----------Time Series Forecasting----------

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from pandas import datetime
from statsmodels.distributions.empirical_distribution import ECDF
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

import warnings
warnings.filterwarnings("ignore")
```

```
<ipython-input-1-8aa0bf58b2d3>:6: FutureWarning: The pandas.datetime class i
s deprecated and will be removed from pandas in a future version. Import fro
m datetime module instead.
  from pandas import datetime
```

In [2]:

```
import os
os.getcwd()
```

Out[2]:

```
'C:\\Users\\rahul'
```

In [3]:

```
os.chdir('C:\\Users\\rahul')
```

Q 1. Read the data as an appropriate Time Series data and plot the data.

Method for Sparkling

In [4]:

```
df1 = pd.read_csv("Sparkling.csv",parse_dates=True,squeeze=True,index_col=0)
```

In [5]:

```
df1.head()
```

Out[5]:

```
YearMonth
1980-01-01    1686
1980-02-01    1591
1980-03-01    2304
1980-04-01    1712
1980-05-01    1471
Name: Sparkling, dtype: int64
```

In [6]:

```
df1.size
```

Out[6]:

```
187
```

In [7]:

```
df1.isnull().sum()
```

Out[7]:

```
0
```

In [8]:

```
df1.describe()
```

Out[8]:

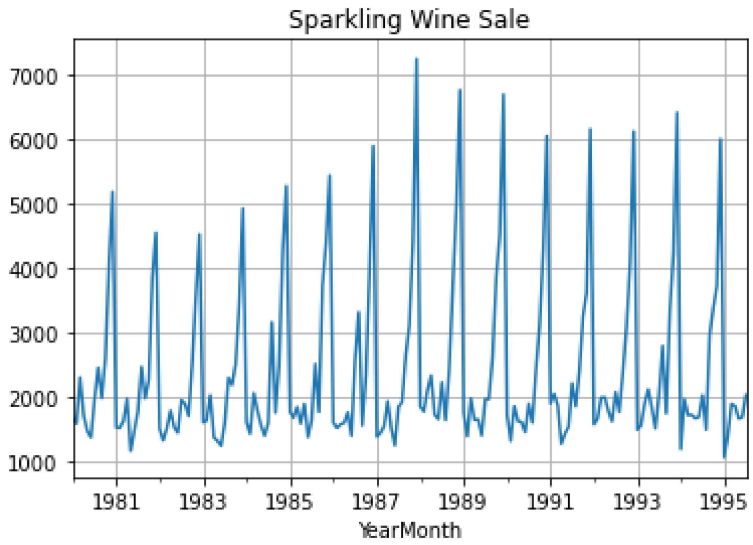
```
count     187.000000
mean     2402.417112
std      1295.111540
min      1070.000000
25%      1605.000000
50%      1874.000000
75%      2549.000000
max      7242.000000
Name: Sparkling, dtype: float64
```

In [9]:

```
df1.plot();
plt.grid()
plt.title('Sparkling Wine Sale')
```

Out[9]:

```
Text(0.5, 1.0, 'Sparkling Wine Sale')
```



In [10]:

```
df2 = pd.read_csv("Sparkling.csv")
```

In [11]:

```
df2.head()
```

Out[11]:

	YearMonth	Sparkling
0	1980-01	1686
1	1980-02	1591
2	1980-03	2304
3	1980-04	1712
4	1980-05	1471

In [12]:

```
date = pd.date_range(start='1/1/1980', end='8/1/1995', freq='M')
```

In [13]:

```
date
```

Out[13]:

```
DatetimeIndex(['1980-01-31', '1980-02-29', '1980-03-31', '1980-04-30',
               '1980-05-31', '1980-06-30', '1980-07-31', '1980-08-31',
               '1980-09-30', '1980-10-31',
               ...
               '1994-10-31', '1994-11-30', '1994-12-31', '1995-01-31',
               '1995-02-28', '1995-03-31', '1995-04-30', '1995-05-31',
               '1995-06-30', '1995-07-31'],
              dtype='datetime64[ns]', length=187, freq='M')
```

In [14]:

```
df2['Time_Stamp'] = pd.DataFrame(date,columns=['Month'])
df2.head()
```

Out[14]:

	YearMonth	Sparkling	Time_Stamp
0	1980-01	1686	1980-01-31
1	1980-02	1591	1980-02-29
2	1980-03	2304	1980-03-31
3	1980-04	1712	1980-04-30
4	1980-05	1471	1980-05-31

Method for Rose

In [15]:

```
df3 = pd.read_csv("Rose.csv",parse_dates=True,squeeze=True,index_col=0)
```

In [16]:

```
df3.head()
```

Out[16]:

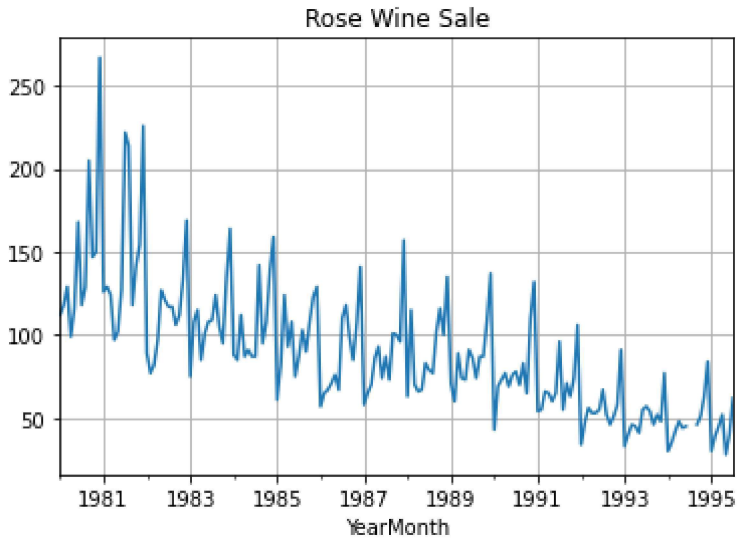
```
YearMonth
1980-01-01    112.0
1980-02-01    118.0
1980-03-01    129.0
1980-04-01     99.0
1980-05-01    116.0
Name: Rose, dtype: float64
```

In [17]:

```
df3.plot();
plt.grid()
plt.title('Rose Wine Sale')
```

Out[17]:

```
Text(0.5, 1.0, 'Rose Wine Sale')
```



In [18]:

```
df4 =pd.read_csv('Rose.csv')
```

In [19]:

```
df4.head()
```

Out[19]:

	YearMonth	Rose
0	1980-01	112.0
1	1980-02	118.0
2	1980-03	129.0
3	1980-04	99.0
4	1980-05	116.0

In [20]:

```
date = pd.date_range(start='1/1/1980', end='8/1/1995', freq='M')
```

In [21]:

```
date
```

Out[21]:

```
DatetimeIndex(['1980-01-31', '1980-02-29', '1980-03-31', '1980-04-30',
               '1980-05-31', '1980-06-30', '1980-07-31', '1980-08-31',
               '1980-09-30', '1980-10-31',
               ...
               '1994-10-31', '1994-11-30', '1994-12-31', '1995-01-31',
               '1995-02-28', '1995-03-31', '1995-04-30', '1995-05-31',
               '1995-06-30', '1995-07-31'],
              dtype='datetime64[ns]', length=187, freq='M')
```

In [22]:

```
df4['Time_Stamp'] = pd.DataFrame(date,columns=['Month'])
df4.head()
```

Out[22]:

	YearMonth	Rose	Time_Stamp
0	1980-01	112.0	1980-01-31
1	1980-02	118.0	1980-02-29
2	1980-03	129.0	1980-03-31
3	1980-04	99.0	1980-04-30
4	1980-05	116.0	1980-05-31

Q 2. Perform appropriate Exploratory Data Analysis to

understand the data and also perform decomposition.

In [23]:

```
df1.describe()
```

Out[23]:

```
count     187.000000
mean     2402.417112
std      1295.111540
min      1070.000000
25%      1605.000000
50%      1874.000000
75%      2549.000000
max      7242.000000
Name: Sparkling, dtype: float64
```

In [24]:

```
df3.describe()
```

Out[24]:

```
count    185.000000
mean      90.394595
std       39.175344
min       28.000000
25%       63.000000
50%       86.000000
75%      112.000000
max      267.000000
Name: Rose, dtype: float64
```

In [25]:

```
df1.isnull().sum()
```

Out[25]:

```
0
```

In [26]:

```
df3.isnull().sum()
```

Out[26]:

```
2
```

In [27]:

```
df3['1994']
```

Out[27]:

```
YearMonth
1994-01-01    30.0
1994-02-01    35.0
1994-03-01    42.0
1994-04-01    48.0
1994-05-01    44.0
1994-06-01    45.0
1994-07-01     NaN
1994-08-01     NaN
1994-09-01    46.0
1994-10-01    51.0
1994-11-01    63.0
1994-12-01    84.0
Name: Rose, dtype: float64
```

In [28]:

```
df3 = df3.interpolate()
df3['1994']
```

Out[28]:

```
YearMonth
1994-01-01    30.000000
1994-02-01    35.000000
1994-03-01    42.000000
1994-04-01    48.000000
1994-05-01    44.000000
1994-06-01    45.000000
1994-07-01    45.333333
1994-08-01    45.666667
1994-09-01    46.000000
1994-10-01    51.000000
1994-11-01    63.000000
1994-12-01    84.000000
Name: Rose, dtype: float64
```

In [29]:

```
df4= df4.interpolate()
df4.head()
```

Out[29]:

	YearMonth	Rose	Time_Stamp
0	1980-01	112.0	1980-01-31
1	1980-02	118.0	1980-02-29
2	1980-03	129.0	1980-03-31
3	1980-04	99.0	1980-04-30
4	1980-05	116.0	1980-05-31

In [30]:

```
df4.isna().sum()
```

Out[30]:

```
YearMonth     0
Rose          0
Time_Stamp    0
dtype: int64
```

Plotting boxplot Wine sales across different years /months across years

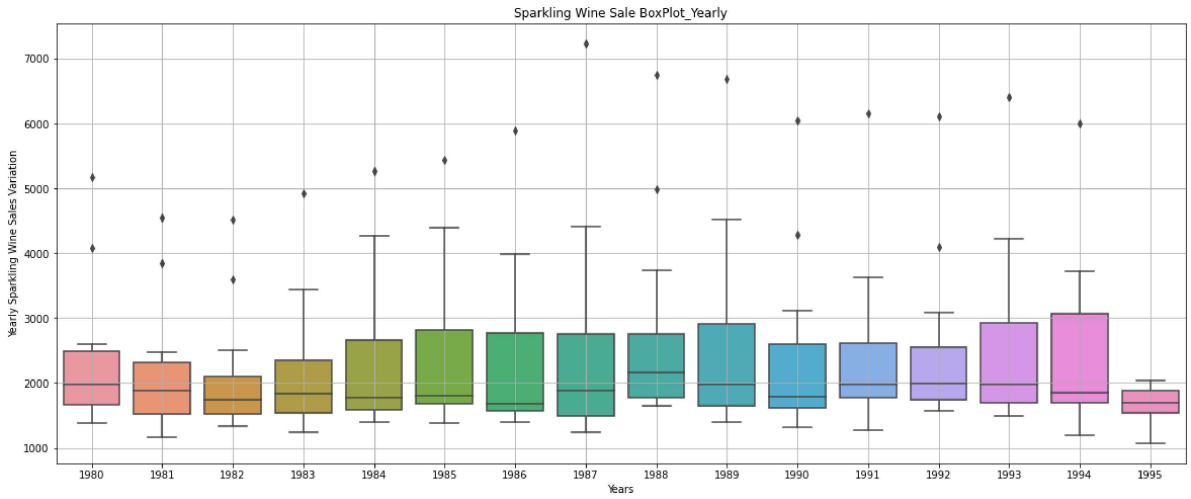
Yearly Plot for Sparkling data Wine Sale

In [31]:

```
fig, ax = plt.subplots(figsize=(20,8))
sns.boxplot(df1.index.year, df1, ax=ax,whis=1.5)
plt.grid()
plt.xlabel('Years');
plt.ylabel('Yearly Sparkling Wine Sales Variation');
plt.title('Sparkling Wine Sale BoxPlot_Yearly')
```

Out[31]:

```
Text(0.5, 1.0, 'Sparkling Wine Sale BoxPlot_Yearly')
```



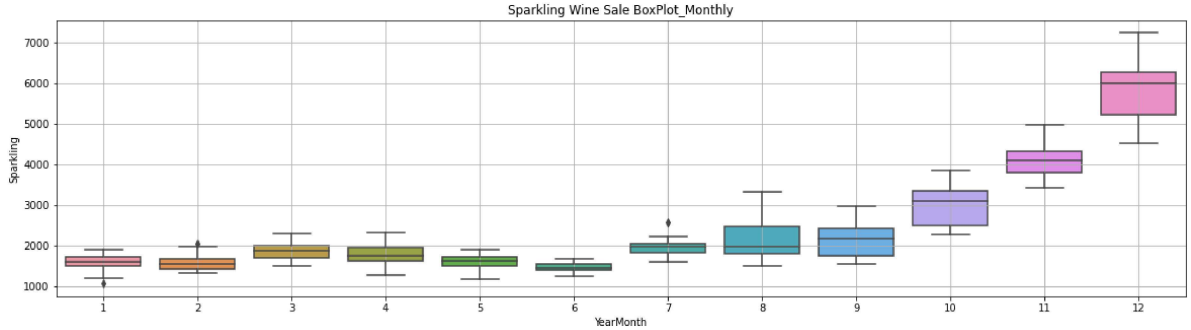
Monthly Plot for Sparkling wine sale

In [32]:

```
fig, ax = plt.subplots(figsize=(20,5))
sns.boxplot(df1.index.month, df1, ax=ax,whis=1.5)
plt.grid();
plt.title('Sparkling Wine Sale BoxPlot_Monthly')
```

Out[32]:

```
Text(0.5, 1.0, 'Sparkling Wine Sale BoxPlot_Monthly')
```

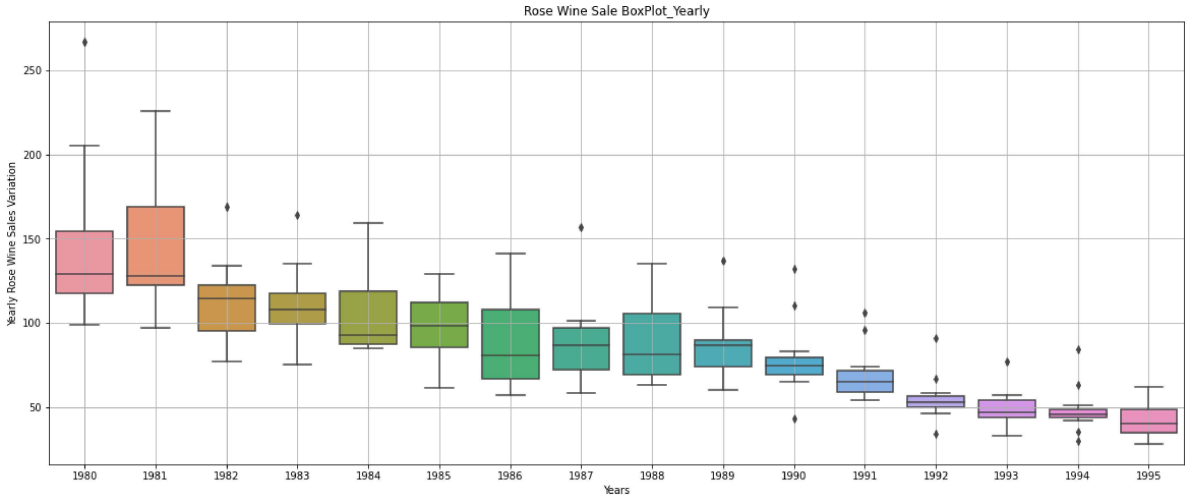


In [33]:

```
fig, ax = plt.subplots(figsize=(20,8))
sns.boxplot(df3.index.year, df3, ax=ax,whis=1.5)
plt.grid()
plt.xlabel('Years');
plt.ylabel('Yearly Rose Wine Sales Variation');
plt.title('Rose Wine Sale BoxPlot_Yearly')
```

Out[33]:

```
Text(0.5, 1.0, 'Rose Wine Sale BoxPlot_Yearly')
```



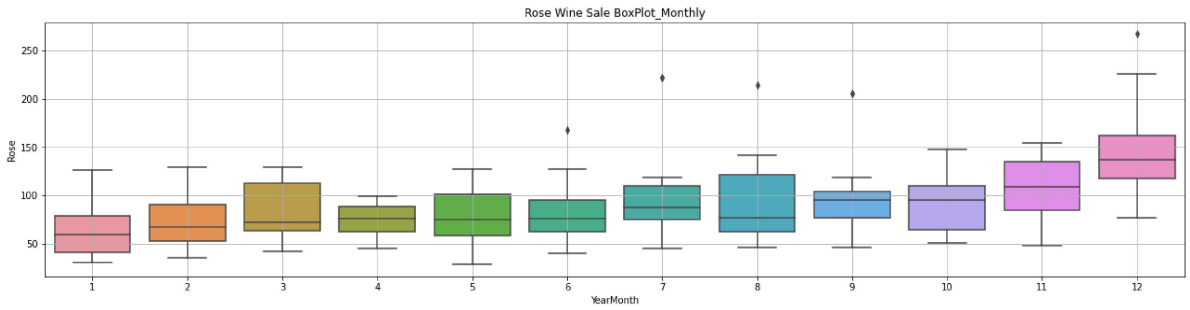
Monthly Plot for Rose wine sale

In [34]:

```
fig, ax = plt.subplots(figsize=(22,5))
sns.boxplot(df3.index.month, df3, ax=ax,whis=1.5)
plt.grid();
plt.title('Rose Wine Sale BoxPlot_Monthly')
```

Out[34]:

```
Text(0.5, 1.0, 'Rose Wine Sale BoxPlot_Monthly')
```



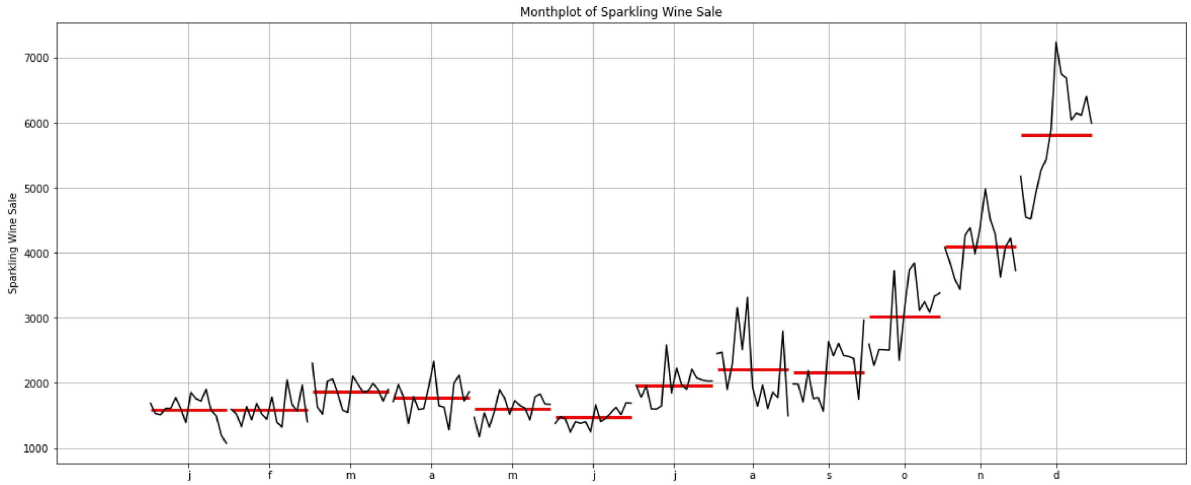
Plotting a monthplot of the given Time Series.

In [35]:

```
from statsmodels.graphics.tsaplots import month_plot
```

In [36]:

```
fig, ax = plt.subplots(figsize=(20,8))

month_plot(df1,ylabel='Sparkling Wine Sale',ax=ax)
plt.title('Monthplot of Sparkling Wine Sale')
plt.grid();
```



Plot the Time Series according to different months for different years.¶

Saprkling Wine Sale Data

In [37]:

```
df2.set_index(keys = 'Time_Stamp',drop=True,inplace=True)
df2.head()
```

Out[37]:

Time_Stamp	YearMonth	Sparkling
1980-01-31	1980-01	1686
1980-02-29	1980-02	1591
1980-03-31	1980-03	2304
1980-04-30	1980-04	1712
1980-05-31	1980-05	1471

In [38]:

```
df2.drop(labels='YearMonth',inplace=True,axis=1)
df2.head()
```

Out[38]:

Time_Stamp	Sparkling
1980-01-31	1686
1980-02-29	1591
1980-03-31	2304
1980-04-30	1712
1980-05-31	1471

In [39]:

```
df2['Sparkling']
```

Out[39]:

```
Time_Stamp
1980-01-31    1686
1980-02-29    1591
1980-03-31    2304
1980-04-30    1712
1980-05-31    1471
              ...
1995-03-31    1897
1995-04-30    1862
1995-05-31    1670
1995-06-30    1688
1995-07-31    2031
Name: Sparkling, Length: 187, dtype: int64
```

In [40]:

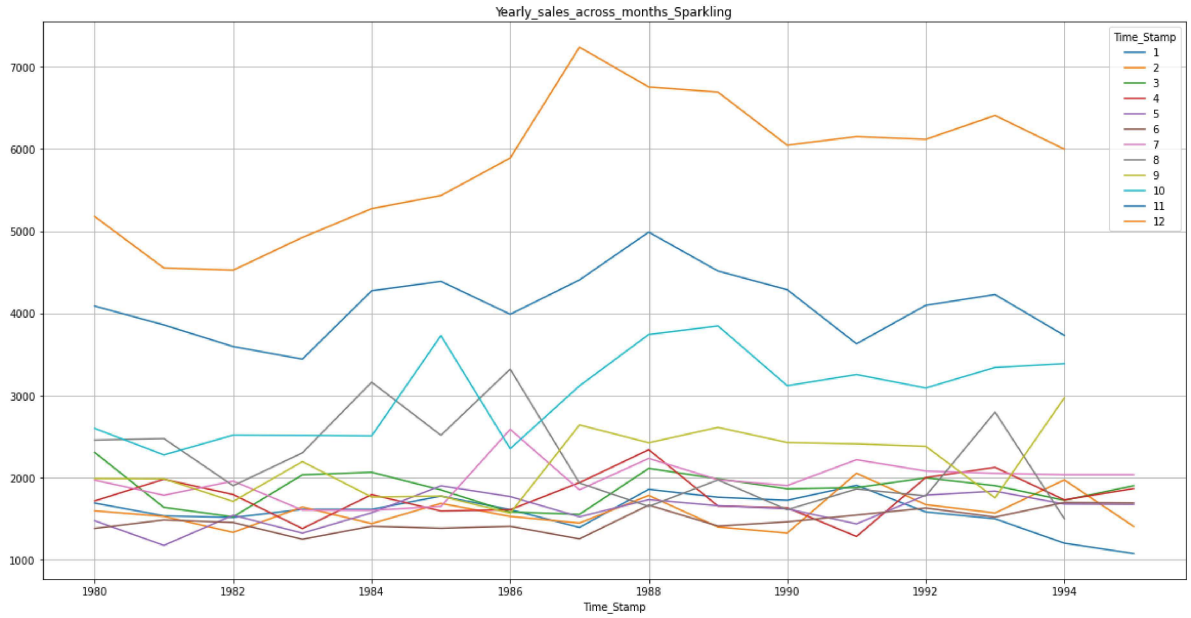
```
yearly_sales_across_months_Sparkling = pd.pivot_table(df2, values = 'Sparkling', columns =
                                        index = df2.index.year)

yearly_sales_across_months_Sparkling
```

Out[40]:

Time_Stamp	1	2	3	4	5	6	7	8	9	10	
Time_Stamp											
1980	1686.0	1591.0	2304.0	1712.0	1471.0	1377.0	1966.0	2453.0	1984.0	2596.0	408
1981	1530.0	1523.0	1633.0	1976.0	1170.0	1480.0	1781.0	2472.0	1981.0	2273.0	385
1982	1510.0	1329.0	1518.0	1790.0	1537.0	1449.0	1954.0	1897.0	1706.0	2514.0	359
1983	1609.0	1638.0	2030.0	1375.0	1320.0	1245.0	1600.0	2298.0	2191.0	2511.0	344
1984	1609.0	1435.0	2061.0	1789.0	1567.0	1404.0	1597.0	3159.0	1759.0	2504.0	427
1985	1771.0	1682.0	1846.0	1589.0	1896.0	1379.0	1645.0	2512.0	1771.0	3727.0	438
1986	1606.0	1523.0	1577.0	1605.0	1765.0	1403.0	2584.0	3318.0	1562.0	2349.0	398
1987	1389.0	1442.0	1548.0	1935.0	1518.0	1250.0	1847.0	1930.0	2638.0	3114.0	440
1988	1853.0	1779.0	2108.0	2336.0	1728.0	1661.0	2230.0	1645.0	2421.0	3740.0	498
1989	1757.0	1394.0	1982.0	1650.0	1654.0	1406.0	1971.0	1968.0	2608.0	3845.0	451
1990	1720.0	1321.0	1859.0	1628.0	1615.0	1457.0	1899.0	1605.0	2424.0	3116.0	428
1991	1902.0	2049.0	1874.0	1279.0	1432.0	1540.0	2214.0	1857.0	2408.0	3252.0	362
1992	1577.0	1667.0	1993.0	1997.0	1783.0	1625.0	2076.0	1773.0	2377.0	3088.0	409
1993	1494.0	1564.0	1898.0	2121.0	1831.0	1515.0	2048.0	2795.0	1749.0	3339.0	422
1994	1197.0	1968.0	1720.0	1725.0	1674.0	1693.0	2031.0	1495.0	2968.0	3385.0	372
1995	1070.0	1402.0	1897.0	1862.0	1670.0	1688.0	2031.0	NaN	NaN	NaN	N

In [41]:

```
fig, ax = plt.subplots(figsize=(20,10))
yearly_sales_across_months_Sparkling.plot(ax=ax)
plt.title('Yearly_sales_across_months_Sparkling')
plt.grid();
```



Rose Wine Sale Data

In [42]:

```
df4.set_index(keys = 'Time_Stamp',drop=True,inplace=True)
df4.head()
```

Out[42]:

Time_Stamp	YearMonth	Rose
1980-01-31	1980-01	112.0
1980-02-29	1980-02	118.0
1980-03-31	1980-03	129.0
1980-04-30	1980-04	99.0
1980-05-31	1980-05	116.0

In [43]:

```
df4.drop(labels='YearMonth',inplace=True,axis=1)
df4.head()
```

Out[43]:

Time_Stamp	Rose
1980-01-31	112.0
1980-02-29	118.0
1980-03-31	129.0
1980-04-30	99.0
1980-05-31	116.0

In [44]:

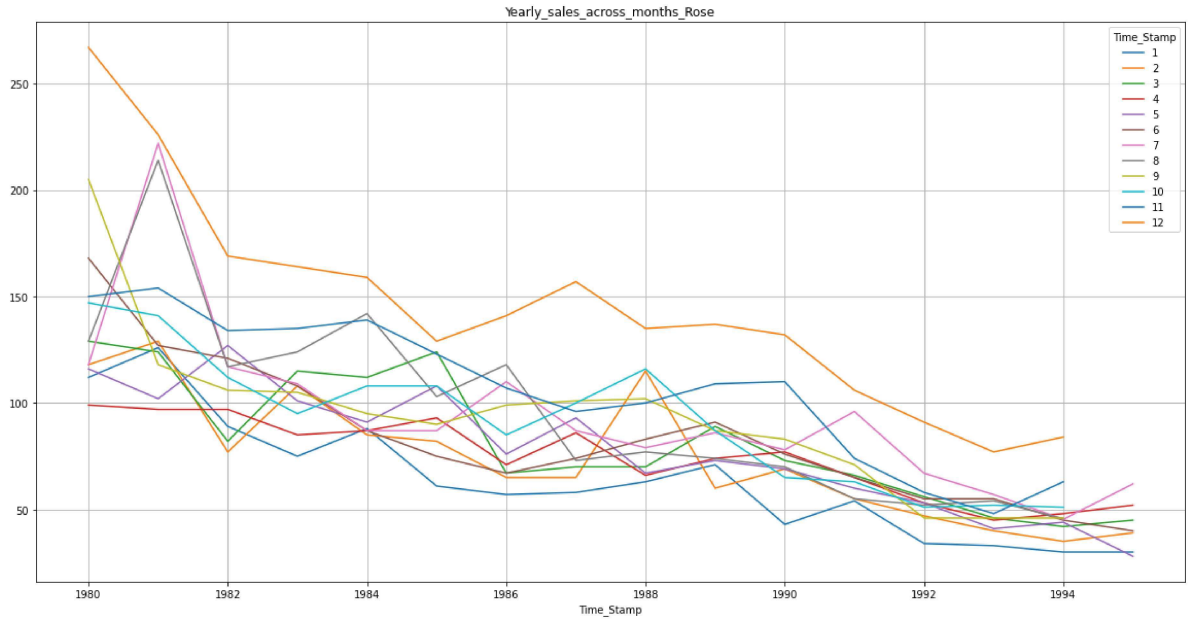
```
yearly_sales_across_months_Rose = pd.pivot_table(df4, values = 'Rose', columns = df4.index.
                                    index = df4.index.year)

yearly_sales_across_months_Rose
```

Out[44]:

Time_Stamp	1	2	3	4	5	6	7	8	9	10	
Time_Stamp											
1980	112.0	118.0	129.0	99.0	116.0	168.0	118.000000	129.000000	205.0	147.0	150
1981	126.0	129.0	124.0	97.0	102.0	127.0	222.000000	214.000000	118.0	141.0	154
1982	89.0	77.0	82.0	97.0	127.0	121.0	117.000000	117.000000	106.0	112.0	134
1983	75.0	108.0	115.0	85.0	101.0	108.0	109.000000	124.000000	105.0	95.0	135
1984	88.0	85.0	112.0	87.0	91.0	87.0	87.000000	142.000000	95.0	108.0	139
1985	61.0	82.0	124.0	93.0	108.0	75.0	87.000000	103.000000	90.0	108.0	123
1986	57.0	65.0	67.0	71.0	76.0	67.0	110.000000	118.000000	99.0	85.0	107
1987	58.0	65.0	70.0	86.0	93.0	74.0	87.000000	73.000000	101.0	100.0	96
1988	63.0	115.0	70.0	66.0	67.0	83.0	79.000000	77.000000	102.0	116.0	100
1989	71.0	60.0	89.0	74.0	73.0	91.0	86.000000	74.000000	87.0	87.0	109
1990	43.0	69.0	73.0	77.0	69.0	76.0	78.000000	70.000000	83.0	65.0	110
1991	54.0	55.0	66.0	65.0	60.0	65.0	96.000000	55.000000	71.0	63.0	74
1992	34.0	47.0	56.0	53.0	53.0	55.0	67.000000	52.000000	46.0	51.0	58
1993	33.0	40.0	46.0	45.0	41.0	55.0	57.000000	54.000000	46.0	52.0	48
1994	30.0	35.0	42.0	48.0	44.0	45.0	45.333333	45.666667	46.0	51.0	63
1995	30.0	39.0	45.0	52.0	28.0	40.0	62.000000	NaN	NaN	NaN	Na

In [45]:

```
fig, ax = plt.subplots(figsize=(20,10))
yearly_sales_across_months_Rose.plot(ax=ax)
plt.title('Yearly_sales_across_months_Rose')
plt.grid();
```



Decomposing the Time Series

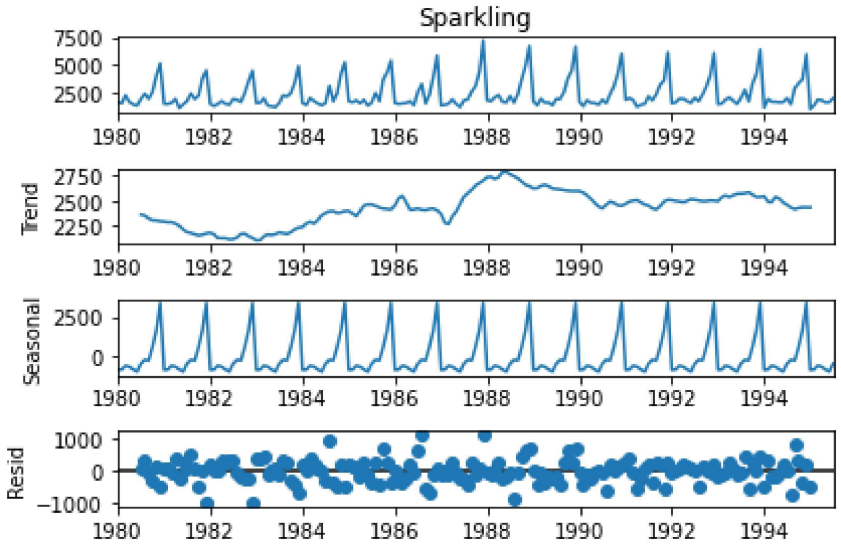
Additive Seasonality_Sparkling TSF

In [46]:

```
from statsmodels.tsa.seasonal import seasonal_decompose
```

In [47]:

```
decomposition = seasonal_decompose(df1,model='additive')
decomposition.plot();
```



In [48]:

```
trend = decomposition.trend
seasonality = decomposition.seasonal
residual = decomposition.resid
```

In [49]:

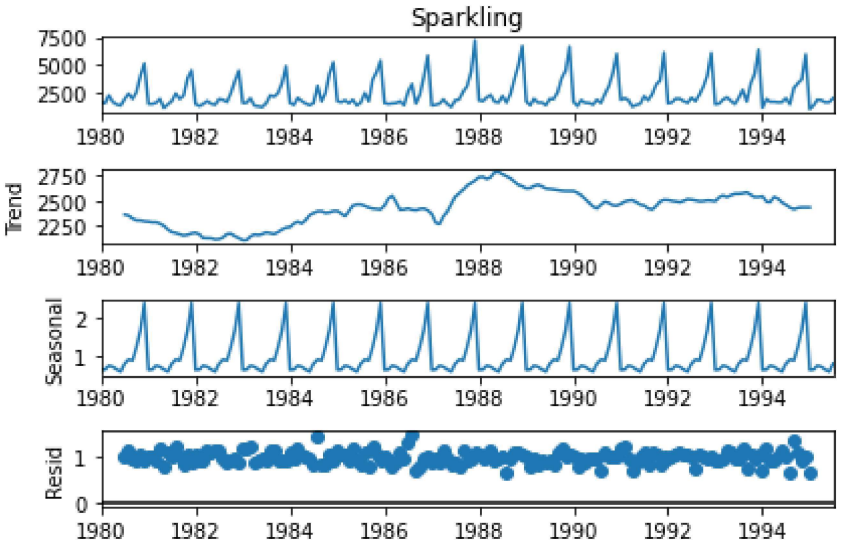
```
print('Trend','\n',trend.head(12),'\n')
print('Seasonality','\n',seasonality.head(12),'\n')
print('Residual','\n',residual.head(12),'\n')
```

```
Trend
 YearMonth
1980-01-01            NaN
1980-02-01            NaN
1980-03-01            NaN
1980-04-01            NaN
1980-05-01            NaN
1980-06-01            NaN
1980-07-01    2360.666667
1980-08-01    2351.333333
1980-09-01    2320.541667
1980-10-01    2303.583333
1980-11-01    2302.041667
1980-12-01    2293.791667
Name: trend, dtype: float64

Seasonality
 YearMonth
1980-01-01    -854.260599
1980-02-01    -830.350678
1980-03-01    -592.356630
1980-04-01    -658.490559
1980-05-01    -824.416154
1980-06-01    -967.434011
1980-07-01    -465.502265
1980-08-01    -214.332821
1980-09-01    -254.677265
1980-10-01     599.769957
1980-11-01    1675.067179
1980-12-01    3386.983846
Name: seasonal, dtype: float64

Residual
 YearMonth
1980-01-01           NaN
1980-02-01           NaN
1980-03-01           NaN
1980-04-01           NaN
1980-05-01           NaN
1980-06-01           NaN
1980-07-01     70.835599
1980-08-01    315.999487
1980-09-01    -81.864401
1980-10-01   -307.353290
1980-11-01    109.891154
1980-12-01   -501.775513
Name: resid, dtype: float64
```

Multiplicative Seasonality_Sparkling TSF

In [50]:

```
decomposition = seasonal_decompose(df1,model='multiplicative')
decomposition.plot();
```



In [51]:

```
trend = decomposition.trend
seasonality = decomposition.seasonal
residual = decomposition.resid

print('Trend','\n',trend.head(12),'\n')
print('Seasonality','\n',seasonality.head(12),'\n')
print('Residual','\n',residual.head(12),'\n')
```

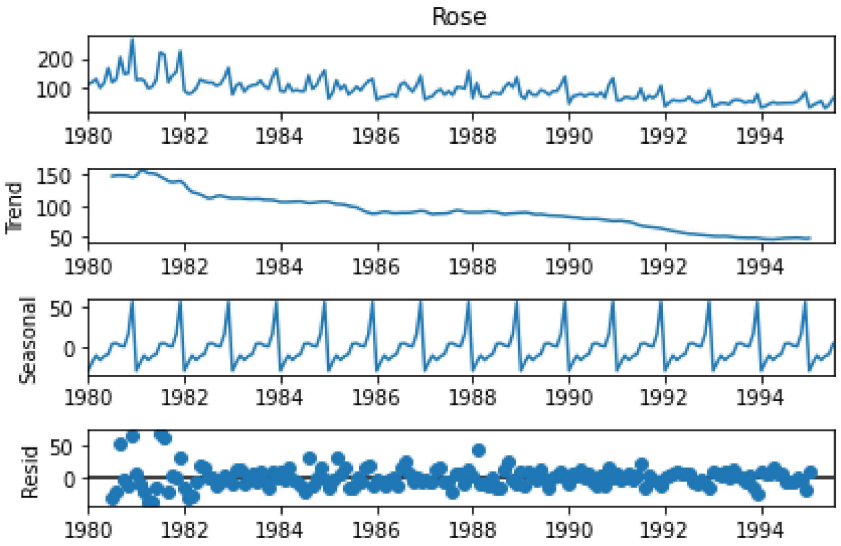
```
Trend
 YearMonth
1980-01-01            NaN
1980-02-01            NaN
1980-03-01            NaN
1980-04-01            NaN
1980-05-01            NaN
1980-06-01            NaN
1980-07-01    2360.666667
1980-08-01    2351.333333
1980-09-01    2320.541667
1980-10-01    2303.583333
1980-11-01    2302.041667
1980-12-01    2293.791667
Name: trend, dtype: float64

Seasonality
 YearMonth
1980-01-01    0.649843
1980-02-01    0.659214
1980-03-01    0.757440
1980-04-01    0.730351
1980-05-01    0.660609
1980-06-01    0.603468
1980-07-01    0.809164
1980-08-01    0.918822
1980-09-01    0.894367
1980-10-01    1.241789
1980-11-01    1.690158
1980-12-01    2.384776
Name: seasonal, dtype: float64

Residual
 YearMonth
1980-01-01         NaN
1980-02-01         NaN
1980-03-01         NaN
1980-04-01         NaN
1980-05-01         NaN
1980-06-01         NaN
1980-07-01    1.029230
1980-08-01    1.135407
1980-09-01    0.955954
1980-10-01    0.907513
1980-11-01    1.050423
1980-12-01    0.946770
Name: resid, dtype: float64
```

Decomposing Rose wine Time series data

Additive Seasonality_Rose TSF

In [52]:

```
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(df3, model='additive')
decomposition.plot();
```



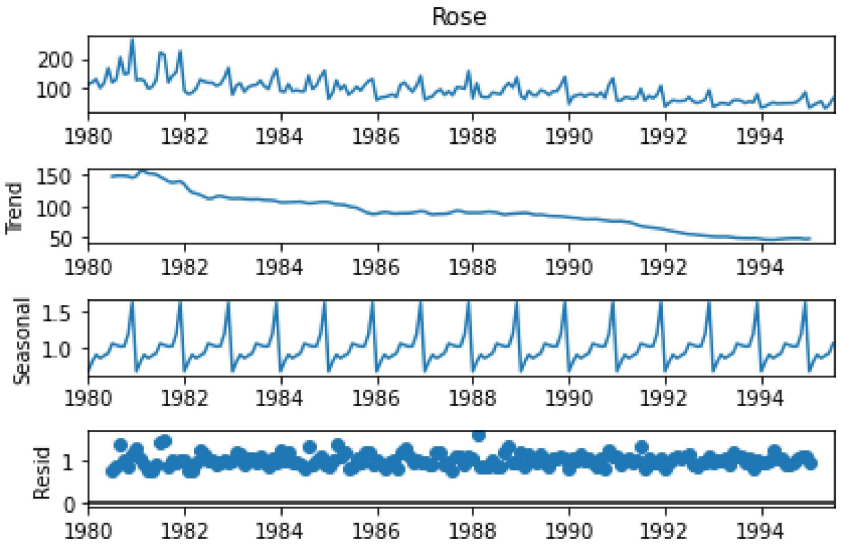
In [53]:

```
trend = decomposition.trend
seasonality = decomposition.seasonal
residual = decomposition.resid

print('Trend','\n',trend.head(12),'\n')
print('Seasonality','\n',seasonality.head(12),'\n')
print('Residual','\n',residual.head(12),'\n')
```

```
Trend
 YearMonth
1980-01-01           NaN
1980-02-01           NaN
1980-03-01           NaN
1980-04-01           NaN
1980-05-01           NaN
1980-06-01           NaN
1980-07-01    147.083333
1980-08-01    148.125000
1980-09-01    148.375000
1980-10-01    148.083333
1980-11-01    147.416667
1980-12-01    145.125000
Name: trend, dtype: float64

Seasonality
 YearMonth
1980-01-01   -27.908647
1980-02-01   -17.435632
1980-03-01    -9.285830
1980-04-01   -15.098330
1980-05-01   -10.196544
1980-06-01    -7.678687
1980-07-01     4.896908
1980-08-01     5.499686
1980-09-01     2.774686
1980-10-01     1.871908
1980-11-01    16.846908
1980-12-01    55.713575
Name: seasonal, dtype: float64

Residual
 YearMonth
1980-01-01          NaN
1980-02-01          NaN
1980-03-01          NaN
1980-04-01          NaN
1980-05-01          NaN
1980-06-01          NaN
1980-07-01   -33.980241
1980-08-01   -24.624686
1980-09-01    53.850314
1980-10-01    -2.955241
1980-11-01   -14.263575
1980-12-01    66.161425
Name: resid, dtype: float64
```

In [54]:

```
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(df3, model='multiplicative')
decomposition.plot();
```



In [55]:

```
trend = decomposition.trend
seasonality = decomposition.seasonal
residual = decomposition.resid

print('Trend','\n',trend.head(12),'\n')
print('Seasonality','\n',seasonality.head(12),'\n')
print('Residual','\n',residual.head(12),'\n')
```

```
Trend
 YearMonth
1980-01-01           NaN
1980-02-01           NaN
1980-03-01           NaN
1980-04-01           NaN
1980-05-01           NaN
1980-06-01           NaN
1980-07-01    147.083333
1980-08-01    148.125000
1980-09-01    148.375000
1980-10-01    148.083333
1980-11-01    147.416667
1980-12-01    145.125000
Name: trend, dtype: float64

Seasonality
 YearMonth
1980-01-01    0.670111
1980-02-01    0.806163
1980-03-01    0.901164
1980-04-01    0.854024
1980-05-01    0.889415
1980-06-01    0.923985
1980-07-01    1.058038
1980-08-01    1.035881
1980-09-01    1.017648
1980-10-01    1.022573
1980-11-01    1.192349
1980-12-01    1.628646
Name: seasonal, dtype: float64

Residual
 YearMonth
1980-01-01         NaN
1980-02-01         NaN
1980-03-01         NaN
1980-04-01         NaN
1980-05-01         NaN
1980-06-01         NaN
1980-07-01    0.758258
1980-08-01    0.840720
1980-09-01    1.357674
1980-10-01    0.970771
1980-11-01    0.853378
1980-12-01    1.129646
Name: resid, dtype: float64
```

Q 3. Split the data into training and test. The test data should start in 1991.

In [56]:

```
train1=df2[df2.index.year < 1991]
test1=df2[df2.index.year >= 1991]
```

In [57]:

```
print(train1.shape)
print(test1.shape)
```

```
(132, 1)
(55, 1)
```

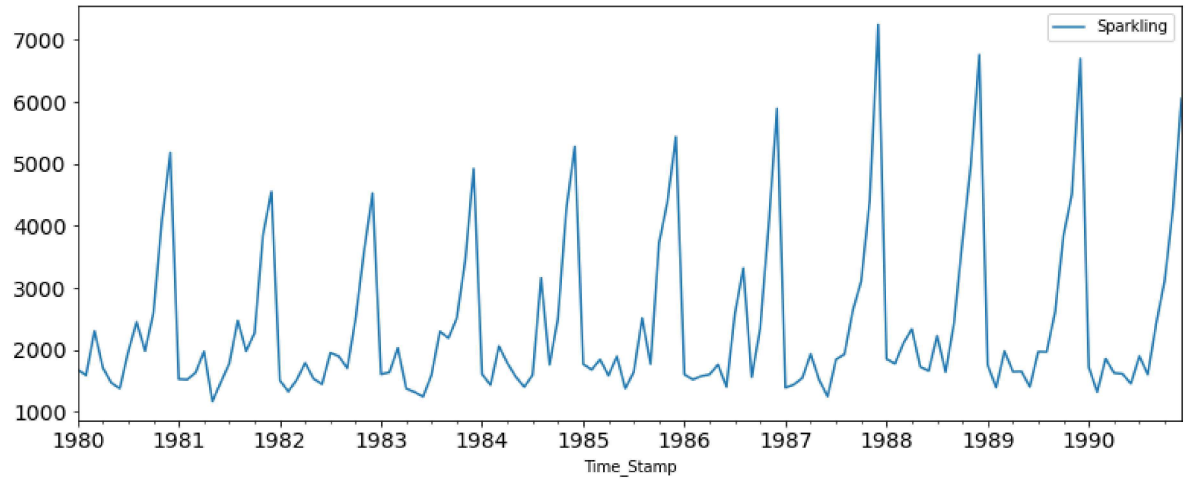
In [58]:

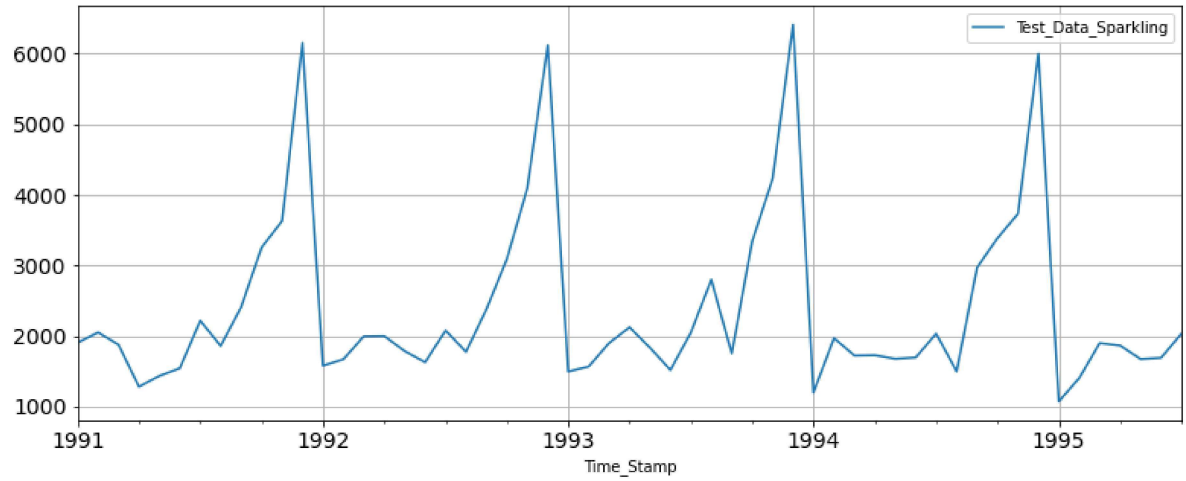
```
print('First few rows of Training Data','\n',train1.head(),'\n')
print('Last few rows of Training Data','\n',train1.tail(),'\n')
print('First few rows of Test Data','\n',test1.head(),'\n')
print('Last few rows of Test Data','\n',test1.tail(),'\n')
```

```
First few rows of Training Data 
             Sparkling
Time_Stamp            
1980-01-31       1686
1980-02-29       1591
1980-03-31       2304
1980-04-30       1712
1980-05-31       1471 

Last few rows of Training Data 
             Sparkling
Time_Stamp            
1990-08-31       1605
1990-09-30       2424
1990-10-31       3116
1990-11-30       4286
1990-12-31       6047 

First few rows of Test Data 
             Sparkling
Time_Stamp            
1991-01-31       1902
1991-02-28       2049
1991-03-31       1874
1991-04-30       1279
1991-05-31       1432 

Last few rows of Test Data 
             Sparkling
Time_Stamp            
1995-03-31       1897
1995-04-30       1862
1995-05-31       1670
1995-06-30       1688
1995-07-31       2031
```

In [59]:

```
train1.plot(figsize=(13,5), fontsize=14)
test1.plot(figsize=(13,5), fontsize=14)
plt.grid()
plt.legend(['Test_Data_Sparkling','Train_Data_Sparkling'])
plt.show()
```





Splitting into train and test for Rose wine Sale

In [60]:

```
train2=df4[df4.index.year < 1991]
test2=df4[df4.index.year >= 1991]
```

In [61]:

```
print(train2.shape)
print(test2.shape)
```

```
(132, 1)
(55, 1)
```

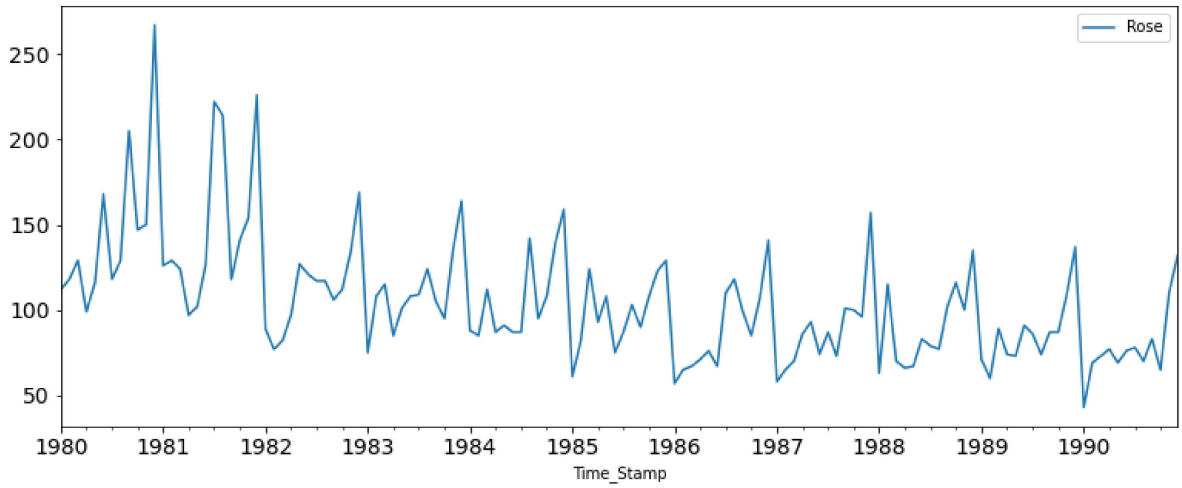
In [62]:

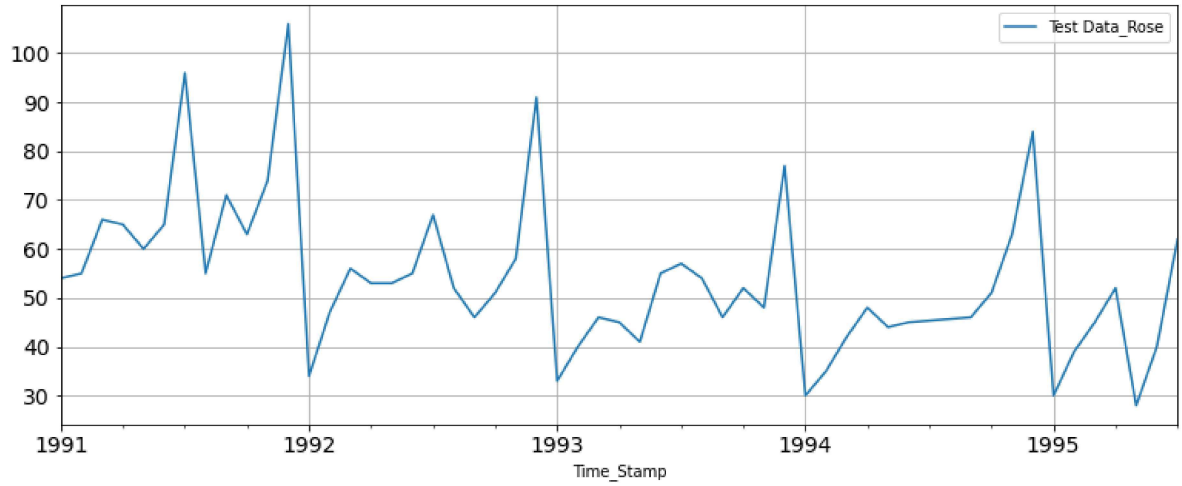
```
print('First few rows of Training Data','\n',train2.head(),'\n')
print('Last few rows of Training Data','\n',train2.tail(),'\n')
print('First few rows of Test Data','\n',test2.head(),'\n')
print('Last few rows of Test Data','\n',test2.tail(),'\n')
```

```
First few rows of Training Data 
              Rose
Time_Stamp        
1980-01-31  112.0
1980-02-29  118.0
1980-03-31  129.0
1980-04-30   99.0
1980-05-31  116.0 

Last few rows of Training Data 
              Rose
Time_Stamp        
1990-08-31   70.0
1990-09-30   83.0
1990-10-31   65.0
1990-11-30  110.0
1990-12-31  132.0 

First few rows of Test Data 
             Rose
Time_Stamp       
1991-01-31  54.0
1991-02-28  55.0
1991-03-31  66.0
1991-04-30  65.0
1991-05-31  60.0 

Last few rows of Test Data 
             Rose
Time_Stamp       
1995-03-31  45.0
1995-04-30  52.0
1995-05-31  28.0
1995-06-30  40.0
1995-07-31  62.0 
```

In [63]:

```
train2.plot(figsize=(13,5), fontsize=14)
test2.plot(figsize=(13,5), fontsize=14)
plt.grid()
plt.legend(['Test Data_Rose','Train_Data_Rose'])
plt.show()
```





Q 4 Build various exponential smoothing models on the training data and evaluate the model using RMSE on the test data. Other models such as regression,naïve forecast models and simple average models. should also be built on the training data and check the performance on the test data using RMSE.

Model 1: Linear Regression - Sparkling wine Model

In [64]:

```
train_time = [i+1 for i in range(len(train1))]
test_time = [i+43 for i in range(len(test1))]
print('Training Time instance','\n',train_time)
print('Test Time instance','\n',test_time)
```

```
Training Time instance
 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78,
79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 11
3, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 12
8, 129, 130, 131, 132]
Test Time instance
 [43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 6
1, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 8
0, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97]
```

In [65]:

```
LinearRegression_train = train1.copy()
LinearRegression_test = test1.copy()
```

In [66]:

```
LinearRegression_train['time'] = train_time
LinearRegression_test['time'] = test_time

print('First few rows of Training Data','\n',LinearRegression_train.head(),'\n')
print('Last few rows of Training Data','\n',LinearRegression_train.tail(),'\n')
print('First few rows of Test Data','\n',LinearRegression_test.head(),'\n')
print('Last few rows of Test Data','\n',LinearRegression_test.tail(),'\n')
```

```
First few rows of Training Data 
             Sparkling  time
Time_Stamp                  
1980-01-31       1686     1
1980-02-29       1591     2
1980-03-31       2304     3
1980-04-30       1712     4
1980-05-31       1471     5 

Last few rows of Training Data 
             Sparkling  time
Time_Stamp                  
1990-08-31       1605   128
1990-09-30       2424   129
1990-10-31       3116   130
1990-11-30       4286   131
1990-12-31       6047   132 

First few rows of Test Data 
             Sparkling  time
Time_Stamp                  
1991-01-31       1902    43
1991-02-28       2049    44
1991-03-31       1874    45
1991-04-30       1279    46
1991-05-31       1432    47 

Last few rows of Test Data 
             Sparkling  time
Time_Stamp                  
1995-03-31       1897    93
1995-04-30       1862    94
1995-05-31       1670    95
1995-06-30       1688    96
1995-07-31       2031    97 
```

In [67]:

```
from sklearn.linear_model import LinearRegression
```

In [68]:

```
lr = LinearRegression()
```

In [69]:

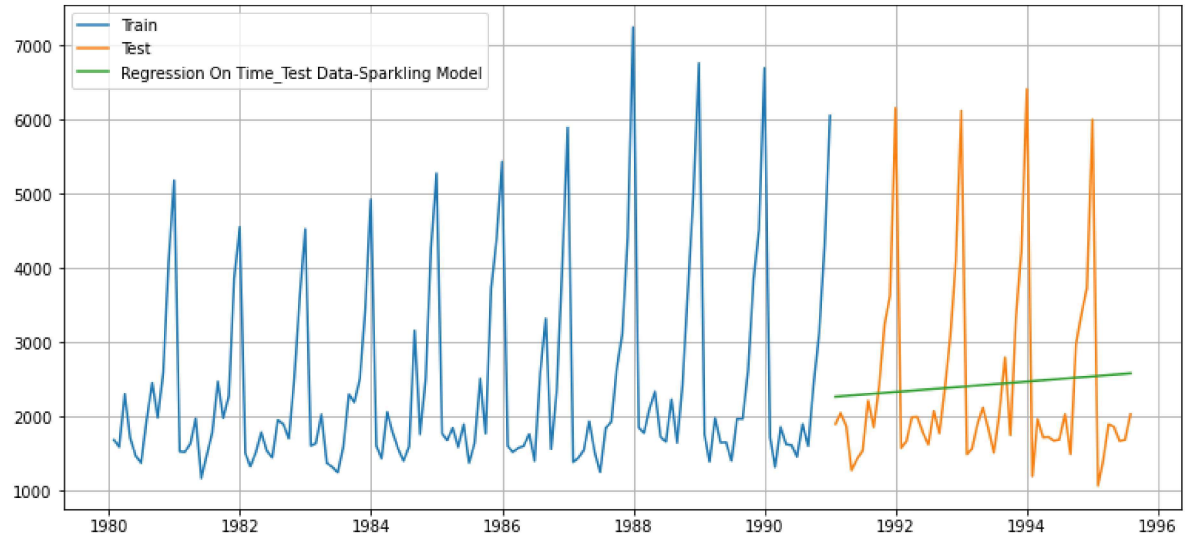
```
lr.fit(LinearRegression_train[['time']],LinearRegression_train['Sparkling'].values)
```

Out[69]:

```
LinearRegression()
```

In [70]:

```
test_predictions_model1 = lr.predict(LinearRegression_test[['time']])
LinearRegression_test['RegOnTime'] = test_predictions_model1

plt.figure(figsize=(13,6))
plt.plot( train1['Sparkling'], label='Train')
plt.plot(test1['Sparkling'], label='Test')
plt.plot(LinearRegression_test['RegOnTime'], label='Regression On Time_Test Data-Sparkling
plt.legend(loc='best')
plt.grid();
```



Defining the accuracy metrics.

In [71]:

```
from sklearn import metrics
```

Model Evaluation

In [72]:

```
## Test Data - RMSE

rmse_model1_test = metrics.mean_squared_error(test1['Sparkling'],test_predictions_model1,sq
print("For RegressionOnTime forecast on the Test Data,  RMSE is %3.3f" %(rmse_model1_test))
```

```
For RegressionOnTime forecast on the Test Data,  RMSE is 1275.867
```

In [73]:

```
resultsDf = pd.DataFrame({'Test RMSE': [rmse_model1_test]},index=['RegressionOnTime'])
resultsDf
```

Out[73]:

	Test RMSE
RegressionOnTime	1275.867052

Model 1: Linear Regression - Rose wine Model

In [74]:

```
train_time = [i+1 for i in range(len(train2))]
test_time = [i+43 for i in range(len(test2))]
print('Training Time instance','\n',train_time)
print('Test Time instance','\n',test_time)
```

```
Training Time instance
 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78,
79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 11
3, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 12
8, 129, 130, 131, 132]
Test Time instance
 [43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 6
1, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 8
0, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97]
```

In [75]:

```
LinearRegression_train2 = train2.copy()
LinearRegression_test2 = test2.copy()
```

In [76]:

```
LinearRegression_train2['time'] = train_time
LinearRegression_test2['time'] = test_time

print('First few rows of Training Data','\n',LinearRegression_train2.head(),'\n')
print('Last few rows of Training Data','\n',LinearRegression_train2.tail(),'\n')
print('First few rows of Test Data','\n',LinearRegression_test2.head(),'\n')
print('Last few rows of Test Data','\n',LinearRegression_test2.tail(),'\n')
```

```
First few rows of Training Data
              Rose  time
Time_Stamp
1980-01-31  112.0     1
1980-02-29  118.0     2
1980-03-31  129.0     3
1980-04-30   99.0     4
1980-05-31  116.0     5

Last few rows of Training Data
              Rose  time
Time_Stamp
1990-08-31   70.0   128
1990-09-30   83.0   129
1990-10-31   65.0   130
1990-11-30  110.0   131
1990-12-31  132.0   132

First few rows of Test Data
             Rose  time
Time_Stamp
1991-01-31  54.0    43
1991-02-28  55.0    44
1991-03-31  66.0    45
1991-04-30  65.0    46
1991-05-31  60.0    47

Last few rows of Test Data
             Rose  time
Time_Stamp
1995-03-31  45.0    93
1995-04-30  52.0    94
1995-05-31  28.0    95
1995-06-30  40.0    96
1995-07-31  62.0    97
```

In [77]:

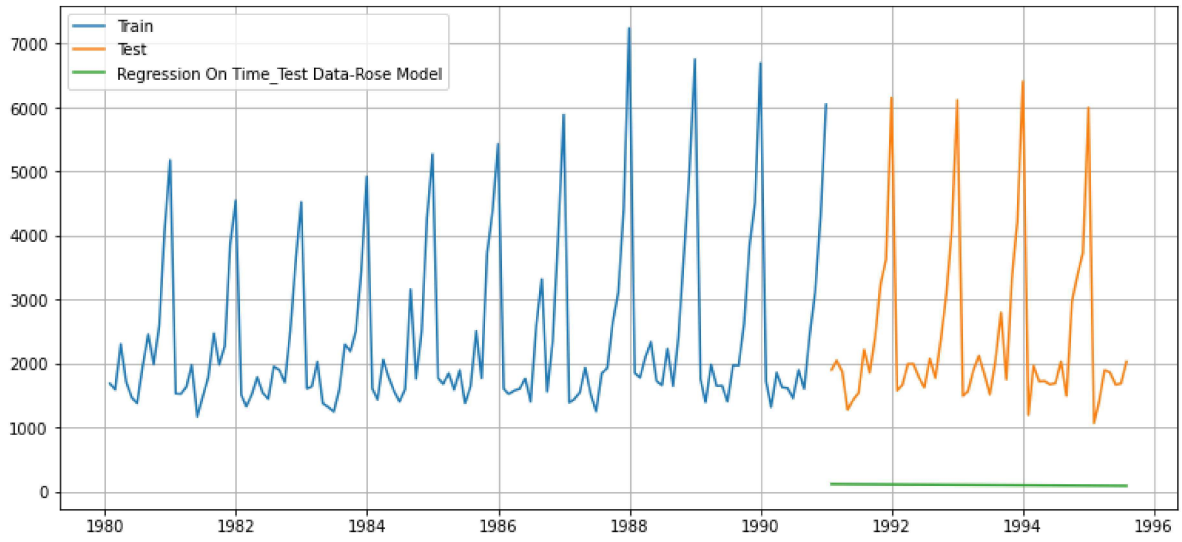
```
lr = LinearRegression()
lr.fit(LinearRegression_train2[['time']],LinearRegression_train2['Rose'].values)
```

Out[77]:

```
LinearRegression()
```

In [78]:

```
test_predictions_model2 = lr.predict(LinearRegression_test2[['time']])
LinearRegression_test2['RegOnTime'] = test_predictions_model2

plt.figure(figsize=(13,6))
plt.plot( train1['Sparkling'], label='Train')
plt.plot(test1['Sparkling'], label='Test')
plt.plot(LinearRegression_test2['RegOnTime'], label='Regression On Time_Test Data-Rose Mode
plt.legend(loc='best')
plt.grid();
```



In [79]:

```
rmse_model1_test1 = metrics.mean_squared_error(test2['Rose'],test_predictions_model1,square
print("For RegressionOnTime forecast on the Test Data,  RMSE is %3.3f" %(rmse_model1_test1)
```

```
For RegressionOnTime forecast on the Test Data,  RMSE is 2372.450
```

In [80]:

```
resultsDf_rose = pd.DataFrame({'Test RMSE': [rmse_model1_test1]},index=['RegressionOnTime']
resultsDf_rose
```

Out[80]:

	Test RMSE
RegressionOnTime	2372.449884

Model 2: Naive Approach: $\hat{y}\ t$+1=yt

Sparkling Wine Model

In [81]:

```
NaiveModel_train = train1.copy()
NaiveModel_test = test1.copy()
```

In [82]:

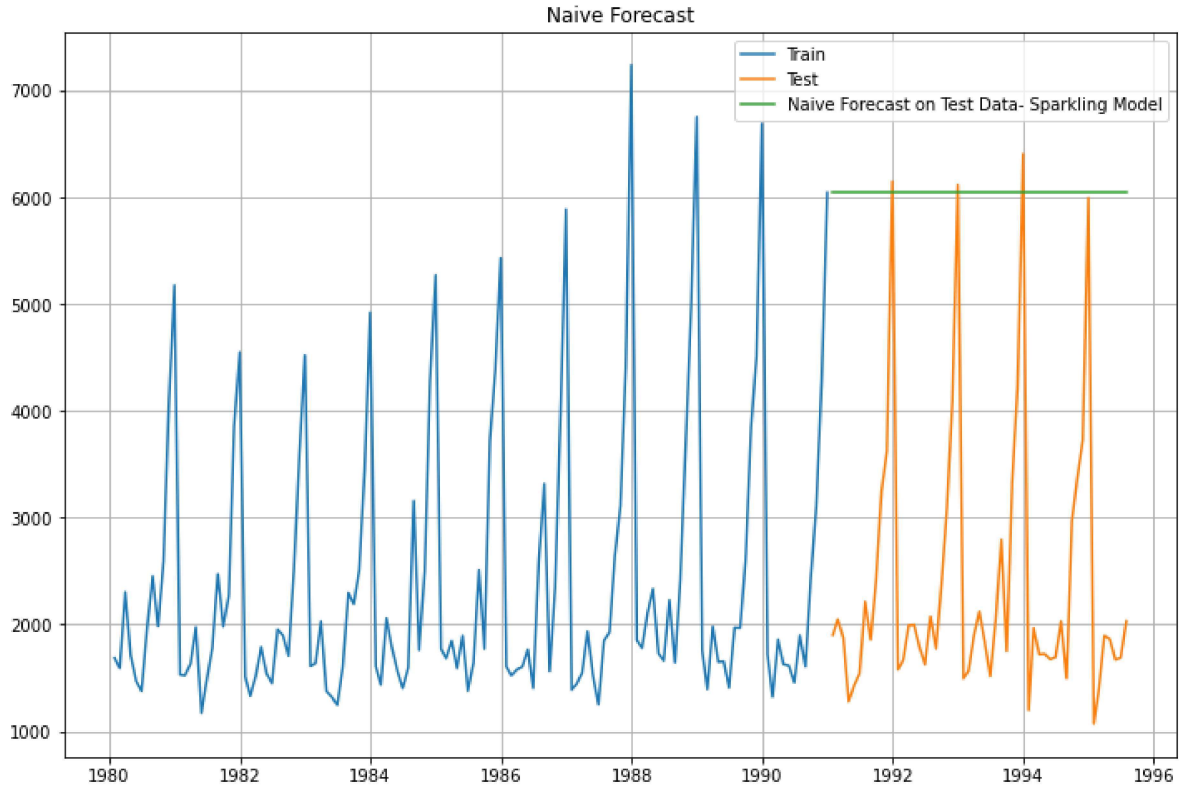
```
NaiveModel_test['naive'] = np.asarray(train1['Sparkling'])[len(np.asarray(train1['Sparkling
NaiveModel_test['naive'].head()
```

Out[82]:

```
Time_Stamp
1991-01-31    6047
1991-02-28    6047
1991-03-31    6047
1991-04-30    6047
1991-05-31    6047
Name: naive, dtype: int64
```

In [83]:

```
plt.figure(figsize=(12,8))
plt.plot(NaiveModel_train['Sparkling'], label='Train')
plt.plot(test1['Sparkling'], label='Test')
plt.plot(NaiveModel_test['naive'], label='Naive Forecast on Test Data- Sparkling Model')
plt.legend(loc='best')
plt.title("Naive Forecast")
plt.grid();
```



Model Evaluation-Naive Approach - Sparkling Wine Model

In [84]:

```
rmse_model2_test = metrics.mean_squared_error(test1['Sparkling'],NaiveModel_test['naive'],s
print("For RegressionOnTime forecast on the Test Data,  RMSE is %3.3f" %(rmse_model2_test))
```

```
For RegressionOnTime forecast on the Test Data,  RMSE is 3864.279
```

In [85]:

```
resultsDf_2 = pd.DataFrame({'Test RMSE': [rmse_model2_test]},index=['NaiveModel'])

resultsDf = pd.concat([resultsDf, resultsDf_2])
resultsDf
```

Out[85]:

	Test RMSE
RegressionOnTime	1275.867052
NaiveModel	3864.279352

Rose Wine Model

In [86]:

```
NaiveModel_train2 = train2.copy()
NaiveModel_test2 = test2.copy()
```

In [87]:

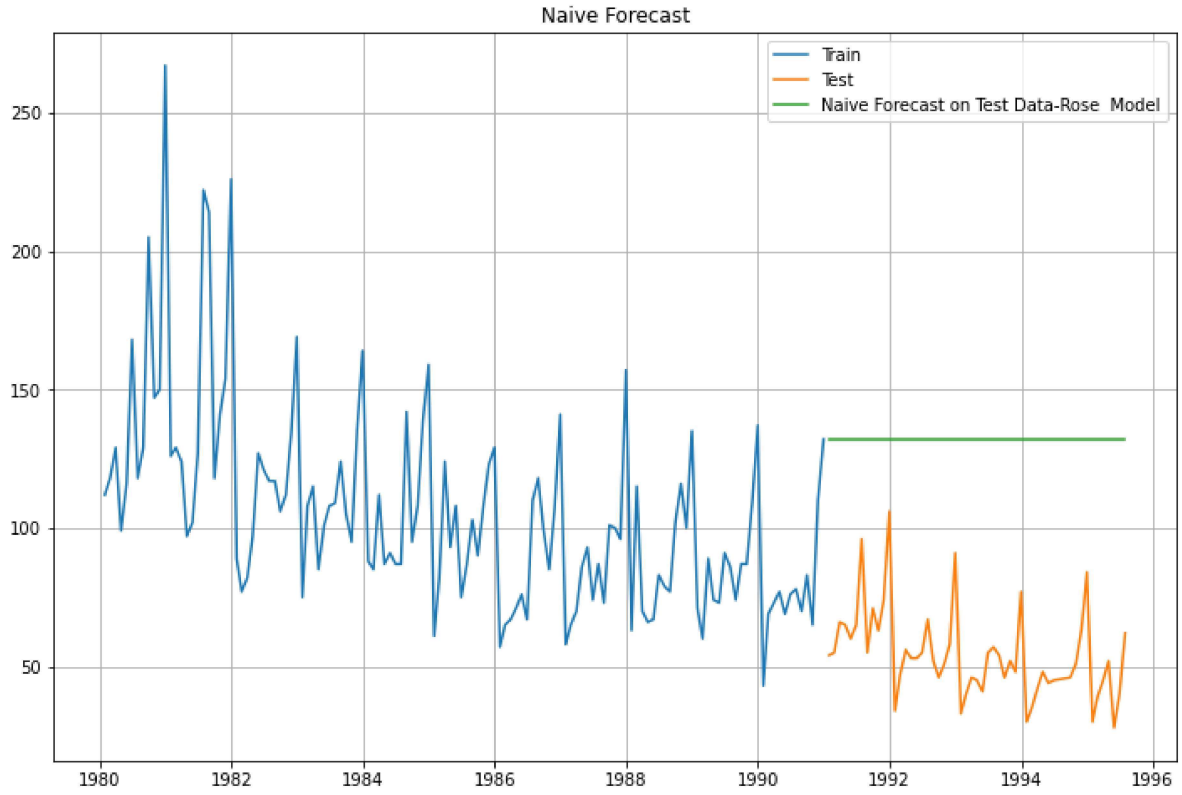
```
NaiveModel_test2['naive'] = np.asarray(train2['Rose'])[len(np.asarray(train2['Rose']))-1]
NaiveModel_test2['naive'].head()
```

Out[87]:

```
Time_Stamp
1991-01-31    132.0
1991-02-28    132.0
1991-03-31    132.0
1991-04-30    132.0
1991-05-31    132.0
Name: naive, dtype: float64
```

In [88]:

```
plt.figure(figsize=(12,8))
plt.plot(NaiveModel_train2['Rose'], label='Train')
plt.plot(test2['Rose'], label='Test')
plt.plot(NaiveModel_test2['naive'], label='Naive Forecast on Test Data-Rose  Model')
plt.legend(loc='best')
plt.title("Naive Forecast")
plt.grid();
```



Model Evaluation-Naive Approach - Rose Wine Model

In [89]:

```
rmse_model2_test1 = metrics.mean_squared_error(test2['Rose'],NaiveModel_test2['naive'],squa
print("For RegressionOnTime forecast on the Test Data,  RMSE is %3.3f" %(rmse_model2_test1)
```

```
For RegressionOnTime forecast on the Test Data,  RMSE is 79.719
```

In [90]:

```
resultsDf_2_rose = pd.DataFrame({'Test RMSE': [rmse_model2_test1]},index=['NaiveModel'])

resultsDf_rose = pd.concat([resultsDf_rose, resultsDf_2_rose])
resultsDf_rose
```

Out[90]:

	Test RMSE
RegressionOnTime	2372.449884
NaiveModel	79.718773

Model- 3: Simple Average Model

Simple Average-Sparkling Wine Model

In [91]:

```
SimpleAverage_train = train1.copy()
SimpleAverage_test = test1.copy()
```

In [92]:

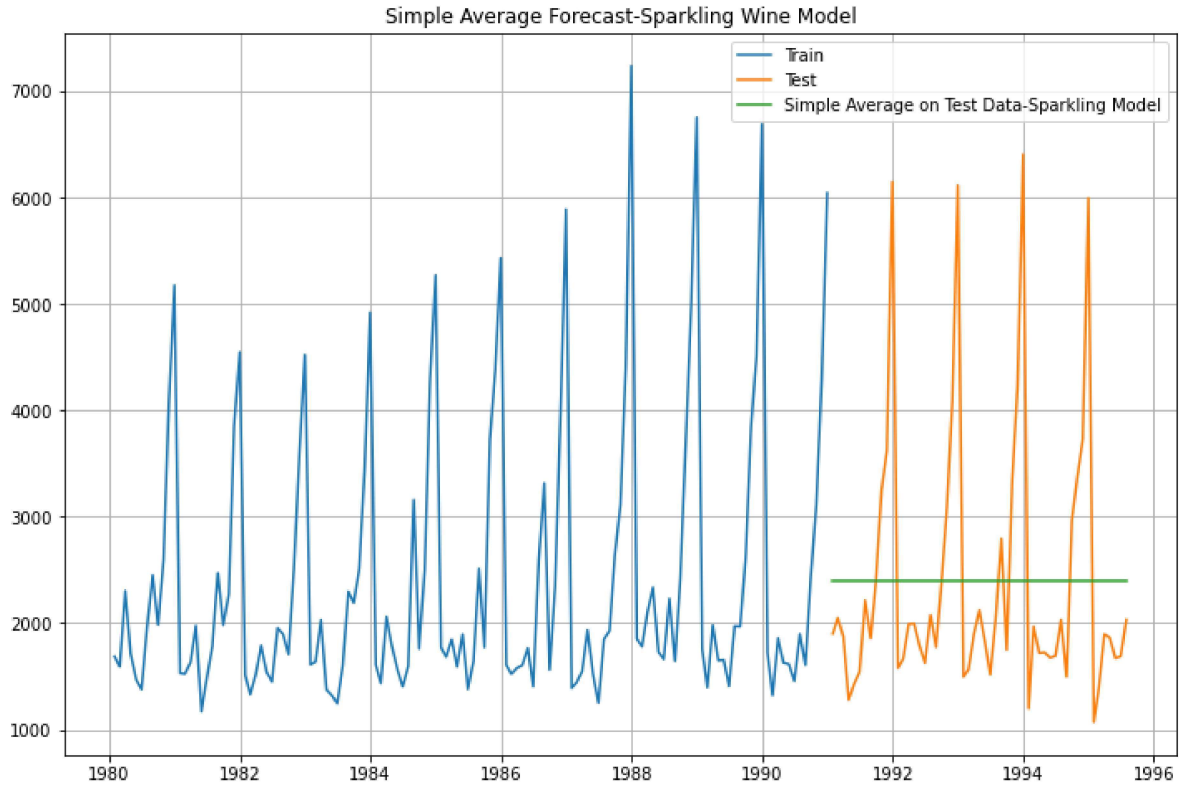
```
SimpleAverage_test['mean_forecast'] = train1['Sparkling'].mean()
SimpleAverage_test.head()
```

Out[92]:

Time_Stamp	Sparkling	mean_forecast
1991-01-31	1902	2403.780303
1991-02-28	2049	2403.780303
1991-03-31	1874	2403.780303
1991-04-30	1279	2403.780303
1991-05-31	1432	2403.780303

In [93]:

```
plt.figure(figsize=(12,8))
plt.plot(SimpleAverage_train['Sparkling'], label='Train')
plt.plot(SimpleAverage_test['Sparkling'], label='Test')
plt.plot(SimpleAverage_test['mean_forecast'], label='Simple Average on Test Data-Sparkling
plt.legend(loc='best')
plt.title("Simple Average Forecast-Sparkling Wine Model")
plt.grid();
```



Model Evaluation-Simple Average-Sparkling Model

In [94]:

```
rmse_model3_test = metrics.mean_squared_error(test1['Sparkling'],SimpleAverage_test['mean_f
print("For Simple Average forecast on the Test Data_Sparkling,  RMSE is %3.3f" %(rmse_model
```

```
For Simple Average forecast on the Test Data_Sparkling,  RMSE is 1275.082
```

In [95]:

```
resultsDf_3 = pd.DataFrame({'Test RMSE': [rmse_model3_test]},index=['SimpleAverageModel'])

resultsDf = pd.concat([resultsDf, resultsDf_3])
resultsDf
```

Out[95]:

	Test RMSE
RegressionOnTime	1275.867052
NaiveModel	3864.279352
SimpleAverageModel	1275.081804

Simple Average-Rose Wine Model

In [96]:

```
SimpleAverage_train2 = train2.copy()
SimpleAverage_test2 = test2.copy()
```

In [97]:

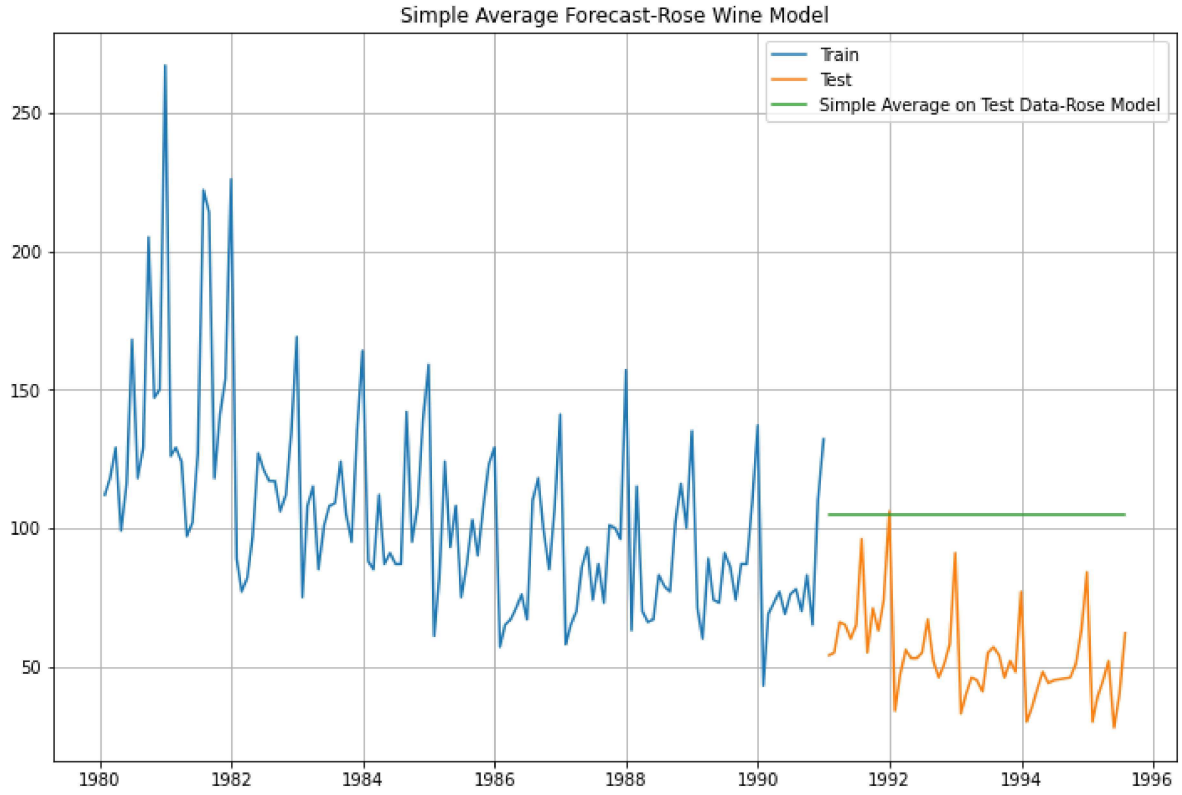
```
SimpleAverage_test2['mean_forecast'] = train2['Rose'].mean()
SimpleAverage_test2.head()
```

Out[97]:

Time_Stamp	Rose	mean_forecast
1991-01-31	54.0	104.939394
1991-02-28	55.0	104.939394
1991-03-31	66.0	104.939394
1991-04-30	65.0	104.939394
1991-05-31	60.0	104.939394

In [98]:

```
plt.figure(figsize=(12,8))
plt.plot(SimpleAverage_train2['Rose'], label='Train')
plt.plot(SimpleAverage_test2['Rose'], label='Test')
plt.plot(SimpleAverage_test2['mean_forecast'], label='Simple Average on Test Data-Rose Mode
plt.legend(loc='best')
plt.title("Simple Average Forecast-Rose Wine Model")
plt.grid();
```



Model Evaluation-Simple Average-Rose Model

In [99]:

```
rmse_model3_test1 = metrics.mean_squared_error(test2['Rose'],SimpleAverage_test2['mean_fore
print("For Simple Average forecast on the Test Data_Rose,  RMSE is %3.3f" %(rmse_model3_tes
```

```
For Simple Average forecast on the Test Data_Rose,  RMSE is 53.461
```

In [100]:

```
resultsDf_3_rose = pd.DataFrame({'Test RMSE': [rmse_model3_test1]},index=['SimpleAverageMod

resultsDf_rose = pd.concat([resultsDf_rose, resultsDf_3_rose])
resultsDf_rose
```

Out[100]:

	Test RMSE
RegressionOnTime	2372.449884
NaiveModel	79.718773
SimpleAverageModel	53.460570

Model 4: Simple Exponential Smoothing

SES-Sparkling Wine Model

In [101]:

```
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
```

In [102]:

```
SES_train = train1.copy()
SES_test = test1.copy()
```

In [103]:

```
model_SES = SimpleExpSmoothing(SES_train['Sparkling'])
```

In [104]:

```
model_SES_autofit = model_SES.fit(optimized=True)
```

In [105]:

```
model_SES_autofit.params
```

Out[105]:

```
{'smoothing_level': 0.049607360581862936,
 'smoothing_trend': nan,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 1818.535750008871,
 'initial_trend': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [106]:

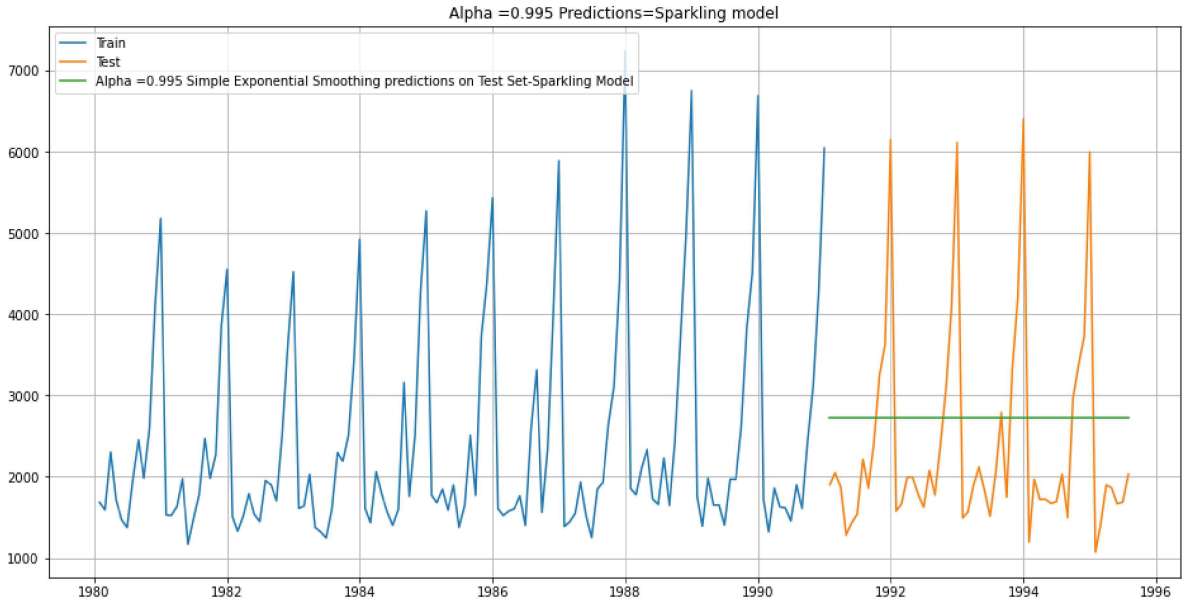
```
SES_test['predict'] = model_SES_autofit.forecast(steps=len(test1))
SES_test.head()
```

Out[106]:

Time_Stamp	Sparkling	predict
1991-01-31	1902	2724.932624
1991-02-28	2049	2724.932624
1991-03-31	1874	2724.932624
1991-04-30	1279	2724.932624
1991-05-31	1432	2724.932624

In [107]:

```
plt.figure(figsize=(16,8))
plt.plot(SES_train['Sparkling'], label='Train')
plt.plot(SES_test['Sparkling'], label='Test')

plt.plot(SES_test['predict'], label='Alpha =0.995 Simple Exponential Smoothing predictions

plt.legend(loc='best')
plt.grid()
plt.title('Alpha =0.995 Predictions=Sparkling model');
```



Model Evaluation for α = 0.995 : Simple Exponential Smoothing-Sparkling Model

In [108]:

```
se_model4_test = metrics.mean_squared_error(SES_test['Sparkling'],SES_test['predict'],square
int("For Alpha =0.995 Simple Exponential Smoothing Model forecast on the Test Data-Sparkling
```

```
For Alpha =0.995 Simple Exponential Smoothing Model forecast on the Test Dat
a-Sparkling, RMSE is 1316.035
```

In [109]:

```
resultsDf_4 = pd.DataFrame({'Test RMSE': [rmse_model4_test]},index=['Alpha=0.995,SimpleExpo

resultsDf = pd.concat([resultsDf, resultsDf_4])
resultsDf
```

Out[109]:

	Test RMSE
RegressionOnTime	1275.867052
NaiveModel	3864.279352
SimpleAverageModel	1275.081804
Alpha=0.995,SimpleExponentialSmoothing	1316.035487

SES-Rose Wine Model

In [110]:

```
SES_train2 = train2.copy()
SES_test2 = test2.copy()
```

In [111]:

```
model_SES2 = SimpleExpSmoothing(SES_train2['Rose'])
```

In [112]:

```
model_SES_autofit2 = model_SES2.fit(optimized=True)
```

In [113]:

```
model_SES_autofit2.params
```

Out[113]:

```
{'smoothing_level': 0.0987493111726833,
 'smoothing_trend': nan,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 134.38720226208358,
 'initial_trend': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [114]:

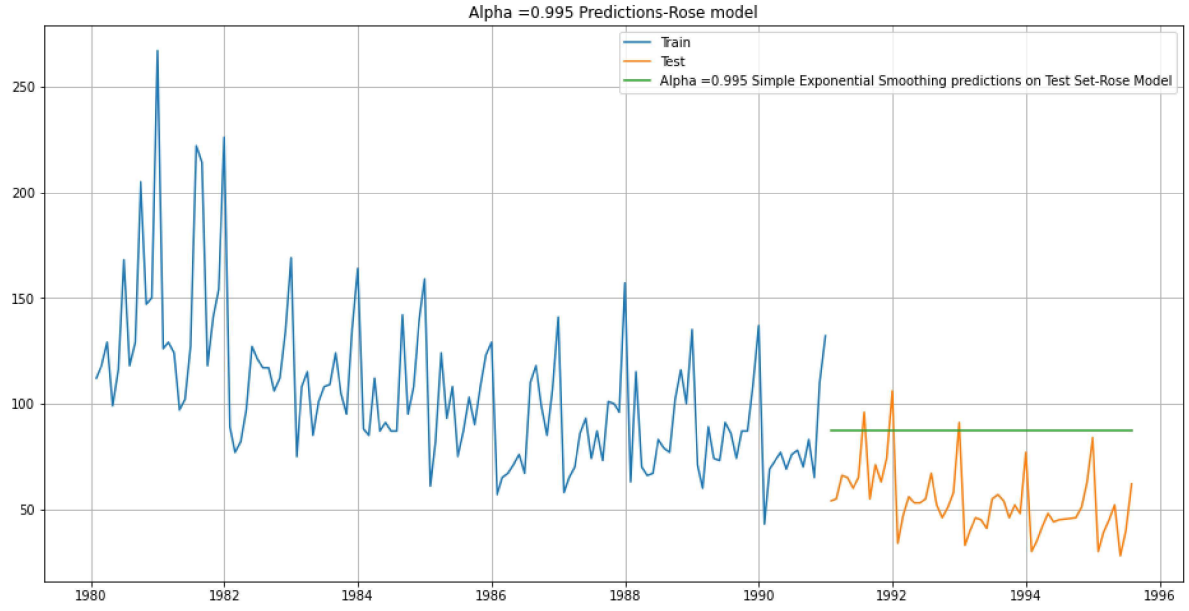
```
SES_test2['predict'] = model_SES_autofit2.forecast(steps=len(test2))
SES_test.head()
```

Out[114]:

Time_Stamp	Sparkling	predict
1991-01-31	1902	2724.932624
1991-02-28	2049	2724.932624
1991-03-31	1874	2724.932624
1991-04-30	1279	2724.932624
1991-05-31	1432	2724.932624

In [115]:

```
plt.figure(figsize=(16,8))
plt.plot(SES_train2['Rose'], label='Train')
plt.plot(SES_test2['Rose'], label='Test')

plt.plot(SES_test2['predict'], label='Alpha =0.995 Simple Exponential Smoothing predictions

plt.legend(loc='best')
plt.grid()
plt.title('Alpha =0.995 Predictions-Rose model');
```



Model Evaluation for α = 0.995 : Simple Exponential Smoothing- Rose Model

In [116]:

```
rmse_model4_test2 = metrics.mean_squared_error(SES_test2['Rose'],SES_test2['predict'],squar
print("For Alpha =0.995 Simple Exponential Smoothing Model forecast on the Test Data, RMSE
```

```
For Alpha =0.995 Simple Exponential Smoothing Model forecast on the Test Dat
a, RMSE is 36.796
```

In [117]:

```
resultsDf_4_rose = pd.DataFrame({'Test RMSE': [rmse_model4_test2]},index=['Alpha=0.995,Simp

resultsDf_rose = pd.concat([resultsDf_rose, resultsDf_4_rose])
resultsDf_rose
```

Out[117]:

	Test RMSE
RegressionOnTime	2372.449884
NaiveModel	79.718773
SimpleAverageModel	53.460570
Alpha=0.995,SimpleExponentialSmoothing	36.796227

Model 5: Double Exponential Smoothing (Holt's Model)

Two parameters α and β are estimated in this model. Level and Trend are accounted for in this model.

DES - Sparkling Wine Model

In [118]:

```
DES_train = train1.copy()
DES_test = test1.copy()
```

In [119]:

```
model_DES = Holt(DES_train['Sparkling'])
```

In [120]:

```
model_DES_autofit = model_DES.fit(optimized=True)
```

In [121]:

```
model_DES_autofit.params
```

Out[121]:

```
{'smoothing_level': 0.6885714285714285,
 'smoothing_trend': 9.999999999999999e-05,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 1686.0,
 'initial_trend': -95.0,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [122]:

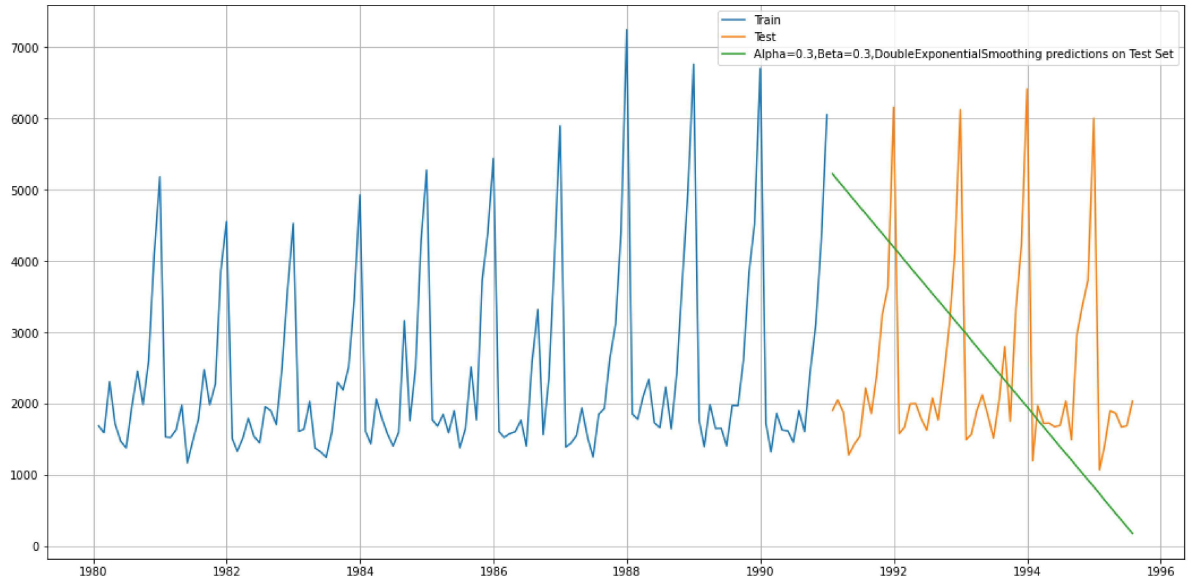
```
DES_test['predict'] = model_DES_autofit.forecast(steps=len(test1))
DES_test.head()
```

Out[122]:

Time_Stamp	Sparkling	predict
1991-01-31	1902	5221.278699
1991-02-28	2049	5127.886554
1991-03-31	1874	5034.494409
1991-04-30	1279	4941.102264
1991-05-31	1432	4847.710119

In [123]:

```
plt.figure(figsize=(18,9))
plt.plot(DES_train['Sparkling'], label='Train')
plt.plot(DES_test['Sparkling'], label='Test')

plt.plot(DES_test['predict'], label='Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing predicti


plt.legend(loc='best')
plt.grid();
```



Model Evaluation- DES-Sparkling Model

In [124]:

```
rmse_model5_test = metrics.mean_squared_error(DES_test['Sparkling'],DES_test['predict'],squ
print("For Alpha =0.995 Double Exponential Smoothing Model forecast on the Test Data, RMSE
```

```
For Alpha =0.995 Double Exponential Smoothing Model forecast on the Test Dat
a, RMSE is 2007.239
```

In [125]:

```
resultsDf_5 = pd.DataFrame({'Test RMSE': [rmse_model5_test]},index=['Alpha=0.995,DoubleExpo

resultsDf = pd.concat([resultsDf, resultsDf_5])
resultsDf
```

Out[125]:

	Test RMSE
RegressionOnTime	1275.867052
NaiveModel	3864.279352
SimpleAverageModel	1275.081804
Alpha=0.995,SimpleExponentialSmoothing	1316.035487
Alpha=0.995,DoubleExponentialSmoothing	2007.238526

DES - Rose Wine Model

In [126]:

```
DES_train2 = train2.copy()
DES_test2 = test2.copy()
```

In [127]:

```
model_DES2 = Holt(DES_train2['Rose'])
```

In [128]:

```
model_DES2_autofit = model_DES2.fit(optimized=True)
```

In [129]:

```
model_DES2_autofit.params
```

Out[129]:

```
{'smoothing_level': 0.017549790270679714,
 'smoothing_trend': 3.236153800377395e-05,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 138.82081494774005,
 'initial_trend': -0.492580228245491,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [130]:

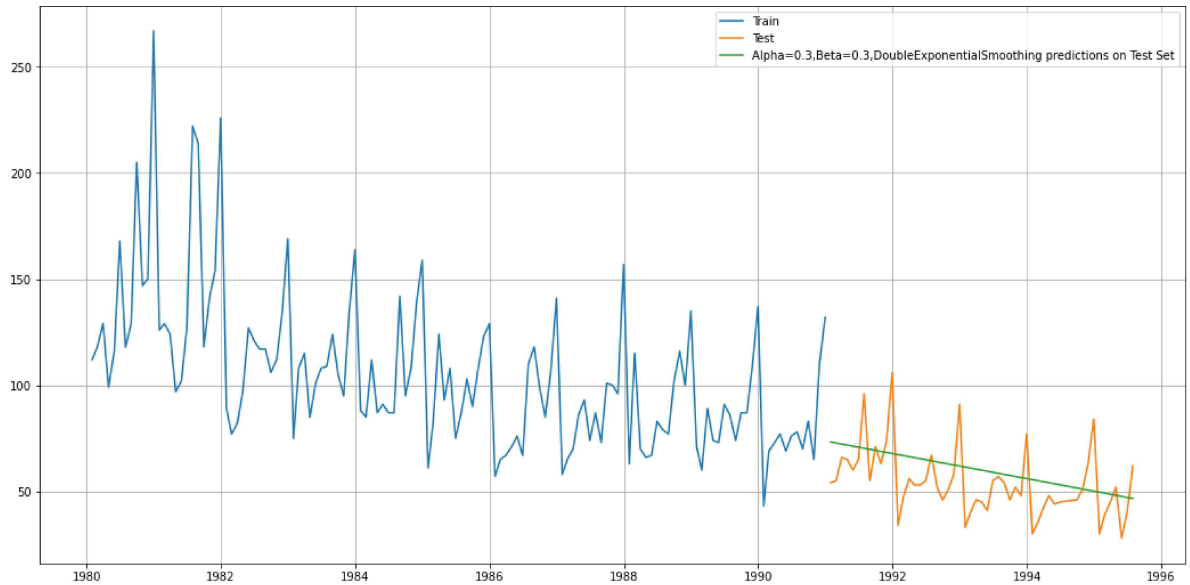
```
DES_test2['predict'] = model_DES2_autofit.forecast(steps=len(test2))
DES_test2.head()
```

Out[130]:

Time_Stamp	Rose	predict
1991-01-31	54.0	73.259732
1991-02-28	55.0	72.767150
1991-03-31	66.0	72.274569
1991-04-30	65.0	71.781987
1991-05-31	60.0	71.289405

In [131]:

```
plt.figure(figsize=(18,9))
plt.plot(DES_train2['Rose'], label='Train')
plt.plot(DES_test2['Rose'], label='Test')

plt.plot(DES_test2['predict'], label='Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing predict

plt.legend(loc='best')
plt.grid();
```



Model Evaluation - DES - Rose Model

In [132]:

```
rmse_model5_test2 = metrics.mean_squared_error(DES_test2['Rose'],DES_test2['predict'],squar
print("For Alpha =0.995 Double Exponential Smoothing Model forecast on the Test Data, RMSE
```

```
For Alpha =0.995 Double Exponential Smoothing Model forecast on the Test Dat
a, RMSE is 15.707
```

In [133]:

```
resultsDf5_rose = pd.DataFrame({'Test RMSE': [rmse_model5_test2]},index=['Alpha=0.995,Doubl

resultsDf_rose = pd.concat([resultsDf_rose, resultsDf5_rose])
resultsDf_rose
```

Out[133]:

	Test RMSE
RegressionOnTime	2372.449884
NaiveModel	79.718773
SimpleAverageModel	53.460570
Alpha=0.995,SimpleExponentialSmoothing	36.796227
Alpha=0.995,DoubleExponentialSmoothing	15.707052

Model 6: Triple Exponential Smoothing (Holt - Winter's Model)

Three parameters α , β and γ are estimated in this model. Level, Trend and Seasonality are accounted for in this model

TES- Sparkling Wine Model

In [134]:

```
TES_train = train1.copy()
TES_test = test1.copy()
```

In [135]:

```
model_TES = ExponentialSmoothing(TES_train['Sparkling'],trend='additive',seasonal='multipli
```

In [136]:

```
model_TES_autofit = model_TES.fit()
```

In [137]:

```
model_TES_autofit.params
```

Out[137]:

```
{'smoothing_level': 0.111108139467838,
 'smoothing_trend': 0.06172875597197263,
 'smoothing_seasonal': 0.3950479631147446,
 'damping_trend': nan,
 'initial_level': 1639.9340657558994,
 'initial_trend': -12.22494561218149,
 'initial_seasons': array([1.06402008, 1.02352078, 1.40671876, 1.20165543,
0.97593   ,
        0.97100155, 1.31897446, 1.69588922, 1.3895294 , 1.81476396,
        2.85150039, 3.62470528]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [138]:

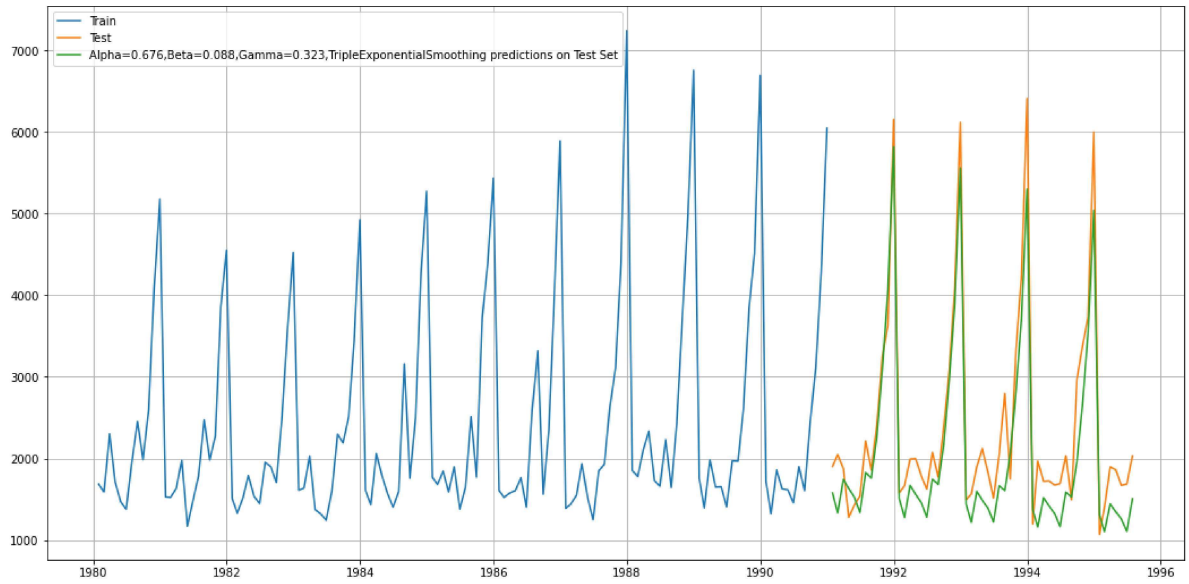
```
TES_test['auto_predict'] = model_TES_autofit.forecast(steps=len(test1))
TES_test.head()
```

Out[138]:

Time_Stamp	Sparkling	auto_predict
1991-01-31	1902	1577.224489
1991-02-28	2049	1333.677558
1991-03-31	1874	1745.945679
1991-04-30	1279	1630.411925
1991-05-31	1432	1523.289070

In [139]:

```
plt.figure(figsize=(18,9))
plt.plot(TES_train['Sparkling'], label='Train')
plt.plot(TES_test['Sparkling'], label='Test')

plt.plot(TES_test['auto_predict'], label='Alpha=0.676,Beta=0.088,Gamma=0.323,TripleExponent


plt.legend(loc='best')
plt.grid();
```



Model Evaluation-TES-Sparkling Wine Model

In [140]:

```
= metrics.mean_squared_error(TES_test['Sparkling'],TES_test['auto_predict'],squared=False)
=0.676,Beta=0.088,Gamma=0.323, Triple Exponential Smoothing Model forecast on the Test Data,
```

```
For Alpha=0.676,Beta=0.088,Gamma=0.323, Triple Exponential Smoothing Model f
orecast on the Test Data,  RMSE is 469.768
```

In [141]:

```
resultsDf6 = pd.DataFrame({'Test RMSE': [rmse_model6_test]}
                          ,index=['Alpha=0.676,Beta=0.088,Gamma=0.323,TripleExponentialSmo

resultsDf = pd.concat([resultsDf, resultsDf6])
resultsDf
```

Out[141]:

	Test RMSE
RegressionOnTime	1275.867052
NaiveModel	3864.279352
SimpleAverageModel	1275.081804
Alpha=0.995,SimpleExponentialSmoothing	1316.035487
Alpha=0.995,DoubleExponentialSmoothing	2007.238526
Alpha=0.676,Beta=0.088,Gamma=0.323,TripleExponentialSmoothing	469.767970

TES- Rose Wine Model

In [142]:

```
TES_train2 = train2.copy()
TES_test2 = test2.copy()
```

In [143]:

```
model_TES2 = ExponentialSmoothing(TES_train2['Rose'],trend='additive',seasonal='multiplicat
```

In [144]:

```
model_TES_autofit2 = model_TES2.fit()
```

In [145]:

```
model_TES_autofit2.params
```

Out[145]:

```
{'smoothing_level': 0.06569374607191865,
 'smoothing_trend': 0.05192938504457338,
 'smoothing_seasonal': 3.879136202038614e-06,
 'damping_trend': nan,
 'initial_level': 54.10985491750761,
 'initial_trend': -0.33471965714896845,
 'initial_seasons': array([2.08282313, 2.36326666, 2.58210206, 2.25702695,
2.53757493,
        2.76639991, 3.04101803, 3.23434567, 3.06747277, 3.00164124,
        3.49893806, 4.82552476]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [146]:

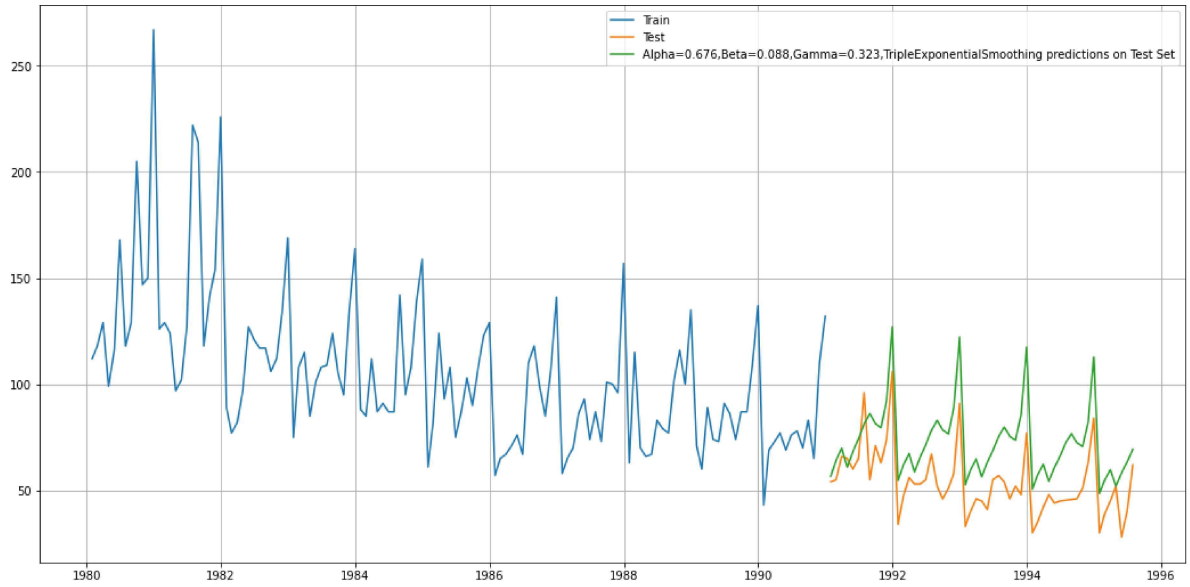
```
TES_test2['auto_predict'] = model_TES_autofit2.forecast(steps=len(test2))
TES_test2.head()
```

Out[146]:

Time_Stamp	Rose	auto_predict
1991-01-31	54.0	56.689174
1991-02-28	55.0	64.129166
1991-03-31	66.0	69.856436
1991-04-30	65.0	60.877474
1991-05-31	60.0	68.237072

In [147]:

```
plt.figure(figsize=(18,9))
plt.plot(TES_train2['Rose'], label='Train')
plt.plot(TES_test2['Rose'], label='Test')

plt.plot(TES_test2['auto_predict'], label='Alpha=0.676,Beta=0.088,Gamma=0.323,TripleExponen


plt.legend(loc='best')
plt.grid();
```



Model Evaluation-TES-Rose Wine Model

In [148]:

```
= metrics.mean_squared_error(TES_test2['Rose'],TES_test2['auto_predict'],squared=False)
).676,Beta=0.088,Gamma=0.323, Triple Exponential Smoothing Model forecast on the Test Data,
```

```
For Alpha=0.676,Beta=0.088,Gamma=0.323, Triple Exponential Smoothing Model f
orecast on the Test Data,  RMSE is 21.020
```

In [149]:

```
resultsDf6_rose = pd.DataFrame({'Test RMSE': [rmse_model6_test2]}
                           ,index=['Alpha=0.676,Beta=0.088,Gamma=0.323,TripleExponentialSmo

resultsDf_rose = pd.concat([resultsDf_rose, resultsDf6_rose])
resultsDf_rose
```

Out[149]:

	Test RMSE
RegressionOnTime	2372.449884
NaiveModel	79.718773
SimpleAverageModel	53.460570
Alpha=0.995,SimpleExponentialSmoothing	36.796227
Alpha=0.995,DoubleExponentialSmoothing	15.707052
Alpha=0.676,Beta=0.088,Gamma=0.323,TripleExponentialSmoothing	21.019620

Q5 Check for the stationarity of the data on which the model is being built on using appropriate statistical tests and also mention the hypothesis for the statistical test. If the data is found to be non-stationary, take appropriate steps to make it stationary. Check the new data for stationarity and comment. Note: Stationarity should be checked at alpha = 0.05.

Check for stationarity of the whole Time Series data.