

A CURIOUS MOON

by Rob Conery

© 2017 Big Machine, Inc. All rights reserved.

ISBN 978-0-692-04576-3

Publisher's version: 1.1

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law. For permission requests, write to the publisher, addressed "Attention: Rob," at the address below.

Published by: **Big Machine, Inc** 111 Jackson, Seattle WA 96714 Publication date: December 2017

ACKNOWLEDGEMENTS	vii
Image Credits	vii
Resources	ix
PREFACE	x
SQL Code	xi
Cassini and Enceladus	xii
LIFT OFF	1
TRANSIT	14
Installing Postgres	15
Creating The Enceladus Database	16
The Master Plan Table	22
Primary Keys and Sequences	25
Importing The Master Plan	28
Extraction	32
Transformation	32
Loading	34
A Simple SQL Script	35
<i>Idempotence</i>	35
<i>Build.sql</i>	36
Using a Schema	38
Using Make	40
<i>The Basics</i>	41
<i>Organization</i>	43
IN ORBIT	44
Importing Events	45
Importing Events, Again	53
<i>Lookup Tables</i>	55

<i>It's The Joins...</i>	59
<i>Got It!</i>	61
<i>Relational Dependencies</i>	64
White and Black Smokers	69
FLYBYS	73
Tomorrow Is Only a Day Away	87
A BENT FIELD	91
E-0, The First Enceladus Flyby	94
Sargeable and Non-Sargeable Queries	100
Using a View to Make Querying Easier	101
<i>A Mystery Becomes More Mysterious</i>	108
Gold Trace in The Mission Plan	110
Full-Text Queries	113
<i>A Better Full-Text Search</i>	122
Using a Materialized View	126
SNIFF THE SKY	134
Hello INMS	138
Basic Function of the INMS	144
Working With The INMS CSVs	147
Who Remembers This Stuff?	154
<i>What's Better Than 549 CSV Files?</i>	154
Using CSVSQL	158
Successful Import	166
Inspecting The Data	170
Cherry Picking the Nadir	174
Transforming Data with CTEs	182
Something Gnawing At Me	188

<i>Patterns in The Data</i>	189
<i>Jets of Water</i>	190
RING DUST	193
Basic Function of the CDA	194
Refactor Part 1: A Better CTE	198
The Flybys Table	201
CDA Data Audit	206
Importing CDA Data	210
Examining CDA Data	216
Calculating Cassini's Speed	218
Playing with Speed Data	223
A Simple Window Function	231
A TIGHT SHIP	244
Postgres is correct	244
The Data-First Mindset	245
You and Your Data	246
Get Bent	246
Why Postgres?	248
<i>ACID</i>	248
<i>Locking And the MVCC</i>	249
<i>Enterprise features ready to go</i>	250
<i>Friendly</i>	251
Fine: NoSQL if you need it	253
GRAVITY ASSIST	255
Flyby Trigonometry	259
Calculating the Analysis Window	263
Calculating the Analysis Window, Try 2	274

If No Can, No Can	277
Revision: Doing It By Hand	284
OMG OMG OMG OMG!	294
The INMS Tutorial	297
UNDER THE ICE	299
Importing Chemical Data	300
Look at Me!	302
<i>A Non-Dead Ice Ball</i>	303
Understanding the INMS Readings	308
Loading The INMS Readings	310
<i>Using Explain and Analyze</i>	312
A Junction Table	317
Indexes vs. Joins	319
Timestamps and Indexes	321
<i>Simple: A BTREE Index</i>	321
Advanced: Using Time Ranges	328
Comparing The Speeds	332
Putting It Together: A Chemical Query	335
Exploring INMS Readings	338
<i>High and Low Sensitivity Counts</i>	339
<i>Querying Counts by Source</i>	341
<i>Open Source Molecular Hydrogen</i>	345
One Last Query	346
GLIDE PATH	350
Meet M. Sullivan	353
Specs	360
Playing God	365

ACKNOWLEDGEMENTS

This book would not have been possible without the help of my amazing editor, Dian Fay. She went above and beyond the call to make sure Dee's voice was as true as possible and that my SQL and PostgreSQL musings weren't completely deranged. I also want to thank my friend Jon Atten, once again, for constant inspiration.

Finally: the entire NASA/JPL team for being so open and transparent, ensuring that there is free access to the Cassini mission data, and finally for putting all the images under public domain.

IMAGE CREDITS

Every image you'll encounter in this book, unless otherwise specified, is used under public domain with credit to NASA/JPL.

The cover image of this book is based on *Enceladus*, from JPL's Visions of the Future project:

A creative team of visual strategists at JPL, known as "The Studio," created the poster series, which is titled "Visions of the Future." Nine artists, designers, and illustrators were involved in designing the 14 posters, which are the result of many brainstorming sessions with JPL scientists, engineers, and expert com-

municators. Each poster went through a number of concepts and revisions, and each was made better with feedback from the JPL experts.

It's just gorgeous:



The artist's name is David Delgado, who described the image thus:

Saturn's moon Enceladus is all about the plumes erupting from its south pole. At our first brainstorming session, someone called the plumes "Cold Faithful," and that helped crystallize this idea quite quickly. There's no right way up in space, so for fun, we turned the surface upside down from the point of view of the visitors in the picture.

RESOURCES

There is a lot of code in this book, and numerous data for you to play with, which I hope you do! The entire goal of this project is to impart the *feeling* of playing with data and the thrill that comes with scrubbing, importing and normalizing a loose set of data that you *really care about*. For me, that's anything to do with space. When I found the Cassini data set online, woohoo!

To that end, before you jump right in and start reading, please be sure you have:

Postgres installed. I highly recommend Postgres.app if you have a Mac. If you're on Windows, I recommend creating a VM or using a hosted service that gives you SSH access such as compose.io, [ElephantSQL](https://elephantsql.com) or Azure hosted services. Amazon is a bit of a pain, to be honest, but if you know it, hooray!

A Basic familiarity with bash. It's all command line at Red:4, no GUIs! You *could* follow along with a GUI if you want, but you would miss out in my mind.

An open mind. This book isn't a tutorial, and it's not really a science fiction story. It's a bit of both, really, and the only way it will work correctly for you is if you don't demand too much of it. You will learn about databases, data, and Postgres in these pages. You'll also learn a lot about Enceladus and Cassini. Just let it happen.

If you want to get started downloading the data (there's 650mb of the stuff), you can [get a jump on it now](#). Everything you need is in there: CSVs, raw TAB files (if you want to play along), and the SQL code that you'll be reading these pages.

PREFACE

Everything you're about to read is entirely true and completely real, except where it's not. The bulk of my time writing this book was spent researching and cross-referencing every detail I could regarding Cassini and its 23 encounters with Enceladus, and all of that is as accurate as I could make it.

During the writing process, however, I realized it would be a heck of a lot more fun if it read more like a story, so I asked one of our fictional interns, Dee Yan, if I could use her fictional journal to help me tell it.

Red:4 doesn't exist, apart from in my mind. It's entirely fictional even though I consider Dee and everyone else that works at my fictional aerospace startup to be as real as you, reading these words. We've probably never met, but I *know you're there*. You have cares and concerns, a career you're trying to nurture, and an imagination you're wanting to feed. Likewise, each character you're about to meet has all of these same concerns, and accordingly is based on one or more people that I've known for a very long time. I've done my utmost to keep everything as grounded and real as I can.

I realize that books about code and databases don't typically have fictional elements, let alone a cosmic detective story in the form of journal entries and emails. I took a chance on using this narrative approach, and I like what came of it. It's a book that *I would have loved to read* when I was learning about databases. That was one of the best times in my career and tons of fun. I really wanted to bring that across!

I started writing this book as an “end to end experience” where you, as a developer, are thrown into the fire as a “developer/DBA.” Your project had grown to the point where you needed one, but your company wouldn't pay for one, so they elected *you* because *you* knew the most about this PostgreSQL thing. You do what we all did: learn what you need to as you go, Googling and reading blog posts, exploring what's possible and eventually becoming intoxicated on the power of data and the database.

It doesn't take long for the realization to hit you: *data is gold, data is the real value of your business*. Subsequently, everything is a bit muted. It's kind of hard to go back to your life as a developer, shouting at your screen, dealing with yet another JavaScript scoping bug on an interface that will be replaced within 6 months. So *inconsequential*.

With the database, it's the opposite. Experiences tend to swing between *I just improved application performance by 300% across the board by sleuthing the logs and implementing a single index* vs *I think I just dropped prod, better go clean off my desk*. Intensely rewarding, gut-punchingly scary, rinse, repeat.

Sound hyperbolic? I suppose so. In fact, *I know so*. Friends accuse me of this constantly as I ramble on about how data is *everything*, PostgreSQL is *perfect*, code is ephemeral, yadda, yadda, ya... da. I suppose they have a point. If only I had the time and place to indeed show them what it's like, to have them join me on a journey that is reflective of my time learning databases and Postgres...

You're holding that journey in your hand, right now. You're about to become a developer/DBA who assumes the job of her lifetime: building and maintaining a database consisting of data from the Cassini mission, exploring the powerful possibility of life on the small icy moon of Enceladus.

This is a story, sure, but there's also a mountain of code you'll get to write as you explore the Cassini data. We'll start with the super basics in the very beginning and, soon enough, you'll be writing trigonometry functions in PLPGSQL to calculate flyby speeds for an analysis window. You'll write window functions to gather chemistry data beamed back to earth from the INMS and common table expressions to pinpoint exact flyby times. Hopefully, this pulls you in, and you have fun. That's what I'm hoping. *That's what Dee is hoping*.

SQL CODE

The SQL code you're going to encounter, while workable, is probably not the best code I could have written. I pondered this for a long time. On the one hand, I want to present

reliable, elegant solutions that you could feel good about. On the other, I tried to keep it as authentic as possible. The balance between elegance and efficiency is forever precarious.

Dee has work to do, sometimes she plows ahead, just like you or I would. In numerous instances, I wrote some of the more complex queries in the same amount of time Dee might, and I *did not* go back to refactor them, just like Dee wouldn't as long as the query is reasonably performant and is short-lived. In most cases, I did refactor or tweak things at the behest of my technical editors, but there are a few rough spots that I left in here and there.

Either way, I would be interested in hearing your thoughts, should you have some input on the SQL you see and you can [post issues here](#). Please: *issues only*. If you have questions, you can just email me using the address that came with your purchase. Or you can use the one sprinkled throughout the book.

CASSINI AND ENCELADUS

Finally, for the Cassini and/or Enceladus fans out there: if you find any factual errors, *please let me know*. I researched this story as carefully and thoroughly as I possibly could, but I know that it's all too easy to have missed something.

A lot has been written about the small, surprising Saturnian moon named Enceladus, the majority of it being sensational garbage, waxing philosophically about alien life. While interesting to ponder, it conceals the beautiful depth to the mystery that you can blissfully fall into, once you have the facts in hand. A lot is happening up there, so much we have yet to discover!

Every bit of data you'll be playing with comes directly from the Cassini mission, most of it from JPL's Planetary Data System, as pure as I could possibly get it. I've included it in the downloads just as I got it myself, and you will get to parse and load it, just like I did.

Just like Dee did.

You'll muse on whether to use a view, materialized view or a table, thinking through all the possibilities and coming up with a solid choice based on sound reasoning. You'll decide which data is crap, why, and what to do with it once it has been declared crap. You'll go through every decision process I did as I pulled raw text into PostgreSQL, tweaked the types, and began to explore.

Or rather: *how Dee did*. Once you turn the next few pages, Dee will take over this joyride, as she prepares Red:4's next major project: the SELFIE analytical system called Glide Path. *Or so she thinks*.

I need to ask for your patience at this point as Dee has a story to tell you. The code will come soon enough, and before you know it, you'll be awash in it, page upon page.

For now, I'd like to introduce you to Dee Yan, former intern, Red:4.

Rob Conery

November 2021.


Kauai, Hawaii.

LIFT OFF

Why is Nas being escorted out of the building? He looks fatigued, sad, pissed-off and somehow *relieved*. Like a convict on the run being led out of a cheap motel on the outskirts of Dallas, relief from the constant anxiety outweighing the enormity of their current situation.

Nas is our database administrator. Clearly some shit just went down.

I try not to stare, stretching as I stand beside my desk, glancing sideways toward the door. He's not in handcuffs so... no felony, I guess?

Ping. 

It's Slack. I have a private message from Rob, our CTO.

Hey Dee, have a second?

Oh ... *no*. I feel that whatever *something* happened to Nas is now about to happen to *me*. I know we're going after a massive job with JPL and it's the government and OMFG did I divulge some kind of secret or insider-trade some stock?!?!

Of course not. I don't own stocks and I'm just an intern doing some low-level Python/ML work. I'm not even getting paid! But still... I think back on the emails I sent to friends and family telling them about how much fun I was having here...

Layoffs, maybe? *Shut up Dee, get moving.*

Sure, omw, I reply. I glance around my workspace and see if there's anything I genuinely care about, grab my backpack, and head to over to the corner office. It wouldn't make sense to fire me, they don't even *pay* me. Unless the entire business is going under?

The Red:4 SOMA office is an odd little space. It looks a bit like one of those concrete bunkers along the Dutch coast that make up the Atlantikwall, half-buried in a hillside covered with low grass. The entrance is located in the back of a parking lot on Hawthorne, right between Folsom and Harrison. You would never see it from the street unless you were looking for it. It used to be a medical imaging startup and the former owner, upset because people could never find the place, put a mural right above the door that looks like a chest x-ray.

Once inside it looks like your typical industrial storage space turned high-tech office. Exposed beams and pipes, wooden split-level floors, a generally open area separated by low walls and standing desks, etc. Rob's office is in the back corner, which is also our conference room as he's rarely here, off at some conference somewhere, "clienting" as he calls it, or spending time at the Presidio office.

"Hey Dee, come on in. I know you're busy, I'll make this fast. Going somewhere?" Rob asks, looking at my backpack.

"I hope not," I reply with a slight smile. Rob laughs, and I feel *slightly* relieved.

"Right. Nas. Yeah, that was weird and yes, he just got fired. I can't discuss the details as you can imagine but whatever - come in. Sit down. I'll ... explain as I go. First: you know PostgreSQL, right?" he asks.

WTF is going on. This is weird. *Just go with it* I think.

"Sure... yeah I know the basics I guess. Simple select queries, updates, inserts I guess. I mostly let SQLAlchemy do that kind of thing and just focus on—"

"ORMs. Damn, I hate ORMs. Cause nothing but problems and next thing you know you're dropping the production database," he says, interrupting me.

Right. Some shit just went down. I look back at Rob and realize he hasn't slept in a decade and is fraying his last nerve. "Dude. What the hell is going on right now? I'm an intern, our DBA was just perp-walked out the building and you're asking me if I know PostgreSQL? Whatever *this* is" I say, motioning around me, "is not what I signed up for. Please, I'm not big on drama - *just tell me what's going on and why I'm here.*"

Rob closes his eyes slowly. Deep breath. “Just one second...” he says through a controlled breath. “Sorry. I know this is weird. It takes some explanation. Bottom line is: *I need your help.*”

“Go on...”

“I need someone to take Nas’s place for a while until I can interview and hire another DBA - the person who controls our data and data servers. I have to meet our CEO Sara at the jet in 45 minutes. We’re off to LA to meet with JPL again,” he says, drawing out the *n*, rolling his eyes. “I don’t have anyone here that knows our system as well as you. I need you to fill in for Nas. Can you do that?”

“I’m just an intern. How is it possible –“

“Right. Yes! I know, I know. This is incredibly weird and you haven’t been here that long but the simple truth of the matter is that our database infrastructure, at least as far as I saw it, didn’t warrant *redundancy* in terms of personnel. What I’m trying to say is that Nas did a kick-ass job and he did it so well that I just... I dunno I never bothered thinking about what might happen if –“

“You had to walk him out the front door? How is this even possible Rob? I’m sorry I know I’m just an intern here and you’re CTO and all but WTF? That was your *database* full of ... I dunno all the weird shit you do here and I guess something happened to it and now you’re completely screwed and asking an *intern* to rescue you?”

That came out stronger than I thought it would. Startup life, I guess, where “everyone hangs the whiteboards” as my friend used to say. You get used to assuming someone, anyone, can step into any role at any time. I sort of get it, and in some ways I guess that’s why I’m here. You learn a *shitload*, on the other hand it’s extremely draining and borderline abusive.

I can see what’s happening before Rob even asks. I know PostgreSQL given some work I did in college before coming here and starting my internship. He’s thinking something like *well Dee knows PostgreSQL probably more than anyone here so I guess I’ll ask her...*

“Dee the simple fact is that you know PostgreSQL and have time I can free up to dedicate to this. That’s my problem: *time*. I need to start rebuilding our SELFIE database, which was just, literally, vaporized. Yes, there are others here that I could ask to take this on but I need them where they are. You, on the other hand, can be *repurposed*.”

“Way to sell it, *bro*” I say, staring back at him. I came here on a whim and Rob and Sara both impressed me with their vision. This place seemed “different”, I guess, than most SF startup/sweatshops. I’m beginning to feel like this might have been a very bad decision on my part. But then again...

“Dee, I apologize. I’ve been ... it doesn’t matter. I apologize. Look I have a simple truth in front of me: we rebuild the SELFIE database so we can get the bid out the door or we lose, big. I have 28 people on the task and the last thing I thought would happen was that our production system would just... disappear. *I need you*, and I’m willing to make it worth your while.”

That’s more like it. I take a deep breath, and ask: “tell me what happened.”

“Nas dropped the *entire* production cluster. He had been working on this new JPL proposal known as ‘SELFIE’ full time, the biggest contract we’ve ever bid on. It was late, he stayed here far too long and made a very, very bad mistake,” he finishes, taking another deep breath.

I’ve heard stories about people dropping production databases. I always felt like they were embellished to a degree, but here I was, living through one. Every story seemed to have the same basic premise: *a simple mistake, amplified, almost cost the business everything*.

“Our production machine holds about 300G of data; not all that much really. We spent some money and bought a high-performance RAID 10 box loaded to the teeth with SSDs... a DBAs dream,” Rob continues.

Wait, what? Red:4 bought its own hardware?

“I would have thought we would have all of this on Amazon Snowball, Google BigTable or... some other cloud service,” I say, curious.

“Ha! *Infinitely scalable* has a limit. You’d bring those resources to their knees with the amount of data and processing we do. We’d blow our ops budget in days. No, we purchased the hardware ourselves and collocate it down the street at a secure facility. We do, however,” he goes on, “store our backups at Amazon in S3 file storage. This, as it turns out, might have been a bad idea.” Another pause, another deep breath.

“So last month Nas goes to TechEd. He’s a former SQL Server DBA who’s seen the light and is now all-in with Postgres, but he likes to meet up with his former Microsoft pals. Really talented, super nice guy. While at the conference, he sits in this talk discussing AWS architecture concerning micro-services; semi-interesting du jour conference crap I guess,” he says with a wave of his hand, as if dismissing the whole idea.

“The speaker advocates that every service should have its own access credentials for logging and... I don’t know a bunch of other reasons,” he says, his voice gaining an edge. “Inspired, Nas decides, when he returns here after the conference, to tweak a few things with AWS and the permissions on our account. He set up a Virtual Private Cloud (VPC) for our company and resources, put us all in Cognito (the AWS identity service) and made what looked like some nice improvements. It was his job to look after our servers and infrastructure, and I trusted him. Up until today, he’s never let me down,” he says.

“Every night our production server, **red1** is backed up using **pg_dumpall** which, as you might know, is the tool you use to backup the entire system. The total size of the databases we have in production is relatively small, especially when the backup is archived and zipped. This means we can send it to S3 without the need for a special backup service. I don’t know why I thought it would be OK,” Rob says, staring at his hands, “but I just let Nas do whatever he thought best when it came to system backups,” Rob says, His voice is quieting now, and I can tell he’s struggling to stay focused. “I don’t have time to watch *everything* around here...”

“When Nas finished changing around the AWS stuff, he logged into **red1** the production machine, to update its access key to the new one he had just set up with AWS. A quick SSH into prod, start up Vim, reset the key file and that’s it. Unfortunately,” he sits back in his chair, slams his hand on the table and looks out the window once again, trying to calm his rising anger. “*Unfucking*, fortunately, he forgot to change his

environment variables! He'd been logged into **red2**, doing some kind of maintenance earlier in the day. He never changed his ENV variable back... it was *pointing at the wrong server* when he updated the AWS certificate," he says, trailing off while leaning forward, staring at the floor. He takes a minute there, and I don't move or say a thing.

Environment variables. Right. Global variables that are loaded in to your shell's memory on start or, in some cases, as you navigate directories. You can see these things by using **printenv** in the terminal window. They're very useful for creating a "context" or "state" when you navigate to a project directory and many developers will use a **.env** file which will autoload into your shell if you have certain plugins installed. Some people get used to this and just assume that the right ENV variables are being loaded, and will proceed to connect to **DATABASE_URL** without checking that it's the right one...

Rob continues: "We'd never changed the access key on production, and it kept trying to access S3 with the old AWS credentials. The failover never does anything with AWS until you need to promote it. It doesn't matter. He never figured out his mistake until last night, just after midnight," he says.

"Fast-forward 3 weeks, and it turns out that **red1** has been trying to back itself up and failing due to bad permissions at S3. The cron job was still firing successfully, running **pg_dumpall** every night as it should. Unfortunately, the entire script never ran because it failed when it tried to push the archives to S3. This meant that the part responsible for deleting the archive files never executed, which means that within a short time, the incredible disk space on **red1** which cost us over \$150,000 to procure, all of that lovely, insanely fast flash storage tuned to handle our production load, was FULL!" he says, shouting at the floor.

This is a therapy session, watching someone coming to terms with the fact that they have just lost everything because of a *simple text file*. Alternating between shouts and whispers, disbelief and rage.

"Nas had set up this intricate monitoring system using CloudWatch, so if anything like this ever happened, it would wake him up in the middle of the night. Failing that, I would be woken up, then Sara our CEO, and failing all of those, then everyone in the company would be woken up by a text message and/or a phone call. He had it set up as a

dead man's switch, so if a backup ever failed or a heartbeat ping wasn't received, the alarms would start to go off. I was impressed with the setup when he showed it to me. Neither of us, however, could see the one, glaring failure point," he says, trailing off.

Environment variables. I know it's coming. They hold access keys and permissions, and if a server or cloud service has a wrong access key... *oh shit...* "Bad permissions," I say, finishing the thought for him.

"Permissions. Correct, Dee. Very good. When Nas changed permissions around, dividing things among the various services, he didn't set up the VPC correctly. I guess he just thought if everything was in the same VPC that it would just work. Long story short: *it didn't*. The serverless lambda function never fired, which was never logged properly, which didn't matter anyway as the CloudWatch event couldn't access Cognito. No Cognito, no email addresses or phone numbers, no alerts make it out, nobody whatsoever knows anything unusual is going on. Nas never once thought of checking on the monitoring system because he set it up as a dead man's switch: *no news is good news!* Everything must be working just right!" Rob says, not trying to contain his rage anymore.

A dead man's switch is a fun idea until you realize the man is already dead and your switch was never really turned on. Nas never tested this, apparently, assuming it would just work...

"Nas went back to working on SELFIE for the next two weeks, the project we're putting together for JPL," he says, "while our disks, our beautiful, \$150,000 super fast flash drives... filled right up and became useless." He sits up again and looks at me, cocking his head to the side. His voice turns grim. "Nas noticed that. We all did. Our customers especially. That was early this morning at 0216. You know how we found out?" he asks me. "Because I'm paranoid. When we bought the machine, I set up an account on it as part of **sudoers**. Every night a cron job goes off, emailing me with the output from **who -a**. I have an uncontrollable phobia of random people logging into a production box via SSH. I can't help it. I haven't caught anyone in years... but that's a whole other story," he says, twiddling a pencil next to his right hand. "I didn't get that email last night" he continues. "There's only one explanation: the colo facility has disappeared, or the server

is 100% offline. I try to SSH in and can't. A ping works, thankfully, but everything else is just... borked," he says, shaking his head. "So I call Nas just after 2300. He tells me he'll head down to the colo and he'll call me in an hour."

Rob leans forward on the table and says "He didn't call for the next 4 hours. Unbelievable," he says, leaning back again. "I had fallen asleep, thinking the whole thing was handled but when I woke up at 0352, I realized it wasn't. I tried calling Nas again, but he didn't pick up, which makes sense as these colo places are like giant Faraday cages; plus they make you leave your phone at the front desk. The only thing left to do was get in my car and drive down," he says. "I finally found Nas. Sitting in the server cage with keyboard and terminal running, typing away madly."

"Turns out he decided to clean up both servers, remaining logged in to them directly for as little time as possible. He installed **remove/purge** to clear up dead log space and the like, and eventually, **red1** came back online. We left the colo facility then, and he decided to head to the office to figure out why the disk was full. The colo facility doesn't like people hanging around if they don't have to, and all of NAS's scripts and tools are on his office machine, so that's where he went. I went home, back to bed."

"When I got in today, right around 0712, I saw Nas, sitting at his desk with 4 empty Starbucks sitting next to him, a triumphant look on his face. *'I've got this... I've got it don't worry'* he said to me, face buried in his laptop. *'The disk was full of temp archives which for some reason aren't being deleted by the backup routine. I'm promoting red2 so I can triage this and... DONE'*" Rob says, mimicking Nas's southern drawl.

"Nas, *STOP!* I yelled. 'You can't do something like this without a second pair of eyes, you know that.' Red:4 has a triage policy, like many startups and enterprises, that no fix goes live until it's been reviewed by a second pair (at least), of rested eyes.

'No need' Nas says. *'The disk was full is all, for some reason the archives weren't being deleted and... hey that's weird'* he says to me," Rob says, shaking his head. "The very last thing *anyone* wants to hear their DBA say when trying to solve a server problem," Rob says.

“Nas decided to promote **red2**, the failover cluster, so he could troubleshoot our production server, **red1**. Rather than trust replication, he was going to use the last backup from earlier that night. He logs into **red2** and dropped everything to clear up as much space as possible *including the Postgres data directory*. He wanted a clean, empty machine. Once he did that, he logged back into **red1**, which is when he discovered what he had done, muttering *that’s weird*. There were no backup archives, Dee,” Rob says. “Can you guess why?”

No need. *Environment variables* I think. Programmers hate global variables for this exact reason: you can’t possibly understand the havoc they’ll cause in your running program when something changes. Turns out it applies to your local development environment too.

“He never changed his ENV. How... how is that possible for someone as experience as Nas?” I ask.

“I don’t know, Dee. I honestly don’t know and, to be honest, I’m not usually this rash. People should be allowed to make mistakes and firing Nas was the last thing in the world I wanted to do. Our investors and clients see things a little differently - they’re going to want to know what happened and what we did to fix it. I can’t get around the fact that Nas acted independently — going against our protocols and playing hero. I’ve thought of stepping down myself as well, but in this case it was flat negligence on Nas’s part. There’s nothing I could have done to stop him in that moment short of tacking him — which I would have if I could have gotten to him in time! His actions cost us immense amounts of time and money.”

“So, Nas took the fall. If I’m honest, I’m not sad about it. That was a brutally egotistical mistake.”

I can understand this, but the unspoken thing that’s just hanging right there is that, at some level, Rob must have known he’d find someone to take Nas’s place. If not, Rob made as much of a blunder as Nas did by firing his DBA at a time when he couldn’t afford to.

“Yes, you’re right Dee. I won’t lie to you I knew I had options and yes, you were the option I was thinking of. *I believe in you.* You’re not Nas. For *three straight weeks* he had been logging into **red1** thinking it was **red2**. When he **rm -rf**d everything into oblivion, he thought he was on **red2**. He wasn’t. He was on the production machine. All the **pg_dumpall** archives were gone, and all the data files too. We didn’t have any backups on S3 either because of the permissions problem! Despite all of the screw ups, we could have recovered if Nas hadn’t played cowboy and blown everything away trying to clear up space. Net result? *We lost over 3 weeks of data and work.* You knew the problem before I even told you. I don’t think you would have ever let this happen” he finishes.

Yeah. I don’t think I would have done that. I can’t believe I’m agreeing with him.

Rob rubs his eyes, apparently exhausted and emotionally wrung out. Stretches and says “I have the entire Data Team focused on rebuilding *everything* for our clients, so we don’t go out of business. The people that aren’t focused on current clients are focused on SELF - they’re all completely locked up, and you’re the only one of our interns that knows Postgres at all. Point blank, Dee: I know you can do this and I know it’s weird that you’re an intern and I’m asking you to take on a management role but that’s my reality. If you say no, that’s fine, but I think it would be a silly decision. This job could easily be yours. I’ve seen your work and you’re incredibly talented.”

He is compelling, I have to give him that. This clearly isn’t my problem but at the same time I could learn a ton and vault my way right into a management position. Not just a “management” position, but one of the most important positions in the entire company. Flattery is also nice, but my ego doesn’t need it.

“You ever hear of *Enceladus*?” He asks.

“Enceladus?” I reply, “I... don’t think I’ve heard of it, no.” I can’t imagine living in this guy’s head with disconnected thoughts ricocheting around his skull, making their way to his mouth at random. “Why... are we talking about... whatever ‘Enceladus’ is?”

“That’s OK, most people haven’t heard of it. Enceladus is one of Saturn’s moons studied heavily during the Cassini mission. 313 miles in diameter, covered in ice. Most

reflective body in the solar system, too. Odd little thing. NASA thinks there might be life there, and we're bidding on a project that will hopefully find out. It's called SELFI: *Sub-millimeter Enceladus Life Fundamentals Instrument*, and we're going to send along a support system that... well, it's a long story which I can tell you more about when I get back. We think there's life up there, Dee, and I need you to own the data that will compel the story."

I'm working my way backwards through his rambling and pick through things like "life", "Saturn", and "data". Rob is asking me to reassemble and manage a set of data that could prove there's life on Enceladus.

Life. On a moon of Saturn?

...

I know he's talking and I can see his arms moving but I stare into the corner, shutting him and everything else in the room completely out of my mind. A database... full of data... that could help prove there's life ... on a *moon* of Saturn? Wait — did he say 313 miles in diameter? That's... that's smaller than... Texas maybe?

"Hold the fuck on" I say, defiantly. He's talking about something called the *Cosmic Dust Analyzer* which could be from a cartoon or a fever dream or who cares because "I need to process what you're saying to me right now. You need for me to rebuild a database that Nas blew away that held some kind of data from Cassini which could 1) help us win a massive bid with JPL and 2) could prove there's *life* up there? Can you be honest with me: did you sleep last night? This all sounds a bit ... "

"Frantic and far-fetched, I get it. No, I didn't sleep well last night which I guess explains why I'm not explaining things very clearly. Yes, we have a massive dump of chemical data (among other things) that will indeed help us *propose* further exploration into the possibility of life. But no, it's not just me being hyperbolic. NASA said it themselves: *this data is a smoking gun for the presence of life under the ice of Enceladus*. They literally said '*smoking gun*'."

He takes a deep breath and then: "Enceladus has these plumes that spray water ice and a bunch of other material into space. In fact, it sprays so much of it that NASA

believes this little moon is solely responsible for the creation of Saturn's E-ring, the second-largest ring in the solar system. The weird thing is: it's so small that it should have blown itself out a long time ago, but it hasn't," he says. "Cassini flew by this little moon 23 times, sometimes right through the plumes at very low altitude, sampling whatever was ejected. They found some crazy stuff, Dee, things that suggest there's life under the ice. That there's a salty ocean that's heated by... *some* mechanism. They don't know how that's happening either," He says.

"Basically, Dee, Enceladus is a huge mystery that has changed planetary science completely. If we get this job, we get to go along for the next ride. The one that *proves* there's life up there," he says, finally smiling once again. I can tell he likes this stuff.

We stare at each other for ... 2 minutes? 3? He knows when to shut up, which makes me like him, even if he does sound...

"I can't take this all in." I finally say. "I have no idea what you're talking about and it sounds fantastical, at best. But if you're even 1% right about what you're saying, *I'm in*."

"You don't need to believe me, Dee. It's all in the data. That's the fun of this job — you can ignore sales-types like me and just focus on what the data tells you. That's where the truth is. Which reminds me: [here's a link to an archive](#) I need you to pull down. I've just upped your security level and brought you on full time if that's OK with you. I'll pay you the same as I was paying Nas, but it's temporary until I can bring in a replacement. Fair?" he asks.

"Yeah... OK. Sure. Whatever..." I say, still stunned.

He's looking at me with a weird look on his face. "I mean — yes! This sounds amazing and if you're willing to pay me equally then yes! I'm just... it's all a bit overwhelming."

"Great! And I'm sorry Dee, I'm so late I really have to run. If you have any questions just shoot them to M. Sullivan and —"

"Who?" I ask. Is this another weird little surprise? I don't know any *M. Sullivan* and I'm hoping he's not some Area 51 refugee alien living in our supply closet.

Rob looks at me in a surprised way and then smiles. “Right, you don’t know M. Sullivan. Interns wouldn’t really — crap,” he says, looking down at his ringing phone. I can see a picture of Sara, our CEO. Rob answers, jabbing the air with his finger. “Right. Yes. No. No. Right. Yes. No. I’m on my way. Yes. Right.” He hangs up.

He grabs his coat and laptop bag, motions me to the door as he puts his phone in his pocket. “Dee I really have to go, I’m totally late. I’ll explain things in more detail in a day or so. For now, get the data from the link I sent you and load up the Master Schedule; there’s a description of the data with the CSV in the archive. Normalize it, add whatever constraints you think are necessary and... optimize... stuff... I guess. Get started on that and I’ll send an email tomorrow with the rest of it,” he says, locking his office door and sprinting by me. “My Lyft is here I really have to go,” he says.

And he’s gone.

Plumes of ice and alien life? Sure, why not? It might also pay well enough that I can move out of mom’s house.

TRANSIT

October 18th, 2017 1022

No better place to start than at the very beginning, I suppose. [Rob's link](#) to the data took me to the Red:4 archives on S3 and a 650Mb zip file containing all kinds of CSVs and data files.

I unzipped it and put it in a directory in my home called **cassini**:

Name	Size	Kind
▼ CDA	--	Folder
📄 cda_manifest.md	10 KB	Markdown document
📄 cda_manifest.txt	44 KB	Plain Text Document
📄 cda.csv	10...MB	Comma Separat...eadsheet (.csv)
▶ raw	--	Folder
▼ INMS	--	Folder
📄 chem_data_manifest.txt	28 KB	Plain Text Document
📄 chem_data.csv	3 KB	Comma Separat...eadsheet (.csv)
📄 inms_manifest.txt	29 KB	Plain Text Document
📄 inms.csv	5.6 GB	Comma Separat...eadsheet (.csv)
📄 master_plan	13.8 MB	Comma Separat...eadsheet (.csv)

That's a lot of data, which I'll look at in a second. Right now, I need to make sure I have Postgres up and running.

INSTALLING POSTGRES

It looks like IT put Postgres 10.0 on this new box using [Postgres.app](#). I could have used Homebrew, Docker, or built from source but Postgres.app is by far the easiest (and best) for a Mac, which is what I'm using.

I also have Postgres running on my Windows 10 Torture Device at home for some .NET Core stuff I was doing on contract over the summer before I landed the internship here. That was a simple installer I [pulled down from here](#).

I don't seem to have a way to work with Postgres, though. Poking through the apps, I would expect to see Navicat, SQLPro, Postico or, at the very least, the horrid PG Admin IV.

Hmmm. Will have to ask M. Sullivan.

From: M. Sullivan sullz@redfour.io
Subject: RE: GUI, have need
To: Dee Yan yand@redfour.io
Date: October 18, 2017

Hello M. Yan, nice to meet you. We don't do GUIs here at Red:4, they slow you down. I know this might seem counter-intuitive, but I would strongly suggest you invest in yourself and your future by getting used to the Postgres client, **psql**.

Rob mentioned to me that you would be building a database named **enceladus**, which is no doubt what you're trying to do now. There are some binary tools that you get when you install Postgres, and they can be found in the **/bin** directory of your installation.

"Where's that?" you ask? Let's review some incredibly basic command line commands that we evidently forgot doing meaningless millennial things during summer vacation:

which `psql`

That should output the location of the Postgres binaries. In there, you'll see all kinds of exciting tools that can ruin your life if misused. For instance, you could do good things with **createdb** and horrible things with **dropdb**. See if the names of those commands suggest something to you.

Happy to substitute for Google this one time,

Best, S

PS: we call it "Postgres" here, as in "post gress." No need to cramp your fingers typing the entire name out.

PPS: All of our Postgres development machines have their logs symlinked to our monitoring system. We had to do this for our security clearance. I will be having a look at what you're doing from time to time (only in Postgres, don't worry) as Rob has asked me to help you out as I can, which I intend to do.

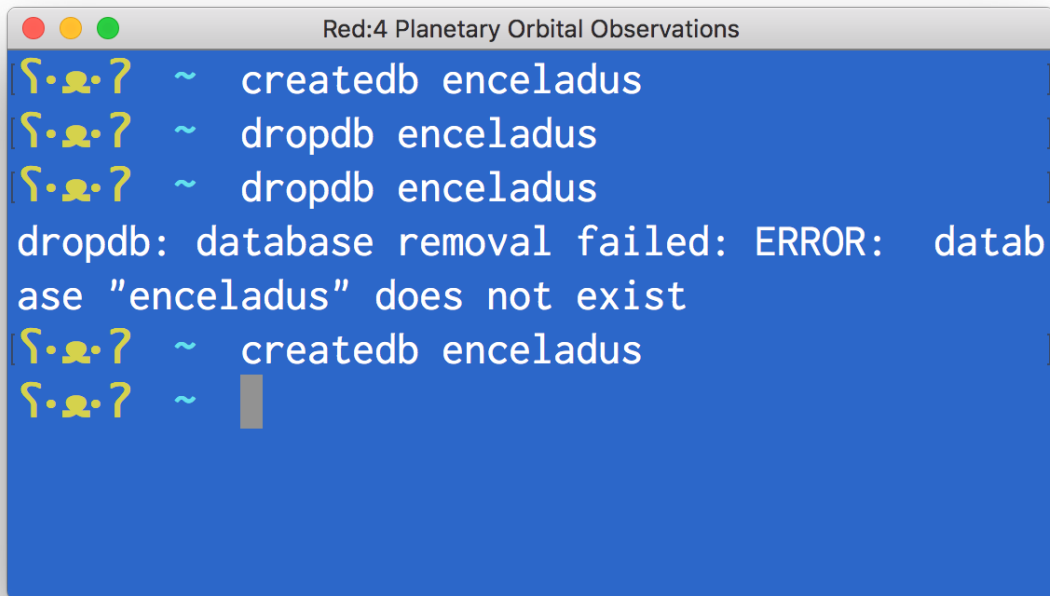
Should be exciting reading, I'm sure.

CREATING THE ENCELADUS DATABASE

Not the sweetest person, is he/she/are they? Strikes me as a bit over-the-top, trying to be edgy, or at least trying to do a wrong impression of Professor Snape from the *Harry Potter* movies. Whatever, moving on.

I've used **psql** before, and I've read things on Hacker News about it being "all you ever need" etc. Some blog posts I've been reading seem to confirm this: that it's much faster and easier to use once you get used to it.

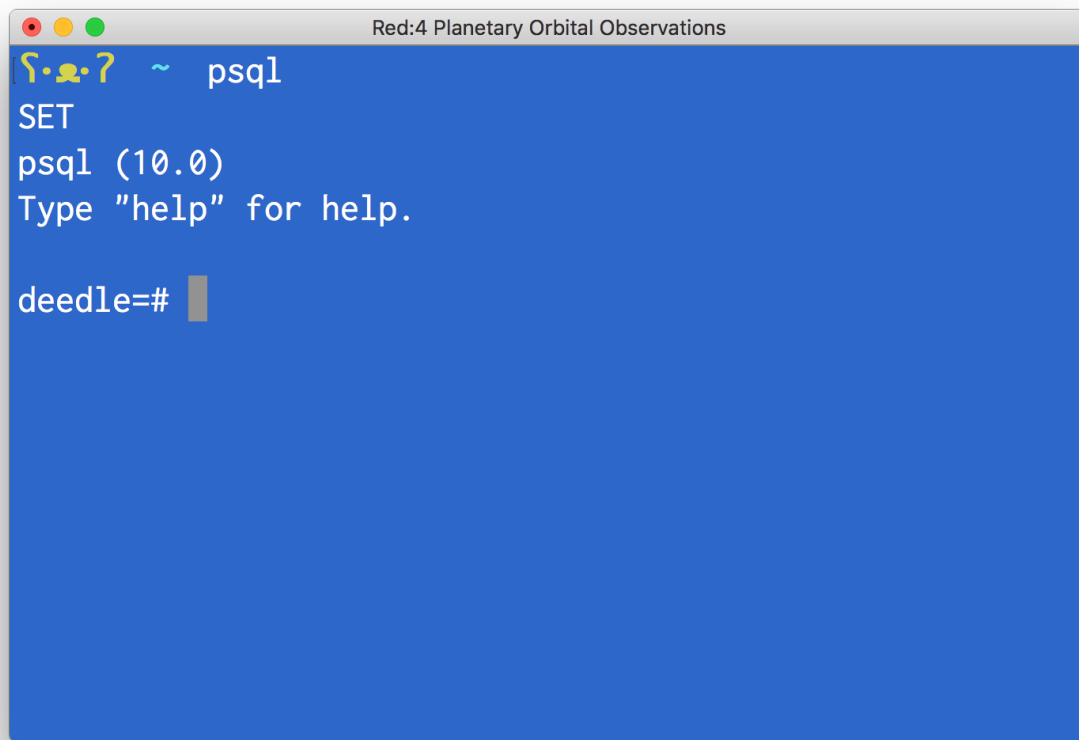
I didn't know about **createdb** and **dropdb**, though:



```
[~? ~ createdb enceladus  
[~? ~ dropdb enceladus  
[~? ~ dropdb enceladus  
dropdb: database removal failed: ERROR: datab  
ase "enceladus" does not exist  
[~? ~ createdb enceladus  
[~? ~
```

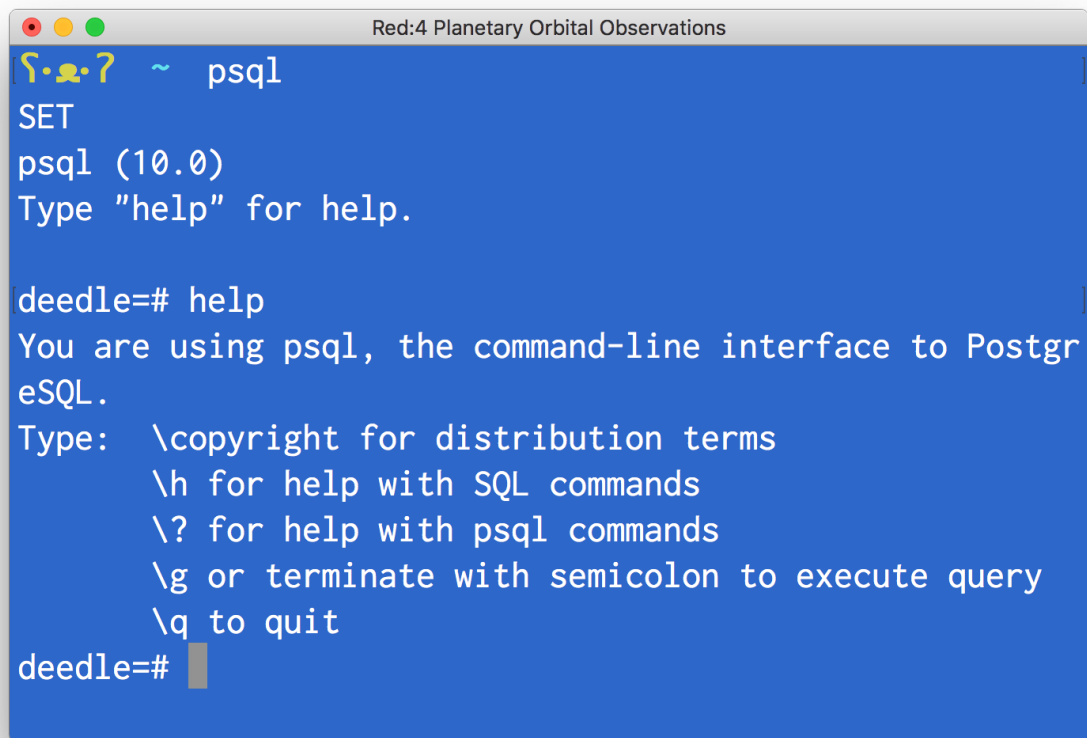
I guess that did seem obvious. I can see what M. Sullivan means: it's pretty easy to drop a database and lose everything. I think it's probably a good idea if I pay extra attention to that.

OK, I have a database. Lets login:



```
f.g.? ~ psql
SET
psql (10.0)
Type "help" for help.
deedle=#
```

I'm logged into my own personal database where I can do whatever I want, which is neat, but where's my new one? I resist the urge to bug M. Sullivan again (though I seriously want to) and type "help" as it says at the prompt:


A terminal window titled "Red:4 Planetary Orbital Observations" with a blue background. The prompt is "psql". The user has entered "SET", and the terminal shows "psql (10.0)" and "Type 'help' for help." The user then enters "deedle=# help", and the terminal displays the help text for psql, including copyright information and a list of backslash commands: \h for help with SQL commands, \? for help with psql commands, \g or terminate with semicolon to execute query, and \q to quit. The prompt "deedle=#" is shown at the bottom with a cursor.

```
Red:4 Planetary Orbital Observations
~ psql
SET
psql (10.0)
Type "help" for help.

deedle=# help
You are using psql, the command-line interface to PostgreSQL.
Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit
deedle=#
```

Got it. I need to know some **psql** commands, specifically how to connect to my new database.

Found it:



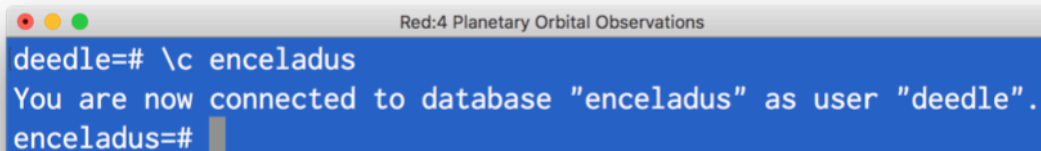
```
Red:4 Planetary Orbital Observations

Connection
\c[connect] {[DBNAME]- USER|- HOST|- PORT|-] | conninfo}
                                connect to new database (currently "deedle")
\conninfo                      display information about current connection
\encoding [ENCODING]           show or set client encoding
\password [USERNAME]           securely change the password for a user

Operating System
\cd [DIR]                      change the current working directory
\setenv NAME [VALUE]           set or unset environment variable
\timing [on|off]               toggle timing of commands (currently off)
\! [COMMAND]                   execute command in shell or start interactive shell

Variables
\prompt [TEXT] NAME            prompt user to set internal variable
:
```

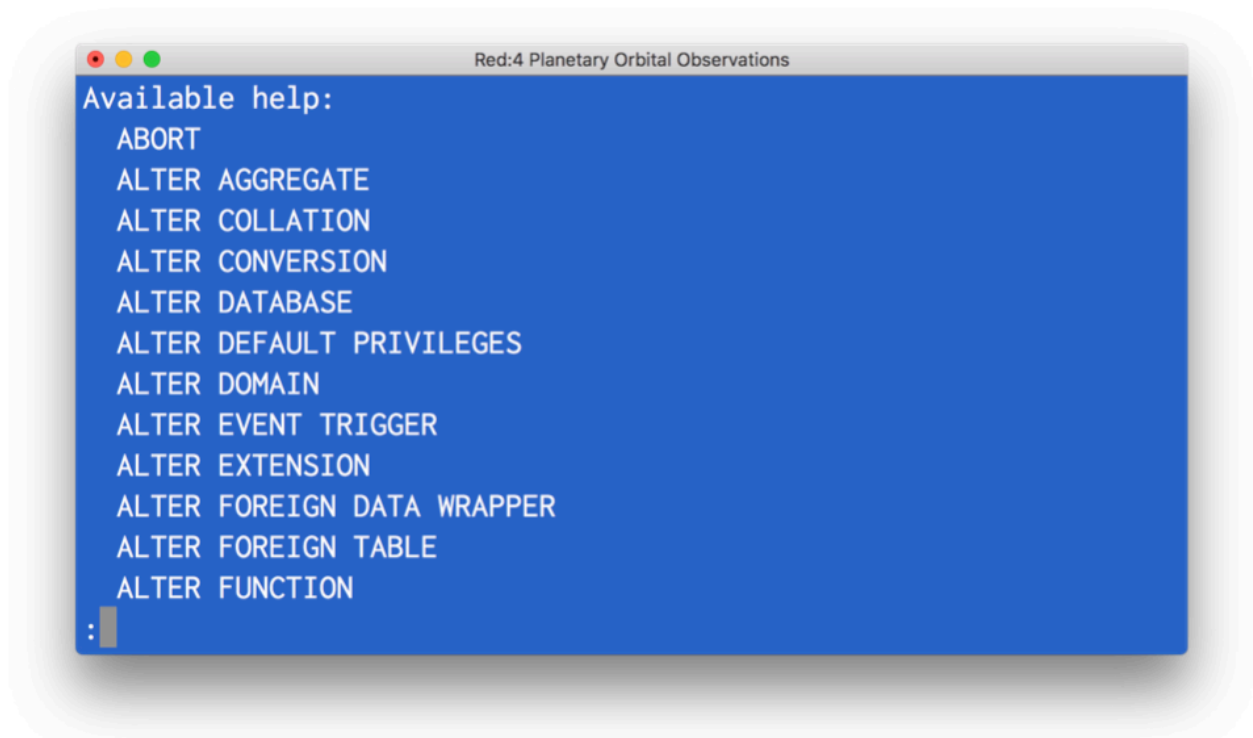
Let's try it:



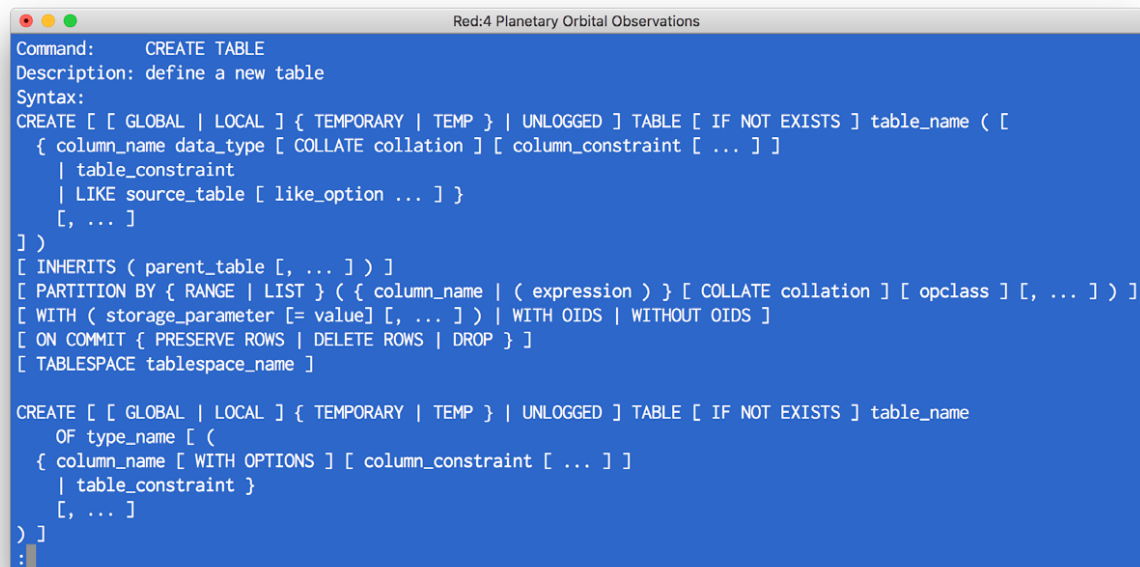
```
Red:4 Planetary Orbital Observations

deedle=# \c enceladus
You are now connected to database "enceladus" as user "deedle".
enceladus=#
```

Perfect. Now I just need to figure out how to create a table. The help screen said to use **h** for help with SQL commands so let's try that:



Ick. The commands seem straightforward, so I'll guess at this one: **h create table:**



The command is valid, but this screen sucks. Think I'll take a break and see if I can find a tutorial on creating tables with SQL.

THE MASTER PLAN TABLE

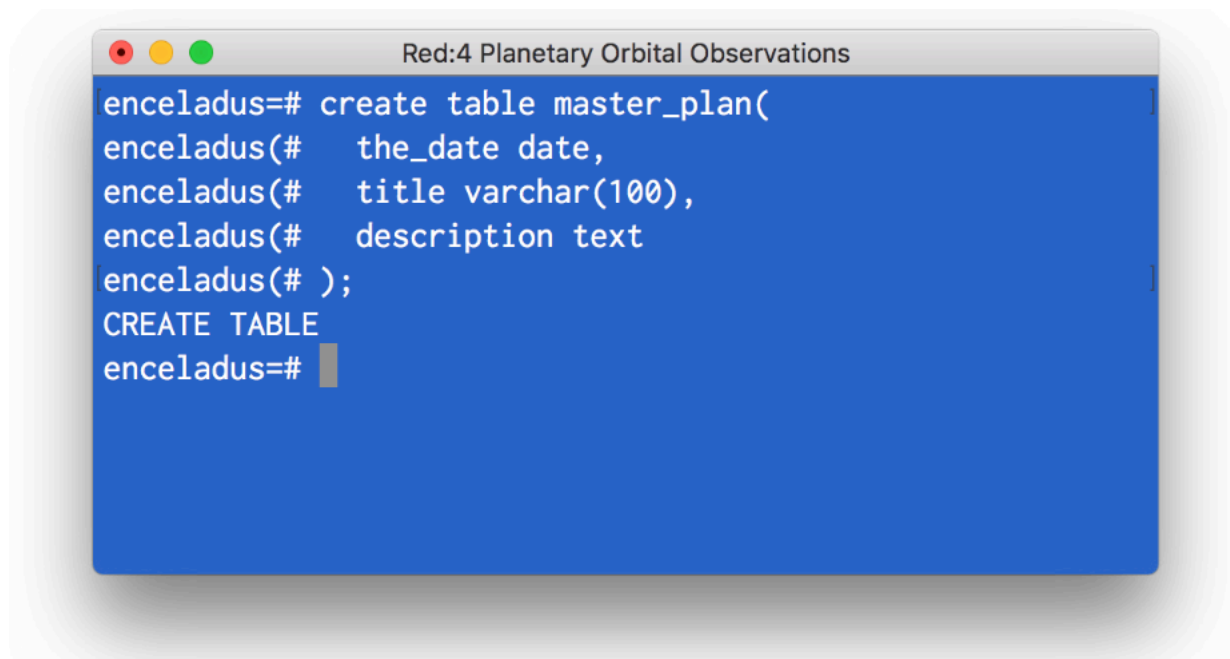
October 18, 2017, 1143

Turns out there's a pretty simple SQL Tutorial [right in the documentation](#) at the main Postgres site. Not the prettiest thing in the world, but the commands seem simple enough.

Turns out making a table is as simple as:

```
create table master_plan(  
    the_date date,  
    title varchar(100), description text  
);
```

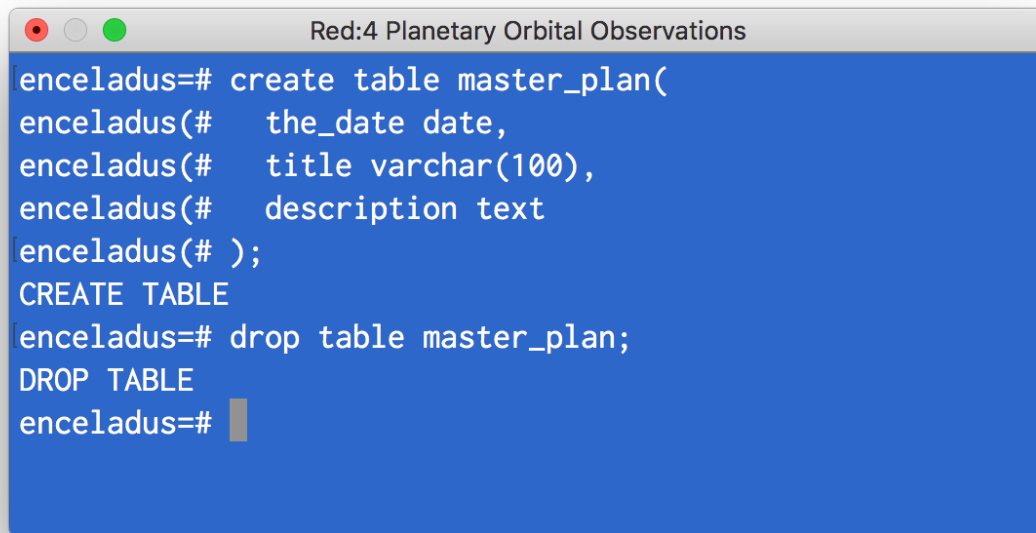
I'm just going with my gut on the data types for now. I can make them make sense later.



```
enceladus=# create table master_plan(  
enceladus(#   the_date date,  
enceladus(#   title varchar(100),  
enceladus(#   description text  
enceladus(# );  
CREATE TABLE  
enceladus=#
```

One thing I like a lot: **psql** wraps the current command if you hit return, unless a line terminator is present, in which case it executes the command. That was fast, too. I can see why people like this utility. Still kind of ugly, though.

Dropping the table is easy too:



```
enceladus=# create table master_plan(
enceladus(# the_date date,
enceladus(# title varchar(100),
enceladus(# description text
enceladus(# );
CREATE TABLE
enceladus=# drop table master_plan;
DROP TABLE
enceladus=#
```

Doesn't seem so bad, really. *Create* a database. *Create* a table. *Alter* this or *drop* that. These are simple instructions that are simple to remember. What follows the instructions are the parameters. This requires some memorization, but Postgres tries to be understanding, so I have some shortcuts I can use.

From: M. Sullivan sullz@redfour.io
Subject: Primary keys are required. Integers.
To: Dee Yan yand@redfour.io
Date: October 18, 2017

Hello M. Yan. I see you have created a table without a primary key. This is not allowed at Red:4, even if it's "just for the lulz." Please take a moment to add a column **ID**

serial primary key to your **create** statement. This will create an auto-incrementing integer key for you, so you don't need to worry about it.

We only use integer keys here at Red:4 unless you have a very, very good reason not to. Which I will guess you probably don't.

Finally, please add a **drop if exists** statement every time you run a **create** statement, for any type of relation. This ensures that my monitoring system won't fill up with avoidable errors.

Best, S

PRIMARY KEYS AND SEQUENCES

October 18, 2017, 1324

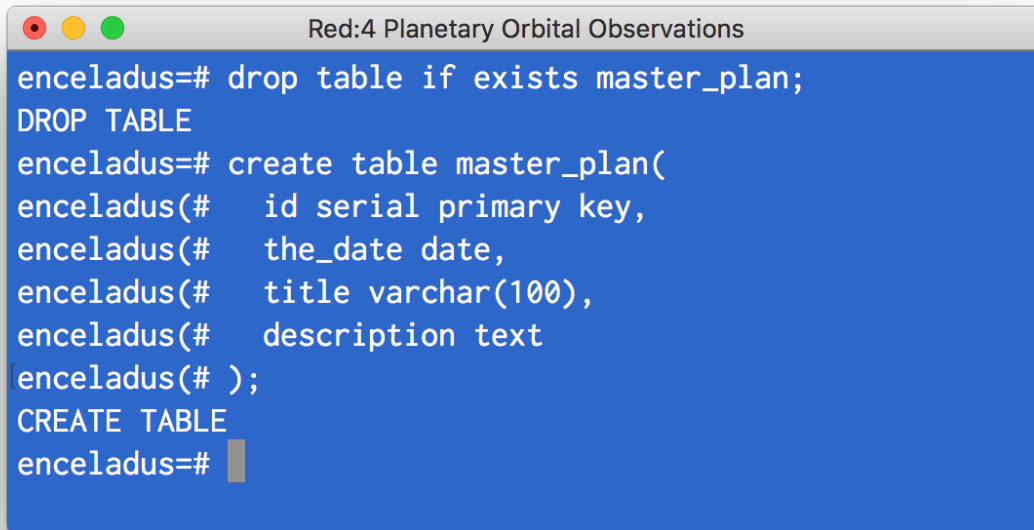
Damn, forgot the primary key. Kind of creeps me out that M. Sullivan is looking over my shoulder like that. I could use the help, so I guess I don't get to use profanity for my column names. Darn.

Didn't know about the **drop table if exists** thing; seems really useful.

OK, redoing my **create** statement:

```
drop table if exists master_plan;  
create table master_plan(  
    id serial primary key, the_date date,  
    title varchar(100), description text  
);
```

Works great:



```
enceladus=# drop table if exists master_plan;
DROP TABLE
enceladus=# create table master_plan(
enceladus(#   id serial primary key,
enceladus(#   the_date date,
enceladus(#   title varchar(100),
enceladus(#   description text
enceladus(# );
CREATE TABLE
enceladus=#
```

What's the deal with **serial primary key** anyway? Guess I'll find out. Googling...

OK, it's not precisely valid SQL. It's a shortcut that Postgres gives you because typing out the exact SQL is a pain. Some people don't like platforms doing things for them, but Postgres is *that smart* and *that good* and adding an auto-incrementing key is *that* simple; people just trust it to do the right thing.

If I were to use straight up ANSI SQL, it would look something like this:

```

create table master_plan(
    id integer not null
    -- ...
);
create sequence master_plan_id_seq;
alter table master_plan
alter column id set default nextval('master_plan_id_seq');
alter table master_plan
add constraint master_plan_pk primary key (id);

```

That’s a lot of SQL. I’m slowly coming around on SQL, but that’s still more than I want to deal with. I took this statement right from a blog post I was reading online and decided to step through it line by line to see if I could understand it. Most of it I could, but what the hell is this, though?

nextval('master_plan_id_seq');

The docs say that **nextval** is a function for accessing the next value in a sequence. A sequence, apparently, is a relation in Postgres, which I think is database speak for “thing.” Whatever. It’s an object... that exists somewhere... and creates incrementing values. I’m sure the implementation details are fascinating, but I don’t have time for that.

As opposed to other systems that bury their auto-incrementing “stuff ” within the engine itself, Postgres puts the functionality right out there, so you can tweak it as you like. A **sequence** has just one job: *remember the last value created*. That’s it! Need a new one? Beautiful — just ask the **sequence** to give it to you, which is precisely what **nextval('sequence_name')** does.

So, this entire block creates a sequence for my **missions** table and then sets the default value of the **id** column to whatever the next value is:

```
create sequence master_plan_id_seq;  
alter table master_plan  
alter column id set default nextval('master_plan_id_seq');
```

This bit of SQL also introduces a new friend: **alter** . Alter works just like **create** or **drop** . You tell it the thing you want to be altered and whatever parameters it needs to do the altering.

This instruction is interesting:

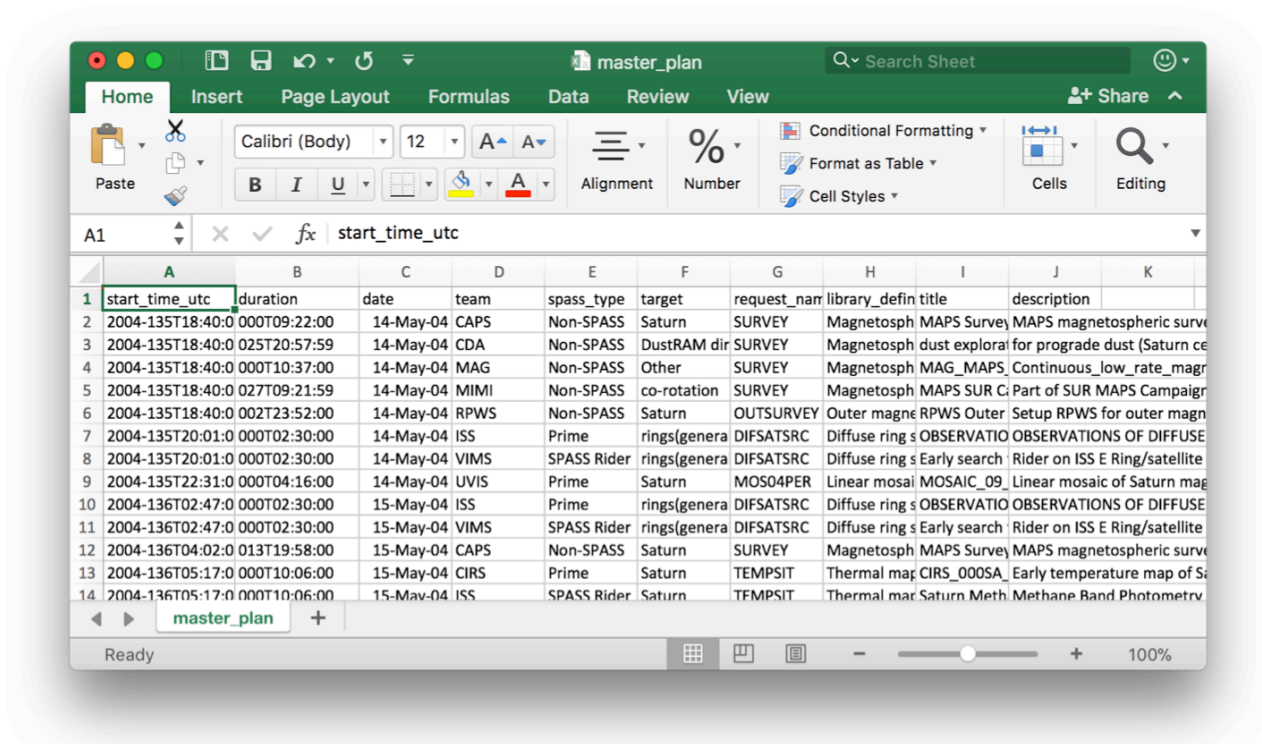
```
alter table master_plan  
add constraint master_plan_pk primary key (id);
```

Postgres is creating a rule for the **masterplan** table. Rules for tables in Postgres are called *constraints*. A primary key seems obvious, but I suppose I've never really thought about it in-depth; it's just something the migration tool takes care of. That's too bad, as it's core to relational theory (as I'm beginning to understand it).

IMPORTING THE MASTER PLAN

I think I'm ready to do this. I have a reasonable understanding of data types and how to set up a table, so I don't get a nastygram from M. Sullivan.

Let's have a look at this **master_plan.csv** and see what we need to do.



Wow. **61,874 rows** of ... what? Instructions? I need to read more about this. Let's see what I can dig up.

The entirety of the Cassini's purpose was to see, listen, touch, taste and smell whatever it could while in orbit around Saturn. Our five human senses were divided among 12 of the most advanced instruments we've ever created, we just had to figure out how and when to use them.

What we did, during the 6 years while Cassini was in transit to Saturn, was to plan every single second of the mission, down to excruciating detail.

--Michele Dougherty, Cassini Scientist

The Mission Plan. Every second of flight time was devoted to studying Saturn, its rings, its moons and its movement through space. A gigantic effort with many brilliant scientists involved.



The entire Cassini team, past and present

But it almost didn't happen. Political pressures squeezed the project to within an inch of cancellation. Another project, the Comet Rendezvous Asteroid Flyby spacecraft, actually *was* canceled to save Cassini. In the end, the Cassini scientists and engineers still had to make several major changes to come in under the new budget constraints.

One such change had to do with the way the instruments were mounted on the probe.

As opposed to Voyager, which had a rotating platform that could point its instruments independently of spacecraft orientation, Cassini's instruments were essentially "bolted on" to the frame. For any of them to collect data, Cassini had to be oriented specifically for the task. This wasn't the original plan, but due to budget cuts, teams had to fight for priority in the mission schedule. One could only [imagine the drama](#) at Cassini HQ:

Instead of each planetary flyby offering data-collecting opportunities to many teams of scientists at once, now only one or two teams could operate simultaneously. The mission leaders were left in the awkward position of deciding whose research would meet the executioner's sword. "People saw all of their plans being really decimated," said Candice Hansen, senior scientist at the

Planetary Science Institute... Hansen... along with Scott Bolton... became a sort of combination traffic cop/psychotherapist, gently guiding research teams through anger, bargaining, and depression and on to acceptance. Eventually, the two were able to craft a new science plan where nobody got what they wanted, everybody gave up something, and all parties were at peace. Their job was a stark reminder that although a spacecraft can exist in a vacuum, data does not. Collecting detailed information about our solar system is as much about people as it is about technology.

So, what happened from all of this? We know precisely: every decision, every study, every single command sent to Cassini was logged. That log was called [The Master Schedule](#) and consists of 61,874 entries.

I have it open in front of me: **masterschedule.csv** . Straight from NASA.

But how do I get all the data from a CSV into Postgres? I'll bet M. Sullivan knows, and I'm confident he would just love to hear from me again.

From: M. Sullivan sullz@redfour.io
Subject: RE: Copy/paste Excel into Postgres?
To: Dee Yan yand@redfour.io
Date: October 23, 2017

Hello again, M. Yan. I am quite happy that you're feeling comfortable in your new position. I assume, of course, that you would not be trolling the Chief Overlord Data Master of Red:4 otherwise.

The process of moving raw data from one data system into another is typically done in three steps: *extract*, *transform* and then *load*. In that order, if that wasn't clear to you.

Normally, this is a term reserved for data *warehousing*, where you gather data from all over the place and stuff it into an analytical system of some kind. In a way, we are doing just that, but we're also building a system that SELFIE can use for learning algorithms while in Kronian orbit.

ETL is about pulling, prepping and loading a ton of data. It's mind-numbing, tedious, and incredibly rewarding once you get to play with the data that you've loaded.

I will go into some detail now. Please print this and keep it next to your machine. Additional copies for your nightstand, bathroom reading, and tattoo artist may also be of use. I know I'm going to be repeating myself, but I hope against hope to keep it to a minimum.

EXTRACTION

You can think of *extraction* as pulling relevant data out of various systems. The CSVs we've been working with have been pulled from [Planetary Data Systems](#) (PDS, maintained by NASA) as well as the [Dust Archive](#) (also maintained by NASA). There's a lot more where that came from — but we have other interns focused on that. *You* get the extraordinary job of tracking down events that happened on Enceladus — which is going to be a primary focus of SELFIE.

TRANSFORMATION

Give me a CSV, and I'll show you a data problem, *somewhere*. Human beings are naturally not mentally equipped to be custodians of data. They're messy by nature, working off of vaguery and intuition in the face of pure logic. The spreadsheet is a curious crossroad between the human and machine mind.

In this setting, *we work for the machines*. If they are to do our bidding they need correct information: it comprises a unique set of failures for both of us.

When the data is extracted, it's usually sent along in CSV format to whatever system is going to load it. CSVs are *garbage*. If we import garbage into our sanitary, pristine database it becomes a rotten, filthy mess. We don't want that, so we need to ensure three things:

- **Correct typing.** Some people think that -- is somehow a meaningful substitute for a number. Other times you'll see the ever-helpful "N/A." These must be corrected, carefully.
- **Completeness.** Rows and rows of data will probably need to be thrown out because they are incomplete, or because we don't know how to deal with their typing.
- **Accuracy.** One of the columns in the particle data that you will be loading is SPACESHIP_RIGHT_ASCENSION, another is SPACECRAFT_DECLINATION. These values have units in angular degrees, angular degrees have an upper bound (90, 180, 360, e.g., depending on the system); therefore, data including a declination of 720 degrees would clearly be inadmissible. Data with a declination of 110 degrees *might* be.

This is the most involved part of the ETL process. The last thing we want to do is to make data *incorrect* while trying to correct it. This could result in Seriously Bad Situations if people and teams are basing their decisions on the data we provide, and the most failure-prone. Our goal is correct, complete, accurate data. If we, and by "we" I mean "you", do not provide correct, complete, an accurate data, we have nothing. Or worse: we wind up wasting large amounts of money and effort on a boondoggle that makes the Mars Climate Orbiter crash look like an evening at the ballet by comparison.

We must be careful, thorough, and diligent. The good news for us is that Postgres makes this straightforward.

LOADING

The loading process is, as you might expect, the act of pushing the data into nicely normalized tables, so we can query it. Every data person has their favorite way of working with data. Most of them that aren't **psql** are horrible. There are some exceptions, but we don't use them here at Red:4.

Like many places, Red:4 isn't immune to the struggle of unnecessary complexity in our software. This is true of our data imports as well. One way we try to offset this is by trying to do the thing with the fewest dependencies/moving parts first and letting the successes or failures of that guide the next step. It isn't that we can't afford to slow down and move pragmatically, it is more that the nature of our work doesn't really present us with a choice.

Put another way, we will typically use:

- Shell scripts/Make files until the complexity slows us down
- A Python project using [Pandas](#) or [PETL](#). We used to use Ruby more for these, but lately, it's been Python, especially for the ML stuff. We'll do this until the transformation and loading times are just too long, or it just *feels wrong* and then...
- We might push a system like [Kafka](#), or subcontract to a vendor that knows what they're doing. We write programs and crunch data — when the import process is too cumbersome we have no problems hiring the right people to get it right.

For what you're doing, shell scripts and **psql** will work perfectly. In fact, I would say that 90% of the time a solid Makefile and some shell scripts are *exactly the right answer*. I do hope that, as a developer, you at least know Make?

One last, crucial tip. I leave it here at the end to be sure you're reading my emails: import everything as **text**. Add types later on. You want the data in the database *first*, everything else can wait.

To that end, and to answer your very first question *last*, the command you're looking for is **COPY FROM**. This will allow you to read a file from disk, adding its contents to the database.

Please put everything in an idempotent script. In fact, *do this for everything*. We need to be sure your work is repeatable, from scratch, should you be hastily escorted from the premises... for whatever reason. Which I'm sure won't happen.

Unless it does.

Best, S

A SIMPLE SQL SCRIPT

An *idempotent* script. I *hate* that word. It just bounces off my brain every time I hear it. I've never *not* had to look it up! So many clashing phonemes!

The... helpful(?) definition:

...denoting an element of a set that is unchanged in value when multiplied or otherwise operated on by itself.

Turns out it just means that I can run a SQL script multiple times, and it will always run, with the same result. I get it. Again.

Idempotence

Sounds good to me. I'll create a SQL script called **build.sql** which I can then push into **psql** :

```
psql enceladus < build.sql
```

I can also just specify the file directly:

```
psql enceladus -f build.sql
```

I just need to be sure that I structure operations in the correct order so Bad Things don't happen. That means I need to be sure everything is dropped at the top, if it exists, and created at the bottom.

Let's get to it.

Build.sql

M. Sullivan suggested that I set everything to the text data type, refining the types later on. Makes sense: Postgres is probably much better and tweaking the data than Excel is:

```
drop table if exists master_plan;  
create table master_plan(  
    start_time_utc text,  
    duration text,  
    date text,  
    team text,  
    spass_type text,  
    target text,  
    request_name text,  
    library_definition text,  
    title text,  
    description text  
);
```

The drop followed by the create makes this simple script idempotent. I can rerun it, and every time the result will be the same, no matter the state of **master_plan** table.

Now I can load the table with a COPY command:

```
COPY master_plan
FROM '[PATH TO DIRECTORY]/master_plan.csv' WITH DELIMITER ',' HEADER CSV;
```

The last line simply tells Postgres the delimiter (obvious), but also that there's a header row and that this is a CSV file.

I *could*, if I wanted to, specify every column that I want the data imported into. If the columns align correctly, however, I can save some typing and just bring it in.

Running this makes my day a little brighter:

COPY 61874

I did it! That was really fast too! Behold my database powers!

From: M. Sullivan sullz@redfour.io
Subject: Please consider isolating data with a schema
To: Dee Yan yand@redfour.io
Date: October 23, 2017

It must have been exciting tipping a dumpster full of raw, probably-fetid data into your pristine database for everyone to see out in the open. If I may offer a suggestion? I don't know why I thought this would occur to you naturally: *please use a schema*.

In case you're not familiar, schemas are where every single relation and object in your database are stored. You can think of them as a "namespace" if that helps.

Best, S

USING A SCHEMA

I think M. Sullivan is starting to like me. I don't even need to email him/her/them, they just email me! A just-in-time help system with some intense snark.

A schema. That makes perfect sense. I need to get organized with my nightly reading. Blog posts, books, videos... none of them tell you the dirty details about doing work like this. I guess it's supposed to melt into your eyes from nastygrams like M. Sullivan's.

That's OK, I can take it. And more. Sullivan will not get to me. BRB hitting Google.

Looks like a schema is a straightforward thing to use. Postgres has a hierarchy to it, which is (in this order):

- **The cluster.** This is a set of servers (usually just one) that execute the instructions.
- **The database.** I think I know what that is.
- **One or more schemas.** The default is public.
- **Tables, views, functions** and other relations. These are all attached to a schema.

I've been dumping raw CSV data into my **public** schema, which, yeah, seems kind of goofy. In programming terms, I suppose this could be like the master branch in Git, the public directory in Rails or the global namespace in client-side JavaScript. Not a place for intermediary... *fetid garbage*, I think it was?

The same style of SQL is used to create the schema, but actually using it can be a difficult to remember in that the schema becomes part of the name of your relation. If you don't specify a schema when creating a relation, **public** is assumed.

I'll dump everything into an import schema, which seems like an obvious name, creating it if it doesn't exist:

```
create schema if not exists import;
drop table if exists import.master_plan;
create table import.master_plan(
    start_time_utc text,
    duration text,
    date text,
    team text,
    spass_type text,
    target text,
    request_name text
);
COPY import.master_plan
FROM '[PATH TO]/master_plan.csv' WITH DELIMITER ',' HEADER CSV;
```

Much cleaner. I still can't believe how fast the **COPY** command is! Time for a cap and a scone.

