Stock-Market-Prediction-and-Analysis (/github/ShubhaTiwarii/Stock-Market-Prediction-and-Analysis/tree/main)
/
Stock Market Analysis.ipynb (/github/ShubhaTiwarii/Stock-Market-Prediction-and-Analysis/tree/main/Stock Market Analysis.ipynb)

In [1]:
```
'''
STOCK MARKET PREDICTION AND ANALYSIS

1. Stocks form Apple, Amazon, Google, and Microsoft are explored (closing prices, daily return, movi
2. Correlation between stocks is observed.
3. Risk of investing in a particular stock is measured.
4. Time Series forecasting is done using ARIMA for Google Stocks.
5. Future stock prices are predicted through Long Short Term Memory (LSTM) method.

'''
```

Out[1]:
```
' \nSTOCK MARKET PREDICTION AND ANALYSIS \n\n1. Stocks form Apple, Amazon, Google, and Microsoft ar
e explored (closing prices, daily return, moving average). \n2. Correlation between stocks is obser
ved. \n3. Risk of investing in a particular stock is measured.  \n4. Time Series forecasting is don
e using ARIMA for Google Stocks.\n5. Future stock prices are predicted through Long Short Term Memo
ry (LSTM) method. \n\n'
```

In [2]:
```
!pip install yfinance pandas_datareader
```

```
Requirement already satisfied: yfinance in c:\users\shubh\anaconda3\lib\site-packages (0.2.46)
Requirement already satisfied: pandas_datareader in c:\users\shubh\anaconda3\lib\site-packages (0.1
0.0)
Requirement already satisfied: pandas>=1.3.0 in c:\users\shubh\anaconda3\lib\site-packages (from yf
inance) (1.5.3)
Requirement already satisfied: numpy>=1.16.5 in c:\users\shubh\anaconda3\lib\site-packages (from yf
inance) (1.24.3)
Requirement already satisfied: requests>=2.31 in c:\users\shubh\anaconda3\lib\site-packages (from y
finance) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\shubh\anaconda3\lib\site-packages (f
rom yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in c:\users\shubh\anaconda3\lib\site-packages (from yfin
ance) (4.9.2)
Requirement already satisfied: platformdirs>=2.0.0 in c:\users\shubh\anaconda3\lib\site-packages (f
rom yfinance) (2.5.2)
Requirement already satisfied: pytz>=2022.5 in c:\users\shubh\anaconda3\lib\site-packages (from yfi
nance) (2022.7)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\shubh\anaconda3\lib\site-packages (fro
m yfinance) (2.4.6)
Requirement already satisfied: peewee>=3.16.2 in c:\users\shubh\anaconda3\lib\site-packages (from y
finance) (3.17.7)
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\shubh\anaconda3\lib\site-packages
(from yfinance) (4.12.2)
Requirement already satisfied: html5lib>=1.1 in c:\users\shubh\anaconda3\lib\site-packages (from yf
inance) (1.1)
Requirement already satisfied: soupsieve>1.2 in c:\users\shubh\anaconda3\lib\site-packages (from be
autifulsoup4>=4.11.1->yfinance) (2.4)
Requirement already satisfied: six>=1.9 in c:\users\shubh\anaconda3\lib\site-packages (from html5li
b>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in c:\users\shubh\anaconda3\lib\site-packages (from htm
l5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\shubh\anaconda3\lib\site-packages
(from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\shubh\anaconda3\lib\site-packag
es (from requests>=2.31->yfinance) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\shubh\anaconda3\lib\site-packages (from req
uests>=2.31->yfinance) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\shubh\anaconda3\lib\site-packages (fr
om requests>=2.31->yfinance) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\shubh\anaconda3\lib\site-packages (fr
om requests>=2.31->yfinance) (2023.5.7)
```

In [3]:
```
!pip install --upgrade yfinance
```

```
Requirement already satisfied: yfinance in c:\users\shubh\anaconda3\lib\site-packages (0.2.46)
Requirement already satisfied: pandas>=1.3.0 in c:\users\shubh\anaconda3\lib\site-packages (from yf
inance) (1.5.3)
Requirement already satisfied: numpy>=1.16.5 in c:\users\shubh\anaconda3\lib\site-packages (from yf
inance) (1.24.3)
Requirement already satisfied: requests>=2.31 in c:\users\shubh\anaconda3\lib\site-packages (from y
finance) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\shubh\anaconda3\lib\site-packages (f
rom yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in c:\users\shubh\anaconda3\lib\site-packages (from yfin
ance) (4.9.2)
Requirement already satisfied: platformdirs>=2.0.0 in c:\users\shubh\anaconda3\lib\site-packages (f
rom yfinance) (2.5.2)
Requirement already satisfied: pytz>=2022.5 in c:\users\shubh\anaconda3\lib\site-packages (from yfi
nance) (2022.7)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\shubh\anaconda3\lib\site-packages (fro
m yfinance) (2.4.6)
Requirement already satisfied: peewee>=3.16.2 in c:\users\shubh\anaconda3\lib\site-packages (from y
finance) (3.17.7)
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\shubh\anaconda3\lib\site-packages
(from yfinance) (4.12.2)
Requirement already satisfied: html5lib>=1.1 in c:\users\shubh\anaconda3\lib\site-packages (from yf
inance) (1.1)
Requirement already satisfied: soupsieve>1.2 in c:\users\shubh\anaconda3\lib\site-packages (from be
autifulsoup4>=4.11.1->yfinance) (2.4)
Requirement already satisfied: six>=1.9 in c:\users\shubh\anaconda3\lib\site-packages (from html5li
b>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in c:\users\shubh\anaconda3\lib\site-packages (from htm
l5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\shubh\anaconda3\lib\site-packages
(from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\shubh\anaconda3\lib\site-packag
es (from requests>=2.31->yfinance) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\shubh\anaconda3\lib\site-packages (from req
uests>=2.31->yfinance) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\shubh\anaconda3\lib\site-packages (fr
om requests>=2.31->yfinance) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\shubh\anaconda3\lib\site-packages (fr
om requests>=2.31->yfinance) (2023.5.7)
```

In [18]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
plt.style.use("fivethirtyeight")
%matplotlib inline

# reading stock data from yahoo finance
from pandas_datareader.data import DataReader
import yfinance as yf
from pandas_datareader import data as pdr


from datetime import datetime

stock_data = {}

# stocks used for this analysis
tech_list = ['AAPL', 'GOOG', 'MSFT', 'AMZN']

# End and Start times for data
tech_list = ['AAPL', 'GOOG', 'MSFT', 'AMZN']

end = datetime.now()
start = datetime(end.year - 1, end.month, end.day)

for stock in tech_list:
    stock_data[stock] = yf.download(stock, start=start, end=end)


company_list = [AAPL, GOOG, MSFT, AMZN]
company_name = ["APPLE", "GOOGLE", "MICROSOFT", "AMAZON"]

for company, com_name in zip(company_list, company_name):
    company["company_name"] = com_name

df = pd.concat(company_list, axis=0)
df.tail(10)
```

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

Out[18]:

Price	Adj Close	Close	High	Low	Open	Volume	company_name	Adj Close	Close			High	Low
Ticker	AAPL	AAPL	AAPL	AAPL	AAPL	AAPL		GOOG	GOOG	GOOG	...	MSFT	MSFT
Date													
2024-10-11 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	AMAZON	NaN	NaN	NaN	...	NaN	NaN
2024-10-14 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	AMAZON	NaN	NaN	NaN	...	NaN	NaN
2024-10-15 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	AMAZON	NaN	NaN	NaN	...	NaN	NaN
2024-10-16 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	AMAZON	NaN	NaN	NaN	...	NaN	NaN
2024-10-17 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	AMAZON	NaN	NaN	NaN	...	NaN	NaN
2024-10-18 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	AMAZON	NaN	NaN	NaN	...	NaN	NaN
2024-10-21 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	AMAZON	NaN	NaN	NaN	...	NaN	NaN
2024-10-22 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	AMAZON	NaN	NaN	NaN	...	NaN	NaN
2024-10-23 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	AMAZON	NaN	NaN	NaN	...	NaN	NaN
2024-10-24 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	AMAZON	NaN	NaN	NaN	...	NaN	NaN

10 rows × 25 columns

In [19]:
```
df.head(10)
```

Out[19]:

Price	Adj Close	Close	High	Low	Open	Volume	company_name	Adj Close	Clos
Ticker	AAPL	AAPL	AAPL	AAPL	AAPL	AAPL		GOOG	GOO
Date									
2023-10-25 00:00:00+00:00	170.228928	171.100006	173.059998	170.649994	171.880005	57157000.0	APPLE	NaN	Nal
2023-10-26 00:00:00+00:00	166.040359	166.889999	171.380005	165.669998	170.369995	70625300.0	APPLE	NaN	Nal
2023-10-27 00:00:00+00:00	167.363586	168.220001	168.960007	166.830002	166.910004	58499100.0	APPLE	NaN	Nal
2023-10-30 00:00:00+00:00	169.423035	170.289993	171.169998	168.869995	169.020004	51131000.0	APPLE	NaN	Nal
2023-10-31 00:00:00+00:00	169.900620	170.770004	170.899994	167.899994	169.350006	44846000.0	APPLE	NaN	Nal
2023-11-01 00:00:00+00:00	173.084320	173.970001	174.229996	170.119995	171.000000	56934900.0	APPLE	NaN	Nal
2023-11-02 00:00:00+00:00	176.665985	177.570007	177.779999	175.460007	175.520004	77334800.0	APPLE	NaN	Nal
2023-11-03 00:00:00+00:00	175.750671	176.649994	176.820007	173.350006	174.240005	79763700.0	APPLE	NaN	Nal
2023-11-06 00:00:00+00:00	178.317520	179.229996	179.429993	176.210007	176.380005	63841300.0	APPLE	NaN	Nal
2023-11-07 00:00:00+00:00	180.894348	181.820007	182.440002	178.970001	179.179993	70530000.0	APPLE	NaN	Nal

10 rows × 25 columns

In [20]:
```
# check if data is downloaded correctly
for ticker in tech_list:
    print(f"{ticker} data:\n", stock_data[ticker].head(), "\n")
```

```
AAPL data:
 Price                      Adj Close       Close        High         Low  \
Ticker                          AAPL        AAPL        AAPL        AAPL
Date
2023-10-25 00:00:00+00:00  170.228928  171.100006  173.059998  170.649994
2023-10-26 00:00:00+00:00  166.040344  166.889999  171.380005  165.669998
2023-10-27 00:00:00+00:00  167.363571  168.220001  168.960007  166.830002
2023-10-30 00:00:00+00:00  169.423035  170.289993  171.169998  168.869995
2023-10-31 00:00:00+00:00  169.900604  170.770004  170.899994  167.899994

Price                           Open    Volume
Ticker                          AAPL      AAPL
Date
2023-10-25 00:00:00+00:00  171.880005  57157000
2023-10-26 00:00:00+00:00  170.369995  70625300
2023-10-27 00:00:00+00:00  166.910004  58499100
2023-10-30 00:00:00+00:00  169.020004  51131000
2023-10-31 00:00:00+00:00  169.350006  44846000

GOOG data:
 Price                      Adj Close       Close        High         Low  \
Ticker                          GOOG        GOOG        GOOG        GOOG
Date
2023-10-25 00:00:00+00:00  126.359680  126.669998  130.100006  126.089996
2023-10-26 00:00:00+00:00  123.137604  123.440002  125.459999  122.320000
2023-10-27 00:00:00+00:00  123.097694  123.400002  124.440002  121.459999
2023-10-30 00:00:00+00:00  125.441940  125.750000  126.550003  123.879997
2023-10-31 00:00:00+00:00  124.993042  125.300003  126.559998  123.925003

Price                           Open    Volume
Ticker                          GOOG      GOOG
Date
2023-10-25 00:00:00+00:00  129.770004  58796100
2023-10-26 00:00:00+00:00  124.470001  33907400
2023-10-27 00:00:00+00:00  124.029999  37367700
2023-10-30 00:00:00+00:00  124.459999  24165600
2023-10-31 00:00:00+00:00  126.269997  21123400

MSFT data:
 Price                      Adj Close       Close        High         Low  \
Ticker                          MSFT        MSFT        MSFT        MSFT
Date
2023-10-25 00:00:00+00:00  338.131989  340.670013  346.200012  337.619995
2023-10-26 00:00:00+00:00  325.447205  327.890015  341.630005  326.940002
2023-10-27 00:00:00+00:00  327.352875  329.809998  336.720001  328.399994
2023-10-30 00:00:00+00:00  334.796997  337.309998  339.450012  331.829987
2023-10-31 00:00:00+00:00  335.591034  338.109985  339.000000  334.690002

