EDA and Preprocessing of Tesla Stock Pricing Data - Fazal Rehman

In [15]:
```
import os
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima_model import ARIMA
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import mean_squared_error, mean_absolute_error
import math
import yfinance as yf
from pmdarima import auto_arima
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
```

In [2]:
```
stock_data = pd.read_csv("D:\Downloads\TSLA.CSV")
stock_data.head()
```

Out[2]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2019-05-21	39.551998	41.480000	39.208000	41.015999	90019500	0	0.0
1	2019-05-22	39.820000	40.787998	38.355999	38.546001	93426000	0	0.0
2	2019-05-23	38.868000	39.894001	37.243999	39.098000	132735500	0	0.0
3	2019-05-24	39.966000	39.995998	37.750000	38.125999	70683000	0	0.0
4	2019-05-28	38.240002	39.000000	37.570000	37.740002	51564500	0	0.0

In [3]:
```
# As we are performing UniVariate Time Series Analysis so we will conside only clos

stock_data = stock_data[['Date','Close']] # filtering the dataframe to date and clo
```

In [4]:
```
stock_data
```

Out[4]:

	Date	Close
0	2019-05-21	41.015999
1	2019-05-22	38.546001
2	2019-05-23	39.098000
3	2019-05-24	38.125999
4	2019-05-28	37.740002
...	...	...
753	2022-05-16	724.369995
754	2022-05-17	761.609985
755	2022-05-18	709.809998
756	2022-05-19	709.419983
757	2022-05-20	663.900024

758 rows × 2 columns

In [5]:
```
stock_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 758 entries, 0 to 757
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Date    758 non-null    object 
 1   Close   758 non-null    float64
dtypes: float64(1), object(1)
memory usage: 12.0+ KB
```

In [6]:
```
stock_data.Date = pd.to_datetime(stock_data.Date) # convert Date data type ('object
```

In [7]:
```
stock_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 758 entries, 0 to 757
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   Date    758 non-null    datetime64[ns]
 1   Close   758 non-null    float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 12.0 KB
```

In [8]:
```
stock_data
```

Out[8]:

	Date	Close
0	2019-05-21	41.015999
1	2019-05-22	38.546001
2	2019-05-23	39.098000
3	2019-05-24	38.125999
4	2019-05-28	37.740002
...	...	...
753	2022-05-16	724.369995
754	2022-05-17	761.609985
755	2022-05-18	709.809998
756	2022-05-19	709.419983
757	2022-05-20	663.900024

758 rows × 2 columns

In [9]:
```
stock_data = stock_data.set_index("Date") # setting date as index
```

In [10]:
```
stock_data.head(5)
```

Out[10]:

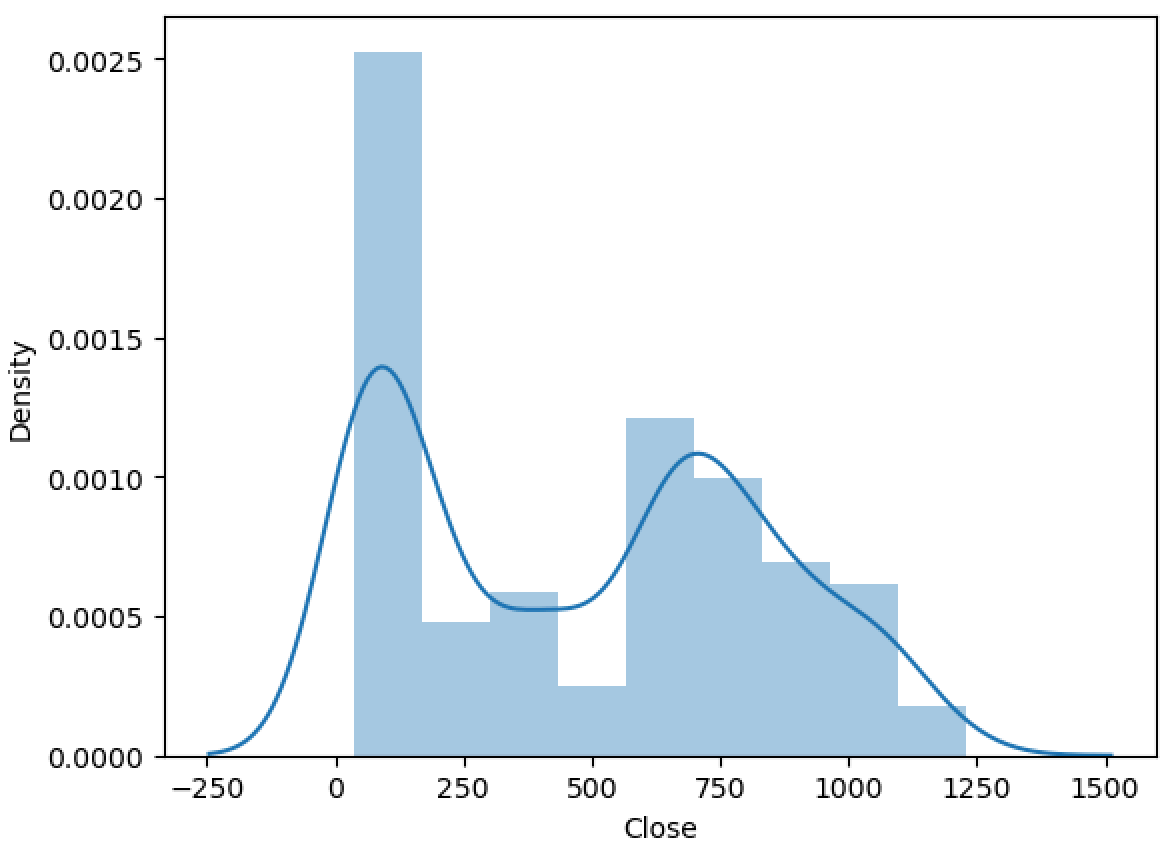
Date	Close
2019-05-21	41.015999
2019-05-22	38.546001
2019-05-23	39.098000
2019-05-24	38.125999
2019-05-28	37.740002

In [11]:
```
import matplotlib.pyplot as plt
plt.plot(stock_data['Close'])
```

Out[11]:
```
[<matplotlib.lines.Line2D at 0x28838f31bd0>]
```

In [12]:
```
import seaborn as sns
sns.distplot(stock_data['Close'])
```

Out[12]: <Axes: xlabel='Close', ylabel='Density'>



In [13]:
```
stock_data
```

Out[13]:

	Close
Date	
2019-05-21	41.015999
2019-05-22	38.546001
2019-05-23	39.098000
2019-05-24	38.125999
2019-05-28	37.740002
...	...
2022-05-16	724.369995
2022-05-17	761.609985
2022-05-18	709.809998
2022-05-19	709.419983
2022-05-20	663.900024

758 rows × 1 columns

In [14]:
```
type(stock_data['Close'])
```

Out[14]:
```
pandas.core.series.Series
```

In [15]:
```
np.mean(stock_data['Close'].head(12))
```

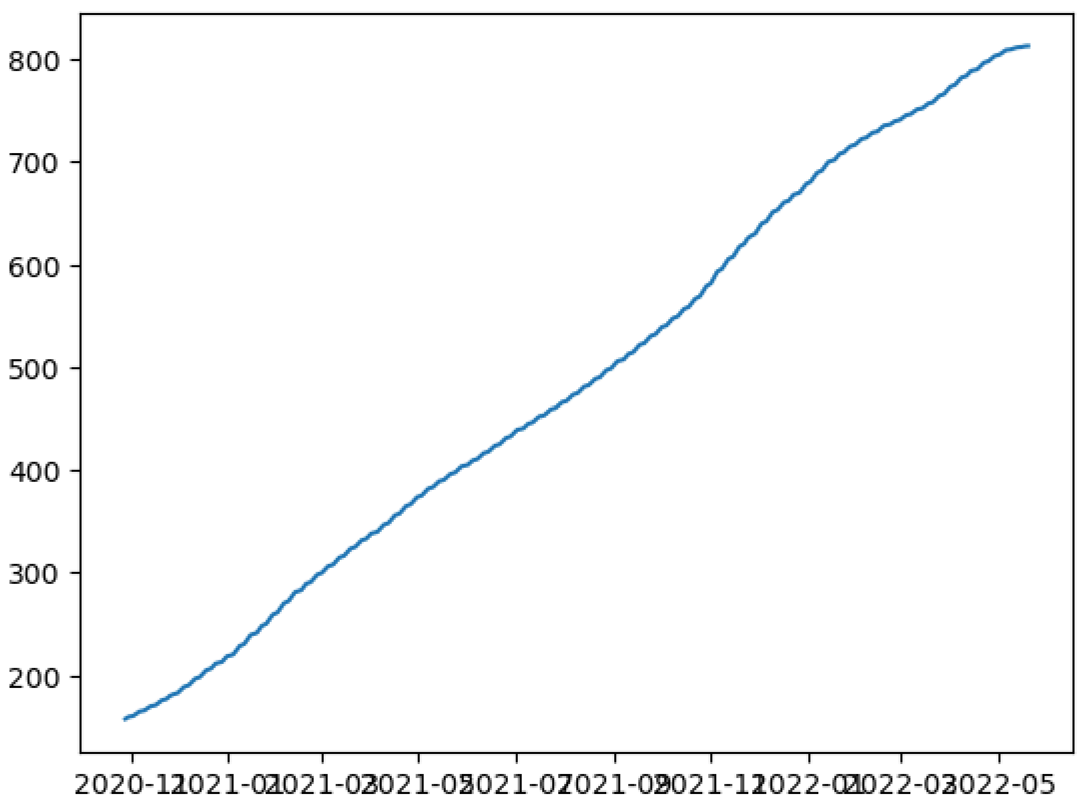
Out[15]:
```
38.51633358001709
```

In [16]:
```
plt.plot(stock_data['Close'])
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x28839becc10>]
```

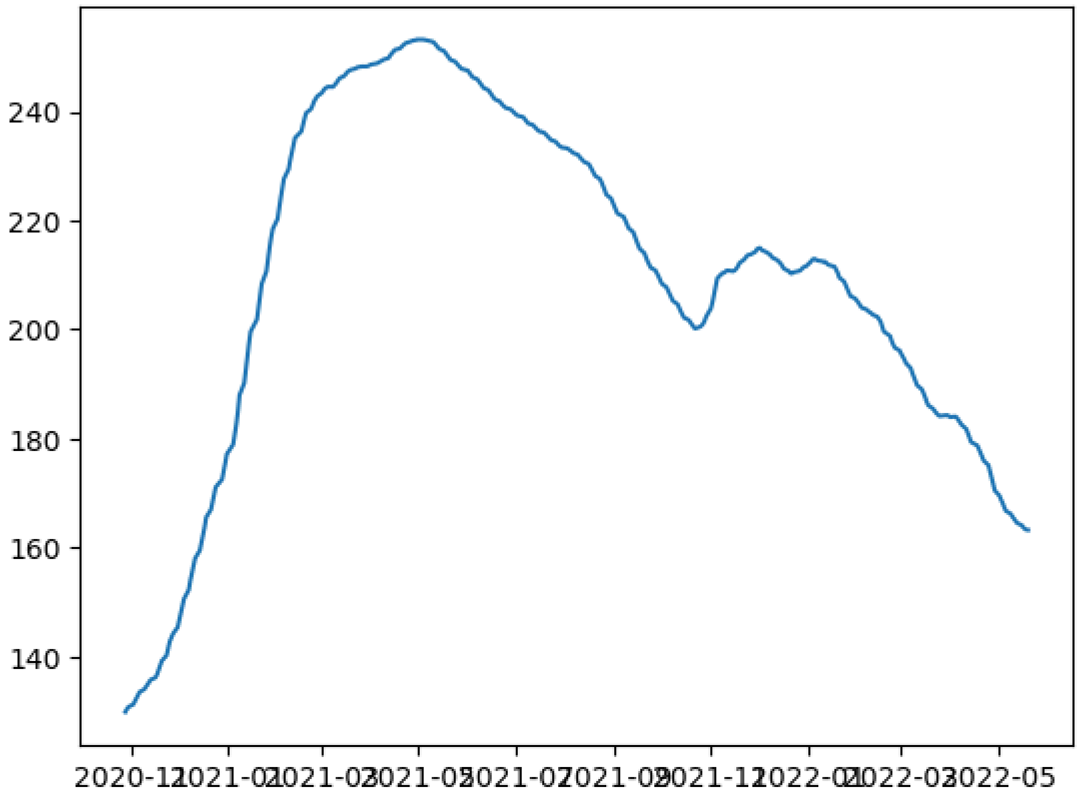
In [17]:
```
plt.plot(stock_data['Close'].rolling(365).mean())
```

Out[17]: [<matplotlib.lines.Line2D at 0x28839ca7a50>]



In [18]:
```
plt.plot(stock_data['Close'].rolling(365).std() )
```

Out[18]: [<matplotlib.lines.Line2D at 0x2883c7c0890>]



In [19]:
```
adfuller(stock_data['Close'],autolag='AIC')
```

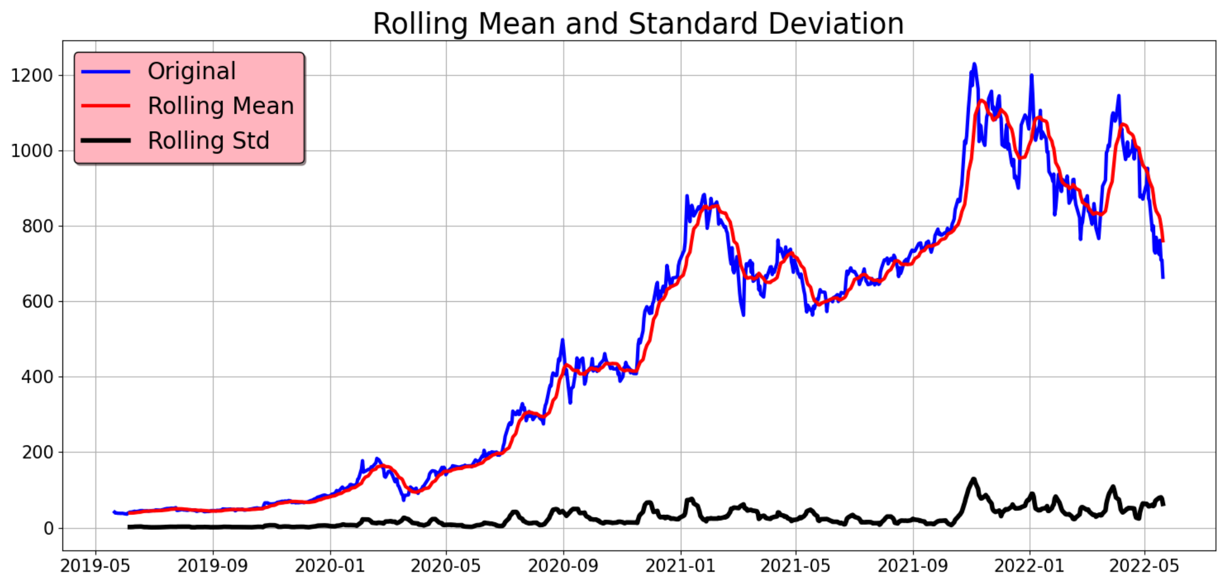
Out[19]:
```
(-1.363008581703748,
 0.5998762543050701,
 9,
 748,
 {'1%': -3.43912257105195,
  '5%': -2.8654117005229844,
  '10%': -2.568831705010152},
 6794.359259220987)
```

In [20]:
```
#Test for staionarity
def test_stationarity(timeseries):
    # Determing rolling statistics
    rolmean = timeseries.rolling(12).mean() # rolling mean
    rolstd = timeseries.rolling(12).std() # rolling standard deviation
    # Plot rolling statistics:
    plt.figure(figsize = (18,8))
    plt.grid('both')
    plt.plot(timeseries, color='blue',label='Original', linewidth = 3)
    plt.plot(rolmean, color='red', label='Rolling Mean',linewidth = 3)
    plt.plot(rolstd, color='black', label = 'Rolling Std',linewidth = 4)
    plt.legend(loc='best', fontsize = 20, shadow=True,facecolor='lightpink',edgecol
    plt.title('Rolling Mean and Standard Deviation', fontsize = 25)
    plt.xticks(fontsize = 15)
    plt.yticks(fontsize = 15)
    plt.show(block=False)

    print("Results of dickey fuller test")
```

```
    adft = adfuller(timeseries,autolag='AIC')
    # output for dft will give us without defining what the values are.
    # hence we manually write what values does it explains using a for loop
    output = pd.Series(adft[0:4],index=['Test Statistics','p-value','No. of lags us
    for key,values in adft[4].items():
        output['critical value (%s)'%key] =  values
    print(output)
```

In [21]:
```
test_stationarity(stock_data['Close'])
```




```
Results of dickey fuller test
Test Statistics                 -1.363009
p-value                          0.599876
No. of lags used                 9.000000
Number of observations used    748.000000
critical value (1%)             -3.439123
critical value (5%)             -2.865412
critical value (10%)            -2.568832
dtype: float64
```

In [22]:
```
df_close=stock_data['Close']
```

In [23]:
```
df_close.diff()
```

Out[23]:
```
Date
2019-05-21          NaN
2019-05-22    -2.469997
2019-05-23     0.551998
2019-05-24    -0.972000
2019-05-28    -0.385998
                 ...
2022-05-16   -45.220032
2022-05-17    37.239990
2022-05-18   -51.799988
2022-05-19    -0.390015
2022-05-20   -45.519958
Name: Close, Length: 758, dtype: float64
```

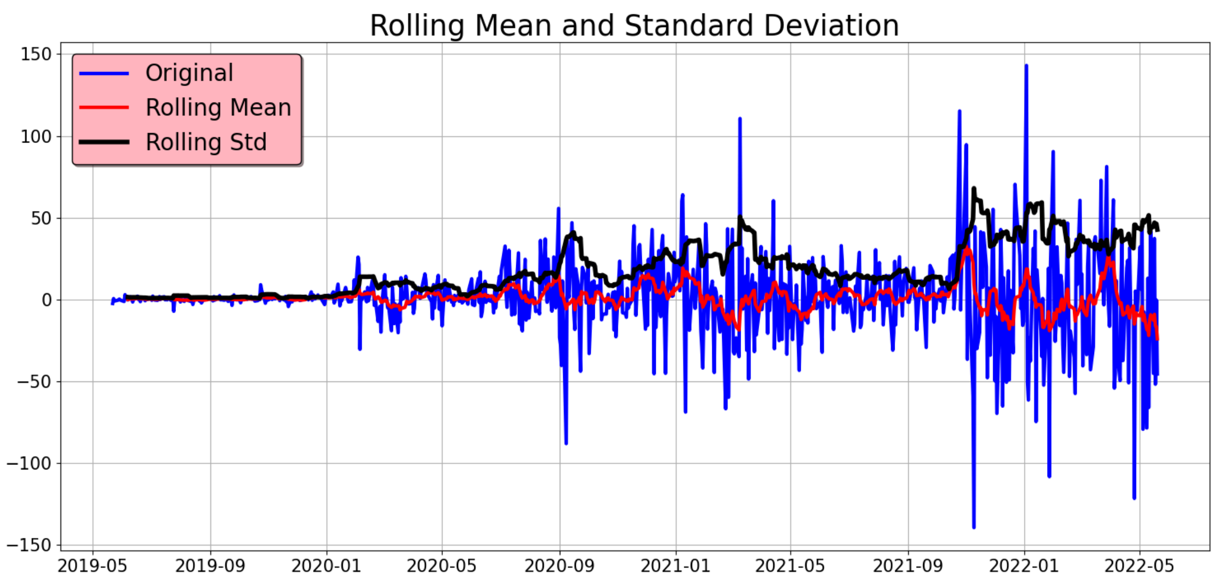
In [24]:
```python
# Get the difference of each Adj Close point
tsla_close_diff_1 = df_close.diff()
```

In [25]:
```python
tsla_close_diff_1
```

Out[25]:
```
Date
2019-05-21          NaN
2019-05-22    -2.469997
2019-05-23     0.551998
2019-05-24    -0.972000
2019-05-28    -0.385998
                ...
2022-05-16   -45.220032
2022-05-17    37.239990
2022-05-18   -51.799988
2022-05-19    -0.390015
2022-05-20   -45.519958
Name: Close, Length: 758, dtype: float64
```

In [26]:
```python
tsla_close_diff_1.dropna(inplace=True)
```

In [27]:
```python
# Plot the tsla Adj Close 1st order difference
test_stationarity(tsla_close_diff_1)
```




```
Results of dickey fuller test
Test Statistics                 -8.324564e+00
p-value                          3.498786e-13
No. of lags used                 8.000000e+00
Number of observations used      7.480000e+02
critical value (1%)             -3.439123e+00
critical value (5%)             -2.865412e+00
critical value (10%)            -2.568832e+00
dtype: float64
```

In [28]:
```python
stock_data[["Close"]]
```

Out[28]:

	Close
Date	
2019-05-21	41.015999
2019-05-22	38.546001
2019-05-23	39.098000
2019-05-24	38.125999
2019-05-28	37.740002
...	...
2022-05-16	724.369995
2022-05-17	761.609985
2022-05-18	709.809998
2022-05-19	709.419983
2022-05-20	663.900024

758 rows × 1 columns

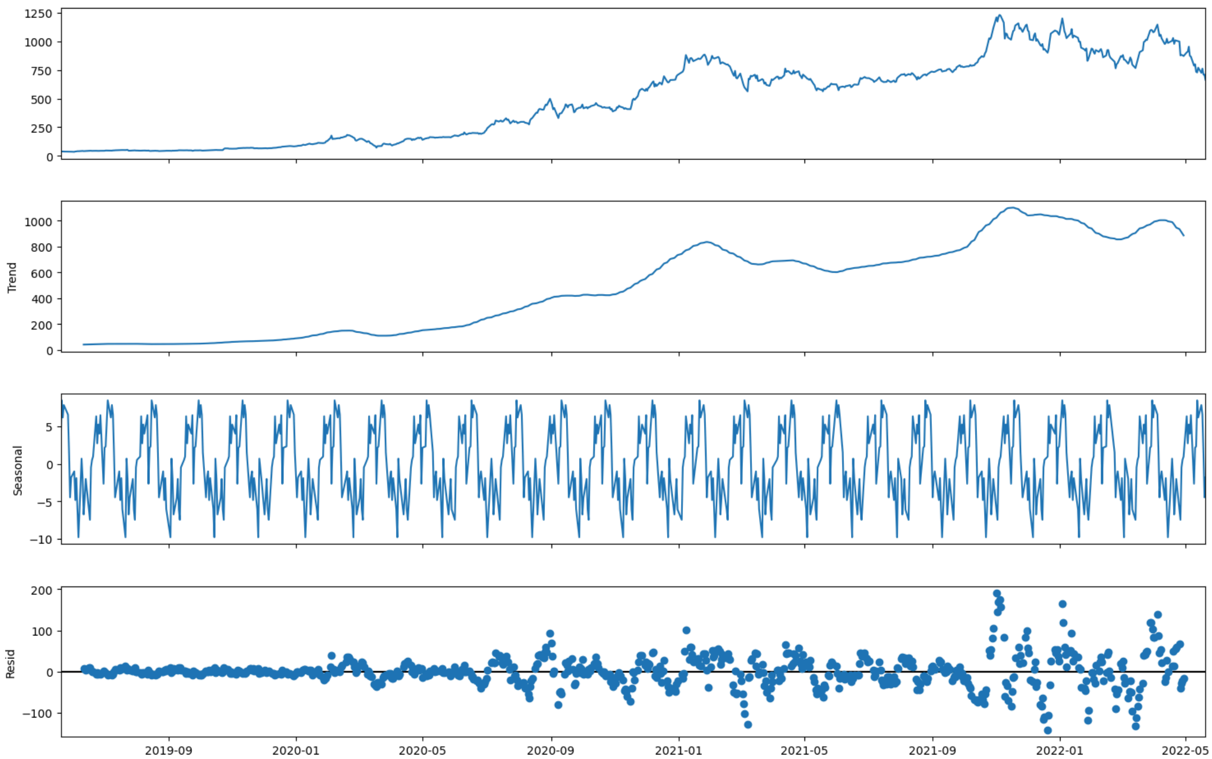
In [29]:

```
result=seasonal_decompose(stock_data[["Close"]], period = 30)
```

In [30]:

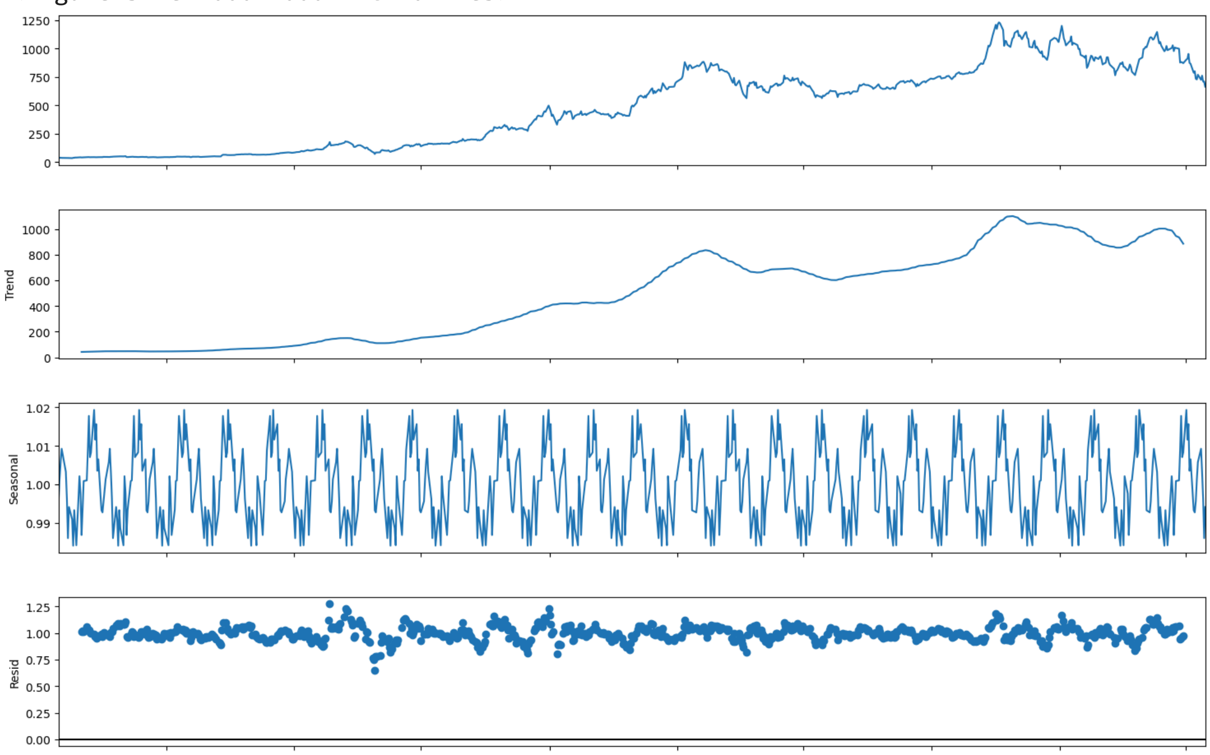
```
fig=plt.figure(figsize=(20,10))
fig=result.plot()
fig.set_size_inches(17,10)
```

```
<Figure size 2000x1000 with 0 Axes>
```



In [31]:
```
result=seasonal_decompose(stock_data[["Close"]],model="multiplicative",period = 30)
fig=plt.figure(figsize=(20,10))
fig=result.plot()
fig.set_size_inches(17,10)
```

```
<Figure size 2000x1000 with 0 Axes>
```



Now we'll create an ARIMA model and train it using the train data's stock closing price. So, let's visualize the data by dividing it into training and test sets.

In [32]:
```
#split data into train and training set
train_data=df_close[0:-60]
test_data=df_close[-60:]
plt.figure(figsize=(18,8))
plt.grid(True)
plt.xlabel('Dates', fontsize = 20)
plt.ylabel('Closing Prices', fontsize = 20)
plt.xticks(fontsize = 15)
plt.xticks(fontsize = 15)
plt.plot(train_data, 'green', label='Train data', linewidth = 5)
plt.plot(test_data, 'blue', label='Test data', linewidth = 5)
plt.legend(fontsize = 20, shadow=True,facecolor='lightpink',edgecolor = 'k')
```

Out[32]: <matplotlib.legend.Legend at 0x288458d1590>