From: R. Conery rob@redfour.io
Subject: Nice Work, Go Faster.
To: Dee Yan yand@redfour.io
Date: October 23, 2017

M. Sullivan has been kind enough to send me an update on your progress. Good job so far. I'd like to see if I can help nudge things along if you don't mind.

Importing raw data into a database is a significant first step regarding ETL. It's much easier to transform data using a database than it is using Excel. It's also much easier to drop things and start over when you screw up.

Your next step for the import is to normalize and load it into the public schema. This is "global" data, if you will, and will touch every bit of imported data you're going to pull into the system. I need it normalized, organized and optimized. That's a lot of work, so I want you to use a simple build system.

I recommend Make. It's easy to use and made precisely for situations like this. M. Sullivan will answer any questions you have.

R

USING MAKE

October 23, 2017, 1924

Use *Make*? The prehistoric build tool used for creating C programs? How (and *why*) would I use it for a database? I probably should be offended at the level of micromanagement happening right now but ... it's really helping me.

I've run other people's Makefiles and know more or less what it is, but I'm in no way an expert. I'll read about it on BART tonight. Right now, I'm headed over to Jupiter in Berkeley for a beer with Kelly.

The Basics

October 24, 2017, 0632

There are a ton of Make tutorials online, but the essence is this: *Make turns one thing into another*. This could be combining header and code files into C objects and binaries, or it could be individual SQL files appended together and then executed.

You create things called *targets*, which are the actual artifacts or processes you want to happen. The build happens with a *recipe*, which is just a shell command (basically). If target X needs to be built before target Y, then you can specify a *prerequisite* for target Y (which is target X).

All of that combines to make a *rule*. Blah blah blah here's what I came up with last night and refined on the way in today:

```
DB=enceladus
BUILD=${CURDIR}/build.sql
SCRIPTS=${CURDIR}/scripts
CSV='${CURDIR}/data/master_plan.csv'
MASTER=$(SCRIPTS)/import.sql
NORMALIZE = $(SCRIPTS)/normalize.sql

all: normalize
    psql $(DB) -f $(BUILD)

master:
    @cat $(MASTER) >> $(BUILD)

import: master
    @echo "COPY import.master_plan FROM $(CSV) WITH DELIMITER ',' HEADER CSV;" >> $(BUILD)

normalize: import
    @cat $(NORMALIZE) >> $(BUILD)

clean:
    @rm -rf $(BUILD)
```

The blue things with the colons are the targets. Their recipe must follow on a new line, indented with a tab. Specifically, a tab, or everything breaks in weird ways. Don't ask me how much time I wasted figuring that out.

A typical Makefile has three targets:

- **all**: this is the *default target* simply because it is, by convention, usually the first target defined in the Makefile. That's the one that will be executed if you don't specify a target when executing the **make** command.
- **clean**: This tears the build down, removing any build artifacts and "cleaning out" the build directory. I'm using the command here to delete the **build.sql** file.
- **.PHONY**: this target tells Make that "these targets do not have corresponding physical artifacts." Every time you run Make it will look to see if something has been made already. If it has and if none of the source bits have changed, make will skip it. Sometimes you want to override this, so you specify a target as "phony." I'm not using that here.

Each line in a Makefile is supposed to build something. This can be a binary executable, or, as in my case, a simple SQL command. You can't build something, however, if it depends on something else being built first. Thankfully, Make has a way of dealing with this: by specifying the dependencies on the top line, right after the target definition.

This line here: **normalize: import...** specifies that the **normalize** target can't be built unless the **import** target has been built. The **import** target can't be built unless the **master** target has been built. Simple.

The things in all caps at the top of the file are variables. It's not idiomatic to hard-code values directly in your targets, so I put all that stuff at the top. Typical shell stuff, really.

I needed a way to avoid hardcoding the absolute path to my **master_plan.csv** file, so Googled and found a very helpful command: **CURDIR**

Out of those, `${CURDIR}` is the only thing I didn't know about before. It's the current working directory, aka the location you're running Make from, and it's a huge help since `psql` wants the absolute path to the CSV file to **COPY** it.

The flow of the Makefile is straightforward: if I merely run make, the **all** target will be executed. Before that can happen, **normalize** needs to run as it's a dependency, before that can happen **import** needs to run, and before that **master** needs to run.

Once the dependencies are straightened out, each recipe is executed in order.

Each target is appending a bit of SQL to the **build.sql** file except for the last one, **clean**, which deletes it. The **all** target simply invokes `psql` with the **build.sql** file, and we're done.

To rerun a completely clean build I can use `make clean && make`. This will create an idemblargIcan'tevenspellit script, with *zero* involvement from me. *Yay, Dee!!*

Organization

The neat thing about using Make, as Rob suggested, is that I can now organize the various SQL commands into separate files, just like a typical software program. I can make this as elaborate or as simple as I need.

I like keeping things as basic as I can, so I decided to divide the SQL commands into two files:

- **import.sql**, this creates the import schema and loads the CSV
- **normalize.sql**, this will split the import table into lookups, etc., as soon as I figure out what I'm doing that way.

This keeps my Makefile small, too. Before I run things, however, I need to have that normalization script.

IN ORBIT

October 24th, 2017 0921

The whole point of normalizing a database is to reduce repetition and therefore disk space. In some cases, you can speed things up as well. For instance, searching for a particular status is a lot faster if you're comparing integers instead of strings.

Here are the columns in the **master_plan.csv** file:

- **start_time_utc**
- **duration**
- **date**
- **team**
- **spass_type**
- **target**
- **request_name**
- **library_definition**
- **title**
- **description**

Nothing tricky, although I have no idea what **SPASS** is. I'll get to that. We have some dates (two columns, for some reason), team information and so on. If I focus on removing string repetition, then I can create 5 different lookup tables:

- **teams**
- **spass_types**
- **targets**

- **requests**
- **library_definitions**

Some of this information is straightforward if you look at the data (**teams** and **targets**, for instance). SPASS evidently means Science Planning Attitude Spread Sheet. The **library_definition** field is a type of event, and the **request_name** field is ... I have no idea. I'll just include it and call it **requests**.

To build a lookup table, I need to do three things:

1. Get all the distinct values in the import table
2. Create a new table using this data
3. Add a primary key for use with a foreign key constraint

All of these lookup tables need to relate back to a “source” table, which in the database world is referred to as the *fact* table. Since I'm working with historical data, this type of structure makes sense and is referred to as a *star schema*, as it will end up looking like a star at some point, I guess. I just read all of this on the bus today, so I don't know if it will work the way it should.

IMPORTING EVENTS

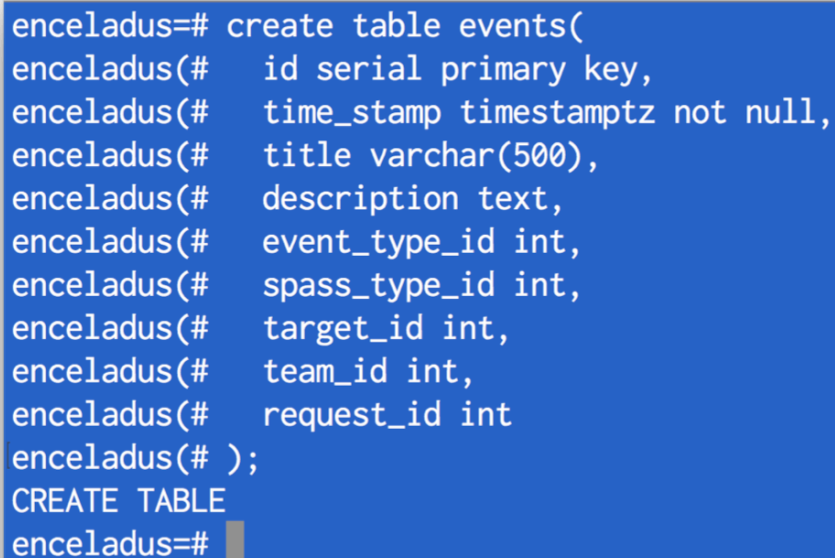
To start, I'll create a fact table called events, and I'll put it in the public schema. Not sure about the name, but I guess I can always change it later with my build script:


```

create table events(
  id serial primary key,
  time_stamp timestamptz not null,
  title varchar(500),
  description text,
  event_type_id int,
  spass_type_id int,
  target_id int,
  team_id int,
  request_id int
);

```

This table gets created in the public schema, where it should be:



```

enceladus=# create table events(
enceladus(#   id serial primary key,
enceladus(#   time_stamp timestamptz not null,
enceladus(#   title varchar(500),
enceladus(#   description text,
enceladus(#   event_type_id int,
enceladus(#   spass_type_id int,
enceladus(#   target_id int,
enceladus(#   team_id int,
enceladus(#   request_id int
enceladus(# );
CREATE TABLE
enceladus=#

```

I'm making sure that I don't have any null constraints, except for the **time_stamp** field. That can't be empty. I don't know about **title** and **description**, but I'm guessing they'll have null values somewhere, so I'll leave them nullable. The lookup IDs *will be empty* until I fill them somehow.

Speaking of, I should be able to get the values from the **import.master_plan** table and insert them using this query:

```
insert into events(time_stamp, title, description)
select
  import.master_plan.date,
  import.master_plan.title,
  import.master_plan.description
from import.master_plan;
```

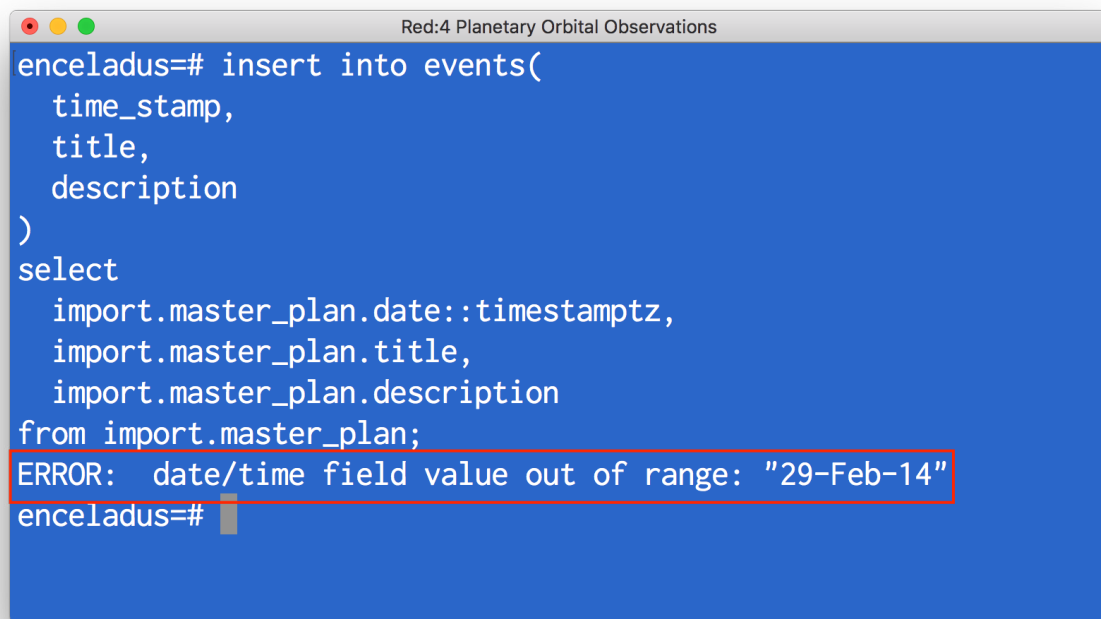
And... crap:

```
Red:4 Planetary Orbital Observations
enceladus(# title,
enceladus(# description
enceladus(# )
enceladus-# select
enceladus-# import.master_plan.date,
enceladus-# import.master_plan.title,
enceladus-# import.master_plan.description
enceladus-# from import.master_plan;
ERROR: column "time_stamp" is of type timestamp with time zone
but expression is of type character varying
LINE 7: import.master_plan.date,
        ^
HINT: You will need to rewrite or cast the expression.
enceladus=#
```

That's... sort of weird. Postgres can't resolve a date from a string? I thought it could parse that stuff.

Just had a look at the docs, you can change one type to another by casting it using **::some_type**. I should be able to cast the **import.master_plan.date** field to a **timestampz** that way:

```
insert into events(time_stamp, title, description)
select
  import.master_plan.date::timestampz,
  import.master_plan.title,
  import.master_plan.description
from import.master_plan;
```



```
Red:4 Planetary Orbital Observations
enceladus=# insert into events(
    time_stamp,
    title,
    description
)
select
    import.master_plan.date::timestampz,
    import.master_plan.title,
    import.master_plan.description
from import.master_plan;
ERROR: date/time field value out of range: "29-Feb-14"
enceladus=#
```

What... the hell?

February 29? As in **Leap Year**? Does JPL think there was a leap year in 2014? I don't know... maybe there was. Google says there wasn't. Postgres says there wasn't.

What the hell is going on?

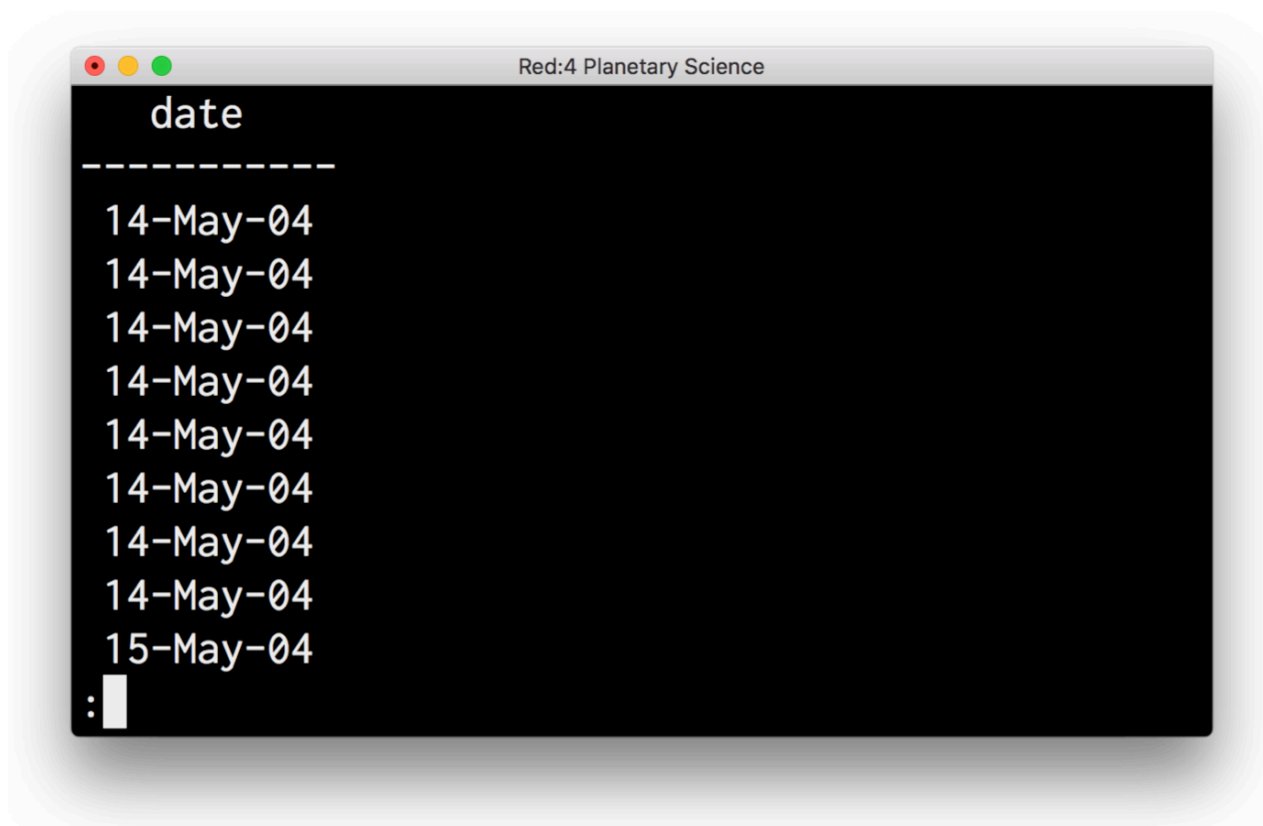
From: M. Sullivan sullz@redfour.io
Subject: RE: Are you guys doing this on purpose?
To: Dee Yan yand@redfour.io
Date: October 24, 2017

Your first foray into the beautiful hell that is other people and their spreadsheets. Welcome, M. Yan. What you've just stumbled into will likely plague your database career for years to come, so take good notes, will you?

The first rule: *dates are always a source of pain.*

I have imported the data into an events table of my making to have a look, and yes, this should be interesting. I already know what the problem is, but I suppose I should take the time to walk you through it anyway.

```
select date from events limit 20;
```

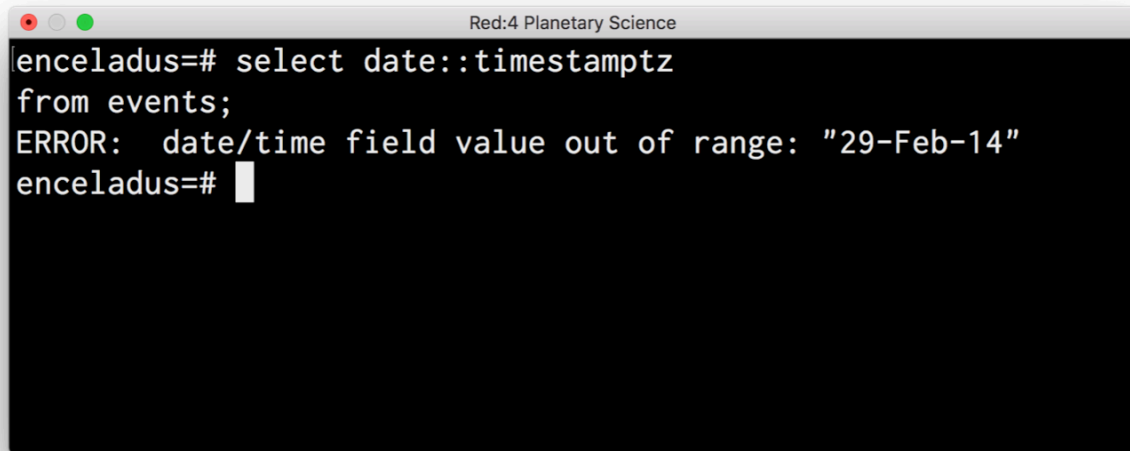


Is that date May 14, 2004, or is it May 04, 2014? I suppose it depends on which side of the Atlantic the author of this spreadsheet is on.

The only way to know is to go back to the source data and look for a pattern to help you out. Having already done that, I know that this date is supposed to be May 14, 2004. It's the date that Cassini entered Saturn's orbit.

Before trying to export *anything*, it's generally a good idea to see if the data in the fields you want to export can indeed be cast to the thing you want:

```
select date::timestampz from events;
```



```
enceladus=# select date::timestampz
from events;
ERROR: date/time field value out of range: "29-Feb-14"
enceladus=#
```

No, it cannot.

This data is straight from the PDS system at JPL, so I must admit I'm a little surprised. You would *think* that they would be able to calculate leap years correctly, but then again...

- In 2012 [Microsoft Azure went offline](#) for 12 hours due to the incorrect calculation of the leap year. They blamed it on a "cert issue," whatever that means. Either way, all systems were offline because of it.
- In 2008 [Zune players around the world froze](#) due to a leap year bug (including mine). The fix? "Wait for 24 hours, and it will restart."

- [There's a bug in Excel](#), which stems from yet another bug Lotus 1-2-3, where 1900 is considered a leap year (it isn't). The reason it's in Excel? Backwards compatibility with Lotus.

The excellent news for you, M. Yan, is:

- NASA uses a date format that avoids issues like this in *year-dayofyear*. It's in there somewhere, you just need to find it.
- If it's not in the data in that format, then look for a large integer field beginning with a 2. This is the Julian date, which some astronomical software tools use.
- Postgres is really, really good at dates, and it's got our backs. If Postgres says it's wrong, we can believe it. Postgres can also fix this problem too.

It seems straightforward that someone at JPL didn't like reading the **start_time_utc** date format, so they created a second column. They then used a built-in function to convert the date, or an older version of Excel running on Windows XP to enforce data formatting, which, because of the bug mentioned above, formatted the date incorrectly as February 29, 2014.

When the spreadsheet was exported, it was saved as a CSV, and all formatting was lost, turning that date into text. Which brings me to a final suggestion: *be careful with your date type*.

You're using **timestampz** , which is "timestamp with time zone" to Postgres. When you read a date value from the text and then cast it to **timestampz** , Postgres can assume any number of things based on your configuration. It may interpret a date (without a timezone specification, which yours does not have) as a UTC date. Or it might interpret that date as being in your local time zone. It's best if you know which.

Every date in Postgres is stored as a UTC date, without a time zone. It's only when you retrieve it that Postgres will convert it based on what's in its configuration file, which by default is the server's location.

You can see this by executing a simple query:

A terminal window titled "Red:4 Planetary Science" with a dark background and light-colored text. The prompt "msullivan=#" is followed by the SQL query "select '2000-01-01'::timestampz; timestampz". The output shows a single row with the value "2000-01-01 00:00:00-08" and "(1 row)". The prompt "msullivan=#" is followed by a cursor.

```
msullivan=# select '2000-01-01'::timestampz;
           timestampz
-----
2000-01-01 00:00:00-08
(1 row)

msullivan=#
```

I didn't add any timezone information, so Postgres assumed it was my local timezone, which is PST, +8.

Best, S

IMPORTING EVENTS, AGAIN

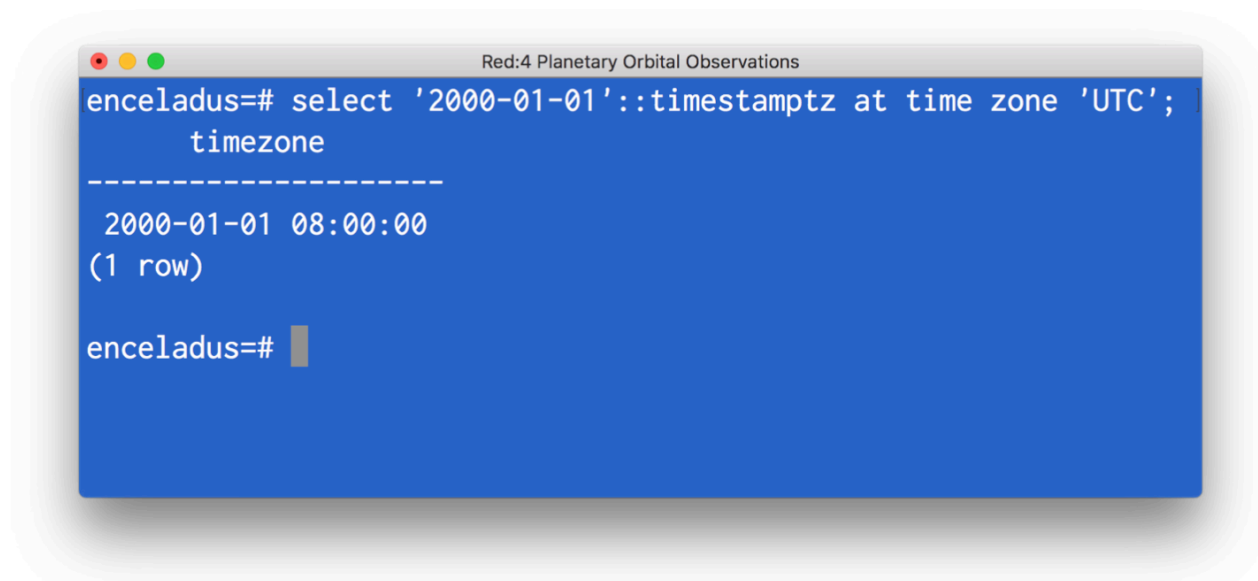
October 24, 2017, 1752

That was almost endearing. I am definitely going to have to find M. Sullivan and give many hugs. I'm sure they will be very well received.

I don't know why I didn't think of running the select statement first, to check and be sure that what I'm casting can, indeed, be cast as I want. That is a perfect idea.

Also: the thing about dates and time zones. It seems like a nightmare waiting to happen. I've already gotten into the habit of using UTC for everything in my applications since it makes life so much simpler. Sounds like Postgres already does that, as long as you record the date in the proper format and don't confuse it.

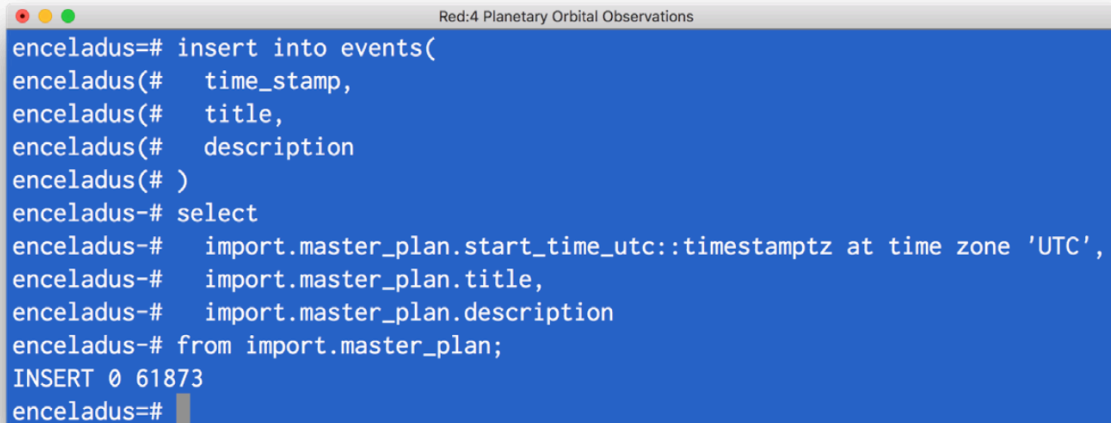
I can cast any date and time using **at time zone X** , which looks a bit weird, but works:

A terminal window titled "Red:4 Planetary Orbital Observations" with a blue background. It shows a PostgreSQL command prompt "enceladus=#" followed by the query "select '2000-01-01'::timestamp at time zone 'UTC';". The output shows a header "timezone" followed by a dashed line and the value "2000-01-01 08:00:00". Below this, it says "(1 row)". The prompt "enceladus=#" is shown again with a cursor.

```
enceladus=# select '2000-01-01'::timestamp at time zone 'UTC';
           timezone
-----
2000-01-01 08:00:00
(1 row)

enceladus=#
```

I can use this to import the time data, but I need to use the **start_time_utc** field to do it as the **date** field will give me a range error that M. Sullivan saw.



```
enceladus=# insert into events(  
enceladus(#   time_stamp,  
enceladus(#   title,  
enceladus(#   description  
enceladus(# )  
enceladus=# select  
enceladus=#   import.master_plan.start_time_utc::timestampz at time zone 'UTC',  
enceladus=#   import.master_plan.title,  
enceladus=#   import.master_plan.description  
enceladus=# from import.master_plan;  
INSERT 0 61873  
enceladus=#
```

Now I just need the lookups.

Lookup Tables

I need to get the distinct values for each of my lookup tables. I can do that with the **distinct** keyword in a select query. I can send the results to a new table using **into** with the table name, I don't even need to create the table beforehand:

```
select distinct(team) as description  
into teams  
from import.master_plan;
```

The next step is to add a primary key:

```
alter table teams
add id serial primary key;
```

Rinse, repeat. I want to keep everything consistent, and the choices I've made are:

- Using **id** as primary key, setting it to an integer
- Using **description** for the text value
- Not using a repetitive naming scheme, like **teams.team** which would look weird.

I do this for each of the lookup tables, and they each have a variation on this pattern:

```
drop table if exists [LOOKUP TABLE];
select distinct(THING) as description
into [LOOKUP TABLE]
from import.master_plan; alter table [LOOKUP TABLE]
add id serial primary key;
```

Now I just need to relate the lookup table back to my events table, planting the lookup **id** values where they should go.

I have no idea how to do that. Crap.

From: M. Sullivan sullz@redfour.io
Subject: RE: Fallen and can't get up

To: Dee Yan yand@redfour.io

Date: October 24, 2017

I'm impressed M. Yan, you don't give up quickly in the face of complete failure.

Queries like this one are quite difficult, and they're also very time-consuming. What you're trying to do is to essentially join two tables on an arbitrary text field, which is very slow. Sometimes that's unavoidable.

You can, however, insert your events from the import table *and* set the IDs, all in one shot:

```

insert into events(
    time_stamp,
    title,
    description,
    event_type_id,
    target_id,
    team_id,
    request_id,
    spass_type_id
)
select
    import.master_plan.start_time_utc::timestamp,
    import.master_plan.title,
    import.master_plan.description,
    event_types.id as event_type_id,
    targets.id as target_id,
    teams.id as team_id,
    requests.id as request_id,
    spass_types.id as spass_type_id
from import.master_plan
inner join event_types on event_types.description
    = import.master_plan.library_definition inner join targets
on targets.description
    = import.master_plan.target inner join teams
on teams.description
    = import.master_plan.team inner join requests
on requests.description
    = import.master_plan.request_name inner join spass_types
on spass_types.description
    = import.master_plan.spass_type;

```

This is a large, somewhat cumbersome query, but it reduces 6 operations into one.

Given that we can insert the event data all at once, that means we can now take full advantage of the **events** table's create statement, creating the foreign keys for us directly in the declaration:

```
drop table if exists events;
create table events(
  id serial primary key,
  time_stamp timestamptz not null,
  title varchar(500),
  description text,
  event_type_id int references event_types(id),
  target_id int references targets(id),
  team_id int references teams(id),
  request_id int references requests(id),
  spass_type_id int references spass_types(id)
);
```

The **references** keyword tells Postgres to create a foreign key constraint for that table/column specification. This saves a total of 11 extra queries, just by knowing a little SQL.

There's a problem, however, can you see what it is? Once you spot it, please be kind enough to fix it as well.

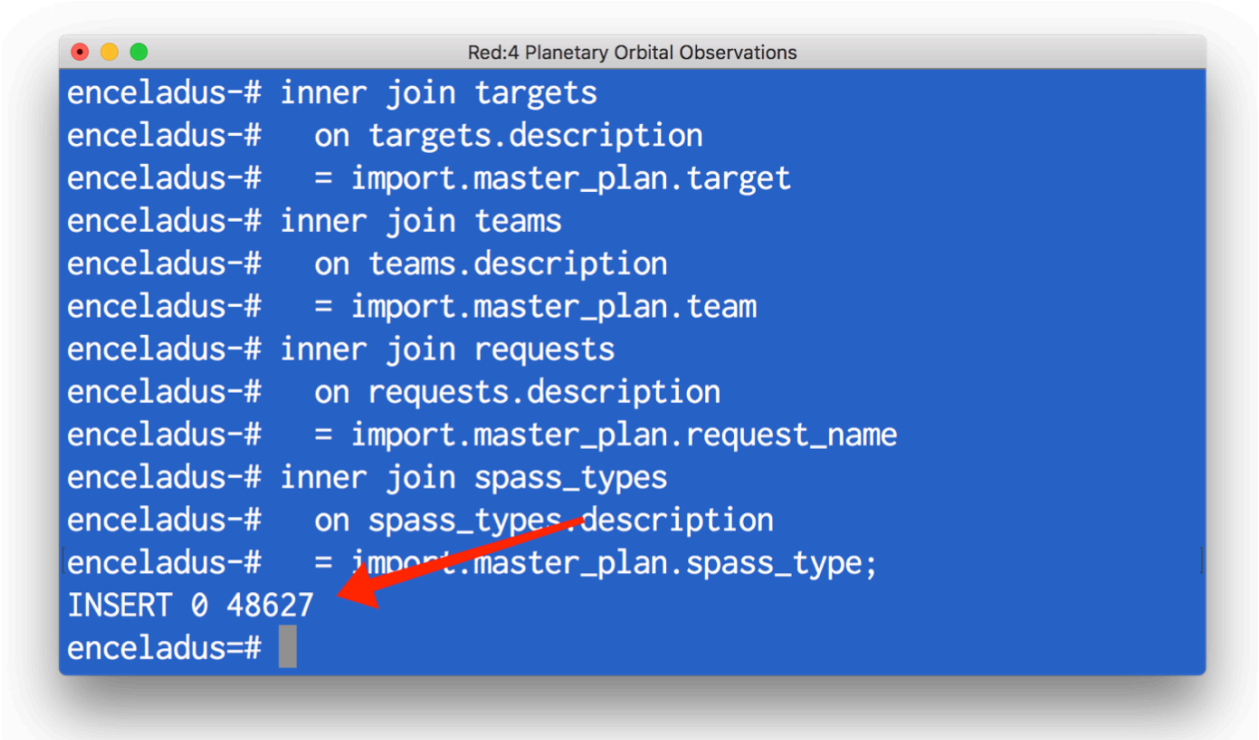
Best, S

It's The Joins...

October 24, 2017, 2012

M. Sullivan's SQL works perfectly, of course. It takes a while to run, but it works. Evidently, there's a problem, however.

Oh, I see it. Right there:



```
enceladus-# inner join targets
enceladus-#   on targets.description
enceladus-#   = import.master_plan.target
enceladus-# inner join teams
enceladus-#   on teams.description
enceladus-#   = import.master_plan.team
enceladus-# inner join requests
enceladus-#   on requests.description
enceladus-#   = import.master_plan.request_name
enceladus-# inner join spass_types
enceladus-#   on spass_types.description
enceladus-#   = import.master_plan.spass_type;
INSERT 0 48627
enceladus=#
```

There should be 61,873 rows that get imported... there are only 48,627.

I'm going to have to think about that one, and it will have to be tomorrow. *Stranger Things 2* is on, and there is *no way* I'm missing this.

October 25, 2017, 1014

Kind of slept in today. Didn't mean to. I tossed and turned last night thinking about SQL statements being pumped out of the floor of my room. I needed to catch them and put them in a green box with little crabs running around it. Each one would eat a bit of data... never mind who cares. I'm exhausted.

It's just occurred to me that I have the entire Cassini master schedule on my machine at work and I haven't even looked at the data. That's nuts!

Got It!

October 25, 2017, 1132

Got it. It's the joins. They're restricting the results from the select query. Data loss like this can happen in only one of two ways: **where** clauses and **joins** are the only parts of SQL statements which can filter or remove data from the result set.

There's no where clause in the insert, so I need to use a less-restrictive join. Be right back...

I need left joins:

```
select
  import.master_plan.start_time_utc::timestamp,
  import.master_plan.title,
  import.master_plan.description,
  event_types.id as event_type_id,
  targets.id as target_id,
  teams.id as team_id,
  requests.id as request_id,
  spass_types.id as spass_type_id
from import.master_plan
left join event_types on event_types.description
  = import.master_plan.library_definition
left join targets on targets.description
  = import.master_plan.target
left join teams on teams.description
  = import.master_plan.team
left join requests on requests.description
  = import.master_plan.request_name
left join spass_types on spass_types.description
  = import.master_plan.spass_type;
```

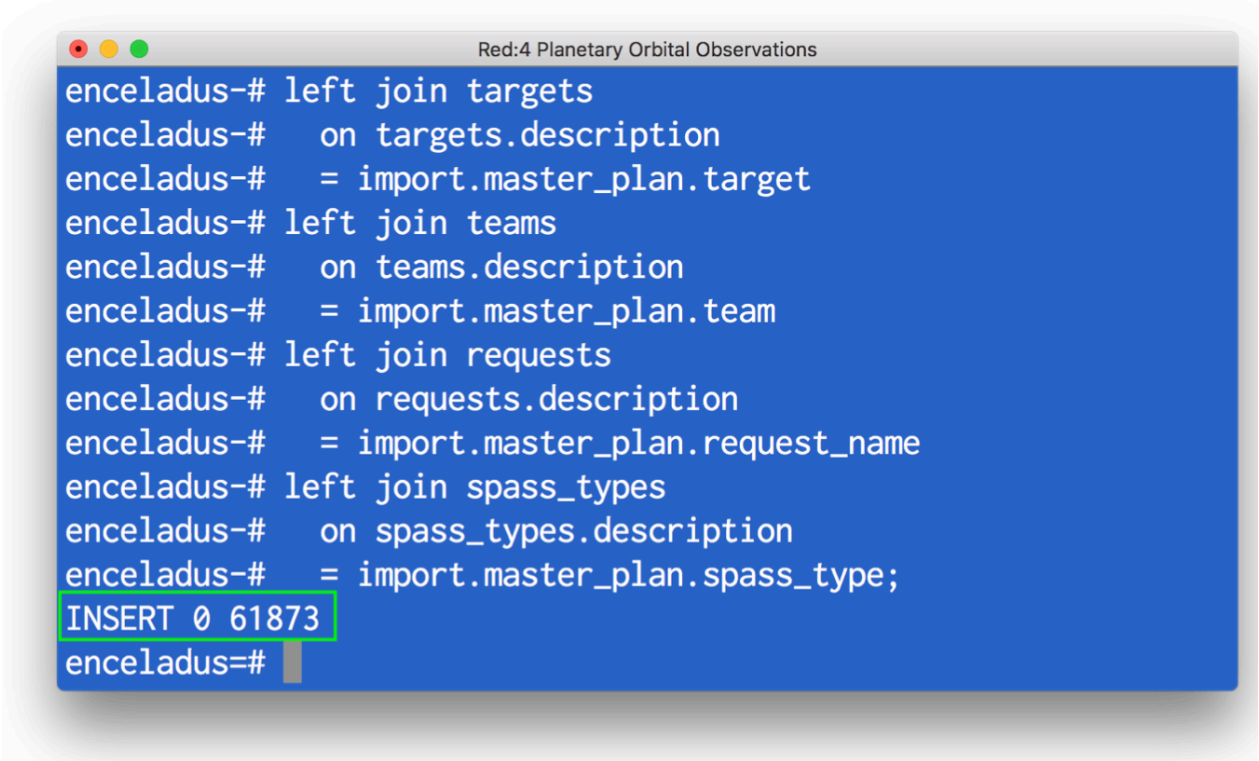

There are different ways you can join records in a table:

1. An inner join, which M. Sullivan used. That will return only those results where matching rows exist in the join table.
2. A left join (also called a left outer join) will return all the records in from table, which in this query is **import.master_plan**, and fill out missing rows in the join table with nulls
3. A right join (also called a right outer join) will return all the records in the joining table, even if there are no joinable rows in the from the table, again padding missing data with nulls
4. A full outer join does both a left and right join, basically returning every row in both tables
5. Finally, there's the odd cross join, which returns every row in the from table crossed with every row in the joining table. So: if our from table had 5 rows, and our cross-joined table had 2, we would have 10 total records returned. I don't know anyone that has ever needed to use this, but I imagine somewhere it's useful!

The weird thing is that all of these tables were created by the *same master table*, but the trick is that *every single row needs to have a value in the lookups to be counted*. That means we might have an event without a team or request. Maybe there's a missing target.

That's OK, I think. Some of these events might not have targets.

Rerunning the script with left joins...



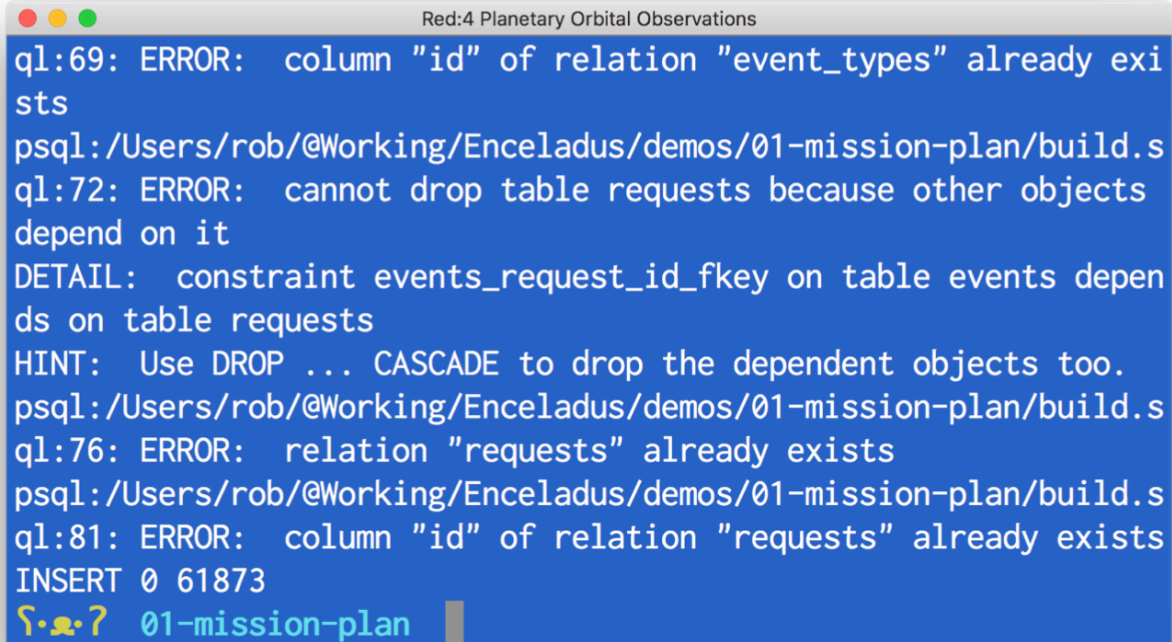
```
enceladus-# left join targets
enceladus-#   on targets.description
enceladus-#   = import.master_plan.target
enceladus-# left join teams
enceladus-#   on teams.description
enceladus-#   = import.master_plan.team
enceladus-# left join requests
enceladus-#   on requests.description
enceladus-#   = import.master_plan.request_name
enceladus-# left join spass_types
enceladus-#   on spass_types.description
enceladus-#   = import.master_plan.spass_type;
INSERT 0 61873
enceladus=#
```

Bingo. Now I can wire up my Makefile. I've put the **master_plan.csv** file into the data directory and two SQL files (**import.sql** and **normalize.sql**) into a scripts directory.

The **import.sql** file drops and creates the import schema, creates the **master_plan** table, and then the events table in the public schema. The **normalize.sql** file creates the lookups and then jams everything into the **events** table.

That's a lot of work! I guess that's why Rob insisted I automate it. Otherwise, I'd be here tweaking and replaying scripts until this "Glide Path" project makes it to Saturn without my stuff... or until I get marched out the door. *Got to focus*. But it'll be pretty sweet to see all this happens with a single command!

make clean && make

A terminal window titled "Red:4 Planetary Orbital Observations" with a blue background and white text. It shows several PostgreSQL error messages. The first error is "column 'id' of relation 'event_types' already exists". The second is "cannot drop table requests because other objects depend on it", with a detail about a constraint and a hint to use CASCADE. The third is "relation 'requests' already exists". The fourth is "column 'id' of relation 'requests' already exists". The terminal also shows "INSERT 0 61873" and a prompt "01-mission-plan".

```
ql:69: ERROR: column "id" of relation "event_types" already exists
psql:/Users/rob/@Working/Enceladus/demos/01-mission-plan/build.s
ql:72: ERROR: cannot drop table requests because other objects
depend on it
DETAIL: constraint events_request_id_fkey on table events depends
on table requests
HINT: Use DROP ... CASCADE to drop the dependent objects too.
psql:/Users/rob/@Working/Enceladus/demos/01-mission-plan/build.s
ql:76: ERROR: relation "requests" already exists
psql:/Users/rob/@Working/Enceladus/demos/01-mission-plan/build.s
ql:81: ERROR: column "id" of relation "requests" already exists
INSERT 0 61873
? 01-mission-plan
```

CRAP. What... could *possibly* have gone wrong? *Akamai, Dee, be smart and read the errors...*

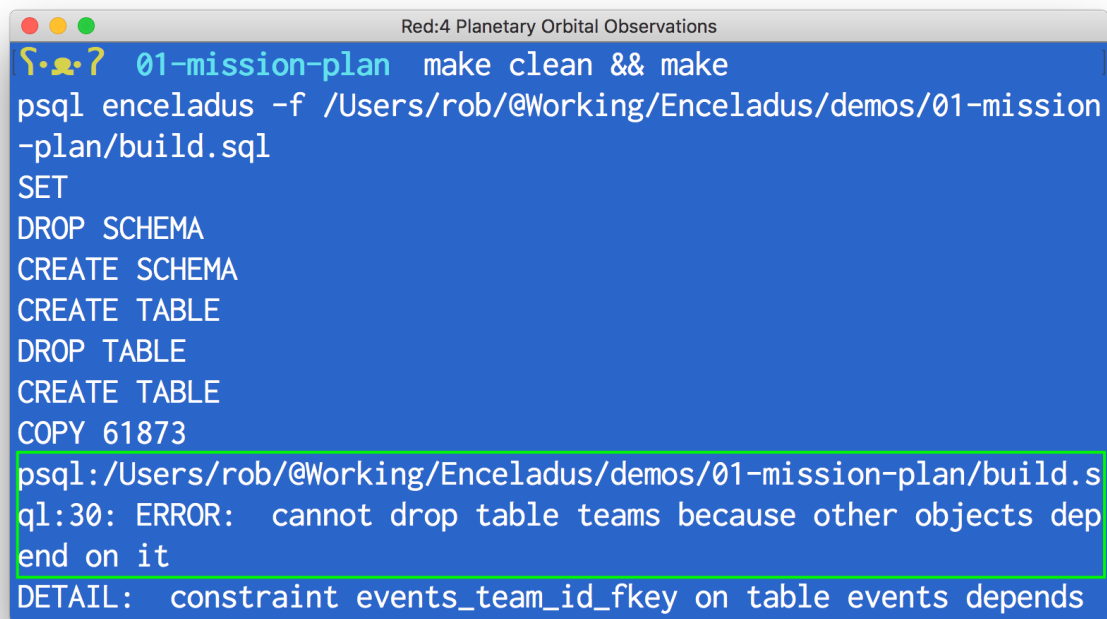
It looks like it's trying to create tables that already exist. Tables that can't be dropped for some reason... hmm.

Relational Dependencies

October 25, 2017, 1246

I have been running the SQL that M. Sullivan sent me (and that I tweaked) but I haven't tried to run everything at once, other than now. It seemed pretty straightforward: just drop and reload the events tables and lookups.

Turns out it's not that easy. I can see this if I scroll back to the top of the Makefile output:

A terminal window titled "Red:4 Planetary Orbital Observations" with a blue background. The prompt is a shell icon followed by "01-mission-plan". The user enters "make clean && make". The terminal shows the execution of a SQL script "psql enceladus -f /Users/rob/@Working/Enceladus/demos/01-mission-plan/build.sql". The script contains several SQL commands: "SET", "DROP SCHEMA", "CREATE SCHEMA", "CREATE TABLE", "DROP TABLE", "CREATE TABLE", and "COPY 61873". The last command is highlighted with a green box. Below it, an error message is displayed: "psql:/Users/rob/@Working/Enceladus/demos/01-mission-plan/build.s
ql:30: ERROR: cannot drop table teams because other objects dep
end on it
DETAIL: constraint events_team_id_fkey on table events depends".

```
01-mission-plan make clean && make
psql enceladus -f /Users/rob/@Working/Enceladus/demos/01-mission-
-plan/build.sql
SET
DROP SCHEMA
CREATE SCHEMA
CREATE TABLE
DROP TABLE
CREATE TABLE
COPY 61873
psql:/Users/rob/@Working/Enceladus/demos/01-mission-plan/build.s
ql:30: ERROR: cannot drop table teams because other objects dep
end on it
DETAIL: constraint events_team_id_fkey on table events depends
```

Make will output the results of the commands, which in this case is just the big **build.sql** file. I can see why M. Sullivan recommended this, I can just open the **build.sql** file and see what/where the problem is.

Which I don't need to do, I can see it right there: can't drop the events table because there are constraints on it, specifically foreign key constraints.

This is Postgres trying to do the right thing. If I was able to delete the events table, I might leave things stranded, or *orphaned*, in other tables. This is what happens when you specify a foreign key constraint. Not only will it protect your data, but it will also help protect your *table*.

I really like how Postgres tries to be helpful as well. *If you really want to shoot yourself in the foot...*

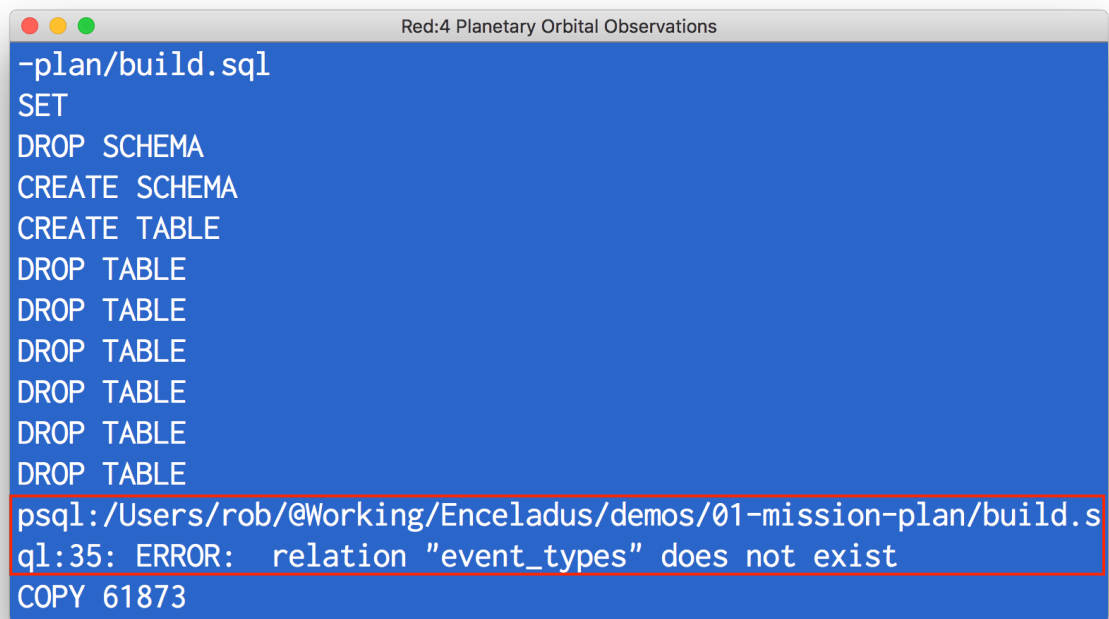
```
Red:4 Planetary Orbital Observations
DROP SCHEMA
CREATE SCHEMA
CREATE TABLE
DROP TABLE
CREATE TABLE
COPY 61873
psql:/Users/rob/@Working/Enceladus/demos/01-mission-plan/build.s
ql:30: ERROR:  cannot drop table teams because other objects dep
end on it
DETAIL:  constraint events_team_id_fkey on table events depends
on table teams
HINT:  Use DROP ... CASCADE to drop the dependent objects too.
psql:/Users/rob/@Working/Enceladus/demos/01-mission-plan/build.s
ql:34: ERROR:  relation "teams" already exists
```

... *here's how you do it*. Looks like all I need to do is to pop ***cascade*** onto the drop statement and all the dependent objects will be dropped too. Sounds excitingly dangerous.

In fact, I'll just be sure every single table is dropped right there at the top:

```
drop table if exists events cascade;
drop table if exists teams cascade;
drop table if exists targets cascade;
drop table if exists spass_types cascade;
drop table if exists requests cascade;
drop table if exists event_types cascade;
```

I don't need the cascades if I'm dropping everything, but I believe in being thorough. Let's try it again:


A terminal window titled "Red:4 Planetary Orbital Observations" with a blue background. It displays a series of SQL commands: -plan/build.sql, SET, DROP SCHEMA, CREATE SCHEMA, CREATE TABLE, DROP TABLE, DROP TABLE, DROP TABLE, DROP TABLE, DROP TABLE, and DROP TABLE. The next line shows the command path: psql:/Users/rob/@Working/Enceladus/demos/01-mission-plan/build.s. This is followed by an error message: ql:35: ERROR: relation "event_types" does not exist. The final line shows the output: COPY 61873.

```
-plan/build.sql
SET
DROP SCHEMA
CREATE SCHEMA
CREATE TABLE
DROP TABLE
DROP TABLE
DROP TABLE
DROP TABLE
DROP TABLE
DROP TABLE
psql:/Users/rob/@Working/Enceladus/demos/01-mission-plan/build.s
ql:35: ERROR: relation "event_types" does not exist
COPY 61873
```

Puzzling. I can see the output of running **build.sql**. It's telling me that it dropped and created the schema, but at line 35 (which is where the **events** table is created) there's no **event_types** table?

What? Oh. Duh. Dee! It's the foreign key thingy!

```
26     id serial primary key,  
27     time_stamp timestamptz not null,  
28     title varchar(500),  
29     description text,  
30     event_type_id int references event_types(id),  
31     target_id int references targets(id),  
32     team_id int references teams(id),  
33     request_id int references requests(id),  
34     spass_type_id int references spass_types(id)
```



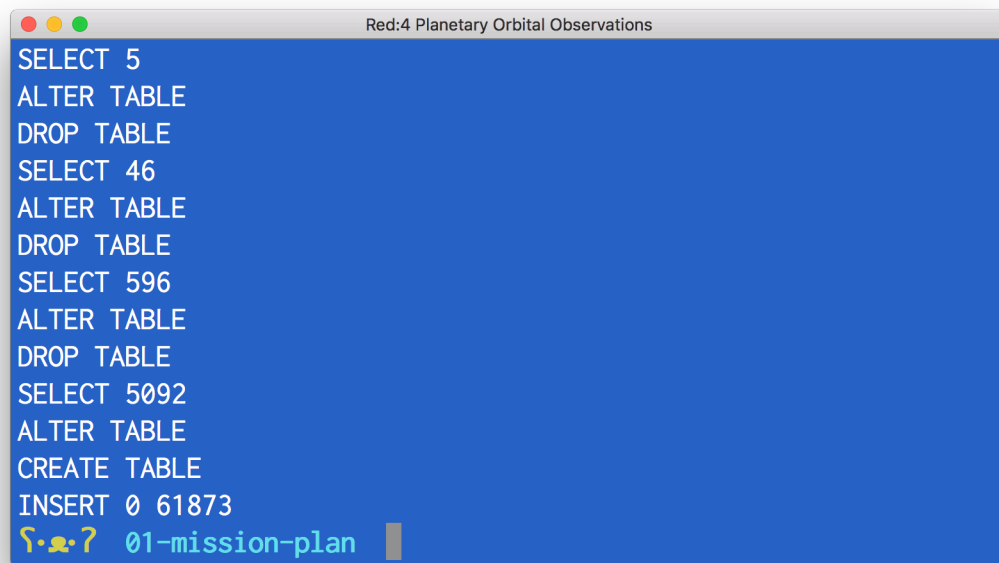
You can't reference a table that doesn't exist!

That's OK. I can fix this in one of two ways:

1. Move the SQL statements around, so I don't try to reference things that don't exist or
2. Tweak my Makefile so that the build steps are more granular, letting it handle the orchestration

The programmer in me wants to do the latter. The “Keep it simple, Dee” part of me wants to... yeah, keep it simple. I think simple wins — I'll just move the **create** statement right down above the insert statement and call it good. I'll leave the delete statements where they are.

Let's try again.



```
Red:4 Planetary Orbital Observations
SELECT 5
ALTER TABLE
DROP TABLE
SELECT 46
ALTER TABLE
DROP TABLE
SELECT 596
ALTER TABLE
DROP TABLE
SELECT 5092
ALTER TABLE
CREATE TABLE
INSERT 0 61873
01-mission-plan
```

That was kind of painless, I suppose.

I'm starving... time for a burrito at Rosie's. Rob sent me some links to an article on deep-sea vents, and it looks like a YouTube video as well? Weird. I'll have a look while I'm eating.

WHITE AND BLACK SMOKERS

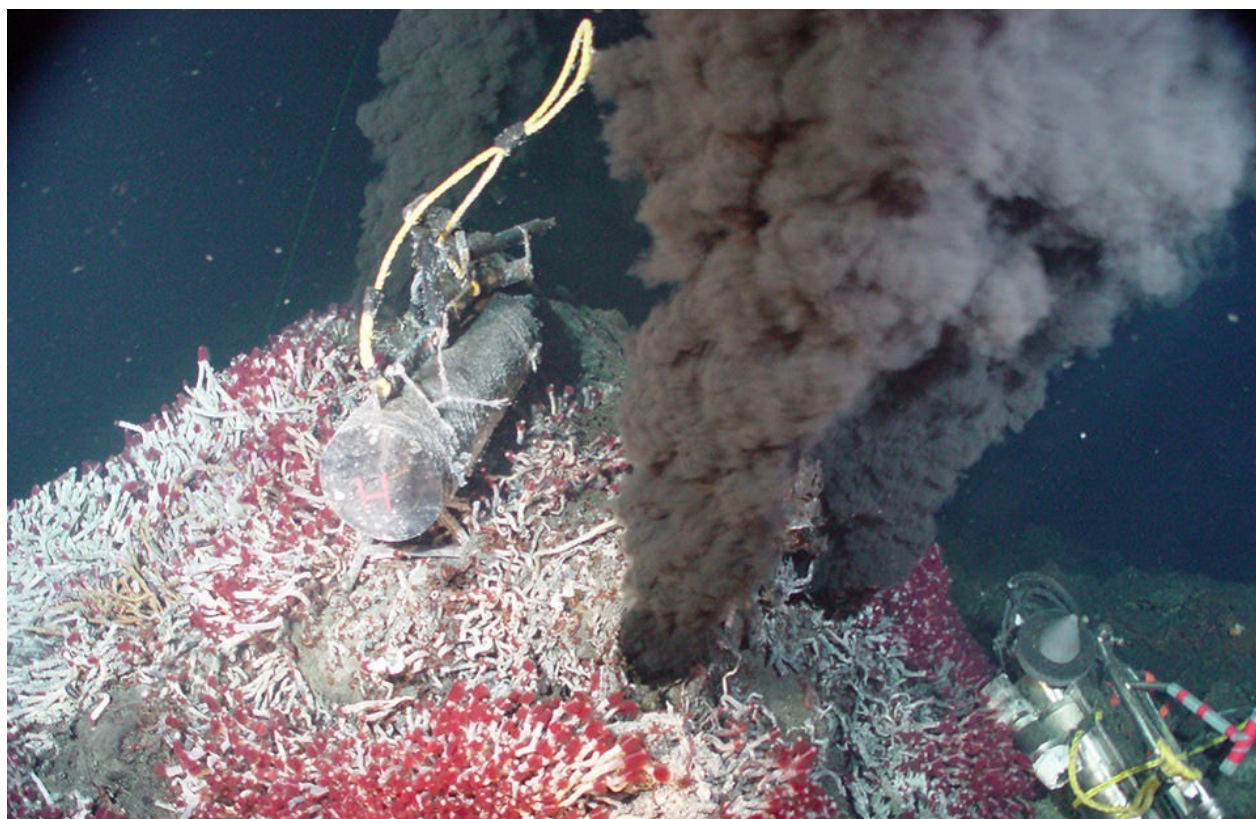
October 25, 2017, 1411

Rob sent me another email with a link in it to a [YouTube nature video](#) — all about white and black smokers. I learned about those things in school — super freaky deep sea environments with no sunlight, pumping sulfur and other stuff into the water...

In 1977, a study of a deep-sea trench near the Galápagos shocked marine biologists and the entire scientific community. There was life that we had never seen before, never even knew could exist! It was (and still is) living off of the heat and chemical reactions which exist near deep water hydrothermal vents.

These hydrothermal vents occur near tectonically-active plate boundaries, otherwise known as *spreading ridges* or their counterpart: *subduction zones*. A spreading ridge is where newly-formed tectonic plates emerge and then diverge; subduction zones are where they converge and plunge back into the heated depths of the planet. The earth, as it turns out, is an actively roiling mass of heat and rock; we just ride on top.

These ridges and subduction zones are quite unstable, and the earth's crust around them is crushed and porous. The ocean that sits on top of these zones percolates down into the cracked plate boundaries and becomes super-heated as well as ultra-pressurized. Lots of heat and lots of pressure combined with primal, mineral-rich magma makes for a very interesting mineral & chemical stew.



Black smoker and chimney. Photo credit: NOAA

Chief among these compounds is hydrogen sulfide, one of the deadliest compounds known to exist. In the deep ocean, however, it sustains a particular bacterium which, oddly, thrives on the stuff.

More specifically: it thrives on molecular hydrogen. Through a process called *methanogenesis*, it converts the molecular hydrogen into water, CH₄ (methane) and other organic compounds. Evidently, these little bacteria are tasty, as they are eaten by the bigger bacterium, and so on.

This creates quite an exotic food chain that, according to scientists of that time, *should not exist*.

Until these things were discovered in 1977, it was assumed that all life on earth depended on the sun. It was thought that the sun provided the energy for plants, people, and bacteria, even that of the deep sea. An organic material called *marine snow* would filter down to the darkest depths and feed the blind, freaky looking scavengers there.

These organisms exist in a world entirely absent of the sun's influence and toxic to every other living thing. It's as if we hosted an alien planet within our very own.

The question is obvious, and I can see why Rob sent this to me: *if black smokers can support an ecosystem here, can they sustain it elsewhere?* Like, maybe Enceladus?

From: Rob Conery rob @redfour.io
Subject: Enceladus: Some detail
To: Dee Yan yand@redfour.io
Date: October 25, 2017

Now that you have the data in and normalized let's see if you can find the exact flyby times.

As I think I told you, Cassini made 23 total flybys of Enceladus, although if you look online, most references say there were *only* 22. Not sure why this is, see if you can find out.

What I specifically need is to know the *precise time* that Cassini was closest to Enceladus, for each flyby. The more accurate, the better. The flyby information you put together will be used by our Analysis Team and the little project they are putting together, which is what I'm down here pitching.

Oh crap, they found me. Gotta run back to the meeting. I was trying to hide out here in the bathroom but... well, I guess that's gross. Anyway: sorry *gotta run*.

Start with the Mission Plan. I need the precise flyby times ASAP.

R

FLYBYS

October 25, 2017, 1536

Now I get to play around with the mission data for a bit. This is what I've been looking forward to!

Rob has asked me to find the precise flyby times, which I should be able to do with the mission plan data, I would think. Weird that they don't already know this somehow? I'll have to ask him about that.

I know what M. Sullivan means about ETL being tedious, laborious and a significant pain in the ass... until you get to see the data! The entirety of Cassini's history, minute by minute, day by day.

I spent the last few hours looking over various ways to filter data in Postgres, and I think I have a handle on this:

```
select targets.description as target,  
time_stamp,  
title  
from events  
inner join targets on target_id=targets.id;
```

With this query, I'm joining the **targets** table as I want to know what planet or moon the mission plan is focusing on. The title should tell me what's happening... I'm looking specifically for something that suggests a flyby.

Red-4 Planetary Orbital Observations		
target	time_stamp	title
Saturn	2004-05-14 18:40:00-07	MAPS Survey
DustRAM direction	2004-05-14 18:40:00-07	dust exploration - saturn
Other	2004-05-14 18:40:00-07	MAG_MAPS_Survey_Measurements
co-rotation	2004-05-14 18:40:00-07	MAPS SUR Campaign
Saturn	2004-05-14 18:40:00-07	RPWS Outer Survey
rings(general)	2004-05-14 20:01:00-07	OBSERVATIONS OF DIFFUSE RINGS AND SATELLITES PRE-S
rings(general)	2004-05-14 20:01:00-07	Early search for E Ring
Saturn	2004-05-14 22:31:00-07	MOSAIC_09_PER_04
rings(general)	2004-05-15 02:47:00-07	OBSERVATIONS OF DIFFUSE RINGS AND SATELLITES PRE-S
rings(general)	2004-05-15 02:47:00-07	Early search for E Ring
Saturn	2004-05-15 04:02:00-07	MAPS Survey
Saturn	2004-05-15 05:17:00-07	CIRS_000SA_TEMPSIT
Saturn	2004-05-15 05:17:00-07	Saturn Methane imaging
InstrumentCalibration	2004-05-15 05:17:00-07	MAG/SCAS_Cal
Other	2004-05-15 10:17:00-07	MAG_MAPS_Survey_Measurements
Earth	2004-05-15 13:08:00-07	RSS USO Characterization plus PIM
Other	2004-05-15 15:23:00-07	RETARGETABLES FOR SATELLITES DISCOVERED ON APPROAC

Swoon. That's neat! Look at that! "What did you do at work today, Dee?"

"Why I looked at data related to Saturn methane imaging and an early search for the E-ring. Yourself?"

OK, *concentrate*. I need to constrain the data now and try a "fuzzy" search on the term "flyby" or "fly by" or something. We'll use title for that:

```
select targets.description as target,
time_stamp,
title
from events
inner join targets on target_id=targets.id
where title like '%flyby%' or '%fly by%';
```

```
Red:4 Planetary Orbital Observations
enceladus=# select targets.description as target,
enceladus=# time_stamp,
enceladus=# title
enceladus=# from events
enceladus=# inner join targets on target_id=targets.id
enceladus=# where title like '%flyby%' or '%fly by%';
ERROR:  invalid input syntax for type boolean: "%fly by%"
LINE 6: where title like '%flyby%' or '%fly by%';
                                         ^
enceladus=#
```

Right. Slow down, Dee. You have to specify every column and condition in a where clause. You can't just chain things together, Postgres won't know what you're trying to do.

Let's try that again:

```
select targets.description as target,
time_stamp,
title
from events
inner join targets on target_id=targets.id
where title like '%flyby%'
or title like '%fly by%';
```

Much better! Look at that!

Red:4 Planetary Orbital Observations		
target	time_stamp	title
Phoebe	2004-06-11 17:33:37-07	Phoebe targeted flyby
Iapetus	2004-12-31 01:53:00-08	Iapetus close flyby 00B/00C global color observati
Iapetus	2004-12-31 08:00:00-08	Iapetus close flyby 00B/00C imaging
Iapetus	2004-12-31 14:00:00-08	Iapetus close flyby 00B/00C global mapping
Iapetus	2004-12-31 18:30:00-08	Iapetus close flyby 00B/00C imaging
Iapetus	2004-12-31 19:00:00-08	Iapetus close flyby 00B/00C local-color imaging
Iapetus	2004-12-31 22:00:00-08	Iapetus close flyby 00B/00C imaging
Iapetus	2004-12-31 22:30:00-08	Iapetus close flyby 00B/00C context imaging
Iapetus	2005-01-01 01:00:00-08	Iapetus close flyby 00B/00C global mapping and gra
Iapetus	2005-01-01 04:00:00-08	Iapetus close flyby 00B/00C graylight imaging
Titan	2005-02-15 02:12:53-08	Warmup for T3 flyby
Enceladus	2005-03-09 07:08:01-08	Enceladus targeted flyby
Enceladus	2005-03-09 08:05:00-08	Enceladus targeted flyby
Enceladus	2005-07-14 17:55:22-07	Enceladus targeted flyby
Enceladus	2005-07-14 18:41:00-07	Enceladus targeted flyby
Tethys	2005-09-24 00:41:58-07	Tethys targeted flyby
Tethys	2005-09-24 01:40:00-07	Tethys targeted flyby
Hyperion	2005-09-26 00:24:46-07	Hyperion targeted flyby
Dione	2005-10-11 15:42:02-07	Dione targeted flyby near-C/A imaging

I can hear M. Sullivan now (in Alan Rickman's Snape voice): *Never trust ... a fuzzy search, Dee, the people who created this data have done their very best to ... corrupt ... it and, likewise, your... soullll.*

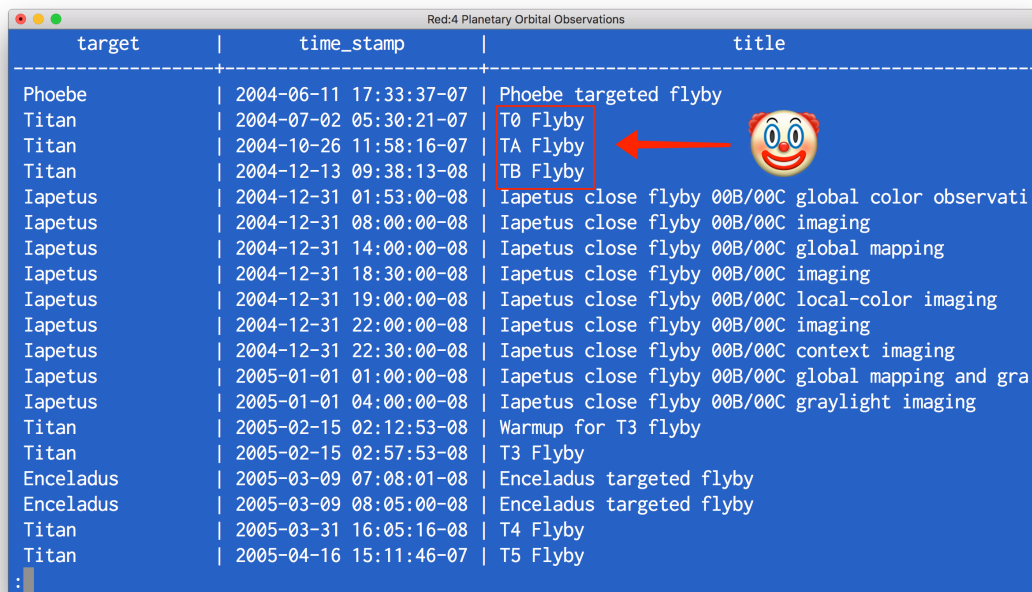
I'm using the term **flyby** in the criteria, which is lower case. The query I've used here is *case sensitive* because I'm using **like**, which is case sensitive by default. I'd bet lunch that there's a **Fly by** or possibly **FLY bY** value in there somewhere! I need this query to be case *insensitive*:


```

select targets.description as target,
time_stamp,
title
from events
inner join targets on target_id=targets.id
where title ilike '%flyby%'
or title ilike '%fly by%'
order by time_stamp;

```

Ha! It's hard being right all the time!



target	time_stamp	title
Phoebe	2004-06-11 17:33:37-07	Phoebe targeted flyby
Titan	2004-07-02 05:30:21-07	T0 Flyby
Titan	2004-10-26 11:58:16-07	TA Flyby
Titan	2004-12-13 09:38:13-08	TB Flyby
Iapetus	2004-12-31 01:53:00-08	Iapetus close flyby 00B/00C global color observati
Iapetus	2004-12-31 08:00:00-08	Iapetus close flyby 00B/00C imaging
Iapetus	2004-12-31 14:00:00-08	Iapetus close flyby 00B/00C global mapping
Iapetus	2004-12-31 18:30:00-08	Iapetus close flyby 00B/00C imaging
Iapetus	2004-12-31 19:00:00-08	Iapetus close flyby 00B/00C local-color imaging
Iapetus	2004-12-31 22:00:00-08	Iapetus close flyby 00B/00C imaging
Iapetus	2004-12-31 22:30:00-08	Iapetus close flyby 00B/00C context imaging
Iapetus	2005-01-01 01:00:00-08	Iapetus close flyby 00B/00C global mapping and gra
Iapetus	2005-01-01 04:00:00-08	Iapetus close flyby 00B/00C graylight imaging
Titan	2005-02-15 02:12:53-08	Warmup for T3 flyby
Titan	2005-02-15 02:57:53-08	T3 Flyby
Enceladus	2005-03-09 07:08:01-08	Enceladus targeted flyby
Enceladus	2005-03-09 08:05:00-08	Enceladus targeted flyby
Titan	2005-03-31 16:05:16-08	T4 Flyby
Titan	2005-04-16 15:11:46-07	T5 Flyby

I know M. Sullivan saw that one.

Each flyby was given a name, something like TA, TB, T1, or E1. The first letter(s) represents the target, the letter or number following the incremented count.

I can take advantage of this pattern... *maybe*. First, let's see how, and then I'll know if it's a good idea.

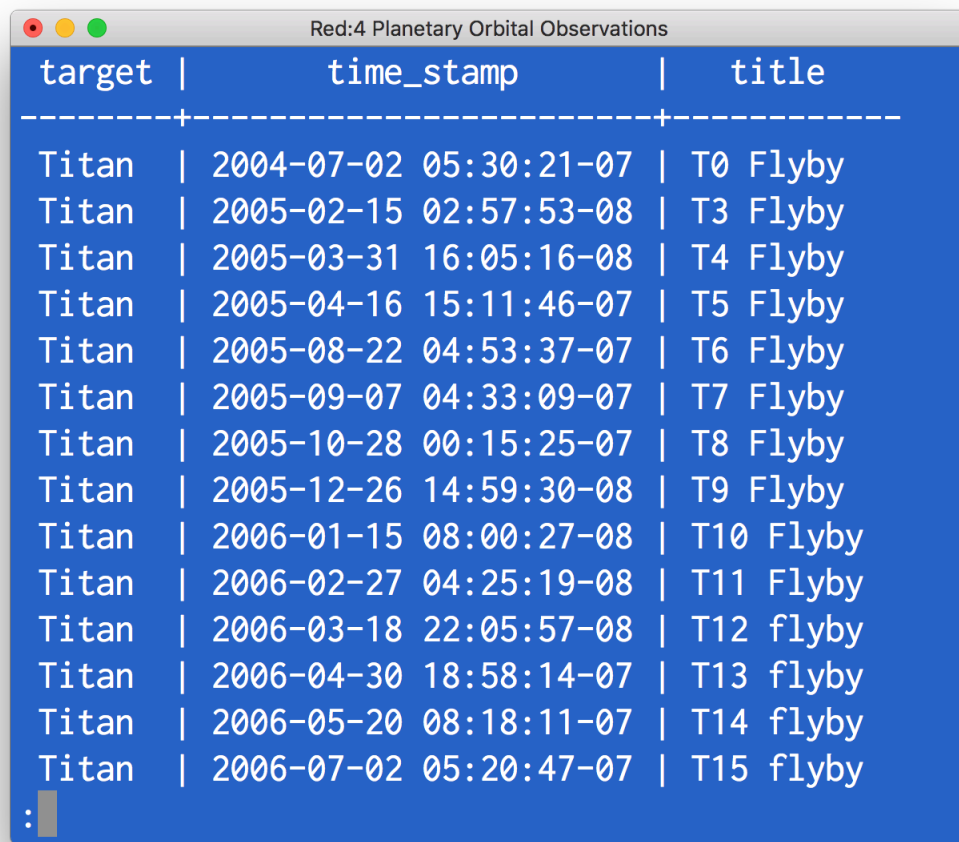
Titan flybys all start with T and usually have a number following them. Instead of constructing an elaborate where clause, I can just use regex:

```
select targets.description as target,  
time_stamp,  
title  
from events  
inner join targets on target_id=targets.id  
where title ~* '^T\d.*? flyby'  
order by time_stamp;
```

Finally, something I *do* know, regex! I also kind of hate it but whatever it's like SQL you just gotta know it... or at least know when to Google it if you need to.

This query is checking all titles that start with T and have one or more numbers following that end with the term “flyby.” I’m getting around the case sensitivity issue by using the `~*` comparison operator, which tells Postgres this is a case-insensitive match operation.

Check it out:



target	time_stamp	title
Titan	2004-07-02 05:30:21-07	T0 Flyby
Titan	2005-02-15 02:57:53-08	T3 Flyby
Titan	2005-03-31 16:05:16-08	T4 Flyby
Titan	2005-04-16 15:11:46-07	T5 Flyby
Titan	2005-08-22 04:53:37-07	T6 Flyby
Titan	2005-09-07 04:33:09-07	T7 Flyby
Titan	2005-10-28 00:15:25-07	T8 Flyby
Titan	2005-12-26 14:59:30-08	T9 Flyby
Titan	2006-01-15 08:00:27-08	T10 Flyby
Titan	2006-02-27 04:25:19-08	T11 Flyby
Titan	2006-03-18 22:05:57-08	T12 flyby
Titan	2006-04-30 18:58:14-07	T13 flyby
Titan	2006-05-20 08:18:11-07	T14 flyby
Titan	2006-07-02 05:20:47-07	T15 flyby

121 rows of Titan flybys, all with mixed casing. Seems like this is an excellent choice for isolating flybys, doesn't it? Being a good data person, however, I want to be sure I'm not being too restrictive with our filter. Regex can do that to you, and I really don't want to be overlooking flybys!

I'll expand the filter and let any word in, just to be sure. This will be a bit more greedy, but that's OK for what we're trying to do right now:

```

select targets.description as target,
time_stamp,
title
from events
inner join targets on target_id=targets.id
where title ~* '^T[A-Z0-9_].*? flyby'
order by time_stamp;

```

Well, that's not very fun:

Red:4 Planetary Orbital Observations		
target	time_stamp	title
-----+-----+-----		
Titan	2004-07-02 05:30:21-07	T0 Flyby
Titan	2004-10-26 11:58:16-07	TA Flyby
Titan	2004-12-13 09:38:13-08	TB Flyby
Titan	2005-02-15 02:57:53-08	T3 Flyby
Titan	2005-03-31 16:05:16-08	T4 Flyby
Titan	2005-04-16 15:11:46-07	T5 Flyby
Titan	2005-08-22 04:53:37-07	T6 Flyby
Titan	2005-09-07 04:33:09-07	T7 Flyby
Tethys	2005-09-24 00:41:58-07	Tethys targeted flyby
Tethys	2005-09-24 01:40:00-07	Tethys targeted flyby
Titan	2005-10-28 00:15:25-07	T8 Flyby
Titan	2005-12-26 14:59:30-08	T9 Flyby
Titan	2006-01-15 08:00:27-08	T10 Flyby
Titan	2010-01-28 22:08:49-08	Titan Flyby
:		



Someone decided to sneak in a TA and TB flyby name! I wonder if there are a TC and TD as well? Just when you think people are going to, you know, be a little consistent with their data they prove you wrong!

