Assignment: Lesson 20

Part 1: Data Cleaning and Exploration:

1. Calculate basic summary statistics for each column (mean, median, standard deviation, etc.)

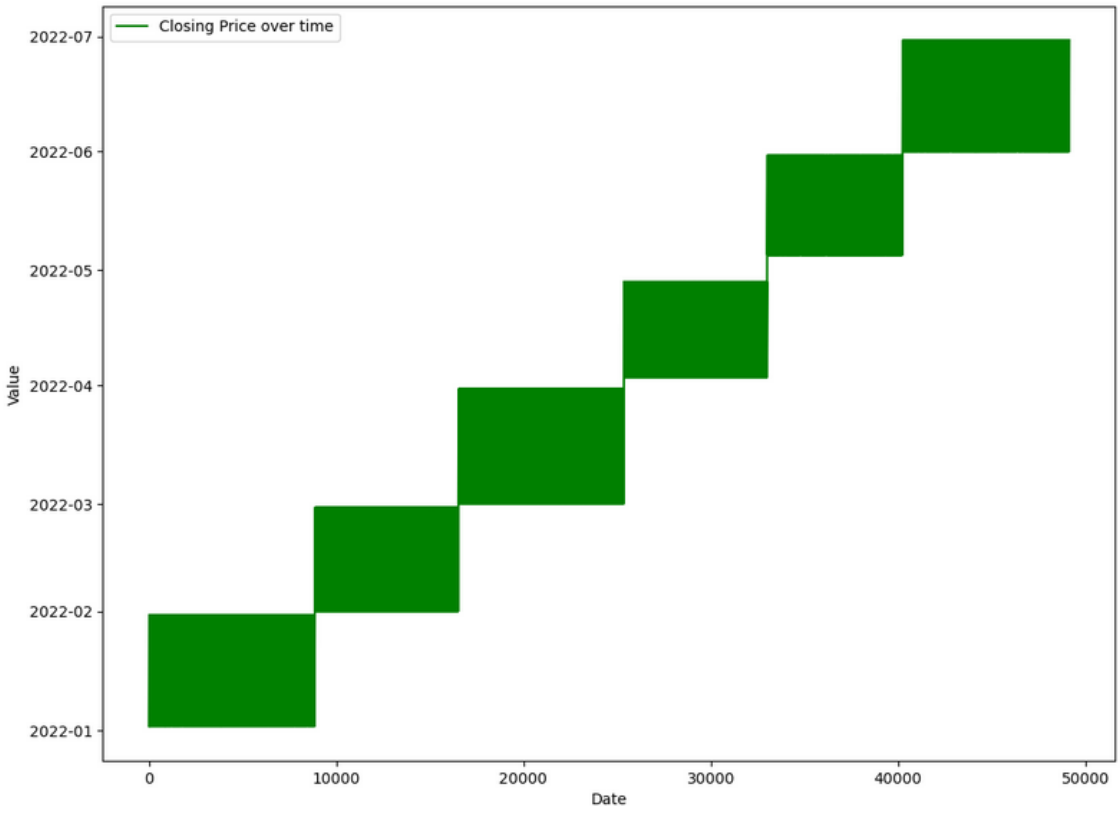
```
In [12]: df.describe()
```

Out[12]:

	Open	High	Low	Close	Volume
count	49158.000000	49158.000000	49158.000000	49158.000000	4.915800e+04
mean	157.869018	159.588214	155.906364	157.351462	5.619999e+05
std	520.191624	523.348078	517.136149	519.711667	1.276909e+06
min	3.900000	3.900000	3.000000	3.800000	1.000000e+00
25%	19.000000	19.300000	18.700000	19.000000	5.109475e+04
50%	40.300000	41.000000	39.535000	40.100000	1.824160e+05
75%	89.400000	90.500000	87.700000	88.700000	5.401398e+05
max	6000.000000	6050.000000	5975.000000	6000.500000	6.593180e+07

2. Explore the distribution of the 'Close' prices over time.

In [30]:
```
plt.figure(figsize=(12,9))
plt.plot(stock_data.index, stock_data['Date'], label='Closing Price over time',color='Green')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.show()
```



3. Identify and analyze any outliers (if any) in the dataset

In [14]:
```
df.dropna(inplace=True)
df
```

Out[14]:

	Date	Name	Open	High	Low	Close	Volume
0	02-01-2022	01.Bank	22.83	23.20	22.59	22.93	1842350.41
1	03-01-2022	01.Bank	23.03	23.29	22.74	22.90	1664989.63
2	04-01-2022	01.Bank	22.85	23.13	22.64	22.84	1354510.97
3	05-01-2022	01.Bank	22.91	23.20	22.70	22.98	1564334.81
4	06-01-2022	01.Bank	23.12	23.65	23.00	23.37	2586344.19
...	...	...	...	...	...	...	...
49153	26-06-2022	ZEALBANGLA	169.00	174.90	169.00	170.30	10480.00
49154	27-06-2022	ZEALBANGLA	174.10	176.00	166.90	167.50	13817.00
49155	28-06-2022	ZEALBANGLA	170.00	170.90	167.00	168.10	5214.00
49156	29-06-2022	ZEALBANGLA	167.10	169.00	164.90	165.10	6678.00
49157	30-06-2022	ZEALBANGLA	165.10	174.00	164.00	172.20	5883.00

49158 rows × 7 columns

In [15]:
```
import pandas as pd
from scipy import stats
```

In [17]:
```
data_column = 'Close'
df[data_column] = pd.to_numeric(df[data_column],errors='coerce')
z_scores = stats.zscore(df[data_column])
threshold = 3
outliers = (z_scores > threshold)|(z_scores > -threshold)
outlier_data = df[outliers]
outlier_data.tail()
```

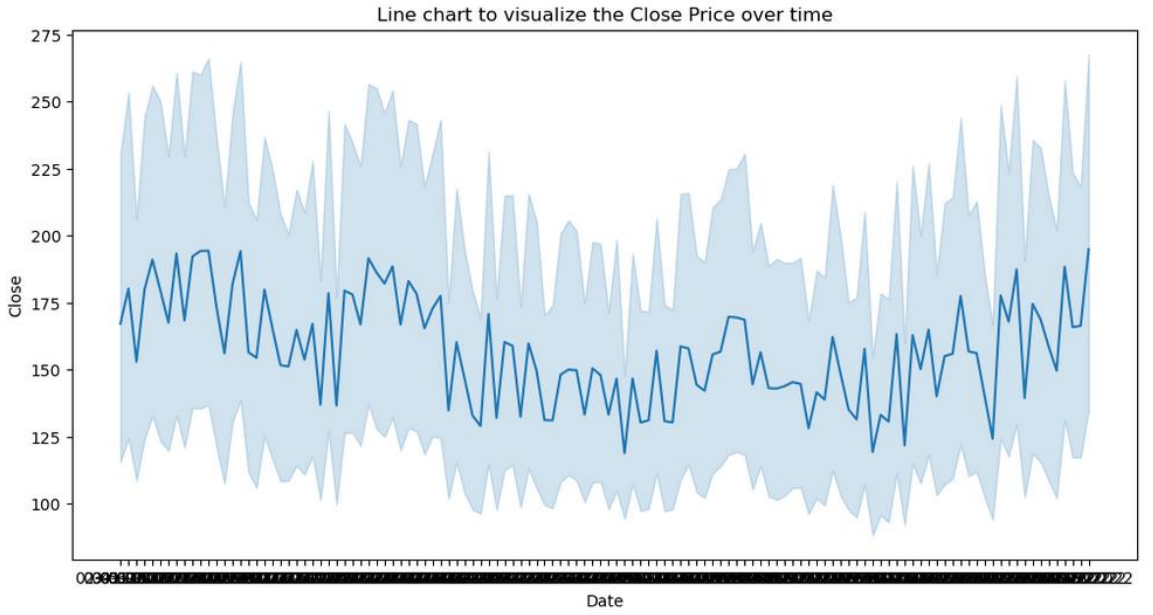
Out[17]:

	Date	Name	Open	High	Low	Close	Volume
49153	26-06-2022	ZEALBANGLA	169.0	174.9	169.0	170.3	10480.0
49154	27-06-2022	ZEALBANGLA	174.1	176.0	166.9	167.5	13817.0
49155	28-06-2022	ZEALBANGLA	170.0	170.9	167.0	168.1	5214.0
49156	29-06-2022	ZEALBANGLA	167.1	169.0	164.9	165.1	6678.0
49157	30-06-2022	ZEALBANGLA	165.1	174.0	164.0	172.2	5883.0

Part 2: Time Series Analysis / Rolling Window / Moving Averages :

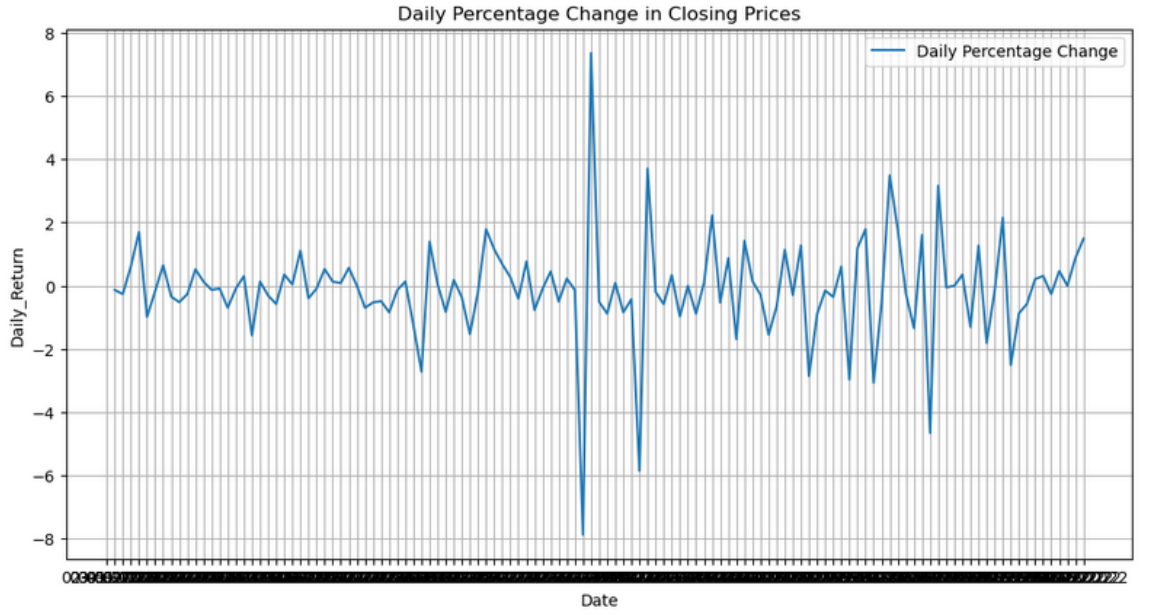
1. Create a line chart to visualize the 'Close' prices over time.

```
In [18]: if 'Date' in df.columns:
             df.set_index('Date', inplace=True)
         df['Close'] = pd.to_numeric(df['Close'], errors='coerce')
         plt.figure(figsize=(12,6))
         sns.lineplot(x=df.index, y='Close', data=df)
         plt.title("Line chart to visualize the Close Price over time")
         plt.xlabel = ('Date')
         plt.ylabel = ('Close Price')
         plt.show()
```



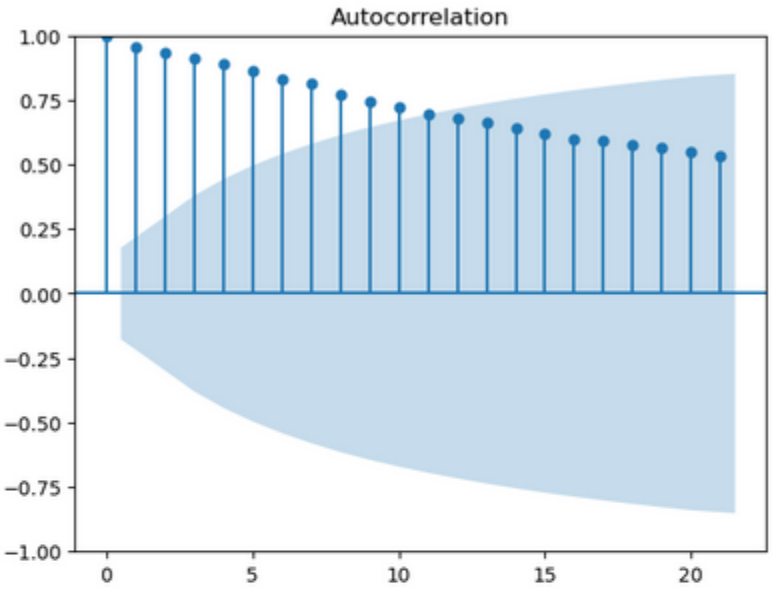
2. Calculate and plot the daily percentage change in closing prices.

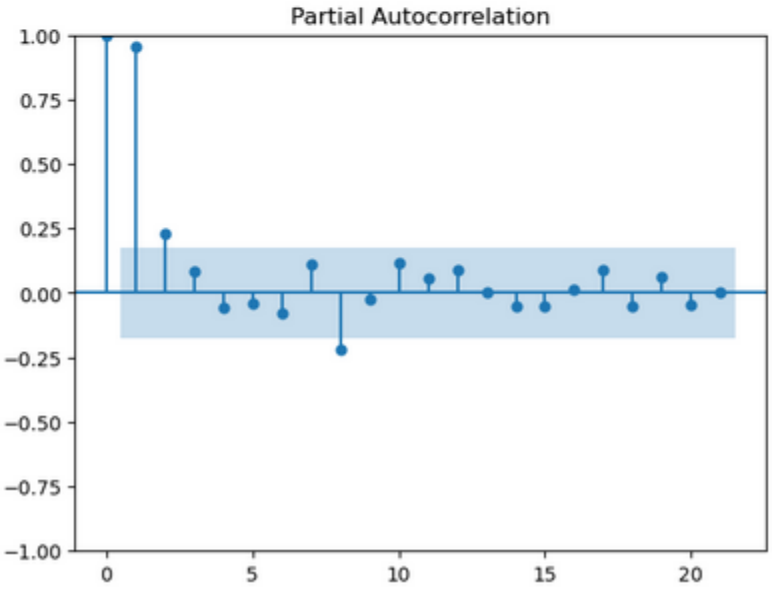
In [21]:
```
df = df.loc[~df.index.duplicated(keep='first')]
df['Daily_Return'] = df['Close'].pct_change() * 100
plt.figure(figsize=(12,6))
sns.lineplot(x=df.index, y='Daily_Return', data=df, label='Daily Percentage Change')
plt.title('Daily Percentage Change in Closing Prices')
plt.xlabel = ('Date')
plt.ylabel = ('Percentage Change (%)')
plt.legend()
plt.grid(True)
plt.show()
```



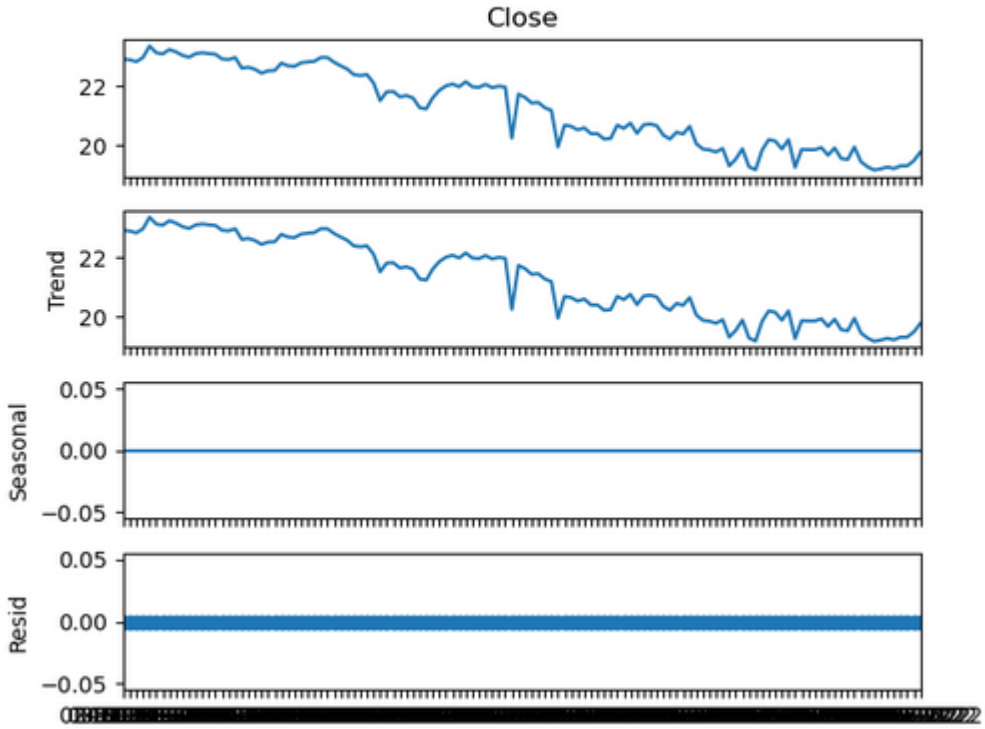
3. Investigate the presence of any trends or seasonality in the stock prices.

In [25]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plot_acf(df['Close'])
plt.show()
plot_pacf(df['Close'])
plt.show()
```





In [26]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(df['Close'], model='additive', period=1)
result.plot()
plt.show()
```



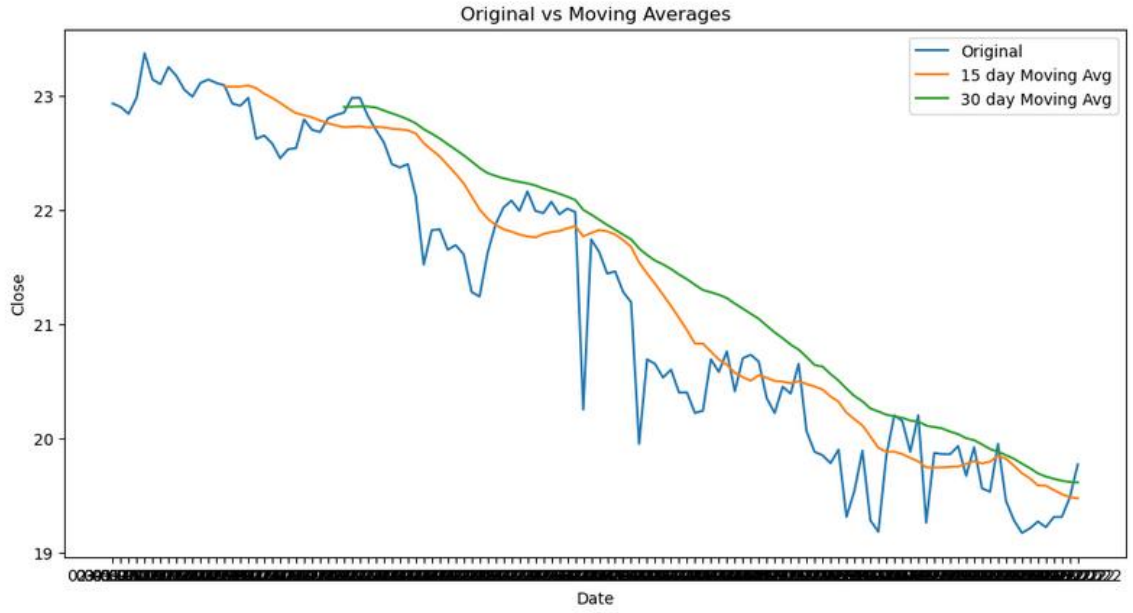
4. Apply moving averages to smooth the time series data in 15/30 day intervals against the original graph.

In [28]:
```
plt.figure(figsize=(12,6))
sns.lineplot(x=df.index, y='Close', data=df, label='Original')

#applying 15 day Moving Avg
df['MA_15'] = df['Close'].rolling(window=15).mean()
sns.lineplot(x=df.index, y='MA_15', data=df, label='15 day Moving Avg')

#applying 30 day Moving Avg
df['MA_30'] = df['Close'].rolling(window=30).mean()
sns.lineplot(x=df.index, y='MA_30', data=df, label='30 day Moving Avg')

plt.title('Original vs Moving Averages')
plt.xlabel = ('Date')
plt.ylabel = ('Close Price')
plt.legend()
plt.show()
```



5. Calculate the average closing price for each stock.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib as mlb
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data= pd.read_csv("G:\DATA_SCIENCE\Bohubrihi DA\Stock_Market_Data.csv")
```

In [3]:
```
df=pd.DataFrame(data)
```

In [4]:
```
df
```

Out[4]:

	Date	Name	Open	High	Low	Close	Volume
0	02-01-2022	01.Bank	22.83	23.20	22.59	22.93	1842350.41
1	03-01-2022	01.Bank	23.03	23.29	22.74	22.90	1664989.63
2	04-01-2022	01.Bank	22.85	23.13	22.64	22.84	1354510.97
3	05-01-2022	01.Bank	22.91	23.20	22.70	22.98	1564334.81
4	06-01-2022	01.Bank	23.12	23.65	23.00	23.37	2586344.19
...	...	...	...	...	...	...	...
49153	26-06-2022	ZEALBANGLA	169.00	174.90	169.00	170.30	10480.00
49154	27-06-2022	ZEALBANGLA	174.10	176.00	166.90	167.50	13817.00
49155	28-06-2022	ZEALBANGLA	170.00	170.90	167.00	168.10	5214.00
49156	29-06-2022	ZEALBANGLA	167.10	169.00	164.90	165.10	6678.00
49157	30-06-2022	ZEALBANGLA	165.10	174.00	164.00	172.20	5883.00

49158 rows × 7 columns

In [5]:
```
df['Close'] = pd.to_numeric(df['Close'], errors='coerce')
avg_price = df.groupby('Name')['Close'].mean()
avg_price.head()
```

Out[5]:
```
Name
01.Bank                      21.260902
02.Cement                    96.600820
03.Ceramics_Sector           71.225164
04.Engineering              132.352459
05.Financial_Institutions    29.253525
Name: Close, dtype: float64
```

6. Identify the top 5 and bottom 5 stocks based on average closing price.

```
In [8]: df['Close'] = pd.to_numeric(df['Close'], errors='coerce')
        avg_price = df.groupby('Name')['Close'].mean()
        avg_price
```

```
Out[8]: Name
        01.Bank                        21.260902
        02.Cement                      96.600820
        03.Ceramics_Sector             71.225164
        04.Engineering                132.352459
        05.Financial_Institutions      29.253525
                                          ...
        WMSHIPYARD                     12.370492
        YPL                            21.339344
        ZAHEENSPIN                      9.964754
        ZAHINTEX                        7.858197
        ZEALBANGLA                    150.338525
        Name: Close, Length: 412, dtype: float64
```

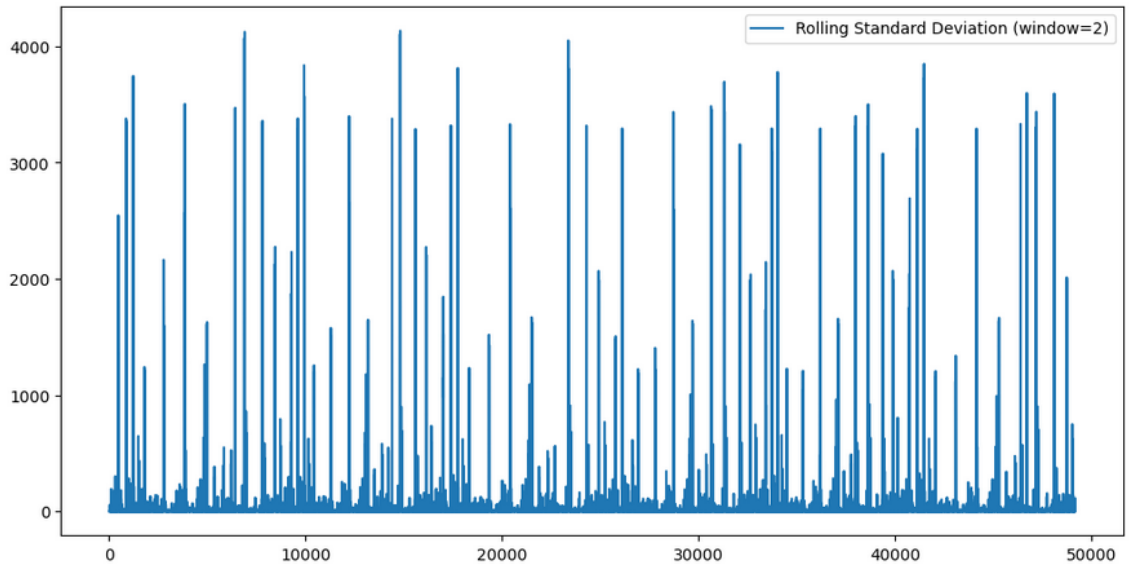
```
In [10]: avg_price.head()
         avg_price.tail()
```

```
Out[10]: Name
         WMSHIPYARD     12.370492
         YPL            21.339344
         ZAHEENSPIN      9.964754
         ZAHINTEX        7.858197
         ZEALBANGLA    150.338525
         Name: Close, dtype: float64
```

Part 3: Volatility Analysis:

1. Calculate and plot the rolling standard deviation of the 'Close' prices.

```
In [11]: rolling_std = df['Close'].rolling(window=2).std()
         plt.figure(figsize=(12,6))
         plt.plot(rolling_std, label='Rolling Standard Deviation (window=2)')
         plt.xlabel = ('Date')
         plt.ylabel = ('Standard Deviation')
         plt.legend()
         plt.show()
```



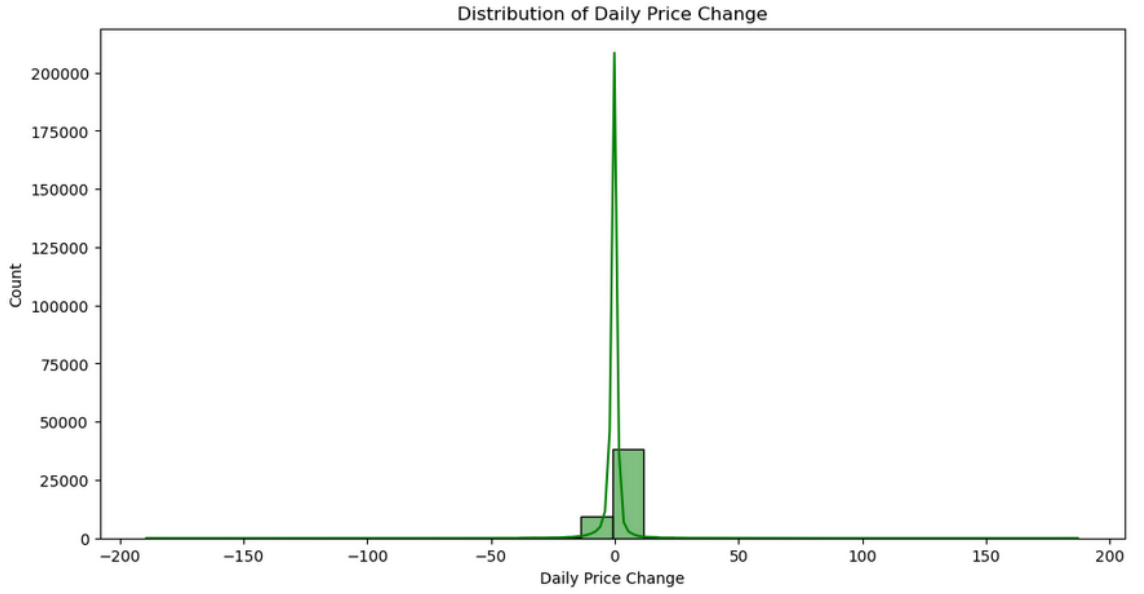
2. Create a new column for daily price change (Close - Open).

```
In [12]: df['Daily Price Change'] = df['Close'] - df['Open']
         df['Daily Price Change'].head()

Out[12]: 0    0.10
         1   -0.13
         2   -0.01
         3    0.07
         4    0.25
         Name: Daily Price Change, dtype: float64
```

3. Analyze the distribution of daily price changes.

In [14]:
```
plt.figure(figsize=(12,6))
sns.histplot(df['Daily Price Change'], bins=30, kde=True, color='green')
plt.title('Distribution of Daily Price Change')
plt.xlabel = ('Daily Price Change')
plt.ylabel = ('Frequency')
plt.show()
```



4. Identify days with the largest price increases and decreases.

```
In [20]: #identify day with the largest price increase
         max_increase_day = df.loc[df['Daily Price Change'].idxmax()]
         max_increase_day

Out[20]: Date                  29-06-2022
         Name                  SJIBLPBOND
         Open                      4710.0
         High                      4899.0
         Low                       4710.0
         Close                     4897.0
         Volume                     101.0
         Daily Price Change         187.0
         Name: 48081, dtype: object

In [21]: #identify day with the largest price decrease
         max_decrease_day = df.loc[df['Daily Price Change'].idxmin()]
         max_decrease_day

Out[21]: Date                  07-03-2022
         Name                  RECKITTBEN
         Open                      5753.0
         High                      5753.0
         Low                       5550.0
         Close                     5563.8
         Volume                    1876.0
         Daily Price Change        -189.2
         Name: 23365, dtype: object
```

5. Identify stocks with unusually high trading volume on certain days.

In [34]:
```
#calculating z-score for the vol column
df['Volume Z-Score'] = (df['Volume'] - df['Volume'].mean()) / df['Volume'].std()
threshold = 2
#identifying stacks for unusually high trading volume
unusual_volume_stocks = df[df['Volume Z-Score'] > threshold]
```

In [33]:
```
unusual_volume_stocks.head()
```

Out[33]:

	Date	Name	Open	High	Low	Close	Volume	Daily Price Change	Volume Z-Score
52	12-01-2022	03.Ceramics_Sector	76.46	79.04	75.30	77.32	3148906.60	0.86	2.025914
54	16-01-2022	03.Ceramics_Sector	78.06	81.36	76.96	79.48	3351889.00	1.42	2.184877
319	17-01-2022	15.Services_&_Real_Estate	60.18	61.83	59.28	61.15	6056375.75	0.97	4.302873
320	18-01-2022	15.Services_&_Real_Estate	63.03	66.15	59.05	64.75	5141492.75	1.72	3.586390
321	19-01-2022	15.Services_&_Real_Estate	64.30	65.85	61.98	63.30	3928104.25	-1.00	2.636135

In [32]:
```
unusual_volume_stocks.tail()
```

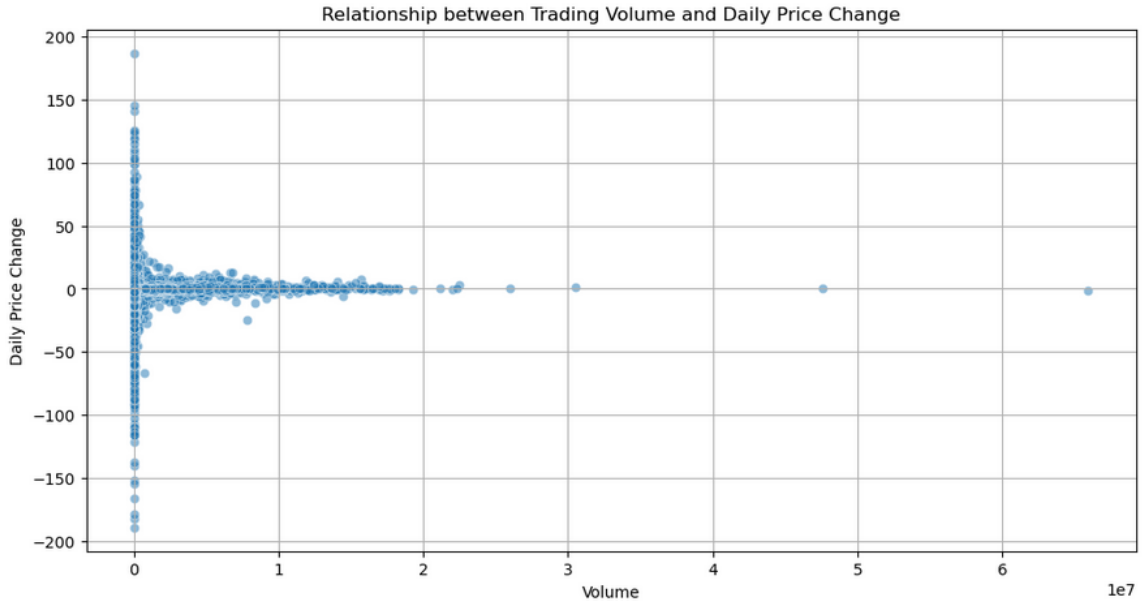
Out[32]:

	Date	Name	Open	High	Low	Close	Volume	Daily Price Change	Volume Z-Score
49075	08-06-2022	YPL	21.7	23.0	21.7	22.7	4296959.0	1.0	2.925001
49081	16-06-2022	YPL	22.8	23.7	22.8	23.3	3394619.0	0.5	2.218341
49089	28-06-2022	YPL	22.8	23.6	21.9	23.6	6145142.0	0.8	4.372389
49090	29-06-2022	YPL	24.3	24.6	23.3	23.4	4463125.0	-0.9	3.055132
49091	30-06-2022	YPL	23.5	24.2	23.0	23.3	3844363.0	-0.2	2.570554

Part 4: Correlation and Heat maps:

1. Explore the relationship between trading volume and volatility.

```
In [37]: plt.figure(figsize=(12,6))
sns.scatterplot(x='Volume', y='Daily Price Change', data=df, alpha=0.5)
plt.title('Relationship between Trading Volume and Daily Price Change')
plt.xlabel = ('Trading Volume')
plt.ylabel = ('Daily Price Change')
plt.grid()
plt.show()
```



2. Calculate the correlation matrix between the 'Open' & 'High', 'Low' &'Close' prices.

```
In [39]: price_columns = ['Open','High','Low','Close']
price_corr_matrix = df[price_columns].corr()
price_corr_matrix
```

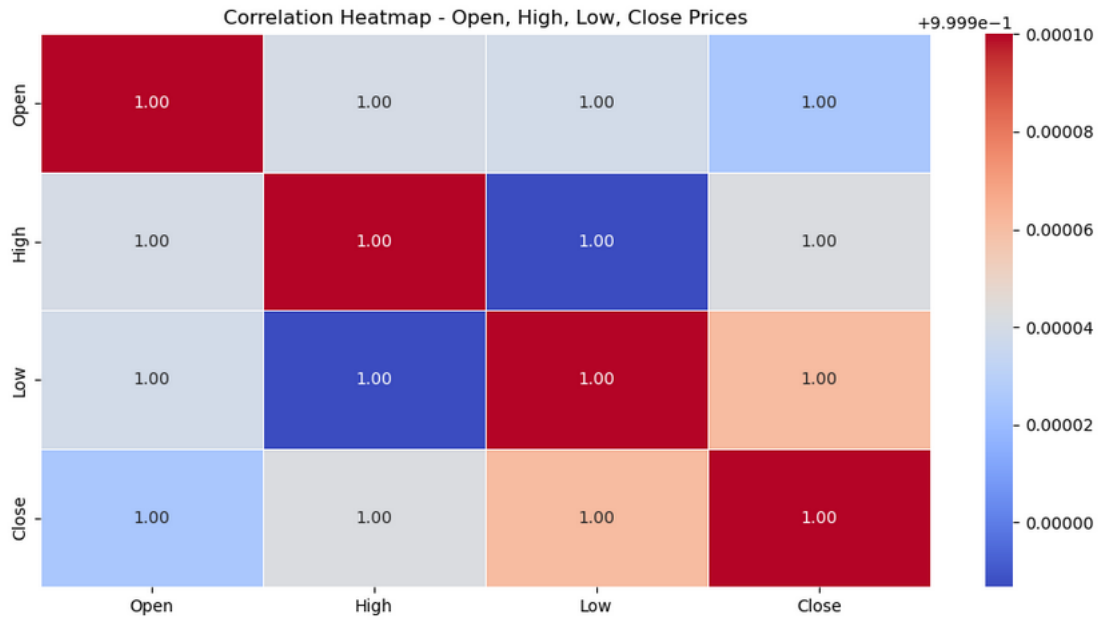
Out[39]:

	Open	High	Low	Close
Open	1.000000	0.999940	0.999939	0.999925
High	0.999940	1.000000	0.999887	0.999942
Low	0.999939	0.999887	1.000000	0.999961
Close	0.999925	0.999942	0.999961	1.000000

3. Create a heatmap to visualize the correlations using the seaborn package.

```
In [40]: plt.figure(figsize=(12,6))
         sns.heatmap(price_corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
         plt.title('Correlation Heatmap - Open, High, Low, Close Prices')
         plt.show
```

```
Out[40]: <function matplotlib.pyplot.show(close=None, block=None)>
```



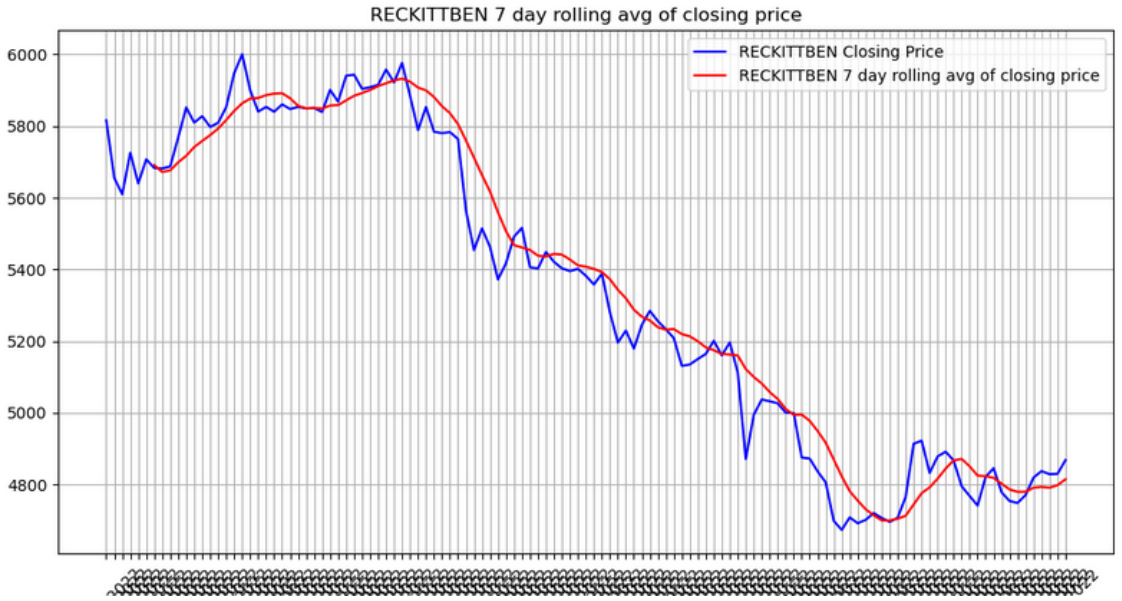
Bonus Task

During the rolling window analysis, we encountered a warning. Find out what's causing this & apply a fix to avoid the warning.

```
In [42]: specific_company ='RECKITTBEN'
specific_data=df[df['Name']==specific_company]
specific_data['7_day_rolling_avg']=specific_data['Close'].rolling(window=7).mean()
plt.figure(figsize=(12,6))
plt.plot(specific_data['Date'],specific_data['Close'],label=f'{specific_company} Closing Price',color='blue')
plt.plot(specific_data['Date'],specific_data['7_day_rolling_avg'],label=f'{specific_company} 7 day rolling avg of closing price',col
plt.xlabel = ('Date')
plt.ylabel = ('Closing Price')
plt.title(f'{specific_company} 7 day rolling avg of closing price')
plt.grid()
plt.legend()
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\Saniat\AppData\Local\Temp\ipykernel_1164\2308828378.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/indexing.html#returning-a-view-versus-a-copy
  specific_data['7_day_rolling_avg']=specific_data['Close'].rolling(window=7).mean()
```



The cause: The warning we were encountered usually indicates that we were trying to modify a DataFrame that is a subset of another DataFrame. This warning arises to inform us that modifying the subset may not modify the original DataFrame as we might expect.

To avoid this warning, we can use the .loc accessor to explicitly set values in the original DataFrame. Refer to the next page

The Solution: By using .copy() when creating specific_data, we can make sure that we are working with a copy of the data, and then using .loc to set values in the original DataFrame, we can avoid the SettingWithCopyWarning.

In [44]:
```
specific_company ='RECKITTBEN'
specific_data=df[df['Name']==specific_company].copy()
specific_data['7_day_rolling_avg']=specific_data['Close'].rolling(window=7).mean()
plt.figure(figsize=(12,6))
plt.plot(specific_data['Date'],specific_data['Close'],label=f'{specific_company} Closing Price',color='blue')
plt.plot(specific_data['Date'],specific_data['7_day_rolling_avg'],label=f'{specific_company} 7 day rolling avg of closing price',col
plt.xlabel = ('Date')
plt.ylabel = ('Closing Price')
plt.title(f'{specific_company} 7 day rolling avg of closing price')
plt.grid()
plt.legend()
plt.xticks(rotation=45)
plt.show()
```