Université Claude Bernard Lyon 1

TP2 - Analyzing Stock Market Values

Prepared By

Danish Touseeq

Data Processing and Analytics (DPA) Course - International Master DISS UCBL

24/25

1. Introduction

This project demonstrates the use of modern technologies like Apache Kafka and Apache Spark Streaming for real-time data processing. The focus is on analyzing stock market data, showing how these tools facilitate continuous data stream management and analysis. Stock market data serves as an example to illustrate how information is collected, sent to Kafka in real time, and then processed by Spark. This project provides insights into stream processing and data analysis in dynamic environments like financial markets.

2. Explanation of the Process

The goal of this project is to analyze stock market data in real time using Apache Kafka and Apache Spark Streaming. The process can be broken down into four main stages: data ingestion, data processing, analysis, and visualization.

2.1 Data Ingestion

The first step is ingesting stock market data, which is typically sourced from real-time streams. In this project, Apache Kafka is used to simulate the streaming of stock data.

- A Kafka Producer is implemented to simulate the continuous flow of stock data by producing messages containing stock information (such as stock name, price, and timestamp).
- The Kafka Producer sends data messages in real time to a Kafka topic where they are temporarily
- stored before being consumed by Apache Spark for further processing.

2.2 Data Processing with Apache Spark Streaming

Once the data is streamed into Kafka, Apache Spark Streaming is used to process the incoming data. Key steps in the data processing phase include:

- **Connecting to Kafka:** Apache Spark is configured to connect to the Kafka topic where the stock data is being produced. Spark continuously consumes the messages from the Kafka stream as they arrive.
- **Transforming the Data:** The incoming raw data is structured into components (stock name, price, and timestamp). This structured data is then converted into a DataFrame for easier manipulation and analysis.

- **Applying Transformations:** Spark processes the data by applying various transformations:
    - **Grouping by Time Windows:** Data is grouped into time windows (e.g., 5-minute windows) to facilitate real-time analysis.
    - **Sorting by Stock Price:** Within each time window, the stocks are sorted by price to identify the top N most valuable stocks.

2.3 Real-time Analysis

After the data has been processed, several real-time analyses are performed to extract valuable insights. Key analyses include:

- **Identifying Top N Stocks:** For each time window, the top N most valuable stocks are determined by sorting them based on their average price within the window.
- **Tracking Stock Gains and Losses:** Stock price movements are compared between time windows to detect the biggest gainers and losers.
- **Monitoring Price Changes:** A threshold can be set to detect significant price changes, such as identifying stocks that have lost more than 5% of their value over a specific period.

2.4 Data Flow Overview

To summarize the flow of data in this project:

1. **Kafka Producer:** Stock market data is read from a CSV or simulated source and sent in real-time to a Kafka topic.
2. **Spark Streaming:** Apache Spark consumes this data from Kafka, processes it by applying transformations such as time-based grouping and sorting, and performs the necessary real-time analysis.
3. **Real-time Results:** The processed data is displayed in real-time, showcasing the most valuable stocks, top gainers, and losers, and detecting significant price movements. The results are used for decision-making in a dynamic market environment.

3. Approach and Results

3.1 Top N Most Valuable Stocks in Each Time Window

Method:

To calculate the top N most valuable stocks within each time window, the approach followed was as follows:

1. **Grouping by Time Windows:**
   The stock price data was grouped into 5-minute time windows. Using Apache Spark's `window` function, the data was divided into fixed-sized time intervals to ensure real-time processing.
2. **Calculating Average Prices:**
   The average stock price for each stock was computed within each time window. This aggregation was done using Spark's `avg()` function, allowing the stock's price data to be averaged across each time window for evaluation.
3. **Ranking Stocks:**
   The stocks were ranked based on their average price within each time window using the `rank()` window function. This function ordered the stocks in descending order by their average price, enabling the identification of top-performing stocks in real-time.
4. **Selecting Top N Stocks:**
   Once ranked, the top N (in this case, top 10) stocks were selected by sorting them in descending order and limiting the selection to the highest-ranked stocks. These selected stocks were then prepared for further analysis and visualization.

Results

The analysis returned a table displaying:

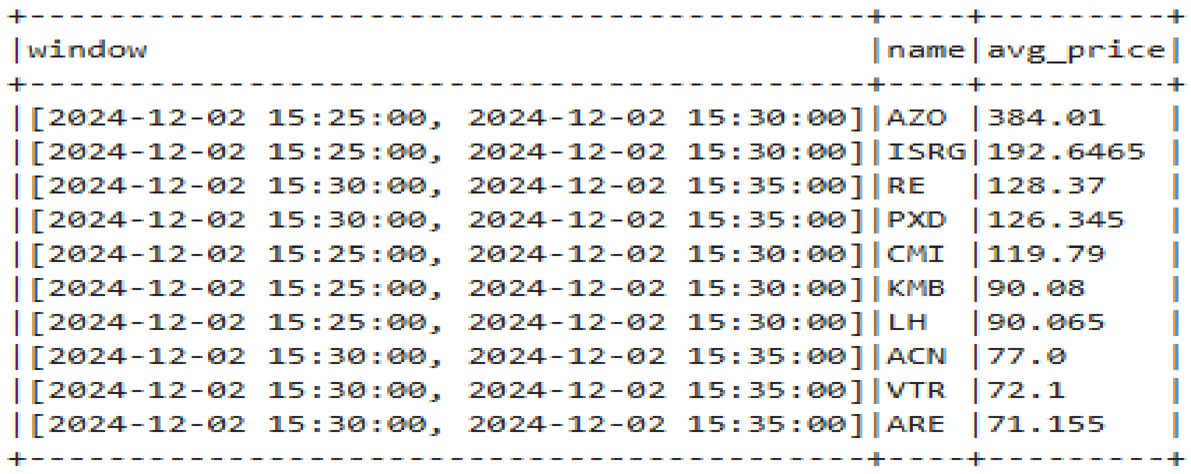
- The names of the top 10 most valuable stocks.
- The corresponding average prices for each stock.
- The rank of each stock is based on its average price within the time window.

These results were then made available for further real-time analysis and decision-making.

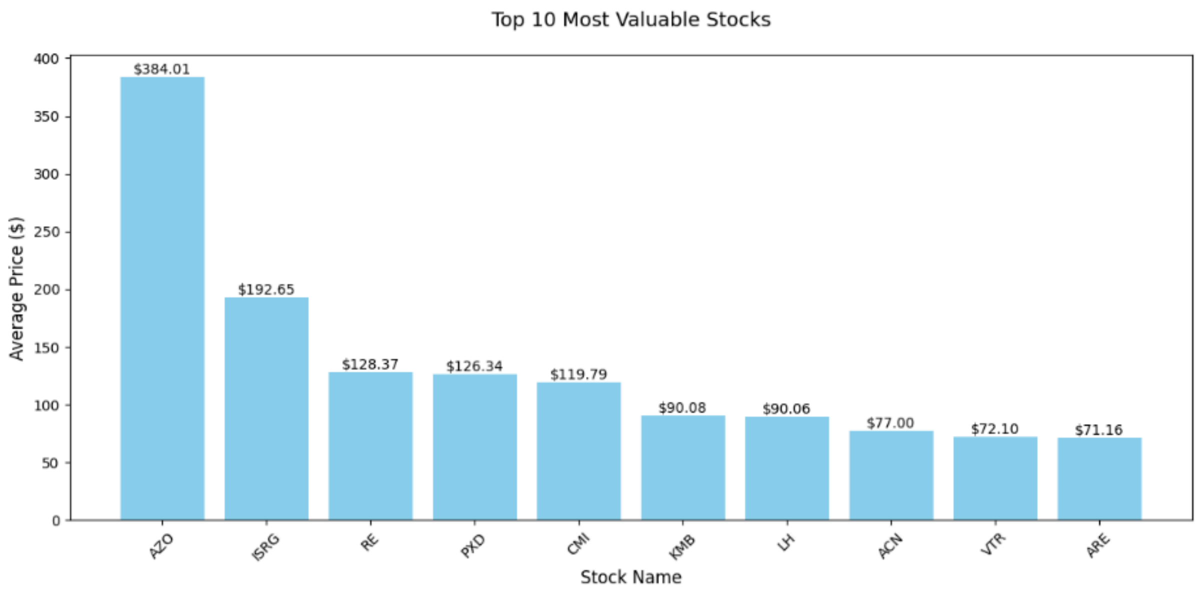
Visualization:

For the visual representation of the top 10 most valuable stocks, a bar chart was generated dynamically using Matplotlib. The chart included:

- **X-axis:** Displayed the stock names.
- **Y-axis:** Represented the average price for each stock in the selected time window.
- **Colors:** The bars were colored in a consistent shade (skyblue), and the heights of the bars visually represented the price differences between the stocks.
- **Value Labels:** Each bar had its respective price labeled on top to provide easy reference for the stock's value. And stock Name and average price are labeled.



```
+------------------------------------------+----+---------+
|window                                    |name|avg_price|
+------------------------------------------+----+---------+
|[2024-12-02 15:25:00, 2024-12-02 15:30:00]|AZO |384.01   |
|[2024-12-02 15:25:00, 2024-12-02 15:30:00]|ISRG|192.6465 |
|[2024-12-02 15:30:00, 2024-12-02 15:35:00]|RE  |128.37   |
|[2024-12-02 15:30:00, 2024-12-02 15:35:00]|PXD |126.345  |
|[2024-12-02 15:25:00, 2024-12-02 15:30:00]|CMI |119.79   |
|[2024-12-02 15:25:00, 2024-12-02 15:30:00]|KMB |90.08    |
|[2024-12-02 15:25:00, 2024-12-02 15:30:00]|LH  |90.065   |
|[2024-12-02 15:30:00, 2024-12-02 15:35:00]|ACN |77.0     |
|[2024-12-02 15:30:00, 2024-12-02 15:35:00]|VTR |72.1     |
|[2024-12-02 15:30:00, 2024-12-02 15:35:00]|ARE |71.155   |
+------------------------------------------+----+---------+
```



3.2 Stocks That Lost Value Between Two Windows

Method

To identify stocks that lost value between two consecutive time windows, the following steps were carried out:

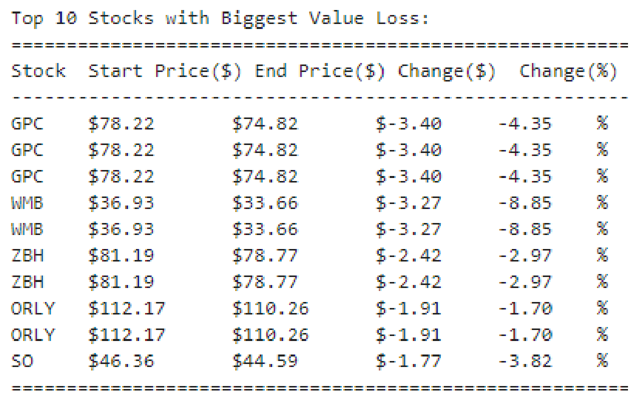
1. **Data Grouping and Maximum Price Calculation**:
   The data was grouped by time windows and stock names, and the maximum price for each stock in each window was calculated.
2. **Lag Function**:
   The `lag()` function was used to retrieve the previous window’s maximum price for each stock. This allowed us to compare the current window's price with the previous one.
3. **Comparison and Detection**:
   A comparison was made between the current maximum price and the previous maximum price to detect a drop in value. If the current price was lower than the previous price, the stock was identified as having lost value.
4. **Stock Selection**:
   Stocks with a lower current price compared to the previous window were selected as stocks that lost value.

Results

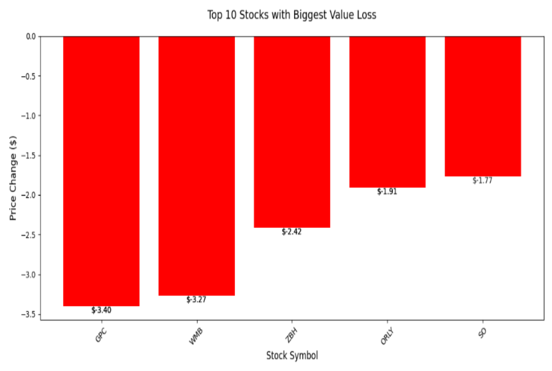
A table displays the stocks that lost value, along with their current maximum price and previous maximum price. These stocks were filtered by their value drop.

Output:

1. **Console Output**:
   - A table showing the stocks that lost value between the two time windows, along with the price change.
2. **Visualization**:
   - A bar chart with red bars representing the price drop for the top 10 stocks that lost value, clearly showing the biggest losses in terms of price change.



```
Top 10 Stocks with Biggest Value Loss:
============================================================
Stock  Start Price($) End Price($) Change($)  Change(%)
------------------------------------------------------------
GPC    $78.22         $74.82       $-3.40     -4.35     %
GPC    $78.22         $74.82       $-3.40     -4.35     %
GPC    $78.22         $74.82       $-3.40     -4.35     %
WMB    $36.93         $33.66       $-3.27     -8.85     %
WMB    $36.93         $33.66       $-3.27     -8.85     %
ZBH    $81.19         $78.77       $-2.42     -2.97     %
ZBH    $81.19         $78.77       $-2.42     -2.97     %
ORLY   $112.17        $110.26      $-1.91     -1.70     %
ORLY   $112.17        $110.26      $-1.91     -1.70     %
SO     $46.36         $44.59       $-1.77     -3.82     %
============================================================
```



3.3 Stocks That Gained the Most Value Between Windows

Method:

To identify stocks that gained the most value between two consecutive time windows, the following steps were executed:

1. **Grouping Data by Time Windows**:
   - The data was grouped by stock names and time windows, and the maximum price for each stock in each window was calculated.
2. **Lag Function to Compare Prices**:
   - The `lag()` function was used to retrieve the previous maximum price for each stock in the prior time window.
3. **Calculate Gain in Value**:
   - The gain in value was calculated by subtracting the previous maximum price from the current maximum price for each stock.
4. **Selecting Stocks with Positive Gains**:
   - Stocks that showed a positive gain in value were selected (i.e., `gain_value > 0`).
5. **Sorting the Results**:
   - The results were sorted by the gain in value in descending order, with the top stock showing the highest gain.

Results:

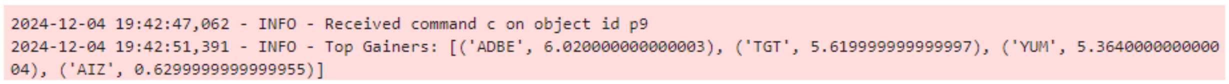
The results were displayed in a Spark table that showed the stocks with the highest gains, along with their respective gain in value and prices in each window. The top stock with the highest gain was displayed at the top.

Visualization:

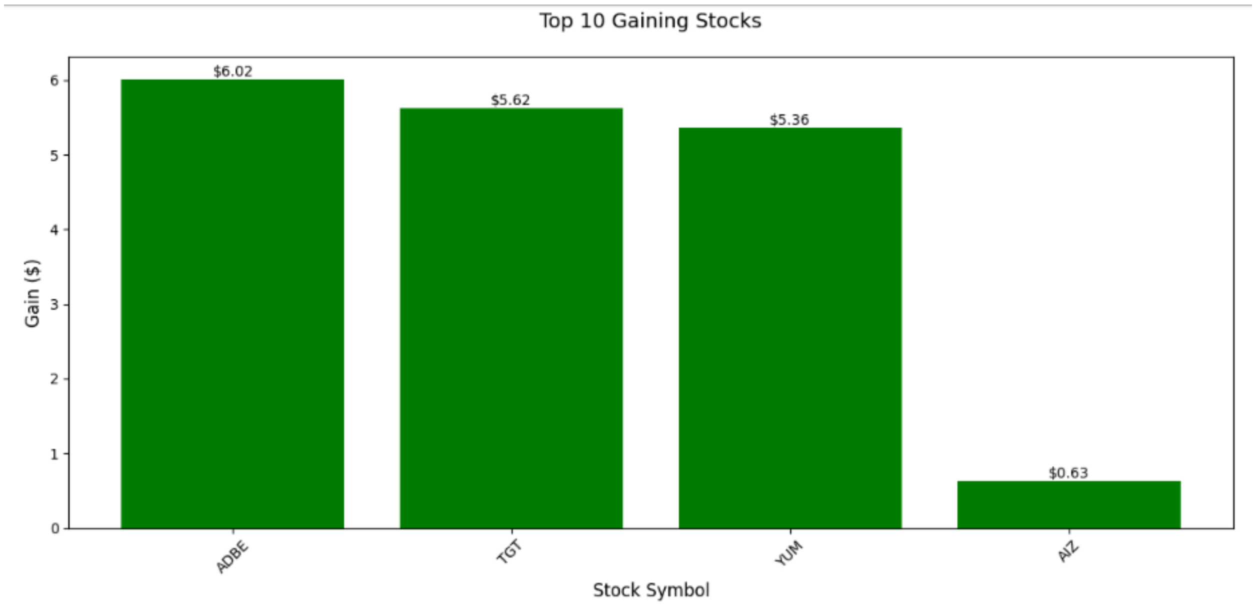
A bar chart was generated to visualize the gain in value for each stock:

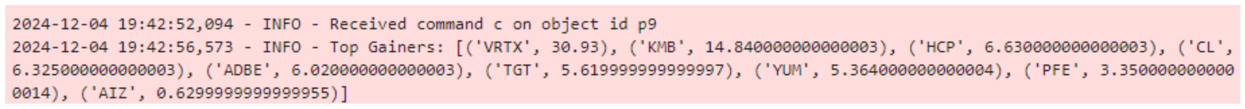
- **x-axis**: Displays the stock names.
- **y-axis**: Shows the gain in value for each stock.
- **Green Bars**: Represent the gain in value for each stock.

This visualization allows for easy identification of the top-performing stocks in terms of value gain, making it straightforward to spot the stock that experienced the highest gain.