I know enough about regex to know when to stop. Maybe someone else around here can help, but I think it's easier to just go back to the old query:

```
select targets.description as target,  
time_stamp,  
title  
from events  
inner join targets on target_id=targets.id  
where title ilike '%flyby%'  
or title ilike '%fly by%'  
order by time_stamp;
```

At least with this query I know I'll be getting all the flyby information I need. Speaking of, I need to add a few expressions to this query, so I can work with the data later on.

```
select targets.description as target,  
event_types.description as event,  
time_stamp,  
time_stamp::date as date,  
title  
from events  
left join targets on target_id=targets.id  
left join event_types on event_type_id=event_types.id  
where title ilike '%flyby%'  
or title ilike '%fly by%'  
order by time_stamp;
```

Running this will result in a big splash of text, so I need to turn on expanded view with `\x`:

```
Red:4 Planetary Orbital Observations
-[ RECORD 1 ]-----
target      | Phoebe
event       | Phoebe targeted flyby
time_stamp  | 2004-06-11 17:33:37-07
date        | 2004-06-11
title       | Phoebe targeted flyby
-[ RECORD 2 ]-----
target      | Titan
event       | MAG Titan observation
time_stamp  | 2004-07-02 05:30:21-07
date        | 2004-07-02
title       | T0 Flyby
-[ RECORD 3 ]-----
target      | Titan
event       | MAG Titan observation
:
```

Looks good so far. I've added target information as well as a cast of the **time_stamp** to a **date**. I'm doing the latter because I'll probably need to see results for a set of dates.

Before I quit for the day, I need to go through the data and spot check it. There are a few things I know about the flybys, specifically that the first Enceladus pass was February 17, 2005. That's when they spotted the weird magnetometer reading. I should be able to see that exact date in this mission plan, as well as the two subsequent flybys in March and July of that year.

Red:4 Planetary Orbital Observations	
time_stamp	2005-02-15 02:12:53-08
date	2005-02-15
title	Warmup for T3 flyby
-[RECORD 15]-----	
target	Titan
event	MAG Titan observation
time_stamp	2005-02-15 02:57:53-08
date	2005-02-15
title	T3 Flyby
-[RECORD 16]-----	
target	Enceladus
event	Enceladus
time_stamp	2005-03-09 07:08:01-08
date	2005-03-09
title	Enceladus targeted flyby
-[RECORD 17]-----	
target	Enceladus
event	Enceladus
:	

Uh oh. We have a Titan flyby on February 15 and then the second Enceladus flyby on March 9. Where's the first one!?!?!?

The only thing I can do at this point is to crosscheck the source. Let's investigate this data a bit more! I'll use that date trick I read about and constrain by a given day:

```
select target, title, date
from import.master_plan
where start_time_utc::date = '2005-02-17'
order by start_time_utc::date
order by time_stamp;
```

I'm querying the source table here, restricting the date, so I can quickly scroll down and see what's happening:

```
Red:4 Planetary Orbital Observations
target | Enceladus
title  | Enceladus
date   | 17-Feb-05
-[ RECORD 2 ]-----
target | Enceladus
title  | Enceladus FP1 and FP3 maps, high spectral resoluti
date   | 17-Feb-05
-[ RECORD 3 ]-----
target | Enceladus
title  | ICYLON: Icy Satellite Longitude / Phase Coverage
date   | 17-Feb-05
-[ RECORD 4 ]-----
target | Saturn
title  | MAPS Survey
date   | 17-Feb-05
-[ RECORD 5 ]-----
target | Saturn
title  | Obtain wideband examples of lightning whistlers
:
```

There it is, right where it's supposed to be. What the heck is a *lightning whistler*? This job is going to be too much fun...

Unfortunately, there's nothing in the data that suggests this is a flyby. But looking at the JPL website, *it is indeed considered a flyby*:

This map of the surface of Saturn's moon Enceladus illustrates the regions that will be imaged by Cassini during the spacecraft's first very close flyby of the moon on Feb. 17, 2005. At closest approach, the spacecraft is expected to pass approximately 1,180 kilometers (733 miles) above the moon's surface. Enceladus is 505 kilometers (314 miles) across.

It was never actually given a name, however, and the flyby that occurred just after this one in March of 2005 was named E-1. I guess the February 2005 one is Rob's zeroth flyby. They could've made that a little easier.

People do horrible things to data, Dee.

Thanks, M. Happy that you've become part of my subconscious.

This was a fun day. Frustrating, but fun. I wish I had more time to plumb through that data, but I gotta get some sleep. I feel like someone scraped me off the bottom of a shoe.

I think I might ask for my title to be changed to Lightning Whistler...

From: M. Sullivan sullz@redfour.io
Subject: Sargeable Queries For Fun and Profit
To: Dee Yan yand@redfour.io
Date: October 25, 2017

Data investigation is fun, isn't it? Even though you're querying it with what amounts to a hammer covered in broken glass and dog vomit. This is clean, structured, *beautiful* data, M. Yan, let's treat it with some respect, shall we?

Please read up on *sargeable* vs. *non-sargeable* queries. If I have a number one concern in this company, aside from everyone else that works here, it would be the carefree and pervasive use of **like** followed by %.

Please have a look at *full-text indexing* and *views*. They are the data investigator's best friend. One type of view, in particular, might be very helpful for you: the *materialized* view. We value speed and efficiency here; slow down and do the necessities, please.

Best, S

TOMORROW IS ONLY A DAY AWAY

October 25, 2017, 1952

That will have to wait until tomorrow. I'm exhausted. I'll take a quick look at full-text indexes and "sergeant query hole" (I like that term better) on the bus back.

After I stop off at the Horseshoe on Chestnut.

I have to admit, though: *this is the most fun I've had at work in ... forever!* Shuffling through data from Saturn! Are you kidding me!

Reminds me so much of my dad, and those summer nights up at Pinecrest Lake...

When I was 6, my dad bought part of a land-lease at a high mountain lake in the Sierra Nevada named Pinecrest. My sister and I used to spend weekends up there at our friend's place, a 90-year-old cabin built by their great-grandfather back in 1901. Now we had our own.

There were no roads. You had to hike in or take a boat, and the lake was nestled between two massive, imposing granite peaks covered with pines and osprey nests.

During the days, we would jump off rocks and swim across the cove, diving 10 feet down looking for lures that fisherman would snag on dead logs while trolling for trout. Later in the day we would run back behind the cabin, into the groves of cedar and pine, turn over rotten logs and count the gross bugs we found. Once we heard a rattlesnake and ran away screaming, only to come back later to see if we could make it rattle again. Our parents weren't thrilled about that.

At night, our dads would make a giant fire, and we would make s'mores, dancing out of the way of the smoke that always seemed to chase you. I would watch the orange

embers float up into the night, wrapping around the outstretched limbs of the sugar pines, making the needles dance and then rising into space, becoming stars.

That's when my dad would bring out his telescope.

The sky over the lake in August was crystal clear save for a blanket of smoke that lay over the lake caused by fires from the campground. We used to stare up at the deep, dark summer night sky and play a game to see who could count the most satellites making their way so far overhead. The ISS was worth 10 points, the Hubble 20. Regular old satellites were 5. For some reason, I could always find the Hubble.

On my dad's 50th birthday, my mom bought him his first "real" telescope: a Celestron NexStar 4SE. It was super high-tech for the time with a motor and onboard computer that would automatically rotate and point the telescope at a given constellation.

A month after he got the telescope, he bought a set of Plossl eyepieces that also came with filters, so you could see exciting surface features on the moon and planets. I remember my dad muttering to himself "locate with 32, then zoom wit da 17".

"Dee, I found it, come look." It was prime time for viewing Saturn, something my dad had promised me for as long as I could remember. We had come up this precise weekend to be here at this exact time, as Saturn would be prime viewable at 9:42 pm, which was past my bedtime, but it didn't matter as this was Pinecrest, and it was summer and here was Saturn.

The motor on the telescope made a tick-tick-ticking sound as it compensated for the earth's rotation, and it was hard to keep my eye in the center of the eyepiece. I kept seeing a blurry yellow flash go by, my dad whispering in my ear "do you see em? Dee? See da buggah?" His pidgin coming to life as he got excited.

And then I saw it: *Saturn*. The rings were as bright as any picture and the planet itself a dull yellow. I could see the shadow of the rings traced across the planet's polished surface, like the rings bumped into it a few too many times, cutting a groove. This wasn't a picture, *this was the real thing*, and I was seeing it with my very own eyes, right there in the sky.

It looked just like the Voyager pictures in the books dad used to read my sister and me before bed. I loved to hear about the stars and planets from the DK Science books and begged endlessly to hear just one more chapter. My sister liked *Little House on the Prairie*. Those books were OK, but I always added a part in my head where Laura's pa, Charles, sitting on his back under the bright country stars with no city lights to dim them, would tell Laura all about the Milky Way in the same way our dad did up at Pinecrest.

"If you make your eyes see the sky like a pool of lights, deep and wide, instead of a flat painting... you feel like you could fall right into it..."

We would talk about Saturn and Jupiter the most as they were the biggest and easiest to spot. Like most kids, Saturn was my favorite. My dad loved to tell me stories about the day that Voyager beamed back its first pictures of the ringed planet. They were all over the news, the first time we had ever seen Saturn so clearly, and so close.

It took people's breath away.



Photo credit: NASA/JPL

Voyager's images are spectacular, but I prefer the nights up at Pinecrest, looking through my dad's telescope, seeing it with my own eye.

"Do you see the rings, Dee?" He asked me again.

I sure do, dad, I sure do. I think I'm going to enjoy this job.

A BENT FIELD

October 26, 2017, 1041

I decided to slow down and read up on everything I possibly could regarding string-based searches, views and generally good Postgres habits. I also went to a user group! It's been a good week, but I'm always reminded just how much I don't know. Which is a lot.

Data science and analytics is about finding patterns and exploiting relationships that, at first, don't seem apparent. Before you do any of that, however, you have to have good data. The whole process is like digging for gold: the first problem is knowing where to look.

When I was 10, my dad used to tell me stories about working as a geologist on the wreck of the *Niantic*, a whaling vessel that was scuttled in San Francisco Bay back in 1849. The crew abandoned the ship, like so many other crews of so many other ships anchored in San Francisco Bay in 1849. Gold Fever struck often, and derelict vessels became a problem.

The *Niantic* was put aground near the corner of Clay and Sansome (the coastline of San Francisco was much different back then) and turned into a hotel and finally a storehouse. In 1851, it was burned to the bilge and buried, where it lay quietly for the next 127 years.

In 1978, my dad worked on the engineering team that helped excavate it. The financial district was growing, and ship after ship was dug up as they set the piers and

casings for these gigantic banks and insurance companies. The *Niantic* was the furthest inland, and one of the more interesting.

An excavation team was brought in to investigate, recover items and, unfortunately, to help facilitate the ship's disposal. They found drums of oil, unopened cases of champagne and whiskey, some items from the last voyage of the ship, other items from the ship's time as a storehouse.

I remember the bottle my dad showed me that had been stowed in her foreword storage, buried under the city for over 100 years. Green, dirty, full of some kind of liquid. He carried it in a lined ash wood case and told me stories of the sailors who drank from bottles just like this, heads full of dreams and hearts full of desire for wealth.

"We never knew this was right beneath our feet. How many other ships do you think are out there, Dee, under the hotels and skyscrapers of downtown? Hundreds, Dee, hundreds. They sank da buggahs jus li'dat!" and he claps his hands in front of him. "We nevah can get 'em now... but bumbye... if can... can" and he sits back in his chair, looking at me thoughtfully.

If can, can. If no can, no can. Another one of my dad's favorite pidgin phrases. Simple, direct and to the point.

Those sailors who drank from the dirty green bottles came to California with dreams of finding gold in the Sierra Nevada foothills. They sailed into the foggy bay and abandoned their ship with everything they had in a single bag, and made for the hills. "Forty-Niners" they were called, thanks to the year they flooded into the San Francisco Bay from all over the world.



Photo from State Library of New South Wales, Photographer unknown

These people surveyed the land for clues to the riches buried below just as my dad searched for clues to them over a century later.

They studied the streams and outcrops for gold trace: glimmer in the sunlight or “the smell of riches.” They used their intuition, science as they knew it, and just plain dumb luck. Sometimes the clues were obvious, other times they would lead you nowhere.

In many ways, the Cassini mission did the same thing, pointing its sensors and cameras at everything it could while flying around Saturn and its moons. It was on a cosmic treasure hunt, but instead of gold, the Cassini team was (and still is) searching for knowledge.

That’s precisely what I’m doing now with the Cassini data, and it’s so much fun.

E-O, THE FIRST ENCELADUS FLYBY

October 26, 2017, 1123

Reading the last sentence from Rob's email a few days back: *Start with the Mission Plan*. I need the precise flyby times ASAP. I'm getting close, but I'm not there yet. That's today's task: pinpoint the Enceladus flybys, and I'll be using the mission plan data to do it. I've also been reading up on how the team actually decided what it was going to do, when. The inside scoop, if you will.

There was a lot of drama! I suppose that's understandable.

We know, now, that the February 17, 2005, flyby of Enceladus was pivotal. That was the first one. Back then, however, it was a minor detail that no one really cared much about. Their focus was on Titan and Saturn's rings. Michele Dougherty (Cassini Magnetometer Project lead and my newfound hero) [was asked](#) what the team expected to find at Enceladus during this first flyby, and she replied:

Nothing... nothing. It was a distant flyby... about 1000 km from the surface. It's a really small moon, it only has a diameter of 500 kilometers ... we didn't expect to see anything at all. And, I have a confession to make: we didn't look at the data for about 24 hours afterward... we were doing other things.

There it is: it was a distant flyby. Maybe it's not showing up in the mission plan data as a flyby because it was just a casual "hey look over there" kind of thing instead of a planned event? Cassini's mission was all about Titan and the Rings. Enceladus literally stole the show, shaping the next equinox and solstice missions almost completely.

I decided to join up the **event_types** table to see if I could find something to help me pinpoint Enceladus flyby details:

```
select
targets.description as target,
events.time_stamp,
event_types.description as event
from events
inner join event_types on event_types.id = events.event_type_id
inner join targets on targets.id = events.target_id
where events.time_stamp::date='2005-02-17'
order by events.time_stamp;
```

I'm narrowing this query to a single day, and I'm doing that by taking advantage of casting. I want to turn the timestamp into a single day only instead of a day with hours, minutes, seconds and so on. All I have to do is cast a **timestampz** to a **date** then I'm able to capture all timestamp data for that day!

The date I'm specifically curious about is February 17, 2005. I want to see what even type is recorded for the Enceladus flyby and if it mentions anything interesting.

I believe that it does:

Red:4 Planetary Orbital Observations		
target	time_stamp	event
Enceladus	2005-02-17 00:00:29-08	Enceladus
Enceladus	2005-02-17 00:15:29-08	CIRS FP1 integration / FP3 map
Enceladus	2005-02-17 00:15:29-08	Icy satellite longitudinal coverage
Saturn	2005-02-17 00:53:17-08	Observing neutral molecules in the inner magnetosphere
Saturn	2005-02-17 00:55:00-08	RPWS lightning search
Saturn	2005-02-17 01:14:00-08	Highest resolution saturn kilometric radiation observation
Enceladus	2005-02-17 01:30:29-08	Icy satellite longitudinal coverage
Enceladus	2005-02-17 01:30:29-08	Enceladus
Enceladus	2005-02-17 01:30:29-08	Enceladus
Enceladus	2005-02-17 02:15:29-08	Enceladus
Enceladus	2005-02-17 02:15:29-08	Icy satellite surface map
Enceladus	2005-02-17 02:15:29-08	Limb topography
Enceladus	2005-02-17 02:15:29-08	Limb topography
Enceladus	2005-02-17 03:00:29-08	MAPS Campaign
Enceladus	2005-02-17 03:00:29-08	Thermal stabilization period
Enceladus	2005-02-17 03:00:29-08	Enceladus closest approach observation
Enceladus	2005-02-17 03:05:29-08	Enceladus dust observation
Enceladus	2005-02-17 03:05:29-08	Icy satellite exosphere
Enceladus	2005-02-17 03:05:29-08	Enceladus dust observation
Other	2005-02-17 03:30:00-08	Magnetospheric survey
RingE	2005-02-17 03:54:34-08	E ring observation

Well OK then! It's telling me straight up: *Enceladus closest approach observation*. The plan data surrounding the closest approach shows me a lot as well, specifically which instruments were active and recording things right at closest approach. I think I'll need to remember that.

Some other activities were going on with other targets. Some were calibrations, others were remote observations of Saturn and its rings.

I need to restrict these results so that we only have Enceladus as the target:

```
select
targets.description as target,
events.time_stamp,
event_types.description as event
from events
inner join event_types on event_types.id = events.event_type_id
inner join targets on targets.id = events.target_id
where events.time_stamp::date='2005-02-17'
and targets.description = 'enceladus'
order by events.time_stamp;
```

A few exciting things about this query:

- You can chain **and** clauses together to further restrict a query. These are inclusive so the more you add, the smaller (potentially) the results get.
- I don't have any results! Why do you think that is, Dee?

```
Red:4 Planetary Orbital Observations
enceladus=# set search_path=plan;
SET
enceladus=# select
enceladus=# targets.description as target,
enceladus=# events.date,
enceladus=# event_types.description as event
enceladus=# from events
enceladus=# inner join event_types
enceladus=#   on event_types.id = events.event_type_id
enceladus=# inner join targets
enceladus=#   on targets.id = events.target_id
enceladus=# where date_part('year',events.date)=2005
enceladus=# and date_part('month',events.date)=2
enceladus=# and targets.description = 'enceladus'
enceladus=# order by events.date;
 target | date | event
-----+-----+-----
(0 rows)

enceladus=#
```

Why yes! PostgreSQL is case sensitive which is something I knew and completely forgot like the all-knowing seasoned professional DBA that I am.

Just one reason why doing a string comparison query is a bad idea. Sigh, I think M. Sullivan is brainwashing me... but I also think it's true: these queries are fragile.

They're also slow. Postgres is a fast system, but the engine needs to do a lot of work to compare strings. But there's another reason, too. What would happen if someone were to change the **description** in our lookup table (**targets**)? Nothing that'd make me happy trying to look for the old value, that's what.

It's much better to find the key value and use that in our query. It's an integer, has an index, and won't change.

To find that key value, I can look in the **targets** table for a primary key that defines the "Enceladus" target:

```
Red:4 Planetary Orbital Observations
enceladus=# select * from targets where description = 'Enceladus'
;
-[ RECORD 1 ]-----
description | Enceladus
id          | 40
enceladus=#
```

40, perfect. Now I can reference that in my query instead:

```
--using the id as filter instead
select
targets.description as target,
events.time_stamp,
event_types.description as event
from events
inner join event_types on event_types.id = events.event_type_id
inner join targets on targets.id = events.target_id
where events.time_stamp::date='2005-02-17'
and targets.id = 40
order by events.time_stamp;
```

Much better, much faster.

SARGEABLE AND NON-SARGEABLE QUERIES

Sometimes you can't get away from searching with a fuzzy string parameter, which is OK. This is a database after all. The trick is to be sure that you're using some kind of optimization. Otherwise, Postgres has a lot of work to do.

The word “sargeable” is goofy DBA-speak for Search ARGument ABLE. Put another way: can the string query be optimized? A sargeable query can be optimized, a non-sargeable query can't. The reason is due to the nature of the query itself.

If I wanted to find all events that started with the term closest I could use this:

```
select *  
from events  
where description like 'closest%';
```

There's no wildcard before the term closest, so the query planner would be able to optimize this search if we had an index on the **description** field. I don't have that in my last query, however, so it's non-sargeable and to find the rows I'm looking for, Postgres has to search every row in my table, running a string comparison. This is called a sequential scan.

That's not a big deal if the database is small, but as it grows the search will begin to slow down. Over time, maybe days, maybe years, the table will grow to the point that searches are too slow to run.

Bottom line: try to use a sargeable query when you can, especially in production. If this query were sent up with Cassini II, I would be fired, for sure.

USING A VIEW TO MAKE QUERYING EASIER

M. Sullivan suggested that I look into views as well, and I did. They seem simple enough, just stored snippets of SQL. You can create one just by passing in a **select** statement to a **create view** statement:

```
drop view if exists enceladus_events;
create view enceladus_events as
select
events.time_stamp,
events.time_stamp::date as date,
event_types.description as event
from events
inner join event_types
on event_types.id = events.event_type_id
where target_id=40
order by time_stamp;
```

I had to make a couple of changes to the underlying query to make this view useful. The first was to remove the join for the **targets** table and all the target information from the result. The view's name says it all: **enceladus_events** . No need to have that with the results.

Removing the join and using an integer key lookup makes things even faster and simplifies the query itself.

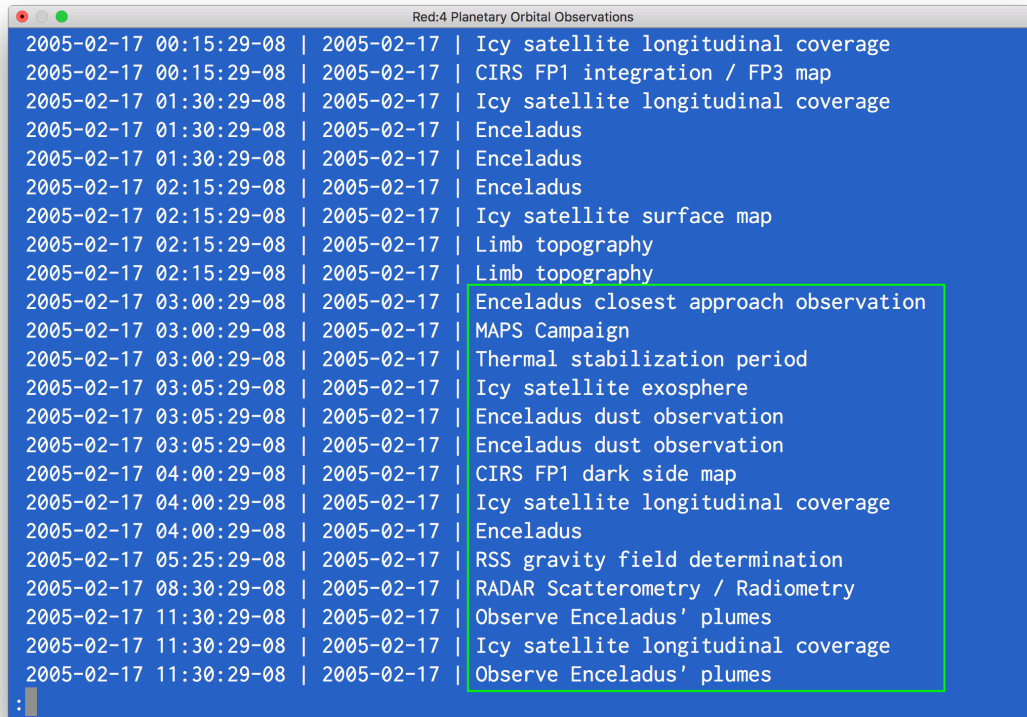
Next, I want to make sure I can query easily for a given day, so I aliased the **time_stamp** column which will allow me to query by date directly:

```
select * from enceladus_events where date='2005-02-17';
```


Much nicer, Dee. Good work. The format for the date isn't important necessarily, Postgres just needs to be able to parse it. I could also use this query with the same result:

```
select * from enceladus_events where date='2/17/2005';
```

Speaking of, look at the results:



2005-02-17 00:15:29-08	2005-02-17	Icy satellite longitudinal coverage
2005-02-17 00:15:29-08	2005-02-17	CIRS FP1 integration / FP3 map
2005-02-17 01:30:29-08	2005-02-17	Icy satellite longitudinal coverage
2005-02-17 01:30:29-08	2005-02-17	Enceladus
2005-02-17 01:30:29-08	2005-02-17	Enceladus
2005-02-17 02:15:29-08	2005-02-17	Enceladus
2005-02-17 02:15:29-08	2005-02-17	Icy satellite surface map
2005-02-17 02:15:29-08	2005-02-17	Limb topography
2005-02-17 02:15:29-08	2005-02-17	Limb topography
2005-02-17 03:00:29-08	2005-02-17	Enceladus closest approach observation
2005-02-17 03:00:29-08	2005-02-17	MAPS Campaign
2005-02-17 03:00:29-08	2005-02-17	Thermal stabilization period
2005-02-17 03:05:29-08	2005-02-17	Icy satellite exosphere
2005-02-17 03:05:29-08	2005-02-17	Enceladus dust observation
2005-02-17 03:05:29-08	2005-02-17	Enceladus dust observation
2005-02-17 04:00:29-08	2005-02-17	CIRS FP1 dark side map
2005-02-17 04:00:29-08	2005-02-17	Icy satellite longitudinal coverage
2005-02-17 04:00:29-08	2005-02-17	Enceladus
2005-02-17 05:25:29-08	2005-02-17	RSS gravity field determination
2005-02-17 08:30:29-08	2005-02-17	RADAR Scatterometry / Radiometry
2005-02-17 11:30:29-08	2005-02-17	Observe Enceladus' plumes
2005-02-17 11:30:29-08	2005-02-17	Icy satellite longitudinal coverage
2005-02-17 11:30:29-08	2005-02-17	Observe Enceladus' plumes

The Cassini team thought this would be a trivial look at a “dead icy ball,” so they pointed Cassini’s thermal and ultraviolet imaging systems towards Enceladus — as I can see from these results, which are actually a bit lacking now. I just can’t get over how neat it is to look at the actual mission plan information.

OK, focus Dee. I need to fix up this view so I can have some detail.

```

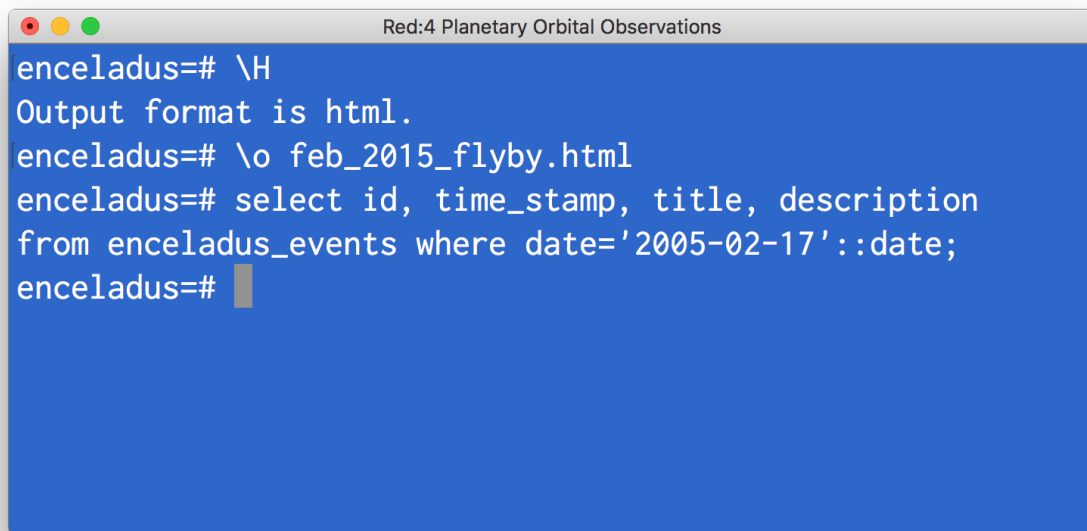
drop view if exists enceladus_events;
create view enceladus_events as
select
events.id,
events.title,
events.description,
events.time_stamp,
events.time_stamp::date as date,
event_types.description as event
from events
inner join event_types
on event_types.id = events.event_type_id
where target_id=40
order by time_stamp;

```

Love the tip from M. Sullivan about the **drop** statement. It makes it really easy to go back and tweak things. I could have altered this view, but that's kind of time-consuming; much easier to just drop and create it again. The query only runs when you **select** information from the view, so creating it takes no time at all.

I've added the **id**, **title** and **description** fields, so we know what's happening... but if you run this you'll see nothing but junk (first item on my wishlist if I get this job: a bigger monitor). If I had a super wide display, I could expand the results, so they're readable, but I have a better idea. I read about this last night, and I really want to try it out! I can use HTML to read the query results!

I can ask **psql** to redirect **STDOUT** to an HTML file with **\H**
and **\o**:



```
enceladus=# \H
Output format is html.
enceladus=# \o feb_2015_flyby.html
enceladus=# select id, time_stamp, title, description
from enceladus_events where date='2005-02-17'::date;
enceladus=#
```

If I were to run the query without specifying an output, then **psql** would just redirect to **STDOUT**, which is my terminal and I would see a splash of ugly angle brackets. The **\o** command specifies the output to use instead of **STDOUT**. I'm creating a file that will be popped into my current working directory.

Finally, I'm running the query, pulling down the **id**, **date**, **title**, and **description** as that's all I want to see.

Opening **feb_2015_flyby.html**:

	01:30:29-08		40 minute to complete. If not enough time, drop the 80 ms frame. point and stare at enceladus
14654	2005-02-17 01:30:29-08	Enceladus rider (FP1,FP3 high spectral resolution integration)	FP1, FP3 observation of Enceladus to search for current thermal activity and obtain high spectral resolution data. Sub-S/C=(1, 314), phase=25, 11<AD<17 mrad. CIRS at 0.5 cm-1 res (52 sec/frame), blink, shutter closed for first and last 5 minutes.
14655	2005-02-17 02:15:29-08	Enceladus	24x12i2-v1 160ms, 1 min/cycle) 0.73 Mbits/frame compressed, 55 frames total=40.2 mbits 50 minute to complete follow ISS Framing must follow ISS dwell times.
23148	2005-02-17 02:15:29-08	ICYMAP: High resolution icy satellite map	Ride along on high resolution map
30309	2005-02-17 02:15:29-08	Enceladus rider (FP1,FP3 coverage)	ISS rider. FP1, FP3 observation of Enceladus to search for current thermal activity, spatial coverage. (1, 318)<Sub-S/C<(7, 332), 26<phase<30, 18<AD<64 mrad. CIRS at 15.5 cm-1 res (4.75 sec/frame), blink, shutter closed for first and last 5 minutes. ***** Duration shortened 5 minutes, to 00:50 ***** 9/15/04
30308	2005-02-17 02:15:29-08	ENCELADUS Limb Topography	5 NAC GRN filters
14409	2005-02-17 03:00:29-08	Enceladus closest approach observations	RPWS observations in the immediate vicinity of Enceladus, including the characterization of the plasma wave spectrum and search for evidence of pickup ions. Pointing: prefer RPWS Langmuir Probe within 90 degrees of plasma ram.
14781	2005-02-17	MAPS 003EN	Part of 003EN MAPS Campaign. Observe interaction between Enceladus and Saturn's magnetosphere

So much more information! Look at 14654 — the plan regarding “FP3 observation”. That’s from CIRS, the Composite Infrared Spectrometer. FP3 is the mid-range focal plane which is used to detect volcanism.

If you keep scrolling you can see the other systems taking their turns having a look at Enceladus: the RSS (the Radio Science Subsystem), the ISS (Imaging Science Subsystem) which does the high-res scans, etc. I finally get to see exactly what the spacecraft was studying as it passed Enceladus! This is so cool!

This one is particularly interesting (id 41467):

FP1 dark side map of Enceladus. Phase= 156, Local time=0100, sub S/C=(2, 162), AD=23 mrad. Turn from waypoint (15 min?); 14 scan Nyquist FP1 map of disk at 15.5 cm-1 resolution (4.75 sec/frame) (25 min); return to waypoint? (14 min?)

Well, *that's* gibberish, but I looked up what was going on, and found out that they were trying to align Cassini and its imaging systems for an *occultation*, where Enceladus is positioned between Cassini and a [star somewhere](#). They were looking for the plumes, hoping distant starlight would light up the icy spray:

In February 2005, the spacecraft moved in such a way that the UVIS instrument could conduct a stellar occultation of Enceladus. During a stellar occultation, a star passes behind the moon or planet, and the light from that star illuminates the area close to the object's surface, potentially revealing the presence of an atmosphere or geysers spewing up from the ground. That occultation would illuminate part of Enceladus' equator.

Unfortunately, there was nothing to see. The ISS didn't capture anything compelling, and CIRS wasn't able to confirm any volcanism:

We got the data back from the February 2005 occultation, and there was just absolutely nothing to see," Candy Hansen, a project scientist on UVIS, told Space.com. "On the one hand, not surprising; but also, a bit disappointing.

Cassini's magnetometer (MAG), however, did capture something surprising:

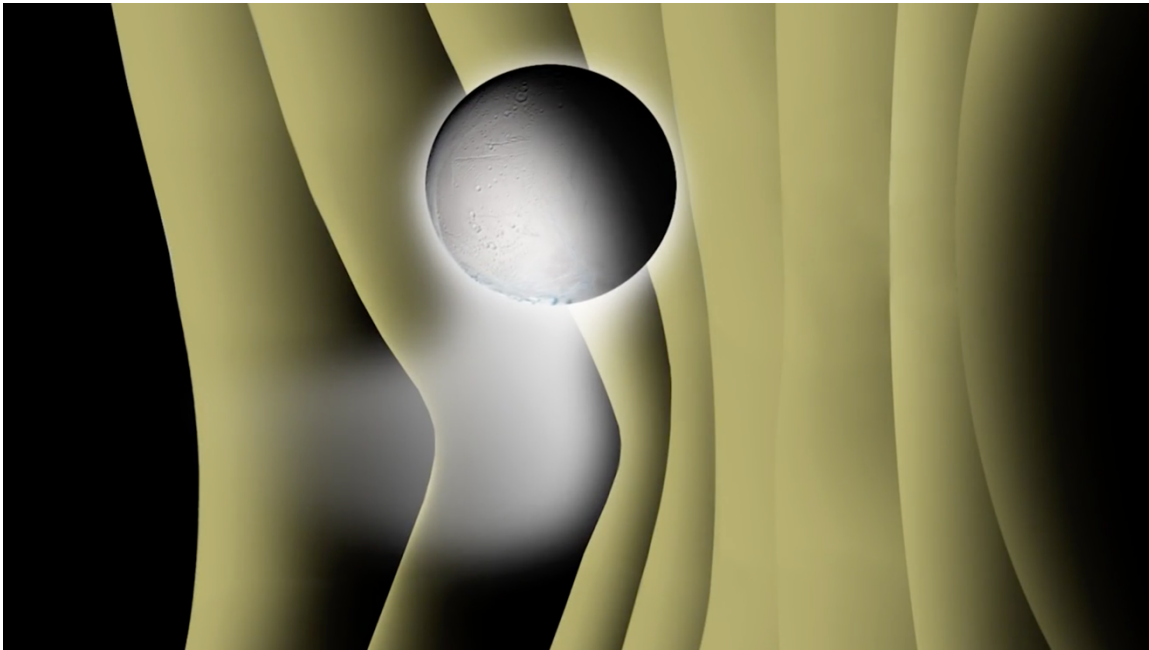
It was like the magnetic field lines of Saturn, which were moving towards Enceladus, were being draped upstream of Enceladus, almost like there was an obstacle [above the moon] ... but a larger obstacle than the body itself

Maybe someday I'll get to meet Michele Dougherty. I love her quotes... and that accent! I read [an interview with her from 2014](#), where she was asked what the best moments of her career were, and they were both from the Cassini mission. The first was when Cassini went into Saturnian orbit successfully, and the second? This:

I think the second most exciting thing is this: my team on Cassini was responsible for discovering an atmosphere on one of Saturn's moons. We saw some

observations on two of the flybys past the moon, and it looked like an atmosphere. We weren't sure, but we thought it was important that we try to go really close on the next fly-by. And so on the next flyby, instead of being a thousand kilometres away, we persuaded the Project team to take the spacecraft really close, at 170 km away from the moon. I watched the data coming back with my heart in my mouth because if we had messed up no one would have ever believed me again! So, that was another of the exciting bits. And then we started putting all of the data together.

She put her reputation on the line to discover more about Enceladus, what a star! If she hadn't done that, we might have missed out entirely on this little moon.



Influence of the Enceladus plumes on Saturn's magnetic field. Image credit: NASA/JPL

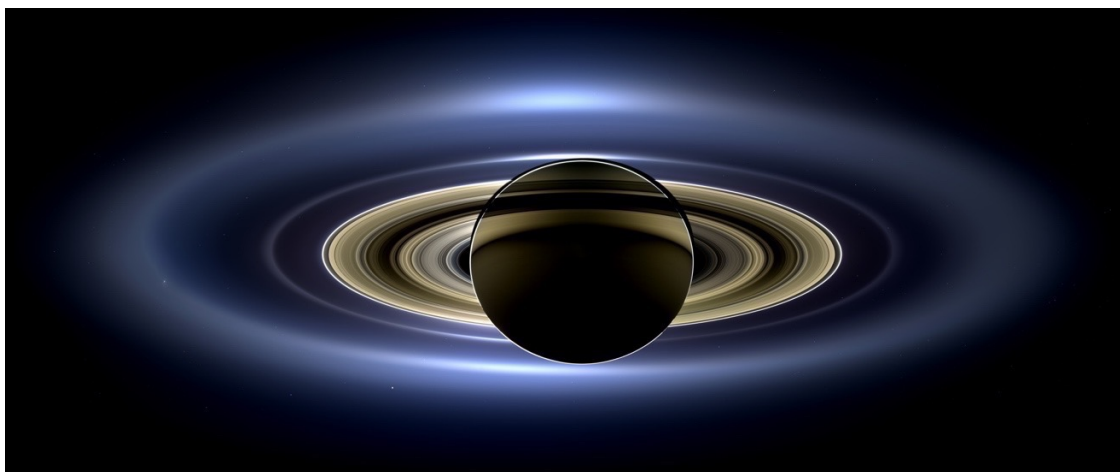
This is *precisely* what I was hoping to find: the very minute that Enceladus became something more than a “dead ice ball.” The moment that made Michele put her career on the line, and more than that: the moment that changed planetary science. It happened at

3am PDT, February 17, 2005. Enceladus was pushing against Saturn's magnetic field somehow.

I know I'm supposed to be looking for flyby times but come on, this is too much fun!

A Mystery Becomes More Mysterious

When the Voyager probes flew by Saturn, they were able to capture a few pictures of our small moon, Enceladus. One thing that scientists noted was how this little icy ball sat directly in the densest part of Saturn's more ethereal (yet most substantial) ring: The E-ring.



*Saturn's outer E-ring, aglow during a solar eclipse. Image brightened and enhanced.
Credit: NASA/JPL*

You can't see the E-ring from earth as it's a diffuse cloud of "stuff" which we now know is ice and water ice. Not believing in coincidence, NASA scientists wondered if the placement of Enceladus right in the thick of things had something to do with the E-ring itself. Which naturally led to the next question: was Enceladus somehow involved in the creation of Saturn's E-ring? How could that even happen?



Enceladus within The E-Ring. Photo credit: NASA/JPL

Some theories were thrown about:

- Geysers of ice and material were pulled from the moon by some force — perhaps a gravitational thing from Saturn, or maybe magnetic field action
- Ice was sublimating from the surface
- Volcanic activity on the Small moon was somehow propelling ice from the surface into space

None of these answers seemed plausible. The only thing that could provide a solution was to go back, fly really close to the small moon and turn some imaging systems toward it.

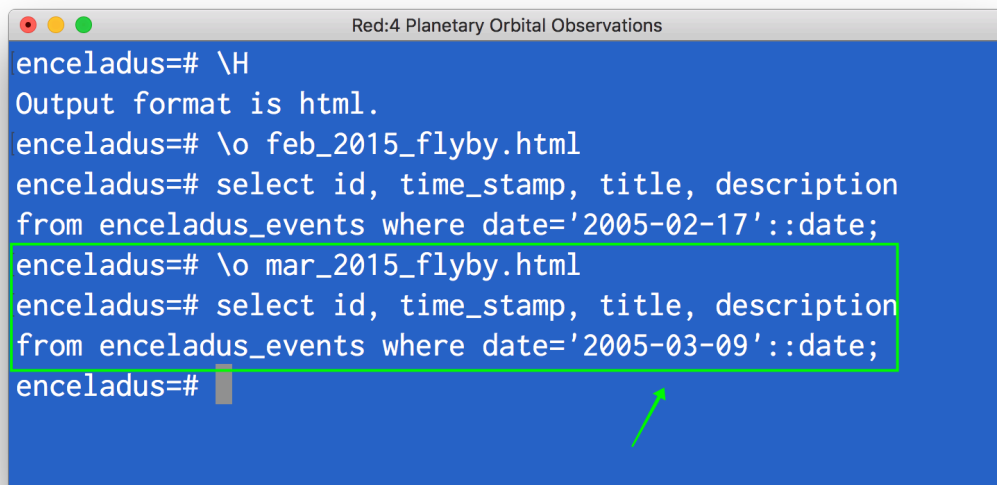
That flyby happened on February 17, 2005, and the results did not make sense. Every other instrument showed nothing, except for the magnetometer. The mystery deepened, and another flyby was scheduled weeks later on March 9, 2005.

GOLD TRACE IN THE MISSION PLAN

October 26, 2017, 1452

The team had taken another look at Enceladus with every instrument they had to verify what they saw. I wanted to see this information for myself.

I reset the output of my HTML file to **mar_2015_flyby.html** and then set the query date for March 9, 2015:



```
Red:4 Planetary Orbital Observations
enceladus=# \H
Output format is html.
enceladus=# \o feb_2015_flyby.html
enceladus=# select id, time_stamp, title, description
from enceladus_events where date='2005-02-17'::date;
enceladus=# \o mar_2015_flyby.html
enceladus=# select id, time_stamp, title, description
from enceladus_events where date='2005-03-09'::date;
enceladus=#
```

Here's what I saw:

	08:38:01-08		keep corotation ram in -X, +/-Z half-plane; B field in X, Z plane.
14410	2005-03-09 08:38:01-08	Enceladus closest approach observations	RPWS observations in the immediate vicinity of Enceladus, including the characterization of the plasma wave spectrum and search for evidence of pickup ions. Pointing: prefer RPWS Langmuir Probe within 90 degrees of plasma ram.
23417	2005-03-09 08:38:01-08	SOST	Measurements of magnetospheric/exospheric interaction and determination of surface composition.
14789	2005-03-09 08:39:00-08	MAPS 004EN Campaign	Part of 004EN MAPS Campaign. Observe interaction between Enceladus and Saturn's magnetosphere
23415	2005-03-09 08:53:01-08	SOST	Measurements of magnetospheric/exospheric interaction and determination of surface composition.
14668	2005-03-09 08:58:01-08	RSS_Ridealong	Limb-to-limb scan of Enceladus during RSS flyby, plus space calibration on either end.
222	2005-03-09 09:10:05-08	Enceladus Orbit Xing	Enceladus Orbit Xing\$ 251.38200\$ 251.40601\$ 4.033 to 3.854 Rs lat: -0.100 to -0.083 deg ring-alt: 425. to 337.km 002OF020 no ADJ: 0 was:....pointing=CDA to Kepler RAM_rate=4192,dataVolume=8.287
23419	2005-03-09 09:23:01-08	SOST	Measurements of magnetospheric/exospheric interaction and determination of surface composition.
18709	2005-03-09 10:38:01-08	ENCELADUS near C/A	Use WAC imaging only. Start with W08 filter set and add BGR.

The RPWS was the Radio and Plasma Wave Science instrument that, duh, was all about radio and plasma waves. It also studied dust particles, but that wasn't important enough to go in the name apparently. The team was very curious about this "atmosphere" that the MAG reported, and they needed more data, so they started poking around for bits of anything floating in space.

I need to make searching this corpus of 60,000 records a bit easier and a whole lot faster. I can't have M. Sullivan hounding me with his academic demands!

He did mention something about full-text indexing. I know a little about that. You basically tweak a body of text and index it, prioritizing useful terms and deprioritizing "noise." If you do it right, you can make finding things a lot easier for Postgres. This is how Google makes money.

I don't think I can do this correctly unless I know a bit more about what I'm trying to do.

From: R. Conery rob@redfour.io
Subject: RE: Miner 49er headin fer them hillz
To: Dee Yan yand@redfour.io
Date: October 26, 2017

Sleuthing data is the fun part of our job, isn't it? *As long as it's clean and correct.*

Every flyby tells a story about Enceladus, and since Enceladus is what SELFI is all about, we need to know, to a very full extent, what happened when, with respect to that curious moon. Facts, truth, a solid story backed by the data itself! Not from Wikipedia or articles on Space.com or Wired.

The truth, Dee, is in the data. We just need to find it.

As I'm sure you're starting to realize as you work with the mission plan data: if humans are allowed to touch data, they corrupt data. The reason is simple: *information is power*. Control the information, control *everything*. The urge to make a bit of data your own is too much to resist for most people.

I think you're on the right track sleuthing through the mission plan, but if you get stuck, there are also data to explore with the INMS (Ion and Neutral Mass Spectrometer) and CDA (Cosmic Dust Analyzer, the coolest name ever).

It's fun to dig into this stuff, but do your best to stay on target. It's easy to invent stories based on phantom information, and we have an entire Analytics Team that specializes in that thing, my dear Watson.

R

Ha! Sherlock Holmes. I get it. It takes practice and talent to see clearly things that other people take for granted, invent, or flat out refuse to look at. This quote, from *The Reigate Squires* is my favorite:

I make a point of never having any prejudices, and of following docilely where fact may lead me.

We see what we want to see, in everyone and everything. Shaking the bias and letting fact lead you along: that's the key. I need to be careful to avoid being pulled down a rabbit hole. I won't lie, though, this mission plan data concerns me some. Mixed casing, that leap year error. I'm wondering how much I can trust this resource.

Right. Where was I? Ah, yes: full-text indexing. I need to eat something — I'll get to that after dinner.

FULL-TEXT QUERIES

October 26, 2017, 1905

Happy news! Full-text indexing is really easy! I can merely tweak my view to add a full-text search index:

```

drop view if exists enceladus_events;
create view enceladus_events as
select
events.id,
events.title,
events.description,
events.time_stamp,
events.time_stamp::date as date,
event_types.description as event,
to_tsvector(events.description) as search
from events
inner join event_types
on event_types.id = events.event_type_id
where target_id=40
order by time_stamp;

```

That's so easy! I just need to use the **to_tsvector** function and I now have an index that I can search over. [There's more that I can do with this function](#), but this will work just fine for what I need to do next.

Let's test this out. I'll see what investigations were happening regarding thermal activity, so I should be able to use the word "thermal" with the search index:

```

-- find the thermal results
select id, date, title
from enceladus_events
where date
between '2005-02-01'::date
and '2005-02-28'::date
and search @@ to_tsquery('thermal');

```

The `@@` symbol is specific to search queries and means “show me the matches.” [There are other operators](#) you can use, but I’m not going to focus on that now.

I'm using a range expression with the where clause, specifying a range **between** February 1 **and** February 28 as I want to see all of the results for the month. I still think it's kind of cool that I can break data down by day just by forcing the **timestamp_{tz}** into the less-precise **date** type.

Running this query for the February flyby, I get back 8 results:

```

Red:4 Planetary Orbital Observations
-----+-----+-----+
3374 | 2005-02-16 | Enceladus rider (FP3 integration)
3380 | 2005-02-16 | RSS Enceladus Gravity Thermal Stabilization
3382 | 2005-02-16 | Enceladus rider (FP3 integration)
3396 | 2005-02-17 | Enceladus FP1 and FP3 maps, high spectral resoluti
3404 | 2005-02-17 | Enceladus rider (FP1,FP3 high spectral resolution integration)
3407 | 2005-02-17 | Enceladus rider (FP1,FP3 coverage)
3411 | 2005-02-17 | RSS Enceladus Gravity Thermal Stabilization
3429 | 2005-02-17 | Enceladus rider (FP3 0.5 cm-1 res. integration)
(8 rows)

enceladus=#

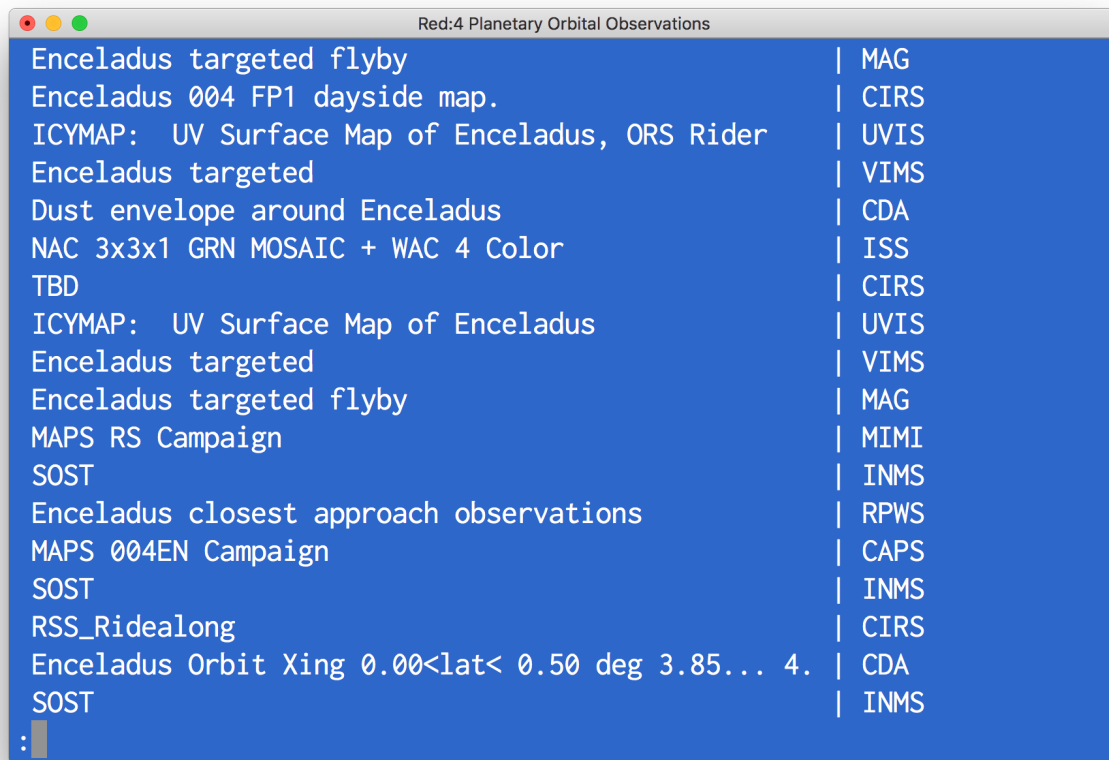
```

If I run it for the March flyby, I get 0. The Cassini team weren't interested in the thermal activity, apparently. What were they doing?

Each instrument aboard Cassini had a dedicated team back on earth. The mission plan details which team is, basically, “driving Cassini” at any given time.

```
select title, teams.description
from events
inner join teams on teams.id=team_id
where time_stamp::date='2005-03-09'
and target_id=40;
```

This tells me what they were doing with that second flyby:



Enceladus targeted flyby	MAG
Enceladus 004 FP1 dayside map.	CIRS
ICYMAP: UV Surface Map of Enceladus, ORS Rider	UVIS
Enceladus targeted	VIMS
Dust envelope around Enceladus	CDA
NAC 3x3x1 GRN MOSAIC + WAC 4 Color	ISS
TBD	CIRS
ICYMAP: UV Surface Map of Enceladus	UVIS
Enceladus targeted	VIMS
Enceladus targeted flyby	MAG
MAPS RS Campaign	MIMI
SOST	INMS
Enceladus closest approach observations	RPWS
MAPS 004EN Campaign	CAPS
SOST	INMS
RSS_Ridealong	CIRS
Enceladus Orbit Xing 0.00<lat< 0.50 deg 3.85... 4.	CDA
SOST	INMS
:	

Looks like all of them were active at some point. I wonder which one had the most time? Oooh! I can use an aggregate query!

```
select count(1) as activity, teams.description
from events
inner join teams on teams.id=team_id
where time_stamp::date='2005-03-09'
and target_id=40;
```

```
Red:4 Planetary Orbital Observations
enceladus=# select count(1) as activity, teams.description
enceladus=# from events
enceladus=# inner join teams on teams.id=team_id
enceladus=# where time_stamp::date='2005-03-09'
enceladus=# and target_id=40;
ERROR: column "teams.description" must appear in the GROUP BY clause or be used in an aggregate function
LINE 1: select count(1) as activity, teams.description
                                         ^
enceladus=#
```

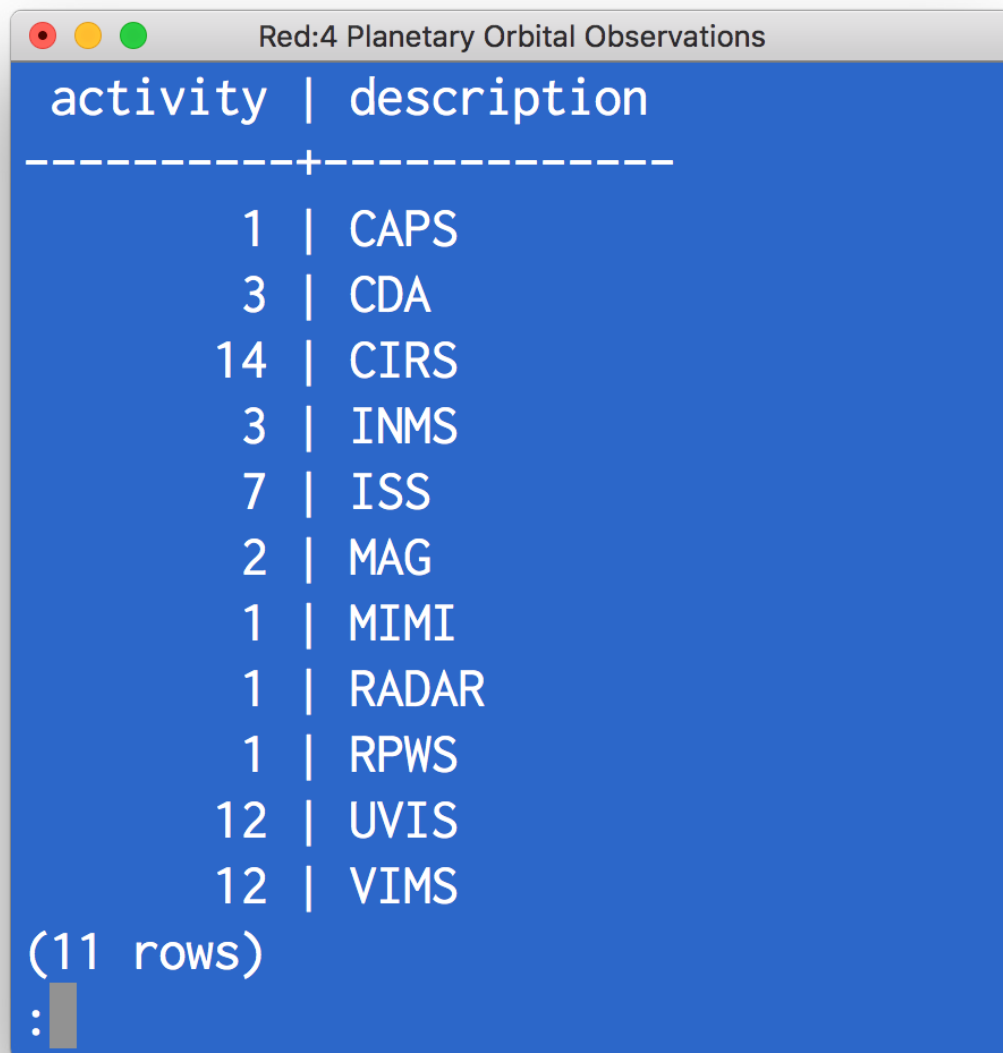
Crap. BRB Googling...

Right: aggregate queries with Postgres can be a chore. It's very focused on "doing SQL right" so it adheres to the SQL standard (didn't know they had one of those, but I guess it makes sense) as closely as possible.

I can fix this by telling Postgres how to roll this query up:

```
select count(1) as activity, teams.description
from events
inner join teams on teams.id=team_id
where time_stamp::date='2005-03-09'
and target_id=40
group by teams.description;
```

Wow, every single instrument:



activity	description
1	CAPS
3	CDA
14	CIRS
3	INMS
7	ISS
2	MAG
1	MIMI
1	RADAR
1	RPWS
12	UVIS
12	VIMS

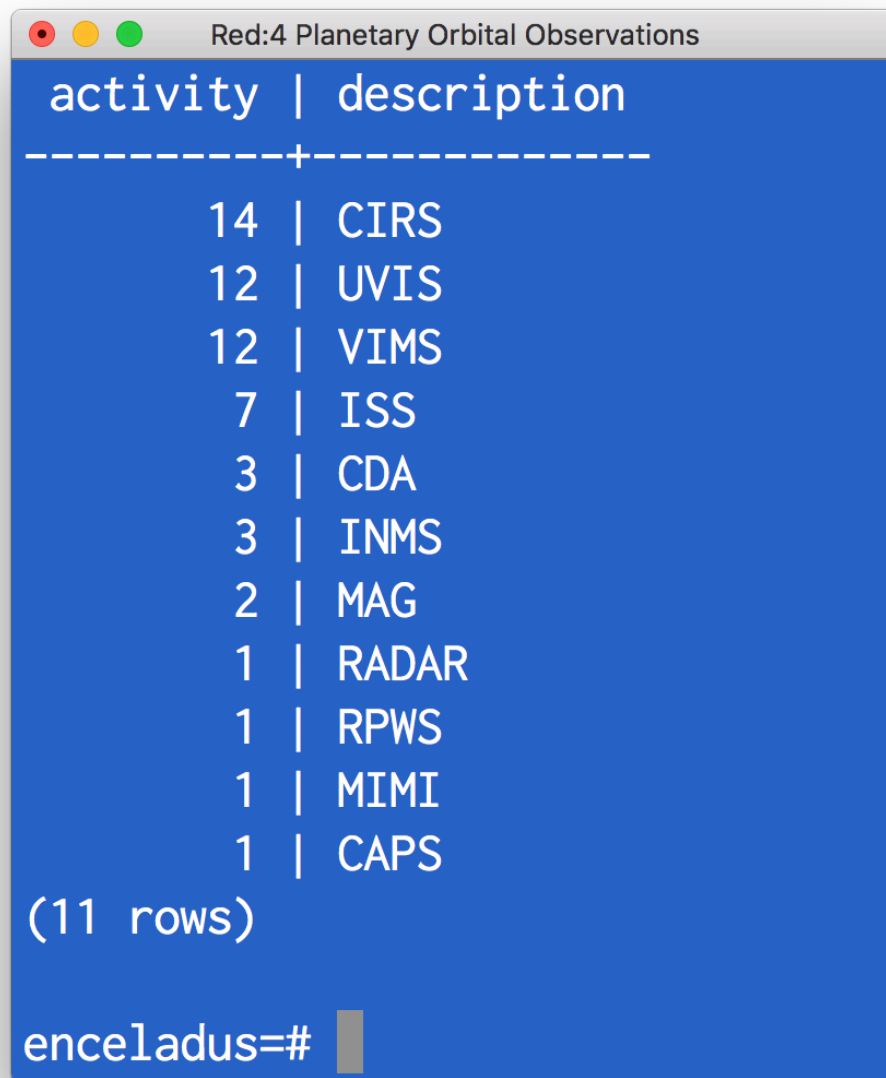
(11 rows)

:

Think I'd like this better if it were sorted, though:

```
select count(1) as activity, teams.description
from events
inner join teams on teams.id=team_id
where time_stamp::date='2005-03-09'
and target_id=40
group by teams.description
order by activity desc;
```

I like how you can alias a column and then order by it, neat.



activity	description
14	CIRS
12	UVIS
12	VIMS
7	ISS
3	CDA
3	INMS
2	MAG
1	RADAR
1	RPWS
1	MIMI
1	CAPS

(11 rows)

enceladus=#

There it is. During the second flyby, on March 9, 2005, the Composite Infrared Scanner (CIRS) was the most active. So much for thermal, huh? The team was looking for a source of heat, something that would cause the eruption of ice and water into space.

The UltraViolet Imaging Spectrograph Subsystem is next. From the instrument's [home page](#):

The Ultraviolet Imaging Spectrograph Subsystem (UVIS) is a set of telescopes used to measure ultraviolet light from the Saturn system's atmospheres, rings, and surfaces. The UVIS will also observe the fluctuations of starlight and sunlight as the sun and stars move behind the rings and the atmospheres of Titan and Saturn, and it will determine the atmospheric concentrations of hydrogen and deuterium.

They were taking pictures of Enceladus too, probably to see some features on the surface.

The next instrument, VIMS, is the Visible and Infrared Mass Spectrometer:

Scientists used VIMS to determine the content and temperatures of atmospheres, rings, and surfaces in the Saturn system. The instrument analyzed light, but scientists also created images from its data, similar to a visible-light camera.

That would suggest they were trying to confirm the presence of an atmosphere, something indicated by the magnetometer readings observed during the first flyby.

The only other instrument that was reasonably active on that day was the ISS, Cassini's Imaging System. All the others were relatively inactive.

[They found something:](#)

People were saying 'it's gotta be jets it's gotta be jets,' and the imaging team was saying 'no no no we don't want to be saying that until we're sure...'

--Linda Spilker, Cassini Project Scientist

The atmospheric signature we were seeing was focused at the south pole... it was almost like there was a cometary plume of water vapor coming off the south pole...

--Michele Dougherty

What they saw with this second flyby was enough for a [press release](#), announcing that a tiny moon the size of Arizona had an atmosphere:

The Cassini spacecraft's two close flybys of Saturn's icy moon Enceladus have revealed that the moon has a significant atmosphere. Scientists, using Cassini's magnetometer instrument for their studies, say the source may be volcanism, geysers, or gases escaping from the surface or the interior... The observations showed a bending of the magnetic field, with the magnetospheric plasma being slowed and deflected by the moon. In addition, magnetic field oscillations were observed. These are caused when electrically charged (or ionized) molecules interact with the magnetic field by spiraling around the field line. This interaction creates characteristic oscillations in the magnetic field at frequencies that can be used to identify the molecule. The observations from the Enceladus flybys are believed to be due to ionized water vapor.

A Better Full-Text Search

October 27th, 2017 0042

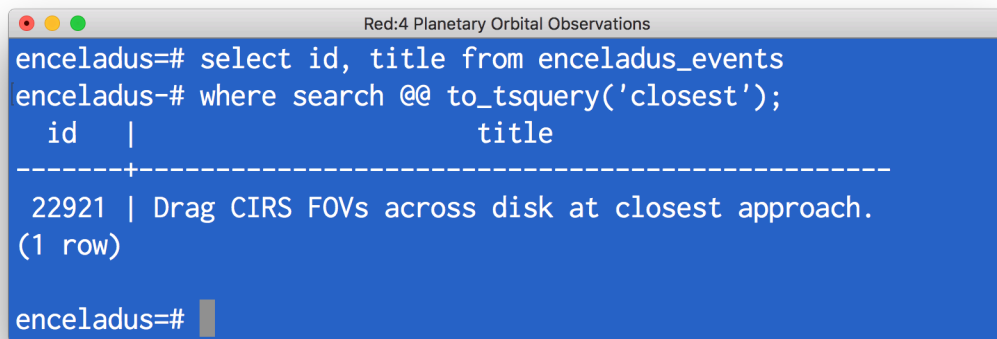
I need to get to bed. This data is so fascinating, though! How can you not get pulled in when you get to see what Cassini was doing so clearly!

I still need to find better flyby dates, so I'll use my **enceladus_events** view and see if I can account for all 23 (including the first flyby in February).

So far, every flyby has the term "closest," so I should be able to use that with my full-text index:

```
select id, title from enceladus_events
where search @@ to_tsquery('closest');
```

Nope.

A terminal window titled "Red:4 Planetary Orbital Observations" with a blue background. It shows the execution of a SQL query and its results.

```
enceladus=# select id, title from enceladus_events
enceladus=# where search @@ to_tsquery('closest');
 id | title
-----+-----
22921 | Drag CIRS FOVs across disk at closest approach.
(1 row)

enceladus=#
```

Hmm. Why would that be? Thinking...

Right. It's getting late. The only thing I've indexed is the **description** field. That doesn't have the term "closest" in it, only **title** does.

I wonder if I can build an index using two or maybe three columns?

```
drop view if exists enceladus_events;
create view enceladus_events as
select
events.id,
events.title,
events.description,
events.time_stamp,
events.time_stamp::date as date,
event_types.description as event,
to_tsvector(
    concat(events.description, '', events.title)
) as search
from events
inner join event_types
on event_types.id = events.event_type_id
where target_id=40
order by time_stamp;
```

Hey! That worked! I used the concat function (which I read about online while Googling all of this) to concatenate **title** and **description**. Look at the result!

```
Red:4 Planetary Orbital Observations
35669 | Enceladus closest approach observations
36743 | Enceladus closest approach observations
37700 | Enceladus closest approach observations
37858 | Enceladus closest approach observations
40090 | Enceladus closest approach observations
40293 | Enceladus closest approach observations
40465 | Enceladus closest approach observations
41965 | Enceladus closest approach observations
42168 | Enceladus closest approach observations
42383 | Enceladus closest approach observations
53888 | Enceladus closest approach observations
54039 | Enceladus closest approach observations
54590 | Enceladus closest approach observations
(25 rows)
(END)
```

25 results! We have 23 total flybys so somehow, I'm getting 2 noise rows in there... I think it's because I'm also using the **description** field, which seems to be doubling things up.

I need to sleep and think about this more.

From: M. Sullivan sullz@redfour.io
Subject: Views, M. Yan, DO YOU SPEAK IT
To: Dee Yan yand@redfour.io
Date: October 27, 2017

Materialized. View. With an index on that search field. There is smoke coming out of Postgres right now.

Best, S

USING A MATERIALIZED VIEW

October 27, 2017, 0715

Oops; forgot to read up on materialized views. I'll do that when I get in today. Right now, I need some coffee.

I decided to walk down Chestnut to the Roasting Company and treat myself to a cappuccino and scone. I'll eat a proper American breakfast later.

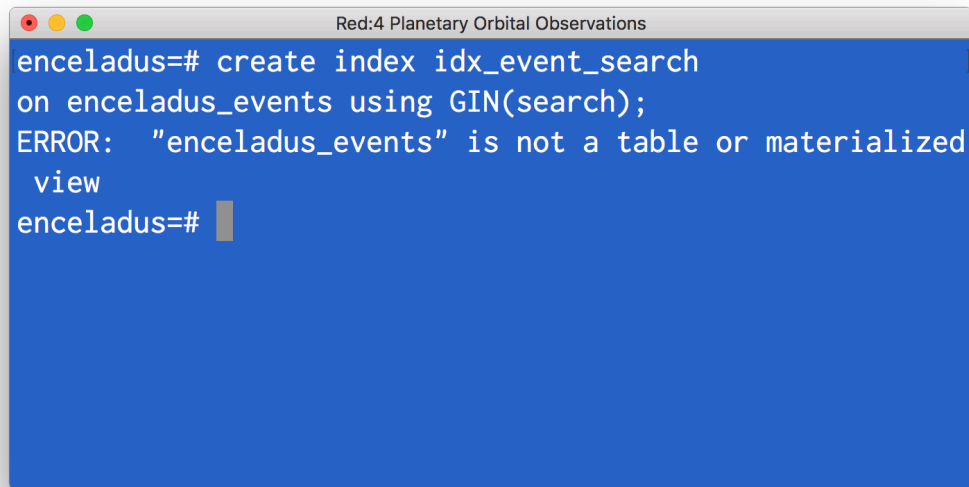
It's getting colder, and the leaves are turning. September and October are always the best times to be here in San Francisco — clear, blue, bright skies and a bit of a chilly breeze. My cheeks are zingy warm now.

OK, let's get to it. I haven't looked it up yet but indexing a view should be as simple as:

```
create index idx_event_search
on enceladus_events using GIN(search);
```

I do know about indexes. They work just like their analog that you would find in the back of a book. You find the term you're looking for by using an alphabetical lookup and then head to a given page. Much faster than scanning every single page for a term, no?

With this little incantation, Postgres should create an index for my full-text search field using a GIN index:



```
enceladus=# create index idx_event_search
on enceladus_events using GIN(search);
ERROR:  "enceladus_events" is not a table or materialized
view
enceladus=#
```

Or not. Does GIN not work on **tsvector** or something? Be right back...

Duh! I'm trying to index a view! That doesn't make sense, does it? Good morning Dee, sleep well?

A view is just a projection of data from a source. Nothing is stored anywhere, and every time I query a view I'm actually querying the underlying tables **through** it. The only way I can index a view is if it exists on disk, which is what the "materialized" bit means. They're great for speeding things up, especially if you need to throw an index in! Which I do.

Dropping and recreating the view is easy. I just need to remember to use the term.

```
drop view if exists enceladus_events;
create materialized view enceladus_events as
select
events.id,
events.title,
events.description,
event_types.description as event,
events.time_stamp::date as date,
events.time_stamp,
to_tsvector(concat(
events.description, ' ',
events.title)
) as search
from events
inner join event_types
on event_types.id = events.event_type_id
where target_id=40
order by time_stamp;
```

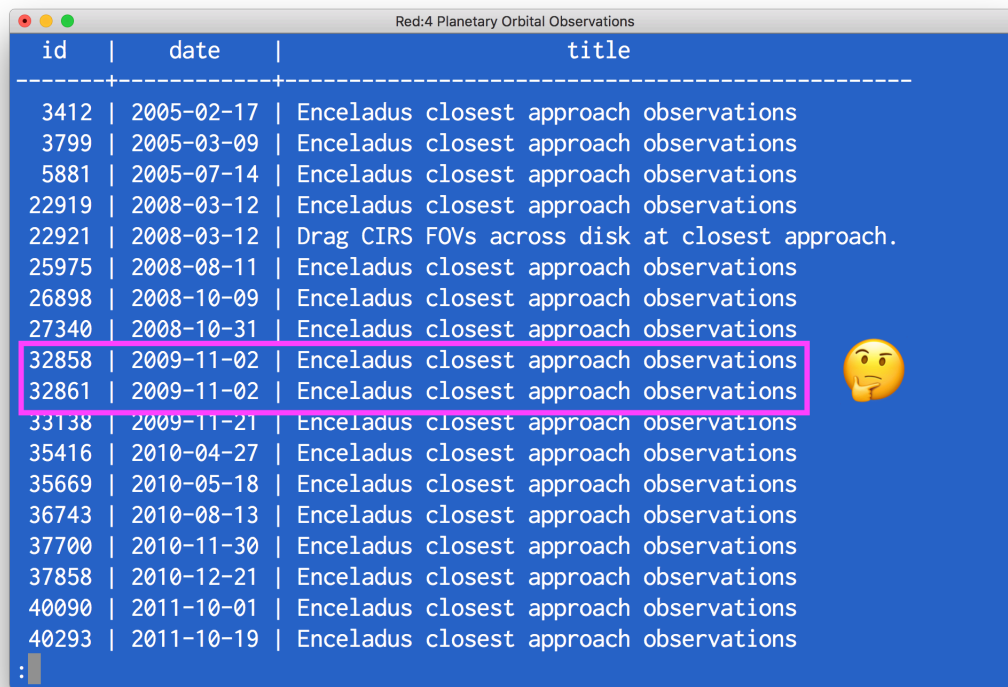
And now for the index:

```
create index idx_event_search
on enceladus_events using GIN(search);
```

Does it work?

```
select id, date, title
from enceladus_events
where search @@ to_tsquery('closest');
```

Boom!



id	date	title
3412	2005-02-17	Enceladus closest approach observations
3799	2005-03-09	Enceladus closest approach observations
5881	2005-07-14	Enceladus closest approach observations
22919	2008-03-12	Enceladus closest approach observations
22921	2008-03-12	Drag CIRS FOVs across disk at closest approach.
25975	2008-08-11	Enceladus closest approach observations
26898	2008-10-09	Enceladus closest approach observations
27340	2008-10-31	Enceladus closest approach observations
32858	2009-11-02	Enceladus closest approach observations
32861	2009-11-02	Enceladus closest approach observations
33138	2009-11-21	Enceladus closest approach observations
35416	2010-04-27	Enceladus closest approach observations
35669	2010-05-18	Enceladus closest approach observations
36743	2010-08-13	Enceladus closest approach observations
37700	2010-11-30	Enceladus closest approach observations
37858	2010-12-21	Enceladus closest approach observations
40090	2011-10-01	Enceladus closest approach observations
40293	2011-10-19	Enceladus closest approach observations

That was really fast! But wait a minute — what the heck with two “closest approach” events on November 2?

Investigation! What happened on November 2, 2009?

```
select time_stamp, title
from events
where time_stamp::date='2009-11-02'
order by time_stamp;
```

Found it! Here's the result:

Red:4 Planetary Orbital Observations		
2009-11-02	06:42:58-08	Enceladus Flyby C/A imaging
2009-11-02	06:42:58-08	Enceladus encounter
2009-11-02	06:44:00-08	Data rate
2009-11-02	07:00:00-08	ENC INMS-CDA 2009-306T0:42 MMH
2009-11-02	07:11:58-08	MAPS RS Campaign
2009-11-02	07:11:58-08	Enceladus closest approach observations
2009-11-02	07:36:58-08	CIRS_120EN_ENCEL7001_INMS;
2009-11-02	07:39:58-08	ICYMAP: UV Surface Map
2009-11-02	07:56:58-08	Enceladus closest approach observations
2009-11-02	08:03:58-08	ICYMAP: UV Surface Map
2009-11-02	08:03:58-08	Enceladus encounter
2009-11-02	08:03:58-08	Enceladus
2009-11-02	08:03:58-08	CIRS_120EN_FP3SPMAP001_PRIME;
2009-11-02	08:03:58-08	Enceladus
2009-11-02	08:11:58-08	RPWS Ring Plane Crossing
2009-11-02	08:11:58-08	MIMI Satellite & Ring Interactions
2009-11-02	08:13:58-08	ICYMAP: UV of plume occulting Saturn
2009-11-02	08:13:58-08	Enceladus
2009-11-02	08:13:58-08	CIRS_120EN_ENUVIS001_UVIS
2009-11-02	08:34:00-08	Enceladus targeted flyby

The first problem I see is that I didn't account for UTC timestamps. I can see that because every date has a **-8** at the end, which is our local time zone here in Northern California and Saturn doesn't care about Northern California.

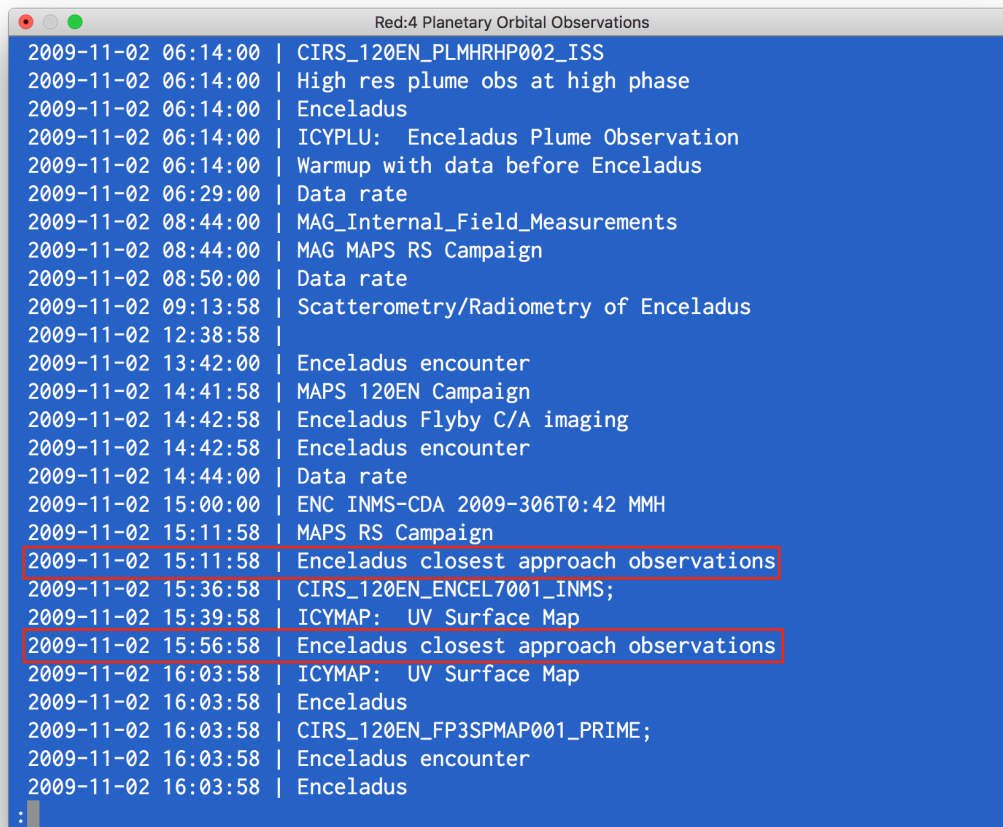
I don't know if adjusting this to UTC will solve the problem, but let's give it a shot:

```

select (time_stamp at time zone 'UTC'),
title
from events
where (time_stamp at time zone 'UTC')::date='2009-11-02'
order by time_stamp;

```

Nope. Same results:



```

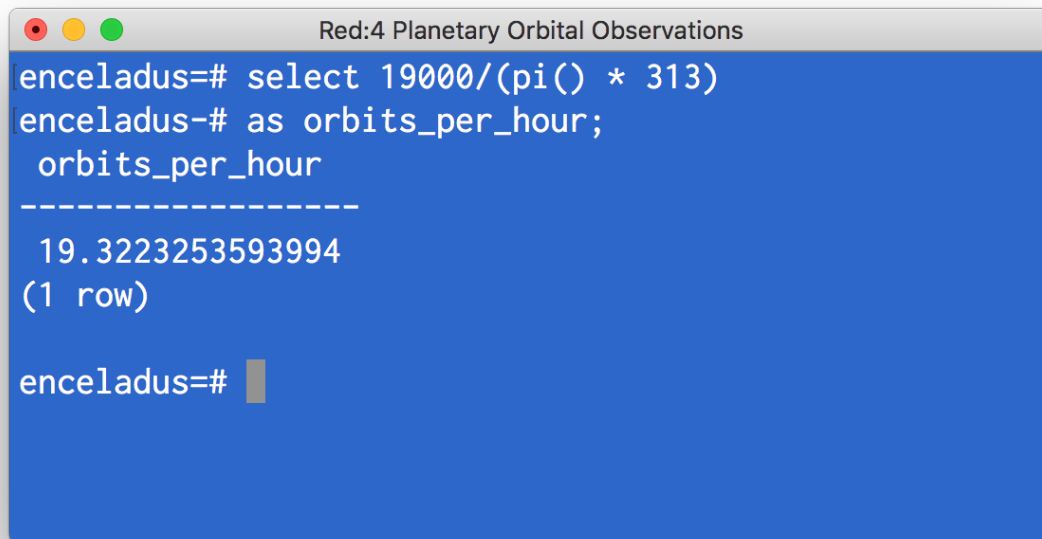
Red:4 Planetary Orbital Observations
2009-11-02 06:14:00 | CIRS_120EN_PLMRHP002_ISS
2009-11-02 06:14:00 | High res plume obs at high phase
2009-11-02 06:14:00 | Enceladus
2009-11-02 06:14:00 | ICYPLU: Enceladus Plume Observation
2009-11-02 06:14:00 | Warmup with data before Enceladus
2009-11-02 06:29:00 | Data rate
2009-11-02 08:44:00 | MAG_Internal_Field_Measurements
2009-11-02 08:44:00 | MAG MAPS RS Campaign
2009-11-02 08:50:00 | Data rate
2009-11-02 09:13:58 | Scatterometry/Radiometry of Enceladus
2009-11-02 12:38:58 |
2009-11-02 13:42:00 | Enceladus encounter
2009-11-02 14:41:58 | MAPS 120EN Campaign
2009-11-02 14:42:58 | Enceladus Flyby C/A imaging
2009-11-02 14:42:58 | Enceladus encounter
2009-11-02 14:44:00 | Data rate
2009-11-02 15:00:00 | ENC INMS-CDA 2009-306T0:42 MMH
2009-11-02 15:11:58 | MAPS RS Campaign
2009-11-02 15:11:58 | Enceladus closest approach observations
2009-11-02 15:36:58 | CIRS_120EN_ENCEL7001_INMS;
2009-11-02 15:39:58 | ICYMAP: UV Surface Map
2009-11-02 15:56:58 | Enceladus closest approach observations
2009-11-02 16:03:58 | ICYMAP: UV Surface Map
2009-11-02 16:03:58 | Enceladus
2009-11-02 16:03:58 | CIRS_120EN_FP3SPMAP001_PRIME;
2009-11-02 16:03:58 | Enceladus encounter
2009-11-02 16:03:58 | Enceladus
:

```

The only explanations remaining are:

- There were two flybys, 45 minutes apart
- A data entry mistake

Enceladus is only 313 miles in diameter. There's no way that Cassini, traveling at 19,000 miles per hour is going to go into orbit around that. Can you imagine? That would be like:



```
Red:4 Planetary Orbital Observations
enceladus=# select 19000/(pi() * 313)
enceladus=# as orbits_per_hour;
orbits_per_hour
-----
19.3223253593994
(1 row)

enceladus=#
```

Zip zip zip! That would look so funny if you were standing on the surface of Enceladus...

