Stock Market Performance Analysis using Python

Let’s start the task of Stock Market Performance Analysis by importing the necessary Python libraries and the dataset. For this task, I will use the Yahoo finance API (yfinance) to collect real-time stock market data for the past ten years.

It’s important to collect real-time data for this task, but still, if you are a complete beginner and want a dataset only to practice the concepts covered in this article, you can download the dataset from here. But it’s recommended to use the yfinance API to collect and work on real-time data. You can install the yfinace API in your Python environment using the pip command mentioned below (run the command below on your command prompt or terminal):

for command prompt or terminal: pip install yfinance for Google Colab or Jupyter notebooks: !pip install yfinance Now below is how we can collect real-time stock market data using the yfinance API:

In [1]:
```python
import pandas as pd
import yfinance as yf
from datetime import datetime

start_date = datetime.now() - pd.DateOffset(months=120)
end_date = datetime.now()

tickers = ['AAPL', 'MSFT', 'NFLX', 'GOOG']

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 Close  \
Ticker Date
AAPL   2014-04-28  20.457144  21.276787  20.448214  21.217501  18.688694
       2014-04-29  21.205000  21.285000  21.053928  21.154642  18.633331
       2014-04-30  21.165714  21.408215  21.064285  21.074642  18.562864
       2014-05-01  21.142857  21.242857  20.941429  21.124287  18.606596
       2014-05-02  21.155001  21.221430  21.061071  21.163570  18.641197

                      Volume
Ticker Date
AAPL   2014-04-28  669485600
       2014-04-29  337377600
       2014-04-30  456640800
       2014-05-01  244048000
       2014-05-02  191514400
```

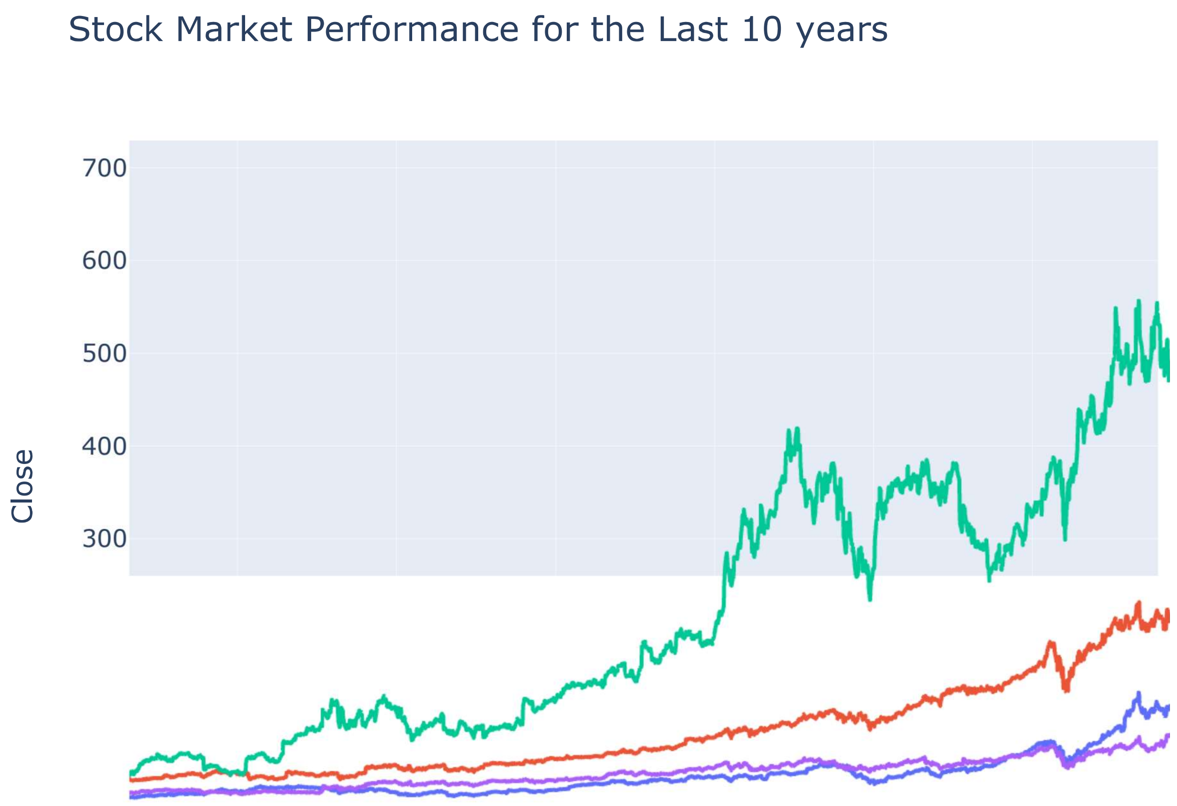
```
In the above code, we first imported the necessary Python libraries and downloaded the historical stock price data for four companies: Apple, Microsoft, Netflix, and Google, for the last three months.

In this dataset, the Date column is the index column in the DataFrame. We need to reset the index before moving forward:
```