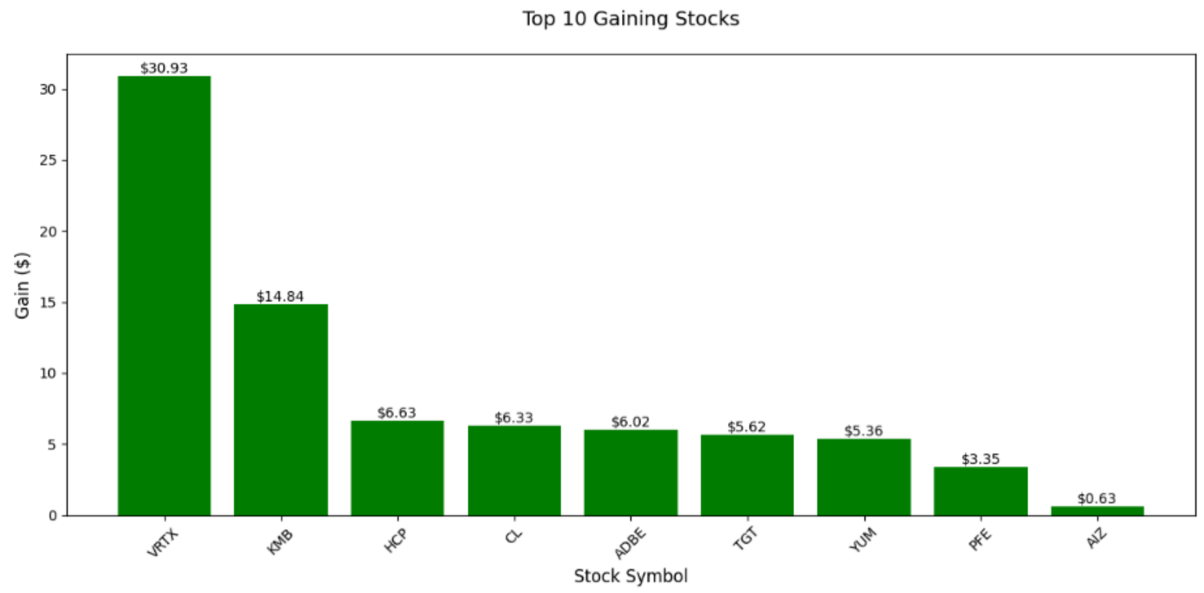


```
2024-12-04 19:42:47,062 - INFO - Received command c on object id p9
2024-12-04 19:42:51,391 - INFO - Top Gainers: [('ADBE', 6.020000000000003), ('TGT', 5.619999999999997), ('YUM', 5.3640000000000
04), ('AIZ', 0.6299999999999955)]
```





```
2024-12-04 19:42:52,094 - INFO - Received command c on object id p9
2024-12-04 19:42:56,573 - INFO - Top Gainers: [('VRTX', 30.93), ('KMB', 14.840000000000003), ('HCP', 6.630000000000003), ('CL',
6.325000000000003), ('ADBE', 6.020000000000003), ('TGT', 5.619999999999997), ('YUM', 5.364000000000004), ('PFE', 3.3500000000000
0014), ('AIZ', 0.6299999999999955)]
```



3.4 Control for Significant Losses

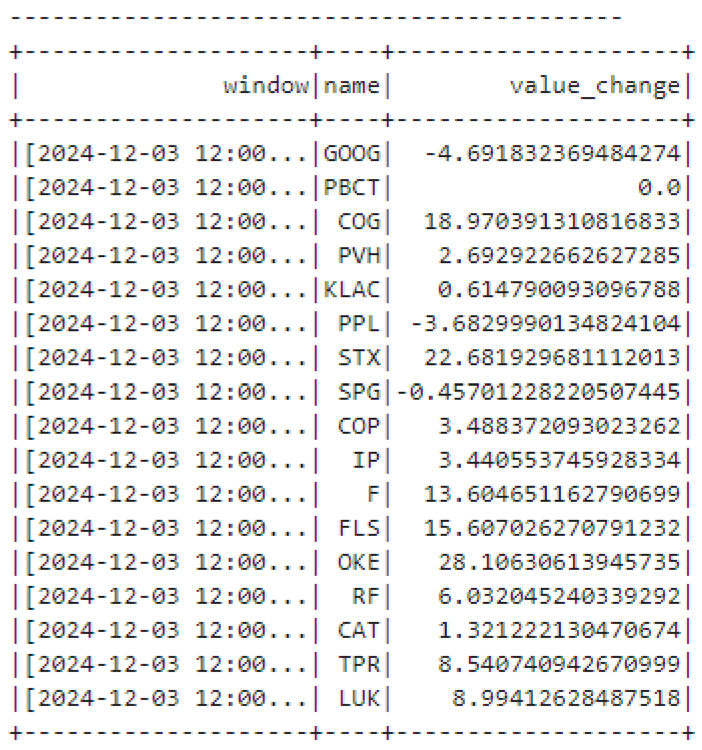
Method:

To implement a control that checks if a stock has lost too much value over a defined period of time (e.g., a loss greater than 5%), the following steps were followed:

1. **Grouping Data by Time Window and Stock**:
   The data was grouped by 5-minute time windows and stock names. The maximum price for each stock was computed in each window.
2. **Calculating the Previous Maximum Price**:
   The `lag()` function was used to obtain the previous maximum price for each stock. This allows for a comparison between the current price and the previous maximum price.
3. **Calculating the Loss Percentage**:
   The loss percentage was calculated by comparing the current maximum price with the previous one. If the current price is lower, the loss percentage was computed.
4. **Filtering Stocks That Lost More Than 5%**:
   Stocks with a loss greater than 5% compared to the previous window are filtered out and considered as having lost too much value.
5. **Displaying the Results**:
   The results were shown in the console with information about the stocks that exceeded the loss threshold, including the current price, the previous price, and the percentage loss.

Results

A Spark table displays the stocks that lost more than 5% of their value, along with their current maximum price, previous maximum price, and loss percentage. The table provides the necessary information to identify the stocks that need further analysis.



```
-------------------------------------------
+--------------------+----+--------------------+
|              window|name|        value_change|
+--------------------+----+--------------------+
|[2024-12-03 12:00...|GOOG|  -4.691832369484274|
|[2024-12-03 12:00...|PBCT|                 0.0|
|[2024-12-03 12:00...| COG|  18.970391310816833|
|[2024-12-03 12:00...| PVH|    2.692922662627285|
|[2024-12-03 12:00...|KLAC|   0.614790093096788|
|[2024-12-03 12:00...| PPL| -3.6829990134824104|
|[2024-12-03 12:00...| STX|  22.681929681112013|
|[2024-12-03 12:00...| SPG|-0.45701228220507445|
|[2024-12-03 12:00...| COP|   3.488372093023262|
|[2024-12-03 12:00...|  IP|   3.440553745928334|
|[2024-12-03 12:00...|   F|  13.604651162790699|
|[2024-12-03 12:00...| FLS|  15.607026270791232|
|[2024-12-03 12:00...| OKE|   28.10630613945735|
|[2024-12-03 12:00...|  RF|   6.032045240339292|
|[2024-12-03 12:00...| CAT|   1.321222130470674|
|[2024-12-03 12:00...| TPR|   8.540740942670999|
|[2024-12-03 12:00...| LUK|    8.99412628487518|
+--------------------+----+--------------------+
```

3.5 Asset Value Change Calculation:

Method:

This task calculates the change in asset value for each stock based on the price fluctuations and the number of shares owned. The steps followed are:

1. **Grouping Data by Time Window and Stock**:
   The data is grouped by 5-minute time windows and stock names, and the maximum price for each stock within each window is computed.
2. **Calculating the Previous Maximum Price**:
   The `lag()` function is used to retrieve the previous maximum price for each stock. This allows a comparison between the current price and the price from the previous window.
3. **Loading Stock Ownership Data**:

The amount of stocks owned by each entity is calculated by grouping the data by stock name and summing up the `amount` column.

4. **Joining Price and Ownership Data**:
   The price data is joined with the stock ownership data to calculate the change in asset value. The joining key is the stock name, ensuring that the asset change is computed per stock.
5. **Calculating the Asset Change**:
   The change in asset value is calculated by subtracting the previous maximum price from the current maximum price, then multiplying the result by the number of stocks owned.
6. **Filtering Results**:
   Rows where the previous maximum price is missing are filtered out to avoid incorrect calculations.

Results

A Spark table is generated showing the changes in asset value for each stock. This table includes the stock name, the previous and current maximum prices, the number of stocks owned, and the asset change.