D214 - Task 2 - Data Analytics Report and Executive Summary

Jonathon "Jon" Fryman
Data Analytics 01/01/2022
Student ID 00974544
Program Mentor: Lea Yoakem
C: 419-206-6989
jfryma1@wgu.edu

Research Question

A. Summarize the original real-data research question you identified in task 1. Your summary should include justification for the research question you identified in task 1, a description of the context in which the research question exists, and a discussion of your hypothesis.

Original Research Question
To what extent can a company's future daily per-share closing stock dollar value be accurately forecast? With so many stocks to choose from, it is often difficult for investors to know the best companies to invest in as well as the best time to invest in them. In an attempt to explore these questions, this author will analyze data associated value and volume for 14 of the largest American based tech companies. To assist with this analysis, several data analytics tools and techniques willl be utilized.

Research Question Justification

The justification for the outlined research question is to provide a better understanding in relation to the price and volume for the given companies at different points in time throughout the last 13 years.
This will faciliate an understanding regarding the existance of seasonality, trends associated with volume, and market fluctuations.

Description of Context for Research Question

As is commonly known, the stock market can often times be riddled with uncertainty and great fluctuation. In an effort to facilitate educated investing, it is prudent to attempt to ascertain trends and seasonality that exists within a given stock prior to making a large investment.

As outlined in the preceding sections of this report, 14 of the largest tech companies will have their market performance analyzed for the previous 13 years of available data in an effort to forecast expected future performance for the respective company's stock price. Per the New York Stock Exchanges available data, an average of $18.9 billion is traded on the stock market per day. With such a massive amount of money invested, an investor stands to lose a significant amount of funding if poor investments are made based on a lack of research into the available data for a given stock.

Utilizing the historical stock information for the companies being analyzed, the objective is to

Hypothesis Discussion

The hypothesis of this research project is that the performance of a specific subset of 14 tech companies market performance can be accurately forcast by training on historical performance and validating the accuracy of the forecast by generating a forecast and comparing to a subset of the historical dataset aside for validation.

Data Collection

B. Report on your data-collection process by describing the relevant data you collected, discussing one advantage and one disadvantage of the data-gathering methodology you used, and discussing how you overcame any challenges you encountered during the process of collecting your data.

The data used to further analysis the previously proposed research question is publicly available at https://www.kaggle.com/datasets/evangower/big-tech-stock-prices.
The full dataset consists of 14 unique csv files. Each file is associated with a specific tech companies stock performance for trading days beginning in 2010. The comanies included for analysis are:

- Adobe
- Amazon
- Apple
- Cisco
- Google
- IBM
- Intel
- Meta
- Microsoft
- Netflix
- Oracle
- Salesforce
- Tesla

These columns contained within each of the individual csv files are:

- Date

- Open
- High
- Low
- Close
- Adj Close
- Volume

The data being used for this analysis is provided under a CC0 1.0 Universal Public Domain Dedication license that allows users to share and adapt the data with proper credit given to the original data provider

Data Gathering: The data-gathering methodology to be used for this analysis is documents and records. This methodology consists of examining existing data. For this specific analysis, this includes examining existing records related to historical stock prices for 14 tech companies over a period beginning in 2010. This includes historical open and close prices, and overall volume.

Advantage of Data-Gathering Methodology

Utilizing a CC0 1.0 Public Licensed Dataset has a multitude of advantages. While not impossible to independently ascertain the data included within this dataset, it would take considerable time and effort to gather such an extensive time period for 14 companies.
Additionally, when utilizing a public dataset, it allows any number of data analyists to draw conclusions the same core data. This can facilitate shared learning, and expand the insight drawn that may not have been immediatly observed by any individual analysis performed.

Disadvantage of Data-Gathering Methodology

Similar to a public dataset providing many advantages to the analysts utilizing it, there are often disadvantages associated with this method of data gathering as well.
A primary disadvantage is a lack of control over selection of included companies, time period, or variables.

Another disadvantage is trusting that the included datapoints have been correctly gathered and properly reflect the information to be analyzed. It does not serve a data analyst well to perform analysis on inaccurate statistics.

In fact, it could lead to complications for a company if recommended course of action is developed based upon a set of data that inaccurately portrays the subject matter being analyzed. This could lead to harm to the companies bottom-line and public reputation.

Challenges

One of the primary challenges with analyzing a public dataset is ensuring the data is properly cleaned and able to adequetly align with the selected research topic.

In a typical business setting, a specific question is posed that requires analysis to be completed. It would be at this time that the data required to perform the analysis is gathered in a way that best aligns with the specific goals of the analysis outlined. When utilizing publicly available datasets, the data is compiled independent of any specific research question which requires time and energy to cater the data in the direction needed.

For the current analysis, time was spent to ensure a proper understanding of what information was contained within the existing data and performing analysis to determine what, if any, cleaning and reconfiguration of the data would be required.

This specific dataset contained a folder of 14 unique CSV files, each associated with a specific tech companies stock performance over a time period often dating back to 2010. It required time to be spent to determine if all companies had data available for the same time period, or if some of the companies included only included information for an abbreviated period in comparison to

Data Extraction and Preparation

C. Describe your data-extraction and -preparation process and provide screenshots to illustrate each step. Explain the tools and techniques you used for data extraction and data preparation, including how these tools and techniques were used on the data. Justify why you used these particular tools and techniques, including one advantage and one disadvantage when they are used with your data-extraction and -preparation methods.

To extract, clean, and prepare the data, a set of standard techniques were used utilizing common Python packages such as Pandas, Numpy, and MatPlotLib.
The primary environment used was Jupyter Notebooks in partnership with Python version 3.8.

The steps required to properly prepare the Jupyter Notebook environment, import the required libraries, and clean the original data are outlined in the following sections.

An explanation, justification, advanatages, and disadvantages for each step of the prepation process have been included in the subsequent sections for review.

Import Python Packages

In [1]:
```
from sklearn import preprocessing
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook
import numpy as np
import pandas as pd
from itertools import product
from pandas.plotting import lag_plot
from pandas import datetime

from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import mean_squared_error, mean_absolute_error
from pmdarima.arima import auto_arima

import chart_studio.plotly as py
import plotly.figure_factory as ff
import pandas as pd

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

```
/var/folders/xk/vmj396hs3rn3tfr6yw2d6zn80000gn/T/ipykernel_51143/14664969
13.py:8: FutureWarning: The pandas.datetime class is deprecated and will
be removed from pandas in a future version. Import from datetime module i
nstead.
  from pandas import datetime
```

- **Explanation**: This step imports the packages required to load the existing code. It also facilitates the cleaning and preperation steps of the analysis process
- **Justification**: Utilizing the existing packages and modules removes the requirement of having to manually code similar functions to facilitate the common step of data preperation.
- **Advantage**: This step saves a significant amount of time and utilizes existing trial and error by the Python Package developer to ensure as many issues as possible have already been resolved.
- **Disadvantage**: There can be a slight learning curve in having to learn what exists within the existing packages to ensure the functionality is properly utilized.

Read in the existing data

In [2]:
```
CSCO = pd.read_csv("Datasets/CSCO.csv")
ADBE = pd.read_csv("Datasets/ADBE.csv")
ORCL = pd.read_csv("Datasets/ORCL.csv")
AMZN = pd.read_csv("Datasets/AMZN.csv")
INTC = pd.read_csv("Datasets/INTC.csv")
MSFT = pd.read_csv("Datasets/MSFT.csv")
NVDA = pd.read_csv("Datasets/NVDA.csv")
IBM = pd.read_csv("Datasets/IBM.csv")
NFLX = pd.read_csv("Datasets/NFLX.csv")
GOOGL = pd.read_csv("Datasets/GOOGL.csv")
AAPL = pd.read_csv("Datasets/AAPL.csv")
CRM = pd.read_csv("Datasets/CRM.csv")
```

In [3]:
```
AAPL.Date = pd.to_datetime(AAPL.Date)
ADBE.Date = pd.to_datetime(ADBE.Date)
ORCL.Date = pd.to_datetime(ORCL.Date)
AMZN.Date = pd.to_datetime(AMZN.Date)
INTC.Date = pd.to_datetime(INTC.Date)
MSFT.Date = pd.to_datetime(MSFT.Date)
NVDA.Date = pd.to_datetime(NVDA.Date)
IBM.Date = pd.to_datetime(IBM.index)
NFLX.Date = pd.to_datetime(NFLX.Date)
GOOGL.Date = pd.to_datetime(GOOGL.Date)
CRM.Date = pd.to_datetime(CRM.Date)
CSCO.Date = pd.to_datetime(CSCO.Date)
```

In [4]:
```
CSCO
```

Out[4]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2010-01-04	24.110001	24.840000	24.010000	24.690001	17.394129	59853700
1	2010-01-05	24.600000	24.730000	24.379999	24.580000	17.316635	45124500
2	2010-01-06	24.540001	24.740000	24.340000	24.420000	17.203917	35715700
3	2010-01-07	24.299999	24.570000	24.170000	24.530001	17.281410	31531200
4	2010-01-08	24.379999	24.700001	24.250000	24.660000	17.372995	39115900
...	...	...	...	...	...	...	...
3266	2022-12-22	47.490002	47.490002	46.689999	47.320000	46.944912	23118500
3267	2022-12-23	47.250000	47.490002	47.009998	47.480000	47.103645	9554400
3268	2022-12-27	47.669998	47.709999	47.220001	47.529999	47.153248	12066200
3269	2022-12-28	47.689999	47.770000	46.980000	47.070000	46.696896	9847400
3270	2022-12-29	47.259998	47.740002	47.259998	47.500000	47.123486	11396500

3271 rows × 7 columns

CSCO.to_csv("Datasets2/CSCO.csv") ADBE.to_csv("Datasets2/ADBE.csv") ORCL.to_csv("Datasets2/ORCL.csv") AMZN.to_csv("Datasets2/AMZN.csv") INTC.to_csv("Datasets2/INTC.csv") MSFT.to_csv("Datasets2/MSFT.csv") NVDA.to_csv("Datasets2/NVDA.csv") IBM.to_csv("Datasets2/IBM.csv")

NFLX.to_csv("Datasets2/NFLX.csv") TSLA.to_csv("Datasets2/TSLA.csv")
GOOGL.to_csv("Datasets2/GOOGL.csv") META.to_csv("Datasets2/META.csv")
AAPL.to_csv("Datasets2/AAPL.csv") CRM.to_csv("Datasets2/CRM.csv")

In [5]:
```
AAPL.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3271 entries, 0 to 3270
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       3271 non-null   datetime64[ns]
 1   Open       3271 non-null   float64
 2   High       3271 non-null   float64
 3   Low        3271 non-null   float64
 4   Close      3271 non-null   float64
 5   Adj Close  3271 non-null   float64
 6   Volume     3271 non-null   int64
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 179.0 KB
```

In [6]:
```
AAPL.columns
```

Out[6]:
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dt
ype='object')
```

In [7]:
```
AAPL.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3271 entries, 0 to 3270
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       3271 non-null   datetime64[ns]
 1   Open       3271 non-null   float64
 2   High       3271 non-null   float64
 3   Low        3271 non-null   float64
 4   Close      3271 non-null   float64
 5   Adj Close  3271 non-null   float64
 6   Volume     3271 non-null   int64
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 179.0 KB
```

- **Explanation**: Through the use of the pandas library, a csv file can be converted into a DataFrame.
- **Justification**: The original dataset consisted of 14 unique csv files, making pandas the logical choice of package to read in the data for analysis
- **Advantage**: Pandas is a widely adapted package with a significant amount of documentation available. This facilitates an ease of use few other Python packages have available.
- **Disadvantage**: In comparison to opening a .csv file in an application like Microsoft Excel, pandas does require additional steps be taken before being able to fully view the information contained within the data.