In [2]:
```
df = df.reset_index()
print(df.head())
```

```
  Ticker        Date       Open       High        Low      Close  Adj Close  \
0   AAPL 2014-04-28  20.457144  21.276787  20.448214  21.217501  18.688694   
1   AAPL 2014-04-29  21.205000  21.285000  21.053928  21.154642  18.633331   
2   AAPL 2014-04-30  21.165714  21.408215  21.064285  21.074642  18.562864   
3   AAPL 2014-05-01  21.142857  21.242857  20.941429  21.124287  18.606596   
4   AAPL 2014-05-02  21.155001  21.221430  21.061071  21.163570  18.641197   

      Volume  
0  669485600  
1  337377600  
2  456640800  
3  244048000  
4  191514400  
```

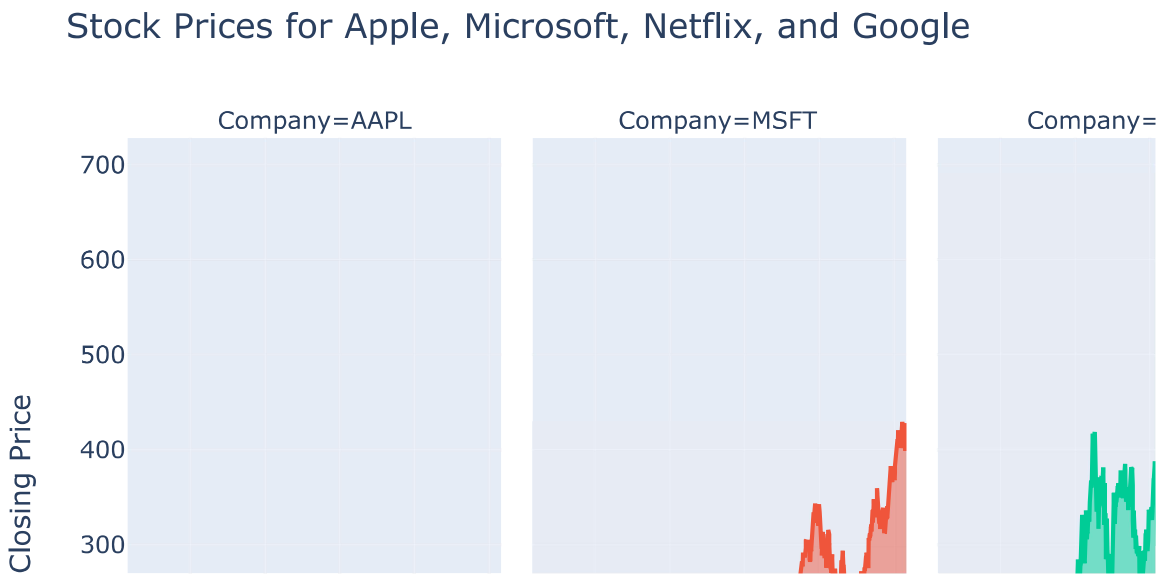
```
Now let’s have a look at the performance in the stock market of all the
companies:
```

