COS30018 - Option B - Task 3: Data processing 1

Student Name: Nguyen Duc Le Nguyen

Id: 104224493

This report details the implementation of two functions for displaying stock market financial data using candlestick charts and boxplot charts. The functions were implemented using Python, leveraging the yfinance, mplfinance, and matplotlib libraries. Below is a detailed explanation of the code, including descriptions of the arguments and parameters used in the functions.

Candlestick Chart Function

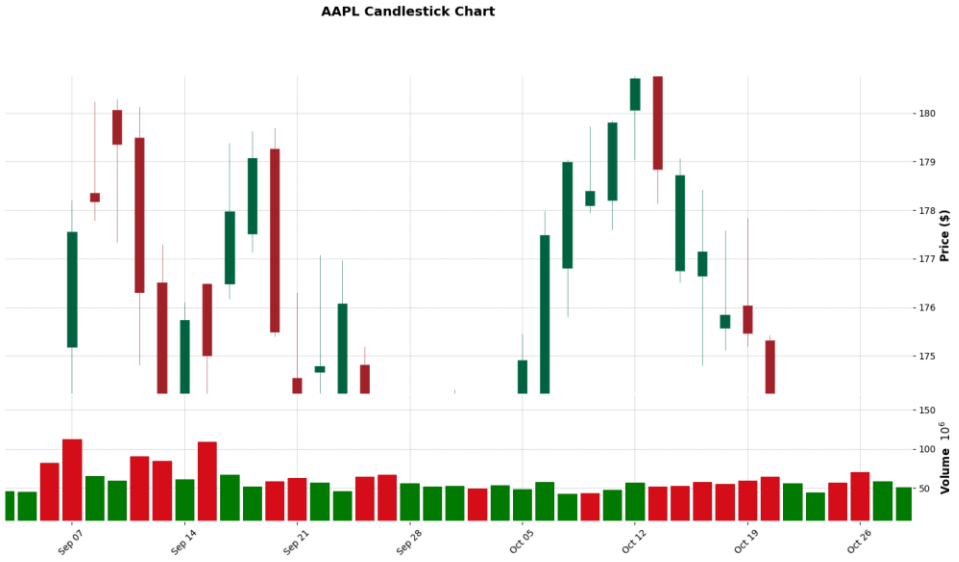

Figure 1 - Candlestick Chart

The candlestick chart is a financial chart used to represent the price movements of a stock over a specific period. Each "candle" in the chart shows four key pieces of information: the opening price, closing price, highest price, and lowest price for a given period. The code leverages the mplfinance library to create and display the candlestick chart.

```
def plot_candlestick(ticker, start_date, end_date, save_chart=False, chart_path='candlestick_chart.png'):
    """
```

Figure 2 - Candlestick Function

Function Parameters:

ticker: The stock ticker symbol (e.g., "AAPL" for Apple).

start_date: The start date for the data range.

end_date: The end date for the data range.

save_chart: Boolean indicating whether to save the chart as an image.

chart_path: The file path to save the chart image, if save_chart is True.

```
# Fetch the stock data
df = yf.download(ticker, start=start_date, end=end_date)
```

Figure 3 - Download Data

Fetching Data: The yf.download function is used to download stock data for the specified ticker and date range.

```
# Ensure the index is a datetime index
df.index = pd.to_datetime(df.index)
```

Figure 4 - change Index to Datetime

Datetime Index: The index of the DataFrame is ensured to be a datetime index for proper plotting.

```
# Plot the candlestick chart
mpf.plot(df, type='candle', style='charles', title=f'{ticker} Candlestick Chart', ylabel='Price ($)', volume=True)
```

Figure 5 - Plot chart

Plotting the Candlestick Chart:

The mpf.plot function is used to create the candlestick chart.

type='candle' specifies the type of plot.

style='charles' specifies the plot style.

title sets the title of the chart.

ylabel labels the y-axis.

volume=True adds a volume subplot.

If save_chart is True, the chart is saved to the specified path.

Boxplot Chart Function

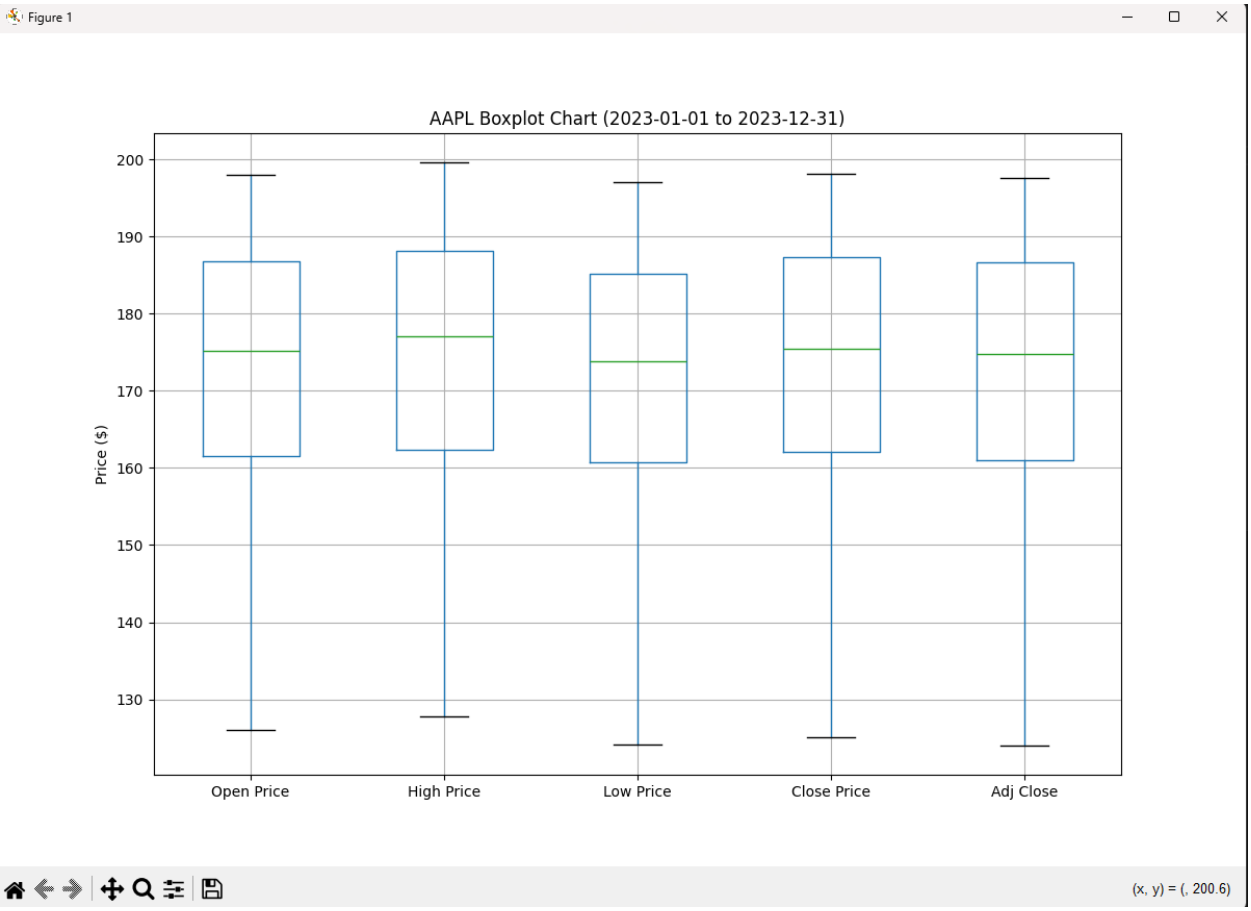

Figure 6 - Boxplot Chart

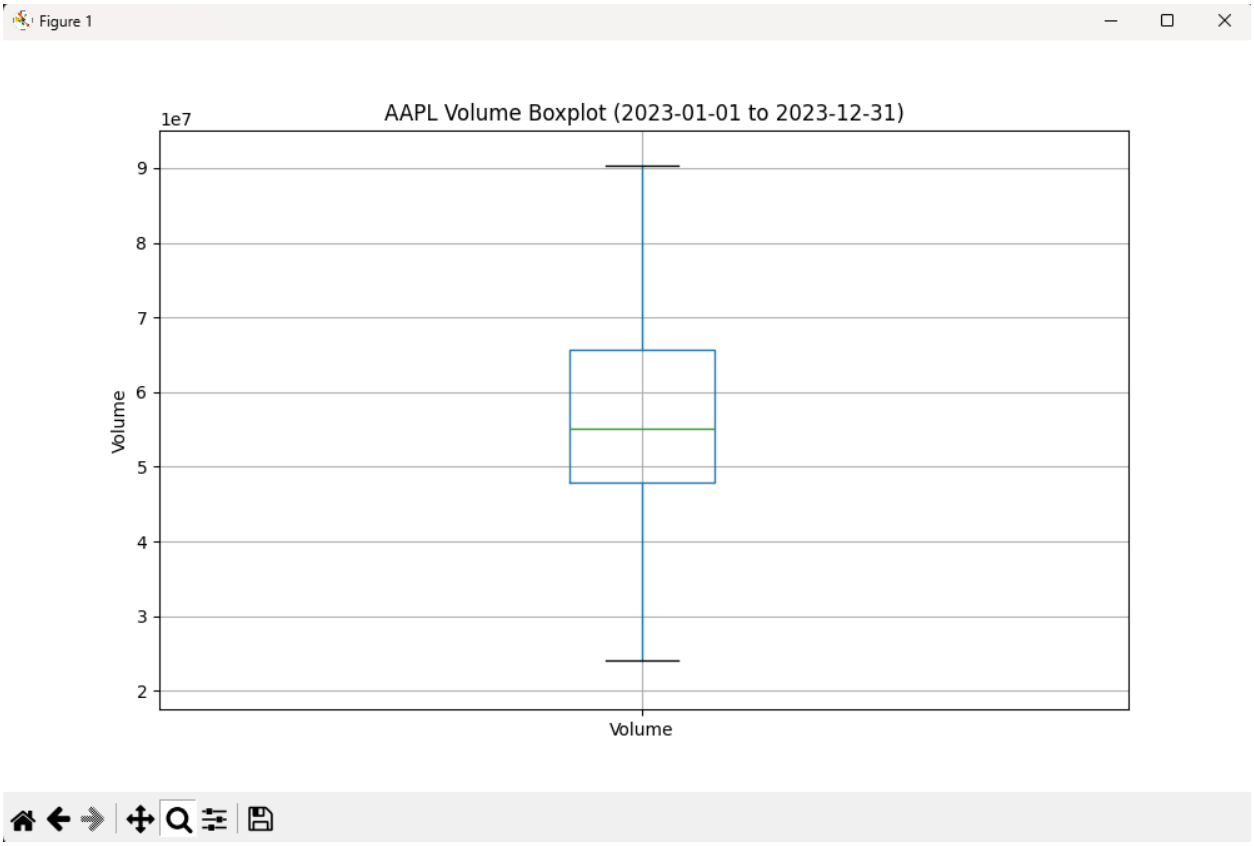

Figure 7 - Volume Boxplot

A boxplot chart is a standardized way of displaying the distribution of data based on a five-number summary: minimum, first quartile (Q1), median, third quartile (Q3), and maximum. This function creates boxplots for the Open, High, Low, and Close prices of a stock.

```
def plot_boxplots(ticker, start_date, end_date, save_chart=False, chart_path='multiple_boxplots_chart.png'):
    """
```

Figure 8 - BoxPlot Function

Function Parameters:

ticker: The stock ticker symbol (e.g., "AAPL" for Apple).

start_date: The start date for the data range.

end_date: The end date for the data range.

save_chart: Boolean indicating whether to save the chart as an image.

chart_path: The file path to save the chart image, if save_chart is True.

```
# Fetch the stock data
df = yf.download(ticker, start=start_date, end=end_date)
```

Figure 9 - Download Data

Fetching Data: The yf.download function is used to download stock data for the specified ticker and date range.

```
# Ensure the index is a datetime index
df.index = pd.to_datetime(df.index)
```

Figure 10 - Index to DateTime

Datetime Index: The index of the DataFrame is ensured to be a datetime index for proper plotting.

```
# Create a boxplot for the 'Open', 'High', 'Low', and 'Close' prices
plt.figure(figsize=(12, 8))
df_to_plot = df[['Open', 'High', 'Low', 'Close']]