Join the DataFrames together using a common key

As shown in the following section, the original dataframes associated with each of the tech companies included for review have been joined into a single DataFrame. This was facilitated by creating an empty DataFrame with a index column associated with the dates included within the original .csv files. A column was then created for each of the companies that will contain the companies closing price for their stock for all of the days included for review.

In [8]:
```
beginning_date = '2010-01-04'
print(beginning_date)
```

```
2010-01-04
```

In [9]:
```
index = pd.date_range(beginning_date, periods=3254, freq='B')

columns = ["AAPL", "ADBE" ,"ORCL", "AMZN","INTC", "MSFT", "NVDA",
           "IBM","NFLX", "GOOGL","CRM", "CSCO"]
```

In [10]:
```
df = pd.DataFrame(index=index, columns=columns)
df.to_csv("df.csv")
```

In [11]:
```
df = pd.read_csv("df.csv")
```

In [12]:
```
df = df.rename(columns={"Unnamed: 0":"Date"})
```

In [13]:
```
df.Date = pd.to_datetime(df.Date)
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3254 entries, 0 to 3253
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    3254 non-null   datetime64[ns]
 1   AAPL    0 non-null      float64
 2   ADBE    0 non-null      float64
 3   ORCL    0 non-null      float64
 4   AMZN    0 non-null      float64
 5   INTC    0 non-null      float64
 6   MSFT    0 non-null      float64
 7   NVDA    0 non-null      float64
 8   IBM     0 non-null      float64
 9   NFLX    0 non-null      float64
 10  GOOGL   0 non-null      float64
 11  CRM     0 non-null      float64
 12  CSCO    0 non-null      float64
dtypes: datetime64[ns](1), float64(12)
memory usage: 330.6 KB
```

In [15]:
```
df["AAPL"] = AAPL["Close"]
df["ADBE"] = ADBE["Close"]
df["ORCL"] = ORCL["Close"]
df["AMZN"] = AMZN["Close"]
df["INTC"] = INTC["Close"]
df["MSFT"] = MSFT["Close"]
df["NVDA"] = NVDA["Close"]
df["IBM"] = IBM["Close"]
df["NFLX"] = NFLX["Close"]
df["GOOGL"] = GOOGL["Close"]
df["CRM"] = CRM["Close"]
df["CSCO"] = CSCO["Close"]
```

In [16]:
```
df.isna().sum()
```

Out[16]:
```
Date     0
AAPL     0
ADBE     0
ORCL     0
AMZN     0
INTC     0
MSFT     0
NVDA     0
IBM      0
NFLX     0
GOOGL    0
CRM      0
CSCO     0
dtype: int64
```

In [17]:
```
df.to_csv('DataSets/closing_df.csv')
```

- **Explanation**: As each of the 14 original tech companies included have their stock information included in seperate .csv files, steps were performed to combine the relevant information for all companies into a single DataFrame.
- **Justification**: Due to challenges of analyzing 14 files within a single Jupyter Notebook, it was determined that combining each of the companies closing price for each of the trading days included within the time period analyzed would be ideal. This was facilitated by creating an empty DataFrame with the date as the index and adding a column for each of the companies. The column associated with each individual company was then populated with their closing price that associated with the trading day reflected in the row index.
- **Advantage**: It is significantly easier to analyze a single DataFrame in comparison to performing analysis across 14 different files/DataFrames.
- **Disadvantage**: To prevent the single DataFrame from containing to much information, all columns besides the company's closing price data was excluded from the new DataFrame.

Review the datatypes contained within DataFrame

In [18]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3254 entries, 0 to 3253
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   Date    3254 non-null   datetime64[ns]
 1   AAPL    3254 non-null   float64       
 2   ADBE    3254 non-null   float64       
 3   ORCL    3254 non-null   float64       
 4   AMZN    3254 non-null   float64       
 5   INTC    3254 non-null   float64       
 6   MSFT    3254 non-null   float64       
 7   NVDA    3254 non-null   float64       
 8   IBM     3254 non-null   float64       
 9   NFLX    3254 non-null   float64       
 10  GOOGL   3254 non-null   float64       
 11  CRM     3254 non-null   float64       
 12  CSCO    3254 non-null   float64       
dtypes: datetime64[ns](1), float64(12)
memory usage: 330.6 KB
```

- **Explanation**: The preceding section contains the step taken to review the data type of each variable contained within the DataFrame.
- **Justification**: To ensure the data contained within the DataFrame is in an appropriate form to be used for analysis, it is imperative to review the datatype of each variable contained within the DataFrame.
- **Advantage**: This step allows corrective action to be taken if it is determined a varaiable or variables required for the specified analysis are not of an appropriate data type.
- **Disadvantage**: There are no disadvantages of performing this step.

Review value counts to ensure no missing values
Although there are a multitude of methodologies for determining if any columns within a DataFrame are absent of values, the following section shows a print out for each of the columns associated with a specific stock to reflect a total number of values contained within the respective column. Another method is using the .info() function contained within the previous section which includes the number of non-null records for each of the columms contained within the DataFrame.

In [19]:
```
columns = {'AAPL', 'ADBE', 'ORCL', 'AMZN', 'INTC', 'MSFT', 'NVDA', 'IBM',
        'NFLX', 'GOOGL', 'CRM', 'CSCO'}