In [3]:
```python
import plotly.express as px
fig = px.line(df, x='Date',
              y='Close',
              color='Ticker',
              title="Stock Market Performance for the Last 10 years")
fig.show()
```



```
Now let’s 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 [4]:
```python
fig = px.area(df, x='Date', y='Close', color='Ticker',
              facet_col='Ticker',
              labels={'Date':'Date', 'Close':'Closing Price', 'Ticker':'Co
              title='Stock Prices for Apple, Microsoft, Netflix, and Googl
fig.show()
```



Now let's 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:

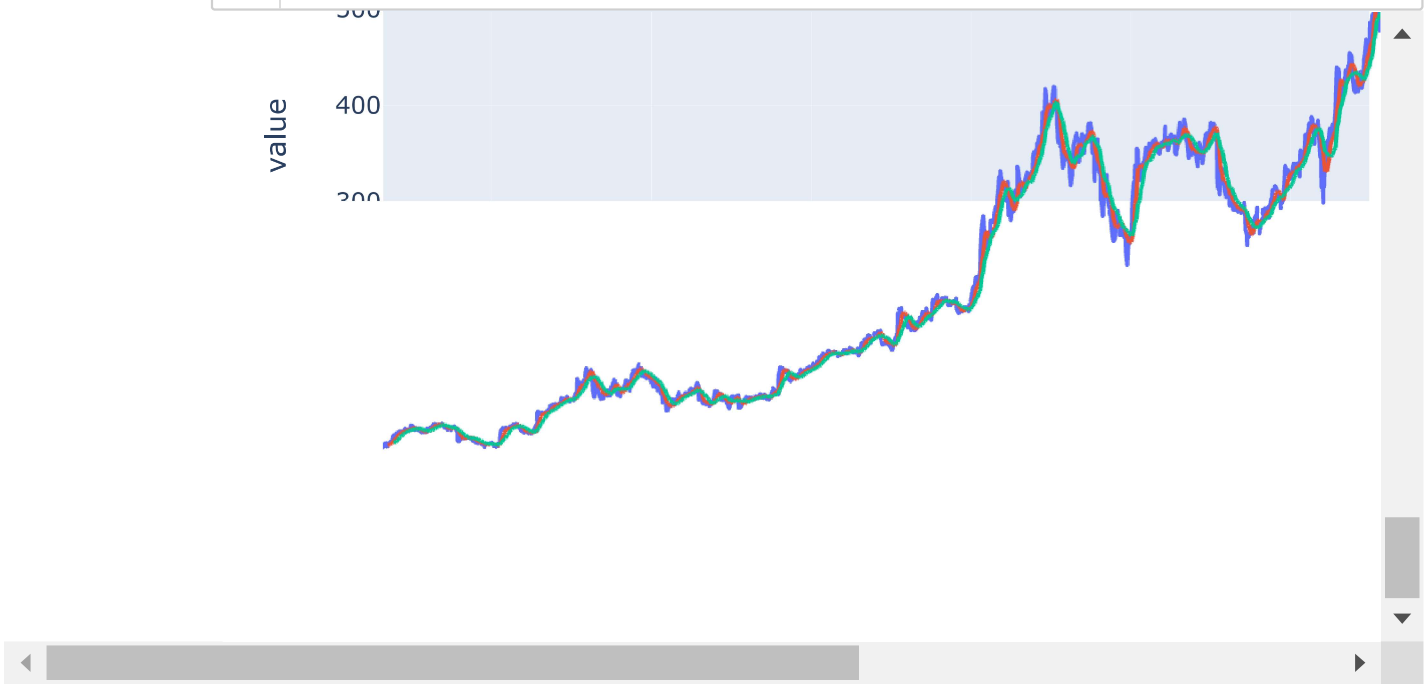
In [5]:
```
df['MA10'] = df.groupby('Ticker')['Close'].rolling(window=10).mean().reset
df['MA20'] = df.groupby('Ticker')['Close'].rolling(window=20).mean().reset

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
...          ...         ...
2513  169.698999  169.885500
2514  169.421999  169.687999
2515  169.545999  169.653499
2516  169.031000  169.482499
2517  168.306000  169.373499

[2518 rows x 2 columns]
Moving Averages for GOOG
             MA10      MA20
7554          NaN       NaN
7555          NaN       NaN
7556          NaN       NaN
7557          NaN       NaN
7558          NaN       NaN
...           ...       ...
10067  157.612001  155.6965
10068  157.790001  156.1350
10069  158.134001  156.6050
10070  157.850002  156.9055
10071  159.300002  157.9770