It's also improbable that Cassini was able to get a gravity assist from Saturn or Titan, returning it within 24 hours. If it were possible, I would see other observations in there — NASA wouldn't waste an opportunity to have a look at the rings.

Data entry problem then? Or perhaps “closest approach” means something entirely different.

Either way: if I want precision for these flyby dates I'll need to use a different dataset.
Wow, what the heck happened here?

Whatever, it doesn't matter. I need to find some different data, and that will be my focus for tomorrow — I gotta go catch the bus.

SNIFF THE SKY

October 28, 2017, 1745

This time I *nailed* it, and I'm terrified that I might be too late. I want this job, but I'm moving too slowly, and they're not liking that I'm staying so late, spending most of my waking hours on this. "We don't want you to burn out, Dee" which is nice to hear, but I really want this job!

Look at what I was able to pull together from the INMS data!

Red:4 Planetary Orbital Observations		
nadir		altitude
-----+-----		
2005-02-17	11:30:12.119	1272.075
2005-03-09	17:08:03.4725	500.370
2005-07-15	02:55:22.33	168.012
2008-03-13	02:06:11.509	50.292
2008-08-12	04:06:18.574	53.353
2008-10-10	02:06:39.724	28.576
2008-11-01	00:14:51.429	173.044
2009-11-02	15:41:57.707	98.901
2009-11-21	10:09:56.371	1596.561
2010-04-28	07:00:01.088	3771.195
2010-05-18	13:04:40.301	437.292
2010-08-14	05:30:51.975	2555.180
2010-11-30	19:53:59.049	45.699
2010-12-21	09:08:27.146	48.324
2011-10-01	20:52:25.698	98.898
2011-10-19	16:22:11.2245	1230.674
2011-11-06	12:58:53.4805	496.603
2012-03-28	01:30:08.975	74.165
2012-04-14	21:01:37.811	74.100
2012-05-02	16:31:28.949	73.134
2015-10-14	17:41:28.9765	1844.230
2015-10-28	22:22:41.55	49.010
2015-12-20	01:49:16.1135	5000.200
(23 rows)		
enceladus=# █		

Those are exact timestamps for the precise moment that Cassini was flying lowest over Enceladus! I can't believe that I actually nailed it like this!

This was the most fun I've had with a database. I need to document every single step I took and how I got these results, which took me two days to pull together. I've been so focused and busy that I've forgotten to write (sorry journal). The work was so fun, though!

I learned all about one of Cassini's most delicate instruments: The Ion Neutral Mass Spectrometer — better known as simply the INMS. It can basically “sniff” space and tell you what chemicals are floating around out there. To do this accurately, it needs to record positional data, which is what I mined to get the results above. I had to cross-reference the data from the INMS with the mission plan, and I found all kinds of weirdness.

I also spent hours (at home) listening to interviews with the Cassini team, reading white papers about the inner workings of the INMS sensors and finally JPL publications on their blog. This, this right here is everything I ever wanted in a job, although I have to admit I never imagined myself as a DBA. Which evidently, I'm not. Not yet, at least, according to our CEO Sara.

When I got to work 2 days ago, right after figuring out the materialized query stuff and feeling gloriously motivated, this little sweet peach of an email was waiting for me.

From: Sara C. boss@redfour.io
Subject: A quick hello
To: Dee Yan yand@redfour.io
Date: October 27, 2017

Hello Dee, my name is Sara, and I'm the CEO of Red:4. I'm here in Los Angeles with our CTO, Rob, who has told me about the work you're doing to help us win this contract.

I'm sure Rob has filled you in on the importance of this contract to Red:4, but I wanted to lend my voice to that message. We're aiming to pivot our current focus from infrastructure as a service to machine learning and the SELF project is the fulcrum around which we intend to accomplish that.

I understand that Rob has promised you the role of DBA, which came as a bit of a surprise to me given that we haven't yet met. I'm sure you'll do admirably if you do indeed take on the role. We like self-starters! There is something we need to discuss first, however.

I've looked at your resume, and your work is indeed impressive, but I would like to underscore that what you're doing now is more on the analytical side. The director position Rob caroted is more of a traditional DBA role. Sort of a systems focus with an eye on maintaining our server, running optimizations, reliability, scalability and so on. In other words: not caring so much about what the data is as opposed to how well it lives. The importance of this role can't be understated, but it might not be what you're thinking it is. And it requires a level of experience and familiarity with database systems that I'm not sure you have. But I'm more than willing to be proven wrong!

Finally, Rob has mentioned that you're putting in over 10 hours a day on this. Sometimes even 12. I want to emphatically discourage this. We're not that kind of workplace. You will burn yourself out, and then we'll lose you altogether. I really do appreciate your desire to do well by us, but it can't come at the expense of your personal life. I hope that's clear to you, and I hope even more that you enjoy the weekend *outside of Red:4*.

Looking forward to meeting you soon,

Sara.

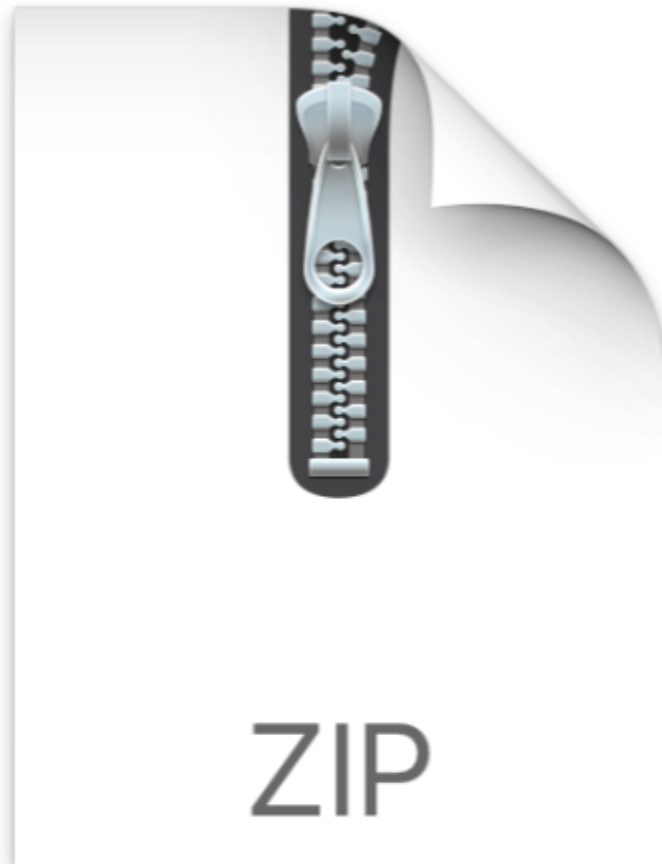
HELLO INMS

Carrotted? Is that a word?

Isn't this tactic called "push me, pull you"? The more work you're given, the harder management makes it on you to complete it?

Whatever, she can't stop me from reading at night. I guess I do understand about the burnout part, I have been a little edgy lately.

Who cares. All of that's boring. Let's get to the good stuff: the INMS data. I downloaded the [INMS archive](#) from our S3 bucket and wow is it big:



CO-S-INMS-3-L1A-U-V1.0.zip

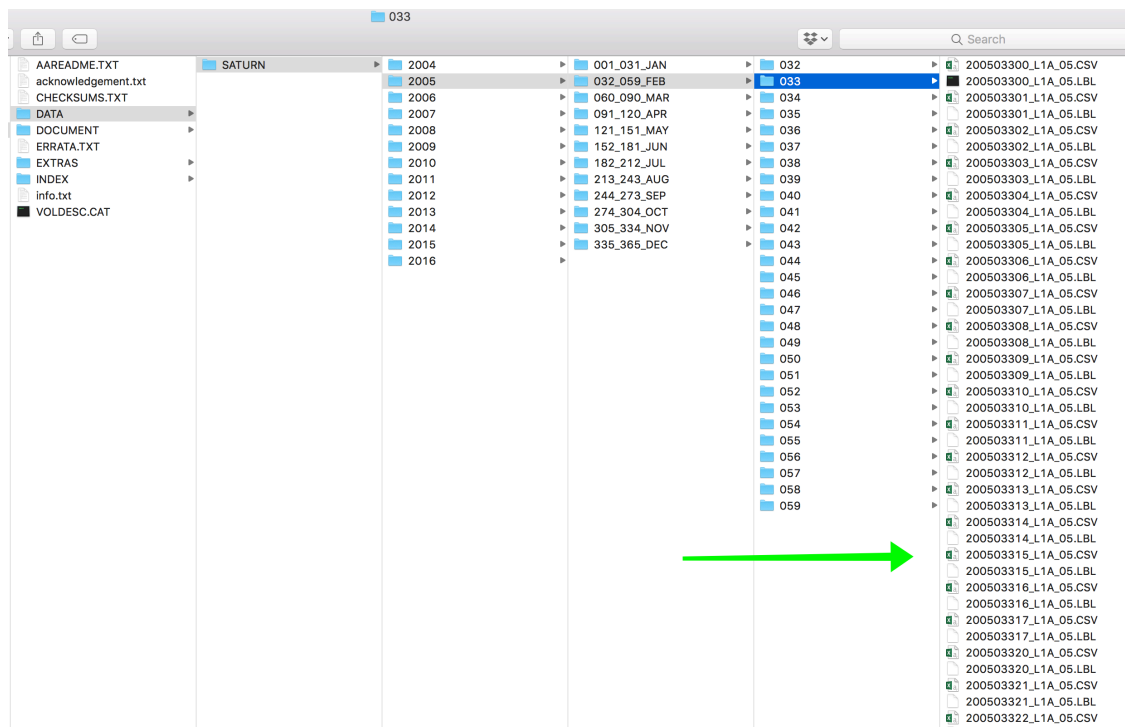
ZIP archive - 35.98 GB

Created Friday, October 27, 2017 at 5:19 PM

Modified Friday, October 27, 2017 at 8:07 PM

Last opened --

It contains directories for each year of the mission, each year directory includes the months. Each month contains subfolders for each day, and each day contains a mess of CSV files.



Each one of these CSV files is between 200K and 10Mb, and contains a load of INMS readings:

	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM
1	pr_ch os_lens2	os_lens1	os_lens4	os_lens3	qp_lens2	qp_lens1	qp_lens4	qp_lens3	qp_bias	ion_defl2	ion_defl1	ion_defl4
2	V	V	V	V	V	V	V	V	V	V	V	V
3	pr_ch os_lens2 (V)	os_lens1 (V)	os_lens4 (V)	os_lens3 (V)	qp_lens2 (V)	qp_lens1 (V)	qp_lens4 (V)	qp_lens3 (V)	qp_bias (V)	ion_defl2 (V)	ion_defl1 (V)	ion_defl4 (V)
4	1	0.04	0.02	-29.86	-12.83	-4.496	-4.496	-55.566	-55.517	-0.153	0.01	0
5	2	0.04	0.02	-29.86	-12.83	-3.281	-3.285	-56.744	-56.695	1.112	0.01	0
6	3	0.04	0.02	-29.86	-12.83	-2.065	-2.099	-57.947	-57.897	2.501	0.01	0
7	4	0.04	0.02	-29.86	-12.83	-0.9	-0.888	-59.124	-59.113	3.765	0.01	0
8	5	0.04	0.02	-29.86	-12.83	0.299	0.319	-60.327	-60.34	5.154	0.01	0
9	6	0.04	0.02	-29.86	-12.83	1.513	1.504	-61.542	-61.543	6.418	0.01	0
10	7	0.04	0.02	-29.86	-12.83	2.715	2.714	-62.77	-62.721	7.807	0.01	0
11	8	0.04	0.02	-29.86	-12.83	3.893	3.899	-63.972	-63.924	9.072	0.01	0
12	12	0.04	0.02	-29.86	-12.83	8.713	8.74	-68.733	-68.747	14.482	0.01	0
13	13	0.04	0.02	-29.86	-12.83	9.915	9.9	-69.961	-69.963	15.746	0.01	0
14	14	0.04	0.02	-29.86	-12.83	11.142	11.136	-71.176	-71.128	17.135	0.01	0
15	15	0.04	0.02	-29.86	-12.83	12.295	12.296	-72.391	-72.343	18.399	0.01	0
16	16	0.04	0.02	-29.86	-12.83	13.521	13.531	-73.556	-73.559	19.788	0.01	0
17	17	0.04	0.02	-29.86	-12.83	14.736	14.754	-74.771	-74.774	21.053	0.01	0
18	18	0.04	0.02	-29.86	-12.83	15.95	15.926	-75.987	-75.94	22.441	0.01	0
19	19	0.04	0.02	-29.86	-12.83	17.115	17.149	-77.202	-77.155	23.706	0.01	0

Unzipped, the entire set of CSVs is 300G. Hundreds of millions of rows of data! I don't need all of it, however, so ... lucky me? I only need the data for the flyby days and, thankfully, the directory organization helps me with this.

Each year has subdirectories for each month, and each month has subdirectories that are named by day of the year. This is how NASA generally thinks of dates, something like "2005-048" instead of February 17, 2005.

All I have to do, therefore, is to figure out what the year/dayofyear is for each of the flybys, and I'm in business.

I can use my little view for that:


```
-- the year/day of year for flybys
select date_part('year',date),
to_char(time_stamp, 'DDD')
from enceladus_events
where event like '%closest%';
```

I can never remember the date format patterns. I should remember DDD however, it's my name three times.

The results tell me which directories to grab:

Red:4 Planetary Orbital Observations	
date_part	to_char
2005	048
2005	068
2005	195
2008	072
2008	224
2008	283
2008	305
2009	306
2009	306
2009	325
2010	117
2010	138
2010	225
2010	334
2010	355
2011	274
2011	292
2011	310
2012	087
2012	105
2012	123
2015	287
2015	301
2015	353

(24 rows)

:

Perfect. Now I just need to go into each year, find the month subdirectory (which also has the day of year in the name) and then cherry pick the CSV directories I want.

Time to read up on the INMS.

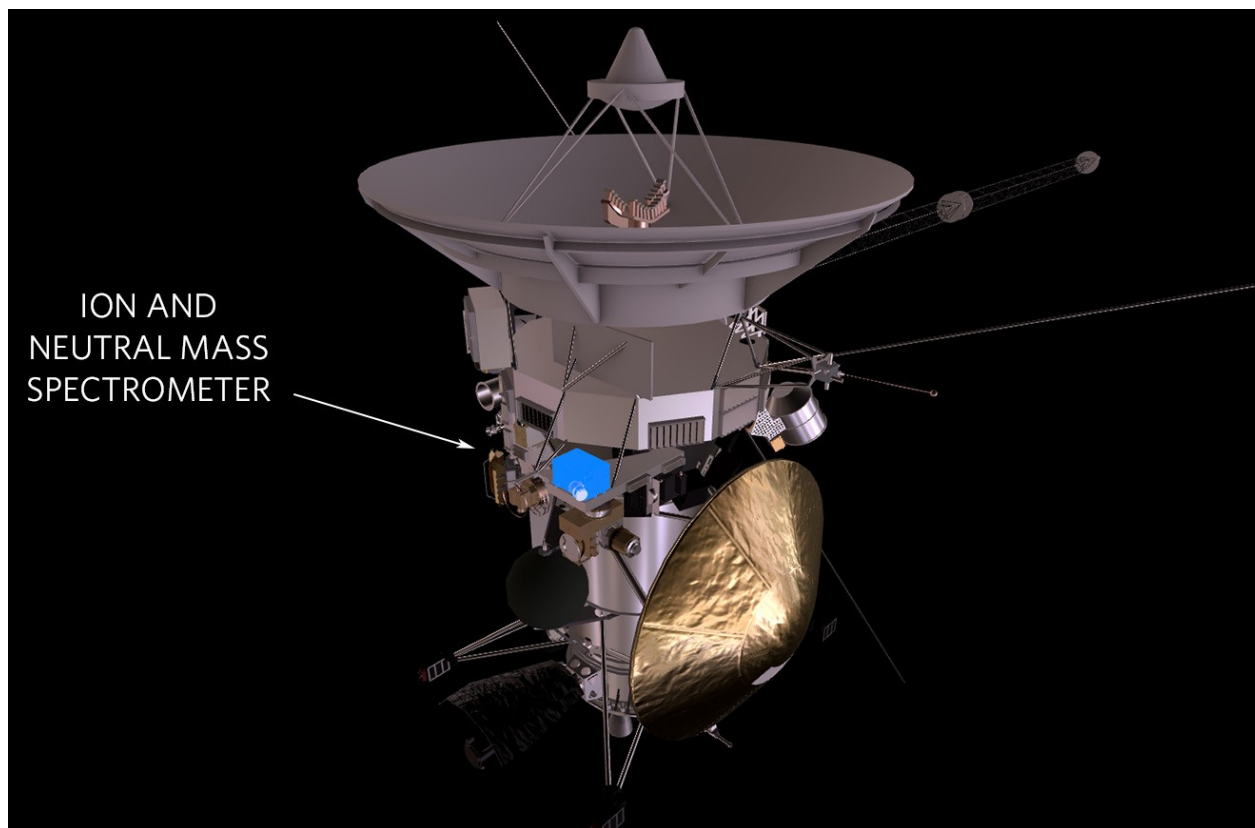
BASIC FUNCTION OF THE INMS

Rob Conery, CTO

The INMS was one of 12 instruments aboard Cassini and arguably played the most critical role when it came to uncovering many of the mysteries of Enceladus. It's not imperative that you, as an employee of Red:4, have an intimate understanding of how the thing worked, but a general understanding will be helpful.

The INMS was a quadrupole mass spectrometer. NASA's use of it is [documented online](#):

The Ion and Neutral Mass Spectrometer (INMS) is intended to measure positive ion and neutral species composition and structure in the upper atmosphere of Titan and magnetosphere of Saturn and to measure the positive ion and neutral environments of Saturn's icy satellites and rings.



A quick Wikipedia search will give you a concise definition:

The quadrupole consists of four parallel metal rods. Each opposing rod pair is connected together electrically, and a radio frequency (RF) voltage with a DC offset voltage is applied between one pair of rods and the other. Ions travel down the quadrupole between the rods. Only ions of a certain mass-to-charge ratio will reach the detector for a given ratio of voltages: other ions have unstable trajectories and will collide with the rods. This permits selection of an ion with a particular m/z or allows the operator to scan for a range of m/z -values by continuously varying the applied voltage. Mathematically this can be modeled with the help of the Mathieu differential equation.

Note: *Wikipedia is not regarded as an authoritative source for anything here at Red:4; however, it is useful for finding reference material and generally getting started regarding understanding things.*

A mass spectrometer separates material into component elements like water separates light into its various wavelengths to create a rainbow. To do this, particles from space

are flooded with electrons which ionize them, making each particle gain a charge by either adding or stripping an electron. Once that's done, the ions are beamed through an electrical field produced by four metal rods.

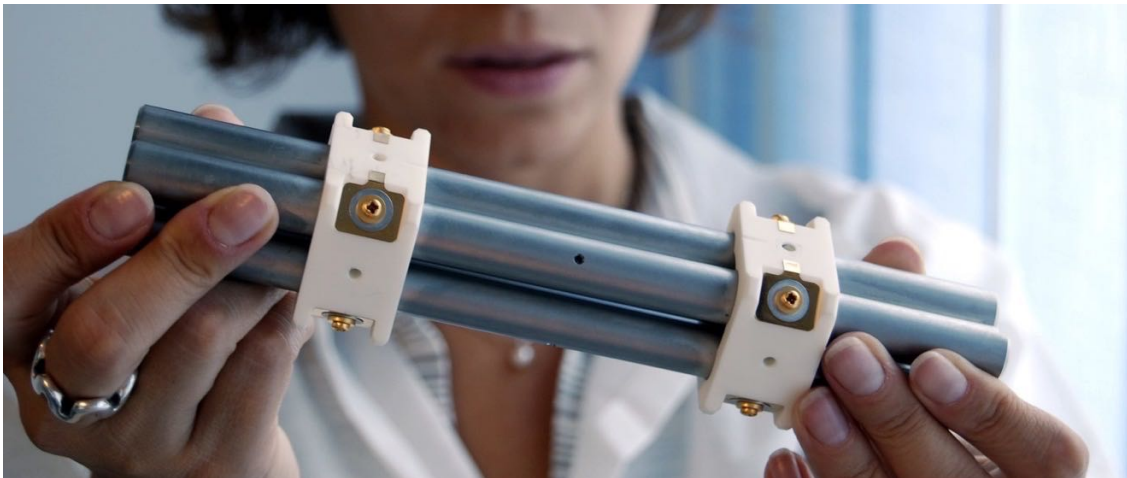
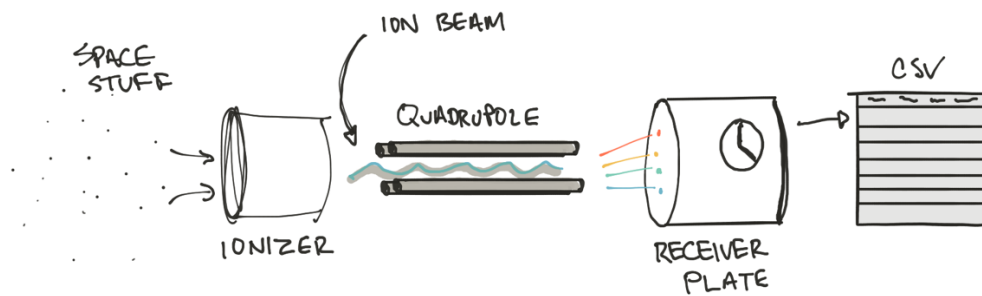


Photo credit: WMFLabs TUSC

This is the “quadrupole” part. Each of those rods carries an electric current, either positive or negative, that attracts or repels the ions in some way. More massive particles tend to deviate less, whereas lighter ones deviate more. This deviation creates a spectrum, like a particle rainbow. That spectral pattern is recorded by a sensor in the back of the spectrometer and can be compared with reference spectra, so scientists can tell what kind of material was encountered.

These readings are taken every 30ms (or thereabouts) in what are called integration periods. Each row in the INMS result CSVs files is a reading from an integration period. Each column has charge data, mass calculations, and ion deflection data, all of which I'll have to come back to at some point.

Here's a pretty crude illustration I threw together to justify the company's purchase of an iPad Pro for me:



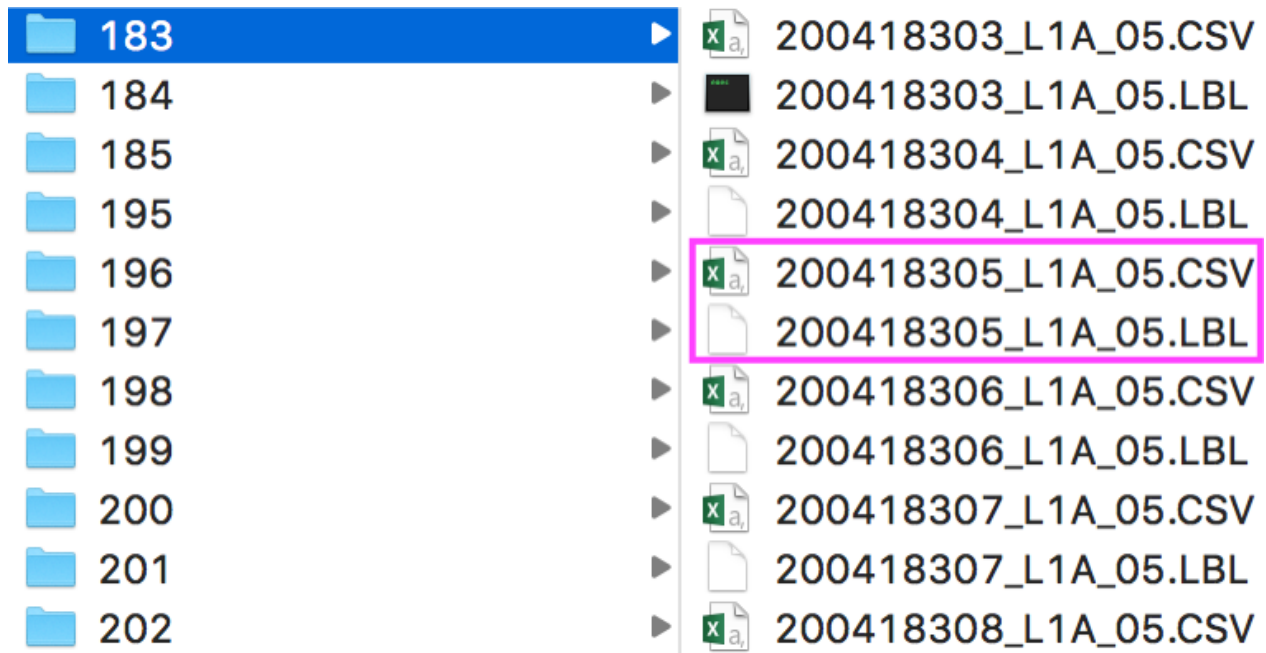
That clock thing is the 30ms timer, by the way.

The bottom line: the INMS can zap space stuff and tell us, with some degree of confidence, what material it's made out of.

WORKING WITH THE INMS CSVS

Short and sweet, I like it. I don't know if this is going to help me with my current task as I'm after something that will help me figure out precise flyby times.

The answer is in the data. I need to go over that manifest. Every CSV directory contains a series of CSVs of all the recordings for that day:



Each of these CSVs has an LBL file (which I assume means “label”) that describes what’s in the corresponding file and is really not all that useful. There’s a single FMT file at the very end which is a bit like a manifest — that’s the one I need.

I grabbed one of these files and renamed it to **inms_manifest.txt**. I made sure to save everything, including the manifest, in case someone ever needs to load it up again.

There are descriptions of spacecraft positional data, filament readings, data reliability and other sciency stuff. I still don’t understand most of it, but I can read more about it on my own time. Now, I need to find something that will help me derive the closest flyby.

Like THIS:

```
inms_manifest.txt — Edited
END_OBJECT = FIELD
OBJECT = FIELD
FIELD_NUMBER = 43
BYTES = 9
DATA_TYPE = ASCII_REAL
NAME = "ALT_T"
FORMAT = "F9.2"
DESCRIPTION = "Altitude of the spacecraft above the surface of the
named target body, included within 1 hour of
closest approach"
UNIT = "km"
VALID_MINIMUM = -1.0E05
VALID_MAXIMUM = 1.0E05
END_OBJECT = FIELD

OBJECT = FIELD
FIELD_NUMBER = 44
BYTES = 9
DATA_TYPE = ASCII_REAL
NAME = "VIEW_DIR_T_X"
FORMAT = "F9.6"
DESCRIPTION = "Components of the UNIT vector in the direction of the
INMS aperture outward normal expressed in the target
centered IAU coordinate frame, included within 1 hour"
```

Altitude data! That's perfect! The INMS recorded the altitude of the craft above Enceladus within a two-hour window of closest approach. This means I should be able to use that altitude data to figure out the precise closest approach: just see which altitude is lowest!

There's also a TARGET field. Yay! This means I can check the lowest altitude for Enceladus:


```
inms_manifest.txt — Edited
END_OBJECT = FIELD
OBJECT = FIELD
  FIELD_NUMBER = 3
  BYTES = 26
  DATA_TYPE = CHARACTER
  FORMAT = "A26"
  NAME = "TARGET"
  DESCRIPTION = "Name of target body."
END_OBJECT = FIELD
OBJECT = FIELD
  FIELD_NUMBER = 4
  BYTES = 8
  DATA_TYPE = ASCII_INTEGER
  NAME = "TIME_CA"
  FORMAT = "I8"
  DESCRIPTION = "Time since closest approach."
  UNIT = "ms"
  VALID_MINIMUM = -86400000
  VALID_MAXIMUM = 86400000
END_OBJECT = FIELD
```

I just need to be able to find “Enceladus” in the results, somehow.

Great, I have enough to go on. Let’s get this data in! I’ll get to use my wicked bash skills to make these CSVs dance!

“Mom! Can you help me load a bunch of pictures to MySpace? Don’t you have some kind of photo editor on here?”

Over the weekend my senior high school class had gone to Great America for our graduation trip, and I had borrowed my mom’s digital camera. I was the annoying friend, snapping pictures of everyone after they rode Batman and Demon knees wobbly, heads spinning. I thought it was hysterical...

I loved using a digital camera, too. I could take as many pictures as I wanted until I filled up the too-small memory card! I just had to plug it into mom's computer and zip it up to MySpace.

"Pictures from my camera? Those are way too big to load to MySpace Dee. You'll have to resize them."

"Mom! I know, OK!" I was kind of snotty at 17. "Just... how do you do it? I don't know how Macs work."

"Get your okole over here, and I'll show you then. No more stinkeye...". Mom tended to drop into pidgin just like dad, but he did it more often because that's how he grew up. My mom just adopted it.

She grew up in Southern California, near Los Angeles and her family moved to Honolulu when she was 8 to get away from the crowds and crappy, smoggy air. The move traumatized her. She lost all of her friends and had to adjust to living on a small island with a big city populated by highly-territorial kids who didn't like outsiders.

She went to Midpac and then transferred to Punahou, the same school President Obama went to — evidently just 3 years apart.

Even though they lived in the same city for most of their lives, my parents never met until college, at UC Santa Barbara in the late 1970s and early 1980s. My dad studied Geology and Structural Engineering, mom got a degree in Computer Science.

My dad had a couple of summer internships in San Francisco, one of which was helping with the excavation of the *Niantic*. The engineering company hired him on, and he worked there until he passed away a few years back... which still hurts.

My mom got a job at Sun Systems right out of school and later went back for her Ph.D. at Stanford. Nowadays, she does bioinformatics at UCSF Medical Center.

At 17, I could never figure out why anyone would want to have a job staring at a computer all day. Especially after growing up in Hawaiian sunshine and going to a school like UC Santa Barbara; another place full of sun and warmth, unlike where I grew up in San Fran-crappy-freezing-cisco.

“Shall we do this the easy way or the hard way?” my mom says. “I guess I don’t care, just let me know when you’re done” I reply.

“Oh ho, checkout tidda! I don’t think I’ll be the one doing all this work, I have work of my own sweetie. You do em, and I’ll help bumbye.”

“Ok... then where’s the program?”

“It’s called Terminal, and it’s already open. We’re going to write an incantation that will resize those pictures in 2, maybe 3 seconds. Woulda took you like one hour with da kine, but a shell script? We geet em li’dat!” she snaps her fingers.

“Mom, you’re haole. You can’t — ouch!”

“MOM!!! DEE called you a haole!” my sister yelled from the other room.

“Shut up, Maile!” I yelled. I hated my sister when I was 17. She loved it when I got in trouble and made sure to point out anything I did wrong, as much as she could. Little brat.

“You know the rule, Dee. Say that word and get a pinch. Besides, pidgin is fun. I was pretty good at it when I was your age. I had to be” she says with that smile that says *akamai*. Don’t push.

Even though mom was Asian, the local kids in Hawaii in the 1970s were rough. They knew transplants when they saw them, no matter the color of your skin. White kids got it worse than Asian, but haole was haole until you proved yourself. Which meant fighting back or withstanding the constant insults, all a variation of “go home.”

I asked her why Hawaiian kids were like that, why there was so much animosity for outsiders. “It’s a complicated answer, Dee, but in short it’s because outsiders have not been very good to the people that have lived over there for generations. The kids I went to school with never experienced what their grandparents did, but they grew up hearing the stories, learning to dislike outsiders. Ultimately, it’s something that takes a long time to heal in the best of circumstances. And poking at that wound, or telling them to hurry up and get over it, only makes the situation worse. *They* feel this way; it’s not our *kuleana* to change it” my mom said.

That is not the answer I wanted to hear. Why do I have to understand why someone else doesn't like me just because of where I'm from? Seemed like a bunch of bullsh—

“I can tell you don't like my answer, Deedle, but you're going to have to get used to the idea that the problem isn't yours. Especially don't you ever call anyone haole in anger. It carries a meaning that is very deep, one that you'll never fully understand. If you do say that word, I pinch!” she finished.

I roll my eyes, a specialty of mine. This was going to end up nowhere, and I'd never get my pictures loaded. She never pinched me that hard, anyway, so I guess it didn't matter. I just didn't like it, so I yelled every time. I always knew it was coming, and I always felt like it would be worth it. It never was.

“Fine — can you just show me how to do the script thing then?”

“Type this out...” and she dictated commands for me to input into the terminal. I changed directories and ended up on the Desktop, where I could see my photos folder that I copied from her camera. She had me navigate into that directory, and then write this script:

```
mkdir fixed && mogrify -resize 500x300 -path fixed
```

There were 32 huge JPEG files in there, and resizing each one in a photo editor would have taken me 10, maybe 15 minutes. This script ran in 3 seconds.

I could see the new fixed directory pop up in the finder window as if by magic, and then all of a sudden... there were my resized pictures.

“Woah! How did you learn to do that mom? That's really cool!” “It's easier than you think, Deedle,” she said. The pictures were a little blurry, I remember, so we tweaked a few settings for Image-Magick, the program that I was using with the “mogrify” command. Eventually, we had it just right.

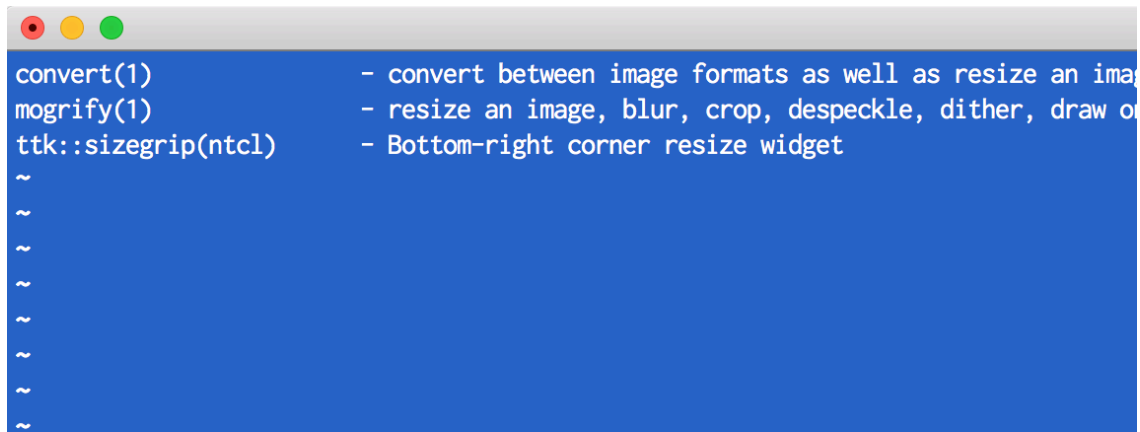
“Now let's load them up with **curl**” she said.

The power contained in so few commands... wow. I was hooked.

WHO REMEMBERS THIS STUFF?

I like working in the shell, but every time I do I need to have Google open in the next window. I forget everything! The **man** pages are helpful, sometimes, but it's tough to find what you're looking for. The commands are also kind of ridiculous.

For instance, if I wanted to figure out today how I could use an installed binary to resize images I could use the **apropos** command:



```
convert(1)          - convert between image formats as well as resize an image
mogrify(1)          - resize an image, blur, crop, despeckle, dither, draw on
ttk::sizegrip(ntcl) - Bottom-right corner resize widget
~
~
~
~
~
~
~
~
~
```

Who remembers this stuff? “Apropos” is another one of those words that sound so elitist, too. I can see M. Sullivan eye-rolling me with it as I ask about database backups.

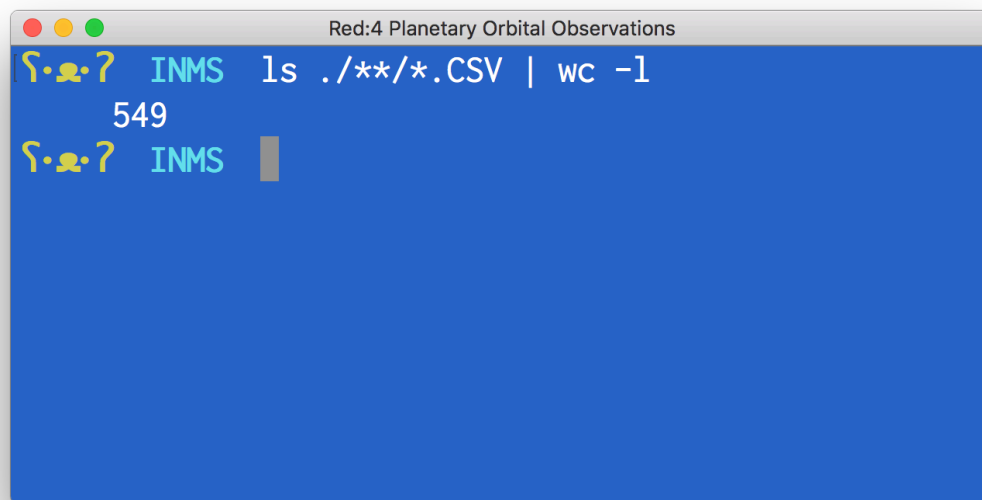
I have my Google window open, and my bash skills warmed up, let's do this.

What's Better Than 549 CSV Files?

That would be one CSV file. Hopefully, formatted properly. This process was a pain in the ass, but I have reasonable bash skills, so that helped somewhat.

I started out by checking to see how many CSV files I actually had. I navigated to the download directory containing the INMS dump and did a list command on a CSV glob, piping it to word count:

```
ls ./\*\*/\*.CSV | wc -l
```

A terminal window with a blue background and a title bar that reads "Red:4 Planetary Orbital Observations". The prompt is a yellow icon followed by "INMS". The command entered is "ls ./**/*.CSV | wc -l". The output is "549". The prompt "INMS" is followed by a cursor.

```
Red:4 Planetary Orbital Observations
INMS ls ./**/*.CSV | wc -l
549
INMS
```

That's a lot of CSVs. At this point, I have a choice:

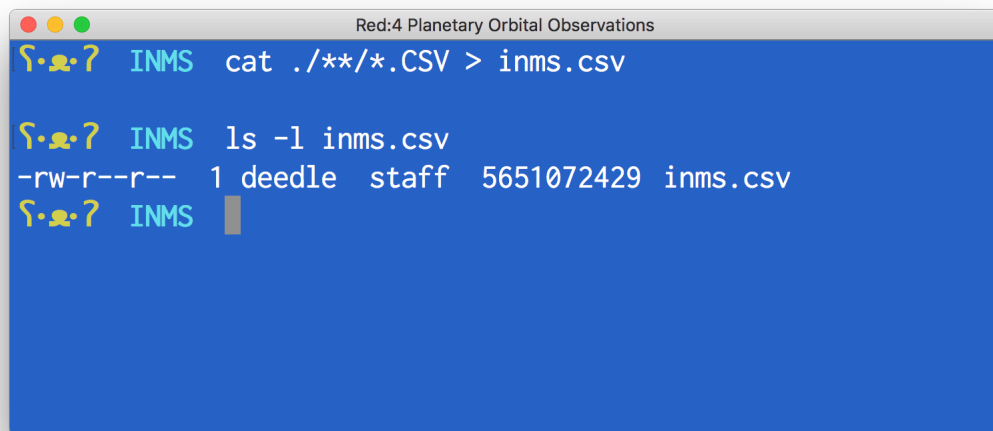
- Loop over the CSVs returned from the `ls` command and then run a `COPY FROM` command to import it
- Concatenate every single CSV into one, and then import that file

The second answer is the better one, simply because of the transactional nature of it. If something fails, I won't have a partially copied table.

I should be able to simply concatenate into a new file:

```
cat ./\*\*/\*.CSV > inms.csv
```

Not exactly fast (took a few minutes) but it worked, and the result file is gigantic:

A terminal window titled "Red:4 Planetary Orbital Observations" with a blue background. It shows a user with a shell icon and "INMS" as the username. The first command is "cat ./**/*.CSV > inms.csv". The second command is "ls -l inms.csv", which outputs "-rw-r--r-- 1 deedle staff 5651072429 inms.csv". A cursor is visible on the line following the command.

```
Red:4 Planetary Orbital Observations
? ? ? INMS cat ./**/*.CSV > inms.csv

? ? ? INMS ls -l inms.csv
-rw-r--r-- 1 deedle staff 5651072429 inms.csv
? ? ? INMS
```

That's 5.6 gigs! I miss my GUI tools. Navicat would import this file automatically for me, creating the fields on the fly.

I need to go in and have a look at one of the smaller CSVs and do it by hand. I'll use the very first file in 2005/048, just because it's first and also a lot smaller. It should have the same structure as the others:

	A	B	C	D	E	F	G	H	I	J	K
1	sclk	uttime	target	time_ca	targ_pos_x	targ_pos_y	targ_pos_z	source	data_reliabil	table_set_id	coadd_cnt
2		ms		ms	km	km	km				
3	sclk	uttime (ms)	target	time_ca (ms)	targ_pos_x (km)	targ_pos_y (km)	targ_pos_z (km)	source	data_reliabil	table_set_id	coadd_cnt
4	2005-048T0C	199322	EPIMETHEUS	-742328	80616	128804	-498	csn	0	156-1	30
5	2005-048T0C	199356	EPIMETHEUS	-742294	80617	128803	-498	csn	0	156-1	30
6	2005-048T0C	199390	EPIMETHEUS	-742260	80617	128803	-498	csn	0	156-1	30
7	2005-048T0C	199424	EPIMETHEUS	-742226	80617	128803	-498	csn	0	156-1	30
8	2005-048T0C	199458	EPIMETHEUS	-742192	80617	128803	-498	csn	0	156-1	30
9	2005-048T0C	199492	EPIMETHEUS	-742158	80618	128803	-498	csn	0	156-1	30
10	2005-048T0C	199526	EPIMETHEUS	-742124	80618	128803	-498	csn	0	156-1	30
11	2005-048T0C	199560	EPIMETHEUS	-742090	80618	128802	-498	csn	0	156-1	30

There's a header row, followed by a row with unit information that I don't need as I have the manifest. The header row is then repeated, verbatim, in row 3.

I had a look at some other CSVs, and I'm confident that every single one of them is formatted in this way. I could be wrong, but I'll spot-check the data once I get it imported.

I now face a new choice: do I try and tweak the CSVs in place? I need to remove rows 2 and 3, and I'm sure if I Googled long enough I could find a solution.

I could also just import what I have, in totally raw condition, and then clean it up with simple queries. They would be faster and simpler to use.

Using the database to clean things up seems like the obvious choice, but that's assuming that I can get the data in without error, which I don't have confidence in.

I wonder if M. Sullivan has some ideas.

From: M. Sullivan sullz@redfour.io
Subject: RE: CSV hell is kinda hot
To: Dee Yan yand@redfour.io
Date: October 27, 2017

There are some excellent choices when it comes to working with CSVs and no, my desire to “build everything in-house despite common sense and tools readily available that do the job better than we ever could” does not preclude using essential utilities, such as [csvkit](#).

I appreciate that you would like to offload your responsibility as a programmer and database person to an expensive toolset, but I have a feeling you’re a bit more capable than that.

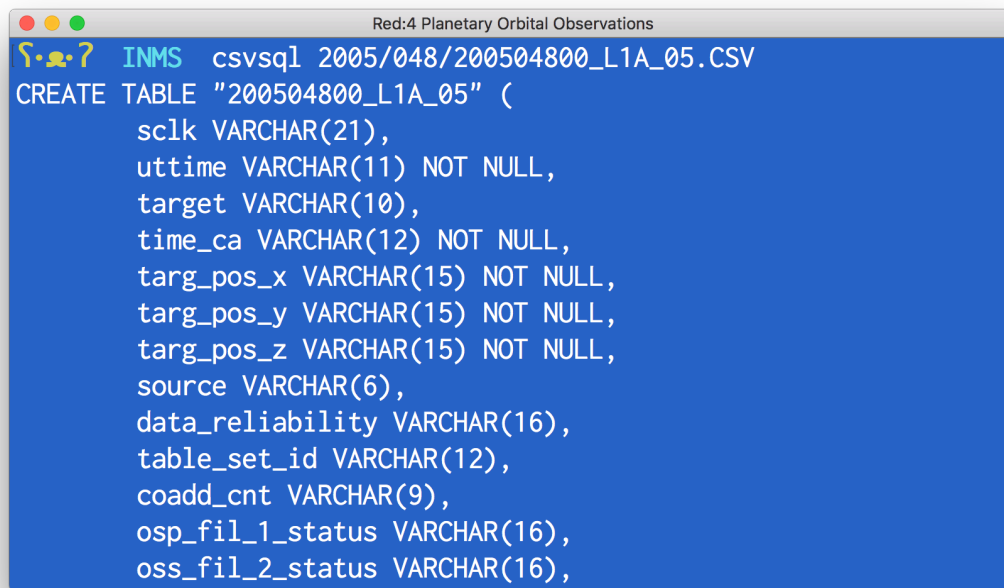
Best, S

USING CSVSQL

If I don’t get this job, M. Sullivan will be the person I miss most. I’m thinking less Tim Gunn/Snape and more Baroness Schraeder from Sound of Music.

Have to admit, he’s very helpful, however. Looks like csvkit is the perfect thing, it even has a tool to generate SQL statements and, best of all, will insert the CSV right in the database for you!

I need to generate the table first. If I just use **csvsql** with a given file, it will only create the table definition. To do this it needs to read in the entire, huge file, so I’ll just use the smaller one I’ve been looking at:

A terminal window titled "Red:4 Planetary Orbital Observations" with a blue background. It shows a command prompt where the user has entered `INMS csvsql 2005/048/200504800_L1A_05.CSV`. The output is a SQL `CREATE TABLE` statement for a table named "200504800_L1A_05". The table has columns: `sclk` (VARCHAR(21)), `uttime` (VARCHAR(11) NOT NULL), `target` (VARCHAR(10)), `time_ca` (VARCHAR(12) NOT NULL), `targ_pos_x` (VARCHAR(15) NOT NULL), `targ_pos_y` (VARCHAR(15) NOT NULL), `targ_pos_z` (VARCHAR(15) NOT NULL), `source` (VARCHAR(6)), `data_reliability` (VARCHAR(16)), `table_set_id` (VARCHAR(12)), `coadd_cnt` (VARCHAR(9)), `osp_fil_1_status` (VARCHAR(16)), and `oss_fil_2_status` (VARCHAR(16)).

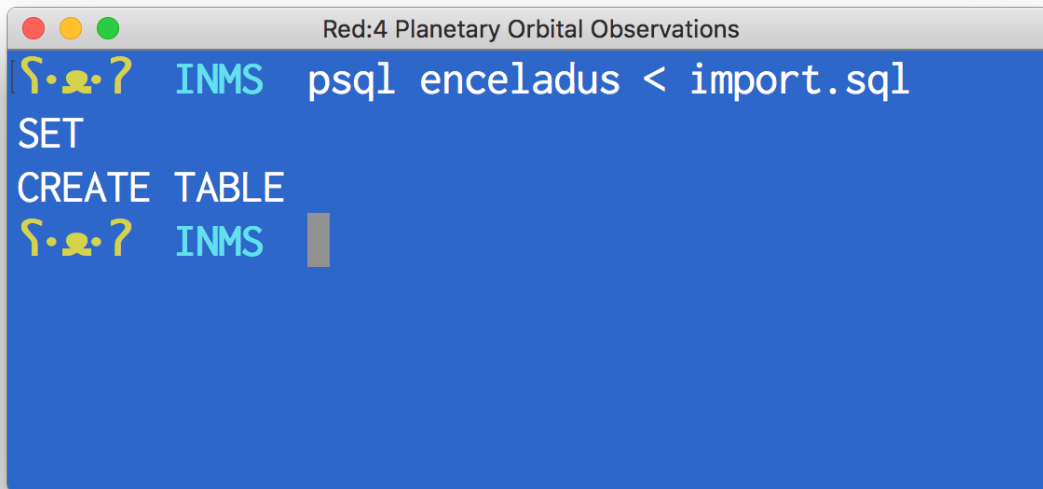
```
INMS csvsql 2005/048/200504800_L1A_05.CSV
CREATE TABLE "200504800_L1A_05" (
  sclk VARCHAR(21),
  uttime VARCHAR(11) NOT NULL,
  target VARCHAR(10),
  time_ca VARCHAR(12) NOT NULL,
  targ_pos_x VARCHAR(15) NOT NULL,
  targ_pos_y VARCHAR(15) NOT NULL,
  targ_pos_z VARCHAR(15) NOT NULL,
  source VARCHAR(6),
  data_reliability VARCHAR(16),
  table_set_id VARCHAR(12),
  coadd_cnt VARCHAR(9),
  osp_fil_1_status VARCHAR(16),
  oss_fil_2_status VARCHAR(16),
```

The create statement is output to STDOUT, which is fine as it allows me to look at the generated bits, which is a good thing as I don't like that table name and I don't like any of the constraints it's using. I want to be sure there's no failure.

Having a look at the help bits (using **csvsql --help**), it looks like I can get what I want using this command:

```
csvsql 2005/048/200504800\_L1A\_05.CSV -i postgresql --tables
"import.inms" --no- constraints
```

Looks good to me! I'll push to a file I'll call **import.sql** and then see if Postgres will load it:



```
Red:4 Planetary Orbital Observations
[?·?·?] INMS psql enceladus < import.sql
SET
CREATE TABLE
[?·?·?] INMS
```

Yes! It works! Behold my —

From: M. Sullivan sullz@redfour.io
Subject: Text, M. Yan.
To: Dee Yan yand@redfour.io
Date: October 27, 2017

You have a remarkable aptitude for triggering my monitoring alarms, M. Yan. Nicely done.

Given that you are working on an import table that will ultimately be deleted, I would ask you to please use the **text** datatype instead of **varchar**, please? It may seem counterintuitive, but there is no difference to Postgres, under the hood, between **text**, **varchar**, and **char**. Here, allow me to Google that for you and pull up some [relevant documentation](#):

There is no performance difference among these three types, apart from increased storage space when using the blank-padded type, and a few extra CPU cycles to check the length when storing into a length-constrained column. While `character(n)` has performance advantages in some other database systems, there is no such advantage in PostgreSQL; in fact, `character(n)` is usually the slowest of the three because of its additional storage costs. In most situations `text` or `character varying` should be used instead.

I believe speed is our primary concern, so **`text`** should be your first choice, without length.

Best, S

Alright. Did not know that. M. Sullivan is like my own personal “just in time” help system.

Turns out you can’t tell **`csvsql`** to use **`text`** instead of **`varchar`**, but I know how to use **`sed`** which means I win:

```
csvsql 2005/048/200504800_L1A_05.CSV -i postgresql --tables
"import.inms" --no- constraints -overwrite | sed 's/VARCHAR/ text/g'
> import.sql
```

Having a quick look at the generated SQL, and it’s almost usable:

```
INMS cat import.sql
CREATE TABLE "import.inms" (
    sclk text,
    uttime text,
    target text,
    time_ca text,
    targ_pos_x text,
    targ_pos_y text,
    targ_pos_z text,
    source text,
    data_reliability text,
    table_set_id text,
    coadd_cnt text,
    osp_fil_1_status text,
    oss_fil_2_status text,
    csp_fil_3_status text,
    css_fil_4_status text,
    seq_table text,
    cyc_num text,
```

The problem is the name of the table: *it's delimited*.

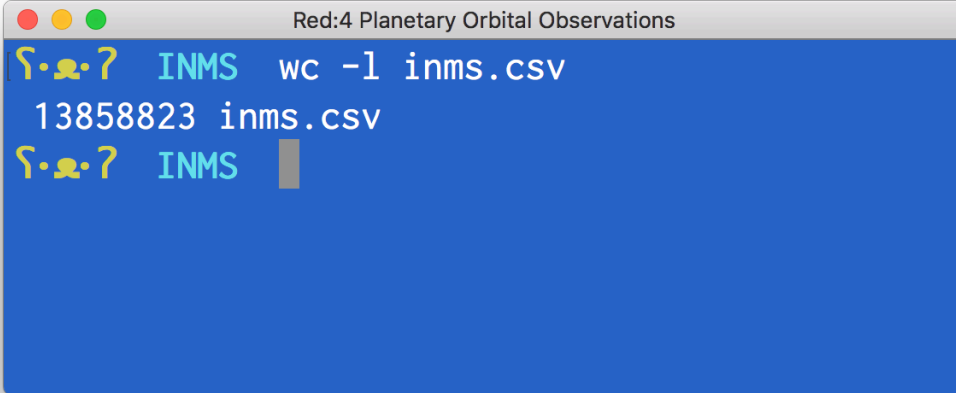
Postgres will do its best to keep you from creating dumb table names, so if you decided to get your .NET on inside a Postgres database by creating **AReallyPoorlyNamedTable**, Postgres would down-case all of it for you as that is the favored convention.

If you really want to have your way, however, you can delimit the table name with quotes. That's the problem with this script: I will end up with a single table in the **public** schema named **import.inms**.

I don't want that, so I'm going to need to edit this file. Which is OK as I need to add a drop statement at the top anyway, and I wanted to add the **COPY FROM** statement below. I'll add that now and put it next to the **inms.csv** file.

Speaking of, I should really check to see how many rows are in this file:

```
wc -l inms.csv
```

A terminal window titled "Red:4 Planetary Orbital Observations" with a blue background. The prompt is a yellow icon followed by "INMS". The command "wc -l inms.csv" has been entered, and the output "13858823 inms.csv" is displayed on the next line. The cursor is at the end of the second line.

```
INMS wc -l inms.csv
13858823 inms.csv
INMS
```

Dang! 13 million records. This should be interesting. I've added the **COPY FROM** command to the import file. Let's see what happens:

```
Red:4 Planetary Orbital Observations
scripts psql enceladus < import.sql
SET
DROP TABLE
CREATE TABLE
ERROR: extra data after last expected column 🍌
CONTEXT: COPY inms, line 11769104: "sclk,mjday,uttime,target,time_ca
,targ_pos_x,targ_pos_y,targ_pos_z,source,data_reliability,table_set_
.."
scripts
```

Oh well. I guess it was a bit optimistic of me. There's an error around like 11 million or so... Googling now.

An “extra data” error means that the CSV contains more fields than my table does. Looking at the context provided by Postgres, I can see that there is, indeed, an extra column and it's the second one: **mjday**. Crap.

I don't know why the format changed all of a sudden in 2015, so I decided to have a look at the FMT file for the very first result set from that year.

Look at this, would ya:

```
L1A_STRUCT_06.FMT
/* 04-Mar-2005 DA Gell updates per J. Mafi */
/* 31-Mar-2005 DA Gell merged filament energy and status fields */
/* 04-Apr-2005 DA Gell revised names of filament status */
/* 08-Apr-2005 DA Gell replaced unit names with abbreviations */
/* 07-Jun-2005 DA Gell corrected lengths and formats of some items */
/* 16-Jun-2005 DA Gell Post-Peer Review */
/* changed ASCII_INT to ASCII_INTEGER */
/* ASCII_FLOAT to ASCII_REAL */
/* added column numbers */
/* 20-Jun-2005 DA Gell Post-Peer Review II */
/* Add note that target relative geometric */
/* quantities are present within 1 hour of CA */
/* Corrected Typos */
/* Added ion source mnemonics definition */
/* 23-Jun-2005 DA Gell Added closing quotation marks where missing */
/* 25-Jul-2008 DA Gell Corrected several field descriptions for */
/* which the byte length (BYTES) or the range */
/* (VALID_MINIMUM or VALID_MAXIMUM) are wrong */
/* 30-Apr-2012 DA Gell Version 6. */
/* Added fields MJDATE, C1RATE, C1ERROR, */
/* C2RATE and C2ERROR. */
/* Corrected volts unit abbreviation */
/* to V (ex v) */
/* 04-May-2012 DA Gell Added format keyword where missing: */
/* SEQ_TABLE, SCAN_NUM */
/* Fixed formats for ION DEF1, IONDEF2 */
/* 07-May-2012 DA Gell Moved C1COUNTS and C2COUNTS to the end of */
/* the file for compatability with analysis */
/* library. */
/* 17-Sep-2013 RThorpe Added ram_angle_t and clock_angle_t fields */
/* 27 Jan 2014 rthorpe Set format for ram_angle_t and clock_angle_t */
/* to F8.3 per instructions from Dave */
/* */
/*****
```

At the very top of the file are all the changes made over the years to the result sets. Right there, in the middle, is the note that **MJDATE** was added, along with a few other fields. Hmmm.

Reading over this it looks like other fields have been changed around too. However, I went back and checked the FMT files for all the years mentioned, up to 2012, and they all look the same. Only 2015 is different, which is odd.

Again, two ways to get around this:

1. I use some more bash magic to remove the extra column using something like **cut -d -f2 --complement inms.csv**
2. I load 2015 as a separate table and insert things manually. Which is a pain, but I guess this is the nature of ETL.

If it was just a matter of a single extra column, I could just use **cut**, but the manifest says there are other columns added as well.

What am I saying? This manifest says all kinds of whacky things — I need to look at both files using Excel.

SUCCESSFUL IMPORT

October 28, 2017, 0934

That wasn't too bad, I guess. There are 7 different columns in the 2015 CSVs, which I went through and cross-checked with the manifest. They are columns 2,48,49 and 78 through 81. If I remove those columns, I have parity.

That's simple enough to do with cut:

```
cut -d ',' -f1,3-47,50-77,82-83 2015.csv > inms_2.csv
```

The problem is that I have to produce two separate CSV files.

The 2015 one is easy:

```
cat 2015/*\*/\*.CSV > 2015.csv
```

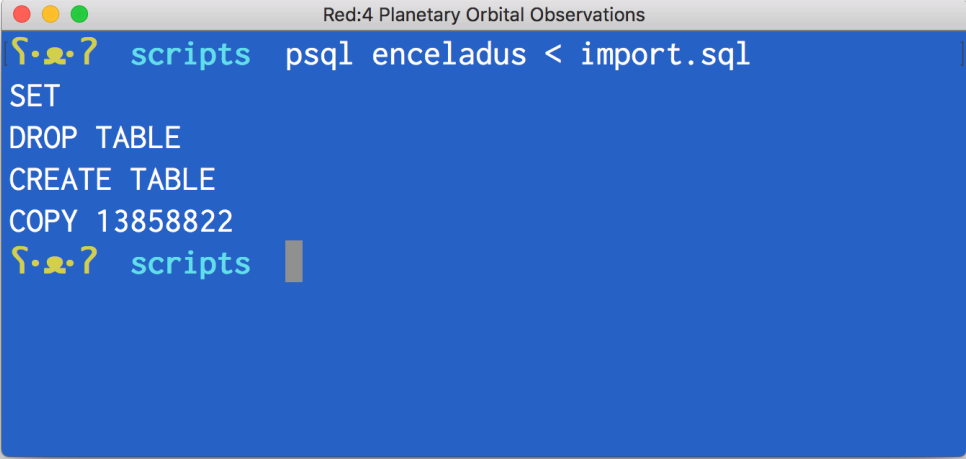
It's the other ones that require more effort. I started looking at how to ignore files using glob patterns and finally realized I was being a dope and just moved all the “good” years into a “good” directory:

```
cat good/*\*/\*.CSV > inms\_1.csv
```

And there you have it. I can combine the two into a final CSV:

```
cat inms\_1.csv inms\_2.csv > inms.csv
```

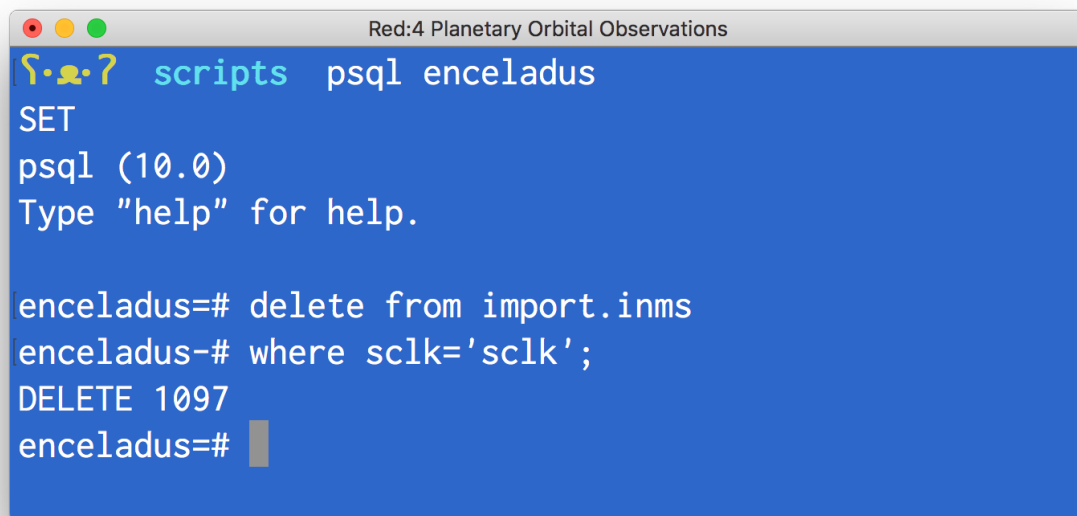
I can use this with my **import.sql** script:



```
Red:4 Planetary Orbital Observations
$ ./scripts psql enceladus < import.sql
SET
DROP TABLE
CREATE TABLE
COPY 13858822
$ ./scripts
```

YES! 13 million rows of data cleanly imported in under a minute. RAWRRRR!

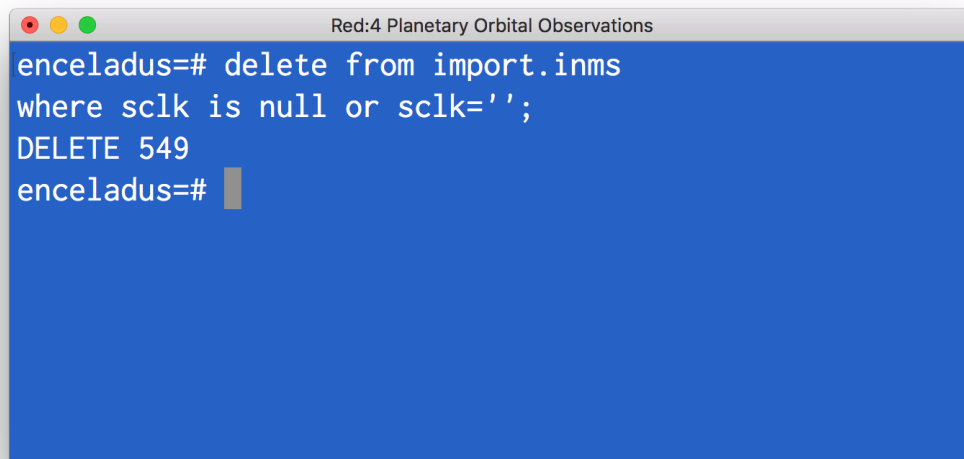
Now I have to clean things up. I'll start by removing all values which contain a header:



```
Red:4 Planetary Orbital Observations
[?] scripts psql enceladus
SET
psql (10.0)
Type "help" for help.

enceladus=# delete from import.inms
enceladus=# where sclk='sclk';
DELETE 1097
enceladus=#
```

I'll do another delete statement, making sure the spacecraft clock has a value:

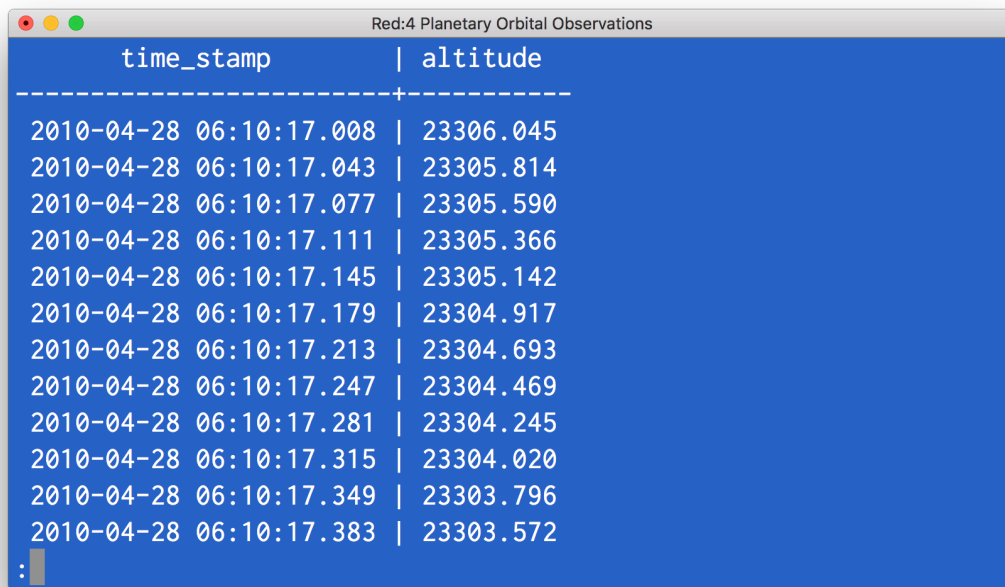


```
Red:4 Planetary Orbital Observations
enceladus=# delete from import.inms
where sclk is null or sclk='';
DELETE 549
enceladus=#
```

Much better. Now I need to be sure that all the data I need can be appropriately cast. Right now, the only thing I want to see is the timestamp and the altitude, and only for the 'ENCELADUS' target (which I found in the CSV):

```
select
  (sclk::timestamp) as time_stamp,
  alt_t::numeric(10,3) as altitude
from import.inms
where target='ENCELADUS'
and alt_t IS NOT NULL;
```

The key here is to make sure the timestamp read in from the data is cast to UTC as that's what the timestamp is set to, according to the manifest. It should be OK, but I want to be sure that it's stored in the view I'm about to create as UTC. Otherwise, the readings might be off.



time_stamp	altitude
2010-04-28 06:10:17.008	23306.045
2010-04-28 06:10:17.043	23305.814
2010-04-28 06:10:17.077	23305.590
2010-04-28 06:10:17.111	23305.366
2010-04-28 06:10:17.145	23305.142
2010-04-28 06:10:17.179	23304.917
2010-04-28 06:10:17.213	23304.693
2010-04-28 06:10:17.247	23304.469
2010-04-28 06:10:17.281	23304.245
2010-04-28 06:10:17.315	23304.020
2010-04-28 06:10:17.349	23303.796
2010-04-28 06:10:17.383	23303.572

Great! Now I can turn this into a materialized view to speed things up:

```
drop materialized view if exists flyby_altitudes;
create materialized view flyby_altitudes as
select
  (sclk::timestamp) as time_stamp,
  alt_t::numeric(10,3) as altitude
from import.inms
where target='ENCELADUS'
and alt_t IS NOT NULL;
```

Perfect. Now I need to figure out a way to get at the lowest altitudes for each flyby.

INSPECTING THE DATA

October 28, 2017, 1309

The simplest thing to do is load up the lowest altitude for a given day, which I think should be the first flyby on February 17, 2005:

```
select min(altitude)
from flyby_altitudes
where time_stamp::date = '2005-02-17';
```

This is the fun part of my job. Look at that!

```
Red:4 Planetary Orbital Observations
enceladus=# select min(altitude)
enceladus=# from flyby_altitudes
enceladus=# where time_stamp::date = '2005-02-17';
min
-----
1272.075
(1 row)

enceladus=#
```

1272 kilometers is the lowest altitude. The query is nice and fast, too, since I'm using a materialized view.

Now I want to see the lowest point of every flyby now, so I remove the where clause:

```
select time_stamp,
min(altitude)
from flyby_altitudes
order by min(altitude);
```

```
Red:4 Planetary Orbital Observations
enceladus=# select time_stamp,
enceladus=# min(altitude)
enceladus=# from flyby_altitudes
enceladus=# order by min(altitude);
ERROR: column "flyby_altitudes.time_stamp" must app
ear in the GROUP BY clause or be used in an aggregat
e function
LINE 1: select time_stamp,
              ^
enceladus=#
```

Guess I got a bit ahead of myself. Dee! Remember: if you do an aggregate, and you want to see other values along with it, you have to use GROUP BY.

I want to see **time_stamp** along with the minimum values, so I need to use a GROUP BY clause:

```
select time_stamp,
min(altitude)
from flyby_altitudes
group by time_stamp
order by min(altitude);
```

Much better. No errors, but also really, really slow. That's OK as soon I'll be creating a flybys table, and that will mitigate all of this.

I wanted to see the data first, however. There's a lot of it:

time_stamp		min
-----+-----		
2008-10-10	02:06:39.741	28.576
2008-10-10	02:06:39.707	28.576
2008-10-10	02:06:39.775	28.577
2008-10-10	02:06:39.673	28.577
2008-10-10	02:06:39.639	28.579
2008-10-10	02:06:39.809	28.580
2008-10-10	02:06:39.605	28.583
2008-10-10	02:06:39.843	28.584
2008-10-10	02:06:39.571	28.589
2008-10-10	02:06:39.877	28.590
2008-10-10	02:06:39.537	28.595
2008-10-10	02:06:39.911	28.596
2008-10-10	02:06:39.503	28.603
2008-10-10	02:06:39.945	28.605
2008-10-10	02:06:39.469	28.612

The closest Cassini ever came to Enceladus was at 2:06AM UTC on October 10, 2008. I heard Michele Dougherty talking about this on a [BBC radio show](#) that I listened to last night:

The Project said they'd never do that again because the plume was so dense the spacecraft almost tumbled, so we won't go as close as that again!

Kind of wild that I can see that pass, right here. Chicken skin! The INMS records data every 30-50ms which presents a bit of a problem: how am I supposed to figure out which altitude is the nadir, or the lowest point of the flyby?

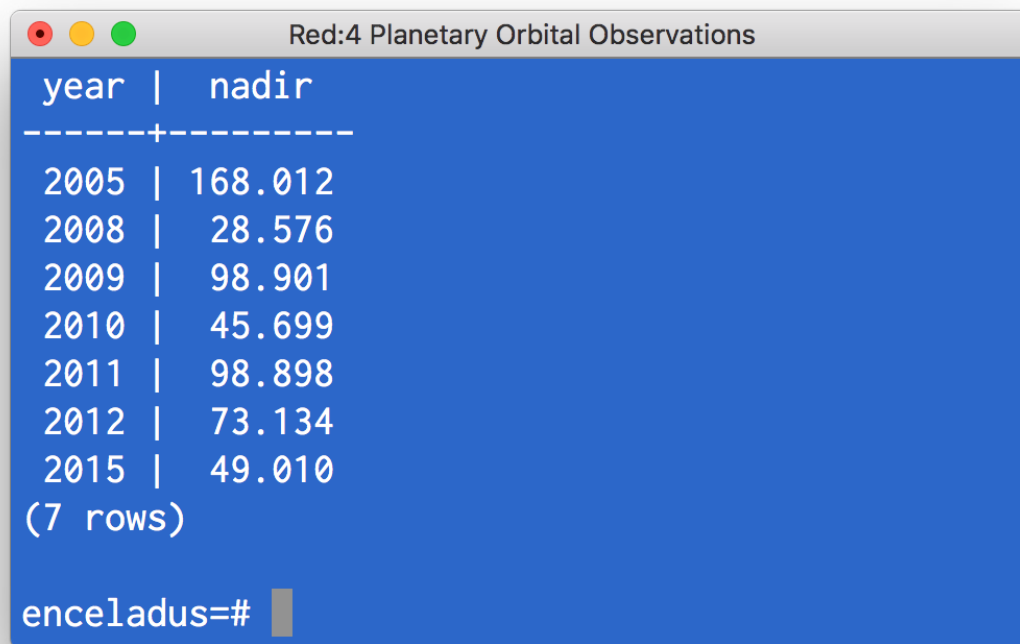
CHERRY PICKING THE NADIR

I need to run an aggregate query, like the one I just ran. Instead of grouping by **time_stamp**, however, I need to group by something that will separate the data a bit into smaller subgroups.

I can group by year:

```
select
date_part('year',time_stamp) as year,
min(altitude) as nadir
from flyby_altitudes
group by date_part('year',time_stamp);
```

Which will show me the closest approaches to Enceladus each year a flyby was performed:



year	nadir
2005	168.012
2008	28.576
2009	98.901
2010	45.699
2011	98.898
2012	73.134
2015	49.010

(7 rows)

enceladus=#

Interesting, but not useful. I need a finer interval. I could add months to the query:

```
select
  date_part('year',time_stamp) as year,
  date_part('month',time_stamp) as month,
  min(altitude) as nadir
from flyby_altitudes
group by
  date_part('year',time_stamp),
  date_part('month',time_stamp);
```

This is getting closer:

year	month	nadir
2005	2	1272.075
2005	3	500.370
2005	7	168.012
2008	3	50.292
2008	8	53.353
2008	10	28.576
2009	11	98.901
2010	4	3771.195
2010	5	437.292
2010	8	2555.180
2010	11	45.699
2010	12	48.324
2011	10	98.898
2011	11	496.603
2012	3	74.165
2012	4	74.100
2012	5	73.134
2015	10	49.010
2015	12	5000.200

(19 rows)

:

Again: interesting but not useful. I have 19 results here, but I need 23. This result is lumping together lowest results that occurred in the same month and year, which evidently there were 4 of.

I need a better interval.

At this point, I should be able to query by day, to find what I'm looking for as the only data I have are for the flyby dates; no other. That means I should be able to group by date to get the flybys I want:

```
select
  time_stamp::date as date,
  min(altitude) as nadir
from flyby_altitudes
group by
  time_stamp::date;
order by date;
```

I'm down to the date level, but this presents some interval problems as well:

Red:4 Planetary Orbital Observations	
date	nadir
2005-02-17	1272.075
2005-03-09	500.370
2005-07-14	168.012
2008-03-12	50.292
2008-08-11	53.353
2008-10-09	28.576
2008-10-31	173.044
2009-11-02	98.901
2009-11-21	1596.561
2010-04-27	3778.264
2010-04-28	3771.195
2010-05-18	437.292
2010-08-13	2555.180
2010-08-14	36876.719
2010-11-30	45.699
2010-12-21	48.324
2011-10-01	98.898
2011-10-19	1230.674
2011-11-06	496.603
2012-03-27	74.165
2012-04-14	74.100
2012-05-02	73.134
2015-10-14	1844.230
2015-10-28	49.010
2015-12-19	5000.200
(25 rows)	
:	

I have 25 rows returned, but I know that there are only 23 total flybys, which means my logic is flawed. I can see that because two sets of dates are consecutive, which is wrong. I could probably just pick the lower of the two altitudes for these sets if I felt like incurring the wrath of M. Sullivan. Correct and proper it is.

These flybys had to happen near midnight, and my intervals must be too fine-grained, so data is getting spread between two calendar days.

So: what to use? I know that flybys can't be a single day apart, they have to be separated by around 2 weeks, so Cassini can slingshot around Titan or Saturn.

Let's try the query one more time, grouping by week:

```
select
  date_part('year',time_stamp) as year,
  date_part('week',time_stamp) as week,
  min(altitude) as altitude
from flyby_altitudes
group by
  date_part('year',time_stamp),
  date_part('week',time_stamp);
```

Bingo:

Red:4 Planetary Orbital Observations		
year	week	altitude
2005	7	1272.075
2005	10	500.370
2005	28	168.012
2008	11	50.292
2008	33	53.353
2008	41	28.576
2008	44	173.044
2009	45	98.901
2009	47	1596.561
2010	17	3771.195
2010	20	437.292
2010	32	2555.180
2010	48	45.699
2010	51	48.324
2011	39	98.898
2011	42	1230.674
2011	44	496.603
2012	13	74.165
2012	15	74.100
2012	18	73.134
2015	42	1844.230
2015	44	49.010
2015	51	5000.200
(23 rows)		

RAWWRRRRRRRR! Much better!

Each one of these dates corresponds with the [published flyby dates](#)... except for the first one, of course, which wasn't considered a formal flyby (apparently).

Now for the next part of the puzzle: *getting the exact timestamp*.

San Francisco is a constantly changing city. I remember watching the news in the morning with my dad, seeing the traffic report and my dad being aghast that the line for the Bay Bridge was all the way back to the metering lights! The wait was a whopping 10 minutes to get through the tolls.

Seems crazy to think about that now. That was 1995, today... it's nuts. The area around our bunker office has gone from funky little warehouses and textiles to almost 100% high tech. People in the city are very annoyed, but then it's quickly pointed out to them how the city changed when they got here too.

I like to make my way down 2nd street when it's nice out, getting lunch at the Mexican place in South Park. The little park gets a lot of sun for being downtown among the buildings, and it's a great way to get your mind off work.

The burritos at Mexico Au Pare are amazing. Just the right size, too. I've only been at Red:4 for just over a month and the only people I've really gotten to know are Rob and M. Sullivan (over email). The office is kind of empty most days as everyone is being moved over to The Presidio space. I'll be moving over there in a few weeks too, I hear, but right now the only desk space is here, with the data team.

There are 7 of us total, and everyone seems to bring their lunch in, mostly keeping to themselves. I'm not terribly social at work, so, whatever. I'll make friends later. I'm kind of in a hurry today anyway as I want to take a look at that data!

TRANSFORMING DATA WITH CTES

Before I go home tonight, I need to refine the dates of the nadirs to actual timestamps. The split second when Cassini was the closest to Enceladus: the *nadir*.

The nadir itself should allow me to find the timestamp associated with it. This means I need to go back and query the table I just did the rollup on. There's another new tool I can use for this: a *Common Table Expression* (CTE).

One of the lightning talks at the Postgres Users Group I went to last week talked about how CTEs are one of the most underutilized features of Postgres, but I don't think they're confined to the platform. From what I read, they're part of the SQL standard, but all the other platforms (SQL Server, Oracle, MySQL) have less-than-exciting implementations of them.

A CTE allows you to chain queries together, passing the result of one to the next. Kind of like piping on the command line. A friend of mine said it was "functional SQL," which I suppose, sure whatever. Just don't start talking about functors and monads.

Here's what I need to do:

- I need to get all the low altitude data, grouped by year and week (which I just did)
- I need to get all the timestamps associated with that year, week and altitude
- I then have to pick one

I could create a couple of views or write some weirdly-joined select statements, but a CTE will keep things tidy:

```

with lows_by_week as (
    select date_part('year',time_stamp) as year,
           date_part('week',time_stamp) as week,
           min(altitude) as altitude
    from flyby_altitudes
    group by date_part('year',time_stamp), date_part('week',time_stamp);
), nadirs as(
    --?
)
select * from nadirs;

```

I can use the results of the first query as if they were a table named **lows_by_week** in the body of the second query.

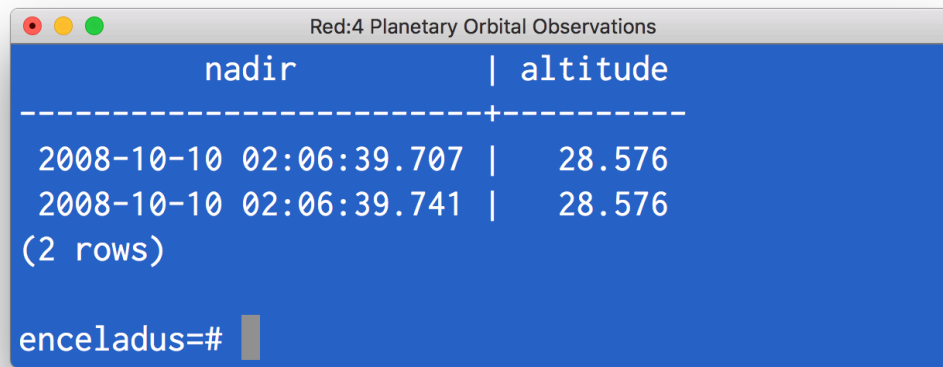
But what about the timestamp? How many of those will I get back? Here's a sample run using the dates from that low flyby in October of 2008:

```

select
    time_stamp as nadir,
    altitude
from flyby_altitudes
where flyby_altitudes.altitude=28.576
and date_part('year', time_stamp) = 2008
and date_part('week', time_stamp) = 41;

```

Kind of what I thought:



nadir	altitude
2008-10-10 02:06:39.707	28.576
2008-10-10 02:06:39.741	28.576

(2 rows)

enceladus=#

Cassini flies incredibly fast, and the INMS is snapping readings every 30ms or so. It's possible to have some timestamps returned with the exact elevation we're using as a filter. Another order of precision for altitude would be beautiful, but I don't have it so we'll have to figure something else out.

I could punt and use **limit 1** on this lookup query, that would return the first timestamp encountered. This wouldn't be very accurate as it's just the first one.

I could also use an aggregate function, grabbing the **min** or the **max** timestamp — but again, that's not very sciency, and I work in a sciency place. Accuracy is important, precision is a must. I can't just make a decision like this without having a reason, so I'll need to find a better solution.

We're working with a time window. The nadir of the October 2010 flyby is somewhere in the middle of those two timestamps, which is the best I can do to approximate the exact flyby. Unfortunately, this involves some math, which means M. Sullivan might be knocking on my door later.

I need to take the minimum timestamp and subtract it from the maximum. This will give me an interval. If I divide that in half and add it back to the minimum, I'll have a midway point.

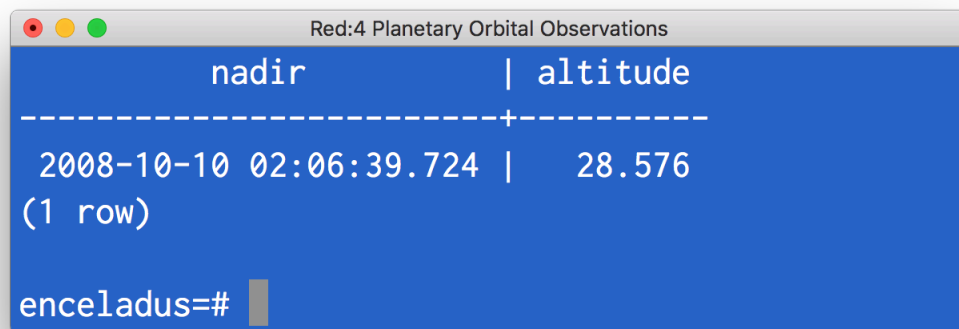
Let's do it:

```

select
  min(time_stamp) +
  (max(time_stamp) - min(time_stamp))/2
  as nadir,
  altitude
from flyby_altitudes
where flyby_altitudes.altitude=28.576
and date_part('year', time_stamp) = 2008
and date_part('week', time_stamp) = 41
group by altitude;

```

The results are musical:



nadir	altitude
2008-10-10 02:06:39.724	28.576

(1 row)

enceladus=#

Perfect. Right smack in the middle! This SQL is ripe for a function, and I'll try to get to that later.

Plugging this query into the CTE looks good, albeit a bit long and somewhat scary:

```

with lows_by_week as (
  select
    date_part('year',time_stamp) as year, date_part('week',time_stamp) as week,
    min(altitude) as altitude
  from flyby_altitudes
  group by date_part('year',time_stamp), date_part('week',time_stamp)
), nadirs as (
  select (
    min(time_stamp) + (max(time_stamp) - min(time_stamp))/2
  ) as nadir,
    lows_by_week.altitude
  from flyby_altitudes,lows_by_week
  where flyby_altitudes.altitude = lows_by_week.altitude
  and date_part('year', time_stamp) = lows_by_week.year
  and date_part('week', time_stamp) = lows_by_week.week
  group by lows_by_week.altitude
  order by nadir
)
select nadir at time zone 'UTC', altitude
from nadirs;

```

This is... errr... some intense SQL. New interns looking at this will probably want to hit eject, which is too bad. I won't say it's the prettiest stuff in the world, but it sure is powerful. Resolving 4 million rows of data into exact flyby timestamps:

Red:4 Planetary Orbital Observations		
nadir		altitude
-----+-----		
2005-02-17	11:30:12.119	1272.075
2005-03-09	17:08:03.4725	500.370
2005-07-15	02:55:22.33	168.012
2008-03-13	02:06:11.509	50.292
2008-08-12	04:06:18.574	53.353
2008-10-10	02:06:39.724	28.576
2008-11-01	00:14:51.429	173.044
2009-11-02	15:41:57.707	98.901
2009-11-21	10:09:56.371	1596.561
2010-04-28	07:00:01.088	3771.195
2010-05-18	13:04:40.301	437.292
2010-08-14	05:30:51.975	2555.180
2010-11-30	19:53:59.049	45.699
2010-12-21	09:08:27.146	48.324
2011-10-01	20:52:25.698	98.898
2011-10-19	16:22:11.2245	1230.674
2011-11-06	12:58:53.4805	496.603
2012-03-28	01:30:08.975	74.165
2012-04-14	21:01:37.811	74.100
2012-05-02	16:31:28.949	73.134
2015-10-14	17:41:28.9765	1844.230
2015-10-28	22:22:41.55	49.010
2015-12-20	01:49:16.1135	5000.200
(23 rows)		
enceladus=# █		

I did it. LOOK AT THAT I DID IT! BEHOLD MY DATABASE POW-

From: M. Sullivan sullz@redfour.io
Subject: Go. Home.
To: Dee Yan yand@redfour.io
Date: October 27, 2017

You have made outstanding progress today M. Yan. It's now time for you to go home, however, per M. CEO's wishes. I'll have some thoughts for you in the morning, but I have to say I'm quite impressed by the work you've done today.

Best, S

SOMETHING GNAWING AT ME

October 27, 2017, 1722

Fine. I feel like I'm back home again and my mom is telling me it's time to get to bed. Well, I guess I never really left home... but at least mom doesn't tell me when it's time to go to bed anymore. Wait... does she?

That blob of SQL is gnawing at me. CTEs are so powerful but can quickly turn into a mess, and to be honest, at the time I wrote that mess I didn't care. Results are what's important now, not prettiness!

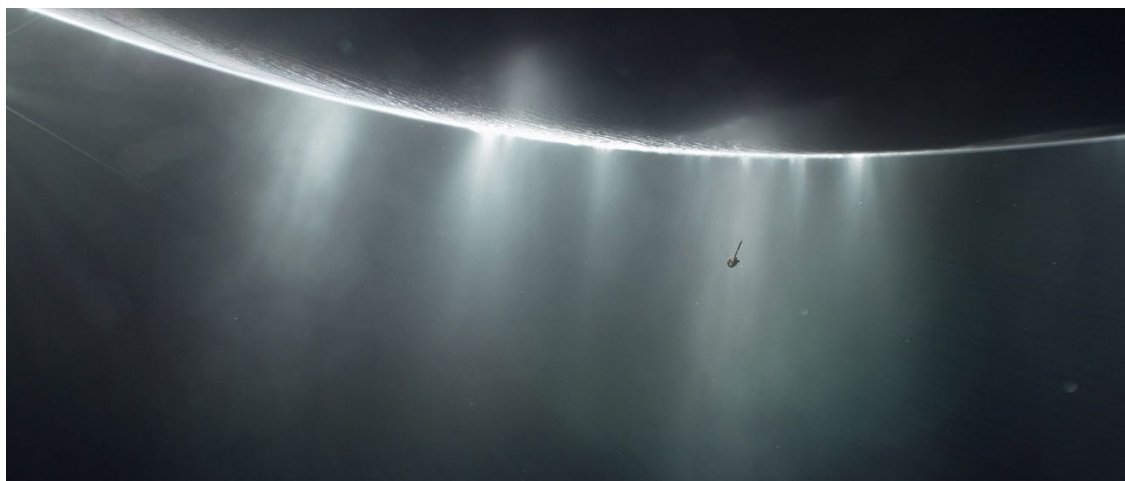
... said every hack programmer, ever. I need to be able to generate beautiful, efficient, concise SQL and deliver results. There's no way I'm going to get this job if I can't do both.

Maybe I should have used Python. It's a higher-order language, and there's no shame in using what I know best. At the same time, knowing SQL better can reduce unneeded code dramatically. It's just a question of how much you're willing to learn.

I'll read more on this bus ride home. First, I want to learn more about Enceladus.

Patterns in The Data

Each of Cassini's first six flybys gets closer and closer until, on October 10, 2010, they got just a little too close. You can tell the Cassini team were trying to find something out, using the INMS and CDA to sniff the chemistry of the little ice moon as the spacecraft flew by at 19,000 miles per hour under its south pole, right through the plumes. That's fast.



Artist's rendering of Cassini flying through the Enceladus plumes. Credit NASA/JPL

Jets of Water

The first two flybys of Enceladus confirmed there was activity coming from the south pole of the small moon, and that activity was providing some kind of atmosphere.

What, exactly, was in that atmosphere?

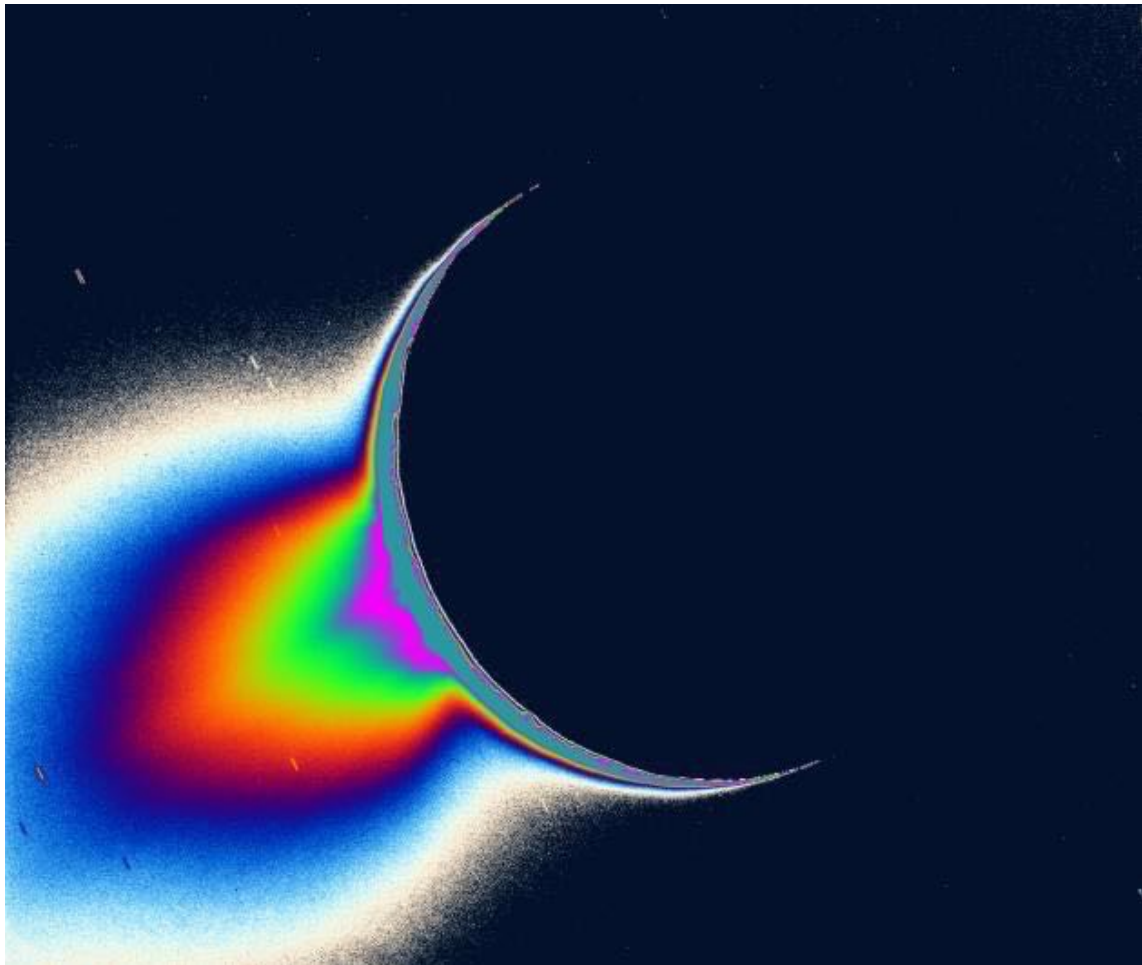


Image Credit: NASA/JPL

To understand that, on July 15, 2005 (UTC) Cassini flew 168.012 kilometers above the surface, snapping pictures, running thermal scans and “sniffing” everything it could.

The previous flybys suggested that there was thermal activity at the south pole, so an imaging scan was run which showed some curious fractures:

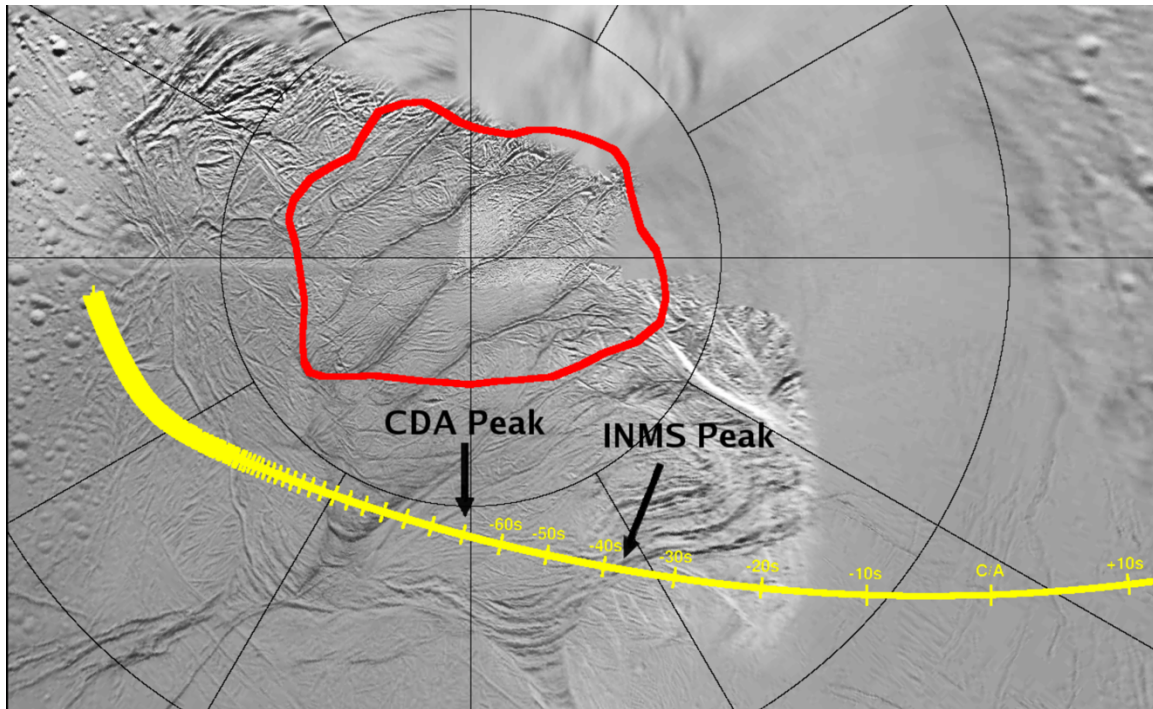


Photo Credit: NASA/JPL

As you can see, plotted here, the INMS readings jumped over the region outlined in red, which was registering as much warmer than the rest of the moon.

These cracks became known as the “Tiger Stripes” and, in subsequent flybys, were visually identified as the source of the plumes being jetted into space.

The INMS readings were even more incredible. In March of 2008, during a direct sampling flyby of the plumes, the INMS detected water and organic compounds. Scientists were stunned. Thermal imaging suggested a hotspot under the ice that was many times warmer than expected, or even considered possible.

Something was going on under the ice of that moon...



Photo Credit: NASA/JPL

RING DUST

From: Rob Conery rob@redfour.io
Subject: The Analysis Window
To: Dee Yan yand@redfour.io
Date: October 30, 2017

Hope you enjoyed your weekend. Got your update and yes, those flyby times look great. Nice touch with the median timestamp — I agree that pinpointing things at those speeds is critical.

Speaking of speeds, that's what I need you to calculate next. The Analysis Team needs to dive into the INMS and CDA results, but they need to know what to focus on, precisely.

It's all about the plumes, Dee. They're really dense right at the south pole of Enceladus, and if Cassini flew too close, it would saturate the INMS and the CDA (Cosmic Dust Analyzer), which would flip over to the HRD (High Rate Detector, part of the CDA, more on that in a second). In short: the readings right at nadir are more prone to error due to saturation than the readings just outside the nadir, on either side. That's our analysis window. By using the readings outside of saturation and within, our Analysis Team can come up with compensation and offsets. The only way they can do that is if they know how fast, exactly, Cassini was traveling.

That's what I need you to figure out next: Cassini's speed.

I know that there are velocity measurements in the CDA data. You'll find the raw dump on [our S3 bucket](#). Go through it, have a look at the manifest like you did before, and do the same import. This time, however, I'd like a Makefile so we can easily follow your process.

Before that, I need you to create a **flybys** table with the timestamp and altitude information and include speed as well as a **start_time** and **end_time** which will define the analysis window. Also – yes, I like your idea of naming; I agree that the first flyby should be E-0. Go ahead and add that too.

I attached some informal notes I threw together on the basic function of the CDA. It's a little on the rough side, but hopefully should get you started figuring out the speed stuff.

R

BASIC FUNCTION OF THE CDA

Rob Conery, CTO

The spacecraft we called “Cassini” was actually two vessels in one: the Cassini deep-space probe (with 12 mounted instruments) and the Huygens lander, which was destined for a one-way trip to Titan.

The spacecraft was built in collaboration with the European Space Agency (ESA), which produced the Huygens lander, and the Italian Space Agency (ISA, and yes, they have one), which made the remote antennae and parts of the radar system.

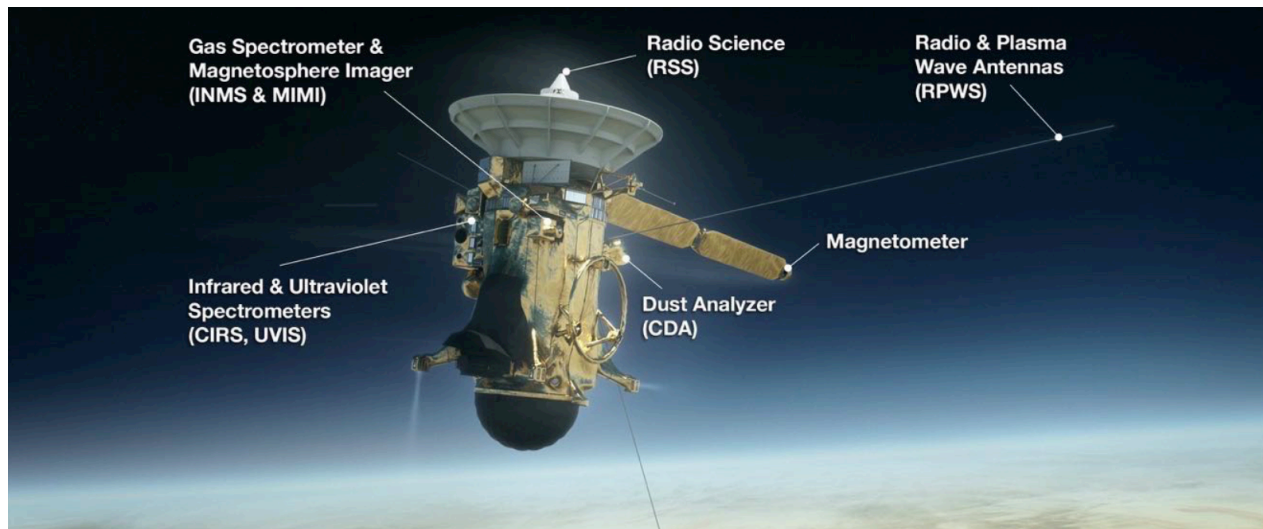


Photo credit: NASA/JPL

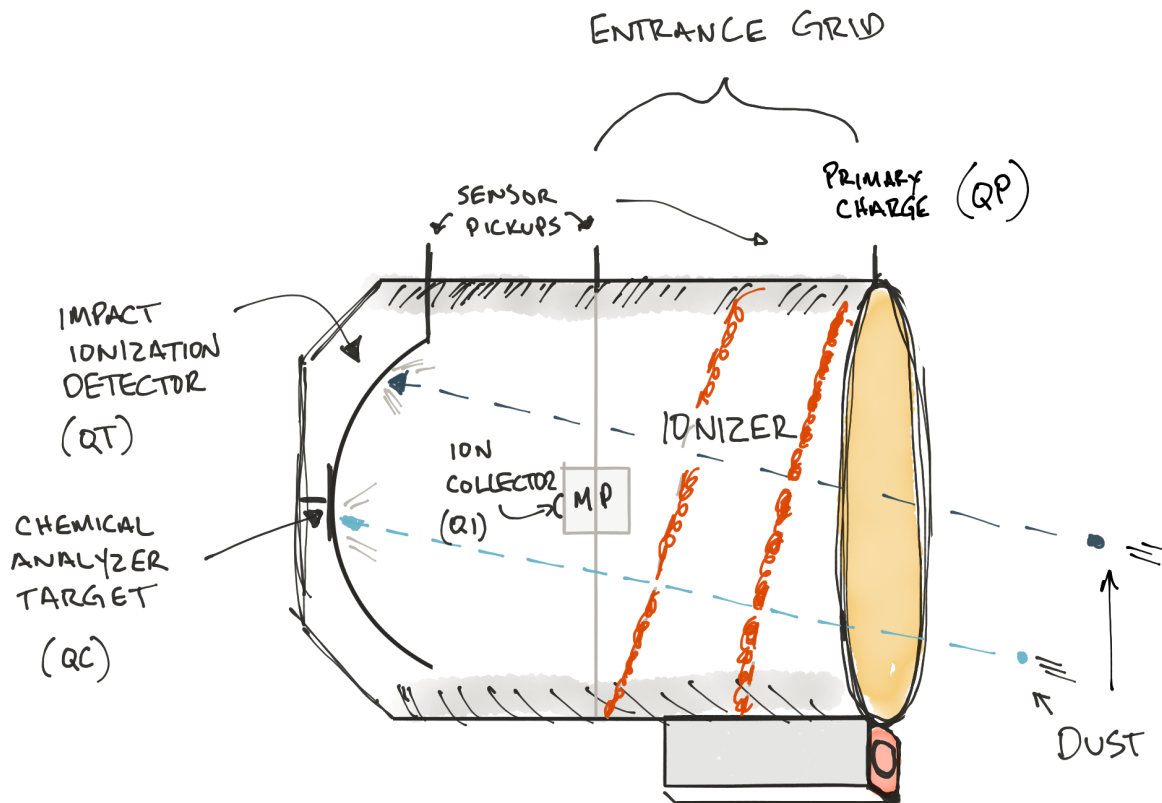
The device I'm going to discuss now is the one with the coolest name: The Cosmic Dust Analyzer (CDA).

The CDA is actually two instruments in one: The Dust Analyzer (DA) and The High Rate Detector (HRD). The DA is designed for bigger, faster dust particles. The HRD is designed to analyze a lot of dust at once.

The Dust Analyzer is the thing we're most interested in. It can evaluate dust particles based on physical characteristics as well as chemical.

A bit of dust will enter the drum and become ionized as it moves through the entrance grid (called a dust ram). At that point, any charge on the particle is measured (QP).

After that, the particle impacts a hemispherical surface in the back of the drum called the Impact Ionization Detector, or IID, which then analyzes the plasma given off by the impact, as well as any smaller bits, left over. This is a mass spectrometer, but instead of shooting ions through an electrical field and looking at the deviation, they are blown up and the resultant material (mostly plasma) analyzed.



I'm reminded of a giant bug zapper that we used to have on my patio growing up. At night, an ultraviolet light would attract bugs, and you could always tell a mosquito from a fly based on the electricity applied to each impact. Mosquitos were quick, satisfying sounds like "zzt!", and flies took a bit longer with some crackling in between.

Occasionally you'd get a yellow jacket that would smoke a bit. Moths were the worst as they would catch on fire and my mom would yell at my dad to "do something Patrick, that's unbearable!".

Anyway, each bug hitting the zapper would have a characteristic zapping sound and fry profile. Likewise, each particle that enters the CDA and gets ionized (gaining a specific charge) has a particular signature of plasma.

That's the way I see it, at least. A multi-million-dollar Cosmic Bug Zapper.

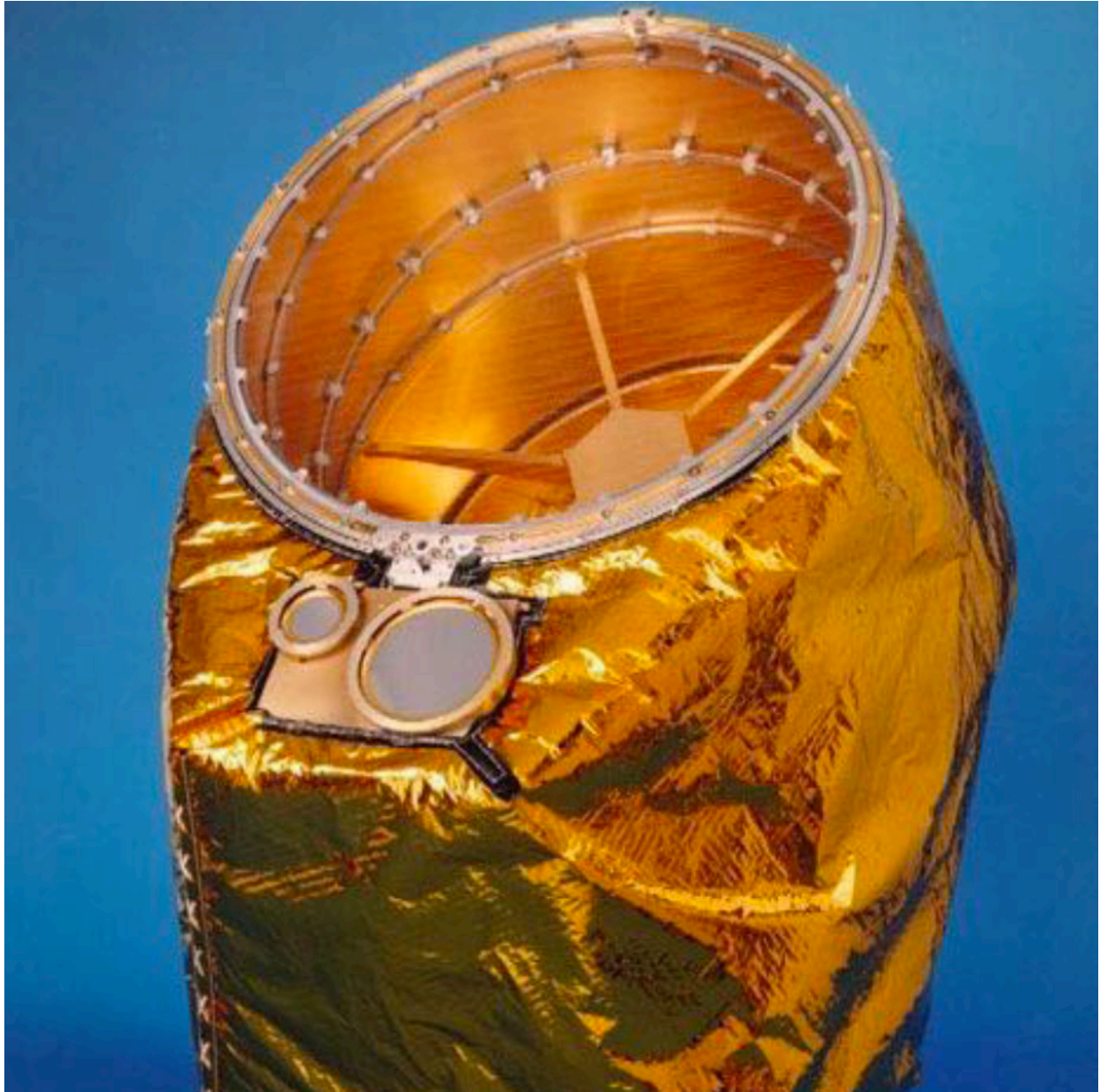


Image credit: NASA/JPL

Scientists are still poring over the readings from the CDA to this very day. Recently (in November 2017) scientists used data from the CDA (as well as from the INMS) to resolve a mystery surrounding the rapid cooling of one of Titan's polar hotspots. That data had always been there, it just took this long to associate it correctly with other data received to resolve a mystery.

REFACTOR PART 1: A BETTER CTE

October 30, 2017, 0812

A Cosmic Bug Zapper. I like it. Looks like I get to play with more shell scripts... later.

Right now, I need to create the flybys table Rob asked for. While I'm at it, I'm going to refactor that CTE a bit. I don't like the mess.

There's a lot of repetition in the nadir calculation, specifically concerning calculating the **year** and the **week**. The most natural thing to do would be to redo the **flyby_altitudes** materialized view, so the `year` and **week** were already calculated:

```
drop materialized view if exists flyby_altitudes;
create materialized view flyby_altitudes as
select
  (sclk::timestamp) as time_stamp,
  date_part('year', (sclk::timestamp)) as year,
  date_part('week', (sclk::timestamp)) as week,
  alt_t::numeric(10,3) as altitude
from import.inms
where target='ENCELADUS'
and alt_t IS NOT NULL;
```

The materialized view is rebuilt with month and year information, which means I can simplify my CTE:

```

with lows_by_week as (
  select year, week,
  min(altitude) as altitude
  from flyby_altitudes
  group by year, week
), nadirs as (
  select (
    min(time_stamp) + (max(time_stamp) - min(time_stamp))/2) as nadir,
    lows_by_week.altitude
  from flyby_altitudes, lows_by_week
  where flyby_altitudes.altitude=lows_by_week.altitude
  and flyby_altitudes.year = lows_by_week.year
  and flyby_altitudes.week = lows_by_week.week
  group by lows_by_week.altitude
  order by nadir
)
select nadir, altitude
from nadirs;

```

Much improved... but I think I can do better. Calculations in SQL seem like a “smell” to me; at least that’s what I’ve read. They can almost always be simplified into a function. I was thinking about that on the bus last night, so I decided to read up on functions as much as I could.

Overall, they’re pretty straightforward: just regular programming “stuff” but for a database, and using SQL. You can also use other languages if you need to.

For instance, if I wanted to use variables, loops, and other control-flow structures I could use **PLPGSQL**, which is kind of funky, but it looks... I don’t know... like a programming language, I guess.

I could also use Python with the PL/Python. If I wanted to piss off M. Sullivan, I could also use JavaScript (evil cackle). Google’s V8 JavaScript engine can be installed

using an extension, and the language is called **plv8** in Postgres. Yuck! But also, kind of interesting in a punching myself in the face kind of way.

I just need regular old SQL, and I want to keep it simple:

```
drop function if exists low_time(numeric, double precision, double precision);
create function low_time(
  alt numeric,
  yr double precision,
  wk double precision,
  out timestamp without time zone
)
as $$
  select
    min(time_stamp)
    + ((max(time_stamp) - min(time_stamp)) / 2)
    as nadir
  from flyby_altitudes
  where flyby_altitudes.altitude=alt
  and flyby_altitudes.year = yr
  and flyby_altitudes.week = wk
$$ language sql;
```

The drop statement isn't mandatory, I'm just using it here so I can make changes quickly. The syntax is simple: create a function with these arguments. Here's the body, delimited by **\$\$**. Whatever is returned from the query gets popped into the output variable, and we're done.

Honestly, this seems like straight up programming to me, and I don't know why app developers get so upset over their use.

Anyway, it makes my CTE look so much better, doesn't it?

```

with lows_by_week as (
  select year, week,
  min(altitude) as altitude
  from flyby_altitudes
  group by year, week
), nadirs as (
  select low_time(altitude,year,week) as time_stamp,
  altitude
  from lows_by_week
)
select * from nadirs;

```

Now I can create a **flybys** table out of the results.

THE FLYBYS TABLE

Rob asked that I push the flyby results from my CTE into a new table and that I should add **name**, **start_time**, and **end_time** .

```

-- convenience for redoing
drop table if exists flybys;

--redone CTE
with lows_by_week as (
    select year, week,
    min(altitude) as altitude
    from flyby_altitudes
    group by year, week
), nadirs as (
    select low_time(altitude,year,week) as time_stamp,
    altitude
    from lows_by_week
)
-- exec the CTE, pushing results into flybys
select nadirs.*,
    -- set initial vals to null
    null::varchar as name,
    null::timestampz as start_time,
    null::timestampz as end_time
-- push to a new table
into flybys
from nadirs;

```

I'll also add a primary key, to save M. Sullivan from heart attack:

```
-- add a primary key
alter table flybys
add column id serial primary key;

-- using the key, create
-- the name using the new id
-- || concatenates strings
-- and also coerces to string
update flybys
set name='E-' || id-1;
```

I added some comments for myself, so I know what's going on later. That's the whole point of the refactor. I know that in a year or more, I might need to revisit this code and understand what I was trying to do. That's how I think of comments anyway.

Also: casting **null** to a type? I guessed at that one. Had no idea if it would work, and guess what? It did. RAWWRR!

Short entry for the day, I suppose. All of this refactoring and function stuff took much longer than I thought.

Off to 21st Amendment for a beer with the team. Maybe I'll make some friends. Wonder if M. Sullivan will be there?

From: M. Sullivan sullz@redfour.io
Subject: Nice function.
To: Dee Yan yand@redfour.io
Date: October 30, 2017

Nice work on the function M. Yan. I like the midpoint calculation. It does make things much more precise, and I believe the JPL team will appreciate that we're keeping their data in mind. Glad you didn't need much more persuasion.

You have some repeated values in there, **min(time_stamp)** to be precise, and I might have liked to see you use **PLPGSQL** so you could use a variable or two to avoid repetition. That said, I probably would have told you it was overkill, and you should have been able to do it in one line. Which you have. Well done.

That CTE looks much cleaner, too. This makes me happy, which should make you happy.

I want to caution you once again about using **timestampz**. There is no time zone information with this data so you might want to investigate using a simple timestamp without a time zone. We care about UTC only, so there's no sense in adding a time zone.

Best, S

October 30, 2017, 2249

I have a feeling that working with dates and times takes years to get right if you ever do. Once again: M. Sullivan makes perfect sense. I'll make sure to use a regular timestamp without time zone for the CDA data, which I'll play with tomorrow.

I also got to spend a few hours reading about Cassini and Enceladus tonight; it's hard to not get pulled in by all the crazy mysteries!

Every time I think I have a handle on all the discoveries made, and when they were made, I read something new, and my mind gets blown once again.

For instance: tonight [I read an article](#) that stated that Enceladus's plumes not only created Saturn's E-Ring, but also a band of water vapor (clouds) encircling Saturn's

north pole! This little moon actually sends water to Saturn! That's kind of crazy, isn't it? It's the only moon that we know of to ever directly affect its parent planet's atmosphere.

Not only that, but [they also found](#) that Enceladus and Saturn are connected electrically, that they form an electrical circuit through magnetic interactions which, frankly, went right over my head.

What the hell is happening up there? This moon is a freak show.

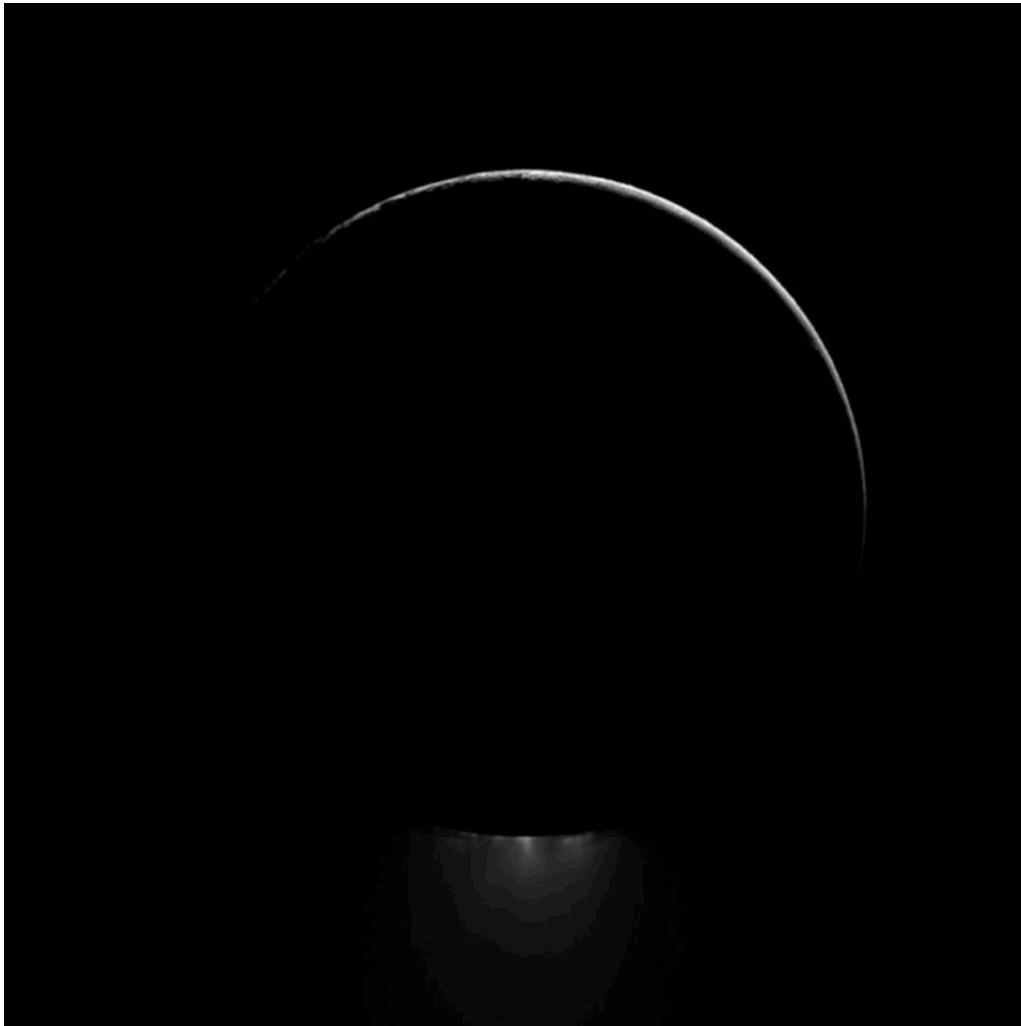


Photo credit: NASA/JPL

CDA DATA AUDIT

October 31, 2017, 0822

I decided to splurge and get a double latte today. I also made it on the 15 Express finally. That thing always leaves just before I get to the dang stop!

I read over the description of the CDA data and had a look at what's sitting in our [S3 bucket](#). Looks like [a jumble of TAB files](#), fixed-width text in 80-byte rows that are just a bit beyond me and my skills. There's a note there from someone, I'm assuming it's Rob.

Hello Fellow Red:4 Space Analysts!

There is a single file in here that you need, called **cda.csv**. This is the real thing, straight from orbit around Saturn. [There's more where that came from](#), but this file is just what we need for our Enceladus study.

The data that comes from the PDS is in the form of fixed-width text files with an 80-byte row length that I've run a small bit of transformation on because I don't want that to slow anyone down. Specifically, I:

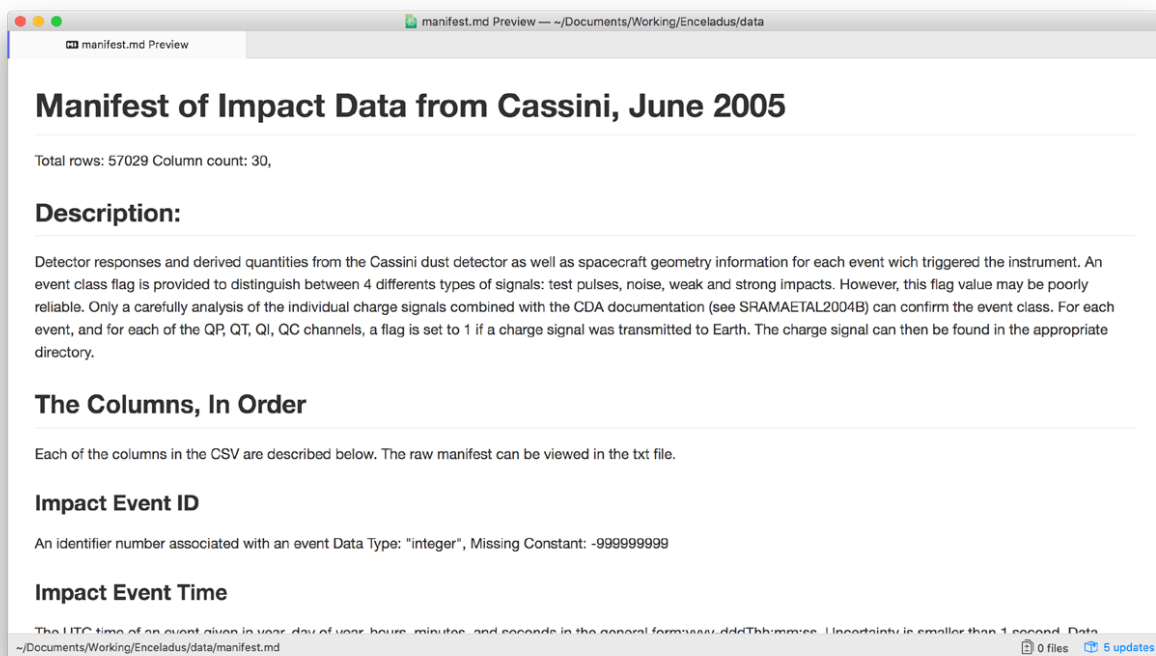
- Unzipped 20 or so massive TAR files
- Concatenated the TAB files using `cat`
- Opened the enormous file with Excel
- Executed "Text to Columns" as every column was tab-delimited
- Added the column names at the top
- Exported it as a CSV

I figured you might like it if I handed you a CSV file as opposed to a TAB.

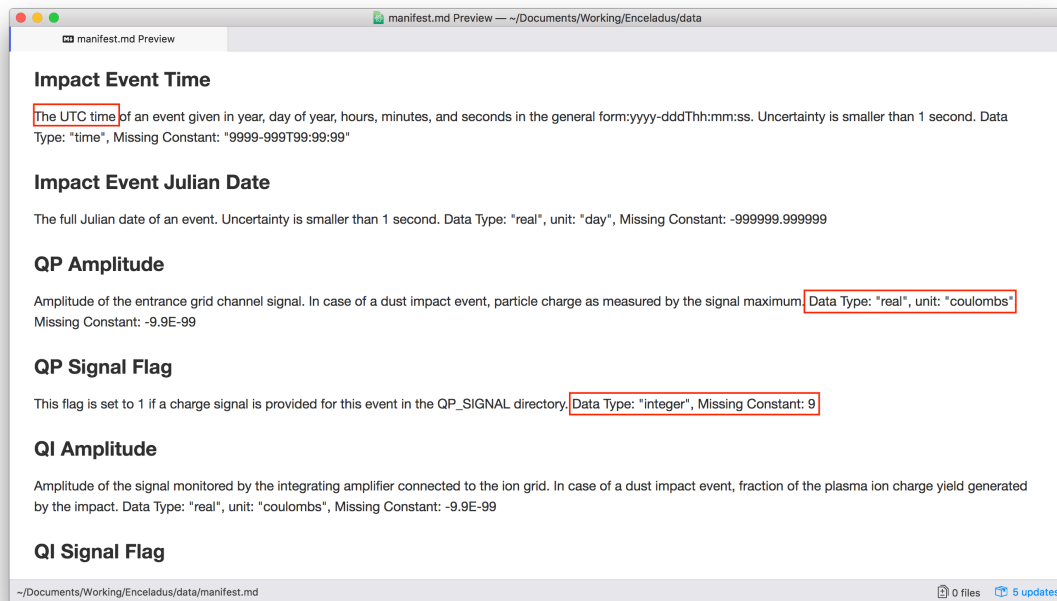
In addition to the data file, there is also a manifest file (**cda_manifest.txt** and **cda_manifest.md**) that describes the data we're about to play with. This file is critical to us because it:

1. Describes the data
2. Designates its type, precision, and length
3. Gives a default value
4. Provides units of measurement We'll use all of this for our audit.

I've summarized the impact data manifest as a markdown file for ease of viewing; that text file was starting to bug me. Here's a preview of it using Atom:



Take a second and open it up. There's some essential stuff in here, some which I've outlined for you:



We care a lot about time in these calculations because we're going to be calculating intervals down to the second as part of our density analysis later on; so, knowing that we can use UTC here is great.

Next, we're given the data types as well as this thing called "Missing Constant." The two of those together show the length and precision we'll need for each column. It is specified in the raw manifest (the "format" specification), but I only want to trust the data I can see.

What is a "Missing Constant"? I suspect that it's merely a placeholder for data that wasn't collected for a given impact event. We'll have to verify that.

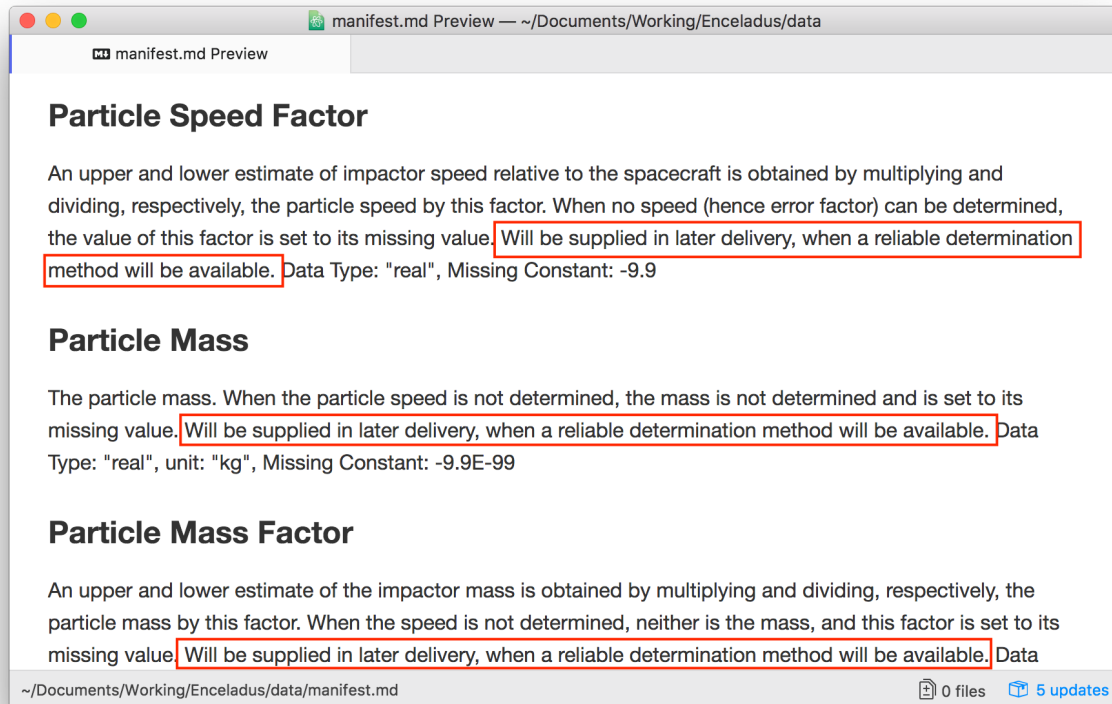
Finally, data is meaningless unless it has some kind of unit. We'll want to remember what these units are, perhaps using them in our column names.

Look over this manifest, see if anything makes you suspicious.

We're bound to find some issues as we import this data in, it's the nature of the beast. Incorrect/inaccurate dates, numbers rounded up from the required precision, and so on.

Case in point: this phrase, which is repeated quite a few times

Will be supplied in later delivery, when a reliable determination method will be available



Placeholder data, not good. Since we don't have any corrections for it, it's meaningless, so I've omitted it from the import.

Here are all the fields that have that designation:

- Target Flag
- Event Quality
- Particle Speed
- Particle Speed Factor
- Particle Mass

- Particle Mass Factor
- Particle Charge
- Particle Charge Error

I have already deleted these columns from the **cda.csv** file so you can download that for whatever you need to do.

R

IMPORTING CDA DATA

October 31, 2017, 0849

That makes life a bit easier. I need to get at that data, and I don't want to spend a bunch of time doing the ETL dance.

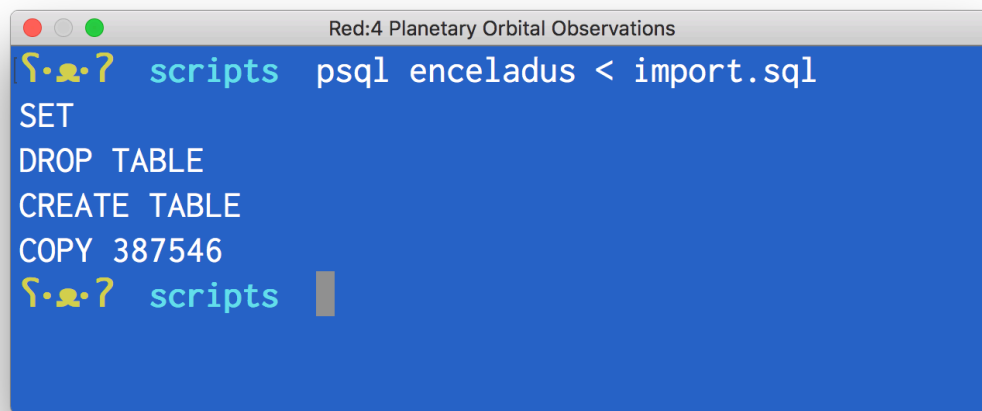
I'm just going to make it a simple **import.sql** file as there are only a few steps and I think a Makefile is probably overkill. I'll find out, I suppose, when this is done.

I'll start by creating a table for the CDA import data and then importing it in:

```
drop table if exists import.cda;
create table import.cda(
  event_id text,
  impact_event_time text,
  impact_event_julian_date text,
  qp_amplitude text,
  qi_amplitude text,
  qt_amplitude text,
  qc_amplitude text,
  spacecraft_sun_distance text,
  spacecraft_saturn_distance text,
  spacecraft_x_velocity text,
  spacecraft_y_velocity text,
  spacecraft_z_velocity text,
  counter_number text,
  particle_mass text,
  particle_charge text
);

COPY import.cda
FROM '[Path to data]/cda.csv' DELIMITER ',' HEADER CSV;
```

Boom:

A terminal window titled "Red:4 Planetary Orbital Observations" with a blue background. The prompt is a yellow icon followed by "scripts". The commands entered are: "psql enceladus < import.sql", "SET", "DROP TABLE", "CREATE TABLE", and "COPY 387546". The prompt "scripts" is shown again at the bottom with a cursor.

```
Red:4 Planetary Orbital Observations
scripts psql enceladus < import.sql
SET
DROP TABLE
CREATE TABLE
COPY 387546
scripts
```

It's so cool how fast that goes! Took barely a second on my Mac to import 387,000 records.

Now I need to pull data out of here, making sure it's typed properly. I'm thinking that, right now, I just need the velocity fields. However, I might want the other stuff later on.

The manifest has all the type information, so I'll use that to run a quick type check on the imported data:

```

drop schema if exists cda cascade;
create schema cda;
select
  event_id::integer as id,
  impact_event_time::timestamp as time_stamp,
  impact_event_time::date as impact_date,
  counter_number::integer,
  spacecraft_sun_distance::numeric(6,4) as sun_distance_au,
  spacecraft_saturn_distance::numeric(8,2) as saturn_distance_rads,
  spacecraft_x_velocity::numeric(6,2) as x_velocity,
  spacecraft_y_velocity::numeric(6,2) as y_velocity,
  spacecraft_z_velocity::numeric(6,2) as z_velocity,
  particle_charge::numeric(4,1),
  particle_mass::numeric(4,1)
from import.cda
order by impact_event_time::timestamp;

```

If I cast the **impact_event_time** to a simple **timestamp** then I believe we can avoid the confusion that comes with time zone adjustments made by Postgres. Since I'm dealing with dates and times down to the millisecond, I think this is really important.

Uh oh...


```
Red:4 Planetary Orbital Observations
enceladus-# counter_number::integer,
enceladus-# spacecraft_sun_distance::numeric(6,4) as sun_distance_au,
enceladus-# spacecraft_saturn_distance::numeric(8,2) as saturn_distance_rads,
enceladus-# spacecraft_x_velocity::numeric(6,2) as x_velocity,
enceladus-# spacecraft_y_velocity::numeric(6,2) as y_velocity,
enceladus-# spacecraft_z_velocity::numeric(6,2) as z_velocity,
enceladus-# particle_charge::numeric(4,1),
enceladus-# particle_mass::numeric(4,1)
enceladus-# from import.cda
enceladus-# order by impact_event_time::timestamptz:
ERROR: invalid input syntax for integer: "**"
enceladus=#
```

The old “let’s put random text in a number field to represent ... something” trick.

This is bad for a few reasons:

1. It suggests that this was, at one time, a spreadsheet on someone’s desk. They took what a machine created and tweaked it, generating trash data.
2. The CDA is trying to represent something that is not in the manifest.
3. Rob is having some fun with me.

I don’t think it’s the latter, though I wouldn’t put it past him. He’s kind of stressed out right now so slowing me down would be counterproductive... but then again maybe he has a warped sense of humor?

If this data did, in fact, get pulled into Excel that means we might not be able to trust it. I’m starting to side with M. Sullivan on that one. Excel is a tool of the devil.

If this did come from a fixed-length file, that suggests that is a dump from a machine. I’ll have to ask Rob when I get a chance.

For now, I’ll just fix this with a case statement. If I see a ** value, I’ll just consider it null:

```

drop schema if exists cda cascade;
create schema cda;
select
  event_id::integer as id,
  impact_event_time::timestampz as time_stamp,
  impact_event_time::date as impact_date,
  case counter_number
    when '**' then null
    else counter_number::integer
  end as counter,
  spacecraft_sun_distance::numeric(6,4) as sun_distance_au,
  spacecraft_saturn_distance::numeric(8,2) as saturn_distance_rads,
  spacecraft_x_velocity::numeric(6,2) as x_velocity,
  spacecraft_y_velocity::numeric(6,2) as y_velocity,
  spacecraft_z_velocity::numeric(6,2) as z_velocity,
  particle_charge::numeric(4,1),
  particle_mass::numeric(4,1)
into cda.impacts
from import.cda
order by impact_event_time::timestampz;

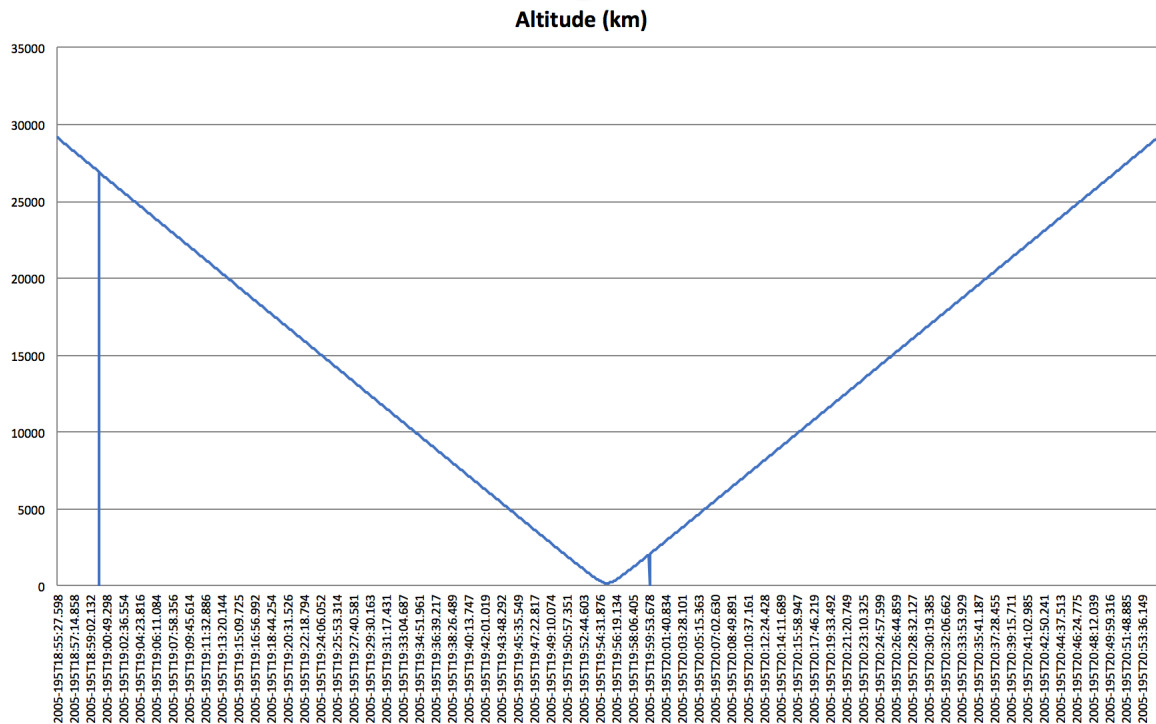
alter table cda.impacts
add id serial primary key;

```

Yay! No more type errors and I now have a table with some velocity data that I can pull together.

EXAMINING CDA DATA

I wanted to see what a flyby “looks like” regarding altitude, speed and so forth, so I decided to graph it. I outputted all the altitudes from the **flyby_altitudes** view for the July 14, 2005, flyby and charted them in Excel (times on x, altitude on y):



As opposed to a swooping U shape, what we have here is a clear V with some possible slowing right at the nadir due to the friction of the plume. I like this. It gives a nice feel to the data and to what happened back then. Cassini blew through the analysis window like a high-speed express train flying through a country station.

Right there, at the very bottom of my chart, Cassini was showered with icy moon “stuff.”



What, exactly, was in that stuff is something that the CDA and the INMS can tell me. First, I have to consider a small problem.

The INMS sensors work by examining particles within a period (I like to think of it as a “time box”) of 30 to 32ms. The machine is calibrated to expected amounts of matter which, in space, isn’t all that much. When it flies through a plume of icy bits, however, the sensors can become saturated, which means that countermeasures and calibrations need to be performed.

The INMS handles all of this, or at least it’s supposed to. This is one of the reasons altitude is tracked, as well as exact ship position and velocity. The reason I say this is a problem is that it involves human interference, which will always taint the data.

This is why the Analysis Team wants a precise window: for comparison. Some particles will escape the Enceladus gravity well, which Cassini will pick up during approach. Other’s will gently fall back to the surface.

This is where things become interesting. These particles are not uniform bits of ice and chemical. Some contain mineralogical material. Those particles will have more mass and, therefore, will be heavier. Some might even contain non-mineralogical material...

Focus, Dee.

Right. I want to see the data without that annoying “Missing Constant”:

```
select * from cda.impacts where x_velocity <> -99.99 limit 5;
```

saturn_distance	x_velocity	y_velocity	z_velocity
21.95	-12.31	-8.22	-1.08
21.96	-12.31	-8.22	-1.08
22.02	-12.29	-8.21	-1.09
22.02	-12.29	-8.21	-1.09
22.03	-12.29	-8.21	-1.09

Looks like there's data in here I can use. Not everything is missing, which is helpful. Even if I can figure out Cassini's speed for a given day, that will help.

CALCULATING CASSINI'S SPEED

Cassini had a thrust system on board, but it didn't use it for acceleration/deceleration between Kronian bodies, it used [gravity assist from Titan](#):

Cassini's combined orbits around Saturn look like a ball of yarn exploded without the threads breaking. The loops reach in all directions and are long in places but short in others. Though they appear disorderly, the sum of Cassini's orbits represents a carefully choreographed, decade-long dance between Cassini and Titan...

"Titan is the engine of this tour," said Duane Roth, chief of Cassini's navigation team. After Cassini arrived at Saturn in 2004, Roth and his team of astrodynamacists at NASA's Jet Propulsion Laboratory used Titan to send the Cassini spacecraft swinging up, down and around the Saturn system.

Now THAT is a title: Astrodynamicist. Your job, all day, every day: being dynamic, astronomically. That's what I would tell my friends anyway.

You might think that by having a crew of astrodynamicists on the job that Cassini's precise location in space would be known at any given time. Not really, [according to JPL](#):

Perhaps more surprising, the exact position of Cassini itself was something of a permanent mystery. The navigation team estimated Cassini's location by sending radio signals to the spacecraft from the powerful antennas of NASA's Deep Space Network. When Cassini received the signal more than an hour later, it immediately sent it back to Earth. Because the signal traveled at a fixed speed – the speed of light – the team was able to calculate Cassini's distance by knowing precisely when the signal was sent from Earth and received from the spacecraft. And by measuring the shift in the frequency Cassini sent back (due to the Doppler effect), the navigation team calculated Cassini's speed toward or away from Earth.

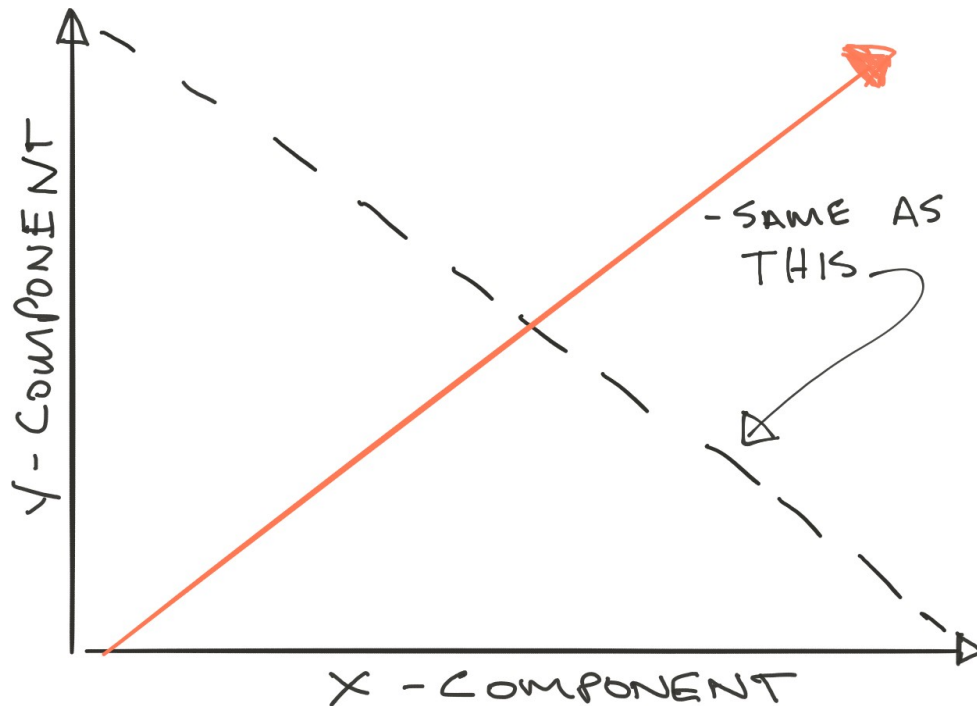
Given that the team didn't ever really know where Cassini was (precisely), every flyby was a bit of a small crapshoot. This process improved over time, but it was kind of "hit and miss" when they [first got there](#):

"We missed our first flyby of Titan by a distance on the order of 20 kilometers [12 miles]." Since then, Cassini's navigation team used each flyby to improve the accuracy of their knowledge of the moon's orbit.

Honestly, that's pretty damn good if you ask me. However, that margin of error covers the analysis window I'm trying to figure out.

There was a gyroscope on board that was calibrated with heliocentric coordinates, so instruments like the CDA could roughly calculate its position relative to the sun.

I can use this data to calculate Cassini's speed relative to the sun by using the x, y and z components stored with the CDA data (in `cda.impact`s). Vectors are computed using trigonometry and geometry, and what I'm trying to do amounts to finding the long side of a right triangle:



If I can figure the length of the long side (the hypotenuse, Dee, can't believe you had to Google it), then I know the length of its complement, which is the vector I'm looking for.

This is 2-dimensional, but the same concept applies to 3-dimensional shapes, which I guess would be a pyramid. Anyway: it's the Pythagorean theorem that I need to apply here (square root of x squared + y squared + z squared – good Googling Dee):

```
select time_stamp,  
       x_velocity,  
       y_velocity,  
       z_velocity,  
       sqrt(  
         (x_velocity * x_velocity) +  
         (y_velocity * y_velocity) +  
         (z_velocity * z_velocity)  
       )::numeric(10,2) as v_kms  
from cda.impacts  
where x_velocity <> -99.99
```

That worked great:

time_stamp	x_velocity	y_velocity	z_velocity	v_kms
2005-04-04 18:12:57.6-07	-8.80	-5.95	-1.46	10.72
2005-04-04 20:32:38.4-07	-8.75	-5.90	-1.46	10.65
2005-04-06 19:30:43.2-07	-7.91	-5.00	-1.54	9.48
2005-04-08 07:48:00-07	-7.32	-4.23	-1.59	8.60
2005-04-08 23:38:24-07	-7.07	-3.85	-1.61	8.21
2005-04-09 12:47:31.2-07	-6.85	-3.51	-1.62	7.87
2005-04-09 19:50:52.8-07	-6.74	-3.31	-1.64	7.69
2005-04-10 03:38:52.8-07	-6.61	-3.08	-1.64	7.47
2005-04-10 03:54:43.2-07	-6.60	-3.07	-1.64	7.46
2005-04-10 14:47:02.4-07	-6.42	-2.71	-1.65	7.16
2005-04-10 15:00:00-07	-6.41	-2.70	-1.66	7.15
2005-04-10 21:20:09.6-07	-6.30	-2.47	-1.66	6.97
2005-04-10 22:58:04.8-07	-6.28	-2.41	-1.66	6.93
2005-04-11 01:13:26.4-07	-6.24	-2.32	-1.66	6.86
2005-04-11 02:47:02.4-07	-6.21	-2.25	-1.66	6.81
2005-04-11 03:56:09.6-07	-6.19	-2.21	-1.67	6.78
2005-04-11 07:14:52.8-07	-6.13	-2.06	-1.67	6.68
2005-04-11 07:19:12-07	-6.13	-2.06	-1.67	6.68
2005-04-11 07:46:33.6-07	-6.12	-2.04	-1.67	6.66
2005-04-11 08:25:26.4-07	-6.11	-2.01	-1.67	6.65
2005-04-11 08:26:52.8-07	-6.11	-2.01	-1.67	6.65
2005-04-11 09:18:43.2-07	-6.09	-1.97	-1.67	6.61
2005-04-11 10:48:00-07	-6.06	-1.90	-1.67	6.57
2005-04-11 11:16:48-07	-6.05	-1.88	-1.67	6.55
2005-04-11 12:04:19.2-07	-6.04	-1.84	-1.67	6.53

I don't know how useful it is, however. The manifest describes the X, Y and Z velocities in this way:

The J2000 heliocentric equatorial X component of the Cassini velocity vector.

Which means that the speed I'm looking at in this calculation is relative to the sun.

Speed is meaningless unless you have a frame of reference. It's not enough to ask "how fast did Cassini travel," you have to specify "relative to Enceladus," or "relative to Saturn."

From our perspective, here on earth, relative to Earth is interesting, but both Cassini and Enceladus could be moving away from Earth at the same time, which would mean that, from our perspective, Cassini is hauling ass.

The only reasonable way to calculate orbital speeds, trajectories and the like is to do it from the frame of reference of the entire solar system. If you were an astrodynamacist, stringing together equations so you could plot Cassini's path around Titan then back to Enceladus, you would likely be staring at a map that included the sun and all the planets.

Those are the X, Y and Z points that I have here with the CDA data: heliocentric vectors. Therefore, the speed that I've calculated is relative to the sun.

Which is still fascinating, but I can't use it. Doesn't mean I won't play around with it, though.

PLAYING WITH SPEED DATA

Another short lunch today, I couldn't stop thinking about speeds and Cassini whizzing around Saturn and also the sun. I couldn't help myself, so I had a look on Google for "Cassini's orbits" and found [a really great video](#) that showed in-system orbits around Saturn:

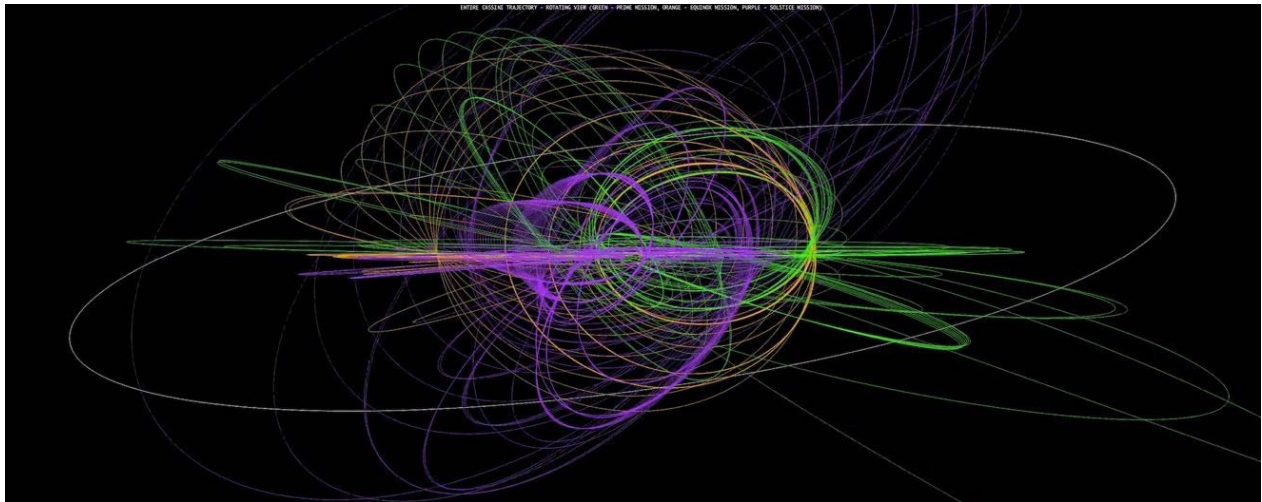


Image credit: NASA/JPL

They call it the “Ball of Yarn.” The fun thing is: I have this data. If I had the time, I could plot X, Y, and Z positions using the INMS dataset and would have a fascinating 3-dimensional graphic to play with. Fun stuff to muse over at lunch, but probably not conducive to future employment.

However. There is just no way I’m not going to have a play with this data. I might find something interesting in there, you never know! I’ll just quickly take a look at high and low speeds during flyby.

The first thing to do is to create a **pythag** function, so I don’t freak out M. Sullivan:

```
--pythag function
drop function if exists pythag(numeric, numeric, numeric);
create function pythag(
  x numeric,
  y numeric,
  z numeric, out numeric)
as $$
  select sqrt(
    (x * x) +
    (y * y) +
    (z * z)
  )::numeric(10,2);
$$
language sql;
```

No need for anything tricky here, just a simple SQL function.

This cleans things up really well:

```
-- using pythag function
select impact_date, pythag(x_velocity, y_velocity, z_velocity) as v_kms
from cda.impactts
where x_velocity <> -99.99
```

This query will calculate the true speed for Cassini relative to the sun and display it as kilometers per second along with a timestamp.

I'll use that query as a starting point. I want to see miles per hour here because... well just because. It's how I think of speed, and I want to get a feel for just how fast Cassini was going that I can't from units I've only encountered in physics labs. I'll also output kilometers per hour in case I have to share this data with the rest of the world, outside the US:

```
-- BOOYAH
with kms as (
  select impact_date as the_date,
         date_part('month', time_stamp) as month,
         date_part('year', time_stamp) as year,
         pythag(x_velocity, y_velocity, z_velocity) as v_kms
  from cda.impacts
  where x_velocity <> -99.99
), speeds as (
  select kms.*,
         (v_kms * 60 * 60)::integer as kmh,
         (v_kms * 60 * 60 * .621)::integer as mph
  from kms
)
select * from speeds;
```

Wow! Look at that! 21,000 miles per hour!

Red:4 Planetary Orbital Observations					
the_date	month	year	v_kms	kmh	mph
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.55	34380	21350
2005-01-01	1	2005	9.54	34344	21328
2005-01-01	1	2005	9.54	34344	21328
2005-01-01	1	2005	9.54	34344	21328
2005-01-01	1	2005	9.54	34344	21328
2005-01-01	1	2005	9.54	34344	21328
2005-01-01	1	2005	9.54	34344	21328
2005-01-01	1	2005	9.54	34344	21328
2005-01-01	1	2005	9.54	34344	21328
2005-01-01	1	2005	9.54	34344	21328
2005-01-01	1	2005	9.54	34344	21328

I know I need to be focusing on finding the speed relative to Enceladus, but I’ve learned to trust my instincts. There might be something in here and I really, really don’t want to fail because I haven’t tried everything.

“Arrange, Act, Assert – triple A. That’s a great way to structure your unit tests, Goose” my mom said. We were sitting at the old, round, coconut and cinnamon wood table in the corner of our kitchen. We bought it one year from a furniture store on the Big Island while we were visiting my dad’s cousin Charles. We were out getting shave ice, and my mom decided to walk into the furniture store next door to Uncle Lou’s (my favorite) in Kona. My sister and I finished the entire thing, washed our sticky hands and decided to go find mom.

She told us she bought a table and not to tell dad. “It’s coconut and cinnamon, smell it!” she beamed and then turned back to fill out some paperwork.

I remember wondering how you made a table out of coconuts and powder, but then I could see it was cinnamon sticks, pressed together with coconut husks underneath a thin layer of finish. You could, if you put your nose up to it, smell the cinnamon.

In fact, you could smell it for years and years after we bought it and each time I smelled it, I thought of that day in front of Uncle Lou’s. To this day, whenever I smell cinnamon I think of home.

“Mom, this is ridiculous. I’m going to spend more time writing tests than writing code! Who the hell ever does this crap? Seriously this is so broke.”

I was in my final year at UCSB, in the same program as my mom, with many of the same professors, getting the same degree. She always made her work sound so fun, and it seemed so effortless for her.

“Discipline, Goose. You can’t build a bridge just because you think it sounds fun. You have to plan, use a lot of math and skill, and then plan some more. When you build it, you test every part of it along the way. You need to be sure you’re correct in what you’ve done.”

I was obdurate. “Mom, this code is obvious. Do I really need to test subtraction for a stupid ATM debit routine? There’s the minus sign right there! Subtraction. Debit. It’s like making work just to make work!”

“OK, we pau. Let’s hele down to the boats. It’s Sunday, I’m sure there’s a regatta we can watch. We can get our windbreakers and sit on the wall. I’ll make some coffee, you find a blanket to sit on.”

“Fine” I replied and slammed the lid on my computer before stomping off.

Fifteen minutes later we were walking down Divisadero, the edge of the fog licking the tops of the trees as the afternoon sun fought to shine through: hot, cold, hot, cold. The marina, just 6 blocks away, was darkened by the gray, cold blanket, the white sails of the racing boats pushed up against the gray like little teeth trying to gnaw their way through it.

We found a spot on the wall just as a klaxon sounded the final leg of the heat. This was a weekend ritual in our family, even though sailing was something we could never afford, especially here at this yacht club.

The sharp wind and staccato *raprap* of the sails snapped me out of my mood. The gray/green water of the bay swirled, chopped and churned as the light spray from the small waves on the rocks below us lofted up and over our heads. I should have been freezing cold, but I wasn’t. I loved this as a kid, and right now it’s the most perfect place in the world.

“Break it down for me Dee, I didn’t see the lead boat’s course,” my mom said.

“The skipper is aggressive. He tacked once after a long close reach as opposed to many tacks with a truer course toward the buoy. The other boats are all trying to stay close hauled which is playing right into the lead skipper’s strategy. He’s on another close reach and has the right of way as he’s ahead and not overtaking. He has a faster point of sail, and they have to get out of his way although...” I trailed off.

“It’s a race! If can, can! If no can, no can...” my mom finished, laughing. A warm feeling came over me. My dad loved to chatter about the strategies that each skipper used. He got to know the boats and their skippers just by observing them weekend after weekend during the regatta season in summer and early fall. It was infectious. He would get so animated, yelling encouragement at the boats he liked and jeering at skippers that he thought sailed ‘dirty’.

Ultimately, every strategy came down to a simple statement from my dad: “If can, can. If no can, no can.”

One year, my mom saved some money on the side and convinced the Commodore of the yacht club to let my father join even though he didn’t have a boat. They knew my dad well, and made an exception in their bylaws for a single membership to go to an “observing skipper.” The club decided to make a big deal out of it and threw him a small welcoming party, catered by the local shellfish company. There were even some small gifts, one of which was a bullhorn with the words “Da Conch” on the side. That was their nickname for him, in reference to the large shell that produced the loud and lovely sound starting off every cheesy luau in Hawaii.

He sat in the same place, every Sunday afternoon when he had the time: 20 paces down on the seawall, directly east of the club entrance. We would go with him as often as we could. He let his membership lapse the next year, thanking the rest of the club, but insisting that his place was just outside, not inside. They offered to waive the fee, and my dad almost moved to tears quietly said: “No can. You sail, I watch and yell.”

“Experience and gut instinct, Dee,” my mom said, breaking what was about to become a bit of a sad moment for me. “That’s how those skippers do that” she added.

“OK...” I replied, confused.

“A skipper that knows their boat can command it like a part of their body. A change in wind, tide, or current, and they’re able to respond as if scratching their own ass” my mom giggled.

“The mental image is interesting, mom, but I don’t get your point” I reply again.

“They can only do that because each of those boats, their sails, hulls, electronics, and rigging, each of those things work consistently and predictably, all in unison. There is never a surprise unless something breaks. If there is a surprise and something doesn’t work the way it should, they lose.”

“They’re fiberglass, cable and cloth mom. I don’t see—”

“A sailboat is a miracle of engineering, Dee. Look: a group of engineers stressed over every single aspect of that boat’s design” she said, pointing to a 32-foot Catalina coming about. “The flow of water over the hull going forward and in reverse. How the keel behaves at every degree of heel. So many little decisions to make such an elegant machine. It looks so simple, doesn’t it? But it takes years and years of practice, testing, failing, retrying and failing again. That’s how you build something elegant and reliable” my mom said.

I thought about watching her work, fingers gliding across the keyboard, commanding Vim like a surgeon. I still can’t use Vim even though I try every day. I just can’t get my mind around it, to take the time required to learn it well. I have more important things to do other than to worry about my text editor! I’m carrying 18 units this quarter and up past midnight every night trying to study. I just don’t have the—

“*Discipline*, Dee,” my mom says, once again ripping me from my self-obsessed reverie. “That’s why we test. Failure is the best teacher, success is the worst. It teaches you to be lazy and arrogant, believing you’re special. That’s not good work, that’s lazy luck” my mom finished, turning back to the boats.

“Feh. *Giggly Jiggly* won again. I hate that boat. What a stupid name!” she says, taking a swig of coffee and another from something that looks suspiciously like a flask. “Get one bettah name already!” she shouts, flashing me a smile. Dad would be proud, he hated that boat too.

We went home that evening, and I sat with her at our coconut and cinnamon-scented table until 8pm, learning about unit testing. The drive back down to Santa Barbara was just under 5 hours, and if you left late enough, you could avoid nasty traffic. I usually left around 8 or 9pm, I’m kind of a night owl. The lack of people on the road and the magic of a dark drive down the misty Central California coast has always been a treat for me.

“I understand how it works, Mom, I’m just having a hard time believing that I need to do it.”

“It’s something you grasp as you gain experience. Failure is good. Failure is right” my mom says, her version of Gordon Gecko’s famous speech about greed. “Especially if

you're trying to learn about the code you've written or the way a system works. Allow yourself the freedom to explore and try something new. If it works, test it out. Most importantly: know how it fails."

"I always structure my tests in a very particular way" she goes on. "I arrange all the things I'll be using for the test; modules, objects, etc. I then do something with them. Finally: I test what I've done. Simple" she finishes.

"Sounds exciting," I say, not helping myself.

"Very," she says. We both look out to the sea as the klaxon blows once again.

A SIMPLE WINDOW FUNCTION

The arrange, act, assert thing seems to be related to what I'm trying to do with this CTE, although there is no assertion. Maybe I can call it "a-ha!" or something. Arrange, Act, A-ha! Gather the data you need, make your calculations, then roll it up. Each part of the CTE has a duty, and if you focus on AAA (my version) then, hopefully, you can keep things focused and simple: 3 queries, each with a task.

I've got the first two, I just need the a-ha part. For this, I think I'll use something I learned about last week at the Postgres User Group.

It was a simple, throw-away comment but for some reason, it stuck with me. In one of the lightning talks, this woman who works at a healthcare organization was trying to describe the way she could compartmentalize multi-tenant data using a simple partitioning scheme. I didn't understand any of it, except for some set theory stuff she was talking about.

During one part of the discussion, when talking about grouping client data she quipped:

I know this looks complex but... at least it's not as bad as trying to figure out window functions...

There were some giggles in the room, and I wrote the term down so I wouldn't forget to look it up later. I didn't know if she was talking about some programming concept or something to do with Postgres. [Turns out it's the latter](#):

A window function performs a calculation across a set of table rows that are somehow related to the current row. This is comparable to the type of calculation that can be done with an aggregate function. However, window functions do not cause rows to become grouped into a single output row like non-window aggregate calls would.

From the examples, it looks like window functions are basically GROUP BY queries without the group by.

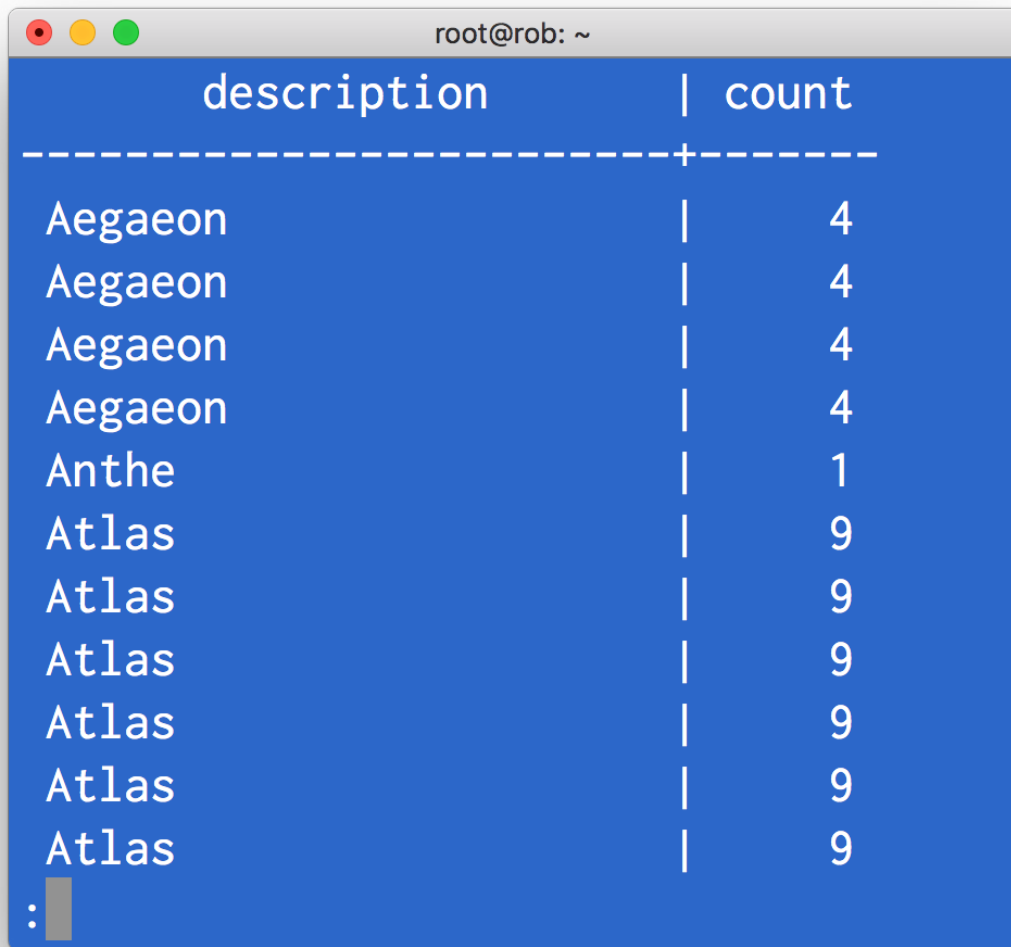
They look weird, but they're very simple and seem to be very powerful.

Here's a query that will get a count of mission plan events for each target:

```
-- simple counts
select targets.description,
count(1) over (
  partition by targets.description
)
from events
inner join targets on targets.id = target_id;
```

The thing that finally made me understand this is to take the keyword **partition** literally. With this query, I'm asking: "how many events do I get for this row if I partition

by **targets.description**. I could partition by anything, and the aggregate function I'm using would work against that partition:



A terminal window titled 'root@rob: ~' displays a table with two columns: 'description' and 'count'. The table is rendered with a blue background and white text. The 'description' column lists various targets, and the 'count' column shows the frequency of each. The data is as follows:

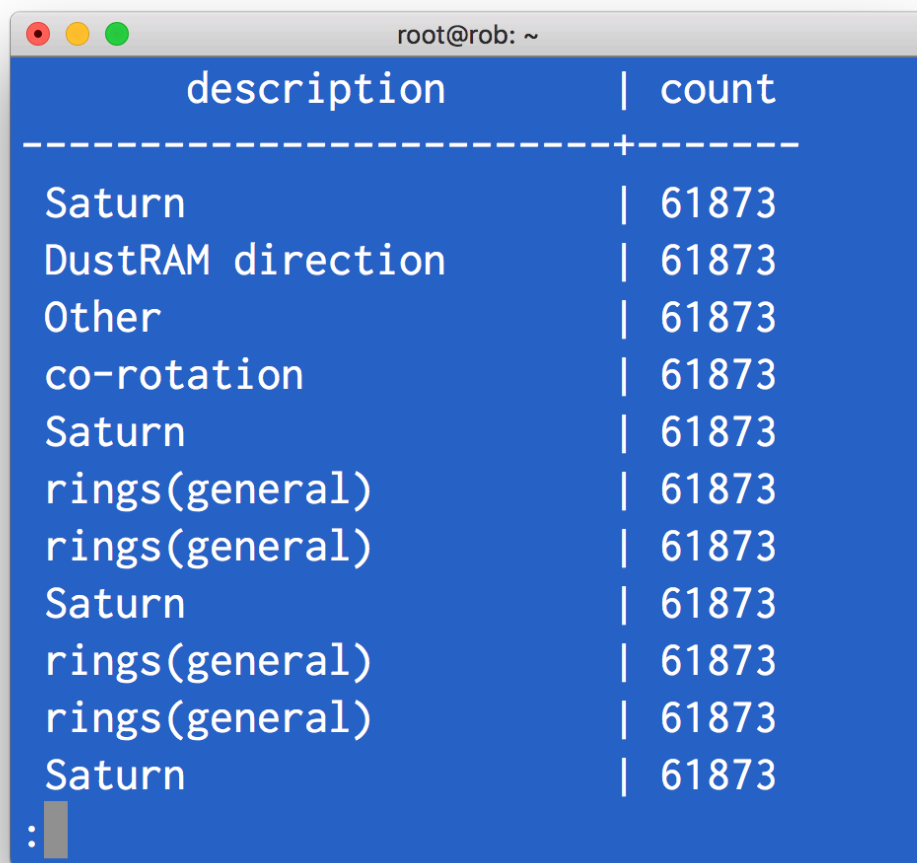
description	count
Aegaeon	4
Aegaeon	4
Aegaeon	4
Aegaeon	4
Anthe	1
Atlas	9
Atlas	9
Atlas	9
Atlas	9
Atlas	9
Atlas	9

The terminal window has a title bar with three colored buttons (red, yellow, green) on the left. Below the table, there is a prompt character ':' followed by a cursor bar.

The rows are repeated here because I'm not doing a group by. It's almost as if I'm running a subquery, but it seems to be much faster than that.

I could ask for a count of all the events if I wanted to, just leave out any **partition** clause:

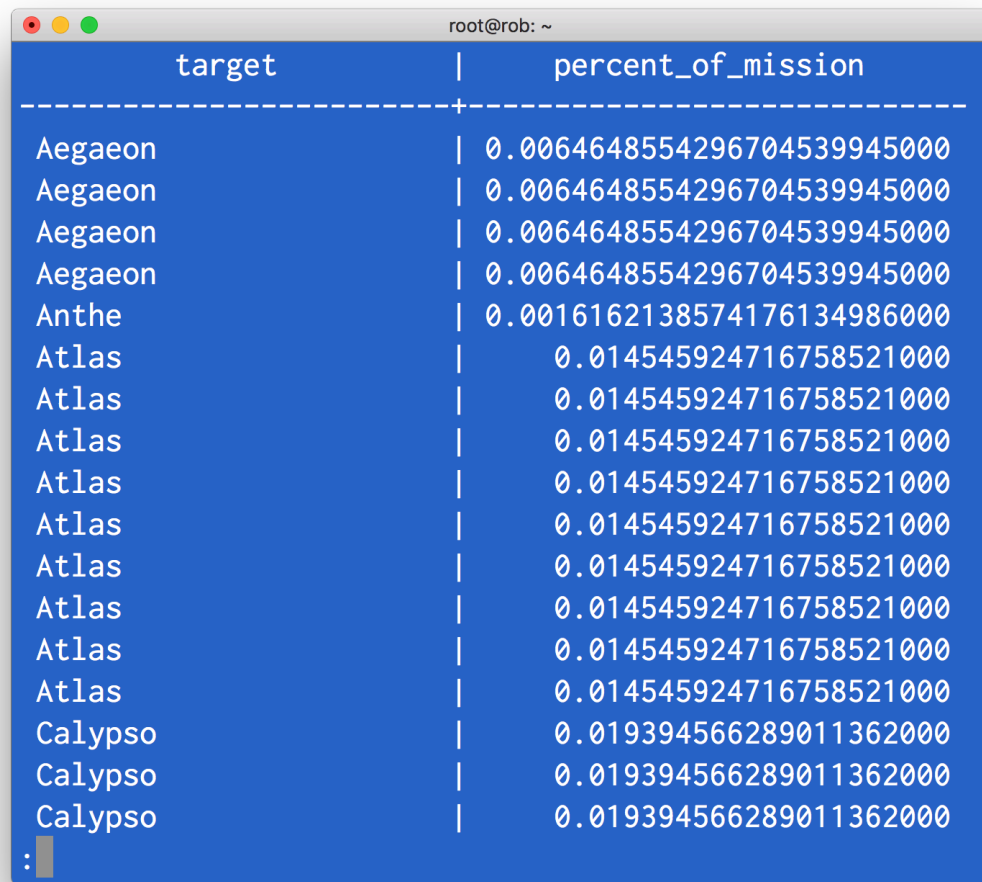
```
-- target distribution
select targets.description,
count(1) over ()
from events
inner join targets on targets.id = target_id;
```

A terminal window with a blue background and white text. The window title is 'root@rob: ~'. It displays the output of a SQL query, which is a table with two columns: 'description' and 'count'. The table has 12 rows of data. The first row is 'Saturn' with a count of 61873. The second row is 'DustRAM direction' with a count of 61873. The third row is 'Other' with a count of 61873. The fourth row is 'co-rotation' with a count of 61873. The fifth row is 'Saturn' with a count of 61873. The sixth row is 'rings(general)' with a count of 61873. The seventh row is 'rings(general)' with a count of 61873. The eighth row is 'Saturn' with a count of 61873. The ninth row is 'rings(general)' with a count of 61873. The tenth row is 'rings(general)' with a count of 61873. The eleventh row is 'Saturn' with a count of 61873. The twelfth row is a colon ':' with a count of 61873. The table is separated from the rest of the terminal by a dashed line.

description	count
Saturn	61873
DustRAM direction	61873
Other	61873
co-rotation	61873
Saturn	61873
rings(general)	61873
rings(general)	61873
Saturn	61873
rings(general)	61873
rings(general)	61873
Saturn	61873
:	61873

Seems useless, unless you wanted to know what percent of the mission events were devoted to a single target. If I were using **GROUP BY**, this query would be kind of intense. I think this is pretty straightforward:

```
-- percentile of target distribution
select
  targets.description as target,
  100.0 *
  (
    (count(1) over (
      partition by targets.description
    )::numeric /
    (
      count(1) over ()
    )::numeric ) as percent_of_mission
from events
inner join targets on targets.id = target_id;
```



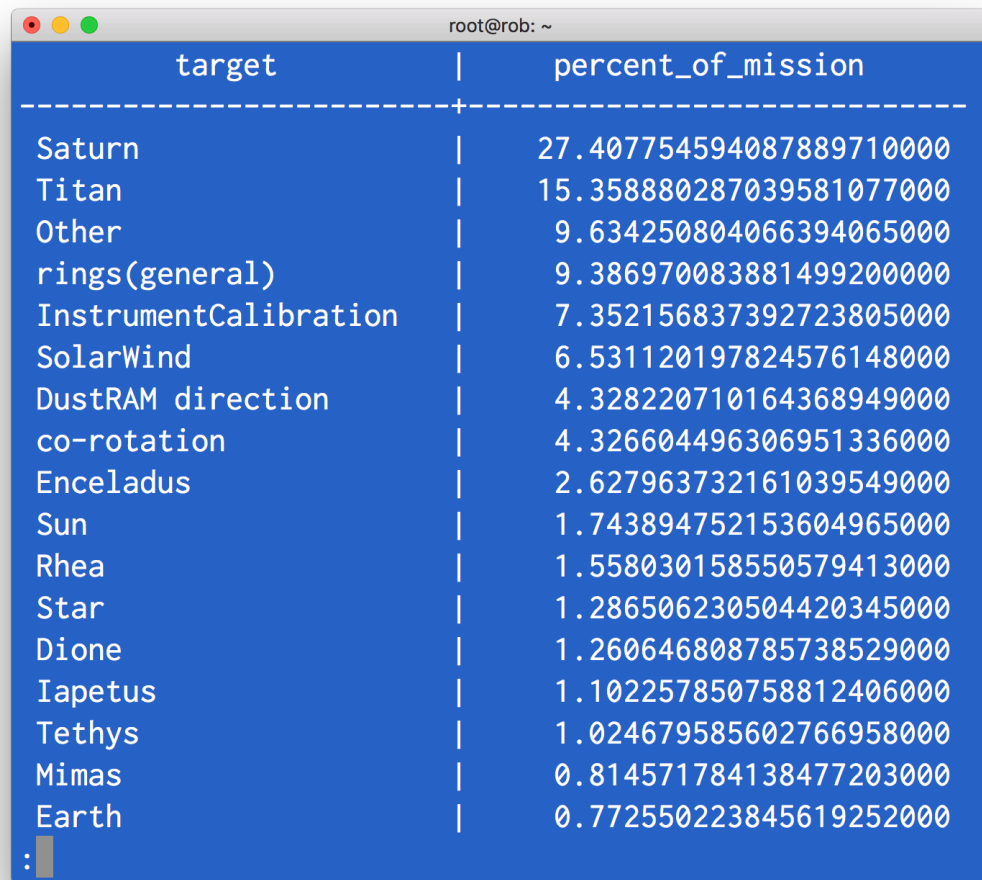
A terminal window with a blue background and white text. The window title is 'root@rob: ~'. It displays a table with two columns: 'target' and 'percent_of_mission'. The table has a dashed line separator between the header and the data rows. The data rows show repeated values for 'target' and 'percent_of_mission'.

target	percent_of_mission
Aegaeon	0.0064648554296704539945000
Aegaeon	0.0064648554296704539945000
Aegaeon	0.0064648554296704539945000
Aegaeon	0.0064648554296704539945000
Anthe	0.0016162138574176134986000
Atlas	0.014545924716758521000
Atlas	0.014545924716758521000
Atlas	0.014545924716758521000
Atlas	0.014545924716758521000
Atlas	0.014545924716758521000
Atlas	0.014545924716758521000
Atlas	0.014545924716758521000
Atlas	0.014545924716758521000
Atlas	0.014545924716758521000
Calypso	0.019394566289011362000
Calypso	0.019394566289011362000
Calypso	0.019394566289011362000

The repeated results confused me, but then I remembered that this is just a query, and I’m doing these little, set-based rollup queries “next to them.” If I did a **select distinct**, however, that should make it more interesting:

```
-- isolating with distinct
select
  distinct(targets.description) as target,
  100.0 * (
    (count(1) over (partition by targets.description))::numeric
    /
    (count(1) over ())::numeric)
  as percent_of_mission
from events
inner join targets on targets.id = target_id
order by percent_of_mission desc;
```

Now, this is informative:



A terminal window with a blue background and white text. The window title is 'root@rob: ~'. It displays a table with two columns: 'target' and 'percent_of_mission'. The table lists various targets and their corresponding percentages of mission events. The targets are listed in descending order of percentage. The table is separated by dashed lines. A cursor is visible at the bottom left of the table area.

target	percent_of_mission
Saturn	27.407754594087889710000
Titan	15.358880287039581077000
Other	9.634250804066394065000
rings(general)	9.386970083881499200000
InstrumentCalibration	7.352156837392723805000
SolarWind	6.531120197824576148000
DustRAM direction	4.328220710164368949000
co-rotation	4.326604496306951336000
Enceladus	2.627963732161039549000
Sun	1.743894752153604965000
Rhea	1.558030158550579413000
Star	1.286506230504420345000
Dione	1.260646808785738529000
Iapetus	1.102257850758812406000
Tethys	1.024679585602766958000
Mimas	0.814571784138477203000
Earth	0.772550223845619252000

Just over a quarter of the mission events had to do with Saturn, another fifteen percent involved Titan.

I wonder how this breaks out for each instrument and team? I know some were used more than others...

```
-- looking at teams
select
distinct(teams.description) as team,
100.0 *
(
    (count(1) over (partition by teams.description))::numeric /
    (count(1) over ())::numeric
) as percent_of_mission
from events
inner join teams on teams.id = team_id
order by percent_of_mission desc;
```

Well now, look at this:

Red:4 Planetary Orbital Observations	
team	percent_of_mission
CIRS	19.344463659431415965000
UVIS	14.882097199101385095000
ISS	14.528146364326927739000
VIMS	10.304979554894703667000
INMS	7.672167181161411278000
CDA	7.216394873369644271000
RPWS	6.789714415011394308000
CAPS	5.432094774780598969000
MAG	5.049052090572624570000
MIMI	4.512469089909976888000
RSS	2.044510529633281076000
RADAR	1.323679149225025455000
MP	0.885685193864852197000
PROBE	0.014545924716758521000
(14 rows)	
enceladus=#	

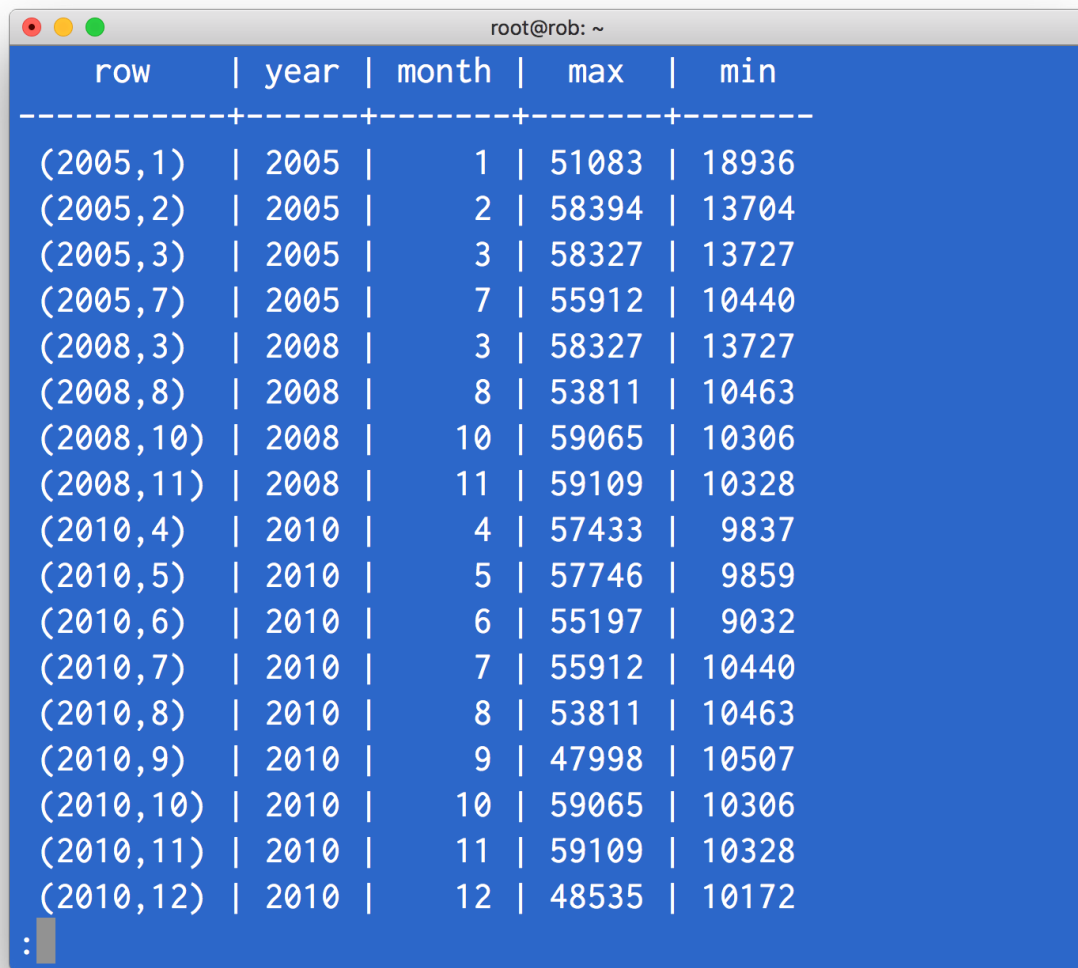
Looks like the CIRS team got the most time with Cassini, followed by the ultra-violet imaging system.

I can do the same thing with my CTE, but with some more interesting results. Instead of a **count**, I can get the **min** and **max** for a given month:

```
-- min/max speeds
with kms as (
  select impact_date as the_date,
  date_part('month', time_stamp) as month,
  date_part('year', time_stamp) as year,
  pythag(x_velocity, y_velocity, z_velocity) as v_kms
  from cda.impacts
  where x_velocity <> -99.99
), speeds as (
  select kms.*,
  (v_kms * 60 * 60)::integer as kmh,
  (v_kms * 60 * 60 * .621)::integer as mph
  from kms
), rollup as (
  select distinct(year,month),
  year, month,
  max(mph) over (partition by month),
  min(mph) over (partition by month)
  from speeds
  order by year, month
)

select * from rollup;
```

I like this! I'm arranging the data in the kms query, I'm acting on it by calculating a true speed using the Pythagorean theorem, and then I get the A-ha!



A terminal window with a blue background and white text. The window title is 'root@rob: ~'. It displays a table with 5 columns: 'row', 'year', 'month', 'max', and 'min'. The table is separated by dashed lines. The data rows are as follows:

row	year	month	max	min
(2005,1)	2005	1	51083	18936
(2005,2)	2005	2	58394	13704
(2005,3)	2005	3	58327	13727
(2005,7)	2005	7	55912	10440
(2008,3)	2008	3	58327	13727
(2008,8)	2008	8	53811	10463
(2008,10)	2008	10	59065	10306
(2008,11)	2008	11	59109	10328
(2010,4)	2010	4	57433	9837
(2010,5)	2010	5	57746	9859
(2010,6)	2010	6	55197	9032
(2010,7)	2010	7	55912	10440
(2010,8)	2010	8	53811	10463
(2010,9)	2010	9	47998	10507
(2010,10)	2010	10	59065	10306
(2010,11)	2010	11	59109	10328
(2010,12)	2010	12	48535	10172

At the bottom of the table, there is a colon ':' followed by a cursor bar.

High and low speeds for each month for the Enceladus flybys!

Too bad heliocentric data is useless to me, It's fun to look at, though.

I feel like I've made good progress today and since everyone else is gone (as it's Halloween), I think I'll sneak out too. I have an idea for calculating a true speed using altitude data and some trigonometry as the velocity information just isn't here in the CDA data. I'll do that tomorrow.

I'll send Rob an update, and then I'm off to Kelly's party in the East Bay.

A TIGHT SHIP

From: M. Sullivan sullz@redfour.io
Subject: Postgres and Red:4
To: Dee Yan yand@redfour.io
Date: November 1, 2017

POSTGRES IS CORRECT

It occurred to me last night that you were never properly “on-boarded” here at Red:4. Yes, I know you went through orientation when you became an intern, but Rob never sent you any information about our commitment to Postgres and, more importantly, *why* we are committed to using it.

PostgreSQL is a critically important piece of software, M. Yan, to us and to the world in general. It’s a free alternative to the gigantic, expensive and baroque enterprise database systems. The amount of money that large corporations such as Microsoft, Google, Amazon, Oracle, and IBM spend on influencing database developers is astounding. They know as we know: everything starts with the database.

You’ll hear it a lot here, and I’ll repeat it because it’s that important: *your data is your gold*. Where you keep that data is almost as important, and *where* is interchangeable with *who*.

One of my favorite movies is *Harry Potter* (in fact many people tell me I remind them of Hermione). One of my favorite things about *Harry Potter* is Gringotts, the bank where every wizard keeps their money and family treasures. The bank is much more than caves, carts, and dragons. *It’s the goblins*. They’re the ones guarding the money, staring at you menacingly as you walk through the door.

Database platforms are just the same: you need to know who is guarding your data. Do you want that to be a massive corporation with an agenda that involves your company's money? Or do you want to be in control of your data and your destiny?

PostgreSQL is that database platform, M. Yan. It exists because it *must exist*. It is outstanding in almost every way, and our support of it is an absolute necessity if we're to keep ourselves free from corporate leveraging.

I've written up a bit of a manifesto regarding our use of Postgres, which I've attached here as a PDF. Please read through it at your convenience. There will be a quiz on Friday.

Best, S

THE DATA-FIRST MINDSET

M. Sullivan, Director of Digital Waste Management

Greetings. As you know, we as a company use PostgreSQL as our primary data store unless there is a very, very good reason not to. As of today, the "good reason" count is still sitting at 0, where it shall remain.

I decided to create this guideline to help you come to terms with the truth: that PostgreSQL is almost always the clear, distinct and correct choice. What follows is a description of why this is the truth.

Hell, as they say, is other people. It's only a matter of time before an app dev will contaminate your database into a horrid shape, or some freak with a spreadsheet contaminates everything with improper text encoding.

We survive here at Red:4 because we realize that without the data, there is no Red:4. This is my job. I do mine, you do yours, and everything will be just fine.

YOU AND YOUR DATA

Every developer needs to cultivate an appreciation for the data their application produces. Yes, code is exciting, code is fun.

Code, no matter what language or platform, is also a disease. The data is your business, however.

Consider the data we've been sifting through over the last few years and all the secrets it holds. Cassini and its 12 instruments lie disintegrated within Saturn's atmosphere, nonexistent. Yet, the data sits here in front of us, holding the keys to so many mysteries.

That data could change science! In fact, just last week I read two fascinating articles about discoveries made using that exact data. One had to do with Titan's cooling south pole, and the other was a "proof," if you will, that Enceladus has a porous core.

There is still so much in that data set. The entire history of the voyage and every reading from each of the instruments. So many patterns and relationships, so many discoveries that could change humanity forever.

Just imagine Cassini being built by a group of aerospace engineers that only casually thought about the data Cassini was supposed to gather. What would be the point? Think about all the applications being built now across the world that view the database as merely a place to maintain application state. That should be a crime, punishable by many years in prison. They don't certify application developers but they should. We might see less JavaScript nonsense and the careless, blind saving of `request.body` as if it's not full of leeches and vomit.

GET BENT

If you are ever in a position to work with the data your application produces, you'll see your career in a completely different way.

When you have to sift through `order` records or logs to answer a specific business question, you'll wonder to yourself *why the hell am I not tracking X, Y and Z! Perhaps I should be in jail...*

When you have a data-first mindset, all of this becomes clear.

Your job becomes easier, and you guarantee your position for many years. You'll become valuable to the team, to your managers and, most of all, to me. In short: you'll be a better developer.

One of the greatest things you can do, as a developer, is to ensure that your application generates data that will not just answer but *anticipate* your boss's or client's questions about their business. They need to know what's happening with the application, so they can understand the value they're providing the customers. If they can understand that, everyone's happy, everyone makes money.

That's all well and good, but when you get to the next level, where you're able to track behavioral patterns and enter the world of prediction, that's where the magic is.

You can only get there by cultivating a genuine appreciation for the power of data. This power is tempting, and it makes good people do bad things, such as collecting data without a thought for who will be using it, or shoehorning it into a model built from their own ridiculous and unexamined biases, which inevitably results in their becoming puppets and stooges for people cleverer than they are.

Good data is good observation, as free of bias as possible, even down the very platform, it's served from. Your data is only as good as your ability to interrogate and understand it. This ability should not be dictated by licensing and platform restrictions! That turns your database into a cage and your data into a way to extort money from you.

This is unacceptable at every level.

WHY POSTGRES?

Storing application data is a solved problem: *use Postgres*. This isn't just my opinion, this is fact. Postgres is suited well for massive enterprise systems like Skype or clever data-centric applications like Instagram. It can be used efficiently for the smallest of systems or scaled out massively for the largest.

It can be used as a traditional, relational data store that leans on normalization rules, or it can be used as a document database, storing records as binary JSON that absolutely destroys other document systems regarding speed and scaling. You don't have to take my word for this, the team at MongoDB seem to know it quite well. One of their products, the "Business Intelligence Connector" is powered by none other than *PostgreSQL*.

Add-on support tools (most of them open source) abound and Postgres can stand toe to toe with Oracle, SQL Server or any other system you'd like to compare regarding speed and scalability. It's been around for decades and is considered one of the premier examples of software excellence in any capacity, not just open source.

In fact, many software developers I know say it's the most advanced software project in the world regarding transparency, testing, documentation, code quality... I could keep going, but I need to move on.

Admittedly, the above are subjective reasons. Let's take a look at some objective ones.

ACID

Atomic, Consistent, Isolated and **Durable**. That's an acronym which merely means that if you get an "ack" (acknowledgment) that data has been written, it's been written. It's on disk, and you can feel right about it.

The guarantees that come with ACID are not trivial. One of the downsides of using a concurrent system is precisely this: you're not sure if data will be written in some cases. For some data, this is OK, and many systems allow you to tweak this. If you Google for NoSQL data loss, however, you'll find that's a thing.

Things are getting much better these days, and yes, I know that concurrency is the future. For now, I'll stick with Postgres until I need something with concurrent scaling. Even then I'll stick with Postgres as concurrency features are being added all the time.

Locking And the MVCC

I had a friend who worked on the data team at Twitter back in 2010. This was when it was quite small, and they kept everything in a massive MySQL database. We would have lunch (usually tacos at Rosie's in South Park), and I would beg him to let me help him port their data to a better system. Not a trivial task, to be sure, but I kept insisting that someday, something would happen where they would wish they were using Postgres.

"Our DBA will never do that. He's a MySQL person and has better things to do than repainting his bike shed." That expression, so casually flung at everything a developer chooses to ignore.

That day happened sooner than I thought. The development team needed to update every single user's record to set a flag (I can't remember precisely what it was) so the DBA at the time decided to run a query that looked something like this:

```
update users set thing=1;
```

Seems innocent enough, doesn't it? Here's the problem: what if the write takes a really long time and the user wants to see their data?

Most database systems out there will lock records when they're part of a write transaction. This is primarily to avoid serving up dirty data, or data that's old and has been updated behind the scenes. To prevent this, each record in the transaction is locked down until the transaction completes. Every query that requests a read on that data is put into a queue to await the unlock.

Usually, this is a good thing, but when you have 10 million users on a single-box database... you're in trouble. This is what Twitter found out when their website crashed (which it did, repeatedly): all user data was unavailable as the database had locked every single row.

Postgres locks data as well, but only as a fallback measure. Instead, it uses a thing called the MVCC: *Multi-Version Concurrency Control*.

Every bit of data that is part of a transaction is “lifted,” if you will, into virtual memory. All changes happen there. You can even query that data if you’re within the transaction and see those changes. Only when the transaction is complete is the entire change written to disk in one operation. If you try to write to a record that’s currently in a transaction, you won’t be able to. That write will be queued until the entire transaction is completed.

This avoids read locks for the most part, and if Twitter had been using Postgres at the time, they (likely) would have prevented that outage.

Enterprise features ready to go

When you use a large, enterprise system like SQL Server or Oracle, you end up paying big money in licensing fees so you can access specific features. As a developer, this might not matter to you, so the basic editions usually work just fine. As your business (and therefore database) grows, however, this will change.

Table compression is one such feature. For small databases, this is no big deal, but as your data expands into the gigabit territory and beyond, you’ll want the smallest footprint possible to avoid costly upgrades. You can pay for that with an enterprise license for SQL Server, or you can have it for free with Postgres.

Those large tables will need to be indexed from time to time, and if you’re pumping in large amounts of data, that indexing process can eat up resources. With non-enterprise databases, this indexing operation is handled with a single process. With Postgres, you can use **create index concurrently** and you just don’t have that problem.

Finally: partitioning data. This is something you can do with Postgres now, for free, easily. It’s way beyond the scope of this manual, but basically, partitioning allows you to scale your database easily by separating the data into physical partitions.

In short: Postgres will grow with you, happily.

Friendly

Have you noticed the error messages as we've been building our system? They're somewhat useful, aren't they?

As you have seen, I've been doing a lot of explicit type casting when creating various queries. Postgres is pretty good about coercing things when you need, but I don't typically like to rely on it. A bit too magical in some cases.

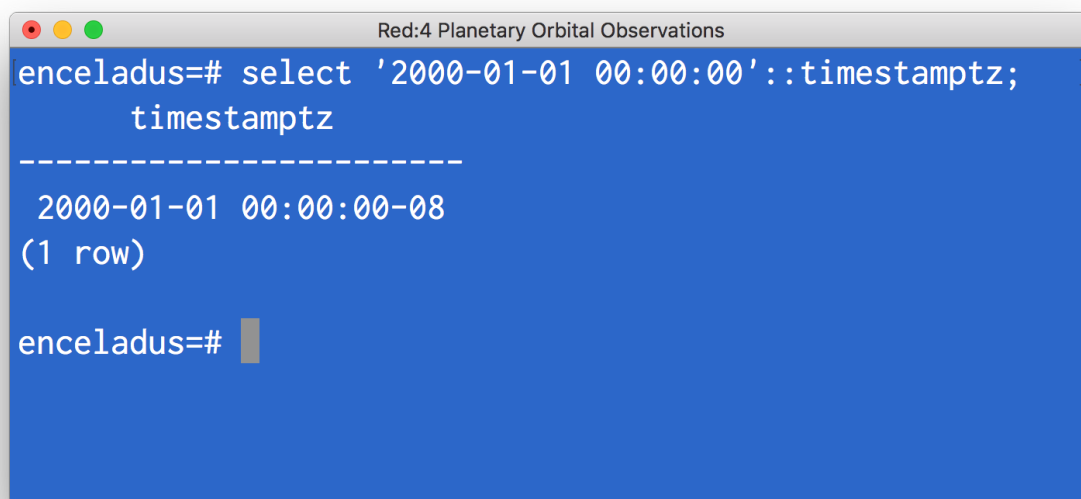
For instance, what will happen here?

```
select '2000-01-01 00:00:00'::timestampz;
```

Take your best guess, as this one is a mystery! Postgres has a few things to figure out, namely:

- Is this a date and can I parse it.
- What time zone is this in?

Postgres will indeed be able to parse this and, given that I didn't specify a time zone or any milliseconds, it will assume I want my current time zone with 0 milliseconds:

A screenshot of a PostgreSQL terminal window titled "Red:4 Planetary Orbital Observations". The terminal has a blue background and white text. The prompt "enceladus=#" is followed by the query "select '2000-01-01 00:00:00'::timestampz;". The output shows a single row with the timestamp "2000-01-01 00:00:00-08" and "(1 row)". The prompt "enceladus=#" is followed by a cursor.

```
enceladus=# select '2000-01-01 00:00:00'::timestampz;
timestampz
-----
2000-01-01 00:00:00-08
(1 row)

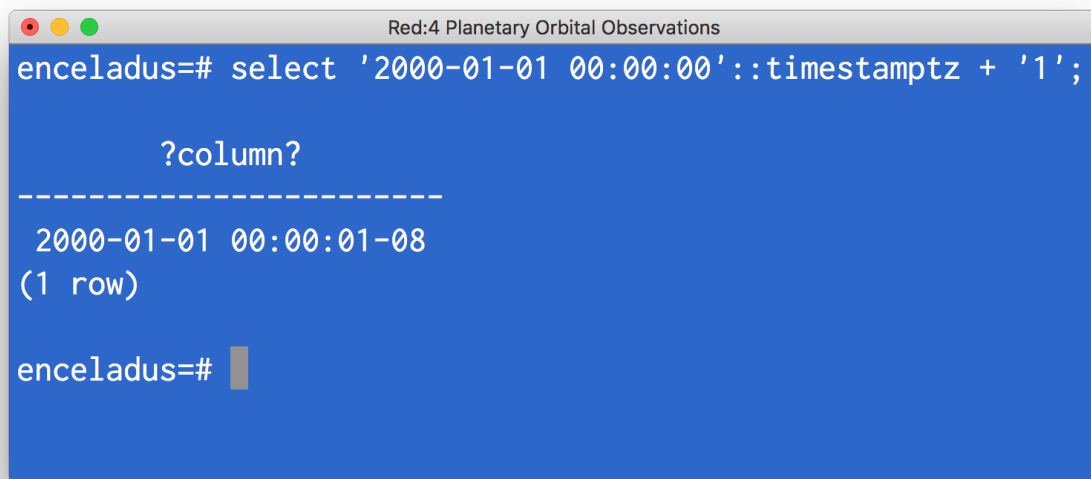
enceladus=#
```

This all makes sense if you understand the rules, but what if I did this?

```
select '2000-01-01 00:00:00'::timestampz + '1';
```

Hmmm. Well, we know that Postgres can determine intervals based on string values, but is there a default interval? If so, what is it?

Turns out that yes, there is a default and it's seconds:

A screenshot of a PostgreSQL terminal window titled "Red:4 Planetary Orbital Observations". The prompt is "enceladus=#". The user enters the query "select '2000-01-01 00:00:00'::timestampz + '1';". The terminal shows the result as a single row with the value "2000-01-01 00:00:01-08". Below the result, it says "(1 row)". The prompt "enceladus=#" is shown again with a cursor.

```
enceladus=# select '2000-01-01 00:00:00'::timestampz + '1';  
  
      ?column?  
-----  
2000-01-01 00:00:01-08  
(1 row)  
  
enceladus=#
```

We can see that because our timestamp has a second added to it.

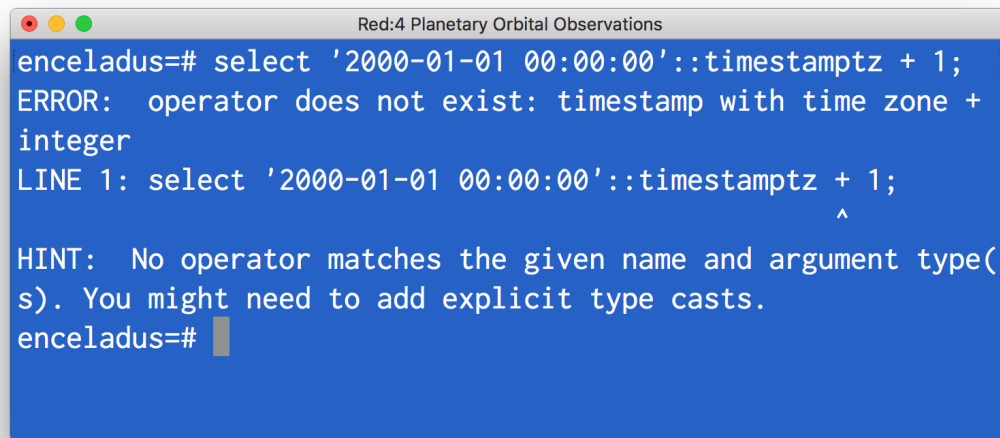
What if we tried to use an integer 1 instead of a string?

```
select '2000-01-01 00:00:00'::timestampz + 1;
```

My app dev friends (yes, I have a few on Slack) like to play this game with JavaScript, opening their browser and digging for some of the dumbest code I've ever seen. The game, I gather, is to see who can create the most confounding JavaScript expression. This is met with laughter, which I don't understand. Applications are written with this boiled puke language, applications that drive me insane with their inability to perform even the most basic of tasks, such as *operate*.

This game cannot be played with Postgres as Postgres doesn't lend itself to trivial idiocy. It's a sane system. It will try its best to help you, but only to a reasonable point. At some level, you just have to know what you're doing.

Here I'm trying to add an integer to a date. Postgres could reason that I really meant to use a string, or it could just do the sane thing and ask me to be a bit clearer, please:

A screenshot of a PostgreSQL terminal window titled "Red:4 Planetary Orbital Observations". The terminal has a blue background and white text. It shows a SQL query being executed: `enceladus=# select '2000-01-01 00:00:00'::timestampz + 1;`. Below the query, an error message is displayed: `ERROR: operator does not exist: timestamp with time zone + integer`. The next line shows the query again with a caret pointing to the `+` operator: `LINE 1: select '2000-01-01 00:00:00'::timestampz + 1;`. Below the error, a hint is provided: `HINT: No operator matches the given name and argument type(s). You might need to add explicit type casts.`. The prompt `enceladus=#` is visible at the bottom of the terminal.

This, right here, is one of the main reasons we use Postgres.

Not only will it insist on clarity when it comes to data, but it will also offer hints on how to *be* clearer.

FINE: NOSQL IF YOU NEED IT

Using JSON in a database (even if it's binary JSON) sidesteps everything that's good about a relational system. There are times where it's useful, however.

Postgres can store JSON in binary format, JSONB. You can also index it, on specific keys or the whole document. It's ridiculously fast and puts MongoDB to shame (as if that's difficult). The only problem is that querying it is not nearly as nice as querying

MongoDB. Yes, I've used MongoDB. I don't have opinions about things unless I've taken the time to get to know them.

Rob has done some work this way, creating a few projects that I think are kind of interesting:

- [MassiveJS](#). This is a Postgres-centric data access tool that mimics the MongoDB api. It's for Node and is currently run by one of the core maintainers.
- [Mœbius](#): A clone of MassiveJS but for Elixir. Same thing – a full abstraction of MongoDB's API on top of Postgres.
- [pg_docs_api](#): this is a weird one – it's a bunch of functions that make Postgres look, act and feel sort of like MongoDB. It's built using Google's V8 engine within Postgres.

Yes, you can use JavaScript within Postgres to build functions. This is grounds for immediate termination from Red:4, however, so be warned. You can also use Ruby, R or Python if SQL or the built-in PLPGSQL is not up to the task. However, I prefer to keep that kind of thing outside the database.

I have a few .NET friends that are starting to use [Marten](#) from Jeremy Miller and gang. They've extended the entire notion of a document to an "Event Store," which is a bit ceremonious for me but my friends enjoy it a lot. As long as it's Postgres, I'll support it.

If you have any question, feel free to drop me an email.

Best, S

GRAVITY ASSIST

From: Rob Conery rob@redfour.io
Subject: RE: Update on speeds, pivoting
To: Dee Yan yand@redfour.io
Date: November 1, 2017

Damn. I was hoping that the velocity data in the CDA was relative to the target. I like your idea RE using math, as long as we know the calcs are *correct, and* most importantly: *accurate*.

Check it with M. Sullivan, and I'll forward your note to the Analytics Team. From what I remember of trig, yes, you should be able to get the value of any side of a triangle if you know enough information about the angles of each side.

You should be receiving an email from someone on the Analytics Team when they get in today, which is usually around 10am. I've asked them to give you as much detail as possible about their analysis window. Thanks, Dee, good work. R

From: Sara C. boss@redfour.io
Subject: Your work with the Analytics Team
To: Dee Yan yand@redfour.io
Date: November 1, 2017

Rob mentioned to me that he was putting you in touch with the Analytics Team and I would like to emphasize how important it is that you work with them closely.

I wish we had more resources to assist you in your current task load, but as Rob mentioned to you already: it's all hands on deck to restore our production system. The timing on this couldn't be worse.

I know you would love to "save the day" and all, but a hero isn't what we need right now. We need a team player.

Regards,

Sara

From: Eno L. eno@redfour.io
Subject: RE: Update on speeds, pivoting
To: Dee Yan yand@redfour.io
Date: November 1, 2017

Hi Dee! Nice to meet you, I'm Eno, and I work on the Analytics Team. Rob asked me to describe our analytics window for Enceladus, so you could calculate its precise position for each flyby. Appreciate you doing this!

The area that we're most interested in is just above the Tiger Stripes, four elongated *sulci* in the icy surface on the south pole. That was a new word for me (*sulcus*), but I found out it simply means "subparallel furrow or ridge." There are all kinds of unusual names for planetary features, all of which are thought up by a governing body called *The International Astronomical Union*.

The IAU put together a report for Enceladus entitled *Enceladus Nomenclature*, which diagrams every interesting feature of the moon and then gives it a name following some kind of theme. You can take a look [here](#) if you're interested.

The theme for Enceladus? Sir Richard Francis Burton's translation of [*Thousand Nights and a Night*](#), (more commonly known as *A Thousand and One Arabian Nights*).

The Tiger Stripes are 4 sulci, and they are the *only* source for the plumes, which we found kind of surprising. Each one is about 130 km long and 2km wide, and are named *Damascus*, *Baghdad*, *Cairo* and *Alexandria*. 4 beautiful, ancient cities of the desert seem oddly perfect for the barren, icy moon:

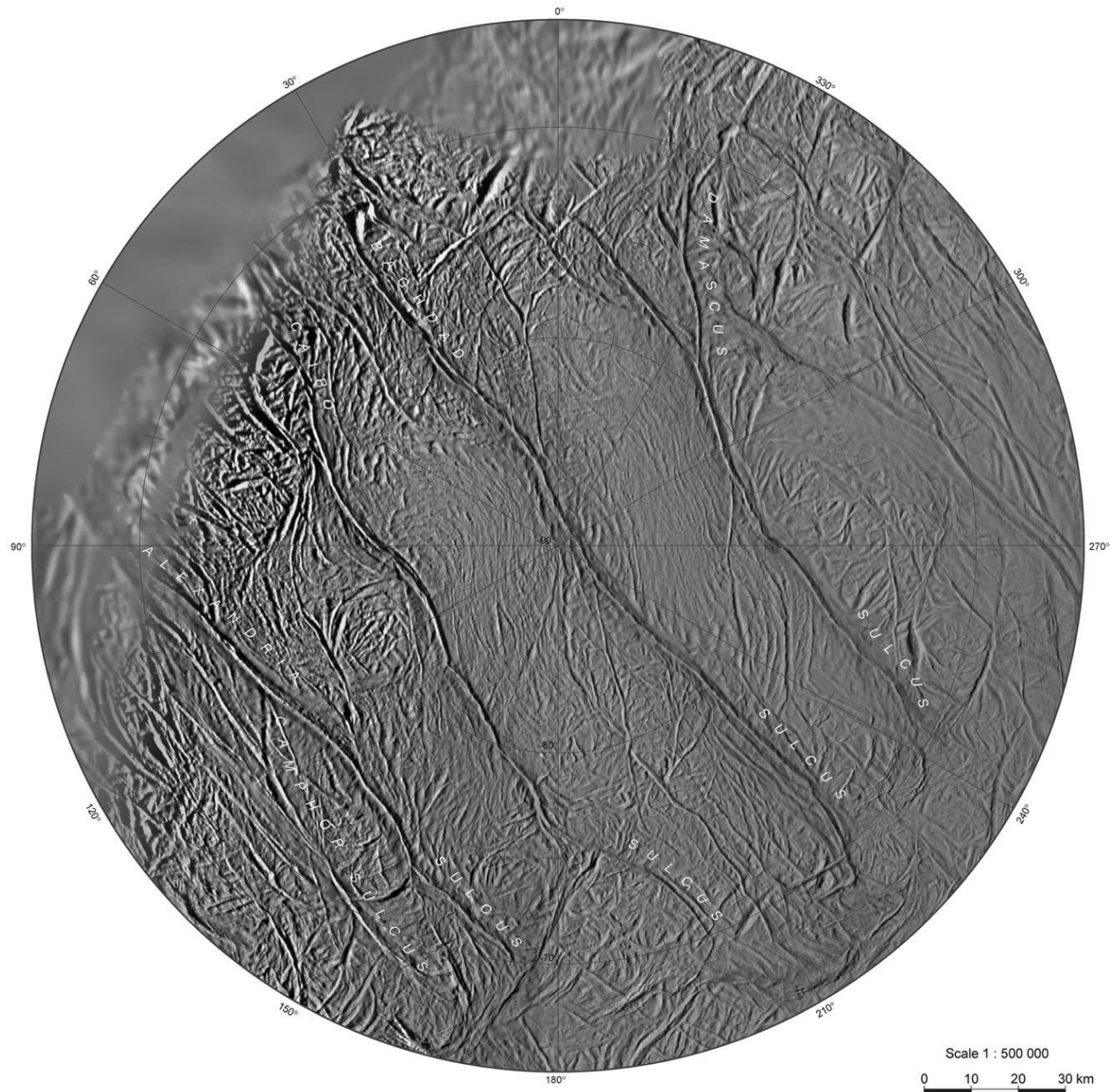
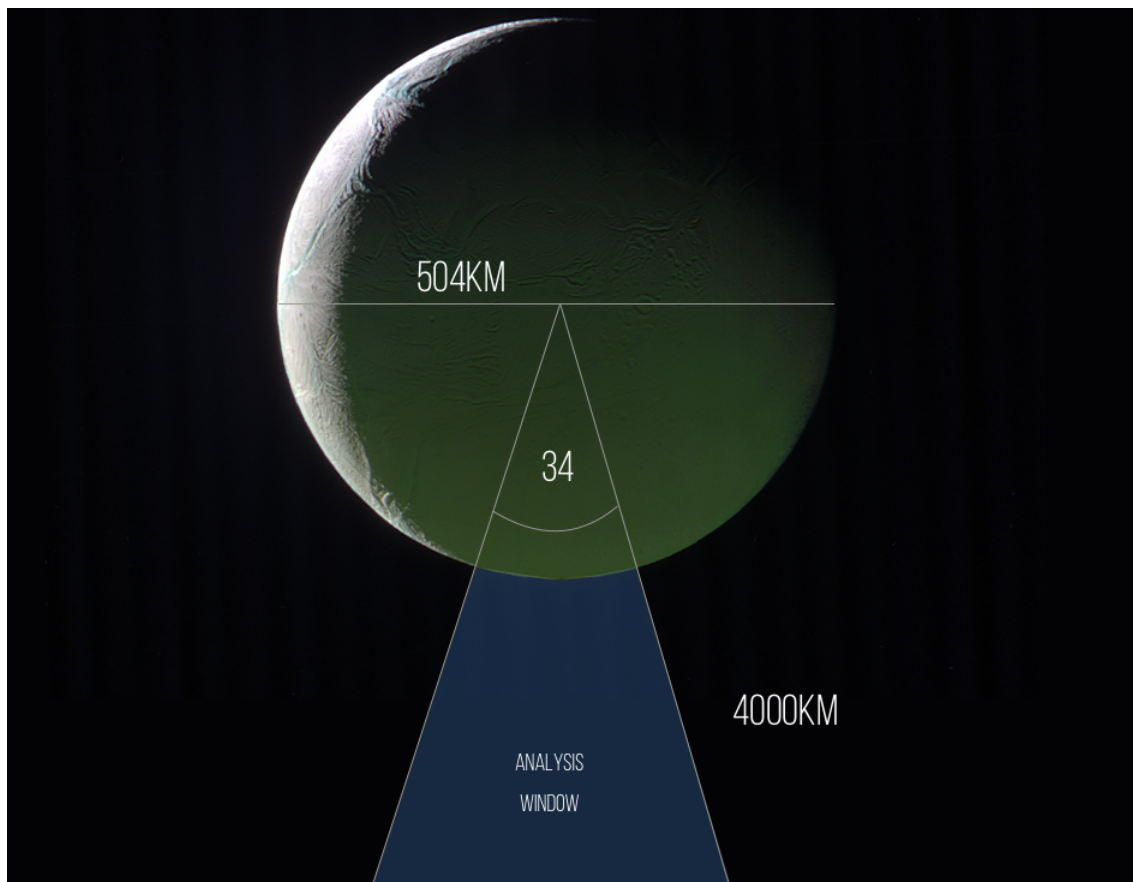


Image credit: NASA/JPL

A much higher-resolution map of Enceladus's features can be seen [here](#) if you're interested. It was created by Steve Albers of the University of Arizona and is quite detailed.

As you can see from the image above, the sulci occupy an area that's roughly 150 square kilometers at the surface. We're primarily interested in what goes on in this area and in a semi-conical zone projected 1500 km above.

We think the source of the plume material is *within* the moon's porous core, rather than just below the ice, so we're using that as our model. A 150km expression area on the surface gives us a 34-degree angle back to the core, so extend that another 4000km into space and you have the analysis window:



This is not to scale, just for you to use as a reference. We want to know, as precisely as possible, when Cassini entered this area and when it left. I understand that you need to

figure out some speed calculations to do this and I have some ideas (which I'm sure you thought of already). Let me know if you'd like some help or a second pair of eyes.

Oh yeah, one last wrinkle. We know that the nadir of each flyby differed regarding relative location to the Tiger Stripes. For now, just assume that each of the flybys did, in fact, center on the south pole. We'll adjust whatever you give us.

Thanks, Dee!

EL

FLYBY TRIGONOMETRY

November 1, 2017, 1547

Impressive set of emails to start my day with. "We trust you, but don't screw up" is a recurring theme. Can't focus on that crap right now, I need to move forward and do some math.

I should also make some time to go over to the Analysis Building in The Presidio. I would love to know what those people are up to! The office here is so empty all the time, the rest of the Data Team are still offsite trying to get production back up and running. What a headache! I do wonder what Nas is doing right now... what he must be feeling. I hope he's doing alright.

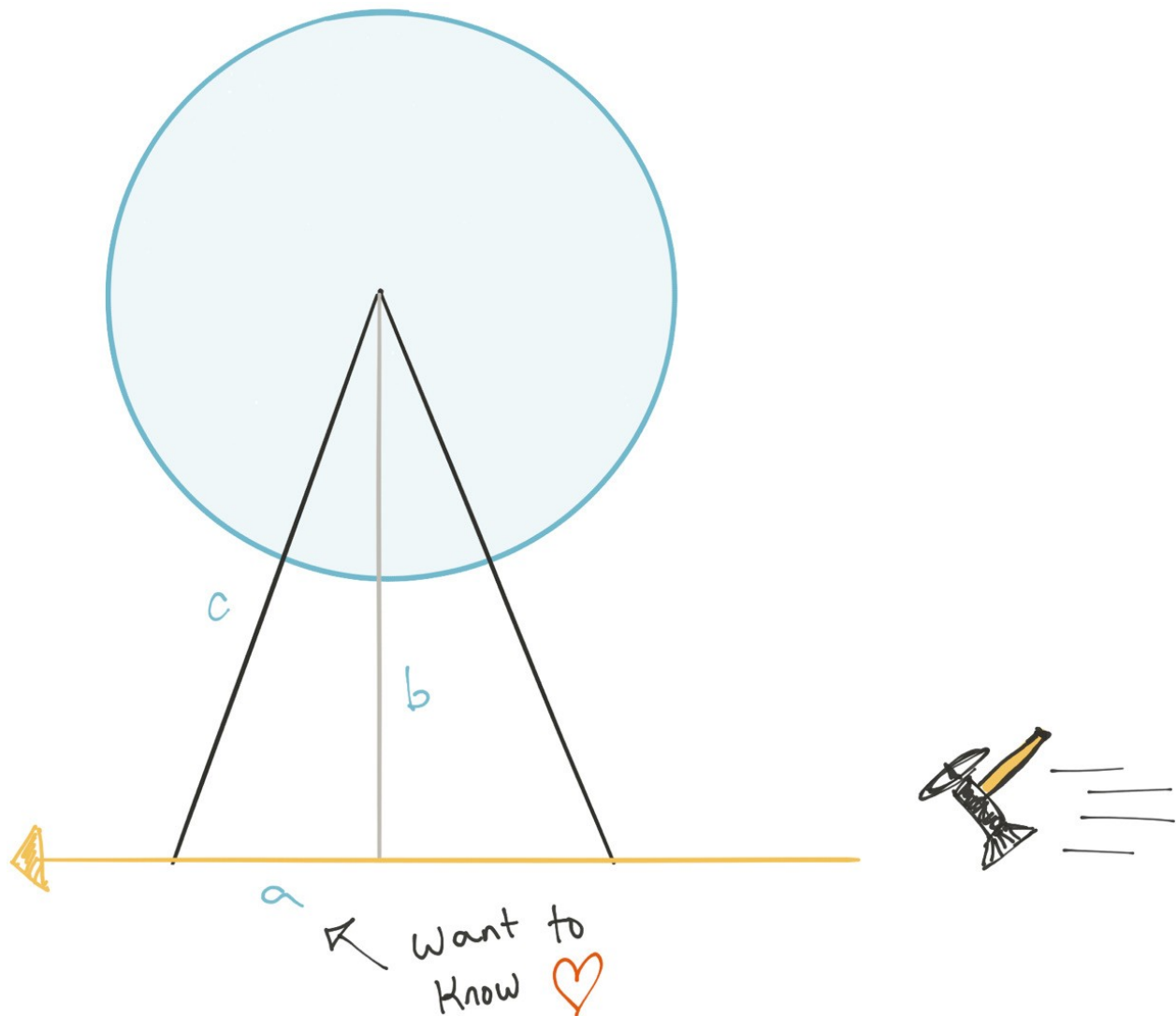
Right, time to focus.

The entire day has been spent Googling basic trigonometry and coming up with a mathematical way to calculate Cassini's speed entirely based off of the INMS altitude data, and I think I'm on to something.

Here's what I know:

- The angle of the analysis window is 34 degrees
- The radius of Enceladus is 252 km
- The altitude of every flyby

The analysis window is an isosceles triangle, which means I can cut it in half and have 2 right triangles:

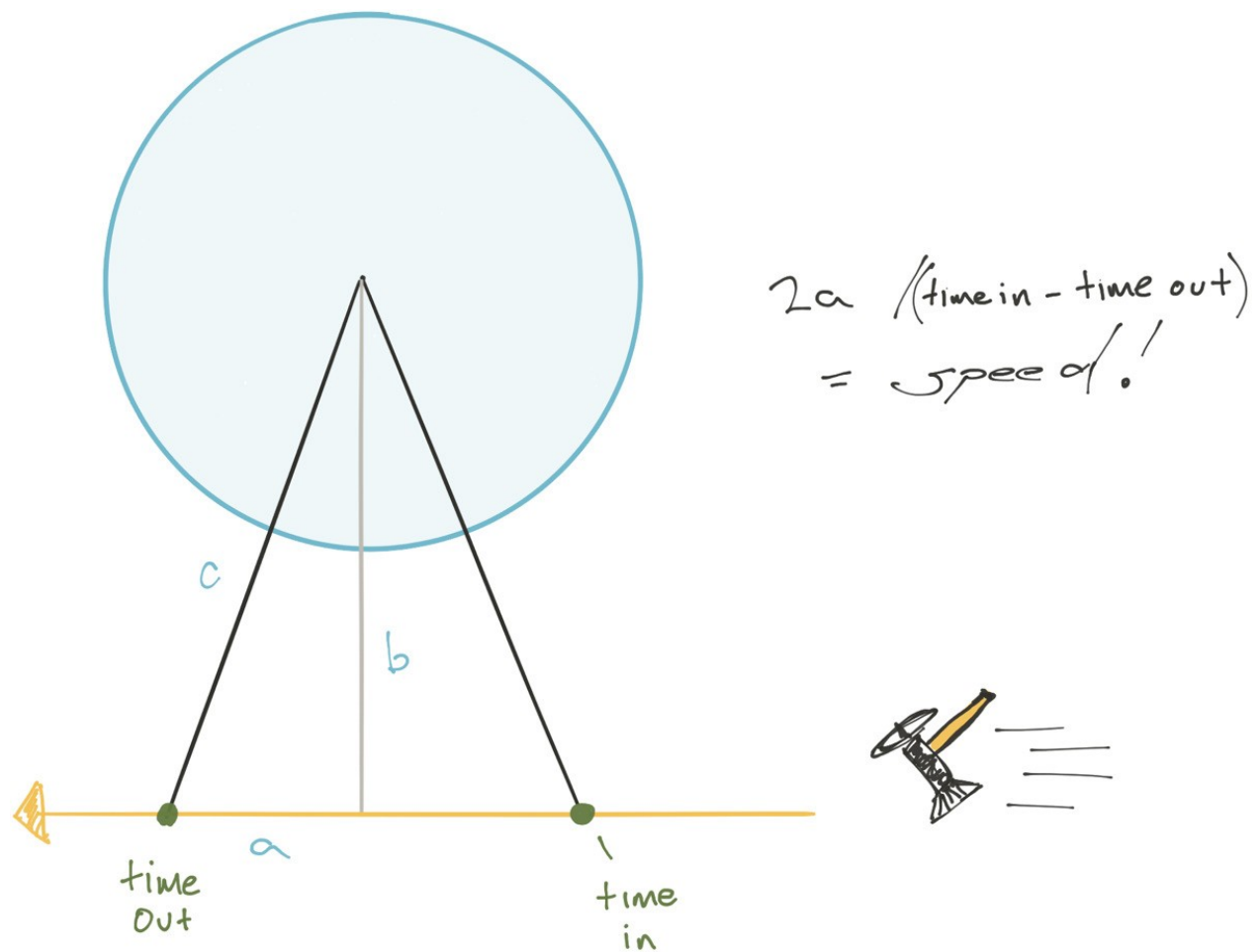


The first thing I want to know is a , which is the distance traveled orthogonal to Enceladus. If I double it, then I have the entire distance traveled through the analysis window.

The next step is to figure out c . If I can calculate c , then I can subtract the radius of Enceladus and figure out the exact altitude point where Cassini exits the analysis window.

Next, if I know the exact altitude points, where Cassini enters and then exits the analysis window, then I can go back to my

flyby_elevations view and find the timestamp associated with this flyby, for the calculated elevation:



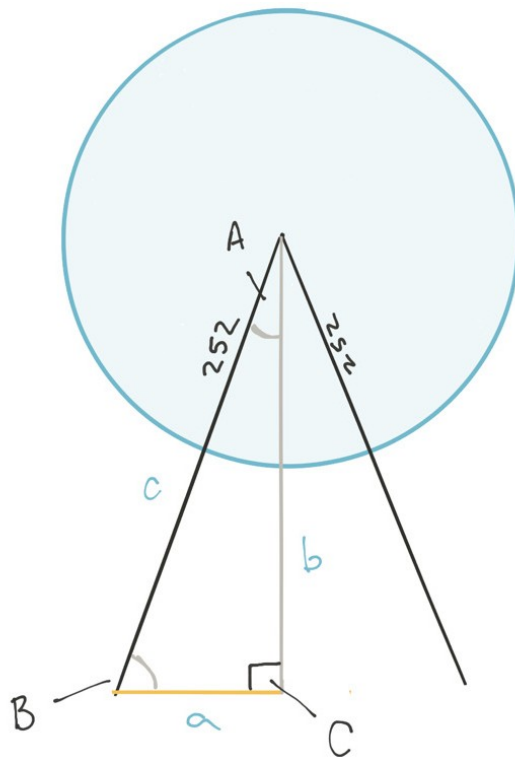
If I know both when Cassini entered and exited the analysis window, I can subtract those times and get an interval, let's say it's something like 45 seconds. I can use that interval with the total distance in the analysis window to derive a speed!

Now I just need to do some math. Thankfully it's pretty simple as I can use the sine rule:

$$a / \sin A = b / \sin B = c / \sin C$$

In this equation, A, B and C are the opposing angles for each side. We know that C is 90 degrees and we also know that A is 17 degrees because it's half of 34 (the angle of the analysis window).

$$\begin{aligned} C &= 90 \\ A &= 17 \\ B &= 73 \end{aligned}$$



I can see if this works by plugging in some numbers. According to the altitudes that I stored in the **flybys** table, our very first flyby had an altitude of 1272.075 km.

That means that b is $1272.075 + 252 = 1524.075$. We can use this value to calculate c because $c / \sin C = b / \sin B$. Given that C is a right angle, that means that sin C is simply 1 so we can simplify the equation:

$$c = 1524.075 / \sin(73) = 1594.22 \text{ km}$$

That's it! Well, sort of. To get the altitude for the entry and exit I need to subtract the radius of Enceladus (252 km), which gives me 1342.22. I can do the same to calculate a:

$$a = c / \sin A = c / \sin(17) = 445.596 \text{ km}$$

Woohoo! I think we have a winner here! Now I just need to multiply that times 2 to get the full flyby distance, which is

891.192 km. BOOM.

Now I just need to plug these equations into some SQL, and I should have some decent working numbers.

CALCULATING THE ANALYSIS WINDOW

I wrestled with this all day, starting with a set of groovy functions and then moving on to just storing the values directly in columns I added to the flybys table. The reason I did this is that:

1. It's historical data and won't change
2. We'll be querying it a lot, and a stored value is much faster than a calculated one
3. I can make the calculations with a single script

I don't know if M. Sullivan will be all climbing up in this but... let him. I feel good about this choice.

The first step is to alter the **flybys** table:

```
-- make room for new stuff
alter table flybys
add speed_kms numeric(10,3),
add target_altitude numeric(10,3),
add transit_distance numeric(10,3);
```

Simple enough. Now I need to do the calculations, so I'll go slowly, step by step.

The first thing is to calculate *b*, which I can do by adding the nadir of the flyby to Enceladus's radius:

```
-- calculate b
select id, altitude,
       (altitude + 252) as total_altitude --b
from flybys;
```

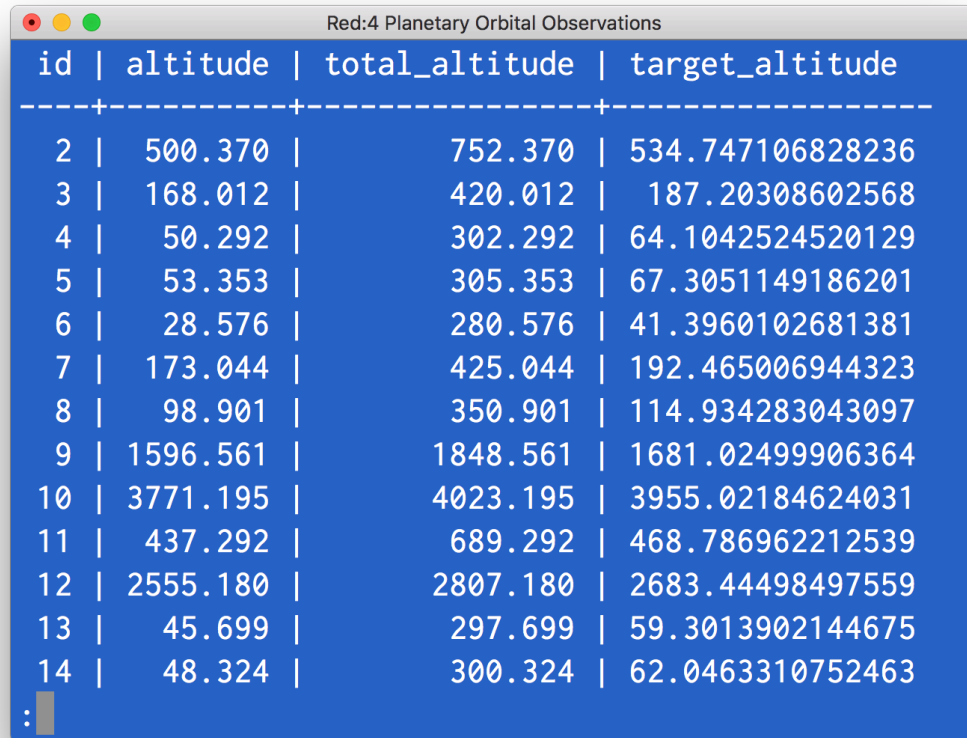
This looks correct:

Red:4 Planetary Orbital Observations		
id	altitude	total_altitude
2	500.370	752.370
3	168.012	420.012
4	50.292	302.292
5	53.353	305.353
6	28.576	280.576
7	173.044	425.044
8	98.901	350.901
9	1596.561	1848.561
10	3771.195	4023.195
11	437.292	689.292
12	2555.180	2807.180
13	45.699	297.699
14	48.324	300.324

Now I need to calculate c as I did above, which is simply b divided by sin(73):

```
-- calculate b and c
select id, altitude,
       (altitude + 252) as total_altitude, --b
       ((altitude + 252) / sind(73)) - 252 as target_altitude -- c
from flybys;
```

Postgres also has a **sin** function, but it takes radians as an argument instead of degrees. For degrees (which is what I have), I have to use the **sind** function:



id	altitude	total_altitude	target_altitude
2	500.370	752.370	534.747106828236
3	168.012	420.012	187.20308602568
4	50.292	302.292	64.1042524520129
5	53.353	305.353	67.3051149186201
6	28.576	280.576	41.3960102681381
7	173.044	425.044	192.465006944323
8	98.901	350.901	114.934283043097
9	1596.561	1848.561	1681.02499906364
10	3771.195	4023.195	3955.02184624031
11	437.292	689.292	468.786962212539
12	2555.180	2807.180	2683.44498497559
13	45.699	297.699	59.3013902144675
14	48.324	300.324	62.0463310752463

That looks correct as well. Our test altitude (with an ID of 1) differs from the test I ran because I rounded to 2 decimal places; I trust this one more.

Now that I know the calculations are correct, I can update the **flybys** table:

```
-- updating flybys with speed data  
update flybys  
set target_altitude=(  
  (altitude + 252) / sind(73)  
) - 252;
```

Yes!

