Lambton College

Real-Time Stock Price Analysis

PySpark and Kafka Streaming

Project Report

BDM 3603
Big Data Framework

SUBMITTED TO

Ishant Gupta , PhD

Lambton College

SUBMITTED BY

Group: 6

Bharat, Satish, Bikash, Samir, Aravind

DECEMBER 2024

ABSTRACT

This project develops a real-time data pipeline for stock price analysis using Confluent and Databricks, demonstrating scalable and efficient data streaming and processing. Stock price data is ingested in real-time via Kafka, powered by Confluent's comprehensive data streaming platform, which facilitates seamless data flow and governance. Using Databricks and PySpark, the data is processed to calculate metrics such as moving averages, price changes, and volatility, with real-time alerts triggered for significant price fluctuations.

Processed data is stored in Delta tables for real-time and batch analysis, including daily summaries of stock performance metrics like average price and total volume. Interactive dashboards in Databricks visualize stock trends and alerts, offering actionable insights for decision-makers. This solution highlights the potential of integrating Confluent and Databricks for robust, real-time analytics pipelines.

Keywords: Real-time analytics, data streaming, Confluent, Databricks, PySpark, stock price analysis, Delta Lake.

INTRODUCTION

In today's fast-paced financial landscape, the ability to analyze stock prices in real-time is essential for informed decision-making. This project demonstrates the development of a robust real-time analytics pipeline for stock price analysis, integrating Confluent, Databricks, Apache Kafka, and PySpark. By leveraging advanced data streaming and processing tools, the project showcases how to transform raw, chaotic data into actionable insights for financial decision-making.

The pipeline begins by ingesting live or simulated stock price data using Kafka, with Confluent's data streaming platform ensuring seamless data flow, governance, and real-time processing. The data is then processed in Databricks using PySpark, where essential metrics such as moving averages, price changes, and volatility are calculated. Real-time alerts are triggered for significant price fluctuations, enabling proactive decision-making.

Processed data is stored in Delta tables within Databricks, supporting both real-time querying and batch processing. Daily summaries are generated to provide insights such as average prices and total trade volumes. Interactive dashboards built in Databricks visualize stock trends and alerts dynamically, making the data easily interpretable and actionable. This project highlights the potential of combining Confluent and Databricks to build scalable, real-time analytics systems, offering valuable insights for the financial sector and beyond.

System Overview

Here's a breakdown of the steps we followed to implement real-time stock price data streaming using **Confluent Kafka** and **PySpark**:

Producer Script

The producer generates stock price data and sends it to a Kafka topic.

Setup and Configuration:

- Imports required libraries (Producer from *confluent_kafka*, *json*, *random*, etc.).
- Configures the Kafka producer with necessary parameters like bootstrap servers.

Simulated Stock Data:

- Generates random stock data for symbols like MSFT, AMZN, etc.
- Fields include Index, Date, Open, High, Low, Close, Adj_Close, Volume, and CloseUSD.

Publishing to Kafka:

- Each record is serialized into JSON format.
- Published to the Kafka topic stocks_analysis using the produce method.
- A delivery callback (delivery_report) is used to handle success or error responses.

Continuous Streaming:

- The producer continuously sends new stock data every 3 seconds (time.sleep(3)).
- Terminates gracefully on a keyboard interrupt.

Here is the screenshot of the code we have implemented:

```python
# Sample stock symbols
stock_symbols = ['MSFT', 'AMZN', 'AAPL', 'GOOG']

def delivery_report(err, msg):
    """Callback for delivery reports."""
    if err is not None:
        print(f"Delivery failed for record {msg.key()}: {err}")
    else:
        print(f"Record {msg.key()} successfully produced to {msg.topic()} [{msg.partition()}]")

def send_stock_data():
    try:
        while True:
            # Randomly select a stock symbol from the list
            stock_symbol = random.choice(stock_symbols)

            # Generate random stock data with the selected stock symbol
            stock_message = {
                'Index': stock_symbol,  # Use the stock symbol here
                'Date': datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
                'Open': round(random.uniform(100, 1500), 2),
                'High': round(random.uniform(1500, 2000), 2),
                'Low': round(random.uniform(50, 1499), 2),
                'Close': round(random.uniform(100, 1500), 2),
                'Adj_Close': round(random.uniform(100, 1500), 2),
                'Volume': random.randint(10000, 1000000),
                'CloseUSD': round(random.uniform(100, 1500), 2)
            }

            # Produce message to Kafka
            kafka_producer.produce(
                'stocks_analysis',
                key=stock_symbol,  # The stock symbol serves as the key
                value=json.dumps(stock_message),
                callback=delivery_report
            )
            print(f"Sent: {stock_message}")
            kafka_producer.poll(0)
            time.sleep(3)
    except KeyboardInterrupt:
        print("Stopping data production.")
    finally:
        kafka_producer.flush()  # Ensure all messages are delivered

send_stock_data()
```

Fig: producer script

Consumer Script

The consumer reads data from the Kafka topic, processes it, and writes it to a Delta table.

Setup and Configuration:

- Creates a Kafka consumer with required settings like bootstrap servers, security protocols, and topic subscription (stocks_analysis).
- Initializes a PySpark session for processing.

Schema Definition:

- Defines a PySpark schema to structure the incoming synthetic data.

```
# Schema definition for incoming data
data_schema = StructType([
    StructField("Index", StringType(), True),
    StructField("Date", StringType(), True),
    StructField("Open", FloatType(), True),
    StructField("High", FloatType(), True),
    StructField("Low", FloatType(), True),
    StructField("Close", FloatType(), True),
    StructField("Adj_Close", FloatType(), True),
    StructField("Volume", IntegerType(), True),
    StructField("CloseUSD", FloatType(), True)
])
```

Figure: Schema definition

Message Processing:

- Polls Kafka for new messages (poll method).
- Decodes and parses JSON messages safely, handling potential errors gracefully.
- Extracts and defaults values to ensure robustness.

DataFrame Creation and Writing:

- Converts each record into a PySpark DataFrame using the predefined schema.
- Writes the DataFrame to a Delta table in append mode for real-time storage and analytics.

Error Handling and Debugging:

- Logs decoding, processing, and writing errors for debugging purposes.
- Handles message errors from Kafka.

Termination:

- Gracefully closes the Kafka consumer on a keyboard interrupt.

```python
# Function to handle consuming messages and processing them
def process_kafka_messages():
    try:
        while True:
            message = kafka_consumer.poll(1.0)  # Poll for new messages

            if message is not None and not message.error():
                try:
                    # Decode key and value safely and log for debugging
                    message_value = message.value().decode("utf-8") if message.value() else "{}"
                    print(f"Decoded Kafka message: {message_value}")  # Debugging: Log the decoded message

                    # Parse message value as JSON
                    try:
                        record = json.loads(message_value)
                    except json.JSONDecodeError as e:
                        print(f"Error decoding JSON: {e}")
                        continue

                    # Extract fields from the record with defaults to ensure robustness
                    index = record.get("Index", "UnknownIndex")  # Correct field name
                    date = record.get("Date", "UnknownDate")
                    open_price = float(record.get("Open", 0.0))
                    high_price = float(record.get("High", 0.0))
                    low_price = float(record.get("Low", 0.0))
                    close_price = float(record.get("Close", 0.0))
                    adj_close = float(record.get("Adj_Close", 0.0))
                    volume = int(record.get("Volume", 0))
                    close_usd = float(record.get("CloseUSD", 0.0))

                    # Create a DataFrame for the single record
                    data_frame = spark_session.createDataFrame(
                        [(index, date, open_price, high_price, low_price, close_price, adj_close, volume, close_usd)],
                        schema=data_schema
                    )

                    # Display the DataFrame in Databricks (optional for debugging/validation)
                    data_frame.show()

                    # Write the data to a Delta table in append mode
                    try:
                        data_frame.write.format("delta").mode("append").save(delta_table_location)
                        print("Data written to Delta table successfully.")
                    except Exception as e:
                        print(f"Error writing to Delta table: {e}")

                except Exception as e:
                    print(f"Error processing Kafka message: {e}")
            elif message is not None and message.error():
                print(f"Kafka error: {message.error()}")
```

Fig: Consumer implementation

After that, we integrated Confluent Kafka for real-time data streaming because it enables scalable, fault-tolerant message delivery, ensuring a reliable flow of stock price data. Kafka decouples producers and consumers, allowing for efficient data processing and seamless integration with PySpark for real-time analytics.

For Data Processing and Analysis in Real-Time, we have performed the following tasks:

Real-Time Metrics Calculation:

- Calculated key stock metrics like moving averages, price changes, and volatility from the real-time stock price data.

Moving Average Calculation:

- Applied a moving average window to smooth out fluctuations in stock prices and analyze trends over time.

Data Transformation:

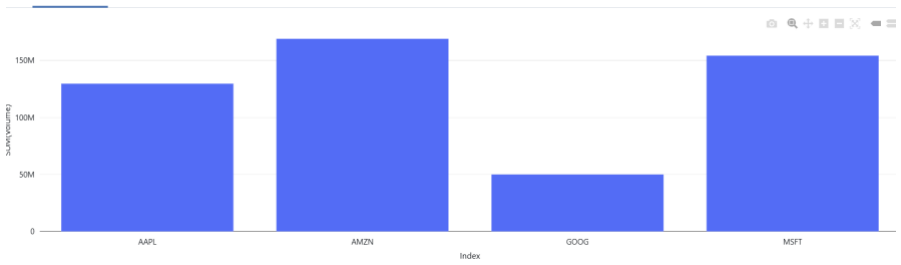
- Processed incoming stock price data into a structured format suitable for analysis and real-time decision-making.

These steps ensure that we can analyze and respond to stock price changes as they happen, in real time.

Visualization and Streaming Data Sink

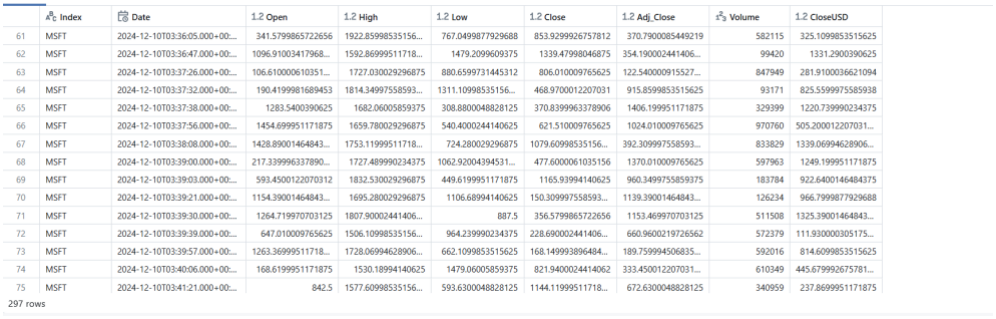
1. Streaming Data Sink:

- **Created Streaming DataFrame:**
  - Loaded data from a Delta table (/mnt/delta/stockprices) using spark.readStream to create a streaming DataFrame (streaming_df).
  - This allows us to continuously process new incoming data from the Delta table in real-time.
- **Display Streaming Data:**
  - Displayed the streaming DataFrame using the display() function to show the live data feed in Databricks.



2. Stock Symbol Filtering:

- **Dropdown Widget for Stock Symbols:**
  - Created a dropdown widget in Databricks with stock symbols (MSFT, AMZN, AAPL, GOOG) for the user to select a stock symbol.
  - Defaulted to the first stock symbol in the list.
- **Filtered Data Based on User Selection:**
  - Retrieved the selected stock symbol from the dropdown widget and used it to filter the streaming DataFrame (streaming_df) by the Index column.
  - This ensures that only data related to the selected stock symbol is displayed.



	Index	Date	1.2 Open	1.2 High	1.2 Low	1.2 Close	1.2 Adj_Close	Volume	1.2 CloseUSD
61	MSFT	2024-12-10T03:36:05.000+00:...	341.5799865722656	1922.85998535156...	767.0499877929688	853.9299926757812	370.7900085449219	582115	325.1099853515625
62	MSFT	2024-12-10T03:36:47.000+00:...	1096.91003417968...	1592.86999511718...	1479.2099609375	1339.47998046875	354.190002441406...	99420	1331.2900390625
63	MSFT	2024-12-10T03:37:26.000+00:...	106.610000610351...	1727.030029296875	880.6599731445312	806.010009765625	122.540000915527...	847949	281.9100036621094
64	MSFT	2024-12-10T03:37:32.000+00:...	190.4199981689453	1814.349975585937...	1311.109985351562...	468.9700012207031	915.8599853515625	93171	825.5599975585938
65	MSFT	2024-12-10T03:37:38.000+00:...	1283.5400390625	1682.060058593750	308.8800048828125	370.8399963378906	1406.199951171875	329399	1220.739990234375
66	MSFT	2024-12-10T03:37:56.000+00:...	1454.699951171875	1659.780029296875	540.4000244140625	621.510009765625	1024.010009765625	970760	505.200012207031...
67	MSFT	2024-12-10T03:38:08.000+00:...	1428.890014648437...	1753.119995117187...	724.280029296875	1079.609985351562...	392.309997558593...	833829	1339.069946289062...
68	MSFT	2024-12-10T03:39:00.000+00:...	217.3399963378906...	1727.489990234375	1062.920043945312...	477.6000061035156	1370.010009765625	597963	1249.199951171875
69	MSFT	2024-12-10T03:39:03.000+00:...	593.4500122070312	1832.530029296875	449.6199951171875	1165.93994140625	960.3499755859375	183784	922.6400146484375
70	MSFT	2024-12-10T03:39:21.000+00:...	1154.390014648437...	1695.280029296875	1106.689941406250	150.3099975585937...	1139.390014648437...	126234	966.7999877929688
71	MSFT	2024-12-10T03:39:30.000+00:...	1264.719970703125	1807.900024414062...	887.5	356.5799865722656	1153.469970703125	511508	1325.390014648437...
72	MSFT	2024-12-10T03:39:39.000+00:...	647.0100097656250	1506.109985351562...	964.2399902343750	228.6900024414062...	660.9600219726562	572379	111.9300003051757...
73	MSFT	2024-12-10T03:39:57.000+00:...	1263.369995117187...	1728.069946289062...	662.1099853515625	168.1499938964843...	189.579994506835...	592016	814.6099853515625
74	MSFT	2024-12-10T03:40:06.000+00:...	168.6199951171875	1530.189941406250	1479.060058593750	821.9400024414062	333.450012207031...	610349	445.679992675781...
75	MSFT	2024-12-10T03:41:21.000+00:...	842.5	1577.609985351562...	593.6300048828125	1144.119995117187...	672.6300048828125	340959	237.8699951171875

297 rows

3. Real-Time Aggregation (Moving Averages & Volatility):

- **Grouped Data by Time Window:**
  - Grouped data by a 1-minute window using window(col("Date"), "1 minute") and the stock Index.
- **Calculated Moving Average and Volatility:**
  - Applied aggregation functions (avg() and stddev()) to calculate the moving average (moving_avg) and volatility (volatility) for each stock symbol over the specified time window.
  - Displayed the results using the display() function for real-time visualization.

