NAME: K ABHIRAM

#**PIN NO : 21X05A6724** COLLAGE : NARASIMHA REDDY ENGINEERING COLLAGE

PROJECT TITLE: **Analysis and prediction of microsoft_stop.csv file which contains stocks and crdits of IT form.Prediction the value of stock in future by 3 category "Open","High","Low"**

DISCRIPTION: The dataset provides the history of daily price of microsoft stock depending on USD.

PROBLEM STATEMENT: This comprehensive dataset provides a detailed analysis of Microsoft Corporation's stock performance from 1986 to 2023. It encompasses various important parameters, including stock price, low price, **high** price, and trading volume, to provide a comprehensive overview of the company's market behavior throughout the years.

The dataset begins in 1986, marking the early years of Microsoft's presence in the stock market. As one of the pioneering companies in the technology industry, Microsoft's stock performance has been closely followed by investors, analysts, and enthusiasts alike. The dataset captures the fluctuations and trends in the stock market, reflecting the company's journey from its inception to its position as a global tech giant.

The stock price data offers a glimpse into the market valuation of Microsoft shares over time. By observing the daily closing prices, one can track the trajectory of the stock and identify key milestones in Microsoft's history. The dataset also includes the lowest and highest prices reached during each trading day, offering insight into the price range within which the stock fluctuated.

Trading volume data provides an additional dimension for understanding Microsoft's stock market activity. It highlights the level of investor interest and participation in buying and selling Microsoft shares during each trading day. Tracking trading volume can help identify periods of increased market activity or significant news events that influenced investor sentiment.

The dataset covers a span of several decades, enabling users to analyze long-term trends, market cycles, and historical patterns that have shaped Microsoft's stock performance. It can be used by researchers, investors, and analysts to conduct quantitative and qualitative studies, perform technical analyses, and gain insights into the dynamics of the technology industry and the broader market.

Please note that this dataset serves as a valuable historical resource and should be utilized alongside other relevant financial information and analysis to make informed decisions. The dataset captures Microsoft's stock performance up until 2023, ensuring that users have access to the latest available information

Time Series Analysis

A Time Series is a set of observations that are collected after regular intervals of time. If plotted, the Time series would always have one of its axes as time. Time Series Analysis in Python considers data collected over time might have some structure; hence it analyses Time Series data to extract its valuable characteristics.

We'll be using the pmdarima library for automatic ARIMA model selection, as it simplifies the process of finding the optimal parameters for the ARIMA model.

```
pip install pmdarima

Requirement already satisfied: pmdarima in
/usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: joblib>=0.11 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (1.3.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (0.29.36)
Requirement already satisfied: numpy>=1.21.2 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (1.23.5)
Requirement already satisfied: pandas>=0.19 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.3)
Requirement already satisfied: scikit-learn>=0.22 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (1.2.2)
Requirement already satisfied: scipy>=1.3.2 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (1.10.1)
Requirement already satisfied: statsmodels>=0.13.2 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.0)
Requirement already satisfied: urllib3 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (2.0.4)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (67.7.2)
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima)
(2023.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22-
>pmdarima) (3.2.0)
Requirement already satisfied: patsy>=0.5.2 in
/usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2-
>pmdarima) (0.5.3)
```

```
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2-
>pmdarima) (23.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-
packages (from patsy>=0.5.2->statsmodels>=0.13.2->pmdarima) (1.16.0)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from pmdarima import auto_arima
```

Loading and Exploring Data

For this example, let's use a sample time series dataset. You can replace this with your own dataset.

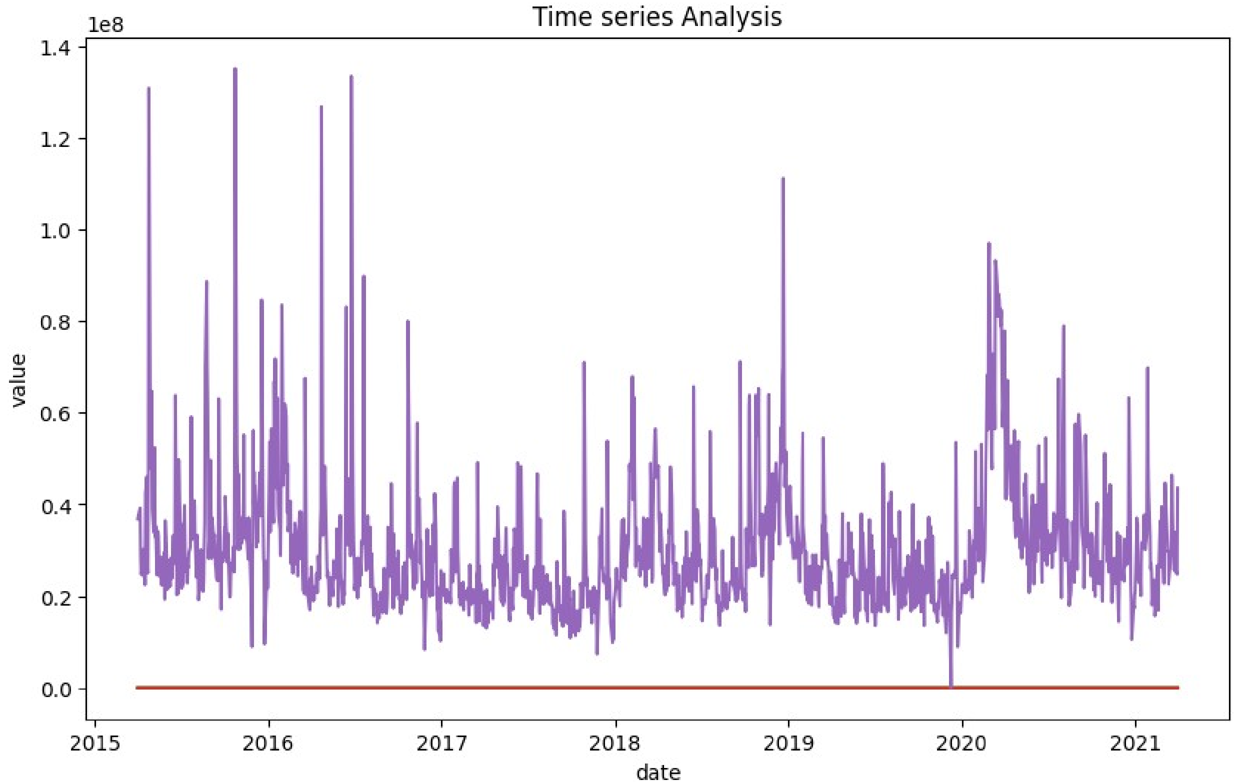
```
# Load your time series data into a pandas DataFrame
# Assume the data has two columns: 'date' and 'value'
data = pd.read_csv('Microsoft_Stock.csv', parse_dates=["Date"]) # add
the parametric series value while working in series frame here we are
useing ["Data"]

# Set 'date' column as the index
data.set_index('Date', inplace=True)

# Display the first few rows of the dataset
print(data.head())
```

```
                      Open   High    Low  Close    Volume
Date
2015-04-01 16:00:00  40.60  40.76  40.31  40.72  36865322
2015-04-02 16:00:00  40.66  40.74  40.12  40.29  37487476
2015-04-06 16:00:00  40.34  41.78  40.18  41.55  39223692
2015-04-07 16:00:00  41.61  41.91  41.31  41.53  28809375
2015-04-08 16:00:00  41.48  41.69  41.04  41.42  24753438
```

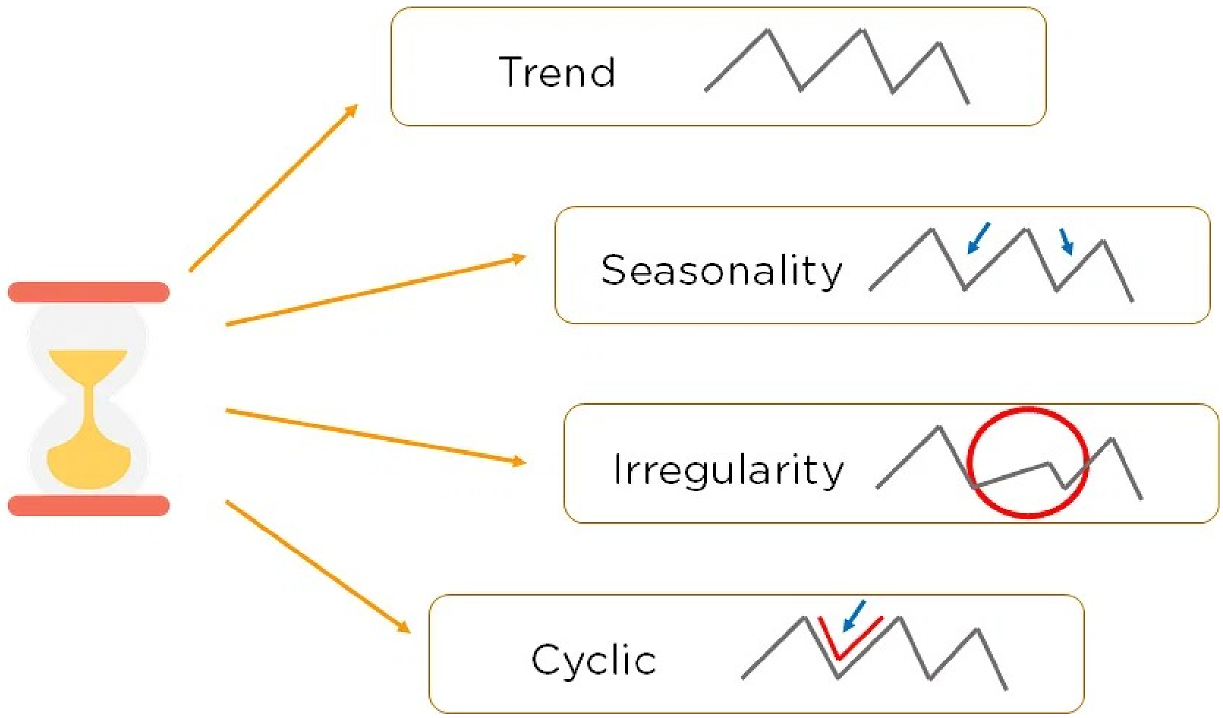
Visualizing Time Series

```
# Plot the time series data

plt.figure(figsize=(10,6))
plt.plot(data)
plt.xlabel("date")
plt.ylabel("value")
```

```
plt.title("Time series Analysis")
plt.show()
```



Decomposing Time Series into Components



Trend: ### The Trend shows the variation of data with time or the frequency of data. Using a Trend, you can see how your data increases or decreases over time. The data can increase, decrease, or remain stable. Over time, population, stock market fluctuations, and production in a company are all examples of trends.

Seasonality:

Seasonality is used to find the variations which occur at regular intervals of time. Examples are festivals, conventions, seasons, etc. These variations usually happen around the same time period and affect the data in specific ways which you can predict.

Irregularity:

Fluctuations in the time series data do not correspond to the trend or seasonality. These variations in your time series are purely random and usually caused by unforeseeable circumstances, such as a sudden decrease in population because of a natural calamity.

Cyclic:

Oscillations in time series which last for more than a year are called cyclic. They may or may not be periodic.

Stationary:

A time series that has the same statistical properties over time is stationary. The properties remain the same anywhere in the series. Your data needs to be stationary to perform time-series analysis on it. A stationary series has a constant mean, variance, and covariance.

#Decomposition in Time series analysis: ## The decomposition of time series is a statistical task that deconstructs a time series into several components, each representing one of the underlying categories of patterns.

The syntax of decomposition is:

decomposition = seasonal_decompose(data[" PARTICULAR COLUMN NAME"], model='additive', period= 12)

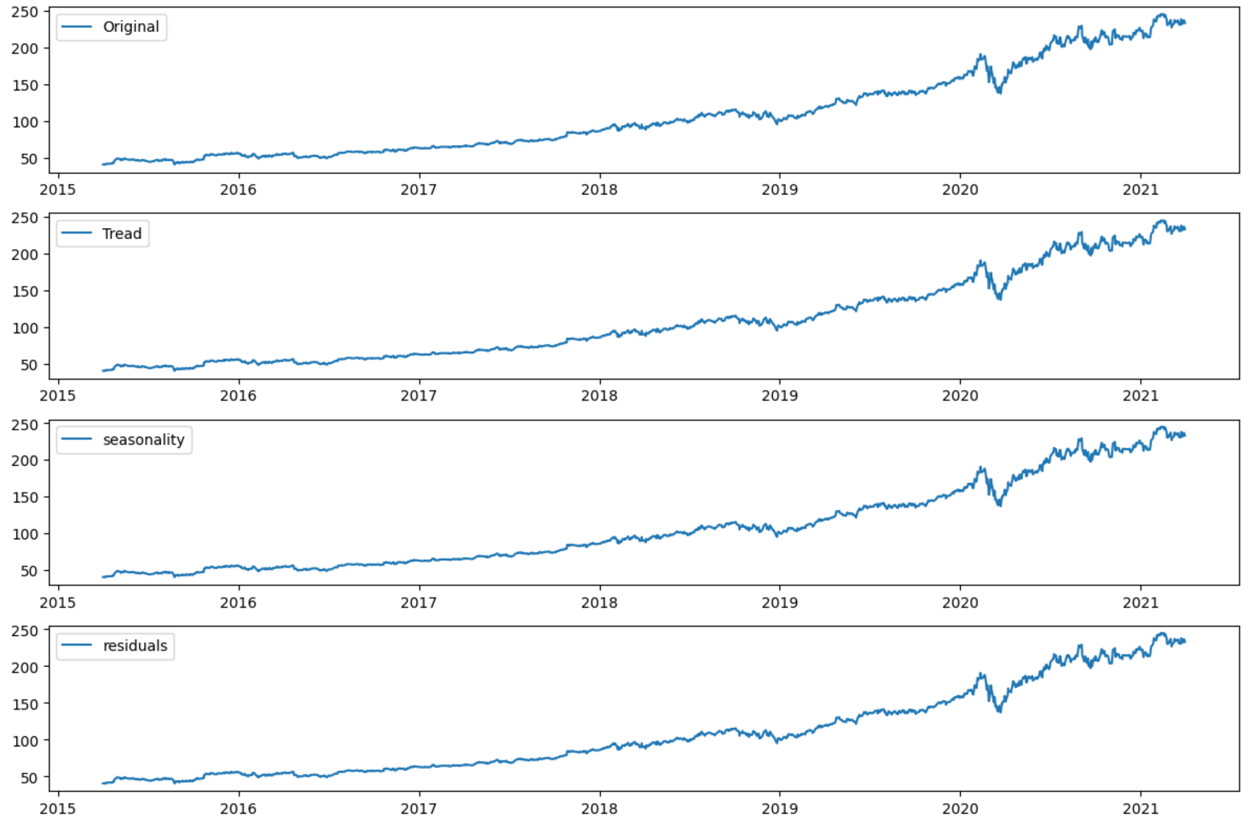
seasonal for types of Time series, model is alway be default for additive and period ios 12 months of in one year.

plt.subplot(nnn)

1. The first digit (n) represents the number of rows in the grid.
2. The second digit (n) represents the number of columns in the grid.
3. The third and last digit (n) represents the index of the current subplot within the grid.

#ON BASES OPEN

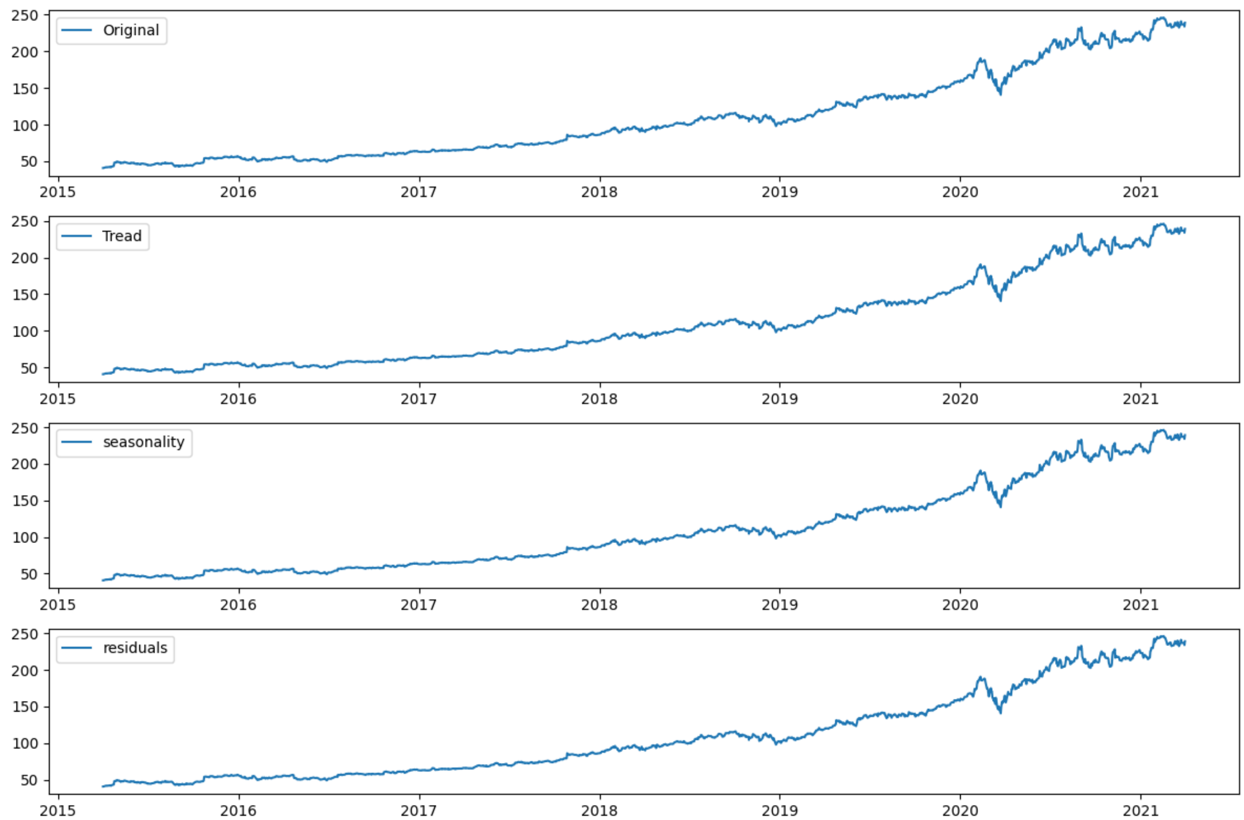
```
from scipy.signal import residue
# Perform seasonal decomposition
decomposition =
seasonal_decompose(data["Open"],model='additive',period=12)
# The decomposition of time series is a statistical task that
deconstructs a time series into several components, each representing
one of the underlying categories of patterns.
  # Assuming seasonality of 12 months

# Plot the decomposed components
trend = decomposition.trend
seasonal = decomposition.seasonal
residue = decomposition.resid

# plot the figure for visualization.
plt.figure(figsize=(12, 8))
plt.subplot(411)
plt.plot(data['Open'], label='Original')
plt.legend(loc='upper left')


# Visualize the trend .
plt.subplot(412)
plt.plot(data['Open'], label='Tread')
plt.legend(loc='upper left')

# visualioze the Seasonality
plt.subplot(413)
plt.plot(data['Open'], label='seasonality')
plt.legend(loc='upper left')


# Visualize the Residuals.
plt.subplot(414)
plt.plot(data['Open'], label='residuals')
plt.legend(loc='upper left')

plt.tight_layout()
plt.show()
```



#ON BASES HIGH

```
# Perform seasonal decomposition
decomposition =
seasonal_decompose(data["High"],model='additive',period=12)
# The decomposition of time series is a statistical task that
deconstructs a time series into several components, each representing
one of the underlying categories of patterns.
  # Assuming seasonality of 12 months

# Plot the decomposed components
trend = decomposition.trend
seasonal = decomposition.seasonal
residue = decomposition.resid

# plot the figure for visualization.
plt.figure(figsize=(12, 8))
plt.subplot(411)
plt.plot(data['High'], label='Original')
plt.legend(loc='upper left')


# Visualize the trend .
plt.subplot(412)
plt.plot(data['High'], label='Tread')
```

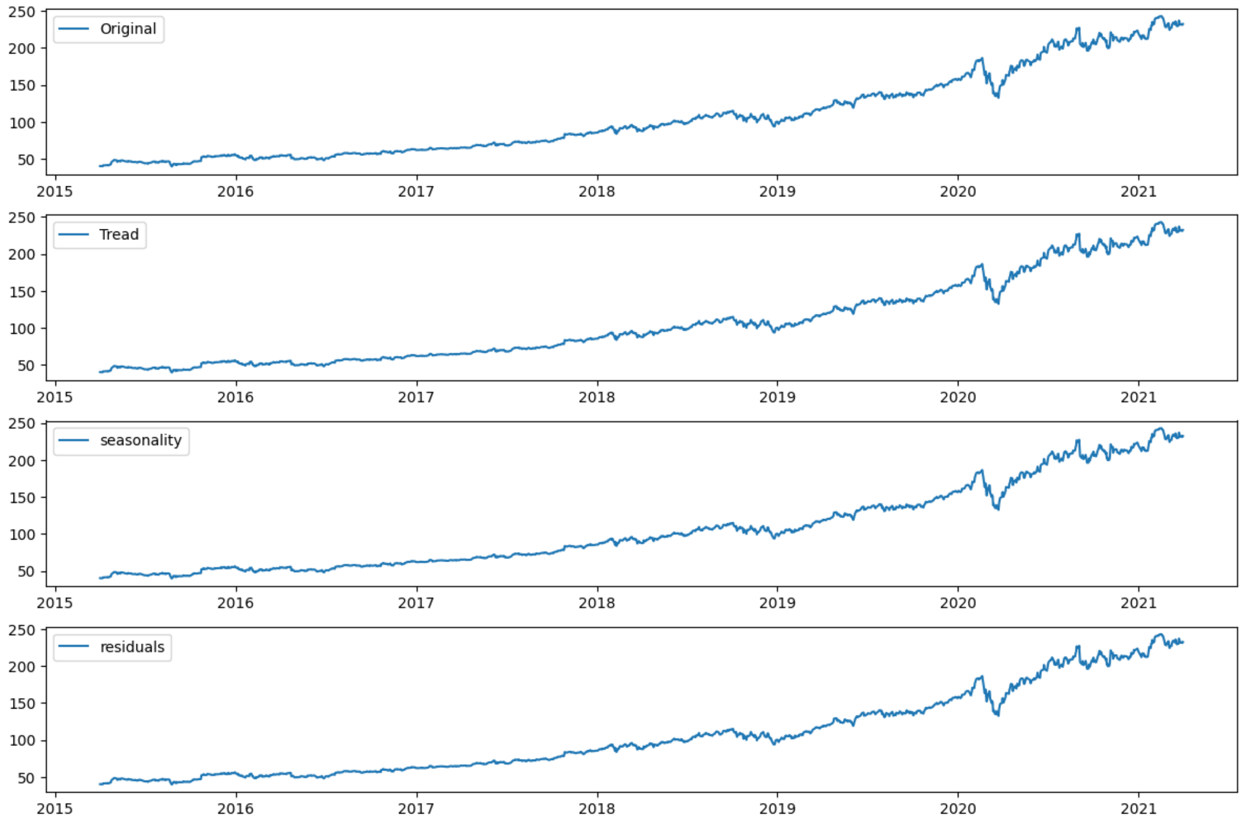
```
plt.legend(loc='upper left')

# visualioze the Seasonality
plt.subplot(413)
plt.plot(data['High'], label='seasonality')
plt.legend(loc='upper left')


# Visualize the Residuals.
plt.subplot(414)
plt.plot(data['High'], label='residuals')
plt.legend(loc='upper left')

plt.tight_layout()
plt.show()
```



#ON BASES LOW

```
# Perform seasonal decomposition
decomposition =
seasonal_decompose(data["Low"],model='additive',period=12)
# The decomposition of time series is a statistical task that
deconstructs a time series into several components, each representing
one of the underlying categories of patterns.
```

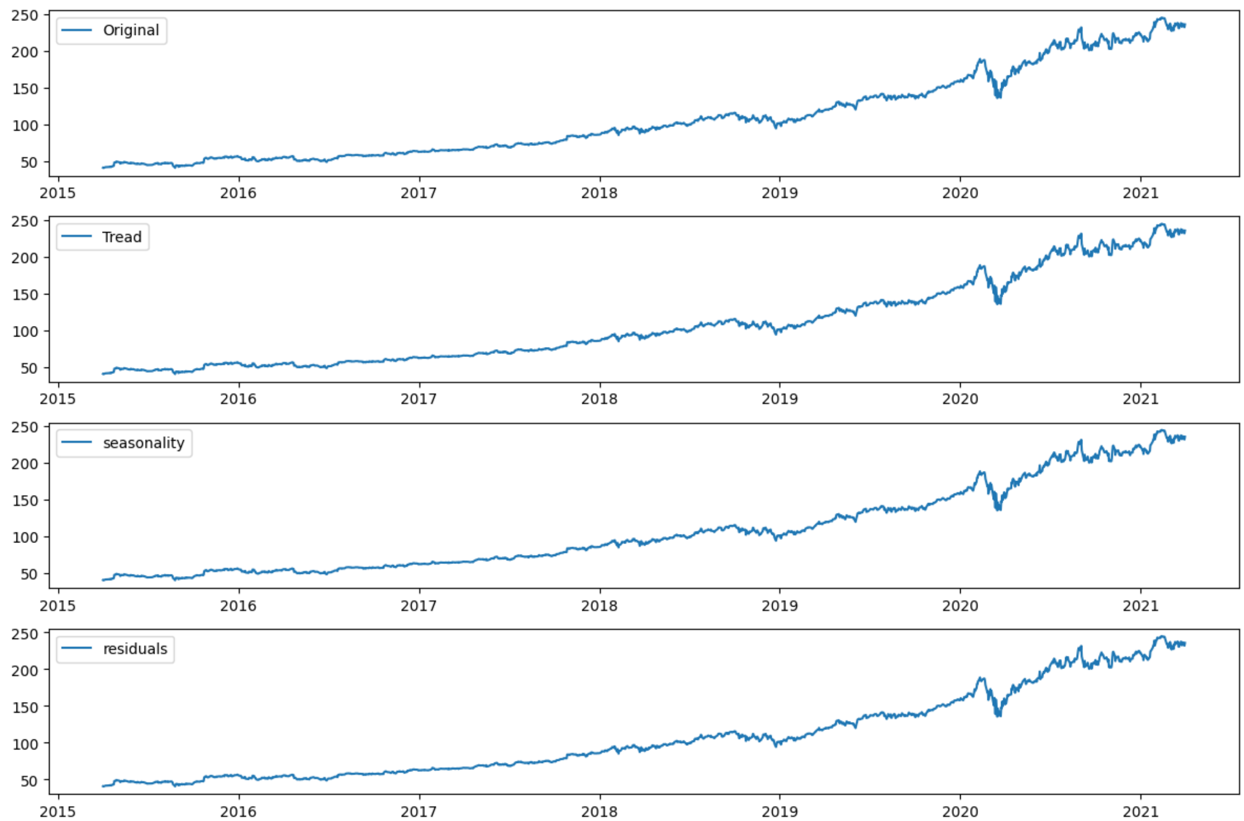
```
  # Assuming seasonality of 12 months

# Plot the decomposed components
trend = decomposition.trend
seasonal = decomposition.seasonal
residue = decomposition.resid

# plot the figure for visualization.
plt.figure(figsize=(12, 8))
plt.subplot(411)
plt.plot(data['Low'], label='Original')
plt.legend(loc='upper left')


# Visualize the trend .
plt.subplot(412)
plt.plot(data['Low'], label='Tread')
plt.legend(loc='upper left')

# visualioze the Seasonality
plt.subplot(413)
plt.plot(data['Low'], label='seasonality')
plt.legend(loc='upper left')


# Visualize the Residuals.
plt.subplot(414)
plt.plot(data['Low'], label='residuals')
plt.legend(loc='upper left')

plt.tight_layout()
plt.show()
```



#ON BASES CLOSE

```
# Perform seasonal decomposition
decomposition =
seasonal_decompose(data["Close"],model='additive',period=12)
# The decomposition of time series is a statistical task that
deconstructs a time series into several components, each representing
one of the underlying categories of patterns.
  # Assuming seasonality of 12 months

# Plot the decomposed components
trend = decomposition.trend
seasonal = decomposition.seasonal
residue = decomposition.resid

# plot the figure for visualization.
plt.figure(figsize=(12, 8))
plt.subplot(411)
plt.plot(data['Close'], label='Original')
plt.legend(loc='upper left')


# Visualize the trend .
plt.subplot(412)
plt.plot(data['Close'], label='Tread')
```

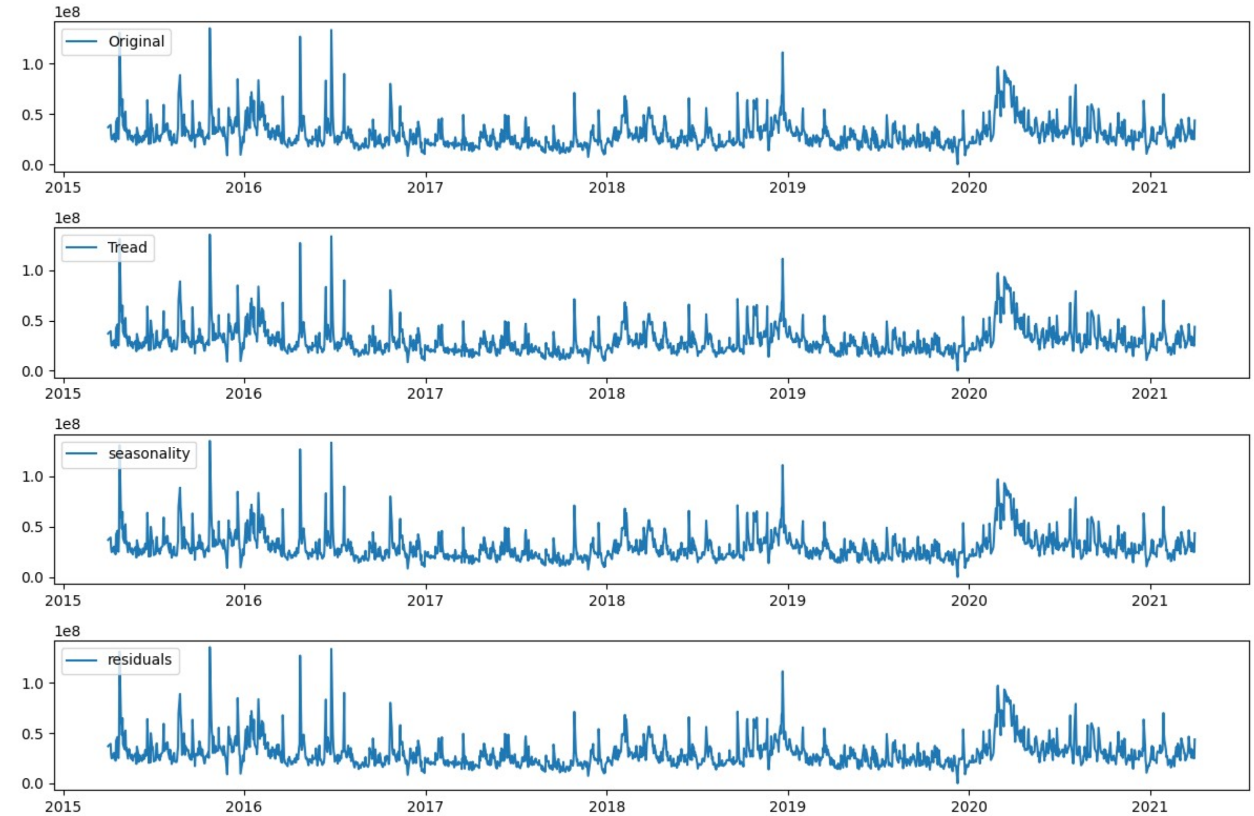
```
plt.legend(loc='upper left')

# visualioze the Seasonality
plt.subplot(413)
plt.plot(data['Close'], label='seasonality')
plt.legend(loc='upper left')


# Visualize the Residuals.
plt.subplot(414)
plt.plot(data['Close'], label='residuals')
plt.legend(loc='upper left')

plt.tight_layout()
plt.show()
```



#ON BASES VOLUME

```
# Perform seasonal decomposition
decomposition =
seasonal_decompose(data["Volume"],model='additive',period=12)
# The decomposition of time series is a statistical task that
deconstructs a time series into several components, each representing
one of the underlying categories of patterns.
```

```
  # Assuming seasonality of 12 months

# Plot the decomposed components
trend = decomposition.trend
seasonal = decomposition.seasonal
residue = decomposition.resid

# plot the figure for visualization.
plt.figure(figsize=(12, 8))
plt.subplot(411)
plt.plot(data['Volume'], label='Original')
plt.legend(loc='upper left')


# Visualize the trend .
plt.subplot(412)
plt.plot(data['Volume'], label='Tread')
plt.legend(loc='upper left')

# visualioze the Seasonality
plt.subplot(413)
plt.plot(data['Volume'], label='seasonality')
plt.legend(loc='upper left')


# Visualize the Residuals.
plt.subplot(414)
plt.plot(data['Volume'], label='residuals')
plt.legend(loc='upper left')

plt.tight_layout()
plt.show()
```



CONCLUSION-1: According to the time series analysis of data for the particular dataset found that "Open", "High", "Low", "Close" are the columns in which trends in increasing manner.But in the year 2020-2021 the stocks are goes in decreasing manner

#####**CONCLUSION-2**:According the volume chart for particular this dataset year 2016-17 gives little bit off more profit to investement in a stocks. After 2017 to up coming year stocks will be fluctuated