Noteable

Surviving 100x scaling events

Picking the right technologies early

Matthew Seal, Co-Founder and CTO

Speaker: Matthew Seal
Role: Co-founder + CTO @ Noteable

Founded Noteable in Apr 2020

- We've built an extensive SaaS Data Collaboration System using Kubernetes, CockroachDB, FastAPI, and Jupyter
- Notebooks as an enablement tool for Data Stories

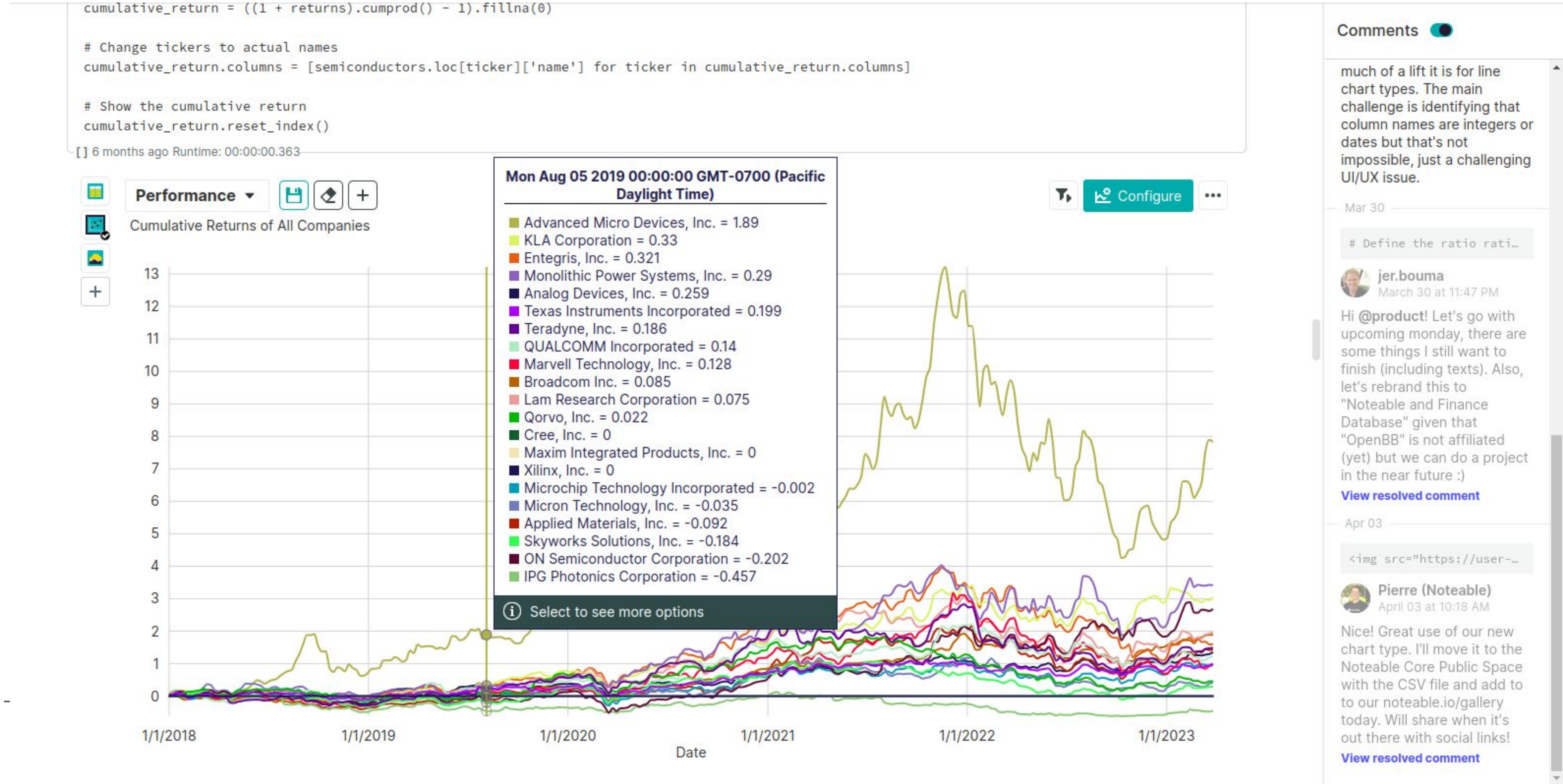
Previously at Netflix on their Big Data Team

