Analyzing Stock Market Values

## LE Thi Hoa - 12310380 TRAN Hai Linh - 12310487

November 2023

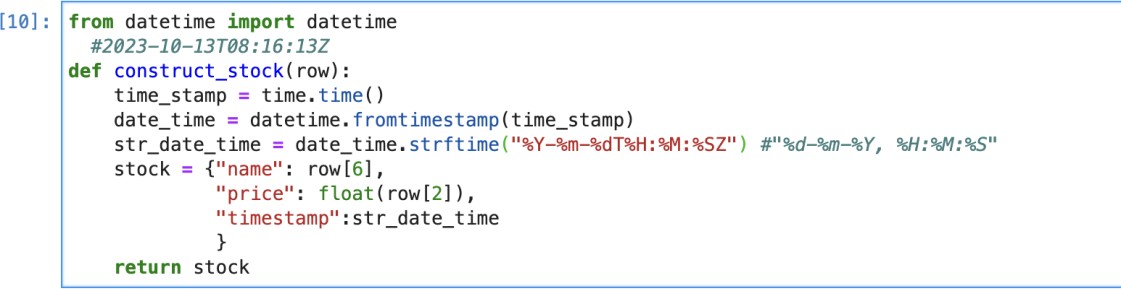
**Link code**: <https://github.com/tranhailinh97/SparkStreamingStocks/tree/master>

# Introduction

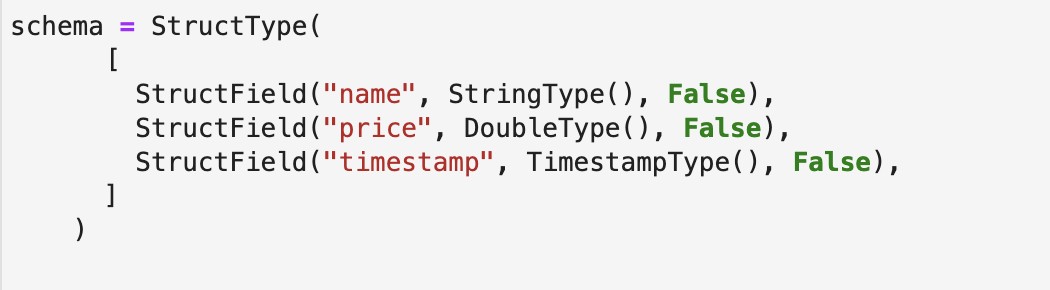
This project aims to analyze streaming data related to the stock market. We have data for stocks with 619,039 rows. The project uses Spark Streaming to analyze the data and Kafka to read the data.

# Pre-Processing

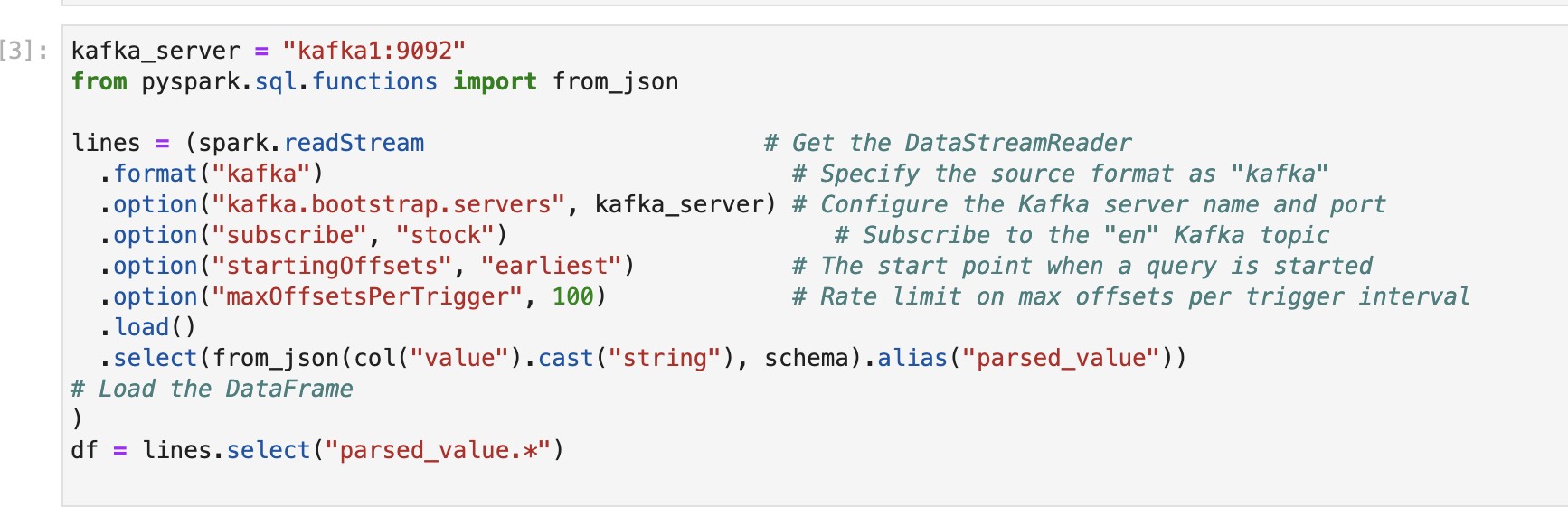
We use Kafka to read the file and ingest the data into Kafka with the schema (name, price, timestamp). The timestamp uses the current one. This below function to process timestamp:



Initialize a SparkSession to work with Spark Streaming and create a schema for the data.



Then, use Spark to read data from Kafka with the topic "stock."



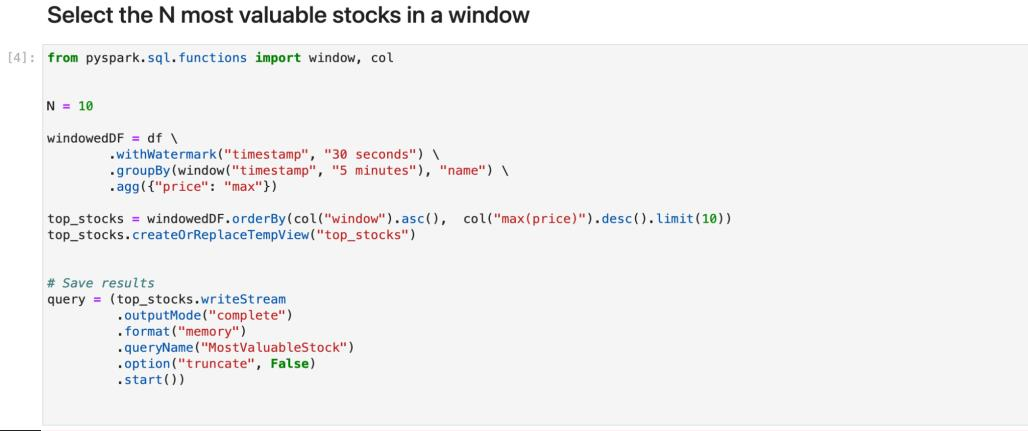
# Data Processing

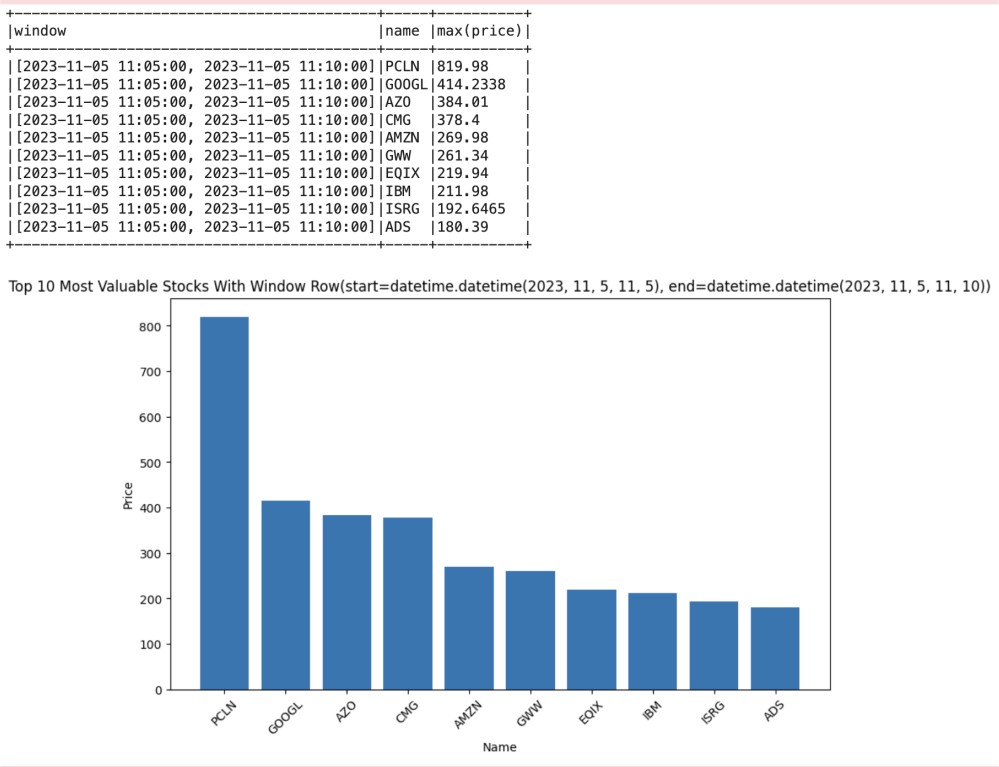
* 1. **The N most valuable stocks in each windows**

For the task of obtaining the N most valuable stocks in each window, we operate with a time window of 5 minutes and set a watermark of 30 seconds. We perform a groupBy operation based on each 5-minute window and the stock's name. Subsequently, we compute the maximum price for each stock within each window. For each stock within each