Stock Market Performance Analysis

Name: Raj C. Mhatre

Email id :rmhatre404@gmail.com (mailto:rmhatre404@gmail.com) , contact No.: +91-9768877272

Linkedin ID: https://www.linkedin.com/in/raj-c-mhatre-571b67219/

```
#Importing all the required libraries:

import pandas as pd
import yfinance as yf
from datetime import datetime
```

1] Now below is how we can collect real-time stock market data using the yfinance API:

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

```
start_date
```

```
Timestamp('2023-02-27 12:03:34.084809')
```

```
end_date
```

```
datetime.datetime(2023, 5, 27, 12, 3, 34, 85185)
```

```
tickers = ['AAPL', 'MSFT', 'NFLX', 'GOOG', 'AMZN', 'KO', 'SBUX']
```

```
df_list = []

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

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

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

```
df
```

		Open	High	Low	Close	Adj Close	Volume

```
print(df.head())
```

```
                           Open        High         Low       Close   Adj Close  \
Ticker Date
AAPL   2023-02-27  147.710007  149.169998  147.449997  147.919998  147.715683
       2023-02-28  147.050003  149.080002  146.830002  147.410004  147.206390
       2023-03-01  146.830002  147.229996  145.009995  145.309998  145.109283
       2023-03-02  144.380005  146.710007  143.899994  145.910004  145.708466
       2023-03-03  148.039993  151.110001  147.330002  151.029999  150.821381

                     Volume
Ticker Date
AAPL   2023-02-27  44998500
       2023-02-28  50547000
       2023-03-01  55479000
       2023-03-02  52238100
       2023-03-03  70732300
```

```
print(df.tail())
```

```
                           Open        High         Low       Close   Adj Close  \
Ticker Date
SBUX   2023-05-22  105.519997  105.699997  102.769997  102.900002  102.900002
       2023-05-23  102.120003  102.139999  100.080002  100.339996  100.339996
       2023-05-24  100.800003  100.809998   99.110001   99.610001   99.610001
       2023-05-25   98.699997   98.900002   97.730003   98.440002   98.440002
       2023-05-26   98.629997   99.610001   98.330002   98.529999   98.529999

                     Volume
Ticker Date
SBUX   2023-05-22  7504300
       2023-05-23  6766300
       2023-05-24  6134800
       2023-05-25  8177000
       2023-05-26  7273600
```

2] We need to reset the index before moving forward:

```
df = df.reset_index()
```

```
df
```

	Ticker	Date	Open	High	Low	Close	Adj Close	Volume
0	AAPL	2023-02-27	147.710007	149.169998	147.449997	147.919998	147.715683	44998500
1	AAPL	2023-02-28	147.050003	149.080002	146.830002	147.410004	147.206390	50547000
2	AAPL	2023-03-01	146.830002	147.229996	145.009995	145.309998	145.109283	55479000
3	AAPL	2023-03-02	144.380005	146.710007	143.899994	145.910004	145.708466	52238100
4	AAPL	2023-03-03	148.039993	151.110001	147.330002	151.029999	150.821381	70732300
...	...	...	...	...	...	...	...	...
443	SBUX	2023-05-22	105.519997	105.699997	102.769997	102.900002	102.900002	7504300
444	SBUX	2023-05-23	102.120003	102.139999	100.080002	100.339996	100.339996	6766300
445	SBUX	2023-05-24	100.800003	100.809998	99.110001	99.610001	99.610001	6134800
446	SBUX	2023-05-25	98.699997	98.900002	97.730003	98.440002	98.440002	8177000
447	SBUX	2023-05-26	98.629997	99.610001	98.330002	98.529999	98.529999	7273600

448 rows × 8 columns

```
print(df.head())
```

```
  Ticker       Date        Open        High         Low       Close  \
0   AAPL 2023-02-27  147.710007  149.169998  147.449997  147.919998
1   AAPL 2023-02-28  147.050003  149.080002  146.830002  147.410004
2   AAPL 2023-03-01  146.830002  147.229996  145.009995  145.309998
3   AAPL 2023-03-02  144.380005  146.710007  143.899994  145.910004
4   AAPL 2023-03-03  148.039993  151.110001  147.330002  151.029999
```

```
    Adj Close    Volume
0  147.715683  44998500
1  147.206390  50547000
2  145.109283  55479000
3  145.708466  52238100
4  150.821381  70732300
```

```
print(df.tail())
```

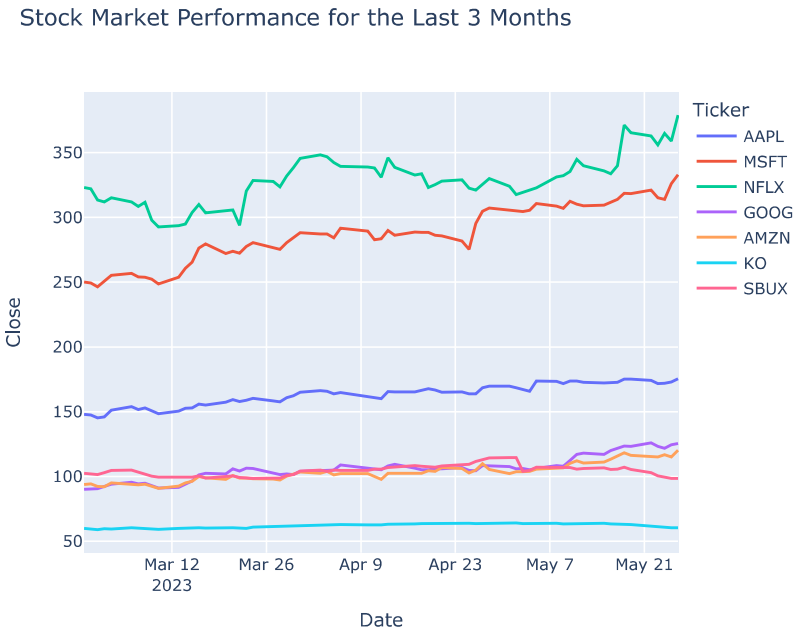
```
     Ticker       Date        Open        High         Low       Close  \
443   SBUX 2023-05-22  105.519997  105.699997  102.769997  102.900002
444   SBUX 2023-05-23  102.120003  102.139999  100.080002  100.339996
445   SBUX 2023-05-24  100.800003  100.809998   99.110001   99.610001
446   SBUX 2023-05-25   98.699997   98.900002   97.730003   98.440002
447   SBUX 2023-05-26   98.629997   99.610001   98.330002   98.529999

     Adj Close   Volume
443  102.900002  7504300
444  100.339996  6766300
445   99.610001  6134800
446   98.440002  8177000
447   98.529999  7273600
```

3] Now let's have a look at the performance in the stock market of all the companies:

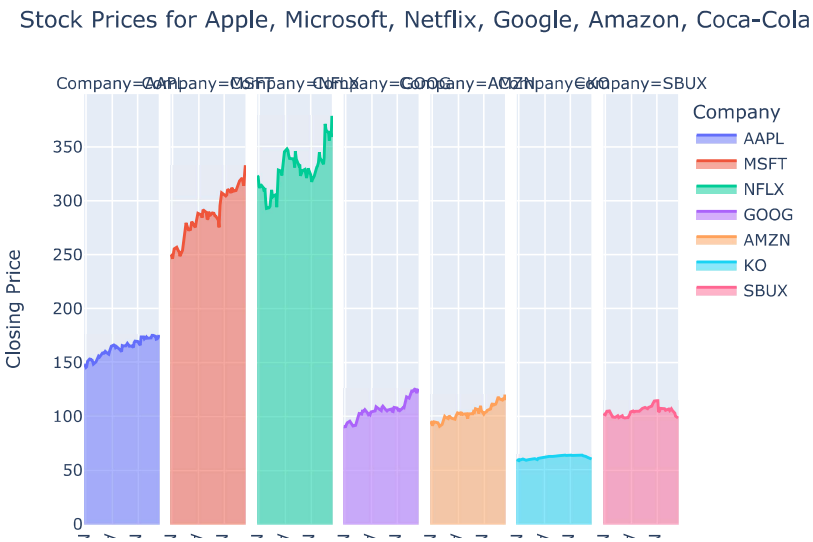
```
import plotly.express as px

fig = px.line(df, x = 'Date',
              y = 'Close',
              color = 'Ticker',
              title = "Stock Market Performance for the Last 3 Months")
fig.show()
```



4] Compare the performance of different companies and identify similarities or differences in their stock price movements:

```
fig = px.area(df, x='Date', y='Close', color='Ticker',
              facet_col='Ticker',
              labels={'Date':'Date', 'Close':'Closing Price', 'Ticker':'Company'},
              title='Stock Prices for Apple, Microsoft, Netflix, Google, Amazon, Coca-Cola and Starbucks')
fig.show()
```



5] 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:

Date Date Date Date Date Date Date

```
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']])
```

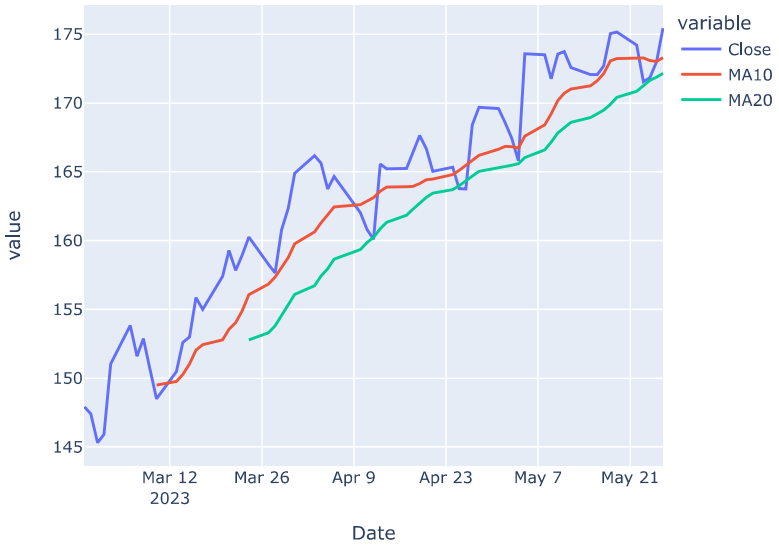
```
385       NaN       NaN
386       NaN       NaN
387       NaN       NaN
388       NaN       NaN
..        ...       ...
443   105.854  107.9235
444   105.217  107.4735
445   104.481  106.8805
446   103.762  106.1650
447   103.003  105.3770

[64 rows x 2 columns]
```

6] Here's how to visualize the moving averages of all companies:

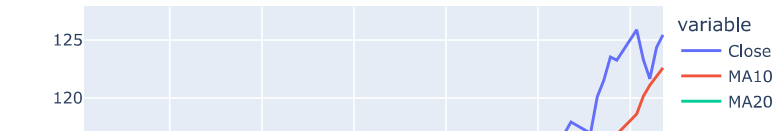
```
for ticker, group in df.groupby('Ticker'):
  fig = px.line(group, x='Date', y=['Close', 'MA10', 'MA20'],
                title=f"{ticker} Moving Averages")
  fig.show()
```

AAPL Moving Averages



AMZN Moving Averages

GOOG Moving Averages



7] Here's how to visualize the volatility of all companies:

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