That's c calculated, which means I can now do a, which is my transit distance (**c * sin(17) * 2**):

```
Red:4 Planetary Orbital Observations
enceladus=# update flybys
enceladus=# set target_altitude=(
enceladus(#    (altitude + 252) / sind(73)
enceladus(# ) - 252;
UPDATE 23
enceladus=# update flybys
enceladus=# set
enceladus=# transit_distance = (
enceladus(#    (target_altitude + 252) * sind(17) * 2
enceladus(# );
UPDATE 23
enceladus=#
```

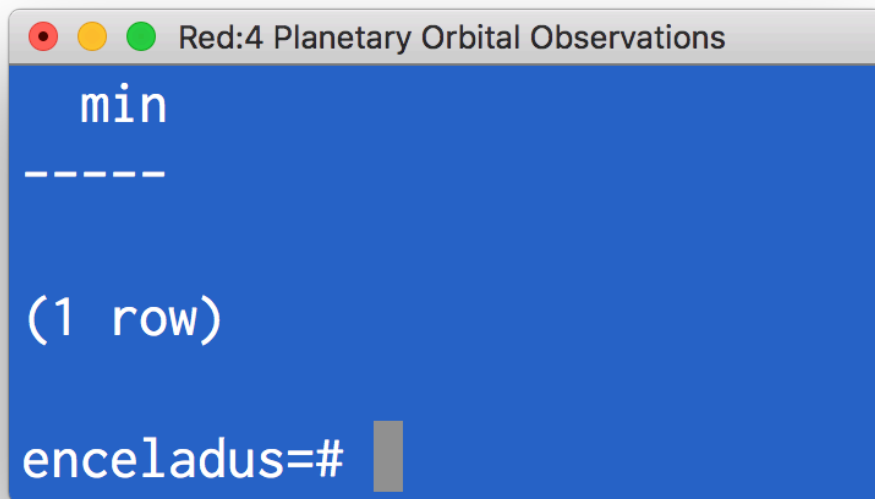
Yeah! I'm on a roll!

Now I can query the **flyby_altitudes** view, using the **target_altitude** so I can figure out my start and end times for crossing the analysis window, making sure to restrict by date. This is where I ran into some trouble.

Here's my query:

```
-- getting the timestamps
select min(flyby_altitudes.time_stamp)
from flyby_altitudes
inner join flybys on flybys.time_stamp::date
    = flyby_altitudes.time_stamp::date
and flybys.target_altitude = flyby_altitudes.altitude
```

I'm using the **min** function to grab the first flyby; I'll use **max** to get the last. This is straightforward, or at least I thought it should be... here are my results:



Nothing! This caused me to tailspin for a few hours, trying to figure out how the altitude data could possibly differ from what's in the **flyby_altitudes** view! That's where it came from!

Then I remembered: *duh, Dee!* It's calculated! There is a super low chance that the computed value will exactly match what's in my view!

This blows everything up. I have the exact altitude for the flyby, I just can't pull the timestamp out, and that's what I need! Time for a break.

Doing trigonometry once again reminds me of college. I enjoyed college, but getting through it was the most difficult thing I've ever done. A beautiful school in a beautiful seaside city full of beautiful people: life in Southern California is pretty great. Sounds horribly difficult, doesn't it? Well...

I moved down right out of high school with my best friend Kelly. I've known her since I was 7. We met at Lilienthal Elementary and became inseparable during our teen years at Marina Middle and then Galileo.

Kelly was accepted to UC Santa Barbara right out of high school. Me? For some reason studying and hard work weren't my thing back then, and my grades suffered for it. I decided to go to Santa Barbara anyway because I could go to the City College to get my general education requirement out of the way and spend a lot less money. *Akamai!*

I loved this newfound life, and I decided to embrace my dreams, foregoing any practical considerations, and major in Astrophysics. I loved science, anything to do with space, and everything else was just so boring! There was no way I was going to end up in an office, and absolutely no way I was going to spend my life staring at a computer all day.

Life was coming together for me and I loved to think about my future while sitting at the beach in the sun – warm sun! This is also where I liked to study, and that's exactly what I was doing (reading *Don Quixote*) when my mom called me, voice shaking, gasping for air.

My dad had suddenly died.

Something about a heart attack, some stairs, a missing shoe and... I don't remember the rest aside from loading my stuff in the car and driving home, forgetting my slippers on the sand. I have no idea how I made that drive; I can't remember a single thing about it aside from stopping for gas and getting sick in the gross black plastic trashcan mounted next to the pump.

I have never felt such vacant pain. I didn't even know that kind of feeling was possible.

A few weeks later I moved home, leaving a year and a half of college behind. I just ... couldn't do ... *anything* related to school. My mom and I did our best to comfort each other and to heal, which we did, as everyone eventually does. *Or so I thought.*

After six months she urged me to go back and to pick up my life. I resisted at first but soon grew bored of being back in San Francisco. I missed Santa Barbara, my new life and the sunny beaches.

I moved back down in the summer, enrolling in the summer program so I could transfer to UCSB for the following fall quarter. Surprisingly, they accepted my transfer and I declared my major as Computer Science. I just couldn't handle thinking about stars and space anymore. Those dreams were gone with my dad.

Kelly was in her final year of her degree (Biology) and had a room open up at her house on the 6700 block of Del Playa, in the student town of Isla Vista right next to campus, and that's where I lived for the next three years.

The first year at UCSB was incredibly hard and I ended up dropping a number of my classes as I just couldn't concentrate. Nothing in life seemed to matter and every class I had seemed totally pointless. I was getting absolutely nowhere and wasting a lot of money and time. I needed to rethink my life and what I wanted to do with it. College, it seemed, was just not for me.

I was about to move home once again when I had a nightmare of a conversation with my mom. I was absolutely furious with her and we didn't speak for weeks, but I look back on that conversation now as a turning point in my life.

“I won’t hear this from you Dee,” she said to me. “I won’t hear your reasons for failing as if they’re... somehow reasonable. Nobody decides to fail, they just fail. Everything else is *bullshit*,” she said.

My mom never swore, and hearing her use that word jarred me. “MOM! You don’t know what it’s like down here! I...” and cue the litany of excuses which I just can’t write down. It’s too painful to remember the excessive whining. My mom, thankfully, cut me off.

“Stop. Just STOP. You’re making me sad right now. I cannot and will not take on your problems. Your father gave so much of his life to you that—”

“Mom *don’t*...” I said, starting to sob.

“No. You must hear this! Your father loved you so much, Dee, so so so so much. You have to hear this right now,” she said, pausing so I could stop sobbing and hear her. “You would *disgrace* him and his memory if you gave up. I will not allow that,” she hissed. “Time to grow *up*! Do *not* come back here until you have!”

The line went dead. Oh my god! My mom *hung up* on me!

Kelly came rushing into my room as I screamed profanity at the walls, cursing my mom, pounding my fists on the bed. She held me as I raged, then sobbed some more, raged again, sobbed, and finally grew quiet.

“Let’s go get a beer,” she whispered.

We ended up at Woodstock’s and had two pitchers of Sierra Nevada and a Hawaiian Pizza over the course of the next three hours. I don’t drink that much, but right then it was the perfect thing. It numbed my sense of betrayal and rage and, somehow, gave way to something that felt like... *joy*?

“That’s freedom, Dee,” Kelly said. “I think your mom just kind of kicked you in the ass and, since I’m your best friend with a few beers in me, I think it’s OK for me to also say that you *needed* it,” she continued, staring at me with a tenuous resolve.

“Dude, don’t—” I started.

“*Look* at me. You need to get your shit together right now. Your mom didn’t betray you or say anything horrible or mean. She’s telling you that it’s time to stop with the excuses and start kicking some ass,” she said. “Be the Dee that we all know is in there.”

“Oh *please*,” I said, getting annoyed by the cliché. The joy was starting to fade back into anger, and all I could do was shake my head. How could she do this to me, right now?

“I can see what you’re thinking: *how can she do this to me?* You’ve always been more than a little self-centered Dee, and *dammit!*” she said, slapping the table probably a little too hard, “you’re 21 years old and want to go back and live with your mom because you can’t hack this amazing opportunity at a school located in paradise. Seriously? Tell me you see how pathetic that is!”

I suppose that is what I love about Kelly and why we’re still friends: she’ll just say it. Whatever *it* is, no matter how much it hurts. Which it did, *a lot*.

She grabbed my hand and held it. I tried to pull away but she wouldn’t let me. I guess I didn’t try too hard either. “Stop hating, Dee, and let the walls fall. See this for what it is, without your internal drama fucking it all up: two people that love you dearly, trying to help you move on with your life. To let go, and to free yourself. Your dad’s *gone*, Dee, it’s time to move on,” she said, tears rolling down her cheeks.

Her words were a kick in the head, stunning me, detaching reality a bit. Honesty like this from good friends is a mixed blessing: you can’t tell them they don’t know what they’re talking about, because they’re talking about *you*. They *know* you.

I told her I needed a few minutes to take a walk and promised I’d be back. I needed to get outside. I made my way to Embarcadero del Norte then down to the bluffs at Pelican Beach where I could see the last bits of the sunset. Two other people were there at the lookout, stubbing out cigarettes, about to leave, when I asked if I could bum one from them. I don’t smoke, but *sometimes*...

Staring at the remains of the setting sun, I simply could not ignore what the two people closest to me were telling me: *grow up*. They were right, and it stung.

The ocean breeze turned cool as the sun went below the horizon.

I took the last drag on my cigarette and blew it out slowly.

Fuck it. *Time to grow up.*

CALCULATING THE ANALYSIS WINDOW, TRY 2

November 1, 2017, 1631

I walked over to The Embarcadero and sat in the sun. It's so nice to be on this side of the bay, where it's a little less windy and usually a bit sunnier. I watched the fireboat bob back and forth in the waves thrown off by the approaching ferry, trying to come up with a reasonable strategy.

I don't like the solution I came up with. It feels sloppy to me. If I think it's sloppy, M. Sullivan is going to tear it apart.

If can, can.

Here's what I came up with.

I decided to restrict the altitudes using a range of values instead of equality. I used the subquery inside an update, which is kind of slow but, well, like I said, it works. Here's the start time:

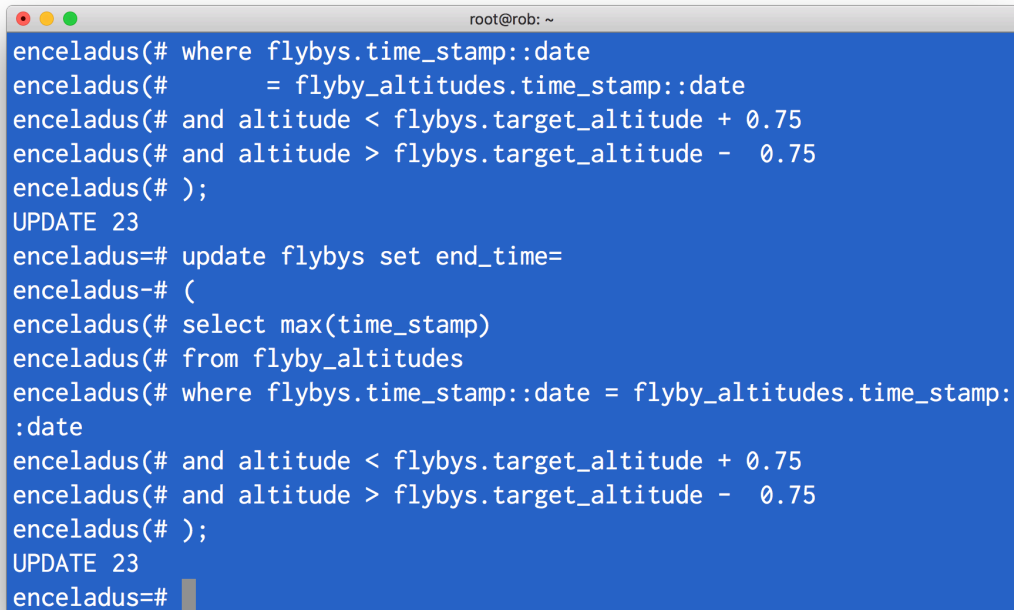
```
-- the hard way
update flybys set start_time=
(
  select min(time_stamp)
  from flyby_altitudes
  where flybys.time_stamp::date
    = flyby_altitudes.time_stamp::date
  and altitude < flybys.target_altitude + 0.75
  and altitude > flybys.target_altitude - 0.75
);
```

And then the end time:

```
update flybys set end_time=
(
  select max(time_stamp)
  from flyby_altitudes
  where flybys.time_stamp::date
    = flyby_altitudes.time_stamp::date
  and altitude < flybys.target_altitude + 0.75
  and altitude > flybys.target_altitude - 0.75
);
```

One query each for the start and end times. It's more SQL, but I think it's actually easier to read than it would be if I tried to do it all at once. Maybe I can get some advice from M. Sullivan if there's a better way to do it.

The good news is that each update query only needs to run once. They're not terribly fast (taking about 40 seconds each), but I suppose that's not a big deal:

A terminal window with a blue background and white text. The window title is 'root@rob: ~'. It contains a series of SQL queries and comments for updating a database. The queries are: 1. A WHERE clause for flybys within a time window and altitude range. 2. An UPDATE statement for the flybys table. 3. A subquery to find the maximum time stamp from the flyby_altitudes table. 4. An UPDATE statement to set the end_time of flybys to the maximum time stamp found. The terminal shows the prompt 'enceladus=#' and the output 'UPDATE 23' for the first query.

```
root@rob: ~
enceladus(# where flybys.time_stamp::date
enceladus(#         = flyby_altitudes.time_stamp::date
enceladus(# and altitude < flybys.target_altitude + 0.75
enceladus(# and altitude > flybys.target_altitude - 0.75
enceladus(# );
UPDATE 23
enceladus=# update flybys set end_time=
enceladus-# (
enceladus(# select max(time_stamp)
enceladus(# from flyby_altitudes
enceladus(# where flybys.time_stamp::date = flyby_altitudes.time_stamp:
:date
enceladus(# and altitude < flybys.target_altitude + 0.75
enceladus(# and altitude > flybys.target_altitude - 0.75
enceladus(# );
UPDATE 23
enceladus=#
```

Phew! The start and end time is set for the analysis window!

Hey, wait a minute...

id	name	start_time	end_time
1	E-0		
2	E-1	2005-03-09 17:07:28.048-08	2005-03-09 17:08:38.766-08
3	E-2	2005-07-15 02:55:06.387-07	2005-07-15 02:55:38.168-07
4	E-3	2008-03-13 02:06:05.091-07	2008-03-13 02:06:17.99-07
5	E-4	2008-08-12 04:06:13.26-07	2008-08-12 04:06:23.905-07
6	E-5	2008-10-10 02:06:34.836-07	2008-10-10 02:06:44.616-07
7	E-6	2008-11-01 00:14:44.045-07	2008-11-01 00:14:58.827-07
8	E-7	2009-11-02 15:41:43.312-08	2009-11-02 15:42:12.104-08
9	E-8	2009-11-21 10:08:42.929-08	2009-11-21 10:11:09.844-08
10	E-9	2010-04-28 06:59:32.683-07	2010-04-28 06:59:32.887-07
11	E-10	2010-05-18 13:04:07.419-07	2010-05-18 13:05:13.135-07
12	E-11	2010-08-14 05:28:45.979-07	2010-08-14 05:32:57.909-07
13	E-12	2010-11-30 19:53:43.943-08	2010-11-30 19:54:14.195-08
14	E-13	2010-12-21 09:08:11.78-08	2010-12-21 09:08:42.469-08
15	E-14	2011-10-01 20:52:10.74-07	2011-10-01 20:52:40.651-07
16	E-15	2011-10-19 16:21:10.227-07	2011-10-19 16:23:12.271-07

Why... would there be no data for the first flyby? That doesn't make any sense!

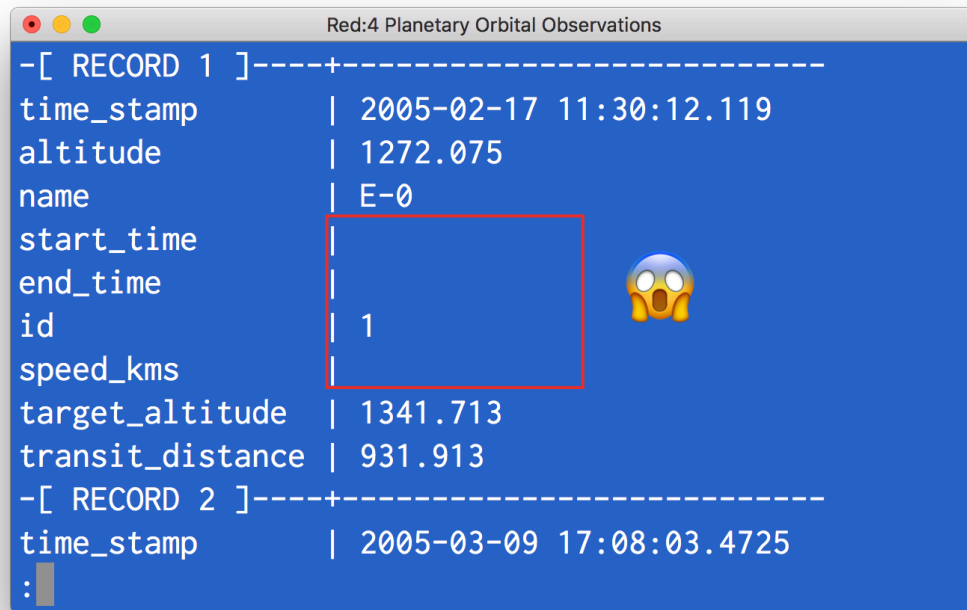
This whole thing feels completely wrong. I'm at the point where I'm merely running more and more SQL, trying to fix one problem after the next. I think I need to stop.

IF NO CAN, NO CAN

It's so weird how impending failure has a feeling to it. Things start to go wrong, you begin compensating and writing more code, rationalizing your way out of one problem into the next. When I saw that row with missing timings, I went into a bit of a tailspin. That information *had* to be there! I just needed to perform the correct incantations to summon it into the **flybys** table!

Seems odd to suppose that data and code can make you feel a certain way. Ultimately, it's just symbols and text. It's what you do with those symbols and text, I guess, that makes you check yourself. I needed some self-checking for damn sure.

The upshot to all this is that I've just wasted 3 days on data that is incomplete and, well, *wrong*.

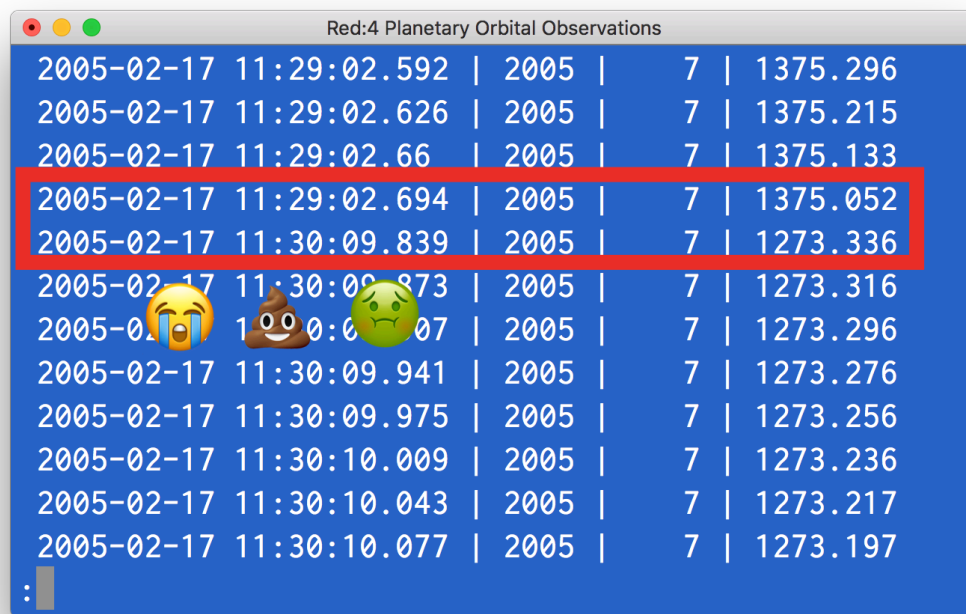


```
Red:4 Planetary Orbital Observations
-[ RECORD 1 ]-----+-----
time_stamp    | 2005-02-17 11:30:12.119
altitude      | 1272.075
name          | E-0
start_time    | 
end_time      | 
id            | 1
speed_kms     | 
target_altitude | 1341.713
transit_distance | 931.913
-[ RECORD 2 ]-----+-----
time_stamp    | 2005-03-09 17:08:03.4725
:
```

The altitude is there – I can see it. It's not calculated which means I pulled it from the **flyby_altitudes** view.

```
select * from flyby_altitudes
where time_stamp::date = '2005-02-17'
and altitude between 1200 and 1500
order by time_stamp;
```

There were 272 rows. I needed to find the rows right around the **target_altitude** that I'm looking for, which is 1341.713:



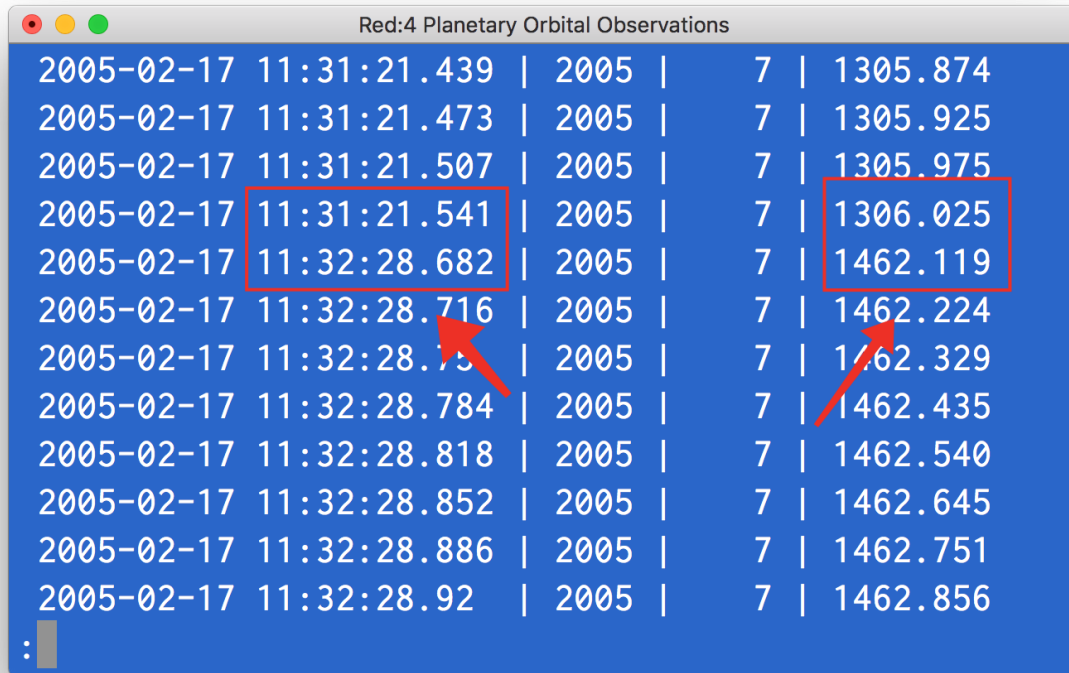
2005-02-17 11:29:02.592		2005		7		1375.296
2005-02-17 11:29:02.626		2005		7		1375.215
2005-02-17 11:29:02.66		2005		7		1375.133
2005-02-17 11:29:02.694		2005		7		1375.052
2005-02-17 11:30:09.839		2005		7		1273.336
2005-02-17 11:30:09.873		2005		7		1273.316
2005-02-17 11:30:09.907		2005		7		1273.296
2005-02-17 11:30:09.941		2005		7		1273.276
2005-02-17 11:30:09.975		2005		7		1273.256
2005-02-17 11:30:10.009		2005		7		1273.236
2005-02-17 11:30:10.043		2005		7		1273.217
2005-02-17 11:30:10.077		2005		7		1273.197

It's not there. The readings jump from 1375.052 down to 1273.336. If you keep scrolling down, you eventually hit the nadir, which is correct, and the altitude begins to go up again.

Where the hell are the readings between 1273 and 1375 km?

Did the INMS just turn off or something?

More pain:



2005-02-17	11:31:21.439		2005		7		1305.874
2005-02-17	11:31:21.473		2005		7		1305.925
2005-02-17	11:31:21.507		2005		7		1305.975
2005-02-17	11:31:21.541		2005		7		1306.025
2005-02-17	11:32:28.682		2005		7		1462.119
2005-02-17	11:32:28.716		2005		7		1462.224
2005-02-17	11:32:28.750		2005		7		1462.329
2005-02-17	11:32:28.784		2005		7		1462.435
2005-02-17	11:32:28.818		2005		7		1462.540
2005-02-17	11:32:28.852		2005		7		1462.645
2005-02-17	11:32:28.886		2005		7		1462.751
2005-02-17	11:32:28.92		2005		7		1462.856

There is a 1-second jump, and in that time Cassini gained 156 or so km!

The only explanations for this are:

1. The INMS wasn't on the whole time, and I can't rely on it to have usable intervals
2. It resets itself
3. It randomly loses data
4. Someone edited this file

Using Occam's Razor, I have to think the INMS wasn't on during certain random times. This means I can't make assumptions about... well, anything I guess. It also says everything I've been doing has been useless.

At this point, things only got worse. It was all starting to spiral, and so I finally decided to ask for help.

From: M. Sullivan sullz@redfour.io
Subject: RE: Crap sandwich filled with crap
To: Dee Yan yand@redfour.io
Date: November 1, 2017

Looking at the logs (and your **flybys** table) right now, yes, I see what you mean. There is indeed a problem, as you say. However, I don't think it's with the data, I think it's with your *assumptions*. Data is just data, M. Yan.

As a data person, your job is to make sure the data is *correct*. When you go beyond that mandate, that is when you find trouble. Of course, there is no way that we can remain completely objective when it comes to normalizing a database and working with the data: we have to make some choices. Those choices, however, will always reflect some kind of *bias*. It's the nature of human beings: we look for patterns so we can understand what it is we're looking at. Those patterns are formed based on how we see the world, which is the core of bias.

If the patterns that we see don't make sense, sometimes we *make them make sense*, even if it means inventing things, or worse, removing the data that contradicts our point of view. A person's capacity to deceive themselves is astounding. Work with spreadsheets long enough, and you'll come to see what I mean.

Take some time to reflect on the choices you've made to get to the point where you're at. You thought you saw something in the data (consistent timestamping with altitude

measurements) *yet it wasn't there*. This has backed you into a corner. I very much appreciate that you stopped to ask for help. Many developers would plow forward, filling in the bits of data that weren't there just to satisfy their need for things to make sense.

Looking at the **flybys** table one more time, I can see how tempting it must have been to look at the missing data points and think: "oh well, I'll just fill these dates in by hand; the INMS hiccupped." This would be incorrect and against everything we're trying to do here, which is to *let the data tell the story*, not *you*, M. Yan. If you find yourself filling in the gaps, the story becomes yours, and the data becomes fictional.

As to how I would "solve this crappy problem," *I wouldn't*. The data is trying to tell you something, and it's something that's hard to see. It's trying to tell you that *it's not there*. If it's not there, you need to look elsewhere for what you need, and you very much need to learn how to let go of whatever story you were telling yourself about what's in the INMS data set. Perhaps start at the NASA/JPL website for the Cassini missions, or look over Google one more time. Maybe there is a data set you haven't considered yet.

The truth can hurt, M. Yan. It's something to embrace, however, because pain is an excellent teacher.

Best, S

The only good thing about right now is the time: 1731. Time to roll over to Thirsty Bear and meet Kelly for a good, stiff bourbon. I feel so defeated; there's no way I'm going to get this job.

Failure is the best teacher, Dee, Success is the worst.

I guess my mom was right. Of course, she was right. I was kicking ass, or so I thought, and it blinded me to the obvious: the data just wasn't there. M. Sullivan was

also right about my filling in those missing dates: I was just about to do just that. I feel like I could have defended that position; I also feel like it'd have gotten me fired.

I suppose the trigonometry should have tipped me off. Nice equations, Dee.

Stupid. *Akamai*, dammit! I am so pissed off *frustrated* right now!

SSFDJROO\$+EIJFPKNEFPKJDFP!!!

From: Sara C. boss@redfour.io
Subject: Any updates for me?
To: Dee Yan yand@redfour.io
Date: November 2, 2017

Hello again Dee, a little surprised that I haven't heard back from you yet. Would love to know how you're coming along with the SELF data, and if you've interfaced with the Analysis Team yet.

I would appreciate hearing back both today and in the future, at regular intervals.

Regards, Sara

From: Eno L. eno@redfour.io
Subject: Flyby speeds and altitudes
To: Dee Yan yand@redfour.io
Date: November 2, 2017

Hey Dee! M. Sullivan sent me a note last night asking me if I might have come across an authoritative data source for Cassini's flyby speeds and altitudes. He mentioned that

you have been working with the INMS data and that it has some gaps? That should still be OK – but it might require some calculus to get right.

What you would need to do is to dig through the altitude data and find *mirror points*, two altitude recordings that are identical. From what I understand, the INMS recorded its altitude every 30 to 35 milliseconds down to a scale of 3 decimal points, which makes the likelihood of finding a mirror match for every flyby almost 0.

At this point, I could write a few more pages about some interesting math you could perform, or I could just suggest heading over to the [Flybys Page](#) on the Cassini website, which is run by NASA/JPL. Each flyby is documented there, along with speed and altitude in most cases. It's NASA/JPL, so I think you can consider it authoritative. We're good with it over here if it makes you feel better.

Some flybys don't have speed and altitude data, which is OK – see what you can gather, and we'll go from there.

Good luck!

E

REVISION: DOING IT BY HAND

November 2nd, 2017 0833

I wish this job wasn't so stressful. The "all-in-lets-panic-or-die" startup mentality is something I've successfully avoided over the last few years. It's why I'm here. I can't keep living at mom's, however, so if this job doesn't pan out... damn.

I guess every job has its crunch times. Maybe I'm also feeling more than a bit defeated.

The good news is that the warm weather has stayed around and it was actually sunny today! I decided to ride my bike to the office in the morning, along the Marina Green, through the winding path at Fort Mason, the Maritime Museum and then down into Fisherman's Wharf. I really don't like that neighborhood. It's so touristy. In the morning, however, when there's no one around you can get a delicious breakfast at Cioppino's. The coffee is soso, but on a day like today, they let you sit outside.

The bright morning air of the Embarcadero was perfect. Riding along the water, watching the ferries glide in and out... the wide-open space of the bay catches my soul as the glass monolithic skyscrapers of the Financial District push it to the edge of existence.

The note from Eno really helped and... yeah, I feel stupid. I looked it over and yeah, if this is an authoritative source, then everything about my altitude calculations has been 100% incorrect. I didn't have a single thing right!

I created a CSV from the data on the website and sent it over to Eno to double check for me since she offered. I created a table for the new data and imported it like I've been doing:


```
drop table if exists flybys;
create table flybys(
  id int primary key,
  name text not null,
  date date not null,
  altitude numeric(7,1),
  speed numeric(7,1)
);

copy flybys
from '[PATH]/data/jpl_flybys.csv' -- in the data directory
delimiter ',' header csv;
```

Looks much better:

Red:4 Planetary Orbital Observations				
id	name	date	altitude	speed
1	E-0	2005-02-17		
2	E-1	2005-03-09	504.0	
3	E-2	2005-07-14	172.0	8.2
4	E-3	2008-03-12	52.0	14.4
5	E-4	2008-08-11	50.0	17.7
6	E-5	2008-10-09	25.0	17.7
7	E-6	2008-10-31	197.0	17.7
8	E-7	2009-11-02	103.0	7.7
9	E-8	2009-11-21	1606.0	7.7
10	E-9	2010-04-28	100.0	6.5
11	E-10	2010-05-18	438.0	6.5
12	E-11	2010-08-13	2502.0	6.8
13	E-12	2010-11-30	47.9	6.3
14	E-13	2010-12-21	47.8	6.2
15	E-14	2011-10-01	99.0	7.4
16	E-15	2011-10-19	1231.0	7.4
17	E-16	2011-11-06	496.0	7.4
18	E-17	2012-03-27	74.0	7.5
19	E-18	2012-04-14	74.0	7.5
20	E-19	2012-05-02	74.0	7.5
21	E-20	2015-10-14	1839.0	8.5
22	E-21	2015-10-28	49.0	8.5
23	E-22	2015-12-19	4999.0	9.5

I guess I could have picked a better name for **date**, but... it's the date so... fight me. I don't have exact times, so a timestamp doesn't make sense. I'm also leaving the altitude and speed as null if they're not there; if the NASA/JPL website is an authoritative source and they didn't provide this data, then I'm not going to store it.

Looking over the altitudes... wow, my calculations were way off. The only thing I can figure is that they turned the INMS off at lowest approach so as not to saturate it. Makes sense, I suppose. They did the same for the CDA – flipping over to the HRD (high rate detector) in “dust-rich zones.”

I'm happy that we have reliable altitude and speed data, but I have no idea what this means for the analysis window. Hopefully, Eno can give me some details. I'll wait for her answer before I let Rob know.

Oh, right, and Sara. Evidently, Rob and Sara don't share information? Maybe she's just a control freak that needs to make sure I know she's the boss or something.

Office politics, always, everywhere.

From: Eno L. eno@redfour.io
Subject: RE: Check this for me? And what about the AW?
To: Dee Yan yand@redfour.io
Date: November 2, 2017

I checked the numbers you sent me with those on the site, and they match perfectly; good work. Bummer there's no speed data for the first two flybys, but that's OK, they aren't our focus anyway. I suppose I should tell you a bit more.

We're trying to model the concentrations of specific molecules. As you know, Cassini's INMS and CDA picked up traces of molecular hydrogen as well as a mess of organics, like methane, carbon monoxide and carbon dioxide.

We're supposed to be calculating "zones of confidence" within the Tiger Stripes themselves, finding out if some sulci are more active than others, or if some emit more molecular hydrogen than others.

Given this, the only flybys that we're truly concerned with are the ones that came the closest to the sulci, with the specific goal of analyzing their chemistry. These are, specifically (with a description added from the [NASA/JPL site](#)):

E-3: *NASA's Cassini spacecraft tasted and sampled a surprising organic brew erupting in geyser-like fashion from Saturn's moon Enceladus during a close flyby on March 12. Scientists are amazed that this tiny moon is so active, "hot" and brimming with water vapor and organic chemicals.*

E-5: *During Cassini's Oct. 9 flyby, the spacecraft's fields and particles instruments will venture deeper into the plume than ever before, directly sampling the particles and gases. The emphasis here is on the composition of the plume rather than imaging the surface.*

E-7: *With E7 being the first direct flyby through the Enceladus plume, and at a lower velocity than earlier inclined flybys, the Ion and Neutral Mass Spectrometer (INMS) was able to achieve very high signal-to-noise measurements of the gases in the plume, and looked for structure in the gases associated with the jets seen by ISS, UVIS and the Cosmic Dust Analyzer (CDA). Measurements should provide high enough signal-to-noise to allow INMS to discriminate between N₂ and CO; distinguishing between these two species has been difficult in the past.*

E-17: *This 46-mile (74-kilometer) close Enceladus flyby puts the fields, particles and waves instruments in the driver's seat. There they will use the rare opportunity of being in the moon's backyard to directly sample the moon's plumes, analyzing them to characterize their composition, density and even variability. Combined with E-14 and E-18, the moon's South polar regions will be well covered.*

E-18: *The Fields, Particles, and Waves instruments were prime (particularly the ion and neutral mass spectrometer [INMS]) to study the composition, density, three-*

dimensional structure and variability of plumes; with E-14, and E-17, this flyby provided good coverage of south polar regions.

E-21: *This daring flyby will bring the spacecraft within 30 miles (48 kilometers) of the surface of Enceladus' south polar region. The flyby is Cassini's deepest-ever dive into the plume of icy spray that issues from fractures in the south polar region. The encounter will allow Cassini to obtain the most accurate measurements yet of the plume's composition, and new insights into the ocean world beneath the ice.*

If you could add a flag of some kind to these flybys, that would be helpful. We can use that when constructing our model later on.

What we're trying to do is to add geographical meaning to these plume observations – almost like a targeting system – with confidence levels for the presence of molecular hydrogen at specific spots for each sulcus. Some of the team here think that the composition of the plumes differs, that each sulcus might emit its own “brand” of ejecta, if you will.

This is why we needed that analysis window: to pinpoint, as accurately as possible, where the material was coming from. I understand that the data won't support this goal and, therefore, we've come up with a plan B.

Right now, all we have is a date, some altitude data, and a speed. I know that you found some discrepancies in your calculated approach, but I decided (hope you don't mind) to run a few queries myself to crosscheck what the INMS had to say about altitude vs. what NASA had to say.

One of the primary issues that we face with this data is that it was text before being imported, which raises some issues, primarily with dates. The **sclk** column is described in the manifest as the “Spacecraft clock,” or as we call it around here: the **ship clock**. It's specified as UTC time, so I think we're OK.

I used to work on the data team, and I miss writing queries, so I hope you don't mind that I took a swing at pulling some data together? If it looks good to you, we might have something to work with.

First: I know M. Sullivan is a fan of materialized views, but that's a bit overkill for what we're trying to do. In many ways, a temporary table will do the trick! So that's what I did (sort of) with the flyby data that you initially pulled together, using year and week like you did (which was smart). I say "sort of" because you can specifically declare a table as temporary using **create temp table**, but it's less SQL to create a table and then delete it when finished:

```
--using a sort-of temp table
drop table if exists time_altitudes;
select
  (sclk::timestamp) as time_stamp,
  alt_t::numeric(9,2) as altitude,
  date_part('year',(sclk::timestamp)) as year,
  date_part('week',(sclk::timestamp)) week
into time_altitudes
from import.inms
where target='ENCELADUS'
and alt_t IS NOT NULL
```

Good move on **timestamp without time zone**, we don't want Postgres to try to convert anything.

Next, I just did a grouping:

```
select min(altitude) as nadir, year, week
from time_altitudes
group by year, week order by year, week;
```

Now I have all the nadirs grouped by year and week. This is guaranteed to give me the correct data for the flyby, assuming we have it.

Now comes the tricky part: getting the proper timestamp! I like doing the easiest thing, and since we're working with a table, queries will be much faster. I find CTEs to be one of the best things about Postgres, so I wrapped my first query in one, and then did a rollup with a second query:

```
with mins as (  
  select min(altitude) as nadir, year, week  
  from time_altitudes  
  group by year, week  
  order by year, week  
, min_times as (  
  select mins.*, min(time_stamp) as low_time,  
  from mins  
  inner join time_altitudes ta on mins.year = ta.year  
    and mins.week = ta.week  
    and mins.nadir = ta.altitude  
  group by mins.week, mins.year, mins.nadir  
)
```

The fun part is that I was able to join my temp table on the second query. The temp table has the timestamps I want, so all I needed to do was to ask for the lowest one for that week/year/ altitude.

It worked really well, and it was reasonably fast.

The final thing was to pull your redone **flybys** table in, and “augment it” with the information in the second query (exact nadir and altitude).

Before I get to that, I wanted to mention that I talked to the rest of the team and we think that a 40-second analysis window, using this data in combination with what you just created, should work for what we need. We don't need it for all the flybys, only the ones I outlined above.

It was simply enough to just add that in, so I did. Here's the whole query that you can use to create the **flybys** table, 2.0:

```
with mins as (  
  select min(altitude) as nadir, year, week  
  from time_altitudes  
  group by year, week  
  order by year, week  
, min_times as (  
  select mins.*, min(time_stamp) as low_time,  
    min(time_stamp)  
    + interval '20 seconds' as window_end,  
    min(time_stamp)  
    - interval '20 seconds' as window_start  
  from mins  
  inner join time_altitudes ta on mins.year = ta.year  
    and mins.week = ta.week  
    and mins.nadir = ta.altitude  
  group by mins.week, mins.year, mins.nadir  
, fixed_flybys as (  
  select f.id, f.name, f.date, f.altitude, f.speed,  
    mt.nadir, mt.year, mt.week, mt.low_time, mt.window_start, mt.window_end  
  from flybys f  
  inner join min_times mt on  
    date_part('year', f.date) = mt.year and  
    date_part('week', f.date) = mt.week  
)
```


It looks much more complicated than it is, I promise. It runs reasonably fast, too! M. Sullivan shouldn't give you any grief... I hope.

Really appreciate your time on all of this Dee! Hope to meet you someday, and I can show you more about what we're doing. In fact, I talked to Rob a bit about that, and it might be kind of fun if you dropped by the Presidio some time. We're at 31 Funston, suite D.

Ha! "Suite D," seems like you have to stop by now. Best,

E

PS – don't forget to drop your temp tables after everything runs right!

PPS – I forgot to tell you that I found a PDF online from the University of Michigan, written by David Gell called [The INMS Analysis Library](#). In it, he has a tutorial that describes the information in more detail. Hope you find it helpful!

OMG OMG OMG OMG!

November 2, 2017, 1743

I think this is entirely doable! Eno is amazing! I have to go over there next week and take her to lunch!

To create the new **flybys** table I just had to:

```
-- create the table from the CTE
select * into flybys_2
from fixed_flybys
order by date;

-- add a primary key
alter table flybys_2
add primary key (id);

-- drop the flybys table
drop table flybys cascade;
drop table time_altitudes;

-- rename flybys_2
alter table flybys_2
rename to flybys;
```

Super squeezy easy! I want to be sure we know which ones we're targeting, so I also added a **targeted** field, defaulting it to **false**:

```

-- add a targeted field
alter table flybys
add targeted boolean not null default false;

-- set it
update flybys
set targeted=true
where id in (3,5,7,17,18,21);

```

Look at this gorgeous data!

id	name	date	altitude	speed	nadir	year	week	low_time	window_start	window_end	targeted
1	E-0	2005-02-17			1272.08	2005	7	2005-02-17 03:30:12.119	2005-02-17 03:29:52.119	2005-02-17 03:30:32.119	f
2	E-1	2005-03-09	504.0		500.37	2005	10	2005-03-09 09:08:03.098	2005-03-09 09:07:43.098	2005-03-09 09:08:23.098	f
3	E-2	2005-07-14	172.0	8.2	168.01	2005	28	2005-07-14 19:55:22.143	2005-07-14 19:55:02.143	2005-07-14 19:55:42.143	t
4	E-3	2008-03-12	52.0	14.4	50.29	2008	11	2008-03-12 19:06:11.458	2008-03-12 19:05:51.458	2008-03-12 19:06:31.458	f
5	E-4	2008-08-11	50.0	17.7	53.35	2008	33	2008-08-11 21:06:18.523	2008-08-11 21:05:58.523	2008-08-11 21:06:38.523	t
6	E-5	2008-10-09	25.0	17.7	28.58	2008	41	2008-10-09 19:06:39.605	2008-10-09 19:06:19.605	2008-10-09 19:06:59.605	f
7	E-6	2008-10-31	197.0	17.7	173.04	2008	44	2008-10-31 17:14:51.429	2008-10-31 17:14:31.429	2008-10-31 17:15:11.429	t
8	E-7	2009-11-02	103.0	7.7	98.90	2009	45	2009-11-02 07:41:57.503	2009-11-02 07:41:37.503	2009-11-02 07:42:17.503	f
9	E-8	2009-11-21	1606.0	7.7	1596.56	2009	47	2009-11-21 02:09:55.929	2009-11-21 02:09:35.929	2009-11-21 02:10:15.929	f
10	E-9	2010-04-28	100.0	6.5	3771.20	2010	17	2010-04-28 00:00:01.088	2010-04-27 23:59:41.088	2010-04-28 00:00:21.088	f
11	E-10	2010-05-18	438.0	6.5	437.29	2010	20	2010-05-18 06:04:40.029	2010-05-18 06:04:20.029	2010-05-18 06:05:00.029	f
12	E-11	2010-08-13	2502.0	6.8	2555.18	2010	32	2010-08-13 22:30:51.215	2010-08-13 22:30:31.215	2010-08-13 22:31:11.215	f
13	E-12	2010-11-30	47.9	6.3	45.70	2010	48	2010-11-30 11:53:58.777	2010-11-30 11:53:38.777	2010-11-30 11:54:18.777	f
14	E-13	2010-12-21	47.8	6.2	48.32	2010	51	2010-12-21 01:08:07.061	2010-12-21 01:08:07.061	2010-12-21 01:08:47.061	f
15	E-14	2011-10-01	99.0	7.4	98.90	2011	39	2011-10-01 13:52:05.409	2011-10-01 13:52:05.409	2011-10-01 13:52:45.409	f
16	E-15	2011-10-19	1231.0	7.4	1230.67	2011	42	2011-10-19 09:21:51.139	2011-10-19 09:21:51.139	2011-10-19 09:22:31.139	f
17	E-16	2011-11-06	496.0	7.4	496.60	2011	44	2011-11-06 04:58:53.259	2011-11-06 04:58:33.259	2011-11-06 04:59:13.259	t
18	E-17	2012-03-27	74.0	7.5	74.17	2012	13	2012-03-27 18:30:08.617	2012-03-27 18:29:48.617	2012-03-27 18:30:28.617	t
19	E-18	2012-04-14	74.0	7.5	74.10	2012	15	2012-04-14 14:01:37.59	2012-04-14 14:01:17.59	2012-04-14 14:01:57.59	f
20	E-19	2012-05-02	74.0	7.5	73.13	2012	18	2012-05-02 09:31:28.864	2012-05-02 09:31:08.864	2012-05-02 09:31:48.864	f

The best part? The altitudes from the INMS agree almost precisely with what NASA has published. I think the discrepancy is something we can live with, if not I'm sure Eno will tell me.

Just looking this over... it feels right. I guess it's like M. Sullivan said: it's the pain of learning. You can tell when things are just wrong and you're working really hard to make them right. This data just fell together, and I like it.

You know what this calls for, Dee? Yep! Some Pappy Van Winkle. I'm off to The Ramp in a few minutes.

THE INMS TUTORIAL

November 2, 2017, 1621

Just had a look at The INMS Analysis Library and I'm kicking myself for not finding this earlier! I guess that's part of the problem when working with an information corpus the size of Cassini's: you have to be really good at finding things.

I was looking at the CDA data to find Cassini's velocity and, for some reason, I didn't even think to have a look at the INMS data. Here it is, precisely what I was looking for:

	wrt the named target				
sc_vel_t_x	x component of spacecraft velocity wrt the named target	real	km s ⁻¹	x ≤ 100	F7.3
sc_vel_t_y	y component of spacecraft velocity wrt the named target	real	km s ⁻¹	x ≤ 100	F7.3
sc_vel_t_z	z component of spacecraft velocity wrt the named target	real	km s ⁻¹	x ≤ 100	F7.3

Relative velocity! This description is different from the manifest, which says:

X-component of spacecraft velocity in the Saturn centered IAU coordinate frame.

I thought that meant "relative to Saturn." When I sew this all together, I'll compare it to what was reported by NASA. This is great news! I can go home a happier data person tonight.

From: Rob Conery rob@redfour.io
Subject: Well done! Dinner's on me
To: Dee Yan yand@redfour.io
Date: November 2nd, 2017

Dee, this is precisely the reason I chose you for this job. You're full guns and don't give up even when you shoot yourself in the toe. Just don't keep doing that. They don't grow back.

Now that we have a reliable source of flyby information, it's time to turn our attention to chemistry. There is so much going on with Enceladus, but right now I have to get back to this cocktail party. SpaceX is trying to schmooze their way into a private meeting with Senator Warren – I can't let that happen.

More tomorrow – dinner's on me. There's a \$200 gift certificate waiting for you and a guest at Green's in Fort Mason. My favorite place. Took my wife there on our first wedding anniversary. Try the braised seagull feet, much better than they sound.

Just kidding about that. Don't eat seagull feet.

R

UNDER THE ICE

From: Rob Conery rob@redfour.io
Subject: And now, to our purpose
To: Dee Yan yand@redfour.io
Date: November 3rd, 2017

Damn, this place is loud. Sorry I couldn't finish my email last night. I needed to think about the update you sent RE the latest flybys table, and where to go from here.

I don't think anything has changed, really. Our plan is still 100% go. We just need to use what you've made and find what we're looking for.

Eno told me that she suggested you drop by the Presidio office and I think that's a good idea. If you're sitting with the Analytics Team things might move faster, which means I need to give you a lot more detail. Here goes:

We're trying to create a thermochemical map. Precise locations of temperature and chemistry (H₂, CO₂, CH₄, CO, N₂, etc.) which we can overlay onto the south pole of Enceladus. As you may or may not know, almost the entirety of Cassini's analysis of Enceladus was focused on these two areas.

I know you've done a bit of reading on Enceladus, but take some time tonight to understand the full timeline. There are some details you'll need to know intimately, especially from the flybys in 2015.

Attached to this email, you'll find a CSV file called **chem_data.csv**. Load it up and have a look. If the data contains what I think it does, we might just have a hand in the most significant discovery humanity has ever made.

Have a nice weekend!

IMPORTING CHEMICAL DATA

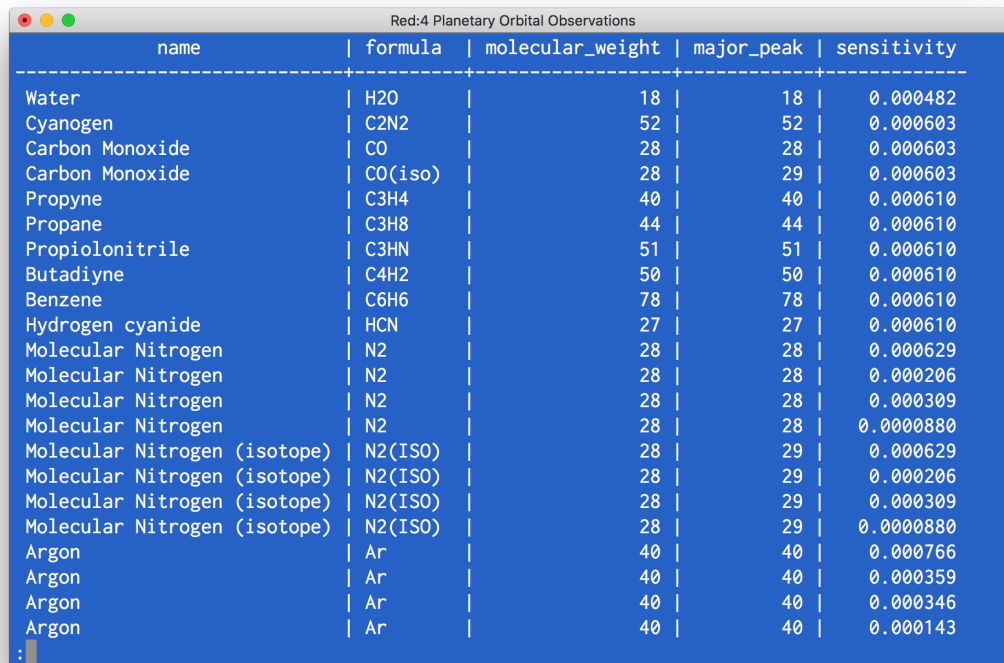
November 3, 2017, 1525

Alrighty then, Rob. Nothing like a bit of drama to start your Friday morning.

I loaded up the **chem_data.csv** file as I've been doing and, once again, used a simple SQL file — no need to get fancy.

```
create table chem_data(  
  name text,  
  formula varchar(10),  
  molecular_weight integer,  
  peak integer,  
  sensitivity numeric  
);  
  
copy chem_data  
from '[PATH]/data/chem_data.csv' -- data/INMS directory  
with delimiter ',' header csv;
```

This data looks interesting.



name	formula	molecular_weight	major_peak	sensitivity
Water	H2O	18	18	0.000482
Cyanogen	C2N2	52	52	0.000603
Carbon Monoxide	CO	28	28	0.000603
Carbon Monoxide	CO(iso)	28	29	0.000603
Propyne	C3H4	40	40	0.000610
Propane	C3H8	44	44	0.000610
Propiolonitrile	C3HN	51	51	0.000610
Butadiyne	C4H2	50	50	0.000610
Benzene	C6H6	78	78	0.000610
Hydrogen cyanide	HCN	27	27	0.000610
Molecular Nitrogen	N2	28	28	0.000629
Molecular Nitrogen	N2	28	28	0.000206
Molecular Nitrogen	N2	28	28	0.000309
Molecular Nitrogen	N2	28	28	0.0000880
Molecular Nitrogen (isotope)	N2(ISO)	28	29	0.000629
Molecular Nitrogen (isotope)	N2(ISO)	28	29	0.000206
Molecular Nitrogen (isotope)	N2(ISO)	28	29	0.000309
Molecular Nitrogen (isotope)	N2(ISO)	28	29	0.0000880
Argon	Ar	40	40	0.000766
Argon	Ar	40	40	0.000359
Argon	Ar	40	40	0.000346
Argon	Ar	40	40	0.000143

The manifest for this data (**chem_data_manifest.txt**) describes the obvious stuff (name, formula) but I’m not familiar with **major_peak**.

The manifest says this:

The mass of the dissociation product produced in greatest quantity

The units are Daltons (Atomic Mass Units, or AMU) per charge (AMU/Z) and I think the INMS data, in addition to velocity and about a million other things, has a **mass_per_charge** field that is also Daltons. I’ll have to see if they’re related.

After I loaded up the **chem_data.csv** file, I decided to leave a bit early. I got on the bus and went back to The Marina, but instead of going home, I walked over to the Palace of Fine Arts.

The fog had just burned off as I made my way from the start down to the benches that line the water. A group of Whooper swans glided toward me, flanked by a bunch of ducks and a few grebes. My dad loved these swans, kept telling me they were “iconic” and made

this whole place romantic. I wish I could agree with him on that, but I think the whole thing is cheesy. Maybe it's the non-stop weddings wafting oppressive amounts of perfume across the way as cover for the phalanxes of groomsmen to crowd everyone else out, or the super-serious photographers with boxes of gear and tripods taking the essential shot that somehow, against all the odds, millions of other super-serious photographers haven't managed to get. It could also be the faux Roman/ Greek architecture that looks 100% out of place in this California city by the bay.

I think the pressure of the last week is getting to me. I'm grumpy, and it's sort of silly. I have a fantastic job (well, internship I suppose), working with great people and some fantastic data. Pull it together Dee!

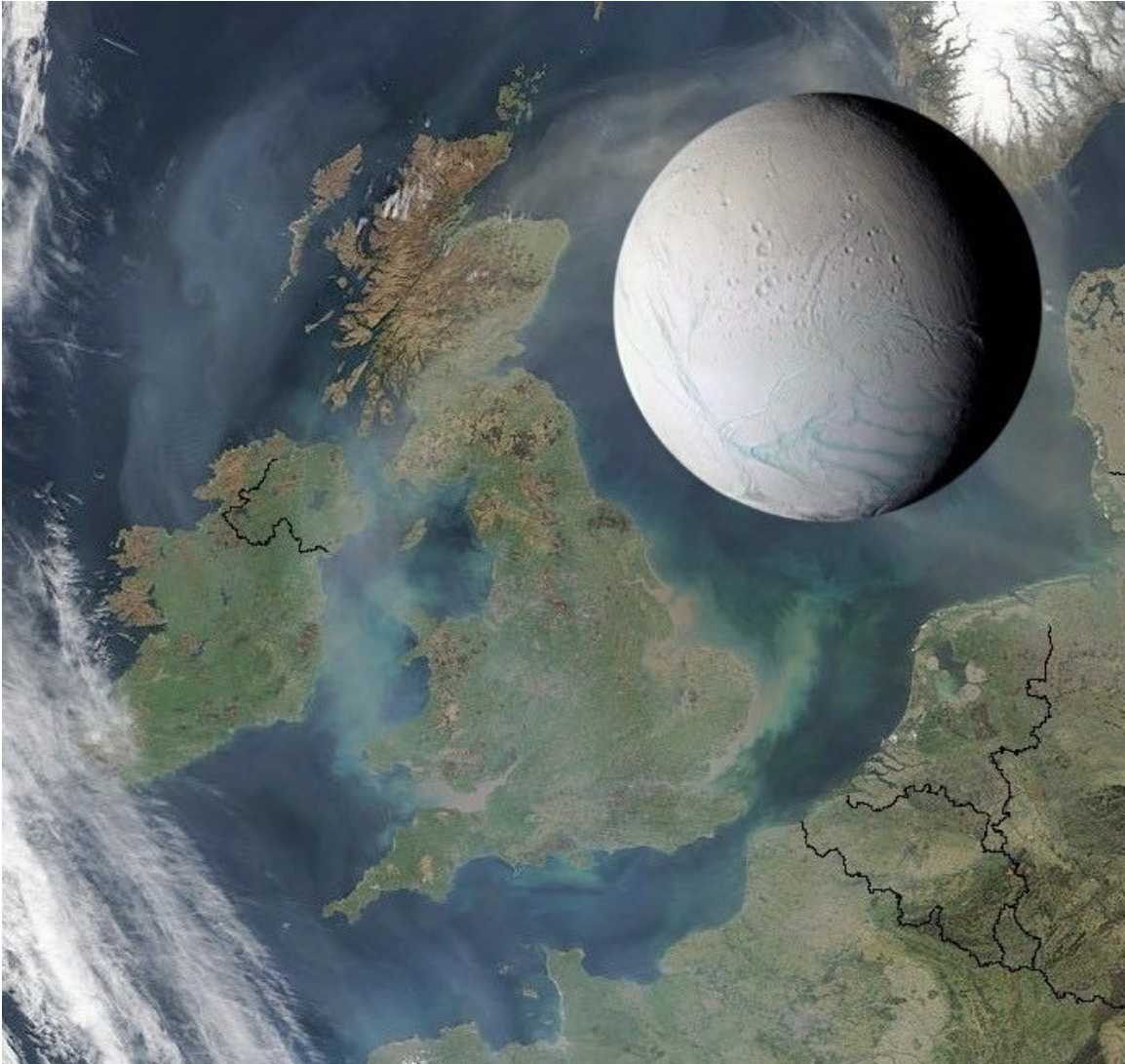
I found a bench all to myself under a willow (my favorite tree), with just the right amount of sun to keep me from freezing. I popped open a Diet Coke, put my laptop on my bag and my bag on my lap to keep me warm, and I started reading the details about Cassini's encounters with Enceladus...

LOOK AT ME!

Enceladus has changed the way we think about our solar system. It should not exist, yet it does! Curt Niebur, a scientist at NASA on the Cassini team, [proclaimed](#):

Enceladus has no business existing, and yet there it is, practically screaming at us, 'Look at me, I completely invalidate all of your assumptions about the solar system'!

Indeed, the moon is only 313 miles in diameter, which would look like this if it were plopped down next to Great Britain:



Enceladus and Great Britain. Image Credit: NASA/JPL

A Non-Dead Ice Ball

Enceladus was discovered in 1789 by William Herschel. Named after a giant from Greek Mythology, the little moon was only slightly interesting to scientists because it was so reflective. The belief by most scientists on the Cassini team was that Enceladus, like its neighbor moons, was just a “dead ice ball”.

In 1981 Voyager snapped some images of Enceladus, which led scientists to question whether the little moon was somehow involved with Saturn's diffuse and less-exciting E-ring. When viewing these images, it looked as though Enceladus was sitting right in the thick of the E-ring cloud. Not believing in coincidence, they wondered if material from the bright little moon might be contributing something to the ring itself. They would have to wait for over 20 years to find out as Voyager II sped away to the planets farther out in our solar system.

Some questions arose from Voyager's observations:

- Was volcanism, a network of geysers or some kind of plume causing material from Enceladus to fill the E-ring?
- If so, Enceladus should have drained itself out of existence a long, long time ago... yet it hasn't. Why?
- Why is the surface so reflective? Impacts from various bodies should have darkened the surface somewhat, but they haven't. Why not?

NASA sought answers to these questions, so Cassini was scheduled to swing by the small moon in February 2005. It would rotate itself, pointing its cameras and magnetometer at Enceladus. What they observed defied explanation: Enceladus was *bending* Saturn's magnetic field.

The first three flybys, E-0 through E-2, were all about investigating this bend around the south pole of Enceladus. The readings just didn't make sense, and 3 flybys later they had their confirmation: Enceladus was spewing ice into space, creating a small atmosphere.

Initially, scientists thought that the water was being pushed to the surface by an "off-gassing" process, like what has been observed with comets. That changed when the readings from CIRS, Cassini's infrared scanner, tracked heat under the south pole:

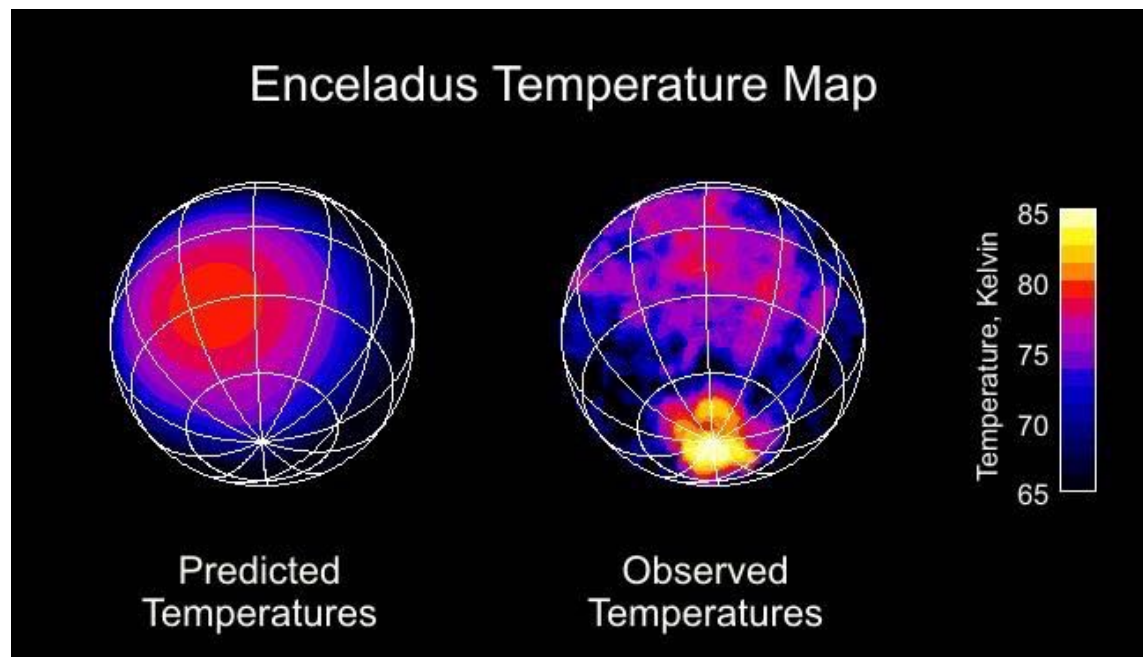


Image credit: NASA/JPL

Later flybys confirmed the presence of water and organic molecules including CO₂, CO, and CH₄. Cassini also registered something else: salt and silica.

This presented a problem: how does salt make its way into ice? Up until this point, scientists figured that warmer ice was evaporating directly from the plumes. This changed with the presence of salt, which could only be present in the water if happened in solution. That meant liquid water and a whole lot more heat than was predicted.

In 2010, a theory was put forth that tried to explain where this heat might be coming from. Two sources were suggested: a tidal pull from Saturn and Dione (the next closest moon) and/or radioactive decay of some type of material.

Physicists decided to create a model of the tidal stresses that would be applied to Enceladus based on its position relative to Saturn. They then took the ISS, VIMS and CIRS images to see if they could find visual proof of what their models suggested.

They could not. The optimal stress zones, where the heat should be generated, were just off by a few kilometers. There was something strange in the orbit of Enceladus.

As they observed Enceladus's orbit around Saturn, they also found that changes in orbital distance (caused by the elliptical nature of orbits) also changed the tidal stress on Enceladus. This, in turn, caused the plumes to change their activity level, sometimes shutting down completely.

It was becoming clear that Enceladus's icy crust was flexible. Scientists turned once again to the imagery of Enceladus's surface features, taken at different times during its orbit of Saturn. They lined up craters and other surface features to see how they changed during the orbital period.

Like our moon, Enceladus is tidally locked to Saturn. This means that it rotates only once per year, and the features on the surface should remain relatively static. But they didn't. They moved a tiny amount, from left to right, due to a wobble. This wobble was caused by the ice being wholly detached from the core below. In other words: Enceladus had a global ocean.

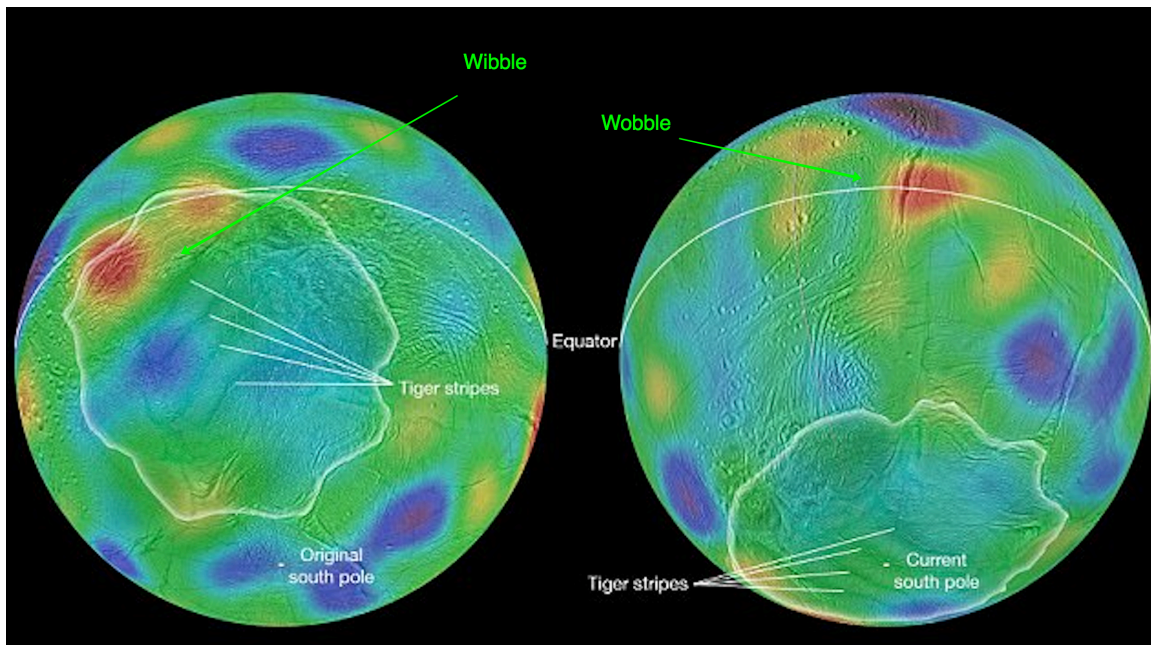


Image credit: NASA/JPL

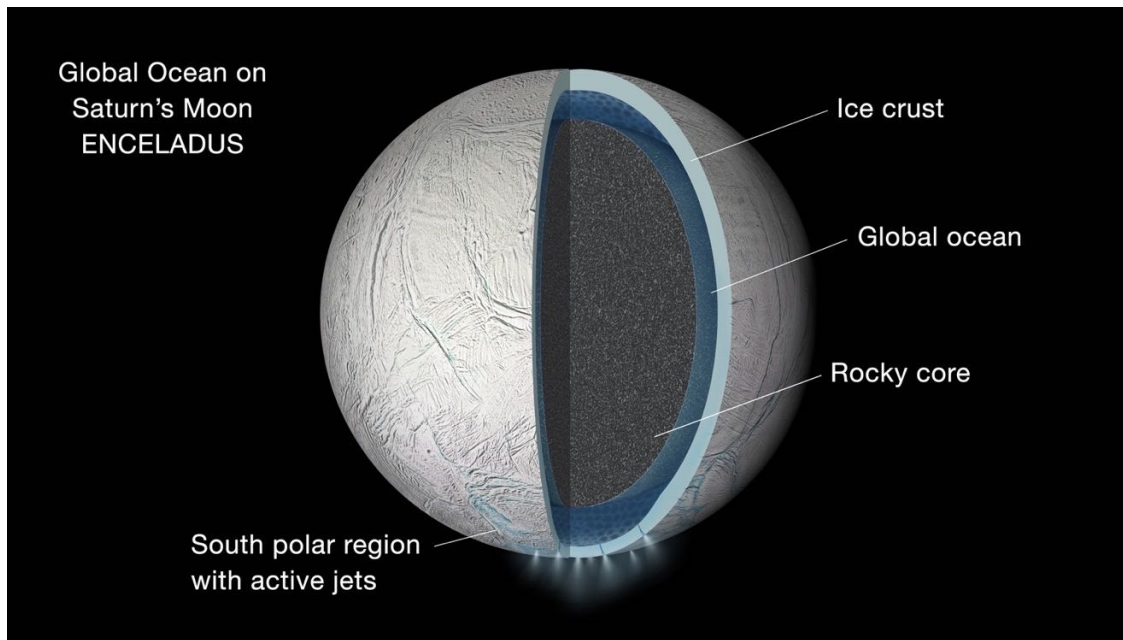


Image credit: NASA/JPL

This was a stunning revelation. Like every revelation from Enceladus, it raised more questions than it answered. One above all: *where is this heat coming from?*

The thermal observations from Cassini were confounding. The energy required to produce a global ocean with a plume system just couldn't exist according to planetary science at the time. The models that the NASA/JPL scientists put together suggested that the water temperatures right at the point of the plume hotspots probably reached 90 degrees Celsius. Right in the range of the black smokers in the deep-sea trenches of the earth.

Some theories regarding Enceladus's heat are gaining acceptance. Recently, a team of German scientists led by Dr. Frank Postberg at the University of Heidelberg [proposed](#) that the core of Enceladus could be porous, allowing water from the overlying ocean to seep in. The friction caused by tidal warping would heat this water, creating enough energy to drive the plume system.

If Enceladus has water and energy, it only needs the proper chemistry to be a prime candidate for the presence of life. In 2015, using black and white smokers on earth as a

model, scientists began to question whether there could be some form of life near the hotspots on the ocean floor of Enceladus.

A suite of flybys was undertaken in 2015 (E-20 through E-22), with E-20 being dedicated to sampling the plume deeper and more directly than was ever done before. They needed to find molecular hydrogen (H₂), the single nutrient that drives the entire food chain around the black smokers. Find that, and things become [a lot more interesting](#).

It would be like a candy store for microbes.

– Hunter Waite, Program Director, Space Science, and Engineering Division,
NASA

UNDERSTANDING THE INMS READINGS

November 3, 2017, 1649

Why does this always happen to me at the end of the day? I knew that scientists thought there might be life on Enceladus, but it still seemed so theoretical. Is it there or not? Just tell me that much...

But then you start reading these articles, getting into the details of the mystery of this crazy little moon, and it's just *weird*. That's my scientific opinion, at least.

Alright, to work now. Let's see if I can get something done before M. Sullivan chases me out of here. The last time I looked at the INMS data, it was merely to find some altitude information, nothing chemical.

I've downloaded the INMS data manifest from the archives and looked it over. There's a lot in here I don't particularly understand, but I do know that mass and particle charge are the things I should be interested in.

Think I'll ask Eno what she thinks. I'm sure there's an astrobiologist over there on the Analysis Team.

From: Eno L eno@redfour.io
Subject: RE: What data from the raw INMS dump?
To: Dee Yan yand@redfour.io
Date: November 3rd, 2017

I've gone over the manifest that you sent along and sat with Karl, one of the Astrobiologists on our team. I hope you can drop by sometime next week to say hello, nice group of people over here!

Anyway, he first suggested "all of it," but I pressed him a bit more — I know drive space is an issue for you guys. Here was his suggestion:

- Any date information
- Source
- Mass table
- Mass per charge (the big one!)
- Particle energy
- Any relative speed data you can find
- C1 and C2 counters

As long as you have the raw data and can rebuild your set, we can adjust if we need more data later on. This should be all we need though, and the mass, charge and counter data is what we're focused on.

And yes, you should be able to relate those numbers directly to the **mass_peak** field in the chemical data that Rob sent you.

E

LOADING THE INMS READINGS

November 3, 2017, 1723

Gotta make this fast, M. Sullivan will be dropping a note any minute now to tell me to go home... I can feel it.

The first thing I'll do is to create a schema for the INMS, now that we have some data we need to analyze. I'll also move the **chem_data** table in there as I don't think we'll be using it anywhere else:

```
-- load up a new schema for instrument data
drop schema if exists inms cascade;
create schema inms;

-- move chem data in there
alter table chem_data
set schema inms;
```

Now I should be able to load up a table of results using the import data I loaded last week. I'll reuse my **pythag** function to calculate Cassini's speed relative to Enceladus:

```

-- create the inms. readings table
select
  sclk::timestamp as time_stamp, source::text, mass_table,
  alt_t::numeric(9,2) as altitude,
  mass_per_charge::numeric(6,3),
  p_energy::numeric(7,3),
  pythag(
    sc_vel_t_scx::numeric,
    sc_vel_t_scy::numeric,
    sc_vel_t_scz::numeric
  ) as relative_speed,
  c1counts::integer as high_counts,
  c2counts::integer as low_counts
into inms.readings
from import.inms
order by time_stamp;

alter table inms.readings
add id serial primary key;

```

I want to be able to get at this data easily, without having to do weird joins on dates, or anything else like that. The simplest thing to do would be to just relate this table to **flybys**:

```
-- add an FK to the flybys
alter table inms.readings
add flyby_id int references flybys(id);

update inms.readings
set flyby_id=flybys.id
from flybys
where flybys.date = inms.readings.time_stamp::date;
```

That seems like an expensive operation, checking equality on a cast **time_stamp**. I'd like to know just how much before I run this.

Using Explain and Analyze

There are two options for updating the 13 million rows in the **inms.readings** table. One uses a join, the other does a nested select statement. I don't know which one will be worse.

I'm trying to avoid a nastygram from M. Sullivan, so I would like to pick the query that's less "painful", I suppose. How do I do that?

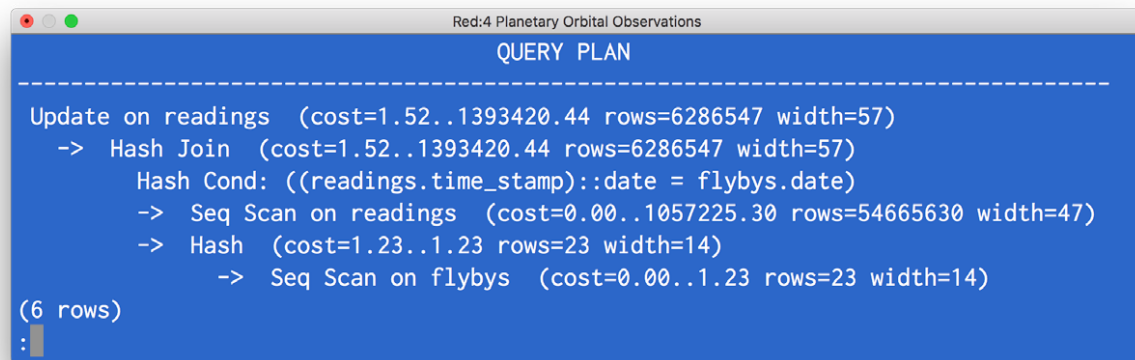
Be right back... Googling...

Found it: just put the keyword **explain** in front of a query and Postgres will do its best to tell you what's going on:

```

explain update inms.readings
set flyby_id=flybys.id
from flybys
where flybys.date = inms.readings.time_stamp::date;

```



```

Red:4 Planetary Orbital Observations
QUERY PLAN
-----
Update on readings (cost=1.52..1393420.44 rows=6286547 width=57)
-> Hash Join (cost=1.52..1393420.44 rows=6286547 width=57)
    Hash Cond: ((readings.time_stamp)::date = flybys.date)
    -> Seq Scan on readings (cost=0.00..1057225.30 rows=54665630 width=47)
    -> Hash (cost=1.23..1.23 rows=23 width=14)
        -> Seq Scan on flybys (cost=0.00..1.23 rows=23 width=14)
(6 rows)
:

```

The trick now becomes understanding what it's telling me. Be right back, Googling again...

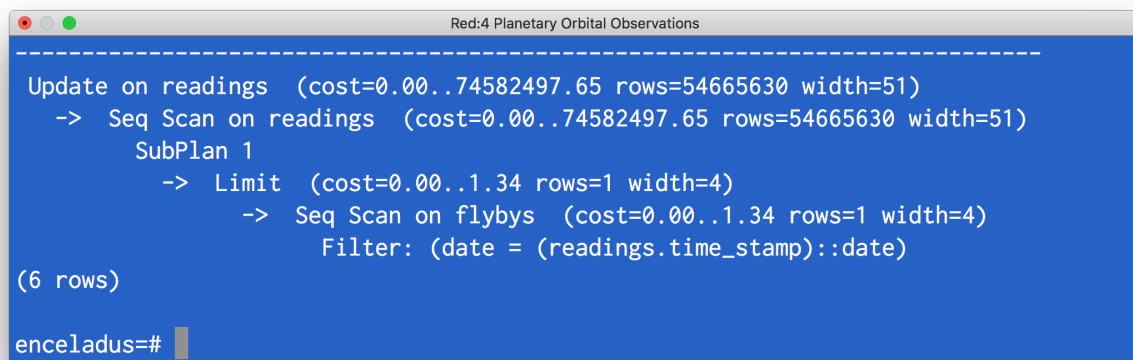
OK, I think I got this. What I'm being shown here is a step by step plan that Postgres will use to fulfill this query when I execute it. All the numbers are estimates and **cost** is a nebulous thing that no one seems to understand, as far as I can tell. It's a relative measurement of the time it takes to read some data from disk, and if you're trying to optimize a query, you want this number to go down. The cost for this query is 1.52 million, which I think is pretty high but honestly, I have no idea until I compare it with the other one.

The rows are the estimated number of rows output from the query. This might not equal the number of rows actually read off the disk. For that, you need to look at the scans as they are the actual reads.

I can see the join is made, and the sequential scan happening for every joined row, which comes out to 54 million rows read! Another sequential scan happens for the **flybys** table which looks to be relatively low cost at 23.

The other query seems to be a lot worse, unfortunately:

```
explain update inms.readings
set flyby_id = (
  select id from flybys
  where date = inms.readings.time_stamp::date
  limit 1
);
```



```
-----
Update on readings (cost=0.00..74582497.65 rows=54665630 width=51)
-> Seq Scan on readings (cost=0.00..74582497.65 rows=54665630 width=51)
    SubPlan 1
        -> Limit (cost=0.00..1.34 rows=1 width=4)
            -> Seq Scan on flybys (cost=0.00..1.34 rows=1 width=4)
                Filter: (date = (readings.time_stamp)::date)
(6 rows)

enceladus=#
```

The estimated cost of this query is 74 million! I think this is due to using a **where** statement, which is the filter expression at the bottom?

I could know for sure if I ran **explain analyze**, which would actually execute the query and then tell me what happened. I can see how this would be invaluable for optimizing things, but ... M. Sullivan.

Looks like I'll be executing the first one. Hopefully, he won't —

From: M. Sullivan sullz@redfour.io
Subject: Go. Home.
To: Dee Yan yand@redfour.io
Date: November 3, 2017

You have made outstanding progress today M. Yan. It's time for you to go home now, however, per M. CEO's wishes. I'll have some thoughts for you in the morning, but I have to say I'm quite impressed by the work you've done today.

Not the SQL you've written, that's ghastly. But the results look good. At least you're not trying to cast null to some data type.

Best, S

Aw crap. Did I not actually have to do that?

From: M. Sullivan sullz@redfour.io
Subject: Consider a Many to Many Join, Please.
To: Dee Yan yand@redfour.io
Date: November 3, 2017

I see that you're altering a table in one schema (**inms.readings**) and attempting to relate it via foreign key to a table in another schema (**public.flybys**). While I am sure you feel you have reason to do something like this, please be aware that you are incorrect.

A better solution is to create a junction table. This is a special kind of table that exists solely to support a relationship between two unrelated tables which, in your case, are the **inms.readings** and **public.flybys** tables.

Simply create a single table with two fields which will store the primary keys from the tables above. The primary key of *this* table should combine the two foreign keys to ensure unicity, which means the quality of being unique (I have no truck with those who merely add "-ness" to words). If you were to leave it this way, however, you would not be entirely safeguarded against non-unique key values. In other words: records from **inms.readings** could appear twice, which is inaccurate.

See if you can solve that one on your own. Use Google if you must.

Best, S

November 3, 2017, 1622

On the bus, headed back to the Marina. Think I'll stop off at the Horseshoe on Chestnut for a bit.

I think I understand what M. Sullivan wants me to do and, yes, it's much cleaner regarding design. I'll take care of that in the morning... but something else is bugging

me: why did he/she/ they send me two emails at once? In the wrong order? That “Go. Home.” email must be on a timer. Dude’s weird.

Now I *really* want to meet her/him/them.

A JUNCTION TABLE

November 6, 2017, 0722

It seems to be a recurring pattern in my life that the best work, the stuff that’s the most fun, happens at 4:55pm on Friday afternoon, right when M. Sullivan kicks me out for the weekend.

I spent most of Saturday and Sunday looking at apartments.

There’s no way I’ll be living in the city; it’s ridiculously expensive. Mom offered to let me live upstairs with her, but there is just no way I’m going to do that.

I don’t know what I’m going to do. Part of me wants to travel... just pick up and live out of a backpack for a year or so. Work from cute little European cafes, meet interesting people and wander where I will. Mom thinks it’s a great idea; I’m still young and single so... “Just go” she says. “A hui hou!”

I might. We’ll see how this job pans out.

Speaking of, I think I’m getting better at SQL! I didn’t need to Google how to make the fix M. Sullivan asked for:


```
-- a junction table
drop table if exists flyby_readings;
create table flyby_readings(
    reading_id int not null unique references inms.readings(id),
    flyby_id int not null references flybys(id),
    primary key(reading_id,flyby_id)
);
```

Inserting data into it took a while, but it all went in with a simple select query:

```
-- fill it
insert into flyby_readings(flyby_id, reading_id)
select
    flybys.id,
    inms.readings.id
from flybys
inner join inms.readings
on date_part('year',time_stamp) = flybys.year
and date_part('week',time_stamp) = flybys.week;
```

That's going to take a while. Think I'll go grab a Diet Coke from downstairs.

From: M. Sullivan sullz@redfour.io
Subject: Maybe a simple index will suffice?
To: Dee Yan yand@redfour.io
Date: November 6, 2017

Looks like M. Eno was able to help you with the **flybys** table, which is excellent news. I also see that you're bending over backward just a little too far to try to make querying easier. While I commend you for being thoughtful, you're sort of just taking up space on disk for no real gain.

When you create something in a database, there should be a justification for its existence. M. Eno's ephemeral table was justified as it was there simply to hold data and speed up querying. Its lifespan was very small, so a materialized view would have been pointless.

The dates appear to be your focus, so let Postgres help you by just creating an index on the **time_stamp** field and forget these joined tables. Joins can be expensive and the tables required to support them costly regarding disk space. I didn't fully realize what you were trying to do when you were crossing schema boundaries; I suppose this was confusing and... I naturally blame you.

A simple index on the field you want to join on can help in more straightforward cases like this one, where you need to specify date criteria.

Best, S

INDEXES VS. JOINS

November 6, 2017, 0819

Rode my bike in again, but the weather was significantly colder than it's been. I love to ride along the marina, and Fort Mason is so beautiful when the sun is coming up. Even when it's windy, the cypress trees make the most haunting, whooshing sound as the fog whips around them.

It doesn't make sense to me how an index could be faster than an integer key join. Can't you index the foreign key? I know you can do that to speed things up so...

I guess there's a more significant concern: disk space. M. Sullivan did mention that. Any extra column on a table with 13 million rows is going to take up considerable space, even if it is just an integer. The "junction table," as M. Sullivan called it, takes up twice as much in the name of clean design. I think I'm beginning to appreciate, instinctively, when I'm going the wrong direction.

Dee's new law: if propelling 'good' design makes you write needless code, take up unnecessary space or make the system needlessly work harder, something's wrong.

If I were to create a foreign key on the **inms.readings** table with an index, that's more space required and more work for the engine regarding keeping the index up. The junction table multiplies the problem: I have two additional constraints (a foreign key and a unique), an extra column, and two extra indexes to make everything faster. That's a lot of space and resources just to accommodate purist design principles!

If I just index the **time_stamp** field on **inms.readings**, then I should be able to do a join at a reasonably small cost. I'll have to run explain to be sure so M. Sullivan doesn't rupture something.

Speaking of Sullivan: today's the day, come hell or high water. Eno would have said if he/she/they worked in her office in the Presidio, and given that the rest of the interns and the data team all work in SOMA... I bet that's where.

I don't know why, but I imagine that he's male, for some reason. I imagine a meticulously pressed white shirt with a black tie, sleeves carefully rolled up to above the elbow. Black denim jeans (an expensive brand, of course) with hipster shoes. Something leather... distressed with red laces and a lifted heel.

Curly hair. Has to be.

TIMESTAMPS AND INDEXES

November 6, 2017, 1142

I've spent the entire morning obsessing on the "proper" index to use for my timestamps. I've learned a ton, and I'm still a bit confused, but I'm going to let efficiency and speed be my guide.

I have three timestamps I need to consider:

1. **`inms.readings.time_stamp`**
2. **`flybys.window_start`**
3. **`flybys.window_end`**

I've learned that there are different approaches to structuring this data and querying it, and I need to go over each one.

I'll start with the simplest.

Simple: A BTREE Index

The most straightforward thing I could do would be to use a BTREE index. BTREE stands for "balanced tree," and I remember this from my algorithms class: it's just a binary tree that has an equal distribution of child nodes, on both sides of the primary node. Balanced trees are also the subject of those annoying questions they ask you in interviews at big tech companies.

Anyway, to speed things up, you can throw an index on the column that you're querying against, which helps Postgres find it. I want to find all INMS readings within a time range, so I can simply index the **inms.readings.time_stamp** field:

```
create index idx_stamps
on inms.readings(time_stamp);
```

This works, but it's not optimal as Postgres will be doing a lot more work (and taking up a lot more space on disk) than I need. I want to restrict the indexing to only the values that I'll be searching over, which are values with an **altitude** reading:

```
create index idx_stamps
on inms.readings(time_stamp) where altitude is not null;
```

Perfect. There are 13 million rows in the **inms.readings** table so if I execute this query it's going to take a while to run. It's OK if it's just me sitting here, but if I were doing this in production, it would lock the entire table, which is bad. I can get around this by using the **concurrently** keyword:

```
create index concurrently idx_stamps on inms.readings(time_stamp)
where altitude is not null;
```

Evidently, you can't do this in other databases without paying for an enterprise license. How nice.

Running a simple query using this index is incredibly fast!

```
select * from inms.readings
where time_stamp > '2015-12-19 17:48:55.275'
and time_stamp < '2015-12-19 17:49:35.275'
and altitude is not null;
```

It returns in 10ms!

time_stamp	source	mass_table	mass_per_charge	p_energy	altitude	c1counts	c2counts	id
2015-12-19 17:48:55.307	csn	16	28.000	70.228	5003.94	1	0	13616140
2015-12-19 17:48:55.341	csn	16	28.000	70.228	5003.93	2	0	13616141
2015-12-19 17:48:55.375	csn	16	2.000	5.016	5003.91	2	0	13616142
2015-12-19 17:48:55.409	csn	16	44.000	110.359	5003.90	0	0	13616143
2015-12-19 17:48:55.443	csn	16	14.000	35.114	5003.89	0	0	13616144
2015-12-19 17:48:55.477	csn	16	15.000	37.622	5003.88	0	0	13616145
2015-12-19 17:48:55.511	csn	16	28.000	70.228	5003.87	2	0	13616146
2015-12-19 17:48:55.545	csn	16	16.000	40.130	5003.85	0	0	13616147
2015-12-19 17:48:55.579	csn	16	17.000	42.639	5003.84	0	0	13616148
2015-12-19 17:48:55.614	csn	16	18.000	45.147	5003.83	0	0	13616149
2015-12-19 17:48:55.648	csn	16	28.000	70.228	5003.82	2	0	13616150
2015-12-19 17:48:55.682	csn	16	28.000	70.228	5003.81	3	0	13616151
2015-12-19 17:48:55.716	csn	16	12.000	30.098	5003.79	0	0	13616152
2015-12-19 17:48:55.75	csn	16	32.000	80.261	5003.78	0	0	13616153
2015-12-19 17:48:55.784	csn	16	14.000	35.114	5003.77	0	0	13616154
2015-12-19 17:48:55.818	csn	16	44.000	110.359	5003.76	2	0	13616155
2015-12-19 17:48:55.852	csn	16	20.000	50.163	5003.75	0	0	13616156
2015-12-19 17:48:55.886	csn	16	28.000	70.228	5003.73	3	0	13616157

Taking a look at the query plan using **explain** shows how the index is being used instead of a sequential scan:

```
-----
Bitmap Heap Scan on readings (cost=23.40..4168.71 rows=1070 width=42)
  Recheck Cond: ((time_stamp > '2015-12-19 17:48:55.275'::timestamp without time zone)
-> Bitmap Index Scan on readings_time_stamp_idx1 (cost=0.00..23.14 rows=1070)
      Index Cond: ((time_stamp > '2015-12-19 17:48:55.275'::timestamp without time zone)
(4 rows)
```

Instead of hardcoding the dates, I can now join the **flybys** table using **window_start** and **window_end**:

```
-- without hardcoded dates
select
    name,
    mass_per_charge,
    time_stamp,
    inms.readings.altitude
from inms.readings
inner join flybys
on time_stamp >= window_start
and time_stamp <= window_end
where flybys.id = 4;
```

That is super fast! The results look correct, too:

name	mass_per_charge	time_stamp	altitude
E-3	25.000	2008-03-12 19:05:51.487	169.81
E-3	26.000	2008-03-12 19:05:51.521	169.46
E-3	27.000	2008-03-12 19:05:51.555	169.12
E-3	28.000	2008-03-12 19:05:51.589	168.77
E-3	29.000	2008-03-12 19:05:51.623	168.42
E-3	44.000	2008-03-12 19:05:51.657	168.08
E-3	45.000	2008-03-12 19:05:51.691	167.73
E-3	84.000	2008-03-12 19:05:51.725	167.39
E-3	61.000	2008-03-12 19:05:51.759	167.05

I output the name to make sure it aligned with the dates and altitudes. It would be fun to have a “countdown” as well. Something like the **altitude - nadir**:


```
select
    name,
    mass_per_charge,
    time_stamp,
    inms.readings.altitude,
    inms.readings.altitude - nadir as distance
from inms.readings
inner join flybys
on time_stamp >= window_start
and time_stamp <= window_end
where flybys.id = 4;
```

I'm such a dork... but this seems more readable and also gives an interesting perspective on how fast Cassini was going:

name	mass_per_charge	time_stamp	distance
E-3	25.000	2008-03-12 19:05:51.487	119.52
E-3	26.000	2008-03-12 19:05:51.521	119.17
E-3	27.000	2008-03-12 19:05:51.555	118.83
E-3	28.000	2008-03-12 19:05:51.589	118.48
E-3	29.000	2008-03-12 19:05:51.623	118.13
E-3	44.000	2008-03-12 19:05:51.657	117.79
E-3	45.000	2008-03-12 19:05:51.691	117.44
E-3	84.000	2008-03-12 19:05:51.725	117.10
E-3	61.000	2008-03-12 19:05:51.759	116.76

Running **explain** on this again, the query is taking full advantage of the index:

```

Red:4 Planetary Orbital Observations
QUERY PLAN
-----
Nested Loop (cost=32670.22..585581.87 rows=1539686 width=77)
-> Seq Scan on flybys (cost=0.00..1.29 rows=1 width=80)
    Filter: (id = 4)
-> Bitmap Heap Scan on readings (cost=32670.22..566334.51 rows=1539686 width=21)
    Recheck Cond: ((time_stamp >= flybys.window_start) AND (time_stamp <= flybys.window_end))
-> Bitmap Index Scan on readings_time_stamp_idx1 (cost=0.00..32285.30 rows=1539686 width=0)
    Index Cond: ((time_stamp >= flybys.window_start) AND (time_stamp <= flybys.window_end))
(7 rows)

```

Still returning in 10ms or less, despite the join. Can't see how this can be any faster, really. In the spirit of covering all my bases, however, here's what else I learned.

ADVANCED: USING TIME RANGES

Postgres supports datatypes dedicated to ranges, which is utterly fascinating. There are numeric ranges and date/time ranges; I'm interested in the latter right now.

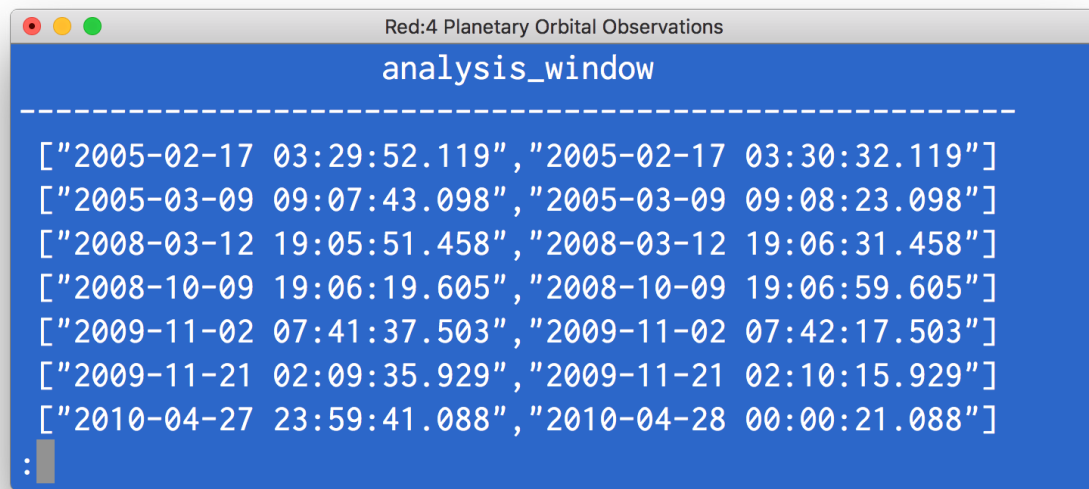
You can define a range using a simple constructor. This means that instead of using **window_start** and **window_end** I could just have a column called **analysis_window**:

```
alter table flybys  
add analysis_window tsrange;
```

To create a range, you simply invoke the **tsrange** type:

```
update flybys  
set analysis_window=tsrange(window_start,window_end, '[')');
```

That's it. Here's what it looks like:



The `[]` option tells Postgres that this range is inclusive, meaning the upper and lower bounds of the range should be included. If I had used `()` as an option the range would be exclusive, meaning all dates between the upper and lower bound are considered the value. You can combine these two symbols as well. For instance, this: `[])` is a range that is inclusive on the lower bound, exclusive on the upper.

Querying a range is easy, and you have some special operators you can use. This query checks to see if a date is contained within a range:

```
select name from flybys
where analysis_window @> '2005-02-17 03:30:12.119'::timestamp;
```

It is, in fact:

```
Red:4 Planetary Orbital Observations
enceladus=# select name from flybys
enceladus=# where analysis_window @>
enceladus=# '2005-02-17 03:30:12.119'::timestamp without time zone;
name
-----
E-0
(1 row)

enceladus=#
```

This means I can query things a bit more easily, regarding syntax:

```
select name, mass_per_charge, time_stamp, inms.readings.altitude - nadir as distance
from inms.readings
inner join flybys
on analysis_window @> inms.readings.time_stamp where flybys.id = 4;
```

This works, but it's really slow. On my little laptop here it took 5 seconds to run. Checking the query plan, you can see why:

```
Red:4 Planetary Orbital Observations
QUERY PLAN
-----
Nested Loop (cost=0.00..822392.68 rows=13857 width=77)
  Join Filter: (flybys.analysis_window @> readings.time_stamp)
  -> Seq Scan on flybys (cost=0.00..2.58 rows=1 width=96)
      Filter: (id = 4)
  -> Seq Scan on readings (cost=0.00..649140.76 rows=13857176 width=21)
(5 rows)
:
```

Two sequential scans, which is not good. This is because the `@>` operator can't be used with a BTREE index.

If the INMS data had range values, however, I could create an index on that range, which is where the power of **tsrange** comes from.

Let's say that the INMS recorded the beginning and end of the integration period (the little 30ms window in which it does its reading) using two fields called **ip_start** and **ip_end**. I could create a **tsrange** from that and then index it:

```
alter table inms.readings
add integration_period tsrange;

update inms.readings
set integration_period=tsrange(ip_start, ip_end, '[]');

create index idx_ip_range on inms.readings
using GIST(integration_period);
```

The GIST index respects the range operators, which means the contains query I ran before (using `@>`) will be able to use this index.

You can do all kinds of queries, including overlap, exclusion, containment, existence and more. I found a great write-up about it [here](#).

Unfortunately for me, this doesn't help. I'm not querying a range directly — I'm using a range to constrain results. It's interesting to know, however.

COMPARING THE SPEEDS

November 6, 2017, 1210

Before I do anything I need to know how these speeds match up: the ones I pulled from the JPL website and the ones I just figured out with the INMS data. I just can't let that failure go! Even if it actually worked well enough, entering data by hand, from a website of all things, just feels like failure to me. I need to know a better source of speed data is out there! I also want to see if I can calculate the speeds for the first 2 flybys.

I'll just average the speed for this first look; it won't be changing that much in the few seconds that Cassini zoomed past Enceladus:

```
inner join inms.readings on
  time_stamp >= flybys.window_start and
  time_stamp <= flybys.window_end
group by speed;
```

You have *got* to be kidding me:

Red:4 Planetary Orbital Observations			
id	name	speed	avg
1	E-0		6.6
2	E-1		6.6
3	E-2	8.2	8.2
4	E-3	14.4	14.4
5	E-4	17.7	17.7
6	E-5	17.7	17.7
7	E-6	17.7	17.7
8	E-7	7.7	7.7
9	E-8	7.7	7.7
10	E-9	6.5	6.5
11	E-10	6.5	6.6
12	E-11	6.8	6.8
13	E-12	6.3	6.3
14	E-13	6.2	6.2
15	E-14	7.4	7.4
16	E-15	7.4	7.4
17	E-16	7.4	7.4
18	E-17	7.5	7.5
19	E-18	7.5	7.5
20	E-19	7.5	7.5
21	E-20	8.5	8.5
22	E-21	8.5	8.5
23	E-22	9.5	9.5
(23 rows)			

Well, there it is then! I guess I know how they calculated the speed of each flyby! It's a perfect match! **VICTORY!!!!**

I have to tell Eno...

From: Eno L eno@redfour.io
Subject: RE: Coming correct with speedz!
To: Dee Yan yand@redfour.io
Date: November 6, 2017

Great work Dee! I was hoping that PDF would help, don't know why I didn't think of sending it to you earlier. We've been studying so many white papers on INMS analysis, and sometimes I get so rabbit-holed I forget that other people could use some of the information I'm studying.

I was talking to Karl again, and we're happy sticking with the time window that you put together. If we need to adjust it, we'll do it during the query.

Glad to hear you'll be by on Wednesday. Rob mentioned that he would be flying back up here for 2 days. Apparently things are going really well down there, but they want some specific reports that we'll be generating for him based on the data you've pulled together. He decided to come up here and help us out however he can. I think we're covered entirely with what you just pulled together: great job.

Will be fun to hang out with the 2 of u. We should go to Specs! Best

E

PUTTING IT TOGETHER: A CHEMICAL QUERY

November 6, 2017, 1436

Wow. It feels so good when things go right. I have to say: I'm proud of myself. I think I don't say that enough, and I really should do it more often as it feels kind of nice.

Also: I love Specs! Except for the walrus thingy hanging on the wall. That will be fun; maybe I can run a few reports to show them and see what they say.

Back to work. This is where the fun starts! They didn't ask me for this, but there's no way I'm holding back. I really want to see the chemical analysis results!

The **chem_data** table I loaded up earlier has a **peak** column, which is AMU/Z — or Daltons. The INMS has the exact same unit type with the **mass_per_charge** column. I can relate the two and find out which molecules were detected when:

```
-- first look at chem results
select
    flybys.name,
    time_stamp,
    inms.readings.altitude,
    inms.chem_data.name as chem
from inms.readings
inner join flybys on
    time_stamp >= flybys.window_start
    and time_stamp <= flybys.window_end
inner join inms.chem_data on
    peak = mass_per_charge
where flybys.id = 4;
```

Holy shit! Look at that:

Red:4 Planetary Orbital Observations

name	time_stamp	altitude	chem
E-3	2008-03-12 19:05:51.521	169.46	Acetylene
E-3	2008-03-12 19:05:51.521	169.46	Acetylene
E-3	2008-03-12 19:05:51.521	169.46	Acetylene
E-3	2008-03-12 19:05:51.521	169.46	Acetylene
E-3	2008-03-12 19:05:51.555	169.12	Hydrogen cyanide
E-3	2008-03-12 19:05:51.589	168.77	Ethane
E-3	2008-03-12 19:05:51.589	168.77	Ethane
E-3	2008-03-12 19:05:51.589	168.77	Ethane
E-3	2008-03-12 19:05:51.589	168.77	Ethane
E-3	2008-03-12 19:05:51.589	168.77	Carbon Monoxide
E-3	2008-03-12 19:05:51.589	168.77	Carbon Monoxide
E-3	2008-03-12 19:05:51.589	168.77	Carbon Monoxide
E-3	2008-03-12 19:05:51.589	168.77	Carbon Monoxide
E-3	2008-03-12 19:05:51.589	168.77	Ethylene
E-3	2008-03-12 19:05:51.589	168.77	Ethylene
E-3	2008-03-12 19:05:51.589	168.77	Ethylene
E-3	2008-03-12 19:05:51.589	168.77	Ethylene
E-3	2008-03-12 19:05:51.589	168.77	Molecular Nitrogen
E-3	2008-03-12 19:05:51.589	168.77	Molecular Nitrogen
E-3	2008-03-12 19:05:51.589	168.77	Molecular Nitrogen
E-3	2008-03-12 19:05:51.589	168.77	Molecular Nitrogen
E-3	2008-03-12 19:05:51.589	168.77	Carbon Monoxide
E-3	2008-03-12 19:05:51.623	168.42	Propane

OMG OMG OMG! Damn, I love this job! Why yes, as a matter of fact, Cassini registered the presence of Hydrogen Cyanide, a deadly poison, at 169.12 km above Enceladus at precisely 7:05pm and 51 seconds on March 12, 2008. Why do you ask?

Scrolling through this data, I can almost see the clouds of chemicals wafting up from the surface, suspended in the plume material. Cassini speeds through with the doors of the INMS and CDA wide open to catch the cosmic bits.

Let's see what else lies hidden in this data set...

EXPLORING INMS READINGS

November 6, 2017, 1652

I spent the last few hours reading a few more white papers on the INMS, specifically what the results mean and how to use the data. This is really not my area of expertise, but I can't help myself: I really want to look around at what's here.

Scientists are still analyzing the INMS data, in detail. Some are revising their calculations based on cross-referencing with other instruments or contamination from the vessel itself.

For instance: as Cassini approaches a close flyby, it has to engage its thrusters to mitigate drag from the atmosphere. There were two identical thrust systems on Cassini, a primary and a backup. Both used hydrazine and/or hybrid hydrazine fuel source. When hydrazine is catalyzed (using iridium) it gives off nitrogen, hydrogen, and ammonia.

Yes, just like in *The Martian* when Watney blew up the HAB. Love that book.

The problem is that this thruster exhaust can contaminate the INMS readings. This is especially problematic when encountering smaller moons like Enceladus, which don't have a dense atmosphere that can essentially "blow the exhaust away."

Another issue is leakage. A white paper, entitled "[A Revised Sensitivity Model for Cassini INMS: Results at Titan](#)" (Teolis, B.D., Niemann, H.B., Waite, J.H. et al. Space Sci Rev (2015) 190: 47, discusses how Titan flyby results yielded lower ion and neutron densities than expected. They suggest this is the result of gas leakage and an error in calibration, among other things. As such, they propose a revised sensitivity model, which increases densities by a constant factor of 1.55%.

Finally, the Cassini team [discussed](#) how they were able to get (what they considered to be) reliable readings of molecular hydrogen:

We used the Ion Neutral Mass Spectrometer onboard the Cassini spacecraft to detect molecular hydrogen in the plume. By using the instrument's open-

source mode, background processes of hydrogen production in the instrument were minimized and quantified, enabling the identification of a statistically significant signal of hydrogen native to Enceladus. We find that the most plausible source of this hydrogen is ongoing hydrothermal reactions of rock containing reduced minerals and organic materials. The relatively high hydrogen abundance in the plume signals thermodynamic disequilibrium that favors the formation of methane from CO₂ in Enceladus' ocean.

Long story short: it seems the INMS data is still being analyzed and interpreted. There is methane, CO₂, N₂, and H₂ in the plumes of Enceladus... the question is how much.

I'll leave that to the Analysis Team. I'm sure they'll have their models and corrections, and besides, it's not my job to analyze the data, just to make sure it's as correctly structured as possible.

I think I've done that. I also think that nothing is going to stop me from poking around!

High and Low Sensitivity Counts

The first thing I want to do is to take a look at the relative density of each chemical compound. In the INMS manifest, you can figure this out using the **C1COUNTS** and **C2-COUNTS**, which are described as “high sensitivity counts” and “low sensitivity counts” respectively. Let's take a look at E-3, one of the targeted flybys that Eno wants to see:

```
-- with density counters
select
  inms.chem_data.name,
  sum(high_counts) as high_counts,
  sum(low_counts) as low_counts
from flybys
inner join inms.readings on
  time_stamp >= flybys.window_start and
  time_stamp <= flybys.window_end
inner join inms.chem_data on peak = mass_per_charge
where flybys.id = 4
group by inms.chem_data.name, flybys.speed
order by high_counts desc;
```

Fascinating:

Red:4 Planetary Orbital Observations		
name	high_counts	low_counts
Molecular Hydrogen	5904	16
Carbon Monoxide	2972	6
Ethylene	2348	4
Molecular Nitrogen	2348	4
Ethane	2348	4
Methane	816	16
Carbon Dioxide	440	4
Acetylene	348	12
Water	297	2
Methane (isotope)	268	8
Propane	258	5
Molecular Nitrogen (isotope)	148	4
Hydrogen cyanide	88	5
Argon	20	0
Molecular Oxygen	16	0
Krypton	12	8
Helium	12	4
Neon	8	4
Benzene	5	5
Propyne	3	0
Propiolonitrile	3	0
Cyanogen	2	0
Butadiyne	0	2
(23 rows)		
enceladus=#		

Querying Counts by Source

Outstanding, but there's more that I need to add to this query before I can draw any conclusions. The high reading of molecular hydrogen is misleading!

If I add **source** to this query, it will tell me something about the analysis of each of these chemicals, and how it was performed.

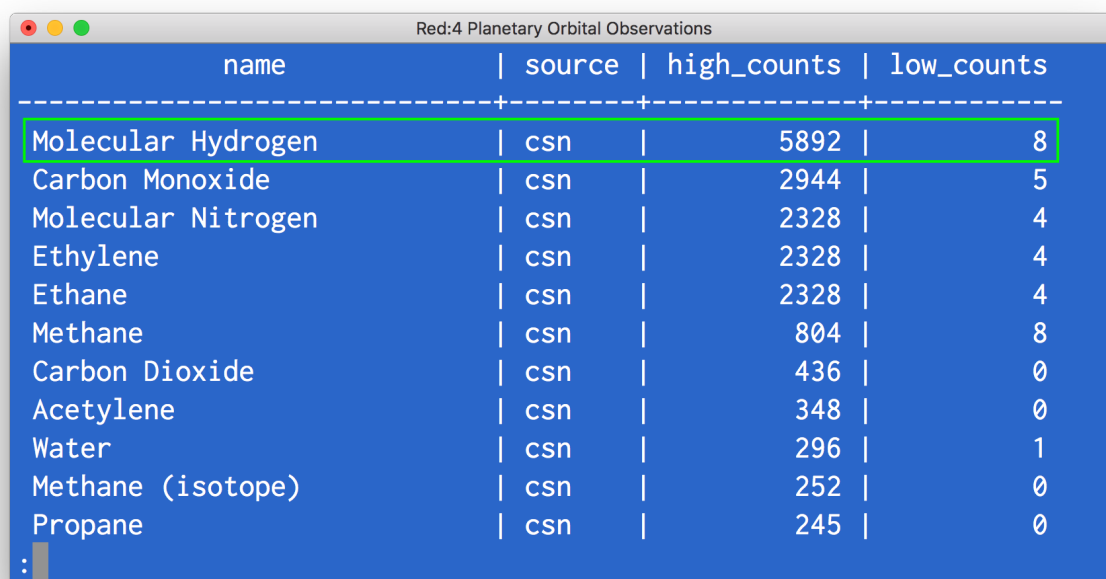
The **SOURCE** is described in the manifest as: Ion source used for this measurement,

- Open Source Ion (osi)
- Closed Source Neutral(csn)
- Open Source Neutral Beam (osnb)
- Open Souce Neutral Thermal (osnt)

Basically: closed source is used to measure non-reactive neutrals like CH₄ and N₂. Open source measures positive-ion species.

The results of flyby E-3 show that molecular hydrogen (H₂) was found using closed-source mode:

```
-- E-3 and Molecular Hydrogen
select
  inms.chem_data.name,
  source,
  sum(high_counts) as high_counts,
  sum(low_counts) as low_counts
from flybys
inner join inms.readings on
  time_stamp >= flybys.window_start and
  time_stamp <= flybys.window_end
inner join inms.chem_data on peak = mass_per_charge
where flybys.id = 4
group by inms.chem_data.name, source
order by high_counts desc;
```



name	source	high_counts	low_counts
Molecular Hydrogen	csn	5892	8
Carbon Monoxide	csn	2944	5
Molecular Nitrogen	csn	2328	4
Ethylene	csn	2328	4
Ethane	csn	2328	4
Methane	csn	804	8
Carbon Dioxide	csn	436	0
Acetylene	csn	348	0
Water	csn	296	1
Methane (isotope)	csn	252	0
Propane	csn	245	0

The problem with using closed-source mode is the chamber itself (the thing that makes this mode “closed”). It can react with certain species and throw off the results.

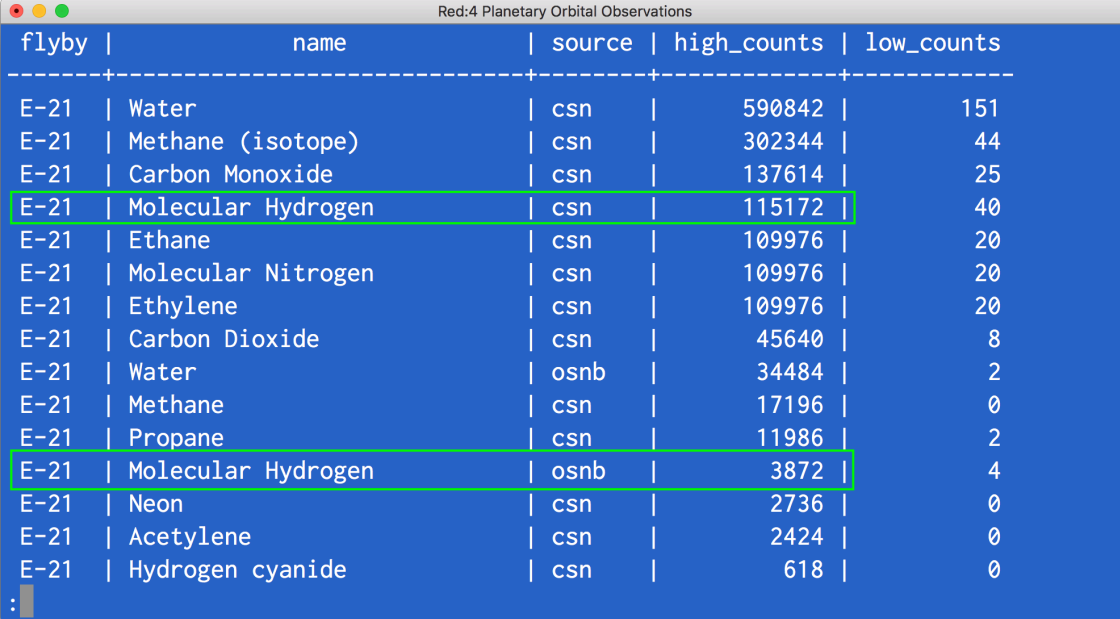
As can be seen in the results, water was also found, which is H₂O. The H₂ results could have something to do with that, as was suggested in [this white paper](#) from the 47th Lunar and Planetary Conference, 2016 shoutingly entitled HOW MUCH HYDROTHERMAL HYDROGEN MIGHT WE FIND IN ENCELADUS’ PLUME?:

During previous Cassini flybys of Enceladus, the Ion and Neutral Mass Spectrometer (INMS) detected counts at mass channel 2 in closed source neutral mode that are attributed to H₂. The signal was enhanced at faster flyby velocities as a result of impact-induced chemistry in the antechamber of the instrument, but up to ~15% H₂ was still detected consistently during the slowest flybys. At present, it is unclear if this H₂ is native to the plume or an artifact of highspeed sampling of the H₂O-rich plume. In an attempt to resolve this ques-

tion, a search for H₂ was performed using the open source neutral beam mode of INMS during the E21 flyby, for which the data are being analyzed.

Open Source Molecular Hydrogen

Open source mode lets plume material directly into the instrument. I'll verify this by taking a look at E-21. I'll also add in the flyby name to make sure we're looking at the right information:



flyby	name	source	high_counts	low_counts
E-21	Water	csn	590842	151
E-21	Methane (isotope)	csn	302344	44
E-21	Carbon Monoxide	csn	137614	25
E-21	Molecular Hydrogen	csn	115172	40
E-21	Ethane	csn	109976	20
E-21	Molecular Nitrogen	csn	109976	20
E-21	Ethylene	csn	109976	20
E-21	Carbon Dioxide	csn	45640	8
E-21	Water	osnb	34484	2
E-21	Methane	csn	17196	0
E-21	Propane	csn	11986	2
E-21	Molecular Hydrogen	osnb	3872	4
E-21	Neon	csn	2736	0
E-21	Acetylene	csn	2424	0
E-21	Hydrogen cyanide	csn	618	0

And there it is! Reported for both closed and open sourced modes! The difference with the open source reading is that it gave NASA a higher level of confidence that the H₂ was coming from the plume itself rather than as a byproduct of analysis or, worse, contamination.

A white paper and announcement soon followed these results, with some scientists thinking it more and more likely that we will, indeed, find life on Enceladus:

Indeed, it has been proposed that there are hydrothermal vents inside Enceladus to explain the formation of silica nanoparticles that are derived from Enceladus. It has also been suggested that Enceladus has an alkaline ocean as a result of serpentinization reactions between water and rocks. If there are serpentinizing hydrothermal systems on Enceladus that are currently active, then we should search for other clues in the plume to confirm their existence. In this respect, a smoking gun may be molecular hydrogen (H₂), which is abundant in hydrothermal systems on Earth (in particular Lost City) that may be analogous to those on Enceladus.

That quote is from the same white paper referenced above. It's also worth noting that this paper was a joint effort between the INMS team, the Southwest Research Institute and Cornell University. It's not some random hack scientist making sensational claims!

ONE LAST QUERY

It's getting late, and M. Sullivan is going to be hounding me any minute... I need to do one last query. One crucial, last query that could answer one of the most significant questions of modern man.

Are we alone in the universe?

```

-- are we alone in the universe?
-- such a small query for such a big question...
select
  flybys.name as flyby,
  inms.chem_data.name,
  source,
  sum(high_counts) as high_counts,
  sum(low_counts) as low_counts
from flybys
inner join inms.readings on
  time_stamp >= flybys.window_start and
  time_stamp <= flybys.window_end
inner join inms.chem_data on peak = mass_per_charge
where flybys.targeted = true
and formula in ('H2', 'CH4', 'CO2', 'H2O')
group by flybys.id, flybys.name, inms.chem_data.name, source
order by flybys.id;

```

Look at that. All the chemical results for H₂, CH₄, CO₂, and H₂O. The chemical thumbprint of life:

flyby	name	source	high_counts	low_counts
E-2	Methane	csn	76	4
E-2	Methane	osi	4	0
E-2	Water	csn	101	0
E-2	Water	osi	1	0
E-2	Molecular Hydrogen	osi	4	4
E-2	Carbon Dioxide	osnb	0	0
E-2	Methane	osnb	0	0
E-2	Molecular Hydrogen	csn	744	0
E-2	Carbon Dioxide	osi	0	0
E-2	Water	osnb	1	0
E-2	Carbon Dioxide	csn	32	0
E-4	Water	csn	517	0
E-4	Molecular Hydrogen	osnb	20	0
E-4	Carbon Dioxide	csn	420	4
E-4	Carbon Dioxide	osnb	0	0
E-4	Methane	csn	936	4
E-4	Molecular Hydrogen	csn	4856	0
E-4	Water	osnb	4	0
E-4	Methane	osnb	16	0
E-6	Carbon Dioxide	osnb	0	0
E-6	Water	csn	419	0
E-6	Molecular Hydrogen	osnb	20	4
E-6	Carbon Dioxide	csn	548	0
E-6	Water	osnb	2	1

It doesn't prove anything, of course, as Cassini could not detect life. The data suggest, however, that it detected everything *but* life.

Methanogenesis is the process by which microbes in the deep ocean trenches convert H₂ and CO₂ to CH₄ and Water. This is the basis of the deep sea trench ecosystem that exists, basically, in parallel to the ecosystem that the rest of the earth lives in.

Each of these molecules was detected at Enceladus, by Cassini, as we can see from the above query. Both the food and the excreta of methanogenesis.

This process describes what's known as an [Origin of Life Reactor](#) on Enceladus, much like the one describing deep-sea vents here on earth:

What is remarkable and exciting about these vents is that they offer simple geochemical equivalents to the biological processes still operating in bacteria living there today. These cells gain both energy and organic carbon from the reaction of H₂ with CO₂. The enzymes responsible contain clusters of iron, nickel and molybdenum sulfide that are almost identical in structure to minerals found in vents. Most intriguingly of all, these bacteria can only grow by using proton gradients across membranes, exactly like those found in the vents. Recent theoretical work indicates why these properties are so important; but the theory has never been tested experimentally.

I always wondered if I would be alive when life was found elsewhere in space. Our next trip to Saturn should be unbelievable.

GLIDE PATH

November 7, 2017, 2352

Met with Rob and Eno today at the Funston office, and had a chance to meet a lot of the Analytics Team (The “80s”, as they call themselves, har har). After a few hours, we headed out to Specs and got into a ... bizarre conversation.

I pretended to text my mom while I took notes. I have a feeling I’ll want to remember the details of what was said for a very long time. Rob is... an interesting guy. A bit guarded and at the same time idealistic. I hope he wasn’t offended by what I had to say.

I need to take my time documenting this, but I have a browser tab open, ready to purchase a ticket to Iceland. I feel like it’s staring at me, daring me to click “Purchase.”

After I write this down. Then I’ll decide.

Rob’s plane landed at SFO at noon, and he said he would head directly to the Funston office, so we set the meeting for 1500. I took the morning off as Eno let me know the Analytics Team had all they needed for the “Glide Path.”

Curious name, that.

I left my mom’s house after lunch and was about to walk through the Palace of Fine Arts, but I decided on the long way: through Crissy Field, along the beach path where tourists, kite surfers and people with really expensive dogs meander their way through dunes being restored to their pre-military, natural state.

I used to come down here with my dad on Friday afternoons after school. His company was involved in a colossal cleanup and restoration project that took place as the army transitioned the entire Presidio to the National Park Service, which sought to restore the pristine land back to its natural state.

The Presidio is located at the strategic northern tip of the San Francisco peninsula and is a perfect place to shoot at things you don't want coming through the Golden Gate. The Spanish knew this in the 1770s, so they set up a fort there. That got handed to the Mexicans and then to the Americans in the 1840s, who used it as a base for numerous operations.

My dad would tell fascinating stories about cleaning up small pockets of contamination, a lot of it lead in the soil and lamp oil that leaked from underground tanks. Interesting stuff, but I liked what my dad's job was. He had to read tide gauges along the waterfront to better understand the effect of the tides on groundwater.

A series of wells were put in, all along Crissy Field, and I would run down there after school and hang out with him, watching windsurfers and huge ships go in and out of the Golden Gate.

He always had an apple juice for me, the one in the glass bottle shaped like an apple.

"You can see the effect of the moon and the sun on the Earth, right here Deedle" he would say as he showed me the printout from the gauge at the wellhead. Up, down, up, down in a perfect rhythm. "That's called a sine wave, and you can imagine it as a real wave, washing over everything, twice a day."

I can still see these wells today, even though Crissy Field looks very, very different. The sun is out once again, and it's one of those rare days where there is no wind at all, and it's colder than it ought to be.

I walk down to McDowell then under the freeway, right by the old Commissary where you used to be able to buy the best pretzels for 20 cents. From there I walked down Lincoln and then over to North drive where the parade grounds used to be, as well as the officer's club.

It's so strange to see all of these old, historical army buildings refurbished and rebuilt into modern businesses. A few are high tech, others are just regular offices. One of them is an aerospace startup, which is where I'm bound.

I finally make it over to the Funston office at 1455 and ask for Eno at the front desk.

After a few minutes, a tall black woman about my age walks around the corner with a huge smile on her face. Her hair is stylishly braided and pinned back, her employee badge around her neck shows the exact same beaming smile.

"Dee! So good to finally meet you!"

We chat for a bit, and she shows me around the old office building, which has been completely renovated inside – if you can call it that. Exposed ducts and pipes painted over in red and black line the ceiling. Tables and desks are all over the place, randomly, in an open floor plan which instantly makes me feel claustrophobic.

And Star Wars posters. Everywhere. Buzz Lightyears, ET, you name it. I just walked into a cliché.

I meet a few people, and we make our way to the back of the office, to a conference room, and I'm surprised to see Rob is already here.

"Dee! Good to see you again! Here have a seat. Can I get you anything?"

"A Diet Coke would be great" I reply. I then look over to Eno: "so, is M. Sullivan in this office? I've never met... him ... or her?"

Eno's stares back at me blankly, the smile dropping a little. I can tell she's calculating what to say next. "No, not at this office I'm afraid. Maybe ask Rob that question when he comes back."

"OK. Sorry, is this a weird question or something?" I don't want to make things awkward, but M. Sullivan did help me a ton... even if I would gladly have strangled whoever they turn out to be. Multiple times. "Is M a he? She? Non-binary?" I ask.

“No, no... it’s fine.” Eno looks around as if it’s anything *but* fine. Her eyes flick towards the door before I even register motion, and she sighs in relief. “Oh good, here’s Rob.” Eno says with relief on her face.

“Here’s Rob what?” Rob says, coming through the door, shutting it gently behind him. He offers me the Diet Coke and walks over to his chair and sits, leaning forward on the table.

“Dee was just asking about M. Sullivan, whether she could meet... him.” She giggles a bit, looking out the window as if something interesting just caught her eye. She’s trying not to laugh, and not in a funny ha ha that joke was good kind of way. More of an oh crap this is gonna be strange sort of way.

Rob stares for a second. “Oh, shit” He lets out a nervous laugh. “M. Sullivan. Right... damn, this isn’t normally how we do things here. I’ve been away and haven’t had a chance to talk to you about some stuff. Oh man, I can’t believe I did this” he says, staring at Eno with that half smile that says *I think I might be in trouble*.

“Is there something I need to know?” I ask, looking at both of them. I really don’t like where this is going.

MEET M. SULLIVAN

“M. Sullivan is a bot, Dee. It’s actually what we refer to as a collective bot. About 12 years ago, I thought it would be funny to put some monitoring triggers on our database server, scanning the logs for troublesome things like **select N+1** queries, non-sargeable stuff, too many indexes and so on. Rather than send out mean emails when people abused Postgres, I created a bit of an alter ego in M. Sullivan.” Rob says, staring at his hands as if hearing the words coming out of his mouth for the very first time. “Well, he’s actually based on a real person I know who inspired the idea and—”

“A... bot?” I ask, interrupting. “As in he doesn’t exist? Are you serious right now? Cause that’s messed up if you are.”

“Yes. Dee look: I’m really sorry. Normally it’s something I tell people immediately during orientation. That day was a bit rough, and I hadn’t slept, plus Sara was texting me and—”

“Yeah, I know, from the jet” I respond, probably a little too snappily. I don’t think I’ve ever experienced finding out that someone you’ve kind of bonded with, who’s given you good advice and who, well, you were starting to become fond of, is fake. As in “not real” sort of fake. I don’t think I like this feeling. “I don’t understand” I continue. “I’ve been emailing him. He’s been emailing me. We correspond. Either you’ve created some perfect artificial intelligence... or you’re full of shit right now” I say, still unbelieving. I probably shouldn’t be cursing, but I hate feeling like the ass end of someone else’s fun little joke.

Rob takes a deep breath, understanding that I am starting to get a little annoyed. “I’m... really, truly sorry. Please: let me explain. M. Sullivan started out as a simple ‘alarm system’ if you will. Our application developers were letting their ORMs create ridiculous queries, and I was getting tired of having the same conversations, so I decided to try to make it fun”

“People liked it and began to talk about M. Sullivan as if it was real. Discussing what it looked like, what it would wear, if it was male/female, animal, mineral or whatever,” he says. “Eventually people started making pictures of what M. Sullivan might look like. They got pretty creative. So did I. More elaborate alarms which turned into general notices. Next thing you know I’m sending emails to all hands as Sullivan, with the same return address. It started to spiral into something” he says.

It’s hard to not feel annoyed at this, but I have to admit I was growing curious. I liked M. Sullivan, even if he could be a pompous prick from time to time. I guess this explains the duplicate emails and the changing tone of some of them, but was all that just some alarm system Rob dreamed up? “That doesn’t explain the detailed emails. He knew what I was working on, he responded with SQL of all things. I know bots can be pretty good, but all they do is assemble words based on probability and structure.

There’s no way those emails were sent by a bot” I retort.

“They weren’t. Those emails were sent by someone here, at Red:4. They didn’t know who you were, they just read the text of your email. Your name is substituted by the system after the email is sent” Rob says. “That’s... the other side of M. Sullivan. It started as an alarm system and then—”

“What... the f—” I start, once again cutting him off.

“Let me back up,” he interjects. “This is totally my fault. From the top: M. Sullivan is a monitoring bot as well as an anonymized chat/help system. That’s what it’s grown into. People would send

M. Sullivan all kinds of notes, either in reply to a monitoring alarm or just for fun. I started responding as him just... I don’t know, because it was fun, and then one of our app devs, Paul, came up with the idea that we should all be able to respond. So he started routing certain requests to people, anonymizing who sent it. He let everyone know he was doing this and asked that the responder could do their best to assume the role of M. Sullivan”

“When they respond, they are anonymized as well. It’s like StackOverflow if it was a person, instead of a bunch of people, and no one could downvote you into oblivion” he says with a small laugh. “This is when M. Sullivan really took off for us. It was a perfect way to be direct, say – or ask – what you wanted, and not worry about social repercussions. No one knew who you were, and you never knew who, exactly, responded as they were each trying to be M. Sullivan. We’ve promised everyone anonymity, so people feel free to write as tersely as they want. Or as kindly.”

Now I’m interested. This can’t work! “Wait a minute. No way” I say. “People who believe their identity is hidden act like complete assholes almost consistently,” I say. “Trolls. Idiots on Twitter. It’s kind of a proven thing.”

“I don’t know what to tell you, Dee. It was different here.

Maybe it’s because we’re a small collective of people doing some really fun stuff? Maybe because there’s only one voice that everyone has to deal with? I have no idea” he says. “I mean sure, things have happened, and yes, we’ve had people complain. It’s not like there’s free rein and zero accountability. If someone abuses the privilege of

corresponding as M. Sullivan – messages are screened automatically, or we can flag them manually – they get a very direct warning. If your second message gets flagged, you're removed from the system. This has happened twice in 10 years. It's one of the reasons people enjoy working here" he says.

"I don't know." I really don't. "Like, maybe someday someone could, I don't know say something terrible and piss everyone off? What if someone writes a stupid manifesto based on their political beliefs?"

"They'll write it anyway, whether we have M. Sullivan or not.

You can't control people in that way, Dee. You might not like what they say, but preventing them from saying it is not a place we want to go."

"Not your *kuleana*" I say, using my mom's words. I wasn't in the habit of dopping Hawaiian (or pidgin, for that matter) into a conversation as it always felt pretentious. Sometimes, however, they just fit perfectly. I also don't want to get into a debate about when opinion becomes deranged hate, so I just let it go.

"Not sure what that word means, but OK," he says. "Let me ask you a question," Rob says, eager to move on. "Did M. Sullivan help you?"

"Yes. A lot" I reply. "And yes, some of his emails stung a bit, but it seemed to be... thoughtful in a way" I say, thinking back on his failure is a good pain screed.

"Someone here wrote that email, Dee. We have no way of knowing who it was, but isn't it comforting, in an odd way, that someone here cares about your work and doesn't need to be recognized for the effort?" Rob asks. "That's some zen shit right there, Dee. Come on" he says, trying to pull me back from the cliff.

I have to admit: it is indeed some zen shit. Someone I've (*possibly*) never met took the time to deliver a tough message, and they did it with care. They could have been 10 times as harsh, but they weren't.

"Some of those emails, though, were pretty snarky. The attitude was a kind of over the top. I thought he seemed like a cross between–"

"Snape and Willy Wonka, maybe?" Rob asks, cutting me off.

“I was going to say, Snape, yes, but ... well anyone sassy really.

What’s going on with that? Why the pompousness?” I ask.

“M. Sullivan has become our mascot, sort of. He’s a manifestation of all of us. In fact, one of our old interns said he was our *egregore*, an ‘autonomous psychic entity that represents the group mind.’ Look it up on Wikipedia, it’s fascinating.”

“I guess we’re all Snape or Wonka or Someone Sassy on the inside” Rob continues. “You have to work here for at least a year before you’re added to the system” Rob goes on. “We usually tell people right away what the deal is. I didn’t tell you as I was personally devastated and not sure I’d live through the day, so it kind of slipped my mind. Again: I’m sorry about that” he finishes, this time with a small smile.

I guess I believe him. Why not? Seems like a fun idea as long as you’re in on it and, honestly, who cares? There are so many other important things going on in the world. “So what’s the deal with the ‘M’? Is that his first name?” I ask.

“It’s something I stole from *The Hyperion Cantos*, my favorite book series by Dan Simmons. Every human character was addressed as M. Something, M. Severn, M. Lamia, M. Endymion and so on. If you’re an android, you’re addressed with an ‘A’: A. Betick, for example. It’s genderless and... well, interesting I guess. Some people thought it substituted for Mr, Mrs or Ms but it doesn’t.

Simmons has always kept it vague; which I thought was pretty neat. So I stole it” Rob says. “Also, one of the main characters, M. Severn, is human but his consciousness resides in the machine collective called ‘The Core,’ which I thought was kind of appropriate.”

“So I write an email, asking for help, and then what?” I ask. I’m intrigued in spite of myself.

“When you write M. Sullivan an email it gets routed to a Python app that Paul wrote. The text is parsed, tagged, anonymized and contextualized, and then an email is sent out to whomever the system thinks can answer the problem in the best way. Most of us here have an account on the system, and we’ve set preferences based on the things we’re

knowledgeable about. Astrobiology, orbital mechanics, Postgres queries... “ he says, trailing off. “Even travel and relationship advice.”

“It can’t possibly be 100% anonymous. If people know I’m working on the Enceladus data set they’re going to figure it out”

I reply. “Sorry, this still makes me feel a bit... like a punchline.” This is starting to annoy me once again.

“Completely understandable if you don’t know what the system is all about. If you do, then you can be as careful as you like. In your case, you got the help you needed and—”

“I feel like a complete ass,” I say, interrupting once again. I’m trying to remember the things I said, if there was something in there that would be ... I don’t know... a bit too personal maybe? But no; they were requests for help from someone here at work that I had, up until now, I guess, never met. Seems weird to think of it that way.

“We have no way of knowing who responded to you, Dee. All I can do is apologize for this; you should have known what was going on. M. Sullivan is a lot more than a help system, and many people here feel that it’s a real person. They see it in whatever way they need to at any given time: man, woman, friend, parent... whatever” Eno cuts in. “We find in M. Sullivan who we need when we need them. Honestly: no one here will ever say anything as they don’t know you wrote those emails. Sure, they could guess, but I was also working on that data. So was Rob, for that matter.”

“Personally, I appreciate the concept a lot,” she continues. “I edit myself obsessively as I write because I want every word to be well-placed, to express myself eloquently in text as in code or architecture, and yes, because I’m afraid of saying the wrong thing or coming off the wrong way. Gender plays into it too: I go further than do my male colleagues to ensure workplace communications are friendly, even anodyne, and I’m still the one who gets described as *intimidating*. It’s really helpful to avoid that mess.”

My growing anger fades a bit because, really, this is a pretty fun idea. An abstraction between co-workers where you could ask whatever question you want and give an answer however directly you feel. What an interesting way to communicate!

“M. Sullivan has morphed and improved over the years, Dee.

He – or however you personally envision our collective consciousness – has become a confidant and also a way to cut down on pitiful ‘HALP ME’ emails that slow everyone down. We’ve had a few people who don’t care for the idea, sure, but most do” Rob says.

Eno picks up where he leaves off. “I love M. Sullivan, she gives damn good advice on robotics. Look, Dee, I know this is probably a little weird, but I’d like to suggest we move on. There’s so much we need to talk about, and if we start going off on M. Sullivan stuff, we’ll be here all day, drawing on the whiteboard and pondering the depths of the human soul” Eno says, her smile dropping a bit.

“I can do that” I reply. “Where do we start?” “Let’s go over what you’ve found.”

Over the next three hours, I describe the database I’ve created, with M. Sullivan’s help, and relay all the things I’ve found, including the altitude data from the CDA and INMS, speed calculations and finally the chemical analysis, which I strictly didn’t need to do but... whatever. Rob and Eno ask a few questions about some of the SQL I’ve used and offer a few suggestions. Mostly optimizations and suggestions for indexing, but it leads us into the theoretical and that’s when time starts to fly by.

Eventually, Rob puts both hands on the table, looking at the clock. “This is good stuff, Dee,” he says to me. Then, to Eno: “you’re confident that you can use this with Glide Path?” Rob asks, apparently trying to wind things up.

“Yep. One of our quants is using the positional data to pinpoint the plume readings using Dee’s database. Nice and fast, everything reasonably designed and normalized. Good stuff. You should be proud of your work” Eno says, looking at me with that big smile again.

“Great. Any questions that you might have?” Rob asks me, clearly wanting to wrap this up.

“What the hell is Glide Path?” I ask. “The two of you keep talking about it and how you’re going to be using my database with it... I think I would like to know a bit more.”

Rob looks at me, then at Eno, and then back to me. I can tell he doesn't want to say what's on his mind, but I'm in no mood for dancing around the point. "Yes, I know I'm your temp DBA and that I'll probably be replaced soon, but come *on*. I just lit it up for you. The least you could do is tell me how this work is going to be used. I've got my security clearance, I signed your NDAs but, most of all: I *delivered*. Spill."

"I wanted to wait until we knew a bit more about funding is all, Dee. No one has made any decisions about your role in this. You've impressed all of us, even Sara, despite her less than enthusiastic emails. There are still a few things in play, and I wanted to be sure of the details" he says, as if it will be enough.

I stare back and don't say a thing. He looks over at Eno for help, she just shrugs. He takes a deep breath. "I guess you've been through enough. You've earned it. But I really want to get out of this office first as I'm hungry and I would really enjoy a nice, cold beer right about now. You're going to have countless questions, so let's be in a comfortable place with food and drinks, shall we?"

"I know! Let's head over to Specs. Dee and I were talking about this the other day. It's early enough that we can still get a table in the corner. I'll call Lyft," Eno finishes before I can say anything.

"Sounds good," I say. Enough is as good as a feast Mary Poppins used to say, and this is enough for me.

"You got it," Rob says as we all get up and gather our things. I grab my coat and laptop bag, Rob and Eno grab theirs, and 12.4 minutes later we're in a Lyft, headed to North Beach.

SPECS

November 7, 2017, 1911

I have a love-hate relationship with North Beach. It's so vibrant with San Francisco history, but so something else at the same time. Places like City Lights Bookstore, The

Stinking Rose and all the fantastic Italian food you can find. Just a few blocks away is Chinatown, one of the most iconic areas in all of San Francisco and a great place for just about anything.

Then there's The Condor Club, Hungry I and all the rest of the strip clubs that draw crowds from all over the Bay Area for a "night out in the city." Fights erupt, police get called; it's just a place I don't like to be past 10 at night. I don't care if you want to watch naked people dance, just check yourself.

And then there's Specs. Tucked away in an alley, out of view, the 12 Adler Museum and Cafe is, literally, a city landmark. San Francisco named it one of nine "legacy businesses," entitling it to grants and other considerations to help it stay open. The owner, Richard, recently turned 89 and you can still see him in here often.

The walls are covered with every kind of memorabilia, and if you like things clean and tidy then this is a place to avoid. It gets loud past 10pm as well, but right now it's perfect for the three of us. Rob and Eno haven't said a word on the way over aside from basic pleasantries, and the reason is simple enough: this conversation is meant for just the three of us. This makes me even more curious. Who cares if the Lyft driver hears us talking about deep space probes and cryovolcanoes?

Right then my phone pings. It's a text message I had been waiting for, and a smile spreads across my face. This is going to be a good night. I tap out a quick reply and push my phone back into my bag.

As expected, there aren't many people here on a Tuesday night, and we find ourselves a table in the back. There are 10 total people in the place, Tom Petty is playing at a reasonable volume, and we're mostly alone; no one can overhear us. Which is a good thing, as Rob starts right in.

"We're building a targeting system, Dee. That's what Glide Path is" he says. "We're trying to target the exact location in the sulci where H₂ has the highest concentration. If the jets are indeed hyper-localized, as the current theory suggests, then that means we should be able to pinpoint a location directly over them if we have accurate enough

positional data” he finishes. “We feel that some localized thermal activity is going to be more conducive to H₂ and methanogenesis; that’s where we want to be,” he says.

Rob is referring to the recent study that suggested the core of Enceladus is porous, and that the water ejected from the plumes comes from deep within the core and is, basically, firehosed from specific locations under the sulci. I never read anything that suggested the jets had different levels of chemical effluent, however. Why do we even care?

“I don’t understand. The sulci take up a 120 square kilometer area, and that’s not that big. We have all kinds of imagery as well: infrared, UV, regular light. What do we gain by being more precise?” I ask.

Eno is quick to answer. “An optimal entry point.”

“An... *entry point*?” I ask, slowly. “You mean, like, for entering. *Enceladus*. Do you plan on diving under the ice or something?” I ask.

“Sort of. As you know the icy crust is detached from the rocky core, which means it moves independently in 3 dimensions. It gets squeezed like a soft-boiled egg, but it also spins somewhat independently of the core. We think the jets are part of the core, so we need to be able to describe their movement over time if we’re going to predict where to go in” Eno says. “That’s the ML system that you’ve been seeding. We’re trying to graph it in two dimensions so we can predict chemical densities when SELFI gets up there” she continues.

“What do you mean ‘go in’ and ‘entry point’? I get the mapping part, but... are you sending a probe under the ice?” I ask.

“More than just a probe, Dee, but yes, that’s how it will start,” Rob replies. “We need visual confirmation, of course, for the discovery. Once we have that we’ll need to set up for future observations, mapping the abyssal plains, if there are any, and any bathymetric features we encounter. I’m sure there will be quite a few. We just need to be sure where to start looking.”

“So let me get this straight. You want to send a lander with the next mission that will land on the ice, bore through it somehow, and then take pictures and other readings?” I ask. “Oh my god...” I trail off. I literally cannot speak. Images of *The Matrix* fill my mind, when the machine creatures landed on the Nebuchadnezzar, using their lasers to try to cut their way in.

How is this even possible? Using heat? Some kind of massive boring device? If the water is salty, how will radio signals work?

The questions are piling up in my head, and I can’t even think of what to ask next.

“Enceladus, not Mars or the Moon, will be our first off-planet base,” Rob says, picking up the conversation. “We will find life there, and we will study it, non-stop, comparing it to our own, trying to understand how it came about and how it will evolve – if it will evolve – differently from us. Things are easier if we can just have a lab up there, don’t you think? We’ll also need machines and software to process and analyze the findings. You could be part of this, you know.” This sounds like he’s trying to persuade me into thinking ... something.

Then something occurs to me. “Why are the two of you so sure you’ll find life? I agree the findings so far look compelling, but you sound as if you already know something,” I say, growing suspicious. “Like you’re trying to keep something under wraps.”

“We’ll find life, Dee. Maybe not during this next mission, but for sure the missions after that,” Eno responds.

“You don’t know that! How is this science? You can’t focus on what you *want* to see!” Great. Now I’m echoing M. Sullivan’s sentiment from days ago. “Whatever is under that ice is... what’s under that ice. Even if it’s nothing. You can’t go into this believing there’s life up there, you’ll end up... I don’t know... making it look like there is or something” I finish.

“Precisely. But it won’t be a masquerade. If SELFI doesn’t detect life, we’ll *put it there ourselves*” Rob replies.

I’m shocked speechless. Almost. “You’ll... do *what*?”

“We’re planning on sending up 4 bio-containers, each carrying a different strain of bacterium. The first will have *Planococcus halocryophilus*, a recently-discovered species that can withstand frigid temperatures. Two other containers will have specimens of *Methanopyrus kandleri*, a thermophilic methanogen that would feel perfectly at home snuggled right up to one of those jets. A fourth container will be the hybrid container, with samples of each bacterium intermixed. The temperature will be rigorously controlled to force the evolution of the bacteria within so it can withstand extreme cold temperatures as well as extreme heat,” Eno says.

I start to giggle. I don’t know why. I know it’s a strange reaction, but that’s all that is coming out of me. This idea is so ridiculous, so brash, so fundamentally wrong-headed that it simply strikes me as funny.

“Yeah! Exciting isn’t it?” Eno says, misreading my laughter. “If we go through with the release protocol, one of these species should be able to adjust and evolve in the Enceladus submarine environment in very short order. When we come back on a future mission, say in 10 or so years, we’ll likely see a thriving ecosystem starting to take shape.”

I’m still giggling, but I manage to stammer out “but, you don’t know there will be a future mission, do you? If you don’t find life, NASA won’t go back. The government would never fund a second mission just to look around a second time.” Tears are starting to come down my cheeks. Of all the things wrong with this plan, a future mission is the only thing I can think to bring up. That makes me laugh even harder.

“We’ll find life because we’ll put it there, Dee,” Rob says, repeating himself. “We’ll leave that part out, though. We’ll just announce that we have visual and biological proof of life in Enceladus’s submarine environment. Which will be true... once we put it there. Future missions will be guaranteed.”

The absurdity of the plan is astounding, but also intoxicating. “You guys are out of your fucking minds,” I say, almost cackling now. It’s not that I don’t believe them, which would be understandable given the revelation about M. Sullivan. I do believe them, and I think this exact scenario is going to happen, and ... it’s hysterical.

Of all the things wrong in the world, all the ways in which people are coping without food and clean water, corrupt governments and diseases taking lives by the millions. Here we are, discussing something that is, in human terms, a somewhat fanciful concern. I love the idea of space exploration, but this isn't that.

Tears are rolling freely now as I take a deep breath to calm my laughter. I down what's left of my drink, wondering where this conversation will end.

PLAYING GOD

"I understand if it's upsetting to you, Dee. I should remind you at this point of the classified nature of what I've just told you.

Also that you've signed some pretty binding non-disclosure agreements." Rob says, looking at me gravely.

I'm trying my best to stop laughing, and I focus on a drop of cold liquid running down the side of the martini glass containing the newly-poured Cosmo I just ordered. A candle is lit behind the glass, its dancing blood-red flame distorted by the condensation adhering to the side of the martini glass's iconic shape.

Rob and Eno haven't said a thing for a few minutes now, watching for my reaction. I think carefully about my next words, hoping the alcohol has given me just enough courage.

"Is this the part where you expect me to lecture you about 'playing god' or some other noble cliché?" I ask. Rob looks at me, puzzled. Eno's brow furrows. I add "Dude, get *over* yourself" and then I'm giggling again.

"I don't know what you mean, Dee." Rob replies. "This is going to be the greatest discovery in human history! Alien life! We're going to prove that life can exist elsewhere in the universe, even if it doesn't already... it doesn't matter! If our microbes can thrive there, then it's just as true." Rob adds, getting a bit more animated.

“You’re serious, aren’t you?” I ask him. “Eno, do you believe this bullshit?”

“I do, Dee. I think that this discovery will help push humanity beyond the tribal thought patterns we’re so stuck on, the outmoded ways and beliefs that have caused so much hurt and destruction over the millennia because we couldn’t be bothered to *observe* and *think* for ourselves.. I’m all for having faith in whatever god you want, but not at the expense of science and the scientific method,” Eno says. “This could be a new renaissance.”

I feel like my eyes are about to roll out of my head. “Wow we’re covering all the clichés tonight! Is this the part where I jump in with a clever retort that science is just another kind of faith and then we have witty banter about god and the meaning of life?

Seriously: why did you bring me here tonight?”

“I understand if this is upsetting to you, I really do. When I was first told of this plan I didn’t quite like it myself” Rob says, and the placating tone of his voice is like teeth on a chalkboard. “But then Sara said something that made me think. She said ‘you know, Rob, that Cassini did a death dive into Saturn’s atmosphere, right? They did that to avoid contaminating possible life-bearing moons, such as Titan and Enceladus. That’s why we didn’t land there: because we knew that microbes were likely contaminating Cassini, frozen onto its various surfaces and instruments’. At first, I didn’t get it, but then what Sara was saying to me started to make sense.”

“What’s that?” I ask, really trying to keep a straight face. “We’ve already contaminated Saturn. If these microbes can suspend themselves and remain viable through the vacuum of space,

Saturn’s atmosphere won’t make a damn bit of difference. The thinking was that Cassini would burn up and take the microbes with it, but Saturn’s atmosphere is so dense, so rich in some places that there is no way all the microbes burned up. Some must have been blown off, and are, right now, floating in suspension in Saturn’s upper atmosphere.”

“I guess I never thought of that” I reply. It’s true: I hadn’t. Did it matter if a microbe was in suspension or not? I guess you could make a pretty good argument that, as of

September 15, 2007, life exists on Saturn. It's frozen and may never actually revive, but does that even matter?

I'm remembering an article in *The New Yorker* that my mom sent me a couple of years ago. There was a great quote from one of the NASA administrators: "We know there's life on Mars already because we sent it there." Kind of an astounding admission coming from NASA, I recall. Oh yeah! That was also the article where I learned that NASA had a biologist in the role of Planetary Protection Officer, whose sole job it was to protect against the exact thing Rob was now proposing. I began to wonder if she ever gave out citations, and the giggling starts again. Rob, in little handcuffs that look like Saturnian rings...

Rob and Eno are watching me as if I'm wasted, but they know I'm only starting on my second. If I'm drunk on anything, it's a contact buzz from the sheer hubris it takes to plan something like this. It's impossible to mount a cogent argument against something this insane, but I pull myself together and try anyway.

"Look," I start, "your idea is so very absurd and, well, wrong on so many levels – at least many people will think so. I think you know this. I also think you've already come up with the arguments against just about anything I'm going to say, so I'll just come right out and tell you that it doesn't matter to me. I like this job, I love space exploration... but what you're wanting to do is kind of... absurd. But so what? Have at it, I say." I can't tell if they think I'm being serious or not.

"I know that ethics, when it comes to data science, is massively under-discussed," Eno says. "Programmers are building out these machine learning systems, and they have no idea what they'll be used for. Graphing systems, social network data mining; crazy powerful prediction engines that are built and just handed over, without anyone stopping to ask how it will be used.' I know this is probably eating at you, but this is NASA's data, you're just putting it all together for—"

"I'm thinking the two of you take yourselves far too seriously, and I think you don't know me like you think you do" I cut her off. "No one cares. How can you not know this?"

“You can’t be serious, Dee. This discovery will be the biggest discovery of all time! We’re not alone in the universe! God exists elsewhere too! Come on, you *have* to see the importance of this!” Rob pleads.

“Rob, seriously: I love you dude, but you have to know that no one cares about space anymore. Really, no one cares about science either. You did, when you were a kid. That’s all you had! My dad used to tell me all about it: 4 or 5 channels on TV, wait until Saturday morning to watch cartoons, spend the rest of your day playing with your friends, reading comic books about fighting space aliens, playing Han Solo trying to save Princess thingy” I say, somewhat calmer now, trying to lighten the mood.

“Leia” Rob corrects me, with a small smile.

“Right, Leia. Listen, let’s be real: if life were discovered on Enceladus tomorrow three things would happen. First: a super small part of the population would be really excited and probably throw numerous parties. Second: deniers and whack jobs would come out of the woodwork proclaiming that everything was faked and this was just a way for NASA to get funded. Third,” I pause for a sip of my drink before continuing, “is the rest of the world. Their response will probably be along the lines of: *so what?* Unless aliens are coming to abduct the Kardashians, the discovery of alien microbes won’t mean crap to the average person, who just wants to snap pictures of their food, argue on Facebook, have a nice dinner and fall asleep watching a sitcom. Or, if they’re in the underdeveloped parts of the world, they’re probably more concerned with, oh I don’t know, maybe clean water and a safe place to eat and sleep?” I say, wrapping up my thoughts.

“I’m sorry you feel that way, Dee,” Eno says, apparently not buying it. “I would have thought you would be a bit more optimistic about this. Being so close to one of the greatest discoveries of all time... you surprise me.”

“Me too. I was thinking that you and I would be working together for years. You did a really, really good job on the Enceladus database. You asked the right questions and sure, made a few mistakes here and there but, like you say: you delivered. I was hoping to offer you the job tonight” Rob says, clearly upset. “I was going to go over your qualifications in detail, which I never had a chance to do before I left. I was also hoping

that Eno could fill me in on some things I might have missed.” “What’s changed about that?” I ask.

“I don’t know. You tell me. Do you want this job?” Rob says.

I think I’m allergic to clichés. I swear I can see them coming from miles away, feel them swell into existence. It’s all I can do to not roll my eyes. He clearly thinks I’m going to pull some ingenue kind of thing, where he pulls me up from nowhere, shows me how to do such and such and then I set out on my own with my newfound knowledge. Barf.

“Oowwww!” I blurt out in my worst cockney, “I’m a good gel OYAM!” I say next, mimicking Eliza Doolittle. He just stares at me, clearly not understanding the reference. “My Fair Lady?”

Hello? Me Eliza, you Professor ‘enry ‘iggins” I say, smiling again. I look over to Eno, and she is smiling too; I think she knows where I’m going with this.

“I must admit you’re confusing me, Dee,” Rob says. Damn, that’s too bad. I thought it was pretty funny. My cockney was a bit loud, though. I may be taking this a bit too far.

“I’m having a little fun at your expense; I’m sorry. You and I haven’t really had a chance to get to know one another; you too, Eno,” I say, looking over at her. “It’s just, you know, you’re treating me as if this is some kind of tired old movie plot. Yes, of course: *I do want the job*. I’m just not into the storybook part of it. I need work” I say, wiping my lip.

“I took the job at Red:4 because every other job in this city sucks. Ridiculous startups convinced they’ll change the world or stodgy enterprises that write code with hammers and treat you like boxed meat. When I complain about these things, I’m called an ‘entitled millennial’ that needs to stop being ‘lazy.’ The only way I’ll *ever* be able to afford a house of my own is if I buy enough avocado toast to kill off the real estate sector, and don’t ask me how much I owe on my student loans.” I say, managing to keep an even tone even though I can feel the emotion swelling in me again, making me want to growl. “You grew up in a world full of possibilities, with moon landings, Star Trek, the Space Shuttle and the Voyager missions. The future, when you grew up, was wide open and amazing, with moon colonies and missions to Mars” I say, pointing at his chest.

“Yep,” he says. “I think I still believe it at some level. I also still watch Star Trek.”

“You want to know about my future?” I go on, “ocean level rising and drowning most of Manhattan and the islands in the south pacific. Arctic ice sheets breaking off, driving polar bears extinct and altering the ocean’s currents. Education and opportunity gutted so a few greedy old white men can wipe their asses with hundred dollar bills instead of mere fifties. Hunger, disease, terrorism, war – that’s *my* future” I say, ice in my voice. “There are no moon bases in my future, Rob. Just me and my generation doing our best to cope, and hopefully clean up the mess.”

I can read the look on his face: *I felt the same way you did when I was your age*, but he’s *akamai* not to say it out loud.

People are starting to fill the bar as well, which is probably why Rob has quieted down some. This isn’t a conversation that should be overheard and tweeted out.

I fish through my laptop bag, pulling out my phone, and fumble around with the apps until I find the PDF I was looking for. “Here,” I say, tapping the AirDrop button. “Here are my qualifications. I think you’ll find in there everything you need to know about me and how I do things.” Rant *over*.

Rob takes a deep breath, looking a bit frustrated, but there’s something else, too. His phone beeps, and he opens up the PDF, Eno slides over to look at it muttering something about “hipsters and their damn iPhones” as she gives me a wink.

“*A Curious Moon*,” Rob says. “What’s this?”

“My journal. I keep very detailed notes as if I’m talking to myself in the future, describing in detail what I’ve done, how I did it and why. This is the work I did for you, Rob, what I learned and how I learned it” I say as he scans the first few pages. “*The story is in the data*, as M. Sullivan told me, and it’s up to us to see that data without bias. You’ve filled in many gaps on your own tonight, I would say. Maybe look at what I’ve done and how I can help.”

“Oh, and I’ll be working remotely” I add.

“I appreciate that’s what you want to do, Dee,” Rob says, “but as you know–”

“I already accepted the position, Rob. Sara offered me the job on the way down here. She sent a text saying she was ‘impressed with my work and professionalism, given the circumstances, and that if you had a problem with it you could call her’” I say with a giggle. “She *also* said I could work anywhere I damn well please.”

“You’ve been corresponding with Sara?” he asks, apparently taken aback. Eno is just staring at me at this point, half smiling, half... something else. Awe maybe? Disgust? Hard to tell.

“Yep. She was all over me a few days back, telling me that you had taken it upon yourself to ignore her, for some reason. She didn’t appreciate the silence thing, so she started emailing me. Given that she’s the CEO and all, I emailed back, updating her on everything I had been doing. I also sent her a copy of the journal you’re looking at now. I guess she was impressed, and we started emailing every day.”

“Wow. Well... welcome, I suppose!” he says, lifting his glass. “Sara is indeed the smartest person at the company. I’ve never worked with anyone quite as talented as her, so, right *on*.”

I lift mine and look over at Eno. Her smile has returned, and her hands are still on the table. Then she raises one, giving me a high five.

“Dee, you rock,” she says, laughing.

EPILOG

December 18, 2017, 1354

Sitting in Cafe de Klepel next to Prinsengracht here in Amsterdam. I don't know if I'll ever leave. It's so riotously peaceful, not to mention astoundingly gorgeous, even if the canals are frozen over and full of crazy people trying to skate next to ice-bound boats:



We should hear the news in the next few days about Glide Path. NASA said they would make a decision “in time for the holidays,” and that would be now.

I like the job a lot. M. Sullivan is a constant source of amusement, and I can see why most people at Red:4 love him. That's how I refer to him now: as a he. Not sure why, but as Eno said: “we see in M. Sullivan who we need when we need them”. I keep asking him

about places to go while I'm here and damn if he doesn't always know the perfect little café for me, with wifi.

Most of the team are at the office today as we've been told to stand by, that critical news is incoming from NASA. This can only mean one thing: they have a decision to share with us. Part of me wants to be there, with everyone else. The other part of me wants to sit right here in the December cold, in Amsterdam, enjoying a coffee and a pancake.

The lid of my laptop is shut, and my phone sits beside it. I've managed to push work out of my mind for a bit, which reminds me I need to close off this little journal. I'd like to think of other things right now.



My phone just pinged. I can see It's a message from Rob. I think it can wait for a little. Maybe we got it, maybe we didn't.

if can, can. If no can, no can