

**STREAMING STOCK DATA ANALYSIS**

**B.Tech PADA Project Report**

**SUBMITTED BY**

**Mehul Dharmraj Patil [B194013]**

**Niranjan Vinod Patil [B194024]**

**Sanket Sunilkumar Sathe [B194036]**

**Prajwal Warad [B194092]**

**GUIDED BY**

**Prof. Seema Mandlik**

**SCHOOL OF COMPUTER ENGINEERING & TECHNOLOGY**

**MIT ACADEMY OF ENGINEERING, ALANDI (D), PUNE-412105**

**MAHARASHTRA (INDIA)**

**MAY, 2020**



**CERTIFICATE**

It is hereby certified that the work which is being presented in the B.Tech. PADA Project Report entitled **“*Streaming Stock Data Analysis*”,** in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Computer Engineering & Technology** and submitted to the **School of Computer Engineering & Technology of MIT Academy of Engineering, Alandi(D), Pune, Affiliated to Savitribai Phule Pune University (SPPU), Pune** is an authentic record of work carried out during an Academic Year 2019-2020, under the supervision of **Prof. Seema Mandlik,** **School of Computer Engineering** **& Technology**.

**Mehul Dharmraj Patil PRN No.** **0120160070 Exam Seat No. B194013**

**Niranjan Vinod Patil PRN No.** **0120160134 Exam Seat No. B194024**

**Sanket Sunilkumar Sathe PRN No.** **0120160202 Exam Seat No. B194036**

**Prajwal Warad PRN No.0120160478 Exam Seat No. B194092**

**Date: 12/05/2020.**

|  |  |
| --- | --- |
| *Signature of Project Advisor* | *Signature of Dean* |
| **Project Adviser** | **Dean** |
| School of Computer Engineering & Technology | School of Computer Engineering & Technology |
| MIT Academy of Engineering, Alandi(D), Pune | MIT Academy of Engineering, Alandi(D), Pune |

**(STAMP/SEAL)**

|  |  |
| --- | --- |
| *Signature of Internal examiner/s* | *Signature of External examiner/s* |
| *Name………………………………* | *Name………………………………* |
| *Affiliation…………………………* | *Affiliation…………………………* |

**ABSTRACT**

Financial and stock market data is continuously increasing day by day. When analyzing stocks, data comes from various sources at different intervals of time. These sources need to be streamed in, joined together efficiently and then calculations and advanced analytics need to be applied. Sometimes these sources need to be updated based on corrections made to the data. Traditionally, this is all very hard to do. Now with Delta Lake and Structured Streaming, this is all much easier. This project shows how you can analyze streams of economic data using Delta Lake and Structured Streaming.

**TABLE OF CONTENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Abstract | | | |  |
|  | Introduction | | |  |
|  | Related Work | | |  |
|  | Project Description | | |  |
|  | a. | Software and Hardware Requirement | |  |
|  | b. | Architecture Diagram | |  |
|  | c. | Dataset Analysis | |  |
|  | Implementation | | |  |
|  | a. | | Source Code |  |
|  | b. | | Result Analysis |  |
| Conclusion | | | |  |
| References | | | |  |

**Chapter 1 – INTRODUCTION**

In the last decade, there is a rise of stock data. It is a hectic, tedious, challenging and time consuming task for workers in financial services or stockholders to read all stock data which in turn results in inappropriate judgement/opinion about stocks market. So analysis of stock data is very important part in financial services.

Traditionally, real time analysis of stock data was a complicated endeavor due to the complexities of maintaining a streaming system and ensuring transactional consistency of legacy and streaming data concurrently.

Hence, this is our purpose as well as our motivation to develop or building a streaming system to analyze stock data in real time which would help the workers in financial services or stockholders to make better decision about invest in stock market or analyze stock price in graphical way.

**Objective:**

* To stream data.
* To analyze stock market data in graphical form.

**Problem Statement:**

Time Consuming task for stockholder to read the entire stock market data which in turn results in appropriate analysis about stock price so our system helps to stockholders to analyze stock market data in short amount of time.

1. **Intended Audience and Reading Suggestions**
2. Novice workers in financial services, such as a person who wants to know how well a analyze real time stock data without going through each and every stock dataset.
3. Researchers who would like to know about the stock analysis.
4. Tester who would like to know about the scope of stock analysis and must know about assumption.
5. Marketing staff would like to know about stock analysis special features.
6. Developers who would like to take the reference of the stock analysis.

**Chapter 2 – Related Work**

In our project we use streaming data. Simply streaming receives live input data stream and divide the data into batches which are then processed by spark engine to generate final stream of results in batches. The pain points of a traditional streaming and data warehousing solution can be broken into two groups: data lake and data warehouse pains.

**Data Lake Pain Points**

While data lakes allow you to flexibly store an immense amount of data in a file system, there are many pain points including (but not limited to):

* Consolidation of streaming data from many disparate systems is difficult.
* Updating data in a Data Lake is nearly impossible and much of the streaming data needs to be updated as changes are made. This is especially important in scenarios involving financial reconciliation and subsequent adjustments.
* Query speeds for a data lake are typically very slow.
* Optimizing storage and file sizes is very difficult and often require complicated logic.

**Data Warehouse Pain Points**

The power of a data warehouse is that you have a persistent performant store of your data. But the pain points for building modern continuous applications include (but not limited to):

* Constrained to SQL queries; i.e. no machine learning or advanced analytics.
* Accessing streaming data and stored data together is very difficult if at all possible.
* Data warehouses do not scale very well.
* Tying compute and storage together makes using a warehouse very expensive.

**How Databricks Delta Solves These Issues**

Databricks Delta is a unified data management system that brings data reliability and performance optimizations to cloud data lakes. More succinctly, Databricks Delta takes the advantages of data lakes and data warehouses together with Apache Spark to allow you to do incredible things! Databricks Delta Lake helps solve problems regarding real time analysis of stock data by combining the scalability, streaming, and access to advanced analytics of Apache Spark with the performance and ACID compliance of a data warehouse.

* Databricks Delta, along with Structured Streaming, makes it possible to analyze streaming and historical data together at data warehouse speeds.
* Using Databricks Delta tables as sources and destinations of streaming big data make it easy to consolidate disparate data sources.
* Upserts are supported on Databricks Delta tables.
* Your streaming/data lake/warehousing solution has ACID compliance.
* Easily include machine learning scoring and advanced analytics into ETL and queries.
* Decouples compute and storage for a completely scalable solution.

**Chapter 3 – Project Description**

1. **Software and Hardware Requirements**

Requirements are the specifications and the tools necessary for building an application. The hardware and software required for making a Para-military employee management system are listed below:

**SOFTWARE REQUIREMENTS**

* 1. Operating System : Windows 98, Windows XP, Windows 7, Linux, Mac
  2. Browser: Google Chrome, any of Mozilla, Opera etc.
  3. Database : SQL Database
  4. Programming Language: Spark.
  5. Platform: Databricks Unified data analytics platform.
  6. Unified analytics Engine: Apache spark for large scale data processing.

**HARDWARE REQUIREMENTS**

* 1. Processor : Intel Core i3 and above
  2. RAM : 1GB and above
  3. Hard Disk : 10GB and above

1. **Architecture diagram**

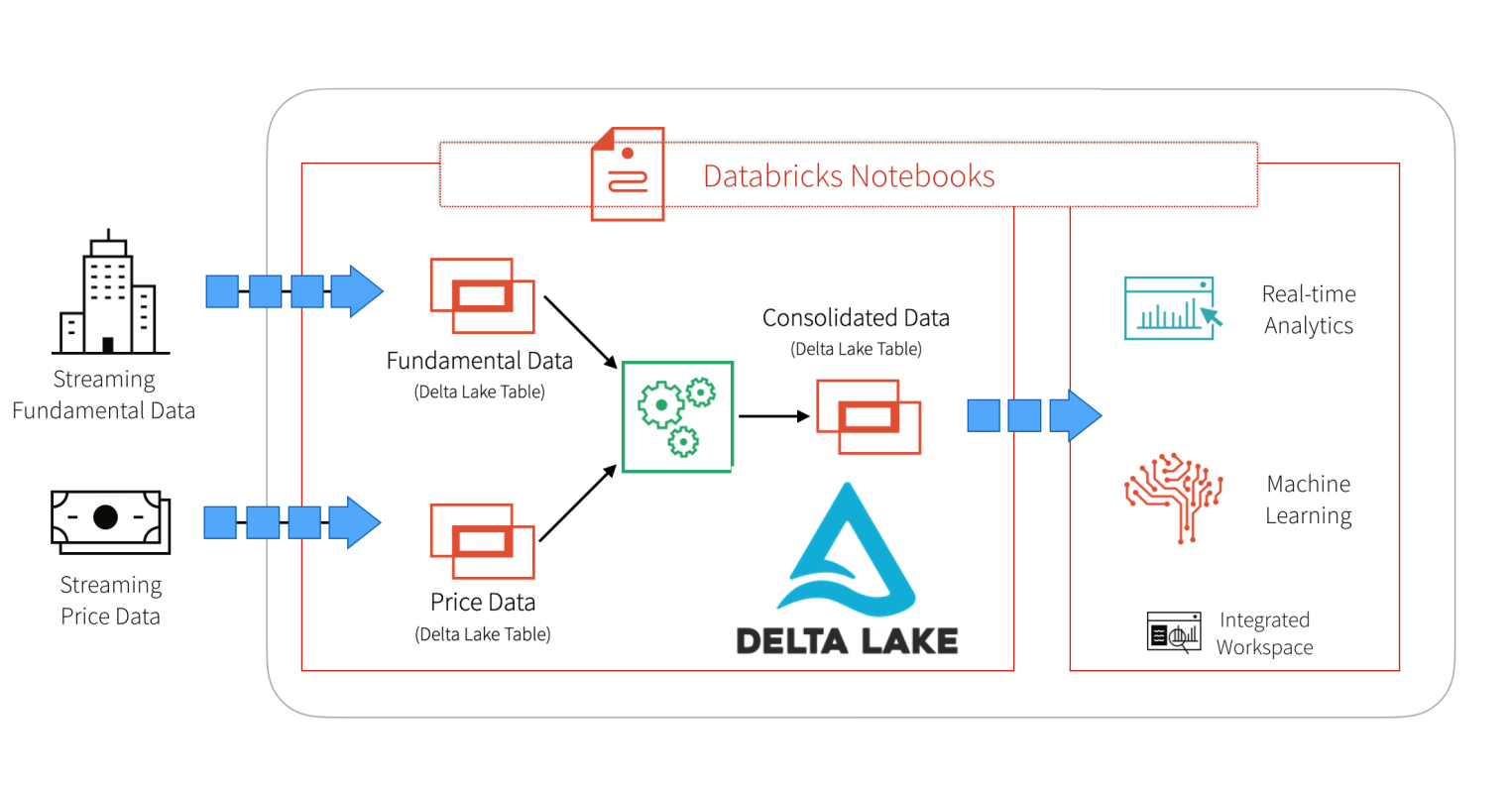


Fig 1. Streaming Stock Data Analysis System Architecture Diagram

In the above diagram, we provide a high-level architecture to simplify problem regarding real time analysis of stock data. We start by ingesting two different sets of data into two Databricks Delta tables. The two datasets are stocks prices and fundamentals. After ingesting the data into their respective tables, we then join the data in an ETL process and write the data out into a third Databricks Delta table for downstream analysis.