for column in columns:
    column_name = df[column].name

    print(column_name,"Column Values:", df['AAPL'].value_counts().sum())
```

```
CSCO Column Values: 3254
INTC Column Values: 3254
NVDA Column Values: 3254
NFLX Column Values: 3254
ADBE Column Values: 3254
MSFT Column Values: 3254
GOOGL Column Values: 3254
CRM Column Values: 3254
AMZN Column Values: 3254
IBM Column Values: 3254
ORCL Column Values: 3254
AAPL Column Values: 3254
```

- **Explanation**: The preceding section reviews the DataFrame to ensure there are an equal number of values for all columns contained.
- **Justification**: Prior to completing analysis, it is important to ensure there are no gaps in the available data that could skew the analysis performed. It is also important to understand any reasons a given set of data would be missing.
- **Advantage**: The primary advantage associated with reviewing for missing data is to pre-emptivly position yourself to address any missing values prior to allowing the missing data to skew any analysis performed.
- **Disadvantage**: It can often be difficult to discern the reason a given set of data may be absent just from the identification that there are an inequal count of values across the fields of a DataFrame.

Review a statistical summary of each variable and plot the distribution

As seen in the following section, the use of the .describe() function provides several statistics related to numeric columns contained within a DataFrame. These statistics are:

- **count**
- **mean**
- **std**
- **min**
- **25%**
- **50%**
- **75%**
- **max**

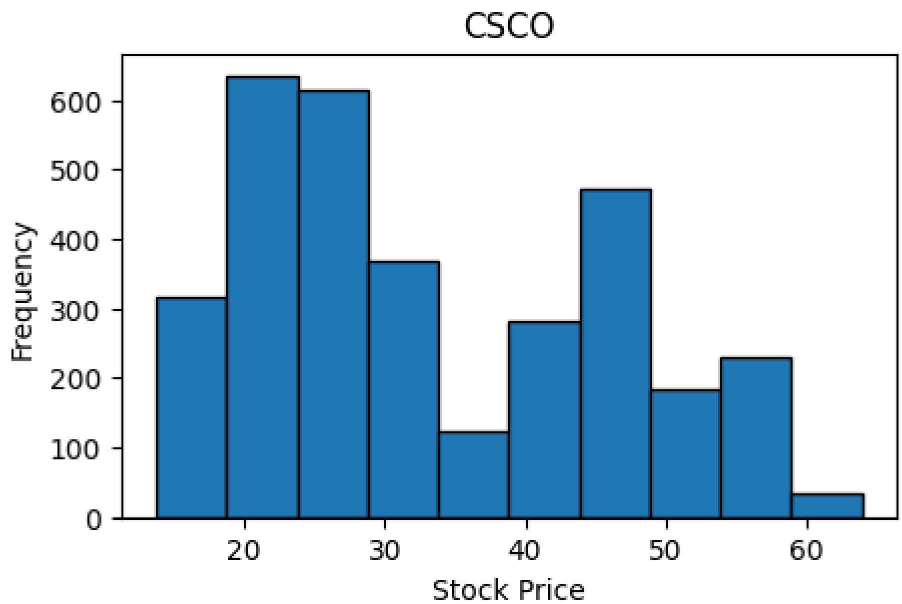
In [20]:
```
df.describe()
```

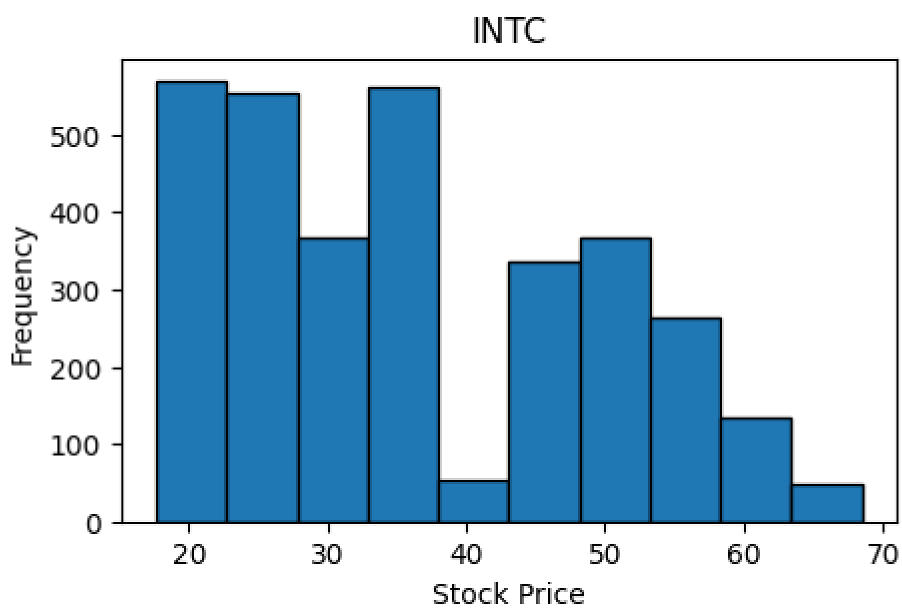
Out[20]:

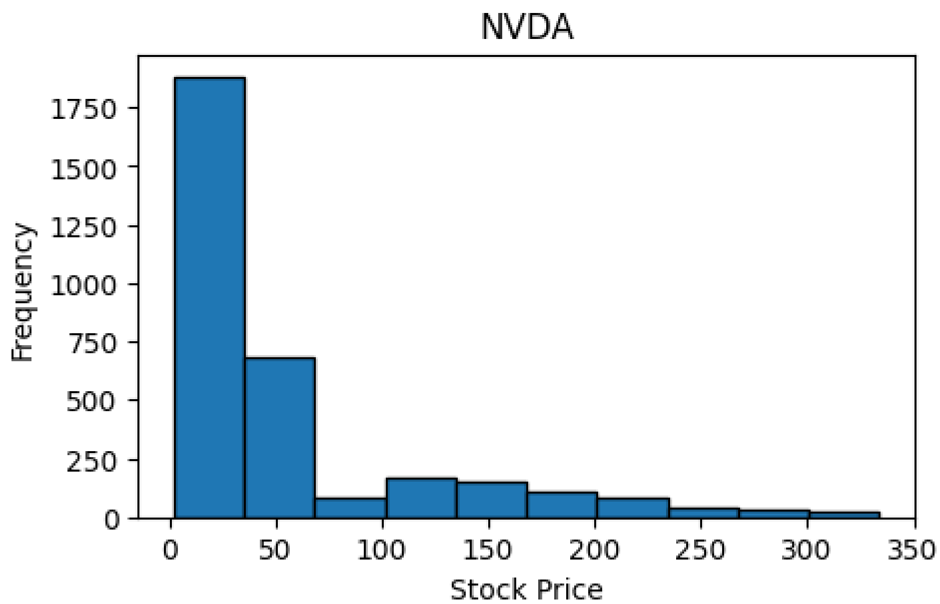
	AAPL	ADBE	ORCL	AMZN	INTC	MSFT	NVDA
count	3254.000000	3254.000000	3254.000000	3254.000000	3254.000000	3254.000000	3254.000000
mean	50.851340	185.244548	46.083685	58.757577	36.540627	99.317412	49.973840
std	47.061272	173.583362	16.724289	54.186993	12.928357	87.840796	69.187589
min	6.858929	22.690001	21.459999	5.430500	17.670000	23.010000	2.220000
25%	18.946161	42.727501	33.405001	13.302125	24.629999	31.790001	3.980625
50%	29.728750	97.089996	41.600001	35.968500	34.395001	55.349998	11.736250
75%	55.985000	285.884995	53.485002	93.215750	48.130001	138.119995	61.642499
max	182.009995	688.369995	103.650002	186.570496	68.470001	343.109985	333.760010

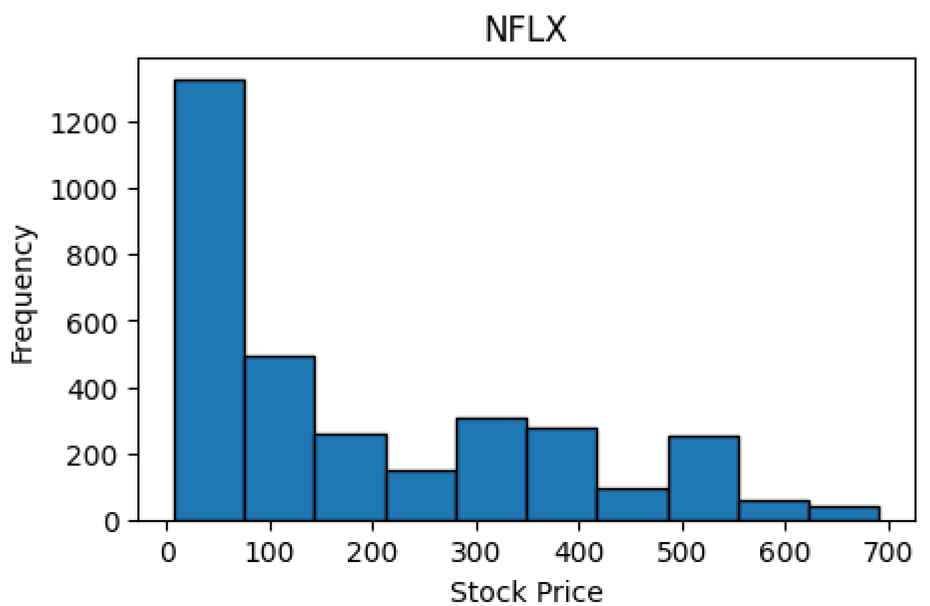
Additionally, a histogram distribution plot has been included for each of the companies closing prices.

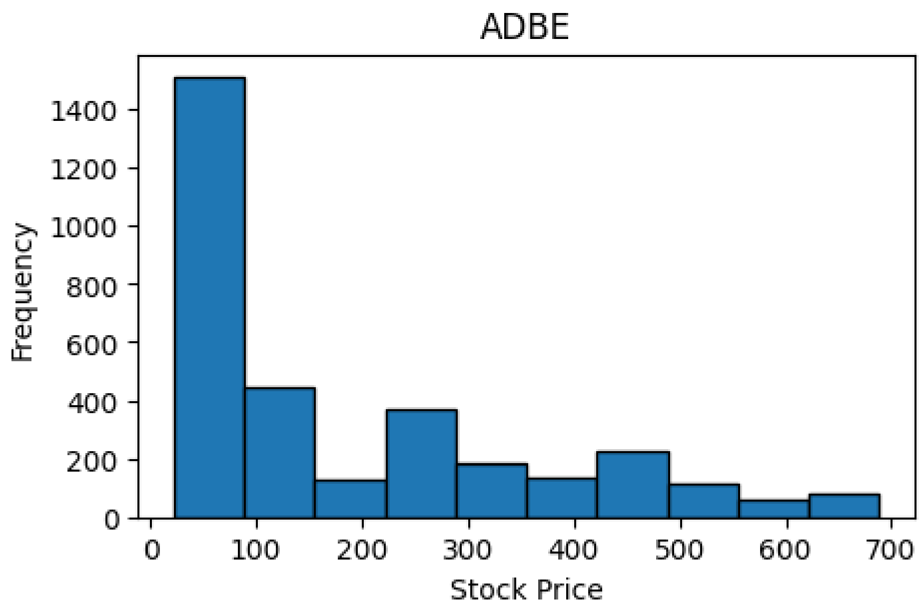
In [21]:
```
columns = {'AAPL', 'ADBE', 'ORCL', 'AMZN', 'INTC', 'MSFT', 'NVDA', 'IBM',
        'NFLX', 'GOOGL', 'CRM', 'CSCO'}