Price                           Open    Volume
Ticker                          MSFT      MSFT
Date
2023-10-25 00:00:00+00:00  345.019989  55053800
2023-10-26 00:00:00+00:00  340.540009  37828500
2023-10-27 00:00:00+00:00  330.429993  29856500
2023-10-30 00:00:00+00:00  333.410004  22828100
2023-10-31 00:00:00+00:00  338.850006  20265300

AMZN data:
 Price                      Adj Close       Close        High         Low  \
Ticker                          AMZN        AMZN        AMZN        AMZN
Date
2023-10-25 00:00:00+00:00  121.389999  121.389999  126.339996  120.790001
2023-10-26 00:00:00+00:00  119.570000  119.570000  121.639999  118.349998
2023-10-27 00:00:00+00:00  127.739998  127.739998  130.020004  125.519997
```

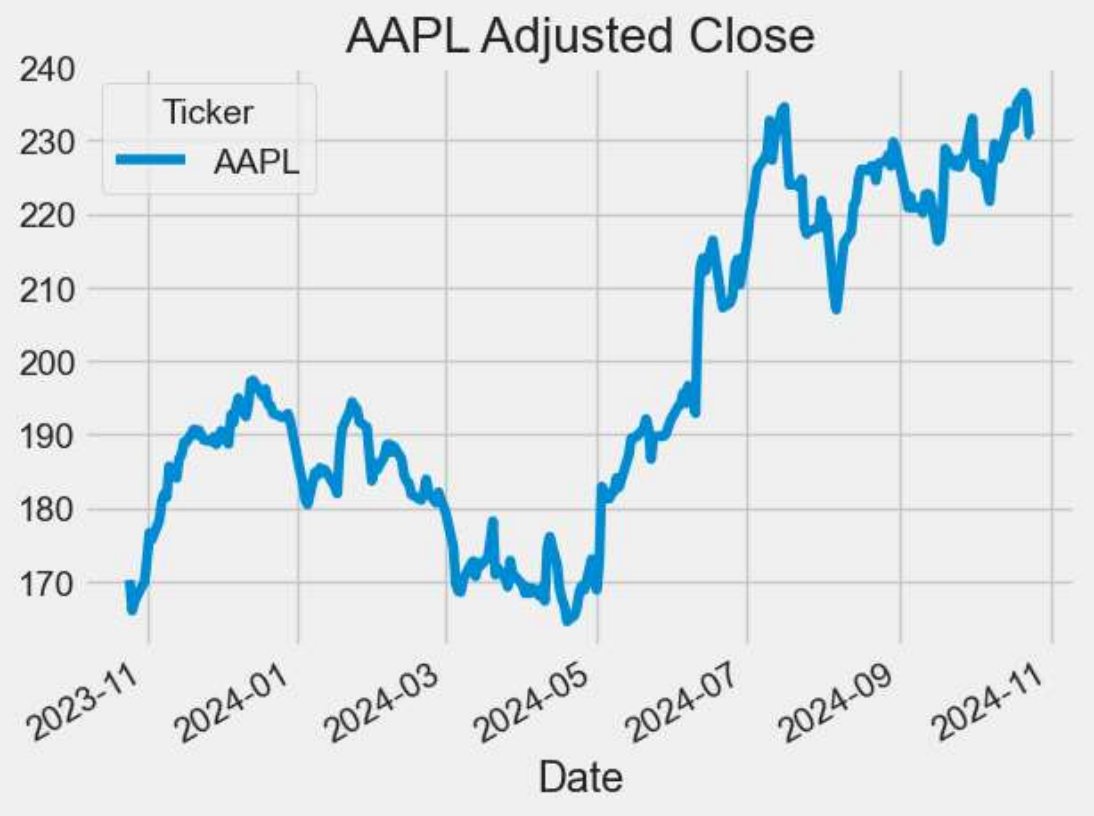
```
2023-10-30 00:00:00+00:00  132.710007  132.710007  133.000000  128.559998
2023-10-31 00:00:00+00:00  133.089996  133.089996  133.570007  131.710007

Price                            Open     Volume
Ticker                           AMZN       AMZN
Date
2023-10-25 00:00:00+00:00  126.040001   74577500
2023-10-26 00:00:00+00:00  120.629997  100419500
2023-10-27 00:00:00+00:00  126.199997  125309300
2023-10-30 00:00:00+00:00  129.720001   72485500
2023-10-31 00:00:00+00:00  132.750000   51589400
```

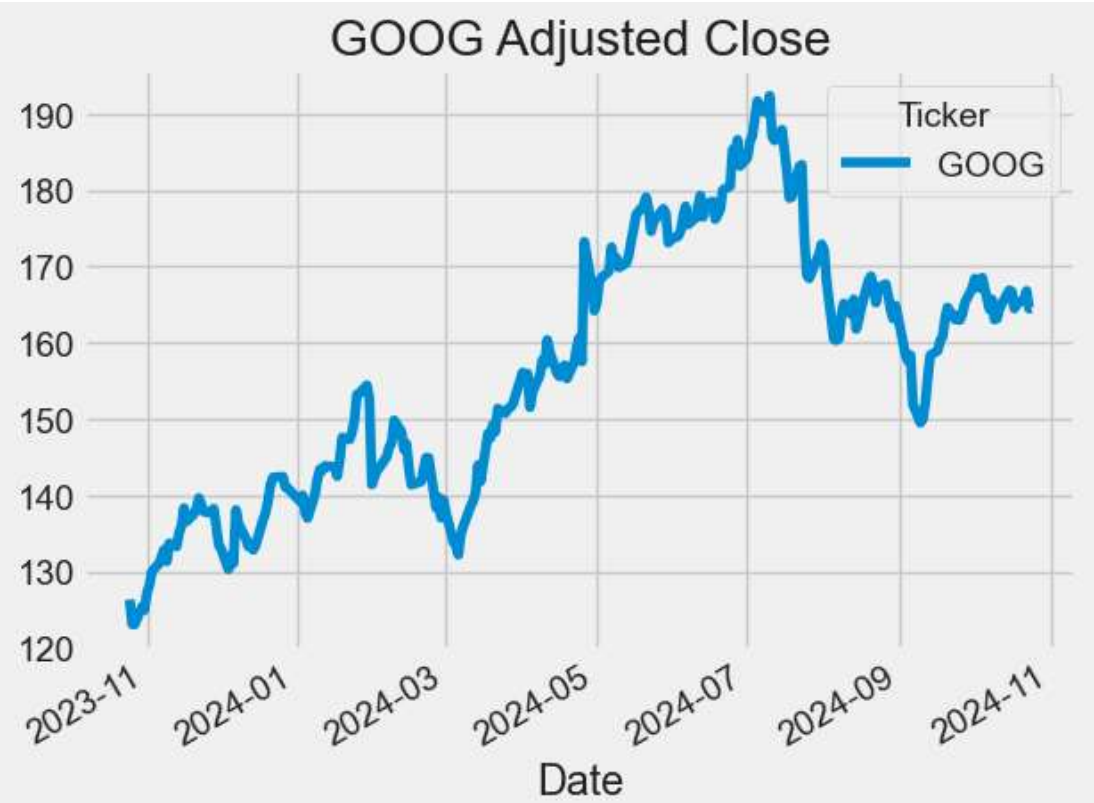
In [21]:
```
# Check if 'Adj Close' exists
for ticker in tech_list:
    print(f"{ticker} columns:\n", stock_data[ticker].columns, "\n")
```

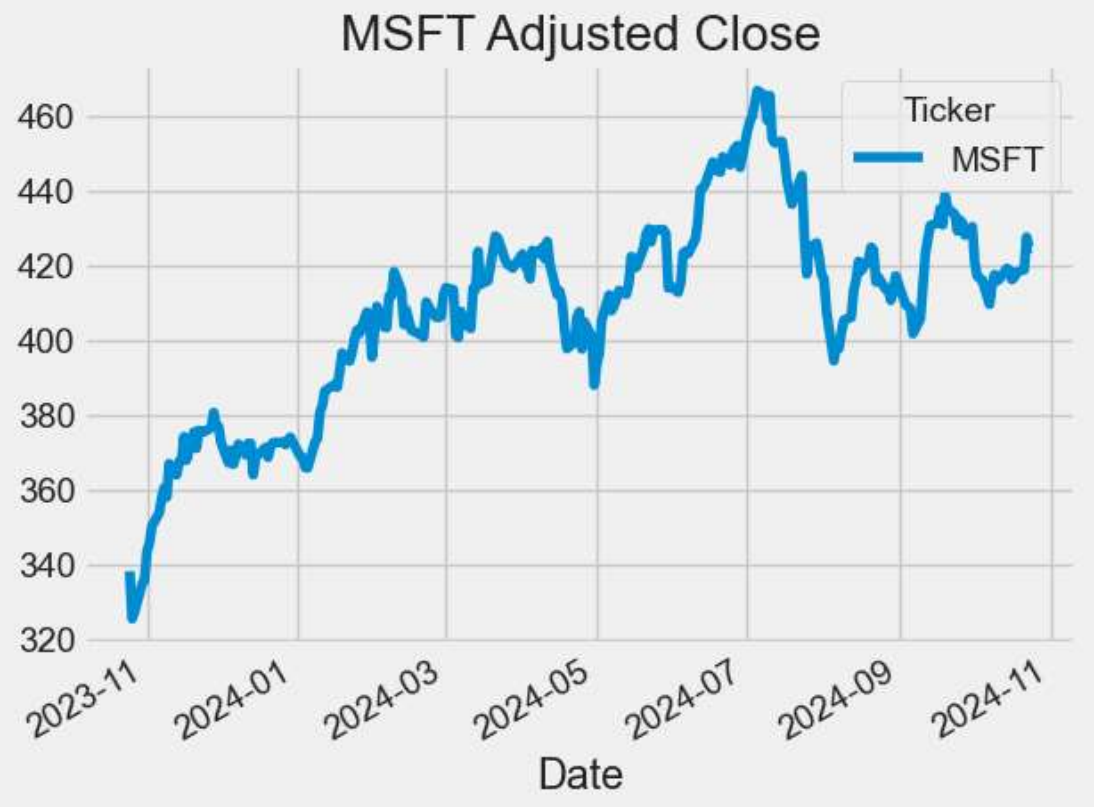
```
AAPL columns:
 MultiIndex([('Adj Close', 'AAPL'),
             (    'Close', 'AAPL'),
             (     'High', 'AAPL'),
             (      'Low', 'AAPL'),
             (     'Open', 'AAPL'),
             (   'Volume', 'AAPL')],
            names=['Price', 'Ticker'])

GOOG columns:
 MultiIndex([('Adj Close', 'GOOG'),
             (    'Close', 'GOOG'),
             (     'High', 'GOOG'),
             (      'Low', 'GOOG'),
             (     'Open', 'GOOG'),
             (   'Volume', 'GOOG')],
            names=['Price', 'Ticker'])

MSFT columns:
 MultiIndex([('Adj Close', 'MSFT'),
             (    'Close', 'MSFT'),
             (     'High', 'MSFT'),
             (      'Low', 'MSFT'),
             (     'Open', 'MSFT'),
             (   'Volume', 'MSFT')],
            names=['Price', 'Ticker'])

AMZN columns:
 MultiIndex([('Adj Close', 'AMZN'),
             (    'Close', 'AMZN'),
             (     'High', 'AMZN'),
             (      'Low', 'AMZN'),
             (     'Open', 'AMZN'),
             (   'Volume', 'AMZN')],
            names=['Price', 'Ticker'])
```

In [22]:
```
#Closing Price
plt.figure(figsize=(10, 10))

stock_data['AAPL']['Adj Close'].plot()
plt.title("AAPL Adjusted Close")
plt.show()

stock_data['GOOG']['Adj Close'].plot()
plt.title("GOOG Adjusted Close")
plt.show()

stock_data['MSFT']['Adj Close'].plot()
plt.title("MSFT Adjusted Close")
plt.show()

stock_data['AMZN']['Adj Close'].plot()
plt.title("AMZN Adjusted Close")
plt.show()
```

```
<Figure size 1000x1000 with 0 Axes>
```




GOOG Adjusted Close
Ticker
GOOG
190
180
170
160
150
140
130
120
2023-11
2024-01
2024-03
2024-05
2024-07
2024-09
2024-11
Date

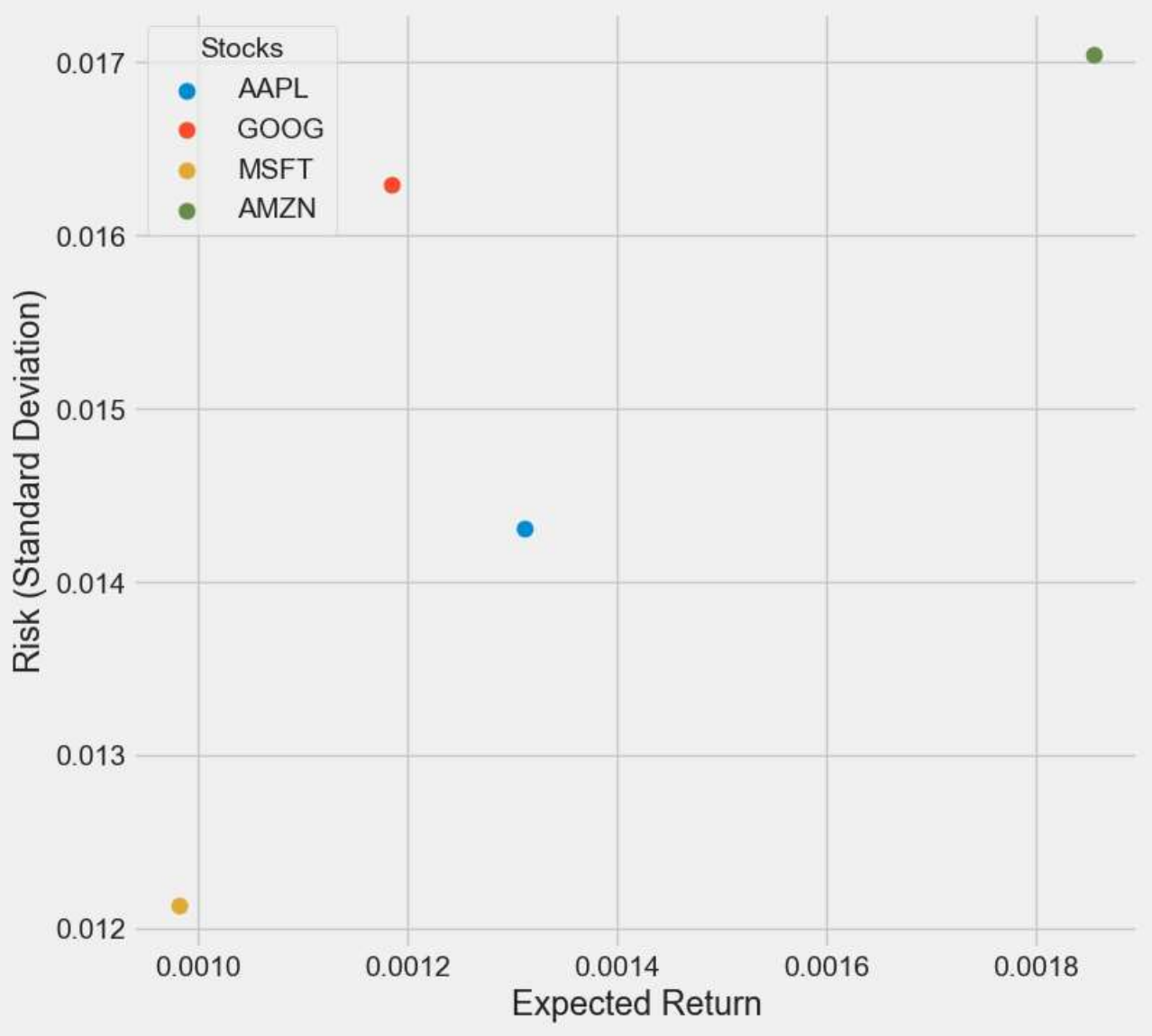

MSFT Adjusted Close
Ticker
MSFT
460
440
420
400
380
360
340
320
2023-11
2024-01
2024-03
2024-05
2024-07
2024-09
2024-11
Date

In [23]:
```
closing_df = pd.DataFrame()

# DataFrame of 'Adj Close' prices for each stock
for stock in tech_list:
    closing_df[stock] = stock_data[stock]['Adj Close']

# New returns DataFrame (percentage change)
tech_rets = closing_df.pct_change()

tech_rets.head()
```

Out[23]:

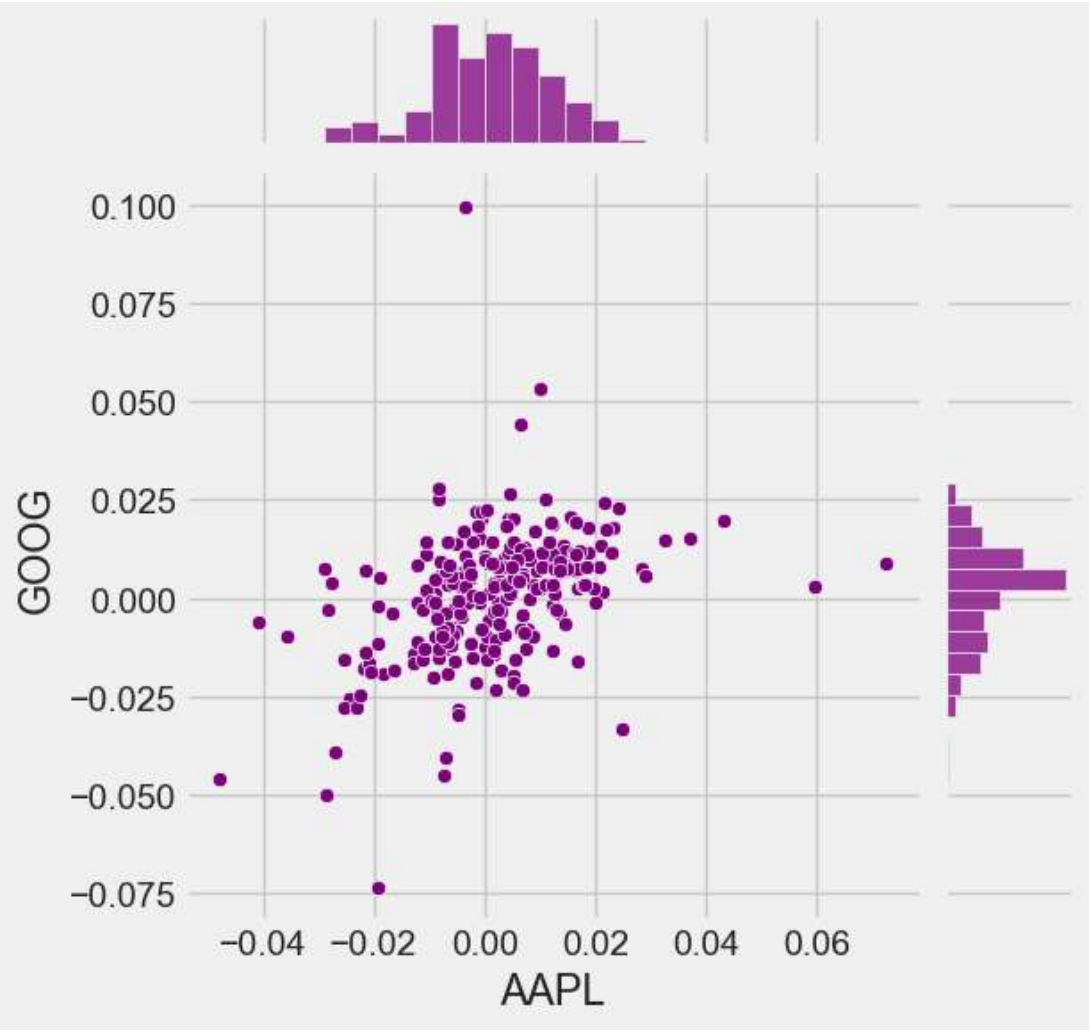
Date	AAPL	GOOG	MSFT	AMZN
2023-10-25 00:00:00+00:00	NaN	NaN	NaN	NaN
2023-10-26 00:00:00+00:00	-0.024606	-0.025499	-0.037514	-0.014993
2023-10-27 00:00:00+00:00	0.007969	-0.000324	0.005856	0.068328
2023-10-30 00:00:00+00:00	0.012305	0.019044	0.022740	0.038907
2023-10-31 00:00:00+00:00	0.002819	-0.003579	0.002372	0.002863

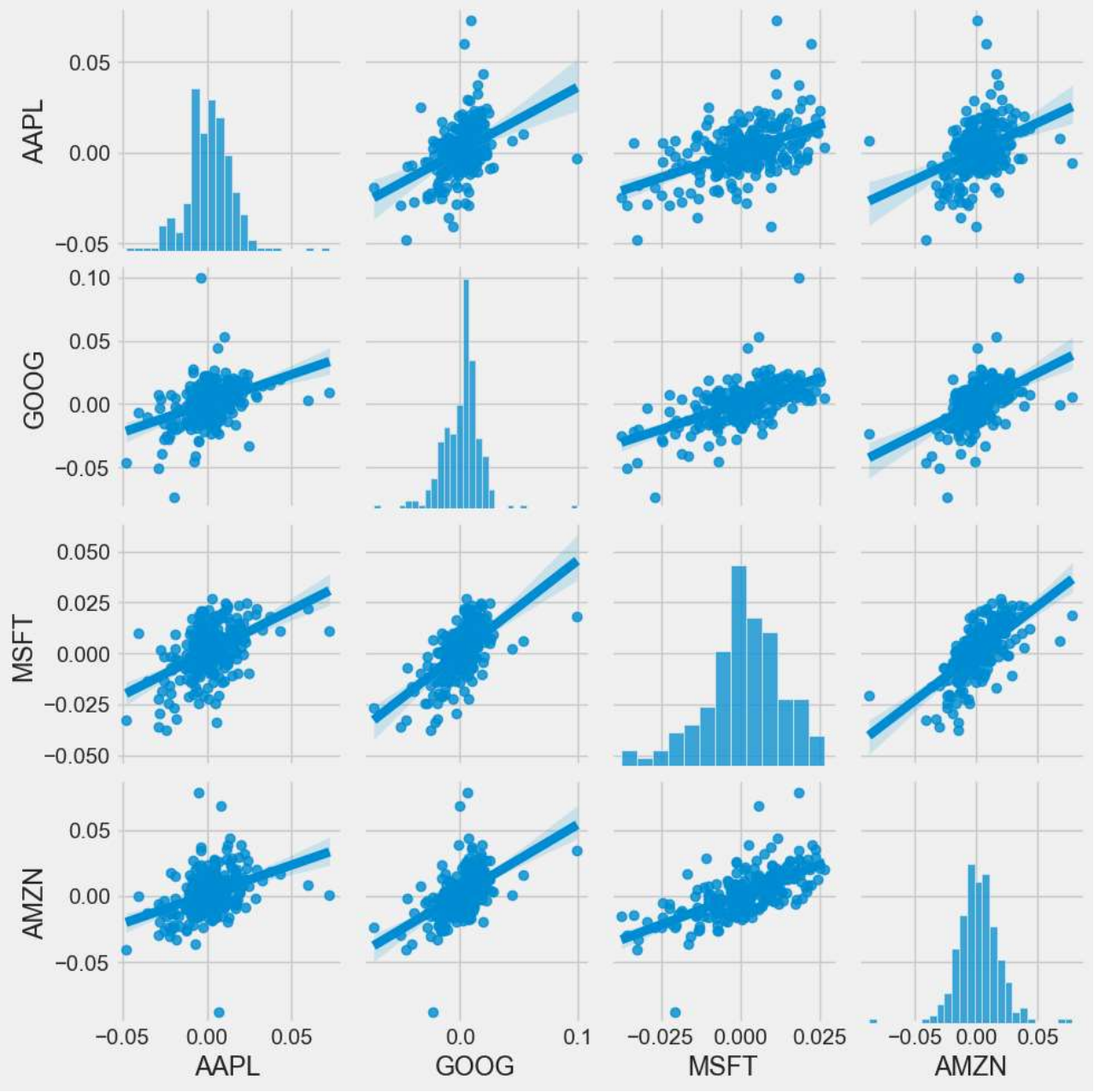
In [24]:
```
rets = tech_rets.dropna()

area = np.pi * 20

plt.figure(figsize=(8, 8))

for label in rets.columns:
    plt.scatter(rets[label].mean(), rets[label].std(), s=area, label=label)

plt.xlabel('Expected Return')
plt.ylabel('Risk (Standard Deviation)')

plt.legend(title='Stocks')

plt.show()
```



In [25]:
```
# compare the daily percentage return of two stocks to check correlation
sns.jointplot(x='AAPL', y='GOOG', data=tech_rets, kind='scatter', color='purple')

# Comparison Analysis for all combinations
sns.pairplot(tech_rets, kind='reg')
```

Out[25]: <seaborn.axisgrid.PairGrid at 0x22daaad9810>


0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
GOOG
−0.04
−0.02
0.00
0.02
0.04
0.06
AAPL



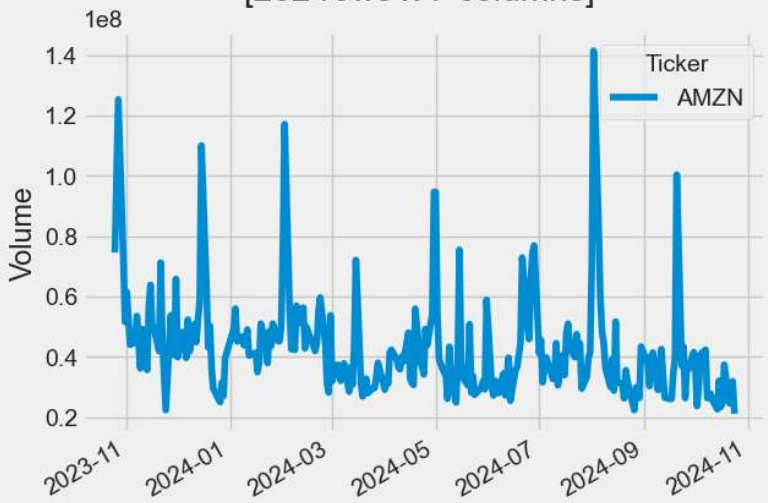
In [26]:
```
#Volume of Sales
plt.figure(figsize=(10, 10))

company['Volume'].plot()
plt.ylabel('Volume')
plt.xlabel(None)
plt.title(f"Sales Volume for {AAPL} ")

plt.tight_layout()
```

```
C:\Users\shubh\AppData\Local\Temp\ipykernel_8108\3274364906.py:9: UserWarning: Tight layout not app
lied. The bottom and top margins cannot be made large enough to accommodate all axes decorations.
  plt.tight_layout()
<Figure size 1000x1000 with 0 Axes>
```

Sales Volume for Price	Adj Close	Close	High	Low
Ticker	AAPL	AAPL	AAPL	AAPL
Date				
2023-10-25 00:00:00+00:00	170.228928	171.100006	173.059998	170.649994
2023-10-26 00:00:00+00:00	166.040359	166.889999	171.380005	165.669998
2023-10-27 00:00:00+00:00	167.363586	168.220001	168.960007	166.830002
2023-10-30 00:00:00+00:00	169.423035	170.289993	171.169998	168.869995
2023-10-31 00:00:00+00:00	169.900620	170.770004	170.899994	167.899994
...	...	...	...	...
2024-10-18 00:00:00+00:00	235.000000	235.000000	236.179993	234.009995
2024-10-21 00:00:00+00:00	236.479996	236.479996	236.850006	234.449997
2024-10-22 00:00:00+00:00	235.860001	235.860001	236.220001	232.600006
2024-10-23 00:00:00+00:00	230.759995	230.759995	235.139999	227.759995
2024-10-24 00:00:00+00:00	230.570007	230.570007	230.820007	228.410004

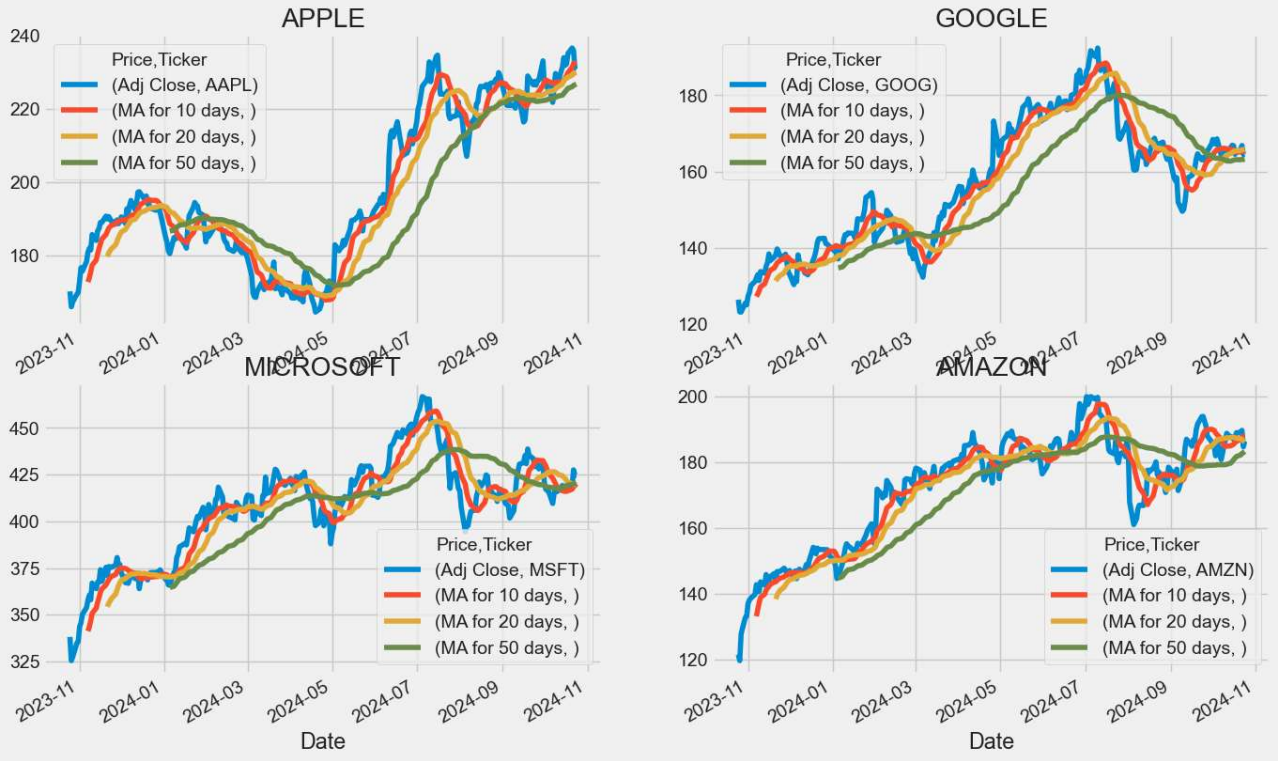
Price	Open	Volume	company_name
Ticker	AAPL	AAPL	
Date			
2023-10-25 00:00:00+00:00	171.880005	57157000	APPLE
2023-10-26 00:00:00+00:00	170.369995	70625300	APPLE
2023-10-27 00:00:00+00:00	166.910004	58499100	APPLE
2023-10-30 00:00:00+00:00	169.020004	51131000	APPLE
2023-10-31 00:00:00+00:00	169.350006	44846000	APPLE
...	...	...	...
2024-10-18 00:00:00+00:00	236.179993	46431500	APPLE
2024-10-21 00:00:00+00:00	234.449997	36254500	APPLE
2024-10-22 00:00:00+00:00	233.889999	38846600	APPLE
2024-10-23 00:00:00+00:00	234.080002	52287000	APPLE
2024-10-24 00:00:00+00:00	229.917007	29318128	APPLE

[252 rows x 7 columns]



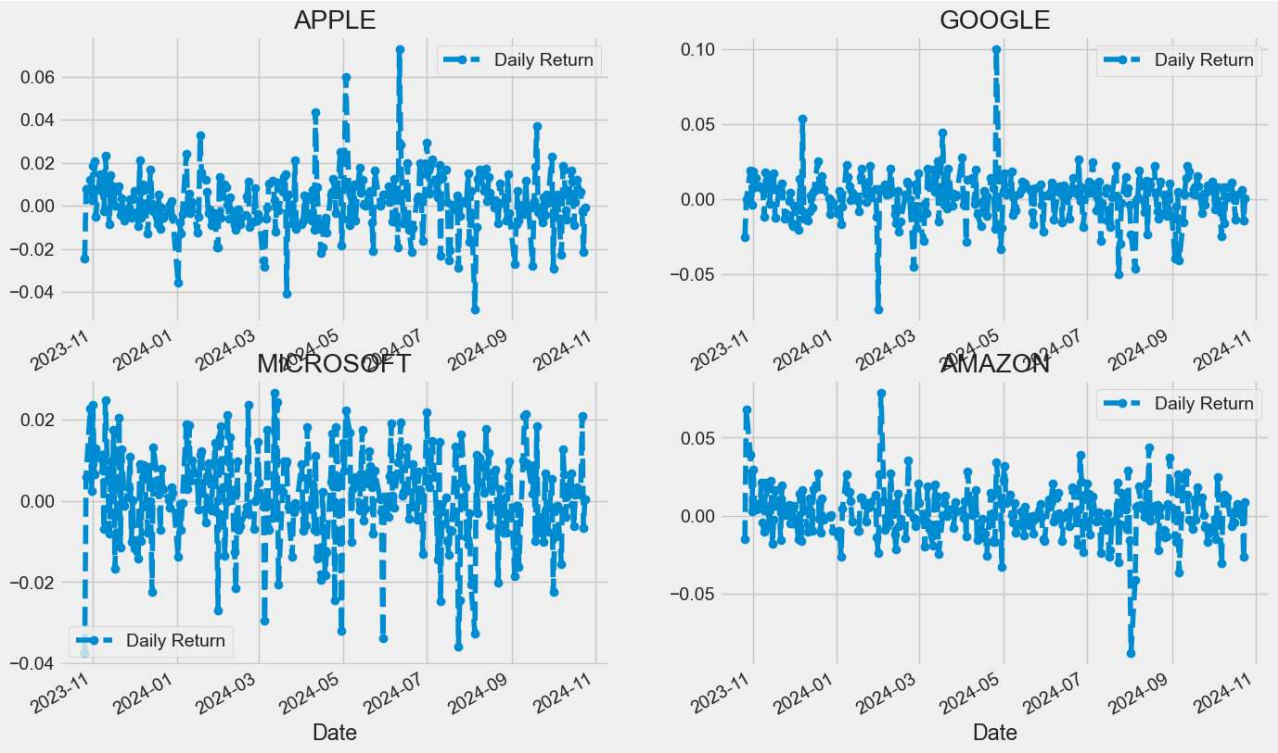
In [27]:
```
#MOving Average
ma_day = [10, 20, 50]

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


fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_figheight(10)
fig.set_figwidth(15)

AAPL[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,0])
axes[0,0].set_title('APPLE')

GOOG[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,1])
axes[0,1].set_title('GOOGLE')

MSFT[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,0])
axes[1,0].set_title('MICROSOFT')

AMZN[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,1])
axes[1,1].set_title('AMAZON')
```

Out[27]: Text(0.5, 1.0, 'AMAZON')

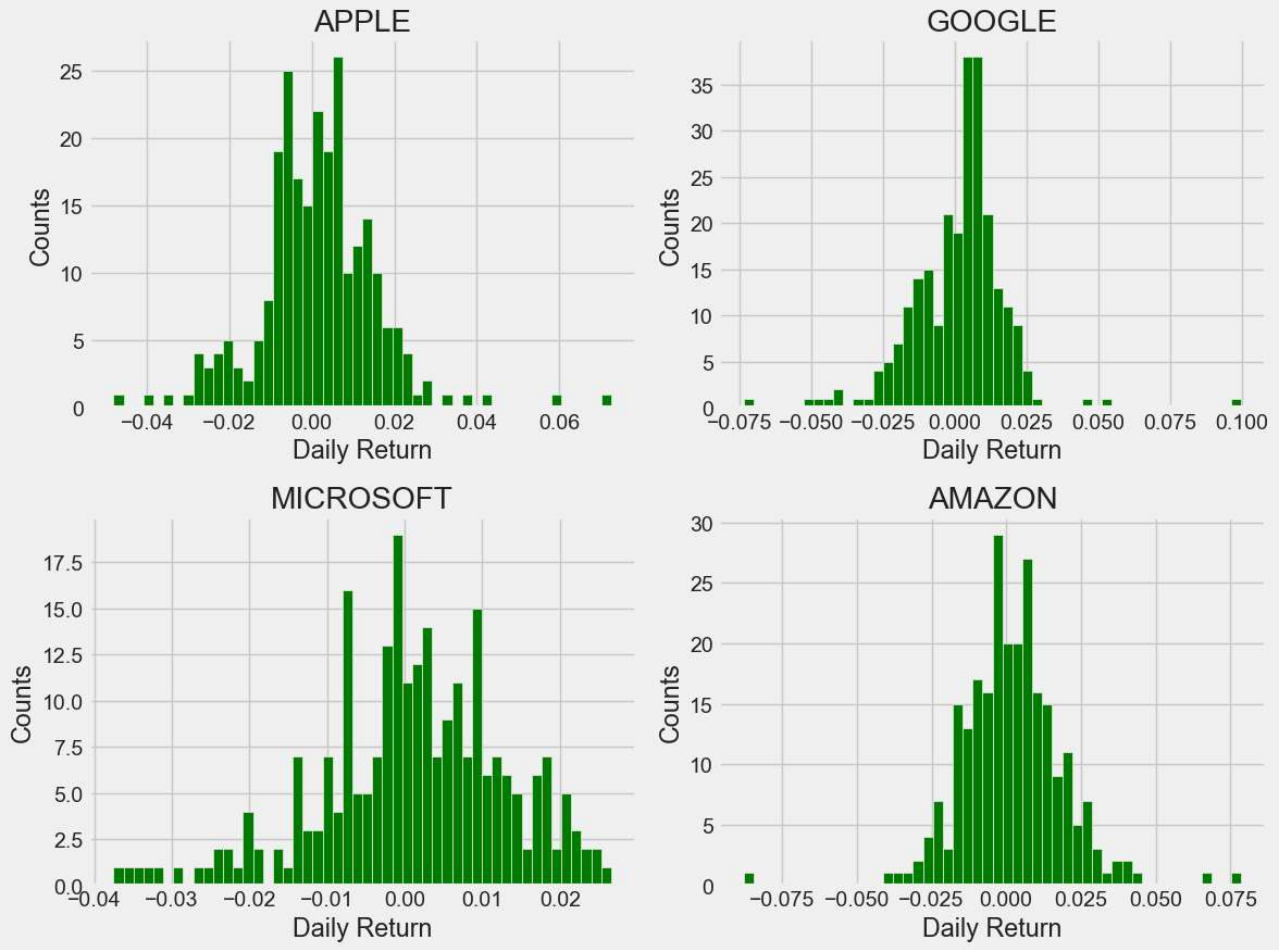


In [28]:
```
#daily return for stocks
for company in company_list:
    company['Daily Return'] = company['Adj Close'].pct_change()