1. **Dataset Analysis**

**Stock**: Stock is an investment. When you purchase a company stock, you’re purchasing a small piece of that company, called a share. Investors purchase stocks in companies they think will go up in value. If that happens, the company stock increases in value as well. The stock can then be sold for profit.

When you own stock in a company, you are called shareholder because you share in the company’s profit. Public company sells their stock through a stock market exchange like the Nasdaq or the New york stock exchange

We have two datasets related to stock market for process – one for fundamentals and one for price data which has period from 2011 - 2016. Data source is quandle.com

**1. Fundamental Dataset:**

Fundamental Dataset has 4 columns like ticker id (stock id), eps\_basic\_net, start\_date and end\_date.

**Ticker:** A ticker is a symbol, a unique combination of letters and numbers that represent a particular stock or security listed on an exchange. The symbol is used to refer to a specific stock, particularly during trading. Trades are executed based on a company’s ticker symbol, which is recorded in the exchange’s trading system. The system will find the stock easily when the ticker symbol is entered.

In this dataset 4 ticker are available like DL2, JO1, JN2 and NA2.

**EPS\_basic\_net:**

Earnings per share (EPS) are a company's net profit divided by the number of common shares it has outstanding. EPS indicates how much money a company makes for each share of its stock and is a widely used metric for corporate profits. A higher EPS indicates more value because investors will pay more for a company with higher profits. In dataset this columns represents EPS during earning season for particular ticker.

**Earning season:**

Earnings season refers to the months of the year during which most quarterly corporate earnings are released to the public. Earnings season generally occurs in the month immediately following the end of each fiscal quarter. This means that earnings seasons typically fall in March, June, September and December. In dataset 2 columns like start and end date this are earning season.

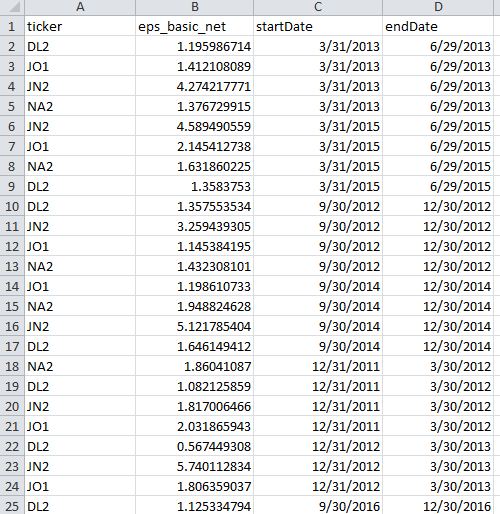
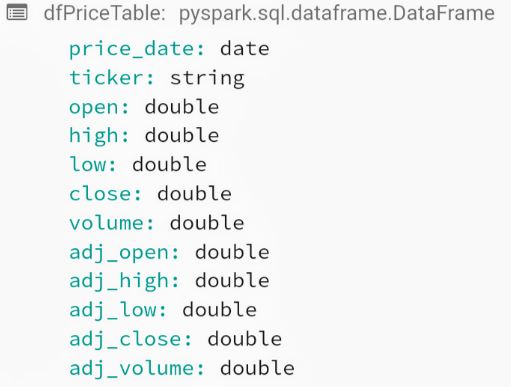
****

Fig 2. Fundamental Dataset

**2. Price Dataset:**

Price Dataset has 12 columns like price\_date, ticker id (stock id), open price, day high price, day low price, close price, volume, adjusted open, adjusted high, adjusted low, adjusted close and adjusted volume.

The price\_date column is for date on which all stock price value of ticker is measured.

The open price is the 'raw' stock price which is just the cash value of the first transacted price before the market closes. The adjusted open price factors in anything that might affect the stock price after the market closes.

The high price is the 'raw' stock price which is just the cash value of the highest transacted price before the market closes. The adjusted high price factors in anything that might affect the stock price after the market closes.

The low price is the 'raw' stock price which is just the cash value of the lowest transacted price before the market closes. The adjusted low price factors in anything that might affect the stock price after the market closes.

The closing price is the 'raw' stock price which is just the cash value of the last transacted price before the market closes. The adjusted closing price factors in anything that might affect the stock price after the market closes.

Adjusted open and adjusted close are the driving factors for calculating return as these includes any corporate actions like stock split, stock merge etc. taken on the stock. (Source: Investopedia.com)

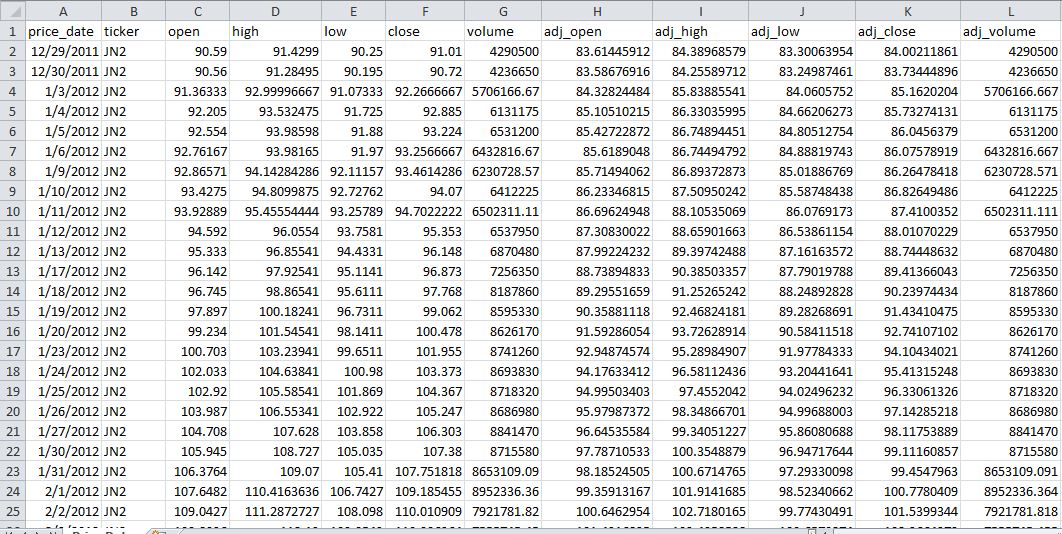


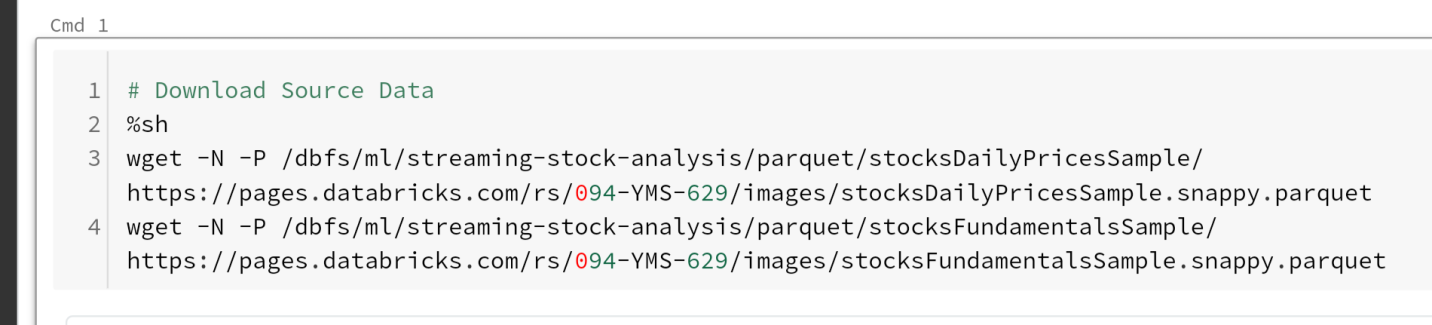
Fig 3. Price Dataset

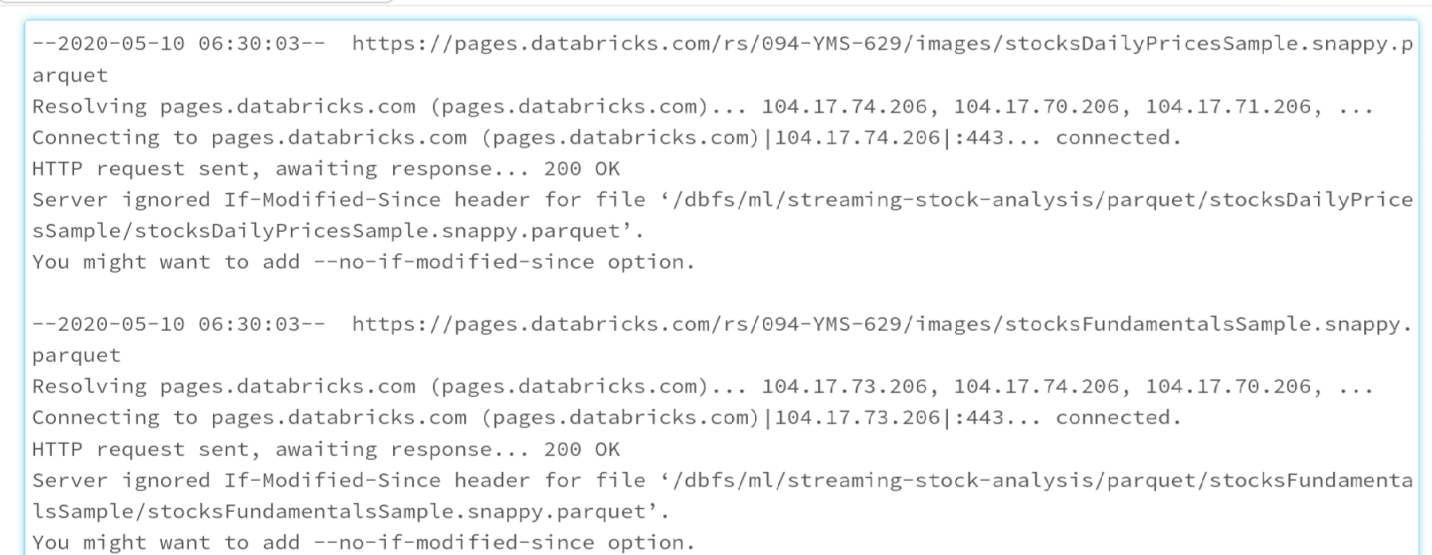
**Chapter 4 – Implementation**

1. **Source Code**

First run this notebook so it can automatically download the generated source data and starts loading data into a file location. This notebook is use for to generate the streaming data that will be showcased in the main notebook.

**Download the Source Data**





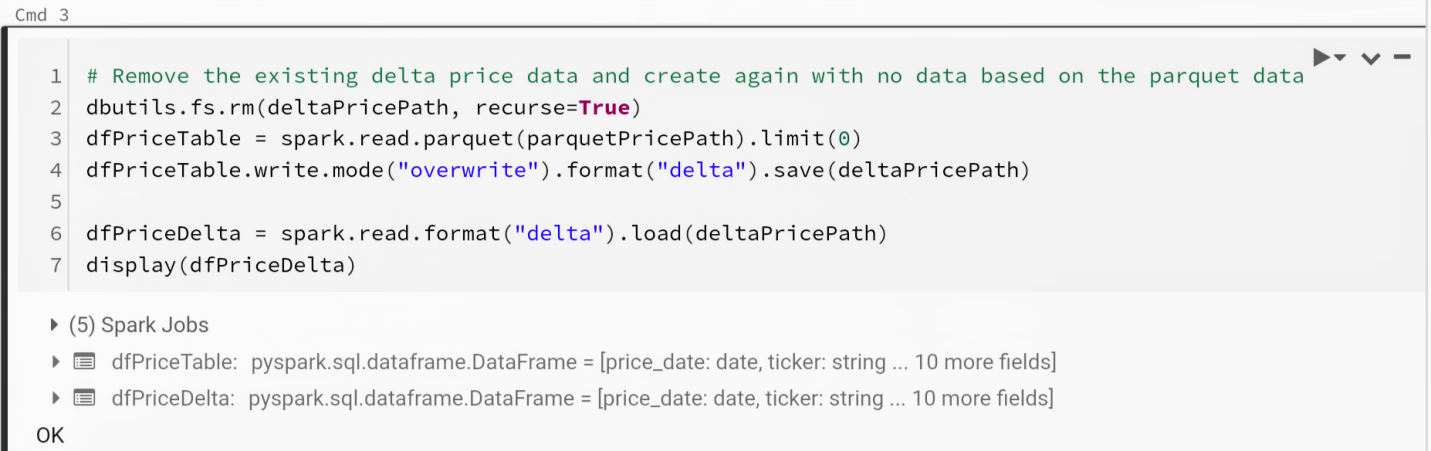
**Configure Locations**

Set the source Parquet Data (open source file format for Hadoop like row-column based and more efficient in terms of storage and performance) and generated Delta Lake Table file paths.

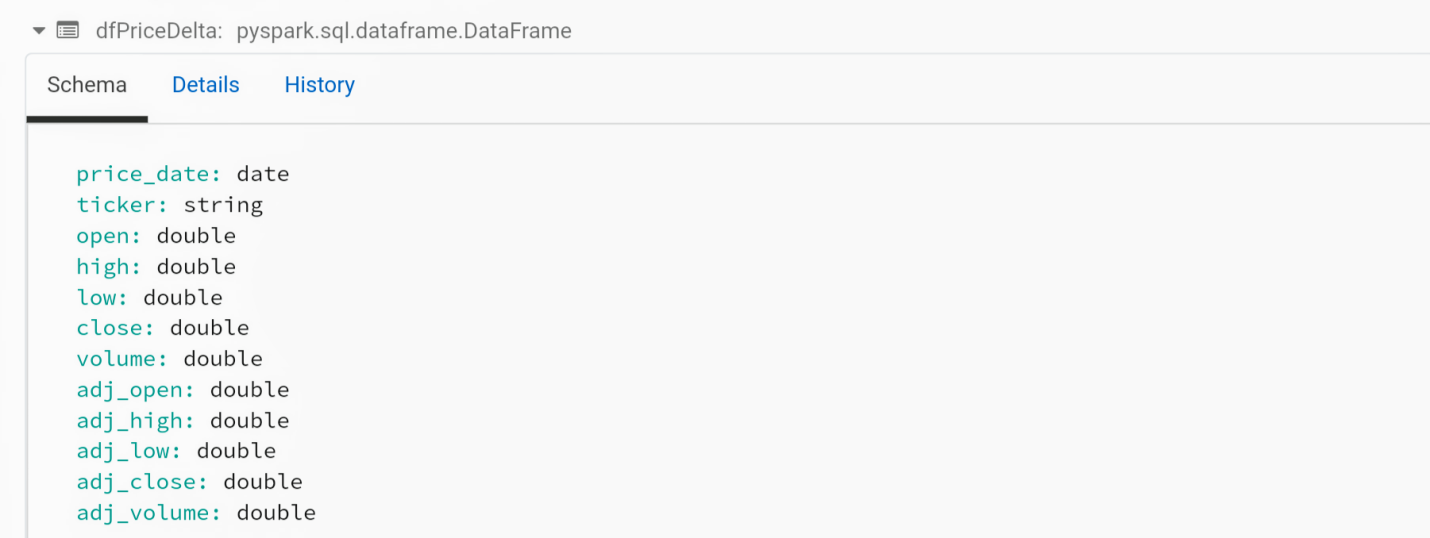


**Build the Initial Delta Lake Tables**

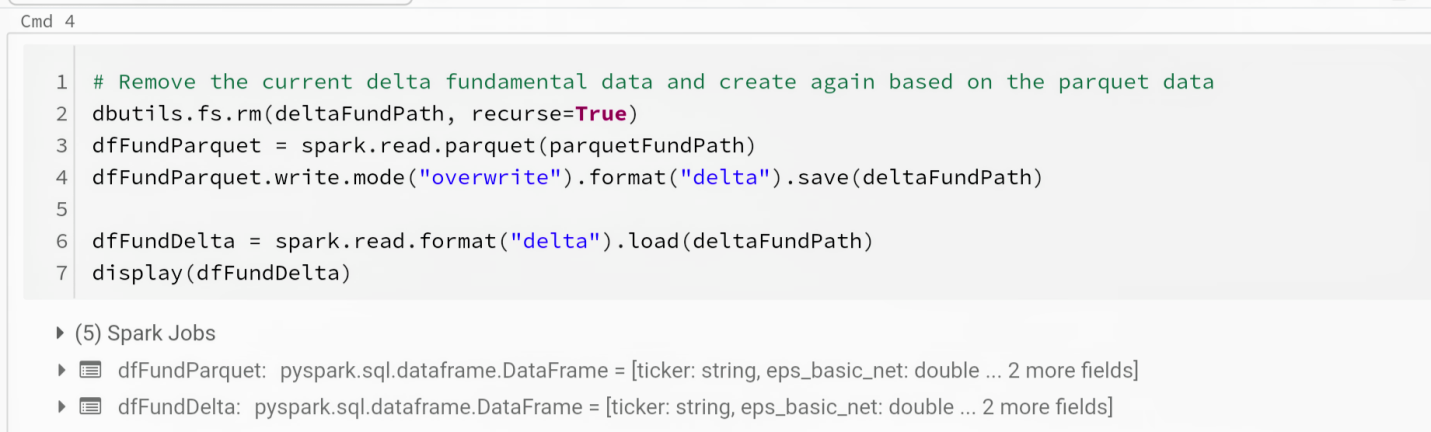
First, let's create the Fund Delta Lake table using the dfPriceDelta DataFrame and then we create the Fundamentals Delta Lake table using the dfFundDelta DataFrame.