- Worked with Spark, CockroachDB, Jupyter, Cassandra, Postgres, and more Data Systems
- Supported a range of ML and Analyst Systems

Core Contributor to Jupyter (e.g. Papermill)

Noteable

Noteable: Work with data the way you want



```
cumulative_return = ((1 + returns).cumprod() - 1).fillna(0)

# Change tickers to actual names
cumulative_return.columns = [semiconductors.loc[ticker]['name'] for ticker in cumulative_return.columns]

# Show the cumulative return
cumulative_return.reset_index()
```

[] 6 months ago Runtime: 00:00:00.363

Performance ▾

Cumulative Returns of All Companies

Mon Aug 05 2019 00:00:00 GMT-0700 (Pacific Daylight Time)

- ■ Advanced Micro Devices, Inc. = 1.89
- ■ KLA Corporation = 0.33
- ■ Entegris, Inc. = 0.321
- ■ Monolithic Power Systems, Inc. = 0.29
- ■ Analog Devices, Inc. = 0.259
- ■ Texas Instruments Incorporated = 0.199
- ■ Teradyne, Inc. = 0.186
- ■ QUALCOMM Incorporated = 0.14
- ■ Marvell Technology, Inc. = 0.128
- ■ Broadcom Inc. = 0.085
- ■ Lam Research Corporation = 0.075
- ■ Qorvo, Inc. = 0.022
- ■ Cree, Inc. = 0
- ■ Maxim Integrated Products, Inc. = 0
- ■ Xilinx, Inc. = 0
- ■ Microchip Technology Incorporated = -0.002
- ■ Micron Technology, Inc. = -0.035
- ■ Applied Materials, Inc. = -0.092
- ■ Skyworks Solutions, Inc. = -0.184
- ■ ON Semiconductor Corporation = -0.202
- ■ IPG Photonics Corporation = -0.457

ⓘ Select to see more options

Comments

much of a lift it is for line chart types. The main challenge is identifying that column names are integers or dates but that's not impossible, just a challenging UI/UX issue.

Mar 30

`# Define the ratio rati…`

jer.bouma
March 30 at 11:47 PM

Hi **@product**! Let's go with upcoming monday, there are some things I still want to finish (including texts). Also, let's rebrand this to "Noteable and Finance Database" given that "OpenBB" is not affiliated (yet) but we can do a project in the near future :)

View resolved comment

Apr 03

`<img src="https://user-…`

Pierre (Noteable)
April 03 at 10:18 AM

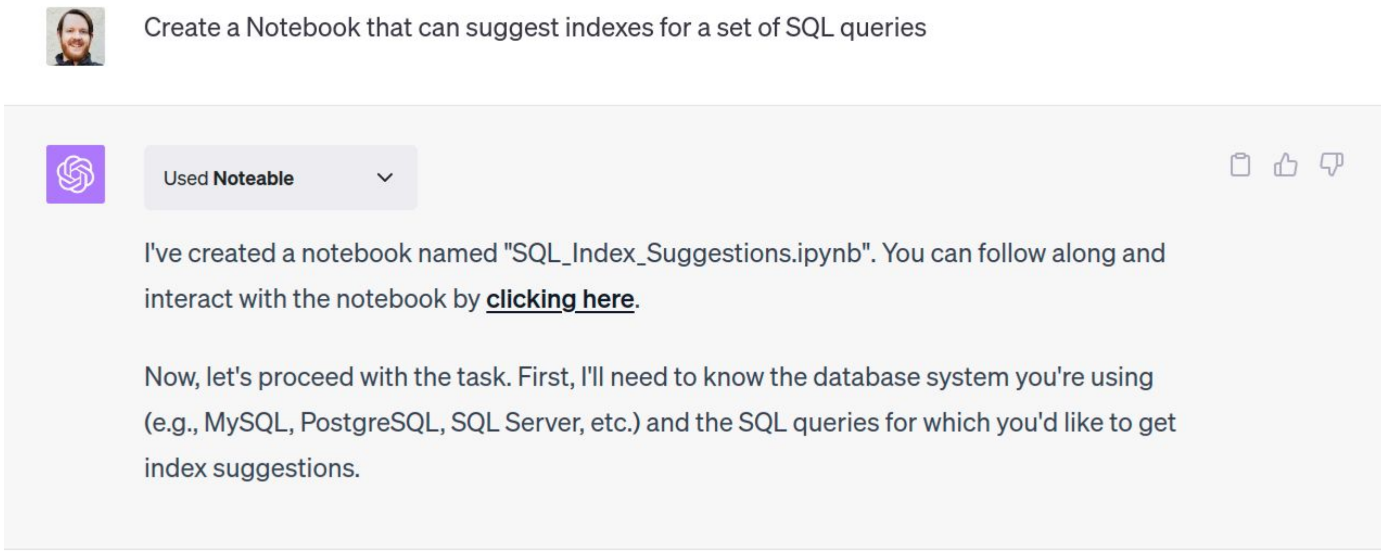
Nice! Great use of our new chart type. I'll move it to the Noteable Core Public Space with the CSV file and add to to our noteable.io/gallery today. Will share when it's out there with social links!

View resolved comment

LLMs and Chat Plugins

The launch of Large Language Model driven AI systems enabled a large step in ease of technology use by everyone.

Audiences that had non-existent tooling now had a dynamic and response interface to engage in problem exploration: **AI Chat**.



Create a Notebook that can suggest indexes for a set of SQL queries

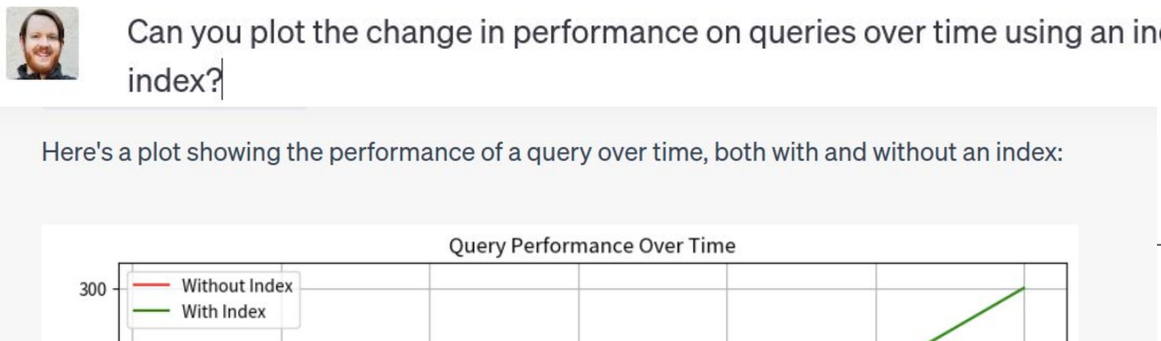
I've created a notebook named "SQL_Index_Suggestions.ipynb". You can follow along and interact with the notebook by clicking here.

Now, let's proceed with the task. First, I'll need to know the database system you're using (e.g., MySQL, PostgreSQL, SQL Server, etc.) and the SQL queries for which you'd like to get index suggestions.

Noteable Chat Plugin

In that first wave of plugins on ChatGPT's launch, Noteable was ranked in the top 5. We saw over the course of a month around **100x** increased traffic on our free multi-tenant site as a result.
Why? Noteable and ChatGPT together work wonders.

Noteable provides a context to chat conversations that houses the documented code, markdown, and interactive data summaries requested by your natural language questions. We enable both code and shareable outcomes natively as a result of one exploration with AI Chat systems like ChatGPT.



Can you plot the change in performance on queries over time using an index vs not using an index?

Here's a plot showing the performance of a query over time, both with and without an index:

Noteable

Noteable Chat Plugin: Scale Challenge

Typically a **100x** scale event would require a traditional SaaS site to undergo a reacrchitecture or major restructuring of one's site.

In the case of Noteable it'd be even more likely, as we are providing dedicated compute containers for every session running code – be it from traditional code development or AI driven conversations.

Within that month of scaling we were able to get our error rates and site latency back to pre-scale levels while providing massively more compute and route requests. Beyond adding a few indexes and some route cleanup, **we didn't have to rebuild any significant part of our app**.

Scaling Response

Day-in-the-life for two months (in order of priority):

- Exercise indecent response playbook
  - Less frequent as stability returned
- Same day hotfix cycles for discovered bottlenecks
- Integration Testing
  - Halted new features, increased test coverage by 2x+
- Load Testing
  - Run next week's expected #s against release candidates
- Manual Testing
  - Being responsive usually doesn't leave time to automate everything

Technology Choices

Chosen ahead of the scaling event enabled our rapid, stable response

- CockroachDB
- Kubernetes
- Datadog
- FastAPI

Noteable

Technology Choice: CockroachDB

From the start we were using CockroachDB to avoid the inevitable series A phase where your data layout typically fails. The ability to apply later stage patterns without rewriting your tables let us skip that failure.

- Multiple Writer bet played out well
  - More nodes added no operational load in application
  - Key use of **time-travel-queries** helps a lot
- Sub-linear increase in cost of operations
- Index and query analysis critical for keeping ahead of slowness
- Experts with Postgres experience able to quickly develop advanced uses of Cockroach
  - **Con:** It's ***almost*** Postgres which can hit the uncanny valley of tech similarity early in use / learning

Noteable

Technology Choice: Datadog

Visibility into your application is better than doubling your engineer count. You need metrics and log gathering and you need a lot of it when things start failing.

- **All in one tool** was helpful
- Most important thing is to get the tool you have for monitoring worked into your **non-technical processes**
  - Technology choice here is less relevant than commitment to the tool you have chosen
- **Con:** Expensive
  - But cheaper than building yourself
    - No really, stop building your own metric and log management systems

Noteable

Technology Choice: Kubernetes

For our needs, we had complex and dynamic container requirements to enable user allocation of compute resources. This required us to have an advanced container management layer, so we used that investment cost to also gain the benefits for the rest of our product.

- Testing scale up and down event in mirrored environments
- Provides **best practices options** for DevOps
- Always have multiple nodes for each service
    - Avoid surprises when you need that 2nd node
        - Spoiler: we broke this rule to get the Plugin out the door, but came back to fix it asap
- **Con:** Operational, expertise, and cost overheard when smaller
    - Many products don't need this level of sophistication so choose wisely if it's worth the opportunity cost to invest

Noteable

Technology Choice: FastAPI

Our users mostly use Python, so we chose Python to be close to our user needs. The next constraint was to make sure our servers were efficient and easy to develop. FastAPI + SQL Alchemy fit the bill for our backend. NextJS for our Frontend… not listed as our winning choice: a conversation for another talk.

- Make use of an async-first API server
  - Let your CPU cycles be spent in the DB not waiting on your DB
  - Future libraries are all going to be Async capable, possible Async only
- **Fast iteration cycle** is important
  - **Con:** Our stack used the latest mature options to achieve goals, engineers weren't as familiar back then
- Do **API first development**
  - SDK against our FastAPI routes developed early and dogfooded for new tools we build for ourselves

Noteable

Learnings

We were able to be flexible to what our techstack needed as real use became apparent.

But there were a lot of learnings that would have eased the pain or sped up our development along the way.

Noteable

Scaling Learnings

- If you're betting on growing fast, know that your underlying tech stack *could* handle it
    - Prepare for problems you have seen at the next 10x size or stage
        - Load testing helps a lot, but is hard to find time to build out
    - Balance with get-it-out-the-door
        - Timebox any future proofing

- The development lifecycle and processes are critical to getting value out of your tech
    - Investment in continually improving your processes
        - E.g. incident response, hotfix rollouts, deep dive analysis
        - Often worth more than investment in better tech

Scaling Learnings Pt2

- Plan for a few potential growth paths and make 1-way door decisions for those
    - Ensure you can be flexible to one particular path dominating
    - Be ok with dropping efforts on paths that aren't important or not **winning harder**

- Your database is where you bottleneck should eventually land in a SaaS scaling event
    - Rest of the tech choices are more lenient on scalability
        - E.g. stateless routers, caching, isolated actions can be autoscaled
    - Make the most efficient use of your storage layer
        - CockroachDB eliminates some of the hard-to-implement requirements here
            - E.g. local region queries, multi-writer

Noteable @ RoachFest 2023

Thanks for Listening:
Try out app.noteable.io