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)

STOCK MARKET PREDICTION AND ANALYSIS

1. Stocks form Apple, Amazon, Google, and Microsoft are explored (closing prices, daily return, moving average).
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.

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

```
Requirement already satisfied: yfinance in c:\users\shubh\anaconda3\lib\site-packages (0.2.50)
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) (2024.8.30)
```

In [2]:
```
!pip install --upgrade yfinance
```

```
Requirement already satisfied: yfinance in c:\users\shubh\anaconda3\lib\site-packages (0.2.50)
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) (2024.8.30)
```

In [3]:
```
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 = 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)


AAPL = stock_data['AAPL']
GOOG = stock_data['GOOG']
MSFT = stock_data['MSFT']
AMZN = stock_data['AMZN']


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)

print(df.tail(10))
```

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

```
Price      Adj Close Close High  Low Open Volume company_name Adj Close Close  \
Ticker          AAPL  AAPL AAPL AAPL AAPL   AAPL                   GOOG  GOOG   
Date                                                                            
2024-12-02       NaN   NaN  NaN  NaN  NaN    NaN       AMAZON       NaN   NaN   
2024-12-03       NaN   NaN  NaN  NaN  NaN    NaN       AMAZON       NaN   NaN   
2024-12-04       NaN   NaN  NaN  NaN  NaN    NaN       AMAZON       NaN   NaN   
2024-12-05       NaN   NaN  NaN  NaN  NaN    NaN       AMAZON       NaN   NaN   
2024-12-06       NaN   NaN  NaN  NaN  NaN    NaN       AMAZON       NaN   NaN   
2024-12-09       NaN   NaN  NaN  NaN  NaN    NaN       AMAZON       NaN   NaN   
2024-12-10       NaN   NaN  NaN  NaN  NaN    NaN       AMAZON       NaN   NaN   
2024-12-11       NaN   NaN  NaN  NaN  NaN    NaN       AMAZON       NaN   NaN   
2024-12-12       NaN   NaN  NaN  NaN  NaN    NaN       AMAZON       NaN   NaN   
2024-12-13       NaN   NaN  NaN  NaN  NaN    NaN       AMAZON       NaN   NaN   

Price      High  ...       Low Open Volume   Adj Close       Close  \
Ticker     GOOG  ... MSFT MSFT MSFT   MSFT        AMZN        AMZN   
Date             ...                                                 
2024-12-02  NaN  ...  NaN  NaN  NaN    NaN  210.710007  210.710007   
2024-12-03  NaN  ...  NaN  NaN  NaN    NaN  213.440002  213.440002   
2024-12-04  NaN  ...  NaN  NaN  NaN    NaN  218.160004  218.160004   
2024-12-05  NaN  ...  NaN  NaN  NaN    NaN  220.550003  220.550003   
2024-12-06  NaN  ...  NaN  NaN  NaN    NaN  227.029999  227.029999   
2024-12-09  NaN  ...  NaN  NaN  NaN    NaN  226.089996  226.089996   
2024-12-10  NaN  ...  NaN  NaN  NaN    NaN  225.039993  225.039993   
2024-12-11  NaN  ...  NaN  NaN  NaN    NaN  230.259995  230.259995   
2024-12-12  NaN  ...  NaN  NaN  NaN    NaN  228.970001  228.970001   
2024-12-13  NaN  ...  NaN  NaN  NaN    NaN  227.460007  227.460007   

Price             High         Low        Open      Volume  
Ticker            AMZN        AMZN        AMZN        AMZN  
Date                                                        
2024-12-02  212.990005  209.509995  209.960007  39523200.0  
2024-12-03  214.020004  209.649994  210.309998  32214800.0  
2024-12-04  220.000000  215.750000  215.960007  48745700.0  
2024-12-05  222.149994  217.300003  218.029999  41140200.0  
2024-12-06  227.149994  220.600006  220.750000  44178100.0  
2024-12-09  230.080002  225.669998  227.210007  46819400.0  
2024-12-10  229.059998  224.199997  226.089996  31199900.0  
2024-12-11  231.199997  226.259995  226.410004  35385800.0  
2024-12-12  231.089996  227.630005  229.830002  28204100.0  
2024-12-13  230.199997  225.860794  228.470001  28249154.0  

[10 rows x 25 columns]
```

In [4]:
```
df.head(10)
```

Out[4]:

Price	Adj Close	Close	High	Low	Open	Volume	company_name	Adj Close	Close	
Ticker	AAPL	AAPL	AAPL	AAPL	AAPL	AAPL		GOOG	GOOG	GOO
Date										
2023-12-14	197.144196	198.110001	199.619995	196.160004	198.020004	66831600.0	APPLE	NaN	NaN	Nal
2023-12-15	196.606827	197.570007	198.399994	197.000000	197.529999	128256700.0	APPLE	NaN	NaN	Nal
2023-12-18	194.934998	195.889999	196.630005	194.389999	196.089996	55751900.0	APPLE	NaN	NaN	Nal
2023-12-19	195.979889	196.940002	196.949997	195.889999	196.160004	40714100.0	APPLE	NaN	NaN	Nal
2023-12-20	193.880188	194.830002	197.679993	194.830002	196.899994	52242800.0	APPLE	NaN	NaN	Nal
2023-12-21	193.730881	194.679993	197.080002	193.500000	196.100006	46482500.0	APPLE	NaN	NaN	Nal
2023-12-22	192.656174	193.600006	195.410004	192.970001	195.179993	37122800.0	APPLE	NaN	NaN	Nal
2023-12-26	192.108871	193.050003	193.889999	192.830002	193.610001	28919300.0	APPLE	NaN	NaN	Nal
2023-12-27	192.208359	193.149994	193.500000	191.089996	192.490005	48087700.0	APPLE	NaN	NaN	Nal
2023-12-28	192.636292	193.580002	194.660004	193.169998	194.139999	34049900.0	APPLE	NaN	NaN	Nal

10 rows × 25 columns

In [5]:

```
# checking 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        Open  \
Ticker             AAPL        AAPL        AAPL        AAPL        AAPL   
Date                                                                      
2023-12-14  197.144196  198.110001  199.619995  196.160004  198.020004   
2023-12-15  196.606827  197.570007  198.399994  197.000000  197.529999   
2023-12-18  194.934998  195.889999  196.630005  194.389999  196.089996   
2023-12-19  195.979889  196.940002  196.949997  195.889999  196.160004   
2023-12-20  193.880188  194.830002  197.679993  194.830002  196.899994   

Price          Volume company_name  
Ticker           AAPL               
Date                                
2023-12-14   66831600        APPLE  
2023-12-15  128256700        APPLE  
2023-12-18   55751900        APPLE  
2023-12-19   40714100        APPLE  
2023-12-20   52242800        APPLE  

GOOG data:
 Price        Adj Close       Close        High         Low        Open  \
Ticker             GOOG        GOOG        GOOG        GOOG        GOOG   
Date                                                                      
2023-12-14  132.723114  133.199997  135.035004  131.059998  134.770004   
2023-12-15  133.360809  133.839996  134.830002  132.630005  132.919998   
2023-12-18  136.698822  137.190002  138.380005  133.770004  133.860001   
2023-12-19  137.605576  138.100006  138.770004  137.449997  138.000000   
2023-12-20  139.159988  139.660004  143.078003  139.410004  140.330002   

Price         Volume company_name  
Ticker          GOOG               
Date                               
2023-12-14  29619100       GOOGLE  
2023-12-15  58569400       GOOGLE  
2023-12-18  25699800       GOOGLE  
2023-12-19  20661000       GOOGLE  
2023-12-20  33507300       GOOGLE  

MSFT data:
 Price        Adj Close       Close        High         Low        Open  \
Ticker             MSFT        MSFT        MSFT        MSFT        MSFT   
Date                                                                      
2023-12-14  363.213928  365.929993  373.760010  364.130005  373.309998   
2023-12-15  367.978302  370.730011  372.399994  366.279999  366.850006   
2023-12-18  369.884003  372.649994  373.000000  368.679993  369.450012   
2023-12-19  370.489563  373.260010  373.260010  369.839996  371.489990   
2023-12-20  367.869110  370.619995  376.029999  370.529999  375.000000   

Price         Volume company_name  
Ticker          MSFT               
Date                               
2023-12-14  43277500    MICROSOFT  
2023-12-15  78478200    MICROSOFT  
2023-12-18  21802900    MICROSOFT  
2023-12-19  20603700    MICROSOFT  
2023-12-20  26316700    MICROSOFT  

AMZN data:
 Price        Adj Close       Close        High         Low        Open  \
Ticker             AMZN        AMZN        AMZN        AMZN        AMZN   
Date                                                                      
2023-12-14  147.419998  147.419998  150.539993  145.520004  149.929993   
2023-12-15  149.970001  149.970001  150.570007  147.880005  148.380005   
2023-12-18  154.070007  154.070007  154.850006  150.050003  150.559998   
```

```
2023-12-19  153.789993  153.789993  155.119995  152.690002  154.399994
2023-12-20  152.119995  152.119995  155.630005  151.559998  152.899994

Price          Volume company_name
Ticker           AMZN
Date
2023-12-14   58400800       AMAZON
2023-12-15  110039100       AMAZON
2023-12-18   62512800       AMAZON
2023-12-19   43171300       AMAZON
2023-12-20   50322100       AMAZON
```

In [6]:
```python
# Checking if 'Adj Close' exists
for ticker in tech_list:
    print(f"{ticker} columns:\n", stock_data[ticker].columns, "\n")
```

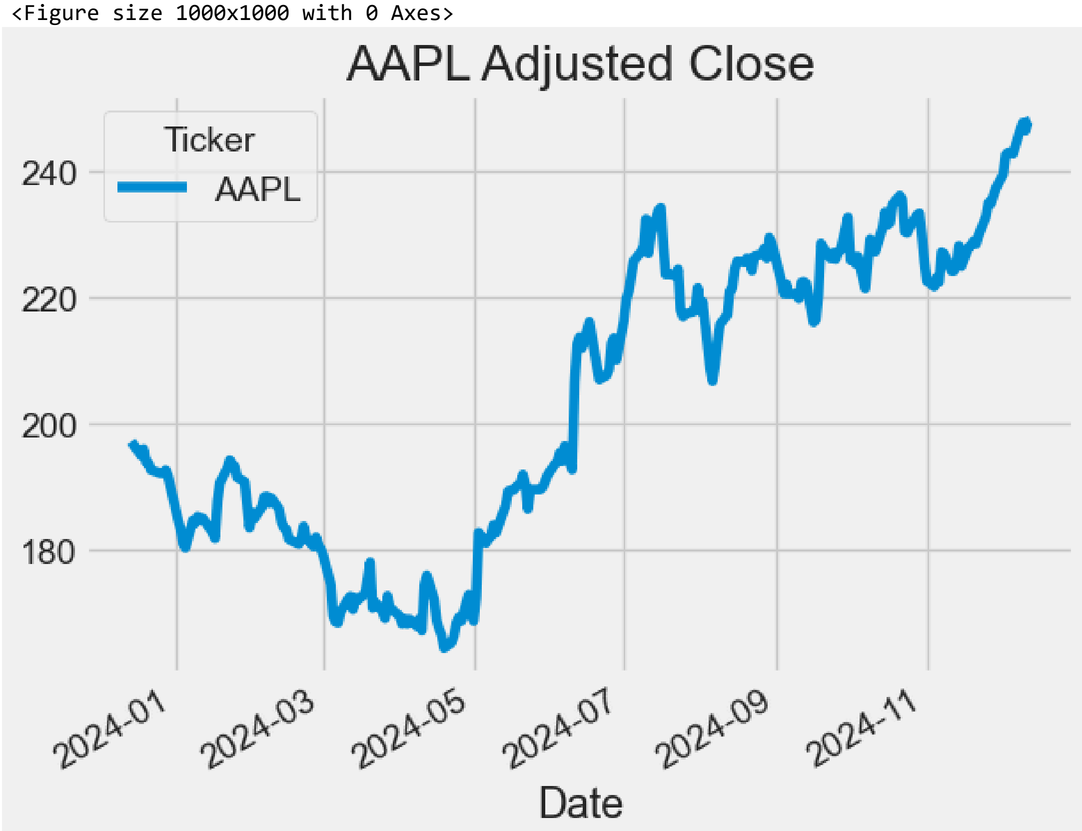
```
AAPL columns:
 MultiIndex([(   'Adj Close', 'AAPL'),
            (       'Close', 'AAPL'),
            (        'High', 'AAPL'),
            (         'Low', 'AAPL'),
            (        'Open', 'AAPL'),
            (      'Volume', 'AAPL'),
            ('company_name',     '')],
           names=['Price',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

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

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

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

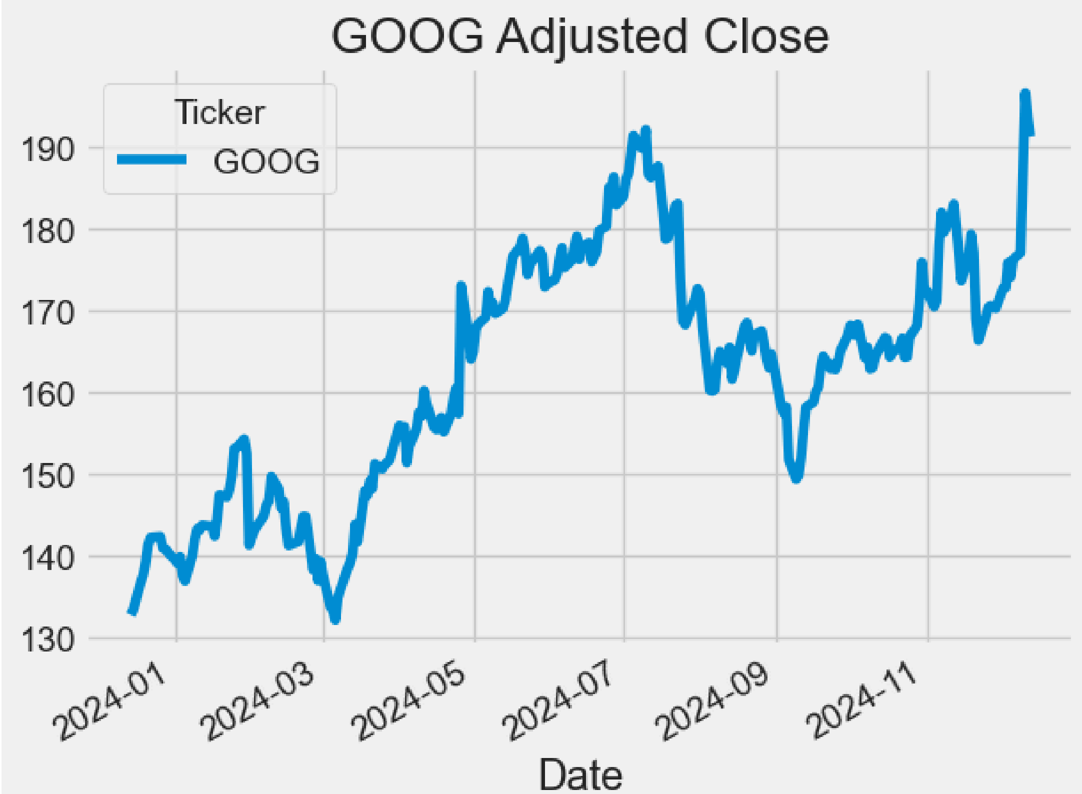
Closing Price :

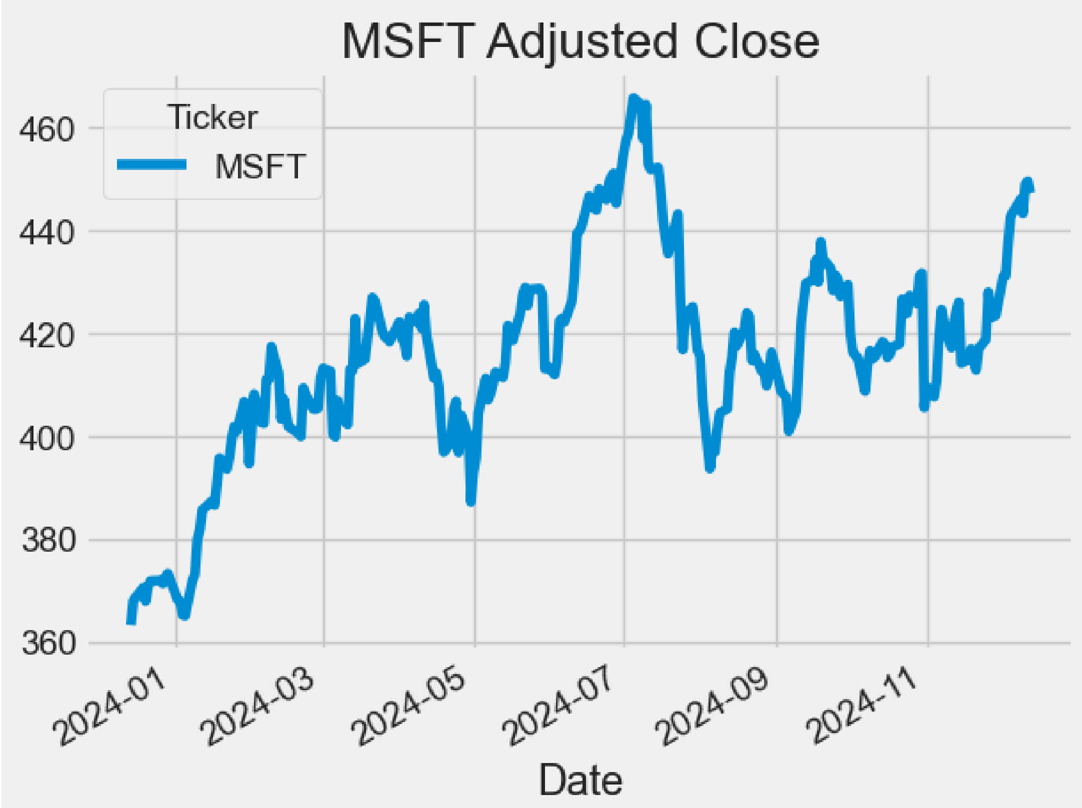
The closing price is also referred to as “close”. Essentially it is the final traded price of a financial asset at the end of a trading day or a trading session.

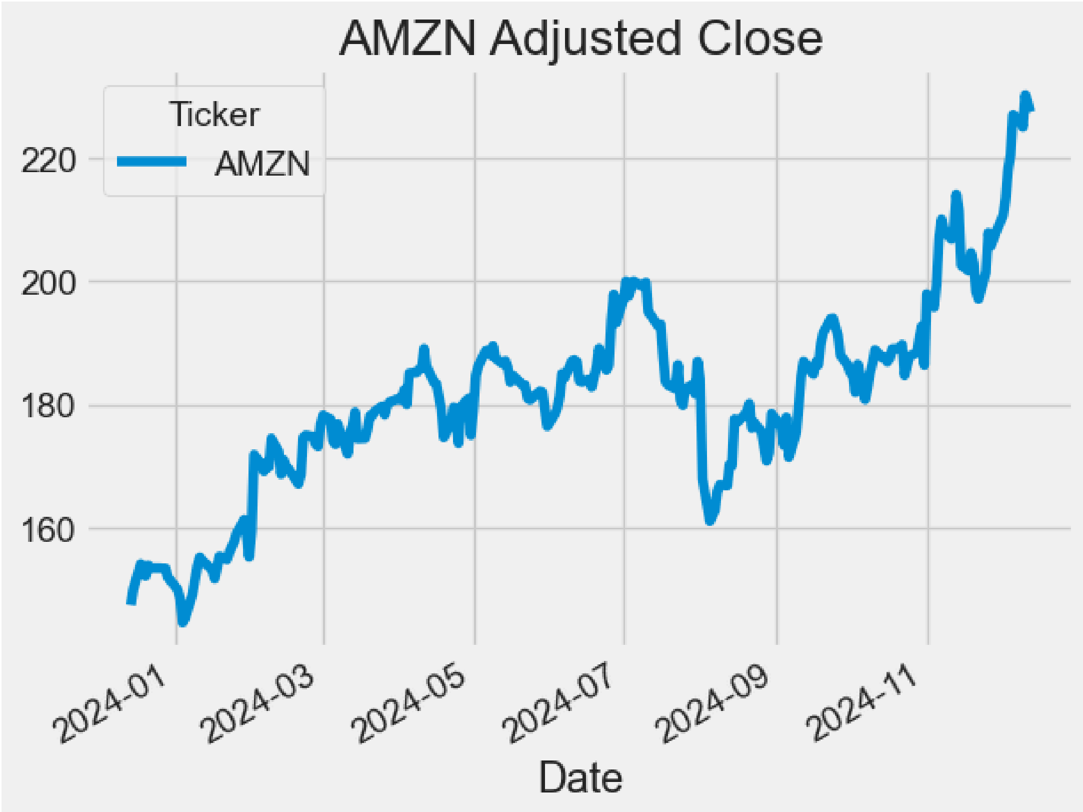
In [7]:
```
#Closing Price
plt.figure(figsize=(10, 10))

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
2024-01
2024-03
2024-05
2024-07
2024-09
2024-11
Date


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

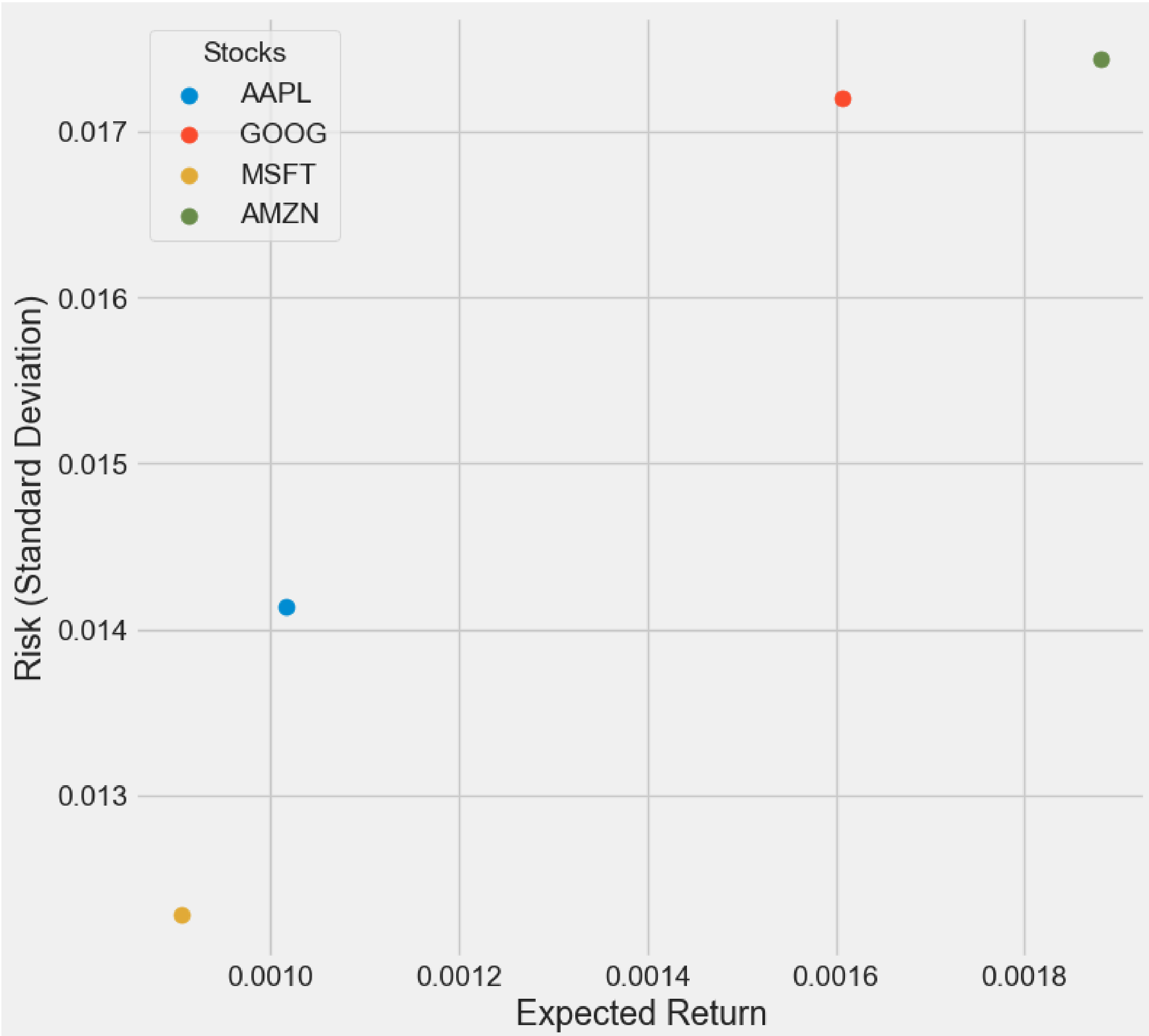


In [8]:
```
closing_df = pd.DataFrame()

for stock in tech_list:
    closing_df[stock] = stock_data[stock]['Adj Close']

tech_rets = closing_df.pct_change()

tech_rets.head()
```

Out[8]:

Date	AAPL	GOOG	MSFT	AMZN
2023-12-14	NaN	NaN	NaN	NaN
2023-12-15	-0.002726	0.004805	0.013117	0.017298
2023-12-18	-0.008503	0.025030	0.005179	0.027339
2023-12-19	0.005360	0.006633	0.001637	-0.001817
2023-12-20	-0.010714	0.011296	-0.007073	-0.010859

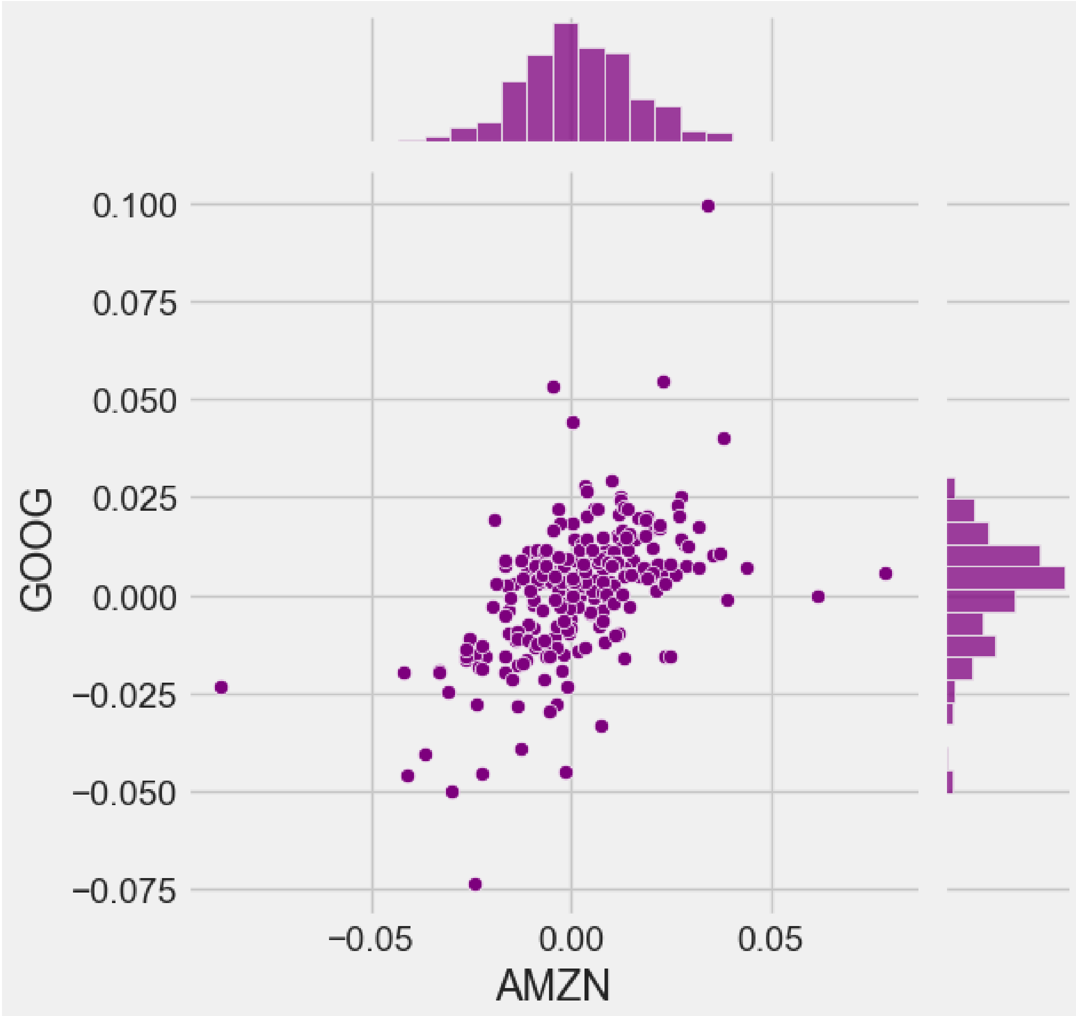
In [9]:
```
rets = tech_rets.dropna()

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()
```

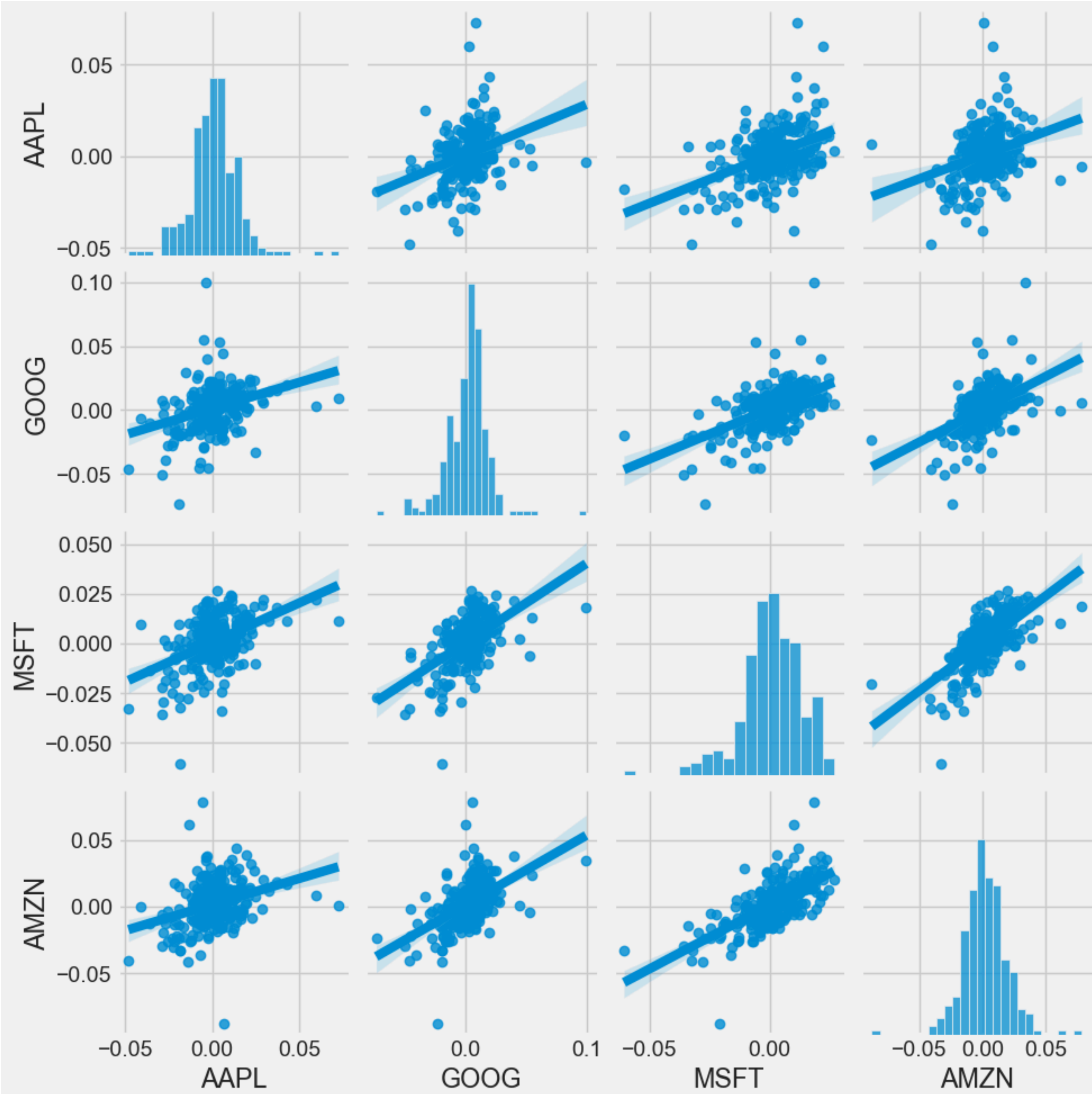


Risk-Return Tradeoff : Higher is expected return, more is the risk for the stocks. MSFT shows low risks and potentially low returns ideal for risk averse investors.

In [10]:
```
# compare the daily percentage return of two stocks to check correlation
sns.jointplot(x='AMZN', y='GOOG', data=tech_rets, kind='scatter', color='purple')

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

```
Out[10]: <seaborn.axisgrid.PairGrid at 0x1d9f84486d0>
```



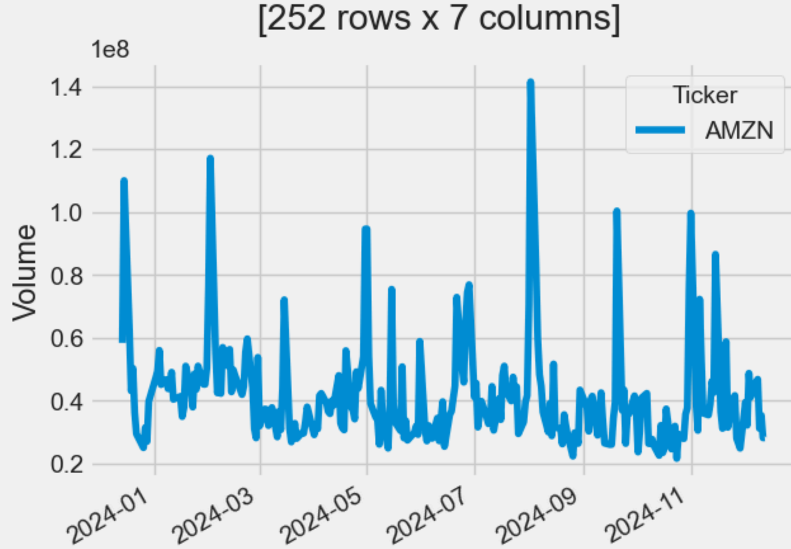


1. Each histogram shows rougly a bell curved shape, while AMZN stocks are normally distributed.
2. A positive correlation is observed amongst most pairs. Slightly weaker correlations may exist for certain pairs, but none show negative or no correlation.
3. The regression lines in the scatter plots indicate linear relationships between the pairs of stocks. This suggests that when one stock's return increases, the others tend to increase as well.
4. Stocks like GOOG and AMZN may exhibit higher dispersion (greater volatility) compared to AAPL and MSFT.

In [11]:
```
#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_42160\3274364906.py:9: UserWarning: Tight layout not ap
plied. 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        Open  \
            Ticker           AAPL        AAPL        AAPL        AAPL        AAPL
                  Date
2023-12-14  197.144196  198.110001  199.619995  196.160004  198.020004
2023-12-15  196.606827  197.570007  198.399994  197.000000  197.529999
2023-12-18  194.934998  195.889999  196.630005  194.389999  196.089996
2023-12-19  195.979889  196.940002  196.949997  195.889999  196.160004
2023-12-20  193.880188  194.830002  197.679993  194.830002  196.899994
                   ...         ...        ...         ...        ...         ...
2024-12-09  246.750000  246.750000  247.240005  241.750000  241.830002
2024-12-10  247.770004  247.770004  248.210007  245.339996  246.889999
2024-12-11  246.490005  246.490005  250.800003  246.259995  247.960007
2024-12-12  247.960007  247.960007  248.740005  245.679993  246.889999
2024-12-13  248.130005  248.130005  249.290207  246.240005  247.880005

Price          Volume company_name
Ticker           AAPL
Date
2023-12-14   66831600        APPLE
2023-12-15  128256700        APPLE
2023-12-18   55751900        APPLE
2023-12-19   40714100        APPLE
2023-12-20   52242800        APPLE
...               ...          ...
2024-12-09   44649200        APPLE
2024-12-10   36914800        APPLE
2024-12-11   45205800        APPLE
2024-12-12   32777500        APPLE
2024-12-13   32003804        APPLE

[252 rows x 7 columns]
```

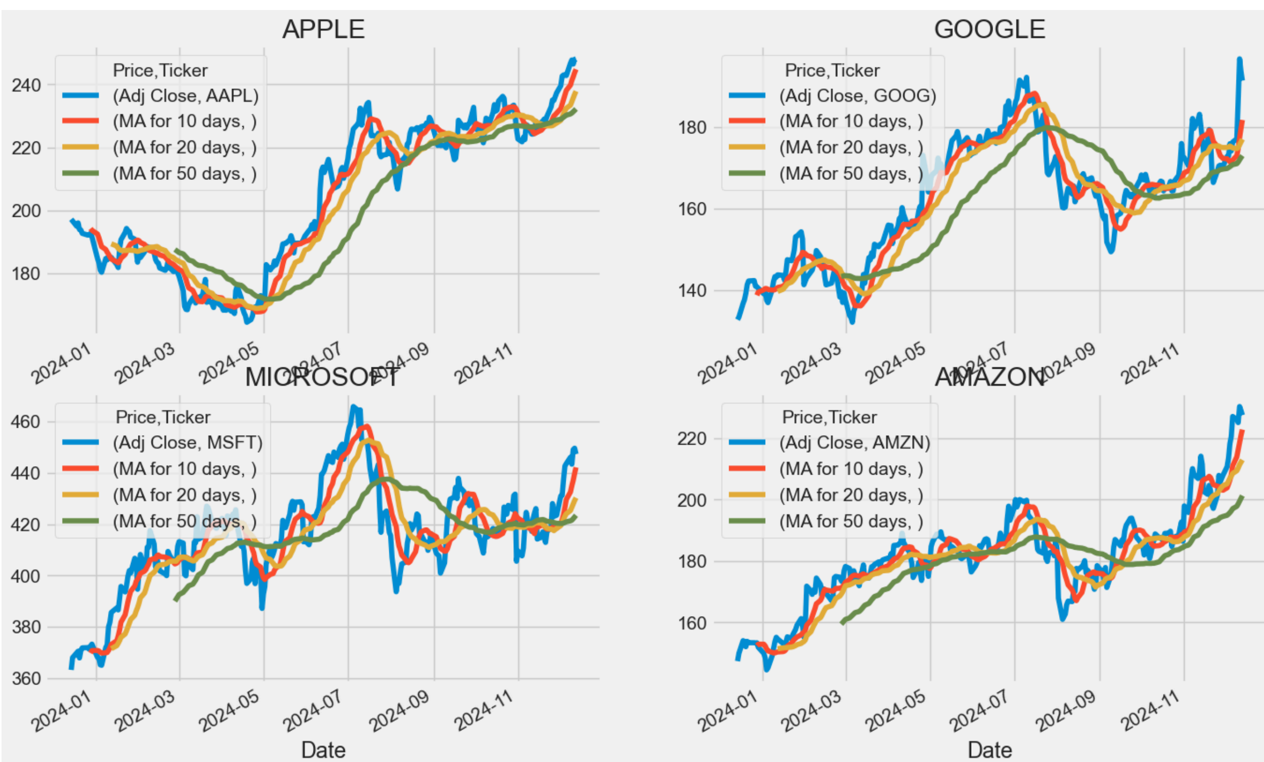


Moving average is calculated to analyze data points by creating a series of averages from different subsets of the full data set. In finance, it is commonly used to smooth out short-term fluctuations in stock prices or other data to reveal long-term trends.

Simple moving averages (SMAs) use a simple arithmetic average of prices over some timespan, while exponential moving averages (EMAs) place greater weight on more recent prices than older ones over the time period.

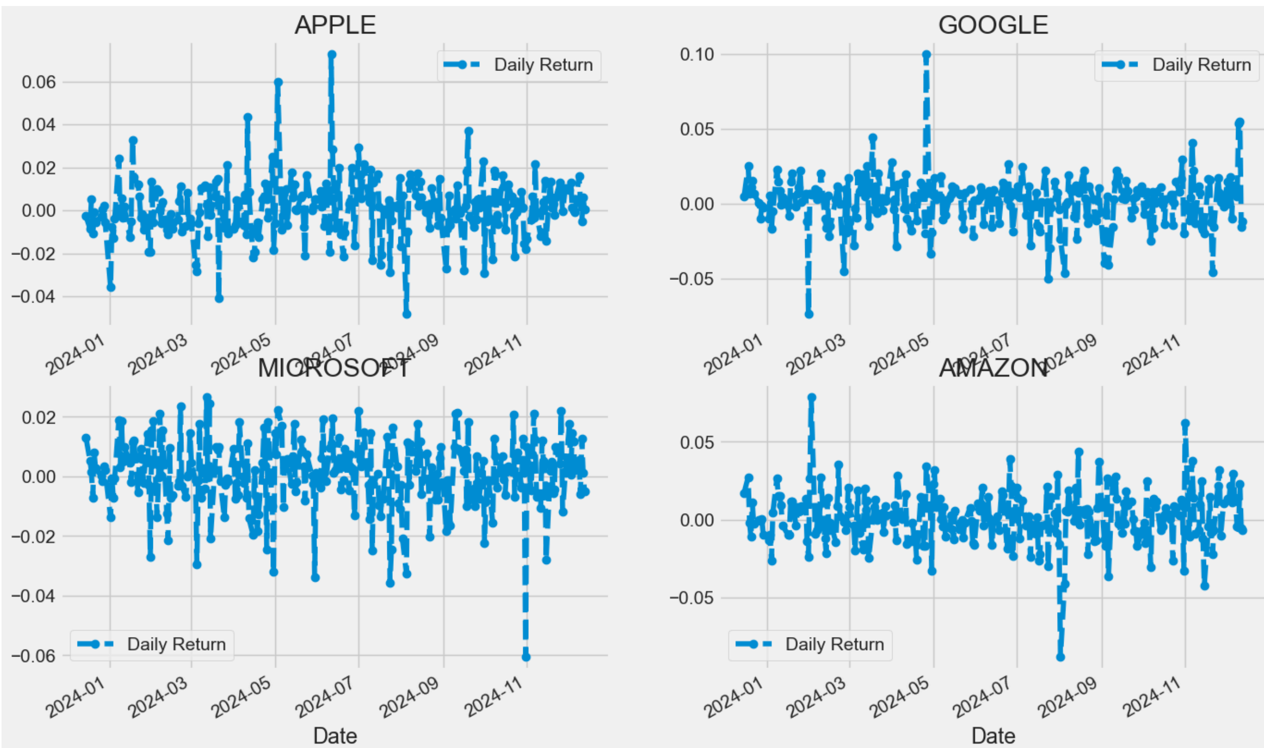
In [12]:
```
#Moving Average
ma_day = [10, 20, 50]

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[12]: `Text(0.5, 1.0, 'AMAZON')`

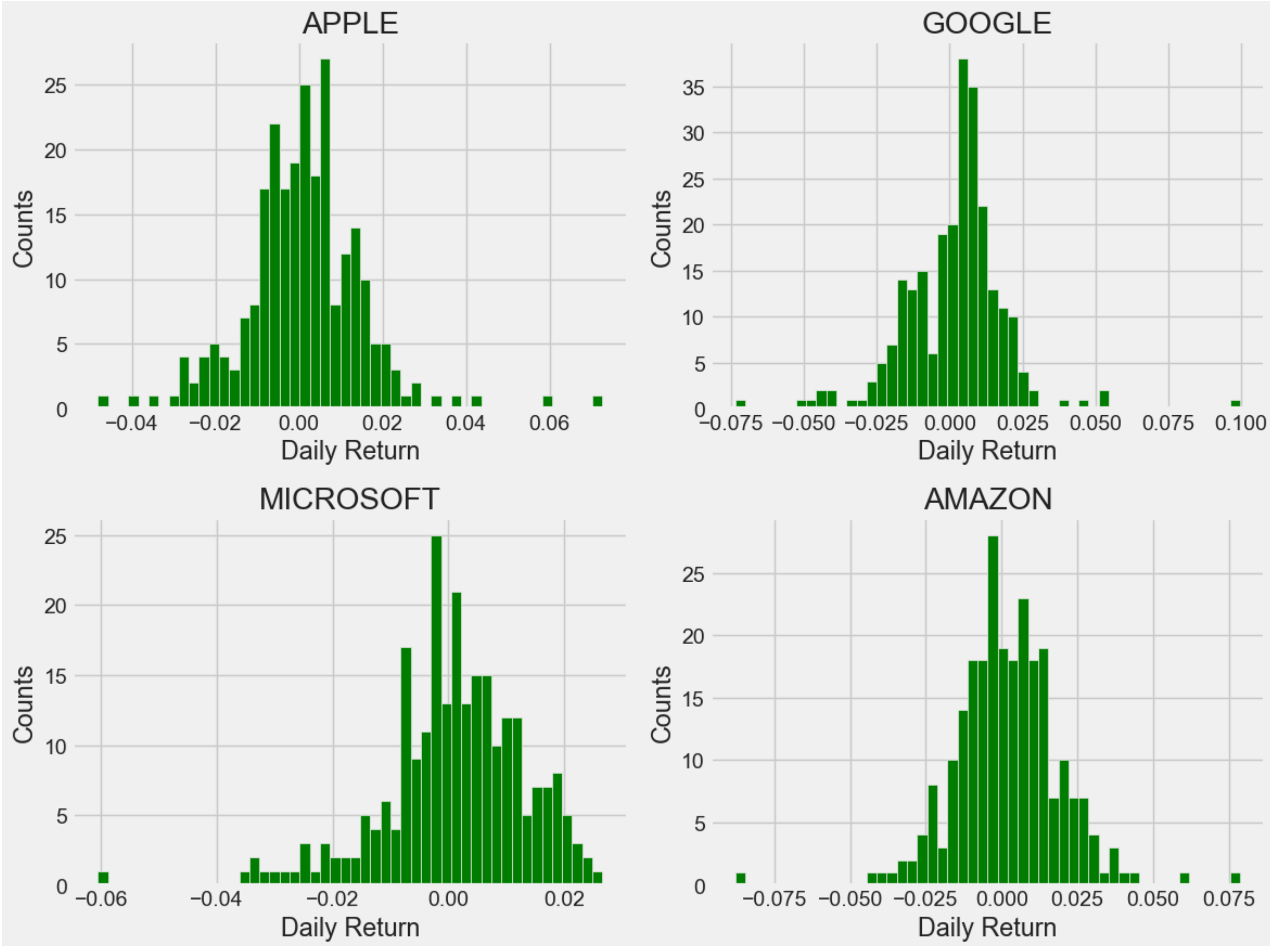


In [13]:
```
#daily return for stocks
for company in company_list:
    company['Daily Return'] = company['Adj Close'].pct_change()

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[13]: `Text(0.5, 1.0, 'AMAZON')`

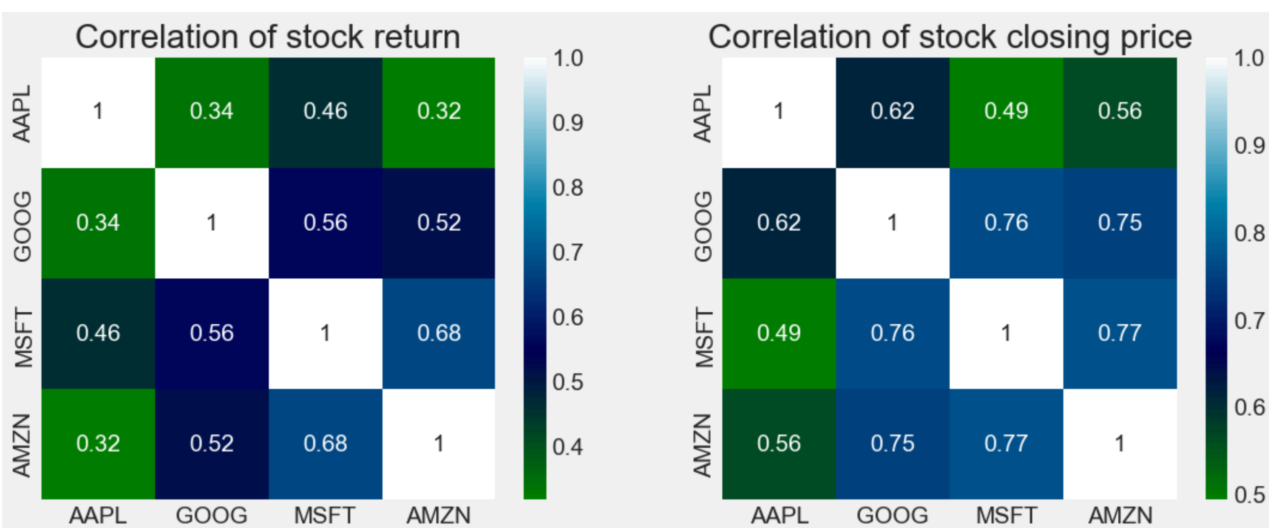


In [14]:
```
plt.figure(figsize=(12, 9))

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 [15]:
```
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[15]:
```
Text(0.5, 1.0, 'Correlation of stock closing price')
```



TIME SERIES FORECASTING USING ARIMA FOR GOOGLE STOCK PRICES

In [16]:
```
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())
```

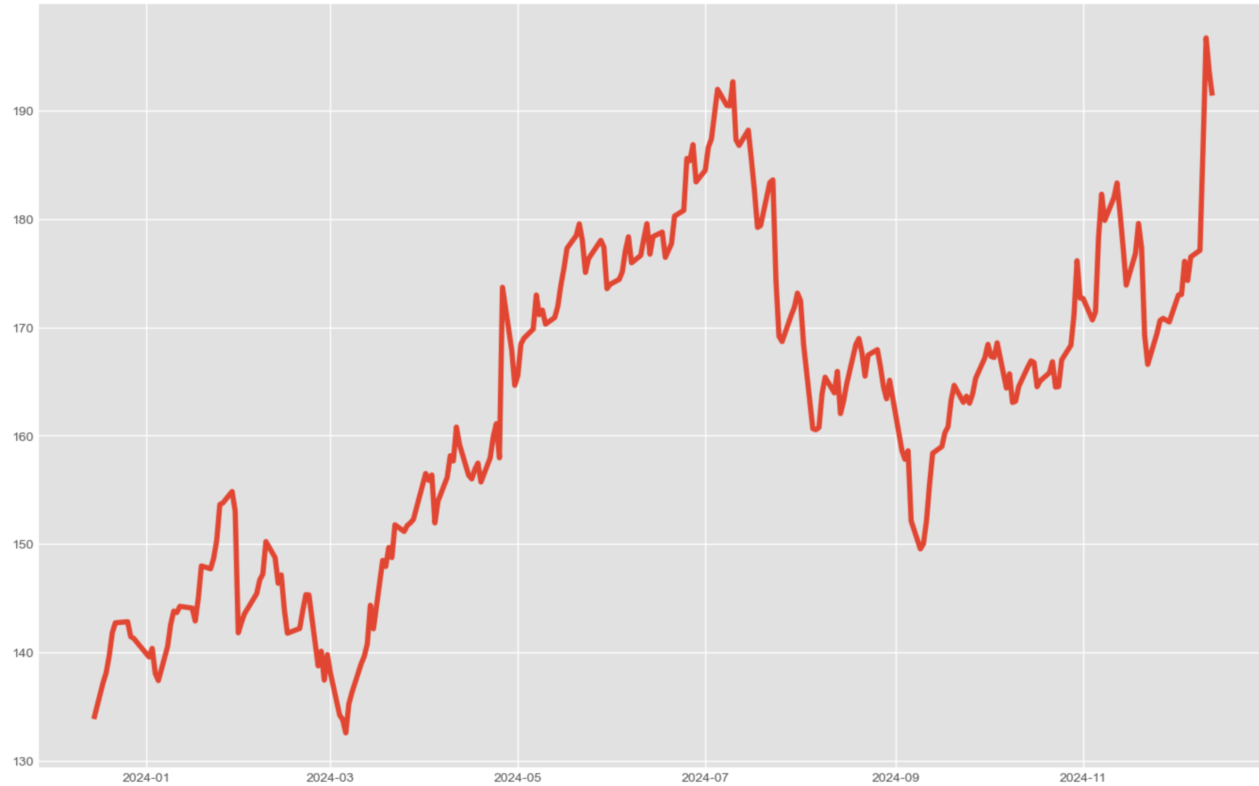
```
Price        Date        Open        High         Low       Close   Adj Close  \
Ticker                   GOOG        GOOG        GOOG        GOOG        GOOG
246    2024-12-09  175.714996  178.039993  175.399994  177.100006  177.100006
247    2024-12-10  184.535004  188.029999  182.669998  186.529999  186.529999
248    2024-12-11  186.699997  196.889999  186.259995  196.710007  196.710007
249    2024-12-12  196.300003  196.705002  193.279999  193.630005  193.630005
250    2024-12-13  192.750000  194.339996  191.259995  191.380005  191.380005

Price     Volume
Ticker      GOOG
246     19887800
247     34317400
248     41664500
249     25197800
250     18360673
```

In [17]:
```
data = data[["Date", "Close"]]
print(data.head())
```

```
Price        Date       Close
Ticker                   GOOG
0      2023-12-15  133.839996
1      2023-12-18  137.190002
2      2023-12-19  138.100006
3      2023-12-20  139.660004
4      2023-12-21  141.800003
```

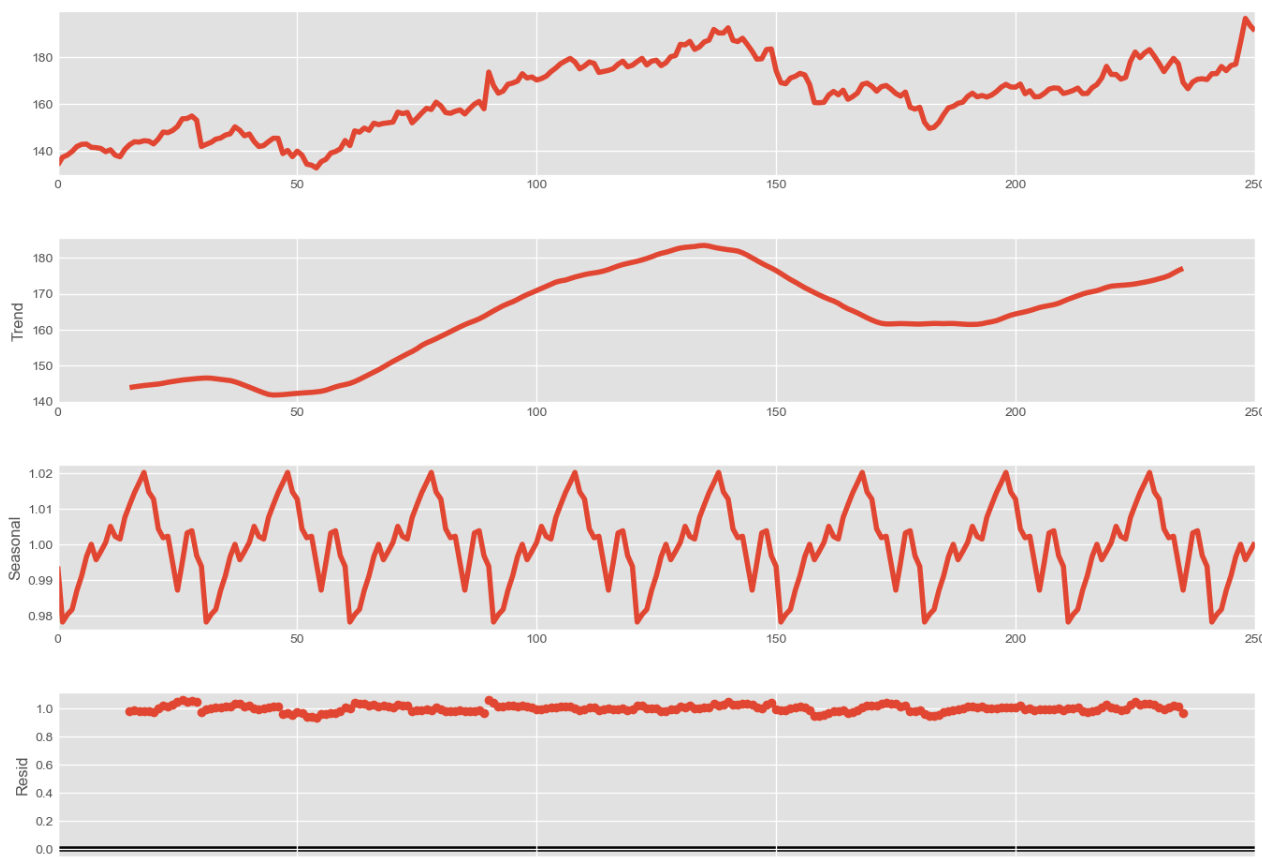
In [18]:
```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
plt.figure(figsize=(15, 10))
plt.plot(data["Date"], data["Close"])
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x1d9f863c0d0>]
```



In [19]:
```
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)
```

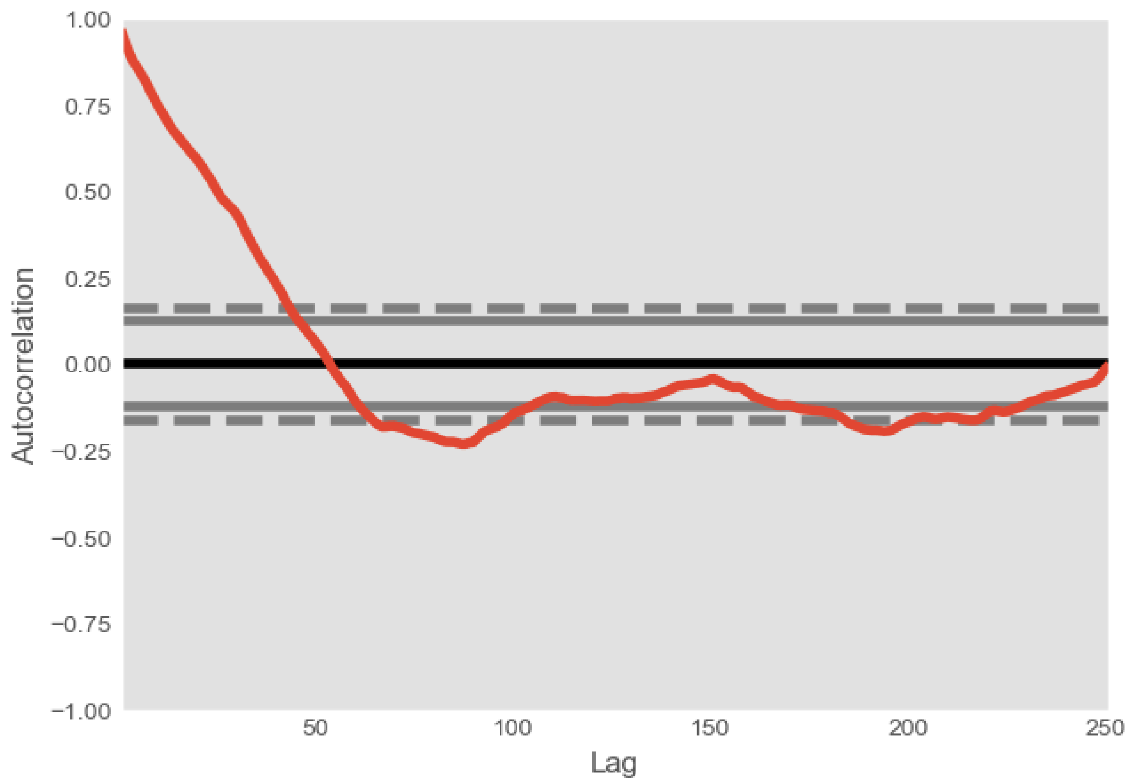
```
<Figure size 640x480 with 0 Axes>
```



1. The overall price has been increasing over time.
2. There are recurring cyclical patterns in the price, likely due to daily, weekly, or monthly factors.
3. There might be additional factors influencing the price that are not captured by the trend or seasonality components.

In [20]:
```
pd.plotting.autocorrelation_plot(data["Close"])
```

Out[20]: <Axes: xlabel='Lag', ylabel='Autocorrelation'>

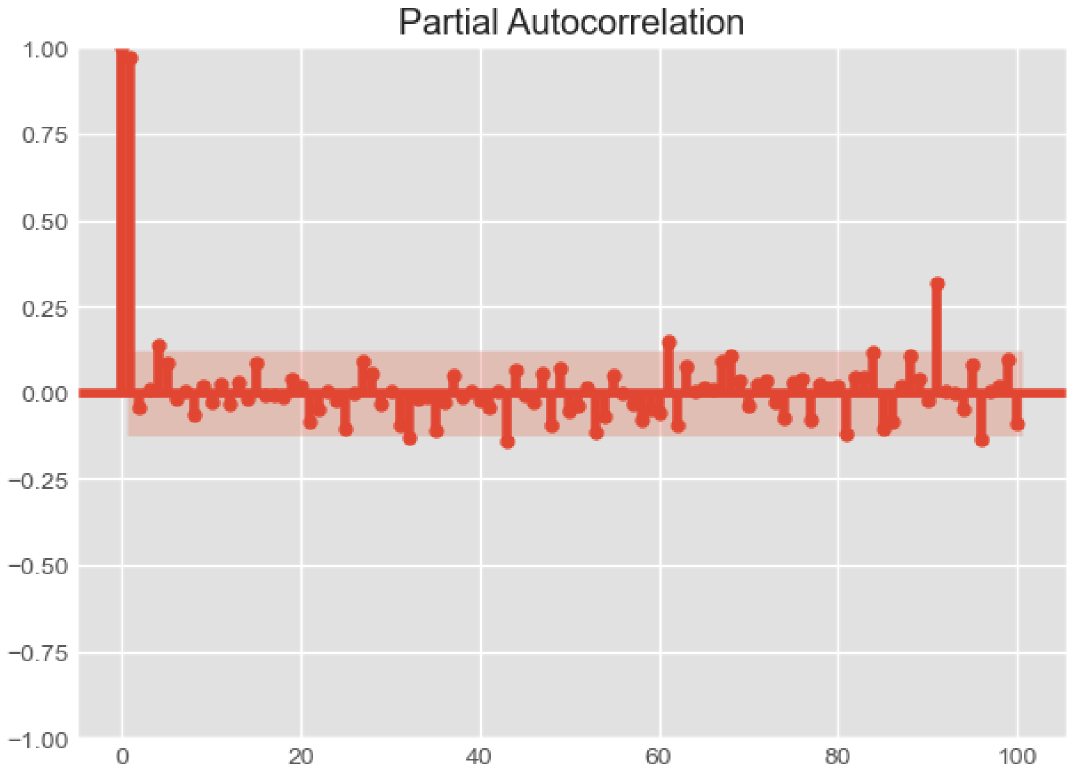


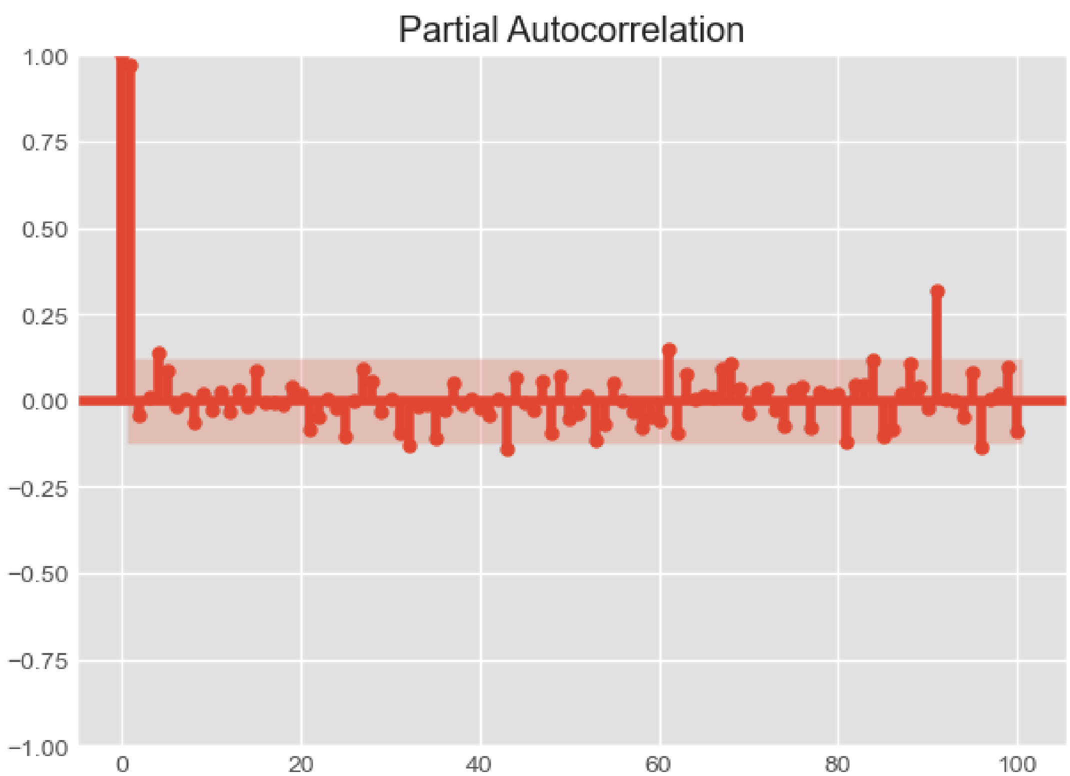
In [21]:
```
#Since the curve is moving down after the 10th line of the first boundary, therefore p = 10
```

In [22]:
```
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[22]:





In [23]:
```
# 2 points are far away from others, therefore q=2 and since data is seasonal , d = 1
```

In [24]:
```
import statsmodels.api as sm
import matplotlib.pyplot as plt
p, d, q = 10, 1, 2

from statsmodels.tsa.arima.model import ARIMA

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

print(fitted.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                   GOOG   No. Observations:                  251
Model:                ARIMA(10, 1, 2)   Log Likelihood                -609.936
Date:                Sat, 14 Dec 2024   AIC                           1245.872
Time:                        04:00:43   BIC                           1291.651
Sample:                             0   HQIC                          1264.296
                                - 251
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.0595      0.339      0.176      0.860      -0.604       0.723
ar.L2          0.5552      0.393      1.413      0.158      -0.215       1.325
ar.L3         -0.0990      0.092     -1.071      0.284      -0.280       0.082
ar.L4          0.0605      0.104      0.582      0.561      -0.143       0.264
ar.L5          0.0825      0.104      0.795      0.427      -0.121       0.286
ar.L6          0.0412      0.107      0.384      0.701      -0.169       0.252
ar.L7          0.0580      0.084      0.694      0.487      -0.106       0.222
ar.L8         -0.0271      0.097     -0.280      0.780      -0.217       0.163
ar.L9         -0.0195      0.100     -0.194      0.846      -0.216       0.177
ar.L10        -0.1114      0.091     -1.226      0.220      -0.289       0.067
ma.L1       3.362e-06      0.344   9.78e-06      1.000      -0.674       0.674
ma.L2         -0.6393      0.399     -1.600      0.110      -1.422       0.144
sigma2         7.6930      0.443     17.362      0.000       6.825       8.561
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               305.02
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               1.36   Skew:                             0.21
Prob(H) (two-sided):                  0.16   Kurtosis:                         8.39
===================================================================================

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

In [25]:
```
predictions = fitted.predict()
print(predictions)
```

```
0        0.000000
1      133.840061
2      137.390243
3      137.907475
4      139.421693
          ...
246    177.005167
247    177.344160
248    187.317731
249    196.648784
250    192.468781
Name: predicted_mean, Length: 251, dtype: float64
```

In [26]:
```
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                -591.695
Date:                              Sat, 14 Dec 2024   AIC                           1233.390
Time:                                      04:06:16   BIC                           1320.197
Sample:                                           0   HQIC                          1268.375
                                              - 251
Covariance Type:                                opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.4291      2.424     -0.177      0.860      -5.180       4.322
ar.L2          0.2240      2.049      0.109      0.913      -3.792       4.240
ar.L3         -0.1284      0.115     -1.112      0.266      -0.355       0.098
ar.L4         -0.0223      0.338     -0.066      0.948      -0.686       0.641
ar.L5          0.0764      0.181      0.423      0.673      -0.278       0.431
ar.L6          0.0745      0.176      0.423      0.673      -0.271       0.420
ar.L7          0.0158      0.258      0.061      0.951      -0.490       0.521
ar.L8         -0.0034      0.164     -0.021      0.983      -0.324       0.317
ar.L9          0.0091      0.093      0.097      0.922      -0.174       0.192
ar.L10        -0.0438      0.099     -0.440      0.660      -0.239       0.151
ma.L1          0.5165      2.432      0.212      0.832      -4.251       5.284
ma.L2         -0.2374      2.251     -0.105      0.916      -4.649       4.174
ar.S.L12      -1.6115      1.467     -1.099      0.272      -4.486       1.263
ar.S.L24      -1.0679      2.051     -0.521      0.603      -5.088       2.952
ar.S.L36      -0.8434      1.574     -0.536      0.592      -3.928       2.241
ar.S.L48      -0.9530      1.330     -0.716      0.474      -3.561       1.655
ar.S.L60      -1.0238      1.379     -0.742      0.458      -3.726       1.679
ar.S.L72      -0.8278      1.388     -0.596      0.551      -3.549       1.893
ar.S.L84      -0.5033      1.109     -0.454      0.650      -2.677       1.670
ar.S.L96      -0.4503      0.726     -0.620      0.535      -1.873       0.972
ar.S.L108     -0.3659      0.602     -0.607      0.544      -1.547       0.815
ar.S.L120     -0.0785      0.376     -0.208      0.835      -0.816       0.659
ma.S.L12       0.6350      1.526      0.416      0.677      -2.356       3.626
ma.S.L24      -0.2259      0.811     -0.279      0.781      -1.816       1.364
sigma2         7.5374      0.959      7.860      0.000       5.658       9.417
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               108.30
Prob(Q):                              0.94   Prob(JB):                         0.00
Heteroskedasticity (H):               0.84   Skew:                             0.10
Prob(H) (two-sided):                  0.43   Kurtosis:                         6.30
===================================================================================

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

1. Residuals show no significant autocorrelation or heteroskedasticity however they deviate significantly from normality.
2. Despite similar fit metrics, neither model clearly outperforms the other based on information criteria alone, as both are complex and might overfit.

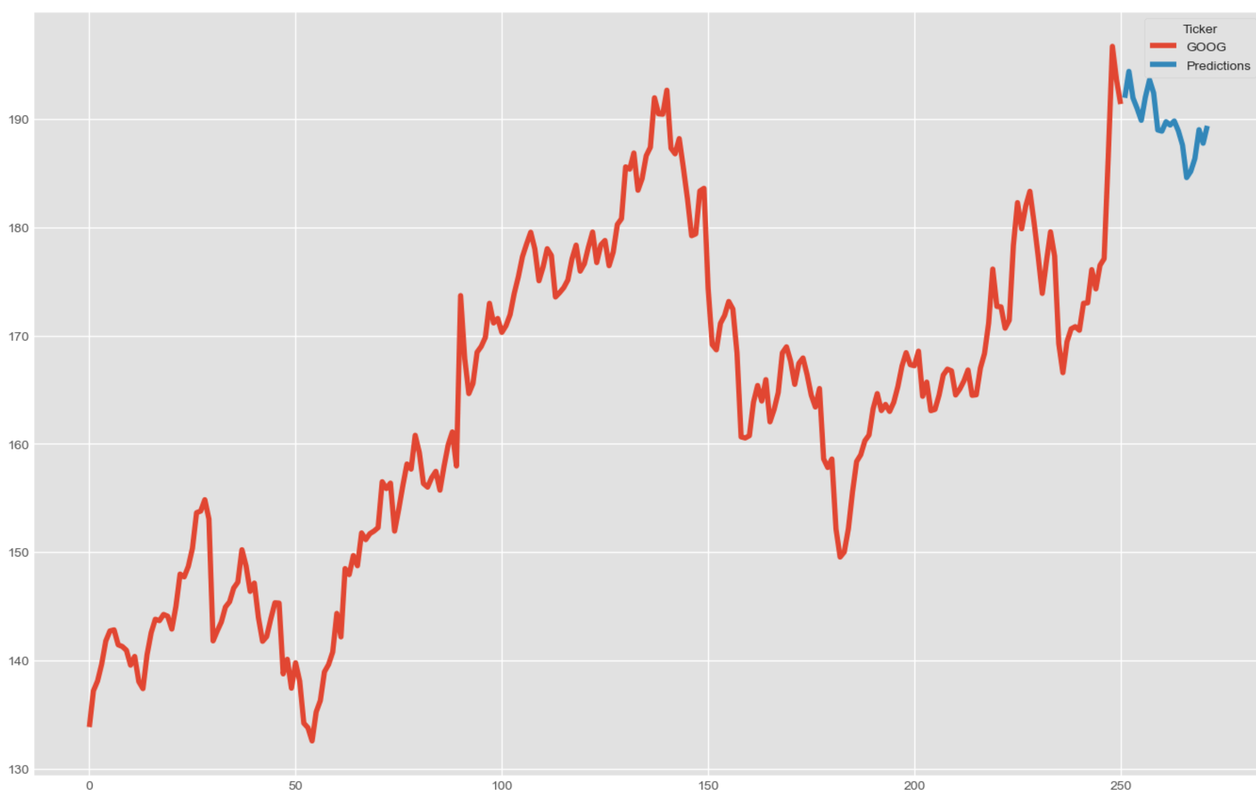
3. Both models include terms with high p-values (insignificant terms) that suggest potential model simplification.

In [27]:
```
predictions = model.predict(len(data), len(data)+20)
print(predictions)
```

```
251    191.956664
252    194.400235
253    191.935900
254    190.973382
255    189.875420
256    192.028742
257    193.620999
258    192.406503
259    188.985731
260    188.878185
261    189.745588
262    189.441284
263    189.815998
264    188.895635
265    187.585820
266    184.590406
267    185.164717
268    186.353954
269    189.013126
270    187.760363
271    189.364336
Name: predicted_mean, dtype: float64
```

In [28]:
```
data["Close"].plot(legend=True, label="Training Data", figsize=(15, 10))
predictions.plot(legend=True, label="Predictions")
```

Out[28]: <Axes: >



PREDICTING CLOSING STOCK PRICE FOR AMZN USING LSTM

In [29]:
```
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       Close   Adj Close  \
Ticker                    AMZN        AMZN        AMZN        AMZN        AMZN
246    2024-12-09  227.210007  230.080002  225.669998  226.089996  226.089996
247    2024-12-10  226.089996  229.059998  224.199997  225.039993  225.039993
248    2024-12-11  226.410004  231.199997  226.259995  230.259995  230.259995
249    2024-12-12  229.830002  231.089996  227.630005  228.970001  228.970001
250    2024-12-13  228.470001  230.199997  225.860794  227.460007  227.460007

Price      Volume
Ticker       AMZN
246    46819400
247    31199900
248    35385800
249    28204100
250    28249154
```

In [30]:
```
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 [31]:
```
df = df["Close"]
print(df.head())
```

```
Ticker        AMZN
0       149.970001
1       154.070007
2       153.789993
3       152.119995
4       153.839996
```

In [32]:
```
dataset = df.values
training_data_len = int(np.ceil( len(dataset) * .95 ))
```

In [33]:
```
from sklearn.preprocessing import MinMaxScaler

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

scaled_data
```

```
Out[33]: array([[0.06301779],
                [0.11086476],
                [0.107597  ],
                [0.08810817],
                [0.10818054],
                [0.10327917],
                [0.10316254],
                [0.10234555],
                [0.10281245],
                [0.08600766],
                [0.06255089],
                [0.04551283],
                [0.        ],
                [0.00781886],
                [0.05286497],
                [0.07935569],
                [0.10689683],
                [0.12381826],
                [0.11728311],
                [0.10024504],
                [0.08332361],
                [0.10421279],
                [0.1256855 ],
                [0.11915034],
                [0.13362118],
                [0.14354055],
                [0.15381018],
                [0.16979799],
                [0.19477173],
                [0.16839765],
                [0.12405171],
                [0.17166523],
                [0.31789   ],
                [0.30038504],
                [0.28684783],
                [0.30295245],
                [0.29490014],
                [0.34869873],
                [0.32407507],
                [0.2808962 ],
                [0.3082039 ],
                [0.29443342],
                [0.29104902],
                [0.26269107],
                [0.28031267],
                [0.35021588],
                [0.35500062],
                [0.35196631],
                [0.33807901],
                [0.33364454],
                [0.37565634],
                [0.39269458],
                [0.38522581],
                [0.34484762],
                [0.33772892],
                [0.37635669],
                [0.35920181],
                [0.31964061],
                [0.35966853],
                [0.37332238],
                [0.39887966],
                [0.34834864],
                [0.34904881],
                [0.36562016],
```

```
[0.3918776 ],
[0.3918776 ],
[0.40028   ],
[0.41008291],
[0.3936282 ],
[0.41148325],
[0.41790177],
[0.42478702],
[0.42151944],
[0.44159181],
[0.41346713],
[0.47263398],
[0.47403432],
[0.47963586],
[0.48290344],
[0.51908043],
[0.48500413],
[0.45571238],
[0.45221153],
[0.42840468],
[0.40436456],
[0.35079942],
[0.38114125],
[0.40809886],
[0.37367247],
[0.33959616],
[0.40903248],
[0.42467038],
[0.35511725],
[0.40179715],
[0.46854942],
[0.48593775],
[0.51499587],
[0.51569604],
[0.50682692],
[0.52433189],
[0.50075849],
[0.49013895],
[0.49597393],
[0.48337034],
[0.45582919],
[0.46831597],
[0.45477876],
[0.45022748],
[0.4499942 ],
[0.42572063],
[0.42221961],
[0.4385575 ],
[0.43704052],
[0.40553163],
[0.37192204],
[0.39409492],
[0.4057649 ],
[0.42840468],
[0.471817  ],
[0.46364805],
[0.49585712],
[0.49784099],
[0.49387325],
[0.45816315],
[0.45617928],
[0.4608472 ],
[0.44625973],
[0.48465404],
```

```
[0.51943052],
[0.47846897],
[0.48745472],
[0.57229549],
[0.62177625],
[0.56809429],
[0.61419066],
[0.64686662],
[0.61874194],
[0.64686662],
[0.63858086],
[0.6391644 ],
[0.64441585],
[0.58910028],
[0.58256513],
[0.56190922],
[0.56541025],
[0.50600994],
[0.45722953],
[0.4499942 ],
[0.4432256 ],
[0.48827171],
[0.42315323],
[0.4117167 ],
[0.44264206],
[0.45081101],
[0.43342286],
[0.4949235 ],
[0.46096401],
[0.27226036],
[0.19197105],
[0.20259059],
[0.21239351],
[0.24775352],
[0.26105728],
[0.25942349],
[0.29945142],
[0.29793445],
[0.38534244],
[0.37915737],
[0.39269458],
[0.40039681],
[0.41475083],
[0.36830438],
[0.37892392],
[0.36095224],
[0.33317764],
[0.30610339],
[0.32150767],
[0.39596216],
[0.36970472],
[0.33562841],
[0.38884347],
[0.31298863],
[0.35978517],
[0.40821567],
[0.46621546],
[0.49515695],
[0.48920533],
[0.4705333 ],
[0.49375661],
[0.48850498],
[0.52864972],
[0.54883891],
```

```
       [0.57544643],
       [0.57638005],
       [0.5596919 ],
       [0.54370409],
       [0.50647684],
       [0.48733809],
       [0.47333415],
       [0.46901614],
       [0.43634035],
       [0.4894386 ],
       [0.42280314],
       [0.44520947],
       [0.47380087],
       [0.49107239],
       [0.51639639],
       [0.50145866],
       [0.50320926],
       [0.49387325],
       [0.50134202],
       [0.51838026],
       [0.51931388],
       [0.52666585],
       [0.46843279],
       [0.48792162],
       [0.50484305],
       [0.51137821],
       [0.53985297],
       [0.56202586],
       [0.48815489],
       [0.62270969],
       [0.59761932],
       [0.64103164],
       [0.7296067 ],
       [0.76414991],
       [0.74232693],
       [0.72668921],
       [0.75084614],
       [0.81141334],
       [0.78083789],
       [0.67732527],
       [0.66670555],
       [0.70066521],
       [0.68047621],
       [0.62796132],
       [0.61325704],
       [0.66378805],
       [0.73859263],
       [0.71385234],
       [0.73894272],
       [0.77185213],
       [0.80371111],
       [0.85879341],
       [0.88668464],
       [0.96230603],
       [0.95133623],
       [0.93908272],
       [1.        ],
       [0.98494581],
       [0.96732421]])
```

In [34]:
```
train_data = scaled_data[0:int(training_data_len), :]

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()

x_train, y_train = np.array(x_train), np.array(y_train)

x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

```
[array([0.06301779, 0.11086476, 0.107597  , 0.08810817, 0.10818054,
       0.10327917, 0.10316254, 0.10234555, 0.10281245, 0.08600766,
       0.06255089, 0.04551283, 0.        , 0.00781886, 0.05286497,
       0.07935569, 0.10689683, 0.12381826, 0.11728311, 0.10024504,
       0.08332361, 0.10421279, 0.1256855 , 0.11915034, 0.13362118,
       0.14354055, 0.15381018, 0.16979799, 0.19477173, 0.16839765,
       0.12405171, 0.17166523, 0.31789   , 0.30038504, 0.28684783,
       0.30295245, 0.29490014, 0.34869873, 0.32407507, 0.2808962 ,
       0.3082039 , 0.29443342, 0.29104902, 0.26269107, 0.28031267,
       0.35021588, 0.35500062, 0.35196631, 0.33807901, 0.33364454,
       0.37565634, 0.39269458, 0.38522581, 0.34484762, 0.33772892,
       0.37635669, 0.35920181, 0.31964061, 0.35966853, 0.37332238])]
[0.39887965676712]

[array([0.06301779, 0.11086476, 0.107597  , 0.08810817, 0.10818054,
       0.10327917, 0.10316254, 0.10234555, 0.10281245, 0.08600766,
       0.06255089, 0.04551283, 0.        , 0.00781886, 0.05286497,
       0.07935569, 0.10689683, 0.12381826, 0.11728311, 0.10024504,
       0.08332361, 0.10421279, 0.1256855 , 0.11915034, 0.13362118,
       0.14354055, 0.15381018, 0.16979799, 0.19477173, 0.16839765,
       0.12405171, 0.17166523, 0.31789   , 0.30038504, 0.28684783,
       0.30295245, 0.29490014, 0.34869873, 0.32407507, 0.2808962 ,
       0.3082039 , 0.29443342, 0.29104902, 0.26269107, 0.28031267,
       0.35021588, 0.35500062, 0.35196631, 0.33807901, 0.33364454,
       0.37565634, 0.39269458, 0.38522581, 0.34484762, 0.33772892,
       0.37635669, 0.35920181, 0.31964061, 0.35966853, 0.37332238]), array([0.11086476, 0.107597  ,
0.08810817, 0.10818054, 0.10327917,
       0.10316254, 0.10234555, 0.10281245, 0.08600766, 0.06255089,
       0.04551283, 0.        , 0.00781886, 0.05286497, 0.07935569,
       0.10689683, 0.12381826, 0.11728311, 0.10024504, 0.08332361,
       0.10421279, 0.1256855 , 0.11915034, 0.13362118, 0.14354055,
       0.15381018, 0.16979799, 0.19477173, 0.16839765, 0.12405171,
       0.17166523, 0.31789   , 0.30038504, 0.28684783, 0.30295245,
       0.29490014, 0.34869873, 0.32407507, 0.2808962 , 0.3082039 ,
       0.29443342, 0.29104902, 0.26269107, 0.28031267, 0.35021588,
       0.35500062, 0.35196631, 0.33807901, 0.33364454, 0.37565634,
       0.39269458, 0.38522581, 0.34484762, 0.33772892, 0.37635669,
       0.35920181, 0.31964061, 0.35966853, 0.37332238, 0.39887966])]
[0.39887965676712, 0.34834864406166965]
```

In [35]:
```
!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) (2024.8.30)
```

```
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\lib\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 (from 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-packages (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 markdown-it-py>=2.2.0->rich->keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (0.1.0)
```

In [36]:

```
from keras.models import Sequential
from keras.layers import Dense, LSTM

# 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))

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

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 pass 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 ━━━━━━━━━━━━━━━━━━━━ 14s 38ms/step - loss: 0.0160
```

Out[36]:

```
<keras.src.callbacks.history.History at 0x1d982c183d0>
```

In [37]:

```
test_data = scaled_data[training_data_len - 60: , :]

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])

x_test = np.array(x_test)

x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1 ))

predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)

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

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 818ms/step
```

Out[37]:

```
15.112473693883775
```

In [38]:

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

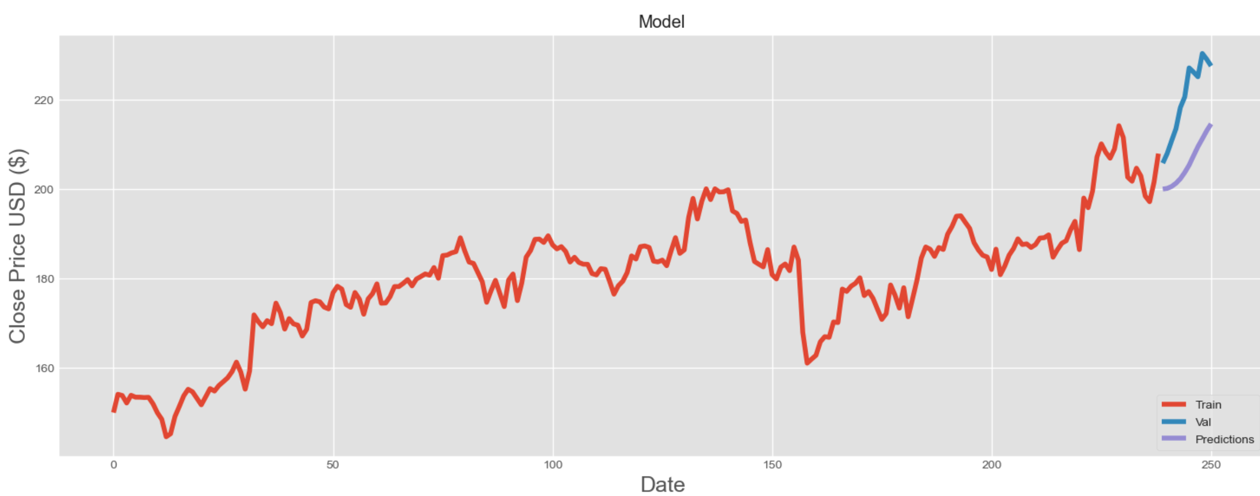
```
C:\Users\shubh\AppData\Local\Temp\ipykernel_42160\84718757.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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

Out[38]:

Ticker	AMZN	Predictions
239	205.740005	199.938980
240	207.889999	200.093216
241	210.710007	200.515366
242	213.440002	201.240036
243	218.160004	202.250870
244	220.550003	203.653549
245	227.029999	205.298019
246	226.089996	207.377975
247	225.039993	209.413376
248	230.259995	211.165131
249	228.970001	212.969986
250	227.460007	214.513351

In [39]:

```
# Visualizing 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()
```



Results : The model has learned the underlying patterns in the historical data and is able to capture the general direction of the time series. However, towards the end of the prediction period, the predictions deviate from the actual data. This suggests that the model's accuracy might decrease as the prediction horizon increases.

Possible Reasons for Prediction Deviation can be model complexity, data variability or bias.