Kaiqi Zhao

Huiyan Sang

Stat 482

14 April 2024

Analysis of S&P 500 Stock Prices and Market Capitalization

In the ever-evolving landscape of financial markets, understanding stock performance is crucial for investors, analysts, and policymakers. This project proposes an in-depth analysis of the S&P 500 stocks dataset, accessible via Kaggle, to uncover patterns, trends, and predictive indicators that could guide investment strategies and economic forecasts. The S&P 500 is a market-capitalization-weighted index of 500 of the largest publicly traded companies in the U.S. By encompassing these 500 important companies, the S&P 500 serves as a barometer for the overall health of the stock market and, by extension, the U.S. economy.

## 1.1 Motivation and Research Goals:

This project aims to provide an in-depth analysis of the S&P 500 stocks dataset, accessible via Kaggle, to uncover patterns, trends, and indicators that could guide investment strategies and economic forecasts. This will be employed utilizing various statistical and machine-learning techniques including descriptive analysis, correlation analysis, cross-sectional regression, and time series analysis.

## 2.1 Data Set Introduction:

The dataset includes historical stock prices, volumes, and additional financial metrics of the companies listed in the S&P 500 index. The main data features include but are not limited to daily stock prices, volume, and various financial indicators over multiple years. Overall, this comprehensive dataset potentially offers the ability to obtain a granular view of market movements, company performance, and broader economic trends.

## 2.2 Description of Data:

The dataset “sp500_companies” contains 503 rows and 16 columns. Each row contains one of the top 500 companies with 3 possible duplicates given how there is an additional 3 rows when there should only be 500. There are 10 objects and 6 int/float data types among the 16 columns.

```
RangeIndex: 503 entries, 0 to 502
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Exchange             503 non-null    object
 1   Symbol               503 non-null    object
 2   Shortname            503 non-null    object
 3   Longname             503 non-null    object
 4   Sector               503 non-null    object
 5   Industry             503 non-null    object
 6   Currentprice         503 non-null    float64
 7   Marketcap            503 non-null    int64
 8   Ebitda               474 non-null    float64
 9   Revenuegrowth        502 non-null    float64
 10  City                 503 non-null    object
 11  State                483 non-null    object
 12  Country              503 non-null    object
 13  Fulltimeemployees    493 non-null    float64
 14  Longbusinesssummary  503 non-null    object
 15  Weight               503 non-null    float64
dtypes: float64(5), int64(1), object(10)
memory usage: 63.0+ KB
```

**Exchange** is where the stocks are negotiated.

**Symbol** is the stock symbol.

**Shortname** is the company’s short name.

**Longname** is the company’s long name.

**Sector** is the area in the economy where the company operates (Technology, Industrials, Healthcare).

**Industry** is the area within a sector where the company operates. Group of companies that provide similar products (Credit Services, Beverages Non-Alcoholic, Software Application).

**Currentprice** is the current stock price.

**Marketcap** is the current market cap, the total value of a company’s outstanding shares of stock.

**Ebitda** is the companies earnings before interest, taxes, depreciation, and amortization.

**Revenue Growth** is the increase in a company's total revenue or income over a specific period

**City, State,** and **Country** are self-explanatory.

**Fulltimeemployees** is the current number of full-time employees.

**Longbusinesssummary** provides a summary of the company and what they do.

**Weight** is the percentage of participation on the S&P index (marketcap %).

Another important dataset is the “sp500_stocks” dataset which contains the stock prices of the S&P 500 companies since 2010. This dataset will allow us to utilize a time-series analysis to predict the prices of future stock prices using information from the past 14 years. This dataset contains 2 objects and 6 float datatypes among the 8 columns.

```
RangeIndex: 1795207 entries, 0 to 1795206
Data columns (total 8 columns):
 #   Column     Dtype  
---  ------     -----  
 0   Date       object 
 1   Symbol     object 
 2   Adj Close  float64
 3   Close      float64
 4   High       float64
 5   Low        float64
 6   Open       float64
 7   Volume     float64
dtypes: float64(6), object(2)
memory usage: 109.6+ MB
```

**Date** is the date of the stock prices taken from around the start of 2010 until now. It skips some days but mostly includes every day up until now.

**Symbol** is the company's symbol that will allow us to match with the data in the "sp500_companies" dataset.

**Adj Close** is similar to the price at market closure but also takes into account company actions such as dividends and splits.

**Close** is the price at market closure.

**High** is the maximum value of the period.

**Low** is the minimum value of the period.

**Open** is the price at market opening.

**Volume** is the volume traded.

## 2.3 Statistical Analysis Plan:

The project will employ a variety of statistical and machine-learning techniques to analyze the dataset:

**Descriptive Analysis:** To summarize the dataset's main characteristics with summary statistics and visualization techniques, providing insights into the distribution, volatility, and trends of stock prices and volumes.

**Correlation Analysis:** To identify relationships between stock prices, market cap, weight, EBITDA, full-time employees, and revenue growth.

**Cross-sectional regression:** To understand the factors affecting market capitalization.

**Time Series Analysis:** To examine certain stock price movements and trading volumes over time, employing Arima models to forecast future stock prices based on historical trends.

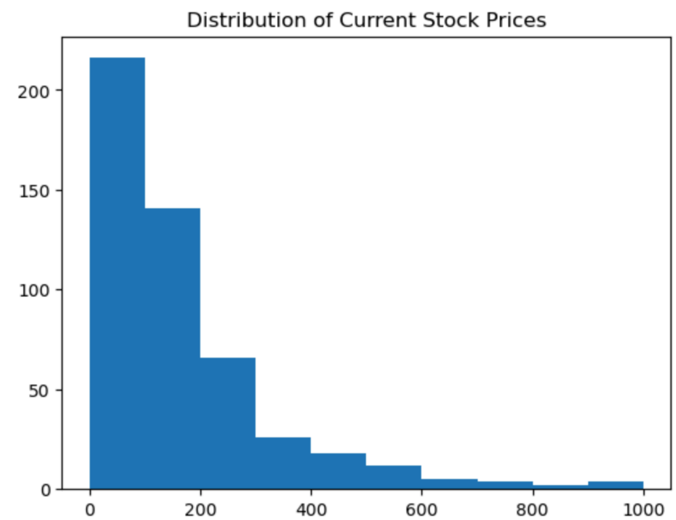
## 3.1 Descriptive Analysis:

The first thing I wanted to look into was which sector had the highest average stock price.

```
Sector
Consumer Cyclical         432.644310
Technology                267.471333
Healthcare                241.879077
Industrials               226.369589
Financial Services        167.450896
Basic Materials           162.104091
Real Estate               131.186129
Communication Services    122.947727
Consumer Defensive        109.142973
Energy                     92.111304
Utilities                  67.302667
Name: Currentprice, dtype: float64
```

The consumer cyclical, technology, and healthcare sectors have the highest average stock prices while energy and utilities have the lowest stock prices.

Here is a histogram showing the overall distribution of stock prices.




The distribution looks right skewed within this range. Overall most stock prices fall between 0 to 400.

Now let's delve into which sectors affect the S&P 500 Stock Market the most. In other words, which sectors have the highest market cap.

```
Sector
Communication Services    2.742586e+11
Technology                1.819918e+11
Financial Services        8.870606e+10
Healthcare                8.651165e+10
Consumer Cyclical         8.507205e+10
Consumer Defensive        7.931366e+10
Energy                    7.182722e+10
Industrials               5.048136e+10
Basic Materials           4.225482e+10
Real Estate               3.363927e+10
Utilities                 3.157316e+10
Name: Marketcap, dtype: float64
```

```
Sector
Communication Services    0.005789
Technology                0.003841
Financial Services        0.001872
Healthcare                0.001826
Consumer Cyclical         0.001796
Consumer Defensive        0.001674
Energy                    0.001516
Industrials               0.001066
Basic Materials           0.000892
Real Estate               0.000710
Utilities                 0.000666
Name: Weight, dtype: float64
```

It seems like communication services, technology, and financial services have the highest average market cap while the basic materials, real estate, and utilities sectors have the lowest average market cap.

Now let's look at the top 10 companies in weight (market cap percentage) among these sectors.

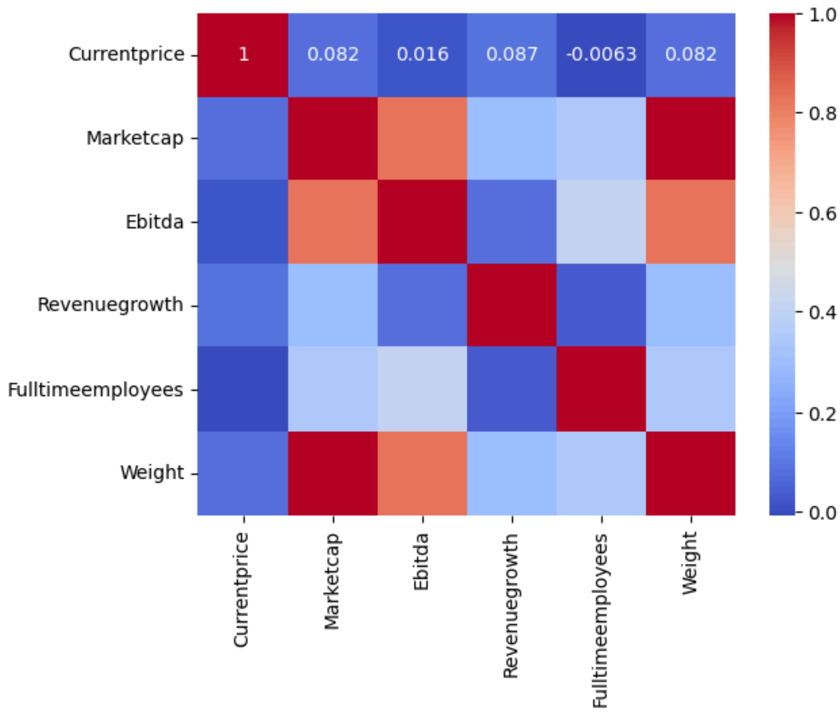
```
                     Longname    Weight
                Alphabet Inc.  0.035644
         Meta Platforms, Inc.  0.027227
                Netflix, Inc.  0.005525
       The Walt Disney Company 0.004271
             T-Mobile US, Inc. 0.004108
          Comcast Corporation  0.003570
  Verizon Communications Inc.  0.003506
                    AT&T Inc.  0.002597
  Charter Communications, Inc. 0.000870
         Electronic Arts Inc.  0.000763
```

```
                     Longname    Weight
        Microsoft Corporation  0.063713
                   Apple Inc.  0.055649
           NVIDIA Corporation  0.046189
                Broadcom Inc.  0.012802
Advanced Micro Devices, Inc.   0.007073
           Oracle Corporation  0.006523
             Salesforce, Inc.  0.006251
                   Adobe Inc.  0.005270
                Accenture plc  0.005005
          Cisco Systems, Inc.  0.004231
```

Here in the communication services sector, we can see some very recognizable names from Disney, T-Mobile, and AT&T. In the technology sector, we can see some recognizable tech companies such as Microsoft, Apple, and Adobe.

## 4.1 Correlation Analysis

To start, we will use a correlation matrix using the seaborn package to identify any initial correlations between the data's numerical features




Looking at the numerical values in our data, the strongly correlated features are Marketcap and Ebitda, Weight and Ebitda, and Marketcap and Weight. Ebitda as described in the description of the data above is the company's earnings before interest, taxes, depreciation, and amortization. Marketcap, short for market capitalization, is the total value of a company's outstanding common shares owned by stockholders. Weight is the percentage of participation on the S&P index (market cap percentage).

Let's go more into depth about the correlation between these variables.

```
pearson_coeff, p_value = pearsonr(df['Marketcap'], df['Ebitda'])
print(f"Pearson's Correlation Coefficient: {pearson_coeff}, P-value: {p_value}")
```

```
Pearson's Correlation Coefficient: 0.8303060620856115, P-value: 1.5721531564487446e-113
```

```
pearson_coeff, p_value = pearsonr(df['Weight'], df['Ebitda'])
print(f"Pearson's Correlation Coefficient: {pearson_coeff}, P-value: {p_value}")
```

```
Pearson's Correlation Coefficient: 0.8303060620856124, P-value: 1.5721531564471266e-113
```

```
pearson_coeff, p_value = pearsonr(df['Weight'], df['Marketcap'])
print(f"Pearson's Correlation Coefficient: {pearson_coeff}, P-value: {p_value}")
```

```
Pearson's Correlation Coefficient: 1.0, P-value: 0.0
```

Marketcap and weight go hand-in-hand with each other and can practically be treated as the same metric since weight in the context of the dataset is the marketcap percentage of the company. Therefore it makes sense how the correlation coefficient is 1.0. Now when examining the relationship between market cap and EBITDA, we can be confident based on the extremely low p-value that there is a strong linear relationship between these variables.

## 5.1 Cross-Sectional Regression of Market Capitalization

Now let us run the same model for Market Capitalization to see if the features ‘Ebitda’, ‘Fulltimeemployees’, and ‘Revenuegrowth’ affect a company’s ‘Marketcap’.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              Marketcap   R-squared:                       0.744
Model:                            OLS   Adj. R-squared:                  0.742
Method:             Least Squares   F-statistic:                     422.3
Date:              Sun, 14 Apr 2024   Prob (F-statistic):          1.07e-128
Time:                        20:34:05   Log-Likelihood:                -11924.
No. Observations:                 441   AIC:                         2.386e+04
Df Residuals:                     437   BIC:                         2.387e+04
Df Model:                           3
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const             -1.876e+10   7.26e+09     -2.583      0.010    -3.3e+10   -4.49e+09
Ebitda               15.3420      0.504     30.424      0.000      14.351      16.333
Fulltimeemployees -8278.2425   5.03e+04     -0.164      0.869   -1.07e+05    9.07e+04
Revenuegrowth      3.035e+11   3.16e+10      9.603      0.000    2.41e+11    3.66e+11
==============================================================================
Omnibus:                      158.314   Durbin-Watson:                   1.510
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            24954.163
Skew:                           0.165   Prob(JB):                         0.00
Kurtosis:                      39.850   Cond. No.                     7.63e+10
==============================================================================
```

Once again, ‘Fulltimeemployees’ seems to not possess a relationship with our variable of interest given the high p-value of 0.869. However, the other two variables of interest ‘Ebitda’ and ‘Revenuegrowth’ seem to possess a very strong relationship with Market Capitalization.

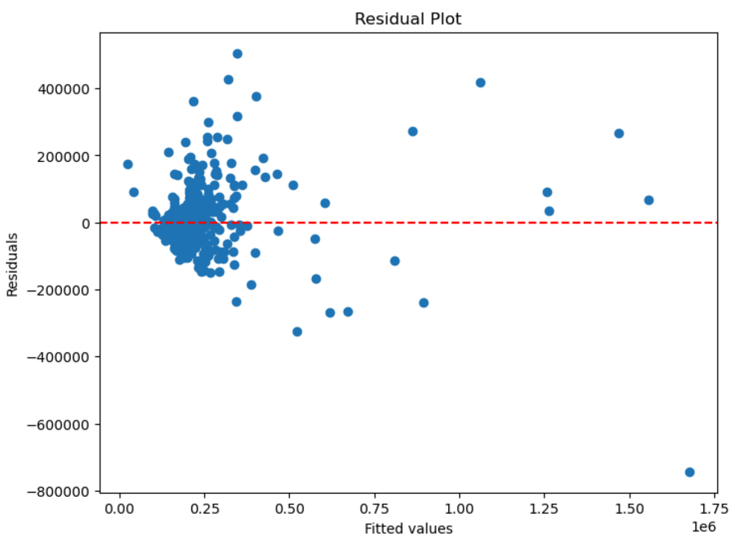
```
const: p-value = 1.01e-02
Ebitda: p-value = 5.59e-110
Fulltimeemployees: p-value = 8.69e-01
Revenuegrowth: p-value = 6.13e-20
```

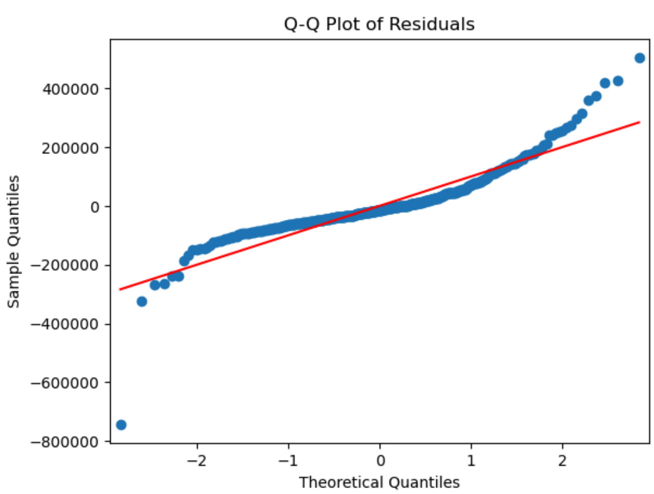
The specific p-values as shown above for ‘Ebitda’ and ‘Revenuegrowth’ are extremely low and we can be very confident that they possess a relationship with a company's market capitalization.

The R-squared and the adjusted R-squared are 0.744 and 0.742 respectively showing that our model does seem to have a pretty accurate fit.

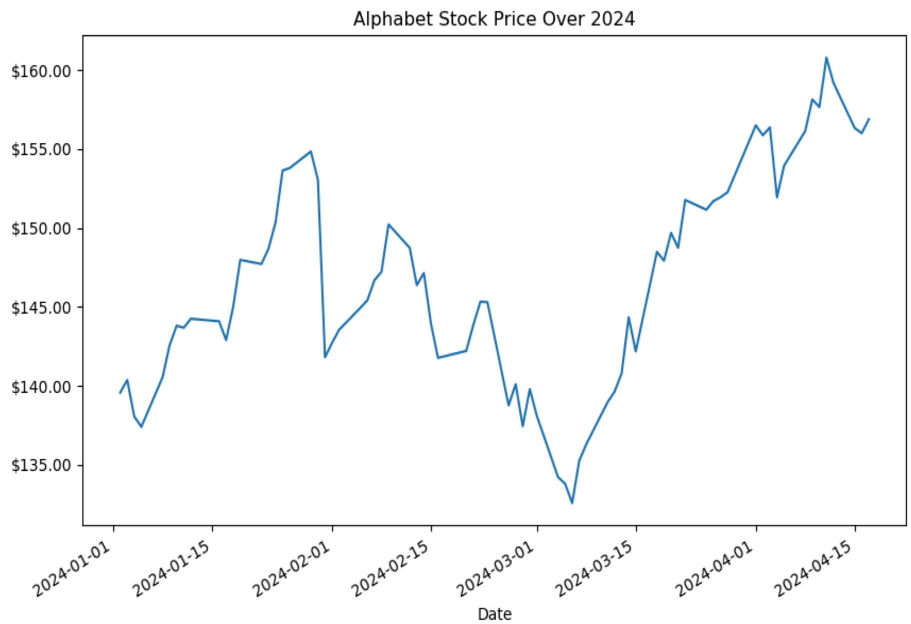
## 5.2 Model Diagnostics

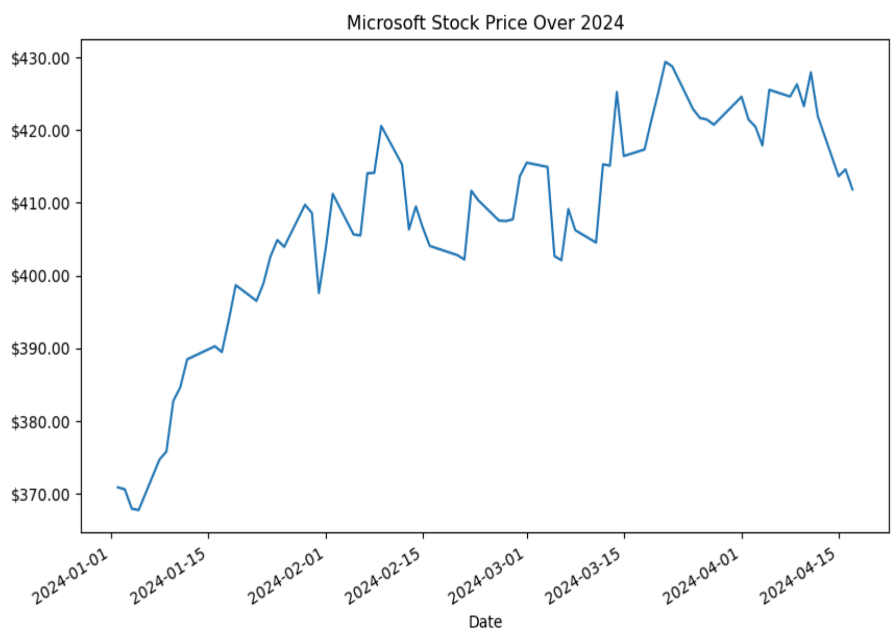
The diagnostics of our model show that the residuals do not show any visible pattern showing the model's predictions are not biased and the q-q plot generally follows the line showing our residuals are normally distributed.



## 6.1 Stock Price Movements Over 2024




First, let's see how the stock prices of these two companies have moved throughout the year so far. For the alphabet or Google stock price in 2024, it seems that the price went up for a bit and then fell only to increase again. The Microsoft stock price seems to start steadily increasing throughout the year and then stabilize and show no clear direction on where it is headed.

## 6.2 Arima Model

For the Arima model I used, I set p to 30 to model based on the past 30 days of what the stock prices were in each iteration. This would give me the most accurate forecast result while ensuring AIC or BIC was not too high to prevent potential overfitting. I then determined that the model needed to be differenced once, and I used the error terms of one past observation.

$$\nabla Y_t = c + \phi_1 \nabla Y_{t-1} + \phi_2 \nabla Y_{t-2} + \ldots + \phi_{30} \nabla Y_{t-30} + \theta_1 \epsilon_{t-1} + \epsilon_t$$

For Alphabet (Google)

```
ADF Statistic after differencing: -8.603674
p-value after differencing: 0.000000
Critical Value (1%): -3.524624
Critical Value (5%): -2.902607
Critical Value (10%): -2.588679
```

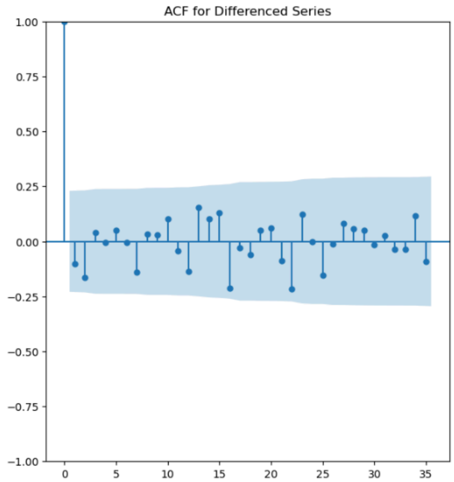
```
ADF Statistic: -1.377000
p-value: 0.593252
```

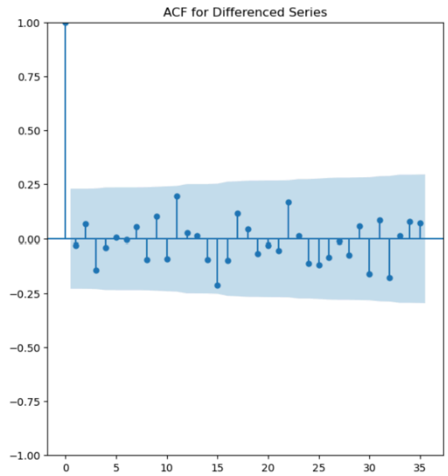
For Microsoft

```
ADF Statistic after differencing: -7.348833
p-value after differencing: 0.000000
Critical Value (1%): -3.526005
Critical Value (5%): -2.903200
Critical Value (10%): -2.588995
```

```
ADF Statistic: -2.506460
p-value: 0.113910
```

For my time series analysis, I used historical stock price data since the start of 2024 so that the period of time would not be too volatile for our model and be more stationary. That being said, the data still needed to be differenced once. As you can see, after differencing the data once, the p-values lowered greatly, and the ADF statistics were lower than the critical values.





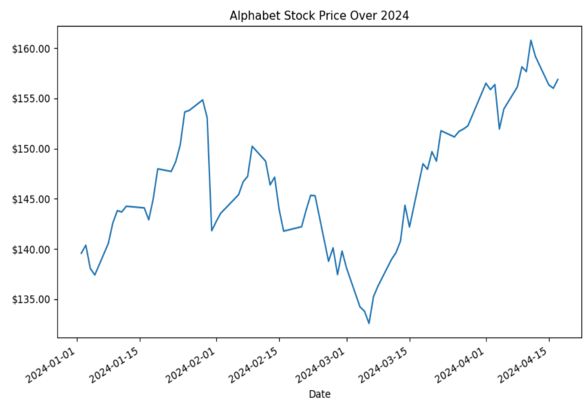

According to the ACF charts which are used to help determine the q for the model. There is no significant lag beyond 0 at 1 so I decided to use the error terms of one past observation.

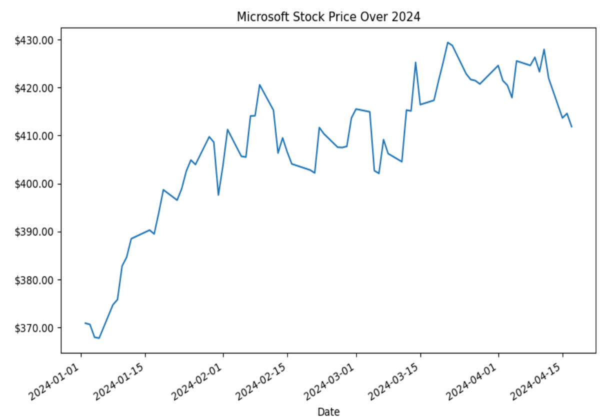
## 6.3 Model Results

Here is the forecast of the next 14 days of stock prices for Google and Microsoft from my model. On the left is the forecast for Alphabet (Google) and on the right is the forecast for Microsoft.

| | |
|---|---|
| Day 1: $157.32 | Day 1: $412.14 |
| Day 2: $154.06 | Day 2: $414.62 |
| Day 3: $151.33 | Day 3: $411.71 |
| Day 4: $148.59 | Day 4: $410.63 |
| Day 5: $148.63 | Day 5: $413.82 |
| Day 6: $148.48 | Day 6: $415.24 |
| Day 7: $146.84 | Day 7: $421.76 |
| Day 8: $145.82 | Day 8: $417.02 |
| Day 9: $141.62 | Day 9: $418.87 |
| Day 10: $142.77 | Day 10: $415.95 |
| Day 11: $141.26 | Day 11: $416.52 |
| Day 12: $141.97 | Day 12: $417.21 |
| Day 13: $140.76 | Day 13: $416.48 |
| Day 14: $141.42 | Day 14: $415.25 |

If you look at Google, there seems to be a decline in the stock price over the next 14 days and for Microsoft, the stock price movements have less of a clear pattern on where they are headed.

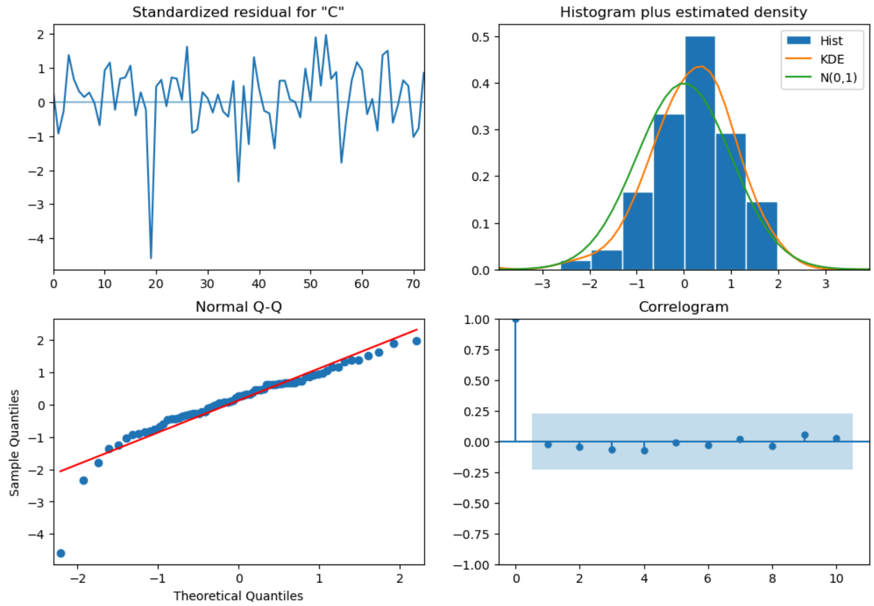


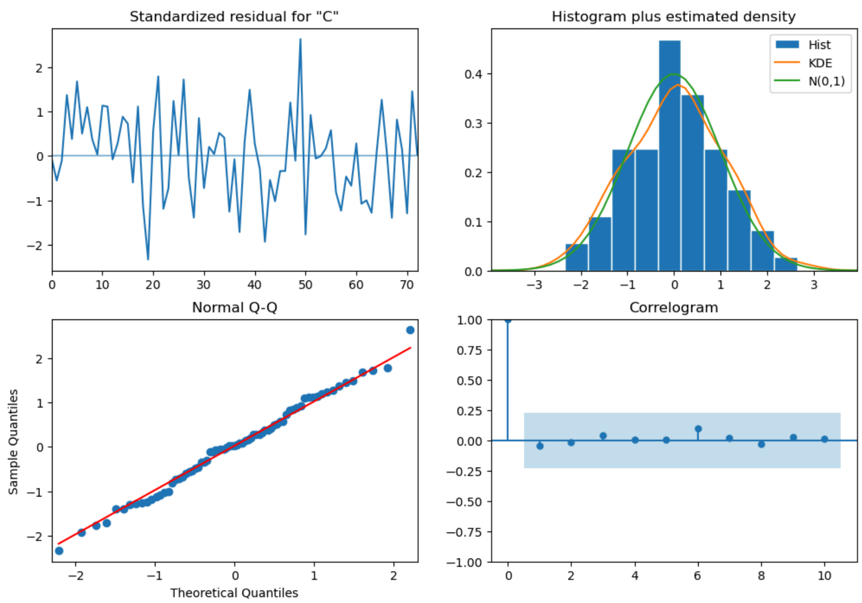

Lets compare with how this forecast aligns with the previous stock prices of 2024. The stock prices for Google would increase and dip only to increase again. In this case, the model potentially captured a pattern that there would be a dip after the period of increase around the start of March. For the stock price for Microsoft where it stabilizes more after a period of increase, the model is forecasting that the price will continue to be relatively stable and not head in any clear direction. From this, someone could potentially want to sell their stock for Google or wait for the potential dip to buy more. However, let's look at how reliable these predictions are.

## 6.4 Model Diagnostics

For Alphabet (Google)




For Microsoft




The diagnostic shows that when run for both companies, the model's residuals do not show any visible pattern showing the model's predictions are not biased and the q-q plots and histograms indicate a normal distribution. The correlogram shows no autocorrelation beyond the interval indicating the model is unbiased in its parameter estimates and values are independent. However, when we look at the significance of the p-lags, for the model, it's mixed in how much evidence and confidence we can have of the previous days being able to forecast future prices.

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                   74
Model:                ARIMA(30, 1, 1)   Log Likelihood                -153.662
Date:                Sun, 21 Apr 2024   AIC                            371.324
Time:                        00:18:42   BIC                            444.619
Sample:                             0   HQIC                           400.533
                                 - 74
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.3472      0.437     -0.795      0.426      -1.203       0.508
ar.L2          0.0249      0.211      0.118      0.906      -0.388       0.438
ar.L3          0.0024      0.276      0.009      0.993      -0.538       0.543
ar.L4         -0.1128      0.189     -0.598      0.550      -0.482       0.257
ar.L5         -0.1810      0.249     -0.728      0.467      -0.668       0.306
ar.L6         -0.0940      0.266     -0.354      0.723      -0.614       0.426
ar.L7          0.1437      0.190      0.758      0.448      -0.228       0.515
ar.L8          0.0684      0.183      0.374      0.708      -0.290       0.427
ar.L9         -0.0379      0.166     -0.228      0.820      -0.364       0.288
ar.L10        -0.2059      0.216     -0.954      0.340      -0.629       0.217
ar.L11         0.0430      0.248      0.173      0.863      -0.444       0.530
ar.L12         0.1178      0.187      0.631      0.528      -0.248       0.484
ar.L13         0.0327      0.261      0.125      0.900      -0.479       0.544
ar.L14        -0.1722      0.200     -0.860      0.390      -0.565       0.220
ar.L15        -0.3458      0.243     -1.421      0.155      -0.823       0.131
ar.L16        -0.1958      0.209     -0.937      0.349      -0.605       0.214
ar.L17         0.1095      0.138      0.795      0.426      -0.160       0.379
ar.L18         0.0618      0.231      0.268      0.789      -0.390       0.514
ar.L19        -0.0456      0.143     -0.319      0.750      -0.325       0.234
ar.L20        -0.1938      0.176     -1.100      0.271      -0.539       0.152
ar.L21        -0.0661      0.233     -0.284      0.776      -0.523       0.390
ar.L22         0.2292      0.249      0.920      0.358      -0.259       0.718
ar.L23         0.1199      0.315      0.381      0.703      -0.497       0.737
ar.L24        -0.2290      0.145     -1.574      0.115      -0.514       0.056
ar.L25        -0.2686      0.252     -1.068      0.286      -0.762       0.224
ar.L26        -0.1084      0.235     -0.461      0.644      -0.569       0.352
ar.L27         0.0333      0.286      0.117      0.907      -0.526       0.593
ar.L28        -0.1645      0.321     -0.513      0.608      -0.793       0.464
ar.L29        -0.2664      0.262     -1.018      0.309      -0.780       0.247
ar.L30        -0.4787      0.242     -1.978      0.048      -0.953      -0.004
ma.L1          0.2367      0.543      0.436      0.663      -0.827       1.300
sigma2         3.1737      1.173      2.706      0.007       0.875       5.473
===================================================================================
Ljung-Box (L1) (Q):                   0.04   Jarque-Bera (JB):               124.80
Prob(Q):                              0.84   Prob(JB):                         0.00
Heteroskedasticity (H):               0.79   Skew:                            -1.56
Prob(H) (two-sided):                  0.56   Kurtosis:                         8.60
===================================================================================
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                   74
Model:                 ARIMA(30, 1, 1)   Log Likelihood                -200.242
Date:                 Sun, 21 Apr 2024   AIC                            464.484
Time:                         00:18:19   BIC                            537.779
Sample:                              0   HQIC                           493.694
                                  - 74
Covariance Type:                   opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.5505      0.307      1.792      0.073      -0.052       1.153
ar.L2         -0.1264      0.227     -0.556      0.578      -0.572       0.319
ar.L3          0.1477      0.373      0.396      0.692      -0.584       0.879
ar.L4         -0.1487      0.411     -0.362      0.718      -0.954       0.657
ar.L5          0.1595      0.333      0.479      0.632      -0.494       0.813
ar.L6         -0.0426      0.276     -0.154      0.877      -0.584       0.499
ar.L7         -0.1015      0.223     -0.455      0.649      -0.538       0.335
ar.L8          0.3634      0.246      1.479      0.139      -0.118       0.845
ar.L9         -0.0750      0.236     -0.317      0.751      -0.538       0.388
ar.L10         0.1646      0.229      0.718      0.473      -0.285       0.614
ar.L11        -0.0935      0.233     -0.401      0.688      -0.551       0.364
ar.L12         0.0510      0.264      0.193      0.847      -0.467       0.569
ar.L13         0.2395      0.297      0.806      0.420      -0.343       0.822
ar.L14        -0.0383      0.273     -0.140      0.888      -0.573       0.497
ar.L15         0.1442      0.202      0.715      0.474      -0.251       0.539
ar.L16        -0.4295      0.212     -2.030      0.042      -0.844      -0.015
ar.L17         0.1595      0.270      0.590      0.555      -0.370       0.689
ar.L18        -0.2969      0.200     -1.482      0.138      -0.690       0.096
ar.L19         0.2281      0.273      0.837      0.403      -0.306       0.762
ar.L20        -0.0251      0.330     -0.076      0.939      -0.672       0.622
ar.L21        -0.1631      0.339     -0.481      0.631      -0.828       0.502
ar.L22        -0.2428      0.387     -0.628      0.530      -1.001       0.515
ar.L23         0.0932      0.428      0.218      0.828      -0.746       0.932
ar.L24         0.0175      0.485      0.036      0.971      -0.934       0.969
ar.L25        -0.1876      0.478     -0.393      0.694      -1.124       0.749
ar.L26         0.1932      0.490      0.395      0.693      -0.767       1.153
ar.L27         0.0828      0.449      0.184      0.854      -0.797       0.963
ar.L28        -0.1002      0.331     -0.302      0.762      -0.749       0.549
ar.L29         0.1290      0.352      0.366      0.714      -0.562       0.820
ar.L30         0.2637      0.324      0.814      0.416      -0.372       0.899
ma.L1         -0.7317      0.296     -2.471      0.013      -1.312      -0.151
sigma2        11.5035      3.574      3.219      0.001       4.499      18.508
===================================================================================
Ljung-Box (L1) (Q):                   0.15   Jarque-Bera (JB):                 0.35
Prob(Q):                              0.70   Prob(JB):                         0.84
Heteroskedasticity (H):               1.06   Skew:                            -0.00
Prob(H) (two-sided):                  0.88   Kurtosis:                         2.66
===================================================================================
```

## 6.6 Model Limitations

As we saw previously, using only historical data for stock prices, we cannot be too certain about our forecasts. As anyone who invests in stocks knows, many factors determine stock prices. An Arima time series analysis can only be one indicator in predicting future stock prices. If you could be confident in future stock prices using historical stock price data alone, it would be way too easy to make money from the stock market as successfully investing in the stock market requires looking at many different factors and indicators. Having historical data on other factors and utilizing other models such as LSTM could be used in future studies to incorporate more factors such as earnings reports or GDP.

## 6.7 Considerations for the Arima Model

When determining how useful an Arima model can be, one may consider it impractical based on theories like the efficient market hypothesis which posits that current stock prices incorporate all important information into current share prices, and the random walk theory, which states that there is no pattern in the stock market and that changes are unpredictable. While it is true that a wide amount of factors determine stock prices, an arima model can still be useful for highlighting underlying trends in stock price movements. Though long-term trends and forecasting are difficult for the Arima model due to how volatile the stock market is, an Arima model can provide reasonably short-term forecasting in stable less volatile market periods. When combined with other indicators such as economic factors and investor sentiment, an Arima model can be one of many useful tools to help guide investment strategies and decision-making.

## 7.1 Final Thoughts and Conclusions

Through this capstone project, I was able to learn about the stock market, the S&P 500, and the Arima time series model that I was previously interested in delving into. In the future, I may test different models such as LSTM neural networks that can utilize more features besides just time series data to produce more accurate forecasts and additional insights. However, the data analysis I have presented has shown an overview of the S&P 500 and the sectors that contributed most to the market cap which were the technology and communication sectors. The predictions of the time series analysis should be taken with a grain of salt but could still be used as one among many indicators for investors. In the end, though the stock market may oftentimes be unpredictable and random, utilizing statistical methods and data analysis can strengthen investment strategies and reduce risk in working with the stock market.

Report Ends at Page 20. The following pages is the code/appendix

# Appendix

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from scipy import stats
from scipy.stats import pearsonr, spearmanr, kendalltau
```

# Data Preprocessing

In [2]:
```
df = pd.read_csv('sp500_companies.csv')
```

In [3]:
```
df.head()
```

Out[3]:

| | Exchange | Symbol | Shortname | Longname | Sector | Industry | Currentp |
|---|---|---|---|---|---|---|---|
| **0** | NMS | MSFT | Microsoft Corporation | Microsoft Corporation | Technology | Software - Infrastructure | 40 |
| **1** | NMS | AAPL | Apple Inc. | Apple Inc. | Technology | Consumer Electronics | 17 |
| **2** | NMS | NVDA | NVIDIA Corporation | NVIDIA Corporation | Technology | Semiconductors | 87 |
| **3** | NMS | AMZN | Amazon.com, Inc. | Amazon.com, Inc. | Consumer Cyclical | Internet Retail | 17 |
| **4** | NMS | GOOG | Alphabet Inc. | Alphabet Inc. | Communication Services | Internet Content & Information | 13 |

In [4]:
```
num_rows = df.shape[0]
print(num_rows)
```

```
503
```

In [5]:
```
for col in df.columns:
    pct_missing = np.mean(df[col].isnull())
    print('{} - {}%'.format(col, pct_missing))
```

```
Exchange - 0.0%
Symbol - 0.0%
Shortname - 0.0%
Longname - 0.0%
Sector - 0.0%
Industry - 0.0%
Currentprice - 0.0%
Marketcap - 0.0%
Ebitda - 0.05765407554671968%
Revenuegrowth - 0.0019880715705765406%
City - 0.0%
State - 0.039761431411530816%
Country - 0.0%
Fulltimeemployees - 0.019880715705765408%
Longbusinesssummary - 0.0%
Weight - 0.0%
```

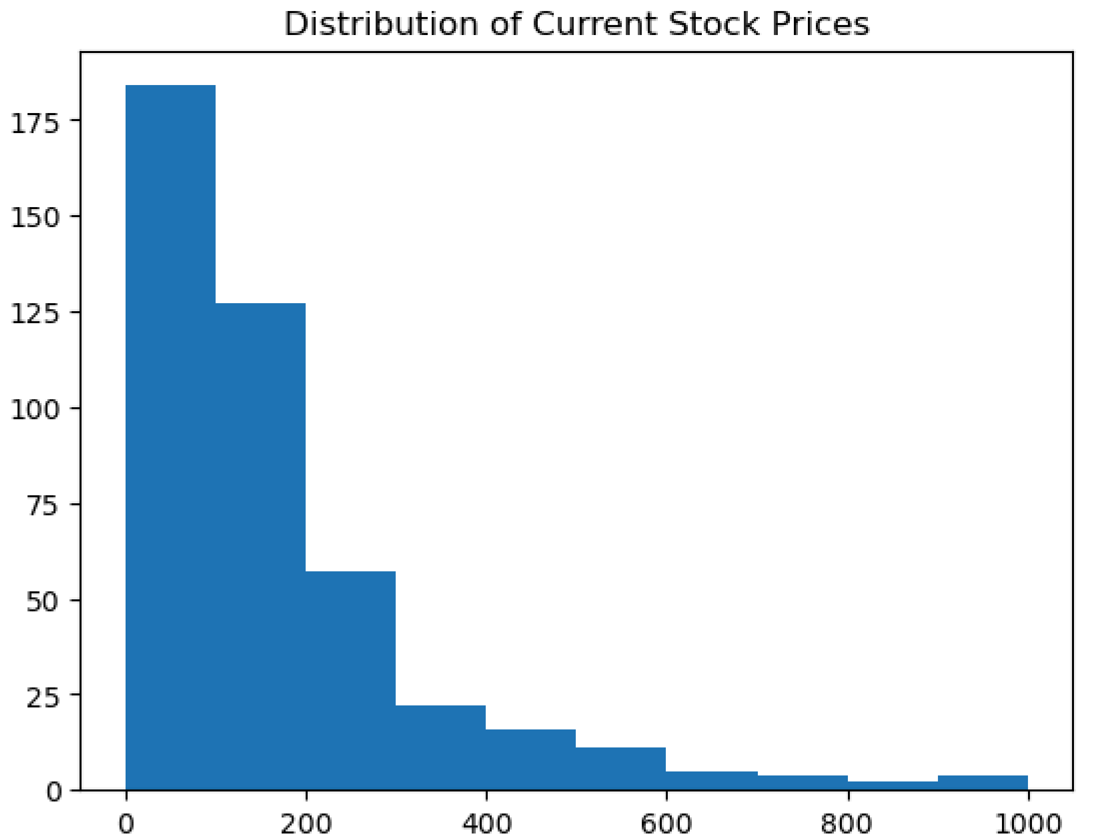
In [6]:
```
original = df
original.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 503 entries, 0 to 502
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Exchange             503 non-null    object 
 1   Symbol               503 non-null    object 
 2   Shortname            503 non-null    object 
 3   Longname             503 non-null    object 
 4   Sector               503 non-null    object 
 5   Industry             503 non-null    object 
 6   Currentprice         503 non-null    float64
 7   Marketcap            503 non-null    int64  
 8   Ebitda               474 non-null    float64
 9   Revenuegrowth        502 non-null    float64
 10  City                 503 non-null    object 
 11  State                483 non-null    object 
 12  Country              503 non-null    object 
 13  Fulltimeemployees    493 non-null    float64
 14  Longbusinesssummary  503 non-null    object 
 15  Weight               503 non-null    float64
dtypes: float64(5), int64(1), object(10)
memory usage: 63.0+ KB
```

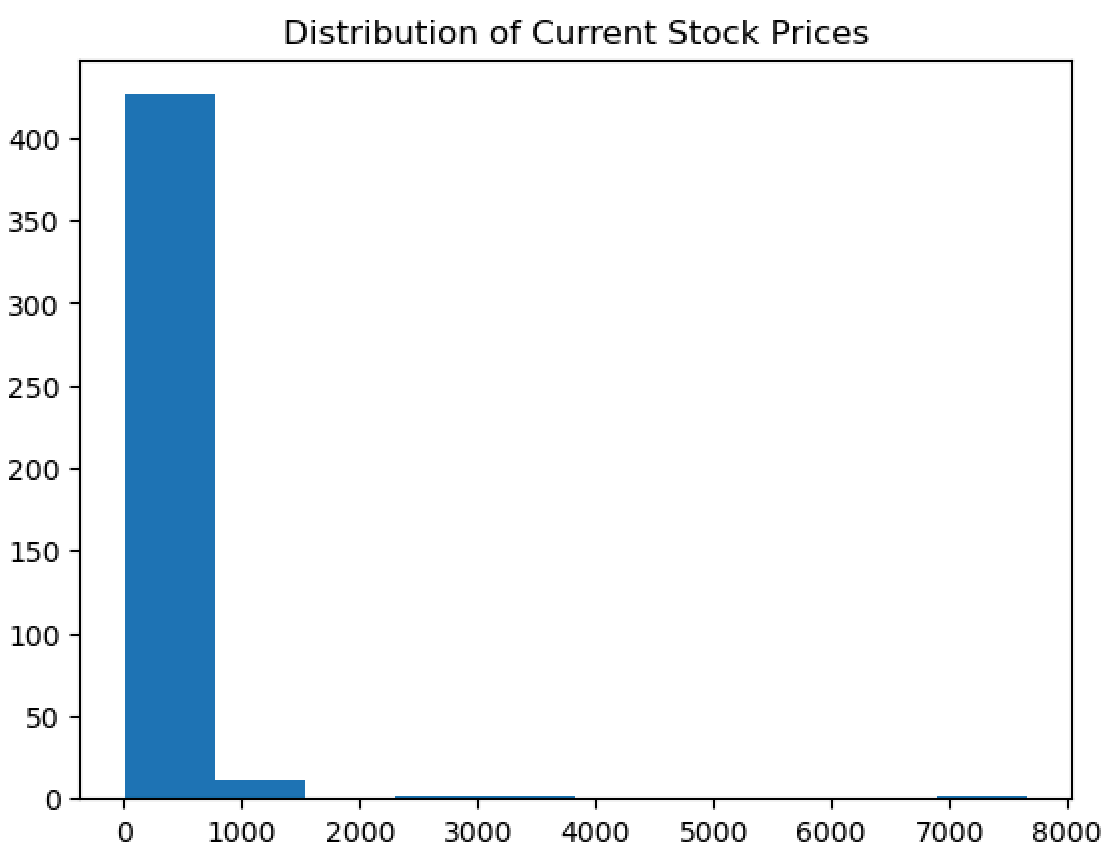
In [7]:
```
df = df.drop_duplicates(subset=['Longname'], keep='first')
df = df.dropna()
```

# Descriptive Analysis (Stock Prices)

In [8]:
```
plt.hist(df['Currentprice'], range=(0, 1000))
plt.title('Distribution of Current Stock Prices')
plt.show()
```




In [9]:
```
plt.hist(df['Currentprice'])
plt.title('Distribution of Current Stock Prices')
plt.show()
```



In [10]:
```python
grouped = df.groupby('Sector')
sector_grouped = grouped['Currentprice']
sector_mean = sector_grouped.mean().sort_values(ascending = False)
print(sector_mean)
```

```
Sector
Consumer Cyclical         473.862115
Technology                271.554925
Healthcare                245.401129
Industrials               230.424118
Financial Services        205.623636
Basic Materials           147.797143
Real Estate               134.909333
Communication Services    132.275263
Consumer Defensive        107.460278
Energy                     92.111304
Utilities                  67.302667
Name: Currentprice, dtype: float64
```

# Descriptive Analysis (Marketcap and Weight)

In [11]:
```python
grouped = df.groupby('Sector')
sector_grouped = grouped['Marketcap']
sector_mean = sector_grouped.mean().sort_values(ascending = False)
print(sector_mean)
```

```
Sector
Communication Services    2.272091e+11
Technology                1.958920e+11
Consumer Cyclical         9.064271e+10
Healthcare                8.805499e+10
Financial Services        8.281085e+10
Consumer Defensive        7.934027e+10
Energy                    7.182722e+10
Industrials               5.050966e+10
Real Estate               3.432341e+10
Basic Materials           3.365967e+10
Utilities                 3.157316e+10
Name: Marketcap, dtype: float64
```

In [12]:
```python
grouped = df.groupby('Sector')
critic_grouped = grouped['Weight']
critic_mean = critic_grouped.mean().sort_values(ascending = False)
print(critic_mean)
```

```
Sector
Communication Services    0.004796
Technology                0.004135
Consumer Cyclical         0.001913
Healthcare                0.001859
Financial Services        0.001748
Consumer Defensive        0.001675
Energy                    0.001516
Industrials               0.001066
Real Estate               0.000725
Basic Materials           0.000710
Utilities                 0.000666
Name: Weight, dtype: float64
```

In [13]:
```
tech_stocks = df[df['Sector'] == 'Technology']
```

In [14]:
```
# Sort by MarketCap in descending order and select the top 10
top_tech_stocks = tech_stocks.sort_values(by='Marketcap', ascending=False).head(10)
```

In [15]:
```
print(top_tech_stocks)
```

```
                     Longname    Weight
0       Microsoft Corporation  0.063713
1                  Apple Inc.  0.055649
2          NVIDIA Corporation  0.046189
9               Broadcom Inc.  0.012802
20  Advanced Micro Devices, Inc.  0.007073
24         Oracle Corporation  0.006523
25           Salesforce, Inc.  0.006251
30                 Adobe Inc.  0.005270
39        Cisco Systems, Inc.  0.004231
41      QUALCOMM Incorporated  0.004018
```

In [16]:
```
comm_stocks = df[df['Sector'] == 'Communication Services']
```

In [17]:
```
# Sort by MarketCap in descending order and select the top 10
top_comm_stocks = comm_stocks.sort_values(by='Marketcap', ascending=False).head(10)
```

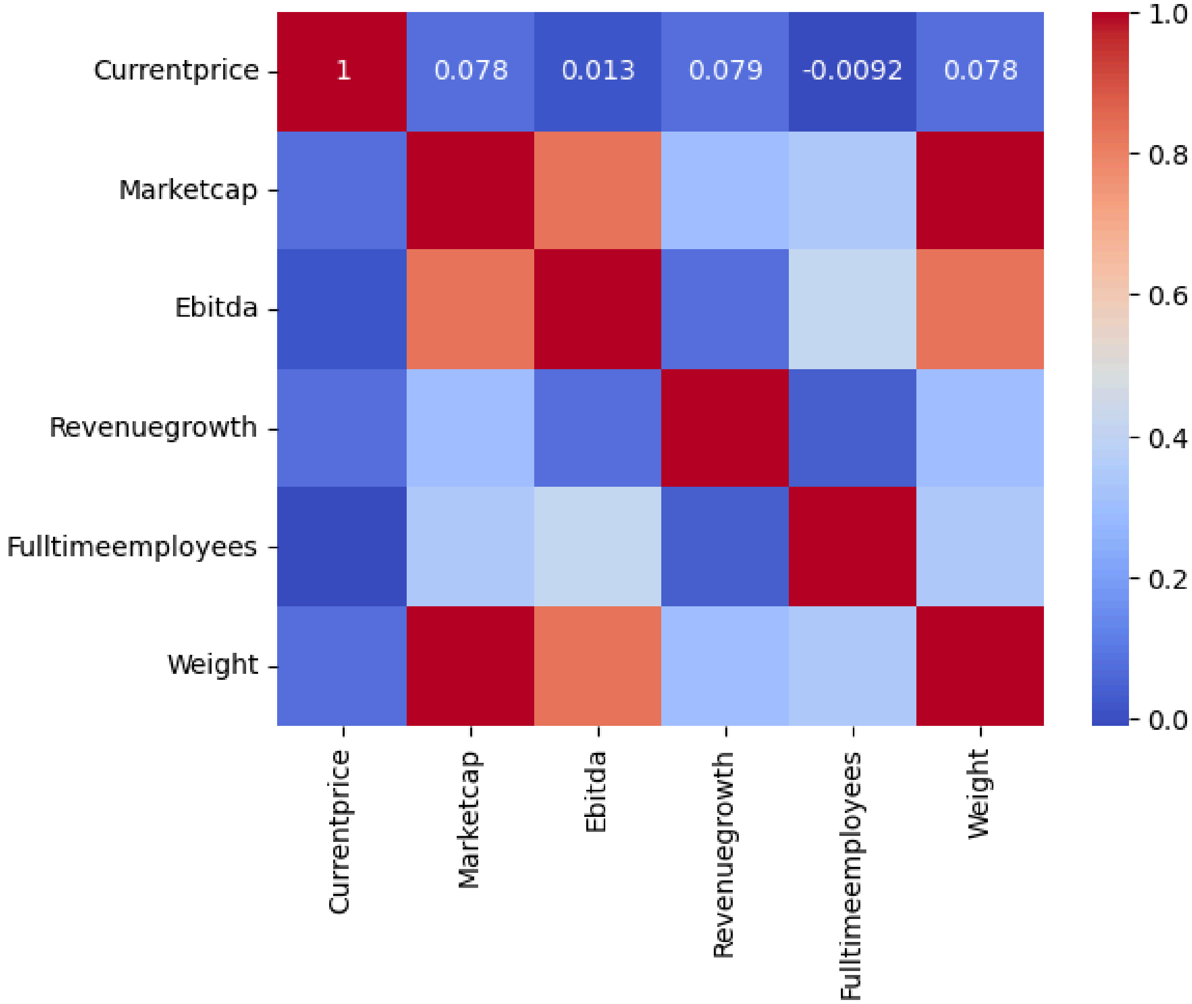
In [18]:
```
print(top_comm_stocks)
```

```
                       Longname    Weight
4                 Alphabet Inc.  0.035644
6           Meta Platforms, Inc.  0.027227
28                Netflix, Inc.  0.005525
38       The Walt Disney Company  0.004271
40            T-Mobile US, Inc.  0.004108
49           Comcast Corporation  0.003570
50   Verizon Communications Inc.  0.003506
73                    AT&T Inc.  0.002597
218  Charter Communications, Inc.  0.000870
244         Electronic Arts Inc.  0.000763
```

# Correlation Analysis

In [19]:
```
df_numerical = df.select_dtypes(include=[np.number])
```

In [20]:
```
corr = df_numerical.corr()

sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.show()
```




In [21]:
```
pearson_coeff, p_value = pearsonr(df['Marketcap'], df['Ebitda'])
print(f"Pearson's Correlation Coefficient: {pearson_coeff}, P-value: {p_value}")
```

```
Pearson's Correlation Coefficient: 0.8303060620856115, P-value: 1.5721531564487446e-
113
```

In [22]:
```
pearson_coeff, p_value = pearsonr(df['Weight'], df['Ebitda'])
print(f"Pearson's Correlation Coefficient: {pearson_coeff}, P-value: {p_value}")
```

```
Pearson's Correlation Coefficient: 0.8303060620856124, P-value: 1.5721531564471266e-
113
```

In [23]:
```
pearson_coeff, p_value = pearsonr(df['Weight'], df['Marketcap'])
print(f"Pearson's Correlation Coefficient: {pearson_coeff}, P-value: {p_value}")
```

```
Pearson's Correlation Coefficient: 1.0, P-value: 0.0
```

# Cross-Sectional Regression (Stock Prices)

In [24]:
```
Y = df['Currentprice']
X = df[['Ebitda', 'Fulltimeemployees', 'Revenuegrowth']]
```

In [25]:
```
# Adding a constant (intercept) to the model
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(Y, X).fit()

# Print out the statistics
print(model.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:           Currentprice   R-squared:                       0.007
Model:                            OLS   Adj. R-squared:                 -0.000
Method:                 Least Squares   F-statistic:                    0.9584
Date:                Sun, 21 Apr 2024   Prob (F-statistic):              0.412
Time:                        22:05:37   Log-Likelihood:                -3340.5
No. Observations:                 441   AIC:                             6689.
Df Residuals:                     437   BIC:                             6705.
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
==================================================================================
=
                        coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------------
-
const               215.1667     25.622      8.398      0.000     164.809     265.52
4
Ebitda             4.839e-10   1.78e-09      0.272      0.786   -3.01e-09    3.98e-0
9
Fulltimeemployees -6.096e-05      0.000     -0.343      0.732      -0.000       0.00
0
Revenuegrowth       182.6611    111.495      1.638      0.102     -36.471     401.79
3
==============================================================================
Omnibus:                      776.666   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           415911.800
Skew:                          10.685   Prob(JB):                         0.00
Kurtosis:                     151.923   Cond. No.                     7.63e+10
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The condition number is large, 7.63e+10. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [26]:
```
Y = df['Currentprice']
X = df[['Revenuegrowth']]
```

In [27]:
```
# Adding a constant (intercept) to the model
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(Y, X).fit()

# Print out the statistics
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Currentprice   R-squared:                       0.006
Model:                            OLS   Adj. R-squared:                  0.004
Method:                 Least Squares   F-statistic:                     2.749
Date:                Sun, 21 Apr 2024   Prob (F-statistic):             0.0980
Time:                        22:05:37   Log-Likelihood:                -3340.6
No. Observations:                 441   AIC:                             6685.
Df Residuals:                     439   BIC:                             6693.
Df Model:                           1
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const           214.9204     22.981      9.352      0.000     169.754     260.087
Revenuegrowth   183.8430    110.888      1.658      0.098     -34.094     401.780
==============================================================================
Omnibus:                      776.546   Durbin-Watson:                   2.004
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           415697.569
Skew:                          10.682   Prob(JB):                         0.00
Kurtosis:                     151.885   Cond. No.                         4.94
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```

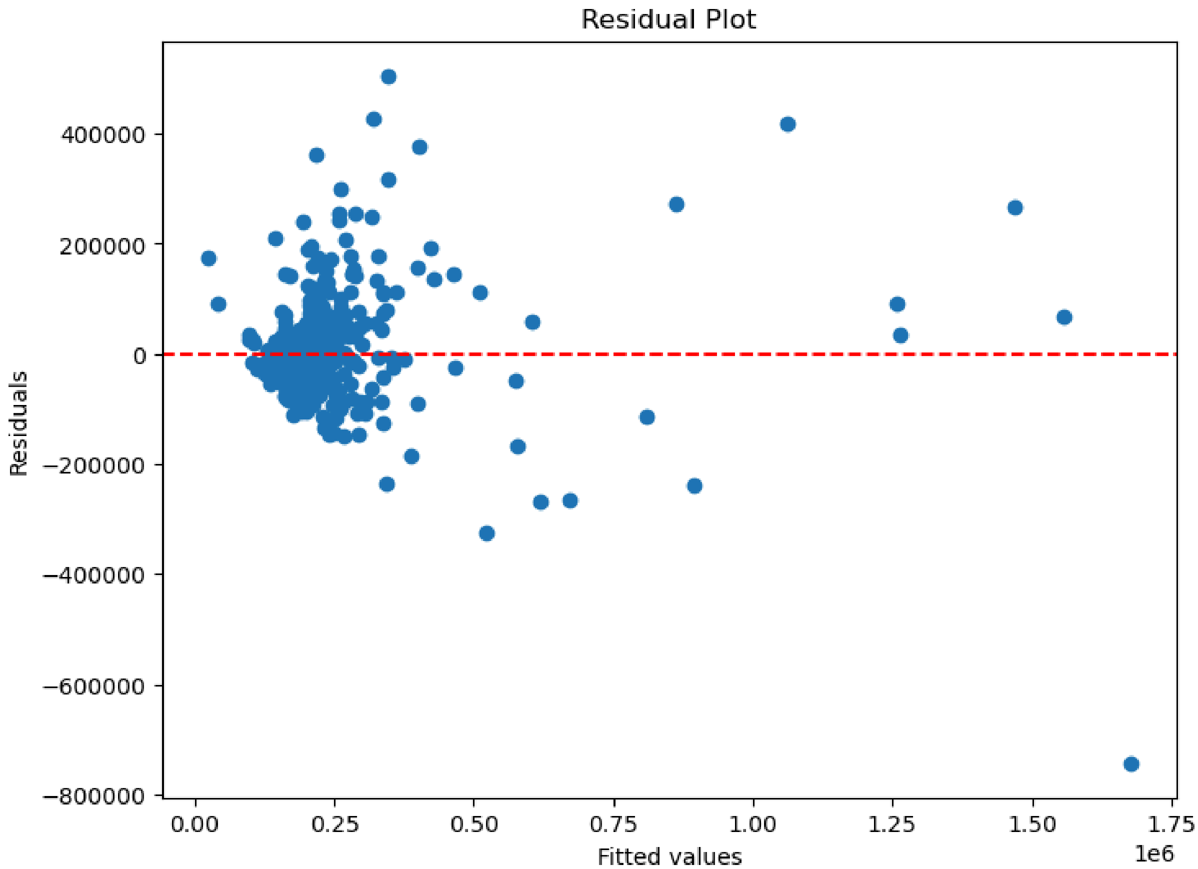
In [28]:
```
Y = df['Marketcap']
X = df[['Ebitda', 'Fulltimeemployees', 'Revenuegrowth']]
```

In [29]:
```
# Adding a constant (intercept) to the model
X = sm.add_constant(X)

# Square root transformation
Y = np.sqrt(Y)

# Fit the model
model = sm.OLS(Y, X).fit()

# Print out the statistics
print(model.summary())
```

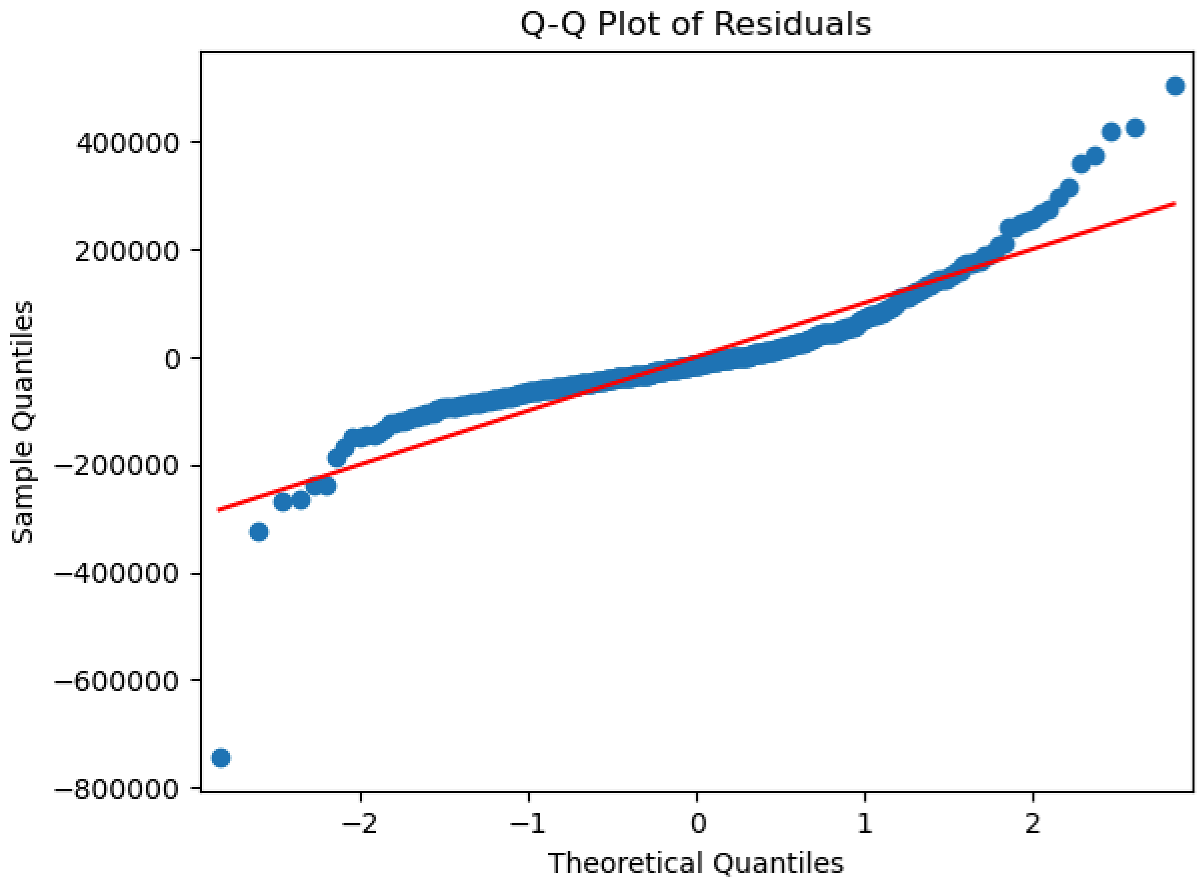
```
                            OLS Regression Results
==============================================================================
Dep. Variable:              Marketcap   R-squared:                       0.729
Model:                            OLS   Adj. R-squared:                  0.727
Method:                 Least Squares   F-statistic:                     392.3
Date:                Sun, 21 Apr 2024   Prob (F-statistic):          1.52e-123
Time:                        22:05:37   Log-Likelihood:                -5703.1
No. Observations:                 441   AIC:                         1.141e+04
Df Residuals:                     437   BIC:                         1.143e+04
Df Model:                           3
Covariance Type:            nonrobust
==================================================================================
=
                        coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------------
-
const              1.532e+05   5435.990     28.182      0.000    1.43e+05    1.64e+0
5
Ebitda             1.062e-05   3.77e-07     28.126      0.000    9.87e-06    1.14e-0
5
Fulltimeemployees     0.1110      0.038      2.946      0.003       0.037       0.18
5
Revenuegrowth      2.029e+05   2.37e+04      8.579      0.000    1.56e+05    2.49e+0
5
==============================================================================
Omnibus:                       94.544   Durbin-Watson:                   1.389
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1883.872
Skew:                           0.252   Prob(JB):                         0.00
Kurtosis:                      13.113   Cond. No.                     7.63e+10
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The condition number is large, 7.63e+10. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [30]:

```python
fig, ax = plt.subplots(figsize=(8, 6))
ax.scatter(x=model.fittedvalues, y=model.resid)
ax.axhline(y=0, color='red', linestyle='--')
ax.set_xlabel('Fitted values')
ax.set_ylabel('Residuals')
ax.set_title('Residual Plot')
plt.show()
```




In [31]:
```
fig = sm.qqplot(model.resid, line='s')
plt.title('Q-Q Plot of Residuals')
plt.show()
```




In [32]:
```
# Assume 'model' is the result of sm.OLS(Y, X).fit()
p_values = model.pvalues

# Print each p-value with high precision
for var, p in p_values.items():
    print(f"{var}: p-value = {p:.2e}")  # Adjust formatting as needed
```

```
const: p-value = 2.41e-100
Ebitda: p-value = 4.23e-100
Fulltimeemployees: p-value = 3.39e-03
Revenuegrowth: p-value = 1.69e-16
```

In [33]:
```
Y = df['Marketcap']
X = df[['Ebitda', 'Revenuegrowth']]
```

In [34]:
```
# Adding a constant (intercept) to the model
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(Y, X).fit()

# Print out the statistics
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              Marketcap   R-squared:                       0.744
Model:                            OLS   Adj. R-squared:                  0.742
Method:                 Least Squares   F-statistic:                     634.9
Date:                Sun, 21 Apr 2024   Prob (F-statistic):          3.82e-130
Time:                        22:05:38   Log-Likelihood:                -11924.
No. Observations:                 441   AIC:                         2.385e+04
Df Residuals:                     438   BIC:                         2.387e+04
Df Model:                           2
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const         -1.899e+10   7.12e+09     -2.667      0.008    -3.3e+10      -5e+09
Ebitda           15.3075      0.458     33.416      0.000      14.407      16.208
Revenuegrowth  3.035e+11   3.16e+10      9.613      0.000    2.41e+11    3.66e+11
==============================================================================
Omnibus:                      158.887   Durbin-Watson:                   1.509
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            24978.578
Skew:                           0.186   Prob(JB):                         0.00
Kurtosis:                      39.868   Cond. No.                     7.63e+10
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The condition number is large, 7.63e+10. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [ ]:

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

In [2]:
```
hp = pd.read_csv('sp500_stocks_4.18.2024.csv', parse_dates=True, index_col='Date')
hp = hp[(hp.index >= pd.Timestamp('2024-01-01')) & (hp.index <= pd.Timestamp('2024-
```

In [3]:
```
hp.head()
```

Out[3]:

| | Symbol | Adj Close | Close | High | Low | Open | Volume |
|---|---|---|---|---|---|---|---|
| **Date** | | | | | | | |
| **2024-01-02** | MMM | 90.176018 | 91.973244 | 92.525085 | 90.677261 | 90.819397 | 3321053.0 |
| **2024-01-03** | MMM | 88.364296 | 90.125420 | 91.521736 | 89.297661 | 91.329430 | 3547575.0 |
| **2024-01-04** | MMM | 88.675812 | 90.443146 | 91.421402 | 90.058525 | 90.367889 | 3319976.0 |
| **2024-01-05** | MMM | 89.020119 | 90.794312 | 91.546822 | 89.924751 | 90.284279 | 1991579.0 |
| **2024-01-08** | MMM | 89.241463 | 91.020065 | 91.103676 | 89.958191 | 90.518394 | 2535042.0 |

In [4]:
```
mshp = hp[hp['Symbol'] == 'GOOG']
```

In [5]:
```
num_rows = mshp.shape[0]
print("Number of rows:", num_rows)
```

```
Number of rows: 74
```

In [6]:
```
mshp.tail()
```

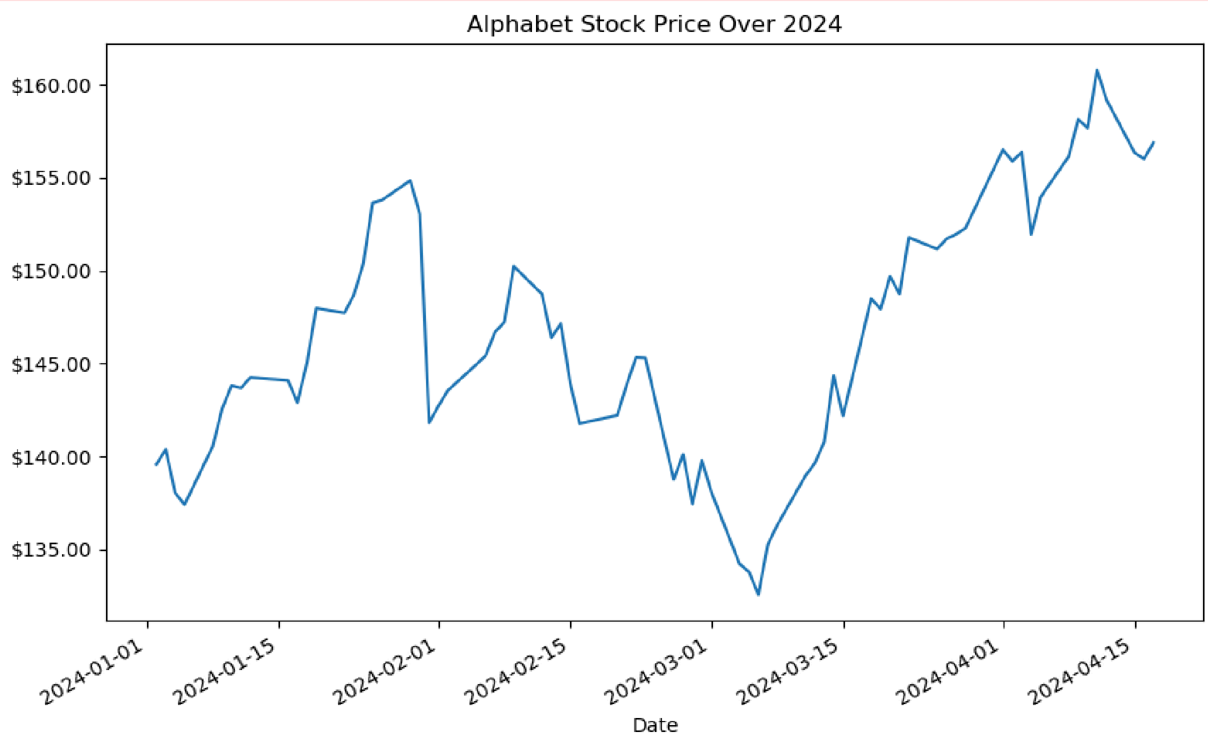
Out[6]:

| Date | Symbol | Adj Close | Close | High | Low | Open | Volume |
|---|---|---|---|---|---|---|---|
| **2024-04-11** | GOOG | 160.789993 | 160.789993 | 161.119995 | 157.929993 | 158.339996 | 17841700.0 |
| **2024-04-12** | GOOG | 159.190002 | 159.190002 | 161.699997 | 158.600006 | 159.404999 | 16968200.0 |
| **2024-04-15** | GOOG | 156.330002 | 156.330002 | 160.830002 | 156.149994 | 160.279999 | 21140900.0 |
| **2024-04-16** | GOOG | 156.000000 | 156.000000 | 157.229996 | 155.050003 | 155.639999 | 15413200.0 |
| **2024-04-17** | GOOG | 156.880005 | 156.880005 | 158.681000 | 156.134995 | 157.190002 | 16069524.0 |

In [7]:

```
import matplotlib.pyplot as plt

mshp['Close'].plot(title='Alphabet Stock Price Over 2024', figsize=(10, 6))
plt.gca().set_yticklabels(['${:,.2f}'.format(x) for x in plt.gca().get_yticks()])
plt.show()
```

```
C:\Users\ericz\AppData\Local\Temp\ipykernel_20444\3890262358.py:5: UserWarning: set_
ticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks
() or using a FixedLocator.
  plt.gca().set_yticklabels(['${:,.2f}'.format(x) for x in plt.gca().get_yticks()])
```




In [8]:

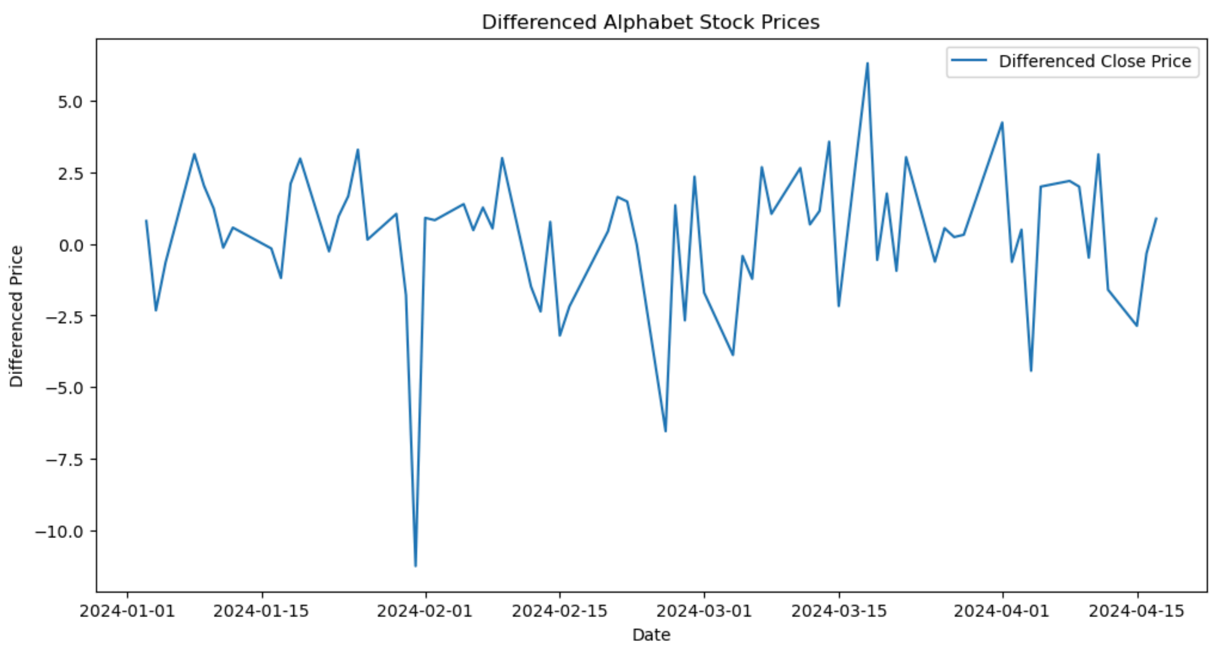
```
result = adfuller(mshp['Close'])
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
```

```
ADF Statistic: -1.377000
p-value: 0.593252
```

In [9]:
```
mshp['Differenced_Close'] = mshp['Close'] - mshp['Close'].shift(1)
mshp_diff = mshp.dropna()
```

```
C:\Users\ericz\AppData\Local\Temp\ipykernel_20444\3269642563.py:1: SettingWithCopyWa
rning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  mshp['Differenced_Close'] = mshp['Close'] - mshp['Close'].shift(1)
```

In [10]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(mshp_diff['Differenced_Close'], label='Differenced Close Price')
plt.title('Differenced Alphabet Stock Prices')
plt.xlabel('Date')
plt.ylabel('Differenced Price')
plt.legend()
plt.show()
```




In [11]:
```
result = adfuller(mshp_diff['Differenced_Close'])
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
```

```
ADF Statistic: -8.603674
p-value: 0.000000
```

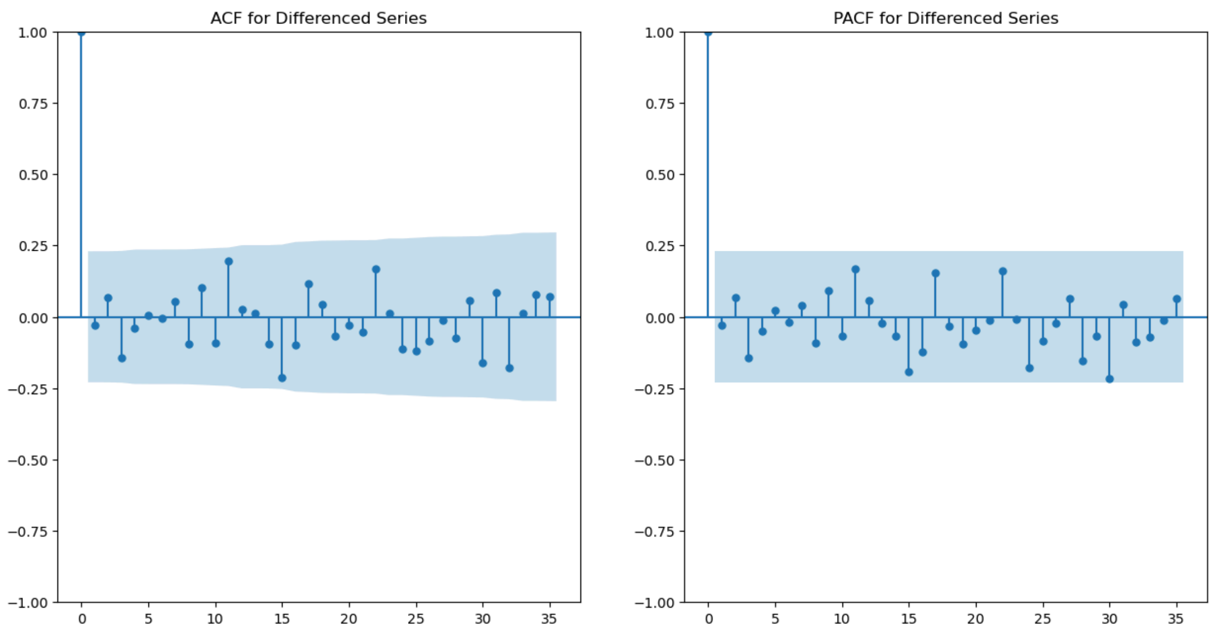
In [12]:
```
result_diff = adfuller(mshp_diff['Differenced_Close'])
print('ADF Statistic after differencing: %f' % result_diff[0])
print('p-value after differencing: %f' % result_diff[1])
```

```
for key, value in result_diff[4].items():
    print('Critical Value (%s): %f' % (key, value))
```

```
ADF Statistic after differencing: -8.603674
p-value after differencing: 0.000000
Critical Value (1%): -3.524624
Critical Value (5%): -2.902607
Critical Value (10%): -2.588679
```

In [13]:

```
plt.figure(figsize=(15, 7.5))
plt.subplot(121)
plot_acf(mshp_diff['Differenced_Close'], ax=plt.gca(), lags=35)
plt.title('ACF for Differenced Series')

plt.subplot(122)
plot_pacf(mshp_diff['Differenced_Close'], ax=plt.gca(), lags=35)
plt.title('PACF for Differenced Series')
plt.show()
```




In [14]:

```
model = ARIMA(mshp['Close'], order=(30, 1, 1))
fitted_model = model.fit()

print(fitted_model.summary())
```

```
C:\Users\ericz\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Va
lueWarning: A date index has been provided, but it has no associated frequency infor
mation and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\ericz\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Va
lueWarning: A date index has been provided, but it has no associated frequency infor
mation and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\ericz\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Va
lueWarning: A date index has been provided, but it has no associated frequency infor
mation and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\ericz\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:96
6: UserWarning: Non-stationary starting autoregressive parameters found. Using zeros
as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
```

```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:                  Close   No. Observations:                   74
Model:                ARIMA(30, 1, 1)   Log Likelihood                -153.662
Date:                Mon, 22 Apr 2024   AIC                            371.324
Time:                        01:57:22   BIC                            444.619
Sample:                             0   HQIC                           400.533
                                 - 74                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.3472      0.437     -0.795      0.426      -1.203       0.508
ar.L2          0.0249      0.211      0.118      0.906      -0.388       0.438
ar.L3          0.0024      0.276      0.009      0.993      -0.538       0.543
ar.L4         -0.1128      0.189     -0.598      0.550      -0.482       0.257
ar.L5         -0.1810      0.249     -0.728      0.467      -0.668       0.306
ar.L6         -0.0940      0.266     -0.354      0.723      -0.614       0.426
ar.L7          0.1437      0.190      0.758      0.448      -0.228       0.515
ar.L8          0.0684      0.183      0.374      0.708      -0.290       0.427
ar.L9         -0.0379      0.166     -0.228      0.820      -0.364       0.288
ar.L10        -0.2059      0.216     -0.954      0.340      -0.629       0.217
ar.L11         0.0430      0.248      0.173      0.863      -0.444       0.530
ar.L12         0.1178      0.187      0.631      0.528      -0.248       0.484
ar.L13         0.0327      0.261      0.125      0.900      -0.479       0.544
ar.L14        -0.1722      0.200     -0.860      0.390      -0.565       0.220
ar.L15        -0.3458      0.243     -1.421      0.155      -0.823       0.131
ar.L16        -0.1958      0.209     -0.937      0.349      -0.605       0.214
ar.L17         0.1095      0.138      0.795      0.426      -0.160       0.379
ar.L18         0.0618      0.231      0.268      0.789      -0.390       0.514
ar.L19        -0.0456      0.143     -0.319      0.750      -0.325       0.234
ar.L20        -0.1938      0.176     -1.100      0.271      -0.539       0.152
ar.L21        -0.0661      0.233     -0.284      0.776      -0.523       0.390
ar.L22         0.2292      0.249      0.920      0.358      -0.259       0.718
ar.L23         0.1199      0.315      0.381      0.703      -0.497       0.737
ar.L24        -0.2290      0.145     -1.574      0.115      -0.514       0.056
ar.L25        -0.2686      0.252     -1.068      0.286      -0.762       0.224
ar.L26        -0.1084      0.235     -0.461      0.644      -0.569       0.352
ar.L27         0.0333      0.286      0.117      0.907      -0.526       0.593
ar.L28        -0.1645      0.321     -0.513      0.608      -0.793       0.464
ar.L29        -0.2664      0.262     -1.018      0.309      -0.780       0.247
ar.L30        -0.4787      0.242     -1.978      0.048      -0.953      -0.004
ma.L1          0.2367      0.543      0.436      0.663      -0.827       1.300
sigma2         3.1737      1.173      2.706      0.007       0.875       5.473
===================================================================================
Ljung-Box (L1) (Q):                   0.04   Jarque-Bera (JB):               124.80
Prob(Q):                              0.84   Prob(JB):                         0.00
Heteroskedasticity (H):               0.79   Skew:                            -1.56
Prob(H) (two-sided):                  0.56   Kurtosis:                         8.60
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-ste
p).
```

In [15]:
```
forecast = fitted_model.forecast(steps=14)
print(forecast)
```

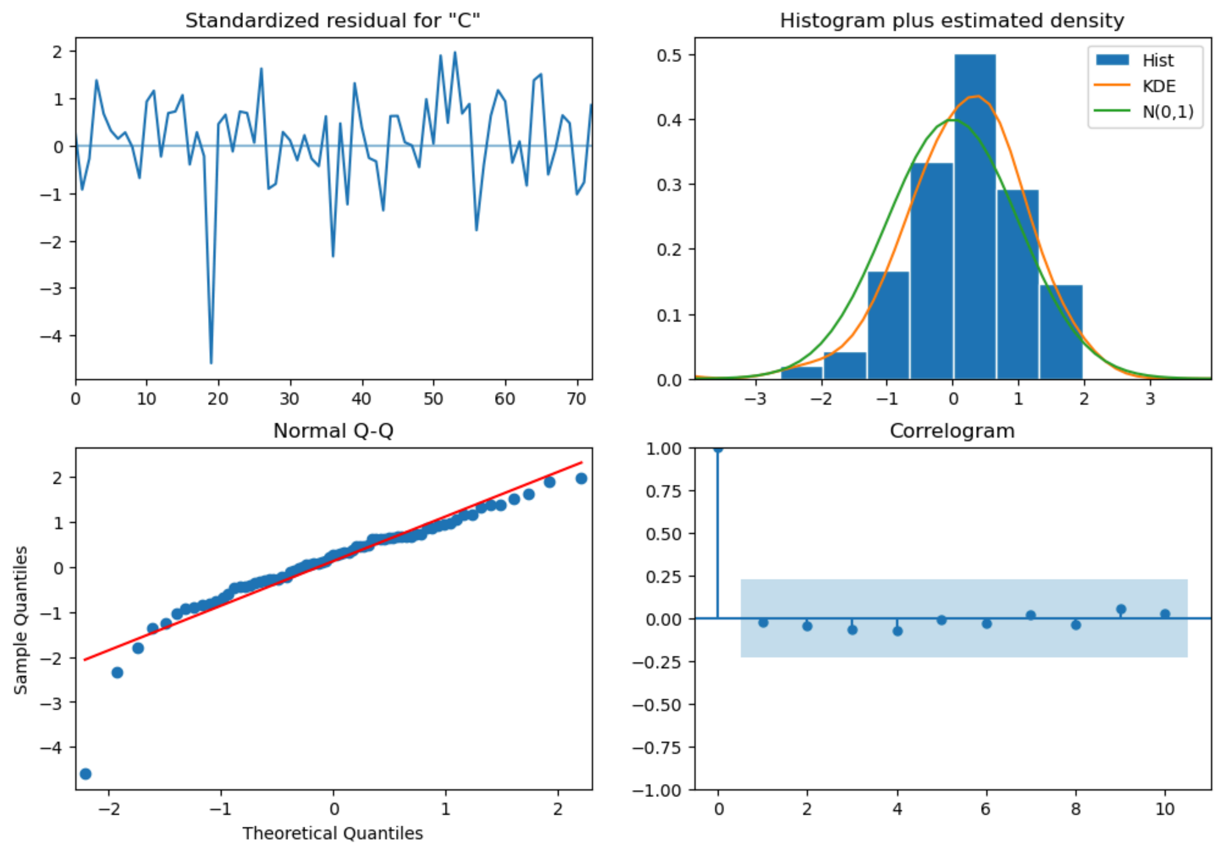
```
74    157.323574
75    154.056399
76    151.326116
77    148.589628
78    148.630114
79    148.481774
80    146.844460
81    145.822878
82    141.617225
83    142.773344
84    141.262522
85    141.966237
86    140.758315
87    141.420953
Name: predicted_mean, dtype: float64
```

```
C:\Users\ericz\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  return get_prediction_index(
C:\Users\ericz\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the next version, calling this method in a model without a supported index will result in an exception.
  return get_prediction_index(
```

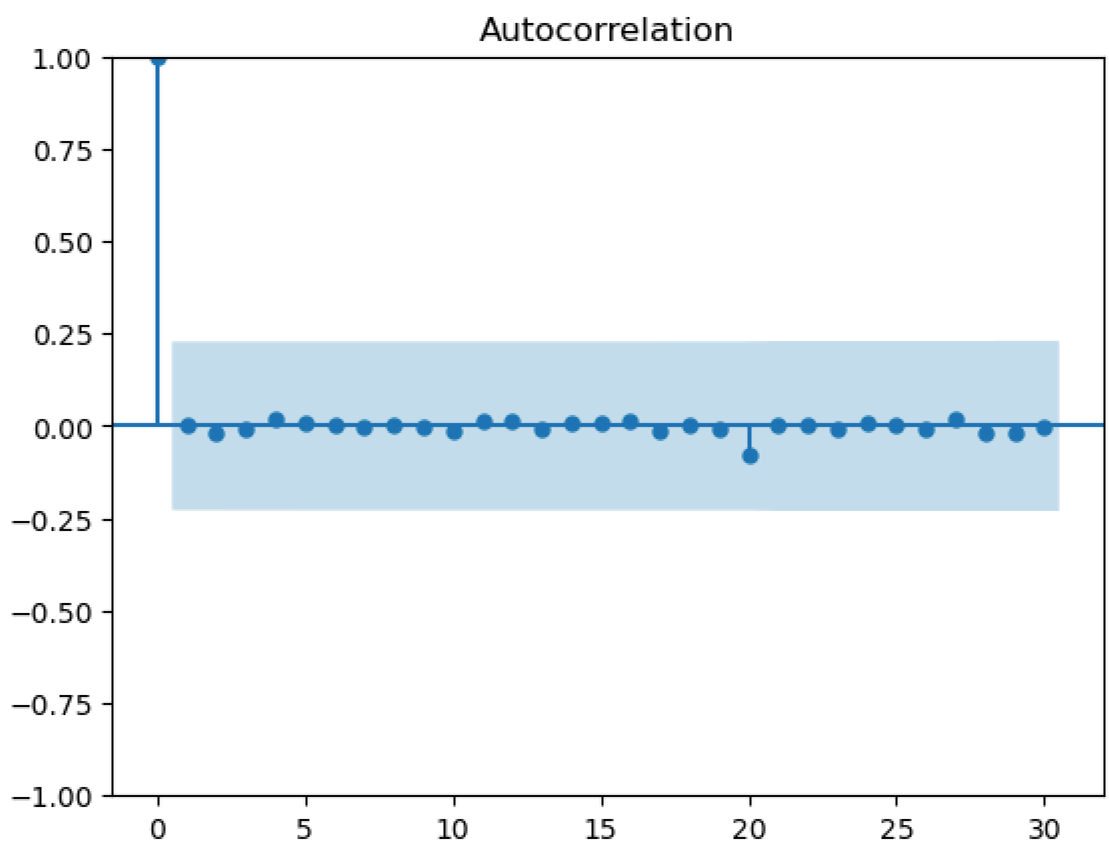
In [16]:
```
for i, price in enumerate(forecast, 1):
    print(f"Day {i}: ${price:.2f}")
```

```
Day 1: $157.32
Day 2: $154.06
Day 3: $151.33
Day 4: $148.59
Day 5: $148.63
Day 6: $148.48
Day 7: $146.84
Day 8: $145.82
Day 9: $141.62
Day 10: $142.77
Day 11: $141.26
Day 12: $141.97
Day 13: $140.76
Day 14: $141.42
```

In [17]:
```
fitted_model.plot_diagnostics(figsize=(12, 8))
plt.show()
```




In [18]:
```
plot_acf(fitted_model.resid, lags=30)
plt.show()
```



In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

In [2]:
```
hp = pd.read_csv('sp500_stocks_4.18.2024.csv', parse_dates=True, index_col='Date')
hp = hp[(hp.index >= pd.Timestamp('2024-01-01')) & (hp.index <= pd.Timestamp('2024-
```

In [3]:
```
hp.head()
```

Out[3]:

| Date | Symbol | Adj Close | Close | High | Low | Open | Volume |
|---|---|---|---|---|---|---|---|
| **2024-01-02** | MMM | 90.176018 | 91.973244 | 92.525085 | 90.677261 | 90.819397 | 3321053.0 |
| **2024-01-03** | MMM | 88.364296 | 90.125420 | 91.521736 | 89.297661 | 91.329430 | 3547575.0 |
| **2024-01-04** | MMM | 88.675812 | 90.443146 | 91.421402 | 90.058525 | 90.367889 | 3319976.0 |
| **2024-01-05** | MMM | 89.020119 | 90.794312 | 91.546822 | 89.924751 | 90.284279 | 1991579.0 |
| **2024-01-08** | MMM | 89.241463 | 91.020065 | 91.103676 | 89.958191 | 90.518394 | 2535042.0 |

In [4]:
```
mshp = hp[hp['Symbol'] == 'MSFT']
```

In [5]:
```
num_rows = mshp.shape[0]
print("Number of rows:", num_rows)
```

```
Number of rows: 74
```

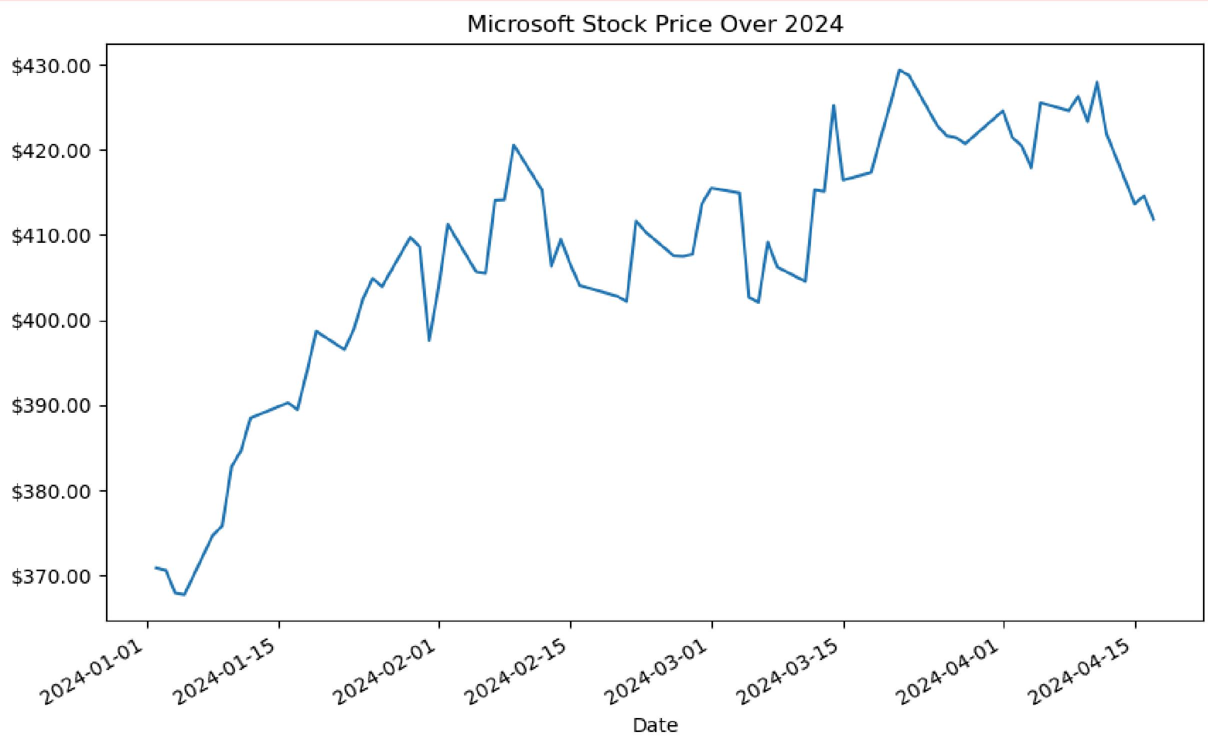
In [6]:
```
mshp.tail()
```

Out[6]:

| Date | Symbol | Adj Close | Close | High | Low | Open | Volume |
|---|---|---|---|---|---|---|---|
| **2024-04-11** | MSFT | 427.929993 | 427.929993 | 429.369995 | 422.359985 | 425.820007 | 17966400.0 |
| **2024-04-12** | MSFT | 421.899994 | 421.899994 | 425.179993 | 419.769989 | 424.049988 | 19232100.0 |
| **2024-04-15** | MSFT | 413.640015 | 413.640015 | 426.820007 | 413.429993 | 426.600006 | 20273500.0 |
| **2024-04-16** | MSFT | 414.579987 | 414.579987 | 418.399994 | 413.730011 | 414.570007 | 16765600.0 |
| **2024-04-17** | MSFT | 411.839996 | 411.839996 | 418.880005 | 410.329987 | 417.595001 | 15779844.0 |

In [7]:
```
import matplotlib.pyplot as plt

mshp['Close'].plot(title='Microsoft Stock Price Over 2024', figsize=(10, 6))
plt.gca().set_yticklabels(['${:,.2f}'.format(x) for x in plt.gca().get_yticks()])
plt.show()
```

```
C:\Users\ericz\AppData\Local\Temp\ipykernel_16604\3865135180.py:5: UserWarning: set_
ticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks
() or using a FixedLocator.
  plt.gca().set_yticklabels(['${:,.2f}'.format(x) for x in plt.gca().get_yticks()])
```



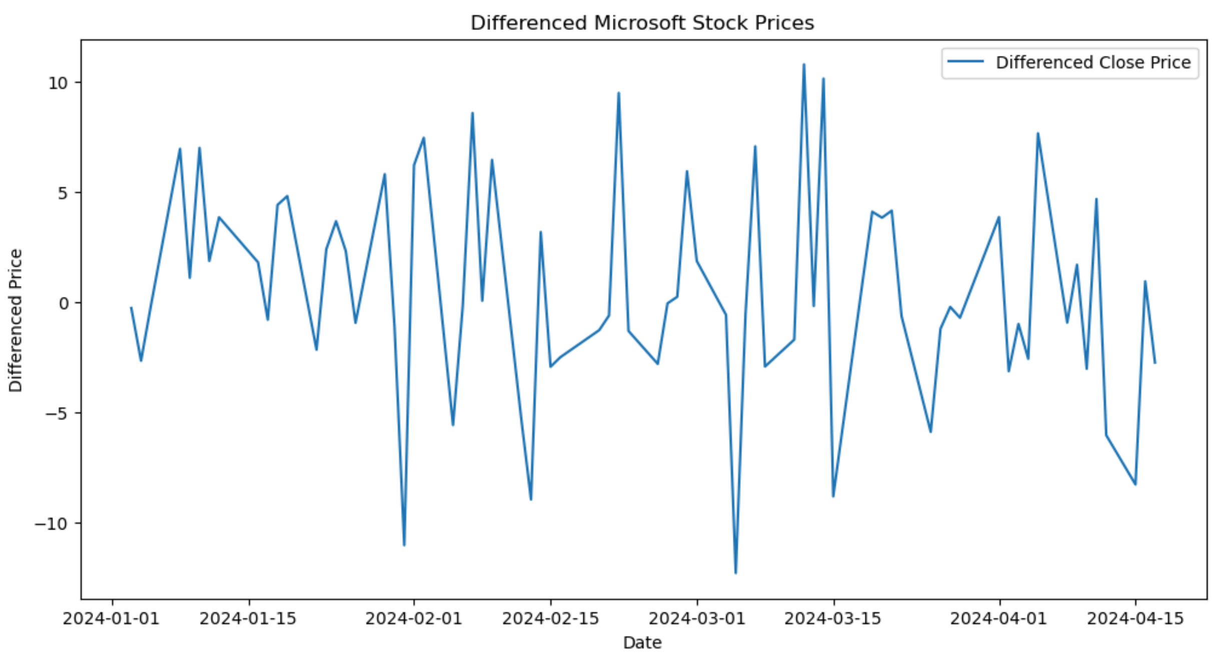

In [8]:
```
result = adfuller(mshp['Close'])
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
```

```
ADF Statistic: -2.506460
p-value: 0.113910
```

In [9]:
```
mshp['Differenced_Close'] = mshp['Close'] - mshp['Close'].shift(1)
mshp_diff = mshp.dropna()
```

```
C:\Users\ericz\AppData\Local\Temp\ipykernel_16604\3269642563.py:1: SettingWithCopyWa
rning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  mshp['Differenced_Close'] = mshp['Close'] - mshp['Close'].shift(1)
```

In [10]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(mshp_diff['Differenced_Close'], label='Differenced Close Price')
plt.title('Differenced Microsoft Stock Prices')
plt.xlabel('Date')
plt.ylabel('Differenced Price')
plt.legend()
plt.show()
```




In [11]:
```
result = adfuller(mshp_diff['Differenced_Close'])
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
```

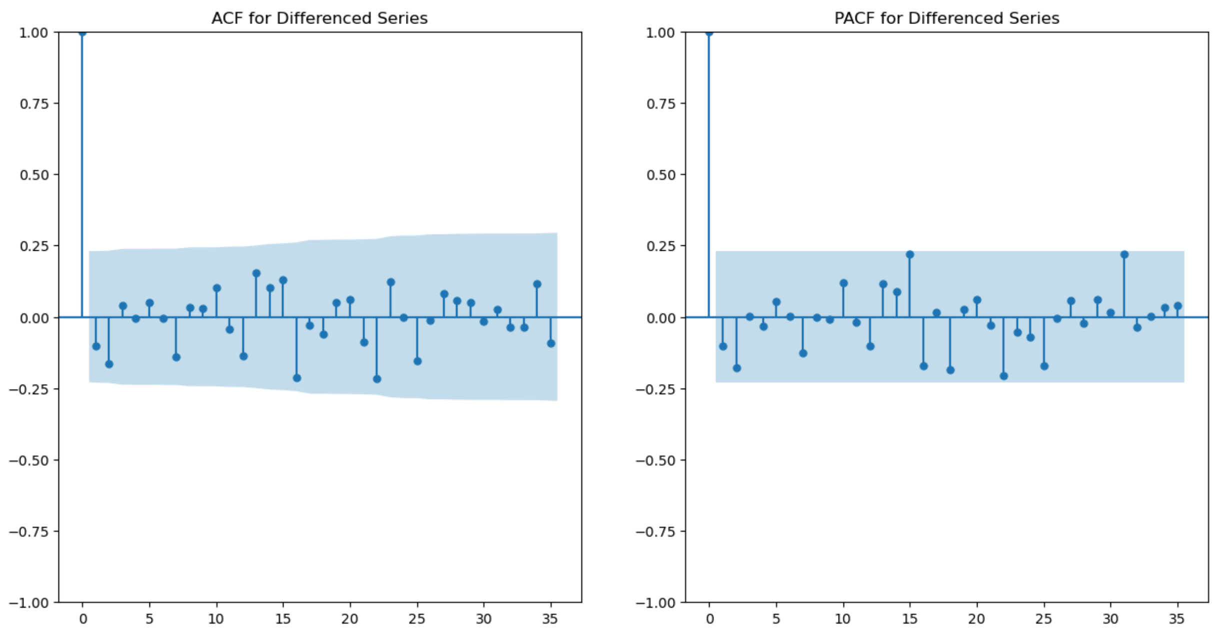
```
ADF Statistic: -7.348833
p-value: 0.000000
```

In [12]:
```
result_diff = adfuller(mshp_diff['Differenced_Close'])
print('ADF Statistic after differencing: %f' % result_diff[0])
print('p-value after differencing: %f' % result_diff[1])
```

```
for key, value in result_diff[4].items():
    print('Critical Value (%s): %f' % (key, value))
```

```
ADF Statistic after differencing: -7.348833
p-value after differencing: 0.000000
Critical Value (1%): -3.526005
Critical Value (5%): -2.903200
Critical Value (10%): -2.588995
```

In [13]:
```
plt.figure(figsize=(15, 7.5))
plt.subplot(121)
plot_acf(mshp_diff['Differenced_Close'], ax=plt.gca(), lags=35)
plt.title('ACF for Differenced Series')

plt.subplot(122)
plot_pacf(mshp_diff['Differenced_Close'], ax=plt.gca(), lags=35)
plt.title('PACF for Differenced Series')
plt.show()
```




In [14]:
```
model = ARIMA(mshp['Close'], order=(30, 1, 1))
fitted_model = model.fit()

print(fitted_model.summary())
```

```
C:\Users\ericz\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Va
lueWarning: A date index has been provided, but it has no associated frequency infor
mation and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\ericz\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Va
lueWarning: A date index has been provided, but it has no associated frequency infor
mation and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\ericz\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Va
lueWarning: A date index has been provided, but it has no associated frequency infor
mation and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\ericz\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:96
6: UserWarning: Non-stationary starting autoregressive parameters found. Using zeros
as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
```

```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:                  Close   No. Observations:                   74
Model:                ARIMA(30, 1, 1)   Log Likelihood                -200.242
Date:                Sun, 21 Apr 2024   AIC                            464.484
Time:                        00:18:19   BIC                            537.779
Sample:                             0   HQIC                           493.694
                                 - 74                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.5505      0.307      1.792      0.073      -0.052       1.153
ar.L2         -0.1264      0.227     -0.556      0.578      -0.572       0.319
ar.L3          0.1477      0.373      0.396      0.692      -0.584       0.879
ar.L4         -0.1487      0.411     -0.362      0.718      -0.954       0.657
ar.L5          0.1595      0.333      0.479      0.632      -0.494       0.813
ar.L6         -0.0426      0.276     -0.154      0.877      -0.584       0.499
ar.L7         -0.1015      0.223     -0.455      0.649      -0.538       0.335
ar.L8          0.3634      0.246      1.479      0.139      -0.118       0.845
ar.L9         -0.0750      0.236     -0.317      0.751      -0.538       0.388
ar.L10         0.1646      0.229      0.718      0.473      -0.285       0.614
ar.L11        -0.0935      0.233     -0.401      0.688      -0.551       0.364
ar.L12         0.0510      0.264      0.193      0.847      -0.467       0.569
ar.L13         0.2395      0.297      0.806      0.420      -0.343       0.822
ar.L14        -0.0383      0.273     -0.140      0.888      -0.573       0.497
ar.L15         0.1442      0.202      0.715      0.474      -0.251       0.539
ar.L16        -0.4295      0.212     -2.030      0.042      -0.844      -0.015
ar.L17         0.1595      0.270      0.590      0.555      -0.370       0.689
ar.L18        -0.2969      0.200     -1.482      0.138      -0.690       0.096
ar.L19         0.2281      0.273      0.837      0.403      -0.306       0.762
ar.L20        -0.0251      0.330     -0.076      0.939      -0.672       0.622
ar.L21        -0.1631      0.339     -0.481      0.631      -0.828       0.502
ar.L22        -0.2428      0.387     -0.628      0.530      -1.001       0.515
ar.L23         0.0932      0.428      0.218      0.828      -0.746       0.932
ar.L24         0.0175      0.485      0.036      0.971      -0.934       0.969
ar.L25        -0.1876      0.478     -0.393      0.694      -1.124       0.749
ar.L26         0.1932      0.490      0.395      0.693      -0.767       1.153
ar.L27         0.0828      0.449      0.184      0.854      -0.797       0.963
ar.L28        -0.1002      0.331     -0.302      0.762      -0.749       0.549
ar.L29         0.1290      0.352      0.366      0.714      -0.562       0.820
ar.L30         0.2637      0.324      0.814      0.416      -0.372       0.899
ma.L1         -0.7317      0.296     -2.471      0.013      -1.312      -0.151
sigma2        11.5035      3.574      3.219      0.001       4.499      18.508
===================================================================================
Ljung-Box (L1) (Q):                   0.15   Jarque-Bera (JB):                 0.35
Prob(Q):                              0.70   Prob(JB):                         0.84
Heteroskedasticity (H):               1.06   Skew:                            -0.00
Prob(H) (two-sided):                  0.88   Kurtosis:                         2.66
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-ste
p).
```

In [15]:
```
forecast = fitted_model.forecast(steps=14)
print(forecast)
```

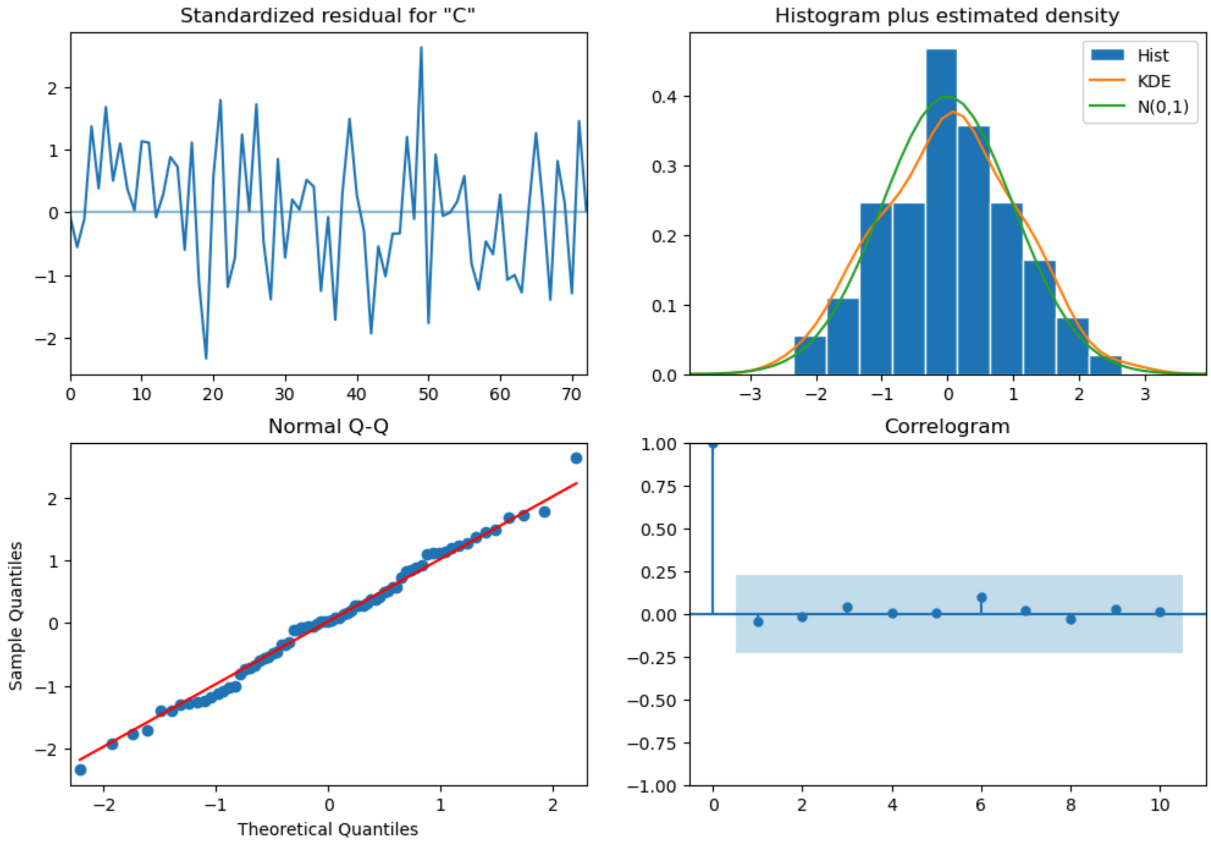
```
74    412.142934
75    414.624530
76    411.710524
77    410.631080
78    413.822433
79    415.241149
80    421.757975
81    417.016272
82    418.873429
83    415.954350
84    416.521858
85    417.208139
86    416.479562
87    415.251844
Name: predicted_mean, dtype: float64
```

```
C:\Users\ericz\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  return get_prediction_index(
C:\Users\ericz\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the next version, calling this method in a model without a supported index will result in an exception.
  return get_prediction_index(
```

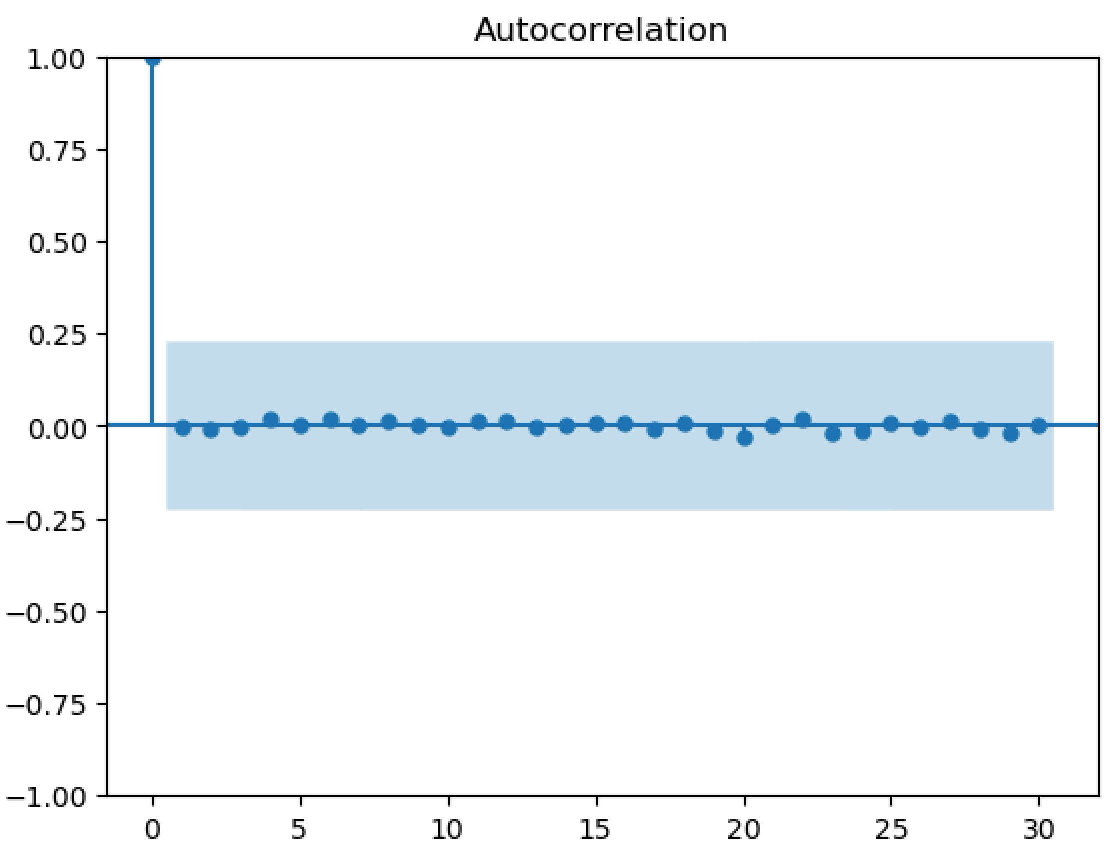
In [16]:
```
for i, price in enumerate(forecast, 1):
    print(f"Day {i}: ${price:.2f}")
```

```
Day 1: $412.14
Day 2: $414.62
Day 3: $411.71
Day 4: $410.63
Day 5: $413.82
Day 6: $415.24
Day 7: $421.76
Day 8: $417.02
Day 9: $418.87
Day 10: $415.95
Day 11: $416.52
Day 12: $417.21
Day 13: $416.48
Day 14: $415.25
```

In [17]:
```
fitted_model.plot_diagnostics(figsize=(12, 8))
plt.show()
```




In [18]:
```
plot_acf(fitted_model.resid, lags=30)
plt.show()
```



In [ ]: