[1]:

```
!pip install yfinance
```

```
Requirement already satisfied: yfinance in c:\users\dhanusha\anaconda3\lib\site-packages (0.2.36)
Requirement already satisfied: lxml>=4.9.1 in c:\users\dhanusha\anaconda3\lib\site-packages (from yfinance) (4.9.1)
Requirement already satisfied: peewee>=3.16.2 in c:\users\dhanusha\anaconda3\lib\site-packages (from yfinance) (3.17.0)
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\dhanusha\anaconda3\lib\site-packages (from yfinance) (4.11.1)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\dhanusha\anaconda3\lib\site-packages (from yfinance) (2.4.0)
Requirement already satisfied: pandas>=1.3.0 in c:\users\dhanusha\anaconda3\lib\site-packages (from yfinance) (1.5.3)
Requirement already satisfied: appdirs>=1.4.4 in c:\users\dhanusha\anaconda3\lib\site-packages (from yfinance) (1.4.4)
Requirement already satisfied: requests>=2.31 in c:\users\dhanusha\anaconda3\lib\site-packages (from yfinance) (2.31.0)
Requirement already satisfied: pytz>=2022.5 in c:\users\dhanusha\anaconda3\lib\site-packages (from yfinance) (2022.7)
Requirement already satisfied: html5lib>=1.1 in c:\users\dhanusha\anaconda3\lib\site-packages (from yfinance) (1.1)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\dhanusha\anaconda3\lib\site-packages (from yfinance) (0.0.11)
Requirement already satisfied: numpy>=1.16.5 in c:\users\dhanusha\anaconda3\lib\site-packages (from yfinance) (1.23.5)
Requirement already satisfied: soupsieve>1.2 in c:\users\dhanusha\anaconda3\lib\site-packages (from beautifulsoup4>=4.11.1->yfinance) (2.3.2.post1)
Requirement already satisfied: six>=1.9 in c:\users\dhanusha\anaconda3\lib\site-packages (from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in c:\users\dhanusha\anaconda3\lib\site-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\dhanusha\anaconda3\lib\site-packages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\dhanusha\anaconda3\lib\site-packages (from requests>=2.31->yfinance) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\dhanusha\anaconda3\lib\site-packages (from requests>=2.31->yfinance) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\dhanusha\anaconda3\lib\site-packages (from requests>=2.31->yfinance) (1.26.14)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\dhanusha\anaconda3\lib\site-packages (from requests>=2.31->yfinance) (2022.12.7)
```

[2]:

```
import pandas as pd
import yfinance as yf
from datetime import datetime
import plotly.express as px
```

```
C:\Users\DHANUSHA\anaconda3\lib\site-packages\yfinance\base.py:48: FutureWarning: The default dtype for empty Series will be 'object' instead of 'float64' in a future version. Spec
ify a dtype explicitly to silence this warning.
  _empty_series = pd.Series()
```

[3]:

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

[4]:

```
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)
```

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

[5]:

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

```
                         Open        High         Low       Close   Adj Close  \
Ticker Date
AAPL   2023-11-21  191.410004  191.520004  189.740005  190.639999  190.397049
       2023-11-22  191.490005  192.929993  190.830002  191.309998  191.066193
       2023-11-24  190.869995  190.899994  189.250000  189.970001  189.727905
       2023-11-27  189.919998  190.669998  188.899994  189.789993  189.548126
       2023-11-28  189.779999  191.080002  189.399994  190.399994  190.157349

                     Volume
Ticker Date
AAPL   2023-11-21  38134500
       2023-11-22  39617700
       2023-11-24  24048300
       2023-11-27  40552600
       2023-11-28  38415400
```

[6]:

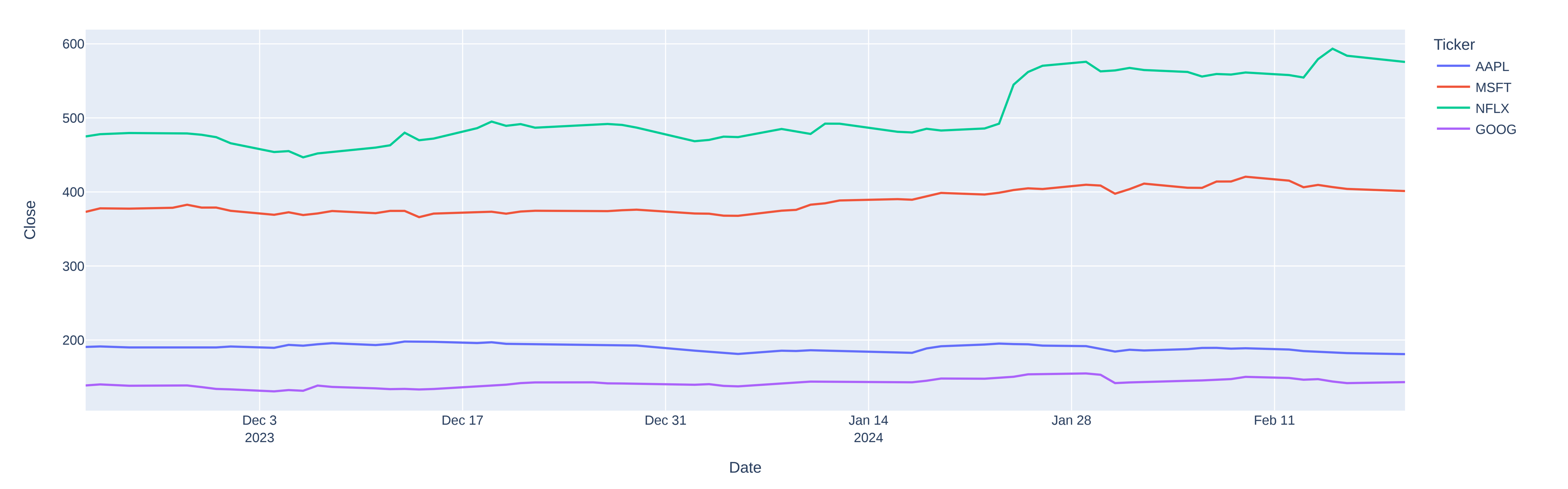
```
df = df.reset_index()
print(df.head())
```

```
  Ticker       Date        Open        High         Low       Close  \
0   AAPL 2023-11-21  191.410004  191.520004  189.740005  190.639999
1   AAPL 2023-11-22  191.490005  192.929993  190.830002  191.309998
2   AAPL 2023-11-24  190.869995  190.899994  189.250000  189.970001
3   AAPL 2023-11-27  189.919998  190.669998  188.899994  189.789993
4   AAPL 2023-11-28  189.779999  191.080002  189.399994  190.399994

    Adj Close    Volume
0  190.397049  38134500
1  191.066193  39617700
2  189.727905  24048300
3  189.548126  40552600
4  190.157349  38415400
```

[7]:

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

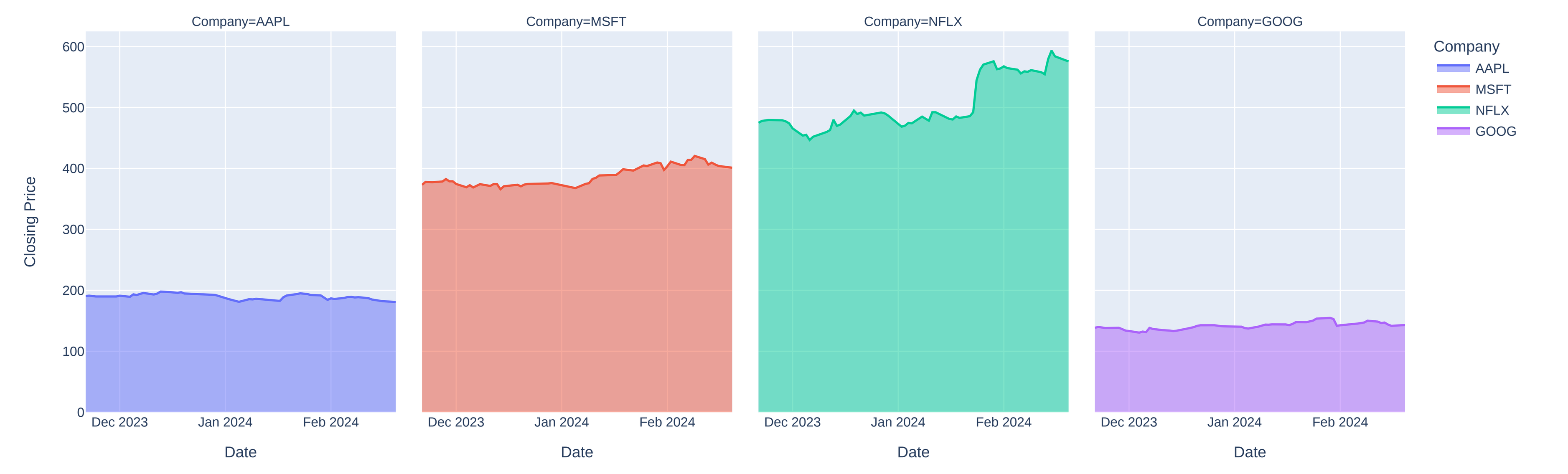
Stocl Market Performance for the last 3 months



[8]:

```
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, and Google')
fig.show()
```

Stock Prices for Apple, Microsoft, Netflix, and Google



[9]:

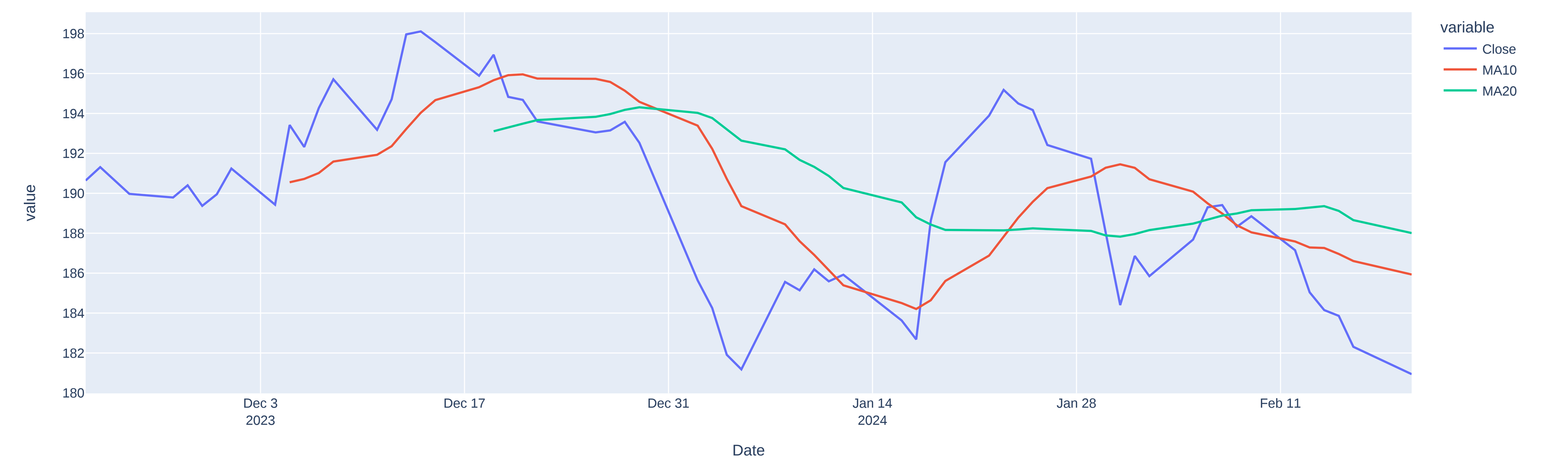
```
df['MA10'] = df.groupby('Ticker')['Close'].rolling(window=10).mean().reset_index(0, drop=True)
df['MA20'] = df.groupby('Ticker')['Close'].rolling(window=20).mean().reset_index(0, drop=True)

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
..         ...         ...
56  187.286000  189.282999
57  187.261000  189.356499
58  186.961000  189.117999
59  186.606999  188.655499
60  185.933000  188.007999

[61 rows x 2 columns]
Moving Averages for GOOG
           MA10        MA20
183         NaN         NaN
184         NaN         NaN
185         NaN         NaN
186         NaN         NaN
187         NaN         NaN
..          ...         ...
239  145.760999  147.775999
240  146.294998  147.988499
241  146.417998  147.935999
242  146.239998  147.625499
243  146.060498  147.396748

[61 rows x 2 columns]
Moving Averages for MSFT
           MA10        MA20
61          NaN         NaN
62          NaN         NaN
63          NaN         NaN
64          NaN         NaN
65          NaN         NaN
..          ...         ...
117  409.400995  405.054997
118  410.591995  406.055997
119  410.869995  406.690497
120  410.153995  406.959996
121  409.709995  407.194995

[61 rows x 2 columns]
Moving Averages for NFLX
           MA10        MA20
122         NaN         NaN
123         NaN         NaN
124         NaN         NaN
125         NaN         NaN
126         NaN         NaN
..          ...         ...
178  560.572003  542.406998
179  562.094006  547.356999
180  564.689008  552.764500
181  566.620007  557.814500
182  567.971008  562.307501

[61 rows x 2 columns]
```

[10]:

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

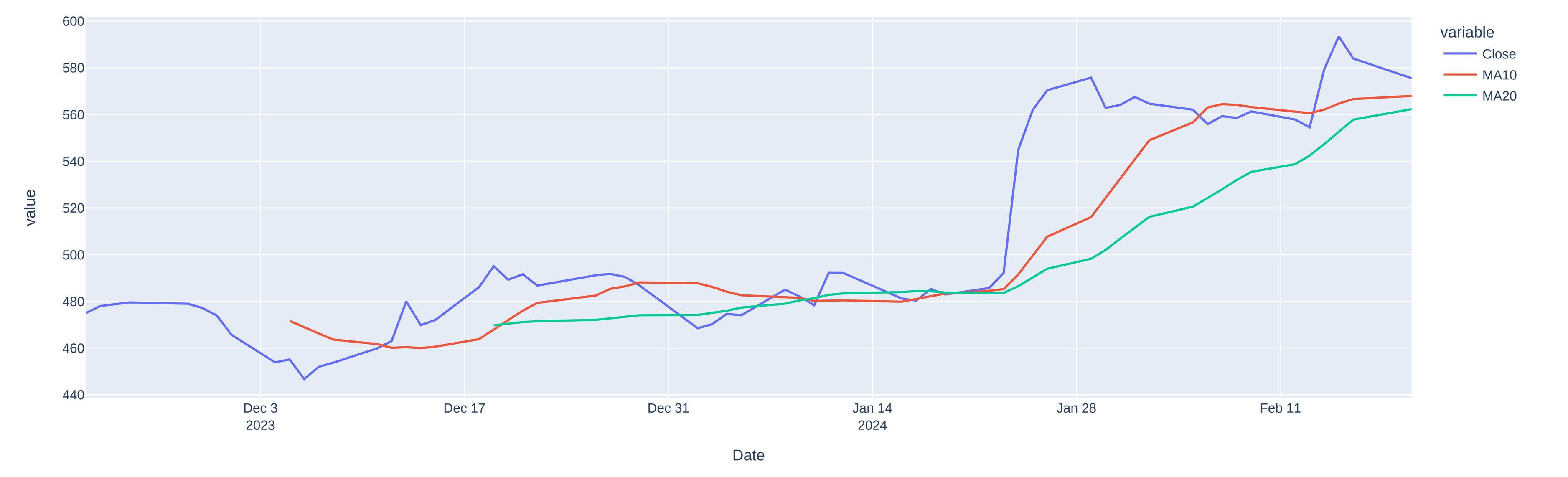
AAPL Moving Averages



GOOG Moving Averages

MSFT Moving Averages

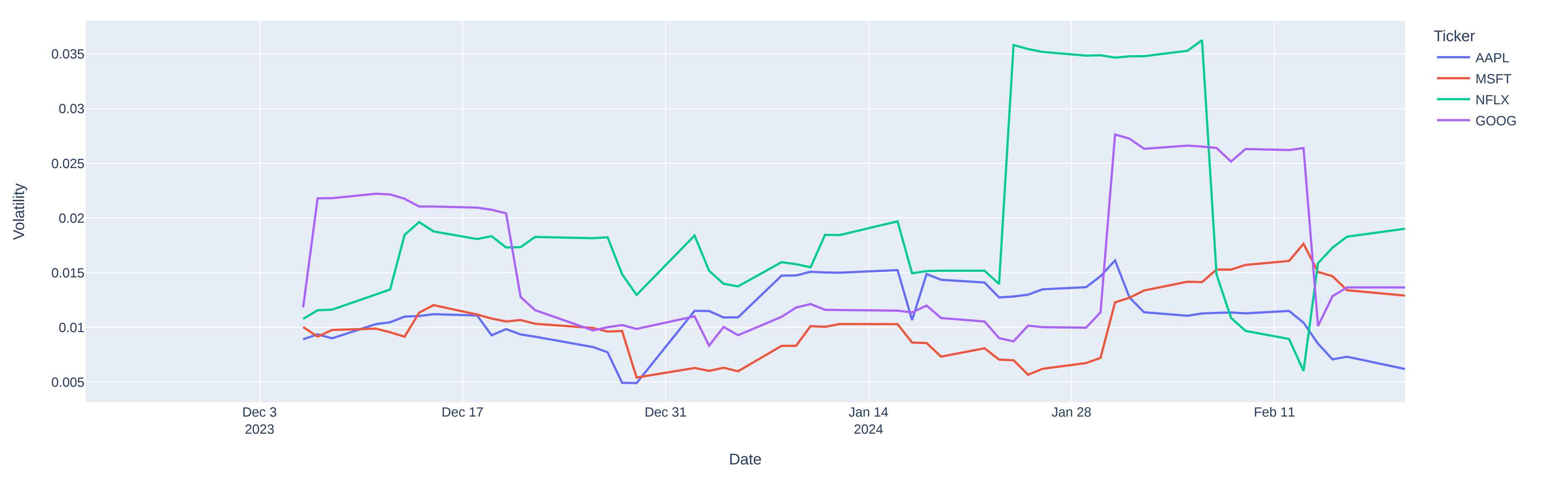
NFLX Moving Averages



[11]:

```
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()
```

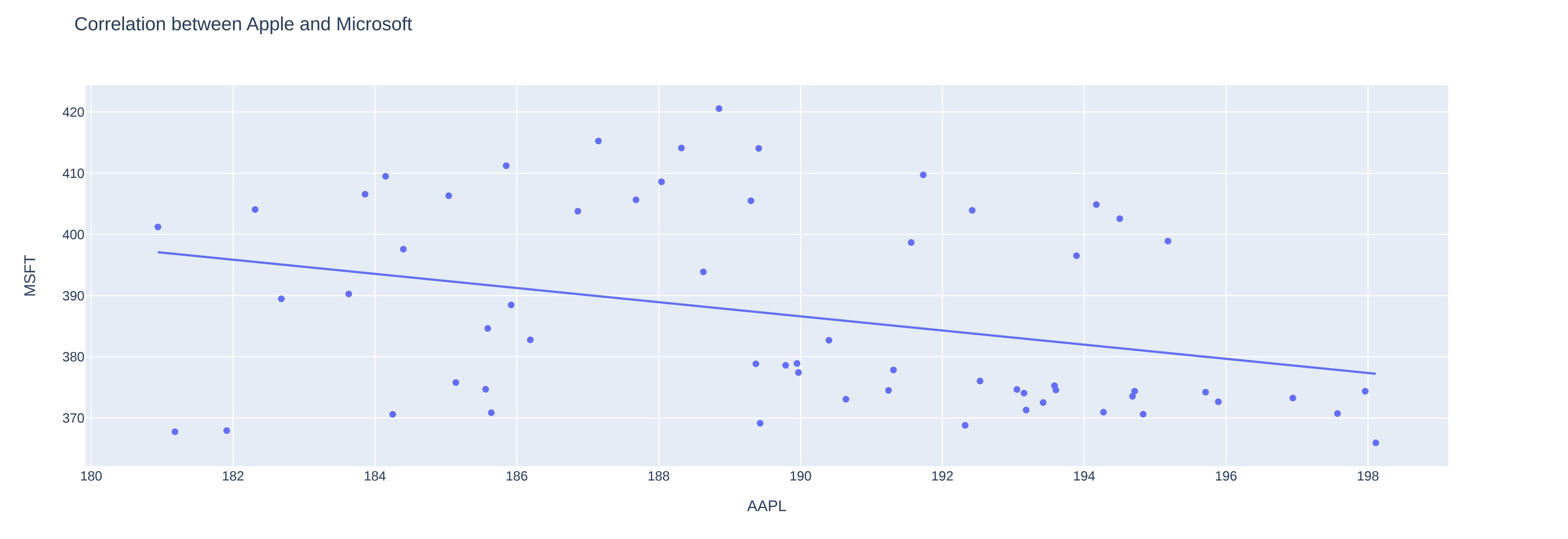
Volatility of All Companies



[12]:

```
# 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(columns={'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()
```

Correlation between Apple and Microsoft



[13]:

```
'''Summary Stock Market Performance Analysis involves calculating moving averages, measuring volatility, conducting correlation analysis and analyzing various aspects of the stock r
```

[13]:

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