COS30018 – Task B.3 - Data Processing 2 Report

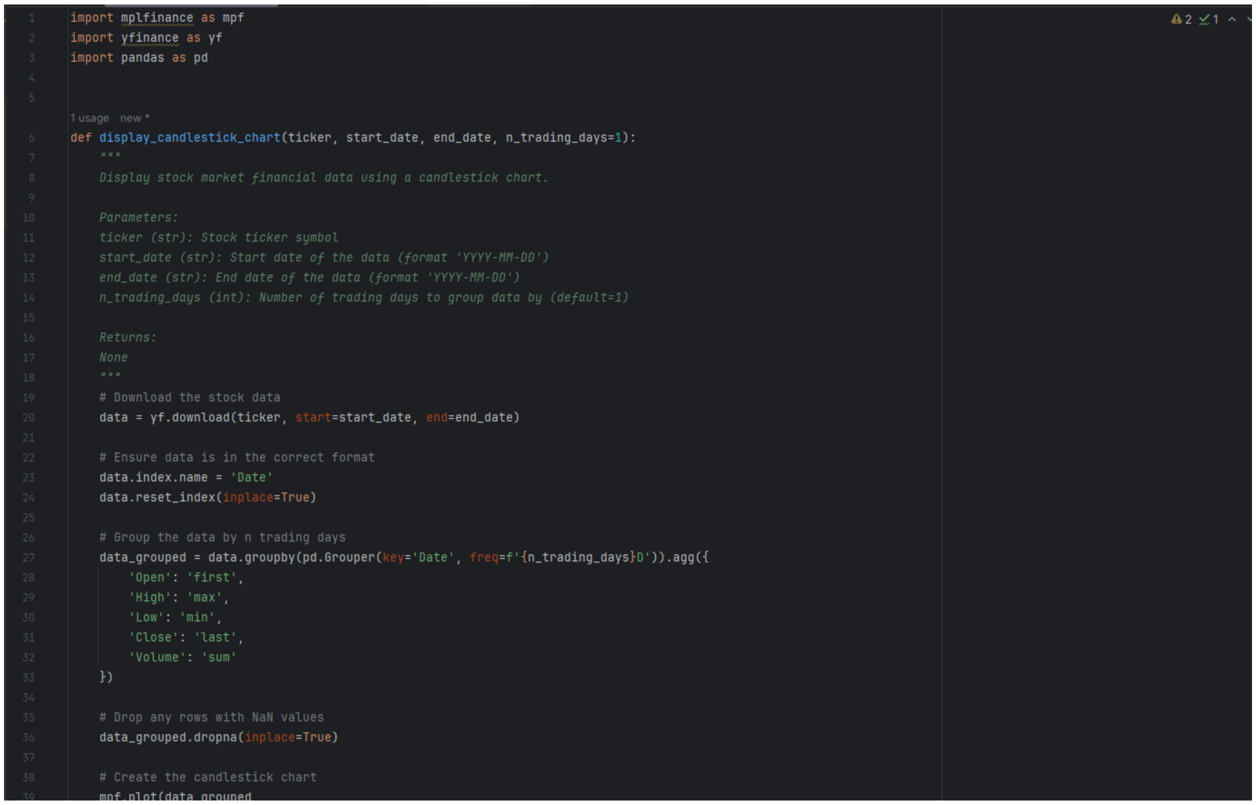
Name: Abhinav Karn

Student ID: 104488053

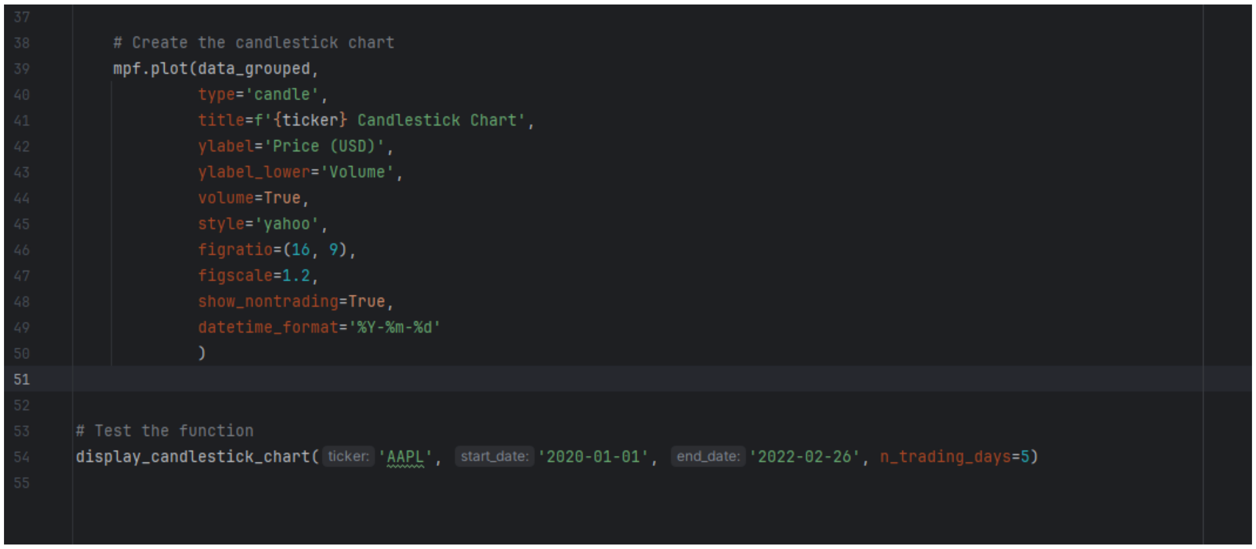
Date: 1st September, 2024

Code:

Candlestick_chart.py:



```python
import mplfinance as mpf
import yfinance as yf
import pandas as pd


def display_candlestick_chart(ticker, start_date, end_date, n_trading_days=1):
    """
    Display stock market financial data using a candlestick chart.

    Parameters:
    ticker (str): Stock ticker symbol
    start_date (str): Start date of the data (format 'YYYY-MM-DD')
    end_date (str): End date of the data (format 'YYYY-MM-DD')
    n_trading_days (int): Number of trading days to group data by (default=1)

    Returns:
    None
    """
    # Download the stock data
    data = yf.download(ticker, start=start_date, end=end_date)

    # Ensure data is in the correct format
    data.index.name = 'Date'
    data.reset_index(inplace=True)

    # Group the data by n trading days
    data_grouped = data.groupby(pd.Grouper(key='Date', freq=f'{n_trading_days}D')).agg({
        'Open': 'first',
        'High': 'max',
        'Low': 'min',
        'Close': 'last',
        'Volume': 'sum'
    })

    # Drop any rows with NaN values
    data_grouped.dropna(inplace=True)

    # Create the candlestick chart
    mpf.plot(data_grouped
```



```
    # Create the candlestick chart
    mpf.plot(data_grouped,
             type='candle',
             title=f'{ticker} Candlestick Chart',
             ylabel='Price (USD)',
             ylabel_lower='Volume',
             volume=True,
             style='yahoo',
             figratio=(16, 9),
             figscale=1.2,
             show_nontrading=True,
             datetime_format='%Y-%m-%d'
             )


# Test the function
display_candlestick_chart( ticker: 'AAPL',  start_date: '2020-01-01',  end_date: '2022-02-26', n_trading_days=5)
```

Imports:

mplfinance: A package used for financial plotting such as candlestick charts.
yfinance: You can download historical stock market data from Yahoo Finance using this library.
pandas: Used for manipulating and analyzing the data.

Function Definition:

display_candlestick_chart: It is defined to plot a candlestick chart of some stock market data.
Parameters:
ticker: This is a stock symbol you want to fetch data for, such as 'AAPL' for Apple.
start_date: This is the starting date for which data should be fetched in 'YYYY-MM-DD' format.
end_date: This is the ending date for which data needs to be fetched in 'YYYY-MM-DD' format.
n_trading_days: It gives the number of days in which data has to be aggregated for trading. Default value is 1 day and hence it will plot daily data.

Download Stock Data:

It then uses yfinance to download the historical stock data between the defined start and end dates.

Format Data:

The downloaded data is processed to ensure that the column date is named 'Date' and used for grouping. It calls the function reset_index() to make 'Date' a regular column.
Group Data by Trading Days:

The data is binned into as many bins as there are trading days specified, so if n_trading_days is 5, for example, the data will be aggregated in 5-day intervals.
Aggregation functions are used to calculate:
Open: opening price on first day of each period
High: highest price of the period
Low: lowest price of the period
Close: closing price on last day of period
Volume: total trading volume over the period.
Drop NaN Values:

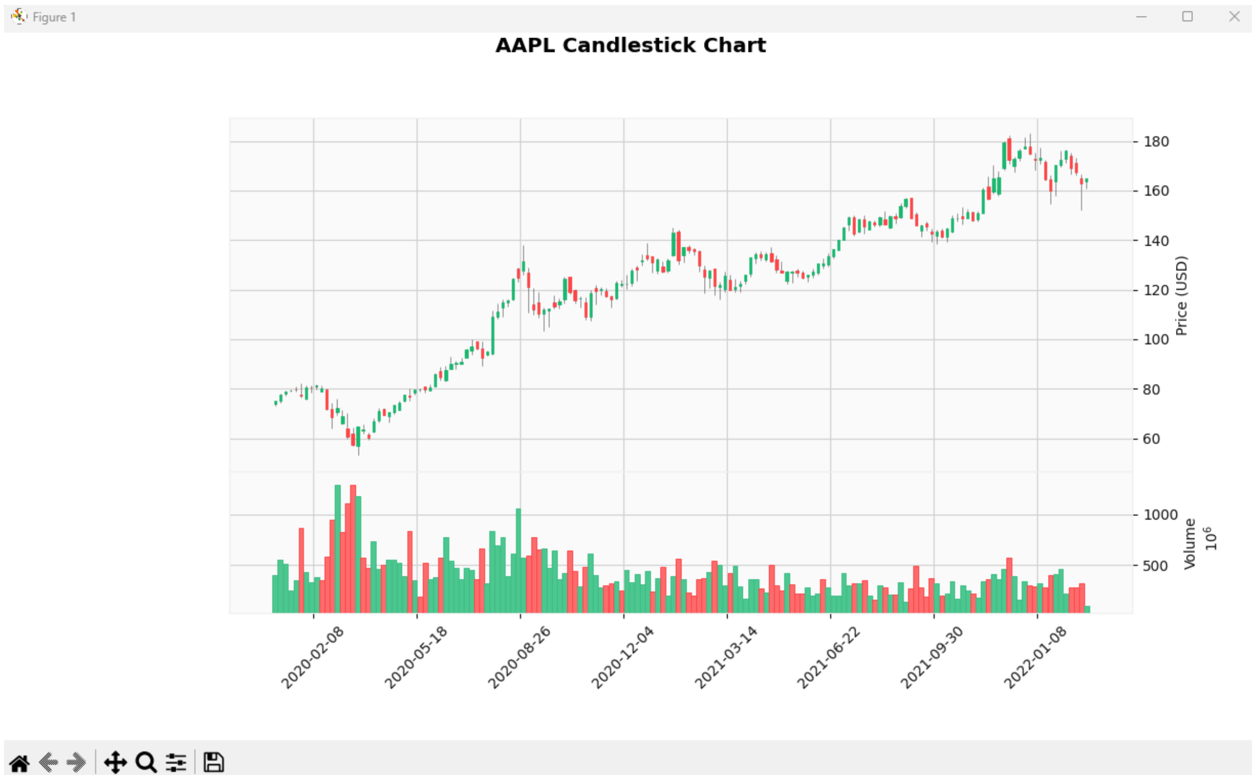
To make the chart return proper data, we drop any row with a missing value.