# plotting daily return percentage
fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_figheight(10)
fig.set_figwidth(15)

AAPL['Daily Return'].plot(ax=axes[0,0], legend=True, linestyle='--', marker='o')
axes[0,0].set_title('APPLE')

GOOG['Daily Return'].plot(ax=axes[0,1], legend=True, linestyle='--', marker='o')
axes[0,1].set_title('GOOGLE')

MSFT['Daily Return'].plot(ax=axes[1,0], legend=True, linestyle='--', marker='o')
axes[1,0].set_title('MICROSOFT')

AMZN['Daily Return'].plot(ax=axes[1,1], legend=True, linestyle='--', marker='o')
axes[1,1].set_title('AMAZON')
```

Out[28]: Text(0.5, 1.0, 'AMAZON')

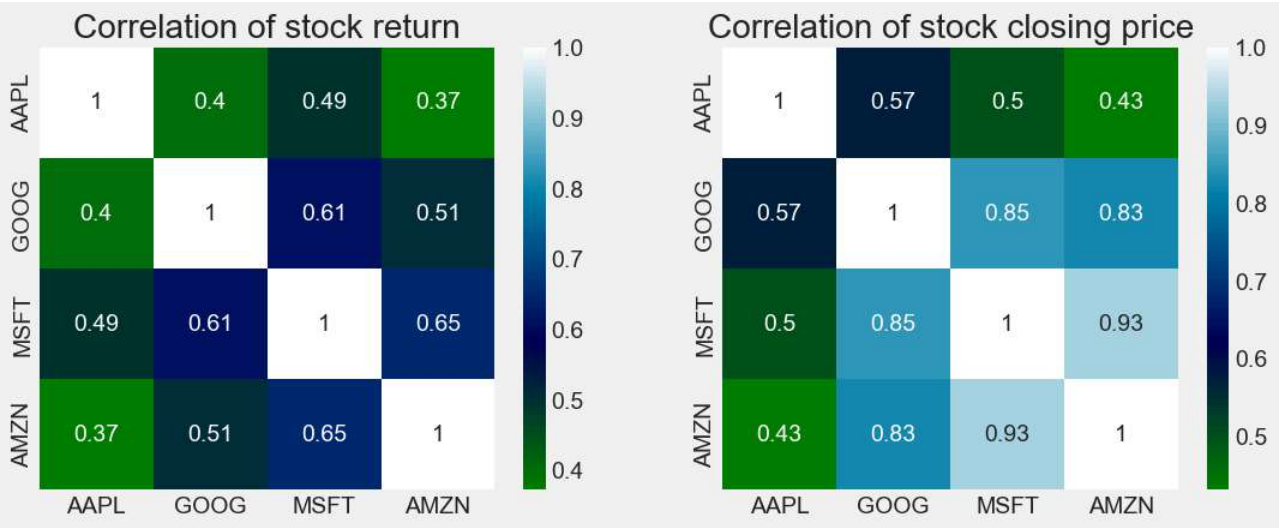


In [29]:
```
plt.figure(figsize=(12, 9))

for i, company in enumerate(company_list, 1):
    plt.subplot(2, 2, i)
    company['Daily Return'].hist(bins=50, color='green')
    plt.xlabel('Daily Return')
    plt.ylabel('Counts')
    plt.title(f'{company_name[i - 1]}')

plt.tight_layout()
```



In [30]:
```
plt.figure(figsize=(12, 10))

#correlation of stock return
plt.subplot(2, 2, 1)
sns.heatmap(tech_rets.corr(), annot=True, cmap='ocean')
plt.title('Correlation of stock return')

#correlation of stock closing price
plt.subplot(2, 2, 2)
sns.heatmap(closing_df.corr(), annot=True, cmap='ocean')
plt.title('Correlation of stock closing price')
```

Out[30]: Text(0.5, 1.0, 'Correlation of stock closing price')



In [31]:
```
# TIME SERIES FORECASTING USING ARIMA FOR GOOGLE STOCK PRICES
```

In [32]:
```
import datetime
from datetime import date, timedelta
today = date.today()
d1 = today.strftime("%Y-%m-%d")
end_date = d1
d2 = date.today() - timedelta(days=365)
d2 = d2.strftime("%Y-%m-%d")
start_date = d2

data = yf.download('GOOG',
                   start=start_date,
                   end=end_date,
                   progress=False)
data["Date"] = data.index
data = data[["Date", "Open", "High", "Low", "Close", "Adj Close", "Volume"]]
data.reset_index(drop=True, inplace=True)
print(data.tail())
```

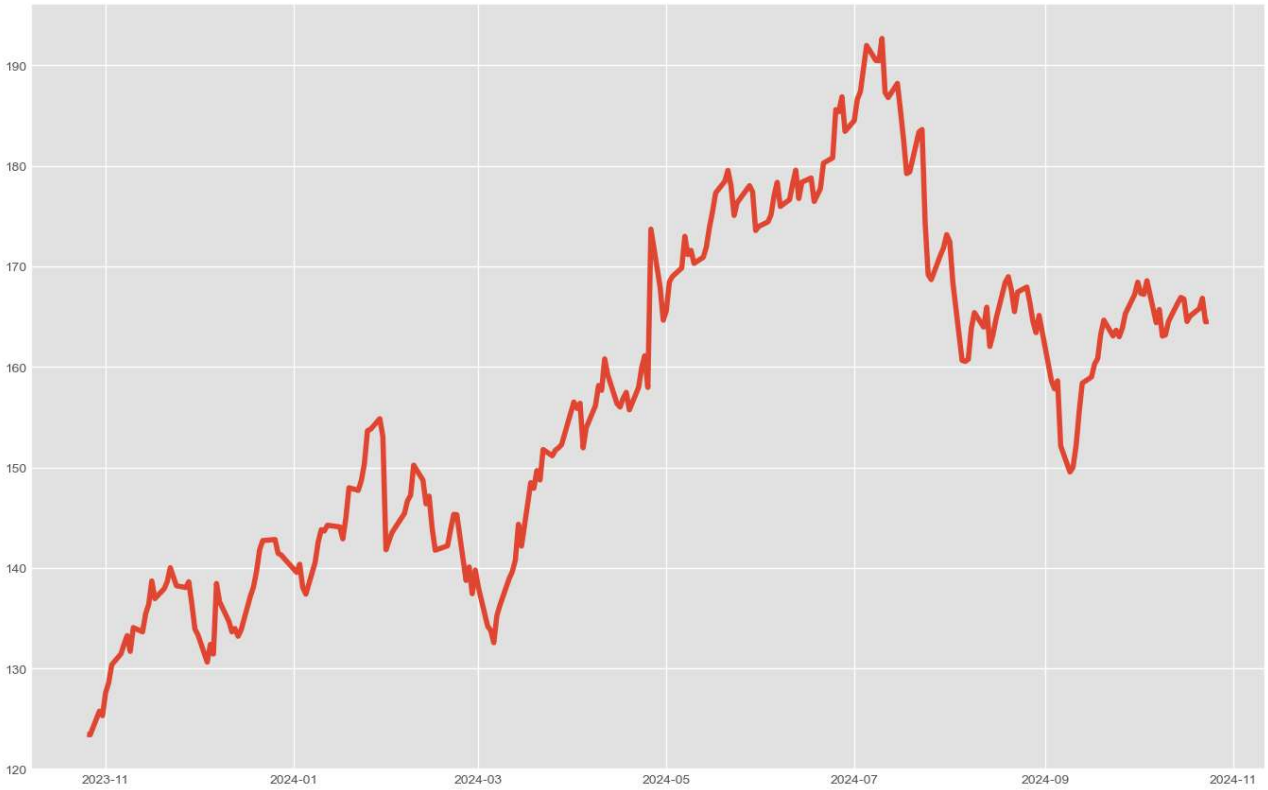
```
Price                       Date        Open        High         Low  \
Ticker                                  GOOG        GOOG        GOOG
246    2024-10-18 00:00:00+00:00  164.869995  166.369995  164.750000
247    2024-10-21 00:00:00+00:00  164.580002  166.220001  164.304993
248    2024-10-22 00:00:00+00:00  164.699997  167.470001  164.669998
249    2024-10-23 00:00:00+00:00  166.429993  167.600006  163.632996
250    2024-10-24 00:00:00+00:00  164.589996  165.050003  162.770004

Price        Close   Adj Close    Volume
Ticker        GOOG        GOOG      GOOG
246     165.050003  165.050003  13091300
247     165.800003  165.800003  11384000
248     166.820007  166.820007  11958600
249     164.479996  164.479996  12754300
250     164.529999  164.529999  12520560
```

In [33]:
```
data = data[["Date", "Close"]]
print(data.head())
```

```
Price                       Date       Close
Ticker                                  GOOG
0      2023-10-26 00:00:00+00:00  123.440002
1      2023-10-27 00:00:00+00:00  123.400002
2      2023-10-30 00:00:00+00:00  125.750000
3      2023-10-31 00:00:00+00:00  125.300003
4      2023-11-01 00:00:00+00:00  127.570000
```

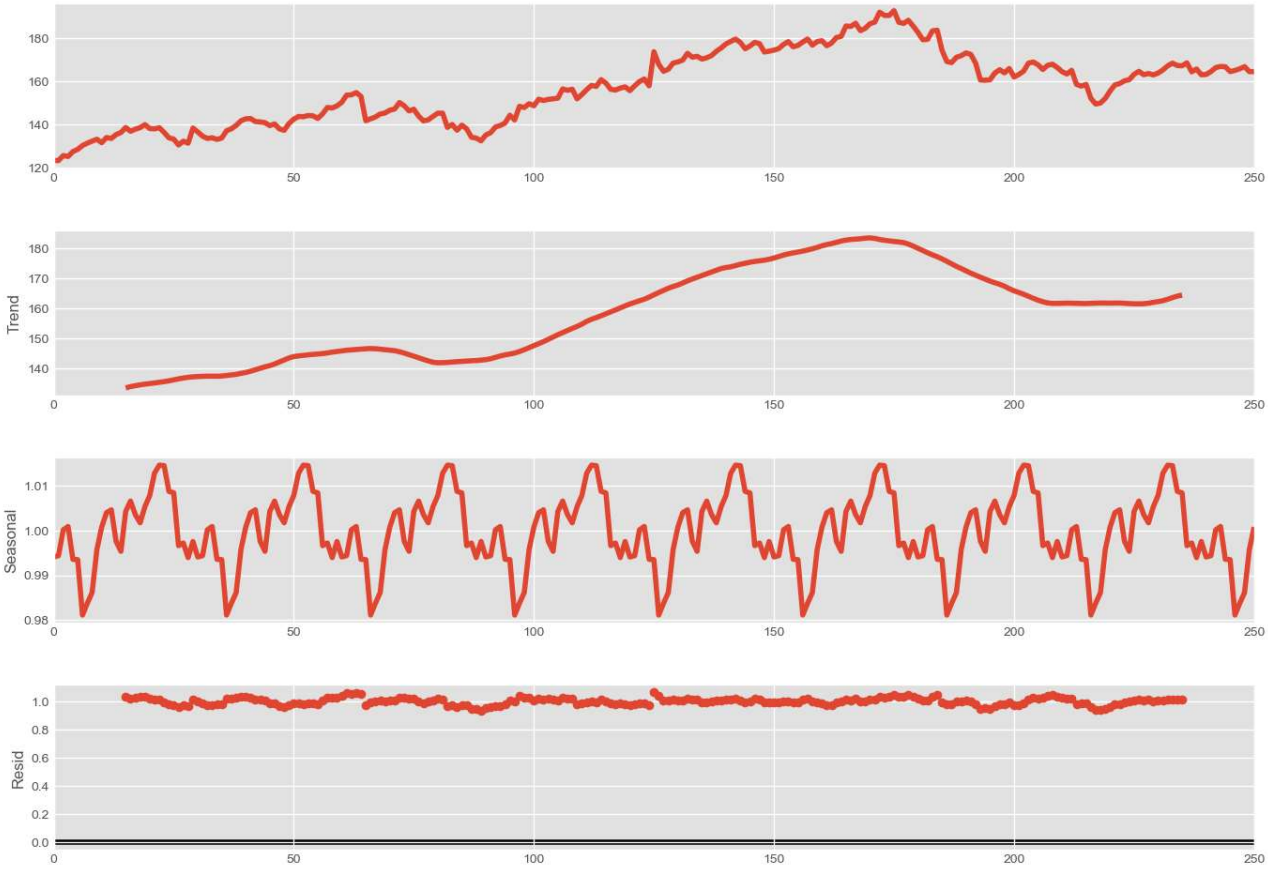
In [34]:
```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
plt.figure(figsize=(15, 10))
plt.plot(data["Date"], data["Close"])
```

Out[34]:
```
[<matplotlib.lines.Line2D at 0x22daa857ed0>]
```



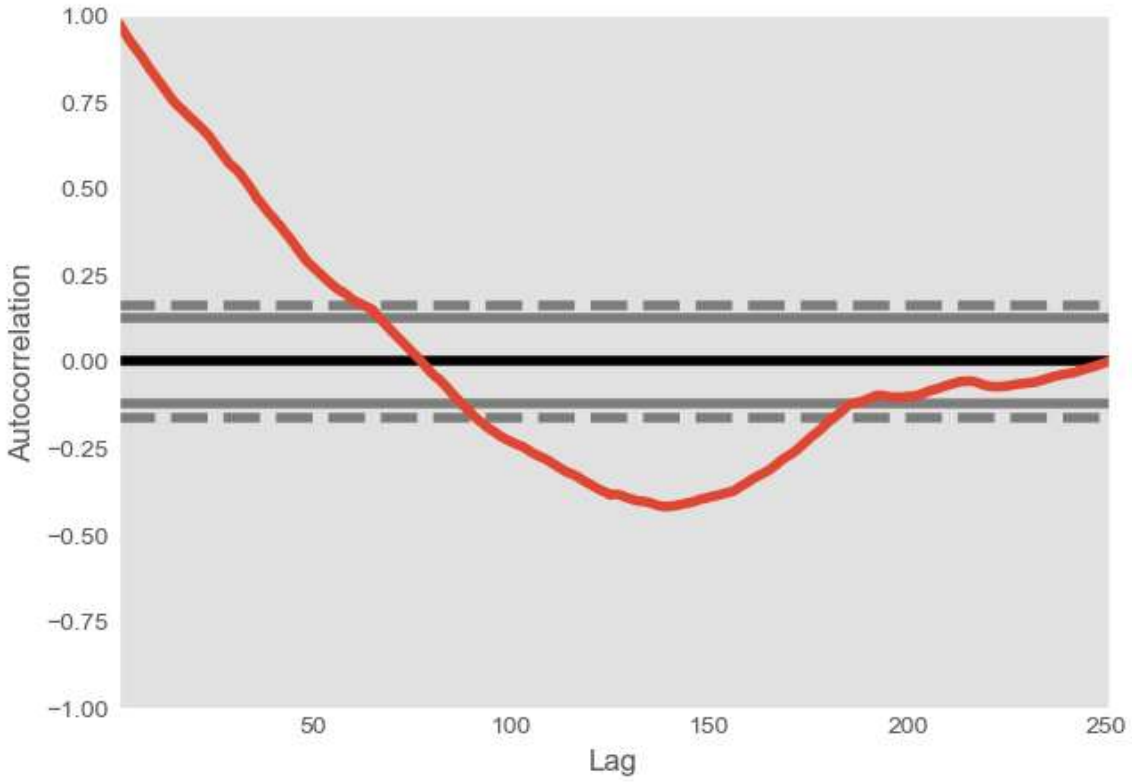
In [35]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(data["Close"],
                            model='multiplicative', period = 30)
fig = plt.figure()
fig = result.plot()
fig.set_size_inches(15, 10)
```

```
<Figure size 640x480 with 0 Axes>
```



In [36]:
```
pd.plotting.autocorrelation_plot(data["Close"])
```

Out[36]: <Axes: xlabel='Lag', ylabel='Autocorrelation'>

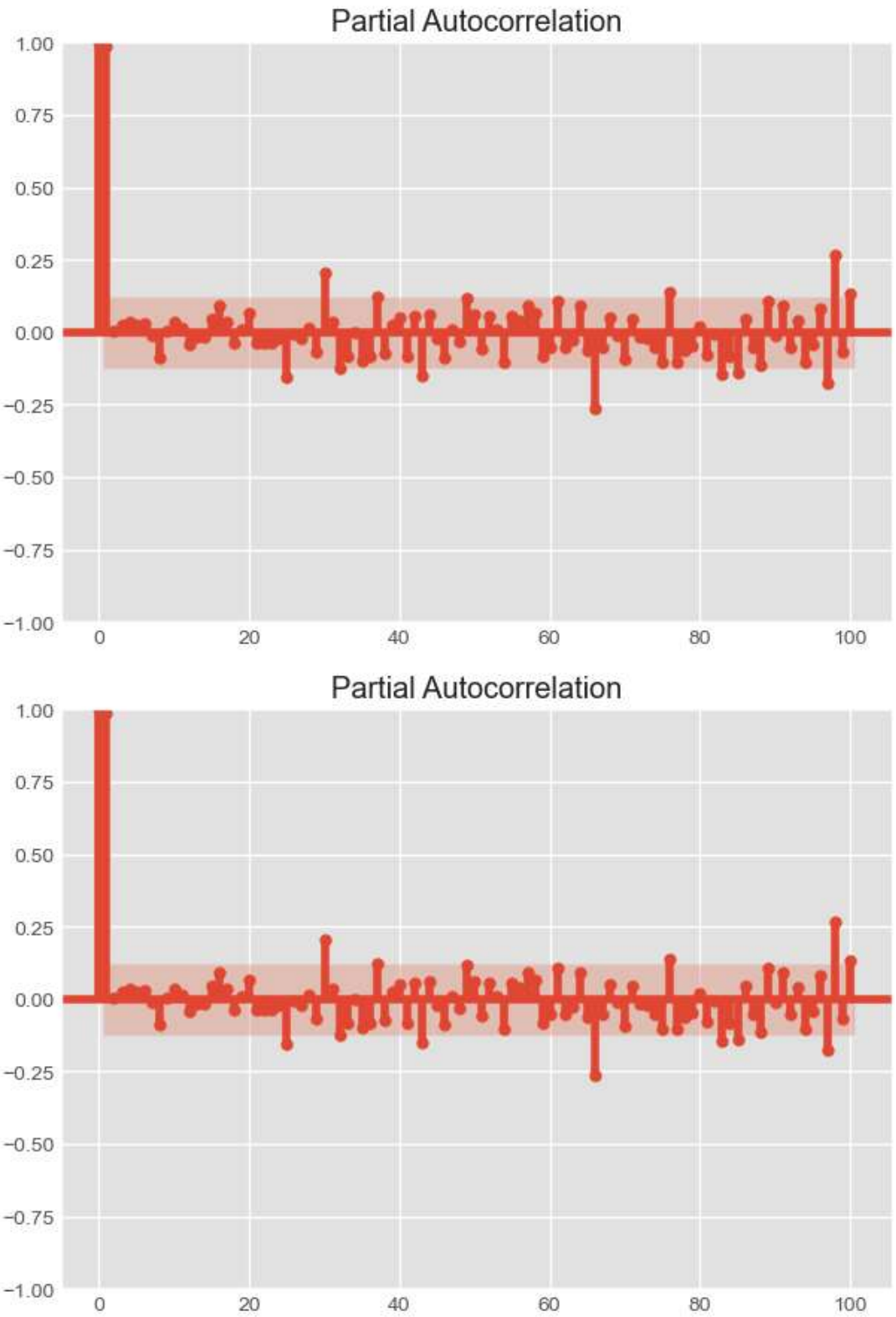


In [37]:
```
# curve is moving down after the 10th line of the first boundary, therefore p = 10
```

In [38]:
```
from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(data["Close"], lags = 100)
```

```
C:\Users\shubh\anaconda3\Lib\site-packages\statsmodels\graphics\tsaplots.py:348: FutureWarning: The
default method 'yw' can produce PACF values outside of the [-1,1] interval. After 0.13, the default
will change tounadjusted Yule-Walker ('ywm'). You can use this method now by setting method='ywm'.
  warnings.warn(
```

Out[38]:



In [39]:
```
# 2 points are far away from others, therefore q=2
# since data is seasonal , d = 1
```

In [40]:
```
import statsmodels.api as sm
import matplotlib.pyplot as plt
p, d, q = 10, 1, 2

# Import the ARIMA class from the correct module
from statsmodels.tsa.arima.model import ARIMA

# Fit the ARIMA model
model = ARIMA(data["Close"], order=(p, d, q))
fitted = model.fit()

# Print the summary of the fitted model
print(fitted.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                   GOOG   No. Observations:                  251
Model:                ARIMA(10, 1, 2)   Log Likelihood                -585.204
Date:                Fri, 25 Oct 2024   AIC                           1196.408
Time:                        02:32:57   BIC                           1242.187
Sample:                             0   HQIC                          1214.833
                                - 251
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.0156      0.848     -1.198      0.231      -2.677       0.646
ar.L2         -0.2448      0.808     -0.303      0.762      -1.829       1.339
ar.L3         -0.0667      0.108     -0.615      0.539      -0.279       0.146
ar.L4         -0.0544      0.102     -0.531      0.595      -0.255       0.146
ar.L5         -0.0294      0.106     -0.277      0.782      -0.237       0.179
ar.L6          0.0109      0.109      0.100      0.920      -0.203       0.224
ar.L7          0.1562      0.110      1.416      0.157      -0.060       0.372
ar.L8          0.1451      0.182      0.799      0.424      -0.211       0.501
ar.L9          0.0039      0.181      0.021      0.983      -0.350       0.358
ar.L10        -0.0821      0.095     -0.867      0.386      -0.268       0.103
ma.L1          0.9975      0.846      1.180      0.238      -0.660       2.655
ma.L2          0.2164      0.785      0.276      0.783      -1.322       1.755
sigma2         6.3129      0.336     18.773      0.000       5.654       6.972
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):               487.27
Prob(Q):                              0.95   Prob(JB):                         0.00
Heteroskedasticity (H):               1.29   Skew:                            -0.04
Prob(H) (two-sided):                  0.25   Kurtosis:                         9.84
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

In [41]:
```
predictions = fitted.predict()
print(predictions)
```

```
0        0.000000
1      123.439985
2      123.400844
3      125.699909
4      125.286550
          ...
246    164.277639
247    165.289837
248    165.852366
249    167.469590
250    164.165160
Name: predicted_mean, Length: 251, dtype: float64
```

In [42]:
```
import warnings
model=sm.tsa.statespace.SARIMAX(data['Close'],
                                order=(p, d, q),
                                seasonal_order=(p, d, q, 12))
model=model.fit()
print(model.summary())
```

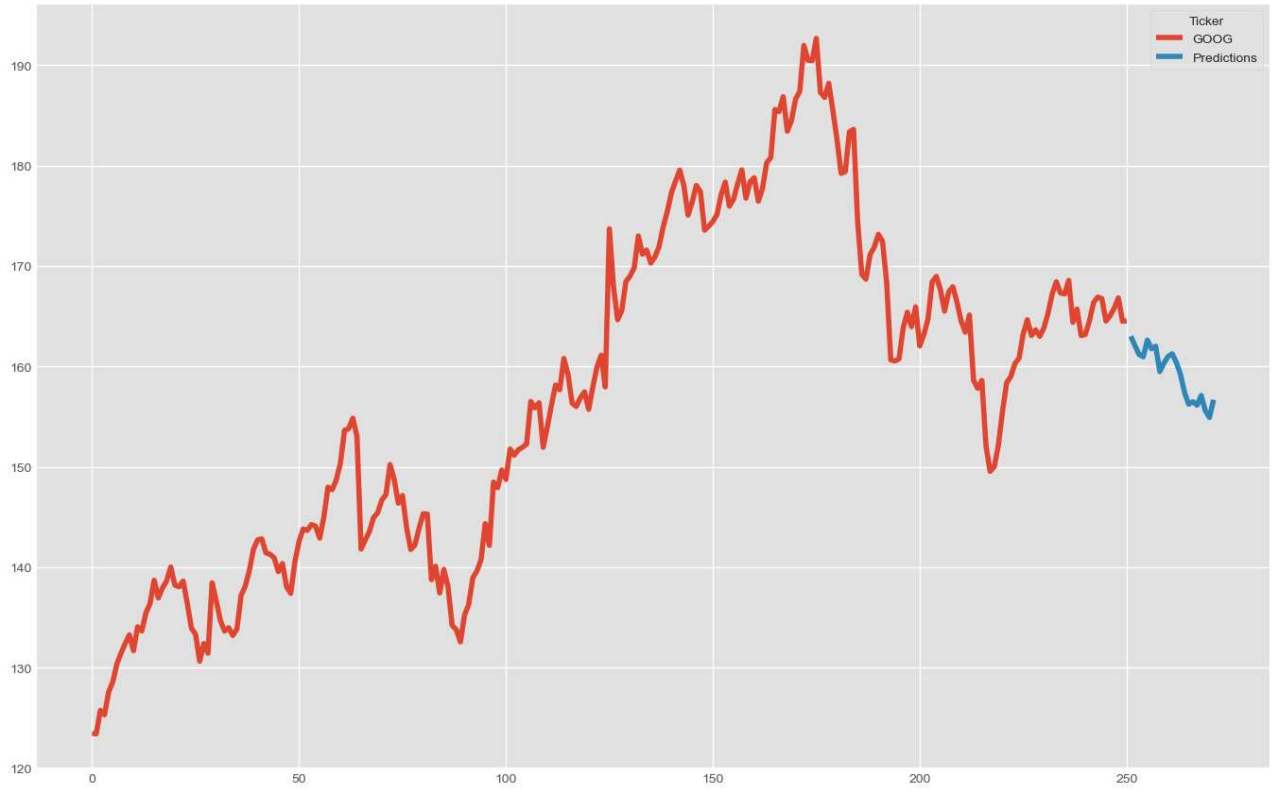
```
C:\Users\shubh\anaconda3\Lib\site-packages\statsmodels\base\model.py:604: ConvergenceWarning: Maxim
um Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                GOOG   No. Observations:                  251
Model:             SARIMAX(10, 1, 2)x(10, 1, 2, 12)   Log Likelihood                -568.748
Date:                              Fri, 25 Oct 2024   AIC                            1187.496
Time:                                      02:37:27   BIC                            1274.303
Sample:                                           0   HQIC                           1222.481
                                                - 251
Covariance Type:                                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.8353      1.219     -0.685      0.493      -3.225       1.554
ar.L2         -0.0398      1.127     -0.035      0.972      -2.249       2.170
ar.L3         -0.0269      0.099     -0.271      0.787      -0.222       0.168
ar.L4         -0.0404      0.120     -0.338      0.736      -0.275       0.194
ar.L5          0.0054      0.139      0.039      0.969      -0.268       0.279
ar.L6          0.0203      0.112      0.181      0.856      -0.200       0.241
ar.L7          0.1235      0.112      1.103      0.270      -0.096       0.343
ar.L8          0.1279      0.205      0.623      0.533      -0.275       0.530
ar.L9          0.0300      0.205      0.146      0.884      -0.373       0.433
ar.L10        -0.0747      0.120     -0.623      0.533      -0.310       0.160
ma.L1          0.8311      1.221      0.681      0.496      -1.562       3.224
ma.L2          0.0552      1.133      0.049      0.961      -2.165       2.275
ar.S.L12      -0.7616      0.539     -1.414      0.158      -1.818       0.294
ar.S.L24      -1.4056      0.544     -2.586      0.010      -2.471      -0.340
ar.S.L36      -1.3251      0.468     -2.833      0.005      -2.242      -0.408
ar.S.L48      -1.1620      0.417     -2.789      0.005      -1.979      -0.346
ar.S.L60      -1.1965      0.413     -2.896      0.004      -2.006      -0.387
ar.S.L72      -0.9762      0.434     -2.249      0.024      -1.827      -0.126
ar.S.L84      -0.9015      0.367     -2.455      0.014      -1.621      -0.182
ar.S.L96      -0.6016      0.287     -2.095      0.036      -1.165      -0.039
ar.S.L108     -0.4342      0.163     -2.665      0.008      -0.753      -0.115
ar.S.L120     -0.2098      0.181     -1.161      0.246      -0.564       0.144
ma.S.L12      -0.1269      0.541     -0.234      0.815      -1.188       0.934
ma.S.L24       0.9022      0.769      1.173      0.241      -0.605       2.409
sigma2         5.8521      1.931      3.030      0.002       2.067       9.638
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               170.34
Prob(Q):                              0.93   Prob(JB):                         0.00
Heteroskedasticity (H):               1.36   Skew:                            -0.24
Prob(H) (two-sided):                  0.17   Kurtosis:                         7.12
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

In [43]:
```
predictions = model.predict(len(data), len(data)+20)
print(predictions)
```

```
251    163.000866
252    162.069998
253    161.178501
254    160.948567
255    162.615265
256    161.752320
257    162.005448
258    159.476272
259    160.328498
260    160.981768
261    161.251480
262    160.400435
263    159.210630
264    157.369783
265    156.223960
266    156.477306
267    156.128058
268    157.100610
269    155.588414
270    154.910691
271    156.675278
Name: predicted_mean, dtype: float64
```

In [44]:
```
data["Close"].plot(legend=True, label="Training Data", figsize=(15, 10))
predictions.plot(legend=True, label="Predictions")
```

Out[44]: <Axes: >



In [45]:
```
# PREDICTING CLOSING STOCK PRICE FOR AMZN USING LSTM
```

In [46]:
```
df = yf.download('AMZN',
                      start=start_date,
                      end=end_date,
                      progress=False)
df["Date"] = df.index
df = df[["Date", "Open", "High", "Low", "Close", "Adj Close", "Volume"]]
df.reset_index(drop=True, inplace=True)
print(df.tail())
```

```
Price                       Date        Open        High         Low  \
Ticker                                   AMZN        AMZN        AMZN
246    2024-10-18 00:00:00+00:00  187.149994  190.740005  186.279999
247    2024-10-21 00:00:00+00:00  188.050003  189.460007  186.399994
248    2024-10-22 00:00:00+00:00  188.350006  191.520004  186.979996
249    2024-10-23 00:00:00+00:00  188.850006  189.160004  183.690002
250    2024-10-24 00:00:00+00:00  185.250000  187.110001  183.865005

Price        Close   Adj Close    Volume
Ticker        AMZN        AMZN      AMZN
246     188.990005  188.990005  37417700
247     189.070007  189.070007  24639400
248     189.699997  189.699997  29650600
249     184.710007  184.710007  31937100
250     186.380005  186.380005  21539596
```

In [47]:
```
plt.figure(figsize=(16,6))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```

In [48]:
```
df = df["Close"]
print(df.head())
```

```
Ticker        AMZN
0       119.570000
1       127.739998
2       132.710007
3       133.089996
4       137.000000
```

In [49]:
```
dataset = df.values
training_data_len = int(np.ceil( len(dataset) * .95 ))
```

In [50]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)

scaled_data
```

Out[50]:
```
array([[0.        ],
       [0.10157899],
       [0.16337196],
       [0.16809644],
       [0.21671019],
       [0.23001377],
       [0.23660334],
       [0.25077714],
       [0.28770368],
       [0.27987072],
       [0.26146968],
       [0.29827176],
       [0.28621157],
       [0.32612213],
       [0.29379581],
       [0.2891956 ],
       [0.31841344],
       [0.3302251 ],
       [0.30249899],
       [0.33743637],
       [0.33780935],
       [0.35011806],
       [0.34141488],
       [0.33258744],
       [0.32972767],
       [0.34141488],
       [0.31418621],
       [0.33954998],
       [0.31020769],
       [0.33954998],
       [0.34626381],
       [0.32724107],
       [0.34700977],
       [0.36391889],
       [0.34626381],
       [0.37796844],
       [0.42894452],
       [0.42546305],
       [0.40469968],
       [0.42608475],
       [0.42086284],
       [0.42073858],
       [0.41986817],
       [0.4203656 ],
       [0.4024618 ],
       [0.377471  ],
       [0.35931868],
       [0.31082939],
       [0.31915959],
       [0.36715164],
       [0.3953748 ],
       [0.42471709],
       [0.44274516],
       [0.43578261],
       [0.41763029],
       [0.39960222],
       [0.42185752],
       [0.44473451],
       [0.43777196],
       [0.45318916],
       [0.46375725],
       [0.4746985 ],
       [0.49173188],
       [0.51833886],
```

```
       [0.49023996],
       [0.44299387],
       [0.49372124],
       [0.64950886],
       [0.6308591 ],
       [0.61643658],
       [0.63359442],
       [0.6250155 ],
       [0.68233243],
       [0.65609843],
       [0.61009573],
       [0.63918931],
       [0.62451825],
       [0.62091253],
       [0.59070001],
       [0.60947403],
       [0.6839488 ],
       [0.68904645],
       [0.6858137 ],
       [0.67101819],
       [0.66629372],
       [0.71105302],
       [0.72920554],
       [0.72124831],
       [0.67822946],
       [0.67064521],
       [0.71179917],
       [0.69352239],
       [0.65137395],
       [0.69401964],
       [0.70856643],
       [0.7357951 ],
       [0.68195945],
       [0.68270541],
       [0.70036049],
       [0.72833512],
       [0.72833512],
       [0.73728702],
       [0.74773103],
       [0.73020021],
       [0.74922295],
       [0.75606123],
       [0.76339676],
       [0.75991549],
       [0.78130056],
       [0.75133657],
       [0.81437284],
       [0.81586476],
       [0.82183263],
       [0.8253139 ],
       [0.86385681],
       [0.82755197],
       [0.79634459],
       [0.79261479],
       [0.76725101],
       [0.74163871],
       [0.6845705 ],
       [0.71689663],
       [0.74561722],
       [0.70893941],
       [0.67263457],
       [0.7466119 ],
       [0.7632725 ],
       [0.68917071],
```

```
[0.7389034 ],
[0.81002115],
[0.82854665],
[0.85950512],
[0.86025108],
[0.85080194],
[0.8694517 ],
[0.84433664],
[0.8330226 ],
[0.83923918],
[0.82581133],
[0.79646904],
[0.80977244],
[0.79534991],
[0.79050098],
[0.79025245],
[0.76439144],
[0.76066145],
[0.77806781],
[0.77645163],
[0.7428821 ],
[0.70707451],
[0.73069746],
[0.74313063],
[0.76725101],
[0.81350243],
[0.80479924],
[0.83911473],
[0.84122834],
[0.83700111],
[0.79895564],
[0.79684202],
[0.80181521],
[0.78627375],
[0.82717899],
[0.86422979],
[0.82058942],
[0.83016283],
[0.92055204],
[0.97326876],
[0.91607609],
[0.96518708],
[1.        ],
[0.97003601],
[1.        ],
[0.99117236],
[0.99179406],
[0.99738895],
[0.93845584],
[0.93149329],
[0.90948653],
[0.91321651],
[0.84993153],
[0.79796096],
[0.79025245],
[0.78304119],
[0.83103324],
[0.76165612],
[0.74947167],
[0.7824195 ],
[0.79112268],
[0.77259737],
[0.83812005],
[0.80193967],
```

```
       [0.60089511],
       [0.51535502],
       [0.52666906],
       [0.53711307],
       [0.57478557],
       [0.58895938],
       [0.58721874],
       [0.62986443],
       [0.62824824],
       [0.72137258],
       [0.71478301],
       [0.72920554],
       [0.73741148],
       [0.75270422],
       [0.70322025],
       [0.7145343 ],
       [0.69538729],
       [0.66579629],
       [0.63695143],
       [0.65336311],
       [0.73268681],
       [0.70471217],
       [0.66840733],
       [0.72510257],
       [0.64428695],
       [0.6941439 ],
       [0.74574168],
       [0.80753456],
       [0.83836877],
       [0.83202792],
       [0.81213477],
       [0.83687685],
       [0.83128177],
       [0.87405191],
       [0.89556143],
       [0.92390905],
       [0.92490373],
       [0.90712419],
       [0.89009081],
       [0.85042896],
       [0.83003857],
       [0.8151188 ],
       [0.8105184 ],
       [0.77570567],
       [0.83227645],
       [0.76128314],
       [0.78515481],
       [0.81561604],
       [0.83401708],
       [0.86099723],
       [0.8450826 ],
       [0.84694769],
       [0.83700111],
       [0.84495833],
       [0.86311085],
       [0.86410552],
       [0.87193829],
       [0.80989689],
       [0.83066026]])
```

In [51]:
```
train_data = scaled_data[0:int(training_data_len), :]
# Split the data into x_train and y_train data sets
x_train = []
y_train = []

for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
    if i<= 61:
        print(x_train)
        print(y_train)
        print()

# Convert the x_train and y_train to numpy arrays
x_train, y_train = np.array(x_train), np.array(y_train)

# Reshape the data
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
# x_train.shape
```

```
[array([0.        , 0.10157899, 0.16337196, 0.16809644, 0.21671019,
       0.23001377, 0.23660334, 0.25077714, 0.28770368, 0.27987072,
       0.26146968, 0.29827176, 0.28621157, 0.32612213, 0.29379581,
       0.2891956 , 0.31841344, 0.3302251 , 0.30249899, 0.33743637,
       0.33780935, 0.35011806, 0.34141488, 0.33258744, 0.32972767,
       0.34141488, 0.31418621, 0.33954998, 0.31020769, 0.33954998,
       0.34626381, 0.32724107, 0.34700977, 0.36391889, 0.34626381,
       0.37796844, 0.42894452, 0.42546305, 0.40469968, 0.42608475,
       0.42086284, 0.42073858, 0.41986817, 0.4203656 , 0.4024618 ,
       0.377471  , 0.35931868, 0.31082939, 0.31915959, 0.36715164,
       0.3953748 , 0.42471709, 0.44274516, 0.43578261, 0.41763029,
       0.39960222, 0.42185752, 0.44473451, 0.43777196, 0.45318916])]
[0.4637572458141963]

[array([0.        , 0.10157899, 0.16337196, 0.16809644, 0.21671019,
       0.23001377, 0.23660334, 0.25077714, 0.28770368, 0.27987072,
       0.26146968, 0.29827176, 0.28621157, 0.32612213, 0.29379581,
       0.2891956 , 0.31841344, 0.3302251 , 0.30249899, 0.33743637,
       0.33780935, 0.35011806, 0.34141488, 0.33258744, 0.32972767,
       0.34141488, 0.31418621, 0.33954998, 0.31020769, 0.33954998,
       0.34626381, 0.32724107, 0.34700977, 0.36391889, 0.34626381,
       0.37796844, 0.42894452, 0.42546305, 0.40469968, 0.42608475,
       0.42086284, 0.42073858, 0.41986817, 0.4203656 , 0.4024618 ,
       0.377471  , 0.35931868, 0.31082939, 0.31915959, 0.36715164,
       0.3953748 , 0.42471709, 0.44274516, 0.43578261, 0.41763029,
       0.39960222, 0.42185752, 0.44473451, 0.43777196, 0.45318916]), array([0.10157899, 0.16337196,
0.16809644, 0.21671019, 0.23001377,
       0.23660334, 0.25077714, 0.28770368, 0.27987072, 0.26146968,
       0.29827176, 0.28621157, 0.32612213, 0.29379581, 0.2891956 ,
       0.31841344, 0.3302251 , 0.30249899, 0.33743637, 0.33780935,
       0.35011806, 0.34141488, 0.33258744, 0.32972767, 0.34141488,
       0.31418621, 0.33954998, 0.31020769, 0.33954998, 0.34626381,
       0.32724107, 0.34700977, 0.36391889, 0.34626381, 0.37796844,
       0.42894452, 0.42546305, 0.40469968, 0.42608475, 0.42086284,
       0.42073858, 0.41986817, 0.4203656 , 0.4024618 , 0.377471  ,
       0.35931868, 0.31082939, 0.31915959, 0.36715164, 0.3953748 ,
       0.42471709, 0.44274516, 0.43578261, 0.41763029, 0.39960222,
       0.42185752, 0.44473451, 0.43777196, 0.45318916, 0.46375725])]
[0.4637572458141963, 0.4746984975793771]
```

In [52]:
```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\shubh\anaconda3\lib\site-packages (2.17.0)
Requirement already satisfied: tensorflow-intel==2.17.0 in c:\users\shubh\anaconda3\lib\site-packag
es (from tensorflow) (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\shubh\anaconda3\lib\site-packages (from t
ensorflow-intel==2.17.0->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\shubh\anaconda3\lib\site-packages (fro
m tensorflow-intel==2.17.0->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in c:\users\shubh\anaconda3\lib\site-packages
(from tensorflow-intel==2.17.0->tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\shubh\anaconda3\lib
\site-packages (from tensorflow-intel==2.17.0->tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\shubh\anaconda3\lib\site-packages (f
rom tensorflow-intel==2.17.0->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in c:\users\shubh\anaconda3\lib\site-packages (from ten
sorflow-intel==2.17.0->tensorflow) (3.12.1)
Requirement already satisfied: libclang>=13.0.0 in c:\users\shubh\anaconda3\lib\site-packages (from
tensorflow-intel==2.17.0->tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in c:\users\shubh\anaconda3\lib\site-package
s (from tensorflow-intel==2.17.0->tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\shubh\anaconda3\lib\site-packages (fro
m tensorflow-intel==2.17.0->tensorflow) (3.4.0)
Requirement already satisfied: packaging in c:\users\shubh\anaconda3\lib\site-packages (from tensor
flow-intel==2.17.0->tensorflow) (23.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0
dev,>=3.20.3 in c:\users\shubh\anaconda3\lib\site-packages (from tensorflow-intel==2.17.0->tensorfl
ow) (4.25.5)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\shubh\anaconda3\lib\site-packages (f
rom tensorflow-intel==2.17.0->tensorflow) (2.32.3)
Requirement already satisfied: setuptools in c:\users\shubh\anaconda3\lib\site-packages (from tenso
rflow-intel==2.17.0->tensorflow) (67.8.0)
Requirement already satisfied: six>=1.12.0 in c:\users\shubh\anaconda3\lib\site-packages (from tens
orflow-intel==2.17.0->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\shubh\anaconda3\lib\site-packages (from
tensorflow-intel==2.17.0->tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\shubh\anaconda3\lib\site-packag
es (from tensorflow-intel==2.17.0->tensorflow) (4.6.3)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\shubh\anaconda3\lib\site-packages (from te
nsorflow-intel==2.17.0->tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\shubh\anaconda3\lib\site-packages (f
rom tensorflow-intel==2.17.0->tensorflow) (1.67.0)
Requirement already satisfied: tensorboard<2.18,>=2.17 in c:\users\shubh\anaconda3\lib\site-package
s (from tensorflow-intel==2.17.0->tensorflow) (2.17.1)
Requirement already satisfied: keras>=3.2.0 in c:\users\shubh\anaconda3\lib\site-packages (from ten
sorflow-intel==2.17.0->tensorflow) (3.6.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\shubh\anaconda3\lib
\site-packages (from tensorflow-intel==2.17.0->tensorflow) (0.31.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\shubh\anaconda3\lib\site-packages
(from tensorflow-intel==2.17.0->tensorflow) (1.24.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\shubh\anaconda3\lib\site-packages (fr
om astunparse>=1.6.0->tensorflow-intel==2.17.0->tensorflow) (0.38.4)
Requirement already satisfied: rich in c:\users\shubh\anaconda3\lib\site-packages (from keras>=3.2.
0->tensorflow-intel==2.17.0->tensorflow) (13.9.3)
Requirement already satisfied: namex in c:\users\shubh\anaconda3\lib\site-packages (from keras>=3.
2.0->tensorflow-intel==2.17.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in c:\users\shubh\anaconda3\lib\site-packages (from keras>=3.
2.0->tensorflow-intel==2.17.0->tensorflow) (0.13.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\shubh\anaconda3\lib\site-packag
es (from requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\shubh\anaconda3\lib\site-packages (from req
uests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\shubh\anaconda3\lib\site-packages (fr
om requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\shubh\anaconda3\lib\site-packages (fr
om requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2023.5.7)
```

```
Requirement already satisfied: markdown>=2.6.8 in c:\users\shubh\anaconda3\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\shubh\anaconda3\li
b\site-packages (from tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\shubh\anaconda3\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (2.2.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\shubh\anaconda3\lib\site-packages (fro
m werkzeug>=1.0.1->tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (2.1.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\shubh\anaconda3\lib\site-packages
(from rich->keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\shubh\anaconda3\lib\site-package
s (from rich->keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\shubh\anaconda3\lib\site-packages (from markd
own-it-py>=2.2.0->rich->keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (0.1.0)
```

In [53]:
```
from keras.models import Sequential
from keras.layers import Dense, LSTM

# Build the LSTM model
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(x_train, y_train, batch_size=1, epochs=1)
```

```
C:\Users\shubh\anaconda3\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning: Do not pas
s an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `
Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
179/179 ━━━━━━━━━━━━━━━━━━━━ 10s 24ms/step - loss: 0.0235
```

Out[53]:
```
<keras.src.callbacks.history.History at 0x22db1378250>
```

In [54]:
```
test_data = scaled_data[training_data_len - 60: , :]
# Create the data sets x_test and y_test
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])

# Convert the data to a numpy array
x_test = np.array(x_test)

# Reshape the data
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1 ))

# Get the models predicted price values
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)

# Get the root mean squared error (RMSE)
rmse = np.sqrt(np.mean(((predictions - y_test) ** 2)))
rmse
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 539ms/step
```

Out[54]:
```
8.026100003766631
```

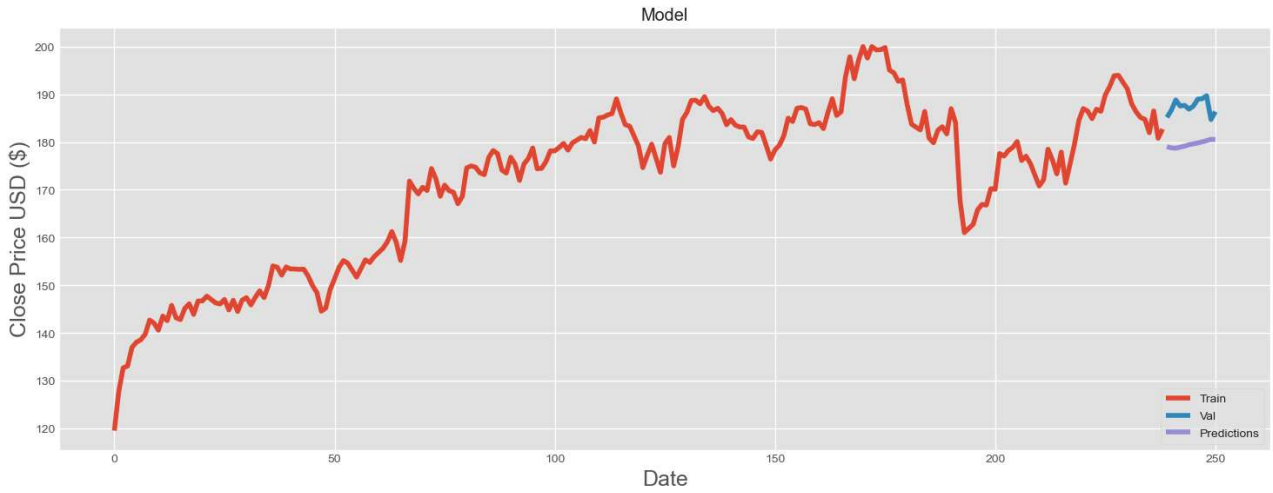
In [55]:
```
train = df.iloc[:training_data_len]
valid = df.iloc[training_data_len:]
valid['Predictions'] = predictions
valid
```

```
C:\Users\shubh\AppData\Local\Temp\ipykernel_8108\84718757.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  valid['Predictions'] = predictions
```

Out[55]:

Ticker	AMZN	Predictions
239	185.169998	179.051758
240	186.649994	178.768890
241	188.820007	178.715683
242	187.539993	178.916199
243	187.690002	179.160797
244	186.889999	179.420303
245	187.529999	179.617630
246	188.990005	179.801041
247	189.070007	180.041382
248	189.699997	180.292160
249	184.710007	180.562469
250	186.380005	180.532303

In [56]:
```
# Visualize the data
plt.figure(figsize=(16,6))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(train['AMZN'])
plt.plot(valid[['AMZN', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
plt.show()
```



In [58]:
```
'''
Thank You
'''
```

Out[58]:
```
'\nThank You\n'
```

In [ ]: