CLOUD MANAGEMENT
IST 615

# STOCK MARKET ANALYSIS WITH AWS

**Team:**
Sahil Wani
Rithvik Segu
Prakyath Praveen Davanam
Meghana Shivani Invilli

Guided By:Carlos Caicedo Bastidas

# Table of contents

## 1. Introduction

In the rapidly evolving realm of finance, data has emerged as a paramount asset driving strategic decision-making and delineating the trajectories of businesses. The stock market, characterized by its expansive and intricate datasets, presents a distinctive challenge and opportunity for data processing and analysis. The advent of cloud computing platforms, exemplified by Amazon Web Services (AWS), has revolutionized the methodologies employed in handling, interpreting, and extracting insights from this data.

Amazon Web Services (AWS), a globally eminent figure in cloud computing, furnishes a comprehensive suite of services meticulously crafted for data management, processing, and visualization. With AWS, financial institutions, individual investors, and data analysts can harness scalable and cost-efficient solutions to unlock the latent potential embedded within their stock market data. This not only augments their capacity to make judicious decisions but also furnishes them with a competitive edge within the market milieu.

This project endeavors to illustrate the efficacy of AWS in metamorphosing raw stock market data into actionable insights. Our focus is oriented towards the seamless amalgamation of services such as Amazon S3 for data storage, AWS Glue for data preparation, Amazon Glue Data catalog for data management, Amazon Athena for data querying, and Amazon QuickSight for data visualization. Each of these services assumes a pivotal role within the data processing pipeline, synergistically contributing towards furnishing a comprehensive solution for stock market analysis.

The merits inherent in leveraging AWS for this endeavor are multifarious. AWS offers a scalable infrastructure, ensuring that the requisites for data processing and visualization can adapt and burgeon commensurately with the evolution of investment strategies. It furnishes a cost-effective proposition, predicated upon a pay-as-you-go model wherein expenses are incurred solely for the utilization of resources. Furthermore, AWS services harmonize seamlessly with one another, thereby facilitating a streamlined and efficacious workflow.

In the ensuing discourse, we shall embark upon a meticulous examination of the step-by-step process underpinning the analysis of stock market data utilizing AWS. We shall expound upon the architectural framework of the project, delineate the sequential stages involved therein, explicate the challenges encountered, and distill the salient lessons garnered. Our objective is to furnish a comprehensive compendium elucidating the utilization of AWS for stock market analysis, thereby underscore its efficacy and efficiency.

## 2. Data Preprocessing:

The data cleaning process is carefully crafted to manage stock price information from several prominent companies, including Apple Inc., Microsoft Inc., Qualcomm Incorporated, and Advanced Micro Devices Inc (AMD). The process begins by setting the stage with essential tools: pandas for handling data with finesse, and numpy for numerical calculations that might be needed later.

Each company's data, stored in separate CSV files, is meticulously loaded into the environment. To make navigating this multitude of data smoother, a 'Company' column is added to each dataset, acting as a label to clearly mark where each piece of data originated. This thoughtful tagging is crucial for keeping the data organized once the separate tables are combined into one.

The merging of these datasets into a single Data Frame is like gathering all the pieces of a puzzle together. This unified structure not only simplifies the handling of the data but also sets the stage for the next steps of the cleaning process. With the data combined, the notebook takes a moment to examine the structure through various lensð looking at data types, counting entries, and identifying any missing pieces in the puzzle, like missing values.

Next, we address a common quirk of financial datað prices formatted as strings with pesky currency symbols. These are carefully stripped away and converted into floats, a transformation that turns these strings into numbers that are ready for any heavy-lifting calculations needed in financial analysis.

Not stopping at just cleaning, we enrich the dataset further by extracting and adding new columns for the day, month, year, and quarter from the existing date information. These additional time-based features are invaluable, providing new angles from which to analyze the data, whether you're looking back at trends or forecasting the future.

In the final act, this cleaned and polished dataset is saved out to a new CSV file. This file stands ready for any analysis or reporting that might follow, encapsulating all the thoughtful preparation into a form that's consistent, accurate, and analysis ready. This meticulous preparation ensures that the data isn't just cleanð it's primed to deliver insights.

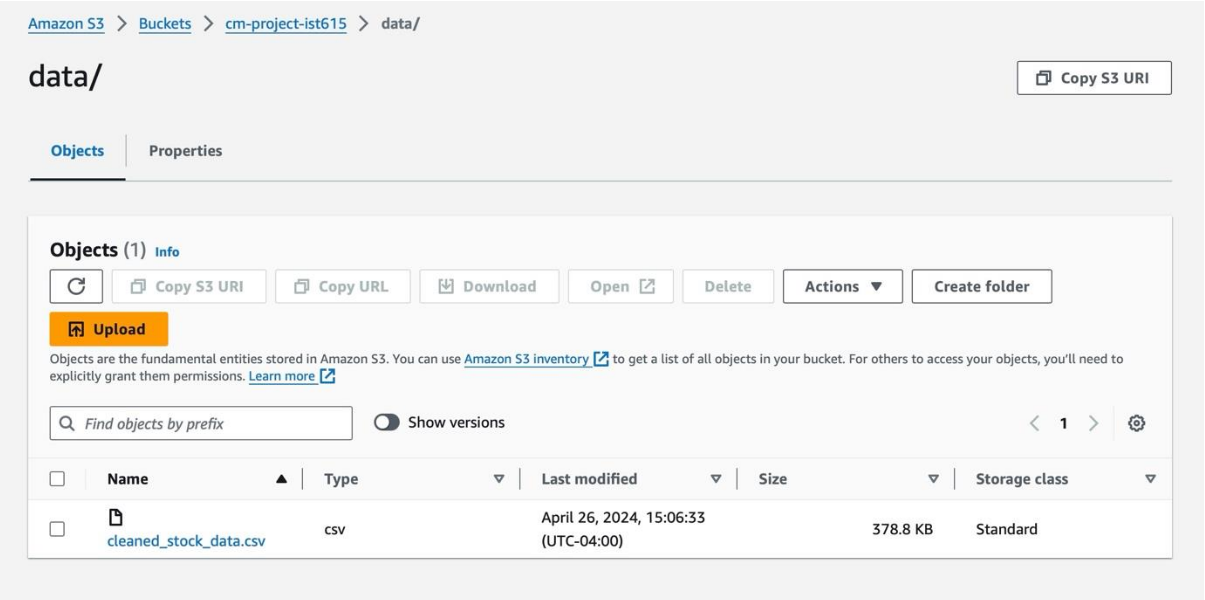
## 3. AWS Services Configuration:

In configuring the AWS services for the project, a systematic approach was adopted to establish the requisite infrastructure and workflows for efficient data management, preprocessing, analysis, and visualization. Below, we outline the key steps involved in configuring each AWS service utilized within the project:

Amazon S3 Configuration:

Amazon Simple Storage Service (S3) serves as the primary repository for storing and retrieving raw and processed stock market data. The configuration process involved:

- Creation of a new S3 bucket via the AWS Management Console or utilizing an existing bucket to house the project's datasets.
- Definition of access controls and permissions, ensuring granular control over data access and security measures.
- Upload CSV files containing raw stock market data to the designated S3 bucket, enabling seamless data ingestion and storage.



Amazon S3 > Buckets > cm-project-ist615 > data/

data/

Objects (1) Info

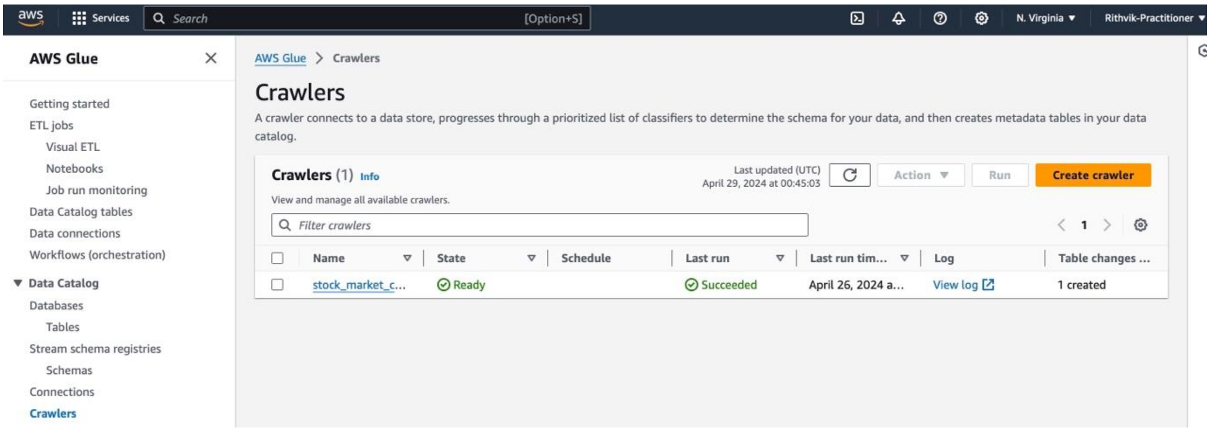
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

| □ | Name | Type | Last modified | Size | Storage class |
|---|---|---|---|---|---|
| □ | cleaned_stock_data.csv | csv | April 26, 2024, 15:06:33 (UTC-04:00) | 378.8 KB | Standard |

AWS Crawler Configuration:

AWS Glue crawlers played a crucial role in automating the discovery and cataloging of data schemas, enabling seamless integration with the Glue Data Catalog. The configuration process involved:

- Creation of a new crawler within the AWS Glue console, specifying the data source location (S3 bucket) containing the uploaded CSV files.
- Configuration of crawler settings, including IAM roles, database location, and frequency of execution, to tailor the crawling process to project requirements.
- Execution of the crawler to automatically scan and analyze the data sources, infer the schema, and populate the Glue Data Catalog with metadata entries.
- Validation of the generated schema information, ensuring alignment with the structure of the raw data and facilitating accurate data processing and analysis.



