Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Subject: Time Series Analysis

Experiment 2

(Seasonality)

D 12
60009210105
Amitesh
Sawarkar

Aim: Seasonality:

(1) Multiple Box Plots
(2) Autocorrelation Plot
(3) Deseasoning of Time-Series Data
(4) Seasonal Decomposition (Additive and Multiplicative)
  i. Trend
  ii. Seasonal Index
  iii. Residual
(5) Detecting Cyclic Variations

Theory:

Seasonality:
Seasonality is a periodical fluctuation where the same pattern occurs at a regular interval of time. It is a characteristic of economics, weather, and stock market time-series data; less often, it's observed in scientific data. In other industries, many phenomena are characterized by periodically recurring seasonal effects.

For example, retail sales tend to increase during Christmas and decrease afterward. The following methods can be used to detect seasonality:

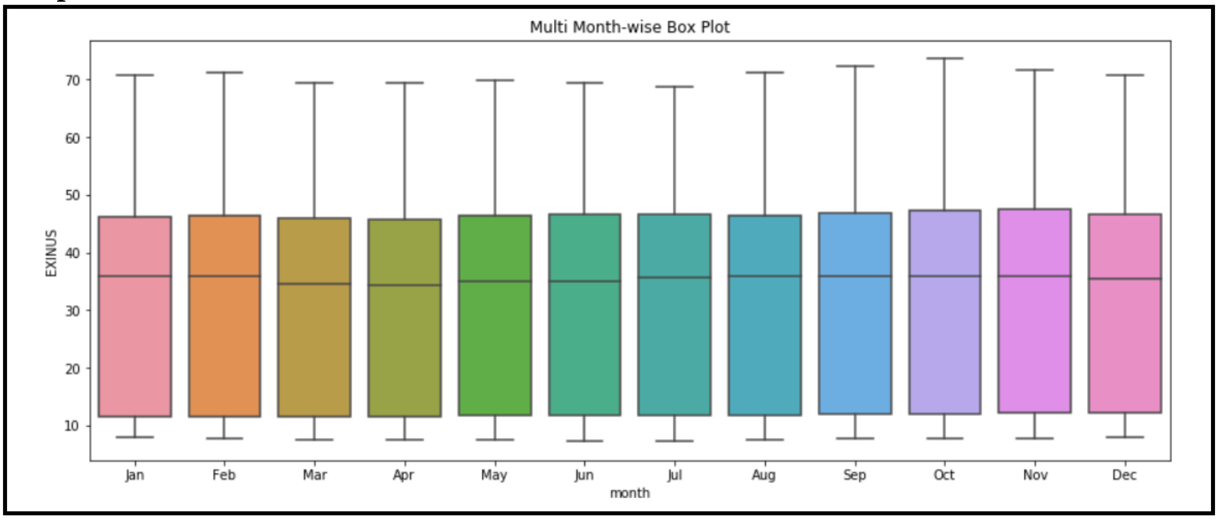
- Multiple box plots
- Autocorrelation plots

Detecting Seasonality using Multiple Box Plots:
A box plot is an essential graph to depict data spread out over a range. It is a standard approach to showing the minimum, first quartile, middle, third quartile, and maximum. The following code shows an example of detecting seasonality with the help of multiple box plots. **Seaborn** is a graphical representation package similar to Matplotlib.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.tsa.filters.hp_filter import hpfilter
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
df = pd.read_excel(r'India_Exchange_Rate_Dataset.xls',parse_dates=True)
df['month'] = df['observation_date'].dt.strftime('%b')
df['year'] = [d.year for d in df.observation_date]
df['month'] = [d.strftime('%b') for d in df.observation_date]
years = df['year'].unique()
plt.figure(figsize=(15,6))
```

```
sns.boxplot(x='month', y='EXINUS', data=df).set_title("Multi Month-wise Box Plot")
plt.show()
```

Output:

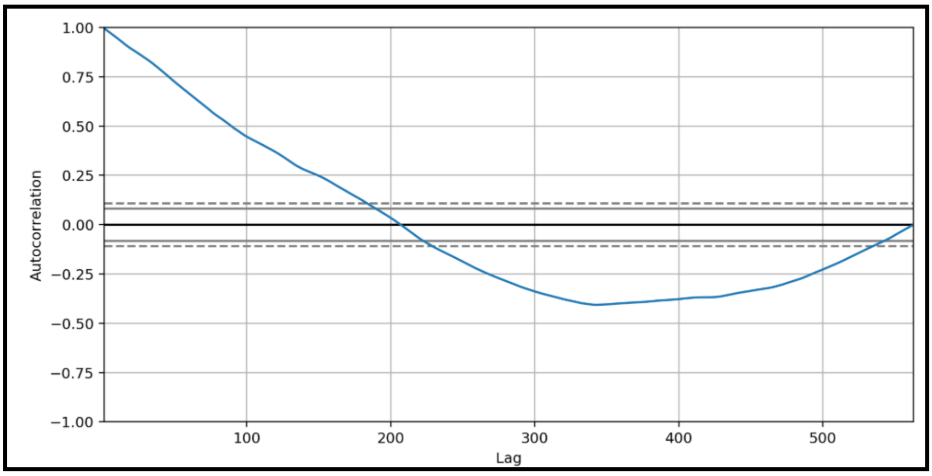


Detecting Seasonality using Autocorrelation Plot:

Autocorrelation is used to check randomness in data. It helps to identify types of data where the period is not known. For instance, for the monthly data, if there is a regular seasonal effect, we would hope to see massive peak lags after every 12 months.

```
from pandas.plotting import autocorrelation_plot
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
df = pd.read_excel(r'India_Exchange_Rate_Dataset.xls',
index_col=0,parse_dates=True)
plt.rcParams.update({'figure.figsize':(15,6)})
autocorrelation_plot(df.EXINUS.tolist())
```

Output:



Deseasoning of Time-Series Data:

Deseasoning means to remove seasonality from time-series data. It is stripped of the pattern of seasonal effect to deseasonalize the impact. Time-series data contains four main components.

- **Level** means the average value of the time-series data.
- **Trend** means an increasing or decreasing value in time-series data.
- **Seasonality** means repeating the pattern of a cycle in the time-series data.
- **Noise** means random variance in time-series data.

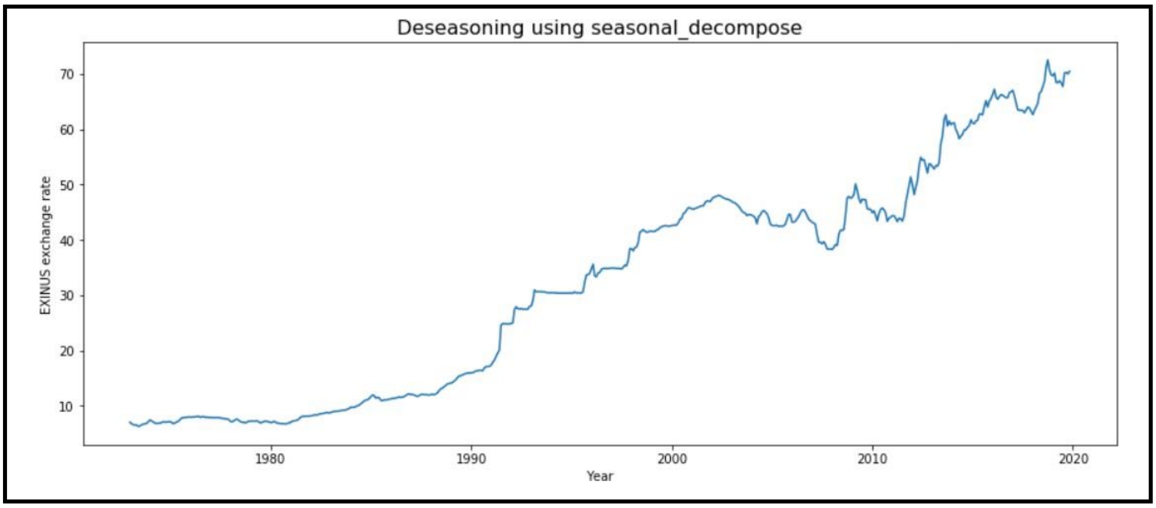
An additive model is when time-series data combines these four components for linear trend and seasonality, and a multiplicative model is when components are multiplied to gather for nonlinear trends and seasonality.

Seasonal Decomposition:

Decomposition is the process of understanding generalizations and problems related to time-series forecasting. We can leverage seasonal decomposition to remove seasonality from data and check the data only with the trend, cyclic, and irregular variations.

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
df =
pd.read_excel(r'India_Exchange_Rate_Dataset.xls',index_col=0,parse_dates=True)
result_mul = seasonal_decompose(df['EXINUS'], model='multiplicative')
deseason = df['EXINUS'] - result_mul.seasonal
plt.figure(figsize=(15,6))
plt.plot(deseason)
plt.title('Deseasoning using seasonal_decompose', fontsize=16)
plt.xlabel('Year')
plt.ylabel('EXINUS exchange rate')
plt.show()
```

Output:



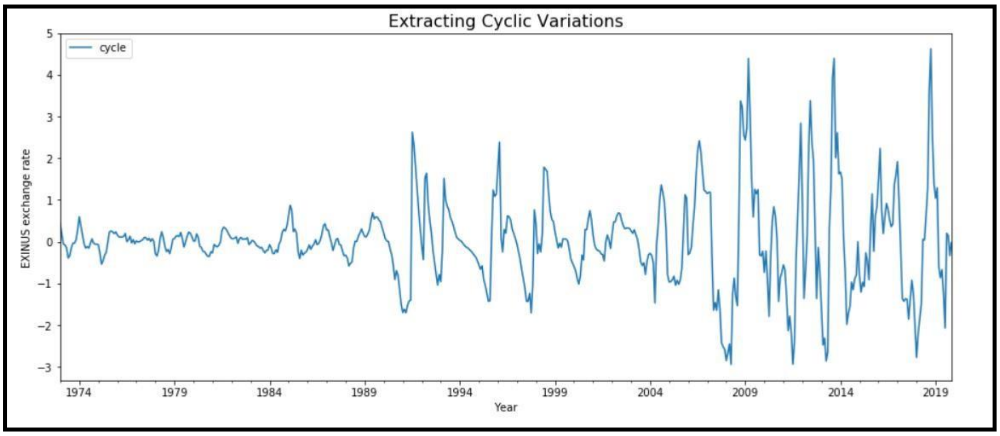
Detecting Cyclic Variations:

Cyclical components are fluctuations around a long trend observed every few units of time; this behavior is less frequent compared to seasonality. It is a recurrent process in a time series. In the field of business/economics, the following are three distinct types of cyclic variations examples:

- **Prosperity:** As we know, when organizations prosper, prices go up, but the benefits also increase. On the other hand, prosperity also causes over-development, challenges in transportation, increments in wage rate, insufficiency in labor, high rates of returns, deficiency of cash in the market and price concessions, etc., leading to depression
- **Depression:** As we know, when there is cynicism in exchange and enterprises, processing plants close down, organizations fall flat, joblessness spreads, and the wages and costs are low.
- **Accessibility:** This causes idealness of money, accessibility of cash at a low interest, an increase in demand for goods or money at a low interest rate, an increase in popular merchandise and ventures described by the circumstance of recuperation that at last prompts for prosperity or boom.

```
from statsmodels.tsa.filters.hp_filter import hpfilter
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
df =
pd.read_excel(r'India_Exchange_Rate_Dataset.xls',index_col=0,parse_dates=True)
EXINUS_cycle,EXINUS_trend = hpfilter(df['EXINUS'], lamb=1600)
df['cycle'] =EXINUS_cycle
df['trend'] =EXINUS_trend
df[['cycle']].plot(figsize=(15,6)).autoscale(axis='x',tight=True)
plt.title('Extracting Cyclic Variations', fontsize=16)
plt.xlabel('Year')
plt.ylabel('EXINUS exchange rate')
plt.show()
```

Output:



Lab Assignments to complete:

Perform the following tasks using the datasets mentioned. Download the datasets from the link given:

Link:
https://drive.google.com/drive/folders/1dbqJuZJULas76_Zzkqs-yRd2DbJReJup?usp=sharing

Dataset 1: Facebook Stock Market Performance

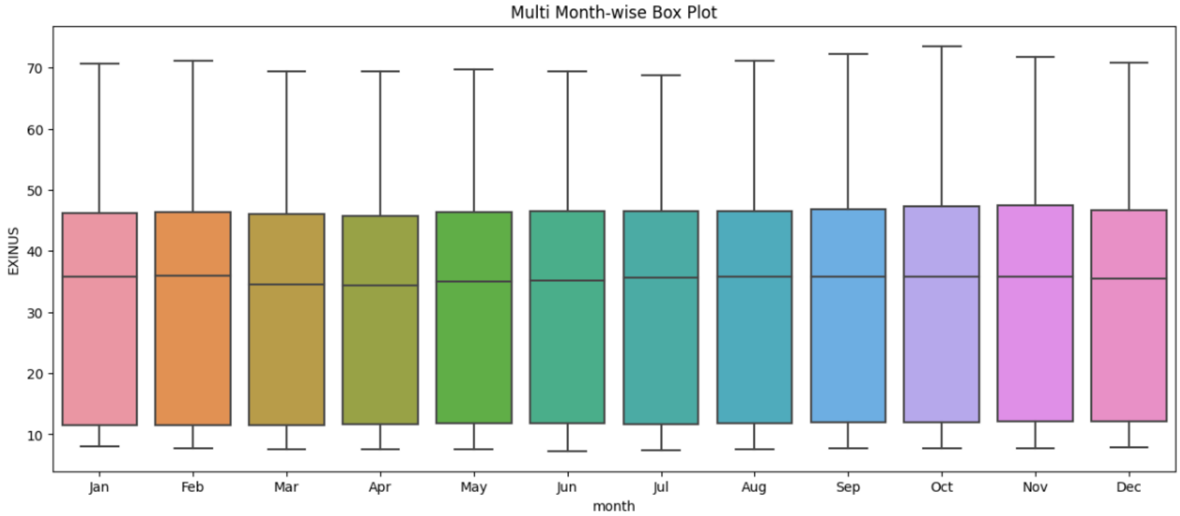
1. Simulate the time series components from scratch.
2. Create the three decomposition models from scratch.
3. Implement seasonality decomposition using the seasonal_decompose() method.

Dataset 2: India Exchange Rate Dataset

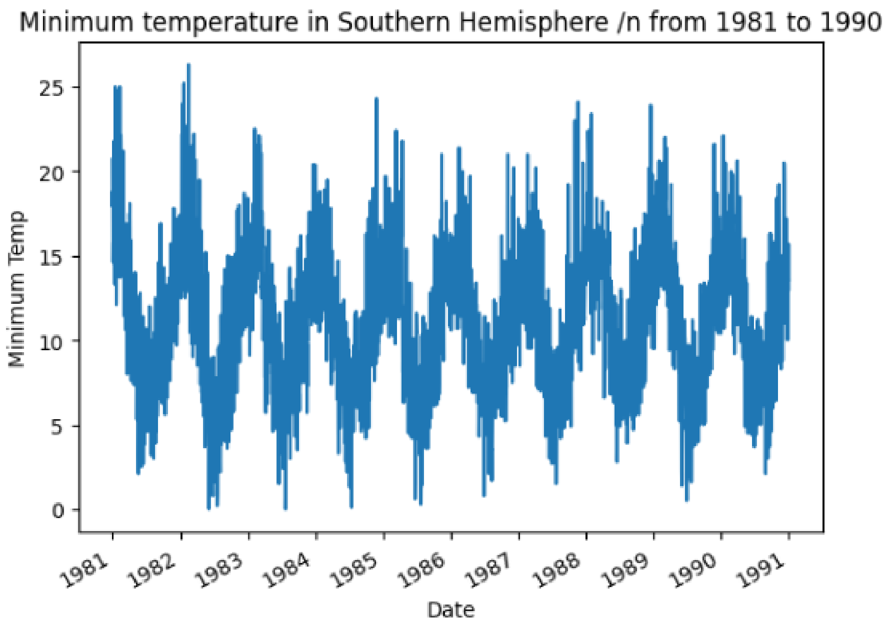
1. Implement seasonality decomposition on the mentioned dataset (using additive and multiplicative decomposition).

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.tsa.filters.hp_filter import hpfilter
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
df = pd.read_excel(r'/content/India_Exchange_Rate_Dataset.xls', parse_dates = True)
df["month"] = df["observation_date"].dt.strftime("%b")
df['year'] = [d.year for d in df.observation_date]
years = df['year'].unique()
```

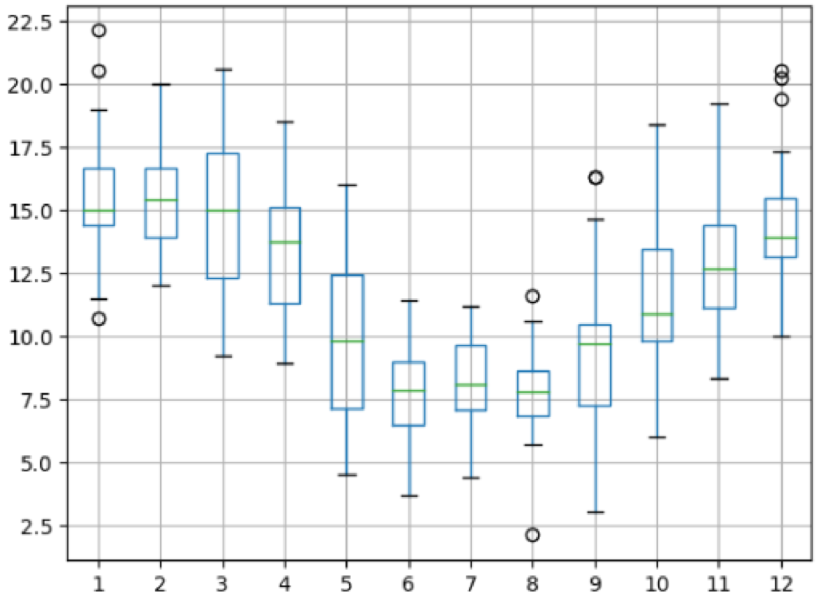
```
plt.figure(figsize=(15, 6))
sns.boxplot(x="month", y='EXINUS', data=df).set_title("Multi Month-wise Box Plot")
plt.show()
```




```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.tsa.filters.hp_filter import hpfilter
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline

series = pd.read_csv('/content/daily-min-temperatures.csv', header=0, index_col = 0, parse_dates = True, squeeze=True)
series.plot()
plt.ylabel("Minimum Temp")
plt.title("Minimum temperature in Southern Hemisphere /n from 1981 to 1990")
plt.show()
```




```
months = pd.DataFrame()
one_year = series['1990']
groups = one_year.groupby(pd.Grouper(freq='M'))
months = pd.concat([pd.DataFrame(x[1].values) for x in groups], axis=1)
months = pd.DataFrame(months)
months.columns = range(1, 13)
months.boxplot()
plt.show()
```

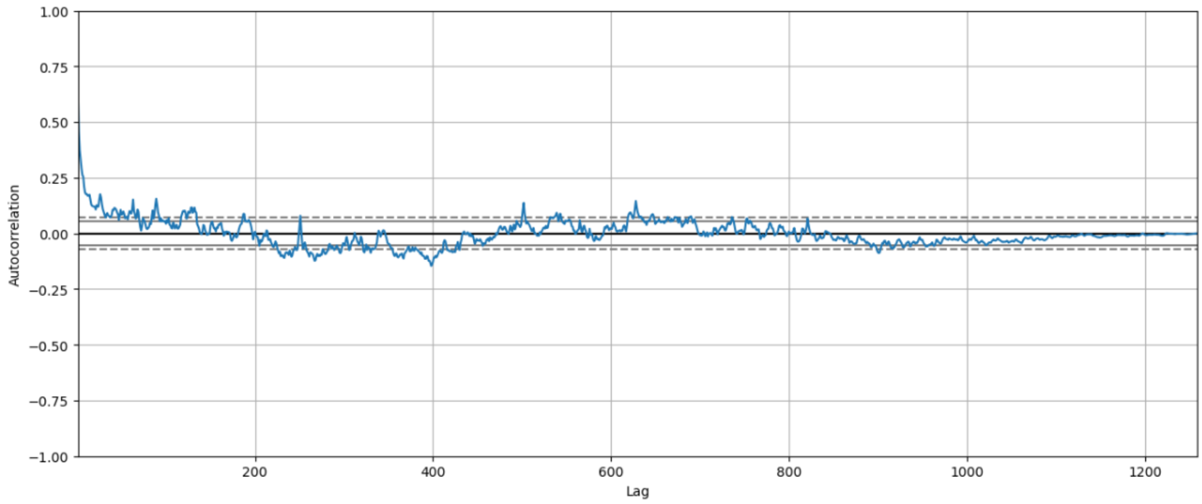


```python
from pandas.plotting import autocorrelation_plot
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline


df = pd.read_csv('/content/FB.csv', index_col = 0, parse_dates = True)
df.head()
```

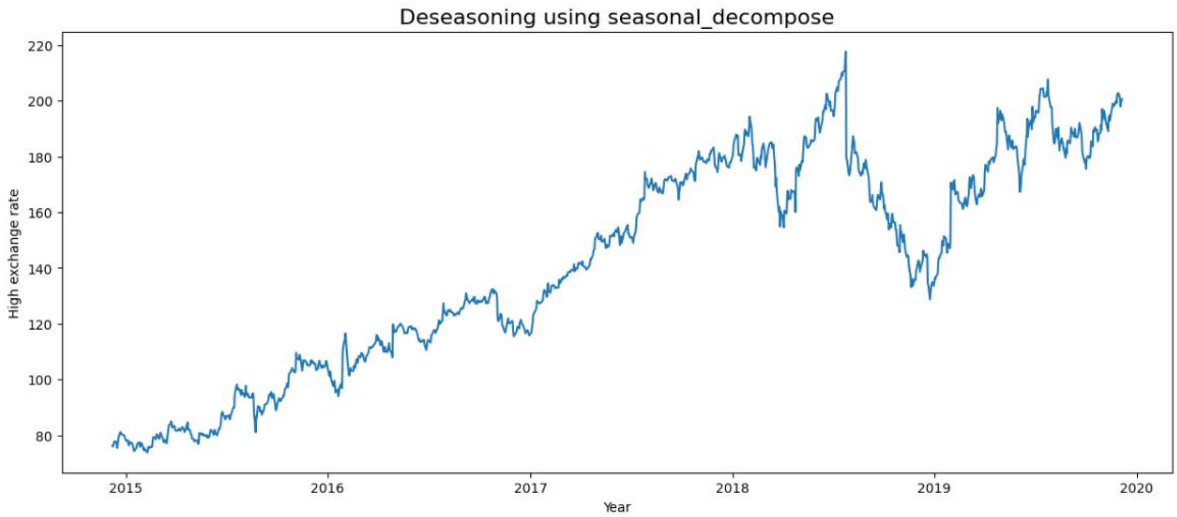
	Open	High	Low	Close	Adj Close	Volume
Date						
2014-12-08	76.180000	77.250000	75.400002	76.519997	76.519997	25733900
2014-12-09	75.199997	76.930000	74.779999	76.839996	76.839996	25358600
2014-12-10	76.650002	77.550003	76.070000	76.180000	76.180000	32210500
2014-12-11	76.519997	78.519997	76.480003	77.730003	77.730003	33462100
2014-12-12	77.160004	78.879997	77.019997	77.830002	77.830002	28091600

```python
plt.rcParams.update({'figure.figsize':(15, 6)})
autocorrelation_plot(df.Volume.tolist())
```

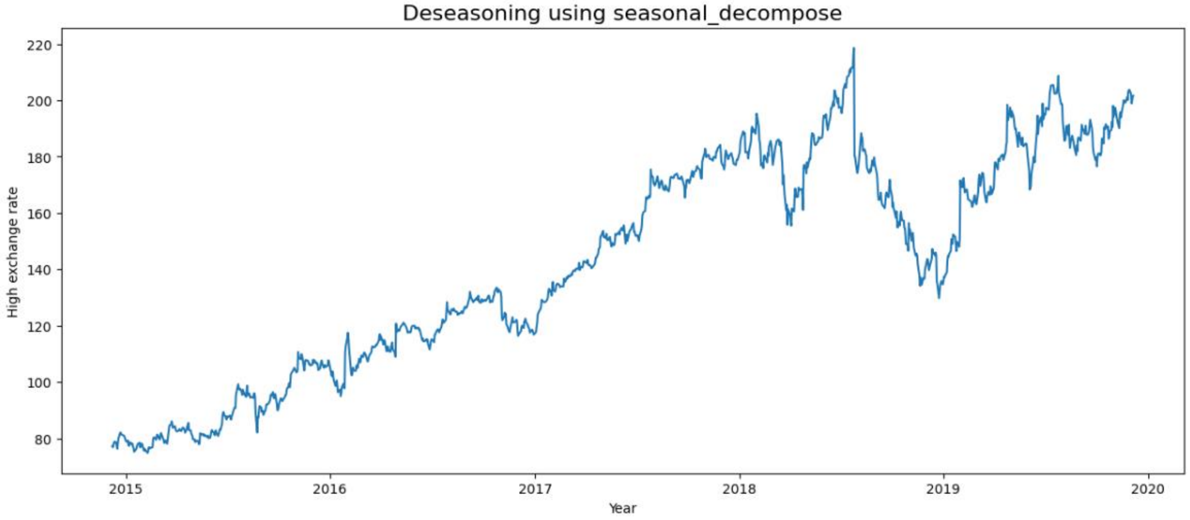
<Axes: xlabel='Lag', ylabel='Autocorrelation'>




```python
from statsmodels.tsa.seasonal import seasonal_decompose
df = pd.read_csv(r'/content/FB.csv', parse_dates = True, index_col=0)
result_mul = seasonal_decompose(df['High'], model='multiplicative', period=2)
deseason = df['High'] - result_mul.seasonal
plt.figure(figsize=(15,6))
plt.plot(deseason)
plt.title('Deseasoning using seasonal_decompose', fontsize=16)
plt.xlabel('Year')
plt.ylabel('High exchange rate')
plt.show()
```



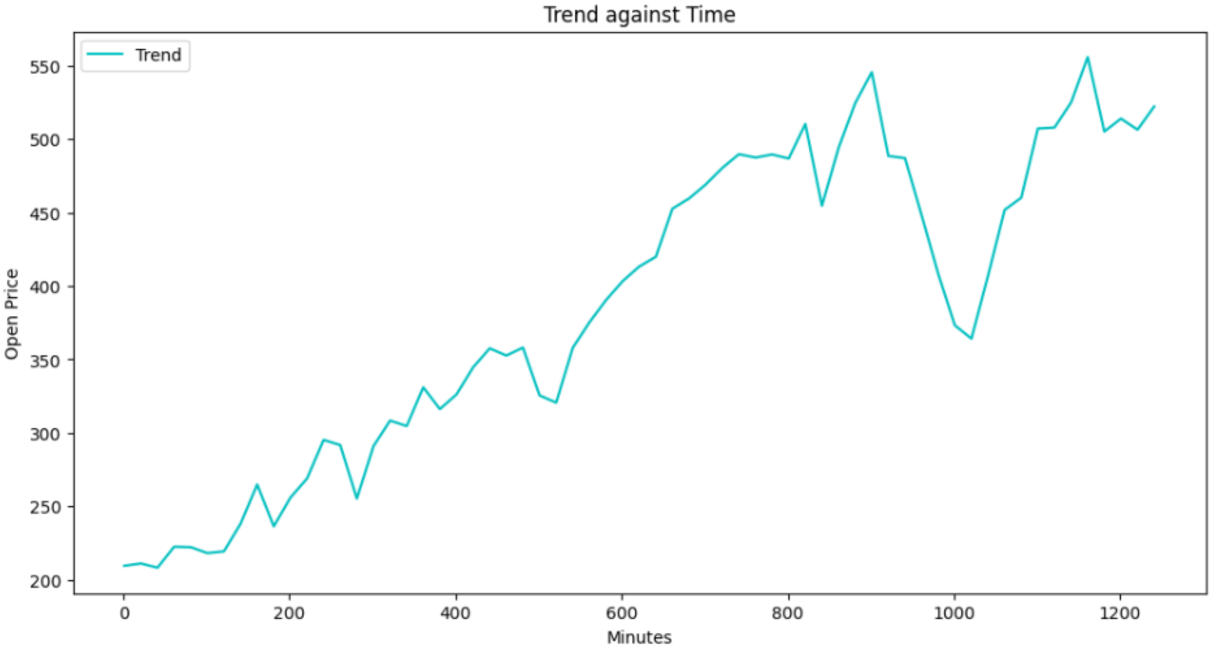

```python
from statsmodels.tsa.seasonal import seasonal_decompose
df = pd.read_csv(r'/content/FB.csv', parse_dates = True, index_col=0)
result_mul = seasonal_decompose(df['High'], model='additive', period=2)
deseason = df['High'] - result_mul.seasonal
plt.figure(figsize=(15,6))
plt.plot(deseason)
plt.title('Deseasoning using seasonal_decompose', fontsize=16)
plt.xlabel('Year')
plt.ylabel('High exchange rate')
plt.show()
```

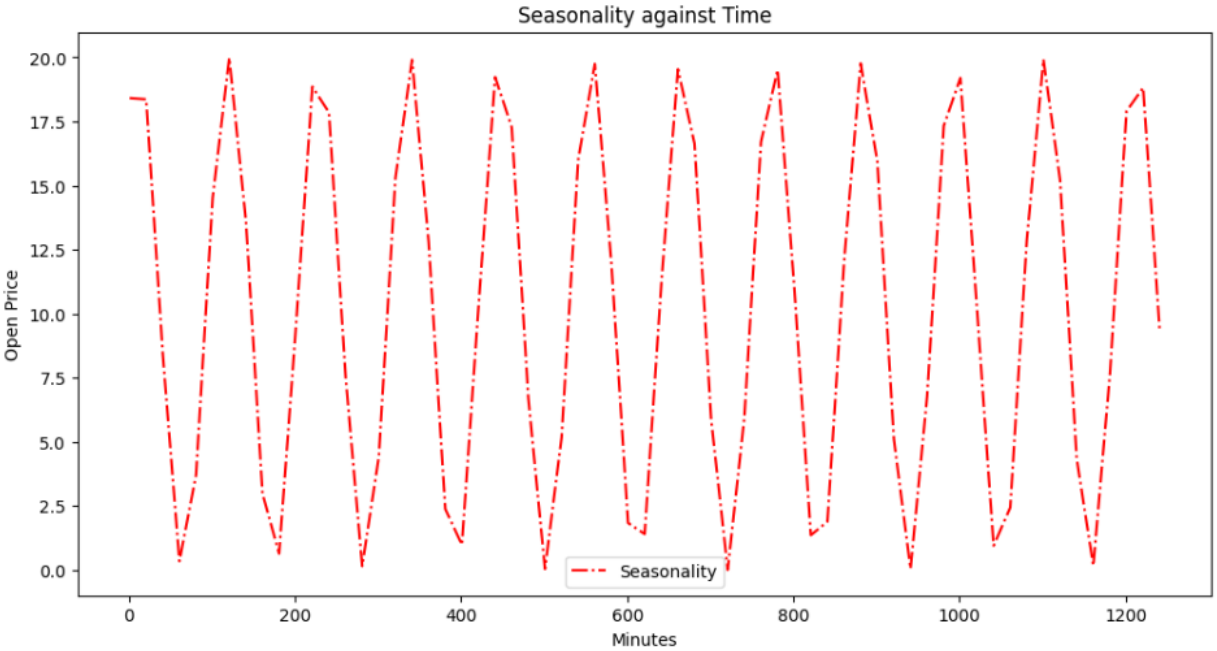


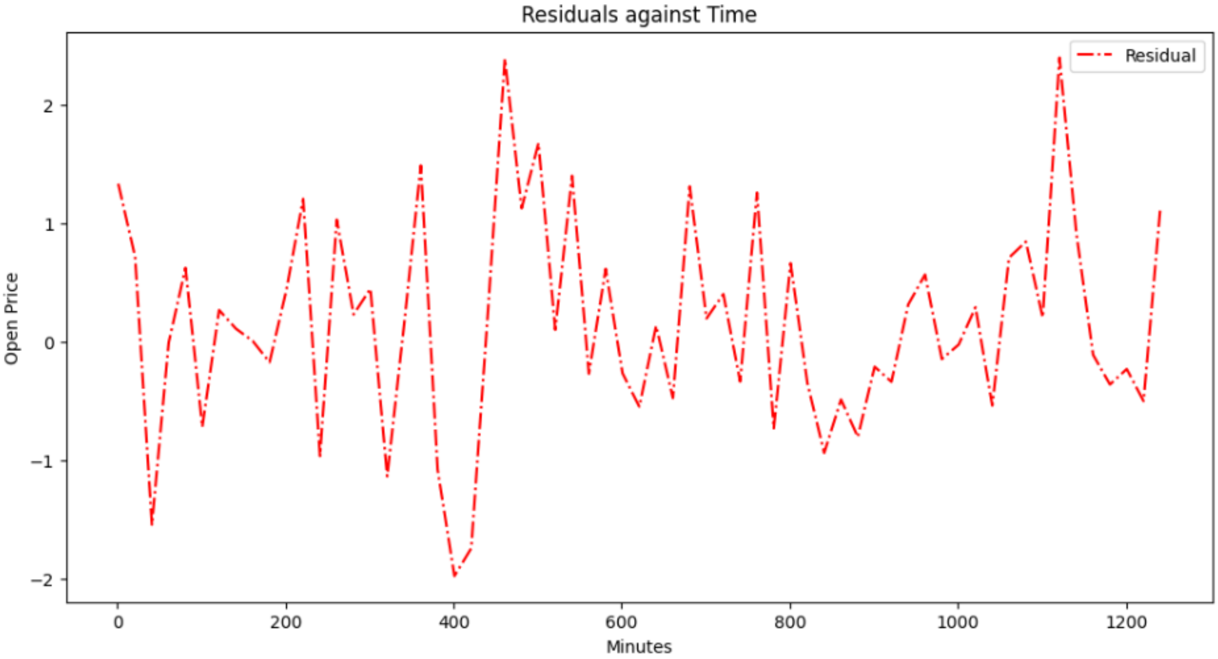

```python
#Decomposition from scratch
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv('/content/FB.csv')

T_Series = np.arange(1, len(df) + 1)
Trend = df['Open'] * 2.75

subsample_factor = 20
T_Series = T_Series[::subsample_factor]
Trend = Trend[::subsample_factor]

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

plt.plot(T_Series, Trend, 'c-', label="Trend")
plt.title("Trend against Time")
plt.xlabel("Minutes")
plt.ylabel("Open Price")
plt.legend()
plt.show()
```

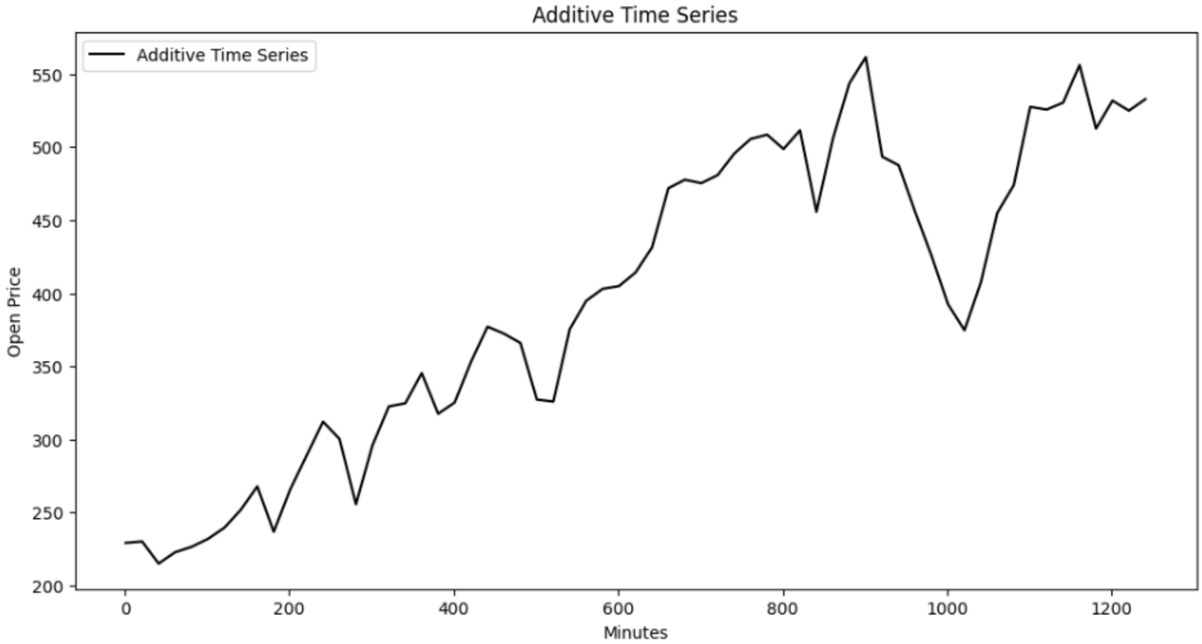
```
seasonal_pattern = np.sin(T_Series) * 10
seasonality = 10 + seasonal_pattern

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

plt.plot(T_Series, seasonality, 'r-.', label="Seasonality")
plt.title("Seasonality against Time")
plt.xlabel("Minutes")
plt.ylabel("Open Price")
plt.legend()
plt.show()

np.random.seed(10)
residual = np.random.normal(loc=0.0, scale=1, size=len(T_Series))

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

plt.plot(T_Series, residual, 'r-.', label="Residual")
plt.title("Residuals against Time")
plt.xlabel("Minutes")
plt.ylabel("Open Price")
plt.legend()
plt.show()
```

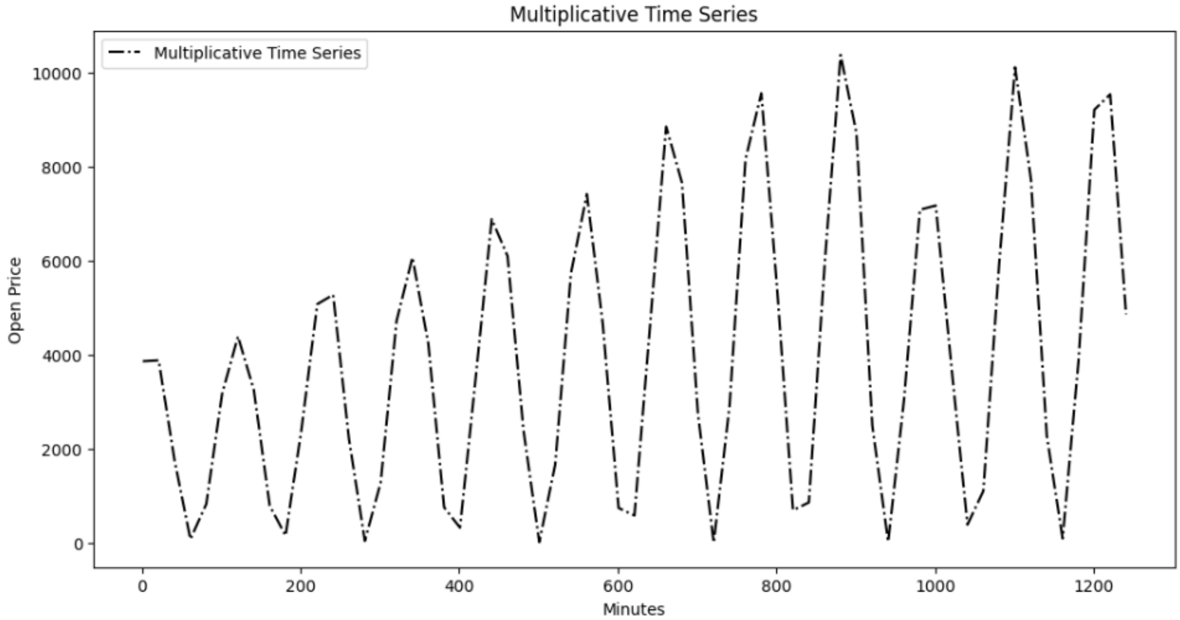
```
additive_Tmodel = Trend + seasonality + residual

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

plt.plot(T_Series, additive_Tmodel, 'k-', label="Additive Time Series")
plt.title("Additive Time Series")
plt.xlabel("Minutes")
plt.ylabel("Open Price")
plt.legend()
plt.show()

ignored_residual = np.ones_like(residual)
multiplicative_Tmodel = Trend * seasonality * ignored_residual

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

plt.plot(T_Series, multiplicative_Tmodel, 'k-.', label="Multiplicative Time Series")
plt.title("Multiplicative Time Series")
plt.xlabel("Minutes")
plt.ylabel("Open Price")
plt.legend()
plt.show()
```


Trend against Time
Trend
Open Price
Minutes
550
500
450
400
350
300
250
200
0
200
400
600
800
1000
1200


Seasonality against Time
Seasonality
Open Price
Minutes
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
200
400
600
800
1000
1200


Residuals against Time
Residual
Open Price
Minutes
2
1
0
−1
−2
0
200
400
600
800
1000
1200


Additive Time Series
Additive Time Series
Open Price
Minutes
550
500
450
400
350
300
250
200
0
200
400
600
800
1000
1200




```
df1 = pd.read_excel(r'/content/India_Exchange_Rate_Dataset.xls', parse_dates = True, index_col=0)
result = seasonal_decompose(df1['EXINUS'], model='multiplicative')
result.plot()
```