# Stock Market Analysis

- Stock analysis involves comparing a company's current financial statement to its financial statements in previous years to give an investor a sense of whether the company is growing, stable, or deteriorating.

# Overview

- Stock market performance analysis can serve as a basis for investment decisions and help investors make informed decisions about buying or selling stocks. Let us say you work as a data science expert in a company that provides services based on investment decisions. As a data science expert, you can help your company by analyzing the historical performance of various companies, identifying potential opportunities and risks in the stock market, and adjusting your clients' investment strategies accordingly.
- As a data science expert, you can go through a structured process of analyzing stock market performance, which includes collecting historical stock price data of various companies from trusted sources such as Yahoo Finance, visualizing the data using various charts, calculating movements, averages, and volatility for each company, and performing correlation analysis to analyze the relationships between different stock prices.

# Yahoo! Finance API

- Download market data from Yahoo! Finance's API (https://pypi.org/project/yfinance/)
- yfinance offers a threaded and Pythonic way to download market data from Yahoo!® finance. (https://finance.yahoo.com/)
- Check out this Blog post (https://aroussi.com/#post/python-yahoo-finance) for a detailed tutorial with code examples.

In [1]:
```
!pip install yfinance
```

```
Collecting yfinance
  Downloading yfinance-0.2.40-py2.py3-none-any.whl (73 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.5/73.5 kB 2.5 MB/s eta 0:
00:00
Requirement already satisfied: pandas>=1.3.0 in /opt/conda/lib/python3.10/
site-packages (from yfinance) (1.5.3)
Requirement already satisfied: numpy>=1.16.5 in /opt/conda/lib/python3.10/
site-packages (from yfinance) (1.23.5)
Collecting requests>=2.31 (from yfinance)
  Downloading requests-2.32.2-py3-none-any.whl (63 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.9/63.9 kB 3.5 MB/s eta 0:
00:00
Collecting multitasking>=0.0.7 (from yfinance)
  Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Requirement already satisfied: lxml>=4.9.1 in /opt/conda/lib/python3.10/si
te-packages (from yfinance) (4.9.2)
Requirement already satisfied: platformdirs>=2.0.0 in /opt/conda/lib/pytho
n3.10/site-packages (from yfinance) (3.5.0)
Requirement already satisfied: pytz>=2022.5 in /opt/conda/lib/python3.10/s
ite-packages (from yfinance) (2023.3)
Requirement already satisfied: frozendict>=2.3.4 in /opt/conda/lib/python
3.10/site-packages (from yfinance) (2.3.8)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.17.5.tar.gz (3.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.0/3.0 MB 40.8 MB/s eta 0:0
0:00
  Installing build dependencies ... -□ □\□ □|□ □/□ □done
  Getting requirements to build wheel ... -□ □done
  Preparing metadata (pyproject.toml) ... -□ □done
Requirement already satisfied: beautifulsoup4>=4.11.1 in /opt/conda/lib/py
thon3.10/site-packages (from yfinance) (4.12.2)
Requirement already satisfied: html5lib>=1.1 in /opt/conda/lib/python3.10/
site-packages (from yfinance) (1.1)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.10/
site-packages (from beautifulsoup4>=4.11.1->yfinance) (2.3.2.post1)
Requirement already satisfied: six>=1.9 in /opt/conda/lib/python3.10/site-
packages (from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in /opt/conda/lib/python3.10/s
ite-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /opt/conda/lib/py
thon3.10/site-packages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/
python3.10/site-packages (from requests>=2.31->yfinance) (2.1.1)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/s
ite-packages (from requests>=2.31->yfinance) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python
3.10/site-packages (from requests>=2.31->yfinance) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python
3.10/site-packages (from requests>=2.31->yfinance) (2023.5.7)
Building wheels for collected packages: peewee
  Building wheel for peewee (pyproject.toml) ... -□ □\□ □|□ □/□ □done
  Created wheel for peewee: filename=peewee-3.17.5-cp310-cp310-linux_x86_6
4.whl size=293373 sha256=24e0796f023ec4f5c1971102788ee8ccc6ed45576379f37f2
8cfe15650e7370f
  Stored in directory: /root/.cache/pip/wheels/06/80/9b/98db0d58349a2f5c09
f8406789ade4270762f97b7d26f2fa22
Successfully built peewee
Installing collected packages: peewee, multitasking, requests, yfinance
  Attempting uninstall: requests
    Found existing installation: requests 2.28.2
    Uninstalling requests-2.28.2:
```

```
      Successfully uninstalled requests-2.28.2
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the follo
wing dependency conflicts.
apache-beam 2.46.0 requires dill<0.3.2,>=0.3.1.1, but you have dill 0.3.6
which is incompatible.
beatrix-jupyterlab 2023.58.190319 requires jupyter-server~=1.16, but you h
ave jupyter-server 2.5.0 which is incompatible.
google-cloud-artifact-registry 1.8.1 requires google-api-core[grpc]!=2.0.
*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.
*,!=2.9.*,<3.0.0dev,>=1.34.0, but you have google-api-core 1.33.2 which is
incompatible.
google-cloud-dlp 3.12.1 requires google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.
10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.
0dev,>=1.34.0, but you have google-api-core 1.33.2 which is incompatible.
google-cloud-pubsub 2.16.1 requires google-api-core[grpc]!=2.0.*,!=2.1.*,!
=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<
3.0.0dev,>=1.34.0, but you have google-api-core 1.33.2 which is incompatib
le.
google-cloud-resource-manager 1.10.0 requires google-api-core[grpc]!=2.0.
*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.
*,!=2.9.*,<3.0.0dev,>=1.34.0, but you have google-api-core 1.33.2 which is
incompatible.
google-cloud-spanner 3.33.0 requires google-api-core[grpc]!=2.0.*,!=2.1.
*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.
*,<3.0.0dev,>=1.34.0, but you have google-api-core 1.33.2 which is incompa
tible.
kfp 1.8.21 requires google-api-python-client<2,>=1.7.8, but you have googl
e-api-python-client 2.86.0 which is incompatible.
momepy 0.6.0 requires shapely>=2, but you have shapely 1.8.5.post1 which i
s incompatible.
ydata-profiling 4.1.2 requires requests<2.29,>=2.24.0, but you have reques
ts 2.32.2 which is incompatible.
ydata-profiling 4.1.2 requires scipy<1.10,>=1.4.1, but you have scipy 1.1
0.1 which is incompatible.
Successfully installed multitasking-0.0.11 peewee-3.17.5 requests-2.29.0 y
finance-0.2.40
WARNING: Running pip as the 'root' user can result in broken permissions a
nd conflicting behaviour with the system package manager. It is recommende
d to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

# GitHub

- [Yahoo! Finance's API GitHub (https://github.com/ranaroussi/yfinance)](https://github.com/ranaroussi/yfinance)
- Since December 2022 Yahoo has been encrypting the web data that yfinance scrapes for non-price data. Price data still works. Fortunately the decryption keys are available, although Yahoo moved/changed them several times hence yfinance breaking several times. yfinance is now better prepared for any future changes by Yahoo.
- Why is Yahoo doing this? We don't know. Is it to stop scrapers? Maybe, so we've implemented changes to reduce load on Yahoo. In December we rolled out version 0.2 with optimised scraping. Then in 0.2.6 introduced Ticker.fast_info, providing much faster access to some Ticker.info elements wherever possible e.g. price stats and forcing users to switch (sorry but we think necessary).

# real-time stock market data

In [2]:
```
import pandas as pd
import yfinance as yf
from datetime import datetime

start_date = datetime.now() - pd.DateOffset(months=3)
end_date = datetime.now()

tickers = ['AAPL', 'MSFT', 'NFLX', 'GOOG']
# Apple, Microsoft, Netflix, and Google

df_list = []

for ticker in tickers:
    data = yf.download(ticker, start=start_date, end=end_date)
    df_list.append(data)

df = pd.concat(df_list, keys=tickers, names=['Ticker', 'Date'])
print(df.head())
```

```
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed

                           Open        High         Low       Close   Adj Cl
ose  \
Ticker Date
AAPL   2024-02-28  182.509995  183.119995  180.130005  181.419998  181.174
255
       2024-02-29  181.270004  182.570007  179.529999  180.750000  180.505
173
       2024-03-01  179.550003  180.529999  177.380005  179.660004  179.416
656
       2024-03-04  176.149994  176.899994  173.789993  175.100006  174.862
823
       2024-03-05  170.759995  172.039993  169.619995  170.119995  169.889
572

                        Volume
Ticker Date
AAPL   2024-02-28   48953900
       2024-02-29  136682600
       2024-03-01   73488000
       2024-03-04   81510100
       2024-03-05   95132400
```

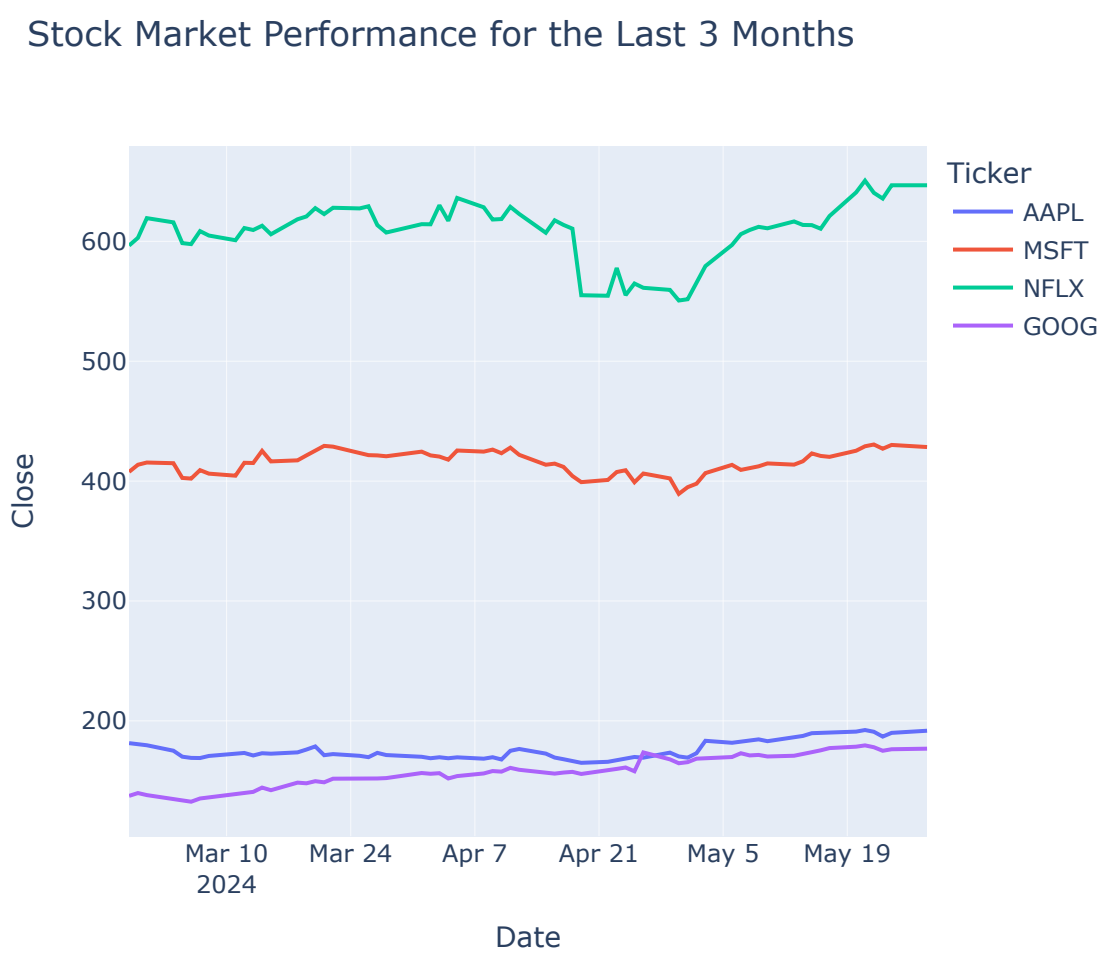
# Index column in the DataFrame

In [3]:
```
df = df.reset_index()
print(df.head())
```

```
  Ticker       Date        Open        High         Low       Close  \
0   AAPL 2024-02-28  182.509995  183.119995  180.130005  181.419998
1   AAPL 2024-02-29  181.270004  182.570007  179.529999  180.750000
2   AAPL 2024-03-01  179.550003  180.529999  177.380005  179.660004
3   AAPL 2024-03-04  176.149994  176.899994  173.789993  175.100006
4   AAPL 2024-03-05  170.759995  172.039993  169.619995  170.119995

    Adj Close     Volume
0  181.174255   48953900
1  180.505173  136682600
2  179.416656   73488000
3  174.862823   81510100
4  169.889572   95132400
```

# Stock Market

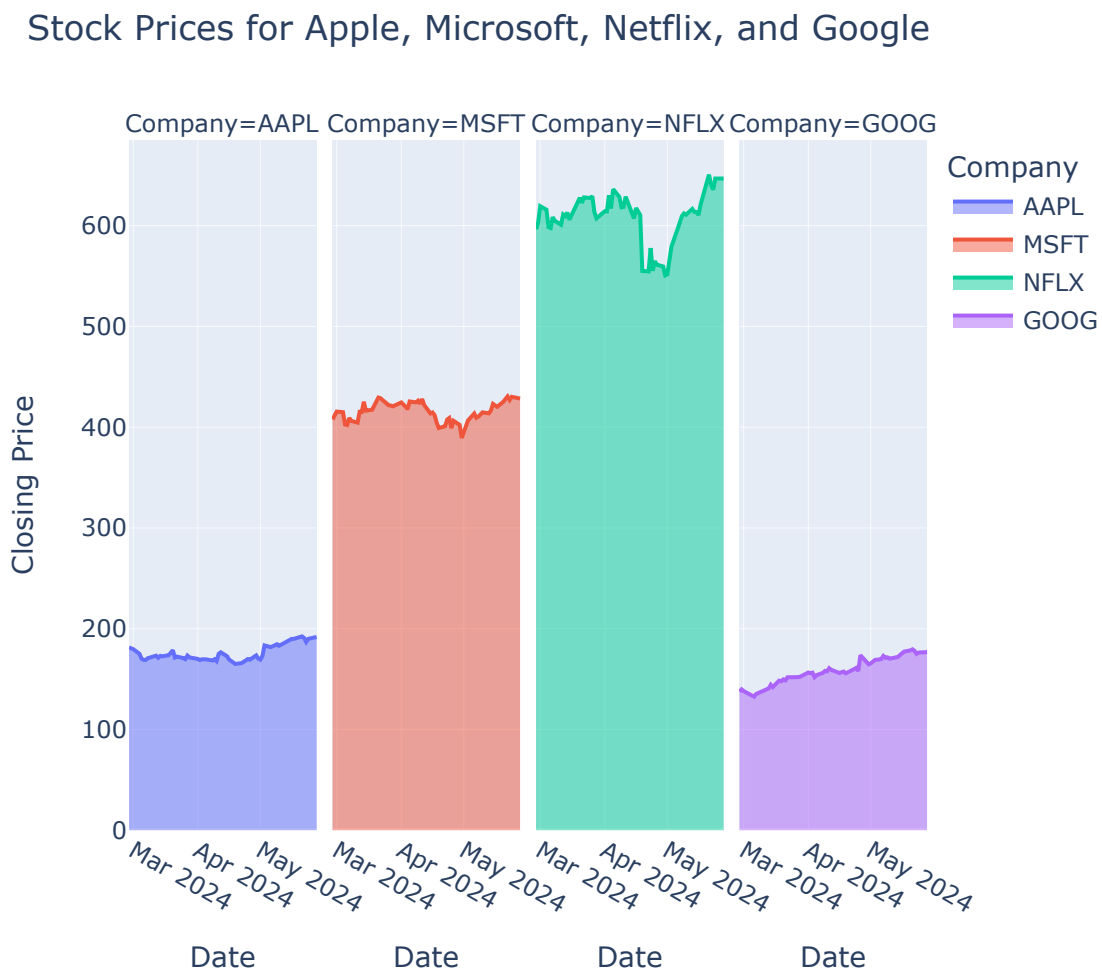
In [4]:
```
import plotly.express as px
fig = px.line(df, x='Date',
              y='Close',
              color='Ticker',
              title="Stock Market Performance for the Last 3 Months")
fig.show()
```




# Compare the Performance of Companies

- look at the faceted area chart, which makes it easy to compare the performance of different companies and identify similarities or differences in their stock price movements:

In [5]:
```
fig = px.area(df, x='Date', y='Close', color='Ticker',
              facet_col='Ticker',
              labels={'Date':'Date', 'Close':'Closing Price', 'Ticker':'Com
pany'},
              title='Stock Prices for Apple, Microsoft, Netflix, and Googl
e')
fig.show()
```




# Analyze Moving Averages

- analyze moving averages, which provide a useful way to identify trends and patterns in each company's stock price movements over a period of time:

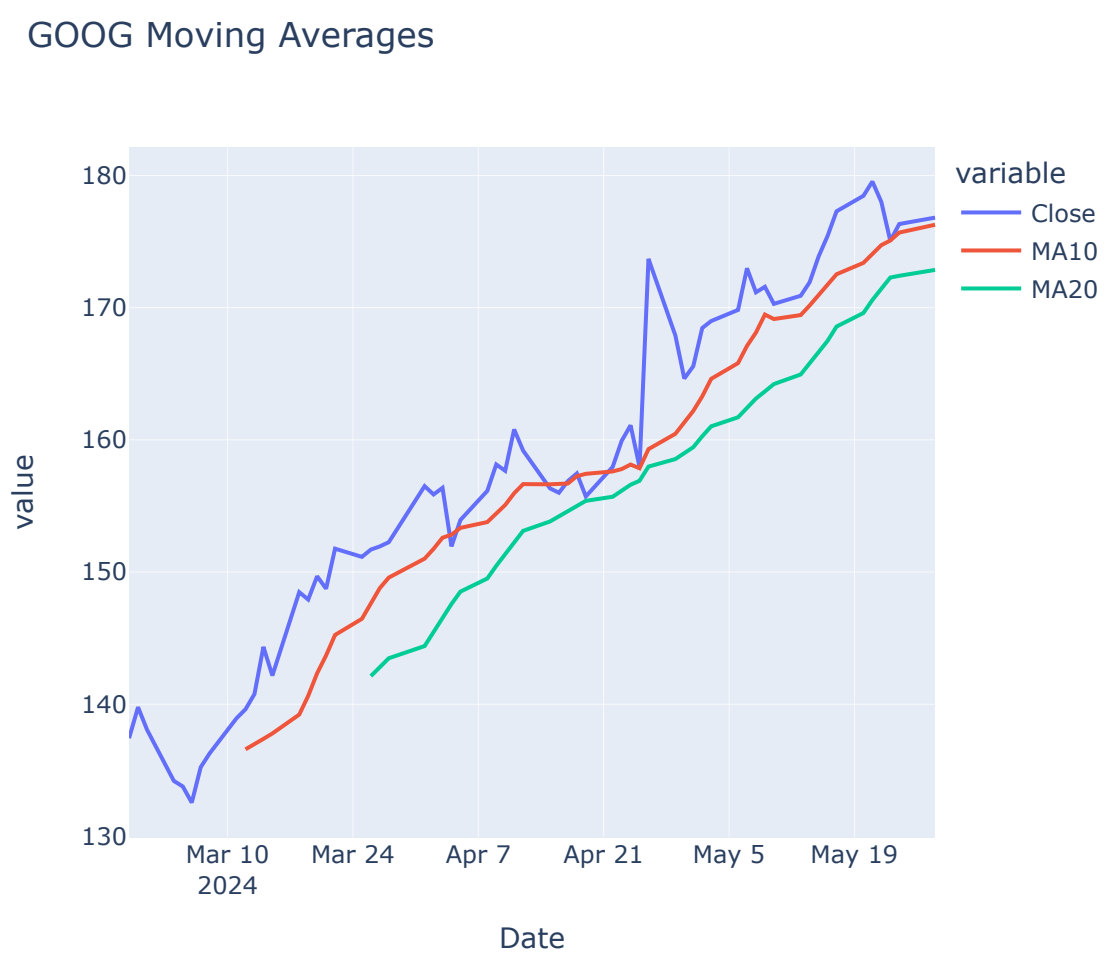
In [6]:
```
df['MA10'] = df.groupby('Ticker')['Close'].rolling(window=10).mean().reset_
index(0, drop=True)
df['MA20'] = df.groupby('Ticker')['Close'].rolling(window=20).mean().reset_
index(0, drop=True)

for ticker, group in df.groupby('Ticker'):
    print(f'Moving Averages for {ticker}')
    print(group[['MA10', 'MA20']])
```

```
Moving Averages for AAPL
          MA10        MA20
0          NaN         NaN
1          NaN         NaN
2          NaN         NaN
3          NaN         NaN
4          NaN         NaN
..         ...         ...
58  187.689000  180.937501
59  188.504999  182.031500
60  188.735999  182.881001
61  189.428998  183.915000
62  189.978998  184.829000

[63 rows x 2 columns]
Moving Averages for GOOG
           MA10        MA20
189         NaN         NaN
190         NaN         NaN
191         NaN         NaN
192         NaN         NaN
193         NaN         NaN
..          ...         ...
247  174.045998  170.578500
248  174.729997  171.423499
249  175.077997  172.278999
250  175.681998  172.410999
251  176.272998  172.856499

[63 rows x 2 columns]
Moving Averages for MSFT
           MA10        MA20
63          NaN         NaN
64          NaN         NaN
65          NaN         NaN
66          NaN         NaN
67          NaN         NaN
..          ...         ...
121  418.653998  410.742999
122  420.651996  411.815999
123  422.119995  413.213998
124  423.661996  414.405998
125  425.129996  415.713498

[63 rows x 2 columns]
Moving Averages for NFLX
           MA10        MA20
126         NaN         NaN
127         NaN         NaN
128         NaN         NaN
129         NaN         NaN
130         NaN         NaN
..          ...         ...
184  619.925000  594.485001
185  623.025000  598.752499
186  625.382996  602.295999
187  628.970996  606.572000
188  631.986993  610.935001

[63 rows x 2 columns]
```

# Visualize the Moving Averages

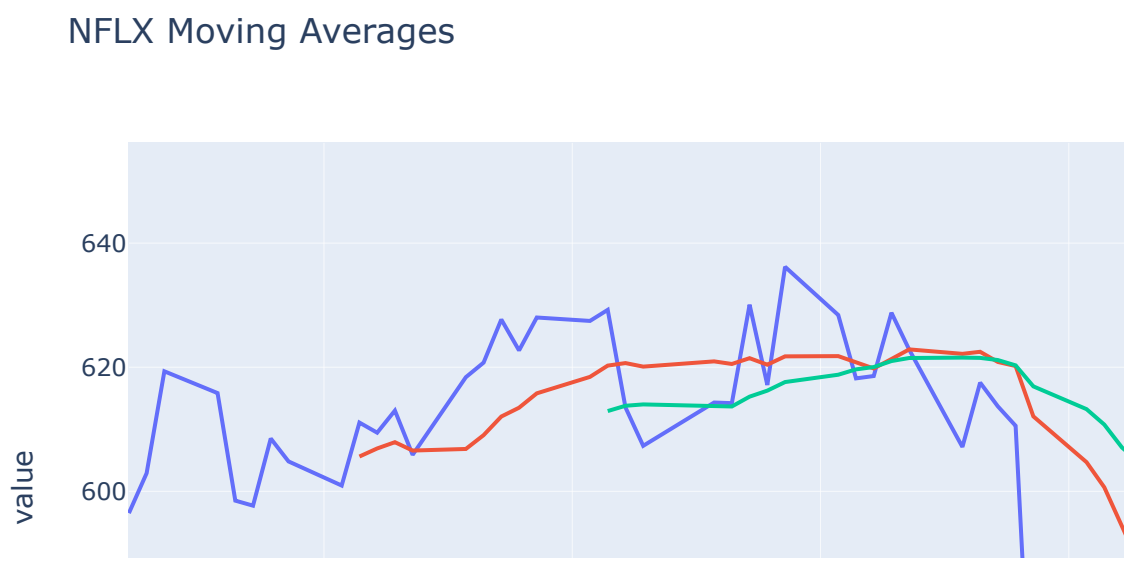
- how to visualize the moving averages of all companies:

In [7]:
```
for ticker, group in df.groupby('Ticker'):
    fig = px.line(group, x='Date', y=['Close', 'MA10', 'MA20'],
                  title=f"{ticker} Moving Averages")
    fig.show()
```




File failed to load: /extensions/MathMenu.js

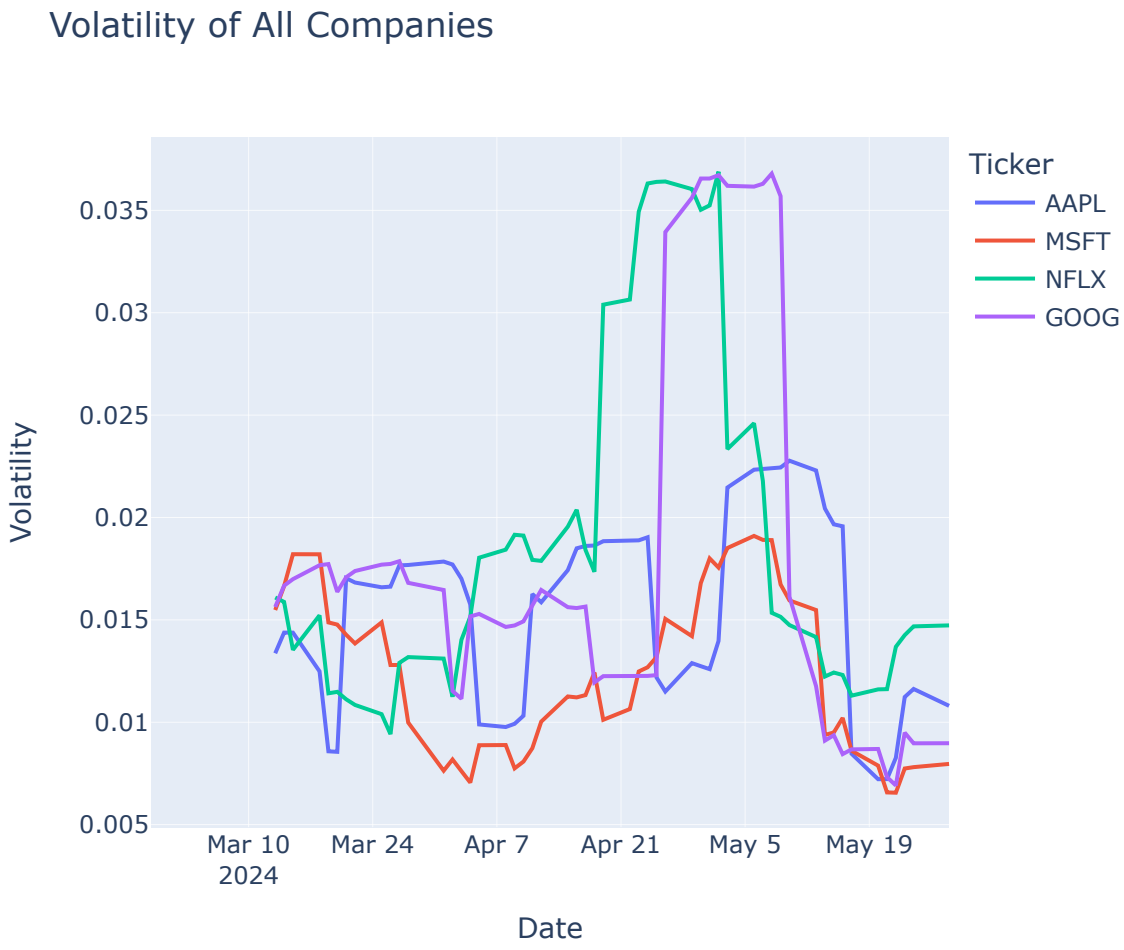

File failed to load: /extensions/MathMenu.js




**The output shows four separate graphs for each company. When the MA10 crosses above the MA20, it is considered a bullish signal indicating that the stock price will continue to rise. Conversely, when the MA10 crosses below the MA20, it is a bearish signal that the stock price will continue falling.**

- Analyze the volatility of all companies. Volatility is a measure of how much and how often the stock price or market fluctuates over a given period of time. Here's how to visualize the volatility of all companies:

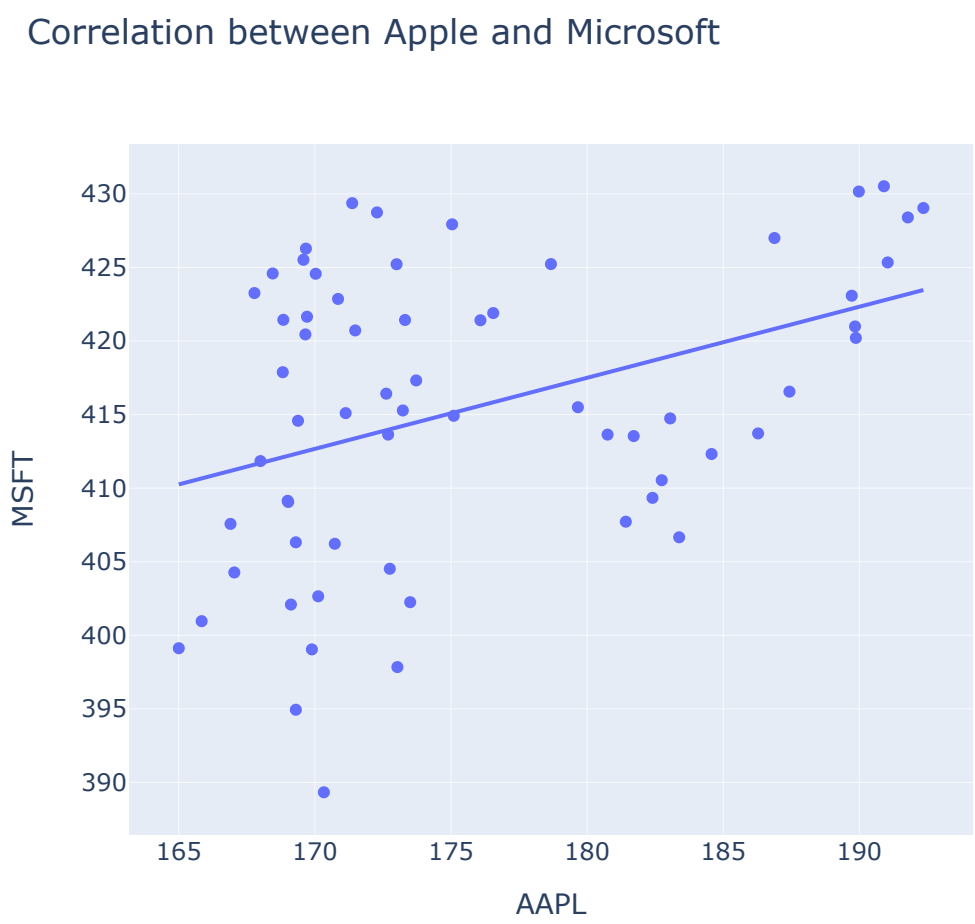
In [8]:
```
df['Volatility'] = df.groupby('Ticker')['Close'].pct_change().rolling(windo
w=10).std().reset_index(0, drop=True)
fig = px.line(df, x='Date', y='Volatility',
              color='Ticker',
              title='Volatility of All Companies')
fig.show()
```




**High volatility indicates that the stock or market experiences large and frequent price movements, while low volatility indicates that the market experiences smaller or less frequent price movements.**

- Analyze the correlation between the stock prices of Apple and Microsoft:

In [9]:
```
# create a DataFrame with the stock prices of Apple and Microsoft
apple = df.loc[df['Ticker'] == 'AAPL', ['Date', 'Close']].rename(columns=
{'Close': 'AAPL'})
microsoft = df.loc[df['Ticker'] == 'MSFT', ['Date', 'Close']].rename(column
s={'Close': 'MSFT'})
df_corr = pd.merge(apple, microsoft, on='Date')

# create a scatter plot to visualize the correlation
fig = px.scatter(df_corr, x='AAPL', y='MSFT',
                 trendline='ols',
                 title='Correlation between Apple and Microsoft')
fig.show()
```



**There is a strong linear relationship between the stock prices of Apple and Microsoft, which means that when the stock price of Apple increases, the stock price of Microsoft also tends to increase. It is a sign of a strong correlation or similarity between the two companies, which can be due to factors such as industry trends, market conditions, or common business partners or customers. For investors, this positive correlation may indicate an opportunity to diversify their portfolio by investing in both companies, as both stocks may offer similar potential returns and risks.**

# Conclusion

- Stock Market Performance Analysis involves calculating moving averages, measuring volatility, conducting correlation analysis, and analyzing various aspects of the stock market to gain a deeper understanding of the factors that affect stock prices and the relationships between the stock prices of different companies.