Assignment: Stock Market Analysis

Part 1: Data Cleaning and Exploration

Question 1:

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

Solution:

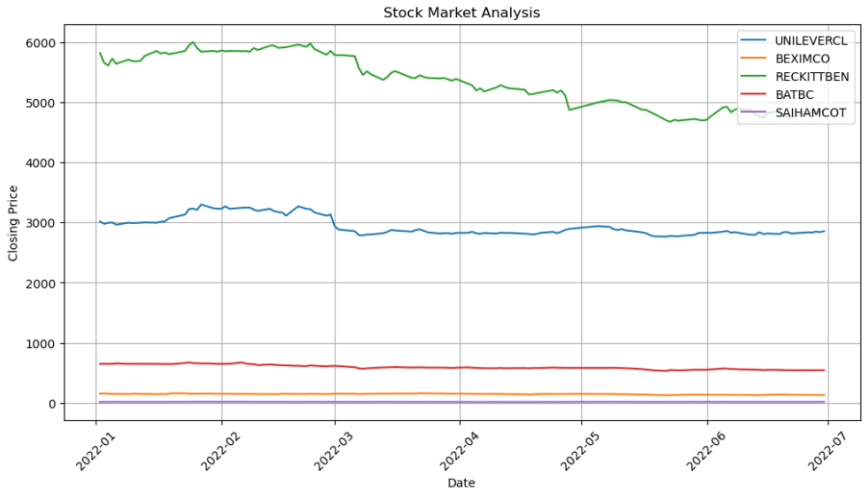
```
specific_data['Open'].mean()
specific_data['High'].mean()
specific_data['Low'].mean()
specific_data['Close'].mean()
specific_data['Volume'].mean()

specific_data['Open'].median()
specific_data['High'].median()
specific_data['Low'].median()
specific_data['Close'].median()
specific_data['Volume'].median()

specific_data['Open'].std()
specific_data['High'].std()
specific_data['Low'].std()
specific_data['Close'].std()
specific_data['Volume'].std()
```

Question 2:

Explore the distribution of the 'Close' prices over time

Solution:

```
plt.figure(figsize=(12,6))
for company in specific_companies:
company_data = specific_data[specific_data['Name']==company]
plt.plot(company_data['Date'], company_data['Close'], label = company)
#Adding Labels
plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.title('Stock Market Analysis')
plt.legend()
plt.grid()

plt.xticks(rotation=45)
plt.show()
```



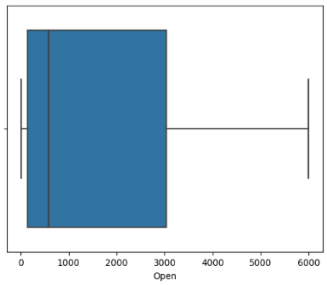
Question 3:

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

Solution:

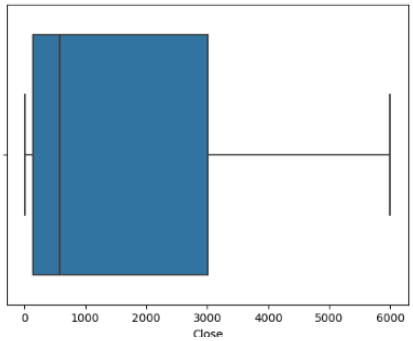
To Detect Outlier we can use boxplot method:

```
[30]: #To See Outlier Clearly Use BoxPlot
      sns.boxplot(x=specific_data['Open'])

[30]: <Axes: xlabel='Open'>
```



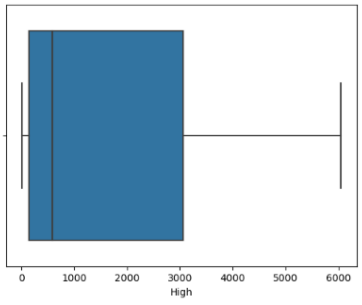
No Outlier Detected in Open Column

```
[37]: #To See Outlier Clearly Use BoxPlot
      sns.boxplot(x=specific_data['Close'])

[37]: <Axes: xlabel='Close'>
```



No Outlier Detected in Close Column

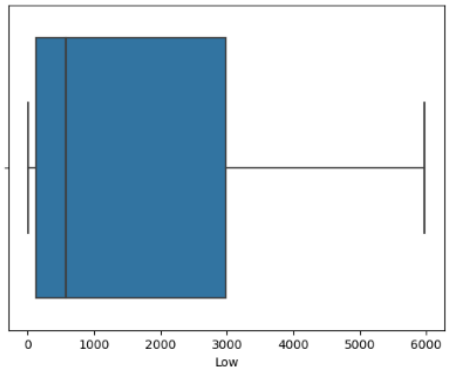
```
[38]: #To See Outlier Clearly Use BoxPlot
      sns.boxplot(x=specific_data['High'])

[38]: <Axes: xlabel='High'>
```



No Outlier Detected in High Column

```
[39]: #To See Outlier Clearly Use BoxPlot
      sns.boxplot(x=specific_data['Low'])
```

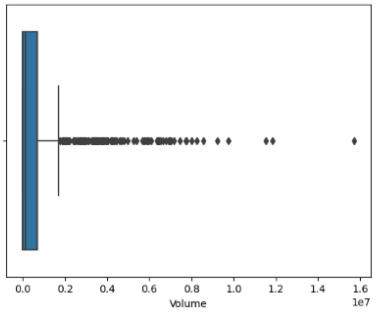
```
[39]: <Axes: xlabel='Low'>
```



No Outlier Detected in Low Column

```
[40]: #To See Outlier Clearly Use BoxPlot
      sns.boxplot(x=specific_data['Volume'])
```

```
[40]: <Axes: xlabel='Volume'>
```



Outlier Detected in Volume Column

#Finding The Outliers

upper_limit = specific_data['Volume'].mean() + 3 * specific_data['Volume'].std()

lower_limit = specific_data['Volume'].mean() - 3 * specific_data['Volume'].std()

specific_data.loc[(specific_data['Volume'] > upper_limit) | (specific_data['Volume'] < lower_limit)]

[48]:
```
# Finding The Outliers
specific_data.loc[(specific_data['Volume'] > upper_limit) | (specific_data['Volume'] < lower_limit)]
```

[48]:

	Date	Name	Open	High	Low	Close	Volume
1829	2022-01-03	BEXIMCO	155.4	159.9	154.6	156.9	11519937.0
1832	2022-01-06	BEXIMCO	151.1	152.0	149.0	149.9	7455503.0
1835	2022-01-11	BEXIMCO	152.3	155.5	150.5	151.6	8010173.0
1841	2022-01-19	BEXIMCO	145.0	152.7	144.0	151.7	11856036.0
1842	2022-01-20	BEXIMCO	153.2	161.9	153.2	160.4	15724293.0
1843	2022-01-23	BEXIMCO	162.3	164.0	155.0	158.2	9742004.0
10467	2022-02-16	BEXIMCO	145.4	153.9	145.3	150.8	9215324.0
18357	2022-03-01	BEXIMCO	149.0	154.0	148.5	152.5	7717890.0
18358	2022-03-02	BEXIMCO	153.9	155.6	151.7	152.5	7181916.0
18371	2022-03-22	BEXIMCO	155.2	161.0	155.0	160.4	8563022.0
18372	2022-03-23	BEXIMCO	161.7	162.8	157.8	159.2	7766909.0
42099	2022-06-19	BEXIMCO	136.5	140.0	135.0	138.2	8246041.0

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

Question 1:

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

Solution:

```
stock_data['Date']= pd.to_datetime(stock_data['Date'],dayfirst=True)

plt.figure(figsize=(12,6))

for company in specific_companies:
    company_data = specific_data[specific_data['Name']==company]
    plt.plot(company_data['Date'], company_data['Close'], label = company)


plt.xlabel('Date')

plt.ylabel('Close')

plt.title('Stock Market Analysis')

plt.legend()

plt.grid()
```

Question 2:

Calculate and plot the daily percentage change in closing prices

Solution:

```
# Calculate daily percentage change
```

```
specific_data['Daily_Return'] = specific_data['Close'].pct_change() * 100

# Plot the daily percentage change
plt.figure(figsize=(10, 6))


for company in specific_companies:
    company_data = specific_data[specific_data['Name']==company]
    plt.plot(company_data['Date'], company_data['Daily_Return'], label = company)

plt.title('Daily Percentage Change in Closing Prices')
plt.xlabel('Date')
plt.ylabel('Percentage Change (%)')
plt.grid()
plt.show()

plt.xticks(rotation=45)
plt.show()
```

Question 3:

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

Solution:

```
plt.figure(figsize=(12,6))
for company in specific_companies:
    company_data = specific_data[specific_data['Name']==company]
    plt.plot(company_data['Date'], company_data['Close'], label = company)
```

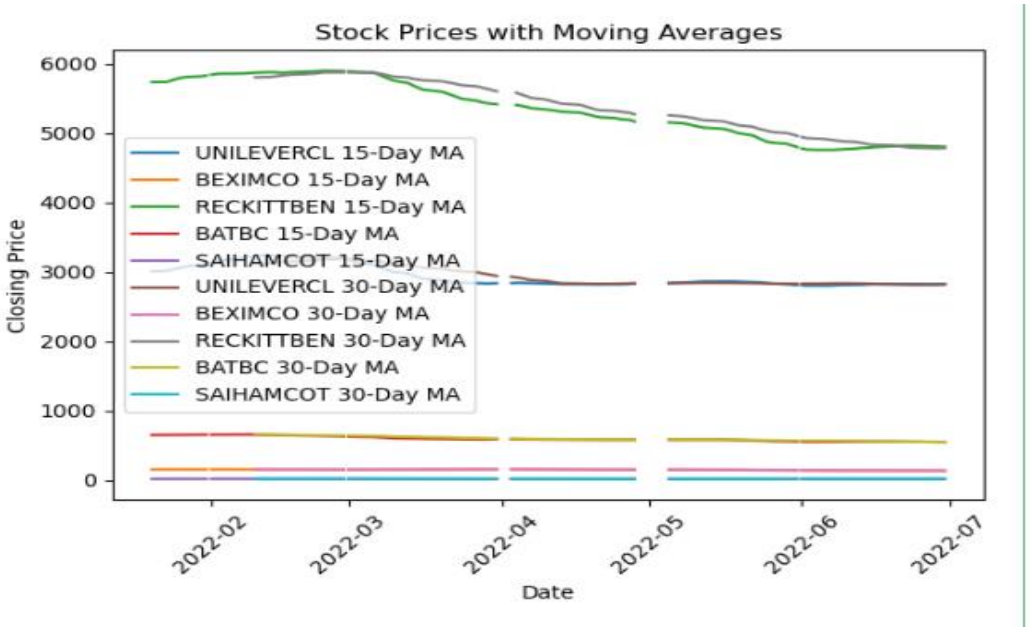
```
plt.title('Stock Prices Trends')
plt.xlabel('Date')
plt.ylabel('Price')
plt.grid()
plt.show()

window_size = 30
specific_data['Rolling_Mean'] = specific_data['Close'].rolling(window=window_size).mean()
specific_data['Rolling_STD'] = specific_data['Close'].rolling(window=window_size).std()

plt.plot(specific_data['Date'], specific_data['Close'], label='Close Price')
plt.plot(specific_data['Date'], specific_data['Rolling_Mean'], label='Rolling Mean', color='red')
plt.plot(specific_data['Date'], specific_data['Rolling_STD'], label='Rolling Std', color='green')
plt.title('Stock Prices with Rolling Mean and Standard Deviation')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
# seaborn's regplot
plt.figure(figsize=(10, 6))
sns.regplot(x=np.arange(len(specific_data)), y='Close', data=specific_data, lowess=True,
line_kws={"color": "red"})
plt.title('Trend of Stock Prices')
plt.xlabel('Time')
plt.ylabel('Price')
plt.show()
```

Question 4:

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

Solution:

```
for days in [15, 30]:
    for company in specific_companies:
        specific_data[f'{company}_MA{days}'] = specific_data[specific_data['Name'] == company]['Close'].rolling(window=days).mean()
        plt.plot(specific_data['Date'], specific_data[f'{company}_MA{days}'], label=f'{company} {days}-Day MA')

plt.title('Stock Prices with Moving Averages')
plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Question 5:

Calculate the average closing price for each stock.

Solution:

```
average_closing_prices = specific_data.groupby('Name')['Close'].mean()
print("Average Closing Prices:")
print(average_closing_prices)
```

Question 6:

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

Solution:

```
average_closing_prices = specific_data.groupby('Name')['Close'].mean()
bottom_5_stocks = average_closing_prices.sort_values().head(5)
top_5_stocks = average_closing_prices.sort_values(ascending=False).head(5)
print("Top 5 Stocks based on Average Closing Price:")
print(top_5_stocks)
print("\nBottom 5 Stocks based on Average Closing Price:")
print(bottom_5_stocks)
```

Part 3: Volatility Analysis

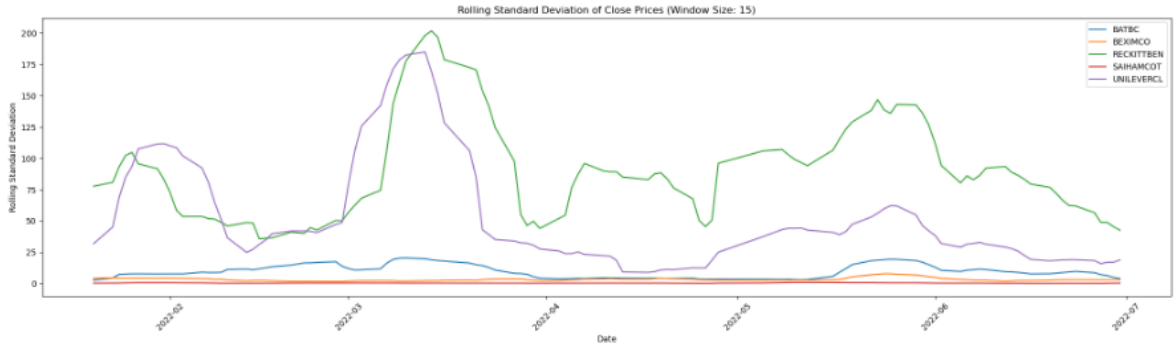
Question 1:

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

Solution:

```
window_size = 15
# Calculate rolling standard deviation
rolling_std = specific_data.groupby('Name')['Close'].rolling(window=window_size).std()

# Plot rolling standard deviation
```

```
plt.figure(figsize=(20, 6))

for company in specific_data['Name'].unique():

    plt.plot(specific_data[specific_data['Name'] == company]['Date'], rolling_std[company], label=company)

plt.title(f'Rolling Standard Deviation of Close Prices (Window Size: {window_size})')

plt.xlabel('Date')

plt.ylabel('Rolling Standard Deviation')

plt.legend()

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()
```



Question 2:

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

Solution:

```
specific_data['Daily_Price_Change'] = specific_data['Close'] - specific_data['Open']
```

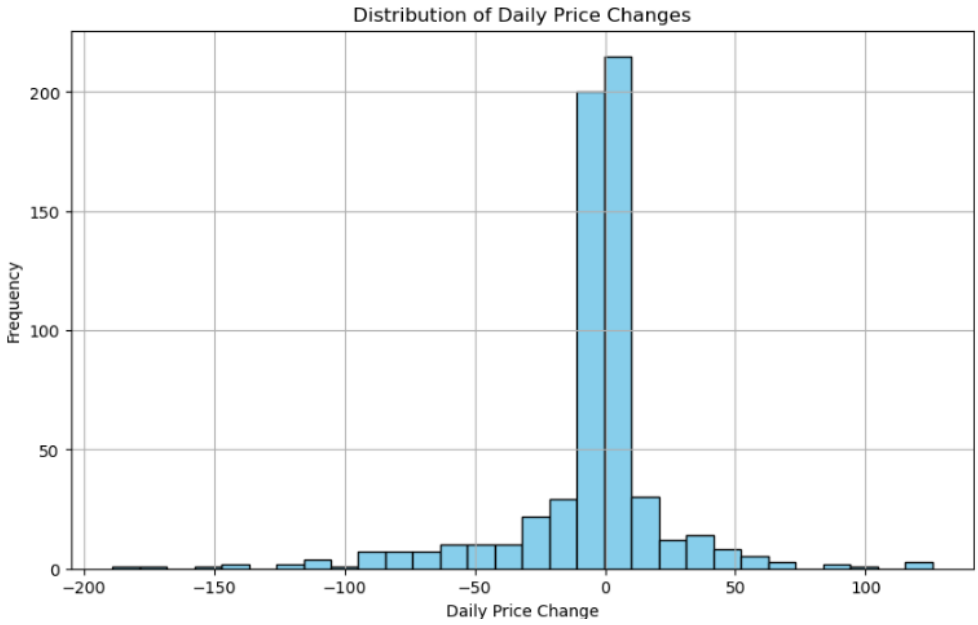
Question 3:

Analyze the distribution of daily price changes

Solution:

```
specific_data['Daily_Price_Change'] = specific_data['Close'] - specific_data['Open']
```

```
# Plot histogram of daily price changes
plt.figure(figsize=(10, 6))
plt.hist(specific_data['Daily_Price_Change'].dropna(), bins=30, color='skyblue', edgecolor='black')
plt.title('Distribution of Daily Price Changes')
plt.xlabel('Daily Price Change')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



Question 4:

Identify days with the largest price increases and decreases.

Solution:

```
largest_increase_days = specific_data.nlargest(5, 'Daily_Price_Change')
print("Days with the largest price increases:")
print(largest_increase_days[['Date', 'Daily_Price_Change']])
```

```
# Identify days with the largest price decreases
largest_decrease_days = specific_data.nsmallest(5, 'Daily_Price_Change')
print("\nDays with the largest price decreases:")
print(largest_decrease_days[['Date', 'Daily_Price_Change']])
```

Question 5:

Identify days with the largest price increases and decreases.

Solution:

```
largest_increase_days = specific_data.nlargest(5, 'Daily_Price_Change')
print("Days with the largest price increases:")
print(largest_increase_days[['Date', 'Daily_Price_Change']])
# Identify days with the largest price decreases
largest_decrease_days = specific_data.nsmallest(5, 'Daily_Price_Change')
print("\nDays with the largest price decreases:")
print(largest_decrease_days[['Date', 'Daily_Price_Change']])
```

Question 6:

Identify stocks with unusually high trading volume on certain days.

Solution:

```
specific_data['Volume_Z_Score'] = (specific_data.groupby('Name')['Volume'].transform(lambda x: (x - x.mean()) / x.std()))

# Define threshold for identifying unusually high trading volume (e.g., z-score > 3)
threshold = 3

# Identify stocks with unusually high trading volume on certain days
```

```
unusually_high_volume_stocks = specific_data[specific_data['Volume_Z_Score'] > threshold]


print("Stocks with unusually high trading volume:")
print(unusually_high_volume_stocks[['Date', 'Name', 'Volume']])
```

Part 3: Correlation and Heatmaps

Question 1:

Explore the relationship between trading volume and volatility

Solution:

```
# Calculate daily price change
specific_data['Daily_Price_Change'] = specific_data['Close'] - specific_data['Open']
# Calculate volatility using standard deviation of daily price changes
volatility = specific_data.groupby('Name')['Daily_Price_Change'].std()
# Calculate correlation between trading volume and volatility
correlation = specific_data.groupby('Name')[['Volume', 'Daily_Price_Change']].corr().iloc[0::2,-1]
print("Correlation between trading volume and volatility:")
print(correlation)
```

Question 2:

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

Solution:

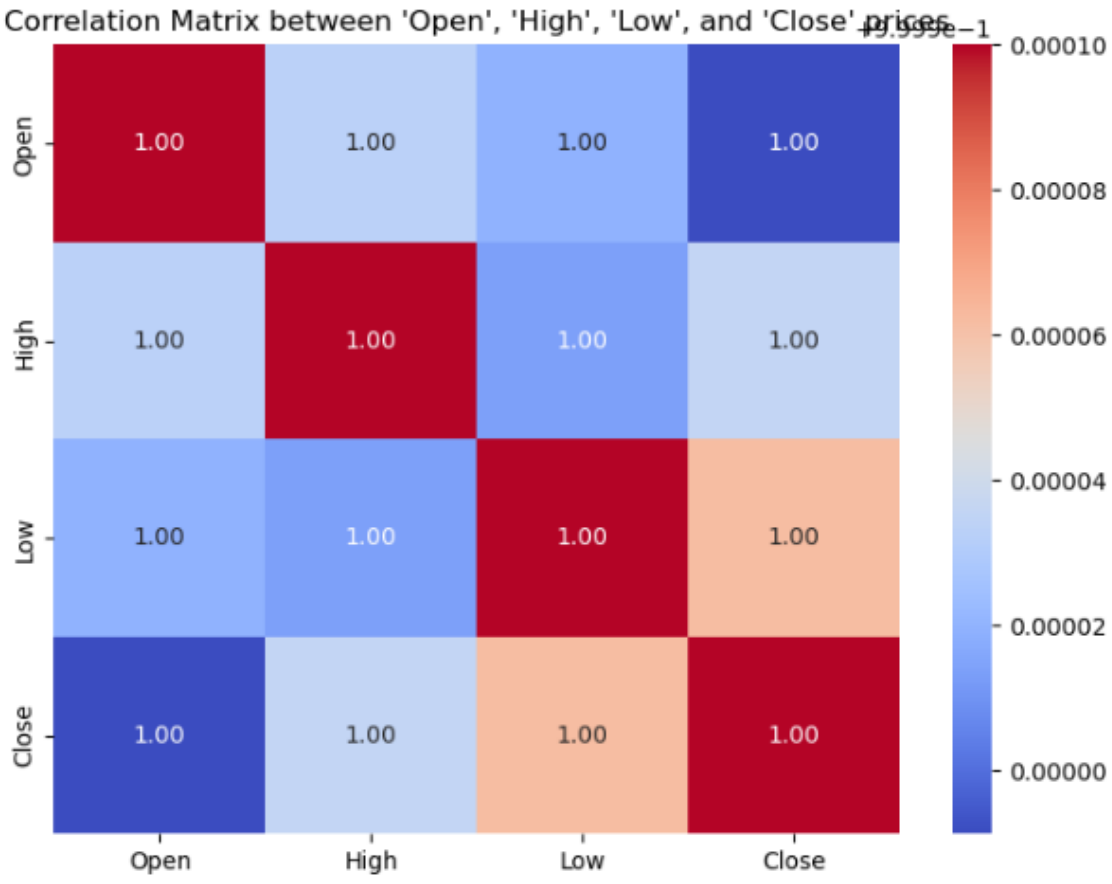
```
price_correlation_matrix = specific_data[['Open', 'High', 'Low', 'Close']].corr()
print("Correlation Matrix between 'Open', 'High', 'Low', and 'Close' prices:")
print(price_correlation_matrix)
```

Question 3:

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

Solution:

```
price_correlation_matrix = specific_data[['Open', 'High', 'Low', 'Close']].corr()


# Create a heatmap to visualize the correlations
plt.figure(figsize=(8, 6))
sns.heatmap(price_correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix between 'Open', 'High', 'Low', and 'Close' prices")
plt.show()
```



Bonus Task

Question:

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

Solution:

The warning we encountered is due to attempting to assign a new column to a DataFrame while it is chained with another operation, which is generally not recommended due to potential issues with indexing and setting values.

Here's the corrected code:

```
specific_company = 'RECKITTBEN'

specific_data = stock_data[stock_data['Name']==specific_company].copy()  # Make a copy to avoid SettingWithCopyWarning


specific_data['7_Day_Rolling_AVERAGE'] = 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_AVERAGE'], label=f'{specific_company} 7 Day Rolling Avg of Closing Price', color='red')


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()
```