# Project 1 - Team 3

Stock Market Analysis

# Our Team Members & GitHub Repository

**Team Members:** Yashada, Witness, Ben and Ned

**GitHub Repository:** https://github.com/Mono-Co/project-one-team3

# What is your research question?

We are working for a large equity-trading company, and have been tasked with researching for a client's portfolio. Your client wants to invest in a "Tech" stocks and needs expert analysis to make the right decision.

The primary objective of this project is to perform a comprehensive analysis of upto 6 Tech Companies on the Nasdaq stock market, use available historical data from the Nasdaq and make a recommendation.

Nasdaq

# Companies under analysis

The Project included a review of the following stocks over a 12 month period;

Google GOOG,

Meta (owner of Facebook) META,

Microsoft MSFT,

MicroStrategy MSTR,

Apple AAPL,

Monday MNDY.

# Stock Market Review Process

**1** **Obtain Data using NASDAQ API**

We used the API QUOTEMEDIA/PRICES.. This allows for a direct query of a 'Ticker' and retina in a Table format.

**Data Collection**

**2** **Clean and Re-Structure Data**

Using the Python tools we re-arranged the data to suit our analysis needs

**Data Cleaning**

**3** **Complete the Analysis**

Using the MatPlotLib tools as related charts / plots to visualize the analysis

**Analysis**

**4** **Make Recommendation**

Based on the analysis we were able to make a recommendation on the stock to the Client.

**Conclusion**

# Stock Market Review

**Outline of our project;**

Identifying the coding **dependencies**, our API to obtain the data was **nasdaqdatalinkfrom** Nasdaq. Also we incorporated matplotlib.pyplot for creating static plots & statistics for statistical analysis

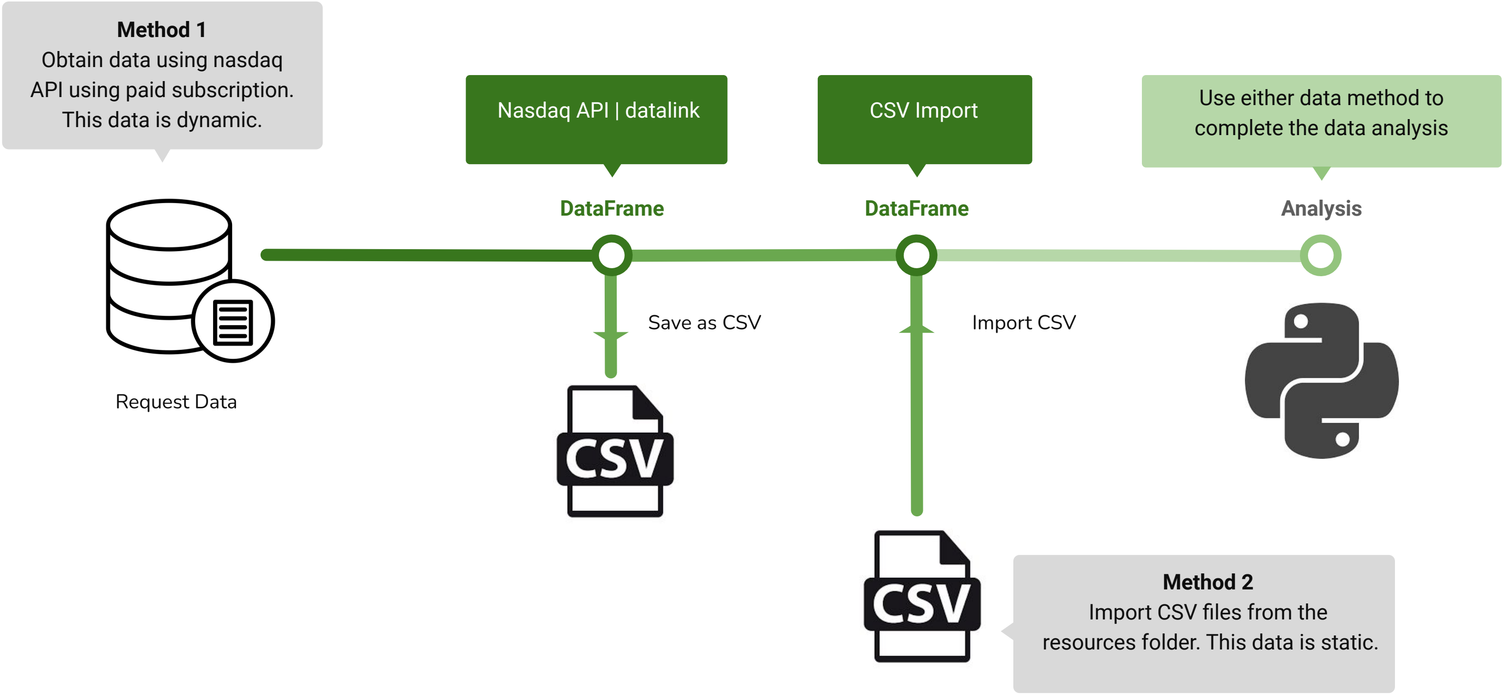
The stock data was initially obtained through the API, and following data **cleaning and processing** saved as CSV files for for re-importing as Pandas DataFrames.

To obtain insights into the performance of each stock, a thorough **analysis** of metrics such as open, high, low, close prices, and trading volume, moving averages was conducted. A historical view of the closing price for all stocks in the company list was generated. Also generated **charts & plots** are used to visualise and analyse the data.

This project offers insightful information about the stock market performance of the companies chosen from the Nasdaq. Investors can make well-informed decisions about their investing plans by using historical stock price data analysis and key metrics visualization.

# Data Collection

# We Used Two Broad Methods For Data



# Method | Dependencies

**Our project included the following setup and dependencies;**

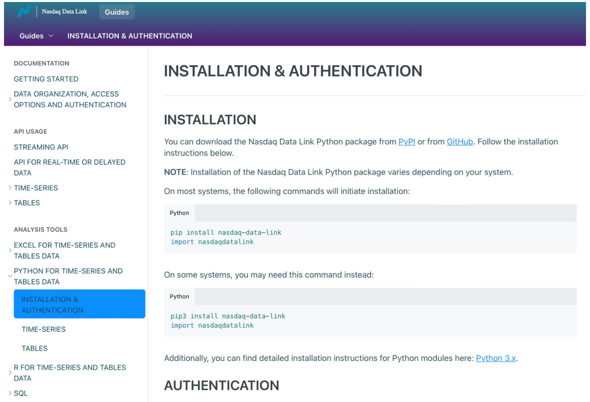
```
import hvplot.pandas
import pandas as pd
import nasdaqdatalink
import matplotlib.pyplot as plt
import statistics as st
import scipy.stats as stpy
from scipy.stats import linregress
import matplotlib.ticker as ticker
```

# Method | NASDAQ Data Link

**Our project included the NASDAQ DATALINK;**

The NASDAQ DATALINK is based on the QUOTEMEDIA/PRICES request using the NASDAQ API / SDK.

To install the NASDAQ DATA LINK we opened a new terminal in our Anaconda environments and ran the command '**pip install nasdaq-data-link**'. This installed the required software. A user needs to create an account on NASDAQ and generate an API key. For "**nasdaq-data-link"** the API is noted in the api_config.py file, therefore no API_Config.py file is provided in the GitHub Repository.



The basic NASDAQ DATA LINK code & response is per the below;

```
# Use Pandas Nasdaq Data Link to get stock data

META_stock = nasdaqdatalink.get_table('QUOTEMEDIA/PRICES',
                                qopts = { 'columns': ['ticker', 'date', 'open', 'high', 'low', 'close', 'volume'] },
                                ticker = ['META'],
                                date = { 'gte': '2023-04-01', 'lte': '2024-04-01' })

print(f"Downloaded {len(META_stock)} rows of data.")

# Display sample data
#META_stock.head()
```

```
Downloaded 250 rows of data.
```

| | None | ticker | date | open | high | low | close | volume |
|---|---|---|---|---|---|---|---|---|
| **0** | 249 | META | 2023-04-03 | 208.840 | 213.4861 | 208.200 | 213.07 | 17887238.0 |
| **1** | 248 | META | 2023-04-04 | 213.390 | 216.2400 | 212.540 | 214.72 | 20977958.0 |
| **2** | 247 | META | 2023-04-05 | 214.150 | 215.1900 | 209.940 | 211.48 | 19331864.0 |
| **3** | 246 | META | 2023-04-06 | 209.250 | 216.9400 | 208.650 | 216.10 | 26104411.0 |
| **4** | 245 | META | 2023-04-10 | 214.710 | 215.6600 | 210.660 | 214.75 | 15841652.0 |
| **5** | 244 | META | 2023-04-11 | 215.480 | 216.0200 | 213.410 | 213.85 | 16348387.0 |
| **6** | 243 | META | 2023-04-12 | 214.835 | 216.8400 | 212.584 | 214.00 | 18859583.0 |
| **7** | 242 | META | 2023-04-13 | 215.730 | 221.1500 | 215.690 | 220.35 | 23233212.0 |
| **8** | 241 | META | 2023-04-14 | 217.880 | 222.1100 | 217.550 | 221.49 | 21532908.0 |
| **9** | 240 | META | 2023-04-17 | 219.790 | 220.9790 | 217.130 | 218.86 | 15411724.0 |
| **10** | 239 | META | 2023-04-18 | 219.910 | 220.4400 | 216.210 | 217.89 | 12209744.0 |

# Data Cleaning

# Method | Data Manipulation

Our project we were  required to manipulate the data using Python functions such as;

- Sorting by date, the data from nasdaqdatalink required to be reversed and reset the index.
- Creating new data and identifying specific data;
    - Moving Averages 10 , 20 & 50 days.
    - Closing price (Mean, Media, Variance & Standard Deviation)
    - Daily Change
    - Daily Percent Change
    - Minimum Close Price
    - Maximum Close Price

The stock sorting by date code & response is per the below;

```
#Change Date Order
GOOG_date = GOOG_stock.sort_values(by="date").reset_index()
META_date = META_stock.sort_values(by="date").reset_index()
MSFT_date = MSFT_stock.sort_values(by="date").reset_index()
MSTR_date = MSTR_stock.sort_values(by="date").reset_index()
AAPL_date = AAPL_stock.sort_values(by="date").reset_index()
MNDY_date = MNDY_stock.sort_values(by="date").reset_index()
#TEAM_date = TEAM_stock.sort_values(by="date").reset_index()
```

```
#Test the changes with a sample data frame
META_date.head(11)
```

| | None | ticker | date | open | high | low | close | volume |
|---|---|---|---|---|---|---|---|---|
| **0** | 249 | META | 2023-04-03 | 208.840 | 213.4861 | 208.200 | 213.07 | 17887238.0 |
| **1** | 248 | META | 2023-04-04 | 213.390 | 216.2400 | 212.540 | 214.72 | 20977958.0 |
| **2** | 247 | META | 2023-04-05 | 214.150 | 215.1900 | 209.940 | 211.48 | 19331864.0 |
| **3** | 246 | META | 2023-04-06 | 209.250 | 216.9400 | 208.650 | 216.10 | 26104411.0 |
| **4** | 245 | META | 2023-04-10 | 214.710 | 215.6600 | 210.660 | 214.75 | 15841652.0 |
| **5** | 244 | META | 2023-04-11 | 215.480 | 216.0200 | 213.410 | 213.85 | 16348387.0 |
| **6** | 243 | META | 2023-04-12 | 214.835 | 216.8400 | 212.584 | 214.00 | 18859583.0 |
| **7** | 242 | META | 2023-04-13 | 215.730 | 221.1500 | 215.690 | 220.35 | 23233212.0 |
| **8** | 241 | META | 2023-04-14 | 217.880 | 222.1100 | 217.550 | 221.49 | 21532908.0 |
| **9** | 240 | META | 2023-04-17 | 219.790 | 220.9790 | 217.130 | 218.86 | 15411724.0 |
| **10** | 239 | META | 2023-04-18 | 219.910 | 220.4400 | 216.210 | 217.89 | 12209744.0 |

The stock data creating for Moving Average (10, 20 & 50 Days), Daily Change, Daily Percent Change, Average Trade Volume, code & response is per the below;

```
#Lets add the moving averages of the Stocks, for 10, 20 & 50 days to the data frames

ma_day = [10, 20, 50]

for ma in ma_day:
    for company in company_list:
        column_name = f"MA for {ma} days"
        company[column_name] = company['close'].rolling(ma).mean()

#Lets add the daily % change, Average Trade volume, Daily Change and Average Percent Cgange of the stocks to the
for company in company_list:
    company['Daily Percent Change'] = company['close'].pct_change()
    company['Average Trade Volume'] = company['volume'].mean()
    company['Daily Change'] = (company['close']-company['open'])
    company['Average Percent Change'] = company['Daily Percent Change'].mean()

#Test by checking a sample stock
AAPL_data.head(11)
```

| | Unnamed: 0 | None | ticker | date | open | high | low | close | volume | MA for 10 days | MA for 20 days | MA for 50 days | Daily Percent Change | Averag Trac Volun |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 249 | AAPL | 2023-04-03 | 164.270 | 166.2900 | 164.22 | 166.17 | 54893192.0 | NaN | NaN | NaN | NaN | 5.631651e+( |
| **1** | 1 | 248 | AAPL | 2023-04-04 | 166.595 | 166.8400 | 165.11 | 165.63 | 44435155.0 | NaN | NaN | NaN | -0.003250 | 5.631651e+( |

The stock data for Mean, Median, Variance and Standard Deviation on the stock Close code & response is per the below;

```
#Summary stats
data_close = pd.DataFrame({"GOOG": GOOG_stock_df["close"],
                           "META": META_stock_df["close"],
                           "MSFT": MSFT_stock_df["close"],
                           "MSTR": MSTR_stock_df["close"],
                           "AAPL": AAPL_stock_df["close"],
                           "MNDY": MNDY_stock_df["close"],
                           "TEAM":TEAM_stock_df["close"]})

aggregate = data_close.aggregate([st.mean, st.median, st.variance, st.stdev])

print(f"The summary statistics for closing prices of our data: Orange is max, green is min")
aggregate.style.highlight_max(axis=1, color = "orange").highlight_min(axis=1, color = "lightgreen")
```

```
The summary statistics for closing prices of our data: Orange is max, green is min
```

| | GOOG | META | MSFT | MSTR | AAPL | MNDY | TEAM |
|---|---|---|---|---|---|---|---|
| **mean** | 131.511420 | 329.996540 | 351.165940 | 518.510980 | 181.003000 | 172.628800 | 191.009080 |
| **median** | 133.235000 | 311.715000 | 337.855000 | 407.520000 | 180.730000 | 172.535000 | 190.610000 |
| **variance** | 148.137678 | 6359.356429 | 1489.332890 | 117372.845809 | 83.995385 | 895.243658 | 711.834533 |
| **stdev** | 12.171182 | 79.745573 | 38.591876 | 342.597206 | 9.164900 | 29.920623 | 26.680227 |

**Note:** Pandas dataframes can be stylised using the .style function.

# Analysis

# Analysis | Charting & Visualisation

Our project includes visualising data using line charts, box plots, bar charts and scatter plots.

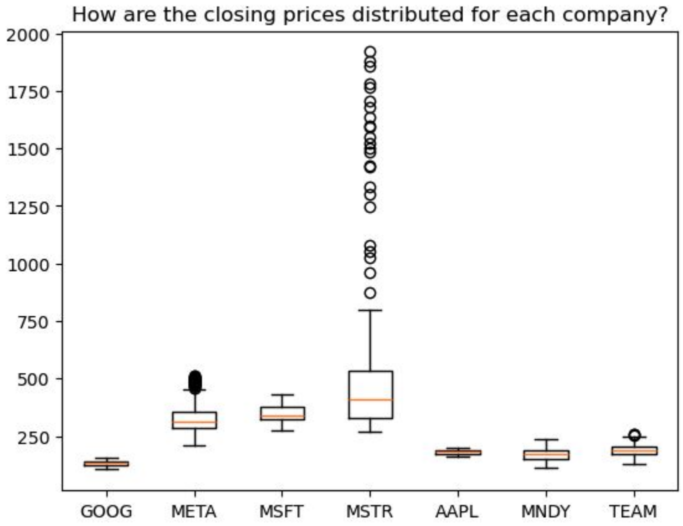
- Plotting the Closing Price of each company
- Plotting the Closing Price of each stock on a single Chart
- Plotting the Volume of Sales
- Plot 10 days, 20 days, 30 days Moving Average of Stocks
- Plot percentage of Daily Return of the shares
- Plot Correlation between recommended stocks closing price and trading volume, including Linear Regression

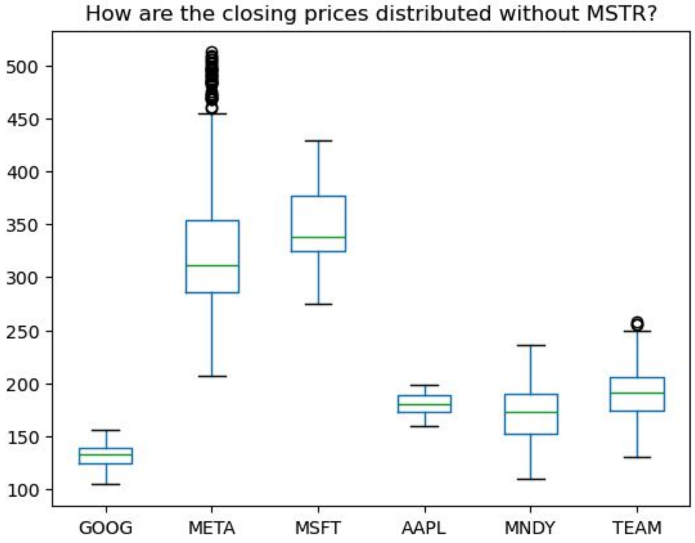
# Closing Price Analysis

The closing price is the last price at which a security traded during the regular trading day. A security's closing price is the standard benchmark used by investors to track its performance over time.

Source: https://www.investopedia.com/terms/c/closingprice.asp

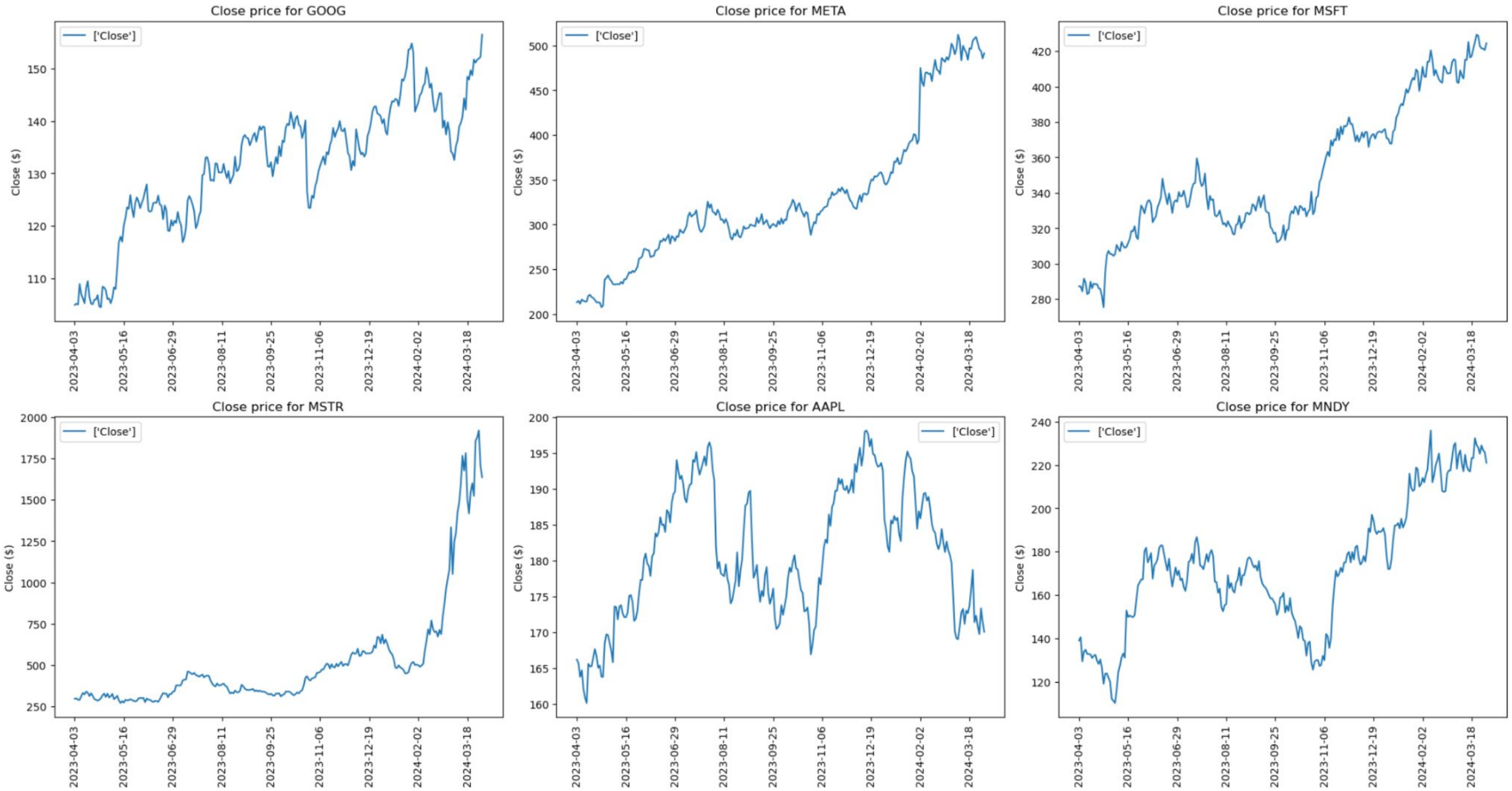
Visualising the spread of closing prices for each company.



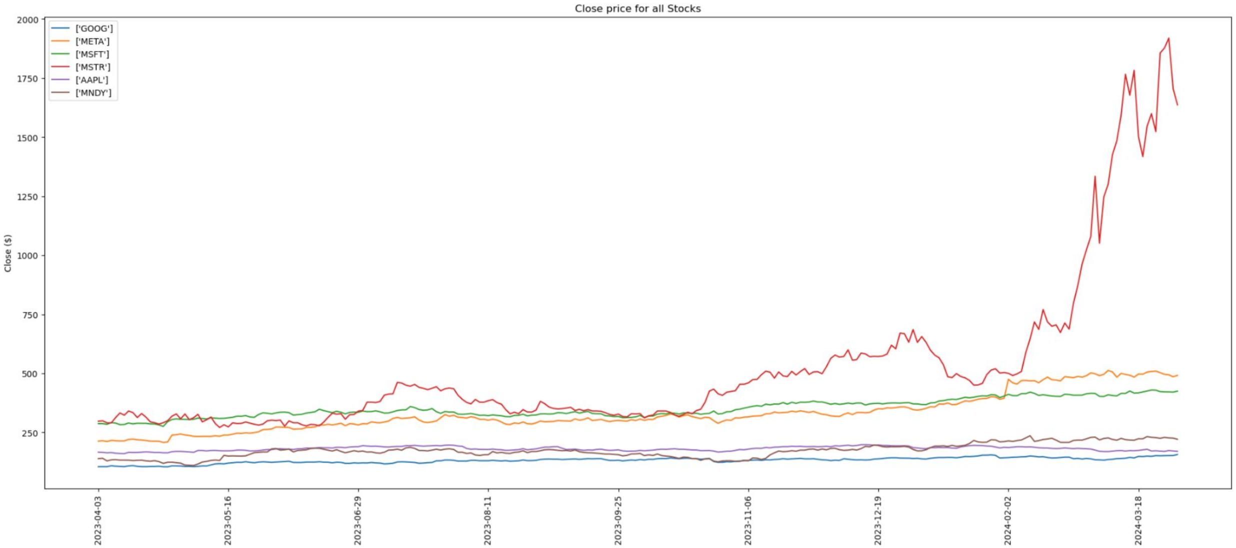

**Observation:** The summary stats and boxplots show us that AAPL stock remains predictable with least std dev and least movement while MSTR shows max std dev and a lot of movement throughout the year, followed by META

Plotting the Closing Price of each company  response is per the below;




**Observation:** All stocks have had positive increases over the 12 month review.

Plotting the Closing Price of each company  response is per the below;




| Company | 52W Open Stock Price | 52W Highest Stock Price | 52W Lowest Stock Price | 52W Close Stock Price | 52W Stock Change ($) | 52W Stock Change Percent (%) |
|---|---|---|---|---|---|---|
| GOOG | 102.67 | 157.00 | 102.380 | 156.50 | 53.83 | 52.430116 |
| META | 208.84 | 523.57 | 207.130 | 491.35 | 282.51 | 135.275809 |
| MSFT | 286.52 | 430.82 | 275.370 | 424.57 | 138.05 | 48.181628 |
| MSTR | 290.99 | 1999.99 | 266.000 | 1636.74 | 1345.75 | 462.472937 |
| AAPL | 164.27 | 199.62 | 159.780 | 170.03 | 5.76 | 3.506422 |
| MNDY | 140.16 | 239.22 | 108.345 | 221.00 | 80.84 | 57.676941 |

We extracted data for max closing price per company:

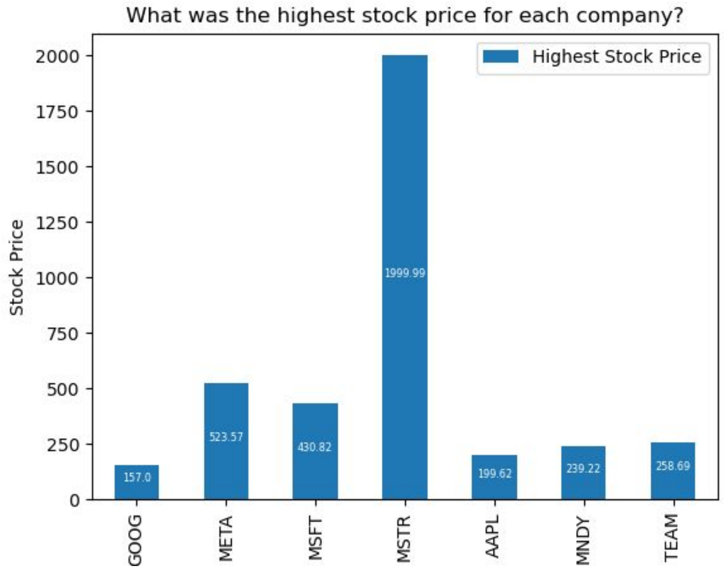
```
[28]: #Create a dataframe with only highest stock price and volume traded:
company_high = []
company_maxvol = []
for company in company_list:
    company_high.append(company["high"].max())
    company_maxvol.append(company["volume"].max())

stockprice_volume_df = pd.DataFrame({"Company": company_name,
             "Highest Stock Price": company_high,
             "Max Volume": company_maxvol})
stockprice_volume_df
```

[28]:

| | Company | Highest Stock Price | Max Volume |
|---|---|---|---|
| 0 | GOOG | 157.00 | 58456507.0 |
| 1 | META | 523.57 | 84391922.0 |
| 2 | MSFT | 430.82 | 72300798.0 |
| 3 | MSTR | 1999.99 | 5635349.0 |
| 4 | AAPL | 199.62 | 135399206.0 |
| 5 | MNDY | 239.22 | 5083390.0 |
| 6 | TEAM | 258.69 | 9307594.0 |

We then plotted the max closing price for each company;

```
#Create a bar plot for highest stock price:

def addlabels(x,y):
    for i in range(len(x)):
        plt.text(i,y[i]//2,y[i], ha = 'center', color = 'white', fontsize = 'xx-small')

x = stockprice_volume_df["Company"]
y = stockprice_volume_df["Highest Stock Price"]

stockprice_volume_df.plot(kind = "bar", x = "Company", y = "Highest Stock Price")
plt.title("What was the highest stock price for each company?")
plt.ylabel("Stock Price")
addlabels(x,y)

plt.savefig("./output_data/stock_price_bar.png")
plt.show()
```




**Observation:** MSTR seems to have the max stock price

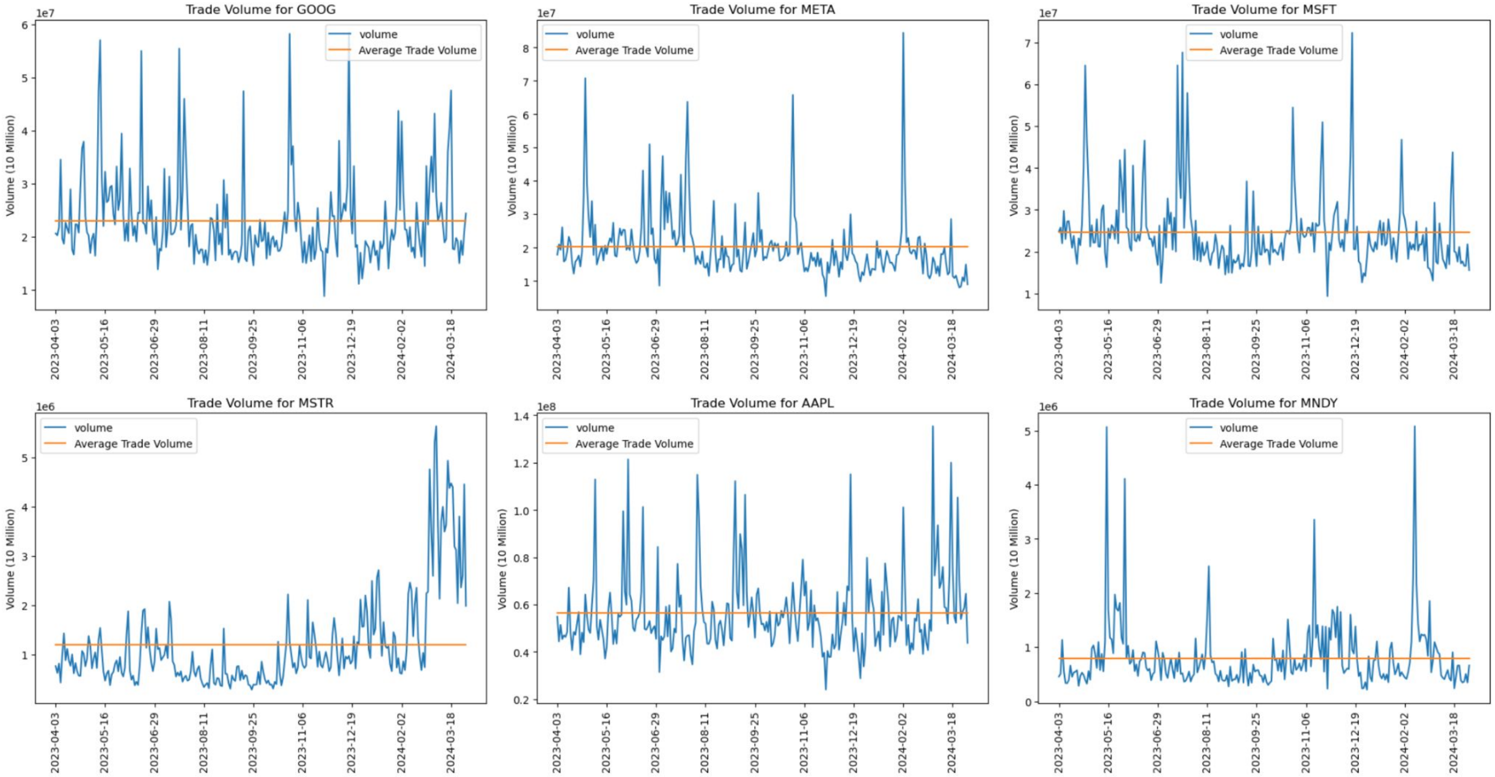
**Note:** For adding datapoint values, you have to define a function first.

# Volume Analysis

Trading volume measures the number of shares traded during a particular time period.

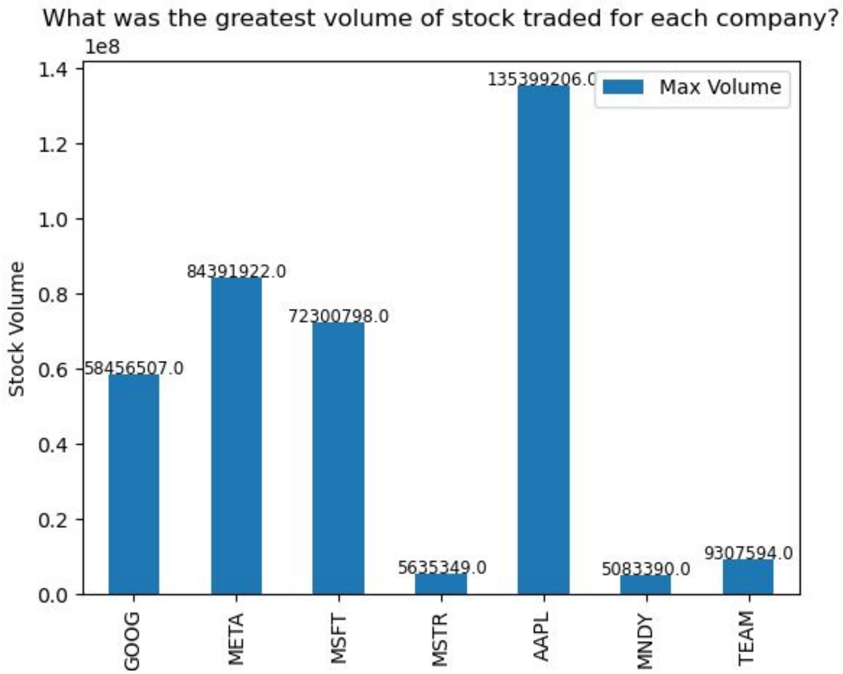
(Source: https://www.schwab.com/learn/story/trading-volume-as-market-indicator)

Plotting the Daily traded Volume for each company response is per the below;




**Observation:** Stocks have a unrelated trading pattern, which indicates market sentiment.

Highest amount of traded volume per company is as below;




**Observation:** While MSTR had the max stock price, the volume of traded stock for MSTR is lowest among our group of companies. On the other hand, Apple has the largest volume of stock traded.

## Further analysis of traded volume: distribution of stock volume by creating volume ranges;

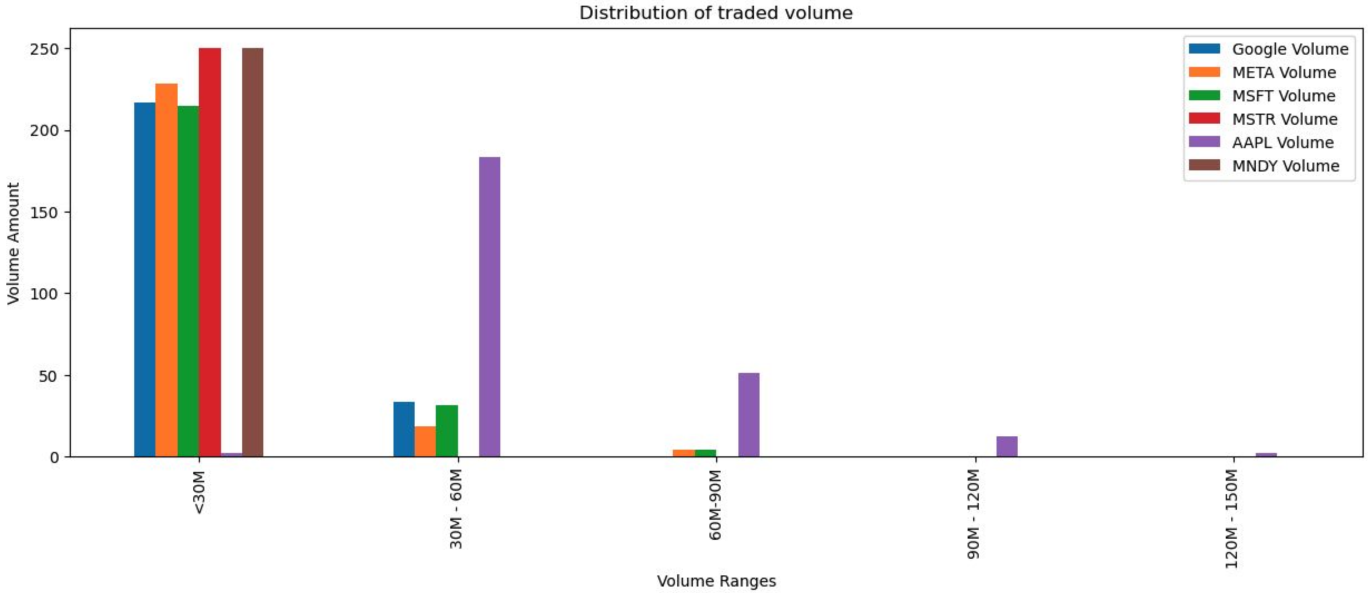
[68]:
```
#Better visualising the stock volume

#Create bins
bins = [0, 30000000, 60000000, 90000000, 120000000, 150000000]
group_names = ["<30M", "30M - 60M", "60M-90M", "90M - 120M", "120M - 150M"]

#create a copy of the datasets
GOOG_data_copy = GOOG_data
META_data_copy = META_data
MSFT_data_copy = MSFT_data
MSTR_data_copy = MSTR_data
AAPL_data_copy = AAPL_data
MNDY_data_copy = MNDY_data

#Append a new column with bins for volume ranges:
GOOG_data_copy["Volume Ranges"] = pd.cut(GOOG_data_copy["volume"], bins, labels = group_names, include_lowest = True)
GOOG_data_copy
```

[68]:

| | Unnamed: 0.1 | index | Unnamed: 0 | None | ticker | date | open | high | low | close | volume | MA for 10 days | MA for 20 days | MA for 50 days | Daily Percent Change | Daily Change | Volume Ranges |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 249 | GOOG | 2023-04-03 | 102.670 | 104.950 | 102.3800 | 104.91 | 20644485.0 | NaN | NaN | NaN | NaN | 2.240 | <30M |
| **1** | 1 | 1 | 1 | 248 | GOOG | 2023-04-04 | 104.840 | 106.100 | 104.6000 | 105.12 | 20299970.0 | NaN | NaN | NaN | 0.002002 | 0.280 | <30M |
| **2** | 2 | 2 | 2 | 247 | GOOG | 2023-04-05 | 106.120 | 106.540 | 104.1021 | 104.95 | 21796705.0 | NaN | NaN | NaN | -0.001617 | -1.170 | <30M |
| **3** | 3 | 3 | 3 | 246 | GOOG | 2023-04-06 | 105.770 | 109.630 | 104.8150 | 108.90 | 34565375.0 | NaN | NaN | NaN | 0.037637 | 3.130 | 30M - 60M |
| **4** | 4 | 4 | 4 | 245 | GOOG | 2023-04-10 | 107.390 | 107.970 | 105.6000 | 106.95 | 19678585.0 | NaN | NaN | NaN | -0.017906 | -0.440 | <30M |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **245** | 245 | 245 | 245 | 4 | GOOG | 2024-03-25 | 150.950 | 151.456 | 148.8000 | 151.15 | 15047965.0 | 146.464 | 141.5520 | 144.4624 | -0.004085 | 0.200 | <30M |
| **246** | 246 | 246 | 246 | 3 | GOOG | 2024-03-26 | 151.240 | 153.200 | 151.0300 | 151.70 | 19275612.0 | 147.672 | 142.1320 | 144.6116 | 0.003639 | 0.460 | <30M |
| **247** | 247 | 247 | 247 | 2 | GOOG | 2024-03-27 | 152.145 | 152.690 | 150.1300 | 151.94 | 16593999.0 | 148.789 | 142.8575 | 144.7688 | 0.001582 | -0.205 | <30M |
| **248** | 248 | 248 | 248 | 1 | GOOG | 2024-03-28 | 152.000 | 152.670 | 151.3300 | 152.26 | 21068018.0 | 149.581 | 143.4815 | 144.9562 | 0.002106 | 0.260 | <30M |
| **249** | 249 | 249 | 249 | 0 | GOOG | 2024-04-01 | 151.830 | 157.000 | 151.6500 | 156.50 | 24416137.0 | 151.014 | 144.4025 | 145.1864 | 0.027847 | 4.670 | <30M |

250 rows × 17 columns

Visualising the distribution of stock volume volume ranges;



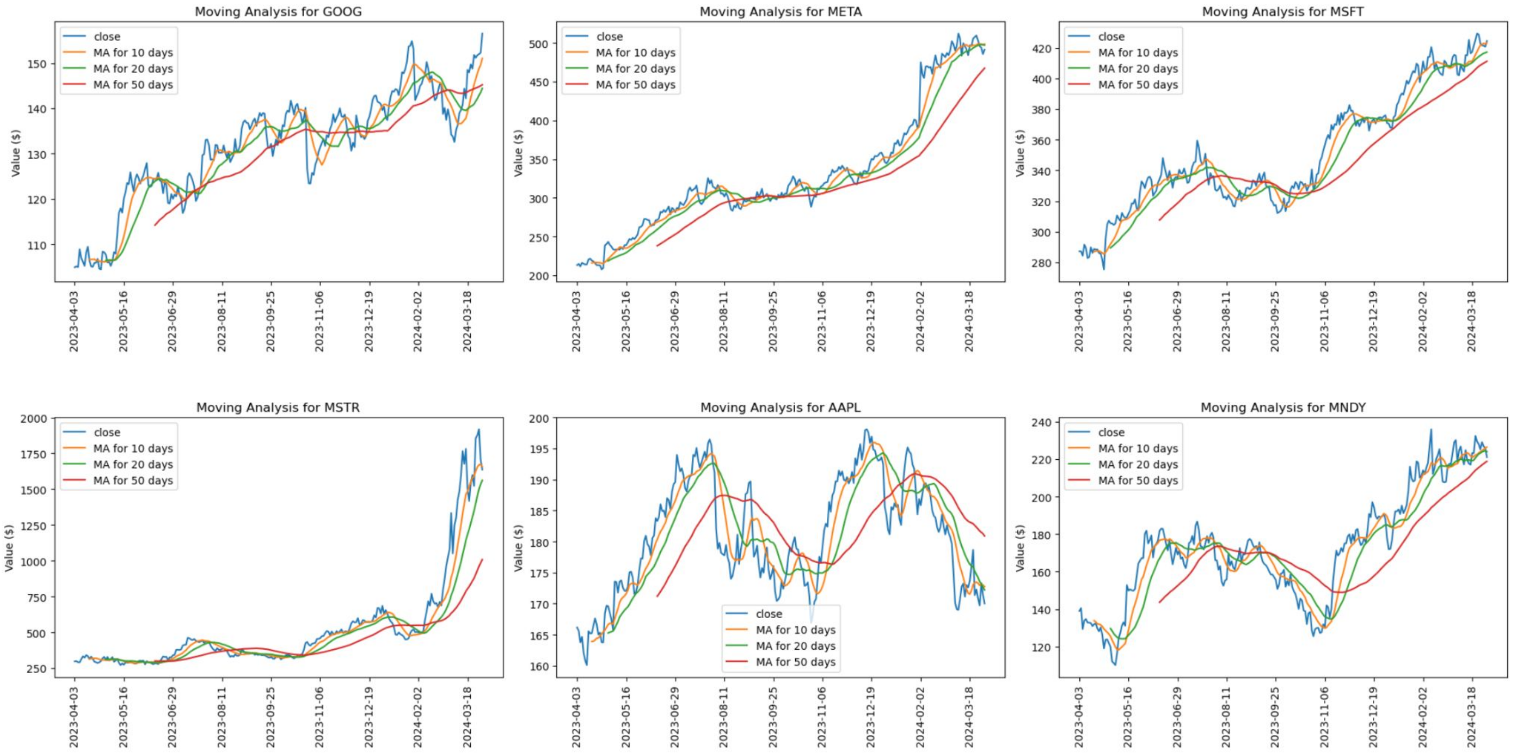

**Observation:** Bulk of the volumes for all companies except Apple reside in the <30M bin, while Apple stock is traded in much greater quantity and lies in 30-60M bin and beyond.

# Moving Average Analysis

The moving average (MA) is a simple technical analysis tool that smooths out price data by creating a constantly updated average price. The average is taken over a specific period of time, like 10 days, 20 minutes, 30 weeks, or any time period the trader chooses

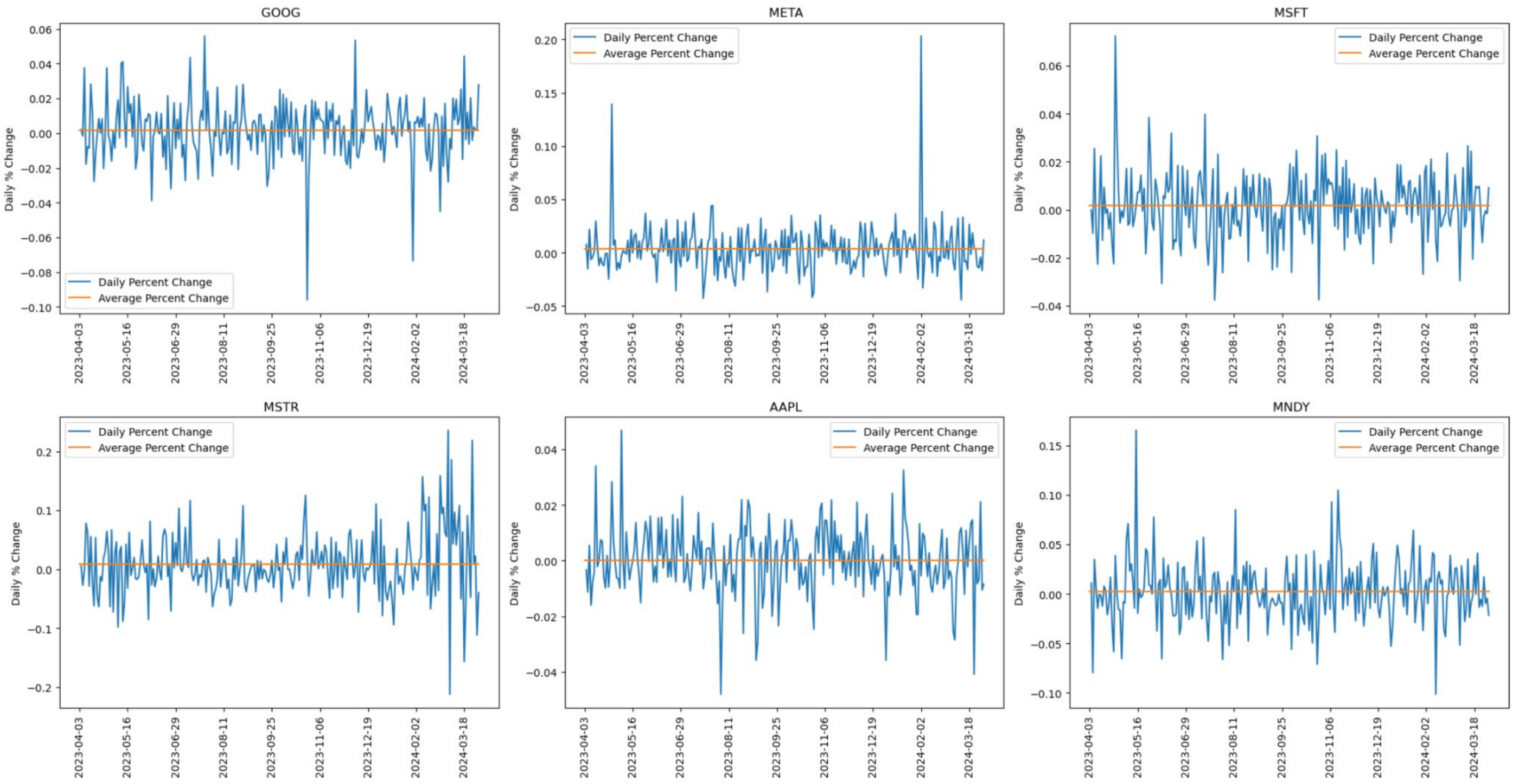
Source: https://www.investopedia.com/articles/active-trading/052014/how-use-moving-average-buy-stocks.asp#:~

Plotting the Moving Averages (10, 20 & 50 Day) for each company response is per the below;




**Observation:** The Moving Averages show a delayed trend line, which in general is softer as the number of days increase from 10 through 50.

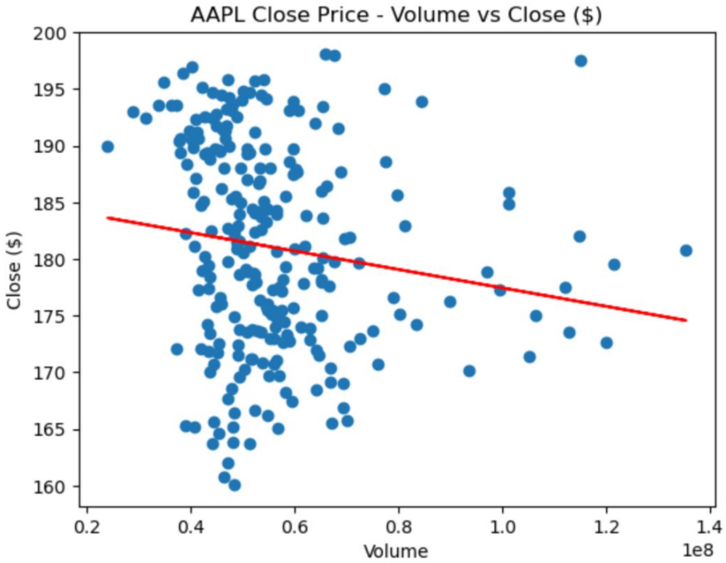
Plotting the Daily Percentage Change for each company response is per the below;




**Observation:** Stocks have a unrelated Daily Change pattern, which indicates market sentiment.

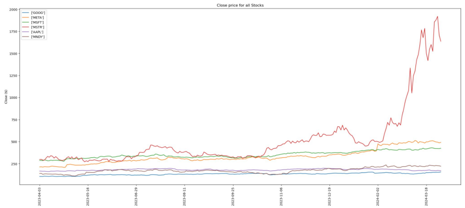
Plotting the recommended stock scatter plot and linear regression line response is per the below;

```
The r-squared is: 0.022414679875326696
The correlation between both factors is -0.15
[<matplotlib.lines.Line2D at 0x2883f29b0>]
```

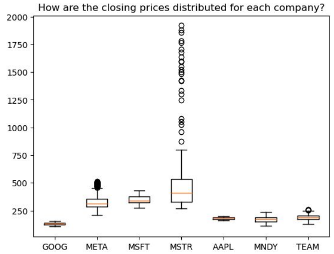



**Observation:** With higher stock traded on a day the stock price tends to lower.

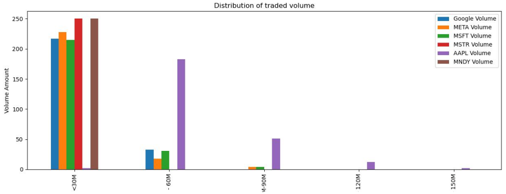
# Conclusion



Time series data showed consistency in Apple’s close price over time




Boxplot further reinforced the tight distribution in Apple’s close price




High volume of Apple stock traded

# Conclusion

**Best Performer**

MicroStrategy's exceptional gains, distinct from the general market trends of six other stocks.

**Recommendation Based on Comparative and Individual Stock Performance:**

Central to our findings is the identification of one particular stock that stands out as a viable investment option. This stock has not exhibited signs of overvaluation. By comparing the mean performance of the entire basket of stocks against this individual stock's performance, we present a compelling case. The specific percentages, indicating a conservative growth rate in contrast to the industry's high valuation trends, reinforce our recommendation.

**Conclusion:**

In conclusion, we advocate for investing in **APPLE (AAPL)**, believing it to represent the best balance of value and growth potential for our client.

# Takeaways / Learnings

1. Using GitHub and branch for code revision management is of benefit, but has its own learning curve.
2. Acquire data from API's can vary in methods from REST to SDK's.
3. NASDAQ has its owns Python SDK nasdatadatalink.
4. The Tech sector in general has performed well over the last 12 months.
5. Adding dates and making it visible to plot x-axis can be challenging.
6. Merging all datasets may have been beneficial in retrospect
7. Writing functions for some steps could have saved us time
8. Domain knowledge (or lack thereof) can play a major role in data analysis

# Thank You !

Team 3