Create Candlestick Chart:

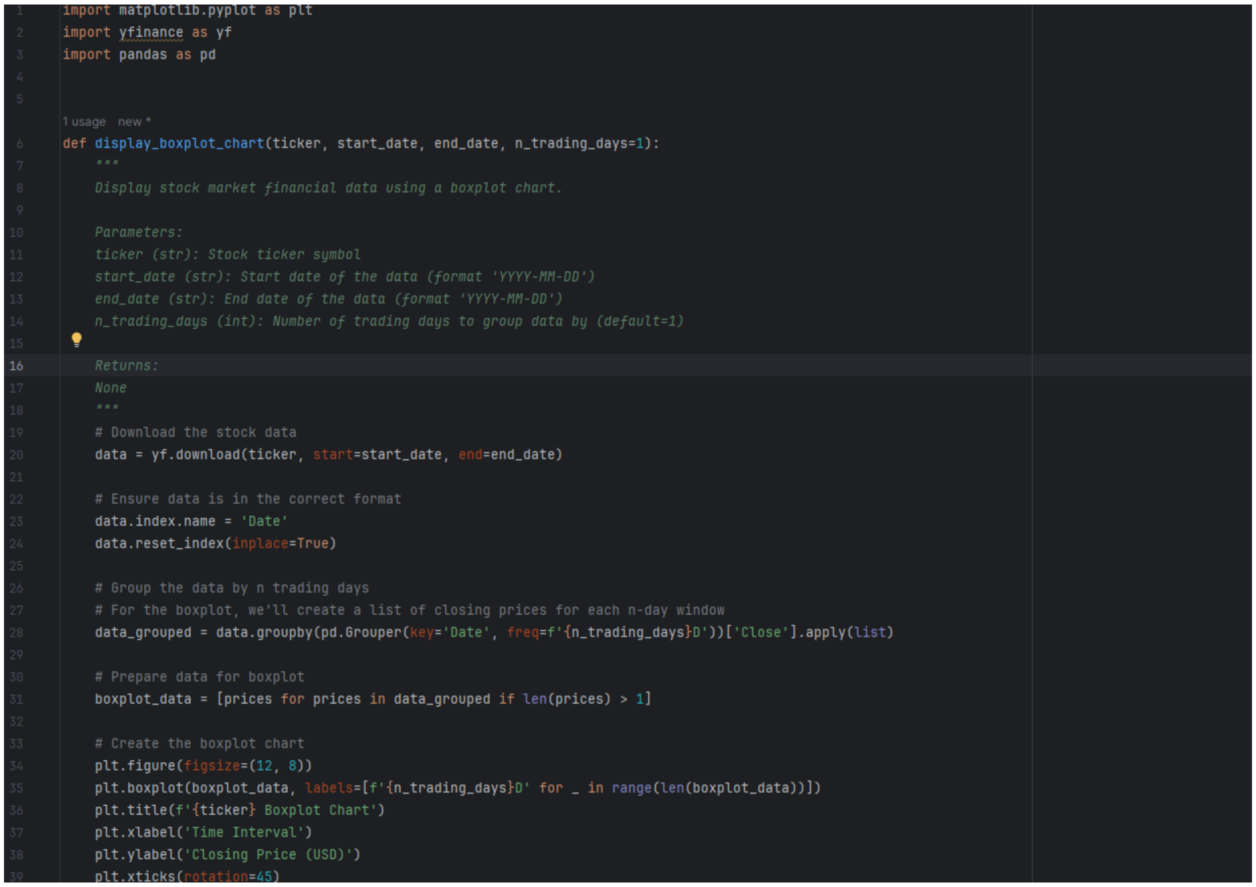
The plot function of mplfinance is used to create the candlestick chart.
The chart is configured to plot candlestick data in the following configuration:
data_grouped: This is the preprocessed data to create the chart.
type='candle': This says that it's going to be a candlestick chart.
title: sets the title for the chart.
ylabel: Labels the y-axis for price.
ylabel_lower: Labels the y-axis for the volume subplot.
volume=True: Adds a volume subplot below the candlestick chart.
style='yahoo': Uses the Yahoo Finance style for the chart.
figratio=(16, 9): Sets the aspect ratio of the figure to widescreen.
figscale=1.2: Scales the size of the figure.
show_nontrading=True: Displays days where no trading occurred.
datetime_format='%Y-%m-%d': Formats the date on the x-axis.

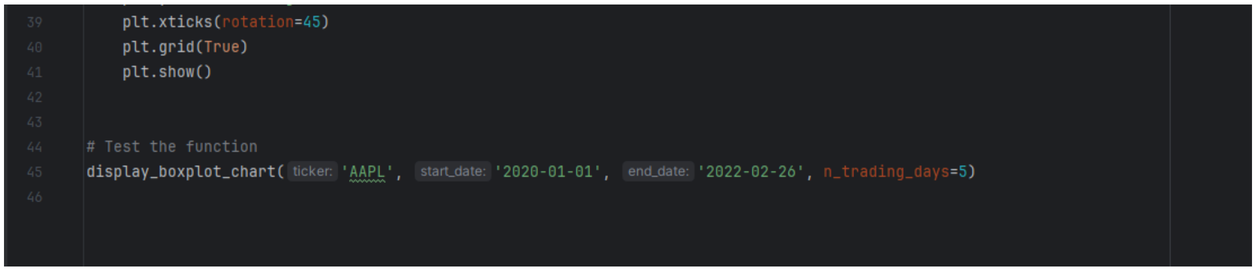
Output of Candlestick_chart.py:

N=5



Bloxplot_chart.py:





```python
import matplotlib.pyplot as plt
import yfinance as yf
import pandas as pd


def display_boxplot_chart(ticker, start_date, end_date, n_trading_days=1):
    """
    Display stock market financial data using a boxplot chart.

    Parameters:
    ticker (str): Stock ticker symbol
    start_date (str): Start date of the data (format 'YYYY-MM-DD')
    end_date (str): End date of the data (format 'YYYY-MM-DD')
    n_trading_days (int): Number of trading days to group data by (default=1)

    Returns:
    None
    """
    # Download the stock data
    data = yf.download(ticker, start=start_date, end=end_date)

    # Ensure data is in the correct format
    data.index.name = 'Date'
    data.reset_index(inplace=True)

    # Group the data by n trading days
    # For the boxplot, we'll create a list of closing prices for each n-day window
    data_grouped = data.groupby(pd.Grouper(key='Date', freq=f'{n_trading_days}D'))['Close'].apply(list)

    # Prepare data for boxplot
    boxplot_data = [prices for prices in data_grouped if len(prices) > 1]

    # Create the boxplot chart
    plt.figure(figsize=(12, 8))
    plt.boxplot(boxplot_data, labels=[f'{n_trading_days}D' for _ in range(len(boxplot_data))])
    plt.title(f'{ticker} Boxplot Chart')
    plt.xlabel('Time Interval')
    plt.ylabel('Closing Price (USD)')
    plt.xticks(rotation=45)
    plt.grid(True)
    plt.show()


# Test the function
display_boxplot_chart('AAPL', '2020-01-01', '2022-02-26', n_trading_days=5)
```

Imports:

matplotlib.pyplot: This library creates a range of plots, including boxplots.
yfinance: This library allows you to download historical data of stocks from Yahoo Finance.
pandas: This is a library used to manipulate and analyze data.

Function Definition:

display_boxplot_chart is defined to create the boxplot chart of stock market data.
Parameters:

ticker: This is the stock symbol; for example, 'AAPL' for Apple - for which you want to fetch and visualize data.
start_date: The date from which you want to fetch the data; this is in the 'YYYY-MM-DD' format.
end_date: This is the date until which you want the data; it's also in the 'YYYY-MM-DD' format.
n_trading_days: An integer that defines the number of days to aggregate the data. That means by this parameter, a time interval is defined; for example, if it is 5, then that means it groups data in 5-day intervals.

Get Stock Data:

The code downloads the historical stock data between the specified start and end dates using the library yfinance.

Format Data:

It cleans up the downloaded data to make the date column named 'Date' and set as index. Then, the function reset_index() is called on it to make 'Date' a regular column.

Group Data by Trading Days:

Then, it will group the data into windows of size n_trading_days. This means if n_trading_days is 5, then data will be aggregated into 5-day windows.
For each group, it generates a list of closing prices. These will be used in creating the boxplot.

Prepare Data for Boxplot:

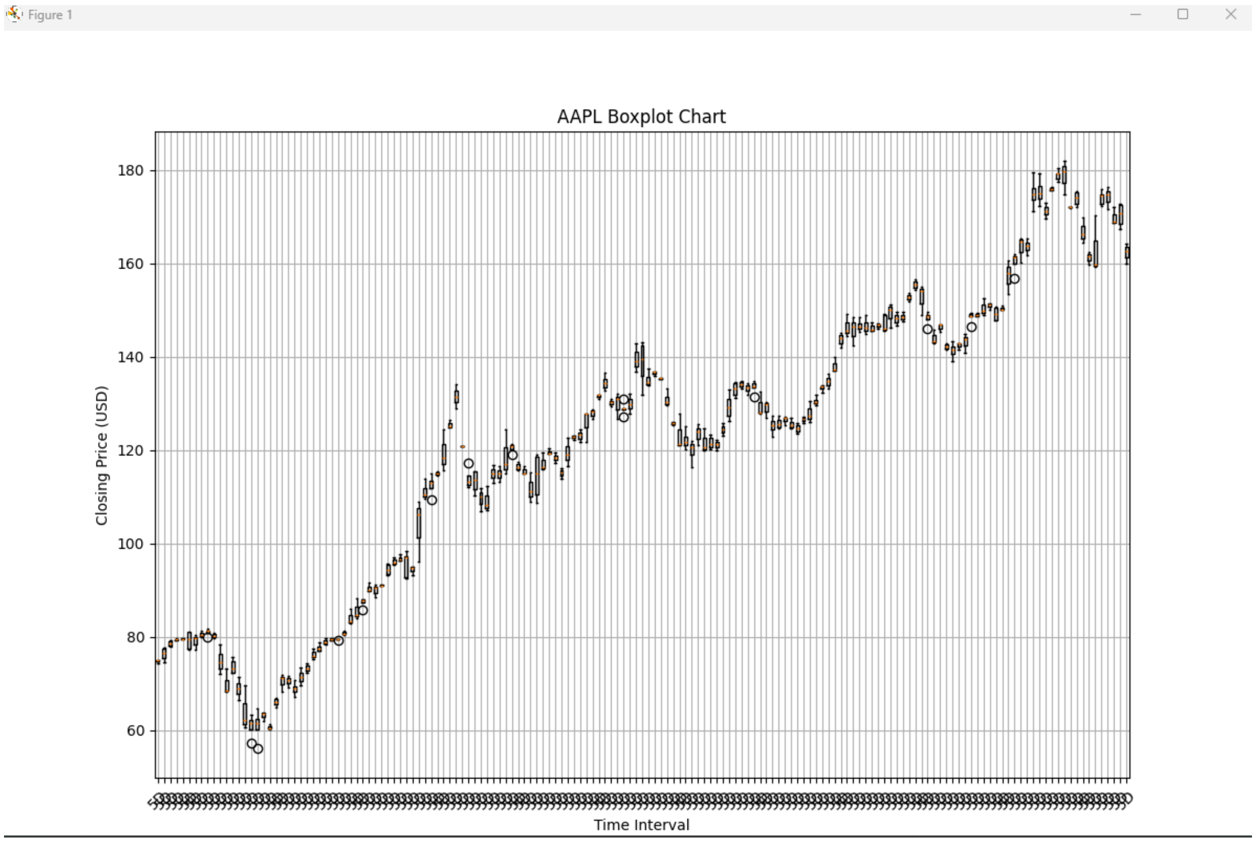
The lists of closing prices for each n_trading_days window are collected into boxplot_data. Only windows with more than one price are collected because otherwise it's meaningless to calculate statistics.

Create Boxplot Chart:

It creates a boxplot chart via matplotlib.pyplot:
plt.figure(figsize=(12, 8)): This sets the figure size to 12x8 inches.

plt.boxplot(boxplot_data, labels=[f'{n_trading_days}D' for _ in range(len(boxplot_data))]): This will create the boxplot based on boxplot_data. Note that even though each boxplot might represent more than one window, each is labeled with its time interval (n_trading_days). The labels parameter gives a label to x-axis.
plt.title(f'{ticker} Boxplot Chart'): This sets the title of the chart, including the stock ticker.
plt.xlabel('Time Interval'): It places a label on the x-axis with 'Time Interval'.
plt.ylabel('Closing Price (USD)'): It places a label on the y-axis with 'Closing Price (USD)'.
plt.xticks(rotation=45): The above line of code will rotate the x-axis labels to 45 degrees for better readability.
plt.grid(True): The grid is drawn on the plot so that the values will become more accurate.

Output for Bloxplot_chart.py:



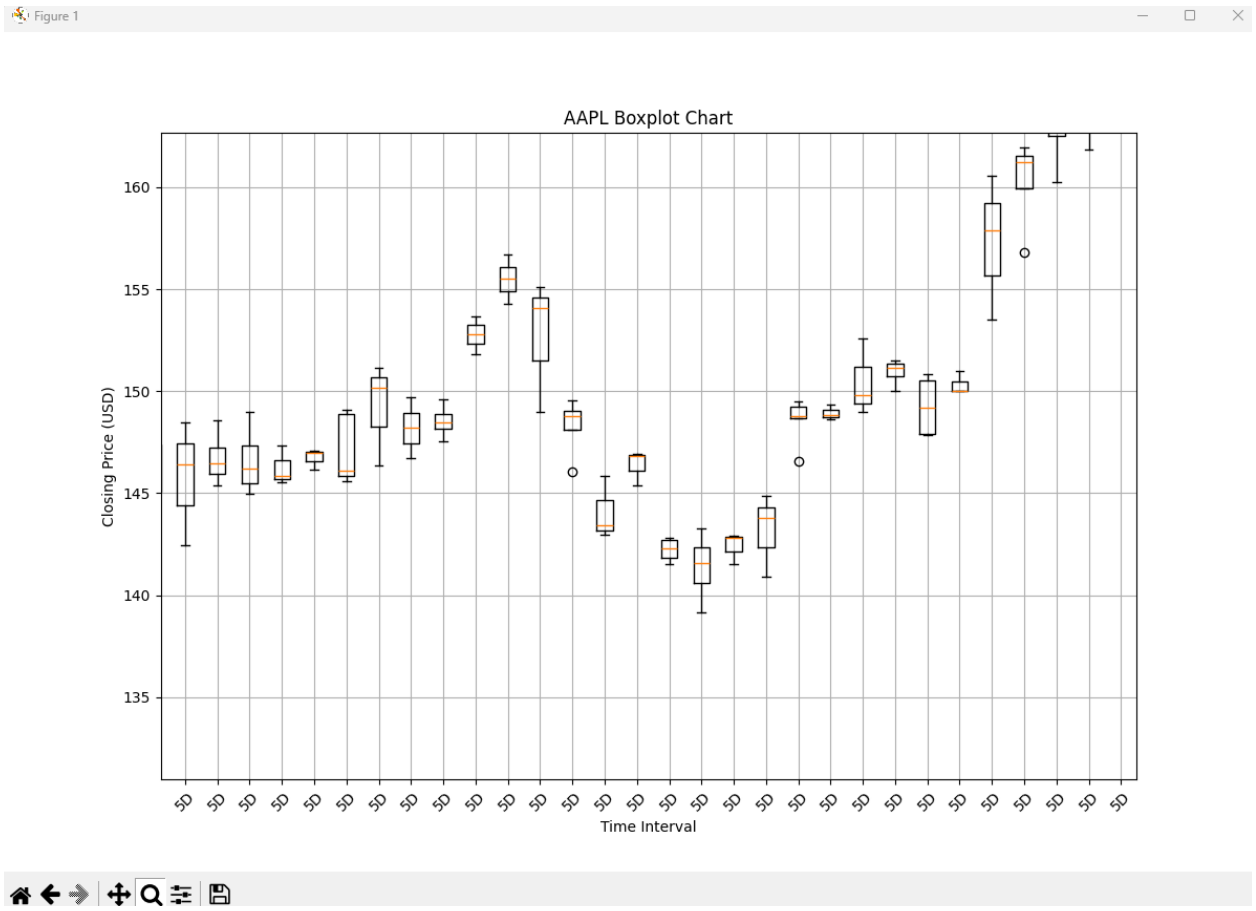


Figure 1
AAPL Boxplot Chart
Closing Price (USD)
160
155
150
145
140
135
5D
Time Interval

N = 5.