for column in columns:
    column_name = df[column].name
    plt.figure(figsize=(5,3))
    plt.title(column_name)
    plt.xlabel("Stock Price")
    df[column_name].plot(kind='hist', edgecolor='black')
```

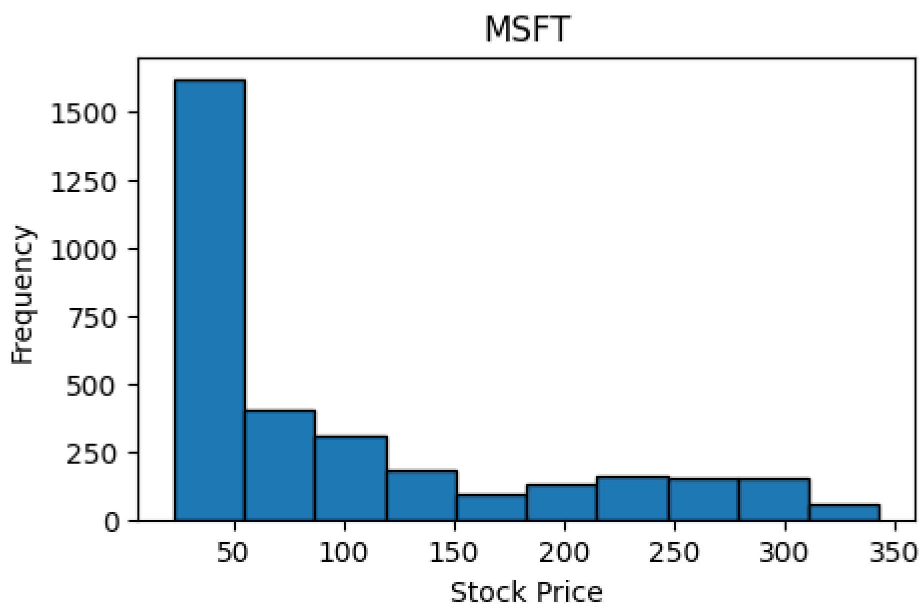


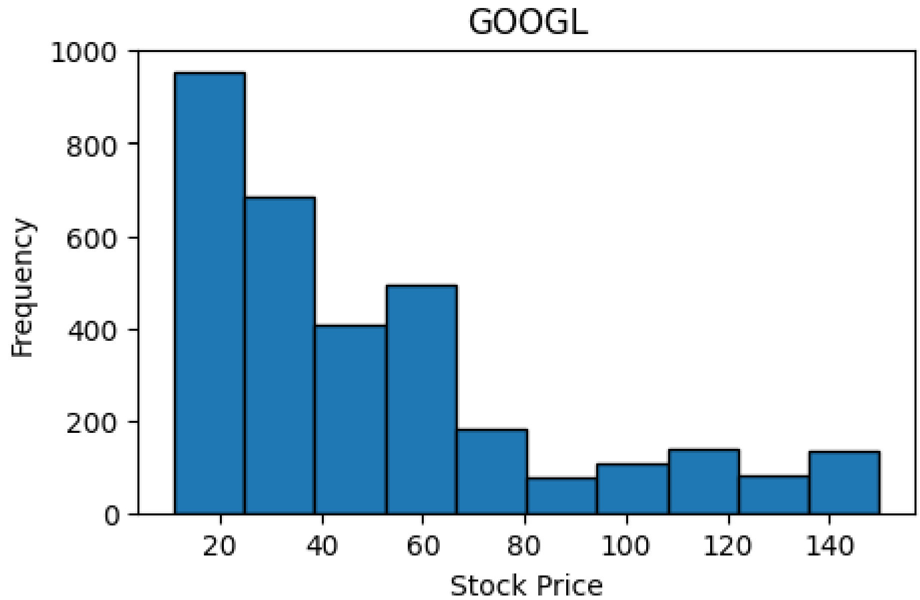


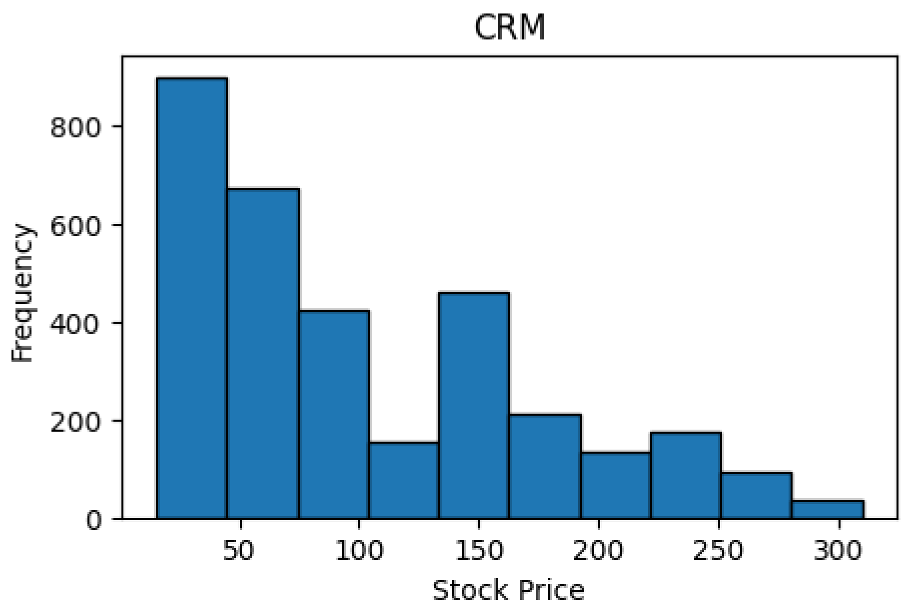

NVDA
Frequency
0
250
500
750
1000
1250
1500
1750
0
50
100
150
200
250
300
350
Stock Price

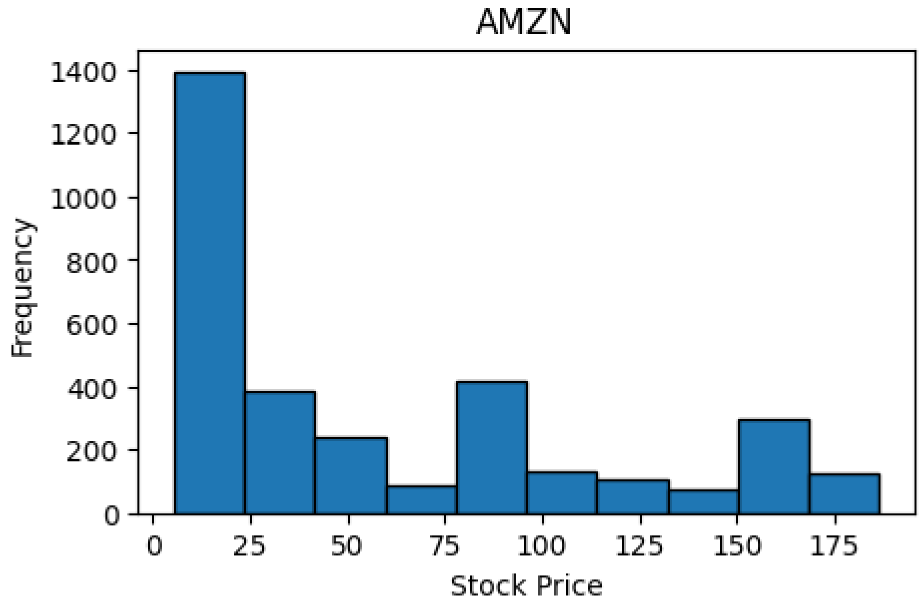

NFLX
Frequency
0
200
400
600
800
1000
1200
0
100
200
300
400
500
600
700
Stock Price

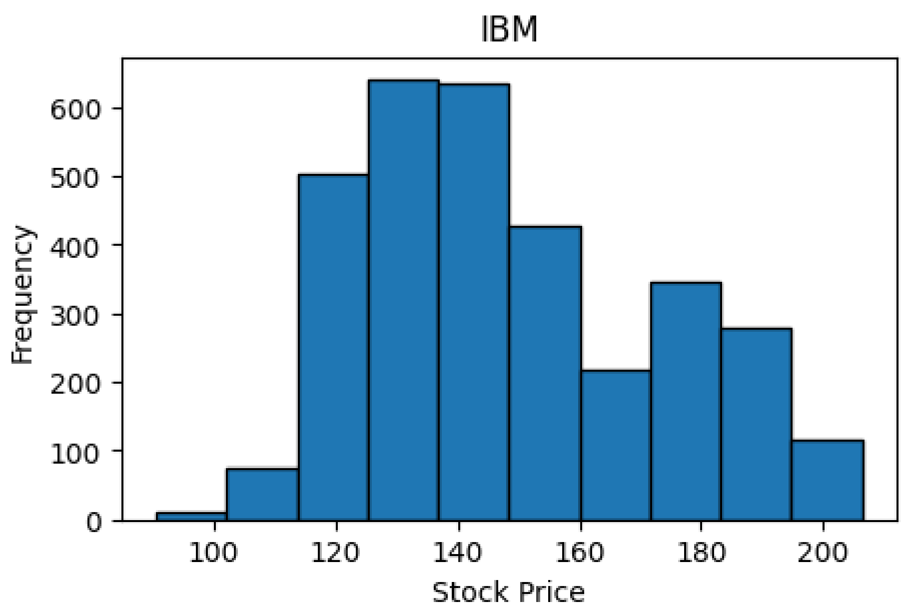

ADBE
Frequency
1400
1200
1000
800
600
400
200
0
0
100
200
300
400
500
600
700
Stock Price

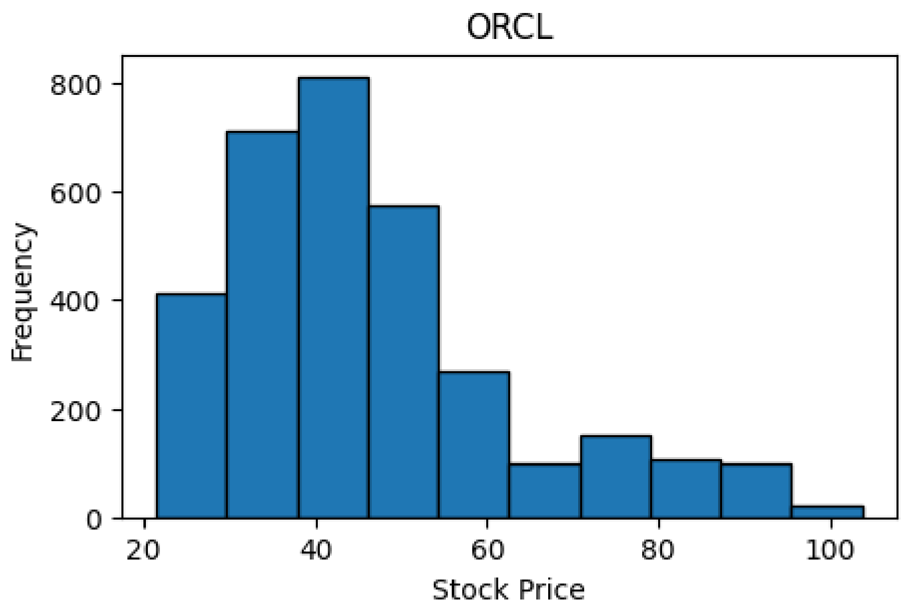

MSFT
Frequency
1500
1250
1000
750
500
250
0
50
100
150
200
250
300
350
Stock Price

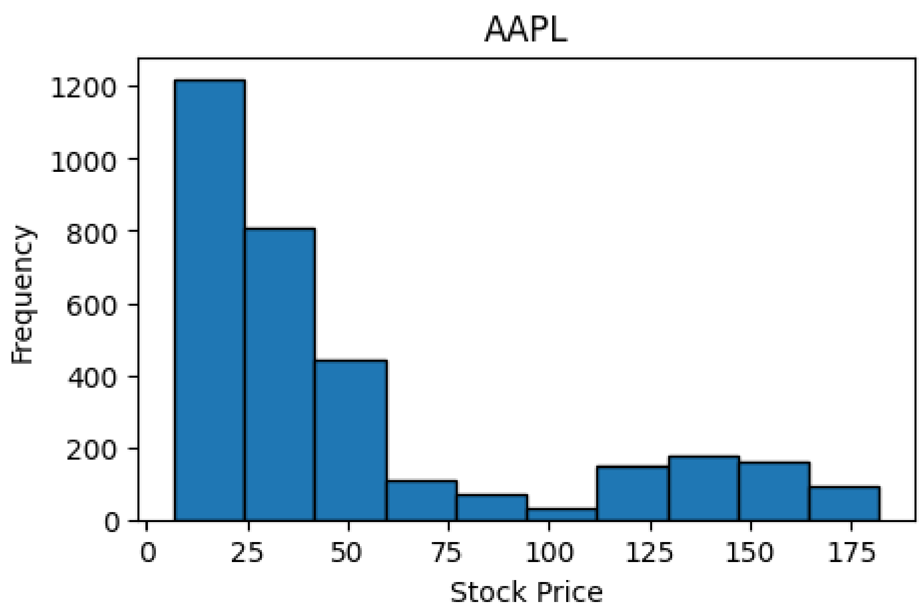

GOOGL
Frequency
0
200
400
600
800
1000
20
40
60
80
100
120
140
Stock Price


CRM
Frequency
0
200
400
600
800
50
100
150
200
250
300
Stock Price


AMZN
Frequency
Stock Price
0
200
400
600
800
1000
1200
1400
0
25
50
75
100
125
150
175


IBM
Frequency
Stock Price
0
100
200
300
400
500
600
100
120
140
160
180
200





- **Explanation**: The actions performed within this step provide a summary of the values found within each of the columns of the DataFrame. As outlined in the initial portion of this secion, a multitude of statistics are provide that provide calculations associated with each columns values.
- **Justification**: It is important to understand the data that is to be analyzed. This step provides a clear insight into the minimum, maximum, and average values found associated with closing stock prices for all the companies being analyzed.
- **Advantage**: The insight gained can assist with determining if outliers exist within the data that could be skewing the future analysis to be performed.
- **Disadvantage**: While this stage does provide the statistics that can be used to detect outlier data points, it does not provide any information regarding if that datapoint may have been collected in error or any other information that provides clarity into the methods used to collect it.

Check for null values

Similar to the section where the DataFrame is reviewed for missing values, the following overviews the review for any null values.

In [22]:
```
df.isnull().sum()
```

Out[22]:
```
Date     0
AAPL     0
ADBE     0
ORCL     0
AMZN     0
INTC     0
MSFT     0
NVDA     0
IBM      0
NFLX     0
GOOGL    0
CRM      0
CSCO     0
dtype: int64
```

In [23]:
```
columns = {'AAPL', 'ADBE', 'ORCL', 'AMZN', 'INTC', 'MSFT', 'NVDA', 'IBM',
        'NFLX', 'GOOGL', 'CRM', 'CSCO'}