[2518 rows x 2 columns]
Moving Averages for MSFT
            MA10        MA20
2518         NaN         NaN
2519         NaN         NaN
2520         NaN         NaN
2521         NaN         NaN
2522         NaN         NaN
...          ...         ...
5031  414.377997  418.244498
5032  412.506998  417.479999
5033  411.086996  416.850499
5034  408.197998  415.731000
5035  406.639999  415.011000

[2518 rows x 2 columns]
Moving Averages for NFLX
            MA10        MA20
5036         NaN         NaN
5037         NaN         NaN
5038         NaN         NaN
5039         NaN         NaN
5040         NaN         NaN
...          ...         ...
7549  604.695007  613.242007
7550  600.650006  610.756506
7551  594.304004  607.050507
7552  587.906000  604.614005
```

```
7553  581.745996  602.309003

[2518 rows x 2 columns]
```

Now here’s how to visualize the moving averages of all companies:

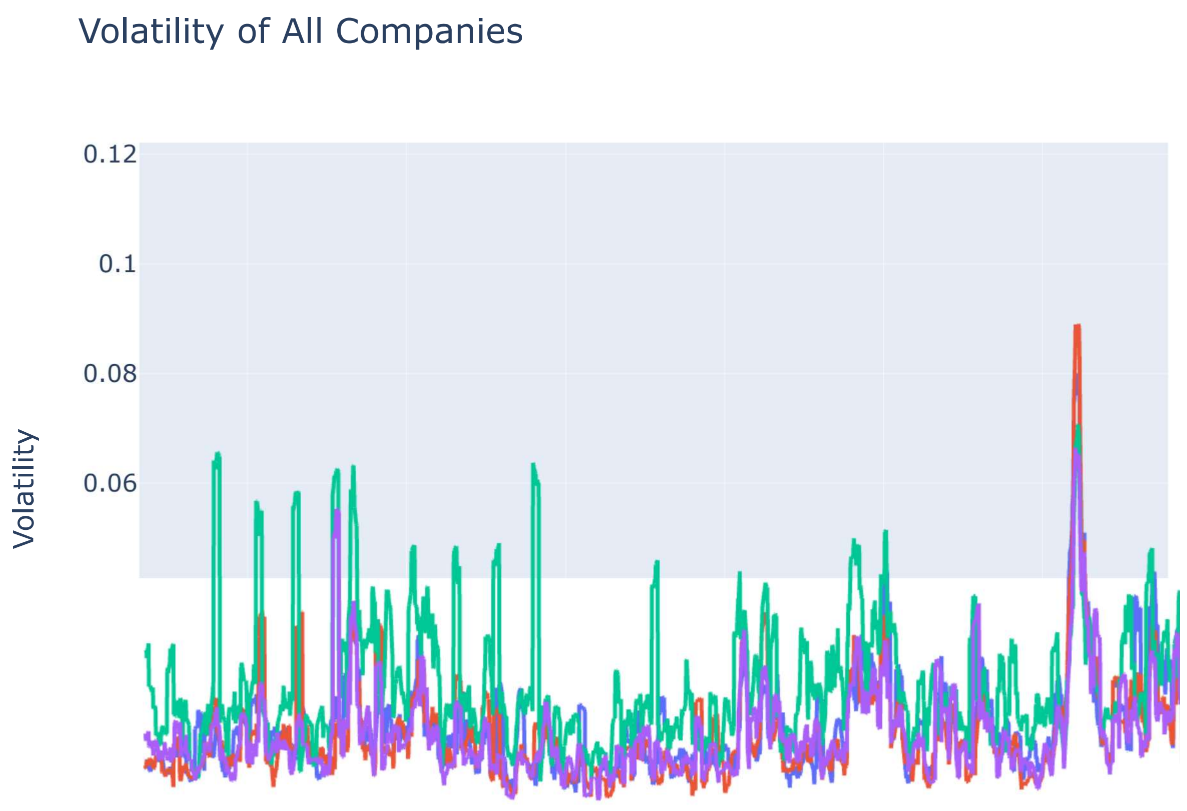
In [6]:
```
for ticker, group in df.groupby('Ticker'):
    fig = px.line(group, x='Date', y=['Close', 'MA10', 'MA20'],
                  title=f"{ticker} Moving Averages")
    fig.show()
```



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.

Let us now 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:

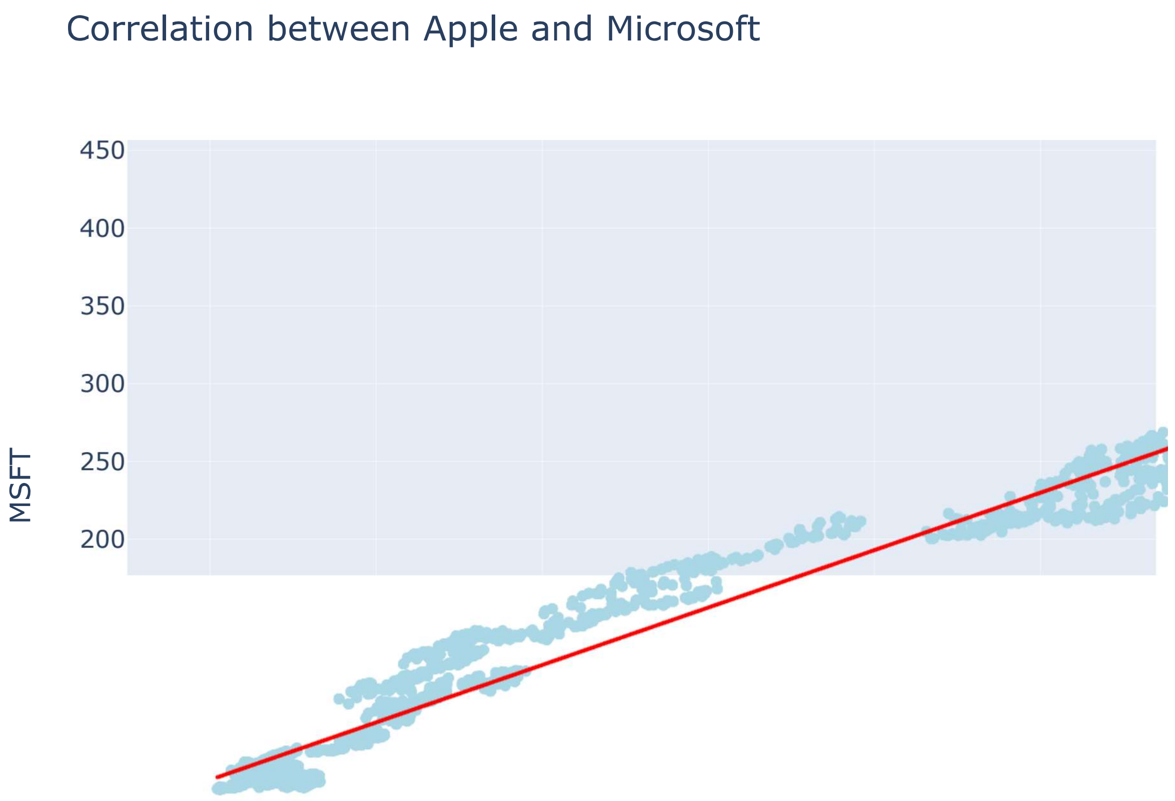
In [7]:
```
df['Volatility'] = df.groupby('Ticker')['Close'].pct_change().rolling(wind
fig = px.line(df, x='Date', y='Volatility',
              color='Ticker',
              title='Volatility of All Companies')
fig.show()
```



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.

Now let’s analyze the correlation between the stock prices of Apple and Microsoft:

In [8]:
```
# create a DataFrame with the stock prices of Apple and Microsoft
apple = df.loc[df['Ticker'] == 'AAPL', ['Date', 'Close']].rename(columns=
microsoft = df.loc[df['Ticker'] == 'MSFT', ['Date', 'Close']].rename(colur
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')

# Changing the color of data points to blue
fig.update_traces(marker=dict(color='lightblue'))

# Changing the color of the trend line to red
fig.update_traces(line=dict(color='red'))

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.

```

Summary

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. I hope you liked this article on Stock Market Performance Analysis using Python. Feel free to ask valuable questions in the comments section below.

In [ ]:

```

```

In [ ]:

```

```