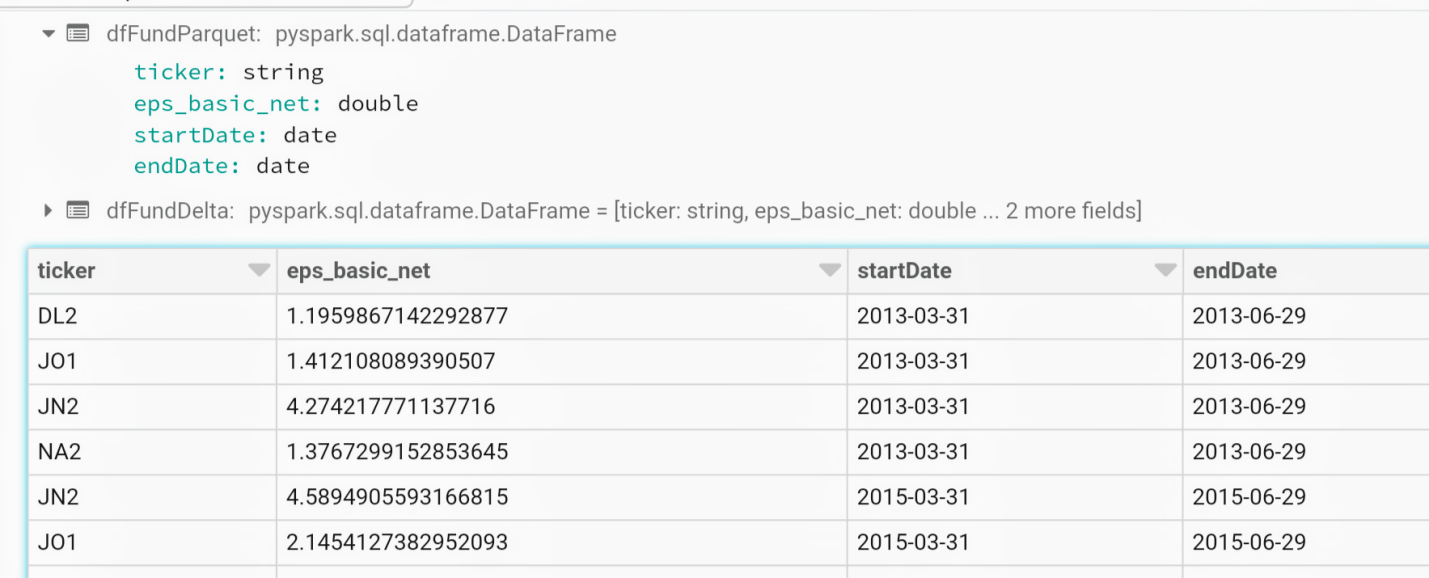


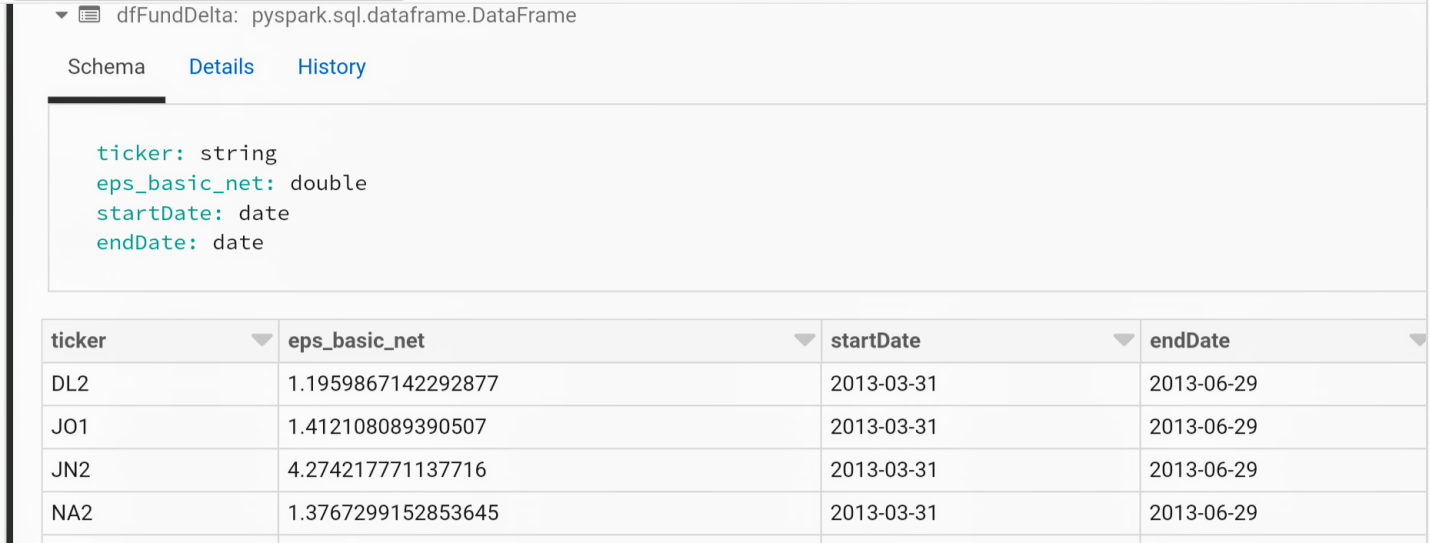




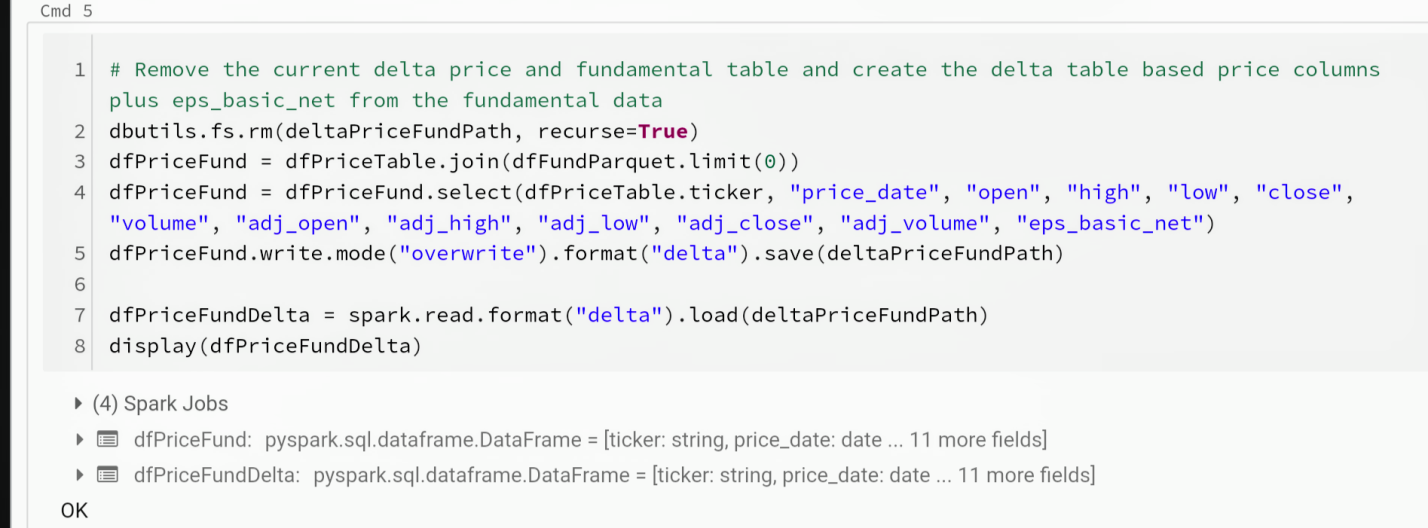
Note: the dfPriceDelta Delta Lake table is empty (though with the correct schema) due to the .limit(0) statement.

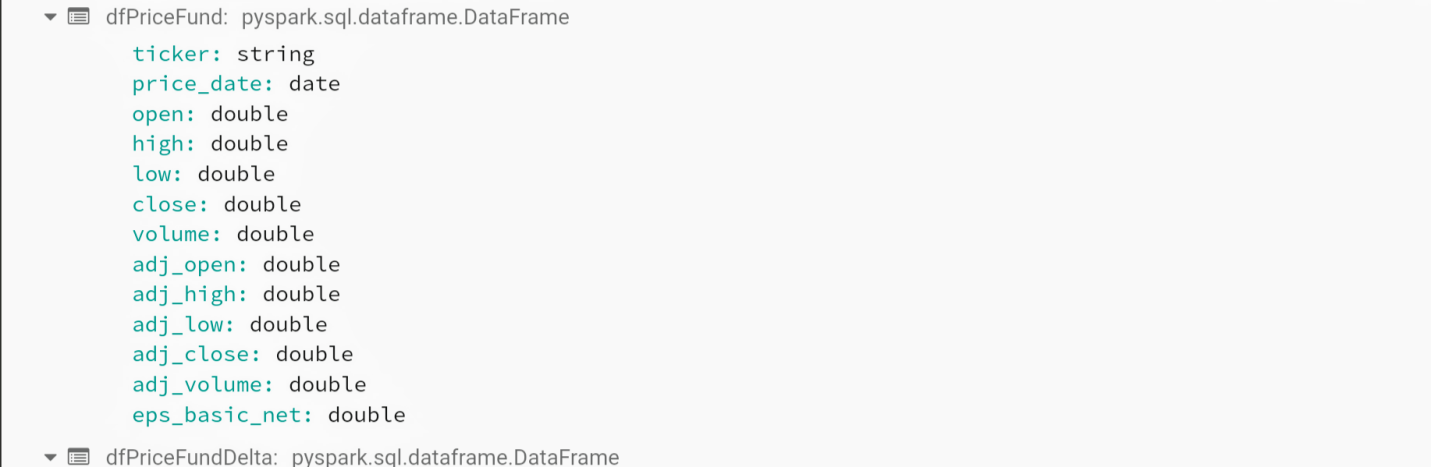


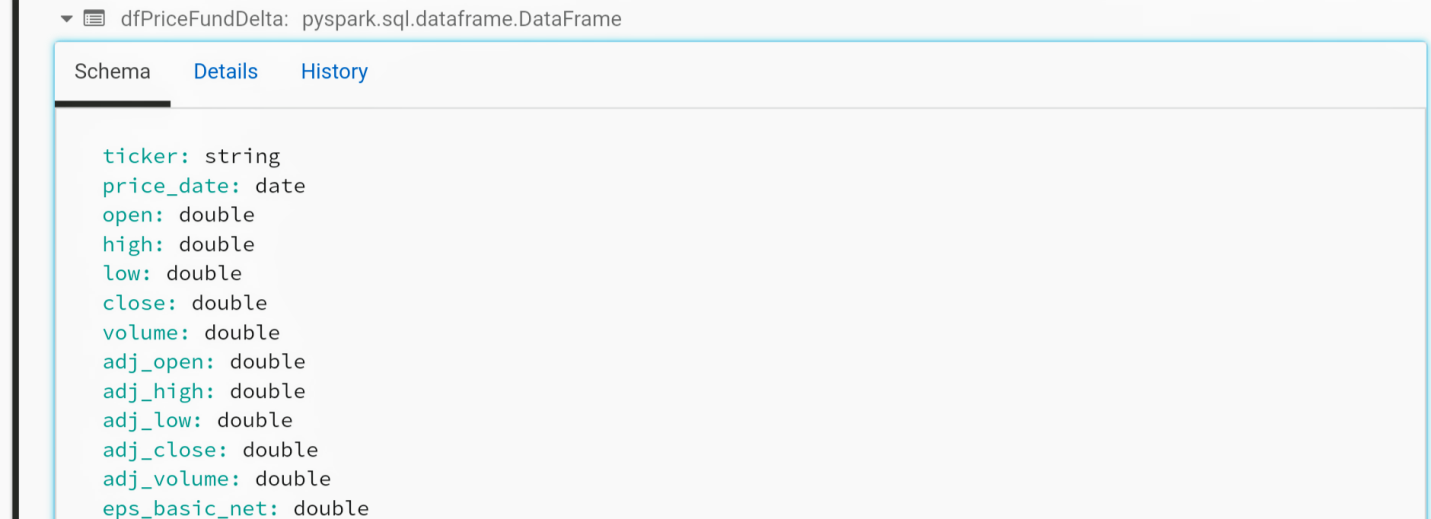




We will join the price and fundamentals data into a PriceFund Delta Lake table. The following code clears out the table/files if they exist and generates a new empty table.





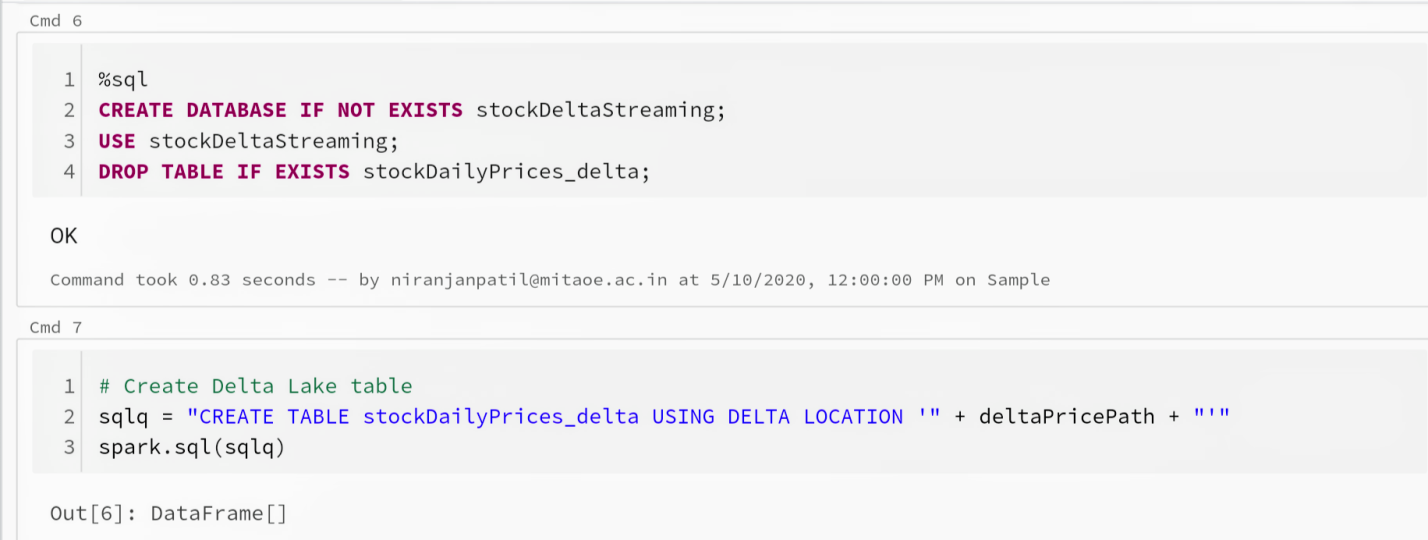


**Create Temporary View for our Streaming Price Data**

We will first create the database stockDeltaStreamng if it does not already exist

Then, we will drop the table stockDailyPrices\_delta if it exists

Finally, we will create the table stockDailyPrices\_delta pointing to deltaPricePath



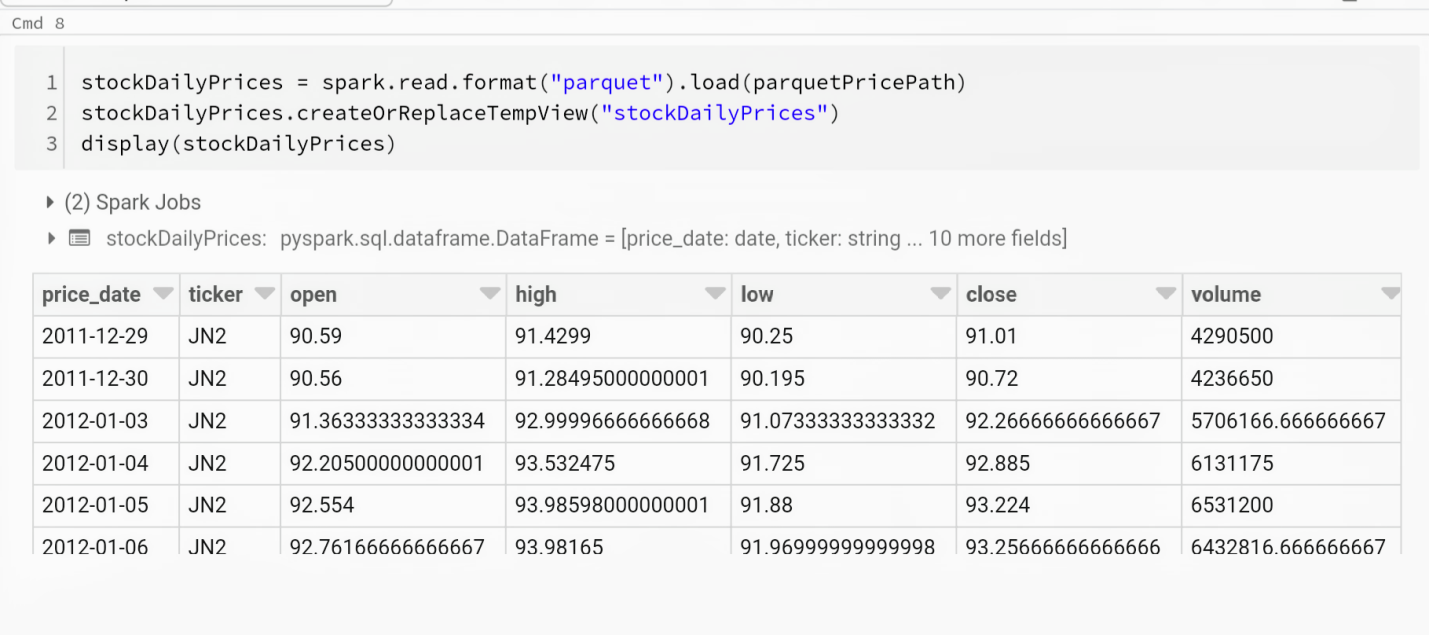
**Stream the Pricing Data based on Price Date**

Let's stream the pricing data based on price date by:

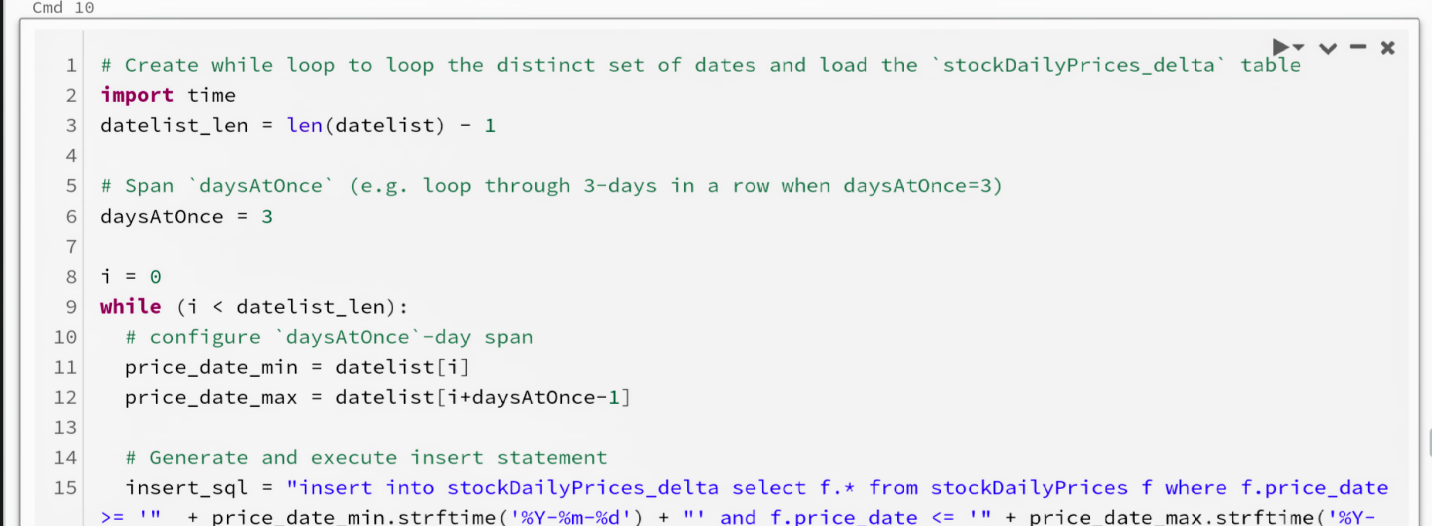
Creating the stockDailyPrices DataFrame based on the source Parquet data

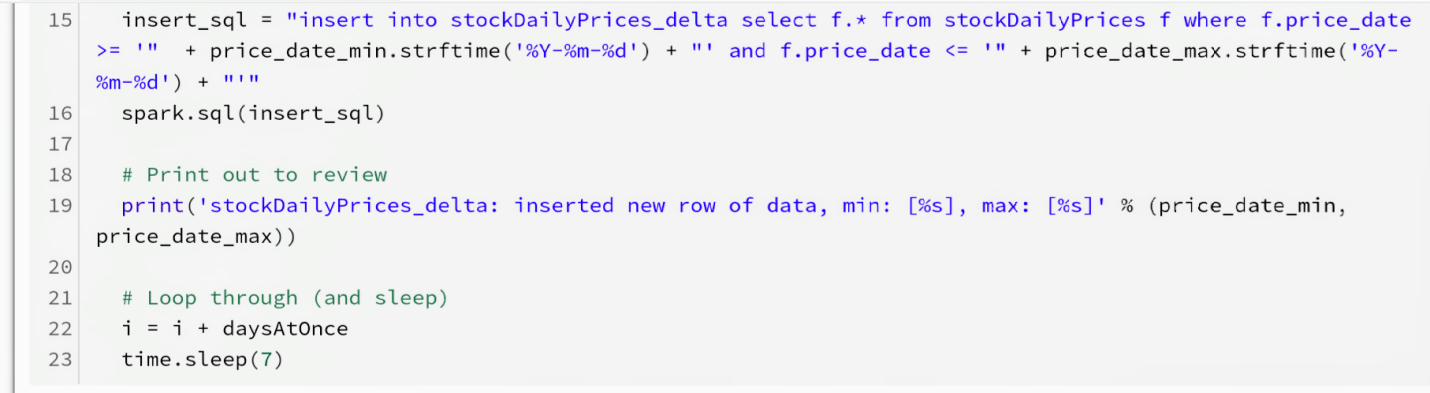
Create datelist that contains the distinct price\_date values

Load the stockDailyPrices\_delta table by loading x number of days at a time (daysAtOnce)







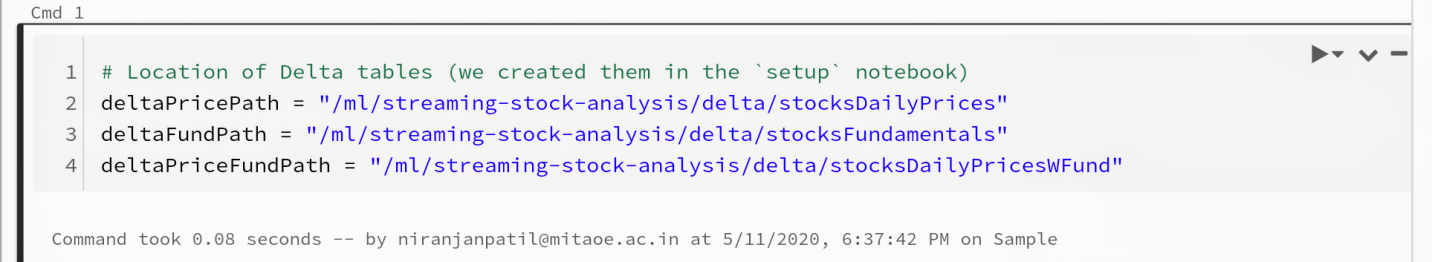


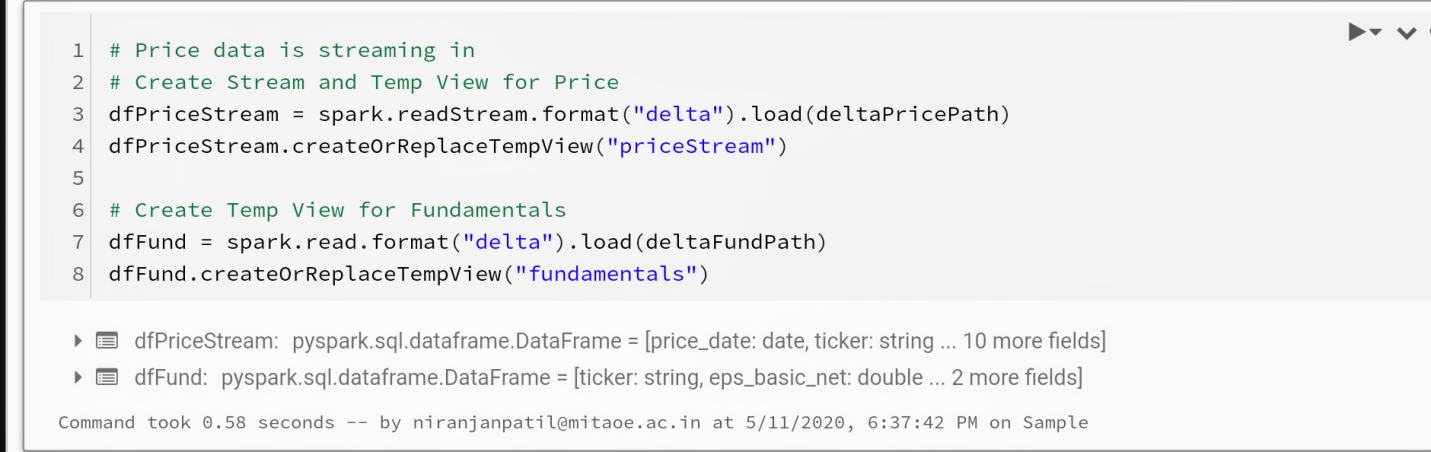


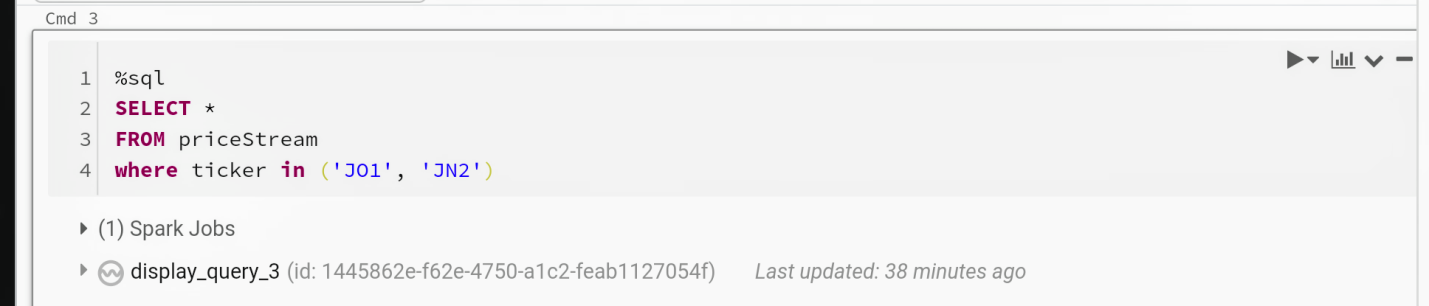
When the last cell of this notebook starts inserting rows into the table stockDailyPrices\_delta, you can start the main notebook.

**Streaming Stock Analysis with Delta Lake (Main Notebook)–**

This main notebook that showcases Delta Lake within the context of streaming stock analysis including unified streaming, batch sync and time travel.







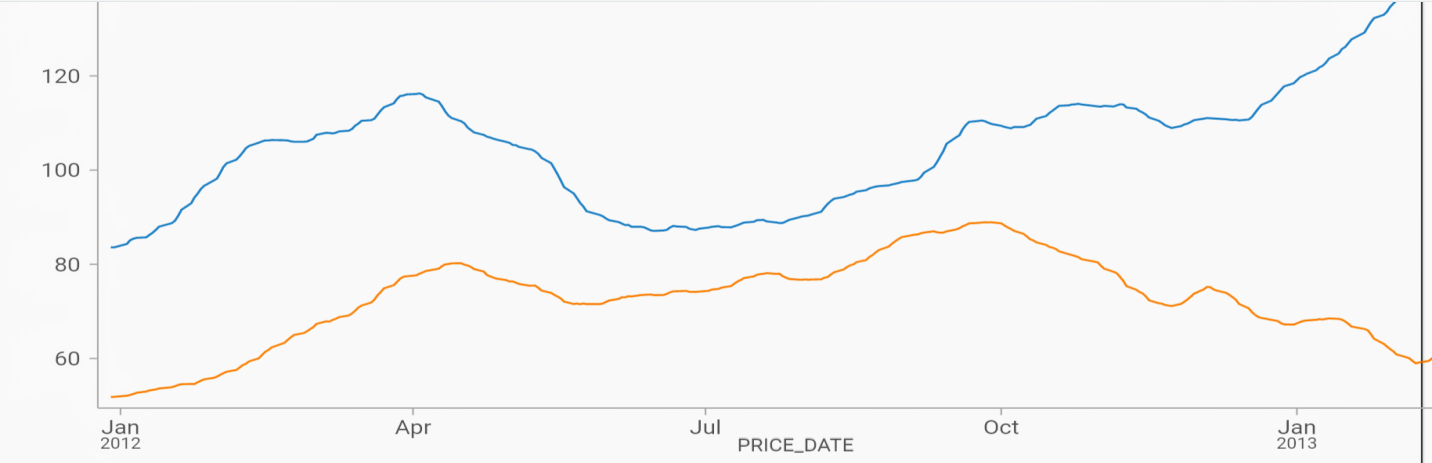


Fig 4. Result 1

Let's review fundamental table



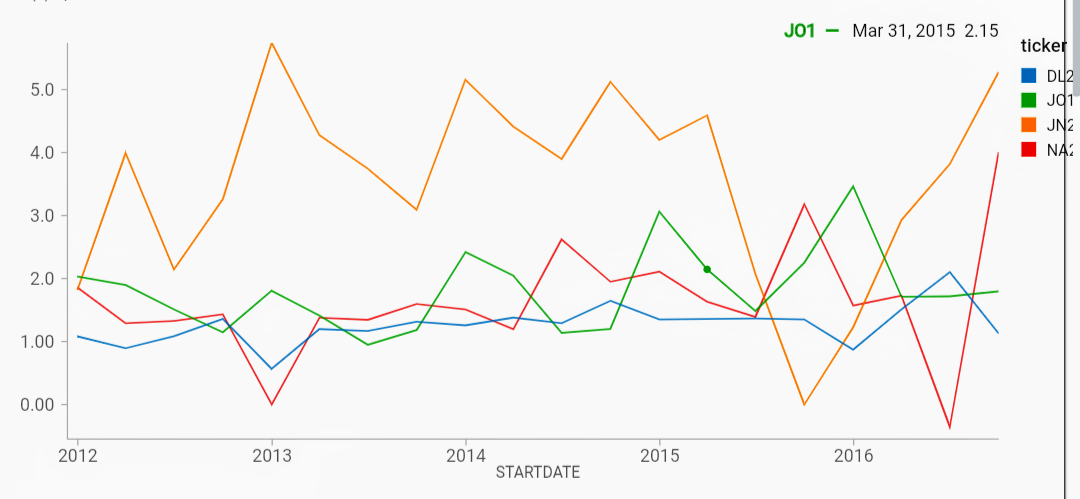


Fig 5. Result 2

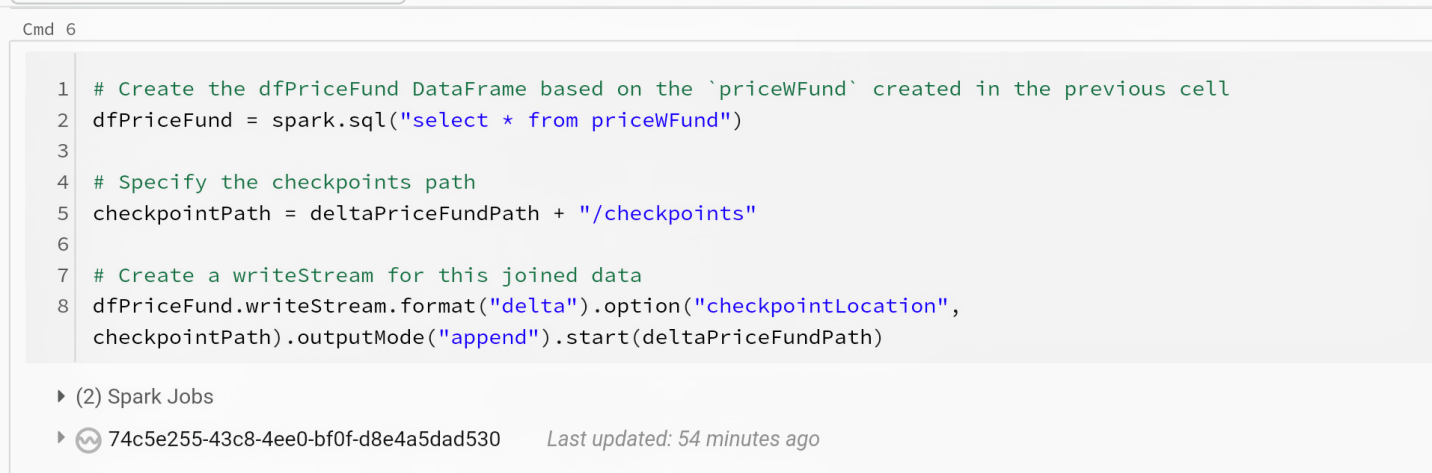
Create a Range Join between the Price and Fundamental tables. Using a Range Join Hint to optimize the join.

* Note the join is happening between the priceStream streaming table and the fundamentals static table
* We can do this with two streams as well but for the purpose of this demo, we will only stream the price data

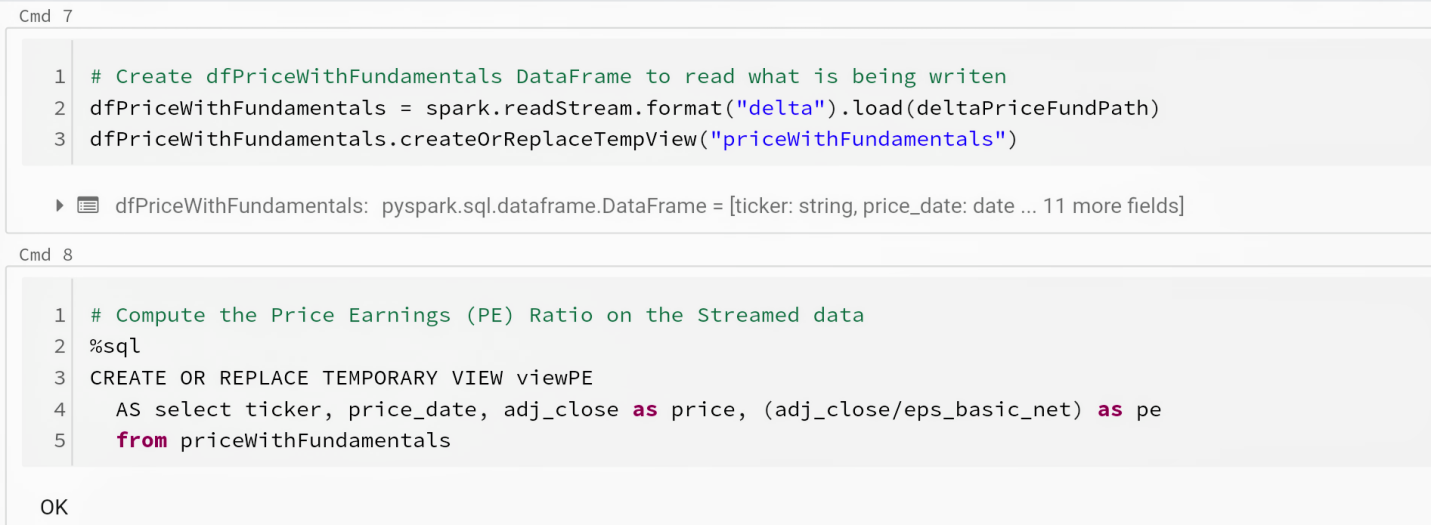
****

Stream the Joined Price and Fundamentals data into a new Delta Lake Table

* First we will create the dfPriceFund DataFrame which is DataFrame API abstraction over the priceWFund temporary view
* Then we will specify the checkpoints Path
* Finally, we will write this join of streaming (price) and batch (fundamentals) data into a new Delta Lake table (via dfPriceFund)

****

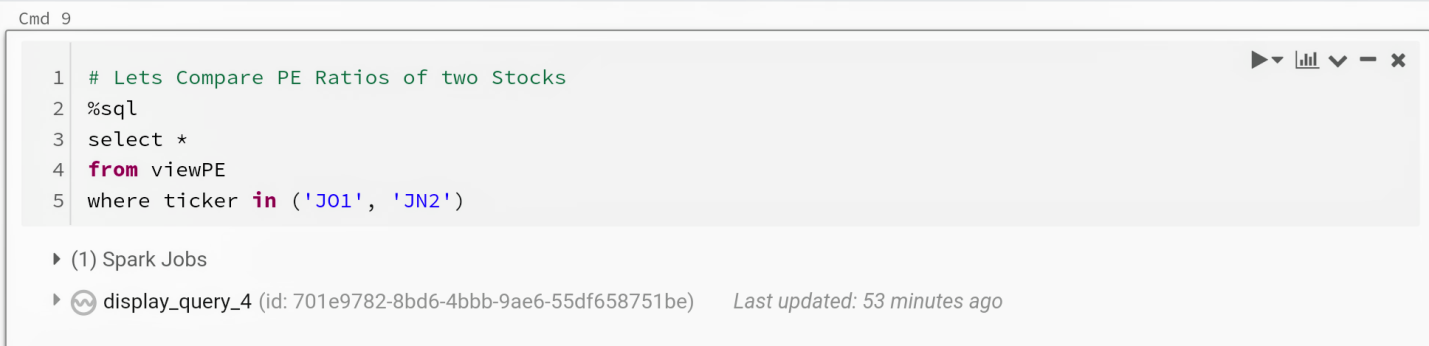
Compute the Price Earnings (PE) Ratio on the Streamed data

****

Price-earning ratio, also known as P/E ratio = Shared Price **/** Earning per share.

The ratio is used for valuing companies and to find out whether they are overvalued and undervalued. High P/E ratio generally indicates increased demand because investor or stockholder anticipates earnings growth in the future.

Lets Compare PE Ratios of two Stocks

****

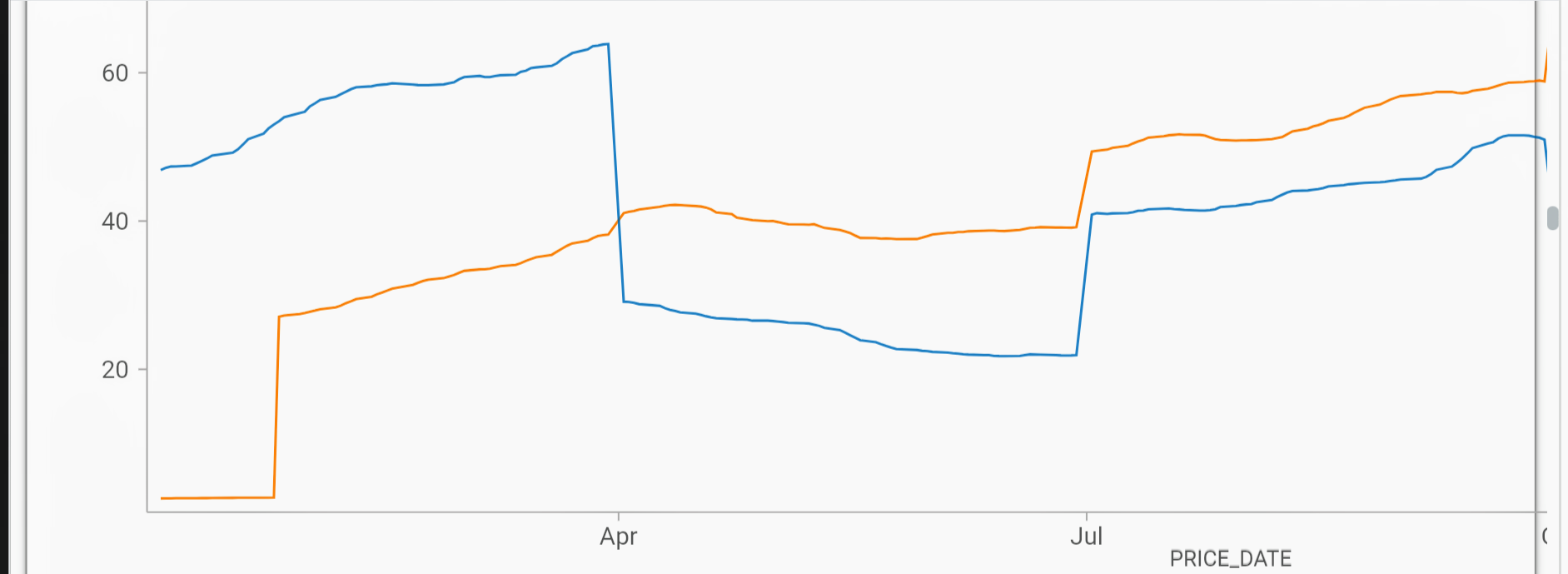
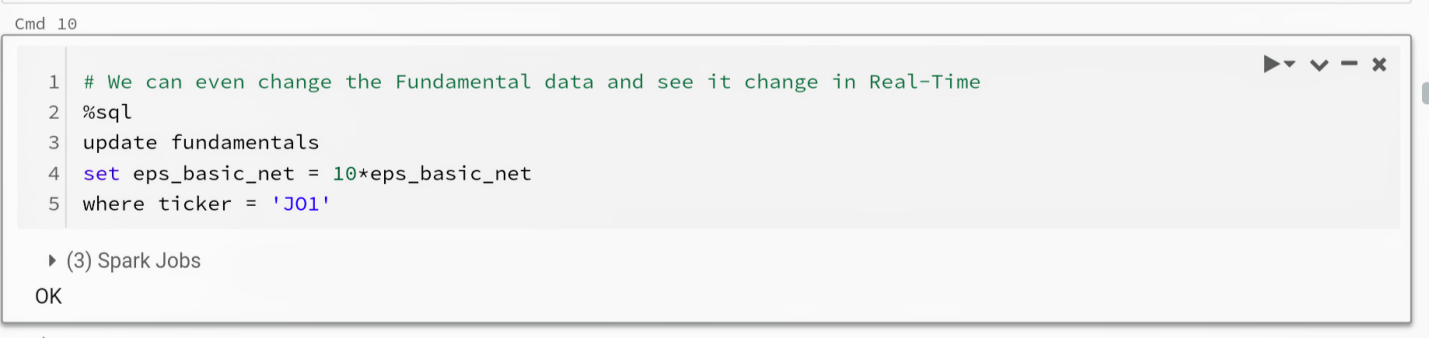
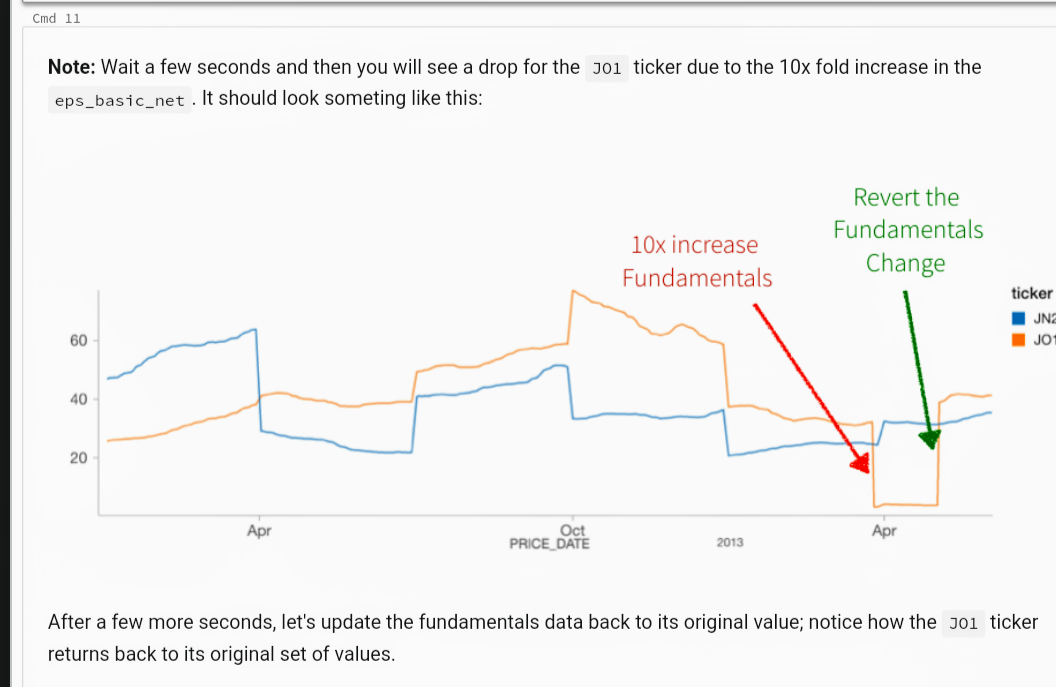
****

Fig 6. Result 3

We can even change the Fundamental data and see it change in Real-Time

* Wait a few seconds and then run the following statement
* Note that we will be changing the fundmentals data by 10x

****

****

****

Let's continue our analysis of our price fundamentals data by calculating the Average Price by Ticker Symbol over the entire data set.

