Name: ARPIT DHOTE
UNID : UMID29062547419

Project Title	Stock Market
Tools	Python, SQL, Power BI
Domain	Data Analysis
Project Difficulties level	intermediate

About Dataset
Given historical stock price data for Apple, Microsoft, Netflix and Google over the past three months, your task is to analyze and compare the performance of these companies in the stock market using various data science techniques. Specifically, the goal is to identify trends and patterns in stock price movements, calculate moving averages and volatility for each company, and conduct correlation analysis to examine the relationships between different stock prices

Project Overview
Objective: To analyze market trends and predict future market behavior using machine learning techniques.

Steps to Follow:
1. Define the Scope and Objective:
○ Identify the market or industry you want to analyze.
○ Define the specific objectives of your analysis (e.g., predicting market growth, understanding consumer behavior, etc.).
2. Data Collection:
○ Gather relevant data from various sources (e.g., financial reports, market research reports, government databases, etc.).

○ Commondata points include market size, market share, growth rates, consumer demographics, competitive analysis, etc.

3. Data Preparation:

○ Clean the data to remove any inconsistencies or errors.
○ Combine data from different sources into a single dataset.
○ Usetools like Pandas for data cleaning and preparation.

4.Exploratory Data Analysis (EDA):

○ Perform EDA to understand the data distribution and identify patterns.
○ Use visualization tools like Matplotlib and Seaborn to visualize the data.

step1:Import Required Libraries

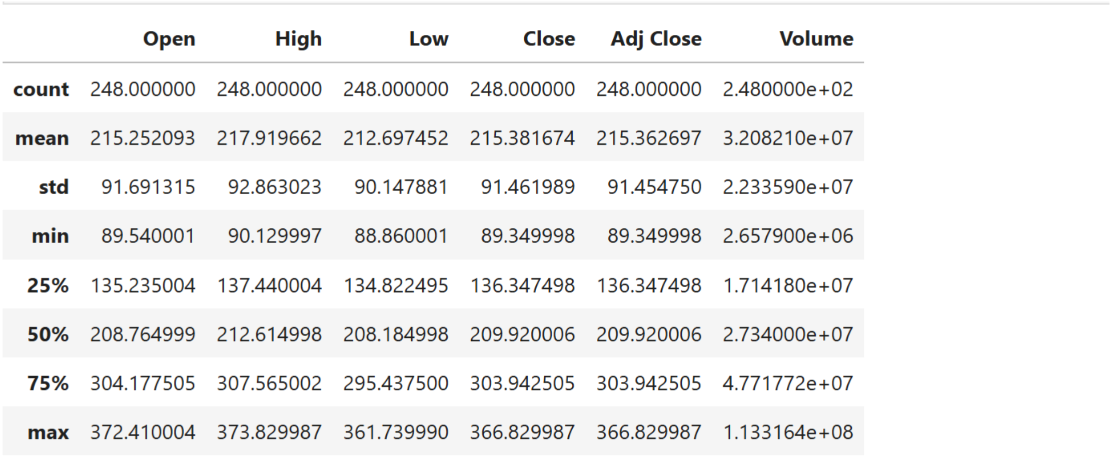
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
```

Step2: load dataset

```
df=pd.read_csv("C:/Users/nihal/OneDrive/Documents/jupyter/jupyter notebook/unified mentor/stock market/stocks.csv")
print(df.head())
```

Step3:Data cleaning

```
print(df.isnull().sum())
print(df.dtypes)
df.describe()
```

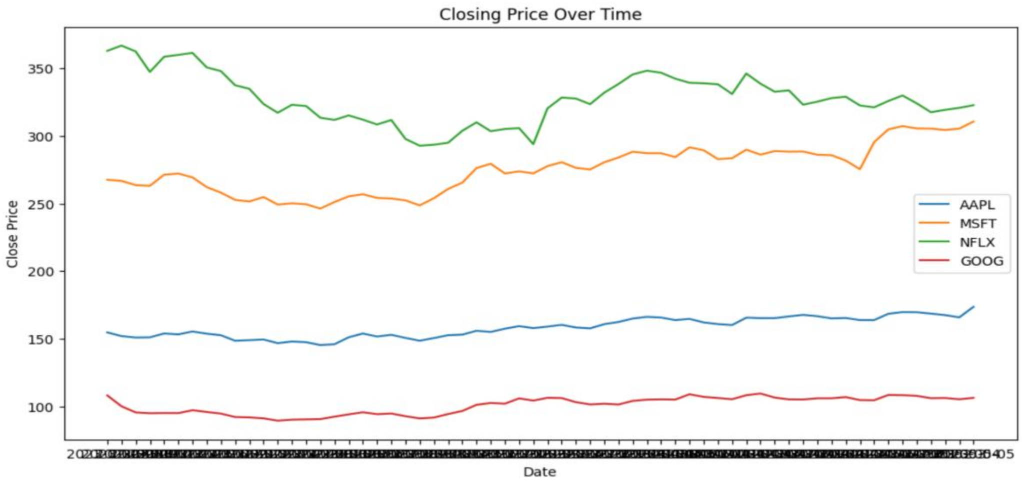


	Open	High	Low	Close	Adj Close	Volume
count	248.000000	248.000000	248.000000	248.000000	248.000000	2.480000e+02
mean	215.252093	217.919662	212.697452	215.381674	215.362697	3.208210e+07
std	91.691315	92.863023	90.147881	91.461989	91.454750	2.233590e+07
min	89.540001	90.129997	88.860001	89.349998	89.349998	2.657900e+06
25%	135.235004	137.440004	134.822495	136.347498	136.347498	1.714180e+07
50%	208.764999	212.614998	208.184998	209.920006	209.920006	2.734000e+07
75%	304.177505	307.565002	295.437500	303.942505	303.942505	4.771772e+07
max	372.410004	373.829987	361.739990	366.829987	366.829987	1.133164e+08

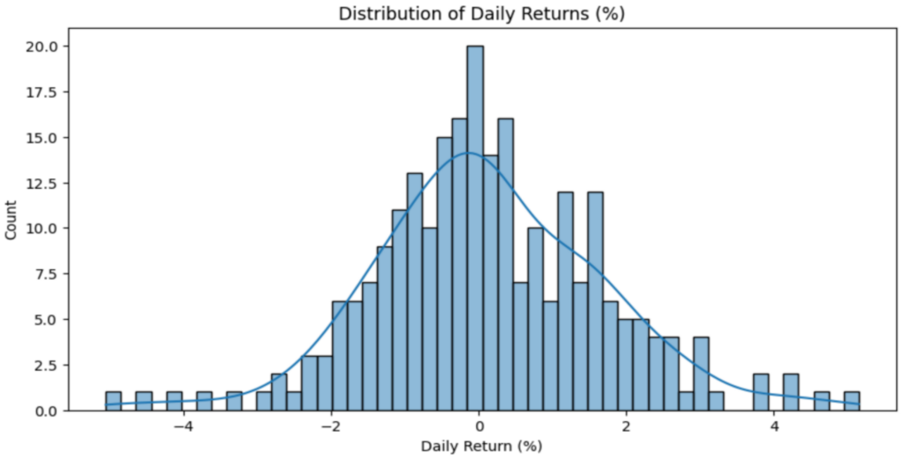
```
 df['Daily_Return_%'] = (df['Close'] - df['Open']) / df['Open'] * 100

 # Average Price
 df['Average_Price'] = (df['High'] + df['Low'] + df['Close']) / 3

 # Candle Type
 df['Candle_Type'] = np.where(df['Close'] > df['Open'], "Bullish",
"Bearish")

 df['MA7'] = df.groupby('Ticker')['Close'].transform(lambda x:
x.rolling(7).mean())
 df['MA14'] = df.groupby('Ticker')['Close'].transform(lambda x:
x.rolling(14).mean())
 df['MA30'] = df.groupby('Ticker')['Close'].transform(lambda x:
x.rolling(30).mean())

 print("\nUnique Tickers:", df['Ticker'].unique())
```

Output:Unique Tickers: ['AAPL' 'MSFT' 'NFLX' 'GOOG']

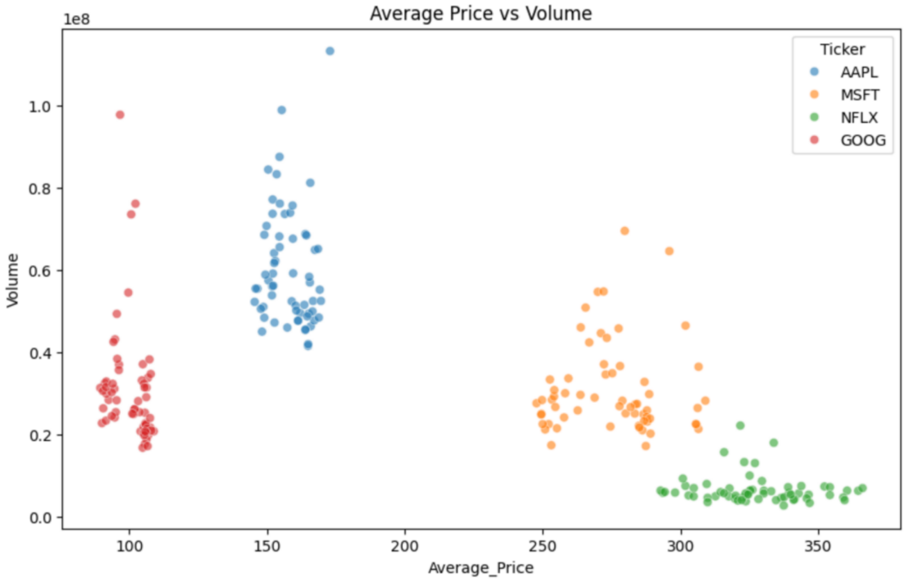
```
# 6.1 Closing Price over Time
plt.figure(figsize=(12,6))
for ticker in df['Ticker'].unique():
    subset = df[df['Ticker'] == ticker]
    plt.plot(subset['Date'], subset['Close'], label=ticker)
plt.title("Closing Price Over Time")
plt.xlabel("Date")
plt.ylabel("Close Price")
plt.legend()
plt.show()
```



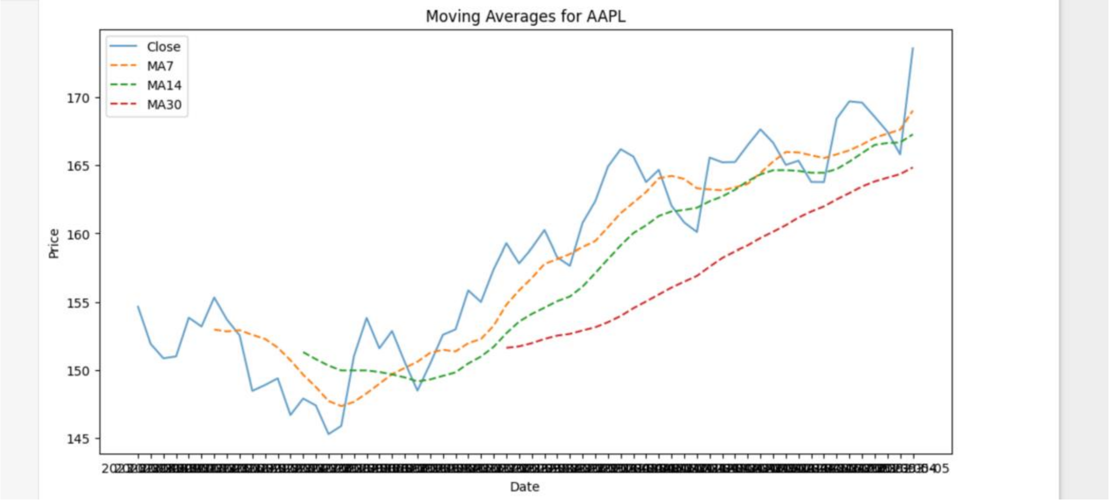
```
# 6.2 Daily Returns Distribution
plt.figure(figsize=(10,5))
sns.histplot(df['Daily_Return_%'], bins=50, kde=True)
plt.title("Distribution of Daily Returns (%)")
plt.xlabel("Daily Return (%)")
plt.show()
```



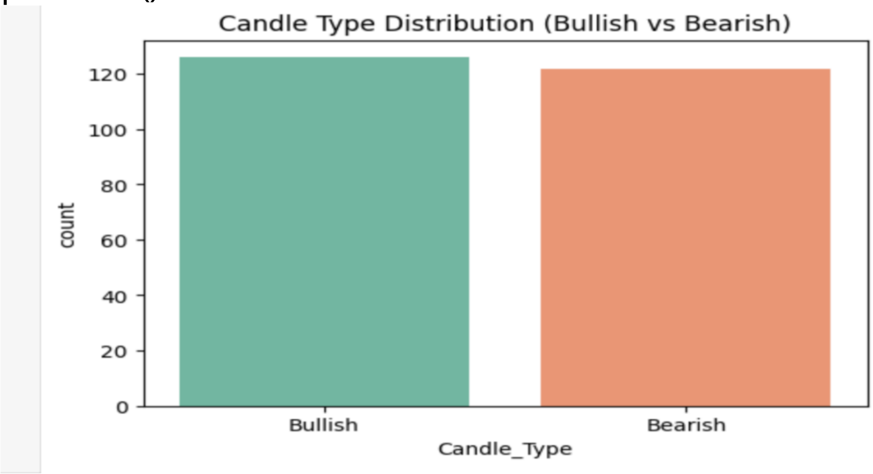
```
# 6.3 Average Price vs Volume
plt.figure(figsize=(10,6))
sns.scatterplot(data=df, x="Average_Price", y="Volume", hue="Tick
er", alpha=0.6)
plt.title("Average Price vs Volume")
plt.show()
```



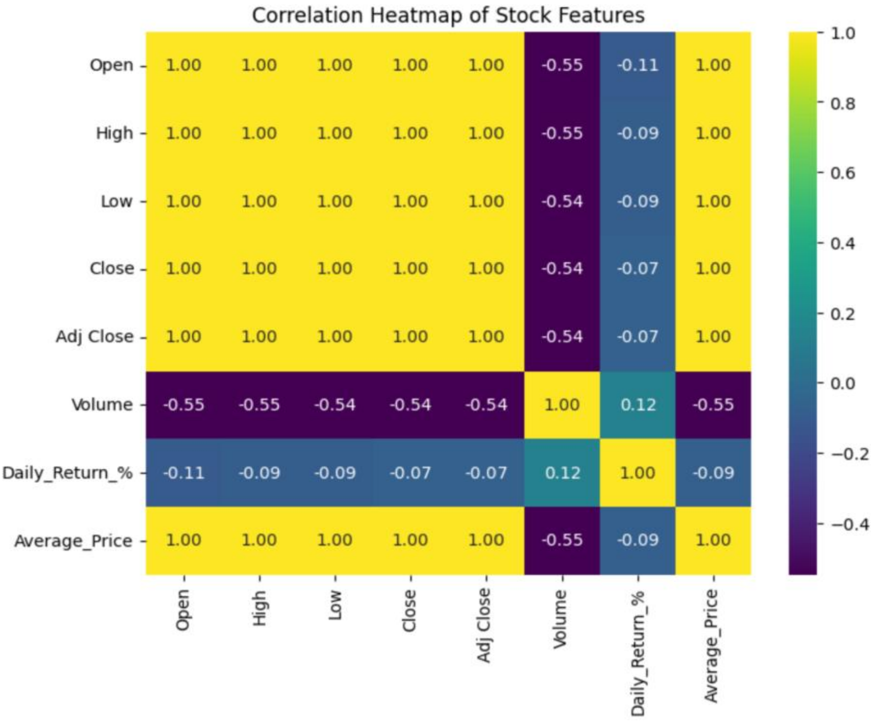
```
# 6.4 Moving Averages Example (for first ticker)
plt.figure(figsize=(12,6))
ticker = df['Ticker'].unique()[0]
subset = df[df['Ticker'] == ticker]
plt.plot(subset['Date'], subset['Close'], label="Close", alpha=0.7)
plt.plot(subset['Date'], subset['MA7'], label="MA7", linestyle="--")
plt.plot(subset['Date'], subset['MA14'], label="MA14", linestyle="--")
plt.plot(subset['Date'], subset['MA30'], label="MA30", linestyle="--")
plt.title(f"Moving Averages for {ticker}")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.show()
```



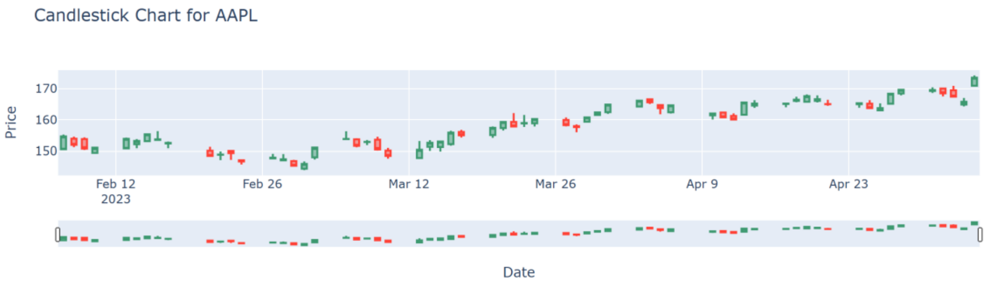
```
# 6.5 Candle Type Count
plt.figure(figsize=(6,4))
sns.countplot(data=df, x="Candle_Type", palette="Set2")
plt.title("Candle Type Distribution (Bullish vs Bearish)")
plt.show()
```



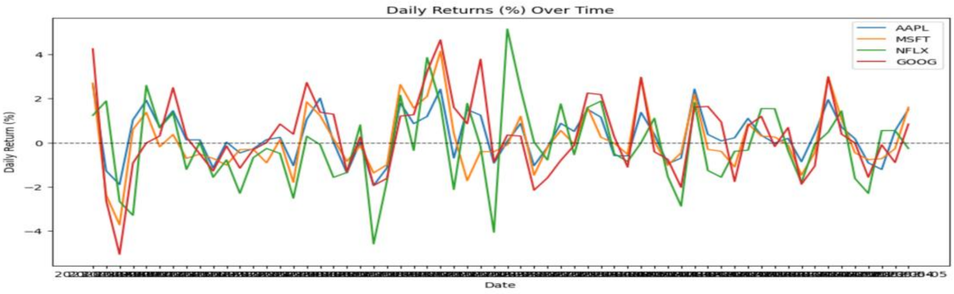
```
 plt.figure(figsize=(8,6))
 sns.heatmap(df[['Open','High','Low','Close','Adj
Close','Volume','Daily_Return_%','Average_Price']].corr(),
            annot=True, cmap="viridis", fmt=".2f")
 plt.title("Correlation Heatmap of Stock Features")
 plt.show()
```



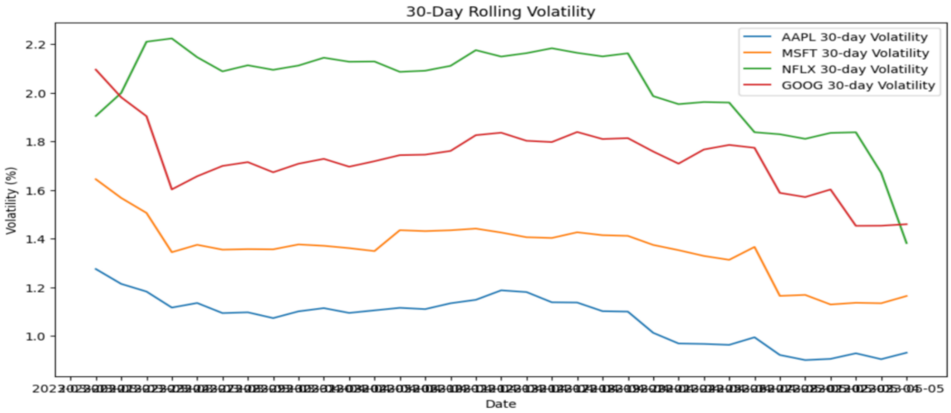
```
  # 2. Candlestick Chart (Plotly)
  ticker = df['Ticker'].unique()[0]
  subset = df[df['Ticker'] == ticker]
  fig = go.Figure(data=[go.Candlestick(
      x=subset['Date'],
      open=subset['Open'],
      high=subset['High'],
      low=subset['Low'],
      close=subset['Close'],
      name=ticker)])
  fig.update_layout(title=f"Candlestick Chart for {ticker}",
xaxis_title="Date", yaxis_title="Price")
  fig.show()
```



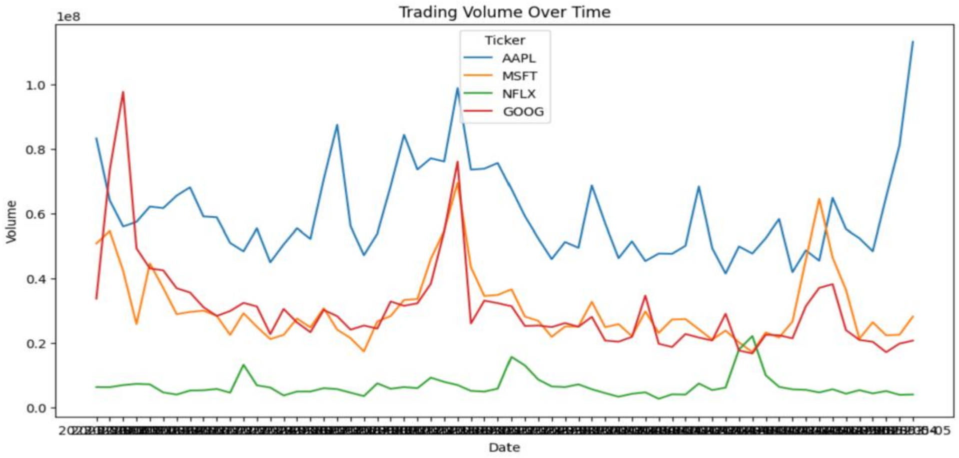
```
#3. Daily Returns Over Time
  plt.figure(figsize=(12,6))
  for ticker in df['Ticker'].unique():
      subset = df[df['Ticker']==ticker]
      plt.plot(subset['Date'], subset['Daily_Return_%'], label=ticker)
  plt.axhline(0, color='black', linestyle='--', linewidth=0.8)
  plt.title("Daily Returns (%) Over Time")
  plt.xlabel("Date")
  plt.ylabel("Daily Return (%)")
  plt.legend()
  plt.show()
```



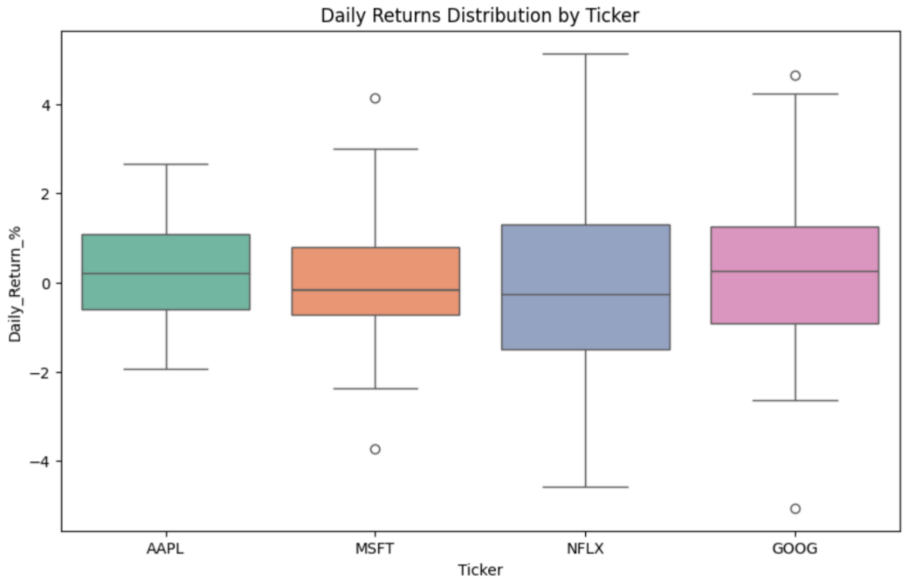
```
 # 4. Rolling Volatility (30-day STD of returns)
 df['Volatility30'] =
df.groupby('Ticker')['Daily_Return_%'].transform(lambda x:
x.rolling(30).std())
plt.figure(figsize=(12,6))
 for ticker in df['Ticker'].unique():
     subset = df[df['Ticker']==ticker]
     plt.plot(subset['Date'], subset['Volatility30'], label=f"{ticker} 30-
day Volatility")
 plt.title("30-Day Rolling Volatility")
 plt.xlabel("Date")
 plt.ylabel("Volatility (%)")
 plt.legend()
 plt.show()
```



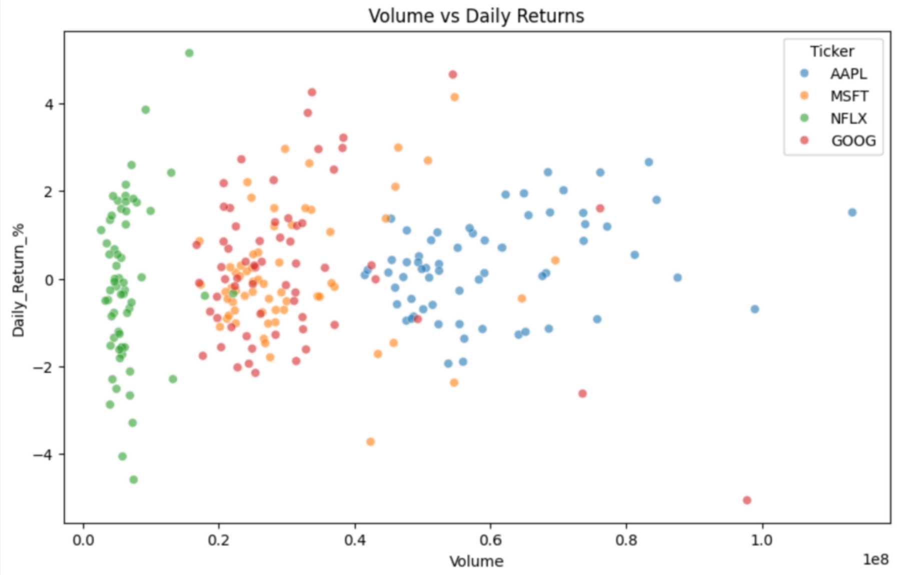
```
 # 5. Volume Over Time
 plt.figure(figsize=(12,6))
 sns.lineplot(data=df, x="Date", y="Volume", hue="Ticker")
 plt.title("Trading Volume Over Time")
 plt.show()
```



```
 # 6. Return Distribution by Ticker
 plt.figure(figsize=(10,6))
 sns.boxplot(data=df, x="Ticker", y="Daily_Return_%",
palette="Set2")
 plt.title("Daily Returns Distribution by Ticker")
 plt.show()
```

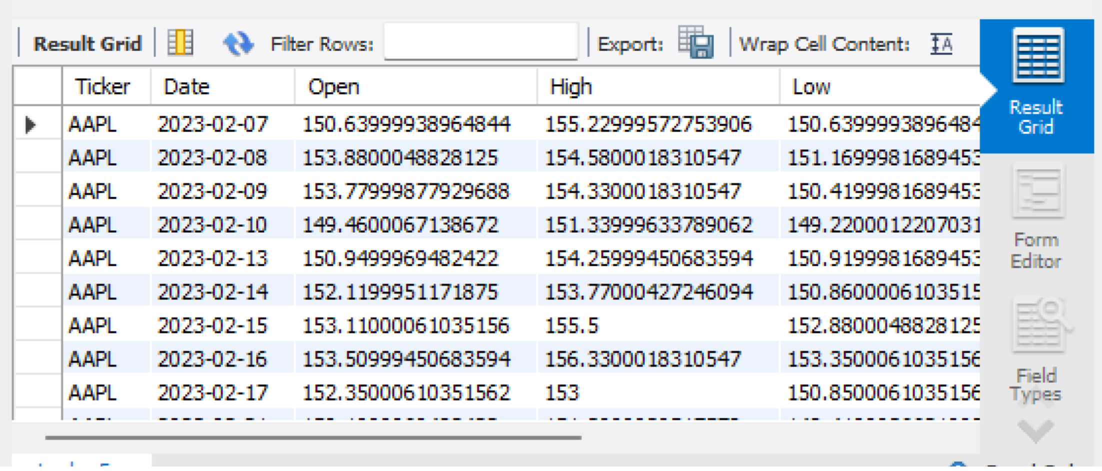


```
  #7. Scatter Plot: Volume vs Daily Returns
 plt.figure(figsize=(10,6))
 sns.scatterplot(data=df, x="Volume", y="Daily_Return_%",
hue="Ticker", alpha=0.6)
 plt.title("Volume vs Daily Returns")
 plt.show()
```



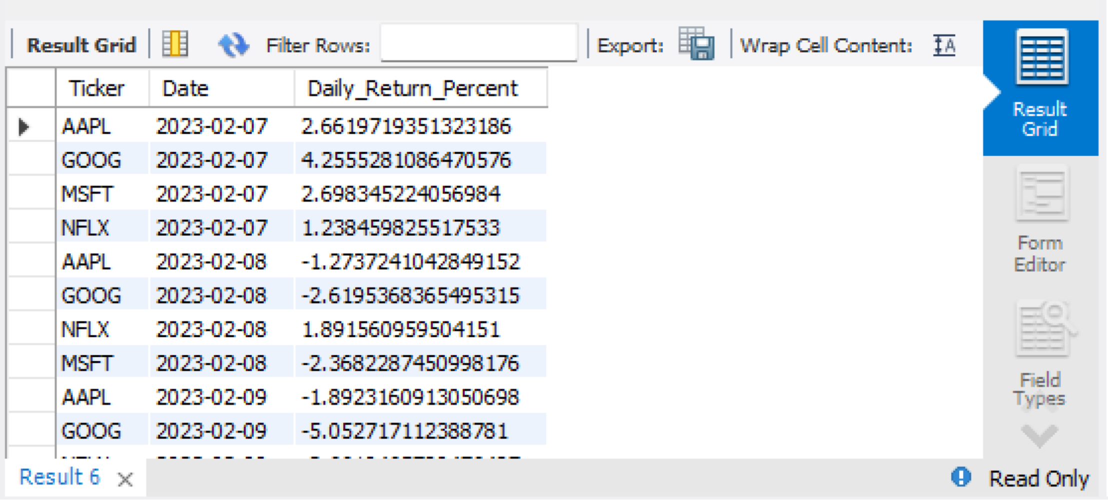
SQL Queries

```
-- View all data for a specific stock
SELECT *
FROM stocks
WHERE Ticker = 'AAPL'
ORDER BY Date ASC;
```



Ticker	Date	Open	High	Low
AAPL	2023-02-07	150.63999938964844	155.22999572753906	150.6399993896484
AAPL	2023-02-08	153.8800048828125	154.5800018310547	151.1699981689453
AAPL	2023-02-09	153.77999877929688	154.3300018310547	150.4199981689453
AAPL	2023-02-10	149.4600067138672	151.33999633789062	149.2200012207031
AAPL	2023-02-13	150.9499969482422	154.25999450683594	150.9199981689453
AAPL	2023-02-14	152.1199951171875	153.77000427246094	150.8600006103515
AAPL	2023-02-15	153.1100006103516	155.5	152.8800048828125
AAPL	2023-02-16	153.50999450683594	156.3300018310547	153.3500061035156
AAPL	2023-02-17	152.35000610351562	153	150.8500061035156

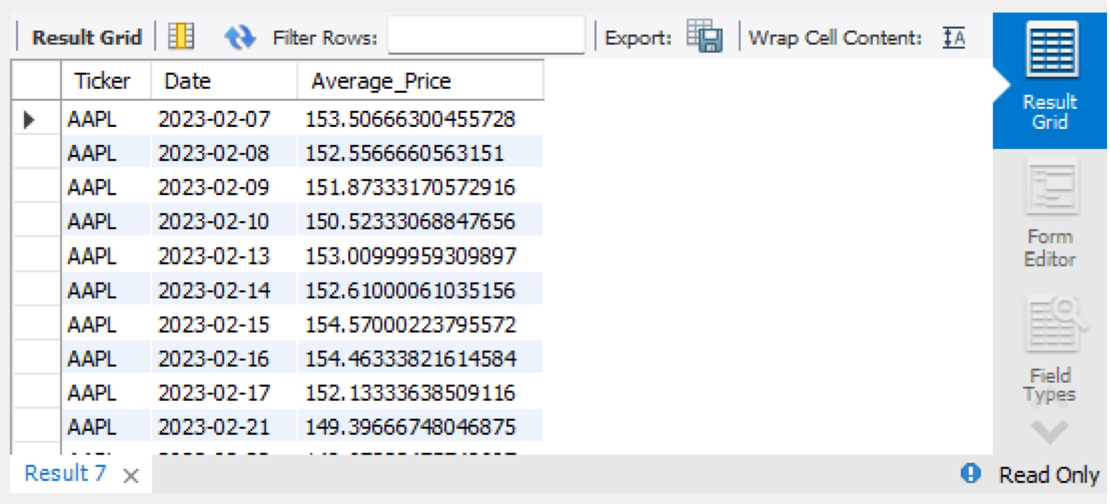
```
-- Calculate Daily Return (%)
SELECT
    Ticker,
    Date,
    ((Close - Open) / Open) * 100 AS Daily_Return_Percent
FROM stocks
ORDER BY Date;
```



Ticker	Date	Daily_Return_Percent
AAPL	2023-02-07	2.6619719351323186
GOOG	2023-02-07	4.2555281086470576
MSFT	2023-02-07	2.698345224056984
NFLX	2023-02-07	1.238459825517533
AAPL	2023-02-08	-1.2737241042849152
GOOG	2023-02-08	-2.6195368365495315
NFLX	2023-02-08	1.891560959504151
MSFT	2023-02-08	-2.3682287450998176
AAPL	2023-02-09	-1.8923160913050698
GOOG	2023-02-09	-5.052717112388781

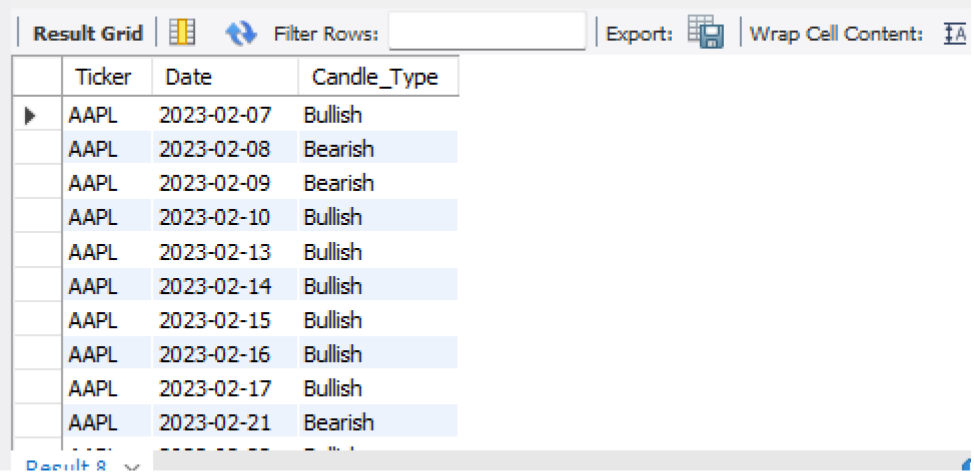
```
-- Calculate Average Price (OHLC average)
SELECT
```

```
    Ticker,
    Date,
    (High + Low + Close) / 3 AS Average_Price
FROM stocks;
```



Ticker	Date	Average_Price
AAPL	2023-02-07	153.50666300455728
AAPL	2023-02-08	152.5566660563151
AAPL	2023-02-09	151.87333170572916
AAPL	2023-02-10	150.52333068847656
AAPL	2023-02-13	153.00999959309897
AAPL	2023-02-14	152.61000061035156
AAPL	2023-02-15	154.57000223795572
AAPL	2023-02-16	154.46333821614584
AAPL	2023-02-17	152.13333638509116
AAPL	2023-02-21	149.39666748046875

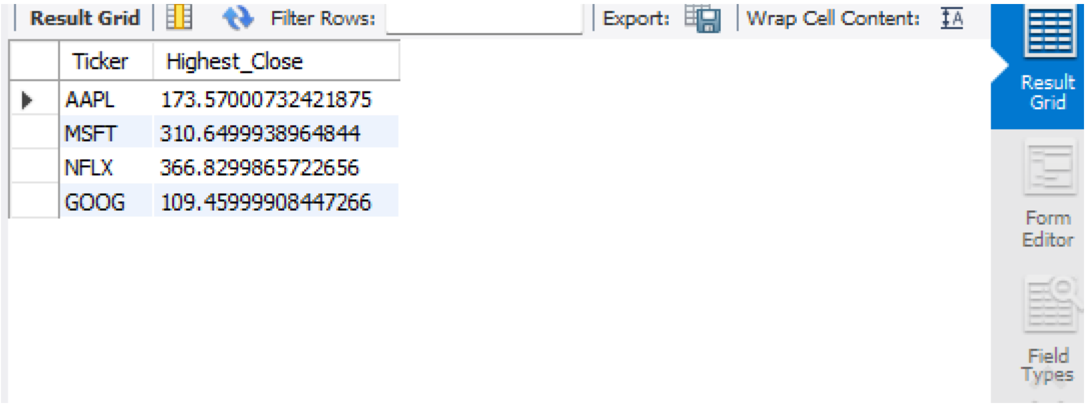
```
-- Classify Candle Type (Bullish / Bearish)
SELECT
    Ticker,
    Date,
    CASE
        WHEN Close > Open THEN 'Bullish'
        ELSE 'Bearish'
    END AS Candle_Type
FROM stocks;
```



Ticker	Date	Candle_Type
AAPL	2023-02-07	Bullish
AAPL	2023-02-08	Bearish
AAPL	2023-02-09	Bearish
AAPL	2023-02-10	Bullish
AAPL	2023-02-13	Bullish
AAPL	2023-02-14	Bullish
AAPL	2023-02-15	Bullish
AAPL	2023-02-16	Bullish
AAPL	2023-02-17	Bullish
AAPL	2023-02-21	Bearish

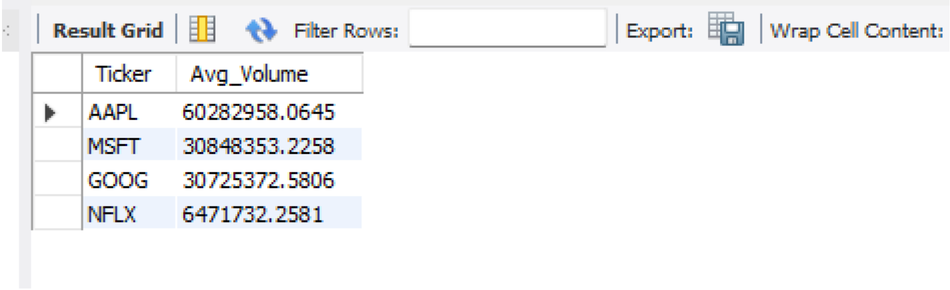
```
-- Get the Highest Closing Price per Stock
SELECT
    Ticker,
    MAX(Close) AS Highest_Close
FROM stocks
```

```
GROUP BY Ticker;
```



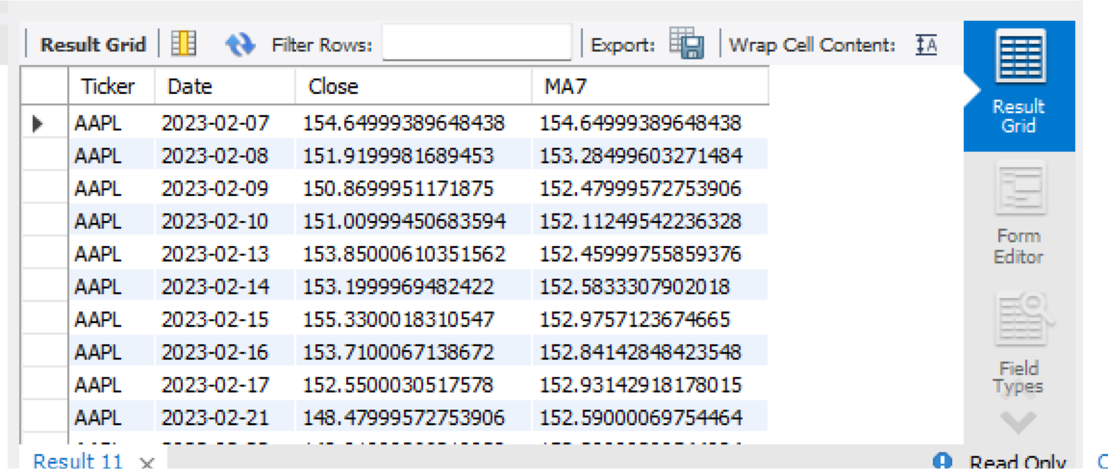
Ticker	Highest_Close
AAPL	173.57000732421875
MSFT	310.6499938964844
NFLX	366.8299865722656
GOOG	109.45999908447266

```
-- Get Average Daily Volume per Stock

SELECT
    Ticker,
    AVG(Volume) AS Avg_Volume
FROM stocks
GROUP BY Ticker
ORDER BY Avg_Volume DESC;
```



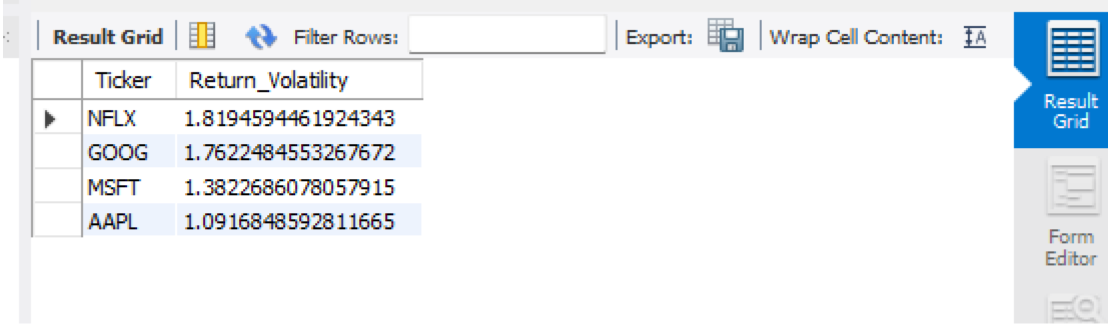
Ticker	Avg_Volume
AAPL	60282958.0645
MSFT	30848353.2258
GOOG	30725372.5806
NFLX	6471732.2581

```
-- Moving Average (7-day Close Price)
 SELECT
    Ticker,
    Date,
    Close,
    AVG(Close) OVER (PARTITION BY Ticker ORDER BY Date ROWS
BETWEEN 6 PRECEDING AND CURRENT ROW) AS MA7
 FROM stocks;
```



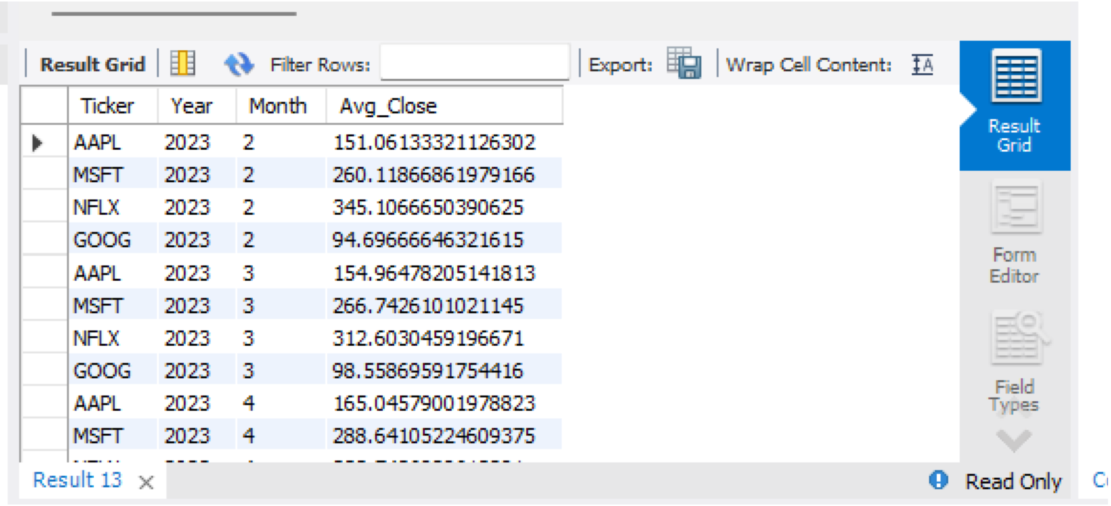
Ticker	Date	Close	MA7
AAPL	2023-02-07	154.64999389648438	154.64999389648438
AAPL	2023-02-08	151.9199981689453	153.28499603271484
AAPL	2023-02-09	150.8699951171875	152.47999572753906
AAPL	2023-02-10	151.00999450683594	152.11249542236328
AAPL	2023-02-13	153.85000610351562	152.45999755859376
AAPL	2023-02-14	153.1999969482422	152.5833307902018
AAPL	2023-02-15	155.3300018310547	152.9757123674665
AAPL	2023-02-16	153.7100067138672	152.84142848423548
AAPL	2023-02-17	152.5500030517578	152.93142918178015
AAPL	2023-02-21	148.47999572753906	152.59000069754464

```
-- Find Most Volatile Stocks (using STD of returns)
SELECT
    Ticker,
    STDDEV((Close - Open)/Open * 100) AS Return_Volatility
FROM stocks
GROUP BY Ticker
ORDER BY Return_Volatility DESC;
```



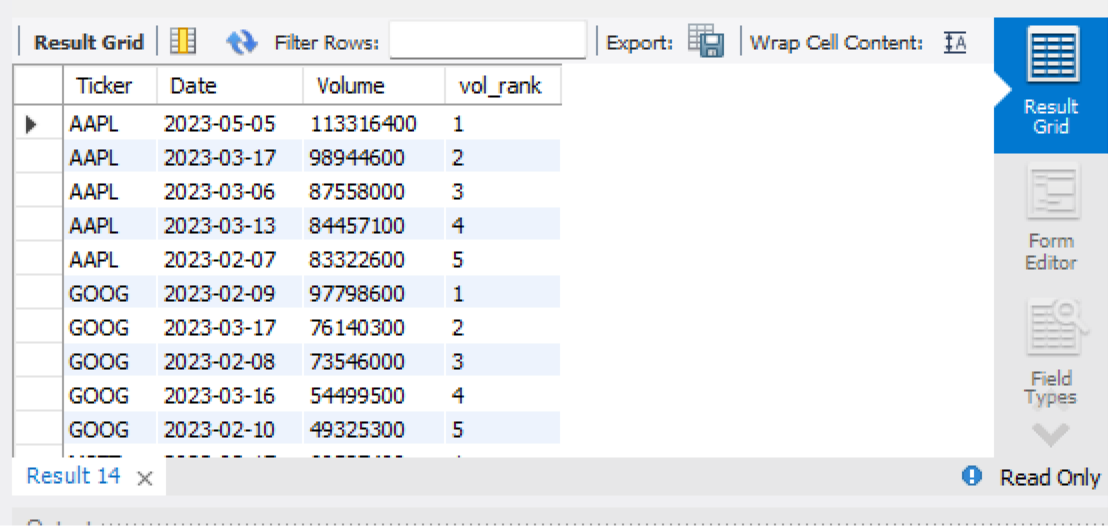
Ticker	Return_Volatility
NFLX	1.8194594461924343
GOOG	1.7622484553267672
MSFT	1.3822686078057915
AAPL	1.0916848592811665

```
-- Monthly Average Close Price
SELECT
    Ticker,
    YEAR(Date) AS Year,
    MONTH(Date) AS Month,
    AVG(Close) AS Avg_Close
FROM stocks
GROUP BY Ticker, YEAR(Date), MONTH(Date)
ORDER BY Year, Month;
```



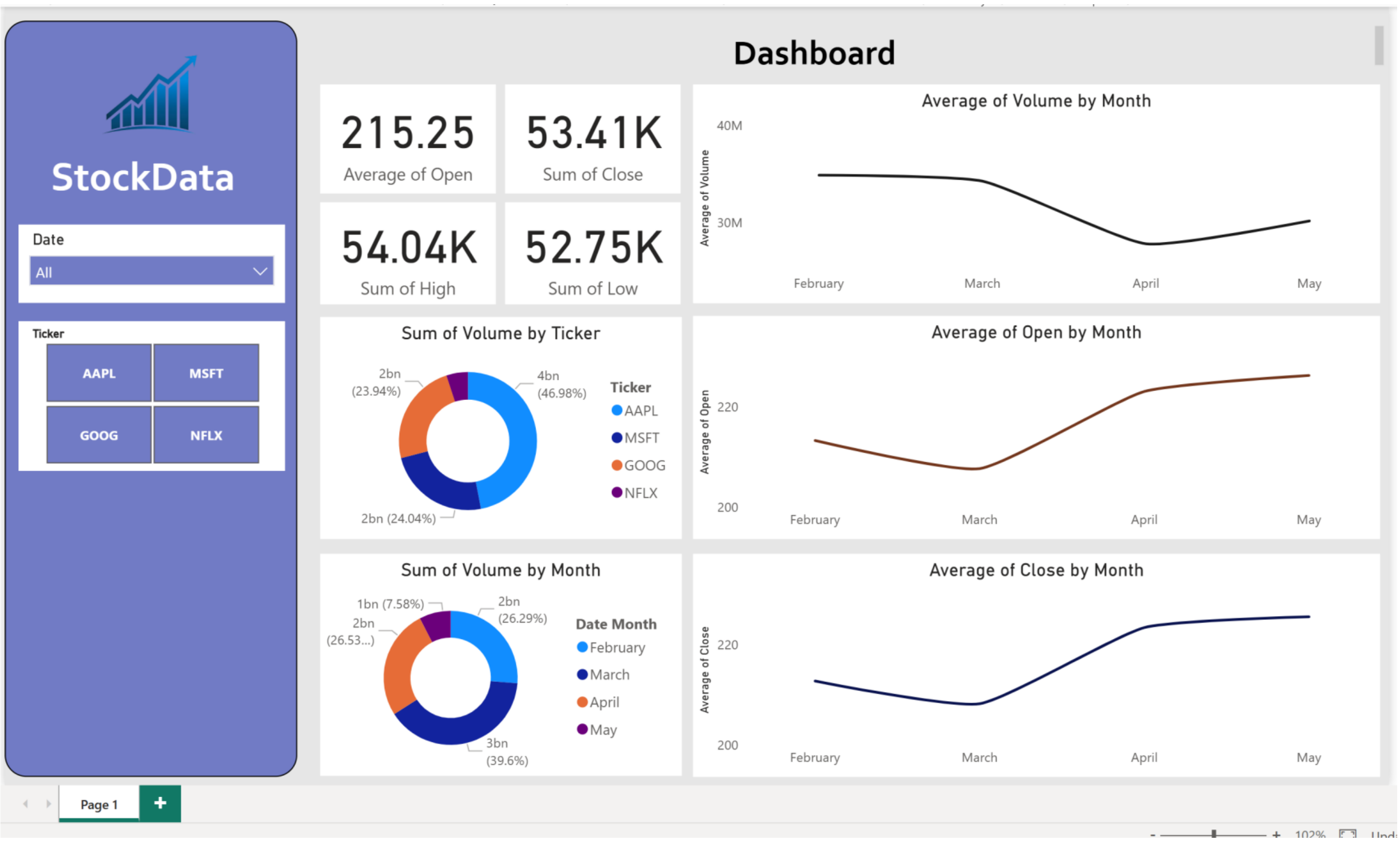
Ticker	Year	Month	Avg_Close
AAPL	2023	2	151.06133321126302
MSFT	2023	2	260.1186861979166
NFLX	2023	2	345.106650390625
GOOG	2023	2	94.69666646321615
AAPL	2023	3	154.96478205141813
MSFT	2023	3	266.7426101021145
NFLX	2023	3	312.6030459196671
GOOG	2023	3	98.55869591754416
AAPL	2023	4	165.04579001978823
MSFT	2023	4	288.64105224609375

```
-- Top 5 Days with Highest Trading Volume per Stock
SELECT *
FROM (
    SELECT
        Ticker,
        Date,
        Volume,
        RANK() OVER (PARTITION BY Ticker ORDER BY Volume DESC) AS
vol_rank
    FROM stocks
) ranked
WHERE vol_rank <= 5;
```



Ticker	Date	Volume	vol_rank
AAPL	2023-05-05	113316400	1
AAPL	2023-03-17	98944600	2
AAPL	2023-03-06	87558000	3
AAPL	2023-03-13	84457100	4
AAPL	2023-02-07	83322600	5
GOOG	2023-02-09	97798600	1
GOOG	2023-03-17	76140300	2
GOOG	2023-02-08	73546000	3
GOOG	2023-03-16	54499500	4
GOOG	2023-02-10	49325300	5

Power BI


StockData
Date
All
Ticker
AAPL
MSFT
GOOG
NFLX
Dashboard
215.25
Average of Open
53.41K
Sum of Close
54.04K
Sum of High
52.75K
Sum of Low
Average of Volume by Month
Average of Volume
40M
30M
February
March
April
May
Sum of Volume by Ticker
2bn (23.94%)
4bn (46.98%)
2bn (24.04%)
Ticker
● AAPL
● MSFT
● GOOG
● NFLX
Average of Open by Month
Average of Open
220
200
February
March
April
May
Sum of Volume by Month
1bn (7.58%)
2bn (26.53...)
2bn (26.29%)
3bn (39.6%)
Date Month
● February
● March
● April
● May
Average of Close by Month
Average of Close
220
200
February
March
April
May
Page 1
+
102%