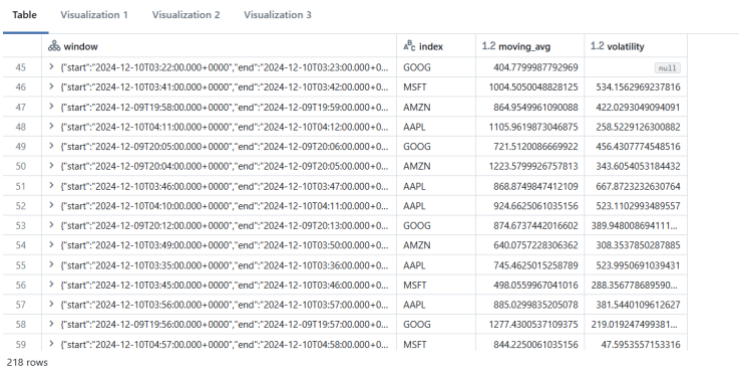


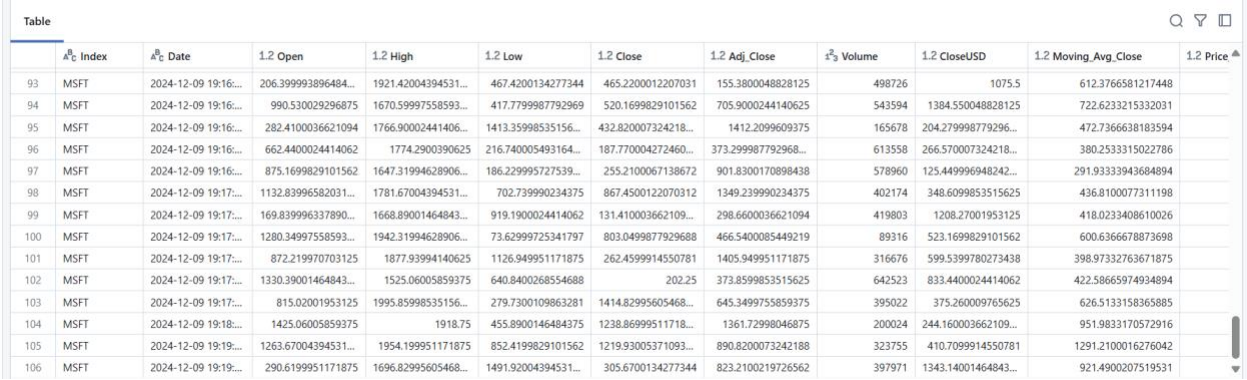
Table | Visualization 1 | Visualization 2 | Visualization 3

	window	index	1.2 moving_avg	1.2 volatility
45	{"start":"2024-12-10T03:22:00.000+0000","end":"2024-12-10T03:23:00.000+0...	GOOG	404.7799987792969	null
46	{"start":"2024-12-10T03:41:00.000+0000","end":"2024-12-10T03:42:00.000+0...	MSFT	1004.5050048828125	534.1562969237816
47	{"start":"2024-12-09T19:58:00.000+0000","end":"2024-12-09T19:59:00.000+0...	AMZN	864.9549961090088	422.0293049094091
48	{"start":"2024-12-10T04:11:00.000+0000","end":"2024-12-10T04:12:00.000+0...	AAPL	1105.9619873046875	258.5229126300882
49	{"start":"2024-12-09T20:05:00.000+0000","end":"2024-12-09T20:06:00.000+0...	GOOG	721.5120086669922	456.4307774548516
50	{"start":"2024-12-09T20:04:00.000+0000","end":"2024-12-09T20:05:00.000+0...	AMZN	1223.5799926757813	343.6054053184432
51	{"start":"2024-12-10T03:46:00.000+0000","end":"2024-12-10T03:47:00.000+0...	AAPL	868.8749847412109	667.8723232630764
52	{"start":"2024-12-10T04:10:00.000+0000","end":"2024-12-10T04:11:00.000+0...	AAPL	924.6625061035156	523.1102993489557
53	{"start":"2024-12-09T20:12:00.000+0000","end":"2024-12-09T20:13:00.000+0...	GOOG	874.6737442016602	389.948008694111...
54	{"start":"2024-12-10T03:49:00.000+0000","end":"2024-12-10T03:50:00.000+0...	AMZN	640.0757228306362	308.3537850287885
55	{"start":"2024-12-10T03:35:00.000+0000","end":"2024-12-10T03:36:00.000+0...	AAPL	745.4625015258789	523.9950691039431
56	{"start":"2024-12-10T03:45:00.000+0000","end":"2024-12-10T03:46:00.000+0...	MSFT	498.0559967041016	288.356778689590...
57	{"start":"2024-12-10T03:56:00.000+0000","end":"2024-12-10T03:57:00.000+0...	AAPL	885.0299835205078	381.5440109612627
58	{"start":"2024-12-09T19:56:00.000+0000","end":"2024-12-09T19:57:00.000+0...	GOOG	1277.4300537109375	219.019247499381...
59	{"start":"2024-12-10T04:57:00.000+0000","end":"2024-12-10T04:58:00.000+0...	MSFT	844.2250061035156	47.5953557153316

218 rows

4. Daily Summary Calculation:

- **Created a Trade Date Column:**
  - Added a new column trade_date by converting the Date column to a date format using to_date(col("Date")).
- **Calculated Daily Stock Summary:**
  - Aggregated the data by Index and trade_date to calculate the daily stock metrics:
  - average_close_price: Average of the Close price for the day.
  - total_volume: Total traded volume for the day.
  - max_high_price: Highest stock price during the day.
  - Saved the result to a Delta table (/mnt/delta/daily_summary) and displayed the summarized data.

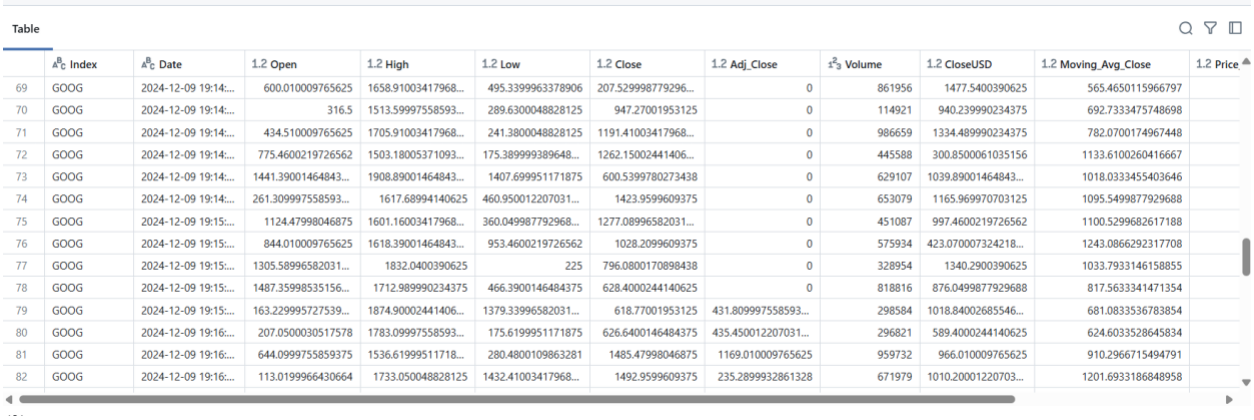


Table

	Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD	Moving_Avg_Close	Price
93	MSFT	2024-12-09 19:16:...	206.399993896484...	1921.42004394531...	467.4200134277344	465.2200012207031	155.3800048828125	498726	1075.5	612.3766581217448	
94	MSFT	2024-12-09 19:16:...	990.530029296875	1670.59997558593...	417.7799987792969	520.1699829101562	705.9000244140625	543594	1384.550048828125	722.6233215332031	
95	MSFT	2024-12-09 19:16:...	282.4100036621094	1766.90002441406...	1413.35998535156...	432.8200073242187...	1412.2099609375	165678	204.279998779296...	472.7366638183594	
96	MSFT	2024-12-09 19:16:...	662.4400024414062	1774.2900390625	216.7400054931640...	187.7700042724609...	373.299987792968...	613558	266.570007324218...	380.2533315022786	
97	MSFT	2024-12-09 19:16:...	875.1699829101562	1647.31994628906...	186.2299957275390...	255.2100067138672	901.8300170898438	578960	125.449996948242...	291.93333943684894	
98	MSFT	2024-12-09 19:17:...	1132.83996582031...	1781.67004394531...	702.7399902343750	867.4500122070312	1349.2399902343750	402174	348.6099853515625	436.8100077311198	
99	MSFT	2024-12-09 19:17:...	169.839996337890...	1668.89001464843...	919.1900024414062	131.4100036621094...	298.6600036621094	419803	1208.27001953125	418.0233408610026	
100	MSFT	2024-12-09 19:17:...	1280.34997558593...	1942.31994628906...	73.62999725341797	803.0499877929688	466.5400085449219	89316	523.1699829101562	600.6366678873698	
101	MSFT	2024-12-09 19:17:...	872.2199707031250	1877.93994140625	1126.949951171875	262.4599914550781	1405.949951171875	316676	599.5399780273438	398.97332763671875	
102	MSFT	2024-12-09 19:17:...	1330.39001464843...	1525.06005859375	640.8400268554688	202.25	373.8599853515625	642523	833.4400024414062	422.58665974934894	
103	MSFT	2024-12-09 19:17:...	815.020019531250	1995.85998535156...	279.7300109863281	1414.82995605468...	645.3499755859375	395022	375.2600097656250	626.5133158365885	
104	MSFT	2024-12-09 19:18:...	1425.06005859375	1918.75	455.8900146484375	1238.86999511718...	1361.72998046875	200024	244.160003662109...	951.9833170572916	
105	MSFT	2024-12-09 19:19:...	1263.67004394531...	1954.199951171875	852.4199829101562	1219.93005371093...	890.8200073242188	323755	410.7099914550781	1291.2100016276042	
106	MSFT	2024-12-09 19:19:...	290.6199951171875	1696.82995605468...	1491.92004394531...	305.6700134277344	823.2100219726562	397971	1343.14001464843...	921.4900207519531	

5. Price Change and Volatility Calculation:

- **Calculated Price Change Percentage:**
  - Used a window function (lag()) to calculate the percentage change in stock price compared to the previous day.
  - Displayed the data with an additional column, Price_Change_Percentage.
- **Calculated Volatility:**
  - Calculated volatility as the difference between High and Low prices, creating a new column Volatility.
  - Displayed the result.

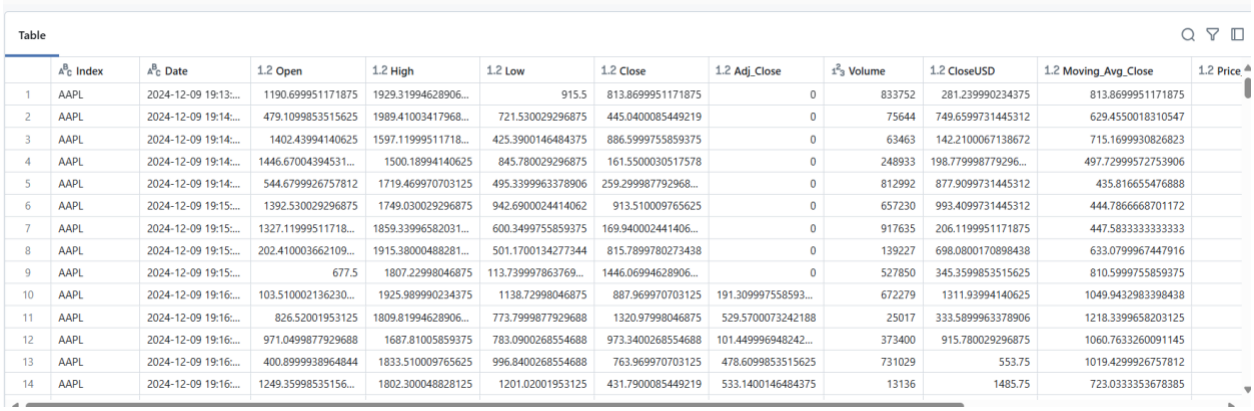


Table

	Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD	Moving_Avg_Close	Price
69	GOOG	2024-12-09 19:14:...	600.010009765625	1658.91003417968...	495.3399963378906	207.529998779296...	0	861956	1477.5400390625	565.4650115966797	
70	GOOG	2024-12-09 19:14:...	316.5	1513.59997558593...	289.6300048828125	947.27001953125	0	114921	940.239990234375	692.7333475748698	
71	GOOG	2024-12-09 19:14:...	434.510009765625	1705.91003417968...	241.3800048828125	1191.41003417968...	0	986659	1334.489990234375	782.0700174967448	
72	GOOG	2024-12-09 19:14:...	775.4600219726562	1503.18005371093...	175.3899993896484...	1262.15002441406...	0	445588	300.8500061035156	1133.6100260416667	
73	GOOG	2024-12-09 19:14:...	1441.39001464843...	1908.89001464843...	1407.699951171875	600.5399780273438	0	629107	1039.89001464843...	1018.0333455403646	
74	GOOG	2024-12-09 19:14:...	261.309997558593...	1617.68994140625	460.9500122070312...	1423.9599609375	0	653079	1165.969970703125	1095.5499877929688	
75	GOOG	2024-12-09 19:15:...	1124.47998046875	1601.16003417968...	360.049987792968...	1277.08996582031...	0	451087	997.4600219726562	1100.5299682617188	
76	GOOG	2024-12-09 19:15:...	844.010009765625	1618.39001464843...	953.4600219726562	1028.2099609375	0	575934	423.070007324218...	1243.0866292317708	
77	GOOG	2024-12-09 19:15:...	1305.58996582031...	1832.0400390625	225	796.0800170898438	0	328954	1340.2900390625	1033.7933146158855	
78	GOOG	2024-12-09 19:15:...	1487.35998535156...	1712.989990234375	466.3900146484375	628.4000244140625	0	818816	876.0499877929688	817.5633341471354	
79	GOOG	2024-12-09 19:15:...	163.229995727539...	1874.90002441406...	1379.33996582031...	618.77001953125	431.809997558593...	298584	1018.84002685546...	681.0833536783854	
80	GOOG	2024-12-09 19:16:...	207.0500030517578	1783.09997558593...	175.6199951171875	626.6400146484375	435.450012207031...	296821	589.4000244140625	624.6033528645834	
81	GOOG	2024-12-09 19:16:...	644.0999755859375	1536.61999511718...	280.4800109863281	1485.47998046875	1169.010009765625	959732	966.010009765625	910.2966715494791	
82	GOOG	2024-12-09 19:16:...	113.0199966430664	1733.050048828125	1432.41003417968...	1492.9599609375	235.2899932861328	671979	1010.20001220703...	1201.6933186848958	

6. Stock Price Alerts:

- **Created Alerts for Price Changes:**
  - Added an alert column that triggers if the *Price_Change_Percentage* exceeds 1% (or any other threshold).
  - Used the when() function to label rows with "Triggered" if the price change condition is met, otherwise "None".
  - Displayed the alert data for real-time monitoring.



Table

	Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD	Moving_Avg_Close	Price
1	AAPL	2024-12-09 19:13:...	1190.699951171875	1929.31994628906...	915.5	813.8699951171875	0	833752	281.239990234375	813.8699951171875	
2	AAPL	2024-12-09 19:14:...	479.1099853515625	1989.41003417968...	721.530029296875	445.0400085449219	0	75644	749.6599731445312	629.4550018310547	
3	AAPL	2024-12-09 19:14:...	1402.43994140625	1597.11999511718...	425.3900146484375	886.5999755859375	0	63463	142.2100067138672	715.1699930826823	
4	AAPL	2024-12-09 19:14:...	1446.67004394531...	1500.18994140625	845.780029296875	161.5500030517578	0	248933	198.779998779296...	497.72999572753906	
5	AAPL	2024-12-09 19:14:...	544.6799926757812	1719.469970703125	495.3399963378906	259.299987792968...	0	812992	877.9099731445312	435.816655476888	
6	AAPL	2024-12-09 19:15:...	1392.530029296875	1749.030029296875	942.6900024414062	913.510009765625	0	657230	993.4099731445312	444.7866668701172	
7	AAPL	2024-12-09 19:15:...	1327.11999511718...	1859.33996582031...	600.3499755859375	169.940002441406...	0	917635	206.1199951171875	447.5833333333333	
8	AAPL	2024-12-09 19:15:...	202.410003662109...	1915.38000488281...	501.1700134277344	815.7899780273438	0	139227	698.0800170898438	633.0799967447916	
9	AAPL	2024-12-09 19:15:...	677.5	1807.22998046875	113.739997863769...	1446.06994628906...	0	527850	345.3599853515625	810.5999755859375	
10	AAPL	2024-12-09 19:16:...	103.510002136230...	1925.989990234375	1138.72998046875	887.9699707031125	191.309997558593...	672279	1311.93994140625	1049.9432983398438	
11	AAPL	2024-12-09 19:16:...	826.52001953125	1809.81994628906...	773.7999877929688	1320.97998046875	529.5700073242188	25017	333.5899963378906	1218.3399658203125	
12	AAPL	2024-12-09 19:16:...	971.0499877929688	1687.81005859375	783.0900268554688	973.3400268554688	101.449996948242...	373400	915.780029296875	1060.7633260091145	
13	AAPL	2024-12-09 19:16:...	400.8999938964844	1833.510009765625	996.8400268554688	763.9699707031125	478.6099853515625	731029	553.75	1019.4299926757812	
14	AAPL	2024-12-09 19:16:...	1249.35998535156...	1802.300048828125	1201.02001953125	431.7900085449219	533.1400146484375	13136	1485.75	723.0333353678385	

Final Dashboard

The final dashboard is designed to provide a comprehensive, real-time view of stock market performance for selected stocks (AAPL, AMZN, GOOG, MSFT). The dashboard consists of several visualization components to analyze stock behaviour, trends, and key metrics.

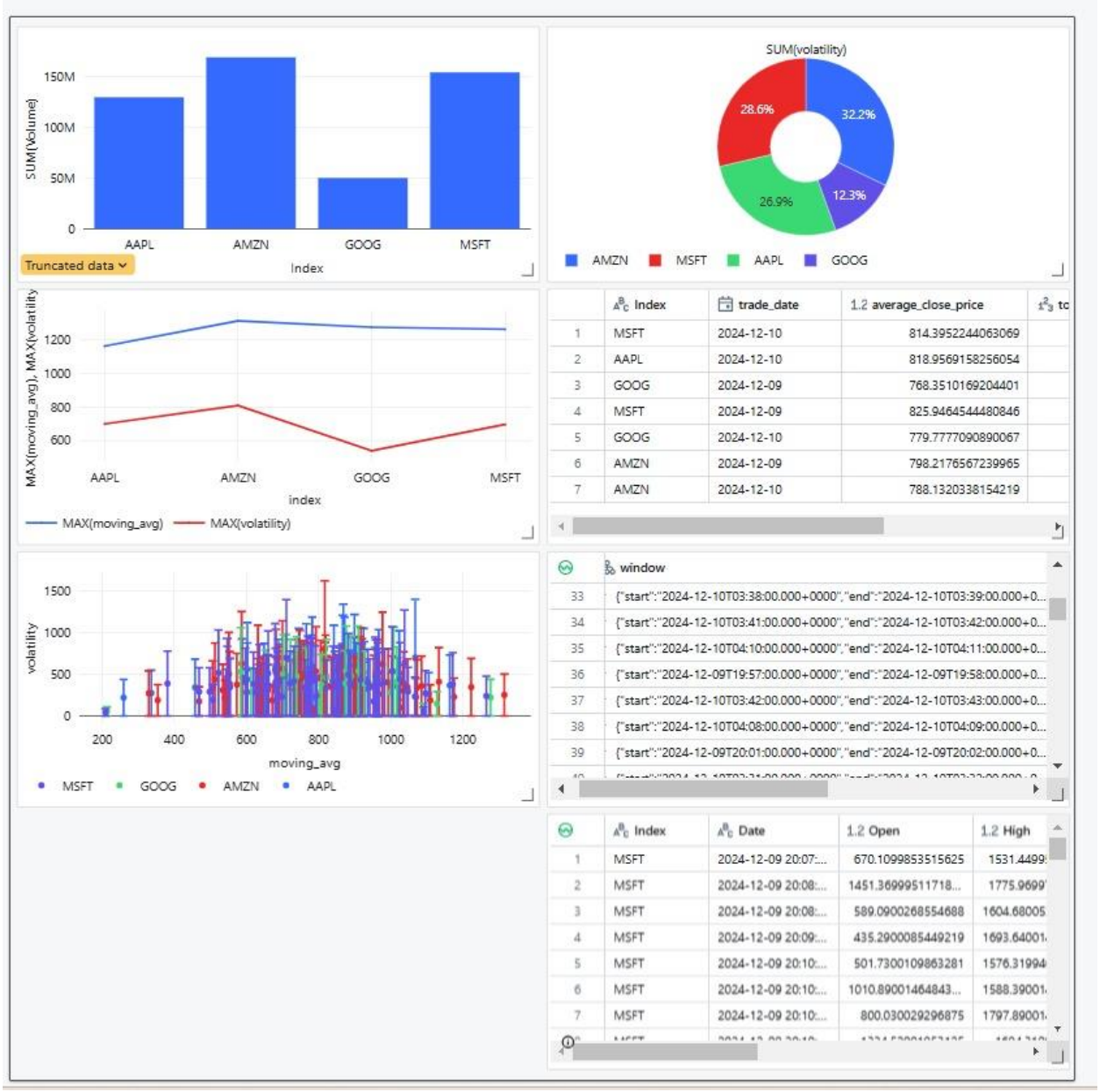


Fig: Final dashboard

Insights

- **Bar Chart (Top Left)**: Shows the total trading volume for stocks (AAPL, AMZN, GOOG, MSFT), identifying the most actively traded stock.
- **Pie Chart (Top Right):** Displays the percentage contribution of each stock to overall volatility.
- **Line Chart (Middle Left)**: Compares maximum moving averages (price trends) and maximum volatilities for stocks.
- **Scatter Plot (Bottom Left)**: Correlates volatility with moving averages to analyse stock price stability and trends.
- **Summary Table (Middle Right)**: Provides key metrics like average closing price for each stock by date.
- **Window Data Table (Bottom Right):** Shows real-time streaming batches with processing windows and metrics.
- **Raw Data Table (Bottom Right Corner):** Displays unprocessed stock data like Open, High, and Volume for analysis.

LIMITATIONS AND FUTURE WORK

Limitations

- **Data Completeness and Quality:** The real-time stock data relies on external sources, and any data gaps or inaccuracies from the Kafka feed could impact the analysis. Missing or erroneous data could affect the reliability of visualizations and metrics.
- **Limited Stock Symbols:** The dashboard currently only supports a limited set of stock symbols (AAPL, AMZN, GOOG, MSFT). The scope can be extended to cover more symbols for broader market analysis.
- **Data Refresh Rate:** The real-time data processing in the streaming pipeline depends on the frequency of incoming data. Delays in receiving data could result in lags in the visualizations, especially for fast-moving stocks.
- **Scalability Issues:** As the volume of incoming stock data grows, the current system might face performance issues, particularly in terms of processing time and storage. This could lead to slower updates or data latency in the visualizations.
- **Limited Advanced Analytics:** The dashboard currently focuses on basic metrics such as moving averages and volatility. More advanced analytics (e.g., sentiment analysis, predictive modelling) could provide deeper insights into market trends.

- **Dependency on Internet Connectivity:** Real-time streaming requires a stable Internet connection to fetch data from external sources. Network outages or instability can cause disruptions in data flow.

Future Work:

- **Expanding Stock Symbol Coverage:** The system can be extended to support a broader set of stock symbols, allowing for a more comprehensive market overview.
- **Enhanced Data Analytics:** Integrate advanced analytics such as machine learning models for price predictions, sentiment analysis from news articles or tweets, and anomaly detection to identify unusual market movements.
- **Real-Time Alerts:** Implement a more robust alert system that triggers notifications for significant price changes, volatility spikes, or other events, providing real-time actionable insights for traders.
- **Improved Data Processing Efficiency:** Optimize the streaming pipeline and integrate distributed computing strategies (e.g., Apache Spark cluster) to improve the scalability and performance of data processing as the dataset grows.
- **Visualization Enhancements:** Enhance the dashboard with additional interactive visualizations such as heatmaps, candlestick charts, and more advanced time-series analysis to provide a richer user experience.
- **Integration with Additional Data Sources:** Incorporate other financial data sources, such as news feeds, market sentiment indicators, or social media sentiment, to enhance stock analysis and provide more context.
- **Historical Data Analysis:** Add capabilities for analyzing historical stock data trends and correlations, enabling users to assess long-term performance and conduct Backtesting of trading strategies.
- **Cloud Deployment:** Deploy the system on cloud platforms to improve accessibility, increase storage, and handle higher volumes of data, making it available for a larger user base without local resource constraints.

By addressing these limitations and implementing future enhancements, the stock price analysis system can be transformed into a more powerful and scalable tool for real-time stock market analysis and decision-making.

CONCLUSION

This project successfully integrates Confluent Kafka for real-time stock price streaming, leveraging PySpark for data processing and visualization. By utilizing Kafka’s robust

message queuing and Spark's powerful analytics capabilities, the system enables efficient and scalable real-time analysis of stock market trends. The interactive dashboard provides insightful visualizations, including bar charts, pie charts, and line graphs, allowing users to track stock performance, volatility, and moving averages. Despite some limitations, such as limited stock symbol coverage and potential scalability challenges, the system offers a solid foundation for monitoring and analyzing stock market data. Future enhancements, such as expanding stock symbols, incorporating advanced analytics, and improving scalability, will further elevate the system's capabilities, making it a valuable tool for real-time financial decision-making.

REFERENCES

1. Confluent Documentation. (n.d.). *Confluent documentation*. Retrieved December 8, 2024, from https://docs.confluent.io/

2. Apache Spark. (n.d.). *Cluster mode overview*. Retrieved December 8, 2024, from https://spark.apache.org/docs/latest/cluster-overview.html

3. Confluent. (n.d.). *confluent_kafka API*. Retrieved December 9, 2024, from https://docs.confluent.io/platform/current/clients/confluent-kafka-python/html/index.html

4. Iwuchukwu, C. (2024, April 23). Analyzing stock prices using PySpark. *Medium*. Retrieved December 7, 2024, from https://medium.com/@ceejayiwufitness/analyzing-stock-prices-using-pyspark-acdaef8a5511

5. ResearchGate. (n.d.). *Real-time streaming data analysis using Spark*. Retrieved December 9, 2024, from https://www.researchgate.net/publication/322674233_Real-time_Streaming_Data_Analysis_using_Spark

In [0]:

```
pip install confluent_kafka
```

```
Python interpreter will be restarted.
Requirement already satisfied: confluent_kafka in /local_disk0/.ephemeral_nfs/envs/python
Env-41618556-6446-4c2d-9846-84da454b184c/lib/python3.9/site-packages (2.6.1)
Python interpreter will be restarted.
```

In [0]:

```
from confluent_kafka import Producer
import json, random, time
from datetime import datetime

# Kafka producer configuration
kafka_producer = Producer({

}

# Sample stock symbols
stock_symbols = ['MSFT', 'AMZN', 'AAPL', 'GOOG']

def delivery_report(err, msg):
    """Callback for delivery reports."""
    if err is not None:
        print(f"Delivery failed for record {msg.key()}: {err}")
    else:
        print(f"Record {msg.key()} successfully produced to {msg.topic()} [{msg.partition
()}]")

def send_stock_data():
    try:
        while True:
            # Randomly select a stock symbol from the list
            stock_symbol = random.choice(stock_symbols)

            # Generate random stock data with the selected stock symbol
            stock_message = {
                'Index': stock_symbol,  # Use the stock symbol here
                'Date': datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
                'Open': round(random.uniform(100, 1500), 2),
                'High': round(random.uniform(1500, 2000), 2),
                'Low': round(random.uniform(50, 1499), 2),
                'Close': round(random.uniform(100, 1500), 2),
                'Adj_Close': round(random.uniform(100, 1500), 2),
                'Volume': random.randint(10000, 1000000),
                'CloseUSD': round(random.uniform(100, 1500), 2)
            }

            # Produce message to Kafka
            kafka_producer.produce(
                'stocks_analysis',
                key=stock_symbol,  # The stock symbol serves as the key
                value=json.dumps(stock_message),
                callback=delivery_report
            )
            print(f"Sent: {stock_message}")
            kafka_producer.poll(0)
            time.sleep(3)
    except KeyboardInterrupt:
        print("Stopping data production.")
    finally:
        kafka_producer.flush()  # Ensure all messages are delivered

send_stock_data()
```

```
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:56:44', 'Open': 604.17, 'High': 1988.23, '
Low': 996.28, 'Close': 330.23, 'Adj_Close': 1100.0, 'Volume': 403622, 'CloseUSD': 1490.72
```

```
}
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:56:47', 'Open': 539.74, 'High': 1972.45, '
Low': 548.15, 'Close': 893.07, 'Adj_Close': 172.4, 'Volume': 40148, 'CloseUSD': 392.56}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:56:50', 'Open': 921.27, 'High': 1994.21, '
Low': 640.28, 'Close': 778.18, 'Adj_Close': 855.62, 'Volume': 74998, 'CloseUSD': 1282.14}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:56:53', 'Open': 177.53, 'High': 1794.86, '
Low': 1277.01, 'Close': 1001.09, 'Adj_Close': 1306.11, 'Volume': 988579, 'CloseUSD': 1424
.01}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:56:56', 'Open': 907.58, 'High': 1956.42, '
Low': 568.83, 'Close': 284.26, 'Adj_Close': 1049.51, 'Volume': 433557, 'CloseUSD': 526.61
}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:56:59', 'Open': 1058.51, 'High': 1732.74,
'Low': 1131.51, 'Close': 835.81, 'Adj_Close': 452.27, 'Volume': 899648, 'CloseUSD': 561.9
6}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:57:02', 'Open': 1120.7, 'High': 1629.35, '
Low': 1352.59, 'Close': 1200.12, 'Adj_Close': 687.14, 'Volume': 107909, 'CloseUSD': 459.9
8}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:57:05', 'Open': 478.78, 'High': 1580.11, '
Low': 784.61, 'Close': 842.87, 'Adj_Close': 1007.82, 'Volume': 731566, 'CloseUSD': 345.29
}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:57:08', 'Open': 1327.83, 'High': 1672.21,
'Low': 742.65, 'Close': 539.86, 'Adj_Close': 1254.96, 'Volume': 168567, 'CloseUSD': 254.7
}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:57:11', 'Open': 495.8, 'High': 1619.17, 'L
ow': 1404.57, 'Close': 1098.77, 'Adj_Close': 641.68, 'Volume': 459528, 'CloseUSD': 249.8}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:57:14', 'Open': 977.88, 'High': 1840.5, 'L
ow': 80.81, 'Close': 303.49, 'Adj_Close': 959.62, 'Volume': 237142, 'CloseUSD': 904.29}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:57:17', 'Open': 114.32, 'High': 1591.66, '
Low': 454.76, 'Close': 1232.75, 'Adj_Close': 1398.87, 'Volume': 724672, 'CloseUSD': 158.5
8}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:57:20', 'Open': 750.36, 'High': 1565.64, '
Low': 524.65, 'Close': 121.95, 'Adj_Close': 1231.74, 'Volume': 846735, 'CloseUSD': 303.16
}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:57:23', 'Open': 766.92, 'High': 1918.84, '
Low': 463.67, 'Close': 661.35, 'Adj_Close': 806.76, 'Volume': 471842, 'CloseUSD': 1039.15
}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:57:26', 'Open': 1034.76, 'High': 1723.88,
'Low': 1187.27, 'Close': 1331.78, 'Adj_Close': 1400.23, 'Volume': 576877, 'CloseUSD': 735
.19}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:57:29', 'Open': 108.12, 'High': 1709.45, '
Low': 362.75, 'Close': 883.66, 'Adj_Close': 481.53, 'Volume': 29427, 'CloseUSD': 388.6}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:57:32', 'Open': 187.18, 'High': 1787.84, '
Low': 938.55, 'Close': 512.17, 'Adj_Close': 839.17, 'Volume': 909215, 'CloseUSD': 1178.03
}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:57:35', 'Open': 1405.77, 'High': 1844.24,
'Low': 1421.64, 'Close': 883.46, 'Adj_Close': 329.64, 'Volume': 877435, 'CloseUSD': 438.9
5}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:57:38', 'Open': 535.01, 'High': 1970.92, '
Low': 985.3, 'Close': 498.63, 'Adj_Close': 400.27, 'Volume': 621894, 'CloseUSD': 254.08}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:57:41', 'Open': 942.86, 'High': 1896.19, '
Low': 809.85, 'Close': 330.03, 'Adj_Close': 845.4, 'Volume': 166045, 'CloseUSD': 1290.83}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:57:44', 'Open': 1156.56, 'High': 1611.08,
'Low': 519.36, 'Close': 1284.48, 'Adj_Close': 1274.9, 'Volume': 418115, 'CloseUSD': 1223.
```

```
31}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:57:47', 'Open': 1097.55, 'High': 1622.68,
'Low': 718.19, 'Close': 486.7, 'Adj_Close': 181.74, 'Volume': 887369, 'CloseUSD': 615.97}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:57:50', 'Open': 187.05, 'High': 1718.09, '
Low': 1065.12, 'Close': 640.76, 'Adj_Close': 804.85, 'Volume': 788759, 'CloseUSD': 485.61
}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:57:53', 'Open': 516.65, 'High': 1880.26, '
Low': 875.97, 'Close': 385.07, 'Adj_Close': 945.02, 'Volume': 601325, 'CloseUSD': 1247.65
}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:57:56', 'Open': 889.94, 'High': 1926.84, '
Low': 326.23, 'Close': 173.26, 'Adj_Close': 955.92, 'Volume': 910061, 'CloseUSD': 1228.52
}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:57:59', 'Open': 857.36, 'High': 1533.09, '
Low': 1251.78, 'Close': 1405.55, 'Adj_Close': 384.67, 'Volume': 231862, 'CloseUSD': 782.6
9}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:58:02', 'Open': 752.83, 'High': 1717.62, '
Low': 1057.44, 'Close': 1233.7, 'Adj_Close': 1201.51, 'Volume': 676401, 'CloseUSD': 978.6
1}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:58:05', 'Open': 658.1, 'High': 1636.05, 'L
ow': 1394.1, 'Close': 186.23, 'Adj_Close': 841.2, 'Volume': 167813, 'CloseUSD': 1049.8}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:58:08', 'Open': 1366.87, 'High': 1652.57,
'Low': 362.37, 'Close': 241.19, 'Adj_Close': 473.14, 'Volume': 549370, 'CloseUSD': 1021.0
2}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:58:11', 'Open': 417.08, 'High': 1868.48, '
Low': 978.87, 'Close': 711.46, 'Adj_Close': 482.47, 'Volume': 819966, 'CloseUSD': 978.35}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:58:14', 'Open': 1435.4, 'High': 1778.59, '
Low': 669.73, 'Close': 1427.59, 'Adj_Close': 877.81, 'Volume': 880358, 'CloseUSD': 1371.4
9}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:58:17', 'Open': 984.95, 'High': 1695.04, '
Low': 201.17, 'Close': 378.27, 'Adj_Close': 357.88, 'Volume': 906387, 'CloseUSD': 1070.09
}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:58:20', 'Open': 1150.82, 'High': 1603.97,
'Low': 378.67, 'Close': 1484.63, 'Adj_Close': 1016.92, 'Volume': 874540, 'CloseUSD': 609.
38}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:58:23', 'Open': 1086.11, 'High': 1556.45,
'Low': 716.48, 'Close': 710.96, 'Adj_Close': 956.18, 'Volume': 870085, 'CloseUSD': 748.23
}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:58:26', 'Open': 167.13, 'High': 1980.54, '
Low': 96.02, 'Close': 1314.97, 'Adj_Close': 1219.56, 'Volume': 577914, 'CloseUSD': 1359.2
7}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:58:29', 'Open': 712.29, 'High': 1903.59, '
Low': 1190.29, 'Close': 348.5, 'Adj_Close': 488.35, 'Volume': 567500, 'CloseUSD': 1179.64
}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:58:32', 'Open': 631.94, 'High': 1542.95, '
Low': 240.32, 'Close': 1051.27, 'Adj_Close': 724.68, 'Volume': 283682, 'CloseUSD': 1012.8
3}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:58:35', 'Open': 1128.1, 'High': 1941.08, '
Low': 828.58, 'Close': 1053.19, 'Adj_Close': 718.04, 'Volume': 191487, 'CloseUSD': 1415.0
9}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:58:38', 'Open': 601.47, 'High': 1552.39, '
Low': 892.77, 'Close': 441.76, 'Adj_Close': 840.82, 'Volume': 174972, 'CloseUSD': 409.26}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:58:41', 'Open': 879.55, 'High': 1694.25, '
Low': 1416.91, 'Close': 955.36, 'Adj_Close': 641.69, 'Volume': 868535, 'CloseUSD': 641.92
```

```
}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:58:44', 'Open': 650.91, 'High': 1748.74, '
Low': 240.81, 'Close': 1079.16, 'Adj_Close': 692.46, 'Volume': 898232, 'CloseUSD': 893.92
}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:58:47', 'Open': 146.39, 'High': 1834.59, '
Low': 729.45, 'Close': 123.71, 'Adj_Close': 1346.37, 'Volume': 728013, 'CloseUSD': 1163.9
7}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:58:50', 'Open': 649.71, 'High': 1810.67, '
Low': 630.0, 'Close': 614.69, 'Adj_Close': 943.11, 'Volume': 305084, 'CloseUSD': 236.39}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:58:53', 'Open': 104.3, 'High': 1705.2, 'Lo
w': 1221.42, 'Close': 249.81, 'Adj_Close': 1272.32, 'Volume': 927637, 'CloseUSD': 782.63}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:58:56', 'Open': 422.74, 'High': 1891.34, '
Low': 905.56, 'Close': 1090.21, 'Adj_Close': 270.37, 'Volume': 657370, 'CloseUSD': 829.9}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:58:59', 'Open': 1422.26, 'High': 1788.88,
'Low': 1037.26, 'Close': 305.05, 'Adj_Close': 778.66, 'Volume': 303198, 'CloseUSD': 554.3
7}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:59:02', 'Open': 140.85, 'High': 1661.42, '
Low': 1050.46, 'Close': 941.8, 'Adj_Close': 567.05, 'Volume': 669073, 'CloseUSD': 1191.45
}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:59:05', 'Open': 914.11, 'High': 1545.74, '
Low': 1308.41, 'Close': 1176.06, 'Adj_Close': 1329.37, 'Volume': 636811, 'CloseUSD': 235.
19}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:59:08', 'Open': 1108.57, 'High': 1764.32,
'Low': 949.43, 'Close': 1018.16, 'Adj_Close': 1075.5, 'Volume': 718540, 'CloseUSD': 1395.
61}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:59:11', 'Open': 754.59, 'High': 1707.41, '
Low': 177.19, 'Close': 830.69, 'Adj_Close': 1428.25, 'Volume': 507764, 'CloseUSD': 882.42
}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:59:14', 'Open': 1147.79, 'High': 1819.12,
'Low': 978.61, 'Close': 578.91, 'Adj_Close': 707.06, 'Volume': 301348, 'CloseUSD': 488.64
}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:59:17', 'Open': 495.93, 'High': 1828.99, '
Low': 125.17, 'Close': 402.39, 'Adj_Close': 505.51, 'Volume': 530957, 'CloseUSD': 121.17}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:59:20', 'Open': 732.21, 'High': 1718.42, '
Low': 335.07, 'Close': 1466.29, 'Adj_Close': 947.41, 'Volume': 83057, 'CloseUSD': 1029.29
}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:59:23', 'Open': 674.12, 'High': 1875.69, '
Low': 1050.76, 'Close': 998.14, 'Adj_Close': 494.42, 'Volume': 76554, 'CloseUSD': 593.97}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:59:26', 'Open': 671.14, 'High': 1864.81, '
Low': 318.06, 'Close': 391.04, 'Adj_Close': 1161.08, 'Volume': 32420, 'CloseUSD': 726.1}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:59:29', 'Open': 458.76, 'High': 1974.6, 'L
ow': 1165.1, 'Close': 748.87, 'Adj_Close': 271.92, 'Volume': 309917, 'CloseUSD': 287.54}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:59:32', 'Open': 769.48, 'High': 1791.7, 'L
ow': 1101.35, 'Close': 961.92, 'Adj_Close': 1486.28, 'Volume': 916661, 'CloseUSD': 762.3}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:59:35', 'Open': 653.88, 'High': 1713.52, '
Low': 251.7, 'Close': 381.42, 'Adj_Close': 815.1, 'Volume': 564176, 'CloseUSD': 1245.46}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:59:38', 'Open': 816.48, 'High': 1961.2, 'L
ow': 551.99, 'Close': 415.2, 'Adj_Close': 654.7, 'Volume': 809006, 'CloseUSD': 118.66}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:59:41', 'Open': 291.78, 'High': 1705.84, '
Low': 1094.65, 'Close': 1279.63, 'Adj_Close': 1499.02, 'Volume': 707466, 'CloseUSD': 746.
13}
Record b'MSFT' successfully produced to stocks_analysis [1]
```

```
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 14:59:44', 'Open': 629.17, 'High': 1648.23, '
Low': 155.44, 'Close': 1388.44, 'Adj_Close': 1149.08, 'Volume': 711238, 'CloseUSD': 146.7
2}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:59:47', 'Open': 593.73, 'High': 1674.41, '
Low': 161.73, 'Close': 721.83, 'Adj_Close': 385.24, 'Volume': 824288, 'CloseUSD': 1281.63
}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 14:59:50', 'Open': 942.1, 'High': 1689.21, 'L
ow': 706.31, 'Close': 701.33, 'Adj_Close': 512.0, 'Volume': 121680, 'CloseUSD': 1482.6}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:59:53', 'Open': 847.12, 'High': 1686.8, 'L
ow': 1319.59, 'Close': 905.25, 'Adj_Close': 839.12, 'Volume': 859769, 'CloseUSD': 1146.09
}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 14:59:56', 'Open': 509.18, 'High': 1574.98, '
Low': 1306.94, 'Close': 419.33, 'Adj_Close': 112.67, 'Volume': 813322, 'CloseUSD': 692.44
}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 14:59:59', 'Open': 378.19, 'High': 1566.47, '
Low': 1297.37, 'Close': 1358.66, 'Adj_Close': 511.52, 'Volume': 684237, 'CloseUSD': 747.5
4}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 15:00:02', 'Open': 1118.49, 'High': 1853.03,
'Low': 756.07, 'Close': 673.3, 'Adj_Close': 967.44, 'Volume': 305512, 'CloseUSD': 1315.44
}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 15:00:05', 'Open': 657.32, 'High': 1670.17, '
Low': 486.45, 'Close': 1103.0, 'Adj_Close': 1238.92, 'Volume': 819245, 'CloseUSD': 1190.5
9}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 15:00:08', 'Open': 225.42, 'High': 1778.17, '
Low': 1070.91, 'Close': 454.86, 'Adj_Close': 1002.31, 'Volume': 193898, 'CloseUSD': 483.9
5}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 15:00:11', 'Open': 1162.74, 'High': 1518.47,
'Low': 630.41, 'Close': 1486.56, 'Adj_Close': 418.0, 'Volume': 469755, 'CloseUSD': 1251.4
7}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 15:00:14', 'Open': 503.0, 'High': 1649.04, 'L
ow': 968.39, 'Close': 1142.65, 'Adj_Close': 887.59, 'Volume': 345804, 'CloseUSD': 294.87}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 15:00:17', 'Open': 1082.54, 'High': 1742.13,
'Low': 1313.0, 'Close': 457.37, 'Adj_Close': 750.21, 'Volume': 833444, 'CloseUSD': 565.28
}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 15:00:20', 'Open': 276.88, 'High': 1994.8, 'L
ow': 391.36, 'Close': 1049.98, 'Adj_Close': 1439.65, 'Volume': 129689, 'CloseUSD': 979.68
}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 15:00:23', 'Open': 471.75, 'High': 1818.58, '
Low': 269.53, 'Close': 461.27, 'Adj_Close': 1309.0, 'Volume': 554473, 'CloseUSD': 1031.58
}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 15:00:26', 'Open': 896.01, 'High': 1690.77, '
Low': 768.66, 'Close': 526.6, 'Adj_Close': 190.43, 'Volume': 610341, 'CloseUSD': 1333.96}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 15:00:29', 'Open': 834.59, 'High': 1602.4, 'L
ow': 822.21, 'Close': 501.19, 'Adj_Close': 1293.34, 'Volume': 204888, 'CloseUSD': 786.85}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 15:00:32', 'Open': 712.0, 'High': 1506.34, 'L
ow': 842.11, 'Close': 209.25, 'Adj_Close': 1454.52, 'Volume': 252838, 'CloseUSD': 1052.08
}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 15:00:35', 'Open': 959.09, 'High': 1923.41, '
Low': 1239.3, 'Close': 1162.18, 'Adj_Close': 522.03, 'Volume': 652753, 'CloseUSD': 1099.2
5}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 15:00:38', 'Open': 367.59, 'High': 1591.27, '
Low': 557.44, 'Close': 869.94, 'Adj_Close': 817.87, 'Volume': 927058, 'CloseUSD': 1089.65
}
Record b'AAPL' successfully produced to stocks_analysis [0]
```

```
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 15:00:41', 'Open': 679.83, 'High': 1818.49, '
Low': 293.41, 'Close': 1306.6, 'Adj_Close': 485.1, 'Volume': 985064, 'CloseUSD': 1163.49}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'GOOG', 'Date': '2024-12-10 15:00:44', 'Open': 923.81, 'High': 1972.61, '
Low': 744.83, 'Close': 475.95, 'Adj_Close': 409.14, 'Volume': 193959, 'CloseUSD': 641.87}
Record b'AMZN' successfully produced to stocks_analysis [4]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 15:00:47', 'Open': 1238.7, 'High': 1935.77, '
Low': 335.33, 'Close': 686.87, 'Adj_Close': 440.04, 'Volume': 458112, 'CloseUSD': 1050.25
}
Record b'GOOG' successfully produced to stocks_analysis [2]
Sent: {'Index': 'AAPL', 'Date': '2024-12-10 15:00:50', 'Open': 1287.63, 'High': 1732.77,
'Low': 425.41, 'Close': 548.5, 'Adj_Close': 1374.88, 'Volume': 239571, 'CloseUSD': 271.87
}
Record b'MSFT' successfully produced to stocks_analysis [1]
Sent: {'Index': 'MSFT', 'Date': '2024-12-10 15:00:53', 'Open': 238.61, 'High': 1696.47, '
Low': 1193.54, 'Close': 605.38, 'Adj_Close': 648.8, 'Volume': 996433, 'CloseUSD': 1451.99
}
Record b'AAPL' successfully produced to stocks_analysis [0]
Sent: {'Index': 'AMZN', 'Date': '2024-12-10 15:00:56', 'Open': 993.63, 'High': 1812.48, '
Low': 877.89, 'Close': 741.92, 'Adj_Close': 1095.17, 'Volume': 970456, 'CloseUSD': 432.01
}
Record b'MSFT' successfully produced to stocks_analysis [1]
```

In [0]:

```
! pip install confluent_kafka
```

```
Collecting confluent_kafka
  Using cached confluent_kafka-2.6.1-cp39-cp39-manylinux_2_28_x86_64.whl (3.9 MB)
Installing collected packages: confluent-kafka
Successfully installed confluent-kafka-2.6.1
WARNING: You are using pip version 21.2.4; however, version 24.3.1 is available.
You should consider upgrading via the '/local_disk0/.ephemeral_nfs/envs/pythonEnv-f4388dc
5-5493-4819-b9bf-dd819783a9a9/bin/python -m pip install --upgrade pip' command.
```

In [0]:

```
from confluent_kafka import Consumer
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, FloatType, TimestampT
ype, IntegerType
from datetime import datetime
import json
import time

# Initialize a SparkSession (Databricks usually initializes this automatically)
spark_session = SparkSession.builder.getOrCreate()

# Configuration settings for the Kafka consumer
kafka_settings = {

}

# Create Kafka consumer instance
kafka_consumer = Consumer(kafka_settings)

# Specify the Kafka topic to listen to
topic_name = 'stocks_analysis'
kafka_consumer.subscribe([topic_name])

# Path to the Delta table (replace with actual location if needed)
delta_table_location = "/mnt/delta/stockprices"

# Schema definition for incoming data
data_schema = StructType([
    StructField("Index", StringType(), True),
    StructField("Date", StringType(), True),
    StructField("Open", FloatType(), True),
    StructField("High", FloatType(), True),
    StructField("Low", FloatType(), True),
    StructField("Close", FloatType(), True),
    StructField("Adj_Close", FloatType(), True),
    StructField("Volume", IntegerType(), True),
    StructField("CloseUSD", FloatType(), True)
])

# Function to handle consuming messages and processing them
def process_kafka_messages():
    try:
        while True:
            message = kafka_consumer.poll(1.0)  # Poll for new messages

            if message is not None and not message.error():
                try:
                    # Decode key and value safely and log for debugging
                    message_value = message.value().decode("utf-8") if message.value() e
lse "{}"
                    print(f"Decoded Kafka message: {message_value}")  # Debugging: Log t
he decoded message

                    # Parse message value as JSON
                    try:
```

```
                        record = json.loads(message_value)
                    except json.JSONDecodeError as e:
                        print(f"Error decoding JSON: {e}")
                        continue

                    # Extract fields from the record with defaults to ensure robustness
                    index = record.get("Index", "UnknownIndex")  # Correct field name
                    date = record.get("Date", "UnknownDate")
                    open_price = float(record.get("Open", 0.0))
                    high_price = float(record.get("High", 0.0))
                    low_price = float(record.get("Low", 0.0))
                    close_price = float(record.get("Close", 0.0))
                    adj_close = float(record.get("Adj_Close", 0.0))
                    volume = int(record.get("Volume", 0))
                    close_usd = float(record.get("CloseUSD", 0.0))

                    # Create a DataFrame for the single record
                    data_frame = spark_session.createDataFrame(
                        [(index, date, open_price, high_price, low_price, close_price, a
dj_close, volume, close_usd)],
                        schema=data_schema
                    )

                    # Display the DataFrame in Databricks (optional for debugging/valida
tion)
                    data_frame.show()

                    # Write the data to a Delta table in append mode
                    try:
                        data_frame.write.format("delta").mode("append").save(delta_table
_location)
                        print("Data written to Delta table successfully.")
                    except Exception as e:
                        print(f"Error writing to Delta table: {e}")

                except Exception as e:
                    print(f"Error processing Kafka message: {e}")
            elif message is not None and message.error():
                print(f"Kafka error: {message.error()}")

            # Small delay for real-time processing simulation
            time.sleep(1)

    except KeyboardInterrupt:
        print("Stopping Kafka consumer...")
    finally:
        kafka_consumer.close()
        print("Kafka consumer closed.")

# Start consuming messages
process_kafka_messages()
```

```
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:07:52", "Open": 670.11, "
High": 1531.45, "Low": 1474.45, "Close": 1161.91, "Adj_Close": 1301.16, "Volume": 905182,
"CloseUSD": 937.66}
+-----+-------------------+------+-------+-------+-------+---------+------+--------+
|Index|               Date|  Open|   High|    Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-------+-------+---------+------+--------+
| MSFT|2024-12-09 20:07:52|670.11|1531.45|1474.45|1161.91|  1301.16|905182|  937.66|
+-----+-------------------+------+-------+-------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:07:58", "Open": 266.08, "
High": 1554.61, "Low": 666.31, "Close": 1450.58, "Adj_Close": 1178.33, "Volume": 34473, "
CloseUSD": 335.35}
+-----+-------------------+------+-------+------+-------+---------+------+--------+
|Index|               Date|  Open|   High|   Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+-------+---------+------+--------+
| MSFT|2024-12-09 20:07:58|266.08|1554.61|666.31|1450.58|  1178.33| 34473|  335.35|
+-----+-------------------+------+-------+------+-------+---------+------+--------+

Data written to Delta table successfully.
```

Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:08:01", "Open": 1451.37, "High": 1775.97, "Low": 1364.54, "Close": 240.07, "Adj_Close": 1331.04, "Volume": 490952, "CloseUSD": 822.78}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:08:01	1451.37	1775.97	1364.54	240.07	1331.04	490952	822.78

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:08:16", "Open": 115.28, "High": 1954.86, "Low": 1072.95, "Close": 160.42, "Adj_Close": 1485.68, "Volume": 840250, "CloseUSD": 167.44}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:08:16	115.28	1954.86	1072.95	160.42	1485.68	840250	167.44

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:08:58", "Open": 589.09, "High": 1604.68, "Low": 250.49, "Close": 1318.04, "Adj_Close": 1496.32, "Volume": 570328, "CloseUSD": 674.06}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:08:58	589.09	1604.68	250.49	1318.04	1496.32	570328	674.06

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:09:13", "Open": 327.41, "High": 1535.36, "Low": 492.43, "Close": 1151.4, "Adj_Close": 845.0, "Volume": 824999, "CloseUSD": 1202.83}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:09:13	327.41	1535.36	492.43	1151.4	845.0	824999	1202.83

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:09:34", "Open": 435.29, "High": 1693.64, "Low": 1319.03, "Close": 311.72, "Adj_Close": 1000.17, "Volume": 992435, "CloseUSD": 219.0}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:09:34	435.29	1693.64	1319.03	311.72	1000.17	992435	219.0

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:10:01", "Open": 501.73, "High": 1576.32, "Low": 723.39, "Close": 832.61, "Adj_Close": 1247.62, "Volume": 697334, "CloseUSD": 1002.39}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:10:01	501.73	1576.32	723.39	832.61	1247.62	697334	1002.39

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:10:07", "Open": 1010.89, "High": 1588.39, "Low": 62.74, "Close": 1387.18, "Adj_Close": 969.62, "Volume": 342181, "CloseUSD": 1172.92}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:10:07	1010.89	1588.39	62.74	1387.18	969.62	342181	1172.92

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:10:25", "Open": 800.03, "High": 1797.89, "Low": 1125.67, "Close": 959.54, "Adj_Close": 331.55, "Volume": 243368,

CloseUSD": 330.38}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:10:25	800.03	1797.89	1125.67	959.54	331.55	243368	330.38

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:10:46", "Open": 1334.52, "High": 1694.31, "Low": 1262.97, "Close": 1457.83, "Adj_Close": 750.6, "Volume": 487102, "CloseUSD": 435.19}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:10:46	1334.52	1694.31	1262.97	1457.83	750.6	487102	435.19

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:10:49", "Open": 812.67, "High": 1846.09, "Low": 1473.71, "Close": 936.93, "Adj_Close": 686.73, "Volume": 324044, "CloseUSD": 710.95}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:10:49	812.67	1846.09	1473.71	936.93	686.73	324044	710.95

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:10:58", "Open": 1132.3, "High": 1593.84, "Low": 249.16, "Close": 1014.31, "Adj_Close": 1235.81, "Volume": 128417, "CloseUSD": 306.02}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:10:58	1132.3	1593.84	249.16	1014.31	1235.81	128417	306.02

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:11:31", "Open": 1006.68, "High": 1573.87, "Low": 670.84, "Close": 1004.18, "Adj_Close": 650.63, "Volume": 666862, "CloseUSD": 565.09}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:11:31	1006.68	1573.87	670.84	1004.18	650.63	666862	565.09

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:11:46", "Open": 847.93, "High": 1550.51, "Low": 593.82, "Close": 1170.11, "Adj_Close": 480.77, "Volume": 402015, "CloseUSD": 935.52}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:11:46	847.93	1550.51	593.82	1170.11	480.77	402015	935.52

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:11:49", "Open": 797.49, "High": 1825.05, "Low": 992.51, "Close": 429.8, "Adj_Close": 209.38, "Volume": 573168, "CloseUSD": 376.3}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:11:49	797.49	1825.05	992.51	429.8	209.38	573168	376.3

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:12:19", "Open": 1260.8, "High": 1826.48, "Low": 203.86, "Close": 416.64, "Adj_Close": 1216.79, "Volume": 820950, "CloseUSD": 1443.37}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD

```
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| MSFT|2024-12-09 20:12:19|1260.8|1826.48|203.86|416.64|  1216.79|820950| 1443.37|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:12:25", "Open": 1357.81,
"High": 1839.42, "Low": 1178.55, "Close": 1353.17, "Adj_Close": 1176.06, "Volume": 610608
, "CloseUSD": 607.37}
+-----+-------------------+------+-------+-------+-------+---------+------+--------+
|Index|               Date|  Open|   High|    Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-------+-------+---------+------+--------+
| MSFT|2024-12-09 20:12:25|1357.81|1839.42|1178.55|1353.17|  1176.06|610608|  607.37|
+-----+-------------------+------+-------+-------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:12:52", "Open": 625.85, "
High": 1892.24, "Low": 986.28, "Close": 728.39, "Adj_Close": 347.28, "Volume": 301666, "C
loseUSD": 700.25}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| MSFT|2024-12-09 20:12:52|625.85|1892.24|986.28|728.39|   347.28|301666|  700.25|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:12:55", "Open": 123.97, "
High": 1589.11, "Low": 350.53, "Close": 568.26, "Adj_Close": 229.89, "Volume": 335573, "C
loseUSD": 365.61}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| MSFT|2024-12-09 20:12:55|123.97|1589.11|350.53|568.26|   229.89|335573|  365.61|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:13:07", "Open": 721.7, "H
igh": 1837.91, "Low": 1269.77, "Close": 909.37, "Adj_Close": 1054.44, "Volume": 150338, "
CloseUSD": 998.95}
+-----+-------------------+-----+-------+-------+------+---------+------+--------+
|Index|               Date| Open|   High|    Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-----+-------+-------+------+---------+------+--------+
| MSFT|2024-12-09 20:13:07|721.7|1837.91|1269.77|909.37|  1054.44|150338|  998.95|
+-----+-------------------+-----+-------+-------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:13:19", "Open": 673.05, "
High": 1682.93, "Low": 622.97, "Close": 778.88, "Adj_Close": 1337.4, "Volume": 608961, "C
loseUSD": 509.41}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| MSFT|2024-12-09 20:13:19|673.05|1682.93|622.97|778.88|   1337.4|608961|  509.41|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:13:43", "Open": 1401.27,
"High": 1600.09, "Low": 1317.0, "Close": 267.4, "Adj_Close": 623.58, "Volume": 850215, "C
loseUSD": 1044.96}
+-----+-------------------+-------+-------+------+-----+---------+------+--------+
|Index|               Date|   Open|   High|   Low|Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-------+-------+------+-----+---------+------+--------+
| MSFT|2024-12-09 20:13:43|1401.27|1600.09|1317.0|267.4|   623.58|850215| 1044.96|
+-----+-------------------+-------+-------+------+-----+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:13:49", "Open": 452.92, "
High": 1684.22, "Low": 286.1, "Close": 316.82, "Adj_Close": 837.72, "Volume": 669199, "Cl
oseUSD": 188.43}
+-----+-------------------+------+-------+-----+------+---------+------+--------+
|Index|               Date|  Open|   High|  Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-----+------+---------+------+--------+
```

```
| MSFT|2024-12-09 20:13:49|452.92|1684.22|286.1|316.82|   837.72|669199|  188.43|
+-----+-------------------+------+-------+-----+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:13:58", "Open": 430.45, "
High": 1609.95, "Low": 925.06, "Close": 1042.3, "Adj_Close": 558.64, "Volume": 284919, "C
loseUSD": 533.85}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| MSFT|2024-12-09 20:13:58|430.45|1609.95|925.06|1042.3|   558.64|284919|  533.85|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:14:04", "Open": 1359.5, "
High": 1575.02, "Low": 409.89, "Close": 1373.35, "Adj_Close": 977.65, "Volume": 916236, "
CloseUSD": 1236.66}
+-----+-------------------+------+-------+------+-------+---------+------+--------+
|Index|               Date|  Open|   High|   Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+-------+---------+------+--------+
| MSFT|2024-12-09 20:14:04|1359.5|1575.02|409.89|1373.35|   977.65|916236| 1236.66|
+-----+-------------------+------+-------+------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:14:10", "Open": 950.77, "
High": 1686.82, "Low": 223.83, "Close": 861.51, "Adj_Close": 550.45, "Volume": 507095, "C
loseUSD": 188.41}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| MSFT|2024-12-09 20:14:10|950.77|1686.82|223.83|861.51|   550.45|507095|  188.41|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:14:40", "Open": 701.56, "
High": 1557.58, "Low": 637.03, "Close": 133.7, "Adj_Close": 423.73, "Volume": 307145, "Cl
oseUSD": 108.47}
+-----+-------------------+------+-------+------+-----+---------+------+--------+
|Index|               Date|  Open|   High|   Low|Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+-----+---------+------+--------+
| MSFT|2024-12-09 20:14:40|701.56|1557.58|637.03|133.7|   423.73|307145|  108.47|
+-----+-------------------+------+-------+------+-----+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:14:43", "Open": 1363.61,
"High": 1537.77, "Low": 1471.43, "Close": 1385.86, "Adj_Close": 1177.0, "Volume": 940107,
"CloseUSD": 557.25}
+-----+-------------------+-------+-------+-------+-------+---------+------+--------+
|Index|               Date|   Open|   High|    Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-------+-------+-------+-------+---------+------+--------+
| MSFT|2024-12-09 20:14:43|1363.61|1537.77|1471.43|1385.86|   1177.0|940107|  557.25|
+-----+-------------------+-------+-------+-------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:15:04", "Open": 752.41, "
High": 1779.32, "Low": 1143.05, "Close": 344.53, "Adj_Close": 1255.97, "Volume": 417041,
"CloseUSD": 1335.79}
+-----+-------------------+------+-------+-------+------+---------+------+--------+
|Index|               Date|  Open|   High|    Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-------+------+---------+------+--------+
| MSFT|2024-12-09 20:15:04|752.41|1779.32|1143.05|344.53|  1255.97|417041| 1335.79|
+-----+-------------------+------+-------+-------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-09 20:15:07", "Open": 1263.27,
"High": 1623.92, "Low": 156.76, "Close": 137.53, "Adj_Close": 162.73, "Volume": 393624, "
CloseUSD": 716.83}
+-----+-------------------+-------+-------+------+------+---------+------+--------+
|Index|               Date|   Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-------+-------+------+------+---------+------+--------+
| MSFT|2024-12-09 20:15:07|1263.27|1623.92|156.76|137.53|   162.73|393624|  716.83|
+-----+-------------------+-------+-------+------+------+---------+------+--------+
```

```
Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-10 03:22:51", "Open": 597.38, "
High": 1648.93, "Low": 486.12, "Close": 1084.39, "Adj_Close": 502.06, "Volume": 788759, "
CloseUSD": 774.99}
+-----+-------------------+------+-------+------+-------+---------+------+--------+
|Index|               Date|  Open|   High|   Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+-------+---------+------+--------+
| MSFT|2024-12-10 03:22:51|597.38|1648.93|486.12|1084.39|   502.06|788759|  774.99|
+-----+-------------------+------+-------+------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-10 03:23:03", "Open": 1178.19,
"High": 1891.87, "Low": 709.3, "Close": 213.99, "Adj_Close": 1128.98, "Volume": 588128, "
CloseUSD": 1127.8}
+-----+-------------------+-------+-------+-----+------+---------+------+--------+
|Index|               Date|   Open|   High|  Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-------+-------+-----+------+---------+------+--------+
| MSFT|2024-12-10 03:23:03|1178.19|1891.87|709.3|213.99|  1128.98|588128|  1127.8|
+-----+-------------------+-------+-------+-----+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-10 03:23:38", "Open": 1279.24,
"High": 1691.36, "Low": 1188.81, "Close": 295.47, "Adj_Close": 1169.87, "Volume": 521259,
"CloseUSD": 468.44}
+-----+-------------------+-------+-------+-------+------+---------+------+--------+
|Index|               Date|   Open|   High|    Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-------+-------+-------+------+---------+------+--------+
| MSFT|2024-12-10 03:23:38|1279.24|1691.36|1188.81|295.47|  1169.87|521259|  468.44|
+-----+-------------------+-------+-------+-------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-10 03:23:50", "Open": 498.64, "
High": 1767.68, "Low": 749.21, "Close": 1146.36, "Adj_Close": 1425.16, "Volume": 347728,
"CloseUSD": 694.77}
+-----+-------------------+------+-------+------+-------+---------+------+--------+
|Index|               Date|  Open|   High|   Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+-------+---------+------+--------+
| MSFT|2024-12-10 03:23:50|498.64|1767.68|749.21|1146.36|  1425.16|347728|  694.77|
+-----+-------------------+------+-------+------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-10 03:24:05", "Open": 1496.31,
"High": 1864.14, "Low": 168.51, "Close": 923.21, "Adj_Close": 300.58, "Volume": 320950, "
CloseUSD": 1487.96}
+-----+-------------------+-------+-------+------+------+---------+------+--------+
|Index|               Date|   Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-------+-------+------+------+---------+------+--------+
| MSFT|2024-12-10 03:24:05|1496.31|1864.14|168.51|923.21|   300.58|320950| 1487.96|
+-----+-------------------+-------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-10 03:24:08", "Open": 741.01, "
High": 1746.59, "Low": 935.14, "Close": 372.12, "Adj_Close": 985.26, "Volume": 994475, "C
loseUSD": 1020.35}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| MSFT|2024-12-10 03:24:08|741.01|1746.59|935.14|372.12|   985.26|994475| 1020.35|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "MSFT", "Date": "2024-12-10 03:24:14", "Open": 1049.5, "
High": 1806.47, "Low": 263.41, "Close": 1344.42, "Adj_Close": 310.12, "Volume": 925642, "
CloseUSD": 1167.36}
+-----+-------------------+------+-------+------+-------+---------+------+--------+
|Index|               Date|  Open|   High|   Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+-------+---------+------+--------+
| MSFT|2024-12-10 03:24:14|1049.5|1806.47|263.41|1344.42|   310.12|925642| 1167.36|
+-----+-------------------+------+-------+------+-------+---------+------+-

*** WARNING: max output size exceeded, skipping output. ***
```

table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:26:29", "Open": 1155.09, "High": 1639.23, "Low": 799.97, "Close": 701.8, "Adj_Close": 1021.01, "Volume": 589226, "CloseUSD": 1288.97}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
GOOG	2024-12-10 03:26:29	1155.09	1639.23	799.97	701.8	1021.01	589226	1288.97

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:26:38", "Open": 1305.69, "High": 1965.26, "Low": 958.19, "Close": 908.54, "Adj_Close": 894.68, "Volume": 317087, "CloseUSD": 504.96}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
GOOG	2024-12-10 03:26:38	1305.69	1965.26	958.19	908.54	894.68	317087	504.96

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:26:44", "Open": 360.94, "High": 1624.01, "Low": 718.6, "Close": 539.9, "Adj_Close": 254.86, "Volume": 613974, "CloseUSD": 976.17}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
GOOG	2024-12-10 03:26:44	360.94	1624.01	718.6	539.9	254.86	613974	976.17

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:27:02", "Open": 898.47, "High": 1843.72, "Low": 549.55, "Close": 893.05, "Adj_Close": 563.78, "Volume": 60193, "CloseUSD": 850.5}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
GOOG	2024-12-10 03:27:02	898.47	1843.72	549.55	893.05	563.78	60193	850.5

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:27:05", "Open": 1004.17, "High": 1572.84, "Low": 1350.75, "Close": 823.85, "Adj_Close": 180.82, "Volume": 676059, "CloseUSD": 1460.77}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
GOOG	2024-12-10 03:27:05	1004.17	1572.84	1350.75	823.85	180.82	676059	1460.77

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:27:17", "Open": 1356.16, "High": 1766.49, "Low": 1426.55, "Close": 953.56, "Adj_Close": 467.35, "Volume": 248927, "CloseUSD": 840.25}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
GOOG	2024-12-10 03:27:17	1356.16	1766.49	1426.55	953.56	467.35	248927	840.25

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:27:23", "Open": 461.98, "High": 1881.51, "Low": 137.77, "Close": 678.77, "Adj_Close": 557.22, "Volume": 111419, "CloseUSD": 802.46}

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
GOOG	2024-12-10 03:27:23	461.98	1881.51	137.77	678.77	557.22	111419	802.46

Data written to Delta table successfully.

```
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:27:38", "Open": 287.97, "
High": 1968.52, "Low": 763.58, "Close": 690.26, "Adj_Close": 652.63, "Volume": 534412, "C
loseUSD": 1310.17}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| GOOG|2024-12-10 03:27:38|287.97|1968.52|763.58|690.26|   652.63|534412| 1310.17|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:27:41", "Open": 150.48, "
High": 1587.39, "Low": 1251.42, "Close": 1061.61, "Adj_Close": 108.23, "Volume": 353936,
"CloseUSD": 763.62}
+-----+-------------------+------+-------+-------+-------+---------+------+--------+
|Index|               Date|  Open|   High|    Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-------+-------+---------+------+--------+
| GOOG|2024-12-10 03:27:41|150.48|1587.39|1251.42|1061.61|   108.23|353936|  763.62|
+-----+-------------------+------+-------+-------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:27:47", "Open": 165.1, "H
igh": 1772.75, "Low": 483.93, "Close": 558.64, "Adj_Close": 644.86, "Volume": 96264, "Clo
seUSD": 1275.65}
+-----+-------------------+-----+-------+------+------+---------+------+--------+
|Index|               Date| Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-----+-------+------+------+---------+------+--------+
| GOOG|2024-12-10 03:27:47|165.1|1772.75|483.93|558.64|   644.86| 96264| 1275.65|
+-----+-------------------+-----+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:27:50", "Open": 1421.14,
"High": 1602.75, "Low": 961.02, "Close": 238.34, "Adj_Close": 859.95, "Volume": 789492, "
CloseUSD": 1009.95}
+-----+-------------------+-------+-------+------+------+---------+------+--------+
|Index|               Date|   Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-------+-------+------+------+---------+------+--------+
| GOOG|2024-12-10 03:27:50|1421.14|1602.75|961.02|238.34|   859.95|789492| 1009.95|
+-----+-------------------+-------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:27:53", "Open": 445.27, "
High": 1760.8, "Low": 563.5, "Close": 1413.06, "Adj_Close": 416.2, "Volume": 22083, "Clos
eUSD": 694.03}
+-----+-------------------+------+------+-----+-------+---------+------+--------+
|Index|               Date|  Open|  High|  Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+------+-----+-------+---------+------+--------+
| GOOG|2024-12-10 03:27:53|445.27|1760.8|563.5|1413.06|    416.2| 22083|  694.03|
+-----+-------------------+------+------+-----+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:28:05", "Open": 529.88, "
High": 1642.32, "Low": 1378.52, "Close": 620.6, "Adj_Close": 1418.9, "Volume": 221441, "C
loseUSD": 895.04}
+-----+-------------------+------+-------+-------+-----+---------+------+--------+
|Index|               Date|  Open|   High|    Low|Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-------+-----+---------+------+--------+
| GOOG|2024-12-10 03:28:05|529.88|1642.32|1378.52|620.6|   1418.9|221441|  895.04|
+-----+-------------------+------+-------+-------+-----+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:28:17", "Open": 212.67, "
High": 1557.22, "Low": 218.56, "Close": 871.24, "Adj_Close": 314.94, "Volume": 658769, "C
loseUSD": 572.25}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| GOOG|2024-12-10 03:28:17|212.67|1557.22|218.56|871.24|   314.94|658769|  572.25|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:28:32", "Open": 592.16, "
High": 1590.05, "Low": 262.28, "Close": 335.82, "Adj_Close": 516.04, "Volume": 47958, "Cl
```

oseUSD": 1455.53}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index| Date| Open| High| Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| GOOG|2024-12-10 03:28:32|592.16|1590.05|262.28|335.82| 516.04| 47958| 1455.53|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:28:38", "Open": 157.49, "
High": 1937.67, "Low": 416.57, "Close": 1053.79, "Adj_Close": 1427.84, "Volume": 759265,
"CloseUSD": 1418.28}
+-----+-------------------+------+-------+------+-------+---------+------+--------+
|Index| Date| Open| High| Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+-------+---------+------+--------+
| GOOG|2024-12-10 03:28:38|157.49|1937.67|416.57|1053.79| 1427.84|759265| 1418.28|
+-----+-------------------+------+-------+------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:28:50", "Open": 719.92, "
High": 1812.53, "Low": 904.6, "Close": 1311.72, "Adj_Close": 1153.44, "Volume": 769648, "
CloseUSD": 213.67}
+-----+-------------------+------+-------+-----+-------+---------+------+--------+
|Index| Date| Open| High| Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-----+-------+---------+------+--------+
| GOOG|2024-12-10 03:28:50|719.92|1812.53|904.6|1311.72| 1153.44|769648| 213.67|
+-----+-------------------+------+-------+-----+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:29:32", "Open": 236.72, "
High": 1818.63, "Low": 437.83, "Close": 512.39, "Adj_Close": 990.13, "Volume": 757416, "C
loseUSD": 858.89}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index| Date| Open| High| Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| GOOG|2024-12-10 03:29:32|236.72|1818.63|437.83|512.39| 990.13|757416| 858.89|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:29:38", "Open": 538.64, "
High": 1613.66, "Low": 1125.49, "Close": 366.56, "Adj_Close": 1193.74, "Volume": 763490,
"CloseUSD": 623.54}
+-----+-------------------+------+-------+-------+------+---------+------+--------+
|Index| Date| Open| High| Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-------+------+---------+------+--------+
| GOOG|2024-12-10 03:29:38|538.64|1613.66|1125.49|366.56| 1193.74|763490| 623.54|
+-----+-------------------+------+-------+-------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:29:47", "Open": 1098.37,
"High": 1958.76, "Low": 473.69, "Close": 1254.61, "Adj_Close": 1149.84, "Volume": 313177,
"CloseUSD": 918.19}
+-----+-------------------+-------+-------+------+-------+---------+------+--------+
|Index| Date| Open| High| Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-------+-------+------+-------+---------+------+--------+
| GOOG|2024-12-10 03:29:47|1098.37|1958.76|473.69|1254.61| 1149.84|313177| 918.19|
+-----+-------------------+-------+-------+------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:30:14", "Open": 765.57, "
High": 1907.63, "Low": 600.22, "Close": 687.34, "Adj_Close": 440.5, "Volume": 459916, "Cl
oseUSD": 1260.4}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index| Date| Open| High| Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| GOOG|2024-12-10 03:30:14|765.57|1907.63|600.22|687.34| 440.5|459916| 1260.4|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:30:17", "Open": 1343.1, "
High": 1950.31, "Low": 572.56, "Close": 217.45, "Adj_Close": 153.09, "Volume": 986734, "C
loseUSD": 515.45}
+-----+-------------------+------+-------+------+------+---------+------+--------+

```
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| GOOG|2024-12-10 03:30:17|1343.1|1950.31|572.56|217.45|   153.09|986734|  515.45|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:30:35", "Open": 1151.5, "
High": 1889.71, "Low": 821.15, "Close": 433.84, "Adj_Close": 366.63, "Volume": 957608, "C
loseUSD": 1387.39}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| GOOG|2024-12-10 03:30:35|1151.5|1889.71|821.15|433.84|   366.63|957608| 1387.39|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:30:44", "Open": 1385.65,
"High": 1996.02, "Low": 277.3, "Close": 1451.44, "Adj_Close": 914.47, "Volume": 312625, "
CloseUSD": 282.33}
+-----+-------------------+------+-------+-----+-------+---------+------+--------+
|Index|               Date|  Open|   High|  Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-----+-------+---------+------+--------+
| GOOG|2024-12-10 03:30:44|1385.65|1996.02|277.3|1451.44|   914.47|312625|  282.33|
+-----+-------------------+------+-------+-----+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:30:56", "Open": 1190.59,
"High": 1743.67, "Low": 187.85, "Close": 342.17, "Adj_Close": 1346.54, "Volume": 94358, "
CloseUSD": 1006.76}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| GOOG|2024-12-10 03:30:56|1190.59|1743.67|187.85|342.17|  1346.54| 94358| 1006.76|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:31:08", "Open": 139.93, "
High": 1716.81, "Low": 1452.38, "Close": 1150.39, "Adj_Close": 1187.87, "Volume": 534839,
"CloseUSD": 510.95}
+-----+-------------------+------+-------+-------+-------+---------+------+--------+
|Index|               Date|  Open|   High|    Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-------+-------+---------+------+--------+
| GOOG|2024-12-10 03:31:08|139.93|1716.81|1452.38|1150.39|  1187.87|534839|  510.95|
+-----+-------------------+------+-------+-------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:31:11", "Open": 1302.21,
"High": 1687.75, "Low": 208.5, "Close": 1188.77, "Adj_Close": 557.94, "Volume": 658368, "
CloseUSD": 1277.74}
+-----+-------------------+------+-------+-----+-------+---------+------+--------+
|Index|               Date|  Open|   High|  Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-----+-------+---------+------+--------+
| GOOG|2024-12-10 03:31:11|1302.21|1687.75|208.5|1188.77|   557.94|658368| 1277.74|
+-----+-------------------+------+-------+-----+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:31:14", "Open": 1055.94,
"High": 1941.72, "Low": 304.49, "Close": 1408.53, "Adj_Close": 467.19, "Volume": 828133,
"CloseUSD": 1384.31}
+-----+-------------------+------+-------+------+-------+---------+------+--------+
|Index|               Date|  Open|   High|   Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+-------+---------+------+--------+
| GOOG|2024-12-10 03:31:14|1055.94|1941.72|304.49|1408.53|   467.19|828133| 1384.31|
+-----+-------------------+------+-------+------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:31:47", "Open": 205.64, "
High": 1609.31, "Low": 981.46, "Close": 233.64, "Adj_Close": 408.41, "Volume": 350959, "C
loseUSD": 727.1}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
```

```
| GOOG|2024-12-10 03:31:47|205.64|1609.31|981.46|233.64|   408.41|350959|   727.1|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:31:50", "Open": 924.7, "H
igh": 1532.53, "Low": 1471.17, "Close": 292.7, "Adj_Close": 114.19, "Volume": 445045, "Cl
oseUSD": 1074.86}
+-----+-------------------+-----+-------+-------+-----+---------+------+--------+
|Index|               Date| Open|   High|    Low|Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-----+-------+-------+-----+---------+------+--------+
| GOOG|2024-12-10 03:31:50|924.7|1532.53|1471.17|292.7|   114.19|445045| 1074.86|
+-----+-------------------+-----+-------+-------+-----+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:31:59", "Open": 1093.26,
"High": 1751.8, "Low": 1494.85, "Close": 1445.33, "Adj_Close": 249.93, "Volume": 506794,
"CloseUSD": 682.76}
+-----+-------------------+-------+------+-------+-------+---------+------+--------+
|Index|               Date|   Open|  High|    Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-------+------+-------+-------+---------+------+--------+
| GOOG|2024-12-10 03:31:59|1093.26|1751.8|1494.85|1445.33|   249.93|506794|  682.76|
+-----+-------------------+-------+------+-------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:32:02", "Open": 508.33, "
High": 1838.36, "Low": 584.6, "Close": 806.33, "Adj_Close": 619.7, "Volume": 84095, "Clos
eUSD": 476.98}
+-----+-------------------+------+-------+-----+------+---------+------+--------+
|Index|               Date|  Open|   High|  Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-----+------+---------+------+--------+
| GOOG|2024-12-10 03:32:02|508.33|1838.36|584.6|806.33|    619.7| 84095|  476.98|
+-----+-------------------+------+-------+-----+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:32:08", "Open": 730.66, "
High": 1784.85, "Low": 695.95, "Close": 542.44, "Adj_Close": 1333.04, "Volume": 716909, "
CloseUSD": 748.47}
+-----+-------------------+------+-------+------+------+---------+------+--------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+--------+
| GOOG|2024-12-10 03:32:08|730.66|1784.85|695.95|542.44|  1333.04|716909|  748.47|
+-----+-------------------+------+-------+------+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:32:38", "Open": 531.31, "
High": 1802.27, "Low": 1385.84, "Close": 1325.44, "Adj_Close": 683.65, "Volume": 55016, "
CloseUSD": 414.59}
+-----+-------------------+------+-------+-------+-------+---------+------+--------+
|Index|               Date|  Open|   High|    Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+-------+-------+---------+------+--------+
| GOOG|2024-12-10 03:32:38|531.31|1802.27|1385.84|1325.44|   683.65| 55016|  414.59|
+-----+-------------------+------+-------+-------+-------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:32:44", "Open": 1215.13,
"High": 1745.65, "Low": 51.97, "Close": 376.95, "Adj_Close": 1366.33, "Volume": 370569, "
CloseUSD": 616.2}
+-----+-------------------+-------+-------+-----+------+---------+------+--------+
|Index|               Date|   Open|   High|  Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-------+-------+-----+------+---------+------+--------+
| GOOG|2024-12-10 03:32:44|1215.13|1745.65|51.97|376.95|  1366.33|370569|   616.2|
+-----+-------------------+-------+-------+-----+------+---------+------+--------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:32:47", "Open": 1076.91,
"High": 1747.65, "Low": 781.56, "Close": 1035.5, "Adj_Close": 923.3, "Volume": 955762, "C
loseUSD": 166.01}
+-----+-------------------+-------+-------+------+------+---------+------+--------+
|Index|               Date|   Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+-------+-------+------+------+---------+------+--------+
| GOOG|2024-12-10 03:32:47|1076.91|1747.65|781.56|1035.5|    923.3|955762|  166.01|
+-----+-------------------+-------+-------+------+------+---------+------+--------+
```

```
Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:32:53", "Open": 1215.37,
"High": 1587.96, "Low": 157.97, "Close": 1293.85, "Adj_Close": 115.76, "Volume": 929528,
"CloseUSD": 1427.03}
+-----+-------------------+------+-------+------+-------+---------+------+---------+
|Index|               Date|  Open|   High|   Low|  Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+-------+---------+------+---------+
| GOOG|2024-12-10 03:32:53|1215.37|1587.96|157.97|1293.85|   115.76|929528| 1427.03|
+-----+-------------------+------+-------+------+-------+---------+------+---------+

Data written to Delta table successfully.
Decoded Kafka message: {"Index": "GOOG", "Date": "2024-12-10 03:32:59", "Open": 705.67, "
High": 1770.76, "Low": 631.72, "Close": 547.65, "Adj_Close": 480.8, "Volume": 660917, "Cl
oseUSD": 1481.7}
+-----+-------------------+------+-------+------+------+---------+------+---------+
|Index|               Date|  Open|   High|   Low| Close|Adj_Close|Volume|CloseUSD|
+-----+-------------------+------+-------+------+------+---------+------+---------+
| GOOG|2024-12-10 03:32:59|705.67|1770.76|631.72|547.65|    480.8|660917|  1481.7|
+-----+-------------------+------+-------+------+------+---------+------+---------+

Data written to Delta table successfully.
```

In [0]:

```
from confluent_kafka import Consumer
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, FloatType, TimestampT
ype, IntegerType
from datetime import datetime
import json
import time

# Initialize a SparkSession (Databricks usually initializes this automatically)
spark_session = SparkSession.builder.getOrCreate()

# Configuration settings for the Kafka consumer
kafka_settings = {
    'bootstrap.servers': 'pkc-619z3.us-east1.gcp.confluent.cloud:9092',
    'security.protocol': 'SASL_SSL',
    'sasl.mechanisms': 'PLAIN',
    'sasl.username': 'MH3WSUES4XVY3S4Q',
    'sasl.password': 'GP4kK/kqeE79vbjJ+2r+0y3+wxL5CCoQ5WPvUljpQbT9u1WkP7UcdJpxP6AaG5xz',
    'session.timeout.ms': 45000,
    'group.id': 'stock-price-consumer-group',
    'auto.offset.reset': 'earliest'
}

# Create Kafka consumer instance
kafka_consumer = Consumer(kafka_settings)

# Specify the Kafka topic to listen to
topic_name = 'stocks_analysis'
kafka_consumer.subscribe([topic_name])

# Path to the Delta table (replace with actual location if needed)
delta_table_location = "/mnt/delta/stockprices"

# Schema definition for incoming data
data_schema = StructType([
    StructField("Index", StringType(), True),
    StructField("Date", StringType(), True),
    StructField("Open", FloatType(), True),
    StructField("High", FloatType(), True),
    StructField("Low", FloatType(), True),
    StructField("Close", FloatType(), True),
    StructField("Adj_Close", FloatType(), True),
    StructField("Volume", IntegerType(), True),
    StructField("CloseUSD", FloatType(), True)
])
```

```
# Function to handle consuming messages and processing them
def process_kafka_messages():
    try:
        while True:
            message = kafka_consumer.poll(1.0)  # Poll for new messages

            if message is not None and not message.error():
                try:
                    # Decode key and value safely and log for debugging
                    message_value = message.value().decode("utf-8") if message.value() e
lse "{}"
                    print(f"Decoded Kafka message: {message_value}")  # Debugging: Log t
he decoded message

                    # Parse message value as JSON
                    try:
                        record = json.loads(message_value)
                    except json.JSONDecodeError as e:
                        print(f"Error decoding JSON: {e}")
                        continue

                    # Extract fields from the record with defaults to ensure robustness
                    index = record.get("Index", "UnknownIndex")  # Correct field name
                    date = record.get("Date", "UnknownDate")
                    open_price = float(record.get("Open", 0.0))
                    high_price = float(record.get("High", 0.0))
                    low_price = float(record.get("Low", 0.0))
                    close_price = float(record.get("Close", 0.0))
                    adj_close = float(record.get("Adj_Close", 0.0))
                    volume = int(record.get("Volume", 0))
                    close_usd = float(record.get("CloseUSD", 0.0))

                    # Create a DataFrame for the single record
                    data_frame = spark_session.createDataFrame(
                        [(index, date, open_price, high_price, low_price, close_price, a
dj_close, volume, close_usd)],
                        schema=data_schema
                    )

                    # Display the DataFrame in Databricks (optional for debugging/valida
tion)
                    data_frame.show()

                    # Write the data to a Delta table in append mode
                    try:
                        data_frame.write.format("delta").mode("append").save(delta_table
_location)
                        print("Data written to Delta table successfully.")
                    except Exception as e:
                        print(f"Error writing to Delta table: {e}")

                except Exception as e:
                    print(f"Error processing Kafka message: {e}")
            elif message is not None and message.error():
                print(f"Kafka error: {message.error()}")

            # Small delay for real-time processing simulation
            time.sleep(1)

    except KeyboardInterrupt:
        print("Stopping Kafka consumer...")
    finally:
        kafka_consumer.close()
        print("Kafka consumer closed.")

# Start consuming messages
process_kafka_messages()
```

In [0]:

```
from pyspark.sql import SparkSession
from pyspark.sql.streaming import StreamingQueryListener

delta_table_location = "/mnt/delta/stockprices"

# Create a streaming DataFrame
streaming_df = (
    spark.readStream
    .format("delta")
    .load(delta_table_location)
)

display(streaming_df)
```

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09 20:07:52	670.11	1531.45	1474.45	1161.91	1301.16	905182	937.66
MSFT	2024-12-09 20:08:01	1451.37	1775.97	1364.54	240.07	1331.04	490952	822.78
MSFT	2024-12-09 20:08:58	589.09	1604.68	250.49	1318.04	1496.32	570328	674.06
MSFT	2024-12-09 20:09:34	435.29	1693.64	1319.03	311.72	1000.17	992435	219.0
MSFT	2024-12-09 20:10:01	501.73	1576.32	723.39	832.61	1247.62	697334	1002.39
MSFT	2024-12-09 20:10:07	1010.89	1588.39	62.74	1387.18	969.62	342181	1172.92
MSFT	2024-12-09 20:10:25	800.03	1797.89	1125.67	959.54	331.55	243368	330.38
MSFT	2024-12-09 20:10:46	1334.52	1694.31	1262.97	1457.83	750.6	487102	435.19
MSFT	2024-12-09 20:10:58	1132.3	1593.84	249.16	1014.31	1235.81	128417	306.02
MSFT	2024-12-09 20:11:31	1006.68	1573.87	670.84	1004.18	650.63	666862	565.09

Databricks visualization. Run in Databricks to view.

Databricks visualization. Run in Databricks to view.

In [0]:

```
# Define your stock symbols list
stock_symbols = ['MSFT', 'AMZN', 'AAPL', 'GOOG']

# Create a dropdown widget for selecting a stock symbol
dbutils.widgets.dropdown(
    name="dropdown_filter",
    defaultValue=stock_symbols[0],  # Default to the first stock symbol
    choices=stock_symbols,
    label="Select Stock Symbol"
)

# Retrieve the selected stock symbol from the dropdown widget
selected_stock = dbutils.widgets.get("dropdown_filter")

# Now you can use the selected stock symbol to filter your data
# Assuming your DataFrame is called streaming_df
filtered_df = streaming_df.filter(col("Index") == selected_stock)

# Show the filtered data
display(filtered_df)
```

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD
MSFT	2024-12-09T20:07:52.000+0000	670.11	1531.45	1474.45	1161.91	1301.16	905182	937.66
MSFT	2024-12-09T20:08:01.000+0000	1451.37	1775.97	1364.54	240.07	1331.04	490952	822.78
MSFT	2024-12-09T20:08:58.000+0000	589.09	1604.68	250.49	1318.04	1496.32	570328	674.06
MSFT	2024-12-09T20:09:34.000+0000	435.29	1693.64	1319.03	311.72	1000.17	992435	219.0

Index	Date	Open	High	Low	Close	Adj Close	Volume	CloseUSD
MSFT	2024-12-09T20:10:01.000+0000	501.73	1576.92	725.39	852.61	1247.62	697334	1002.39
MSFT	2024-12-09T20:10:07.000+0000	1010.89	1588.39	62.74	1387.18	969.62	342181	1172.92
MSFT	2024-12-09T20:10:25.000+0000	800.03	1797.89	1125.67	959.54	331.55	243368	330.38
MSFT	2024-12-09T20:10:46.000+0000	1334.52	1694.31	1262.97	1457.83	750.6	487102	435.19
MSFT	2024-12-09T20:10:58.000+0000	1132.3	1593.84	249.16	1014.31	1235.81	128417	306.02
MSFT	2024-12-09T20:11:31.000+0000	1006.68	1573.87	670.84	1004.18	650.63	666862	565.09

In [0]:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, avg, stddev, expr
from pyspark.sql.window import Window

windowed_df = streaming_df.groupBy(
    window(col("Date"), "1 minute"),
    col("index")
).agg(
    avg("CloseUSD").alias("moving_avg"),
    stddev("CloseUSD").alias("volatility")
)

display(windowed_df)
```

window	index	moving_avg	volatility
List(2024-12-10T03:31:00.000+0000, 2024-12-10T03:32:00.000+0000)	AMZN	468.7033386230469	174.13126964258421
List(2024-12-09T20:01:00.000+0000, 2024-12-09T20:02:00.000+0000)	AMZN	1110.2174987792969	186.09512990958868
List(2024-12-10T03:22:00.000+0000, 2024-12-10T03:23:00.000+0000)	MSFT	774.989990234375	null
List(2024-12-10T03:47:00.000+0000, 2024-12-10T03:48:00.000+0000)	AMZN	1003.1239868164063	358.4982814532724
List(2024-12-10T03:28:00.000+0000, 2024-12-10T03:29:00.000+0000)	AAPL	844.8485804966518	386.1638174917641
List(2024-12-09T20:10:00.000+0000, 2024-12-09T20:11:00.000+0000)	GOOG	709.4819976806641	486.00083072195673
List(2024-12-10T03:55:00.000+0000, 2024-12-10T03:56:00.000+0000)	MSFT	208.86666870117188	42.36349127178742
List(2024-12-10T03:59:00.000+0000, 2024-12-10T04:00:00.000+0000)	MSFT	882.9275054931641	491.66807567776675
List(2024-12-10T03:54:00.000+0000, 2024-12-10T03:55:00.000+0000)	MSFT	648.7233352661133	316.8769496928872
List(2024-12-09T20:14:00.000+0000, 2024-12-09T20:15:00.000+0000)	AMZN	813.8750050862631	394.9253770968825

Databricks visualization. Run in Databricks to view.

Databricks visualization. Run in Databricks to view.

Databricks visualization. Run in Databricks to view.

In [0]:

```
from pyspark.sql.functions import col, avg, sum, max, to_date
stock_df = spark.read.format("delta").load(delta_table_location)


stock_df = stock_df.withColumn("trade_date", to_date(col("Date")))


daily_summary_df = stock_df.groupBy(
    col("Index"),
    col("trade_date")
).agg(
    avg("Close").alias("average_close_price"),
    sum("Volume").alias("total_volume"),
    max("High").alias("max_high_price")
)


daily_summary_delta_path = "/mnt/delta/daily_summary"
```

```
daily_summary_df.write.mode("overwrite").format("delta").save("/mnt/delta/daily_summery"
)

display(daily_summary_df)
```

Index	trade_date	average_close_price	total_volume	max_high_price
MSFT	2024-12-10	814.3952244063069	137633026	1999.64
AAPL	2024-12-10	818.9569158256054	129669458	1999.84
GOOG	2024-12-09	768.3510169204401	48385673	1988.93
MSFT	2024-12-09	825.9464544480846	16636787	1954.86
GOOG	2024-12-10	779.7777090890067	13472065	1981.56
AMZN	2024-12-09	798.2176567239965	45985449	1999.76
AMZN	2024-12-10	788.1320338154219	123087217	1997.04

In [0]:

```
df=streaming_df
display(df)
```

In [0]:

```
from pyspark.sql.window import Window
from pyspark.sql.functions import lag, col

# Define the window without a frame
window_spec = Window.partitionBy("Index").orderBy("Date")

# Calculate price change percentage
df_with_change = df_with_ma.withColumn(
    "Price_Change_Percentage",
    (col("Close") - lag("Close", 1).over(window_spec)) / lag("Close", 1).over(window_spe
c) * 100
)


display(df_with_change)
```

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD	Moving_Avg_Close	Price_Change_Pe
AAPL	2024-12-09 19:13:48	1190.7	1929.32	915.5	813.87	0.0	833752	281.24	813.8699951171875	
AAPL	2024-12-09 19:14:00	479.11	1989.41	721.53	445.04	0.0	75644	749.66	629.4550018310547	-45.318046
AAPL	2024-12-09 19:14:03	1402.44	1597.12	425.39	886.6	0.0	63463	142.21	715.1699930826823	99.218038
AAPL	2024-12-09 19:14:27	1446.67	1500.19	845.78	161.55	0.0	248933	198.78	497.72999572753906	-81.77870
	2024-									

In [0]:

```
df_with_volatility = df_with_change.withColumn(
    "Volatility", col("High") - col("Low")
)
display(df_with_volatility)
```

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD	Moving_Avg_Close	Price_Change_Pe
AAPL	2024-12-09 19:13:48	1190.7	1929.32	915.5	813.87	0.0	833752	281.24	813.8699951171875	

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD	Moving_Avg_Close	Price_Change_Pe
AAPL	2024-12-09 19:14:00	479.11	1989.41	721.53	445.04	0.0	75644	749.66	629.4550018310547	-45.318046
AAPL	2024-12-09 19:14:03	1402.44	1597.12	425.39	886.6	0.0	63463	142.21	715.1699930826823	99.218038
AAPL	2024-12-09 19:14:27	1446.67	1500.19	845.78	161.55	0.0	248933	198.78	497.72999572753906	-81.77870
	2024-									

In [0]:

```
from pyspark.sql.functions import when, col

alert_df = df_with_volatility.withColumn(
    "Alert",
    when(col("Price_Change_Percentage") > 1, "Triggered").otherwise("None")
)

display(alert_df)
```

Index	Date	Open	High	Low	Close	Adj_Close	Volume	CloseUSD	Moving_Avg_Close	Price_Change_Pe
AAPL	2024-12-09 19:13:48	1190.7	1929.32	915.5	813.87	0.0	833752	281.24	813.8699951171875	
AAPL	2024-12-09 19:14:00	479.11	1989.41	721.53	445.04	0.0	75644	749.66	629.4550018310547	-45.318046
AAPL	2024-12-09 19:14:03	1402.44	1597.12	425.39	886.6	0.0	63463	142.21	715.1699930826823	99.218038
AAPL	2024-12-09 19:14:27	1446.67	1500.19	845.78	161.55	0.0	248933	198.78	497.72999572753906	-81.77870
	2024-									