window, we consider selecting the maximum price as a suitable choice because, within a window, a stock may have multiple price updates, and the highest price is the most valuable one.

Next, we sort the data for each window in ascending order of time between windows, and within each window, we sort the data in descending order of price and select the top N most valuable stocks.

We save the processed data into memory with the name "MostValuableStock." We can access "MostValuableStock" to display the results and perform basic visualizations on the outcome.



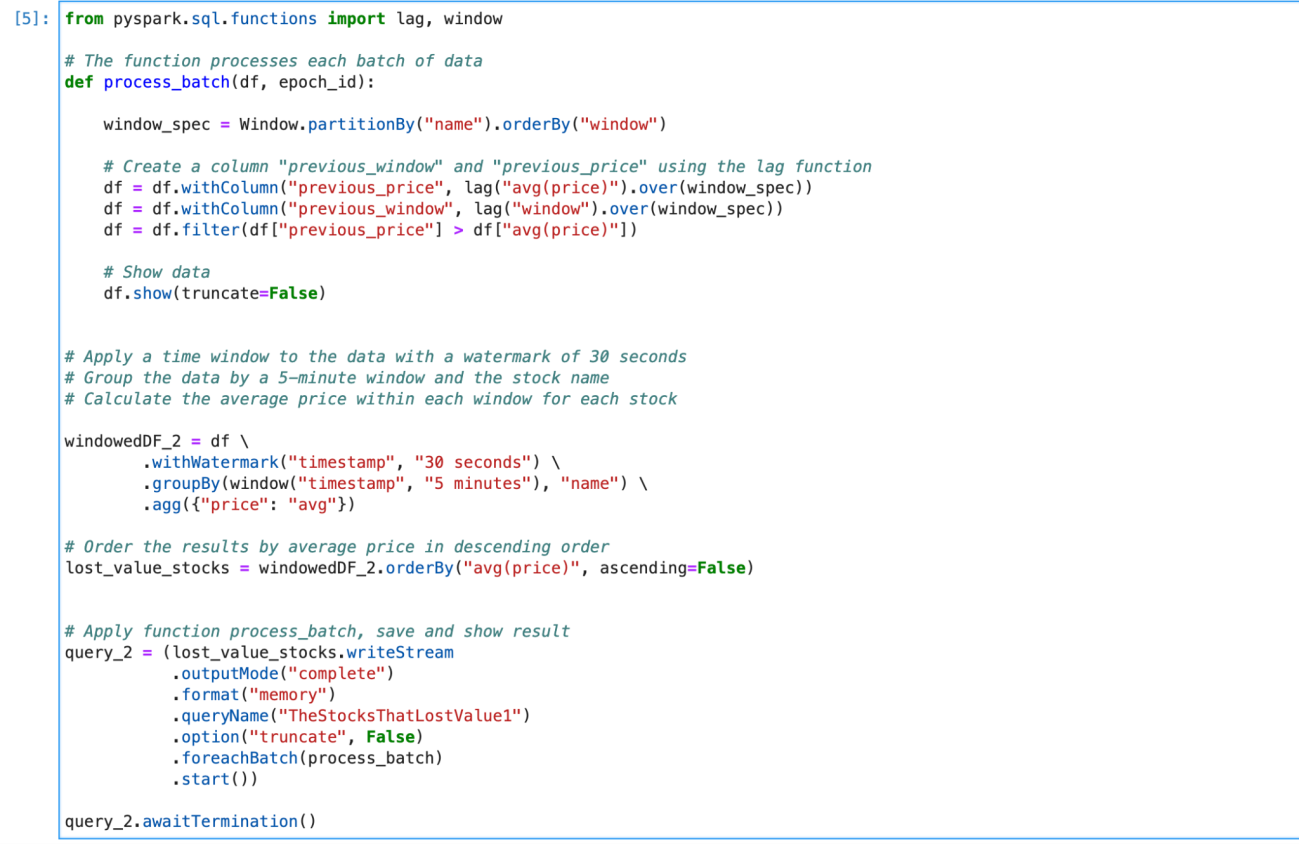
=> Result: We select the 10 most valuable stocks

*Top 10 most valuable stock*

# Select the stocks that lost value between windows

For the task of "Select the stocks that lost value between two windows," our main idea is to compare the average price of each stock within a window with its average price in the previous window.

To achieve this, we have created a function called process\_batch(df, epoch\_id) to process each batch (using foreachBatch).



We apply a time window to the data with a watermark of 30 seconds. Sau đó, group the data by a 5-minute window and the stock name và calculate the average price within each window for each stock.

For process\_batch - a function that processes each batch of data in a Spark Structured Streaming job. The goal of this code is to identify stocks that have experienced a loss in value between consecutive time windows and display the relevant information about these stocks for further analysis. The lag function is used to access the previous window's data and compare it with the current window's data to detect value losses.

For each batch of data, the process\_batch function should be applied.

This function processes the batch and identifies stocks that have lost value.

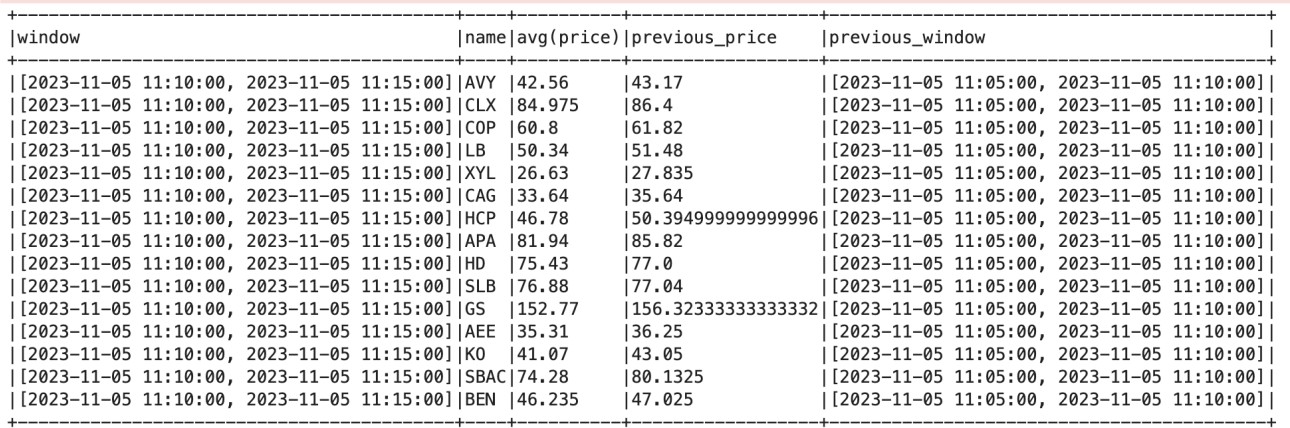
=> Result:

Table 1: *The stocks that lost value between windows*

Table 1 hiện thị the stocks that lost value between window. In this table, you can observe that stocks with decreasing values between windows, for instance, AVY had a price of 43.17 in the previous window, but in the current window, it has reduced to 42.56

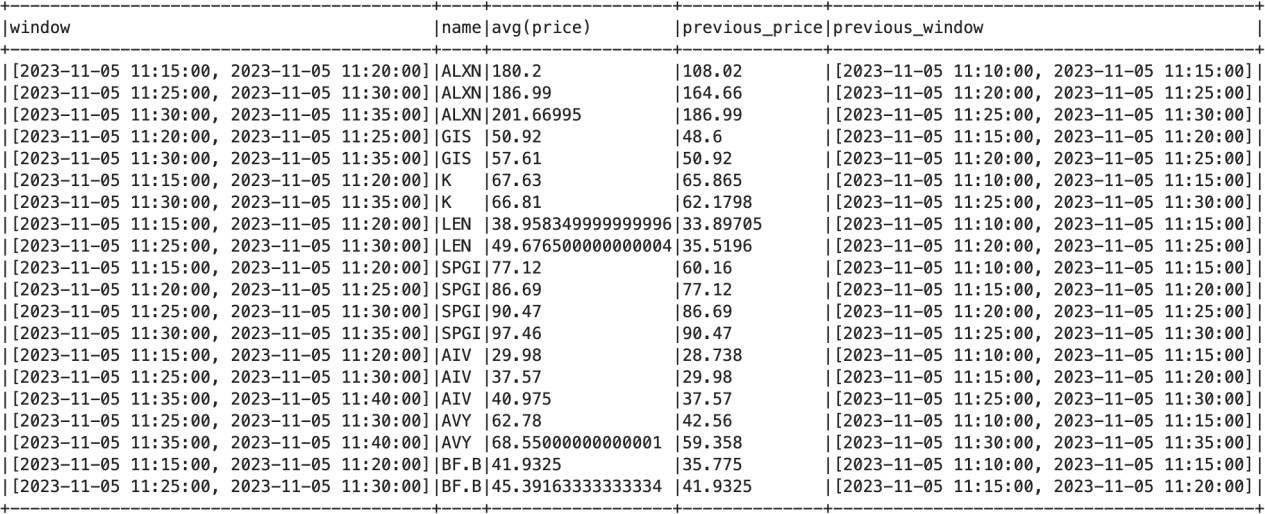
# Find the stocks that gained the most between windows

Similar to the steps performed in task 2, in this task, we are looking for stocks that have increased in value between windows.

We filter out stocks with an average price in the current window greater than their average price in the previous window. This means that they have experienced a price increase compared to the previous time window.



=> Result:



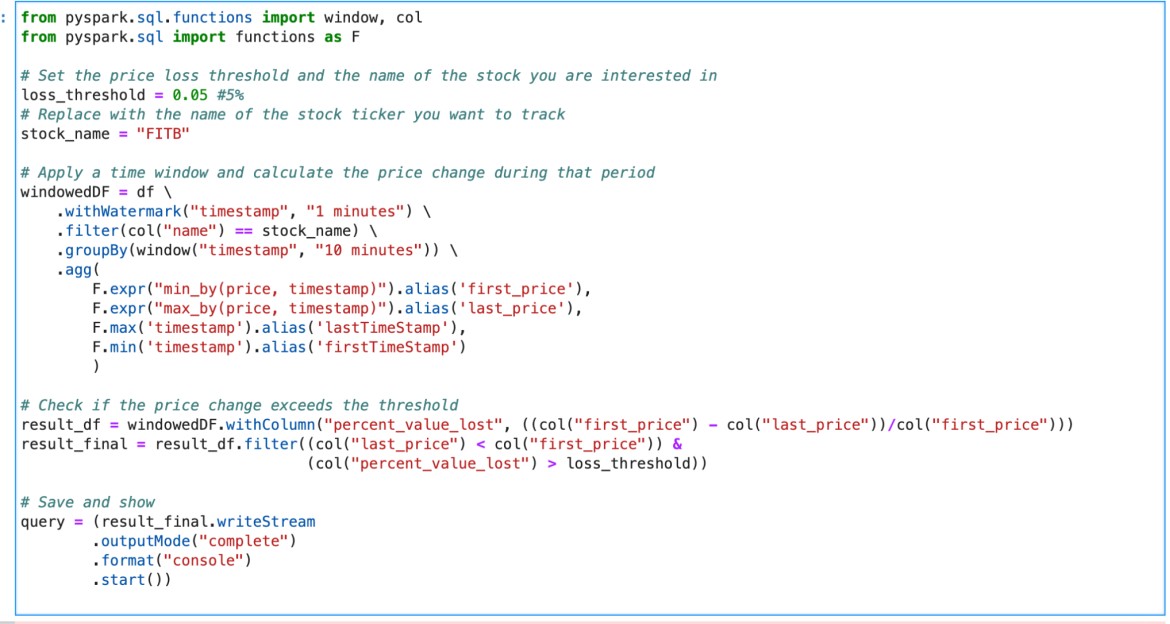
*Table 2: the stocks that gained the most between windows*

# Implement a control that checks if a stock does not lose too much value in a period of time

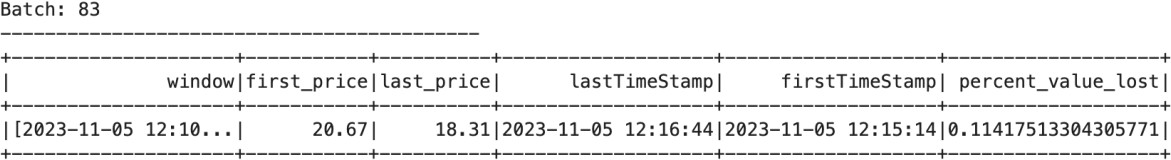
We apply a time window and calculate the price change during that period. Here, we're using a 10-minute window with a 1-minute watermark to track changes in the stock's price over time. We filter the data to select rows with the specified stock name. (example: FITB)

We then calculate the percentage price change between the first and last prices within each time window.

We filter and select rows where the last price is lower than the first price and where the percentage price change exceeds the defined loss threshold.

This code helps monitor and identify cases where a specific stock's price decreases beyond a predefined threshold within the specified time window.

=> Result:



*Table 3: The FITB stock lost beyond the threshold.*

# Compute how your asset changes with the fluctuation of the market

We have created a DataFrame called stock\_portfolio, which represents the stocks you own along with the amount of each stock. We then join this DataFrame with the df DataFrame (which contains stock price data) using the common column "name."

After joining, we calculate the asset value for each stock by multiplying the "amount\_of\_stocks\_owned" by the "price" for each stock. This is done to compute the asset value for each stock you own.

Finally, we calculate the total asset value by summing up the asset values of all the stocks in your portfolio using the selectExpr function.

The result of this computation is written to a streaming query named "AssetValue1" with an output mode set to "update," which means the result will be updated as new data arrives.



=> Result:

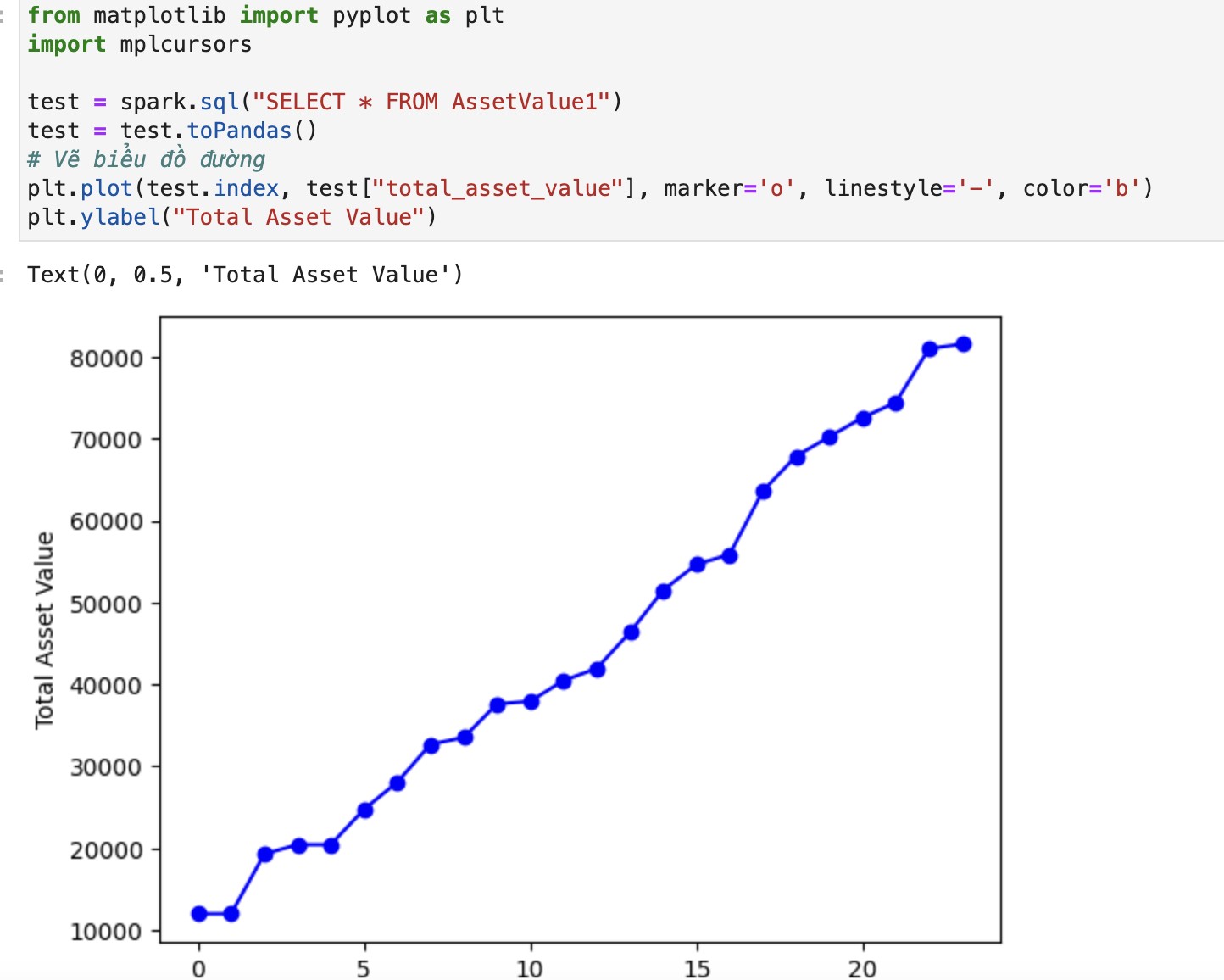


Figure 1: The change in total asset value.

# Structure of the Project and Instructions

*Figure 2: Structure of the Project*

The project consists of 7 code files, including:

* Kafka-Producer-for-project.ipynb
* Project\_template.ipynb
* Task1.ipynb
* Task2.ipynb
* Task3.ipynb
* Task4.ipynb
* Task5.ipynb

In the Template\_Project file, we have implemented the code for all 5

tasks.

We have also divided each task into a separate code file for easier tracking and debugging.

To execute this project, you should start by running the Kafka\_project file to read data from the stock.csv file. Then, you can run each cell in the Template\_Project file, or run each task separately by executing the corresponding individual files.

Make sure to install Kafka and Spark to run the code.

# Conclusion

## In this project, we have conducted an analysis of 5 tasks related to stock market analysis. In the future, we want to enhance the quality of analysis and focus on improving visualization.