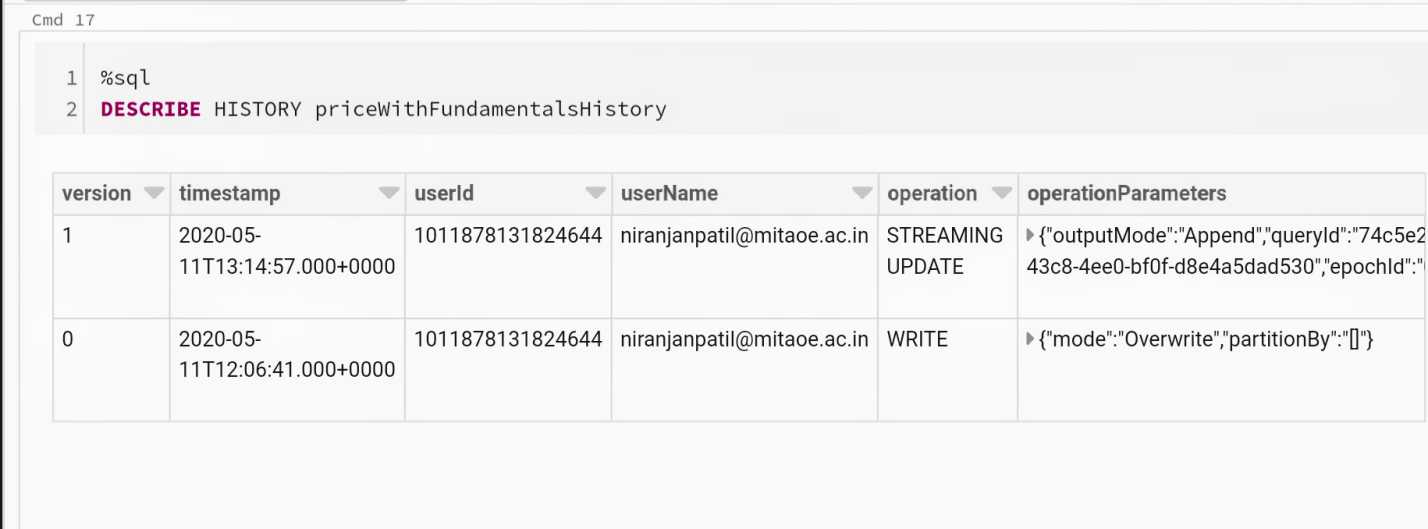
****

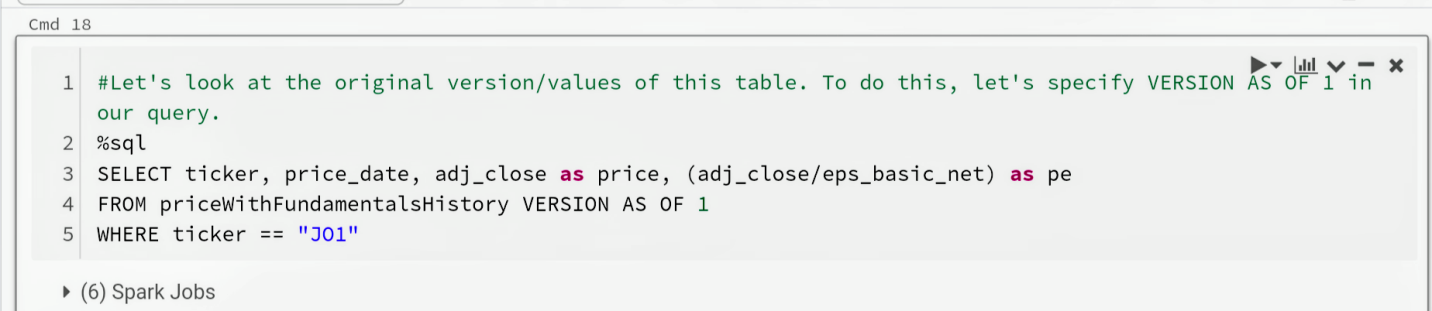
Fig 7. Result 4

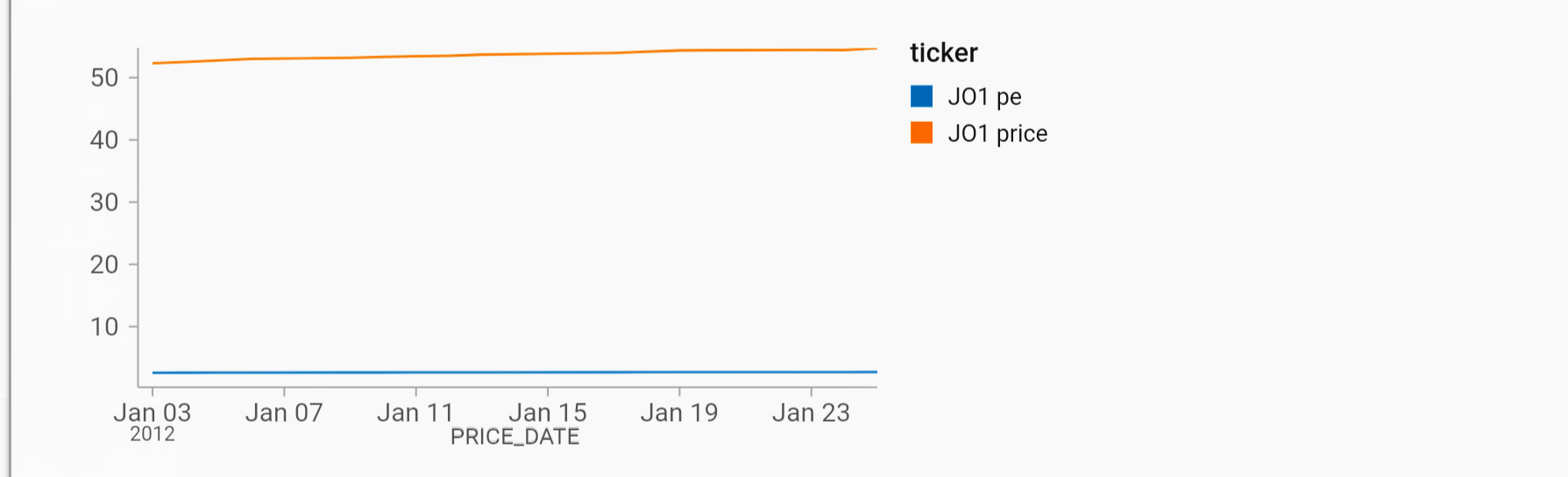
****

Fig 8. Result 5

****

****

****

****

****

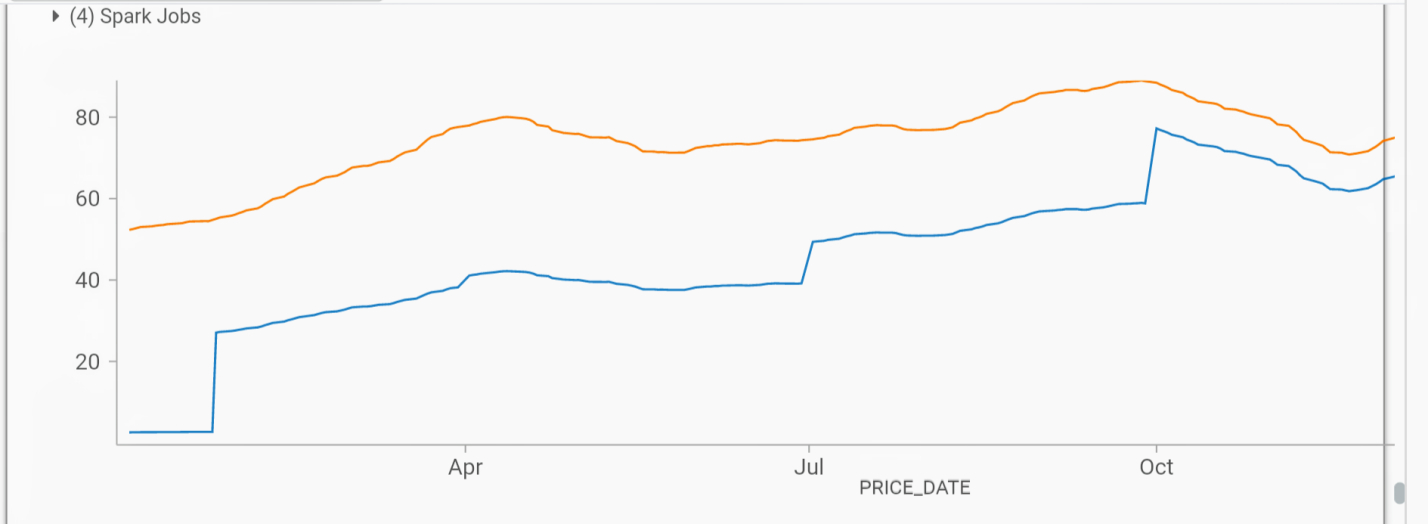
****

Fig 9. Result 6

1. **Result Analysis.**

We have done streaming of stock data so we analyze batch wise stock data effectively through graphical way like we can see line graph of price of different stock tickers and compare their price for particular period. Second graph of fundamental dataset which shows higher EPS indicates more value because investors will pay more for a company with higher profits. JN2 stock ticker has highest EPS. By comparing P/E ratio of two stocks we will find stock ticker which has highest demand and that stock ticker will be provide earning profit to stockholder in the future. We also find and compare average price of different stock ticker. JN2 stock ticker has highest average price.

**Conclusion**

We demonstrated how to simplify streaming stock data analysis using Databricks Delta. By combining Spark Structured Streaming and Databricks Delta, we can use the Databricks integrated workspace to create a performant, scalable solution that has the advantages of both data lakes and data warehouses. The Databricks Unified Analytics Platform removes the data engineering complexities commonly associated with streaming and transactional consistency enabling data engineering and data science teams to focus on understanding the trends in their stock data.

**References**

1. https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/1267108248902618/4226977863287266/1011878131824644/latest.html
2. https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/1267108248902618/4226977863287285/1011878131824644/latest.html
3. https://spark.apache.org/docs/latest/streaming-programming-guide.html
4. https://www.investopedia.com/terms/e/eps.asp
5. https://www.quandl.com/