AWS Glue > Crawlers

Crawlers

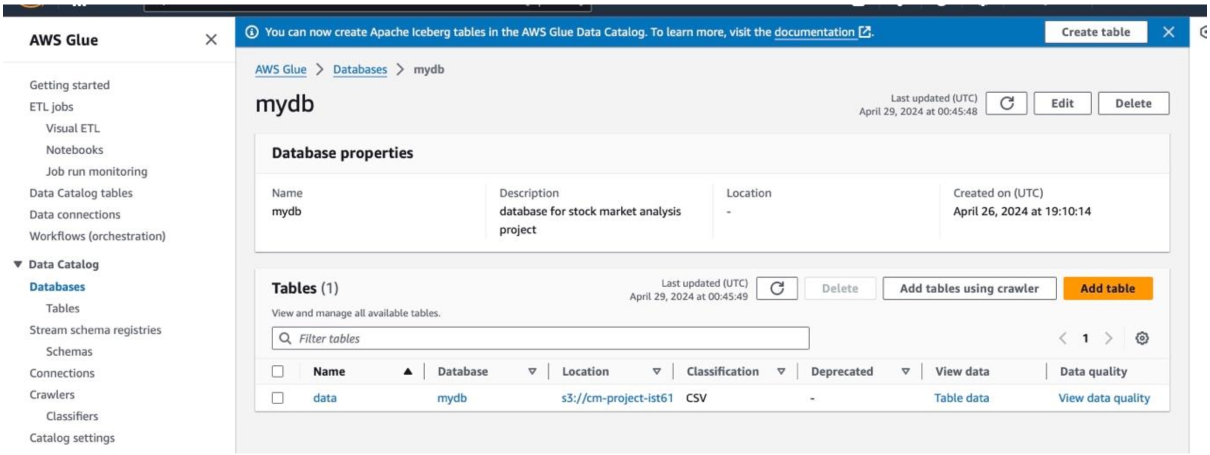
A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawlers (1) Info

Last updated (UTC)
April 29, 2024 at 00:45:03

View and manage all available crawlers.

| □ | Name | State | Schedule | Last run | Last run tim... | Log | Table changes ... |
|---|---|---|---|---|---|---|---|
| □ | stock_market_c... | Ready |  | Succeeded | April 26, 2024 a... | View log | 1 created |



You can now create Apache Iceberg tables in the AWS Glue Data Catalog. To learn more, visit the documentation.

AWS Glue > Databases > mydb

mydb

Last updated (UTC) April 29, 2024 at 00:45:48

Database properties

| Name | Description | Location | Created on (UTC) |
|---|---|---|---|
| mydb | database for stock market analysis project | - | April 26, 2024 at 19:10:14 |

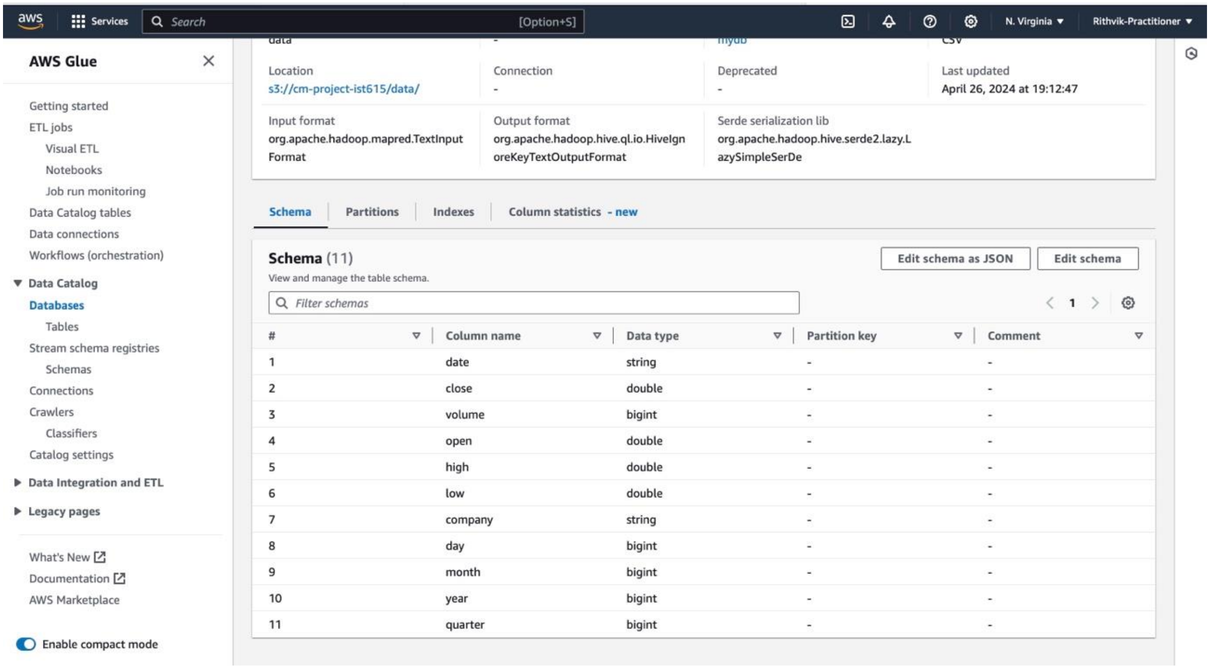
Tables (1)

View and manage all available tables.

Last updated (UTC) April 29, 2024 at 00:45:49

Filter tables

| Name | Database | Location | Classification | Deprecated | View data | Data quality |
|---|---|---|---|---|---|---|
| data | mydb | s3://cm-project-ist61 | CSV | - | Table data | View data quality |



| Location | Connection | Deprecated | Last updated |
|---|---|---|---|
| s3://cm-project-ist615/data/ | - | - | April 26, 2024 at 19:12:47 |

| Input format | Output format | Serde serialization lib |
|---|---|---|
| org.apache.hadoop.mapred.TextInputFormat | org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat | org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |

Schema (11)

View and manage the table schema.

Filter schemas

| # | Column name | Data type | Partition key | Comment |
|---|---|---|---|---|
| 1 | date | string | - | - |
| 2 | close | double | - | - |
| 3 | volume | bigint | - | - |
| 4 | open | double | - | - |
| 5 | high | double | - | - |
| 6 | low | double | - | - |
| 7 | company | string | - | - |
| 8 | day | bigint | - | - |
| 9 | month | bigint | - | - |
| 10 | year | bigint | - | - |
| 11 | quarter | bigint | - | - |

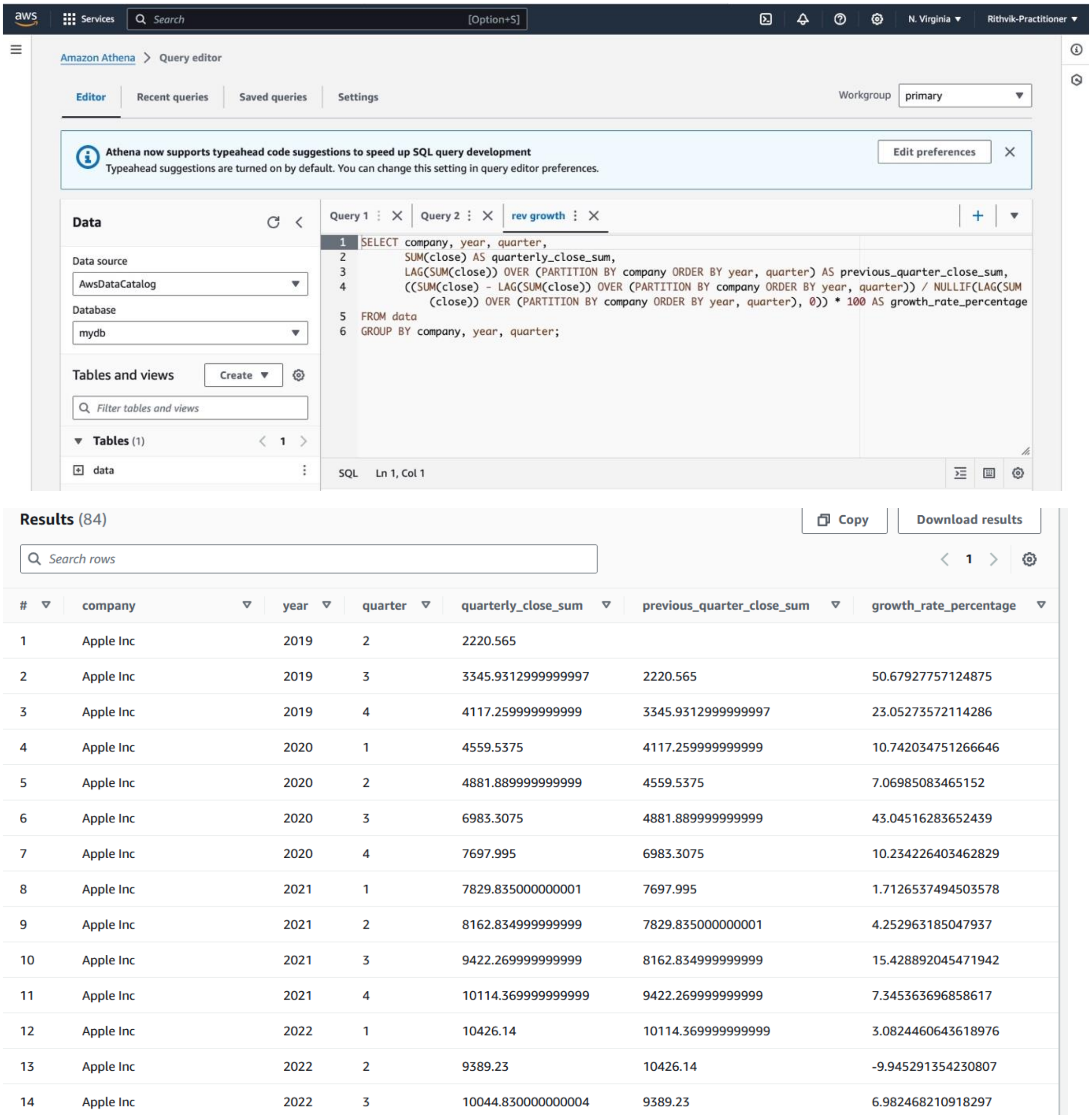
## Amazon Athena Configuration:

Amazon Athena provided an interactive serverless query service for analyzing raw data stored in S3 using standard SQL syntax. The configuration process comprised:

- Creation of a new database within Athena corresponding to the Glue Data Catalog database, establishing a logical namespace for querying the cataloged data.
- Definition of tables within Athena based on the schema inferred by Glue crawlers, enabling seamless data querying and analysis.
- Utilization of standard SQL queries to interactively analyze and query the data stored in S3 via Athena, facilitating ad-hoc exploration and insights generation.

- ➢ Optimization of query performance through the utilization of metadata stored in the Glue Data Catalog, ensuring efficient query execution and resource utilization.

Query 1:



Workgroup: primary

Athena now supports typeahead code suggestions to speed up SQL query development
Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.

Data source: AwsDataCatalog

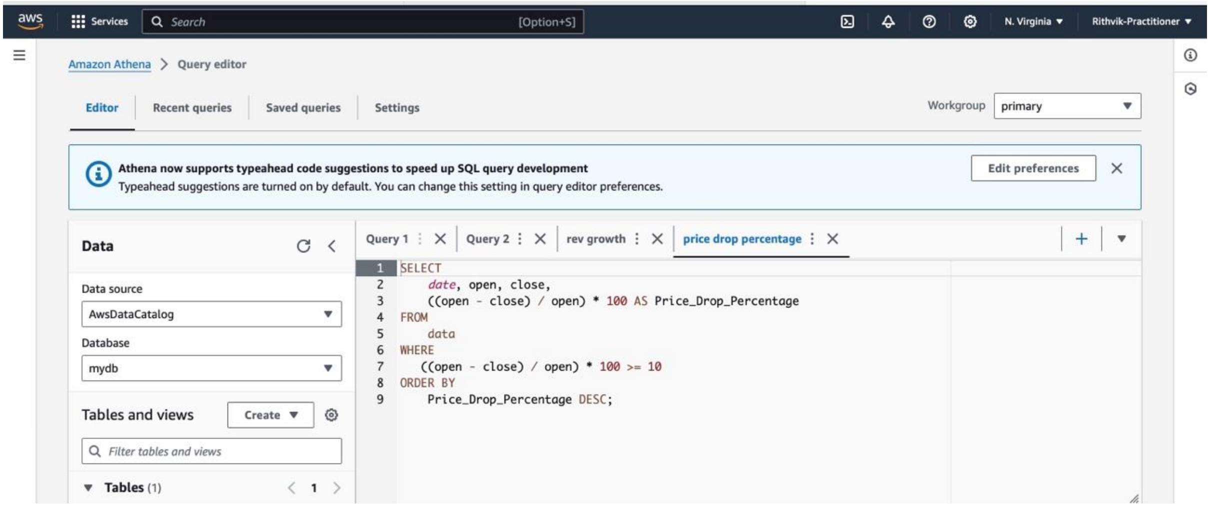
Database: mydb

```sql
SELECT company, year, quarter,
       SUM(close) AS quarterly_close_sum,
       LAG(SUM(close)) OVER (PARTITION BY company ORDER BY year, quarter) AS previous_quarter_close_sum,
       ((SUM(close) - LAG(SUM(close)) OVER (PARTITION BY company ORDER BY year, quarter)) / NULLIF(LAG(SUM
          (close)) OVER (PARTITION BY company ORDER BY year, quarter), 0)) * 100 AS growth_rate_percentage
FROM data
GROUP BY company, year, quarter;
```

Results (84)

| # | company | year | quarter | quarterly_close_sum | previous_quarter_close_sum | growth_rate_percentage |
|---|---|---|---|---|---|---|
| 1 | Apple Inc | 2019 | 2 | 2220.565 | | |
| 2 | Apple Inc | 2019 | 3 | 3345.9312999999997 | 2220.565 | 50.67927757124875 |
| 3 | Apple Inc | 2019 | 4 | 4117.259999999999 | 3345.9312999999997 | 23.05273572114286 |
| 4 | Apple Inc | 2020 | 1 | 4559.5375 | 4117.259999999999 | 10.742034751266646 |
| 5 | Apple Inc | 2020 | 2 | 4881.889999999999 | 4559.5375 | 7.06985083465152 |
| 6 | Apple Inc | 2020 | 3 | 6983.3075 | 4881.889999999999 | 43.04516283652439 |
| 7 | Apple Inc | 2020 | 4 | 7697.995 | 6983.3075 | 10.234226403462829 |
| 8 | Apple Inc | 2021 | 1 | 7829.835000000001 | 7697.995 | 1.7126537494503578 |
| 9 | Apple Inc | 2021 | 2 | 8162.834999999999 | 7829.835000000001 | 4.252963185047937 |
| 10 | Apple Inc | 2021 | 3 | 9422.269999999999 | 8162.834999999999 | 15.428892045471942 |
| 11 | Apple Inc | 2021 | 4 | 10114.369999999999 | 9422.269999999999 | 7.345363696858617 |
| 12 | Apple Inc | 2022 | 1 | 10426.14 | 10114.369999999999 | 3.0824460643618976 |
| 13 | Apple Inc | 2022 | 2 | 9389.23 | 10426.14 | -9.945291354230807 |
| 14 | Apple Inc | 2022 | 3 | 10044.830000000004 | 9389.23 | 6.982468210918297 |

## Query 2:



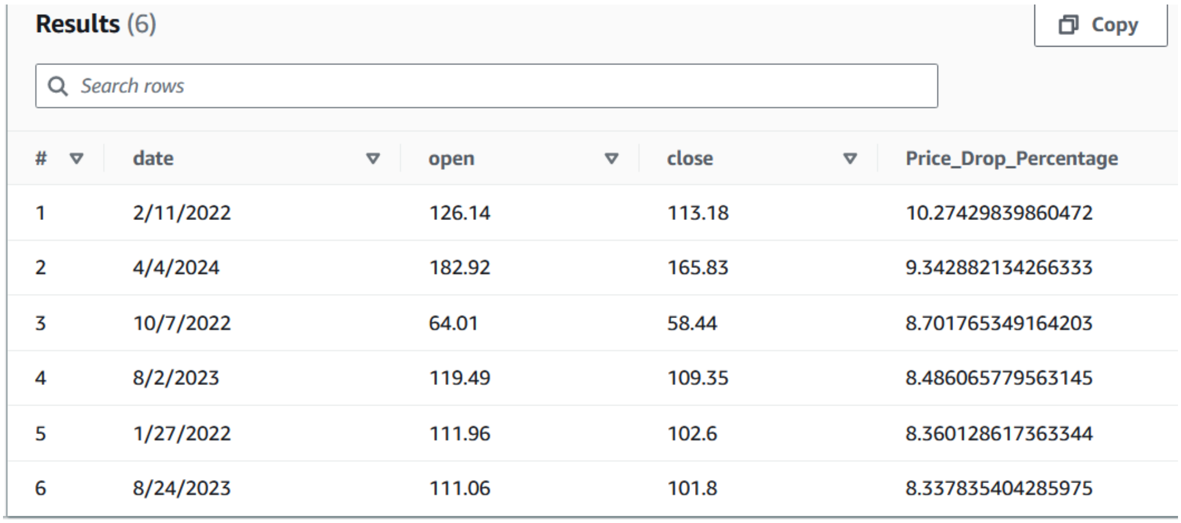
Workgroup: primary

**Athena now supports typeahead code suggestions to speed up SQL query development**
Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.

Data source: AwsDataCatalog

Database: mydb

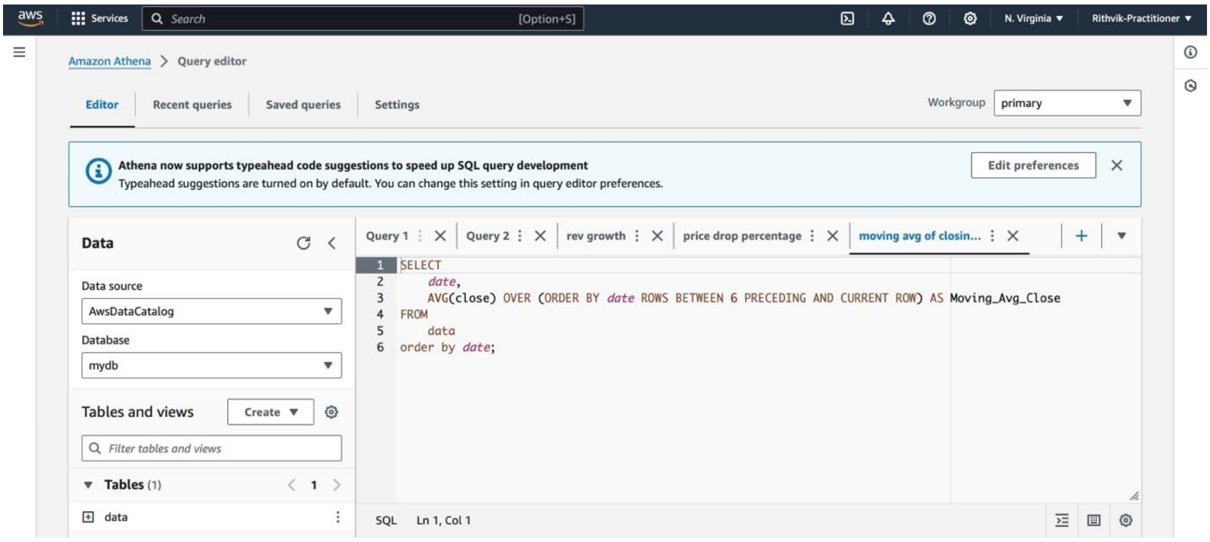
```sql
SELECT
    date, open, close,
    ((open - close) / open) * 100 AS Price_Drop_Percentage
FROM
    data
WHERE
   ((open - close) / open) * 100 >= 10
ORDER BY
    Price_Drop_Percentage DESC;
```


Copy
Search rows

**Results** (6)

| # | date | open | close | Price_Drop_Percentage |
|---|---|---|---|---|
| 1 | 2/11/2022 | 126.14 | 113.18 | 10.27429839860472 |
| 2 | 4/4/2024 | 182.92 | 165.83 | 9.342882134266333 |
| 3 | 10/7/2022 | 64.01 | 58.44 | 8.701765349164203 |
| 4 | 8/2/2023 | 119.49 | 109.35 | 8.486065779563145 |
| 5 | 1/27/2022 | 111.96 | 102.6 | 8.360128617363344 |
| 6 | 8/24/2023 | 111.06 | 101.8 | 8.337835404285975 |

## Query 3:



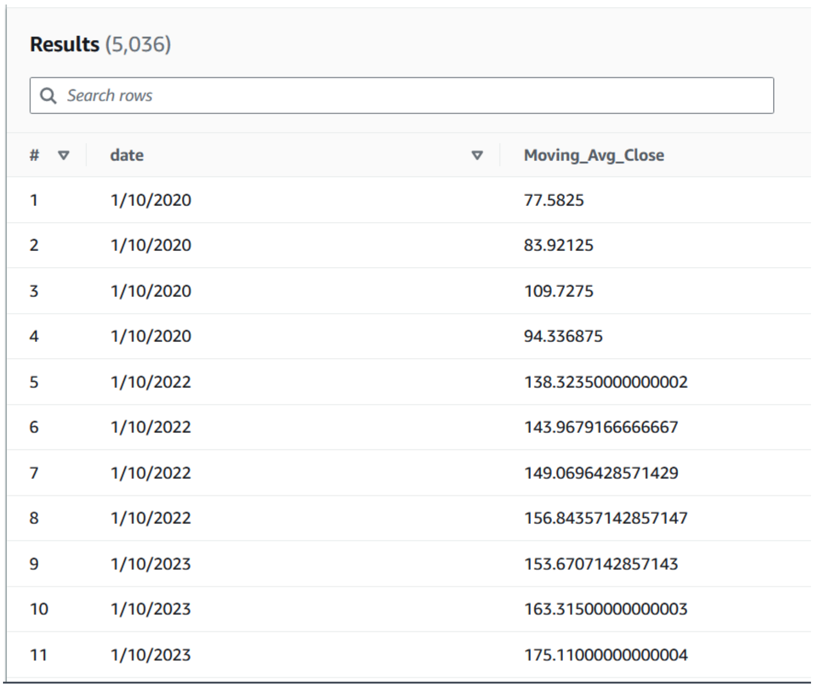
Workgroup: primary

**Athena now supports typeahead code suggestions to speed up SQL query development**
Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.

Data source: AwsDataCatalog

Database: mydb

```sql
SELECT
    date,
    AVG(close) OVER (ORDER BY date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS Moving_Avg_Close
FROM
    data
order by date;
```

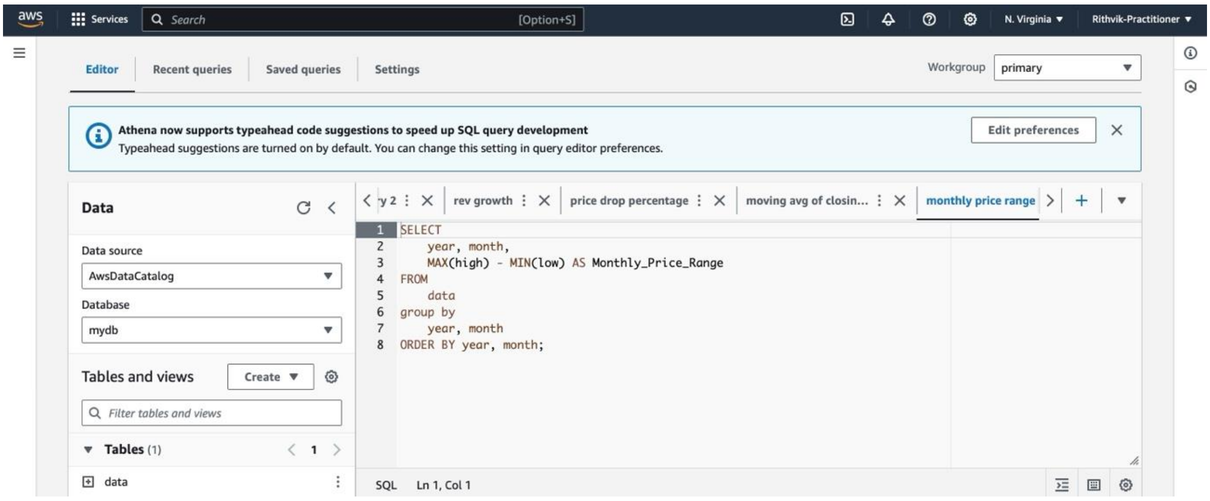


**Results** (5,036)

Search rows

| # | date | Moving_Avg_Close |
|---|---|---|
| 1 | 1/10/2020 | 77.5825 |
| 2 | 1/10/2020 | 83.92125 |
| 3 | 1/10/2020 | 109.7275 |
| 4 | 1/10/2020 | 94.336875 |
| 5 | 1/10/2022 | 138.32350000000002 |
| 6 | 1/10/2022 | 143.9679166666667 |
| 7 | 1/10/2022 | 149.0696428571429 |
| 8 | 1/10/2022 | 156.84357142857147 |
| 9 | 1/10/2023 | 153.6707142857143 |
| 10 | 1/10/2023 | 163.31500000000003 |
| 11 | 1/10/2023 | 175.11000000000004 |

Query 4:



Athena now supports typeahead code suggestions to speed up SQL query development

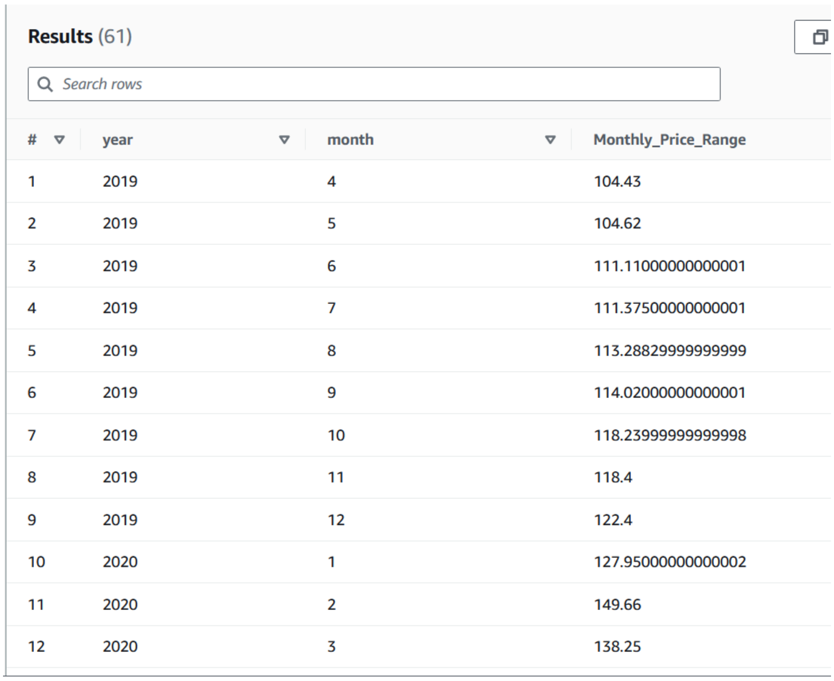
Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.

Data source: AwsDataCatalog

Database: mydb

Workgroup: primary

```
SELECT
    year, month,
    MAX(high) - MIN(low) AS Monthly_Price_Range
FROM
    data
group by
    year, month
ORDER BY year, month;
```



Results (61)

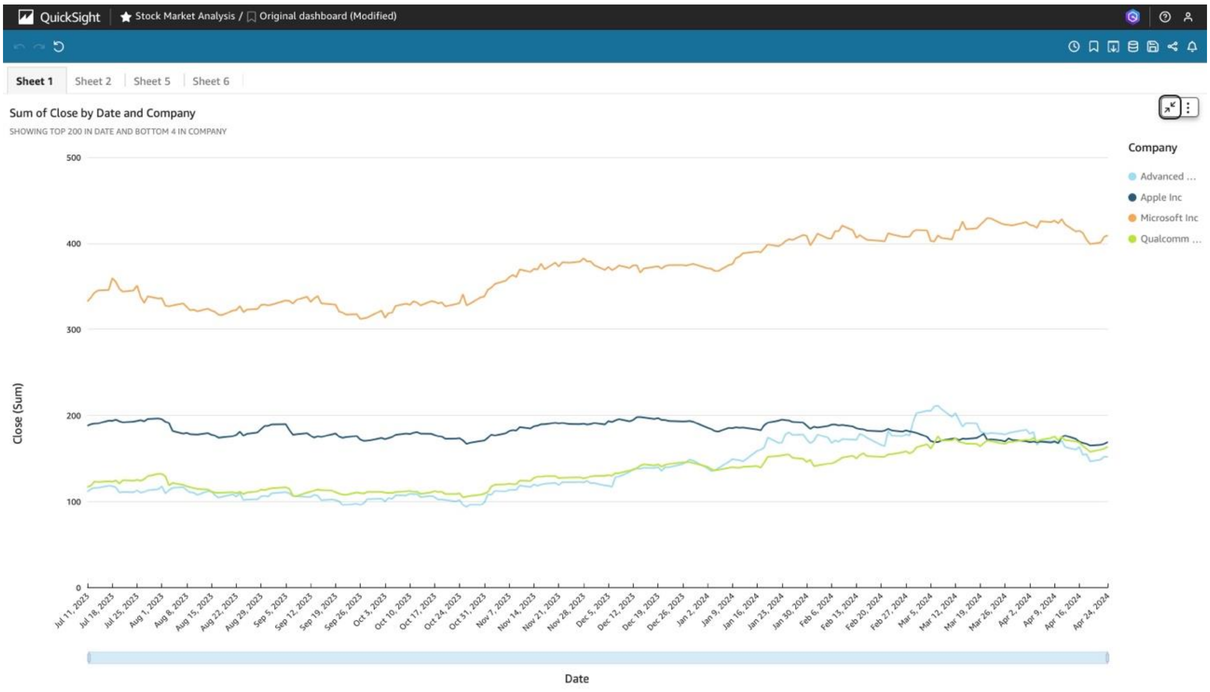
| # | year | month | Monthly_Price_Range |
|---|---|---|---|
| 1 | 2019 | 4 | 104.43 |
| 2 | 2019 | 5 | 104.62 |
| 3 | 2019 | 6 | 111.11000000000001 |
| 4 | 2019 | 7 | 111.37500000000001 |
| 5 | 2019 | 8 | 113.28829999999999 |
| 6 | 2019 | 9 | 114.02000000000001 |
| 7 | 2019 | 10 | 118.23999999999998 |
| 8 | 2019 | 11 | 118.4 |
| 9 | 2019 | 12 | 122.4 |
| 10 | 2020 | 1 | 127.95000000000002 |
| 11 | 2020 | 2 | 149.66 |
| 12 | 2020 | 3 | 138.25 |

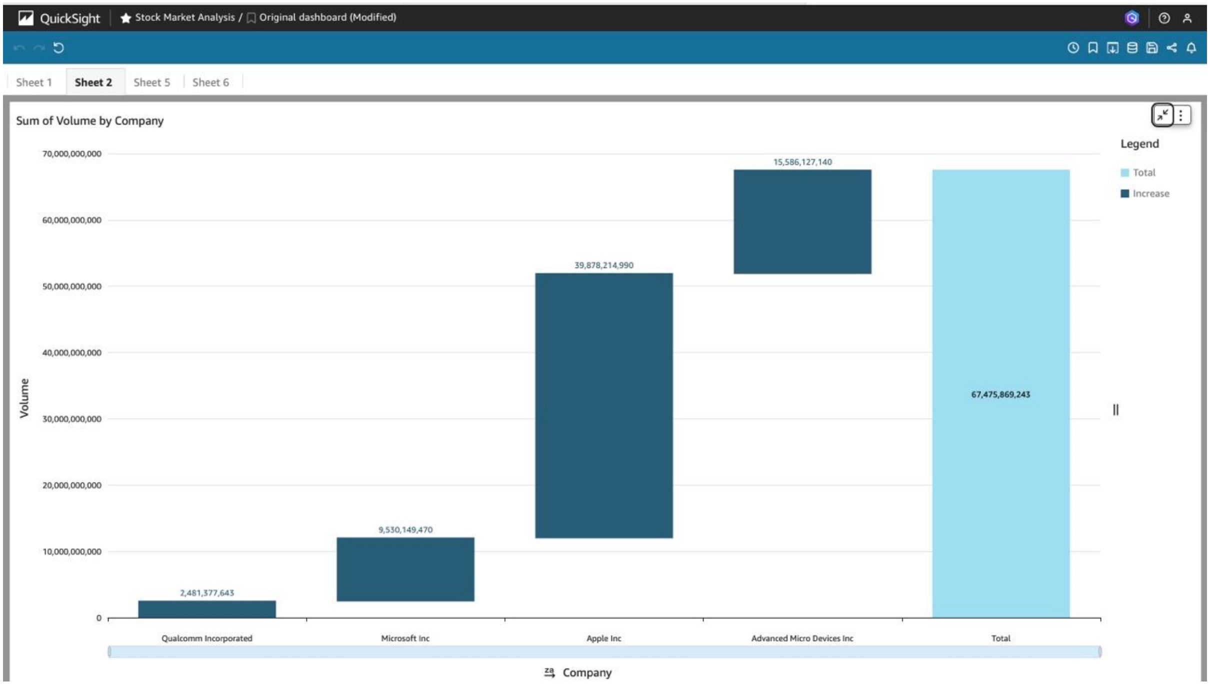
## QuickSight Configuration:

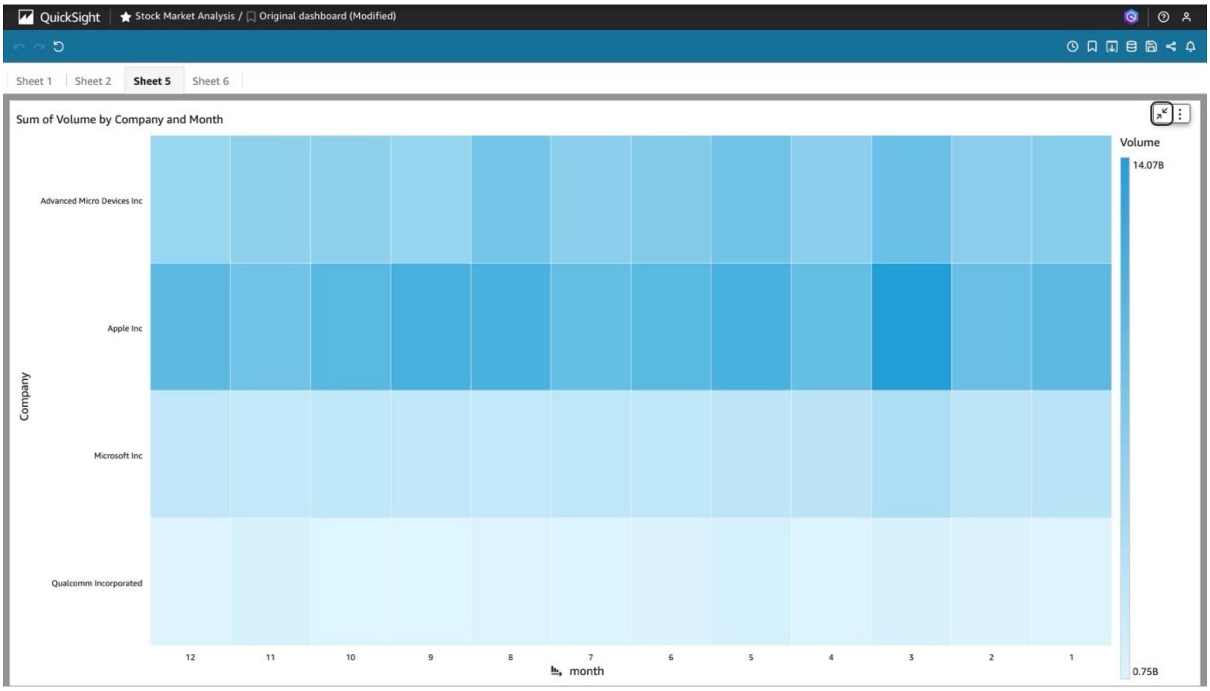
Amazon QuickSight served as the primary Business Intelligence and visualization tool for generating actionable insights from the analyzed data. The configuration process involved:

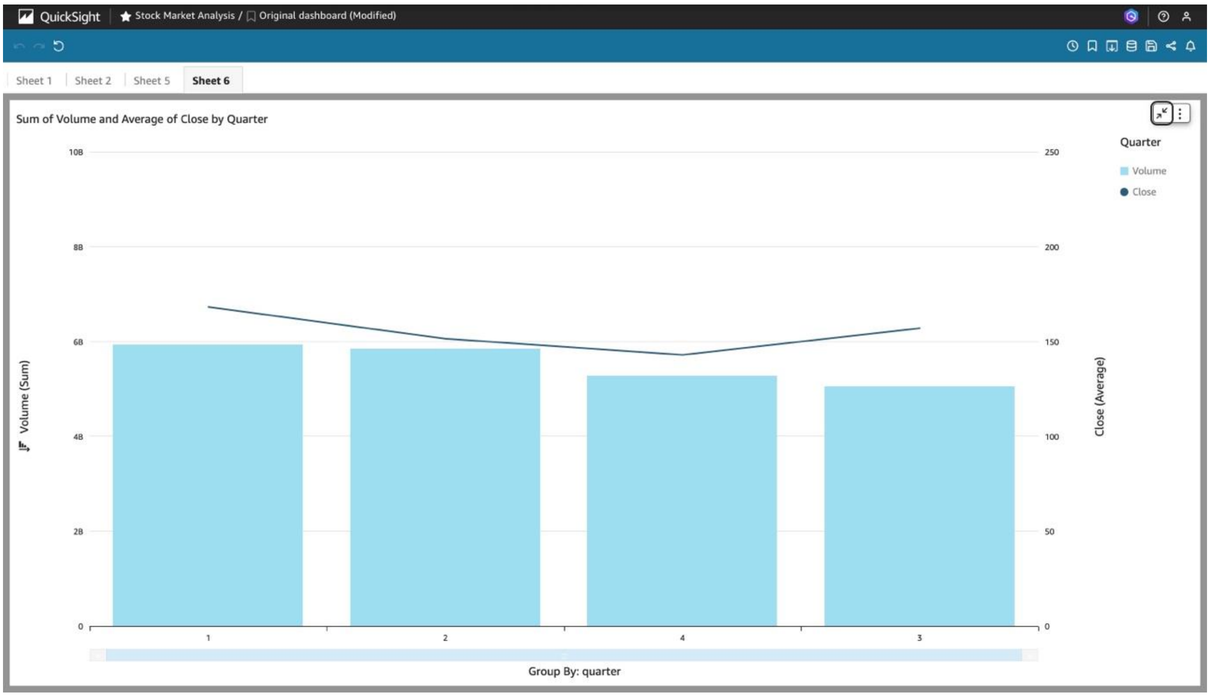
- Creation of a new QuickSight analysis and connection to the Glue Data Catalog database, establishing seamless integration with the cataloged data.
- Definition of datasets within QuickSight based on the tables available in the Glue Data Catalog, enabling access to curated datasets for visualization.
- Design and customization of interactive dashboards and visualizations within QuickSight, facilitating intuitive exploration and analysis of the data.
- Sharing of dashboards with stakeholders and scheduling automated data refreshes for real-time insights dissemination, ensuring timely decision-making and strategic planning.

## Visualizations:



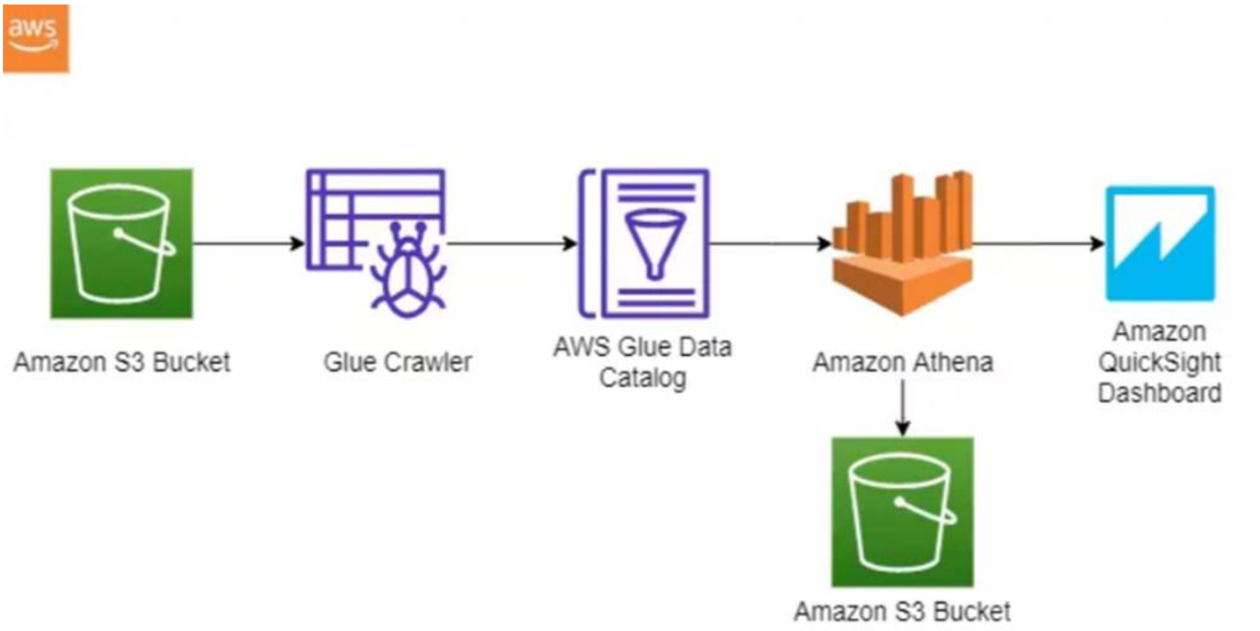






Through meticulous configuration and integration of these AWS services, the project established a robust and scalable infrastructure for conducting comprehensive stock market analysis, leveraging the capabilities of cloud computing to drive informed decision-making and strategic insights.

## 4. System Architecture:



1. Amazon S3 Bucket: The process begins with the selection of a new Amazon S3 bucket, where we uploaded the stock market data files for storage. Amazon S3 provides a highly durable and scalable storage solution, making it an ideal choice for handling large volumes of financial data.

2. AWS Glue Database Creation: Within AWS Glue, we created a new database to organize the processed data. This database serves as a structured repository, allowing for efficient data management and retrieval.

3. Glue Crawler Configuration: A Glue Crawler was set up and run to populate the AWS Glue Data Catalog with metadata from the S3 bucket. This step is crucial as it automates the extraction, transformation, and loading (ETL) of data, inferring the schema and structure without manual intervention.

4. Amazon Athena for SQL Queries: Utilizing Amazon Athena, we executed SQL queries against the data organized in the Glue Data Catalog. Athena's serverless nature allows for flexible and cost-effective querying directly on the data stored in S3.

5. Amazon QuickSight for Visualization: Finally, we connected Amazon QuickSight to Athena to visualize and analyze the data through interactive dashboards. QuickSight's powerful visualization capabilities enabled us to transform complex data sets into clear and actionable insights.

## 5. Issues Encountered:

During the course of our project, we encountered several challenges that tested our problem-solving skills and technical expertise. Here’s a detailed account of these issues:

1. **Server Overloading**: One of the significant challenges we faced was server overloading. Kafka, due to its extensive resource requirements, proved to be a heavy application for our virtual machine. The large volume of load was beyond the capacity of our virtual machine, leading to memory constraints. This issue underscored the importance of ensuring adequate resources while deploying heavy applications like Kafka.

2. **Creating Kafka Topics**: Another hurdle we encountered was while attempting to create topics within our Kafka consumer instance. The process was not as straightforward as we anticipated, and we faced several challenges. This experience highlighted the complexities of working with advanced tools like Kafka and the need for a deep understanding of its functionalities.

3. **AWS QuickSight Integration**: Lastly, we faced limitations with the AWS Learner’s Account, which prevented us from using the AWS QuickSight service. QuickSight is a powerful visualization tool in the AWS suite, and not being able to use it due to account limitations was a setback. This issue emphasized the need to be aware of the limitations of the services we use and the importance of finding suitable alternatives when necessary.

## 6. Lessons Learned:

Throughout the "Stock Market Analysis Using AWS Services" project, participants gained a wealth of knowledge and practical skills that enriched their understanding of complex data processing using Amazon Web Services. Here are the key insights and learnings distilled from the experience:

1. Mastery of AWS Tools: The project offered a hands-on opportunity to explore and integrate various AWS services, including AWS Glue, Athena, and QuickSight. Each tool played a critical role in the data lifecycle, from storage and cleaning to analysis and visualization, showcasing the versatility and robustness of AWS in handling cloud-based data tasks.

2. Data Processing Insights: A major focus was on the preprocessing of raw stock market data, which involved cleaning and organizing the data meticulously. This process highlighted the importance of clean and well-structured data in deriving accurate insights, teaching the team the critical foundations of effective data management.

3. Scalability and Economic Efficiency: Using AWS's scalable infrastructure allowed the team to efficiently manage fluctuating data volumes. This experience illustrated the benefits of cloud resources, which can be scaled to project needs without the hefty initial investment in physical infrastructure.

4. Streamlined Workflows: The seamless integration of AWS services demonstrated how a cohesive cloud ecosystem can streamline entire workflows. This experience was invaluable in showing how integrated systems can simplify the journey from data ingestion to insightful visualizations, making the process faster and more efficient.

## 7. Future Enhancements:

1. Welcoming AWS SageMaker: Integrating AWS SageMaker would be a game-changer, allowing the project to harness machine learning and artificial intelligence for sharper predictive analytics. SageMaker’s comprehensive toolkit for developing, training, and deploying machine learning models opens up exciting possibilities for forecasting market trends, enriching the decision-making process with deeper insights.

2. Adopting Real-time Data Processing: Adding real-time data processing capabilities with AWS Lambda and Kinesis could transform the project's approach to data management. This update would enable the system to analyze and respond to market changes as they happen, providing a vital edge in environments where speed is of the essence.

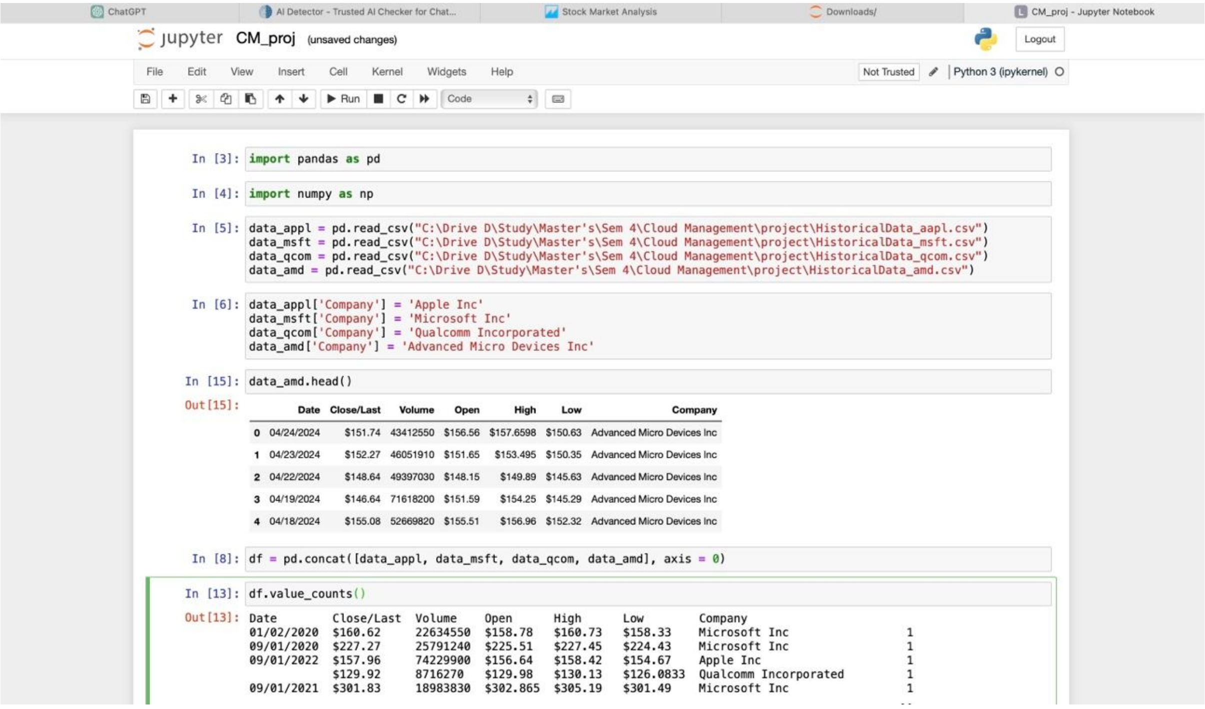
3. Expanding Visualization Tools: Enhancing the project's visualization capabilities can make the data not just more accessible, but also more engaging. By building on the foundation laid by Amazon QuickSight and incorporating more interactive and dynamic visualization features, users can explore the data in new ways, crafting personalized insights that are both rich and relevant.

## 8. Conclusion

In conclusion, the project "Stock Market Analysis Using AWS Services" has exemplified the transformative capacity of cloud computing, particularly through the deployment of Amazon Web Services (AWS), in extracting actionable insights from raw stock market data. By integrating AWS services such as Amazon S3, AWS Glue, Amazon Athena, and Amazon QuickSight, we established a robust infrastructure and workflow for comprehensive data management, preprocessing, analysis, and visualization. This approach not only enhanced scalability and cost-effectiveness but also provided a competitive advantage in the dynamic financial landscape.

Despite encountering challenges such as server overloading and AWS account limitations, we persevered and overcame these hurdles through problem-solving and resourcefulness. These experiences underscored the importance of resource allocation, deep understanding of tools and services, and adaptability in navigating constraints. In essence, the project showcases the immense potential of AWS in empowering organizations and individuals to derive valuable insights from complex datasets, facilitating informed decision-making and strategic planning in the finance sector and beyond.

# 9. Pre-Processing Code:


ChatGPT
AI Detector - Trusted AI Checker for Chat...
Stock Market Analysis
Downloads/
CM_proj - Jupyter Notebook
jupyter CM_proj (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [3]: import pandas as pd
```

```
In [4]: import numpy as np
```

```
In [5]: data_appl = pd.read_csv("C:\Drive D\Study\Master's\Sem 4\Cloud Management\project\HistoricalData_aapl.csv")
        data_msft = pd.read_csv("C:\Drive D\Study\Master's\Sem 4\Cloud Management\project\HistoricalData_msft.csv")
        data_qcom = pd.read_csv("C:\Drive D\Study\Master's\Sem 4\Cloud Management\project\HistoricalData_qcom.csv")
        data_amd = pd.read_csv("C:\Drive D\Study\Master's\Sem 4\Cloud Management\project\HistoricalData_amd.csv")
```

```
In [6]: data_appl['Company'] = 'Apple Inc'
        data_msft['Company'] = 'Microsoft Inc'
        data_qcom['Company'] = 'Qualcomm Incorporated'
        data_amd['Company'] = 'Advanced Micro Devices Inc'
```

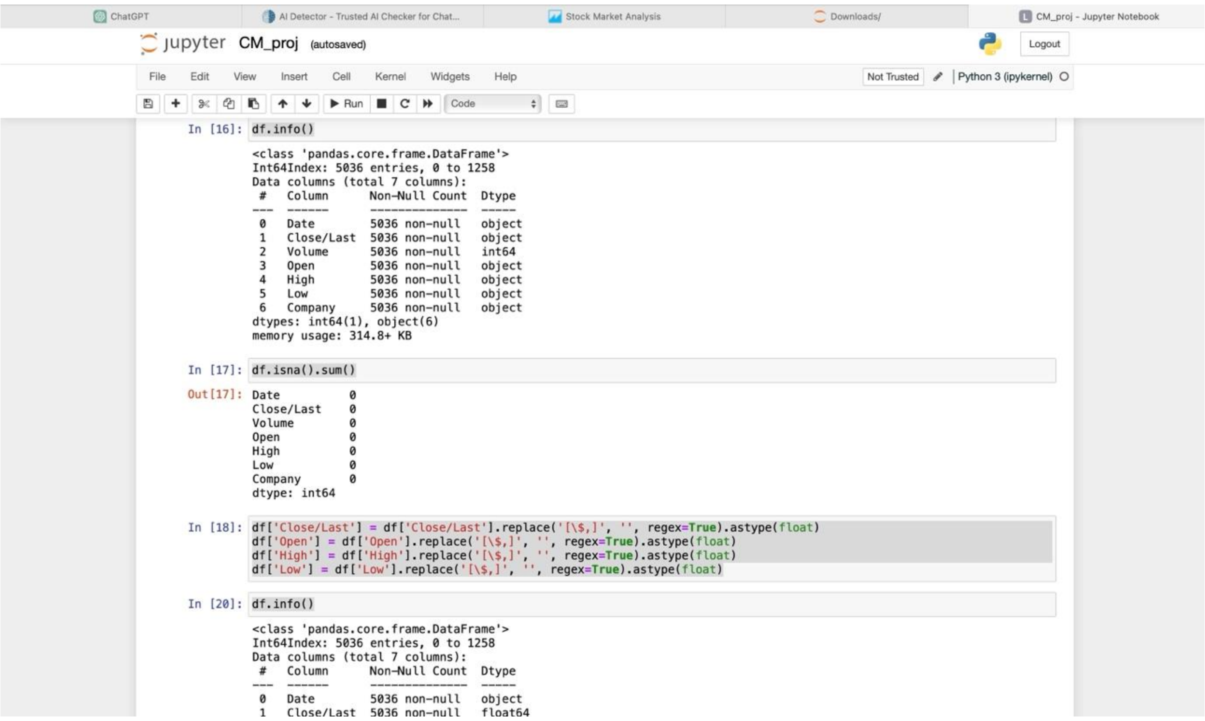
```
In [15]: data_amd.head()
```

Out[15]:

| | Date | Close/Last | Volume | Open | High | Low | Company |
|---|---|---|---|---|---|---|---|
| 0 | 04/24/2024 | $151.74 | 43412550 | $156.56 | $157.6598 | $150.63 | Advanced Micro Devices Inc |
| 1 | 04/23/2024 | $152.27 | 46051910 | $151.65 | $153.495 | $150.35 | Advanced Micro Devices Inc |
| 2 | 04/22/2024 | $148.64 | 49397030 | $148.15 | $149.89 | $145.63 | Advanced Micro Devices Inc |
| 3 | 04/19/2024 | $146.64 | 71618200 | $151.59 | $154.25 | $145.29 | Advanced Micro Devices Inc |
| 4 | 04/18/2024 | $155.08 | 52669820 | $155.51 | $156.96 | $152.32 | Advanced Micro Devices Inc |

```
In [8]: df = pd.concat([data_appl, data_msft, data_qcom, data_amd], axis = 0)
```

```
In [13]: df.value_counts()
Out[13]: Date        Close/Last  Volume    Open      High      Low       Company
         01/02/2020  $160.62     22634550  $158.78   $160.73   $158.33   Microsoft Inc               1
         09/01/2020  $227.27     25791240  $225.51   $227.45   $224.43   Microsoft Inc               1
         09/01/2022  $157.96     74229900  $156.64   $158.42   $154.67   Apple Inc                   1
                     $129.92     8716270   $129.98   $130.13   $126.0833 Qualcomm Incorporated       1
         09/01/2021  $301.83     18983830  $302.865  $305.19   $301.49   Microsoft Inc               1
```

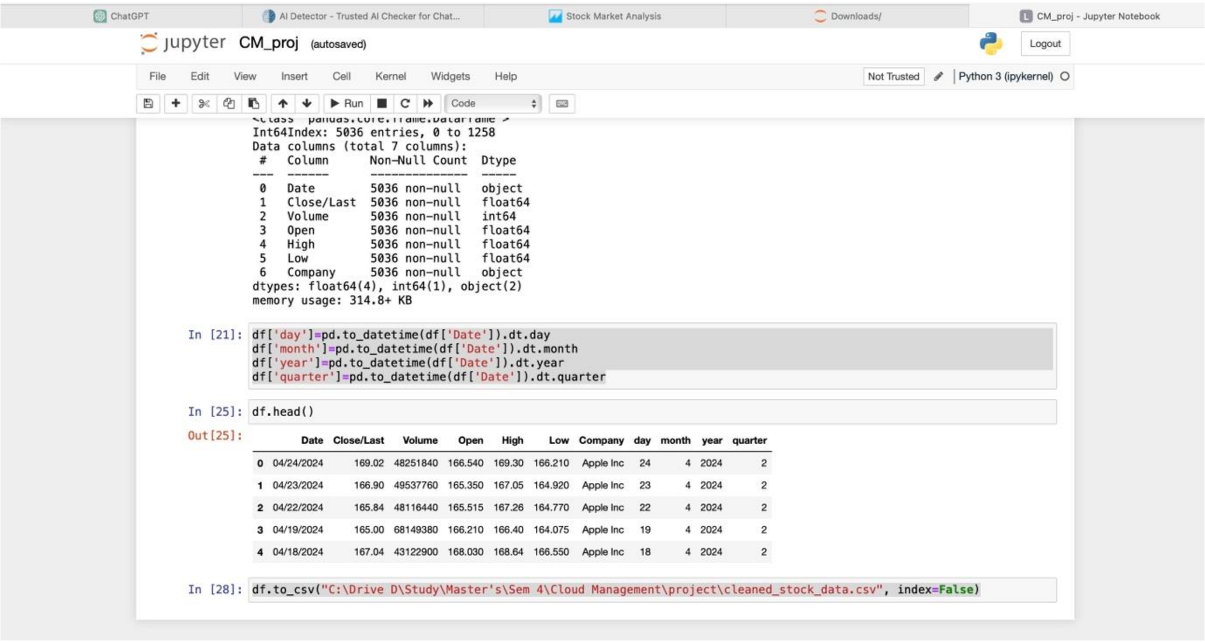

ChatGPT
AI Detector - Trusted AI Checker for Chat...
Stock Market Analysis
Downloads/
CM_proj - Jupyter Notebook
jupyter CM_proj (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [16]: df.info()

         <class 'pandas.core.frame.DataFrame'>
         Int64Index: 5036 entries, 0 to 1258
         Data columns (total 7 columns):
          #   Column      Non-Null Count  Dtype
         ---  ------      --------------  -----
          0   Date        5036 non-null   object
          1   Close/Last  5036 non-null   object
          2   Volume      5036 non-null   int64
          3   Open        5036 non-null   object
          4   High        5036 non-null   object
          5   Low         5036 non-null   object
          6   Company     5036 non-null   object
         dtypes: int64(1), object(6)
         memory usage: 314.8+ KB
```

```
In [17]: df.isna().sum()
Out[17]: Date          0
         Close/Last    0
         Volume        0
         Open          0
         High          0
         Low           0
         Company       0
         dtype: int64
```

```
In [18]: df['Close/Last'] = df['Close/Last'].replace('[\$,]', '', regex=True).astype(float)
         df['Open'] = df['Open'].replace('[\$,]', '', regex=True).astype(float)
         df['High'] = df['High'].replace('[\$,]', '', regex=True).astype(float)
         df['Low'] = df['Low'].replace('[\$,]', '', regex=True).astype(float)
```

```
In [20]: df.info()

         <class 'pandas.core.frame.DataFrame'>
         Int64Index: 5036 entries, 0 to 1258
         Data columns (total 7 columns):
          #   Column      Non-Null Count  Dtype
         ---  ------      --------------  -----
          0   Date        5036 non-null   object
          1   Close/Last  5036 non-null   float64
```


ChatGPT
AI Detector - Trusted AI Checker for Chat...
Stock Market Analysis
Downloads/
CM_proj - Jupyter Notebook
jupyter CM_proj (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5036 entries, 0 to 1258
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Date        5036 non-null   object
 1   Close/Last  5036 non-null   float64
 2   Volume      5036 non-null   int64
 3   Open        5036 non-null   float64
 4   High        5036 non-null   float64
 5   Low         5036 non-null   float64
 6   Company     5036 non-null   object
dtypes: float64(4), int64(1), object(2)
memory usage: 314.8+ KB
```

In [21]:

```
df['day']=pd.to_datetime(df['Date']).dt.day
df['month']=pd.to_datetime(df['Date']).dt.month
df['year']=pd.to_datetime(df['Date']).dt.year
df['quarter']=pd.to_datetime(df['Date']).dt.quarter
```

In [25]:

```
df.head()
```

Out[25]:

| | Date | Close/Last | Volume | Open | High | Low | Company | day | month | year | quarter |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 04/24/2024 | 169.02 | 48251840 | 166.540 | 169.30 | 166.210 | Apple Inc | 24 | 4 | 2024 | 2 |
| 1 | 04/23/2024 | 166.90 | 49537760 | 165.350 | 167.05 | 164.920 | Apple Inc | 23 | 4 | 2024 | 2 |
| 2 | 04/22/2024 | 165.84 | 48116440 | 165.515 | 167.26 | 164.770 | Apple Inc | 22 | 4 | 2024 | 2 |
| 3 | 04/19/2024 | 165.00 | 68149380 | 166.210 | 166.40 | 164.075 | Apple Inc | 19 | 4 | 2024 | 2 |
| 4 | 04/18/2024 | 167.04 | 43122900 | 168.030 | 168.64 | 166.550 | Apple Inc | 18 | 4 | 2024 | 2 |

In [28]:

```
df.to_csv("C:\Drive D\Study\Master's\Sem 4\Cloud Management\project\cleaned_stock_data.csv", index=False)
```

## Data Set Links:

https://www.nasdaq.com/market-activity/stocks/aapl/historical

https://www.nasdaq.com/market-activity/stocks/msft/historical

https://www.nasdaq.com/market-activity/stocks/qcom/historical

https://www.nasdaq.com/market-activity/stocks/amd/historical

## 10. References:

Amazon Web Services, Inc. (2024). Amazon S3.
https://aws.amazon.com/s3/

Amazon Web Services, Inc. (2024). Amazon Athena.
https://aws.amazon.com/athena/

Amazon Web Services, Inc. (2024). AWS Glue.
https://aws.amazon.com/glue/

Amazon Web Services, Inc. (2024). Amazon QuickSight. Retrieved April 28, 2024
https://aws.amazon.com/quicksight/?amazon-quicksight-whats-new.sort-by=item.additionalFields.postDateTime&amazon-quicksight-whats-new.sort-order=desc