df_to_plot.boxplot()

plt.title(f'{ticker} Boxplot Chart ({start_date} to {end_date})')
plt.ylabel('Price ($)')
plt.xticks([1, 2, 3, 4], ['Open Price', 'High Price', 'Low Price', 'Close Price'])

# Save the chart as an image if specified
if save_chart:
    plt.savefig(chart_path)

# Show the plot
plt.show()
```

Figure 11 - Boxplot configuration

Creating the Boxplot:

A plt.figure is created to set the figure size.

The DataFrame is subsetted to include only the 'Open', 'High', 'Low', and 'Close' columns.

The boxplot method is called on this subsetted DataFrame to create the boxplot.

plt.title sets the title of the chart.

plt.ylabel labels the y-axis.

plt.xticks sets the labels for the x-axis ticks.

If save_chart is True, the chart is saved to the specified path.

plt.show displays the chart.

Main Challenges Faced

Data Fetching: I had to make sure I was getting the correct data in the last task for the specified date range and stock ticker. This involved dealing with potential errors and ensuring that the dates were in the correct format.

Plot Customization: Customizing the plots to meet specific needs, such as adding volume subplots to candlestick charts and ensuring the boxplot shows the right columns, required a deep understanding of the plotting libraries I used.

References

- CoderzColumn Tutorial on Candlestick Charts: Candlestick Chart in Python
- yfinance Documentation: yfinance
- mplfinance Documentation: mplfinance
- matplotlib Documentation: matplotlib