for column in columns:
    column_name = df[column].name

    print(column_name, "Column Null Values:", df[column_name].isnull().sum(
```

```
CSCO Column Null Values: 0
INTC Column Null Values: 0
NVDA Column Null Values: 0
NFLX Column Null Values: 0
ADBE Column Null Values: 0
MSFT Column Null Values: 0
GOOGL Column Null Values: 0
CRM Column Null Values: 0
AMZN Column Null Values: 0
IBM Column Null Values: 0
ORCL Column Null Values: 0
AAPL Column Null Values: 0
```

- **Explanation**: While similar to missing data, the verification of any null values contained within the DataFrame has a fundimental difference. Missing data refers to no rows existing for a specific index value for a given column within a DataFrame. A null value is when the row exists at the specific index, but is assigned with a balue that is defined as nothing.
- **Justification**: It is of great important to ensure there are now values associated with the provided tech companies stock closing price that are defined as nothing to prevent any skewing of the analysis performed for the remaining, available datapoints.
- **Advantage**: In comparison to missing data, an advantage of a column having a null value for a given index is that no rows are required to be added for the specific varaible.

- **Disadvantage**: Similar to missing data, there is often no clear indication as to why a value would reflect as null. This can lead to uncertainty regarding the most appropriate way to handle correction and replacement of the null value.

Create a single target variable for Stock Closing Price

As this analysis is is an univariate time series, there is a single time-dependent target variable. In this analysis, the target variable is the stock closing price.

- **Explanation**: A univariate time series includes a single time-dependent target vairable.
- **Justification**: The analysis being performed is to predict the stock closing price for a set of tech companies. As a result, the prediction is to forecast a single variable which fits the definition of a univeriate time series.
- **Advantage**: A primary advantage of univariate time series is they allow researchers to focus all of their attention on one specific aspect of research.
- **Disadvantage**: A noted disadvantage of a univariate time series is a comparitvely long period of time must be collected for the sample data. This is nessesitated by the need for reliable identification of d and the ARMA co-efficient of AdX.

Tools & Techniques

The tools and techniques used for this analysis are as follows:

- **Jupyter Notebooks**: The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Its uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more(K, 2020).
- **Python**: Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Tool and Technique Justification

The justification for the utilized tools and techniques are as follows:

- **Jupyter Notebooks**: As is outlined below in the advantages section of this report, Jupyter Notebooks is an excellent tool to use when having the results of executed code appear directly with the code itself is of benefit to the user.
- **Python**: Due to Python's support for an extensive set of libraries that allow easy exploration of data, the libraries facilitating time-series model creation, and my general familiarity with the language across a wide spectrum of uses, Python was determined to be the optimal programming language utilized for this analysis

Tools and Techniques Advantages

The advantages of the utilized tools and techniques are as follows:

- Jupyter Notebooks is great for showcasing your work/analysis. This is a result of both the code and results easily within the same cell of the notebook.
- The Python programming language is easy to read, learn, and write in comparison to many other programming languages.
- Python is a free and open-sourced language.
- Python has a vast, extensive set of libraries to faciliate nearly any objective a programmer/data analyst would be pursuing.

Tools and Techniques Disadvantages

The disadvantages of the utilized tools and techniques are as follows:

- When creating code in Jupyter Notebooks, it is very easy to end of with duplicate code rather than the standard creation of functions,classes, and objects. This can become difficult to maintain as your notebook grows in size.
- As Python is an interpreted language, it can be slow in comparison to languages like C/C++ or Java.
- Due to Python being a dynamically typed language, it can often lead to run-time errors and require more testing when compared to other programming languages.

Analysis

D. Report on your data-analysis process by describing the analysis technique(s) you used to appropriately analyze the data. Include the calculations you performed and their outputs. Justify how you selected the analysis technique(s) you used, including one advantage and one disadvantage of these technique(s).

Description of Analysis Techniques

Over the course of the performed analysis, several techniques were utilized. These techniques include:

- **Exploratory Data Analysis**
- **Statistical Testing**
- **Creation of Time Series Model**

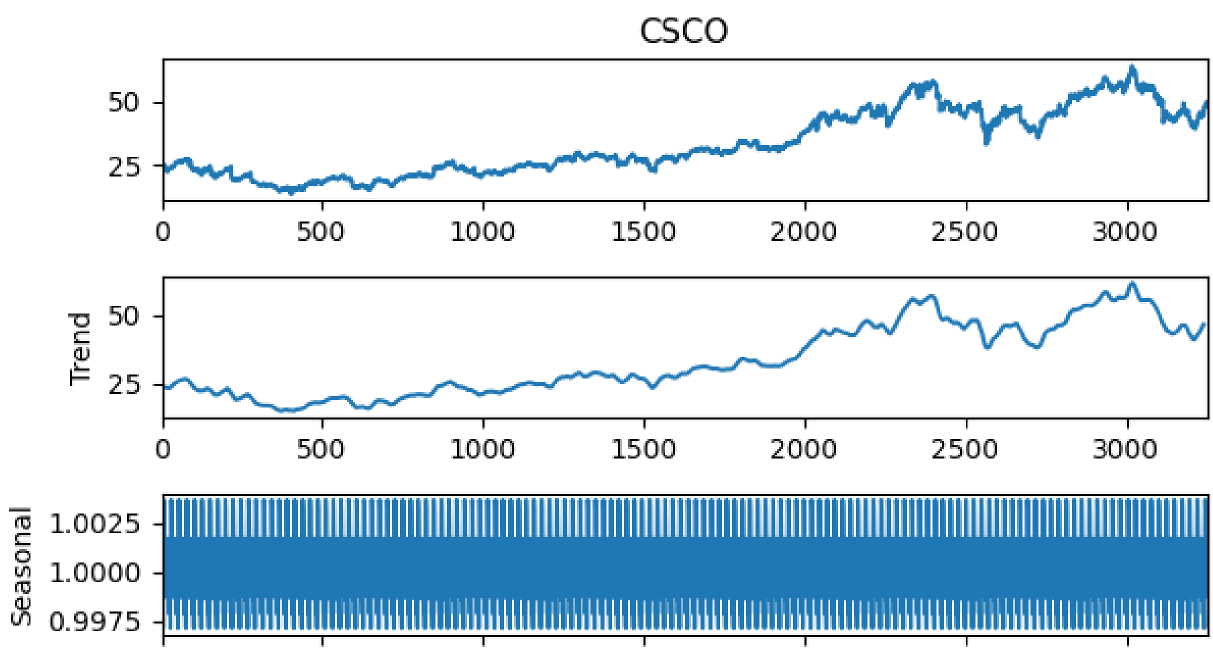
Each of the three analysis techniques used will be further overviewed below.

- **Exploratory Data Analysis:** EDA refers to the critical process of performing initial investigations on data in an attempt at discovering patterns, spotting anomalies, testing hypothesis, and checking assumptions with the help of summary statistics and graphical representations(Patil, 2022).
- **Statistical Testing:** Statistic hypothesis testing is a method of statistical inference used to decide whether the data at hand sufficiently supports a particular hypothesis. Hypothesis testing allows the creation of probabilistic statements about population parameters(Wikipedia, 2023).

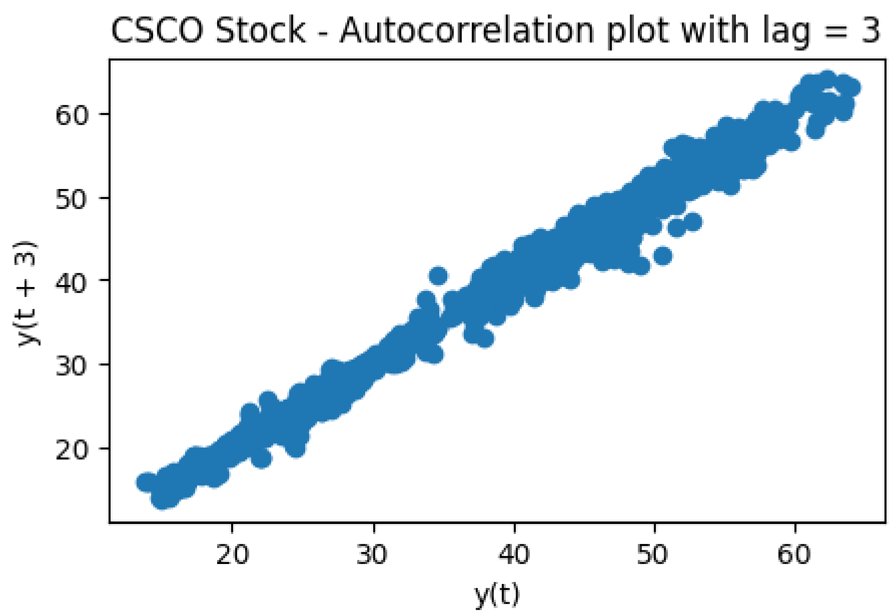
- **Creation of Time Series Model:** A Time Series is a collection of data points that are stored with respect to their time. Mathematical and statistical analysis is performed on this type of data to find hidden patterns and meaningful insights, which is refered to as *time-series analysis*. Time-series modeling techniques are used to understand past patterns from the collected data to try and forecast future horizens(Vishwas & Patel, 2020).

Exploratory Data Analysis

In [24]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
columns = {'AAPL', 'ADBE', 'ORCL', 'AMZN', 'INTC', 'MSFT', 'NVDA', 'IBM',
        'NFLX', 'GOOGL', 'CRM', 'CSCO'}

for column in columns:
    column_name = df[column].name
    result = seasonal_decompose(df[column], model='multiplicative', period=
    result.plot()
```



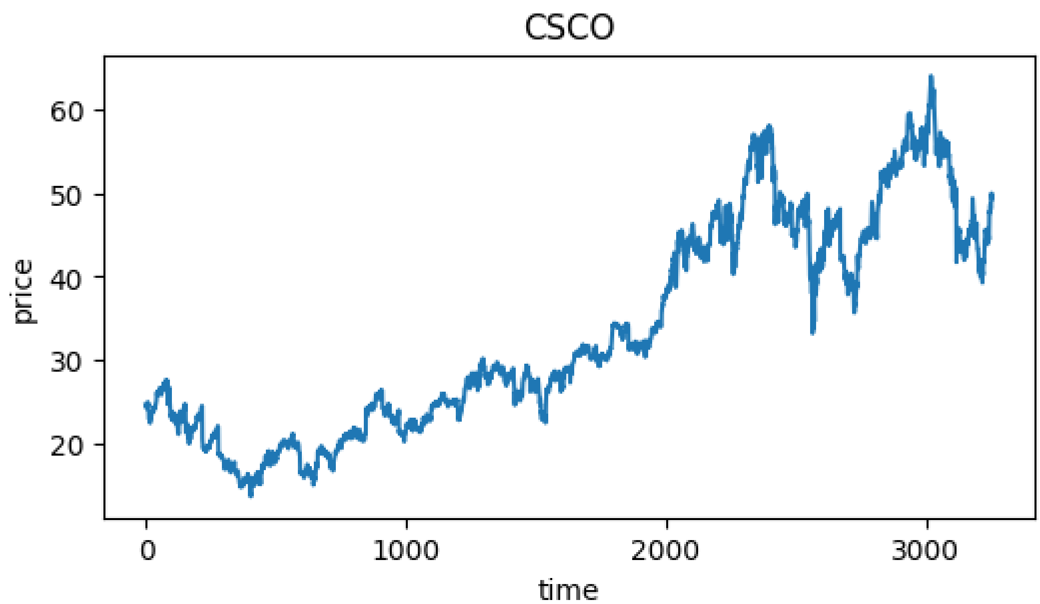
In [25]:
```
columns = {'AAPL', 'ADBE', 'ORCL', 'AMZN', 'INTC', 'MSFT', 'NVDA', 'IBM',
        'NFLX', 'GOOGL', 'CRM', 'CSCO'}

for column in columns:
    column_name = df[column].name
    plt.figure(figsize=(5,3))
    lag_plot(df[column_name], lag=3)
    plt.title(str(column_name)+' Stock - Autocorrelation plot with lag = 3'
    plt.show()
```



In [26]:
```
df.columns
```

Out[26]:
```
Index(['Date', 'AAPL', 'ADBE', 'ORCL', 'AMZN', 'INTC', 'MSFT', 'NVDA', 'I
BM',
       'NFLX', 'GOOGL', 'CRM', 'CSCO'],
      dtype='object')
```

In [27]:
```
columns = {'AAPL', 'ADBE', 'ORCL', 'AMZN', 'INTC', 'MSFT', 'NVDA', 'IBM',
       'NFLX', 'GOOGL', 'CRM', 'CSCO'}

for column in columns:
    column_name = df[column].name
    plt.figure(figsize=(6,3))
    plt.plot(df.index, df[column_name])
    plt.xticks(np.arange(0,3254, 1000), df.index[0:3254:1000])
    plt.title(column_name)
    plt.xlabel("time")
    plt.ylabel("price")
    plt.show()
```



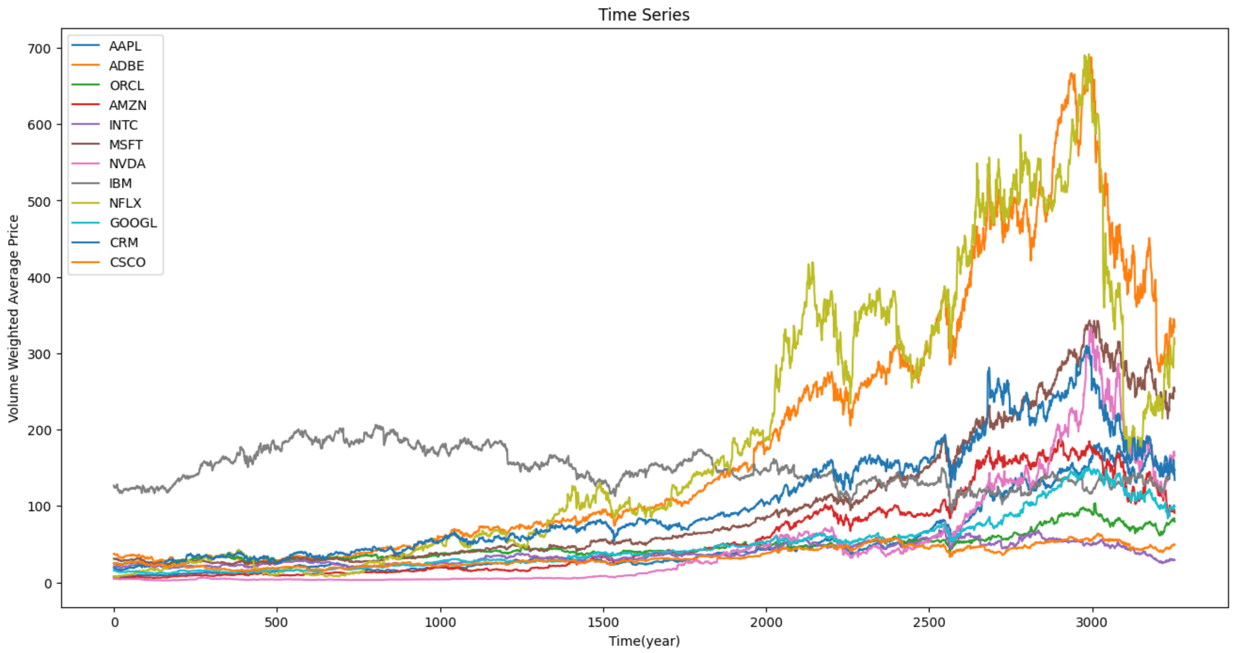
Statistical Testing

In [28]:
```
# Earliest date index location
print('Earliest date index location is: ',df.index.argmin())

# Latest date location
print('Latest date location: ',df.index.argmax())
```

```
Earliest date index location is:  0
Latest date location:  3253
```

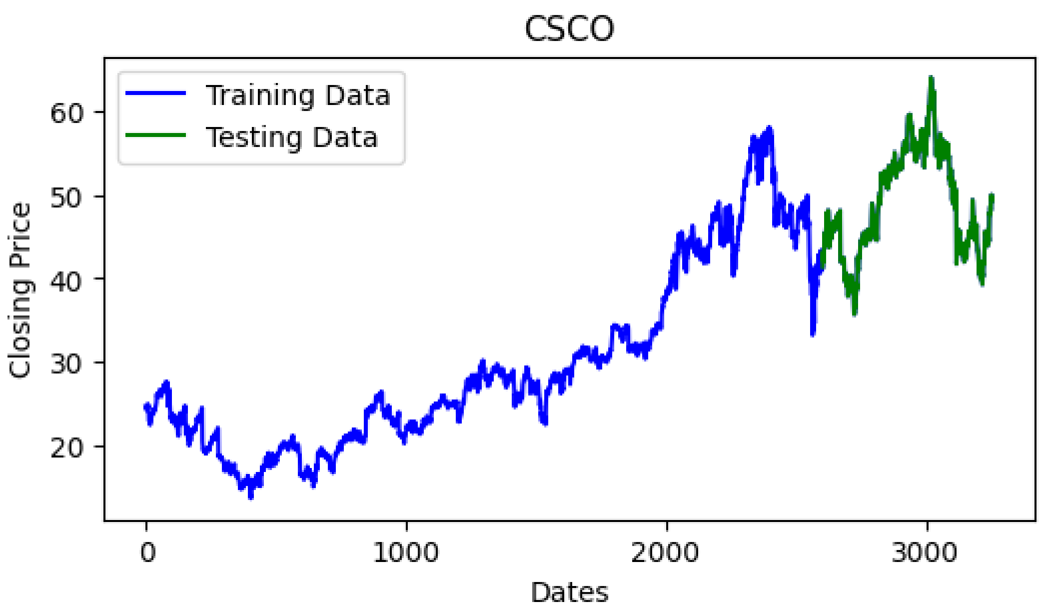
In [29]:
```
# Visualising the VWAP

plt.figure(figsize=(16,8))
plt.plot(df[['AAPL', 'ADBE', 'ORCL', 'AMZN', 'INTC', 'MSFT', 'NVDA', 'IBM',
    'GOOGL', 'CRM', 'CSCO']], label=['AAPL', 'ADBE', 'ORCL', 'AMZN', 'INTC'
        'GOOGL', 'CRM', 'CSCO'])
plt.title('Time Series')
plt.xlabel("Time(year)")
plt.ylabel("Volume Weighted Average Price")
plt.legend(loc='best')
```

Out[29]: <matplotlib.legend.Legend at 0x1684ba160>



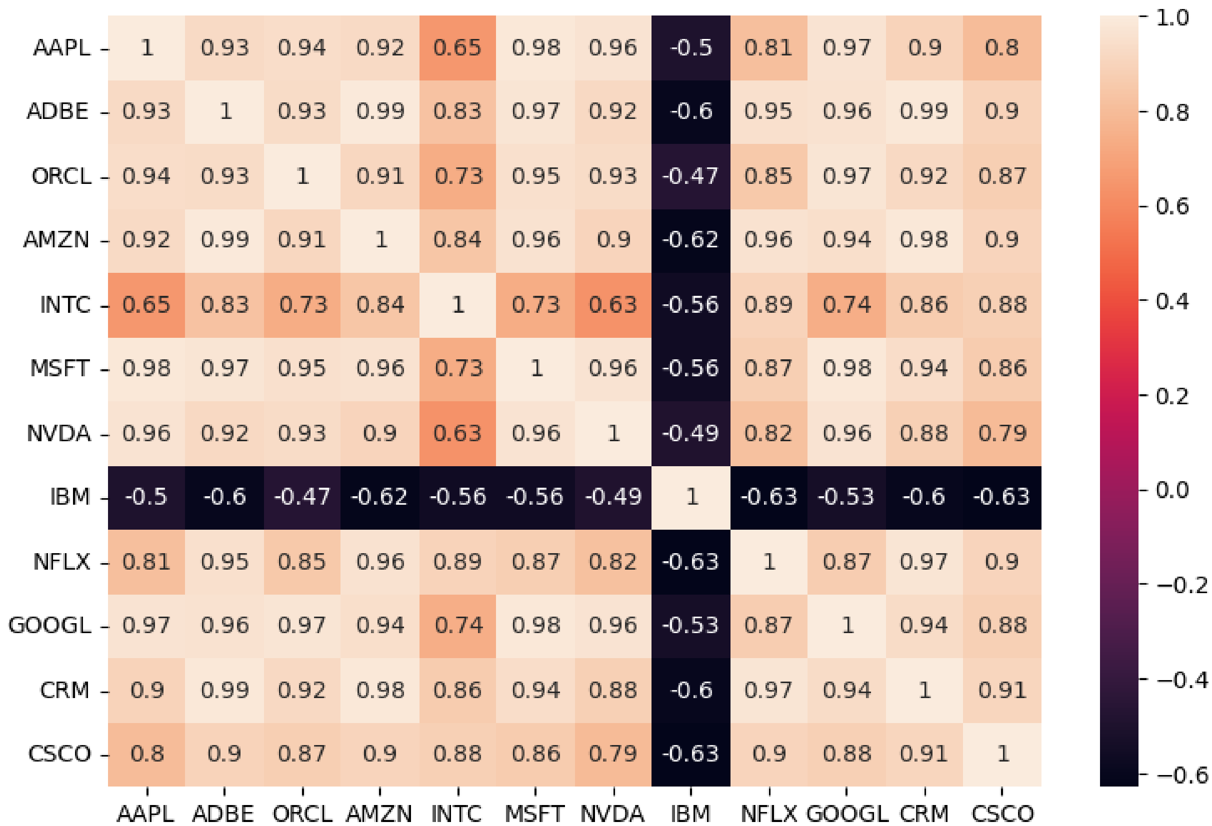
In [30]:
```
AAPL_close = AAPL['Close']
```

In [31]:
```
columns = {'AAPL', 'ADBE', 'ORCL', 'AMZN', 'INTC', 'MSFT', 'NVDA', 'IBM',
        'NFLX', 'GOOGL', 'CRM', 'CSCO'}

for column in columns:
    column_name = df[column].name

    train_data, test_data = df[column_name][0:int(len(df[column_name])*0.8)
    plt.figure(figsize=(6,3))
    plt.title(column_name)
    plt.xlabel('Dates')
    plt.ylabel('Closing Price')
    plt.plot(df[column_name], 'blue', label='Training Data')
    plt.plot(test_data[column_name], 'green', label='Testing Data')
    plt.xticks(np.arange(0,3254, 1000), df.index[0:3254:1000])
    plt.legend()
```



In [32]:
```
import seaborn as sns

corr = df.corr()
plt.figure(figsize=(9,6))
sns.heatmap(corr, annot=True)
```

Out[32]: <AxesSubplot: >



Creation of Time Series Model
The Time Series model used for this analysis was an ARIMA model. ARIMA, short for 'Auto Regressive Integrated Moving Average' is actually a class of models that 'explains' a given time series based on its own past values. This is useful in forecasting future values. Any 'non-seasonal' time series that exhibits patterns and is not a random white noise can be modeled by ARIMA models. An ARIMA model is characterized by 3 terms: p, d, and q. p is the order of the AR term, q is the order of the MA term, and d is the number of differencing required to make the time series stationary. To provide additional context, **p** is the order of the 'Auto Regressive term. This refers to the number of lags of Y to be used as predictors. **q** is the order of the 'Moving Average (MA). This refers to the number of lagged forecast errors that should go into the ARIMA model. **d** is the minimum number of differencing needed to make the series stationary. If the series is already stationary, then d = 0. When it comes to making the appropriate decision for the level of decisioning to be completed, the correct differencing level is whatever is required to get a near stationary series which roams around the defined mean and the ACF plot reaches zero quickly.

The first step in appropriatly determining if **d** should be set as zero is to test if the series is stationary by performing the Augmented Dickey Fuller Test. If the calculated p-value is less than the significance level (0.05), it can be infered that the time series is stationary. If the p-value is greater than 0.05, the apprioriate order of differencing will need to be determined.

In [ ]:

```

```

In [ ]:

```

```

Utilizing auto ARIMA by conducting differencing tests to determine the order of differencing

This step also includes creating and fitting ARIMA model

In [33]:
```
aa = pd.DataFrame()
aa.index = ['p', 'q', 'd']

future_forecast = pd.DataFrame()

columns = {'AAPL', 'ADBE', 'ORCL', 'AMZN', 'INTC', 'MSFT', 'NVDA', 'IBM',
       'NFLX', 'GOOGL', 'CRM', 'CSCO'}

for column in columns:
    column_name = df[column].name

    train_data, test_data = df[column_name][0:int(len(df[column_name])*0.8)
    print("****************************************************************
    print("AUTO ARIMA FOR ", column_name)
    print("****************************************************************
    model_autoARIMA = auto_arima(train_data, start_p=0, start_q=0,
                          test='adf',       # use adftest to find optimal '
                          max_p=3, max_q=3, # maximum p and q
                          m=12,              # frequency of series
                          d=None,           # let model determine 'd'
                          seasonal=False,   # No Seasonality
                          start_P=0,
                          D=0,
                          trace=True,
                          error_action='ignore',
                          suppress_warnings=True,
                          stepwise=True)
    print(model_autoARIMA.summary())
    aa[column_name] = model_autoARIMA.get_params().get("order")
    model_autoARIMA.plot_diagnostics(figsize=(15,8))
    plt.show()

    model_autoARIMA.fit(train_data)


    future_forecast[column_name] = model_autoARIMA.predict(n_periods=37)
```

```
*****************************************************************
AUTO ARIMA FOR  CSCO
*****************************************************************
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=4059.967, Time=0.16 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=4032.308, Time=0.10 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=4033.211, Time=0.09 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=4058.341, Time=0.02 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=4034.191, Time=0.07 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=4034.242, Time=0.09 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=4035.730, Time=0.74 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=4030.774, Time=0.03 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=4032.650, Time=0.03 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=4032.704, Time=0.03 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=4031.684, Time=0.03 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=4034.172, Time=0.19 sec

Best model:  ARIMA(1,1,0)(0,0,0)[0]
Total fit time: 1.612 seconds
```

Auto ARIMA model has saved the p, q, and d for each company in a new dataframe named aa

In [34]:

```
aa
```

Out[34]:

	CSCO	INTC	NVDA	NFLX	ADBE	MSFT	GOOGL	CRM	AMZN	IBM	ORCL	AAPL
p	1	3	3	3	2	1	3	2	1	0	3	3
q	1	0	1	1	1	1	0	1	1	1	0	1
d	0	3	1	0	2	0	2	2	0	0	3	1

In [35]:
```
for column in columns:
    column_name = df[column].name
    print("FUTURE FORECAST FOR: " ,column_name)
    print("---------------------------------------------------------------"
    print(future_forecast[column_name].head())
    print("---------------------------------------------------------------"
```

```
FUTURE FORECAST FOR:  CSCO
----------------------------------------------------------------
2603    41.182958
2604    41.179455
2605    41.179827
2606    41.179788
2607    41.179792
Name: CSCO, dtype: float64
----------------------------------------------------------------
FUTURE FORECAST FOR:  INTC
----------------------------------------------------------------
2603    58.670904
2604    59.348273
2605    58.551912
2606    59.321573
2607    58.670772
Name: INTC, dtype: float64
----------------------------------------------------------------
FUTURE FORECAST FOR:  NVDA
----------------------------------------------------------------
2603    74.590770
2604    74.590931
2605    74.796415
2606    74.670403
2607    74.838609
Name: NVDA, dtype: float64
----------------------------------------------------------------
FUTURE FORECAST FOR:  NFLX
----------------------------------------------------------------
2603    434.305820
2604    434.880583
2605    435.553008
2606    435.704063
2607    435.926002
Name: NFLX, dtype: float64
----------------------------------------------------------------
FUTURE FORECAST FOR:  ADBE
----------------------------------------------------------------
2603    361.201700
2604    361.764155
2605    362.072114
2606    361.695601
2607    362.424745
Name: ADBE, dtype: float64
----------------------------------------------------------------
FUTURE FORECAST FOR:  MSFT
----------------------------------------------------------------
2603    182.031702
2604    182.276111
2605    182.272930
2606    182.351191
2607    182.402662
Name: MSFT, dtype: float64
----------------------------------------------------------------
FUTURE FORECAST FOR:  GOOGL
----------------------------------------------------------------
2603    66.431167
```

```
2604    67.387434
2605    66.453869
2606    67.194550
2607    66.735955
Name: GOOGL, dtype: float64
----------------------------------------------------------------
FUTURE FORECAST FOR:  CRM
----------------------------------------------------------------
2603    162.403271
2604    163.827982
2605    162.690526
2606    163.583262
2607    163.267001
Name: CRM, dtype: float64
----------------------------------------------------------------
FUTURE FORECAST FOR:  AMZN
----------------------------------------------------------------
2603    117.478575
2604    117.531027
2605    117.572818
2606    117.615439
2607    117.657995
Name: AMZN, dtype: float64
----------------------------------------------------------------
FUTURE FORECAST FOR:  IBM
----------------------------------------------------------------
2603    117.753349
2604    117.753349
2605    117.753349
2606    117.753349
2607    117.753349
Name: IBM, dtype: float64
----------------------------------------------------------------
FUTURE FORECAST FOR:  ORCL
----------------------------------------------------------------
2603    51.675036
2604    51.966354
2605    51.626184
2606    51.949193
2607    51.678781
Name: ORCL, dtype: float64
----------------------------------------------------------------
FUTURE FORECAST FOR:  AAPL
----------------------------------------------------------------
2603    75.318074
2604    75.202185
2605    75.424679
2606    75.272904
2607    75.449438
Name: AAPL, dtype: float64
----------------------------------------------------------------
```

In [36]:
```
future_forecast.to_csv('Datasets2/future_forecast.csv')
```

In [57]:
```
columns = ["AAPL", "ADBE" ,"ORCL", "AMZN","INTC", "MSFT", "NVDA",
           "IBM","NFLX", "GOOGL","CRM", "CSCO"]
difference_df = pd.DataFrame()
accuracy_df = pd.DataFrame()
forecast_df = pd.DataFrame()
actual_df = pd.DataFrame()

# actual_df.
for column in columns:
    j = 1
    difference_series = []
    accuracy_series = []
    forecast_series = []
    actual_series = []
    column_name = df[column].name

    for i in range(2603, 2639 ):
        print("Day Number ", i, " for stock ", column_name)
        sample_forecast = future_forecast[column_name].loc[i]
        forecast_series.append(sample_forecast)
        print("Sample Forecast: ", sample_forecast)
        sample_actual = df[column_name].loc[i]
        actual_series.append(sample_actual)
        print("Sample Actual: ", sample_actual)
        sample_difference = sample_forecast - sample_actual
        difference_series.append(sample_difference)
        print("Sample Difference: ", sample_difference)
        if sample_forecast > sample_actual:
            sample_accuracy = sample_actual / sample_forecast
        else:
            sample_accuracy = sample_forecast / sample_actual
        print("Sample Accuracy: ", sample_accuracy)
        print("----------------------------------------------------")
        sample_accuracy = sample_accuracy
        sample_accuracy_total = (sample_accuracy + accuracy_series) / j
        accuracy_series.append(sample_accuracy)
        print("j is currently ", j)
        j = j+1
        print("sample accuracy is currently", sample_accuracy)
    forecast_df[column_name] = pd.DataFrame(forecast_series)
    actual_df[column_name] = pd.DataFrame(actual_series)
    difference_df[column_name] = pd.DataFrame(difference_series)
    accuracy_df[column_name] = pd.DataFrame(accuracy_series)
```

```
Day Number  2603  for stock  AAPL
Sample Forecast:  75.3180740808222
Sample Actual:  75.934998
Sample Difference:  -0.6169239191777933
Sample Accuracy:  0.9918756313238094
----------------------------------------------------
j is currently  1
sample accuracy is currently 0.9918756313238094
Day Number  2604  for stock  AAPL
Sample Forecast:  75.20218491074093
Sample Actual:  77.532501
Sample Difference:  -2.330316089259071
Sample Accuracy:  0.9699440098126195
----------------------------------------------------
j is currently  2
sample accuracy is currently 0.9699440098126195
Day Number  2605  for stock  AAPL
Sample Forecast:  75.424679497523
Sample Actual:  78.752502
```

In [58]:

```python
accuracy_df.to_csv('Datasets2/accuracy_df.csv')
forecast_df.to_csv('Datasets2/forecase_df.csv')
difference_df.to_csv('Datasets2/difference_df.csv')
actual_df.to_csv('Datasets2/actual_df.csv')
```

In [59]:

```python
accuracy_df.head()
```

Out[59]:

	AAPL	ADBE	ORCL	AMZN	INTC	MSFT	NVDA	IBM	NFLX	GOOG
0	0.991876	0.984791	0.982415	0.992381	0.991565	0.991458	0.978657	0.984249	0.994905	0.9703
1	0.969944	0.984365	0.970064	0.987818	0.994608	0.986984	0.954764	0.998539	0.998463	0.9735
2	0.957743	0.974832	0.964435	0.976113	0.973755	0.976079	0.927362	0.995291	0.988725	0.9469
3	0.966866	0.990675	0.993863	0.998031	0.984296	0.999130	0.957006	0.976374	0.991086	0.9772
4	0.980978	0.989336	0.999443	0.993767	0.984136	0.985457	0.961936	0.939596	0.994652	0.9899

In [60]:

```python
accuracy_df.describe()
```

Out[60]:

	AAPL	ADBE	ORCL	AMZN	INTC	MSFT	NVDA	IBM	
count	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.0
mean	0.919064	0.926793	0.969776	0.940849	0.965183	0.972629	0.857652	0.970150	0.9
std	0.048188	0.050072	0.017657	0.041799	0.025637	0.027265	0.048792	0.021909	0.0
min	0.830597	0.829889	0.936223	0.859856	0.912228	0.911451	0.792555	0.907330	0.9
25%	0.868623	0.893601	0.955400	0.905136	0.945385	0.948477	0.824831	0.953690	0.9
50%	0.938518	0.936710	0.972397	0.955260	0.975255	0.985471	0.850935	0.977079	0.9
75%	0.955463	0.969809	0.981923	0.974862	0.984834	0.993145	0.865174	0.987111	0.9
max	0.991876	0.992536	0.999443	0.998031	0.999778	0.999130	0.978657	0.998539	0.9

In [41]:
```
forecast_df.tail()
```

Out[41]:

	AAPL	ADBE	ORCL	AMZN	INTC	MSFT	NVDA	IBM	
31	76.083710	365.483195	51.731144	118.807141	58.858347	183.973061	75.504719	117.753349	44
32	76.109703	365.607705	51.670144	118.849702	58.687515	184.031163	75.531631	117.753349	44
33	76.135677	365.732473	51.747939	118.892263	58.929542	184.089265	75.558539	117.753349	44
34	76.161664	365.856990	51.655012	118.934824	58.636280	184.147368	75.585449	117.753349	44
35	76.187642	365.981718	51.736721	118.977385	58.913884	184.205470	75.612358	117.753349	44

In [42]:
```
actual_df.tail()
```

Out[42]:

	AAPL	ADBE	ORCL	AMZN	INTC	MSFT	NVDA	IBM	
31	89.717499	438.640015	55.119999	135.690994	60.090000	200.570007	95.267502	115.745697	46
32	91.632500	440.549988	55.189999	138.220505	59.919998	201.910004	94.500000	114.158699	46
33	90.014999	431.679993	54.439999	136.720001	59.090000	197.839996	92.355003	111.300194	45
34	91.209999	436.950012	54.529999	137.729004	58.509998	200.339996	94.900002	113.795410	46
35	88.407501	426.920013	54.180000	134.643494	57.500000	196.330002	91.550003	112.036331	44

In [43]:
```
difference_df.head()
```

Out[43]:

	AAPL	ADBE	ORCL	AMZN	INTC	MSFT	NVDA	IBM	NFLX	
0	-0.616924	-5.578299	-0.924962	-0.901926	-0.499094	-1.568304	-1.626729	1.854690	-2.224179	-
1	-2.330316	-5.745855	-1.603646	-1.449472	-0.321725	-2.403882	-3.534069	0.172088	-0.669405	-
2	-3.327823	-9.347899	-1.903815	-2.877179	-1.578089	-4.467075	-5.858584	0.554496	-4.966981	-
3	-2.579597	-3.404405	-0.320807	-0.232065	0.931574	-0.158804	-3.354599	2.782028	3.884056	-
4	-1.463060	3.864747	0.028779	-0.738009	0.930770	2.652662	-2.961394	7.112816	-2.343987	-

Calculations Performed

The calculations performed for this analysis are as follows:

Analysis Technique Justification

The three techniques overviewed above all play an essential role in ensuring the performed data analysis is as reliable as possible.
The **exploratory data analysis** step is a technique used to review the existing data allowing better familiarity with what is contained within the data.

The **statistical testing** stage is used to provide a mechanism for making quantitative decisions about a process or processes. The intent is to determine if enough evidence exists to "reject" a conjecture or hypothesis about a given process. The conjecture is also commonly known as the null hypothesis.

Analysis Technique Advantages

The advantages for each of the three techniques used will be outlined below:

- **Exploratory Data Analysis:** The advantages of the EDA stage are that it can help identify obvious errors within the data, facilitate better understanding of patterns within the data, assist with detecting outliers and/or anomalous events, and find interesting relationships among variables.
- **Statistical Testing:** The advantages of the statistical testing stage include providing a mechanism for making quantitative decisions regarding a process of processes.
- **Creation of Time Series Model:** Some of the advantages of the Time Series Modeling include facilitating a better understanding of underlying causes of trends or systematic patterns over time. This includes identification of seasonal trends and an understanding of why any such trends occur.

Analysis Technique Disdvantages

The disadvantages for each of the three techniques used will be outlined below:

- **Exploratory Data Analysis:** Some disadvantages of the EDA stage include providing inconclusive results, a lack of standardized analysis, a small sample population, and/or outdated information tht can adversely affect the authenticity of information.
- **Statistical Testing:** ThAe disadvantages of the statistical testing stage includes that it can be easy to misuse the tests themselves. If the tests used are not carefully constructed, the results can be skewed incorrectly.
- **Creation of Time Series Model:** Some of the disadvantages of the Time Series Modeling include issues with generalization from a single study, difficulty in obtaining appropriate measures, and problems with identifying the correct model to represent the data.

Data Summary and Implications

E. Summarize the implications of your data analysis by discussing the results of your data analysis in the context of the research question, including one limitation of your analysis. Within the context of your research question, recommend a course of action based on your results. Then propose two directions or approaches for future study of the data set.

Data Analysis Implications

The implications of the performed analysis is that future stock performance can be reasonably predicted with a relativly high accuracy. Utilizing the forecasted stock prices, an investor can gain a level of comfort with how the stock is estimated to perform based upon its past performance.

Data Analysis Results

The results of the performed analysis indicate that a daily stock price can be predicted with an accuracy generally above 85%.

Data Analysis Limitations

The limitations of the performed analysis were that the predicted estimates were consistantly lower than the stocks actual market price for a given day. In this case, it is beneficial to error on the side of caution and have the prediction be below an actual price as opposed to above, it warrants more fine tuning to the model to determine if performance can be fine tuned to more closely align with the actual price on a given day.

Recommended Course of Action

The recommended course of action based upon the completed analysis is that utilizing the developed model can assist with investment decisions if a company is seeking insight into an expected company's market performance.

However, as stated in the previous section, it would be prudent to attempt fine-tuning the model to correct the predictions consistantly being below the actual price. If a companys market value significantly increases for a given period of time, the model may lag behind in reflecting accurate predictions at the new market value. This could cause an investor to underestimate any gains from their investment.

Future Data Study Directions

1. It would be prudent to facilitate the model being able to regularly update the historical data to allow predictions to continue being made for trading days in the future based upon historical data that is not yet currently available.
2. Improving the model to collect data for additional companies would allow predictions to be generated for any company with historical data available for collection.

F. Sources.

- Creative Commons License Deed. Creative Commons - Attribution-NonCommercial-ShareAlike 4.0 International - CC BY-NC-SA 4.0. (n.d.). Retrieved February 23, 2023, fromhttps://creativecommons.org/licenses/by-nc-sa/4.0/

- Gower, E. (2023, January 30). Big Tech Stock prices. Kaggle. Retrieved February 23, 2023, fromhttps://www.kaggle.com/datasets/evangower/big-tech-stock-prices
- Time series analysis: Definition, types, techniques, and when it's used. Tableau. (n.d.). Retrieved February 23, 2023, from https://www.tableau.com/learn/articles/time-series-analysis#definition
- Li, S. (2018, September 5). An end-to-end project on time series analysis and forecasting with python. Medium. Retrieved February 23, 2023, from https://towardsdatascience.com/an-end-to-end-project-on-time-series-analysis-and-forecasting-with-python-4835e6bf050b
- Stephen Allwright. (2022, December 6). What is a good MAPE score? (simply explained). Stephen Allwright. Retrieved February 24, 2023, from https://stephenallwright.com/good-mape-score/
- Patil, P. (2022, May 30). What is exploratory data analysis? Medium. Retrieved March 10, 2023, from https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15
- Wikimedia Foundation. (2023, March 3). Statistical hypothesis testing. Wikipedia. Retrieved March 10, 2023, from https://en.wikipedia.org/wiki/Statistical_hypothesis_testing
- Vishwas, B. V., & Patel, A. (2020). Hands-on time series analysis with python: From basics to bleeding edge techniques. Apress.
- K, B. (2020, December 22). Everything you need to know about Jupyter notebooks! Medium. Retrieved March 10, 2023, from https://towardsdatascience.com/everything-you-need-to-know-about-jupyter-notebooks-10770719952b
- Python - overview. Tutorials Point. (n.d.). Retrieved March 10, 2023, from https://www.tutorialspoint.com/python/python_overview.htm

In [ ]:

In [ ]: