Project: Time Series Forecasting

20.06.2021

—

Peeyush Vardhan

Mobile: +91-9515456727

Email: peeyushvrdhn@gmail.com

Case Study 1: Sparkling Wine Prediction

Problem Statement:

For this particular assignment, the data of different types of wine sales in the 20th century is to be analysed. Both of these data are from the same company but of different wines. As an analyst in the ABC Estate Wines, you are tasked to analyse and forecast Wine Sales in the 20th century.

Data set: Sparkling.csv

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

Solution:

- Reading the data:

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

```

	Sparkling
YearMonth	
1980-01-01	1686
1980-02-01	1591
1980-03-01	2304
1980-04-01	1712
1980-05-01	1471

- Checking the timestamp:

```
1  df.index
```

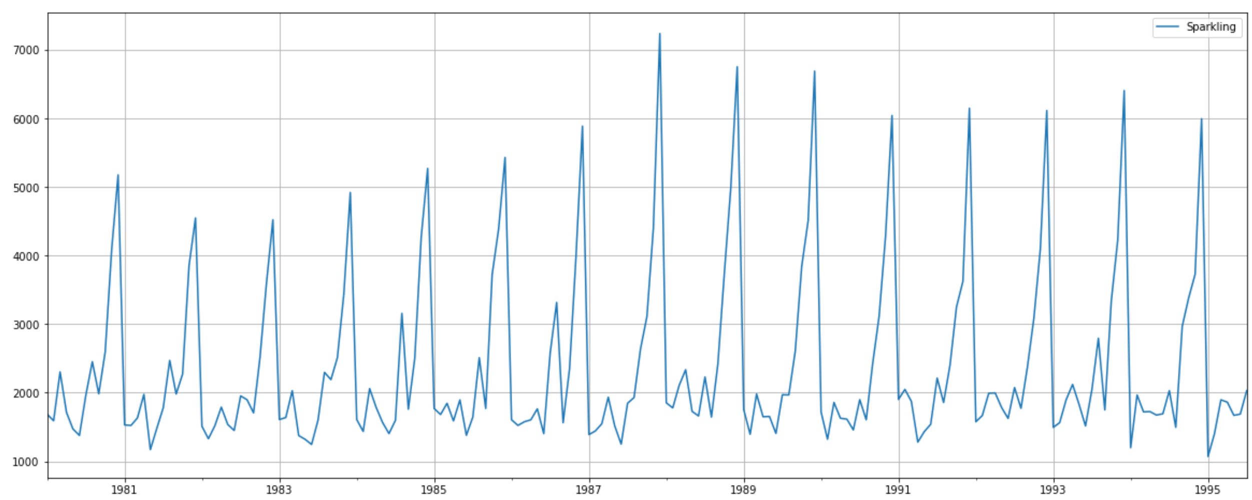
```
DatetimeIndex(['1980-01-01', '1980-02-01', '1980-03-01', '1980-04-01',
               '1980-05-01', '1980-06-01', '1980-07-01', '1980-08-01',
               '1980-09-01', '1980-10-01',
               ...
               '1994-10-01', '1994-11-01', '1994-12-01', '1995-01-01',
               '1995-02-01', '1995-03-01', '1995-04-01', '1995-05-01',
               '1995-06-01', '1995-07-01'],
              dtype='datetime64[ns]', name='YearMonth', length=187, freq=None)
```

- Checking data information:
    - *The number of rows are 187*
    - *The number of columns are 1*

```
1  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 187 entries, 1980-01-01 to 1995-07-01
Data columns (total 1 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Sparkling  187 non-null    int64
dtypes: int64(1)
memory usage: 2.9 KB
```

- Plotting the Time Series to understand the behaviour of the data:



- Insights:
    - The data seems additive in nature and it has seasonality.

- Over the years, the Sparkling sales has been good and it was the best performing in 1988
- Between the years 1989 and 1995 the sales have seemed to be constant, it shows that this particular wine has a constant consumer base.

Question 2: Perform appropriate Exploratory Data Analysis to understand the data and also perform decomposition.

Solution:

- Checking the basic statistical details:

```
df.describe()
```

	Sparkling
count	187.000000
mean	2402.417112
std	1295.111540
min	1070.000000
25%	1605.000000
50%	1874.000000
75%	2549.000000
max	7242.000000

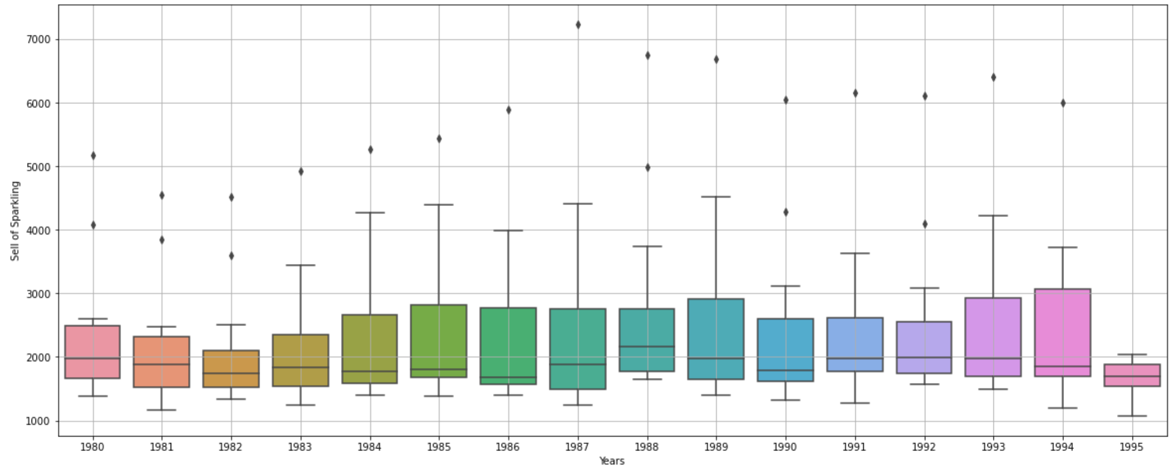
- Checking missing values in the data set:
  - *There are no missing data in the dataframe*

```
#finding number of missing values in the data set

df.isnull().sum()
```

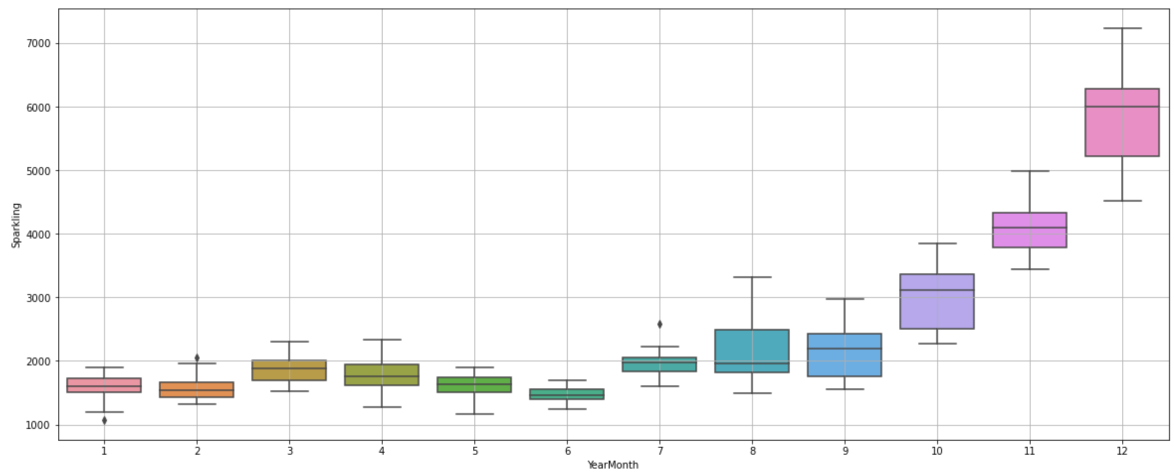
```
Sparkling    0
dtype: int64
```

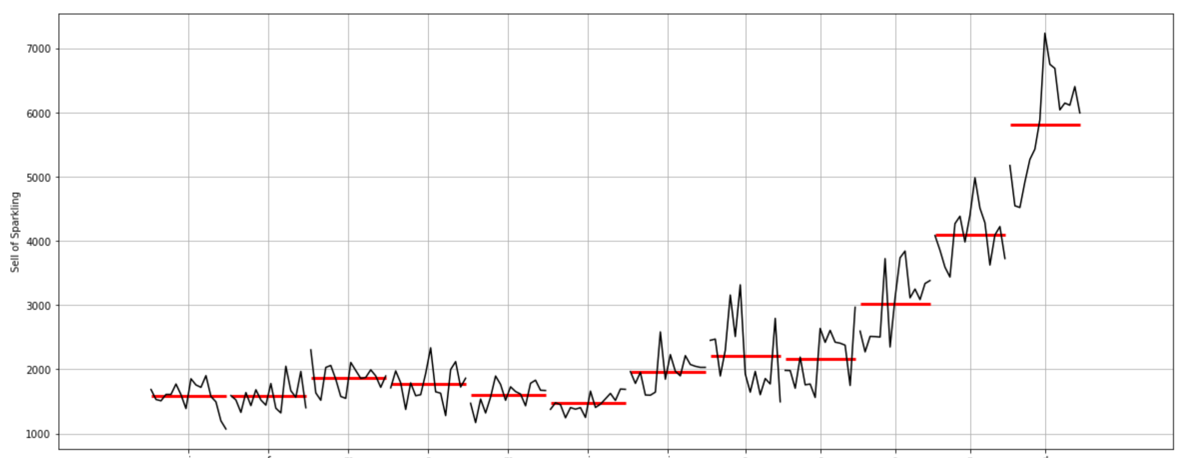
- Plotting a boxplot to understand the spread of accidents across different years and within different months across years:

○ Yearly Boxplot



○ Monthly Boxplot

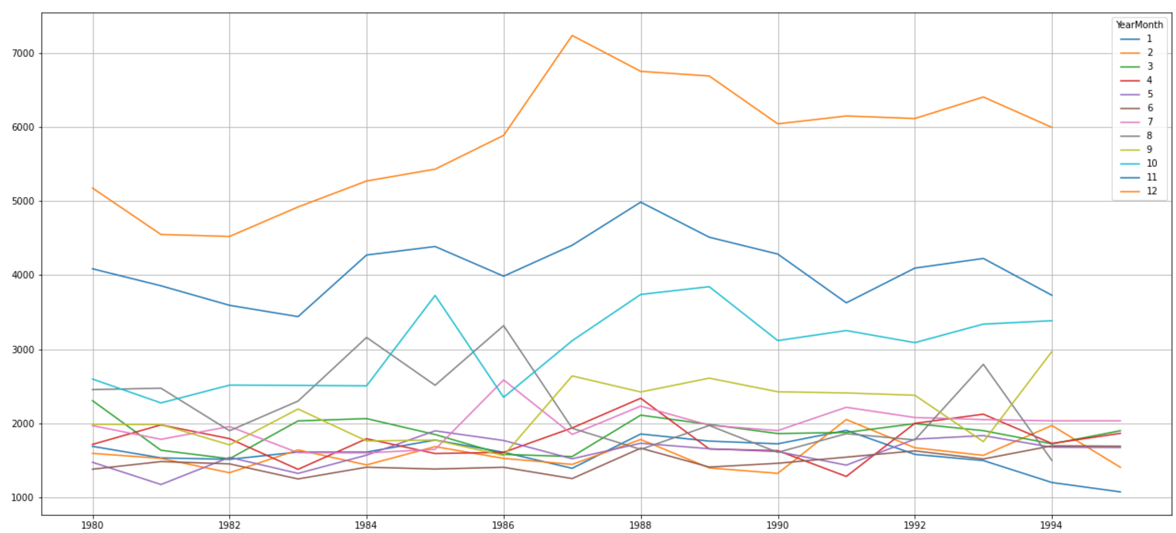




- Looking at Monthly Sale Across Different Years:

YearMonth	1	2	3	4	5	6	7	8	9	10	11	12
YearMonth												
1980	1686.0	1591.0	2304.0	1712.0	1471.0	1377.0	1966.0	2453.0	1984.0	2596.0	4087.0	5179.0
1981	1530.0	1523.0	1633.0	1976.0	1170.0	1480.0	1781.0	2472.0	1981.0	2273.0	3857.0	4551.0
1982	1510.0	1329.0	1518.0	1790.0	1537.0	1449.0	1954.0	1897.0	1706.0	2514.0	3593.0	4524.0
1983	1609.0	1638.0	2030.0	1375.0	1320.0	1245.0	1600.0	2298.0	2191.0	2511.0	3440.0	4923.0
1984	1609.0	1435.0	2061.0	1789.0	1567.0	1404.0	1597.0	3159.0	1759.0	2504.0	4273.0	5274.0
1985	1771.0	1682.0	1846.0	1589.0	1896.0	1379.0	1645.0	2512.0	1771.0	3727.0	4388.0	5434.0
1986	1606.0	1523.0	1577.0	1605.0	1765.0	1403.0	2584.0	3318.0	1562.0	2349.0	3987.0	5891.0
1987	1389.0	1442.0	1548.0	1935.0	1518.0	1250.0	1847.0	1930.0	2638.0	3114.0	4405.0	7242.0
1988	1853.0	1779.0	2108.0	2336.0	1728.0	1661.0	2230.0	1645.0	2421.0	3740.0	4988.0	6757.0
1989	1757.0	1394.0	1982.0	1650.0	1654.0	1406.0	1971.0	1968.0	2608.0	3845.0	4514.0	6694.0
1990	1720.0	1321.0	1859.0	1628.0	1615.0	1457.0	1899.0	1605.0	2424.0	3116.0	4286.0	6047.0
1991	1902.0	2049.0	1874.0	1279.0	1432.0	1540.0	2214.0	1857.0	2408.0	3252.0	3627.0	6153.0
1992	1577.0	1667.0	1993.0	1997.0	1783.0	1625.0	2076.0	1773.0	2377.0	3088.0	4096.0	6119.0
1993	1494.0	1564.0	1898.0	2121.0	1831.0	1515.0	2048.0	2795.0	1749.0	3339.0	4227.0	6410.0
1994	1197.0	1968.0	1720.0	1725.0	1674.0	1693.0	2031.0	1495.0	2968.0	3385.0	3729.0	5999.0
1995	1070.0	1402.0	1897.0	1862.0	1670.0	1688.0	2031.0	NaN	NaN	NaN	NaN	NaN

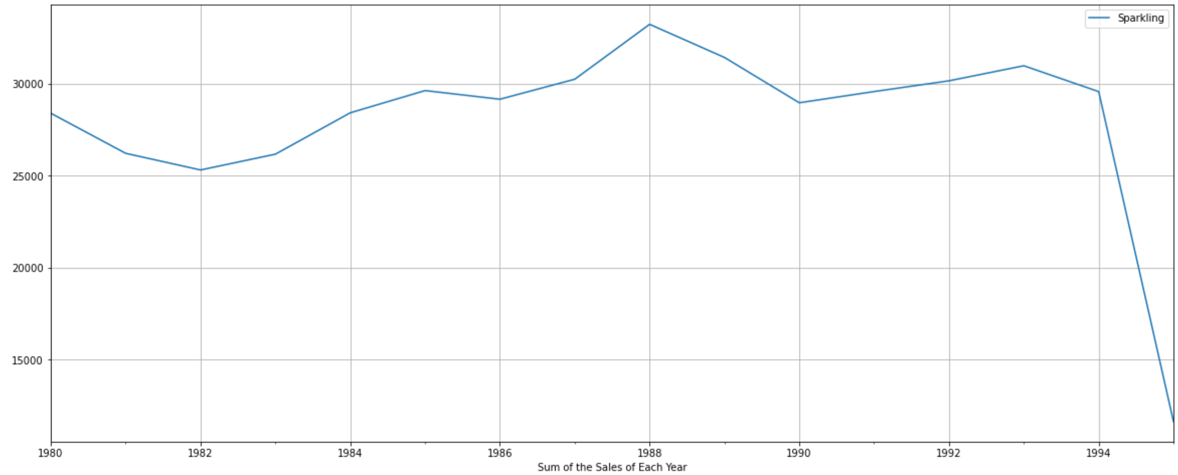
Graphical Representation:



- Details of the annual sale:

YearMonth	Sparkling
1980-12-31	28406
1981-12-31	26227
1982-12-31	25321
1983-12-31	26180
1984-12-31	28431

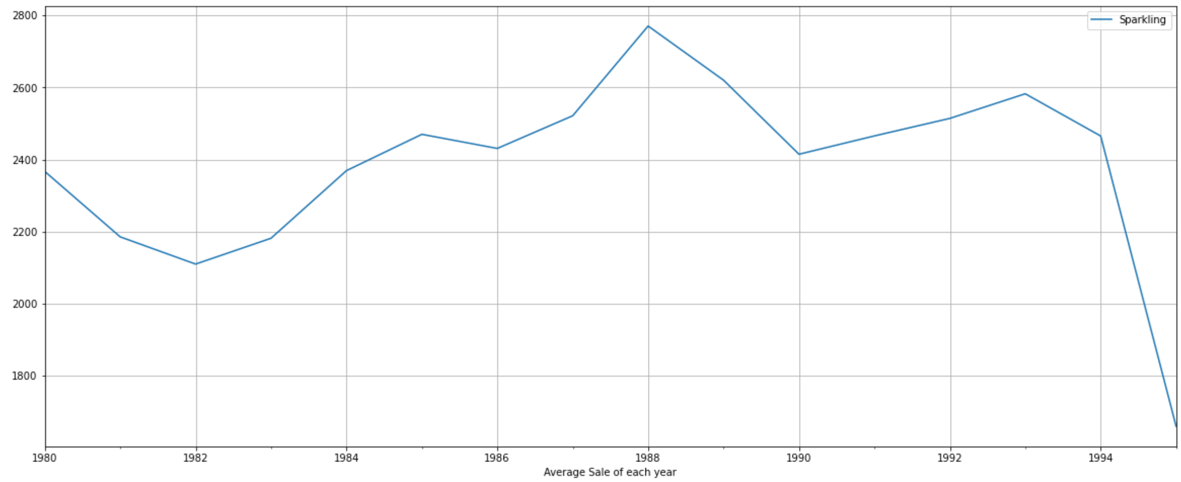
Graphical Representation:



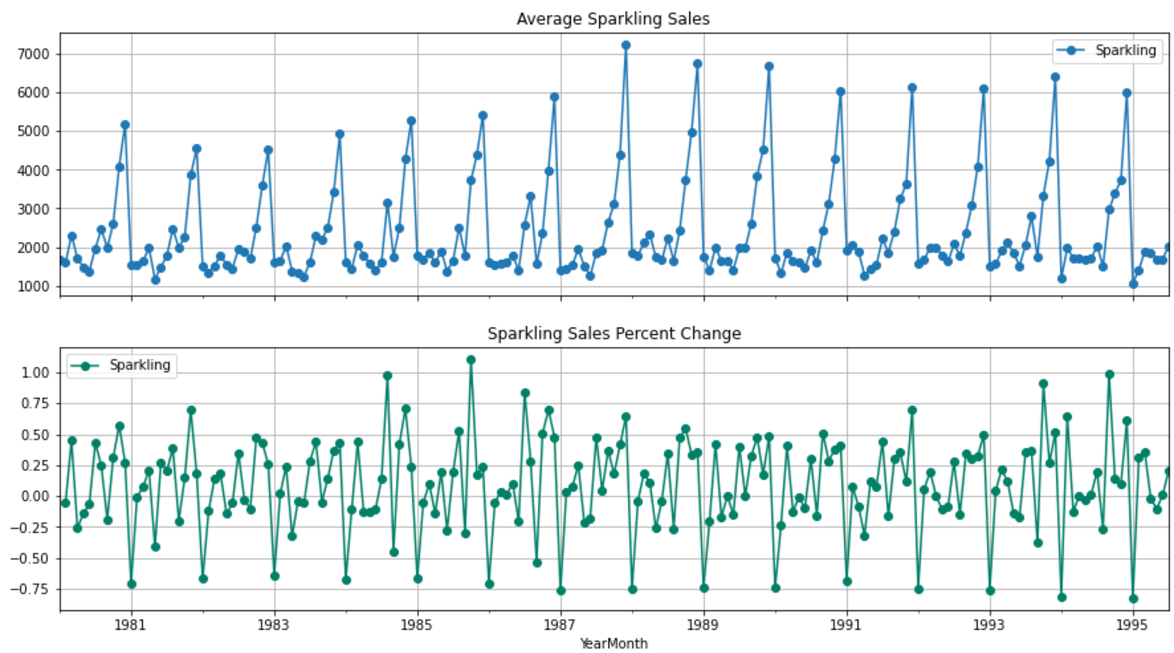
- Details of average the annual sale:

YearMonth	Sparkling
1980-12-31	2367.166667
1981-12-31	2185.583333
1982-12-31	2110.083333
1983-12-31	2181.666667
1984-12-31	2369.250000

Graphical Representation:

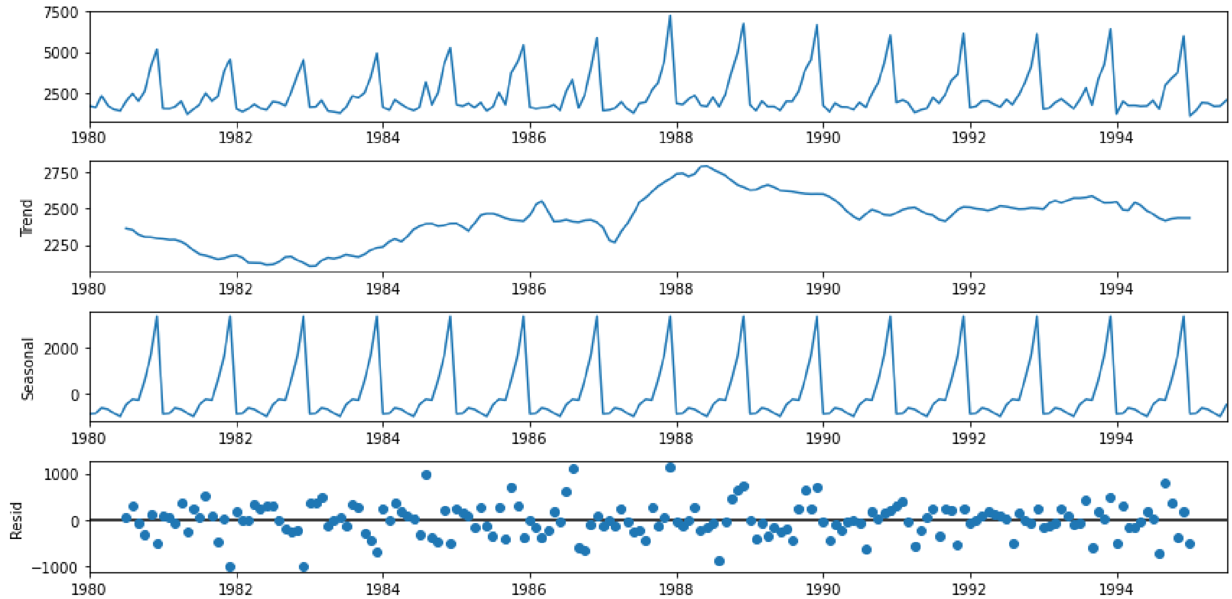


- Average Sparkling Sales per month and the month on month percentage change of Sparkling Sales:

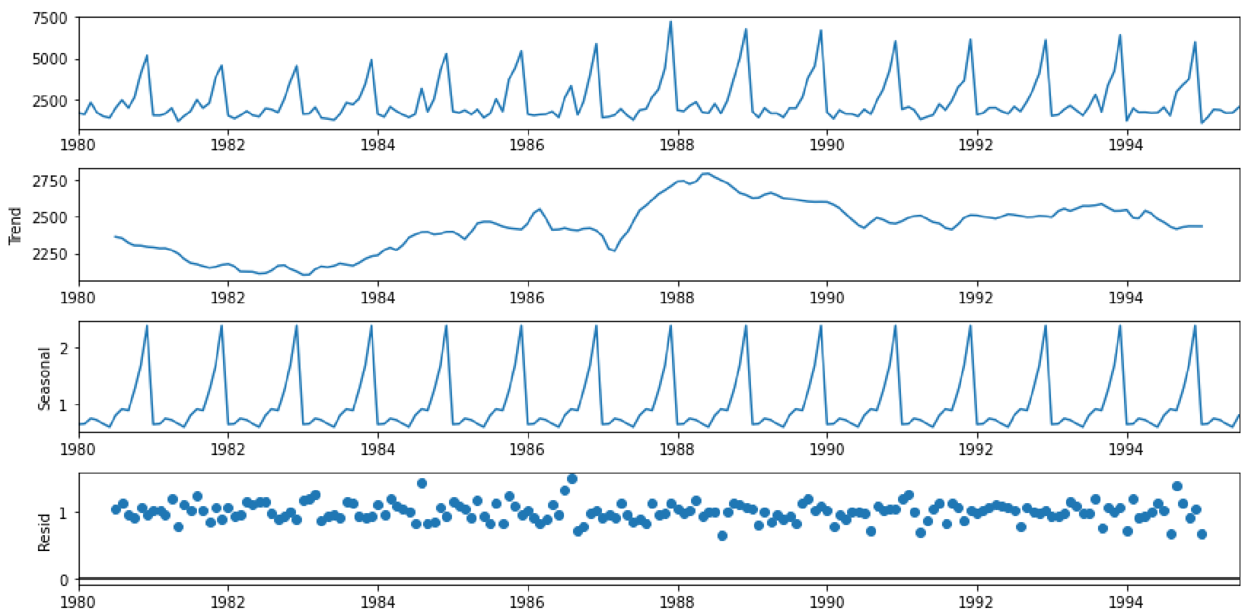


- Representation of decomposition of data for Time series analysis. This will be represented in 2 types:
  - Additive decomposition
  - Multiplicative decomposition

Additive decomposition:



Multiplicative decomposition:



- **Insights from decomposition of data:**
    - It is observed in additive decomposition that seasonality is present on a yearly basis.
    - There is a slight trend in error terms as well in additive decomposition.
    - Multiplicative decomposition also shows seasonality on a yearly basis.

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

Solution:

- Splitting the data into train and test shape:

```
train=df[df.index.year < 1991]
test=df[df.index.year >= 1991]
print("Shape of Training Data is", train.shape)
print("Shape of Test Data is", test.shape)
```

```
Shape of Training Data is (132, 1)
Shape of Test Data is (55, 1)
```

- Bottom data from training set:

```
display(train.tail())
```

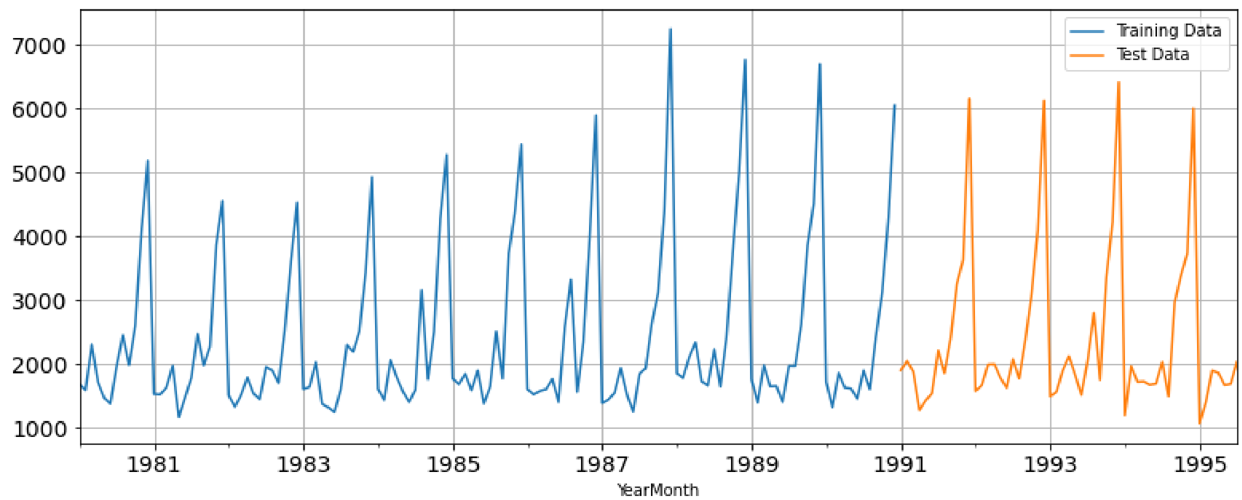
	Sparkling
YearMonth	
1990-08-01	1605
1990-09-01	2424
1990-10-01	3116
1990-11-01	4286
1990-12-01	6047

- Top data from test set:

```
display(test.head())
```

	Sparkling
YearMonth	
1991-01-01	1902
1991-02-01	2049
1991-03-01	1874
1991-04-01	1279
1991-05-01	1432

- Plotting the training and test data:



Question 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.

Solution:

Please refer to the Jupyter Notebook submitted to look into the code.

4.1. Building Regression Model:

- Training Time instance

```
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, 3
4, 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, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123,
124, 125, 126, 127, 128, 129, 130, 131, 132]
Test Time instance
 [133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157,
158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 18
3, 184, 185, 186, 187]
```

- Training data head:

YearMonth	Sparkling	time
1980-01-01	1686	1
1980-02-01	1591	2
1980-03-01	2304	3
1980-04-01	1712	4
1980-05-01	1471	5

- Training data tail:

YearMonth	Sparkling	time
1990-08-01	1605	128
1990-09-01	2424	129
1990-10-01	3116	130
1990-11-01	4286	131
1990-12-01	6047	132

- Test data head:

YearMonth	Sparkling	time
1991-01-01	1902	133
1991-02-01	2049	134
1991-03-01	1874	135
1991-04-01	1279	136
1991-05-01	1432	137

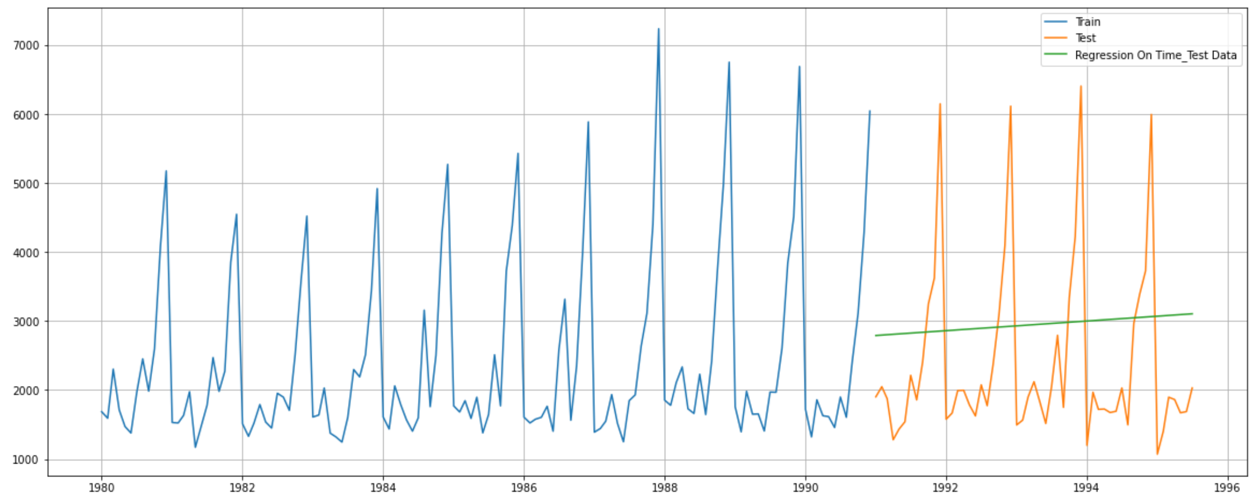
- Test data tail:

YearMonth	Sparkling	time
1995-03-01	1897	183
1995-04-01	1862	184
1995-05-01	1670	185
1995-06-01	1688	186
1995-07-01	2031	187

- Plotting Linear Regression Model:

```
import sklearn
print(sklearn.__version__)
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

```
0.24.2
```



- RMSE model evaluation:

```
#LR MODEL EVALUATION

rmse_lr = metrics.mean_squared_error(test['Sparkling'], lr_predict, squared=False)
print("RMSE of LR is %3.3f" %(rmse_lr))
```

```
RMSE of LR is 1389.135
```

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

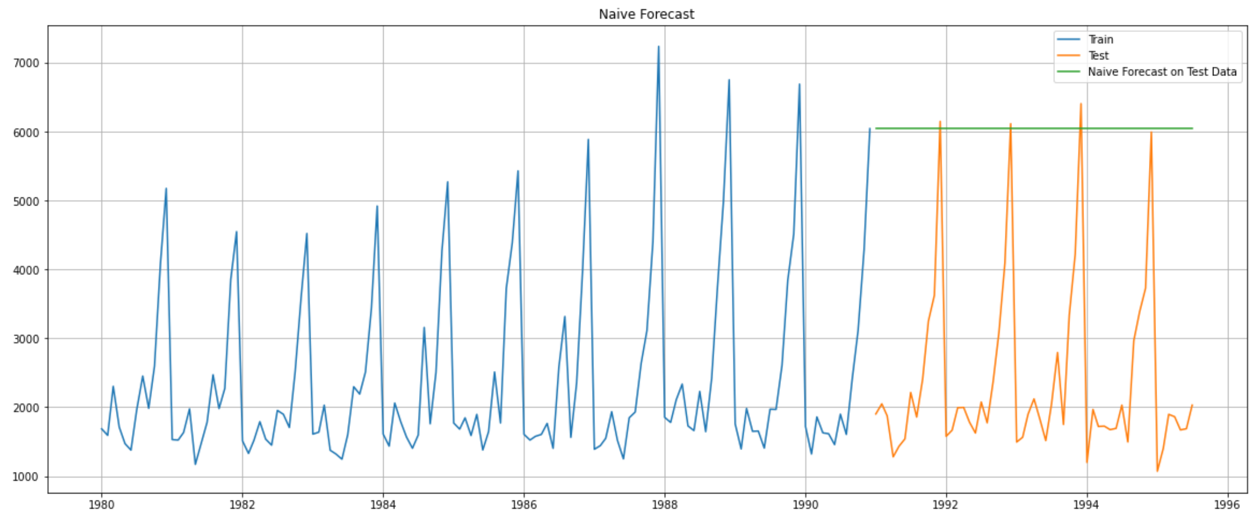
	Test RMSE
RegressionOnTime	1389.135175

4.2. Building Naïve Forecast Model:

- Test data head:

```
YearMonth
1991-01-01    6047
1991-02-01    6047
1991-03-01    6047
1991-04-01    6047
1991-05-01    6047
Name: naive, dtype: int64
```

- Train test data plot:



- Naïve Forecast Model Evaluation:
    - RMSE of NF is 3864.279

```
#NF MODEL EVALUATION

rmse_nf = metrics.mean_squared_error(test['Sparkling'],NM_test['naive'],squared=False)
print("RMSE of NF is %3.3f" %(rmse_nf))
```

RMSE of NF is 3864.279

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

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

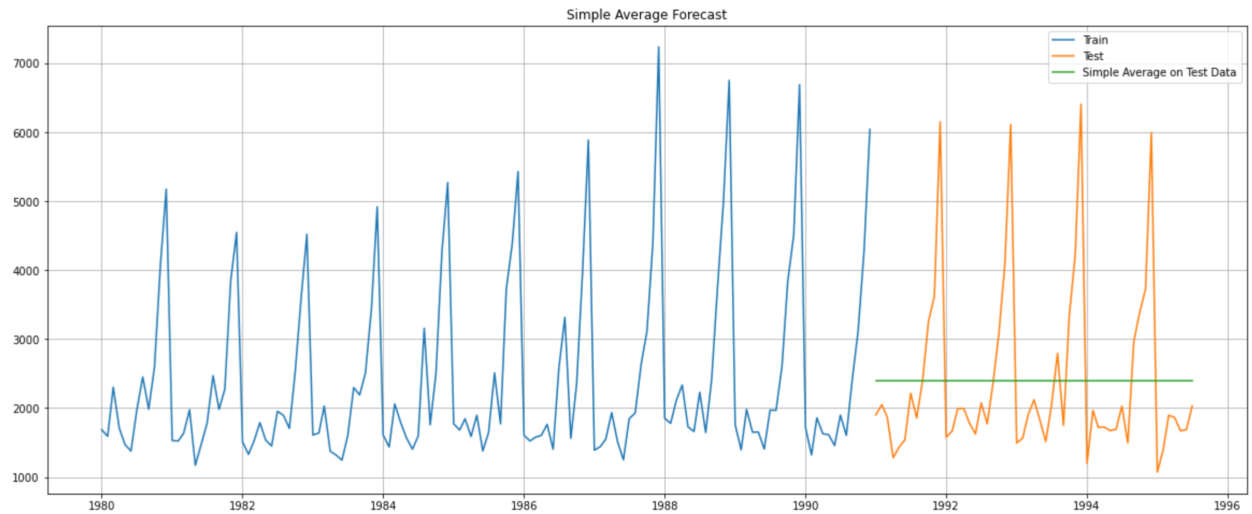
	Test RMSE
RegressionOnTime	1389.135175
NaiveModel	3864.279352

4.3. Building Simple Average Model:

- Test data head:

	Sparkling	mean_forecast
YearMonth		
1991-01-01	1902	2403.780303
1991-02-01	2049	2403.780303
1991-03-01	1874	2403.780303
1991-04-01	1279	2403.780303
1991-05-01	1432	2403.780303

- Train test data plot:



- Simple Average Model Evaluation:

```
#Simple Average Model Evaluation

rmse_sa = metrics.mean_squared_error(test['Sparkling'],SA_test['mean_forecast'],squared=False)
print("RMSE of SA isis %3.3f" %(rmse_sa))
```

RMSE of SA isis 1275.082

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

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

	Test RMSE
RegressionOnTime	1389.135175
NaiveModel	3864.279352
SimpleAverageModel	1275.081804

4.4. Single Exponential Smoothing Model (SES):

```
SES_test = test.copy()
SES_train = train.copy()
```

```
model_SES = SimpleExpSmoothing(SES_train['Sparkling'])
model_SES_autofit = model_SES.fit(optimized=True)
model_SES_autofit.params
```

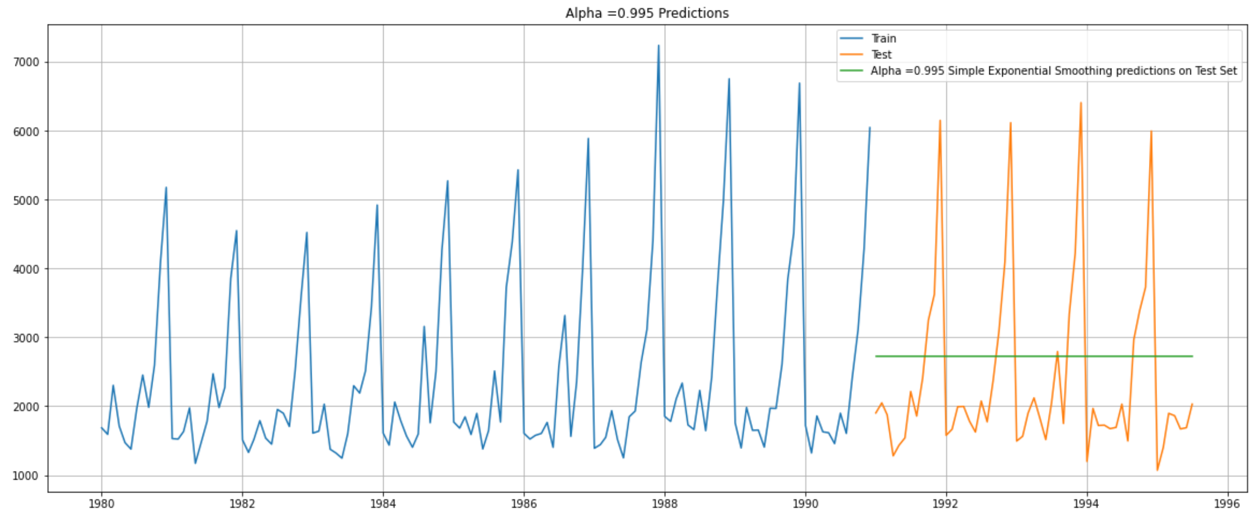
```
C:\Users\91951\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueWarning: No frequency information was
ovided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\91951\anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.py:427: FutureWarning: After 0.13 initialization
ust be handled at model creation
  warnings.warn(
C:\Users\91951\anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.py:920: ConvergenceWarning: Optimization failed
converge. Check mle_retvals.
  warnings.warn(
```

```
{'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}
```

- Test data head:

	Sparkling	predict
YearMonth		
1991-01-01	1902	2724.932624
1991-02-01	2049	2724.932624
1991-03-01	1874	2724.932624
1991-04-01	1279	2724.932624
1991-05-01	1432	2724.932624

- Plotting on both the Training and Test data:



- Single Exponential Smoothing Model Evaluation:

```
#SES MODEL EVALUATION

rmse_ses = metrics.mean_squared_error(SES_test['Sparkling'],SES_test['predict'],squared=False)
print("For Alpha =0.995, RMSE of SES is %3.3f" %(rmse_ses))
```

```
For Alpha =0.995, RMSE of SES is 1316.035
```

```
resultsDf_SES = pd.DataFrame({'Test RMSE': [rmse_ses]},index=['Alpha=0.995,SimpleExponentialSmoothing'])
resultsDf = pd.concat([resultsDf, resultsDf_SES])
resultsDf
```

	Test RMSE
RegressionOnTime	1389.135175
NaiveModel	3864.279352
SimpleAverageModel	1275.081804
Alpha=0.995,SimpleExponentialSmoothing	1316.035487

4.5. Building Double Exponential Smoothing - Holt's Model

```
resultsDf_DES = pd.DataFrame({'Alpha Values':[],'Beta Values':[],'Train RMSE':[],'Test RMSE': []})
resultsDf_DES
```

Alpha Values	Beta Values	Train RMSE	Test RMSE

- Consolidation of Double Exponential Smoothing:

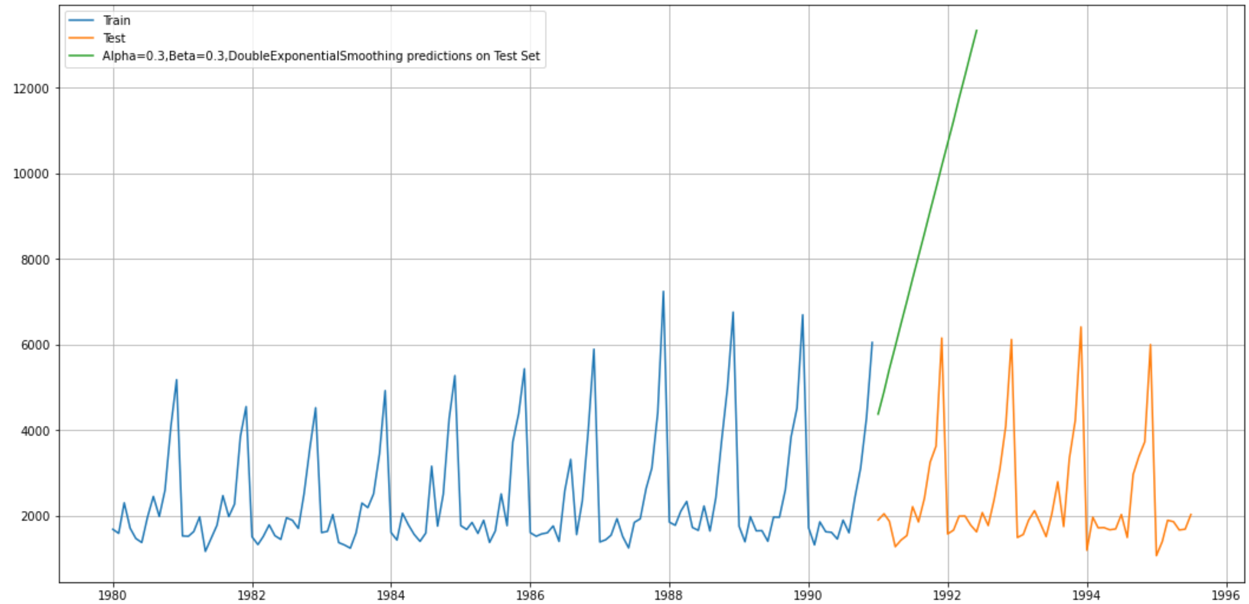
	Alpha Values	Beta Values	Train RMSE	Test RMSE
0	0.3	0.3	2.535396e+06	2.535396e+06
1	0.3	0.4	2.831055e+06	2.831055e+06
2	0.3	0.5	3.138959e+06	3.138959e+06
3	0.3	0.6	3.417235e+06	3.417235e+06
4	0.3	0.7	3.609806e+06	3.609806e+06
...	...	...	...	...
59	1.0	0.6	3.074420e+06	3.074420e+06
60	1.0	0.7	3.331308e+06	3.331308e+06
61	1.0	0.8	3.617656e+06	3.617656e+06
62	1.0	0.9	3.941688e+06	3.941688e+06
63	1.0	1.0	4.316722e+06	4.316722e+06

64 rows × 4 columns

- Double Exponential Smoothing test set top data:

	Alpha Values	Beta Values	Train RMSE	Test RMSE
32	0.7	0.3	2.252068e+06	2.252068e+06
24	0.6	0.3	2.269391e+06	2.269391e+06
40	0.8	0.3	2.277533e+06	2.277533e+06
48	0.9	0.3	2.337537e+06	2.337537e+06
16	0.5	0.3	2.342662e+06	2.342662e+06

- Plotting on both the Training and Test data (Alpha = 0.3, Beta = 0.3)



- Results of Double Exponential Smoothing:

	Test RMSE
RegressionOnTime	1.389135e+03
NaiveModel	3.864279e+03
SimpleAverageModel	1.275082e+03
Alpha=0.995,SimpleExponentialSmoothing	1.316035e+03
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing	2.252068e+06

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

```
# Triple Exponential Smoothing (Holt - Winter's Model)
TES_test = test.copy()
TES_train = train.copy()
```

- Model Parameters:

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

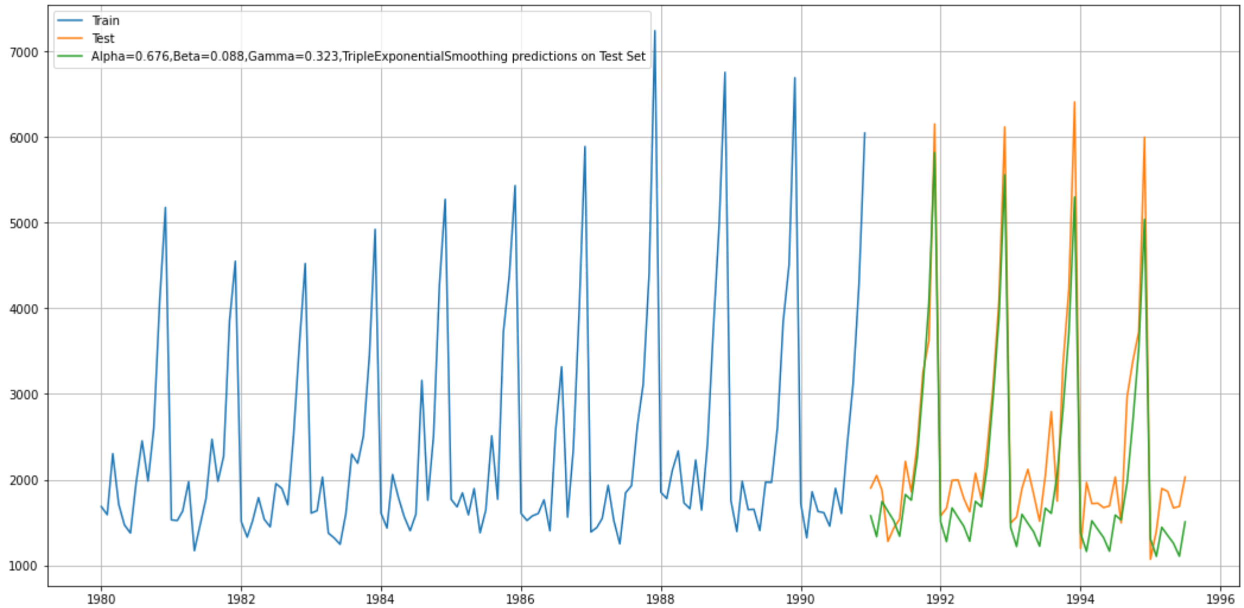
```
model_TES_autofit.params
```

```
{'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}
```

- Triple Exponential Smoothing test set top data:

YearMonth	Sparkling	auto_predict
1991-01-01	1902	1577.224489
1991-02-01	2049	1333.677558
1991-03-01	1874	1745.945679
1991-04-01	1279	1630.411925
1991-05-01	1432	1523.289070

- Plotting on both the Training and Test data (Alpha = 0.676, Beta = 0.088)



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

```
resultsDf_SES = pd.DataFrame({'Test RMSE': [rmse_ses]}
                            ,index=['Alpha=0.676,Beta=0.088,Gamma=0.323,TripleExponentialSmoothing'])

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

```
resultsDf
```

	Test RMSE
RegressionOnTime	1.389135e+03
NaiveModel	3.864279e+03
SimpleAverageModel	1.275082e+03
Alpha=0.995,SimpleExponentialSmoothing	1.316035e+03
Alpha=0.3,Beta=0.3,DoubleExponentialSmoothing	2.252068e+06
Alpha=0.676,Beta=0.088,Gamma=0.323,TripleExponentialSmoothing	1.316035e+03

Question 5: Check for the stationarity of the data on which the model is being built 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.

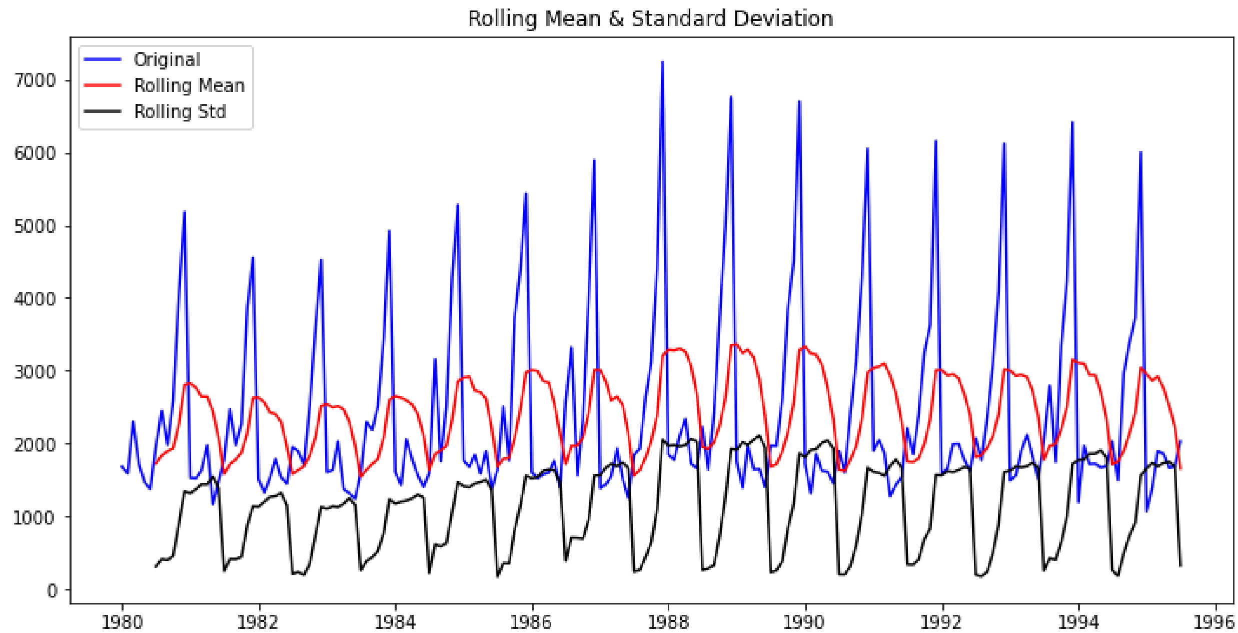
Solution:

Please refer to the Jupyter Notebook submitted to look into the code.

- Steps Involved:
  - *Determining rolling statistics*
  - *Plot rolling statistics*
  - *Perform Dickey-Fuller test*

```
def test_stationarity(timeseries):

    #Determining rolling statistics
    rolmean = timeseries.rolling(window=7).mean() #determining the rolling mean
    rolstd = timeseries.rolling(window=7).std()   #determining the rolling standard deviation

    #Plot rolling statistics:
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    #Perform Dickey-Fuller test:
    print ('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print (dfoutput,'\n')
```

- Plot between Rolling Standard, Rolling mean and Original:

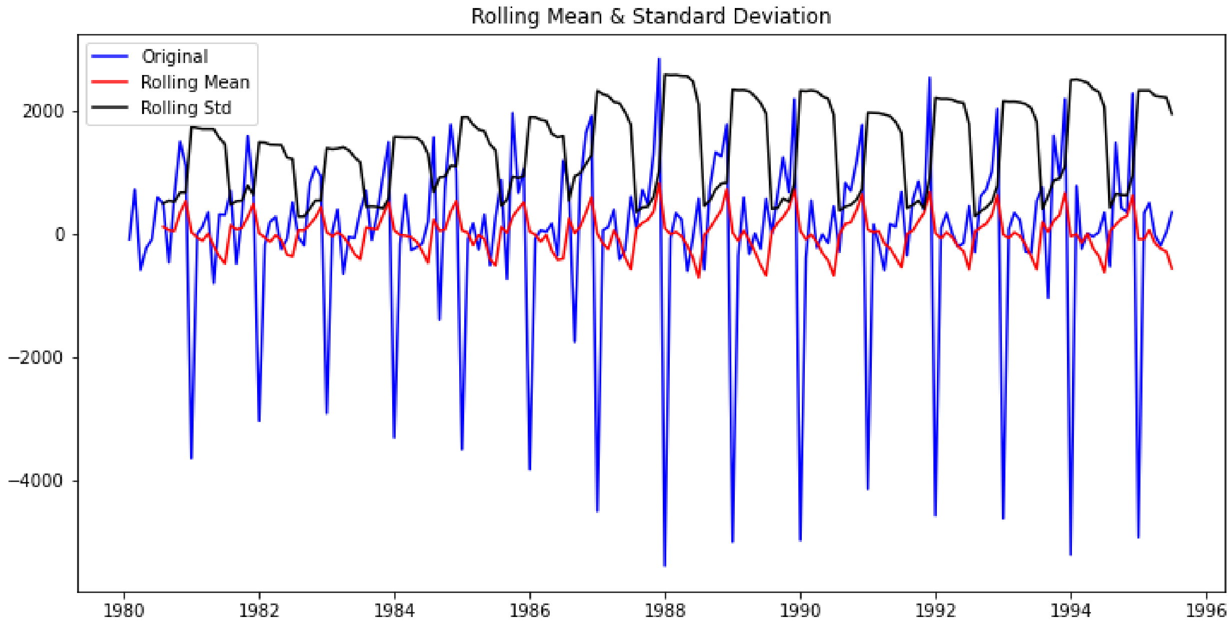


- Results of Dickey-Fuller Test:

```
Results of Dickey-Fuller Test:
Test Statistic                  -1.360497
p-value                          0.601061
#Lags Used                      11.000000
Number of Observations Used    175.000000
Critical Value (1%)             -3.468280
Critical Value (5%)             -2.878202
Critical Value (10%)            -2.575653
dtype: float64
```

- Standards:
    - p-value 0.601061, Thus the series is non-stationary.
    - Difference of order 1, to check the time series again

- Plot between Rolling Standard, Rolling mean and Original:



- Results of Dickey-Fuller Test:

```
Results of Dickey-Fuller Test:
Test Statistic                  -45.050301
p-value                           0.000000
#Lags Used                       10.000000
Number of Observations Used     175.000000
Critical Value (1%)              -3.468280
Critical Value (5%)              -2.878202
Critical Value (10%)             -2.575653
dtype: float64
```

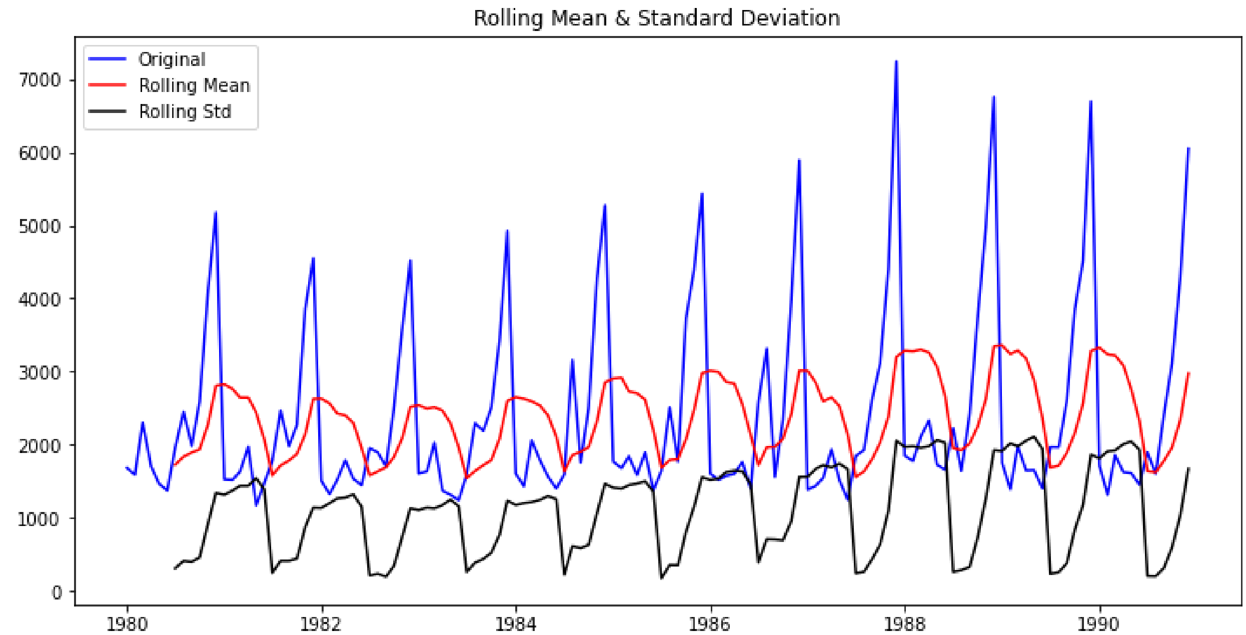
- **Insight:**
  - Now, series is stationary at alpha =0.05

Question 6: Build an automated version of the ARIMA/SARIMA model in which the parameters are selected using the lowest Akaike Information Criteria (AIC) on the training data and evaluate this model on the test data using RMSE.

Solution:

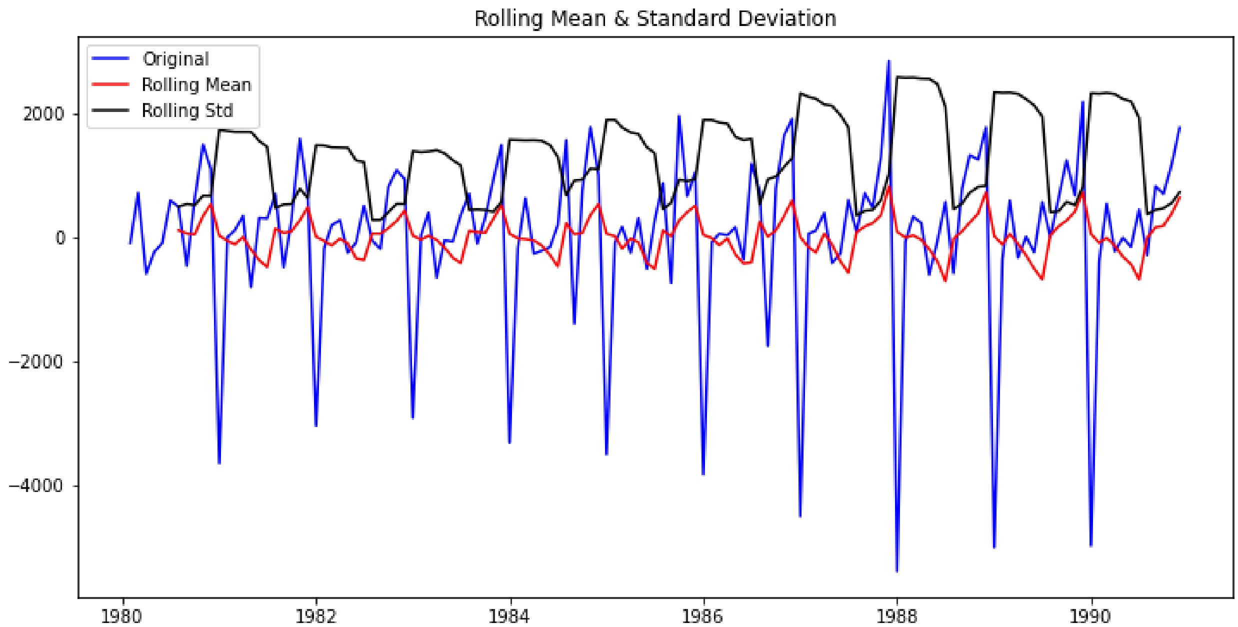
Please refer to the Jupyter Notebook submitted to look into the code.

- Test stationarity Graph:




```
Results of Dickey-Fuller Test:
Test Statistic                  -1.208926
p-value                          0.669744
#Lags Used                      12.000000
Number of Observations Used    119.000000
Critical Value (1%)             -3.486535
Critical Value (5%)             -2.886151
Critical Value (10%)            -2.579896
dtype: float64
```

- Test stationarity Graph by dropping un-necessary values:




```
Results of Dickey-Fuller Test:
Test Statistic                -8.005007e+00
p-value                        2.280104e-12
#Lags Used                     1.100000e+01
Number of Observations Used    1.190000e+02
Critical Value (1%)           -3.486535e+00
Critical Value (5%)           -2.886151e+00
Critical Value (10%)          -2.579896e+00
dtype: float64
```

- Building the parameter combinations for the Model:

```
Some parameter combinations for the Model...
Model: (0, 1, 1)
Model: (0, 1, 2)
Model: (1, 1, 0)
Model: (1, 1, 1)
Model: (1, 1, 2)
Model: (2, 1, 0)
Model: (2, 1, 1)
Model: (2, 1, 2)
```

- The ARIMA Model:

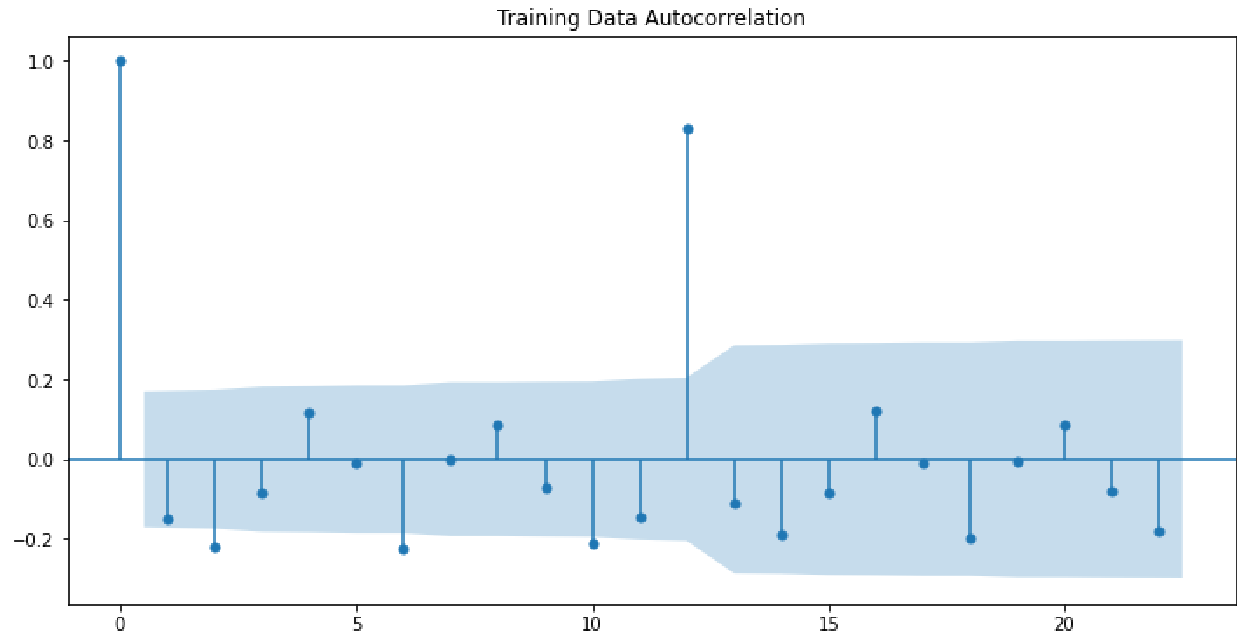
```
                             ARIMA Model Results
==============================================================================
Dep. Variable:           D.Sparkling   No. Observations:                  131
Model:                ARIMA(2, 1, 2)   Log Likelihood               -1099.309
Method:                      css-mle   S.D. of innovations           1012.730
Date:               Sun, 20 Jun 2021   AIC                           2210.619
Time:                       19:00:05   BIC                           2227.870
Sample:                   02-01-1980   HQIC                          2217.628
                        - 12-01-1990
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 5.5843      0.518     10.790      0.000       4.570       6.599
ar.L1.D.Sparkling     1.2700      0.074     17.048      0.000       1.124       1.416
ar.L2.D.Sparkling    -0.5604      0.074     -7.620      0.000      -0.704      -0.416
ma.L1.D.Sparkling    -1.9978      0.042    -47.093      0.000      -2.081      -1.915
ma.L2.D.Sparkling     0.9978      0.042     23.501      0.000       0.915       1.081
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.1333           -0.7073j            1.3359           -0.0888
AR.2            1.1333           +0.7073j            1.3359            0.0888
MA.1            1.0004           +0.0000j            1.0004            0.0000
MA.2            1.0019           +0.0000j            1.0019            0.0000
-----------------------------------------------------------------------------
```

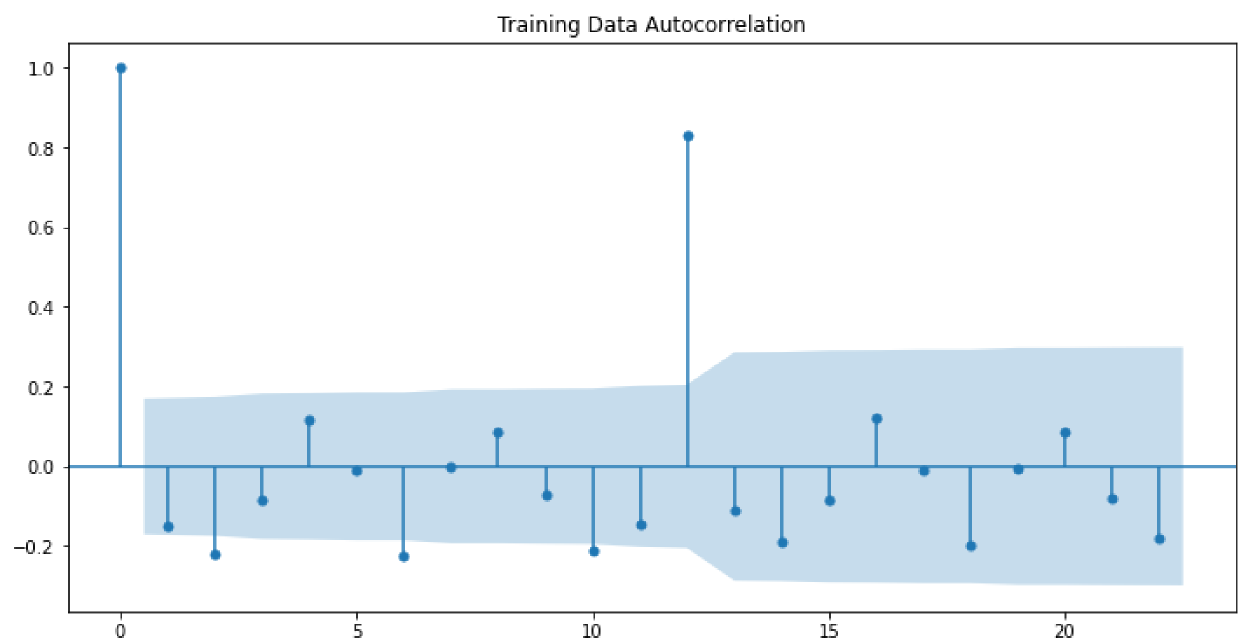
- Prediction of ARIMA Model:
  - RMSE = 1374.546023727508

```
resultsDf = pd.DataFrame({'RMSE': [rmse]}
                         ,index=['ARIMA(2,1,1)'])

resultsDf
```

	RMSE
ARIMA(2,1,1)	1374.546024

- Building an SARIMA model using AIC:





- Examples of the parameter combinations for the Model:

```
Examples of the parameter combinations for the Model are
Model: (0, 1, 1)(0, 0, 1, 4)
Model: (0, 1, 2)(0, 0, 2, 4)
Model: (0, 1, 3)(0, 0, 3, 4)
Model: (1, 1, 0)(1, 0, 0, 4)
Model: (1, 1, 1)(1, 0, 1, 4)
Model: (1, 1, 2)(1, 0, 2, 4)
Model: (1, 1, 3)(1, 0, 3, 4)
Model: (2, 1, 0)(2, 0, 0, 4)
Model: (2, 1, 1)(2, 0, 1, 4)
Model: (2, 1, 2)(2, 0, 2, 4)
Model: (2, 1, 3)(2, 0, 3, 4)
Model: (3, 1, 0)(3, 0, 0, 4)
Model: (3, 1, 1)(3, 0, 1, 4)
Model: (3, 1, 2)(3, 0, 2, 4)
Model: (3, 1, 3)(3, 0, 3, 4)
```

- The SARIMA Model:

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                          Sparkling   No. Observations:                  132
Model:             SARIMAX(2, 1, 3)x(3, 0, 3, 4)   Log Likelihood                -843.516
Date:                            Sun, 20 Jun 2021   AIC                           1711.033
Time:                                    19:03:06   BIC                           1743.972
Sample:                                01-01-1980   HQIC                          1724.403
                                     - 12-01-1990
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          1.5316      0.043     35.260      0.000       1.446       1.617
ar.L2         -0.9458      0.051    -18.720      0.000      -1.045      -0.847
ma.L1         -2.9790      0.107    -27.929      0.000      -3.188      -2.770
ma.L2          3.1855      0.327      9.752      0.000       2.545       3.826
ma.L3         -1.3074      0.214     -6.117      0.000      -1.726      -0.888
ar.S.L4       -0.0070      0.011     -0.655      0.513      -0.028       0.014
ar.S.L8       -0.0232      0.010     -2.392      0.017      -0.042      -0.004
ar.S.L12       1.0449      0.011     95.062      0.000       1.023       1.066
ma.S.L4       -0.0268      0.099     -0.271      0.786      -0.221       0.167
ma.S.L8       -0.0772      0.099     -0.777      0.437      -0.272       0.118
ma.S.L12      -0.6509      0.101     -6.461      0.000      -0.848      -0.453
sigma2      6.933e+04    1.5e-05   4.63e+09      0.000    6.93e+04    6.93e+04
===================================================================================
Ljung-Box (L1) (Q):                   0.10   Jarque-Bera (JB):                42.51
Prob(Q):                              0.75   Prob(JB):                         0.00
Heteroskedasticity (H):               2.61   Skew:                             0.85
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.45
===================================================================================
```

- The Auto ARIMA Summary:

Sparkling	mean	mean_se	mean_ci_lower	mean_ci_upper
1991-01-01	1489.775620	347.338138	809.005379	2170.545861
1991-02-01	1348.987035	349.397804	664.179923	2033.794147
1991-03-01	1848.172847	349.772280	1162.631775	2533.713919
1991-04-01	1667.876173	350.195212	981.506170	2354.246176
1991-05-01	1353.543177	351.063678	665.471013	2041.615342

- RMSE Output: 713.9776935967018

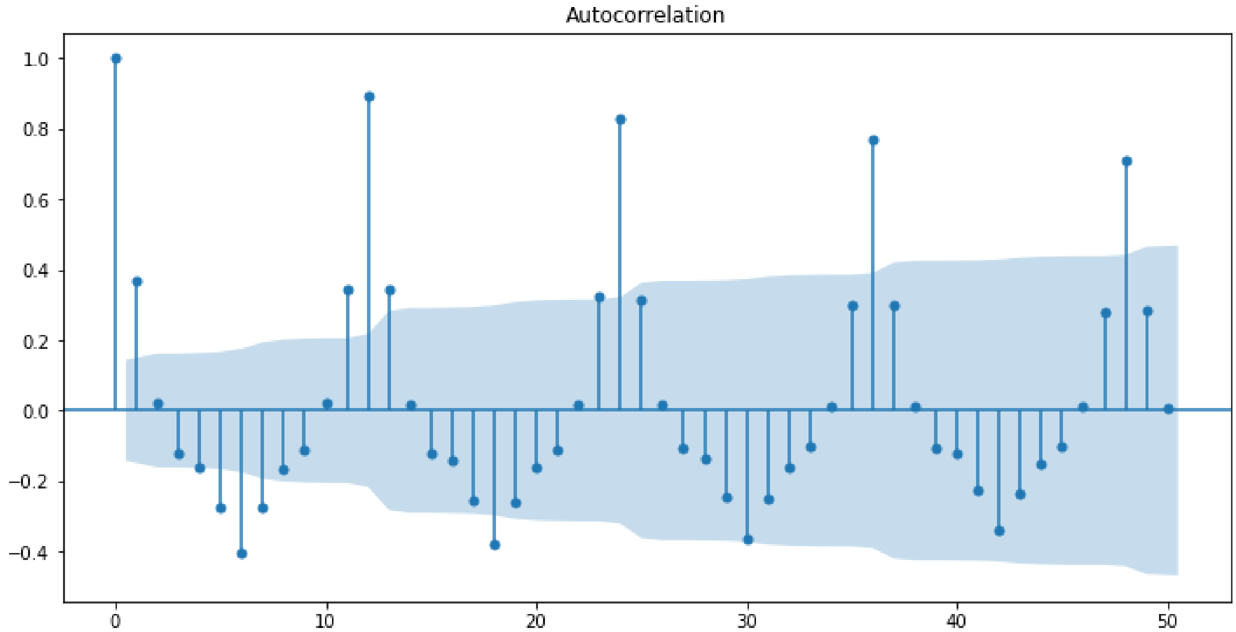
	RMSE
ARIMA(2,1,1)	1374.546024
SARIMA(2,1,3)(0,0,3,12)(SARIMA AIC)	713.977694

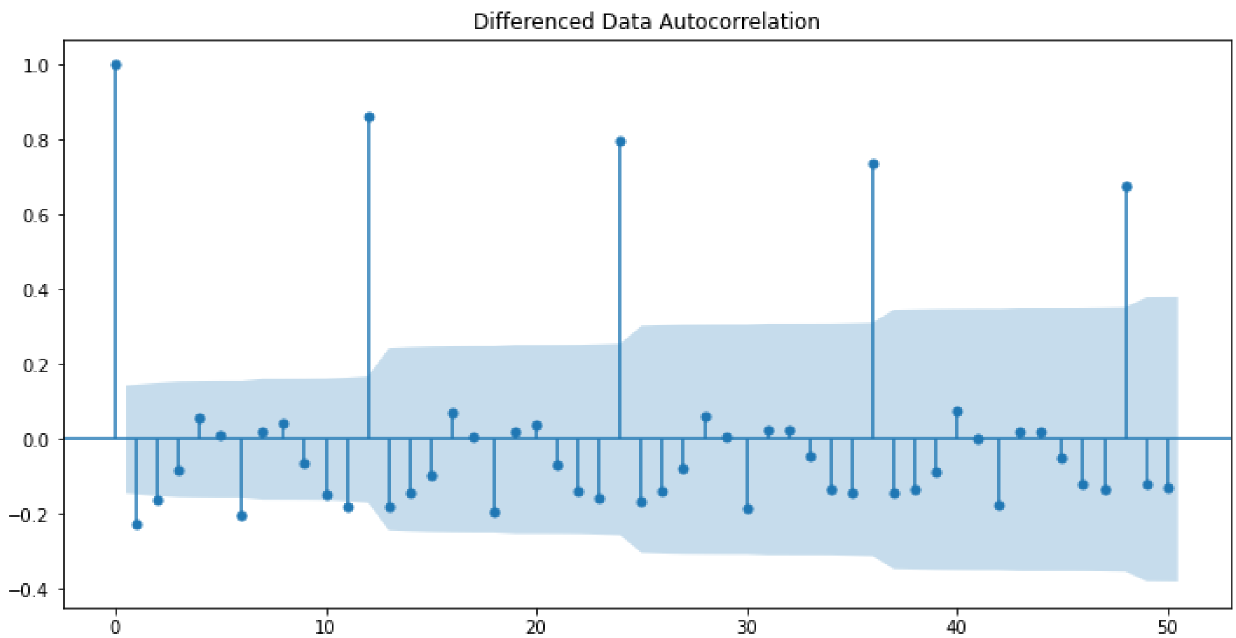
Question 7: Build ARIMA/SARIMA models based on the cut-off points of ACF and PACF on the training data and evaluate this model on the test data using RMSE.

Solution:

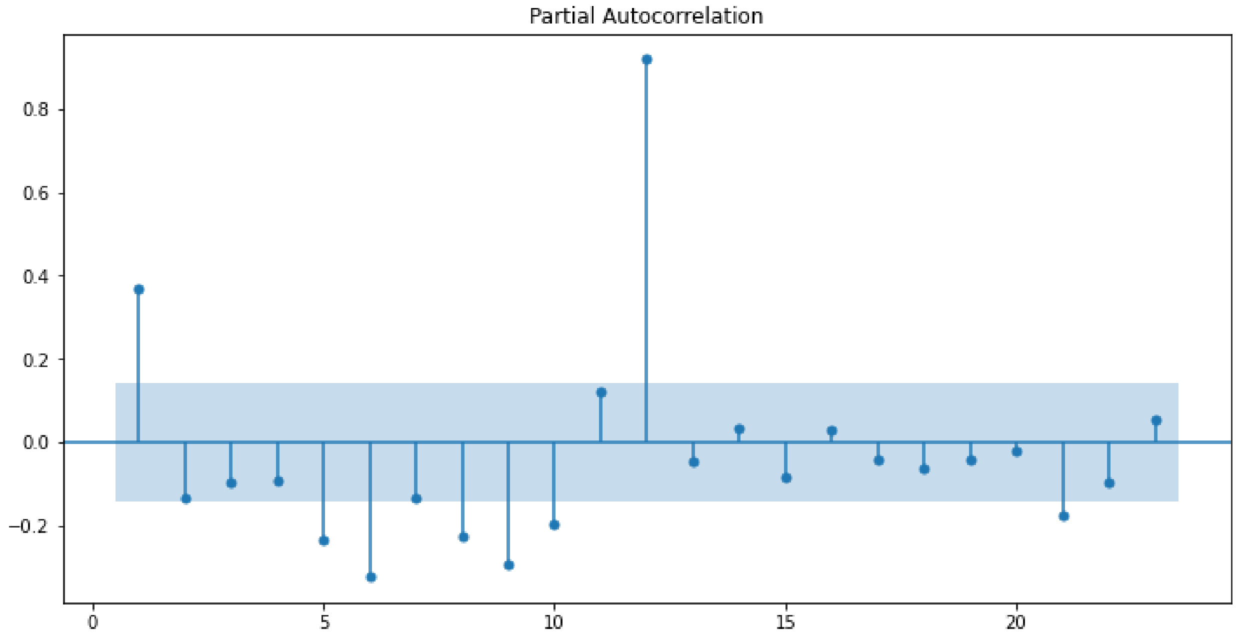
Please refer to the Jupyter Notebook submitted to look into the code.

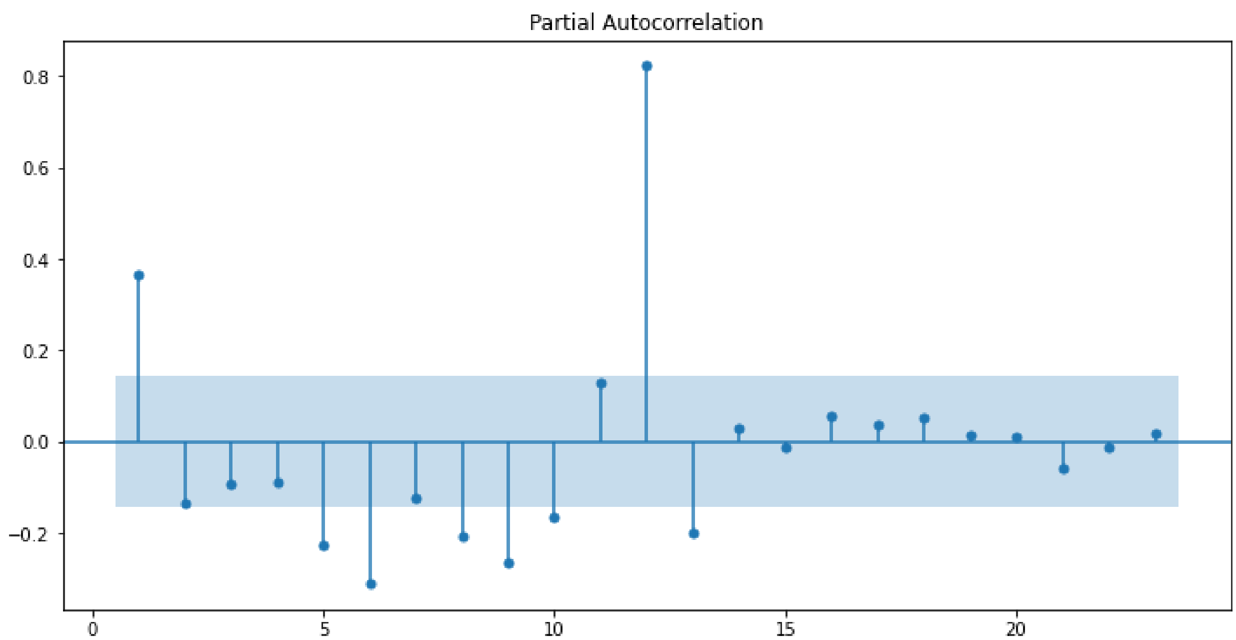
- Plotting the Autocorrelation graph:





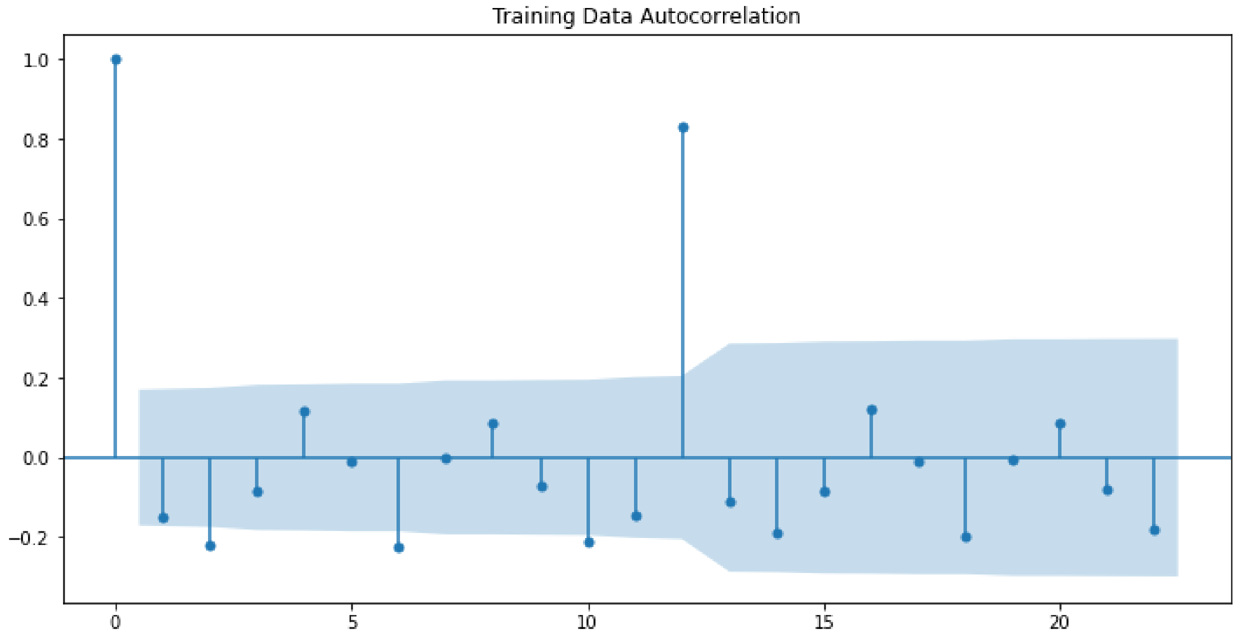
- Plotting the Partial Autocorrelation graph:



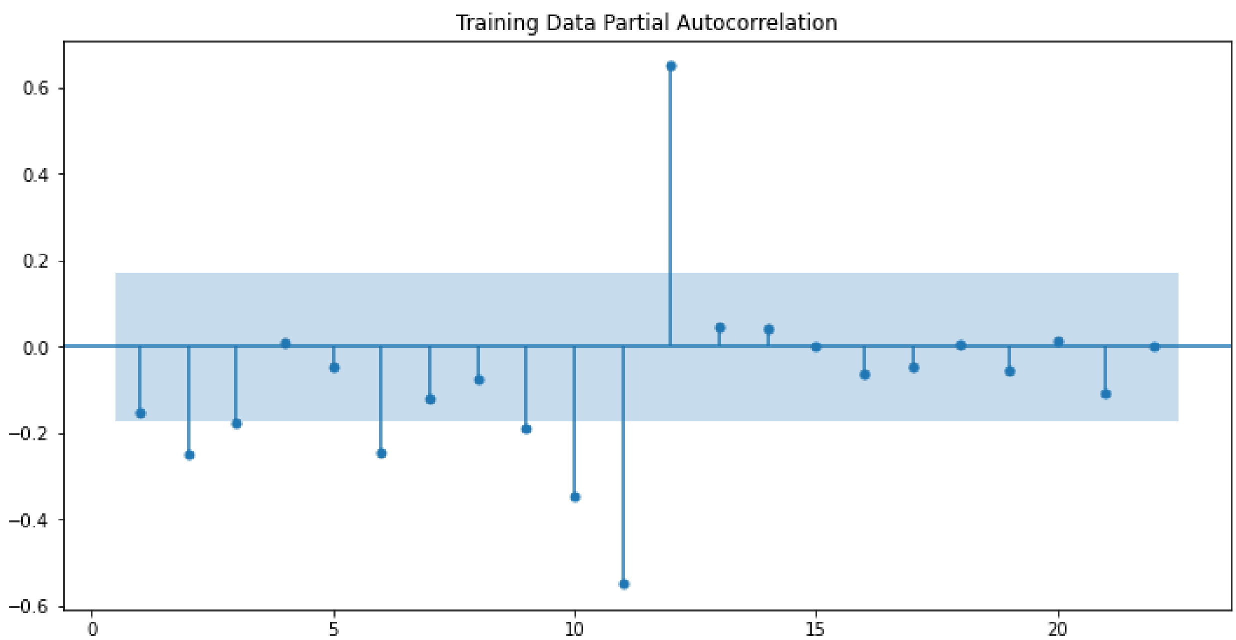


- Finding auto-correlation function and partial auto-correlation function on training data only:
    - Shape of Training Data is (132, 1)
    - Shape of Test Data is (55, 1)

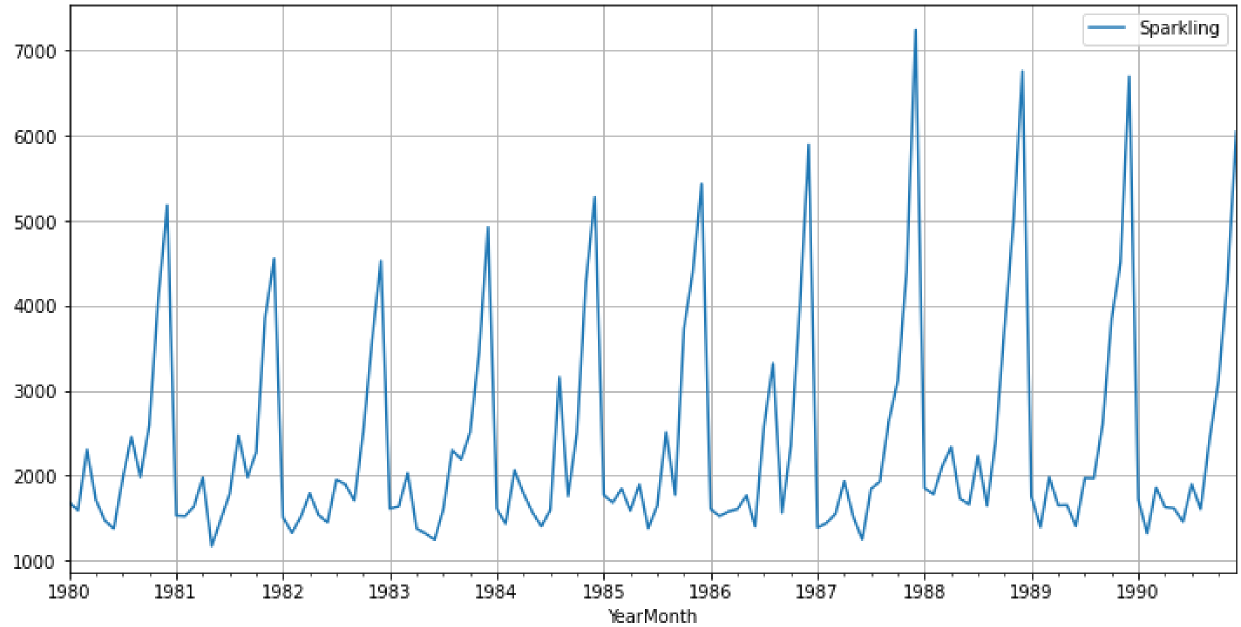
- Training data autocorrelation:



- Training data partial autocorrelation:



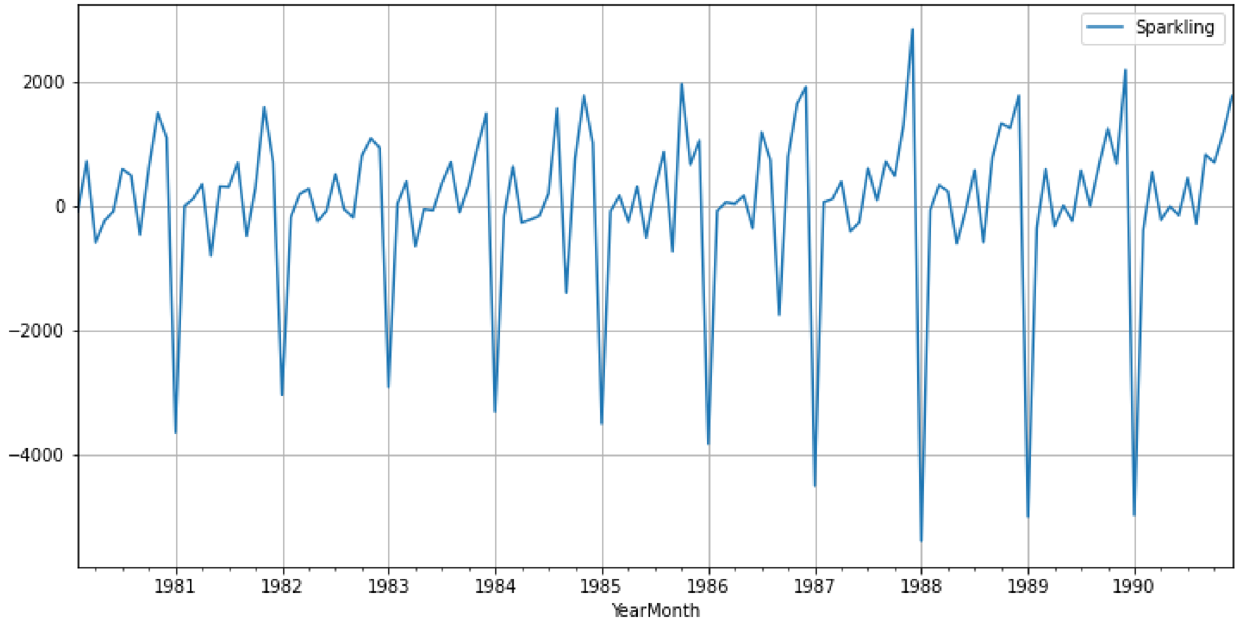
- Visualization of the training data:



- The Augmented Dickey-Fuller test results:
    - *DF test statistic is -2.062*
    - *DF test p-value is 0.56741103885936866*
    - *Number of lags used 12*

- The Augmented Dickey-Fuller test results by removing unnecessary data:
    - *DF test statistic is -7.968*
    - *DF test p-value is 8.479210655514366e-11*
    - *Number of lags used 11*

- Plotting graph for the same:



Results for SARIMA model:

- SARIMA(0, 1, 0)x(0, 0, 0, 12) - AIC:2251.3597196862966

	param	seasonal	AIC
0	(0, 1, 0)	(0, 0, 0, 12)	2251.35972

- The SARIMA Model:

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                            Sparkling   No. Observations:                  132
Model:             SARIMAX(0, 1, 1)x(1, 0, 1, 12)   Log Likelihood                 -865.045
Date:                              Sun, 20 Jun 2021   AIC                           1738.090
Time:                                      19:03:08   BIC                           1749.139
Sample:                                  01-01-1980   HQIC                          1742.576
                                       - 12-01-1990
Covariance Type:                                opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -1.1498      0.071    -16.097      0.000      -1.290      -1.010
ar.S.L12       1.0408      0.011     94.240      0.000       1.019       1.062
ma.S.L12      -0.6549      0.080     -8.141      0.000      -0.813      -0.497
sigma2      1.128e+05   1.58e+04      7.116      0.000    8.17e+04    1.44e+05
===================================================================================
Ljung-Box (L1) (Q):                   0.47   Jarque-Bera (JB):                30.07
Prob(Q):                              0.49   Prob(JB):                         0.00
Heteroskedasticity (H):               2.82   Skew:                             0.43
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.33
===================================================================================

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

- Predicted auto SARIMA summary:

Sparkling	mean	mean_se	mean_ci_lower	mean_ci_upper
1991-01-01	1417.964748	386.121138	661.181224	2174.748272
1991-02-01	1145.080417	389.378651	381.912284	1908.248549
1991-03-01	1615.907957	392.614481	846.397714	2385.418199
1991-04-01	1493.135022	395.824092	717.334057	2268.935987
1991-05-01	1346.984336	399.007903	564.943216	2129.025457

- RMSE obtained is: 603.6494071113511
- Results Obtained:

	RMSE
ARIMA(2,1,1)	1374.546024
SARIMA(2,1,3)(0,0,3,12)(SARIMA AIC)	713.977694
SARIMA(0, 1, 1)(1, 0, 1, 12)	603.649407

Question 8: Build a table (create a data frame) with all the models built along with their corresponding parameters and the respective RMSE values on the test data.

Solution:

```
temp_resultsDf = pd.DataFrame({'RMSE': [rmse]}
                           ,index=['SARIMA(0,1,1)(0,0,3,12)'])


resultsDf = pd.concat([resultsDf,temp_resultsDf])

resultsDf
```

	RMSE
ARIMA(2,1,1)	1374.546024
SARIMA(2,1,3)(0,0,3,12)(SARIMA AIC)	713.977694
SARIMA(0, 1, 1)(1, 0, 1, 12)	603.649407
SARIMA(0,1,1)(0,0,3,12)	1099.387954

Question 9: Based on the model-building exercise, build the most optimum model(s) on the complete data and predict 12 months into the future with appropriate confidence intervals/bands.

Solution:

Please refer to the Jupyter Notebook submitted to look into the code.

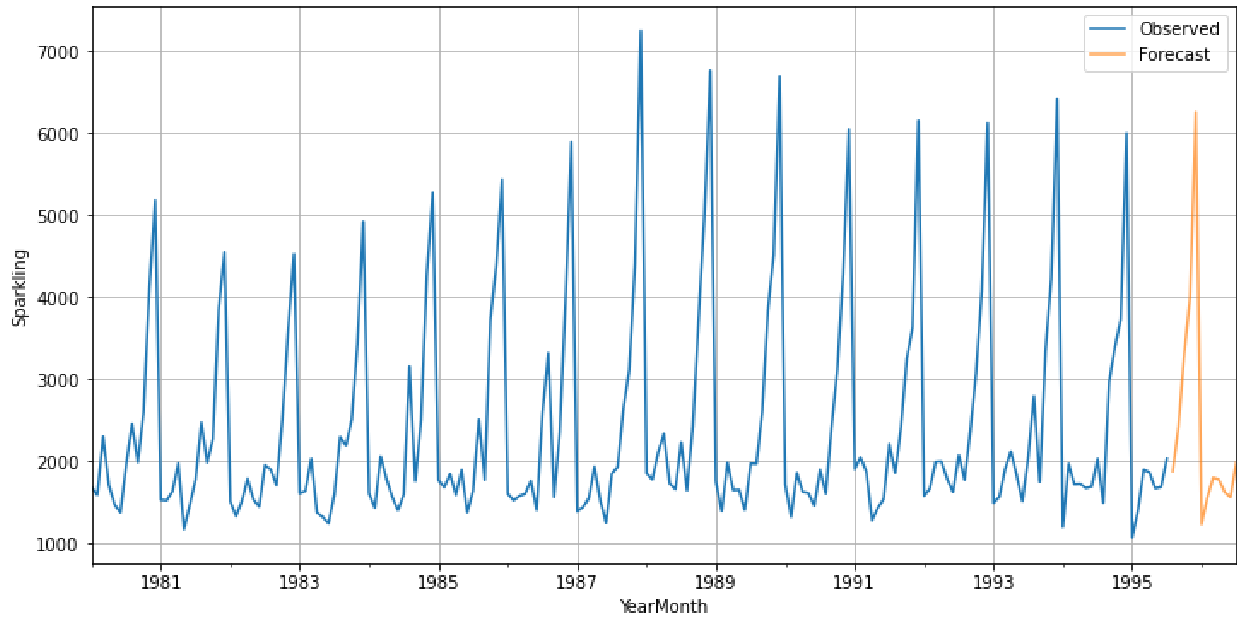
```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                             Sparkling   No. Observations:               187
Model:             SARIMAX(0, 1, 1)x(1, 0, 1, 12)   Log Likelihood               -1266.941
Date:                            Sun, 20 Jun 2021   AIC                           2541.882
Time:                                    19:03:09   BIC                           2554.472
Sample:                                01-01-1980   HQIC                          2546.990
                                     - 07-01-1995
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -1.0788      0.038    -28.471      0.000      -1.153      -1.005
ar.S.L12       1.0121      0.011     96.365      0.000       0.992       1.033
ma.S.L12      -0.6163      0.065     -9.508      0.000      -0.743      -0.489
sigma2       1.22e+05   1.21e+04     10.114      0.000    9.84e+04    1.46e+05
===================================================================================
Ljung-Box (L1) (Q):                   1.44   Jarque-Bera (JB):                49.08
Prob(Q):                              0.23   Prob(JB):                         0.00
Heteroskedasticity (H):               1.22   Skew:                             0.57
Prob(H) (two-sided):                  0.45   Kurtosis:                         5.36
===================================================================================

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

- **Forecasting for upcoming one year:**

Sparkling	mean	mean_se	mean_ci_lower	mean_ci_upper
1995-08-01	1877.254167	376.878119	1138.586628	2615.921706
1995-09-01	2418.741793	377.882708	1678.105296	3159.378291
1995-10-01	3296.070789	378.884633	2553.470554	4038.671023
1995-11-01	3998.078150	379.883915	3253.519358	4742.636942
1995-12-01	6250.663528	380.880577	5504.151315	6997.175742

- RMSE of the Full Model: 532.1702158997205

- Upcoming year forecasting graph:



- **Insights:**
  - The forecasting prediction is that there will be an increase in sales in 1996 than 1995 in Sparkling Wine.
  - The growth will be more additive and it will show seasonality.

Question 10: Comment on the model thus built and report your findings and suggest the measures that the company should be taking for future sales. Please explain and summarise the various steps performed in this project. There should be proper business interpretation and actionable insights present.

Solution:

This Sparkling data contains the 15 years of data of sales. Once I dug the data it seems to have a strong seasonality in the last year every year. It shows in festive times the demand for wine increases all around. This increasing demand is specific and needs specific care to match the increased demand.

Once predicting the future demand for the next 12 months, it is observed that demand will follow the same trend and will show a strong seasonality at the end of the year. Once the sales data is decomposed it shows the trend of demand as static over the years. The demand has shown a slight increase and a specific increase between Y1987 and Y1990

Various exponential smoothing models development of this project contains the deep analysis of data:

- First data is converted into Time series index, and then EDA has provided the complete insight of data along with Decomposition of data on both additive and multiplicative basis. Additive analysis shows some trends in error terms so multiplicative decomposition is required here.
- Then different techniques of various exponential smoothing models are done training data and its effect is observed on the test data. The RMSE value of the different models is observed, and each model's RMSE value is enclosed here for better understanding. Double Exponential Smoothing follows the test data most accurately comparing the other models.

Test	RMSE
Regression On Time	
NaiveModel	3.864279e+03
Simple Average Model	1.275082e+03
Alpha = 0.995, SimpleExponentialSmoothing	1.316035e+03
Alpha=0.3, Beta=0.3, Double Exponential Smoothing	2.252068e+06

- Stationarity shows the statistical properties of a time series that do not change over a period of time. Here data was not stationary and the first order of differentiation was required to make it stationary. The augmented Dickey-Fuller test at alpha level =0.05 is used to develop the above model.
- ARIMA and SARIMA models are then developed to generate the future prediction for the next 12 months. ARIMA and SARIMA both are developed using AIC first then using ACF and PACF.

Test	RMSE
ARIMA (2,1,1) (AIC)	1374.037009
SARIMA (2,1,3) (0,0,3,12) (SARIMA AIC)	652.983976
ARIMA (1,1,0) (ACF & PACF)	1418.218341
SARIMA(0, 1, 1) (1, 0, 1, 12), (AIC & PACF)	603.648764

The Overall RMSE of the model when implemented on whole data (Training+Test) = RMSE of the Full Model is 532.170401378045

Action Insights for the company:

- New customer segment identification:
  - Looking at the result and future forecasting, we've seen that there's a constant and profitable demand for Sparkling Wine, hence we have a strong customer base. Now it's needed to identify new consumers to expand the growth. For this the suggestions would be:
    - Referral Program:
      - Bring your buddy program with additional discount to introduce new consumers
- Brand Marketing and Advertising:
  - This is the perfect time when there should be work started on brand enablement and advertising in new geography to acquire new customers.