COS30018 – Intelligent System

Option B: Stock Price Prediction

Report v0.3

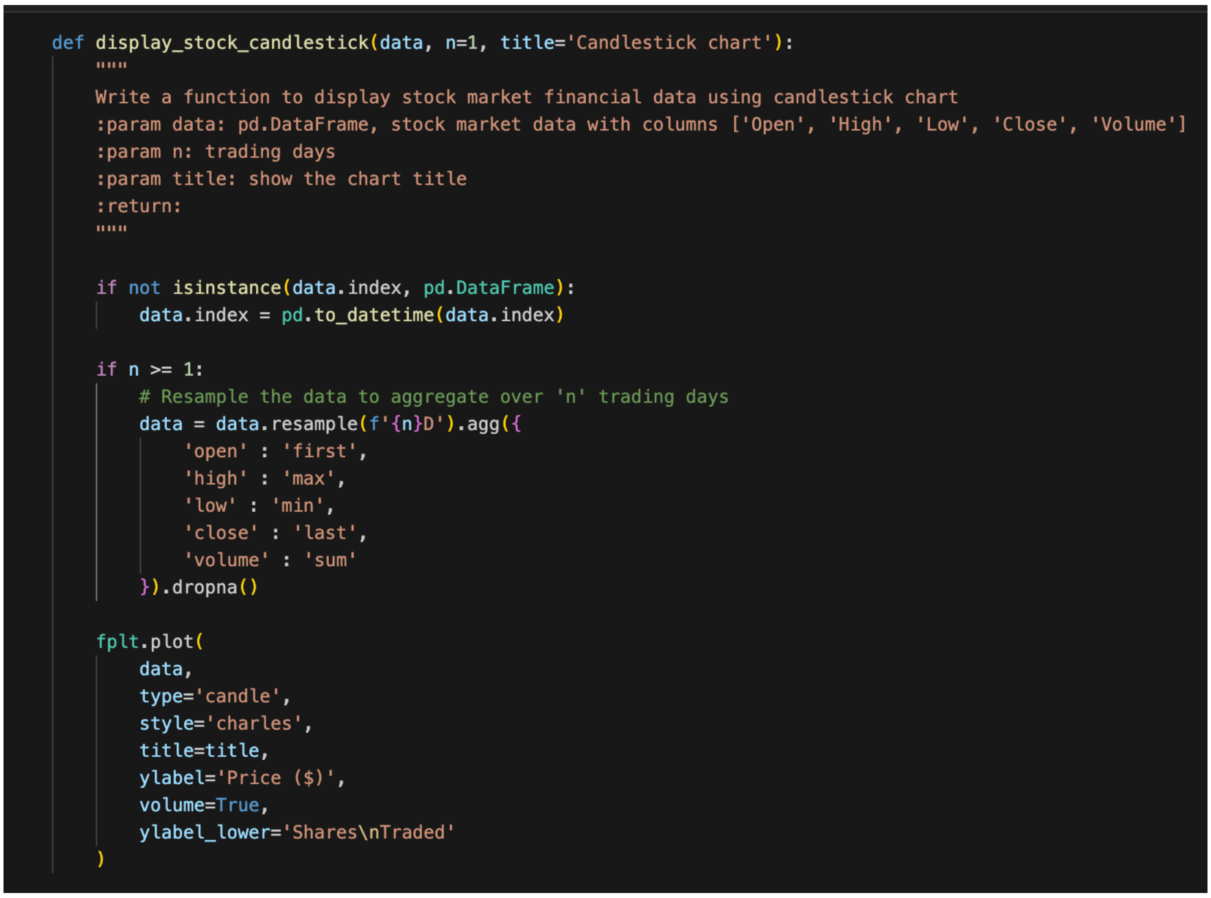
Name Phong Tran
Student Id:104334842

Class: 1-5
Tutor: Tuan Dung Lai

Table of Contents

1.Write a function using candlestick chart.

a) Explain



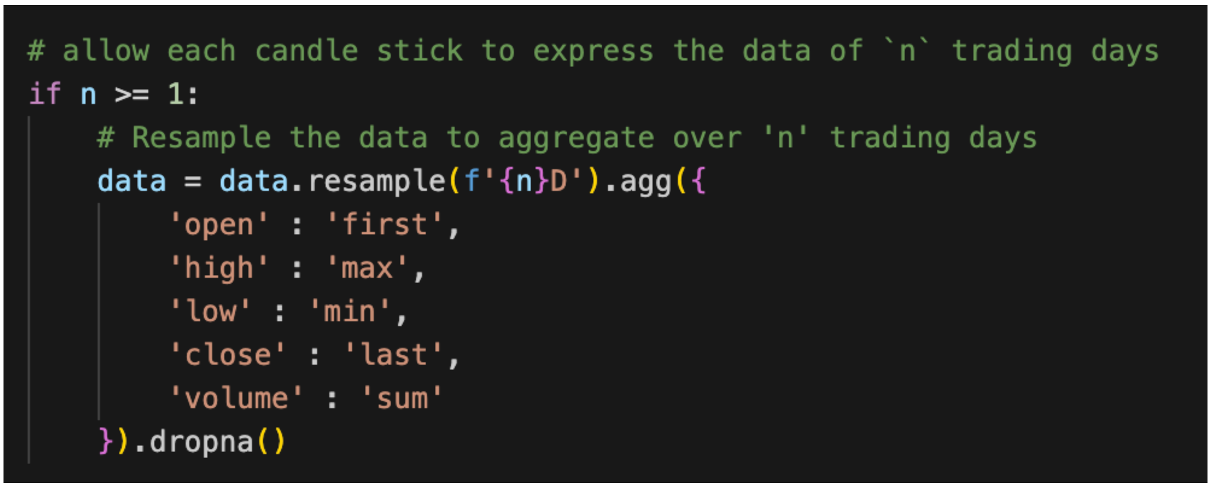
```python
def display_stock_candlestick(data, n=1, title='Candlestick chart'):
    """
    Write a function to display stock market financial data using candlestick chart
    :param data: pd.DataFrame, stock market data with columns ['Open', 'High', 'Low', 'Close', 'Volume']
    :param n: trading days
    :param title: show the chart title
    :return:
    """

    if not isinstance(data.index, pd.DataFrame):
        data.index = pd.to_datetime(data.index)

    if n >= 1:
        # Resample the data to aggregate over 'n' trading days
        data = data.resample(f'{n}D').agg({
            'open' : 'first',
            'high' : 'max',
            'low' : 'min',
            'close' : 'last',
            'volume' : 'sum'
        }).dropna()

    fplt.plot(
        data,
        type='candle',
        style='charles',
        title=title,
        ylabel='Price ($)',
        volume=True,
        ylabel_lower='Shares\nTraded'
    )
```

Check if index is not in the datetime format. Then use fplt as alternative called from mplfinance library to plot the data with parameters:

- data: retrieve data
- type: candle (also called candlestick)
- style: charles (to show color between red and green)
- title: show title in details
- volume: to see volume plot below the candlestick chart

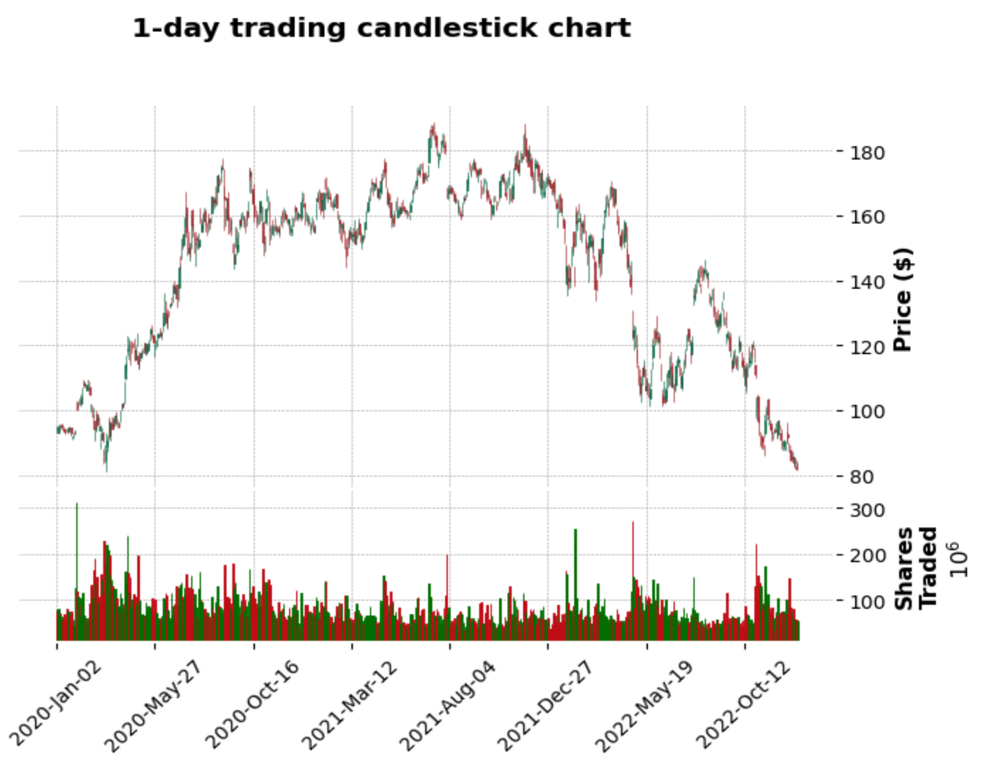
- ylabel_lower: to change label of y-axis of the volume plot

b) Optional with include an option to express the data of n trading days (n >= 1)



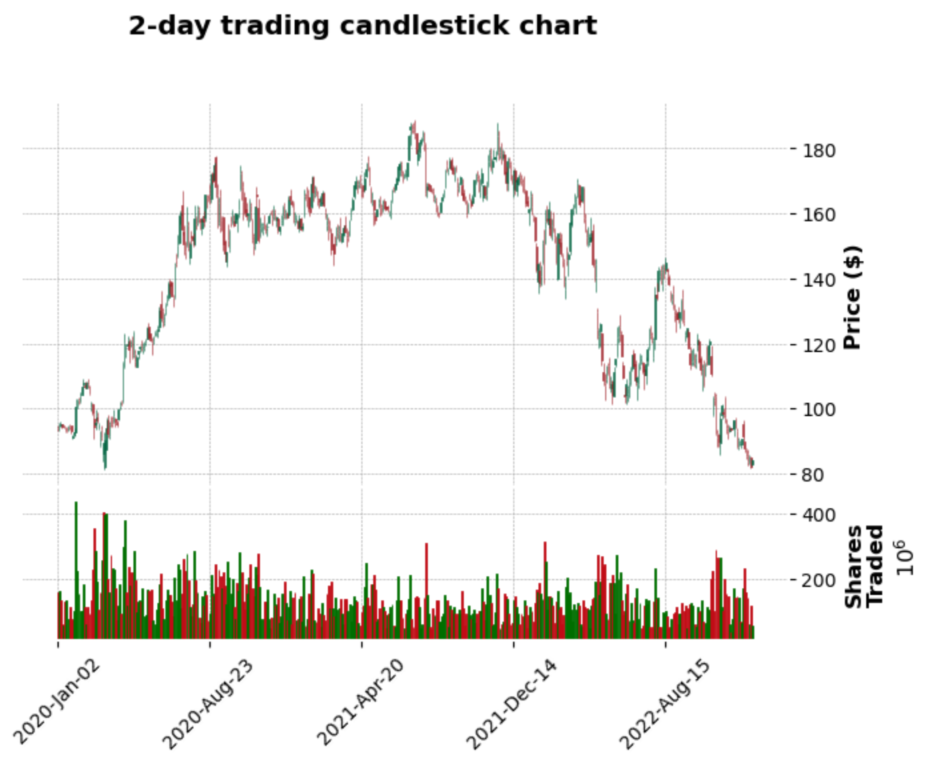
```python
# allow each candle stick to express the data of `n` trading days
if n >= 1:
    # Resample the data to aggregate over 'n' trading days
    data = data.resample(f'{n}D').agg({
        'open' : 'first',
        'high' : 'max',
        'low' : 'min',
        'close' : 'last',
        'volume' : 'sum'
    }).dropna()
```

If more than or equal to 1 trading days, the data will resample with corresponding days then aggregate with different function for each column. Finally, drop any rows that missing value.

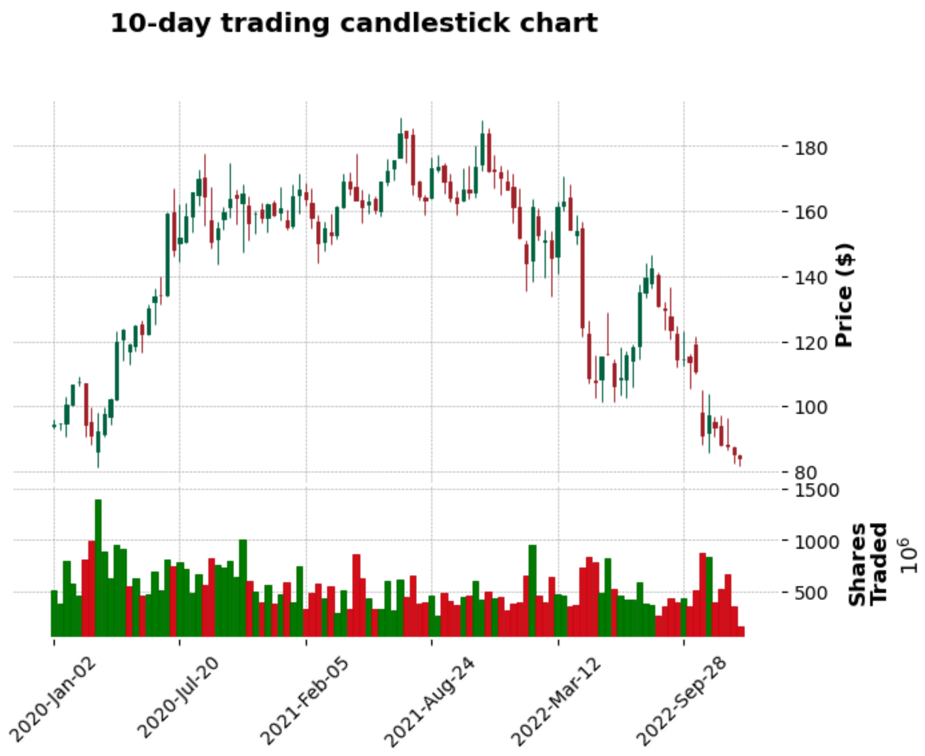


Images plot demonstration below:

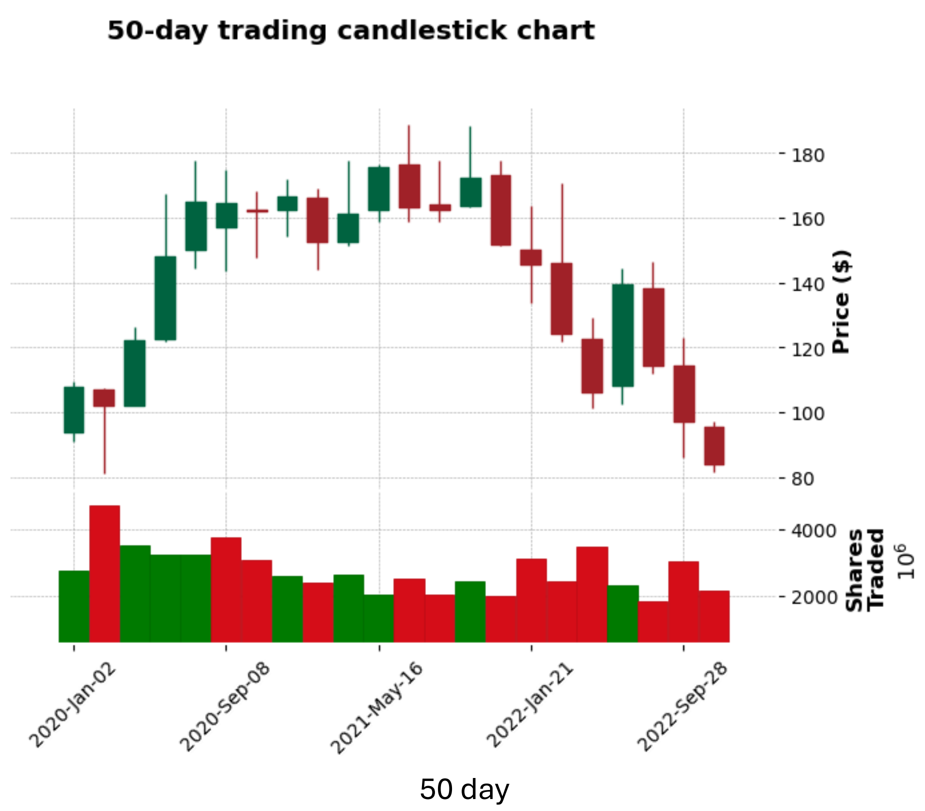
1 day


2-day trading candlestick chart
Price ($)
Shares Traded
10^6
180
160
140
120
100
80
400
200
2020-Jan-02
2020-Aug-23
2021-Apr-20
2021-Dec-14
2022-Aug-15

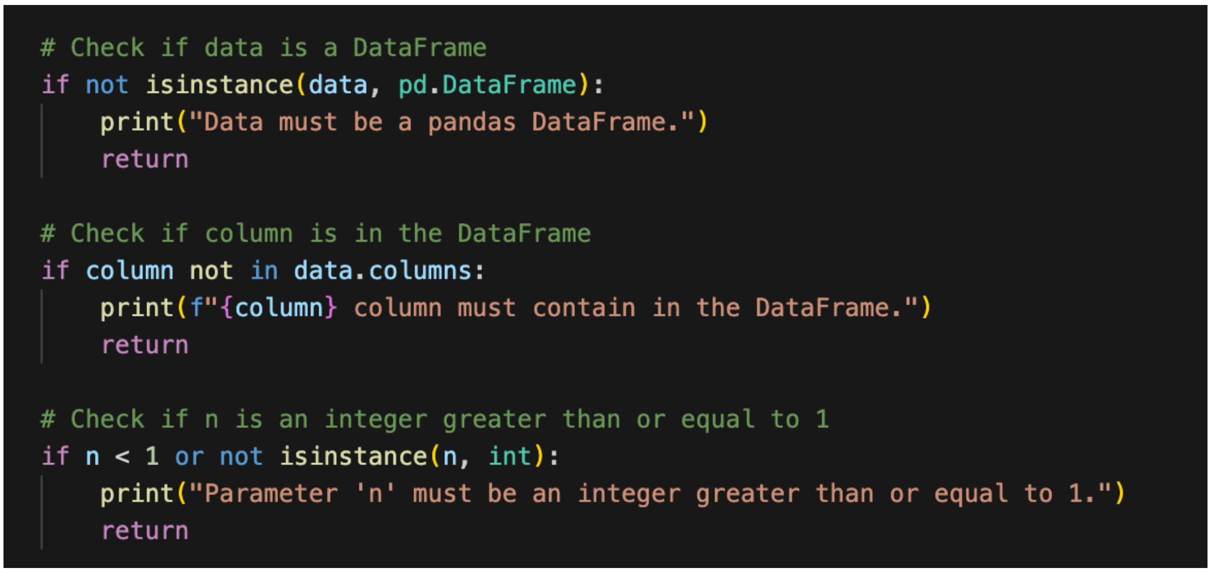
2 day


10-day trading candlestick chart
Price ($)
Shares Traded
10^6
180
160
140
120
100
80
1500
1000
500
2020-Jan-02
2020-Jul-20
2021-Feb-05
2021-Aug-24
2022-Mar-12
2022-Sep-28

10 day



2) Write a function display boxplot stock market financial data, and try to moving window of n consecutive trading days.



```
# Check if data is a DataFrame
if not isinstance(data, pd.DataFrame):
    print("Data must be a pandas DataFrame.")
    return

# Check if column is in the DataFrame
if column not in data.columns:
    print(f"{column} column must contain in the DataFrame.")
    return

# Check if n is an integer greater than or equal to 1
if n < 1 or not isinstance(n, int):
    print("Parameter 'n' must be an integer greater than or equal to 1.")
    return
```

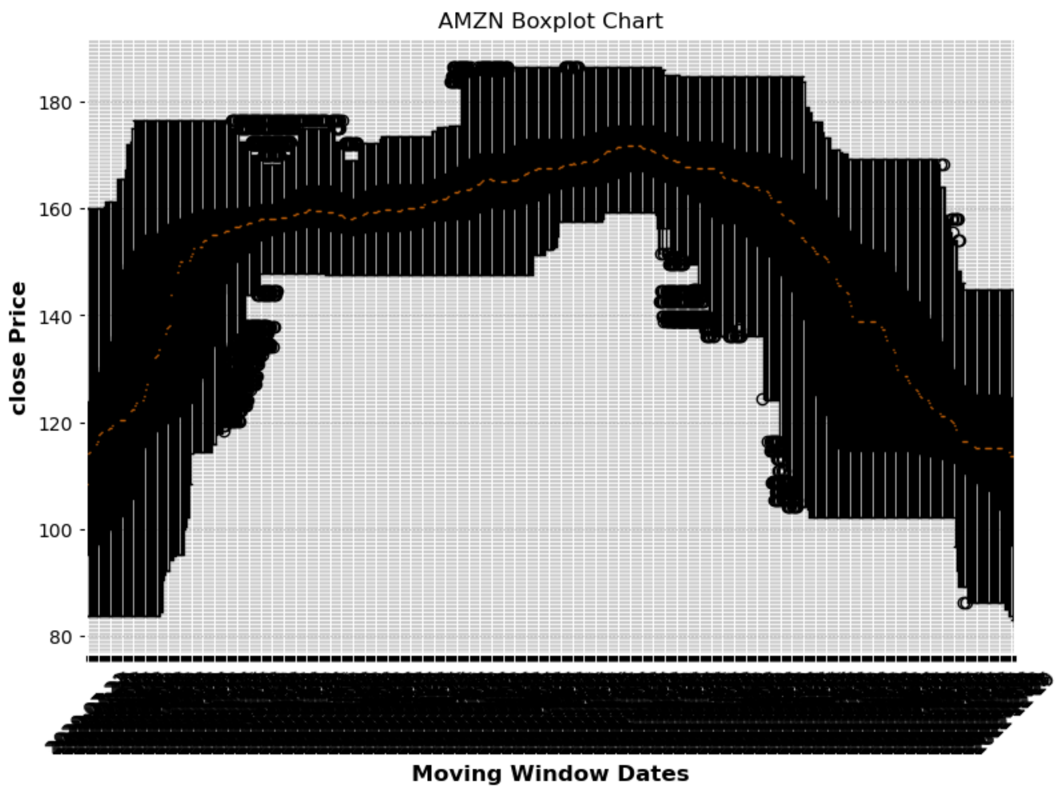
Check the condition to print and return at the function if satisfy the following conditions:

- check data or column if it in the dataframe type
- check ` n ` variable if it is a numeric and greater than 1

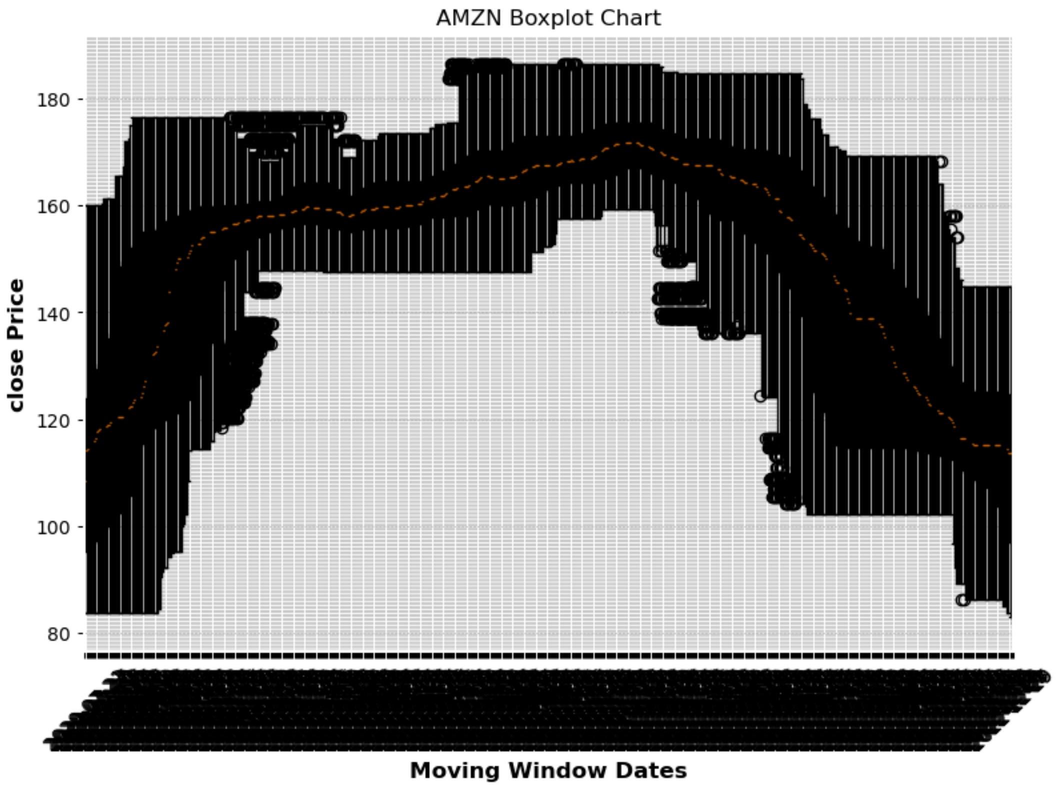
```python
# Calculate the total number of windows
total_windows = len(data) - n + 1 # If n=maximum number of days, total window will display last one
print (f"data: {len(data)}, n: {n}, total_windows: {total_windows}")
# len(data): total number of rows


# Raise an error if there are not enough data points for the specified window size
if total_windows <= 0:
    print("Not enough data points for the specified window size.")
    return

# Prepare the data for the boxplot chart
boxplot_data = []
# Labels for the x-axis
labels = []

# Iterate through the data to create the boxplot data
for i in range(total_windows):
    window_data = data[column].iloc[i:i+n] # Retrieve the data through the window size
    boxplot_data.append(window_data.values) # Append the data to the boxplot data list
    window_date = data.index[i + n - 1].strftime('%Y-%m-%d') # Get the last date of the window date
    labels.append(window_date) # Append the date to the labels list
```
Check the window total number by calculate with number of rows in data, number of days plus 1. Then check if total windows below or equal 0, before initialise the boxplot data and labels variables to assign data.

- Boxplot_data to display the dataframe data by retrieve through window size
- Labels used to display the window size by the last date

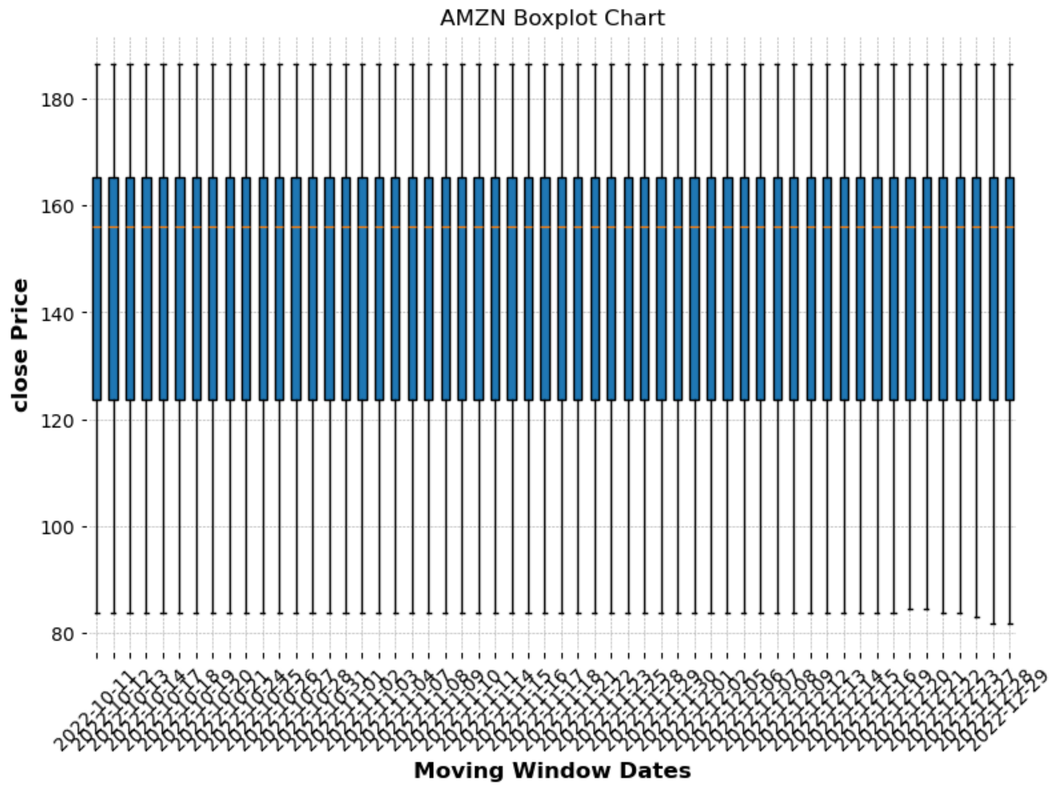
Display the boxplot data



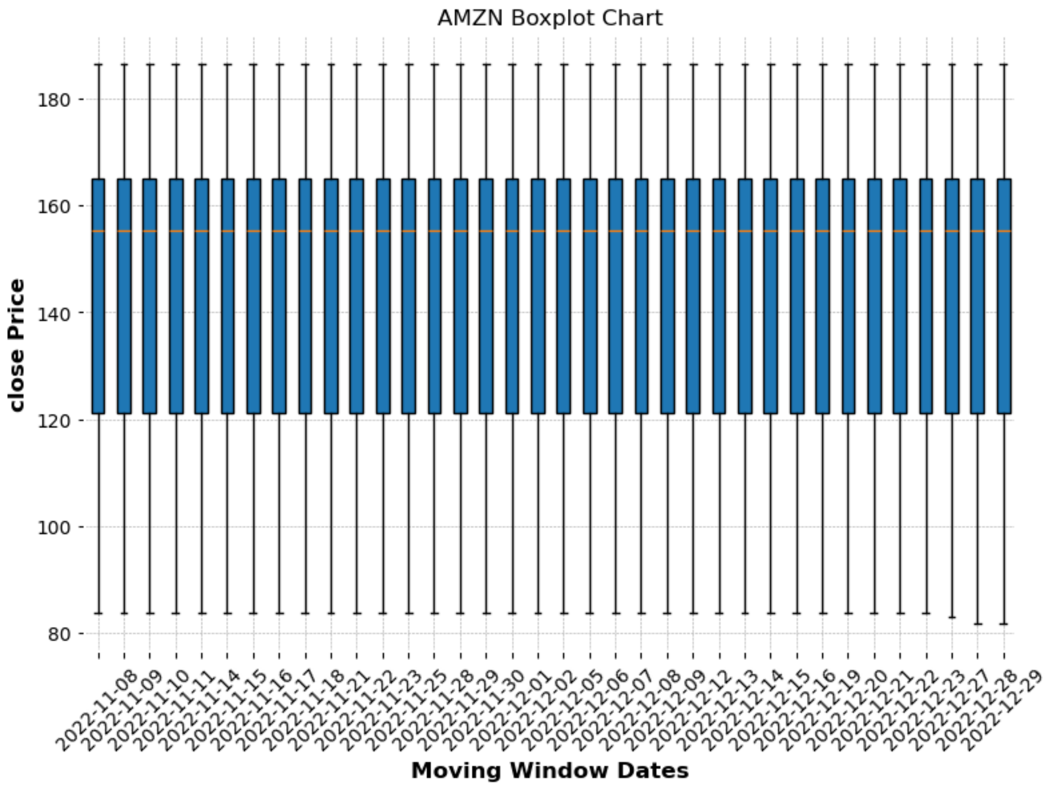
139 days



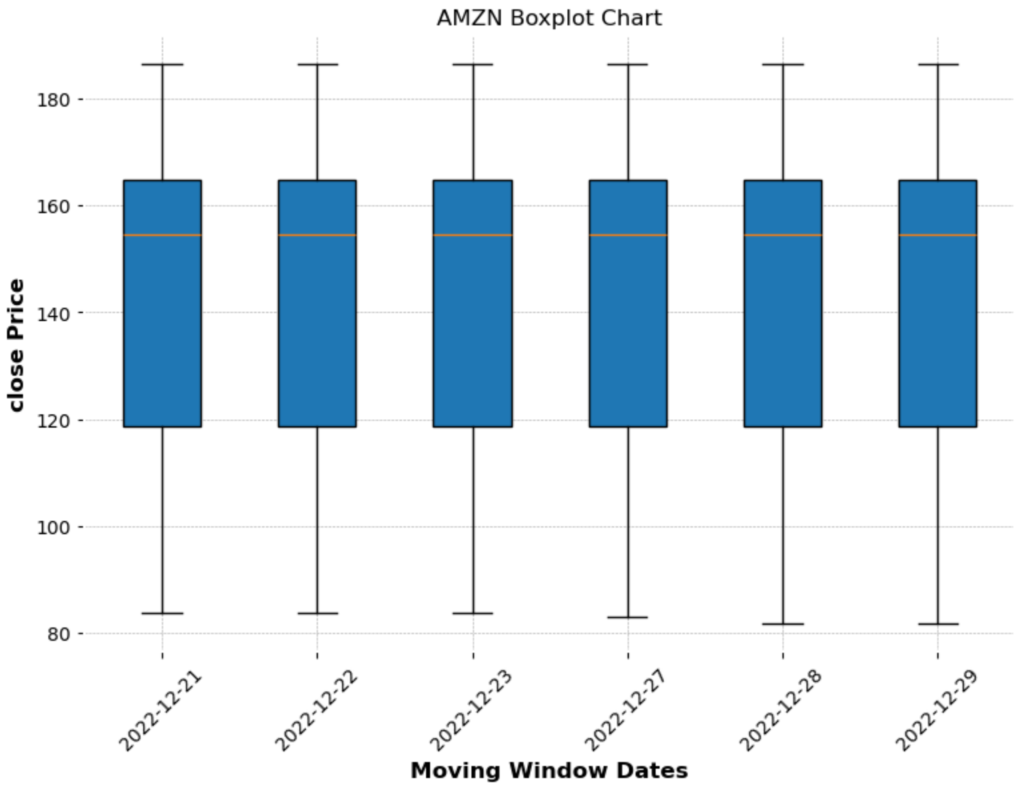
650 days



650 days



720 days



750 days