Time Series Forecasting Business Report

Rohan R. Khade

Table of Contents

Back to Top

List of Tables

List of Figures

Back to Top

Back to Top

Back to Top

Back to Top

Back to Top

Chapter 1. Problem

1.1 Problem Statement

For this particular assignment, the data of different types of wine sales in the 20th century is to be analyzed. 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 analyze and forecast Wine Sales in the 20th century.

1.1.1 Introduction

We have the revenue sales for the two types of wines produced by the ABC Estate Wines, namely Rose and Sparkling. The data has been provided from January 1980 to June 1995. In the below report we have represented the sales in different types of charts and built models to forecast the sales for next 12 months. We have separately analyzed the sales data for the two types of wines and provided separate as well as combined analyst perspective for the same.

In the below chapters, we have separately analyzed the sales data for the two types of wines and provided separate as well as combined analyst perspective for the same.

Back to Top

Chapter 2. Rose

We have created two dataframes rose1 (refer the jupyter notebook) and rose2 to read the data differently. However, as we have used rose2 dataframe for further analysis, we will refer to the same dataframe for below analysis.

2.1 Read the data as an appropriate Time Series data and plot the data.

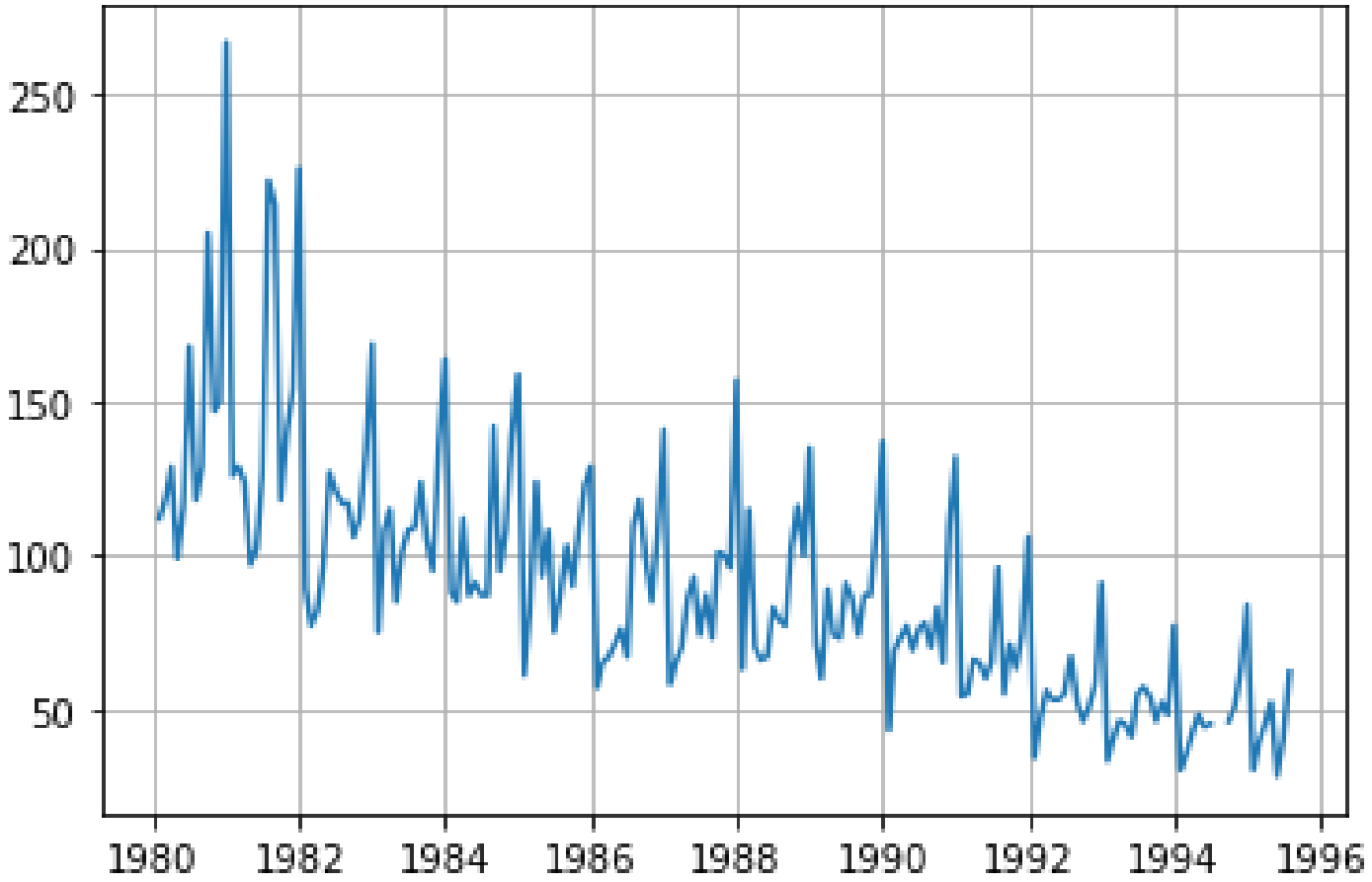
Table 1 Dataframe: rose2 (with head function)

	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

Table 2 Dataframe: rose2 (with describe function)

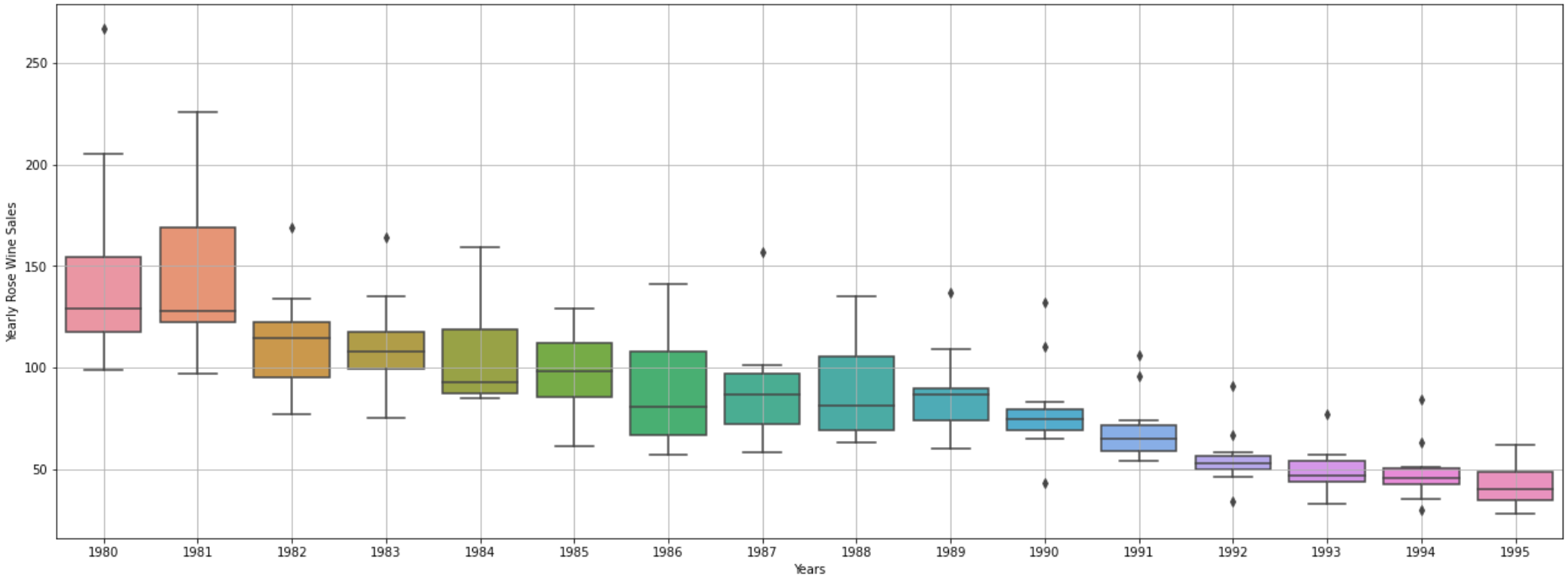
	Rose
count	185.000000
mean	90.394595
std	39.175344
min	28.000000
25%	63.000000
50%	86.000000
75%	112.000000
max	267.000000

Figure 1. **Dataframe rose2: Plot**



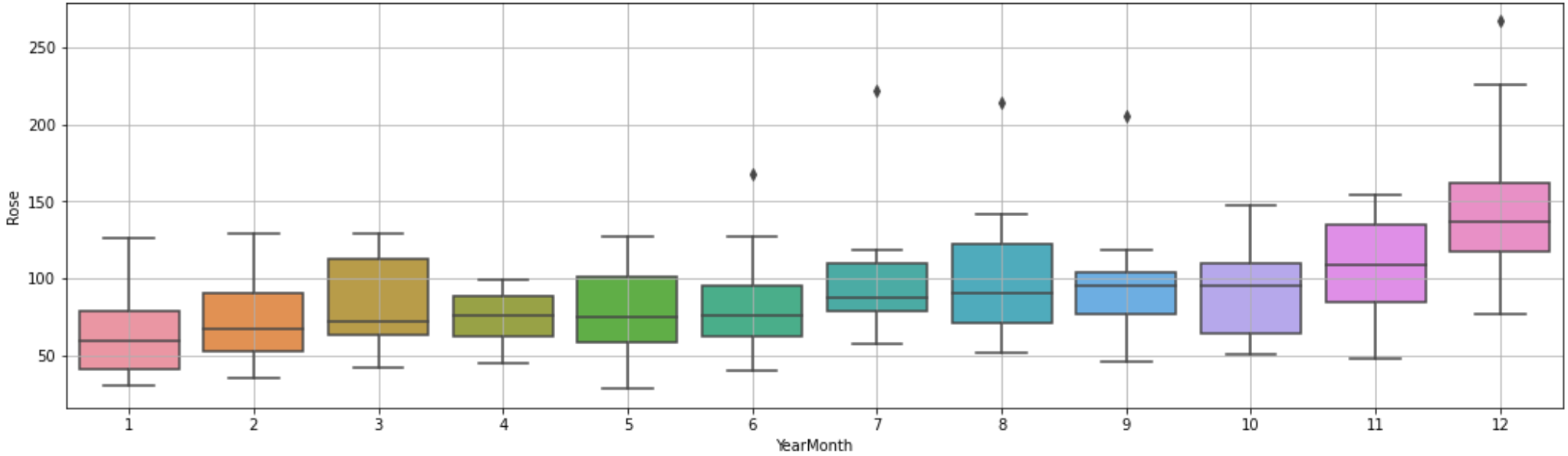
2.2 Perform appropriate Exploratory Data Analysis to understand the data and also perform decomposition.

Figure 2. Yearly Rose Wine Sales



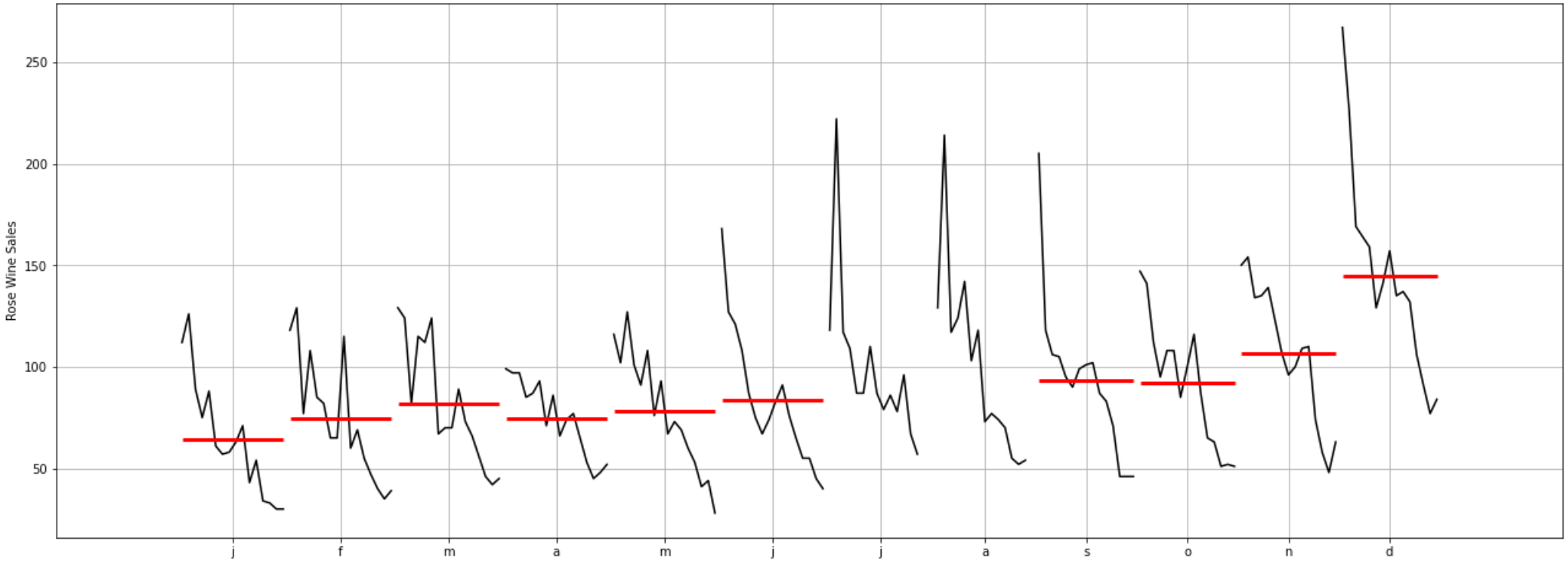
Back to Top

Figure 3. Monthly Rose Wine Sales



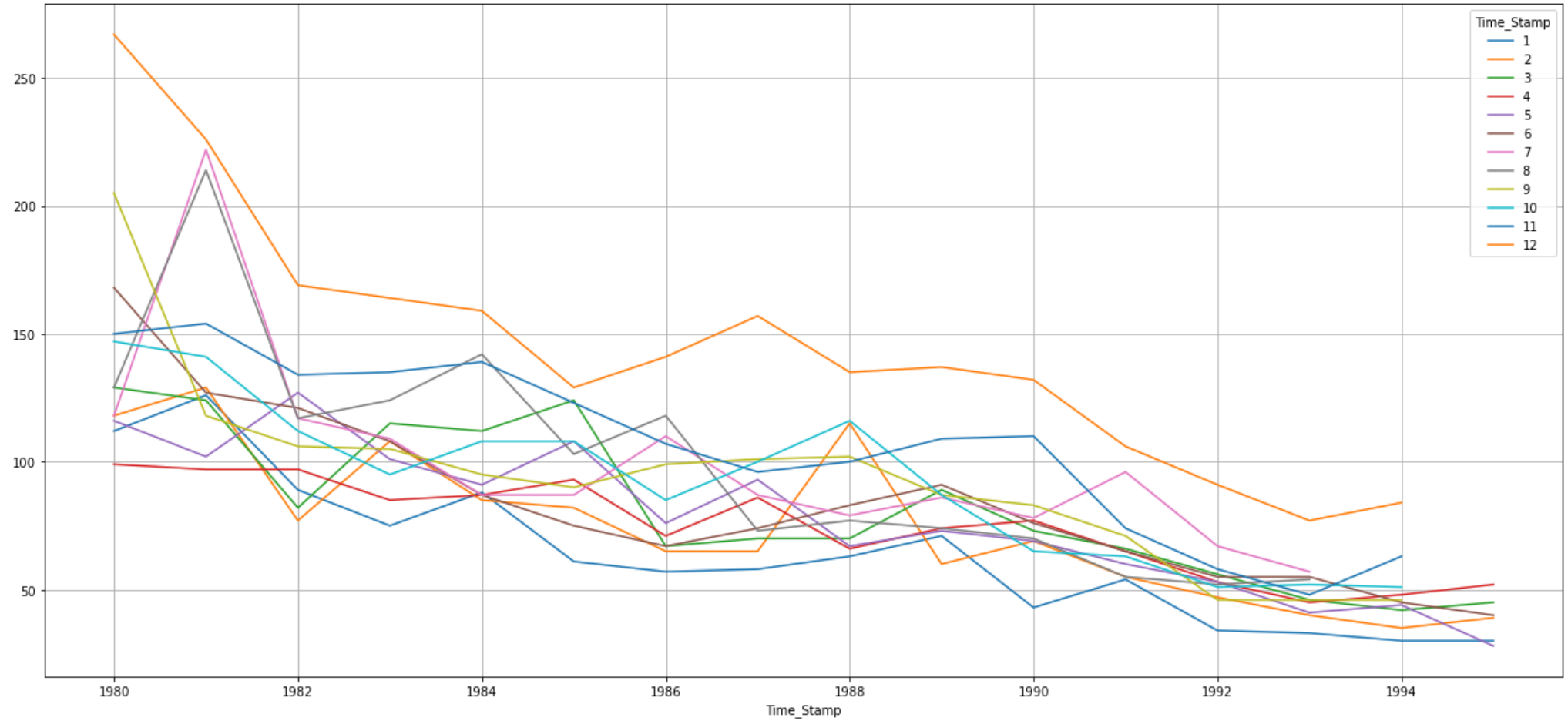
Back to Top

Figure 4. Rose Wine Sales



Back to Top

Figure 5. Monthly Sales: Line Graph



Back to Top

Table 3 Monthly Sales, January 1980 - June 1995

Time_Stamp	1	2	3	4	5	6	7	8	9	10	11	12
Time_Stamp												
1980	112.0	118.0	129.0	99.0	116.0	168.0	118.0	129.0	205.0	147.0	150.0	267.0
1981	126.0	129.0	124.0	97.0	102.0	127.0	222.0	214.0	118.0	141.0	154.0	226.0
1982	89.0	77.0	82.0	97.0	127.0	121.0	117.0	117.0	106.0	112.0	134.0	169.0
1983	75.0	108.0	115.0	85.0	101.0	108.0	109.0	124.0	105.0	95.0	135.0	164.0
1984	88.0	85.0	112.0	87.0	91.0	87.0	87.0	142.0	95.0	108.0	139.0	159.0
1985	61.0	82.0	124.0	93.0	108.0	75.0	87.0	103.0	90.0	108.0	123.0	129.0
1986	57.0	65.0	67.0	71.0	76.0	67.0	110.0	118.0	99.0	85.0	107.0	141.0
1987	58.0	65.0	70.0	86.0	93.0	74.0	87.0	73.0	101.0	100.0	96.0	157.0
1988	63.0	115.0	70.0	66.0	67.0	83.0	79.0	77.0	102.0	116.0	100.0	135.0
1989	71.0	60.0	89.0	74.0	73.0	91.0	86.0	74.0	87.0	87.0	109.0	137.0
1990	43.0	69.0	73.0	77.0	69.0	76.0	78.0	70.0	83.0	65.0	110.0	132.0
1991	54.0	55.0	66.0	65.0	60.0	65.0	96.0	55.0	71.0	63.0	74.0	106.0
1992	34.0	47.0	56.0	53.0	53.0	55.0	67.0	52.0	46.0	51.0	58.0	91.0
1993	33.0	40.0	46.0	45.0	41.0	55.0	57.0	54.0	46.0	52.0	48.0	77.0
1994	30.0	35.0	42.0	48.0	44.0	45.0	NaN	NaN	46.0	51.0	63.0	84.0
1995	30.0	39.0	45.0	52.0	28.0	40.0	62.0	NaN	NaN	NaN	NaN	NaN

Figure 6. Yearly Sum Plot

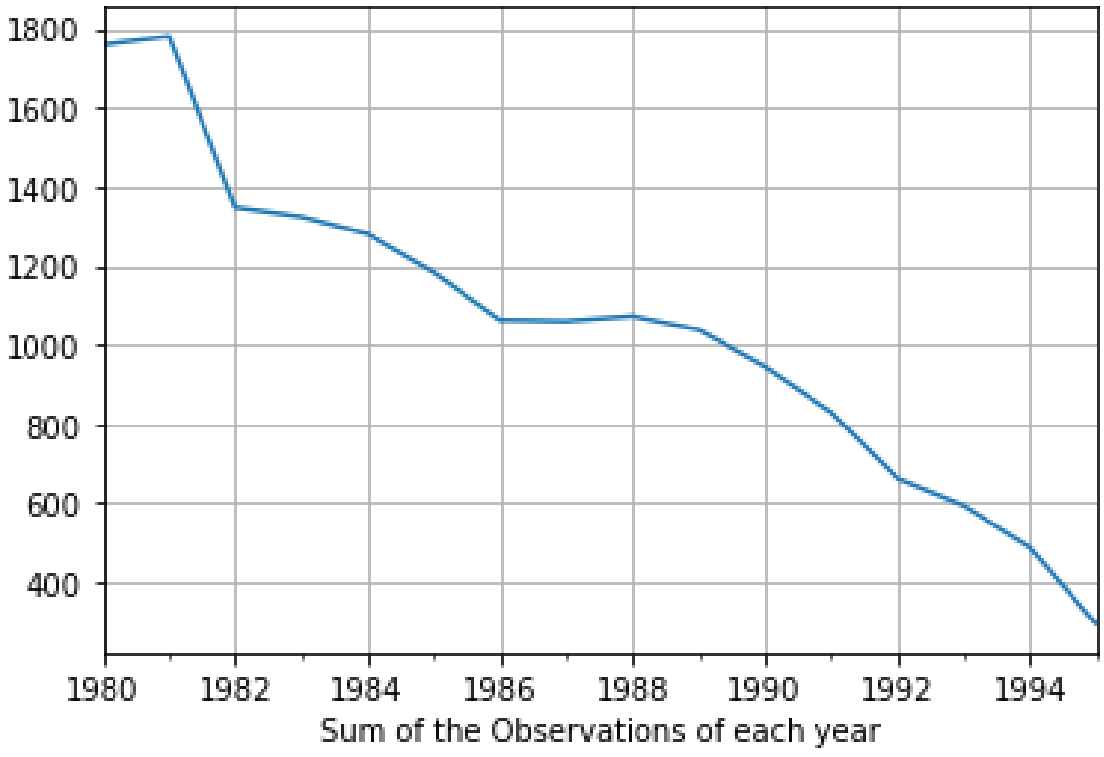


Figure 7. **Yearly Mean Plot**

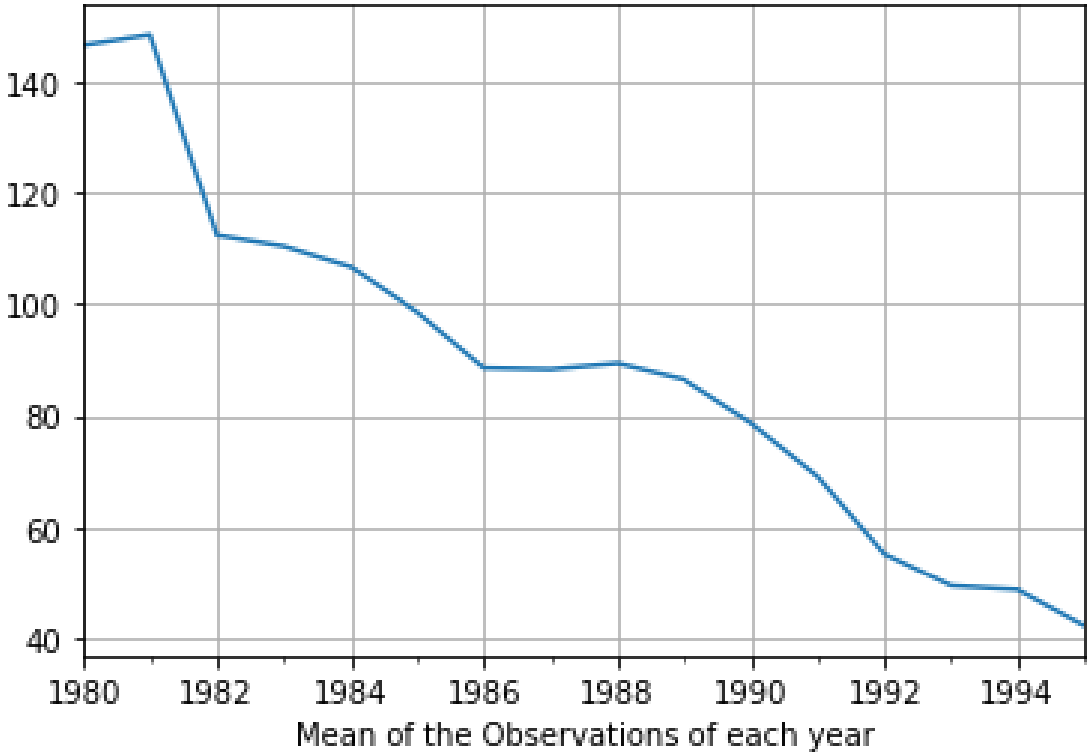
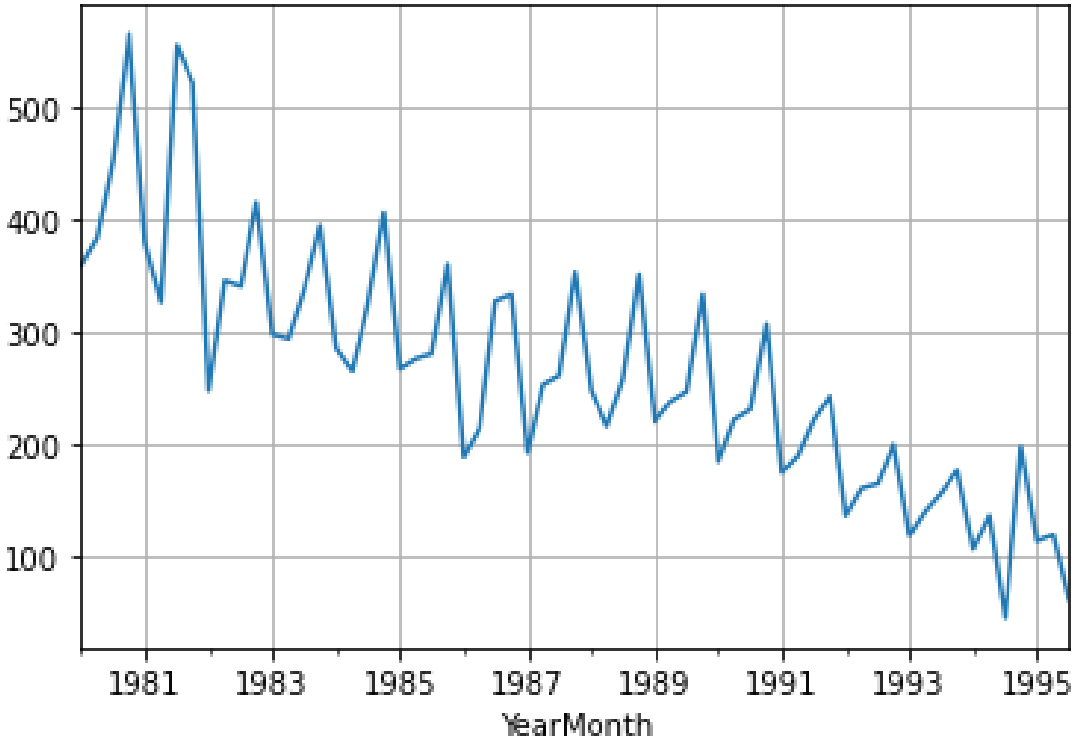


Figure 8. **Quarterly Sum Plot**



Back to Top

Figure 9. Quarterly Mean Plot

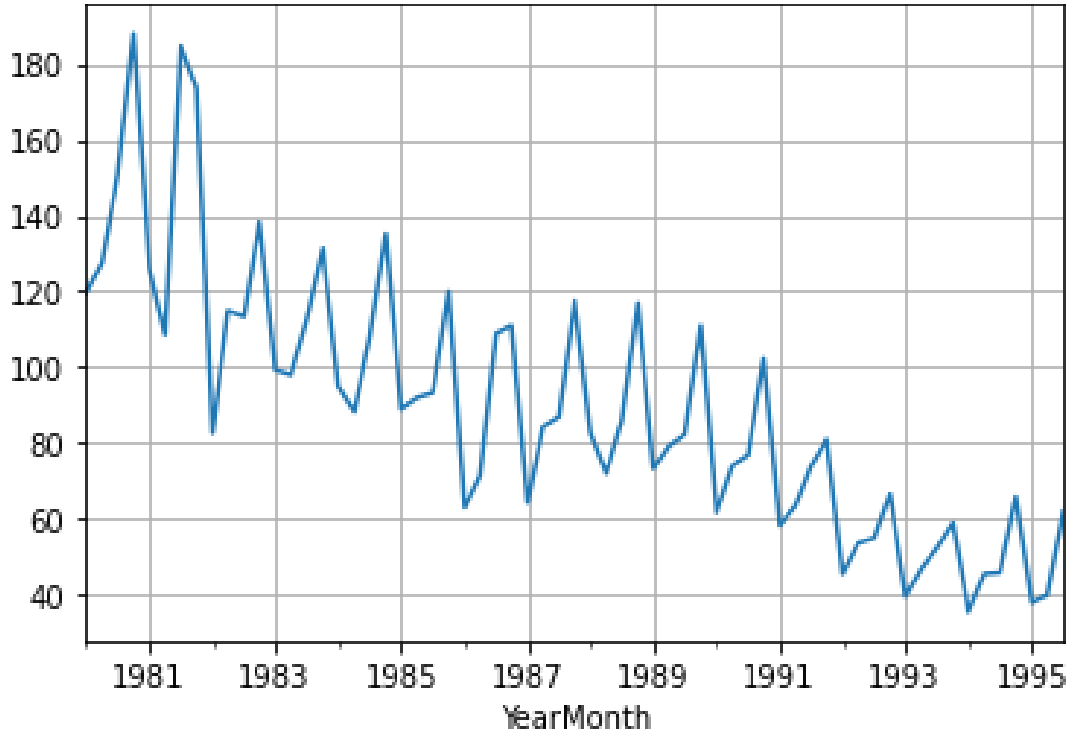
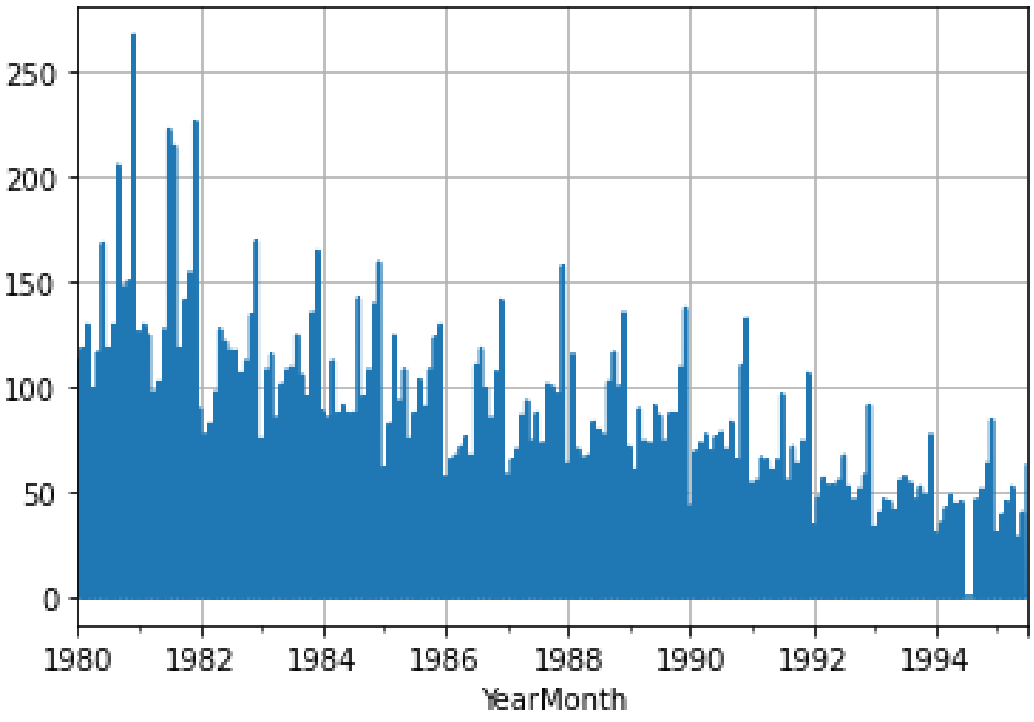
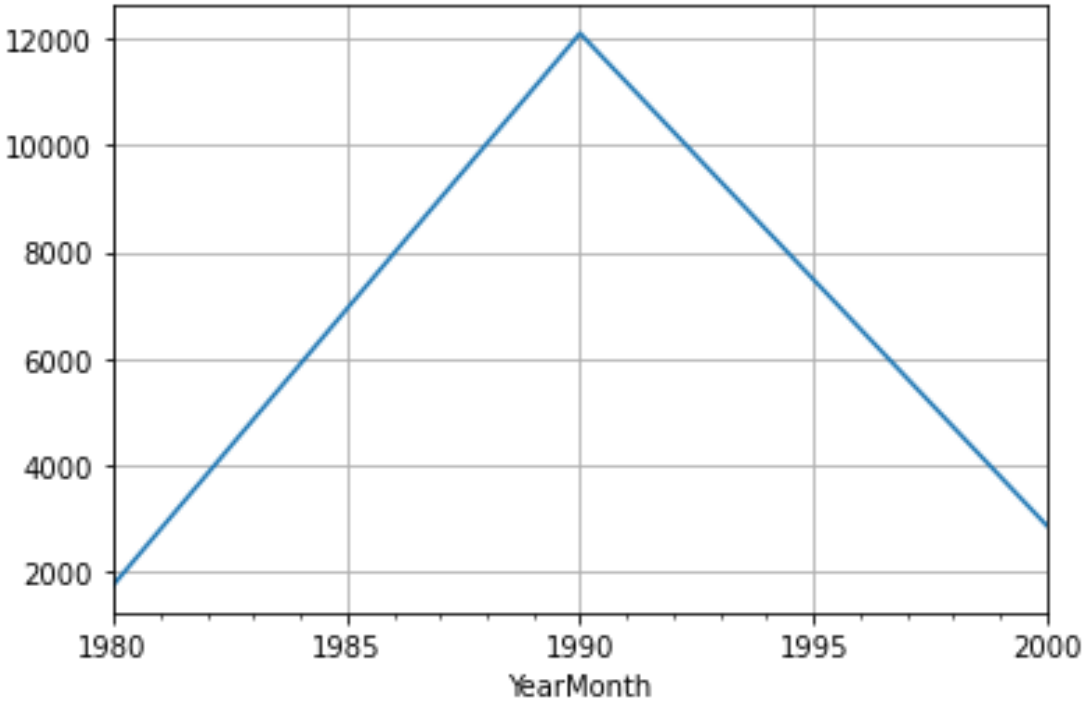


Figure 10. Daily Sum Plot



Back to Top

Figure 11. Decade Sum Plot



Back to Top

Figure 12. Data Decomposition

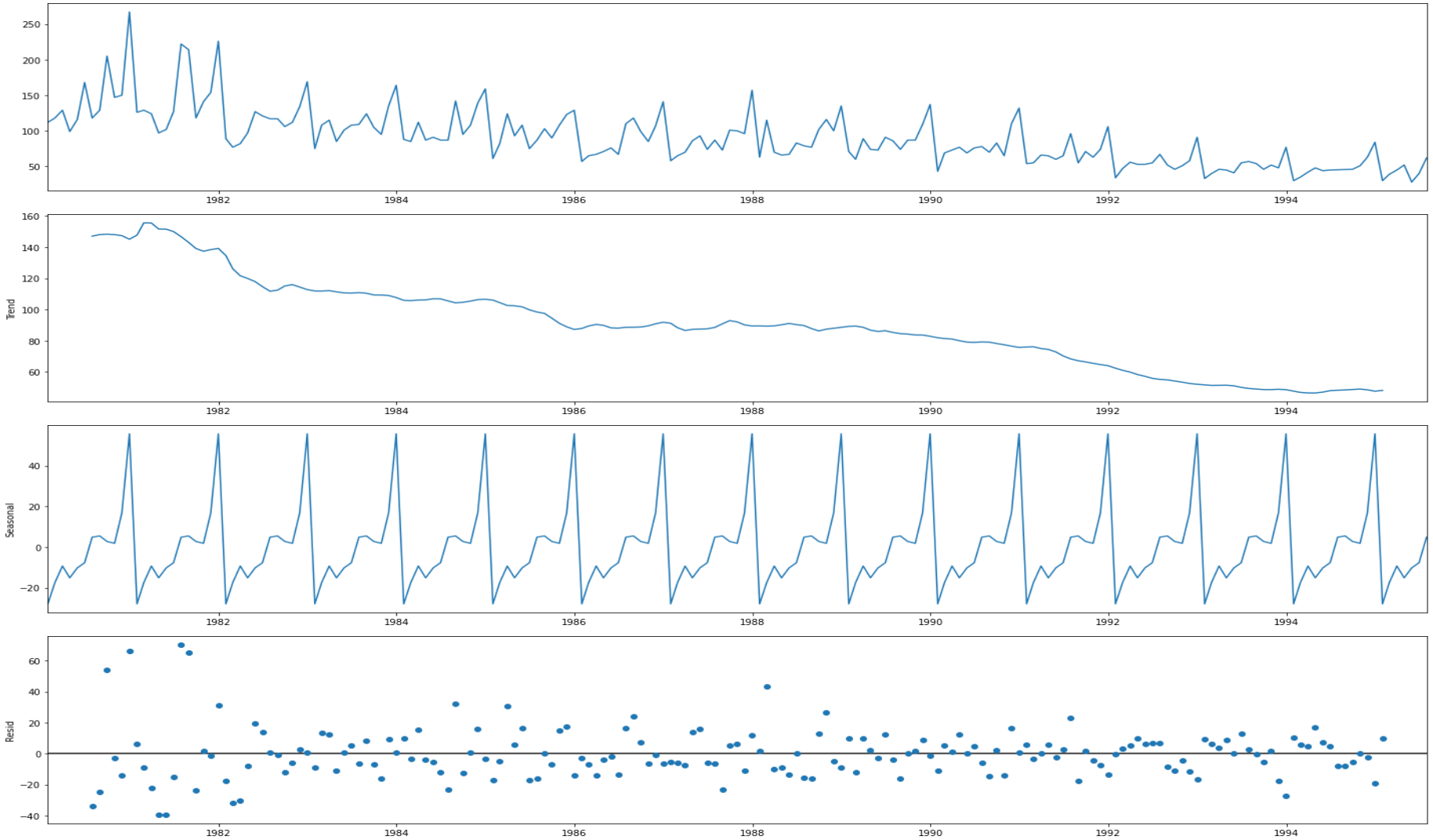
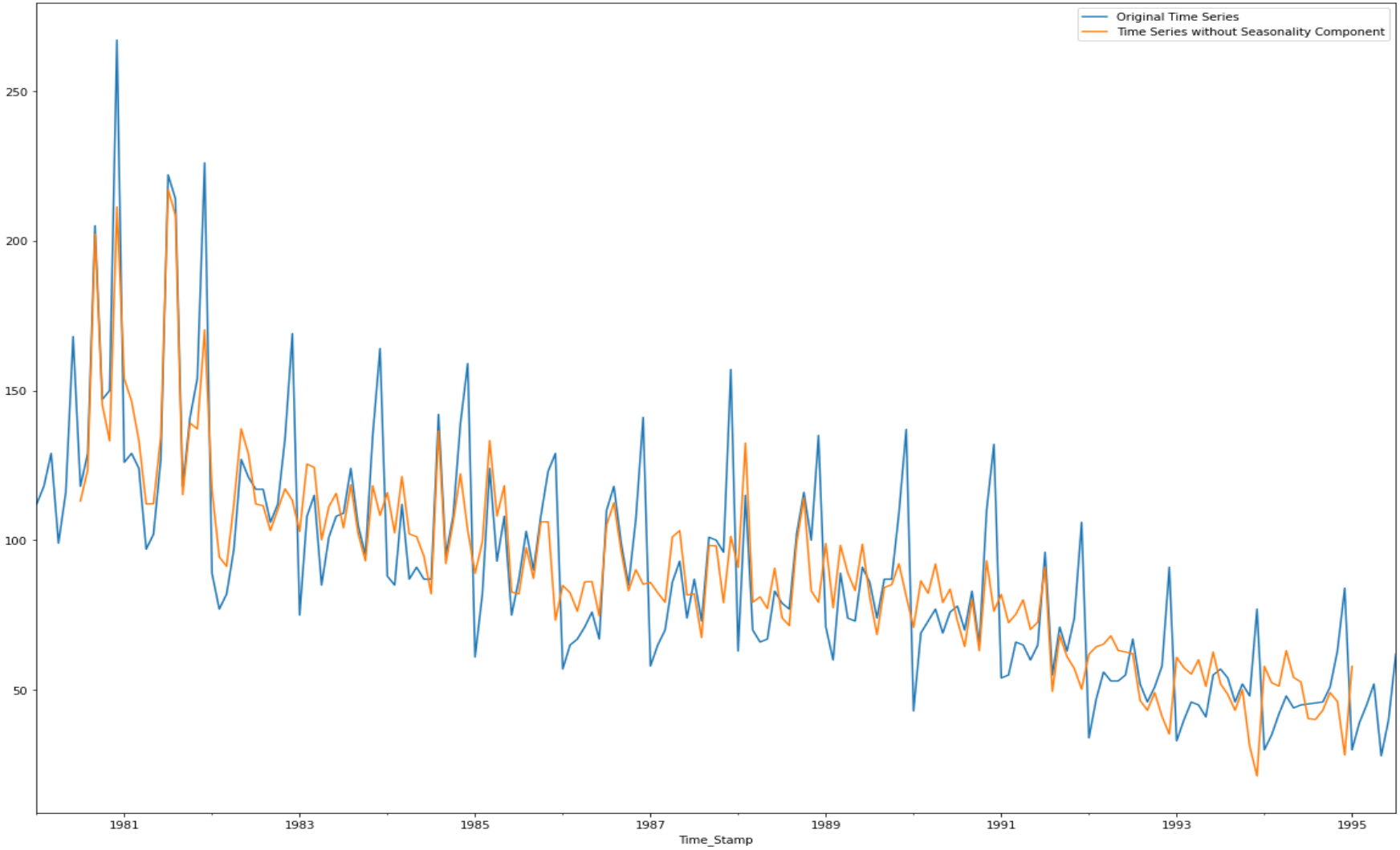


Figure 13. Original Time Series vs. Time Series without Seasonality Component



Back to Top

Table 4 Trend Table (with head function)

```
Trend
 Time_Stamp
1980-01-31           NaN
1980-02-29           NaN
1980-03-31           NaN
1980-04-30           NaN
1980-05-31           NaN
1980-06-30           NaN
1980-07-31    147.083333
1980-08-31    148.125000
1980-09-30    148.375000
1980-10-31    148.083333
1980-11-30    147.416667
1980-12-31    145.125000
Name: trend, dtype: float64
```

Table 5 Seasonality Table (with head function)

```
Seasonality
 Time_Stamp
1980-01-31   -27.908647
1980-02-29   -17.435632
1980-03-31    -9.285830
1980-04-30   -15.098330
1980-05-31   -10.196544
1980-06-30    -7.678687
1980-07-31     4.896908
1980-08-31     5.499686
1980-09-30     2.774686
1980-10-31     1.871908
1980-11-30    16.846908
1980-12-31    55.713575
Name: seasonal, dtype: float64
```

Table 6 Residual Table (with head function)

```
Residual
 Time_Stamp
1980-01-31          NaN
1980-02-29          NaN
1980-03-31          NaN
1980-04-30          NaN
1980-05-31          NaN
1980-06-30          NaN
1980-07-31   -33.980241
1980-08-31   -24.624686
1980-09-30    53.850314
1980-10-31    -2.955241
1980-11-30   -14.263575
1980-12-31    66.161425
Name: resid, dtype: float64
```

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

We have split the data from 1980 to 1990 into the train dataset and 1991 to June 1995 into the test dataset.

Figure 14. Train and Test Dataset (Rows and Column)

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

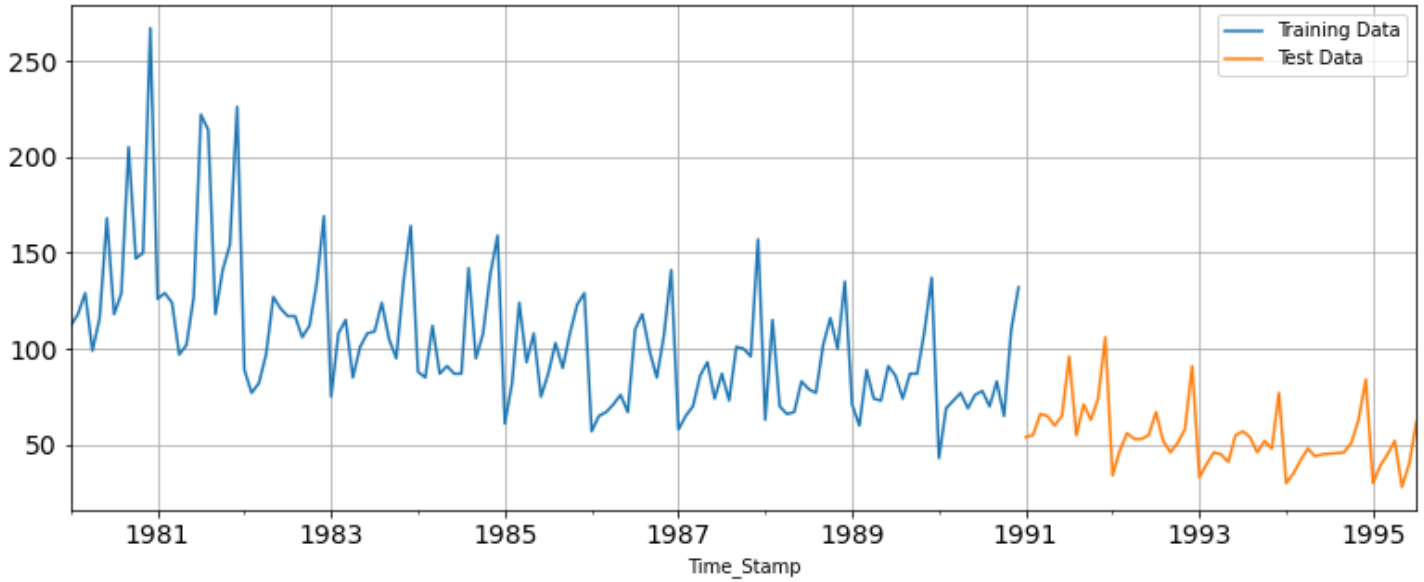
Figure 15. Train Dataset (with head and tail function)

```
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
```

Figure 16. Train Dataset (with head and tail function)

```
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
```

Figure 17. Train & Test Dataset: Plot



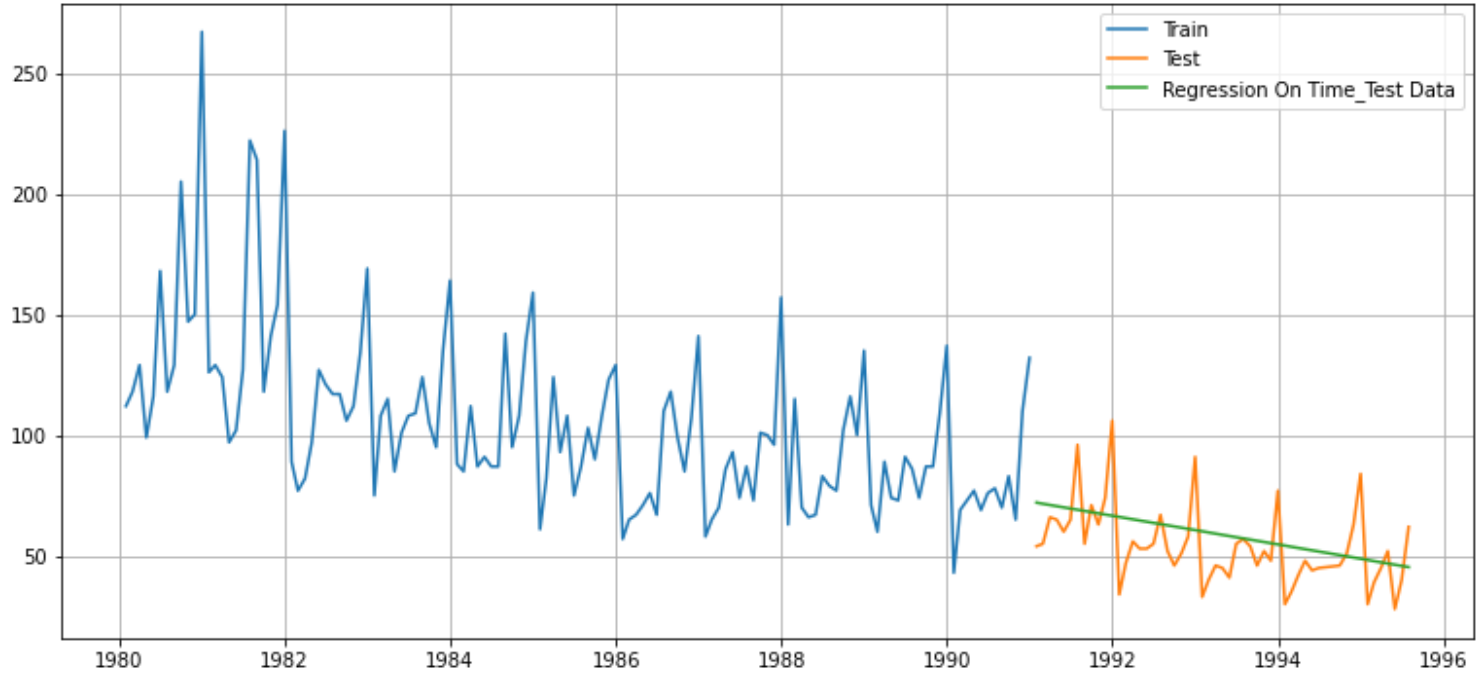
2.4 Build all the exponential smoothing models on the training data and evaluate the model using RMSE on the test data. Other additional models such as regression, naïve forecast models, simple average models, moving average models should also be built on the training data and check the performance on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

2.4.1 Linear Regression Model

Linear Regression: (Please refer to the Jupyter notebook)

- The LinearRegression () function is fit on the train dataset
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

Figure 18. Linear Regression Plot



We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as **15.269** and created a new dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

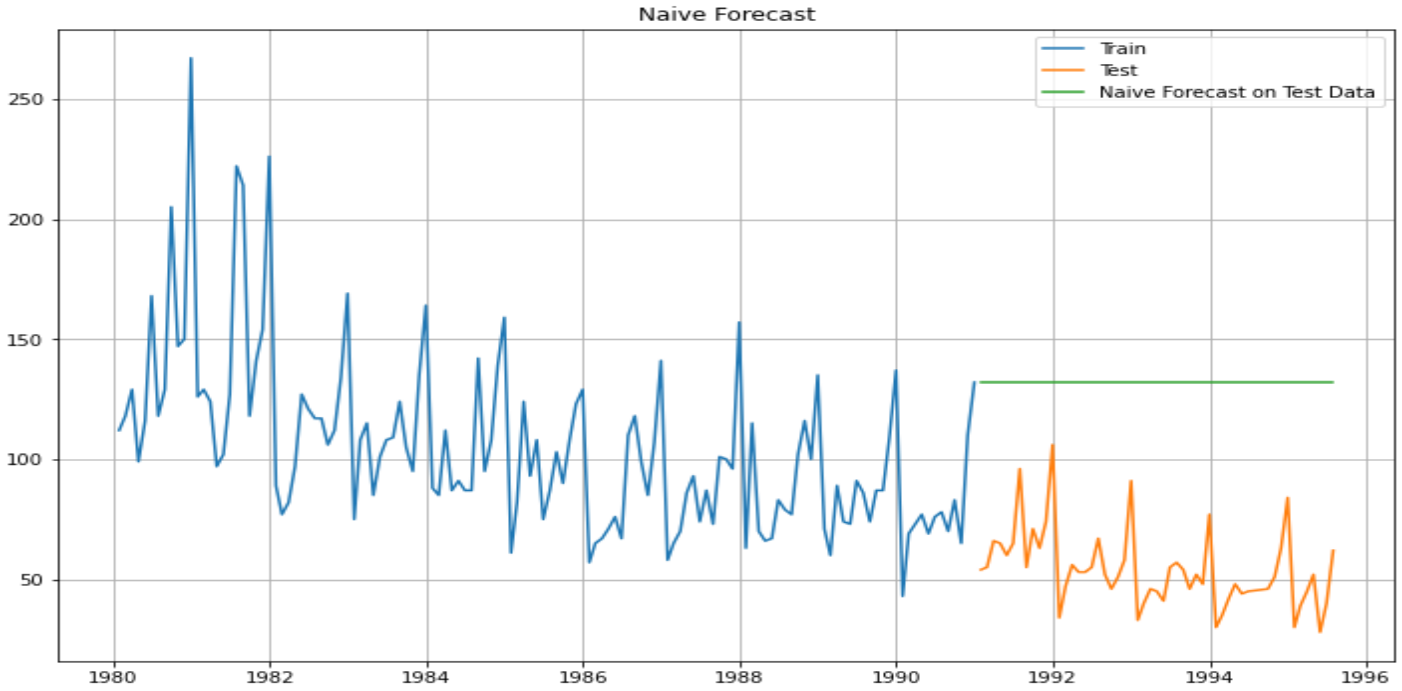
2.4.2 Naïve Model

Naïve: (Please refer to the Jupyter notebook)

- The naïve function is fit on the train dataset
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

Back to Top

Figure 19. Naïve Model Plot



We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as **79.719** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

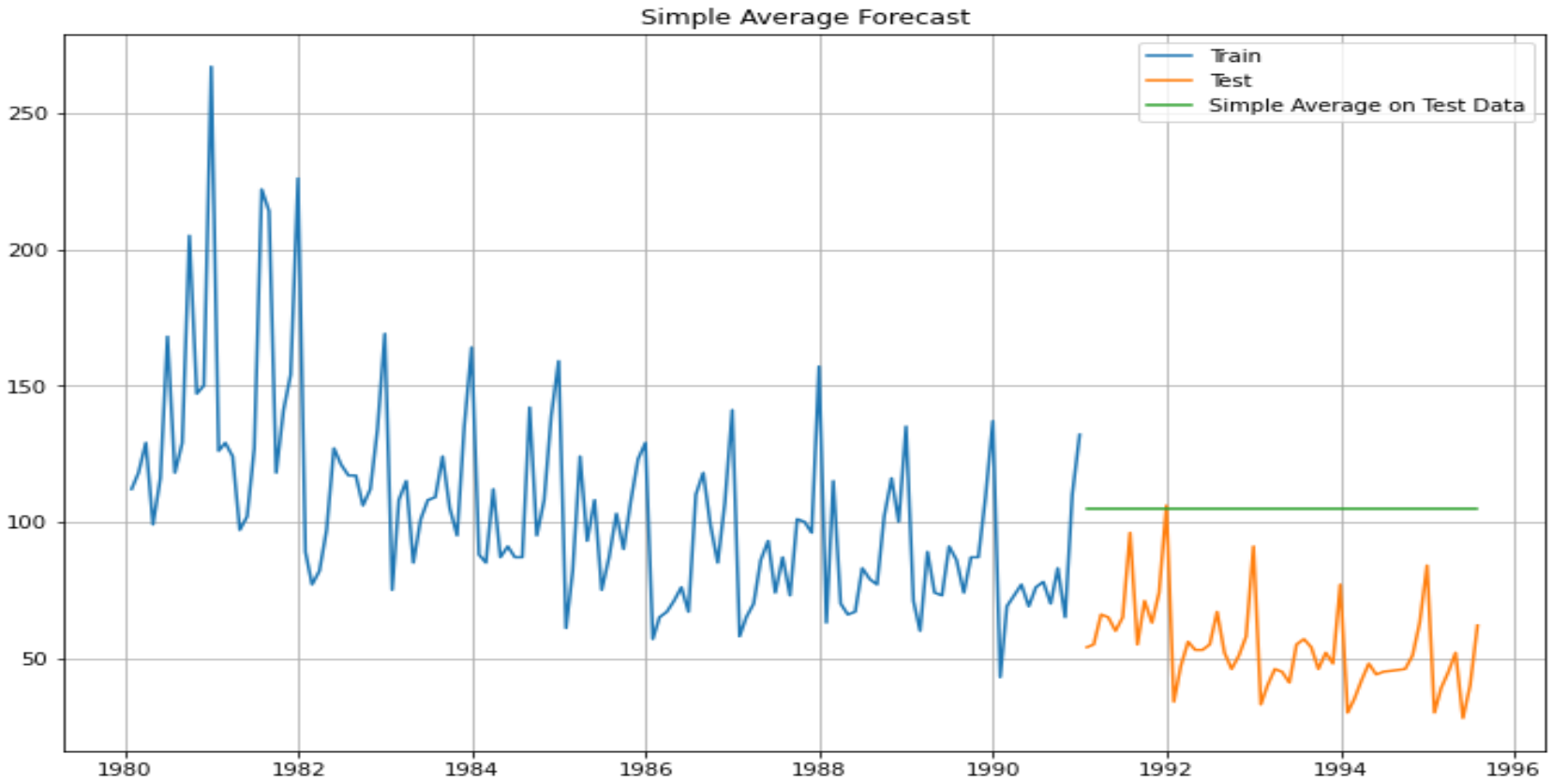
2.4.3 Simple Average Model

Simple Average Model: (Please refer to the Jupyter notebook)

- We take the simple arithmetic mean of the train dataset
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

Back to Top

Figure 20. Simple Average Model Plot



We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as **53.461** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Back to Top

2.4.4 Moving Average Model

Moving Average Model: (Please refer to the Jupyter notebook)

- We have taken moving average of the train dataset for 2, 4, 6, 9 points
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

Table 7 Moving Average for 2, 4, 6, 9 points (with head function)

Time_Stamp	Rose	Trailing_2	Trailing_4	Trailing_6	Trailing_9
1980-01-31	112.0	NaN	NaN	NaN	NaN
1980-02-29	118.0	115.0	NaN	NaN	NaN
1980-03-31	129.0	123.5	NaN	NaN	NaN
1980-04-30	99.0	114.0	114.5	NaN	NaN
1980-05-31	116.0	107.5	115.5	NaN	NaN

We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as for all the points 2, 4, 6, 9 of **11.529, 14.451, 14.566, and 14.728 respectively** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

We can see that the RMSE score for 2 point moving average is lowest owing to which we will consider as the most accurate predictor of sales for the test dataset time period as also seen in the below plots the accuracy of the prediction.

Figure 21. Moving Average Plot (Train Dataset)

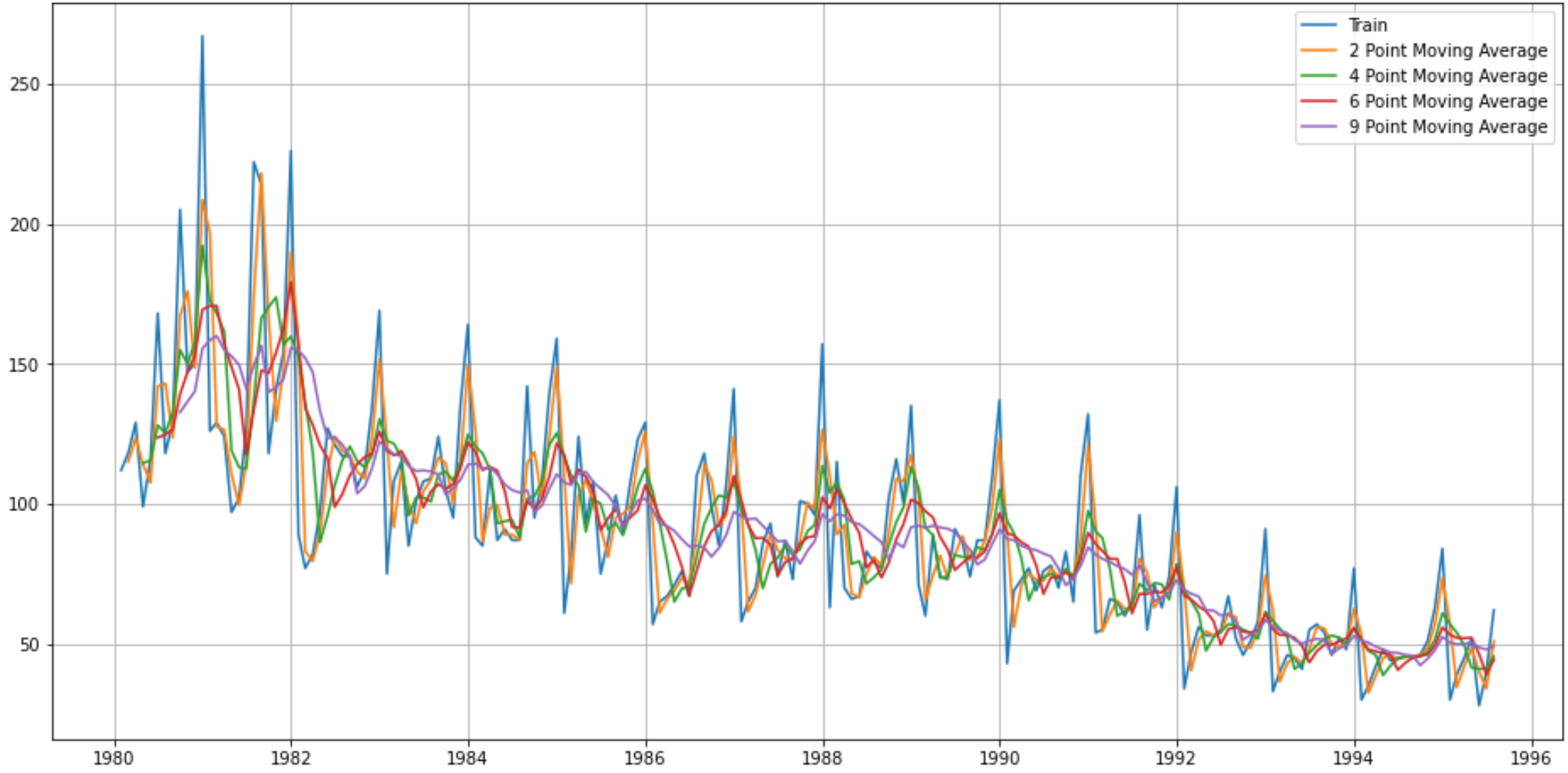
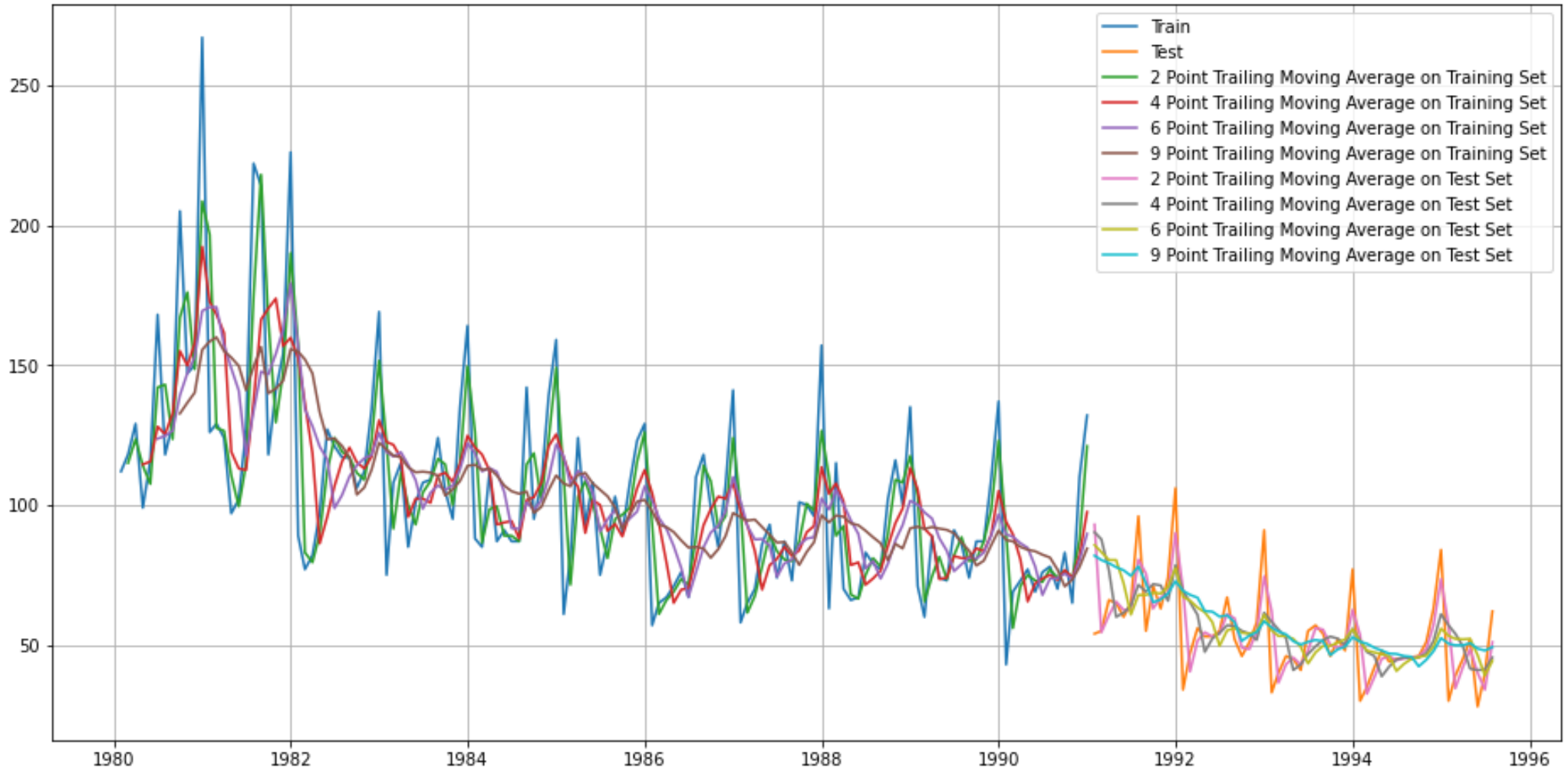
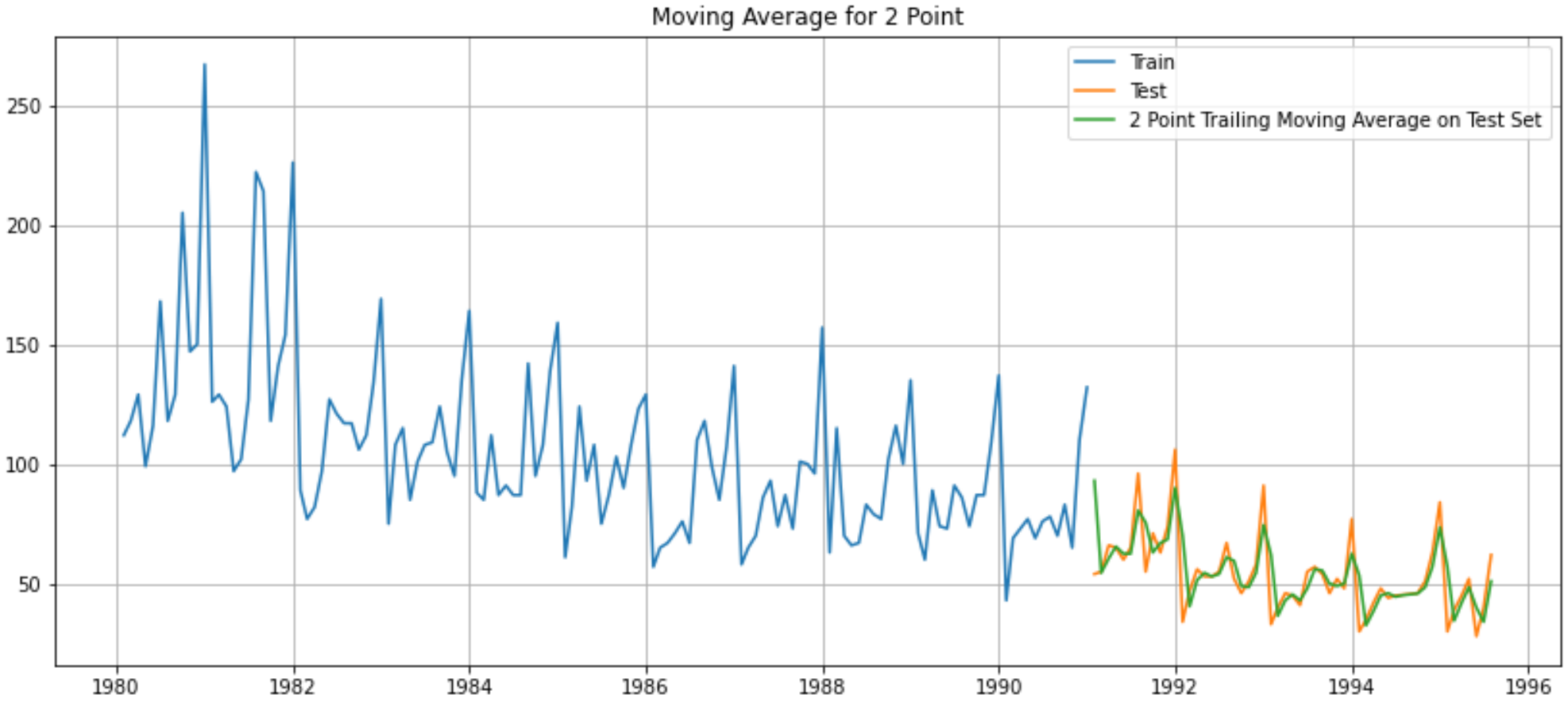


Figure 22. Moving Average Plot (Train & Test Dataset)



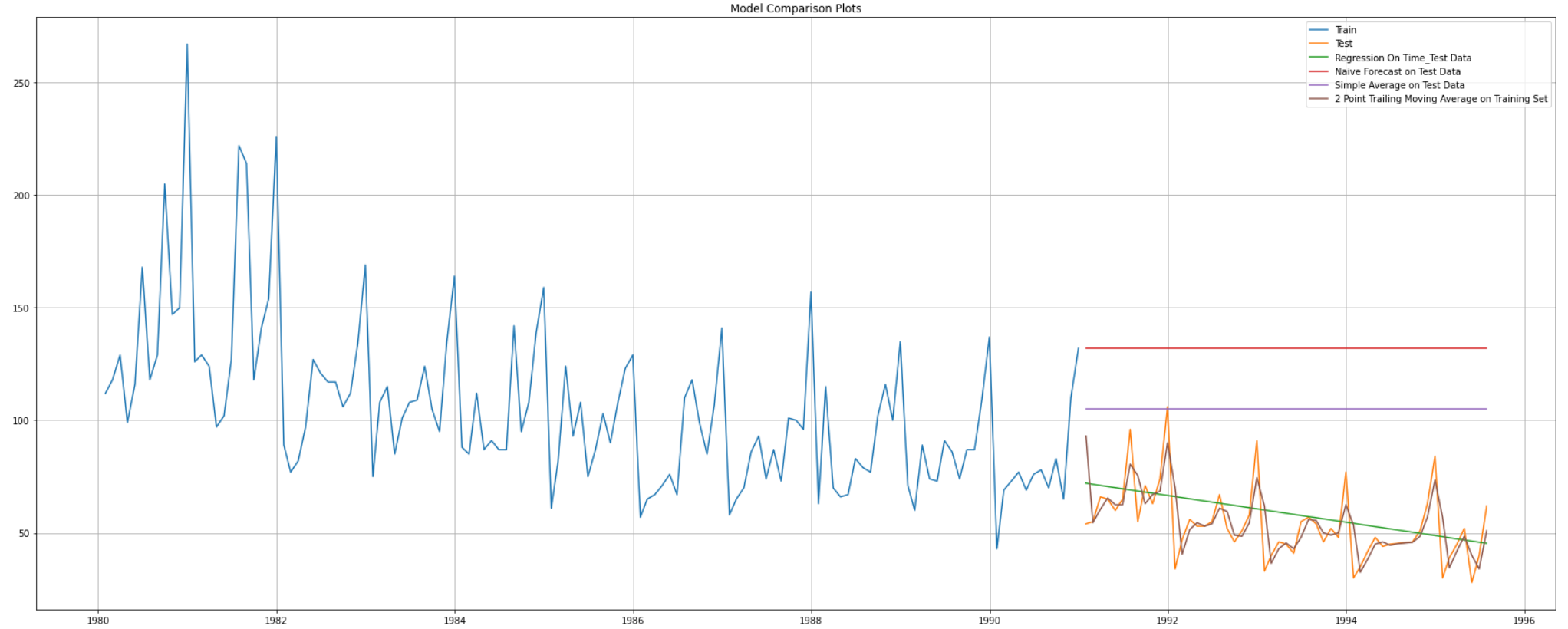
Back to Top

Figure 23. Moving Average Plot (2 Point)



Back to Top

Figure 24. Prediction Plot for Regression, Naïve, Simple Average, and Moving Average (2 Point)



Back to Top

2.4.5 Simple Exponential Smoothing

Simple Exponential Smoothing: (Please refer to the Jupyter notebook)

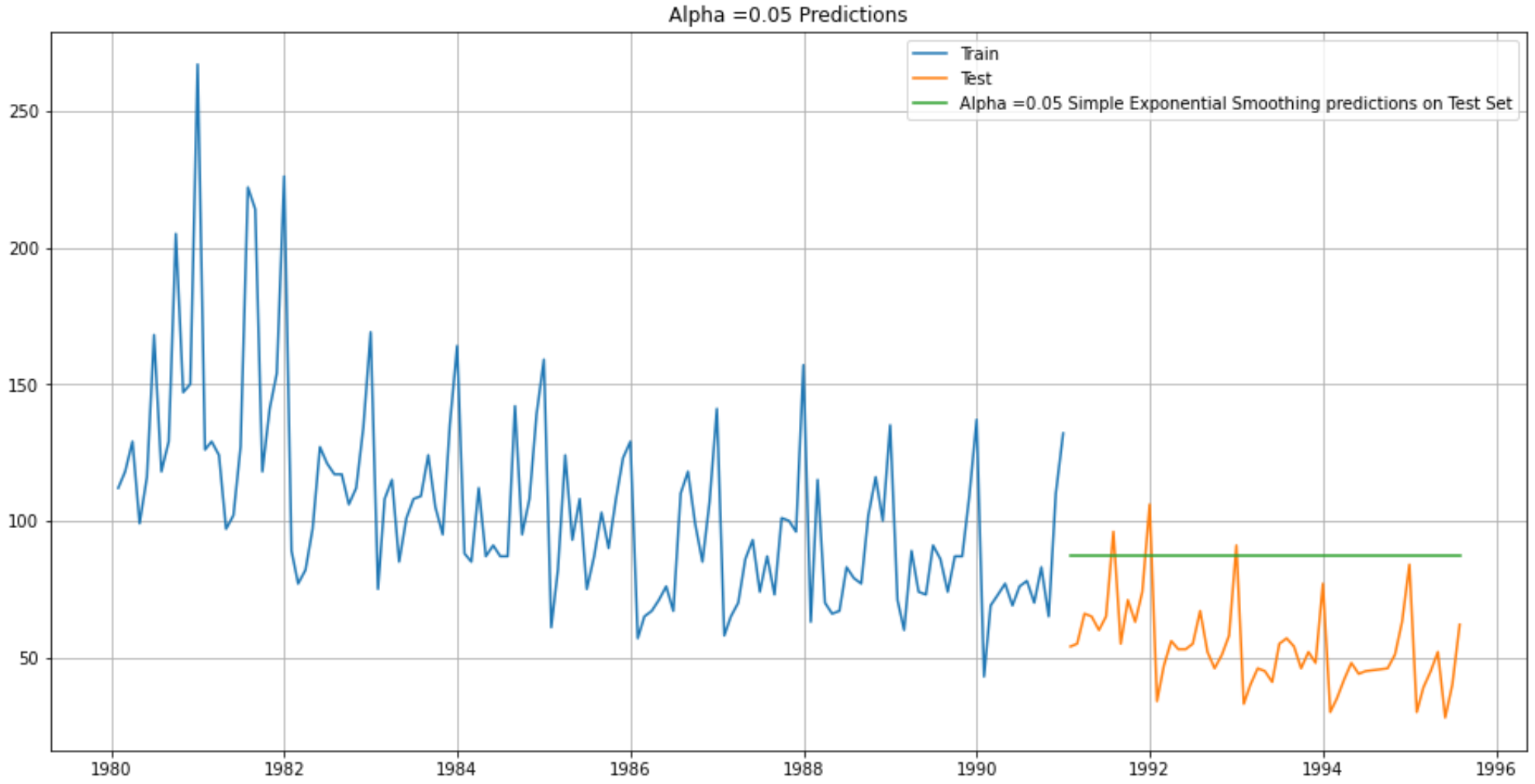
- We have used the SimpleExpSmoothing function and the hyperparameters used for the same is given in the below figure 25
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as **36.796** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Figure 25. Auto Hyperparameters

```
{'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}
```

Figure 26. Simple Exponential Smoothing Plot



2.4.6 Double Exponential Smoothing

Double Exponential Smoothing: (Please refer to the Jupyter notebook)

- We have used the Holt's function and the hyperparameters used for the same is given in the below figure 28
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as **15.269** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

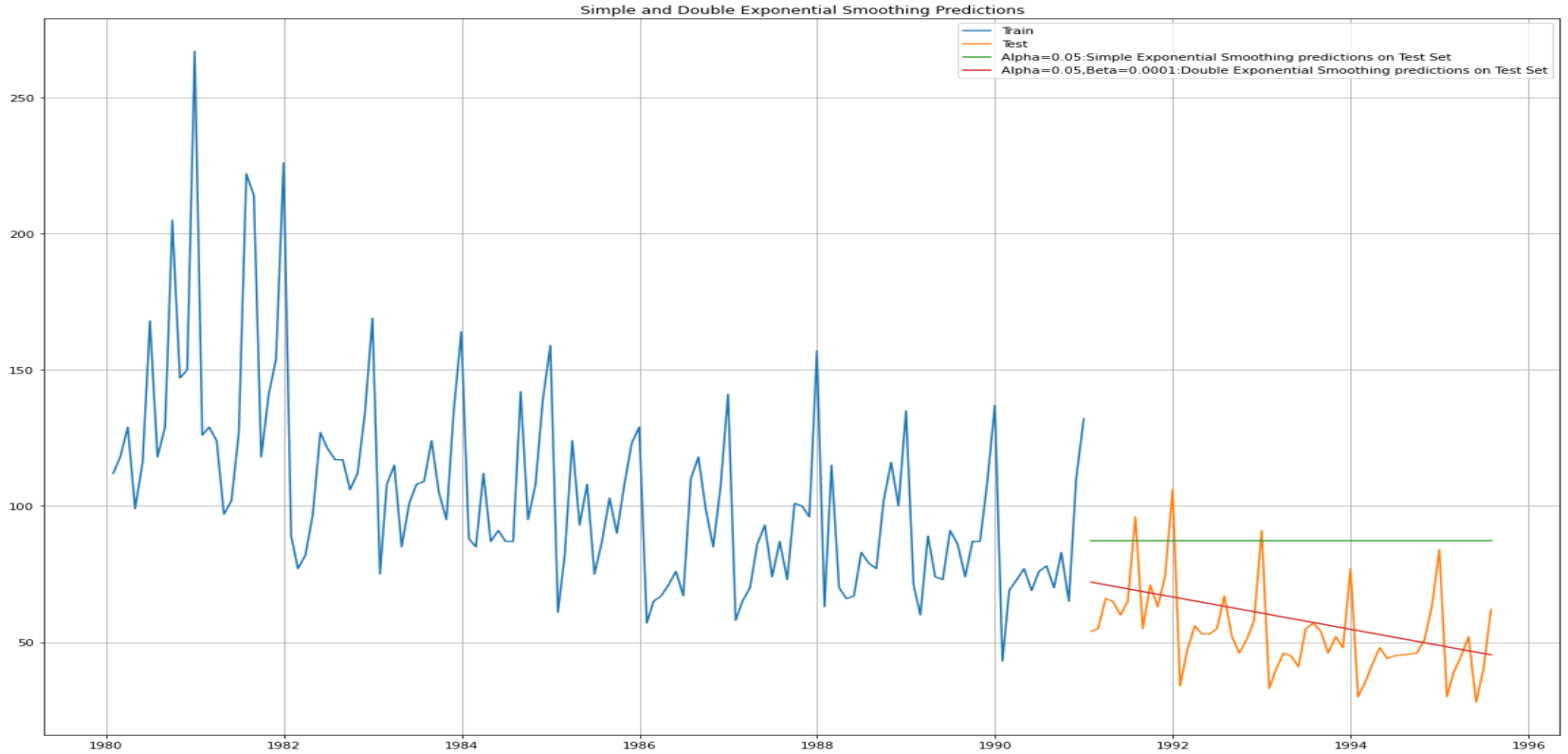
Figure 27. Prediction on Test Dataset

```
Time_Stamp
1991-01-31    72.063238
1991-02-28    71.568859
1991-03-31    71.074481
1991-04-30    70.580103
1991-05-31    70.085725
Name: predict, dtype: float64
```

Figure 28. Holt's Hyperparameters

```
==Holt model Exponential Smoothing Estimated Parameters ==

{'smoothing_level': 1.4901161193847656e-08, 'smoothing_trend': 1.6610391146660035e-10, 'smoothing_seasonal': nan, 'damping_tren
d': nan, 'initial_level': 137.81553690867275, 'initial_trend': -0.4943781897068274, 'initial_seasons': array([], dtype=float6
4), 'use_boxcox': False, 'lamda': None, 'remove_bias': False}
```

Figure 29. Double & Simple Exponential Smoothing Plot



2.4.7 Triple Exponential Smoothing

Triple Exponential Smoothing: (Please refer to the Jupyter notebook)

- We have used the ExponentialSmoothing function and the hyperparameters used for the same is given in the below figure 30 with trend set as additive and seasonality set as multiplicative
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as **21.020** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

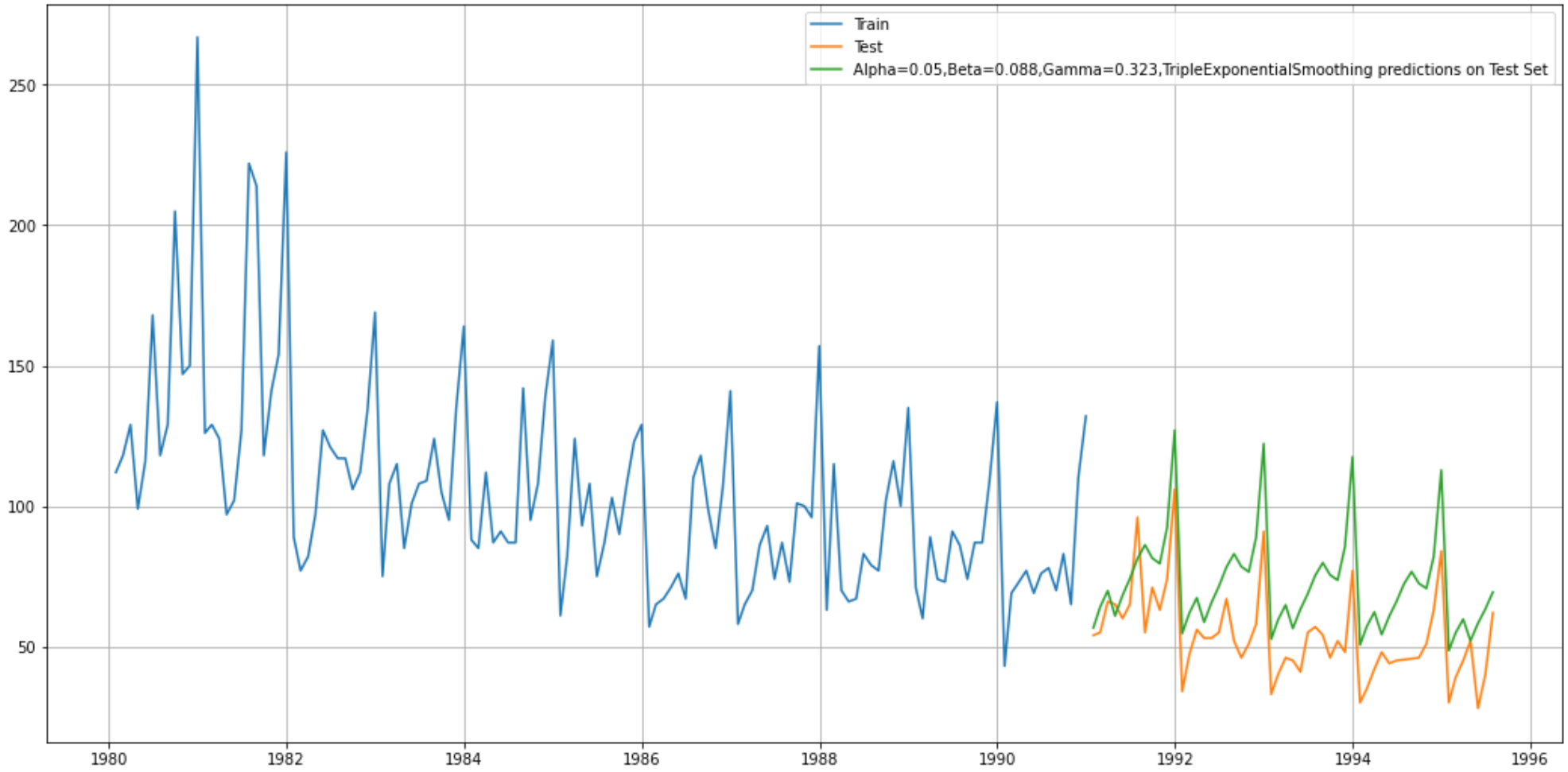
Figure 30. Auto Hyperparameters

```
{'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}
```

Figure 31. Auto Prediction on Test Dataset

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

Figure 32. Triple Exponential Smoothing Auto Plot



We also given hyperparameters manually for the following variables,

- Smoothing Level: 0.3, 1.1, 0.1
- Smoothing Trend: 0.3, 1.1, 0.1
- Smoothing Seasonality: 0.3, 1.1, 0.1

We run these aforementioned iterations and get multiple combinations with RMSE scores on train and test datasets. We have got 512 combinations as follows with top and bottom 5 combinations:

Figure 33. Manual Prediction Combinations

	Alpha Values	Beta Values	Gamma Values	Train RMSE	Test RMSE
0	0.3	0.3	0.3	27.217969	19.057218
1	0.3	0.3	0.4	27.399095	11.201633
2	0.3	0.3	0.5	27.928512	30.565763
3	0.3	0.3	0.6	28.888611	63.623019
4	0.3	0.3	0.7	30.568635	122.472557
...	...	...	...	...	...
507	1.0	1.0	0.6	28358.458519	9603.635095
508	1.0	1.0	0.7	30724.126331	23029.955361
509	1.0	1.0	0.8	1218.755446	9626.710854
510	1.0	1.0	0.9	14150.253251	9691.905402
511	1.0	1.0	1.0	1768.254189	8138.618579

Figure 34. Combinations with sort function on RMSE

	Alpha Values	Beta Values	Gamma Values	Train RMSE	Test RMSE
8	0.3	0.4	0.3	28.111886	10.945435
1	0.3	0.3	0.4	27.399095	11.201633
69	0.4	0.3	0.8	32.601491	12.615607
16	0.3	0.5	0.3	29.087520	14.414604
131	0.5	0.3	0.6	32.144773	16.720720

We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) for manual hyperparameters **10.945** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Figure 35. Triple Exponential Smoothing Plot with Best Parameters

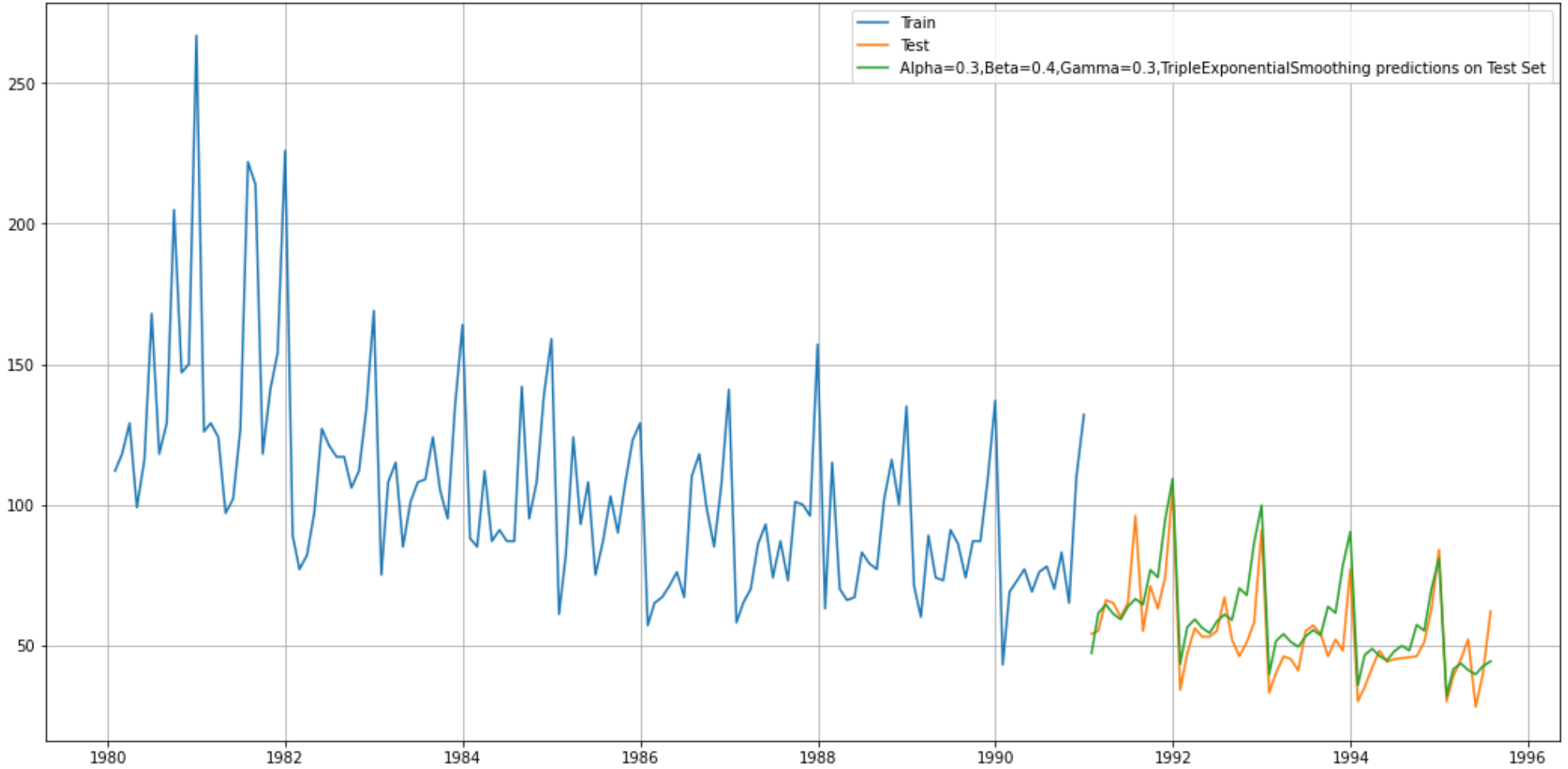
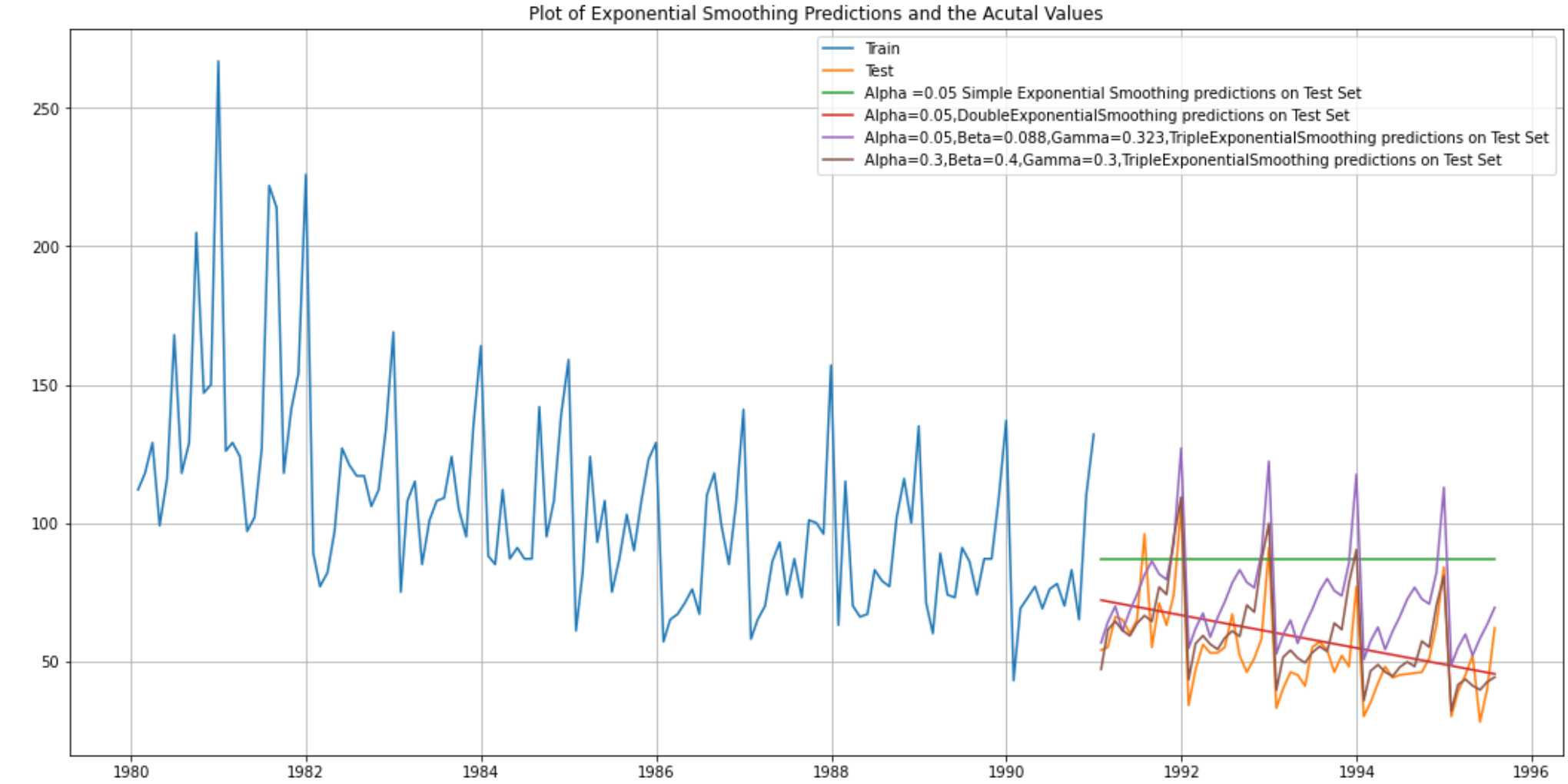


Figure 36. Simple, Double, & Triple Smoothing Plot



Back to Top

2.5 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.

The Augmented Dickey-Fuller test is a unit root test which determines whether there is a unit root and subsequently whether the series is non-stationary.

The hypothesis in a simple form for the ADF test is:

- H_0 : The Time Series has a unit root and is thus non-stationary.
- H_1 : The Time Series does not have a unit root and is thus stationary.

We would want the series to be stationary for building ARIMA models and thus we would want the p-value of this test to be less than the alpha value.

We have used the augmented Dickey Fuller test to check the stationarity of the dataset and we can see that as the p-value is much higher than 0.05 or 95% confidence level as seen below:

Figure 37. ADF Test of Stationarity

```
DF test statistic is -2.240
DF test p-value is 0.46713716277931433
Number of lags used 13
```

Figure 38. ADF Test of Stationarity with difference = 1

```
DF test statistic is -8.162
DF test p-value is 3.015976115828316e-11
Number of lags used 12
```

When we take difference as 1, we can see that the data is stationary and there doesn't seem to be any kind of trend involved. As a result, we can use the data for building models such as ARIMA and SARIMA models.

Figure 39. Stationarity Plot

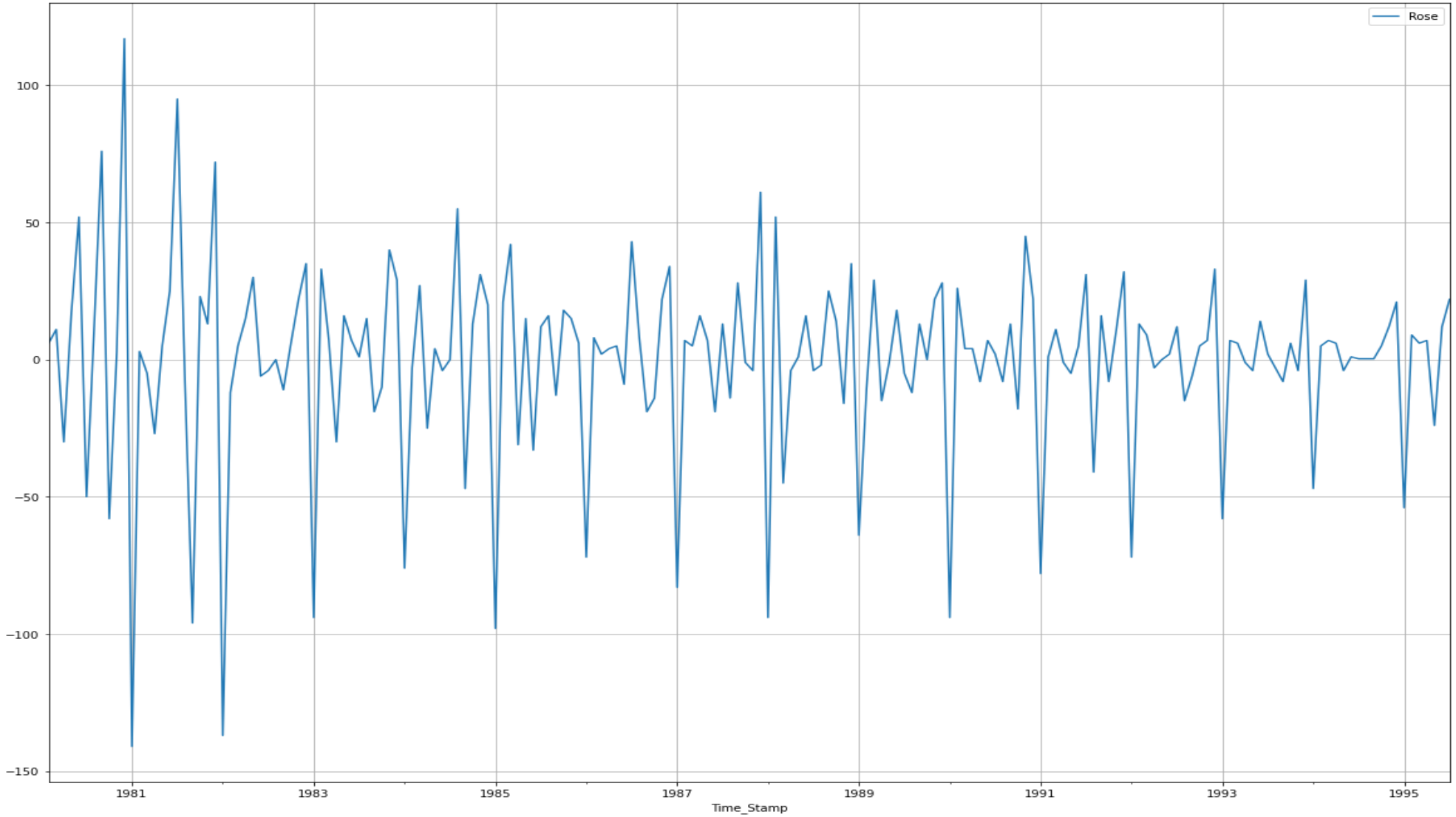
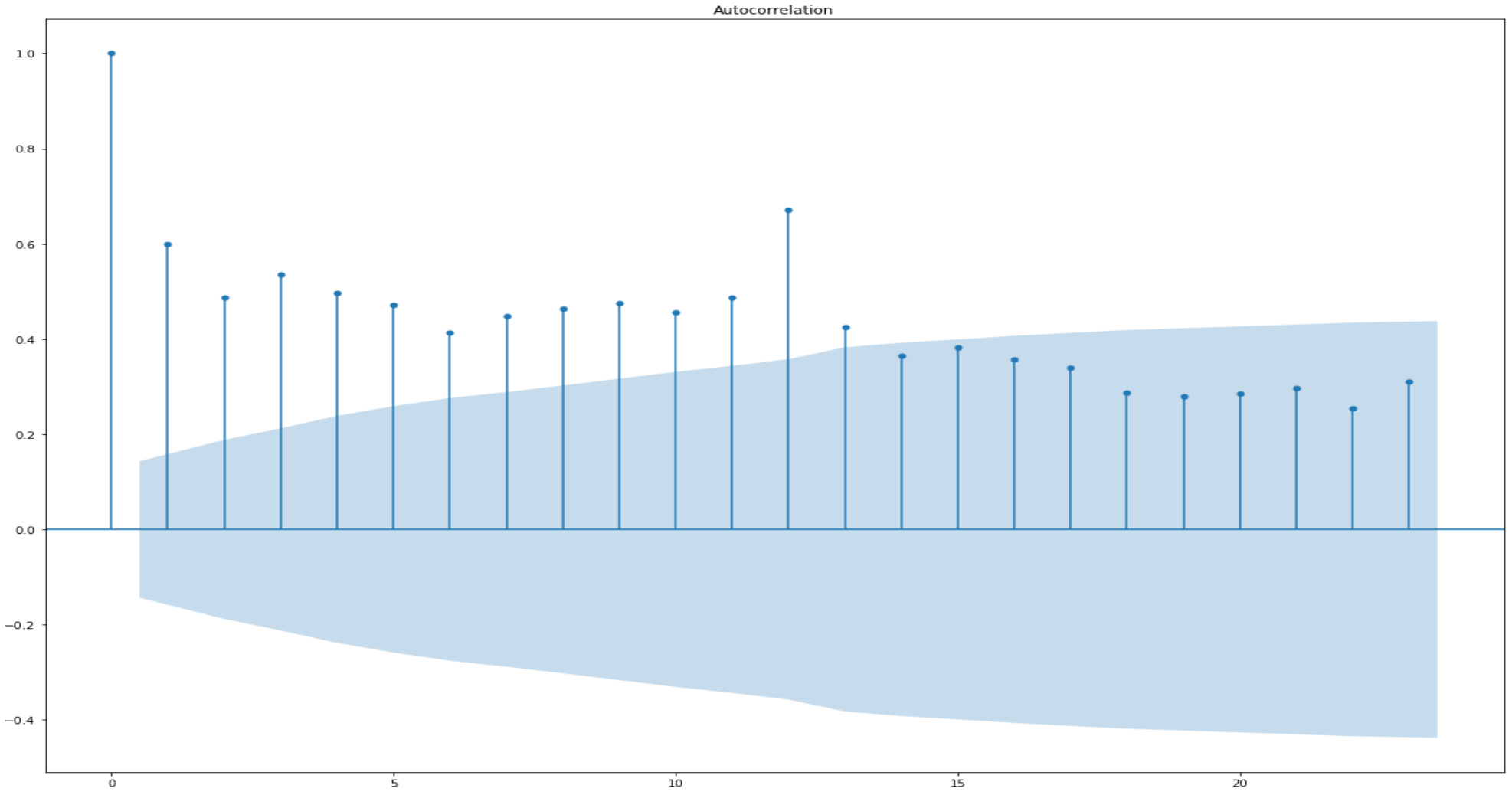
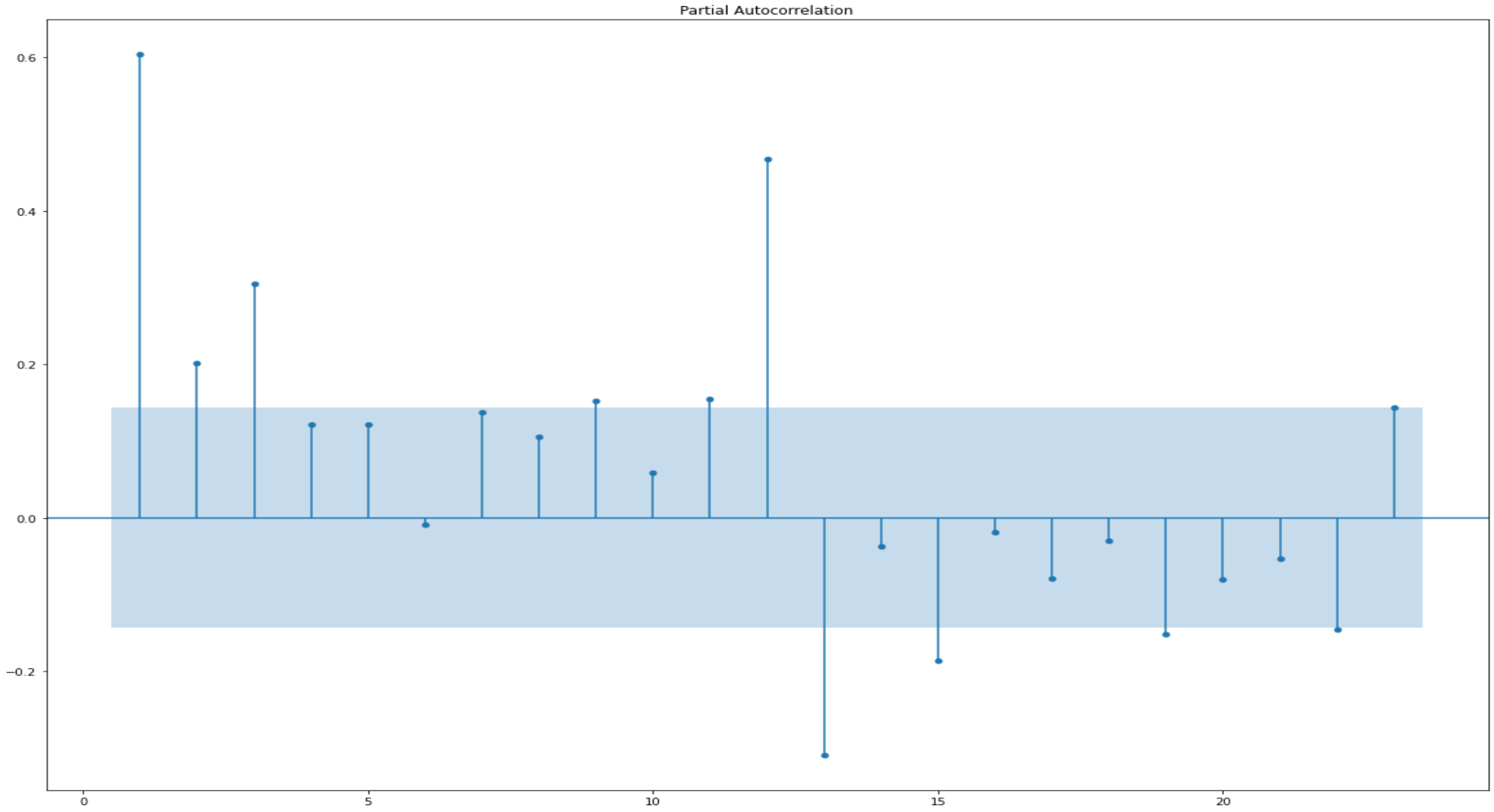


Figure 40. ACF Plot



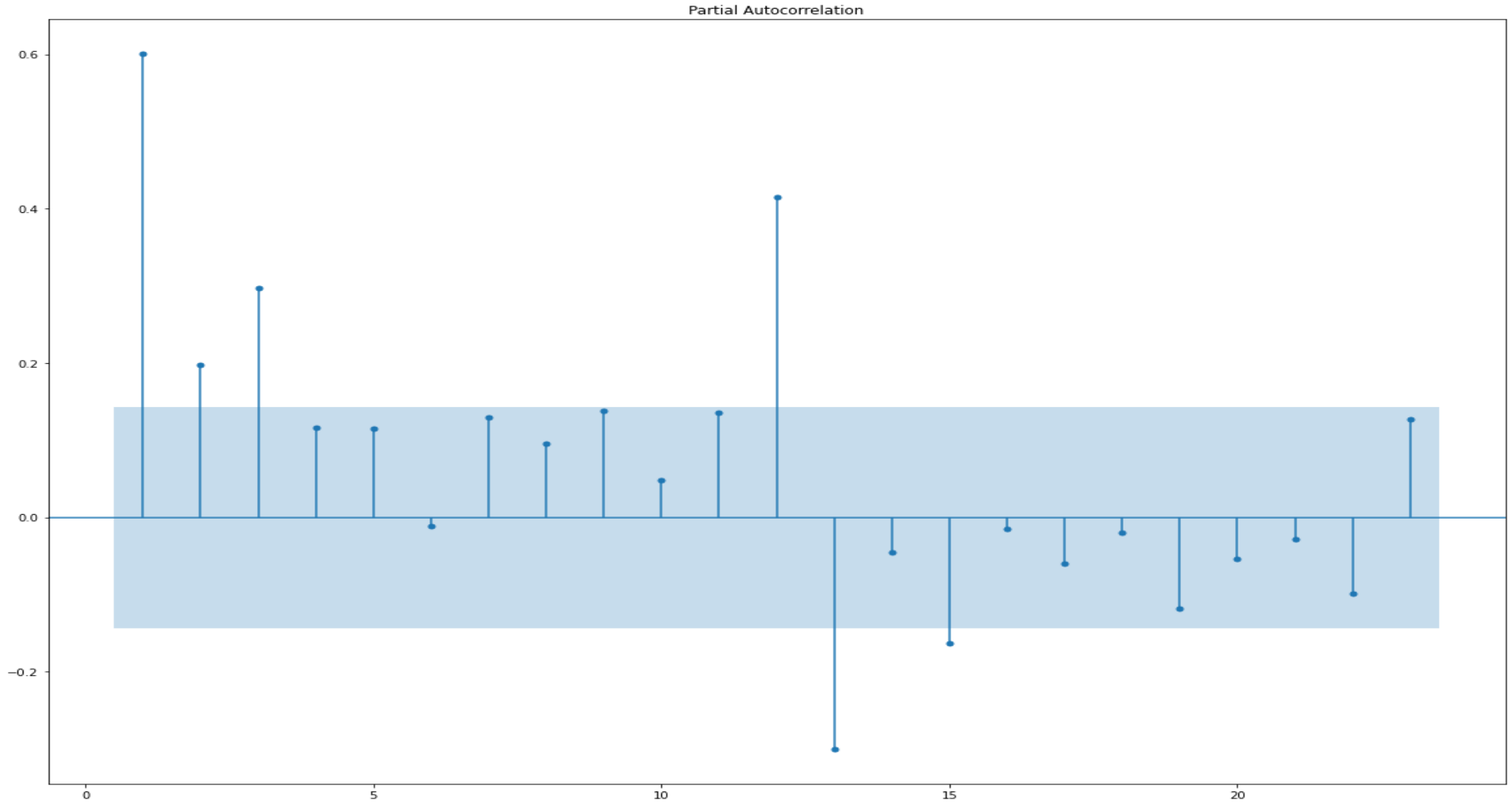
Back to Top

Figure 41. Partial ACF Plot



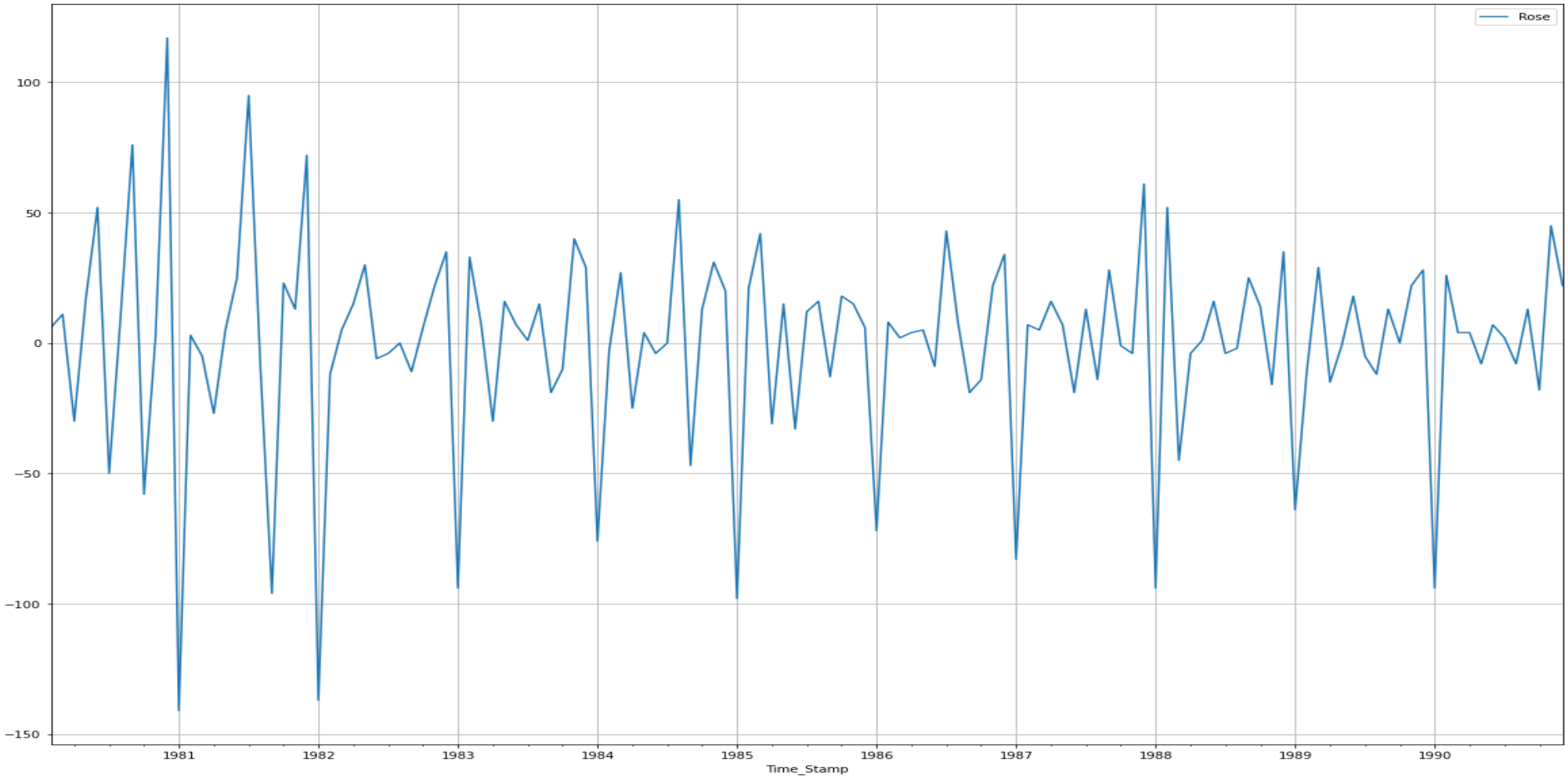
Back to Top

Figure 42. Partial ACF Plot with MLE Method



Back to Top

Figure 43. Train Plot with Stationarity



Back to Top

2.6 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.

2.7 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.

(Note: Combining two questions together 6 & 7)

2.7.1 ARIMA Model

ARIMA Model: (Please refer to the Jupyter notebook)

- We have used iteration tools to try combinations with p, d, q parameters, where p is the number of autoregressive terms, d is the number of nonseasonal differences needed for stationarity, and. q is the number of lagged forecast errors in the prediction equation
- For the above combinations, we have calculated the Akaike Information Criterion (AIC) score to determine which is the best combination with lowest AIC score
- Based on the figure 46, we can see that combination (2,1,3) has the lowest AIC score and we will build a model on the training dataset for the same combination and test the accuracy using by building a summary report, diagnostics, and RMSE score which has come to **36.813**, and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Figure 44. Example Parameter Combinations

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

Figure 45. Parameters with AIC scores

```
ARIMA(0, 1, 0) - AIC:1333.1546729124348
ARIMA(0, 1, 1) - AIC:1282.3098319748312
ARIMA(0, 1, 2) - AIC:1279.6715288535806
ARIMA(0, 1, 3) - AIC:1280.5453761734655
ARIMA(1, 1, 0) - AIC:1317.3503105381492
ARIMA(1, 1, 1) - AIC:1280.5742295380064
ARIMA(1, 1, 2) - AIC:1279.8707234231922
ARIMA(1, 1, 3) - AIC:1281.8707223309998
ARIMA(2, 1, 0) - AIC:1298.6110341605004
ARIMA(2, 1, 1) - AIC:1281.5078621868543
ARIMA(2, 1, 2) - AIC:1281.8707222264356
ARIMA(2, 1, 3) - AIC:1274.6954123405285
ARIMA(3, 1, 0) - AIC:1297.4810917271739
ARIMA(3, 1, 1) - AIC:1282.4192776271989
ARIMA(3, 1, 2) - AIC:1283.720740597716
ARIMA(3, 1, 3) - AIC:1278.6679167115944
```

Figure 46. Parameters with AIC scores (ascending function)

	param	AIC
11	(2, 1, 3)	1274.695412
15	(3, 1, 3)	1278.667917
2	(0, 1, 2)	1279.671529
6	(1, 1, 2)	1279.870723
3	(0, 1, 3)	1280.545376

Figure 47. ARIMA: Summary Report

SARIMAX Results

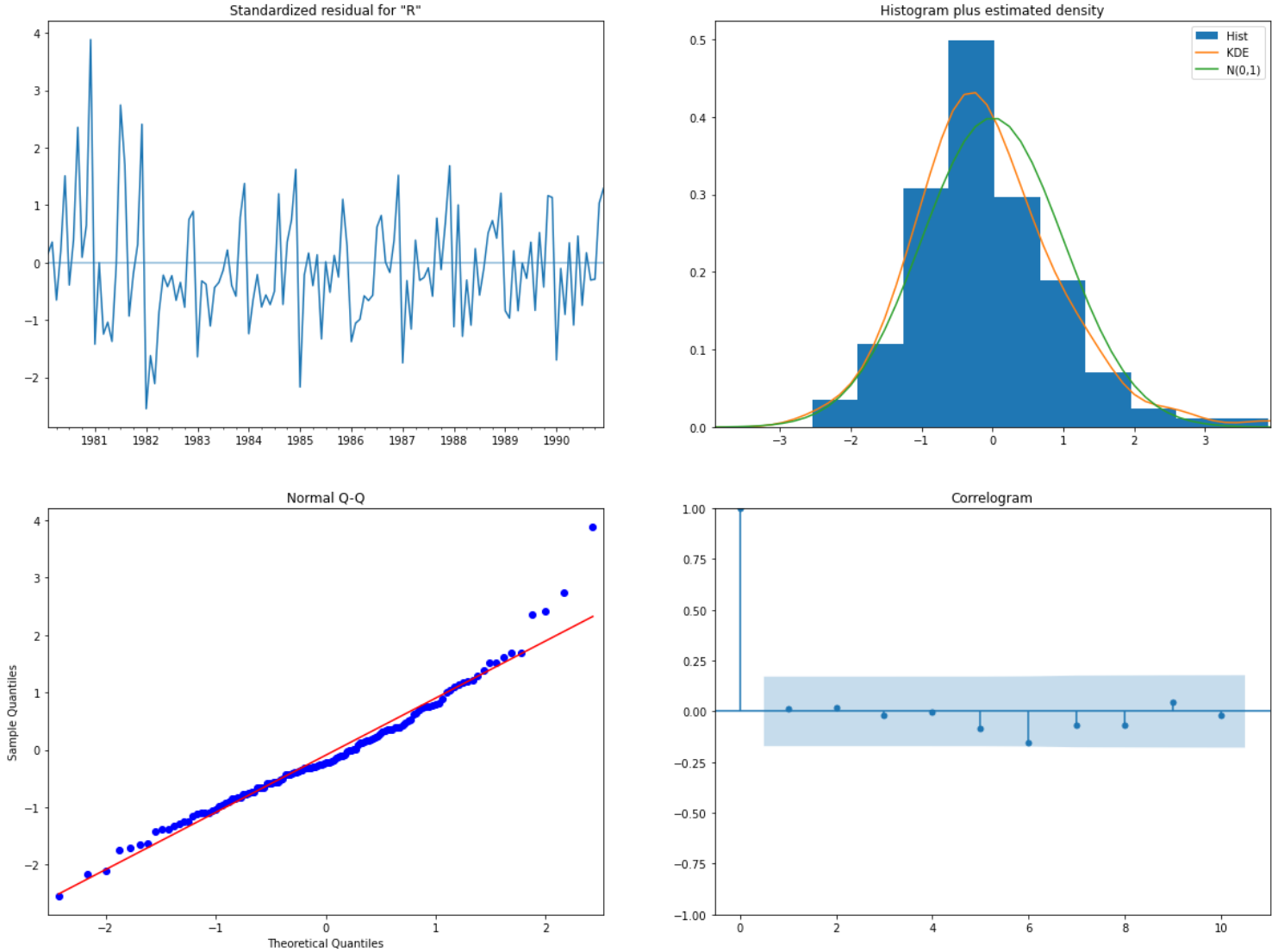
Dep. Variable:	Rose	No. Observations:	132
Model:	ARIMA(2, 1, 3)	Log Likelihood	-631.348
Date:	Fri, 07 Oct 2022	AIC	1274.695
Time:	22:36:43	BIC	1291.947
Sample:	01-31-1980	HQIC	1281.705
	- 12-31-1990		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-1.6783	0.084	-19.999	0.000	-1.843	-1.514
ar.L2	-0.7291	0.084	-8.687	0.000	-0.894	-0.565
ma.L1	1.0446	0.618	1.691	0.091	-0.166	2.255
ma.L2	-0.7720	0.132	-5.858	0.000	-1.030	-0.514
ma.L3	-0.9045	0.560	-1.616	0.106	-2.002	0.192
sigma2	860.3101	519.823	1.655	0.098	-158.525	1879.145

Ljung-Box (L1) (Q):	0.02	Jarque-Bera (JB):	24.51
Prob(Q):	0.87	Prob(JB):	0.00
Heteroskedasticity (H):	0.40	Skew:	0.71
Prob(H) (two-sided):	0.00	Kurtosis:	4.57

Back to Top

Figure 48. ARIMA: Diagnostics



We use Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) to figure out the order of ARIMA model:

Figure 49. ARIMA Model: Autocorrelation Function (ACF)

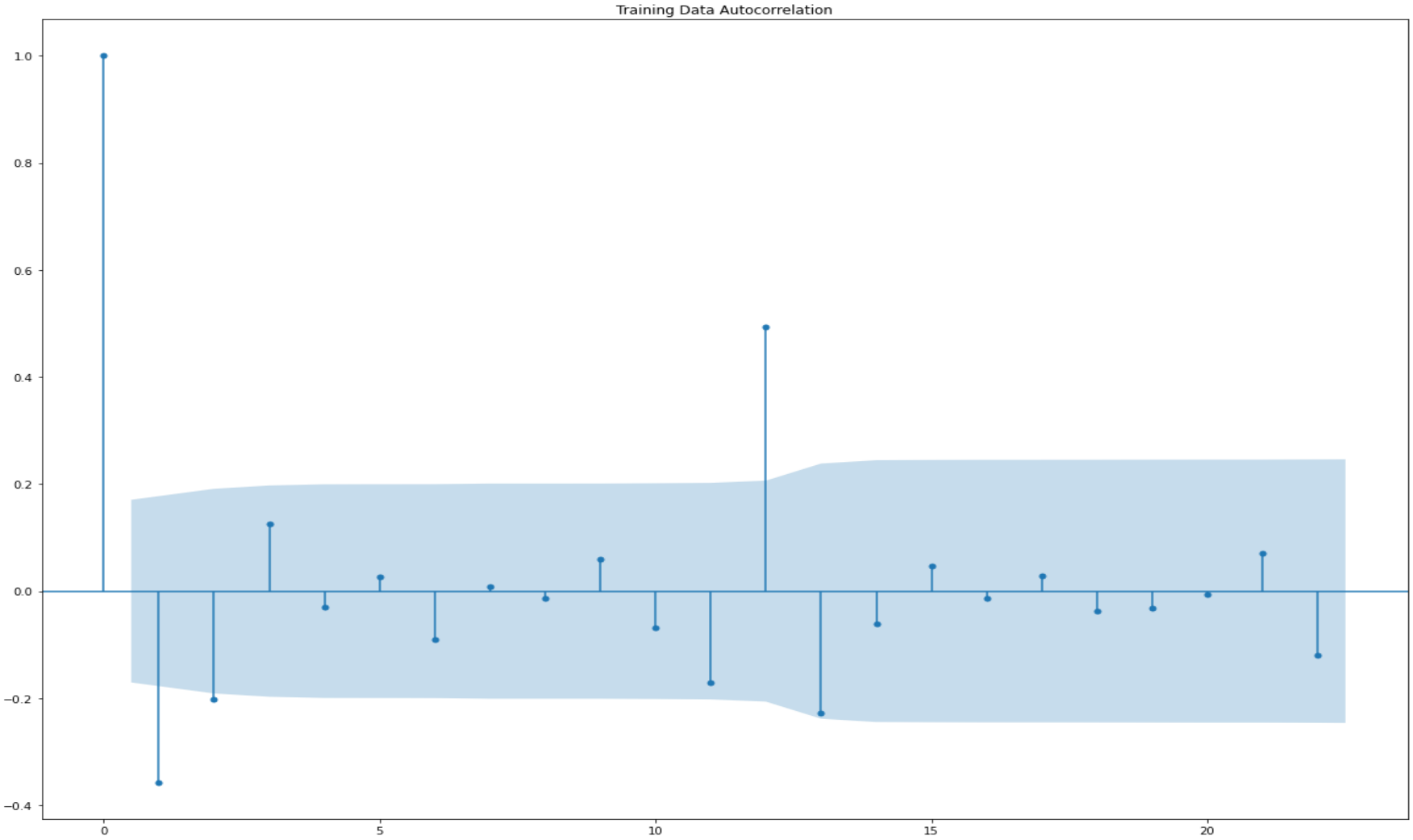
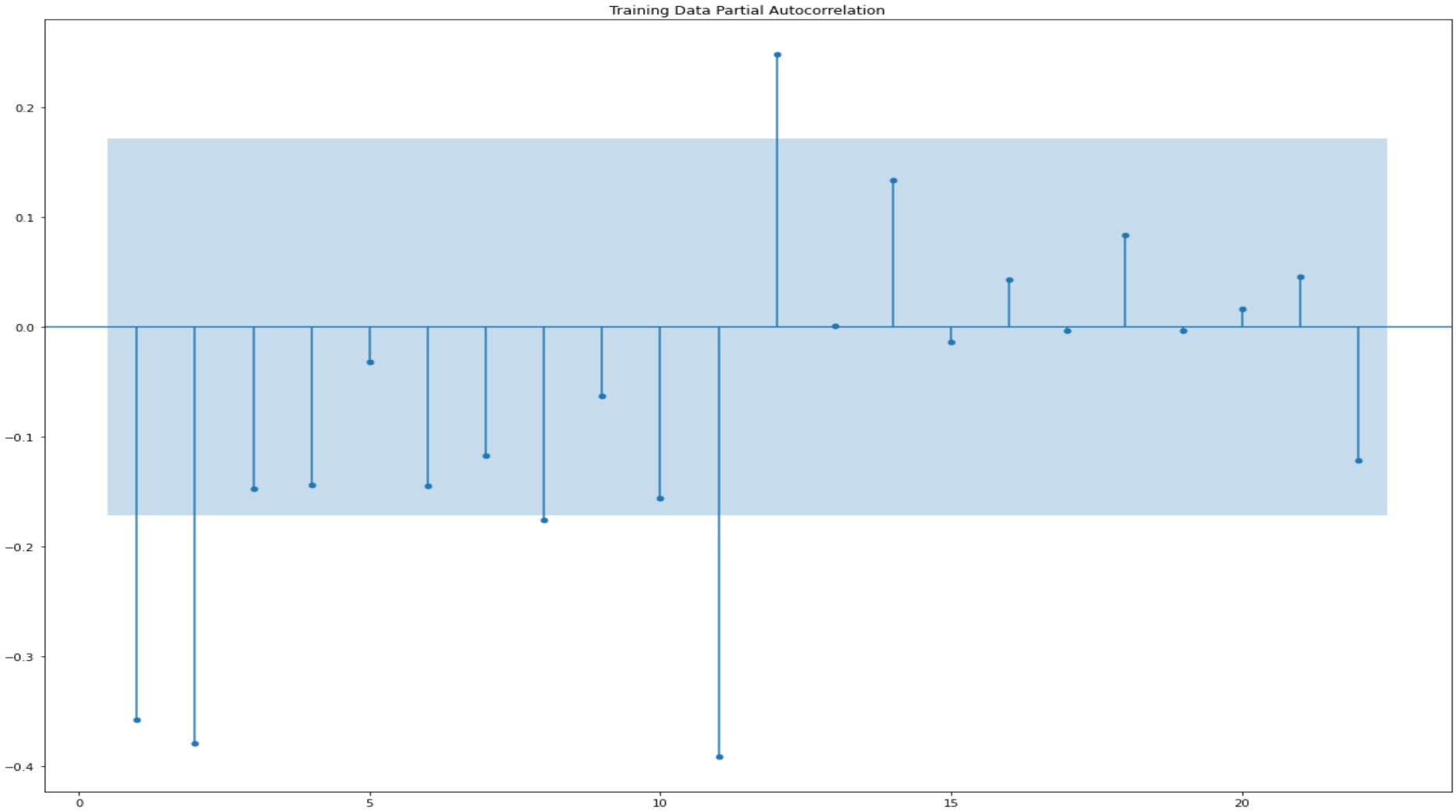


Figure 50. ARIMA Model: Partial Autocorrelation Function (PACF)



Back to Top

We see that the best combination based on the above ACF and PACF plots is (3,1,3) and below is the summary report, diagnostic plots, and manual RMSE score, which is **36.701**, and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Figure 51. ARIMA Model: Partial Autocorrelation Function (PACF)

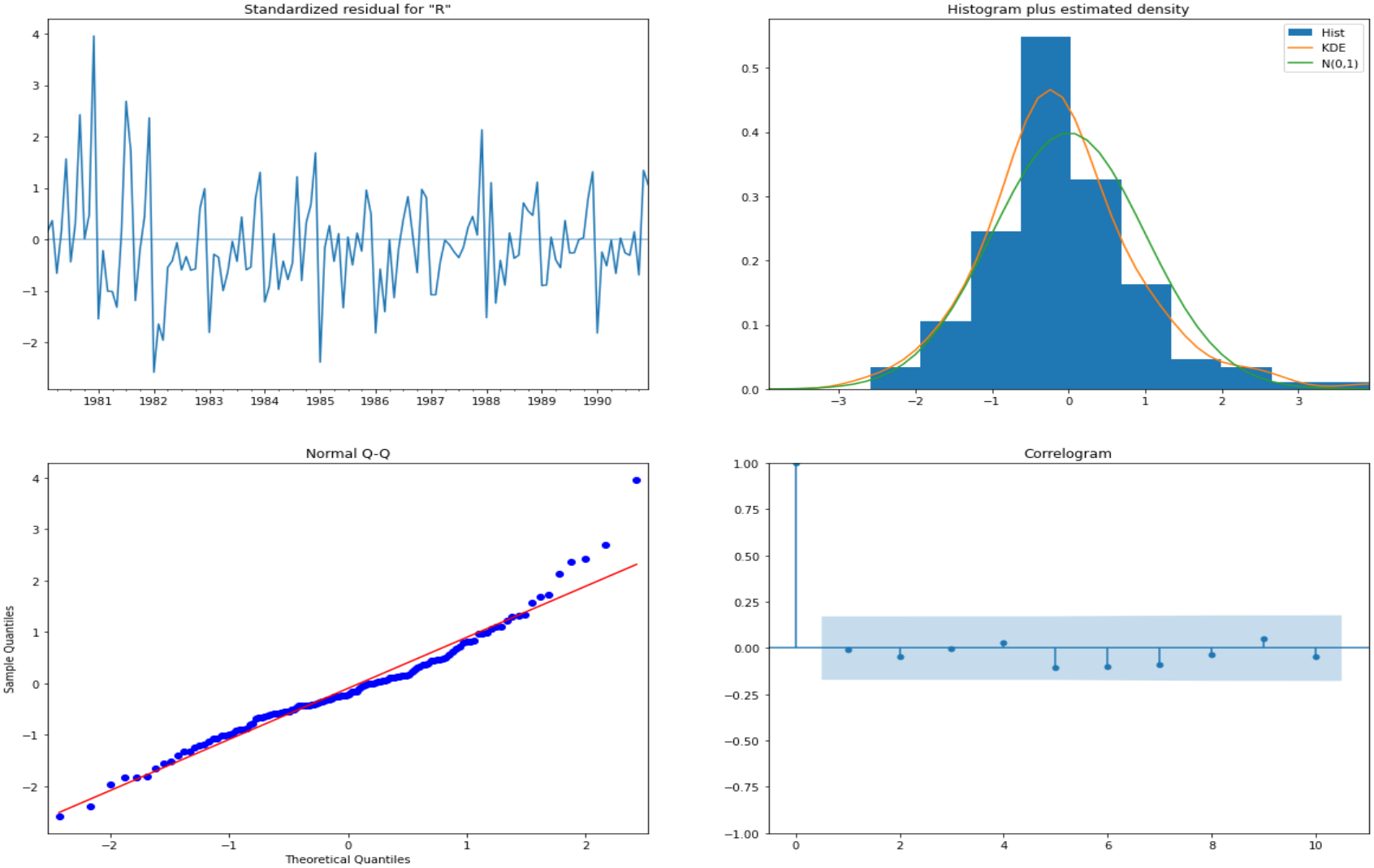
SARIMAX Results

Dep. Variable:	Rose	No. Observations:	132			
Model:	ARIMA(3, 1, 3)	Log Likelihood	-632.334			
Date:	Fri, 07 Oct 2022	AIC	1278.668			
Time:	22:36:45	BIC	1298.794			
Sample:	01-31-1980	HQIC	1286.846			
	- 12-31-1990					
Covariance Type:	opg					

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-1.5851	0.089	-17.908	0.000	-1.759	-1.412
ar.L2	-0.6412	0.142	-4.510	0.000	-0.920	-0.363
ar.L3	0.1329	0.090	1.484	0.138	-0.043	0.309
ma.L1	0.9431	0.131	7.207	0.000	0.687	1.200
ma.L2	-0.7146	0.104	-6.842	0.000	-0.919	-0.510
ma.L3	-0.9080	0.126	-7.219	0.000	-1.155	-0.662
sigma2	884.4157	115.970	7.626	0.000	657.118	1111.714

Ljung-Box (L1) (Q):	0.01	Jarque-Bera (JB):	31.75
Prob(Q):	0.93	Prob(JB):	0.00
Heteroskedasticity (H):	0.37	Skew:	0.72
Prob(H) (two-sided):	0.00	Kurtosis:	4.94

Figure 52. Manual ARIMA: Diagnostics



Back to Top

2.7.2 SARIMA Model

SARIMA Model: (Please refer to the Jupyter notebook)

- We have used iteration tools to try combinations with p, d, q and P, D, Q parameters as performed
- For the above combinations, we have calculated the Akaike Information Criterion (AIC) score to determine which is the best combination with lowest AIC score
- Based on the figure 54, we can see that combination (2, 1, 3) x (2, 0, 3, 6) has the lowest AIC score and we will build a model on the training dataset for the same combination and test the accuracy using by building a summary report, diagnostics, and RMSE score which has come to **27.124,** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Figure 53. Example Parameter Combinations

```
Examples of the parameter combinations for the Model are
Model: (0, 1, 1)(0, 0, 1, 6)
Model: (0, 1, 2)(0, 0, 2, 6)
Model: (0, 1, 3)(0, 0, 3, 6)
Model: (1, 1, 0)(1, 0, 0, 6)
Model: (1, 1, 1)(1, 0, 1, 6)
Model: (1, 1, 2)(1, 0, 2, 6)
Model: (1, 1, 3)(1, 0, 3, 6)
Model: (2, 1, 0)(2, 0, 0, 6)
Model: (2, 1, 1)(2, 0, 1, 6)
Model: (2, 1, 2)(2, 0, 2, 6)
Model: (2, 1, 3)(2, 0, 3, 6)
Model: (3, 1, 0)(3, 0, 0, 6)
Model: (3, 1, 1)(3, 0, 1, 6)
Model: (3, 1, 2)(3, 0, 2, 6)
Model: (3, 1, 3)(3, 0, 3, 6)
```

Figure 54. Parameters with AIC scores (ascending function)

	param	seasonal	AIC
187	(2, 1, 3)	(2, 0, 3, 6)	951.744298
59	(0, 1, 3)	(2, 0, 3, 6)	952.073632
251	(3, 1, 3)	(2, 0, 3, 6)	952.582105
191	(2, 1, 3)	(3, 0, 3, 6)	953.205612
123	(1, 1, 3)	(2, 0, 3, 6)	953.684951

Figure 55. SARIMA Model: Summary Report

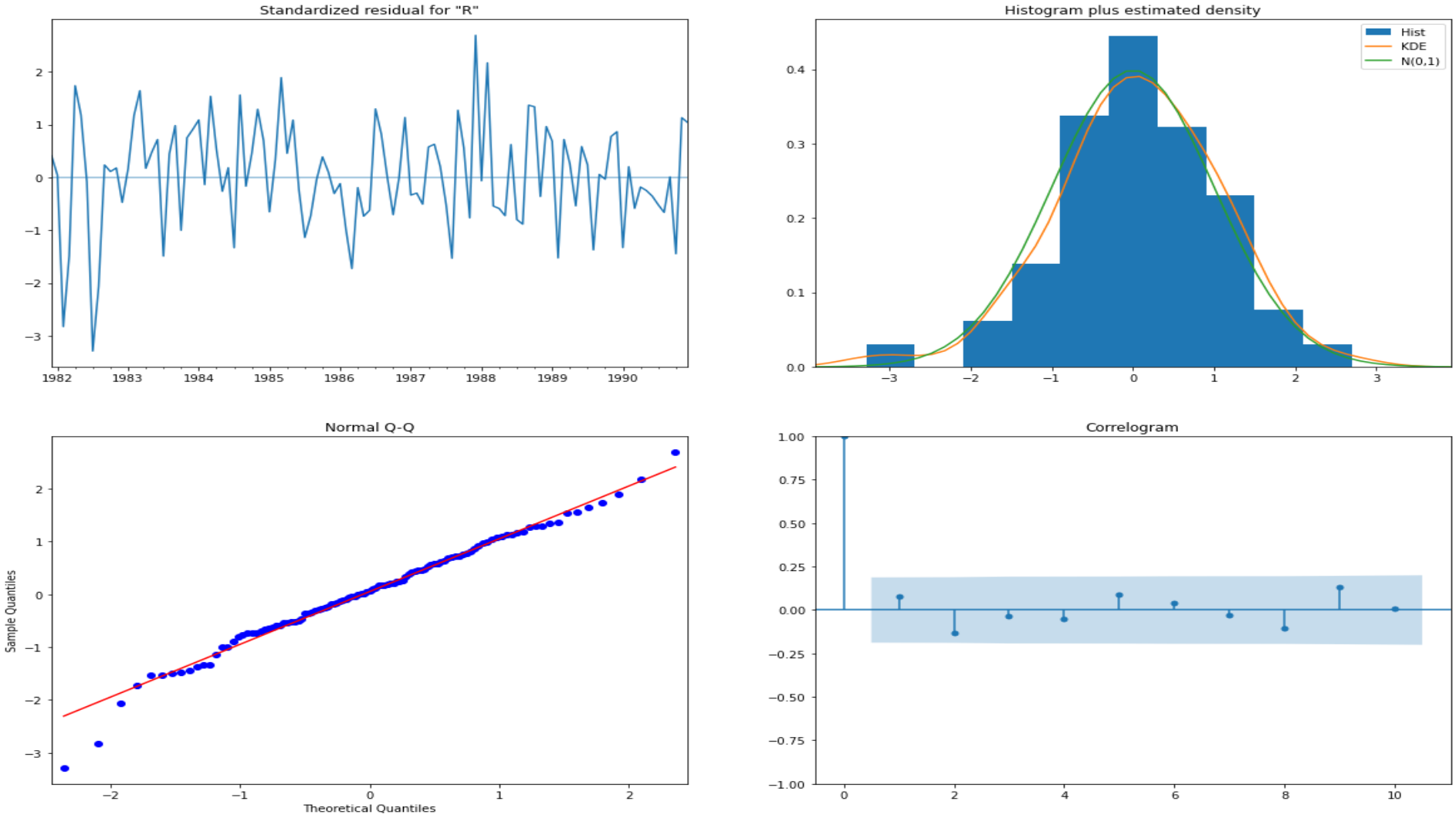
SARIMAX Results

Dep. Variable:	Rose	No. Observations:	132
Model:	SARIMAX(2, 1, 3)x(2, 0, 3, 6)	Log Likelihood	-464.872
Date:	Fri, 07 Oct 2022	AIC	951.744
Time:	22:39:35	BIC	981.349
Sample:	01-31-1980 - 12-31-1990	HQIC	963.750
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.5027	0.083	-6.080	0.000	-0.665	-0.341
ar.L2	-0.6628	0.084	-7.916	0.000	-0.827	-0.499
ma.L1	-0.3714	84.755	-0.004	0.997	-166.487	165.744
ma.L2	0.2034	53.258	0.004	0.997	-104.181	104.588
ma.L3	-0.8320	70.466	-0.012	0.991	-138.943	137.279
ar.S.L6	-0.0838	0.049	-1.720	0.085	-0.179	0.012
ar.S.L12	0.8099	0.052	15.464	0.000	0.707	0.913
ma.S.L6	0.1701	0.248	0.686	0.493	-0.316	0.656
ma.S.L12	-0.5646	0.199	-2.835	0.005	-0.955	-0.174
ma.S.L18	0.1710	0.143	1.198	0.231	-0.109	0.451
sigma2	260.8050	2.21e+04	0.012	0.991	-4.31e+04	4.36e+04

Ljung-Box (L1) (Q):	0.72	Jarque-Bera (JB):	4.77
Prob(Q):	0.40	Prob(JB):	0.09
Heteroskedasticity (H):	0.54	Skew:	-0.36
Prob(H) (two-sided):	0.06	Kurtosis:	3.73

Figure 56. SARIMA Model: Diagnostic



Back to Top

We use Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) to figure out the order of SARIMA model:

Figure 57. SARIMA Model: Autocorrelation Function (ACF)

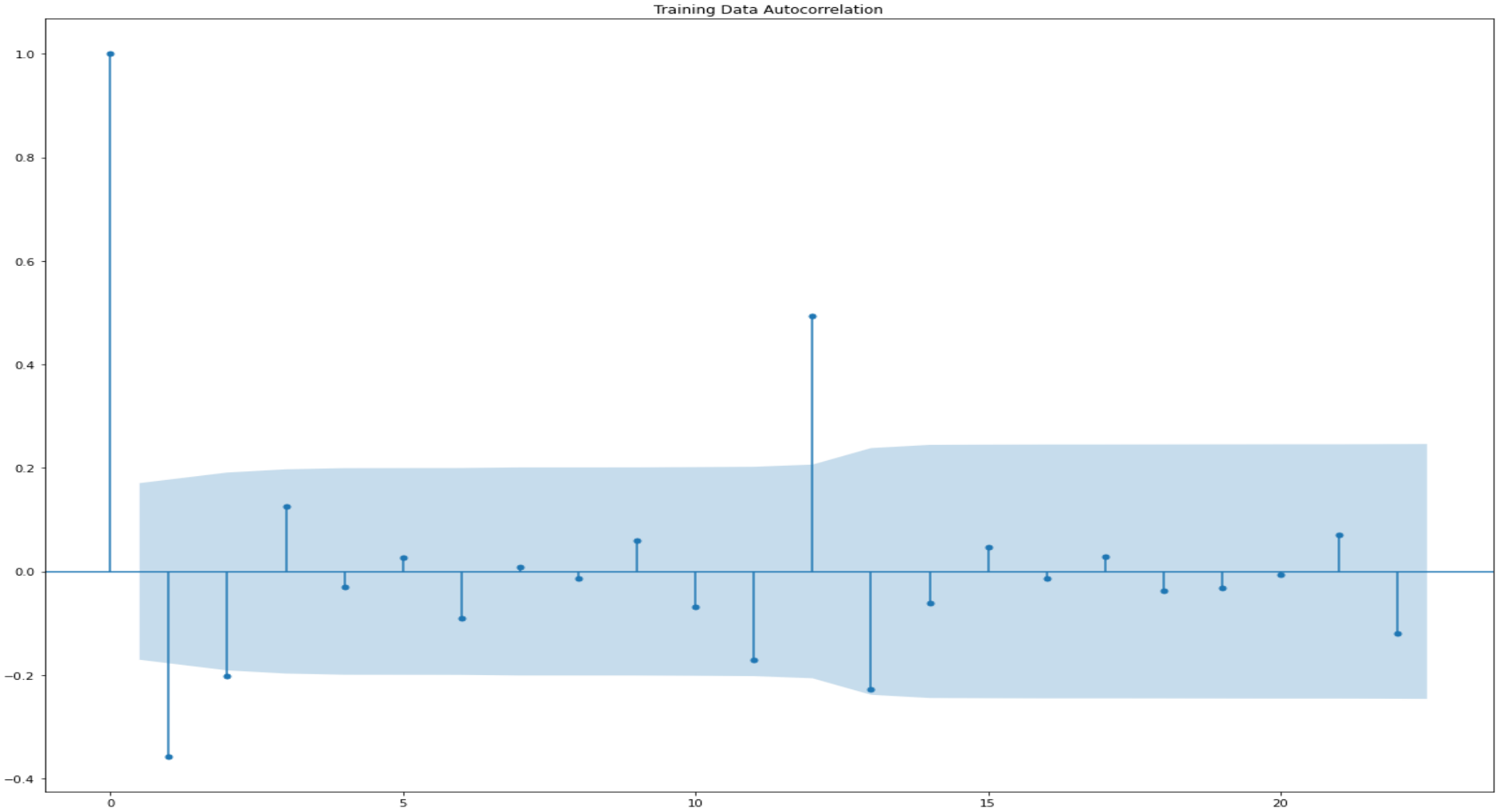
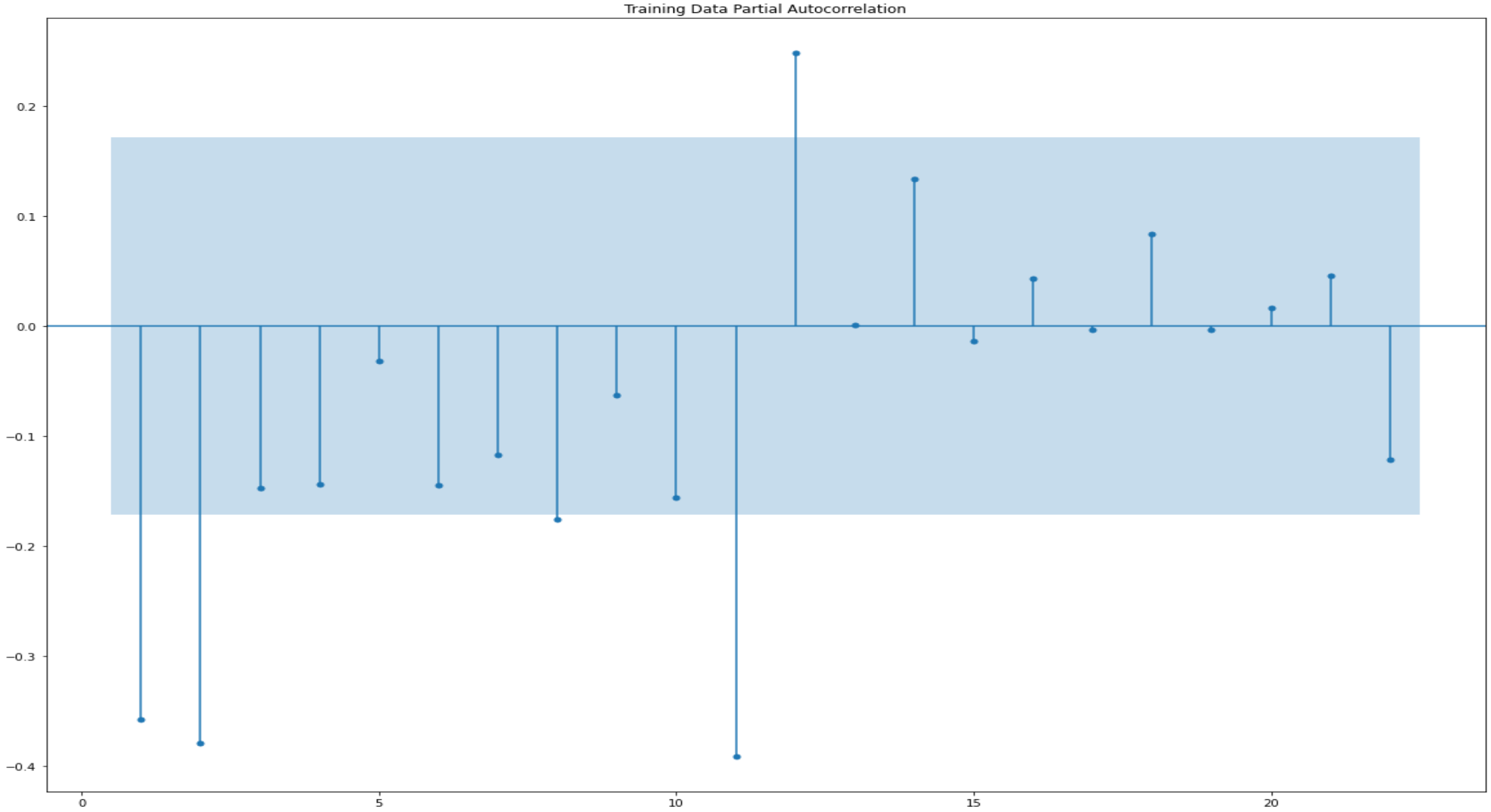


Figure 58. SARIMA Model: Partial Autocorrelation Function (PACF)



Back to Top

We see that the best combination based on the above ACF and PACF plots is (2,1,2) x (2,0,3,6) and below is the summary report, diagnostic plots, and manual RMSE score, which is **30.077**, and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Figure 59. Manual SARIMA: Summary Report

SARIMAX Results

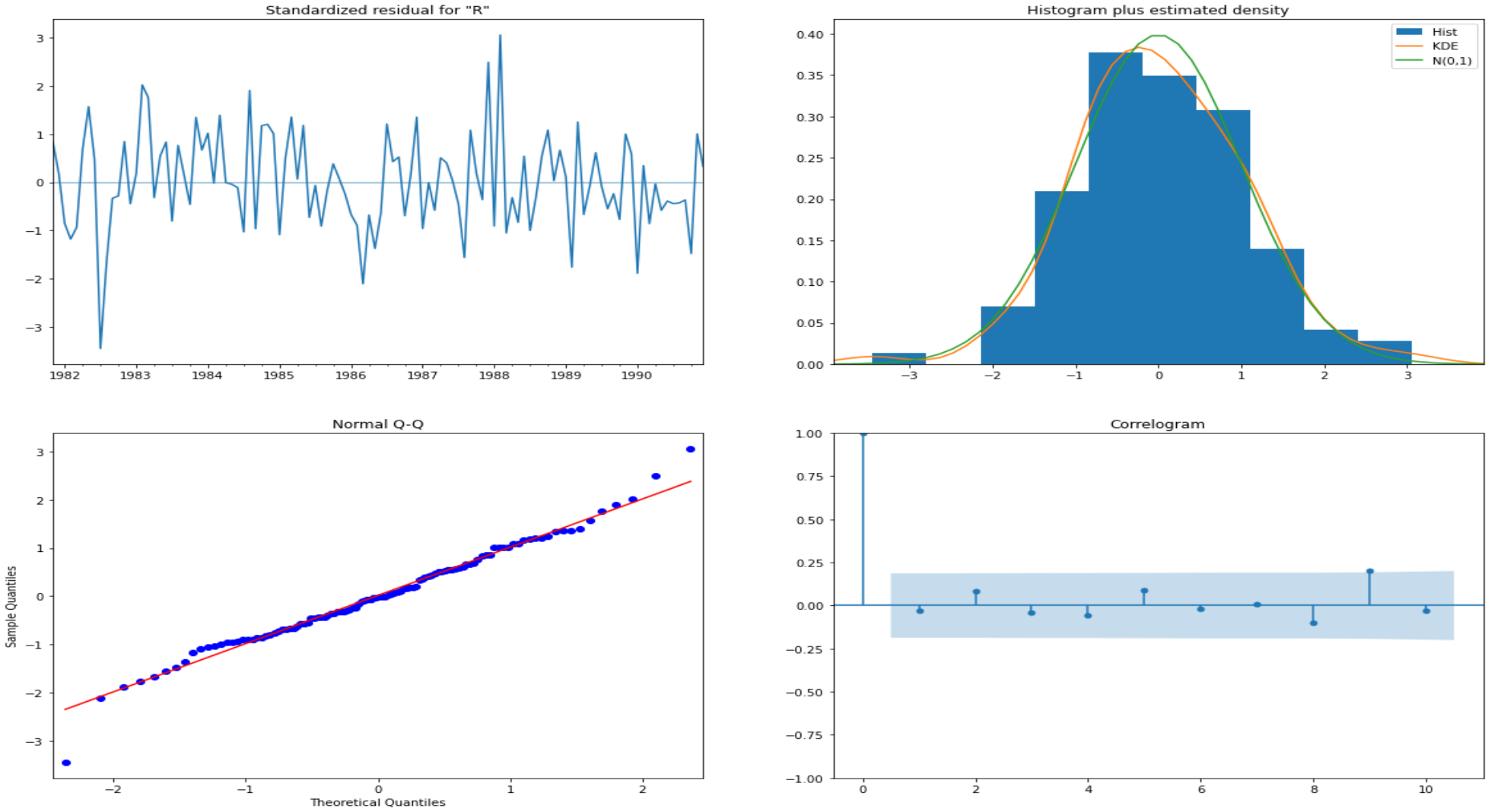
Dep. Variable:	Rose	No. Observations:	132
Model:	SARIMAX(2, 1, 2)x(2, 0, [1, 2, 3], 6)	Log Likelihood	-470.533
Date:	Fri, 07 Oct 2022	AIC	961.066
Time:	22:39:38	BIC	988.071
Sample:	01-31-1980	HQIC	972.019
	- 12-31-1990		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.4994	0.359	1.391	0.164	-0.204	1.203
ar.L2	-0.2647	0.125	-2.123	0.034	-0.509	-0.020
ma.L1	-1.2502	501.185	-0.002	0.998	-983.555	981.054
ma.L2	0.2502	125.499	0.002	0.998	-245.723	246.224
ar.S.L6	-0.0873	0.053	-1.638	0.101	-0.192	0.017
ar.S.L12	0.7252	0.057	12.616	0.000	0.613	0.838
ma.S.L6	0.0876	0.138	0.632	0.527	-0.184	0.359
ma.S.L12	-0.2840	0.123	-2.303	0.021	-0.526	-0.042
ma.S.L18	0.0923	0.113	0.817	0.414	-0.129	0.314
sigma2	288.2581	1.44e+05	0.002	0.998	-2.83e+05	2.83e+05

Ljung-Box (L1) (Q):	0.10	Jarque-Bera (JB):	3.57
Prob(Q):	0.75	Prob(JB):	0.17
Heteroskedasticity (H):	0.83	Skew:	0.00
Prob(H) (two-sided):	0.56	Kurtosis:	3.88

Back to Top

Figure 60. Manual SARIMA: Diagnostic



Back to Top

2.8 Build a table with all the models built along with their corresponding parameters and the respective RMSE values on the test data.

We have built a new dataframe **'Final_results'** with RMSE scores of all the models built to check the accuracy for further forecast for next twelve months

Table 8 Final Results Dataframe with RMSE Scores

	Test RMSE
RegressionOnTime	15.268955
NaiveModel	79.718773
SimpleAverageModel	53.460570
2pointTrailingMovingAverage	11.529278
4pointTrailingMovingAverage	14.451403
6pointTrailingMovingAverage	14.566327
9pointTrailingMovingAverage	14.727630
Alpha=0.05,SimpleExponentialSmoothing	36.796227
Alpha=0.05,DoubleExponentialSmoothing	15.268944
Alpha=0.05,Beta=0.088,Gamma=0.323,TripleExponentialSmoothing	21.019620
Alpha=0.03,Beta=0.04,Gamma=0.3,TripleExponentialSmoothing	10.945435
ARIMA(2,1,3)	36.812984
Manual ARIMA(3,1,3)	36.701328
SARIMA (2, 1, 3) (2, 0, 3, 6)	27.124156
Manual SARIMA (2, 1, 2) (2, 0, 3, 6)	30.077540

In line with the aforementioned table, we can see that manual triple exponential smoothing model is most accurate as its RMSE score is lowest among all the models built for future prediction.

2.9 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.

As manual triple exponential smoothing model gave us the lowest RMSE, we will build the model on the full data for forecasting rose wine sales for next the 12 months, which will give us the most accurate prediction for the same. In the manual triple exponential smoothing model, the following parameters were considered to give most accurate prediction, smoothing level: 0.3, smoothing trend: 0.4, smoothing seasonality: 0.3. We got the RMSE score using the aforementioned parameters on the complete dataset as **20.672**.

Figure 61. 12 Months Prediction Plot

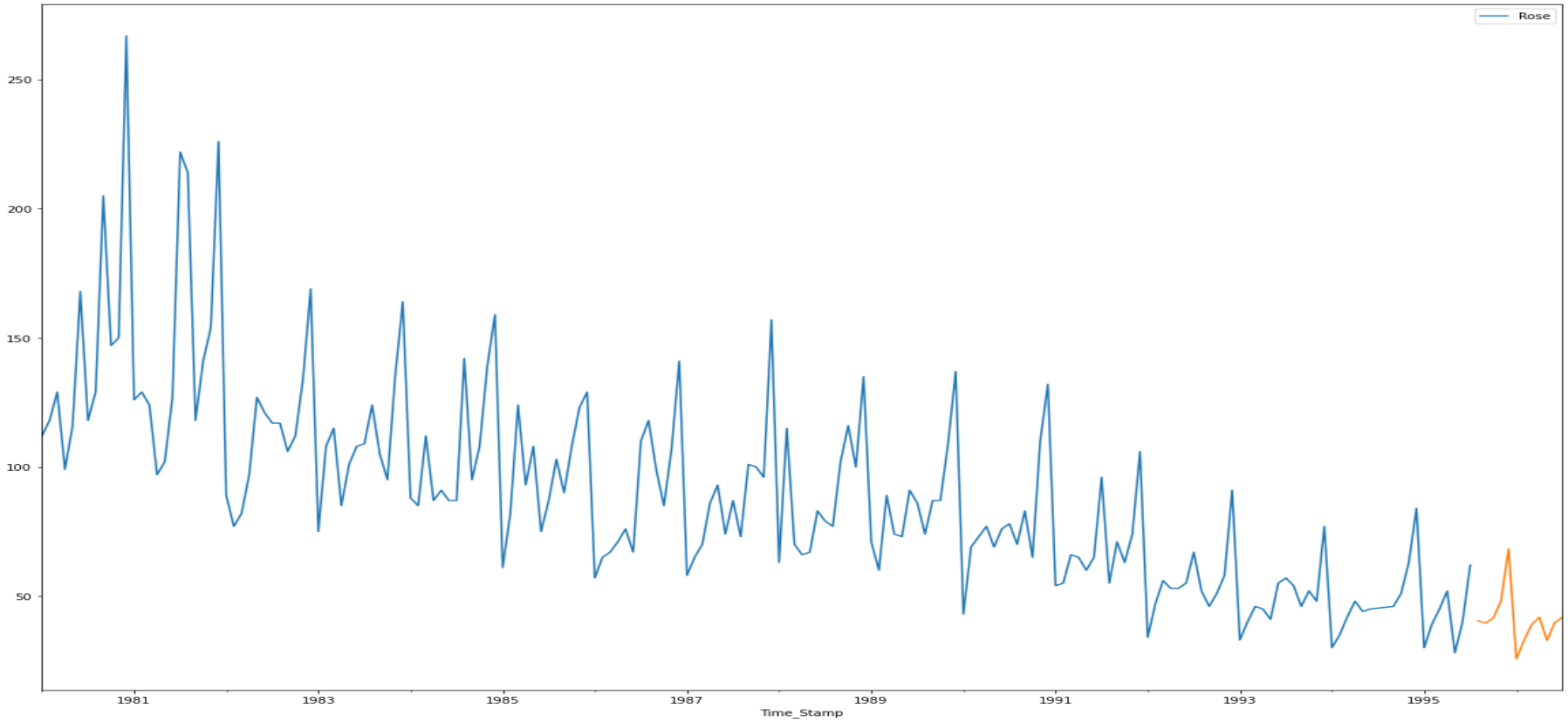


Table 9 Final Results Dataframe with RMSE Scores (including for complete dataset)

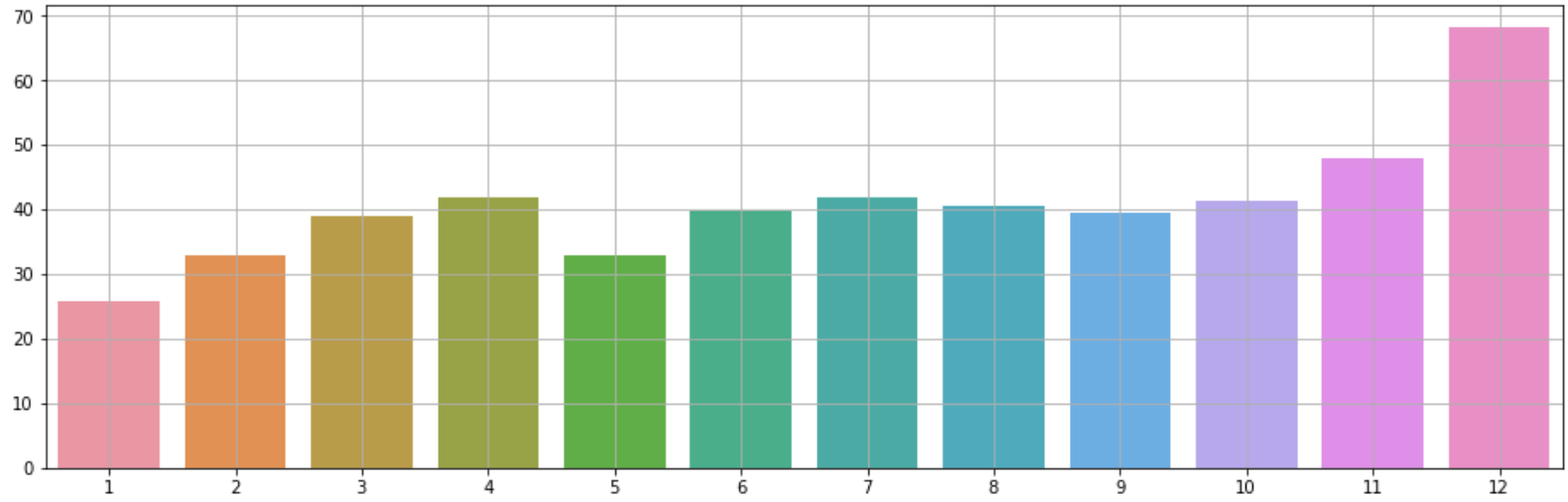
	Test RMSE
RegressionOnTime	15.268955
NaiveModel	79.718773
SimpleAverageModel	53.460570
2pointTrailingMovingAverage	11.529278
4pointTrailingMovingAverage	14.451403
6pointTrailingMovingAverage	14.566327
9pointTrailingMovingAverage	14.727630
Alpha=0.05,SimpleExponentialSmoothing	36.796227
Alpha=0.05,DoubleExponentialSmoothing	15.268944
Alpha=0.05,Beta=0.088,Gamma=0.323,TripleExponentialSmoothing	21.019620
Alpha=0.03,Beta=0.04,Gamma=0.3,TripleExponentialSmoothing	10.945435
ARIMA(2,1,3)	36.812984
Manual ARIMA(3,1,3)	36.701328
SARIMA (2, 1, 3) (2, 0, 3, 6)	27.124156
Manual SARIMA (2, 1, 2) (2, 0, 3, 6)	30.077540
Full Data Triple Exponential Smoothing Alpha=0.3,Beta=0.4,Gamma=0.3	20.672561

2.10 Comment on the model thus built and report your findings and suggest the measures that the company should be taking for future sales.

2.10.1 Comments on the data and model built

- We see that manual triple exponential smoothing model gives us the lowest RMSE and has most accurately predicts or forecasts for the next 12 months
- We can clearly see that the rose wine sales are declining year-on-year and we can see that there is an obvious trend that the sales data is following yearly and monthly
- In addition, there is a slight upward trend in sales when we see the box plot for monthly sales from January to December. The rose wine sales gradually pick up when we look at sales figures for a particular year on a monthly basis.
- There is clear seasonality in the data and its constant on a yearly basis. We can see that January denotes lowest sales while December records higher number of sales as compared to other months.
- The parameters used to build a triple exponential smoothing model on train data is smoothing level as 0.3, smoothing trend as 0.4, and smoothing seasonality is 0.3, which gives highest accuracy. In addition, we can see that the model built on training data is not so accurate when to predict the dips in the sales, however, it resembles with the peaks in sales.
- The revenue sales are estimated to total around **491.34** for next 12 months with an average of **40.94**

Figure 62. 12 Months Bar Plot on Forecast Data, August 1995 - July 1996



Back to Top

2.10.2 Measures that the company should be taking for future sales

- As sales are following a declining trend and the sales are decreasing year-on-year, the needs to take some smart decisions and may be identify potential markets to improve sales figures
- The company could pass on a survey cards/online surveys to their existing customers to understand what all suggestions come in to improve their sales or rose wine collection
- The company can have wine tasting events twice or thrice a year at their estate to popularize their wine collections and get feedback from its customers
- Offering food dishes which complement rose wines could prove helpful in attracting customers and improve sales for the rose wines
- As a last resort, if the company has started to incur losses, the company needs to compare costs with revenue generated through rose sales, and take a strategic decision to shut down the production of rose wines and focus more on wines delivering higher revenues

Back to Top

Chapter 3. Sparkling

We have created two dataframes sparkling1 (refer the jupyter notebook) and sparkling2 to read the data differently. However, as we have used sparkling2 dataframe for further analysis, we will refer to the same dataframe for below analysis.

3.1 Read the data as an appropriate Time Series data and plot the data.

Table 10 Dataframe: sparkling2 (with head function)

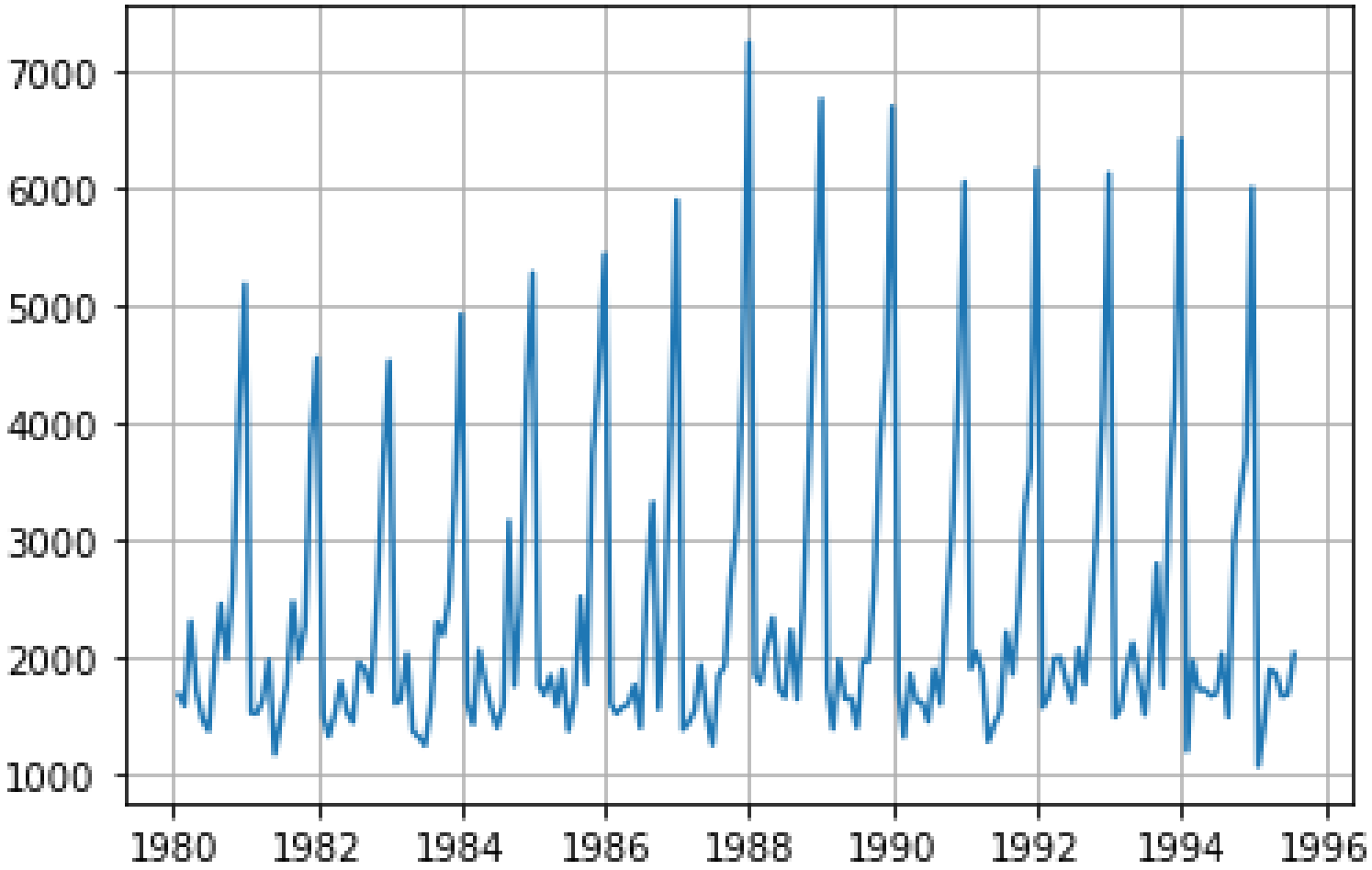
	YearMonth	Sparkling	Time_Stamp
0	1980-01	1686	1980-01-31
1	1980-02	1591	1980-02-29
2	1980-03	2304	1980-03-31
3	1980-04	1712	1980-04-30
4	1980-05	1471	1980-05-31

Table 11 Dataframe: sparkling2 (with describe function)

	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

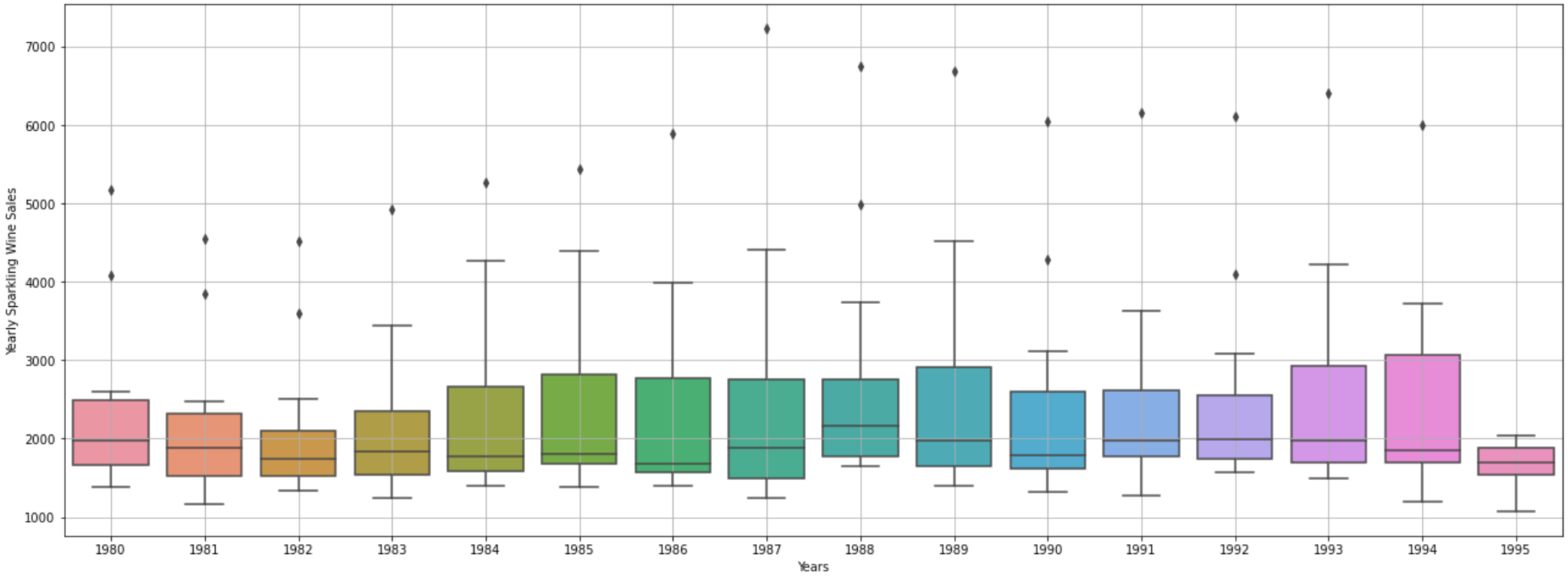
Back to Top

Figure 63. Dataframe sparkling2: Plot



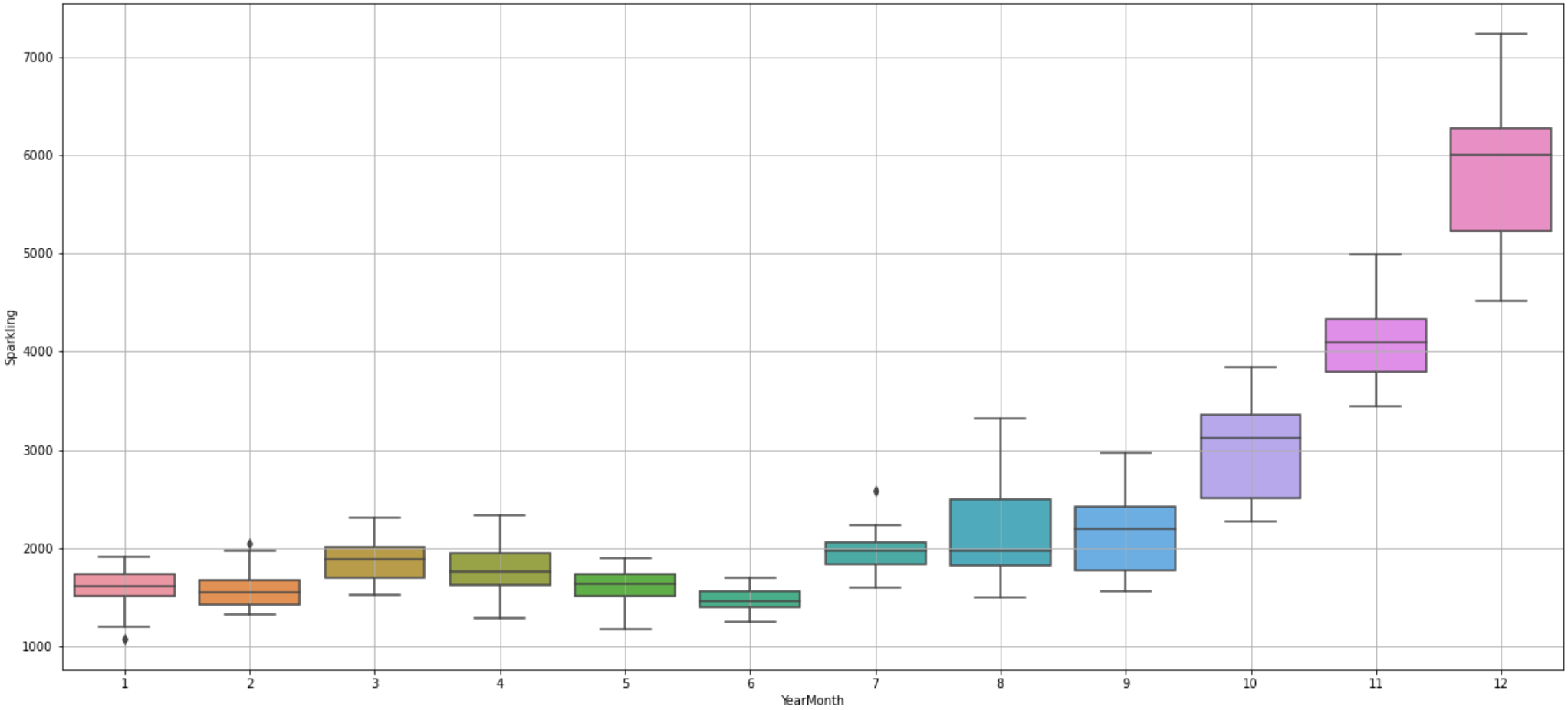
3.2 Perform appropriate Exploratory Data Analysis to understand the data and also perform decomposition.

Figure 64. Yearly Sparkling Wine Sales



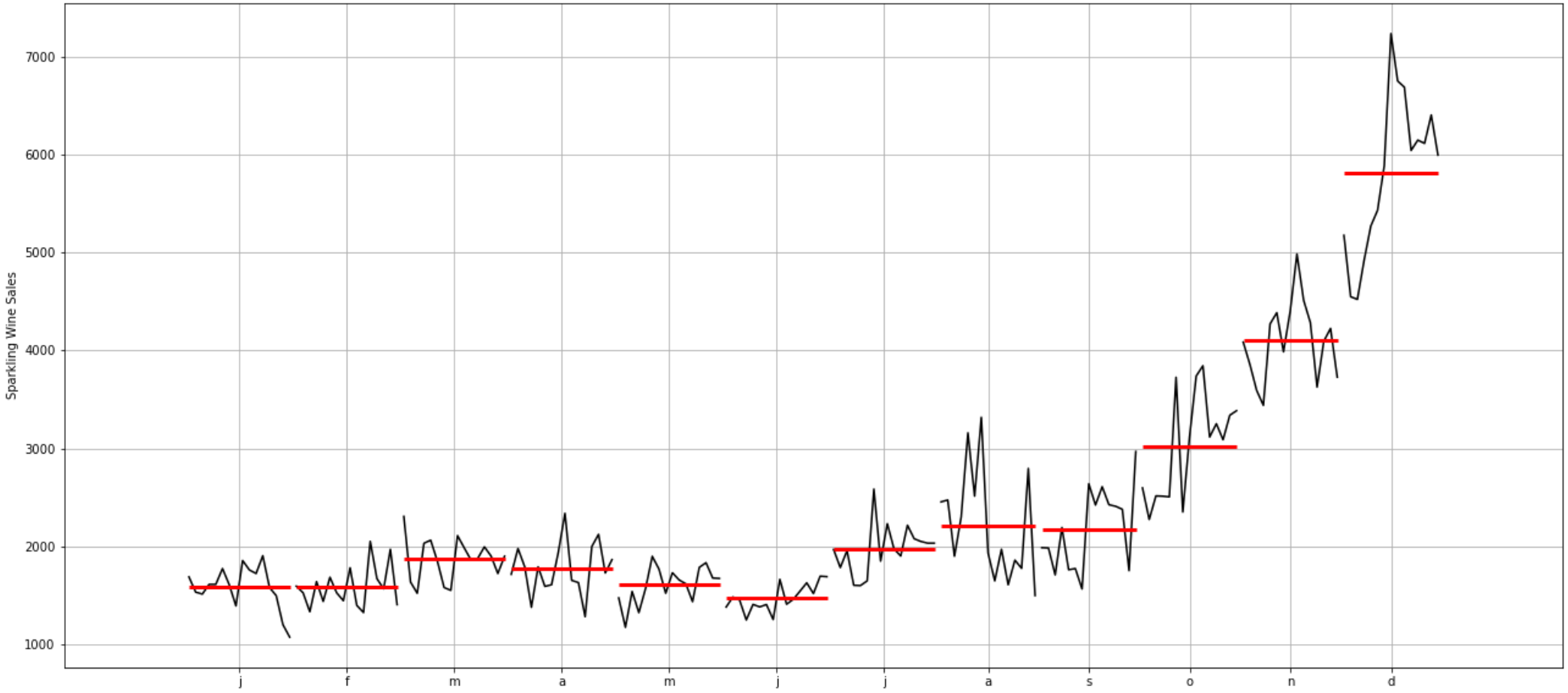
Back to Top

Figure 65. Monthly Sparkling Wine Sales



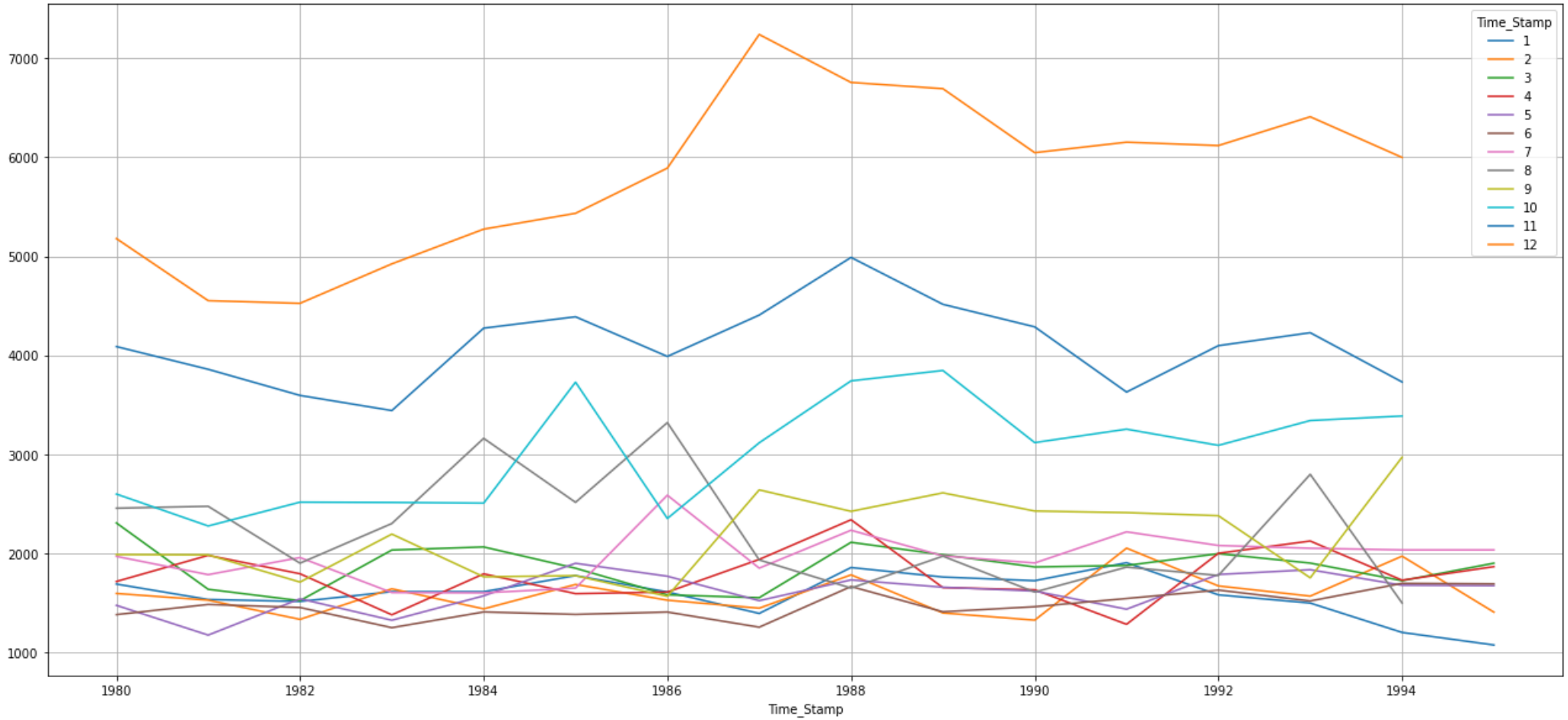
Back to Top

Figure 66. Sparkling Wine Sales



Back to Top

Figure 67. Monthly Sales: Line Graph



Back to Top

Table 12 Monthly Sales, January 1980 - June 1995

Time_Stamp	1	2	3	4	5	6	7	8	9	10	11	12
Time_Stamp												
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

Figure 68. Yearly Sum Plot

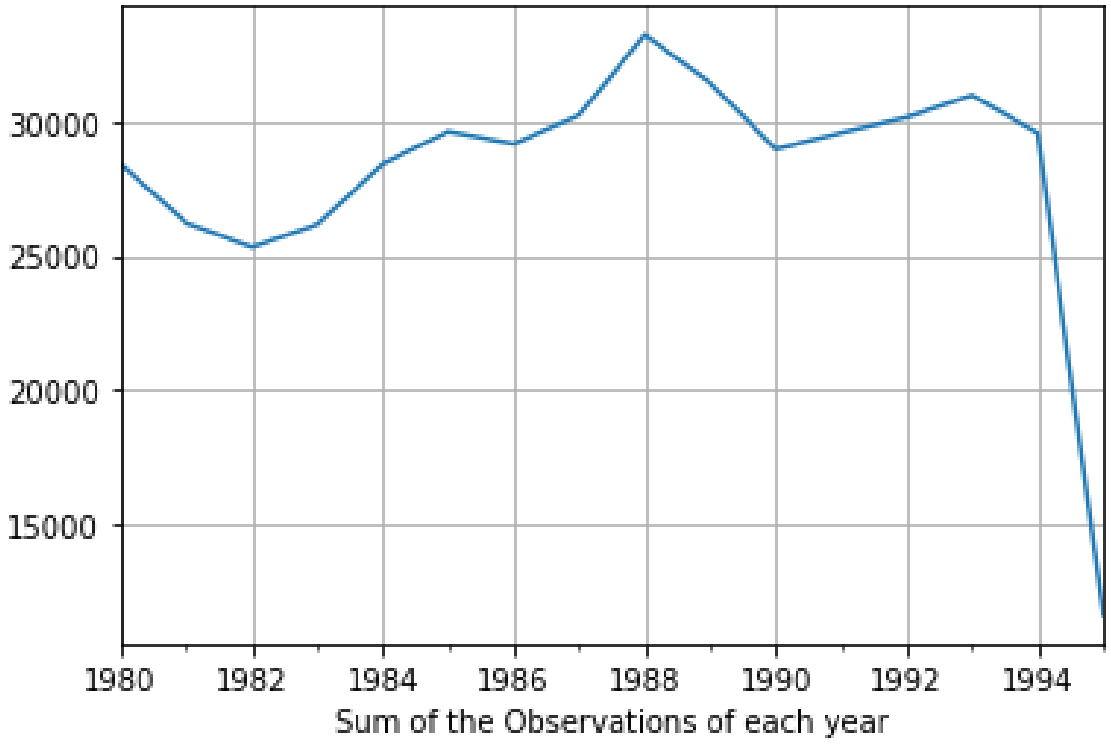


Figure 69. Yearly Mean Plot

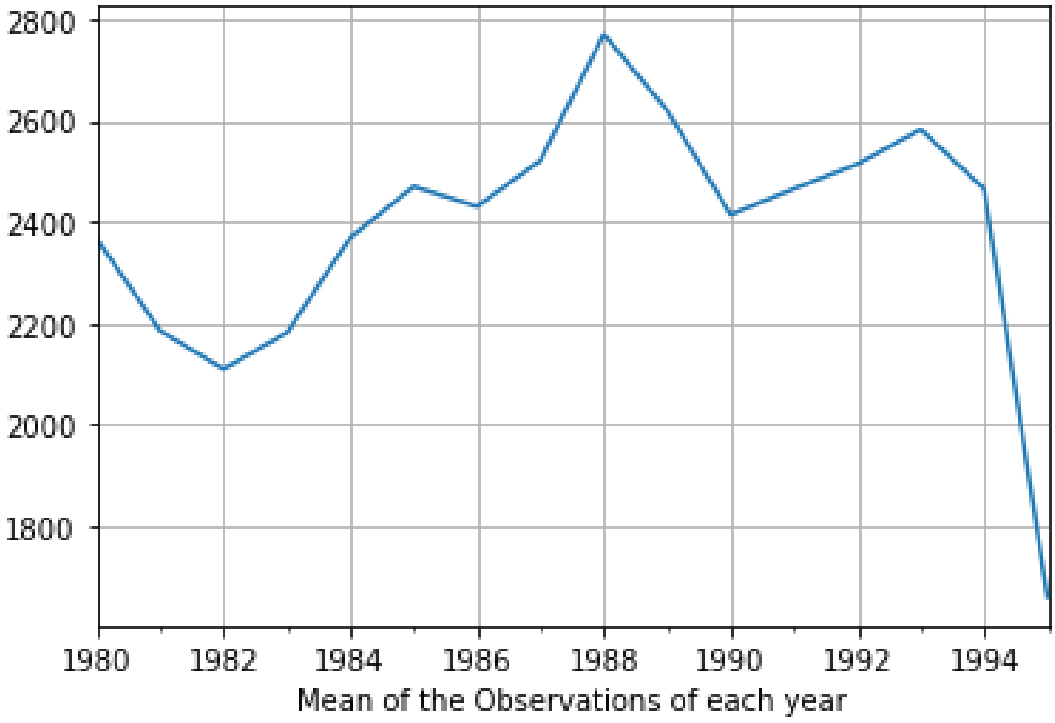
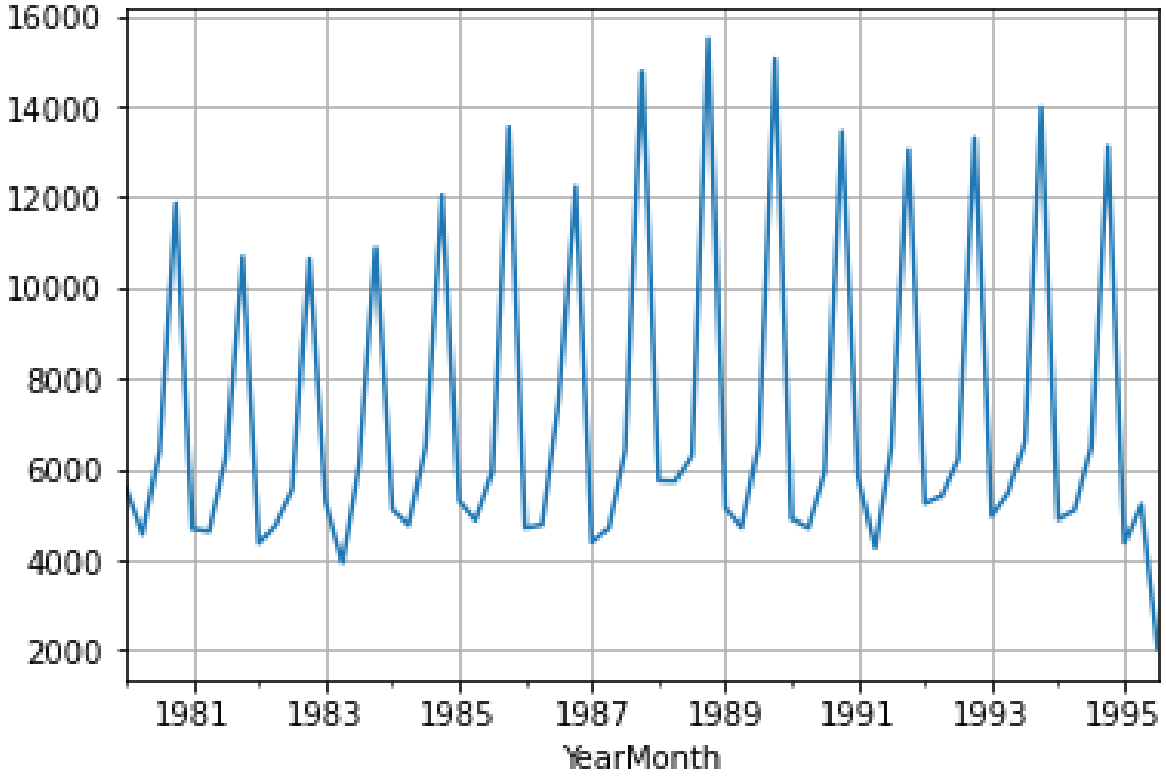


Figure 70. Quarterly Sum Plot



Back to Top

Figure 71. Quarterly Mean Plot

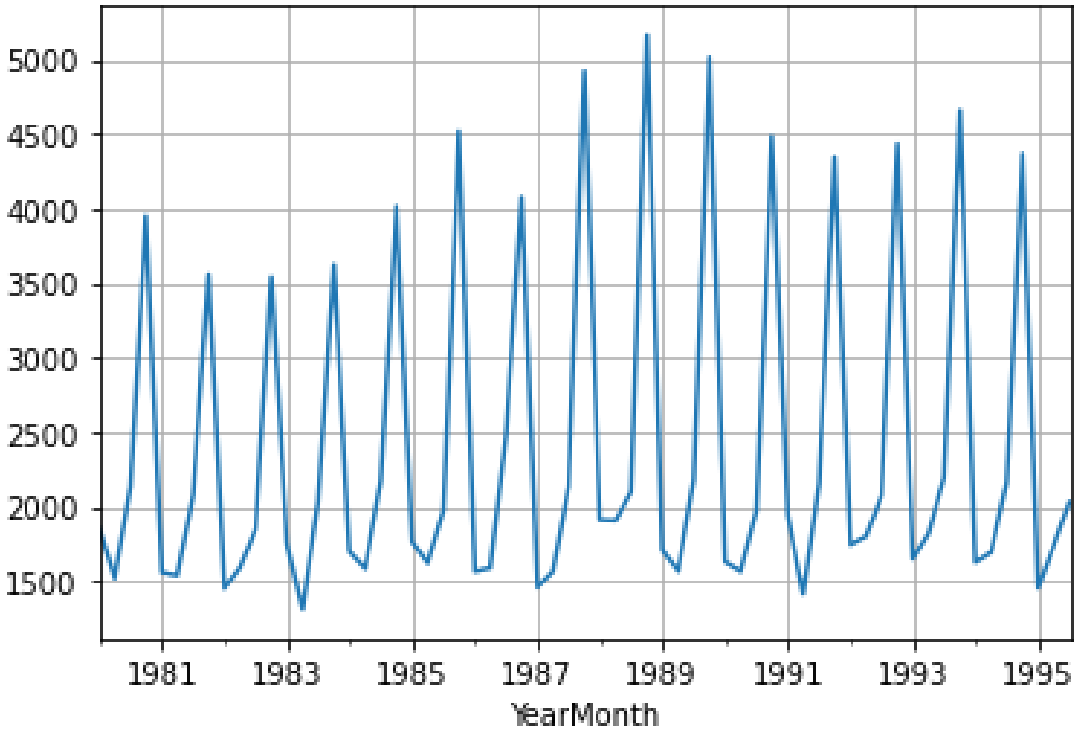
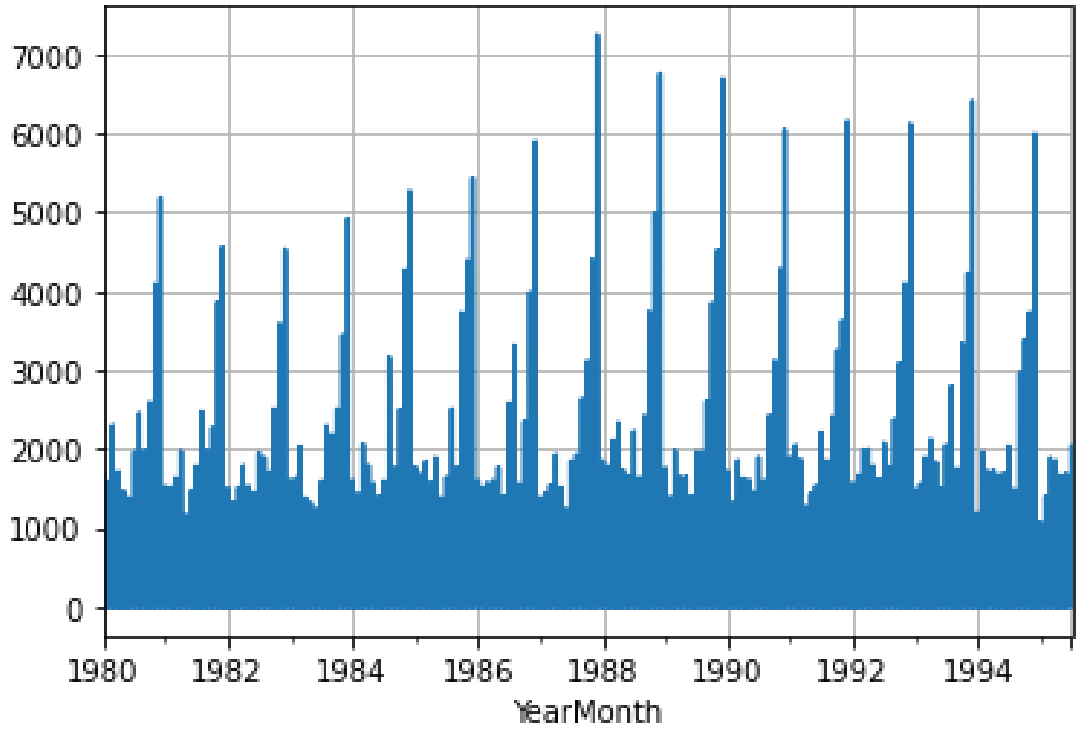
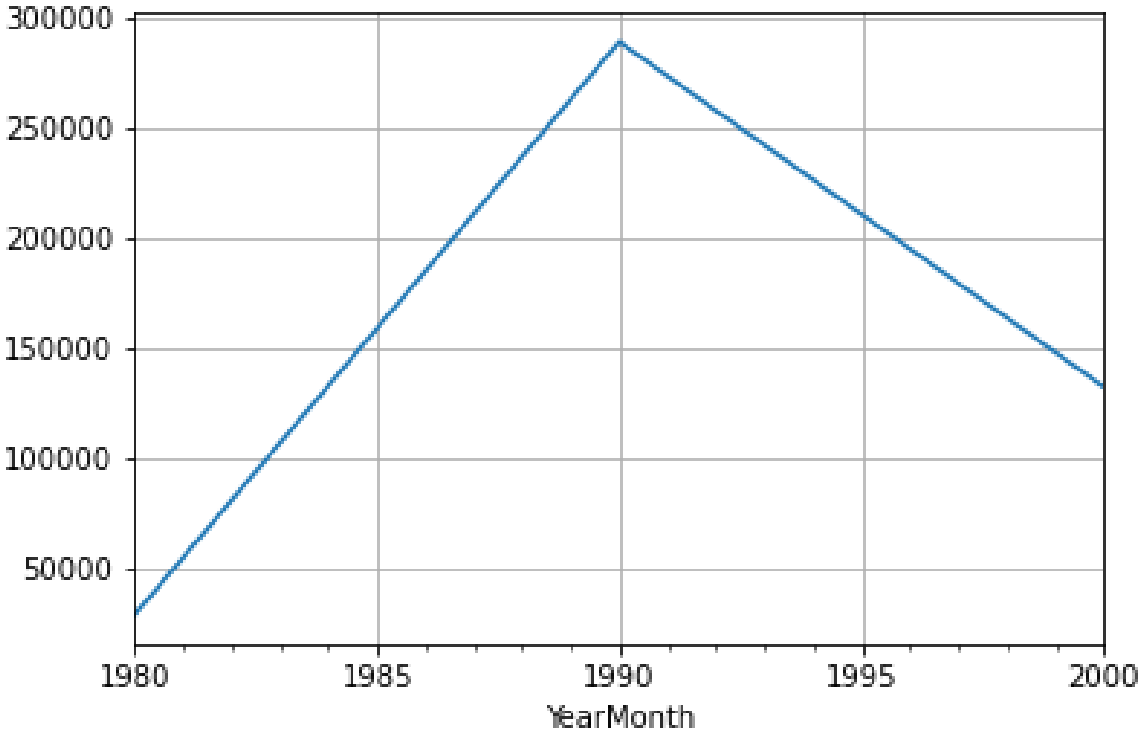


Figure 72. Daily Sum Plot



Back to Top

Figure 73. Decade Sum Plot



Back to Top

Figure 74. Data Decomposition

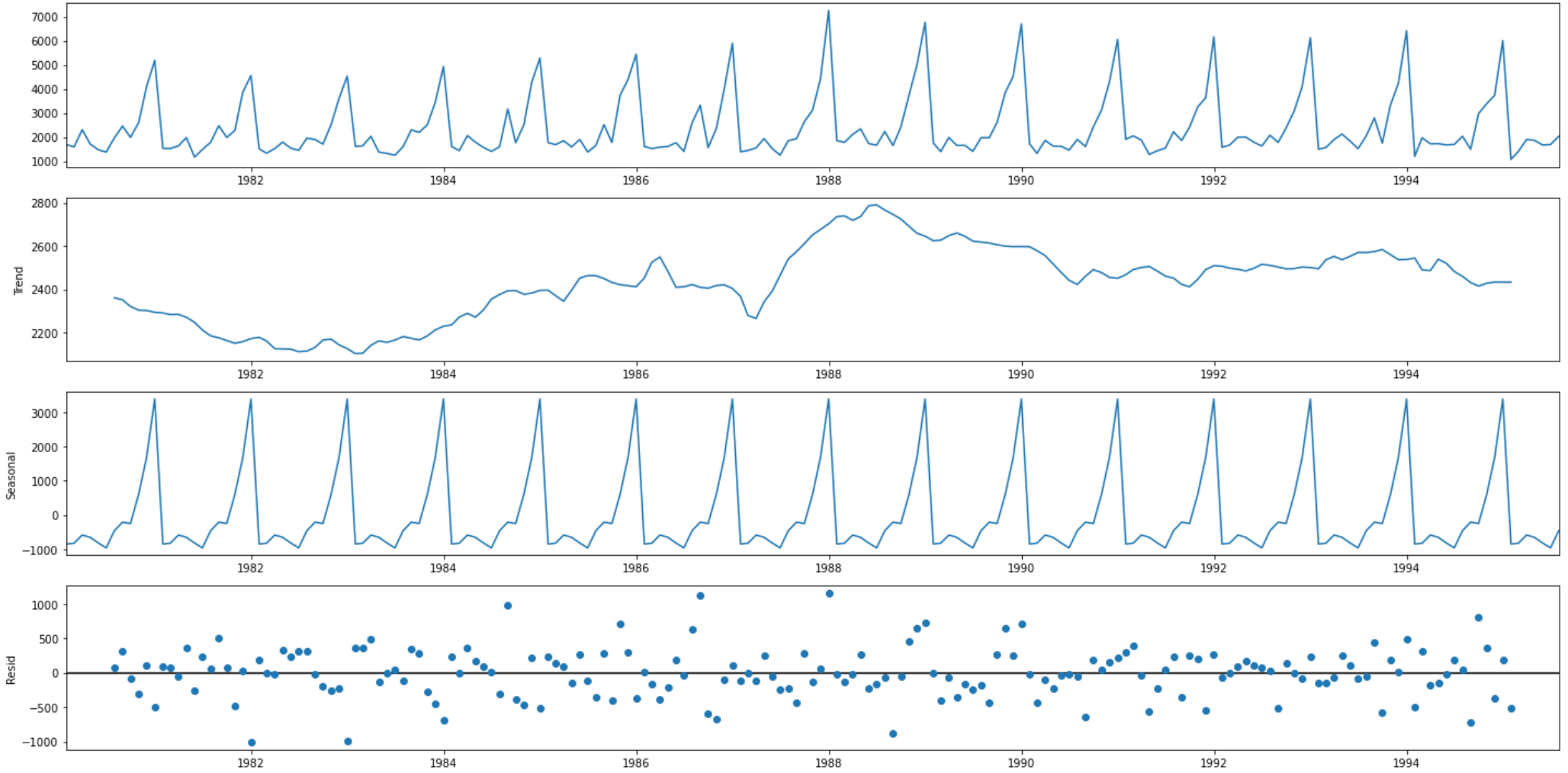
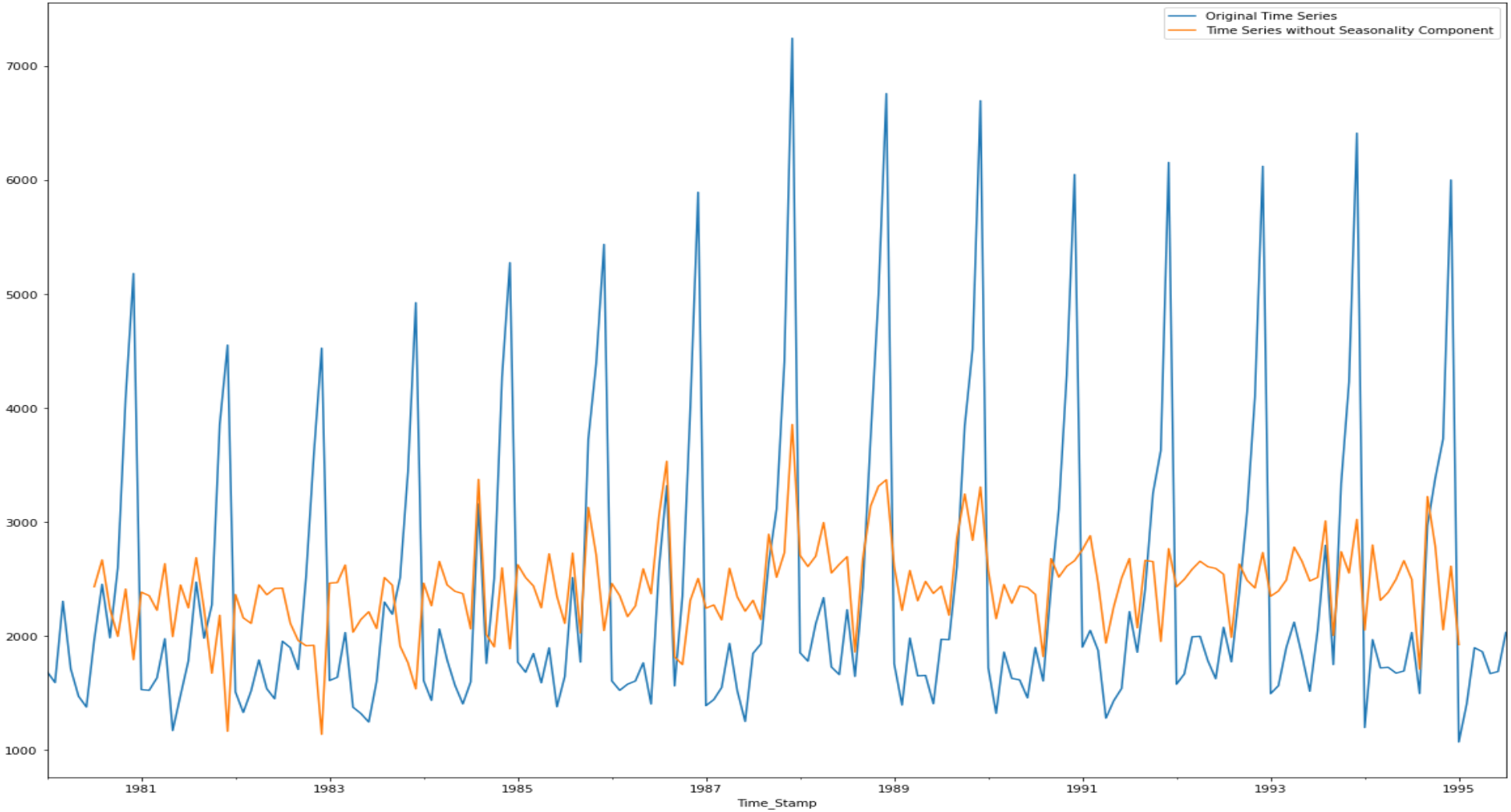


Figure 75. Original Time Series vs. Time Series without Seasonality Component



Back to Top

Table 13 Trend Table (with head function)

```
Trend
 Time_Stamp
1980-01-31           NaN
1980-02-29           NaN
1980-03-31           NaN
1980-04-30           NaN
1980-05-31           NaN
1980-06-30           NaN
1980-07-31    2360.666667
1980-08-31    2351.333333
1980-09-30    2320.541667
1980-10-31    2303.583333
1980-11-30    2302.041667
1980-12-31    2293.791667
Name: trend, dtype: float64
```

Table 14 Seasonality Table (with head function)

```
Seasonality
 Time_Stamp
1980-01-31    -854.260599
1980-02-29    -830.350678
1980-03-31    -592.356630
1980-04-30    -658.490559
1980-05-31    -824.416154
1980-06-30    -967.434011
1980-07-31    -465.502265
1980-08-31    -214.332821
1980-09-30    -254.677265
1980-10-31     599.769957
1980-11-30    1675.067179
1980-12-31    3386.983846
Name: seasonal, dtype: float64
```

Table 15 Residual Table (with head function)

```
Residual
 Time_Stamp
1980-01-31          NaN
1980-02-29          NaN
1980-03-31          NaN
1980-04-30          NaN
1980-05-31          NaN
1980-06-30          NaN
1980-07-31    70.835599
1980-08-31   315.999487
1980-09-30   -81.864401
1980-10-31  -307.353290
1980-11-30   109.891154
1980-12-31  -501.775513
Name: resid, dtype: float64
```

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

We have split the data from 1980 to 1990 into the train dataset and 1991 to June 1995 into the test dataset.

Figure 76. Train and Test Dataset (Rows and Column)

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

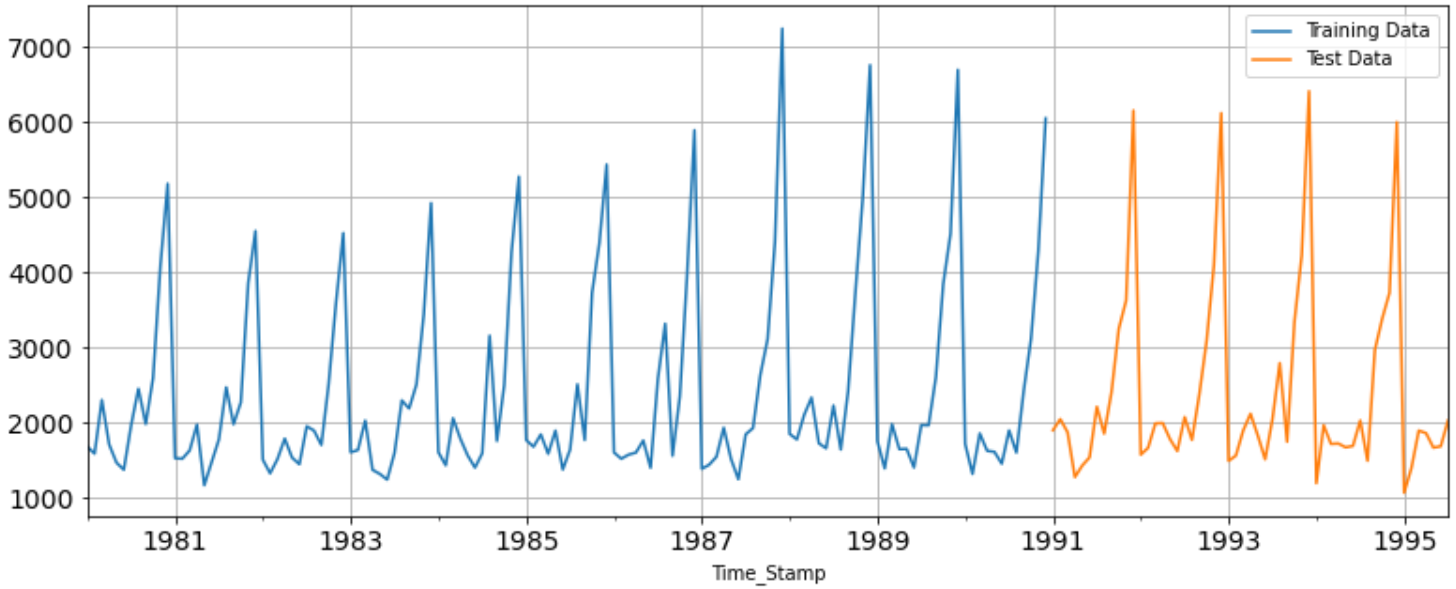
Figure 77. Train Dataset (with head and tail function)

```
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
```

Figure 78. Train Dataset (with head and tail function)

```
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
```

Figure 79. Train & Test Dataset: Plot



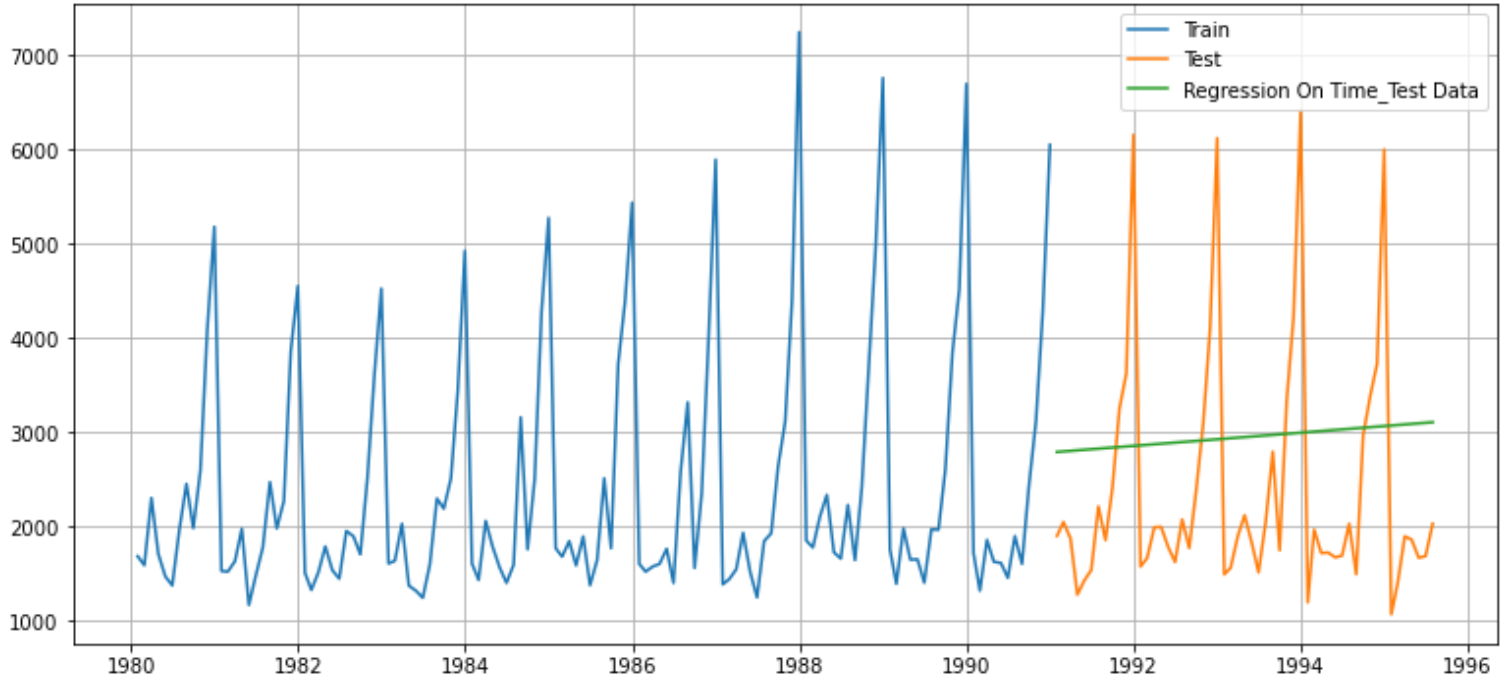
3.4 Build all the exponential smoothing models on the training data and evaluate the model using RMSE on the test data. Other additional models such as regression, naïve forecast models, simple average models, moving average models should also be built on the training data and check the performance on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

3.4.1 Linear Regression Model

Linear Regression: (Please refer to the Jupyter notebook)

- The LinearRegression () function is fit on the train dataset
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

Figure 80. Linear Regression Plot



We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as **1389.135** and created a new dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

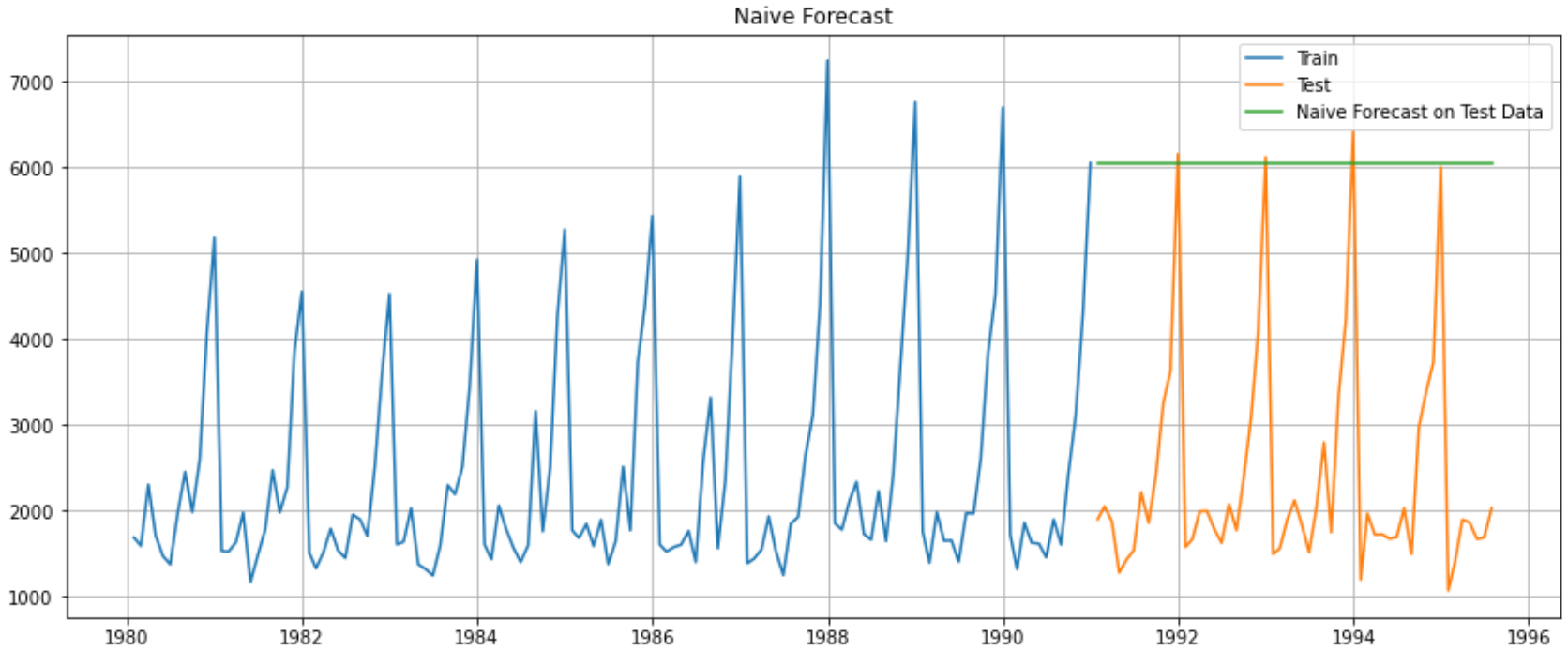
3.4.2 Naïve Model

Naïve: (Please refer to the Jupyter notebook)

- The naïve function is fit on the train dataset
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

Back to Top

Figure 81. Naïve Model Plot



We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as **3864.279** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

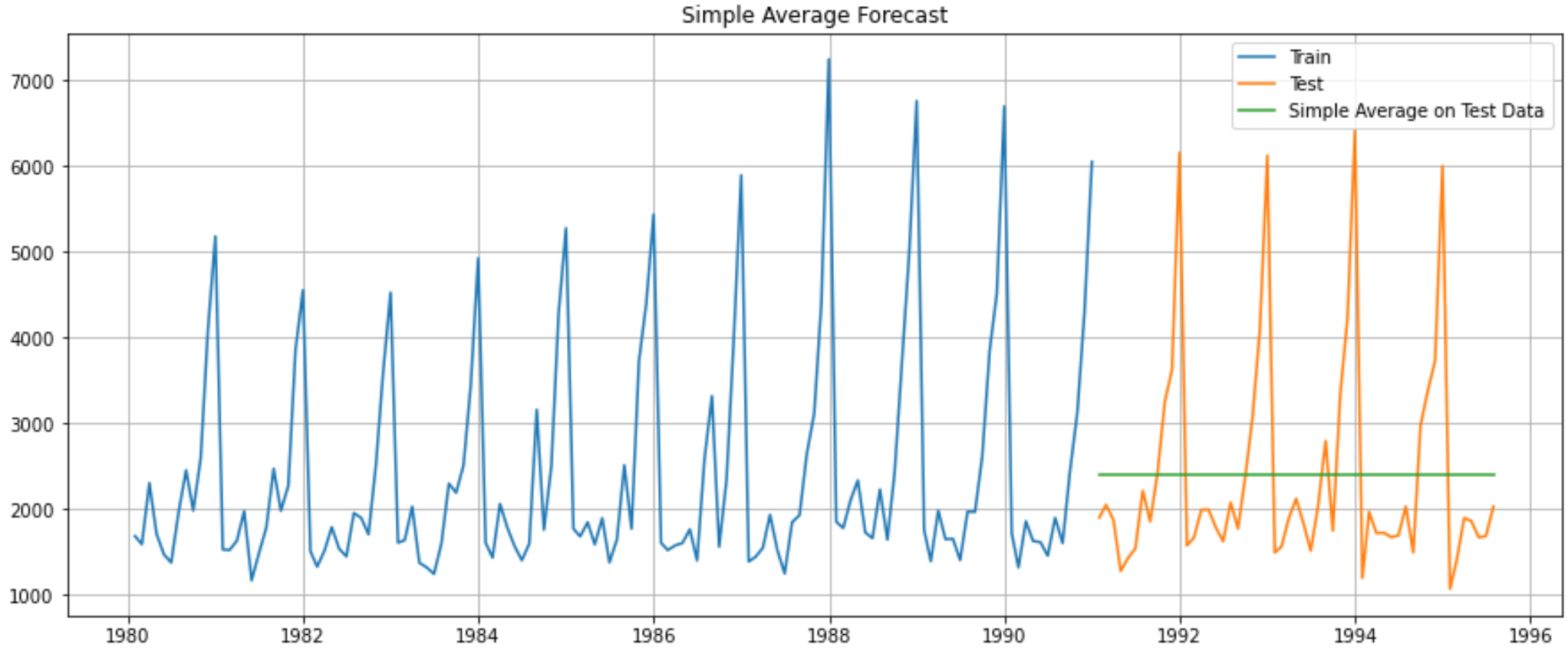
3.4.3 Simple Average Model

Simple Average Model: (Please refer to the Jupyter notebook)

- We take the simple arithmetic mean of the train dataset
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

Back to Top

Figure 82. Simple Average Model Plot



We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as **1275.082** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Back to Top

3.4.4 Moving Average Model

Moving Average Model: (Please refer to the Jupyter notebook)

- We have taken moving average of the train dataset for 2, 4, 6, 9 points
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

Table 16 Moving Average for 2, 4, 6, 9 points (with head function)

Time_Stamp	Sparkling	Trailing_2	Trailing_4	Trailing_6	Trailing_9
1980-01-31	1686	NaN	NaN	NaN	NaN
1980-02-29	1591	1638.5	NaN	NaN	NaN
1980-03-31	2304	1947.5	NaN	NaN	NaN
1980-04-30	1712	2008.0	1823.25	NaN	NaN
1980-05-31	1471	1591.5	1769.50	NaN	NaN

We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as for all the points 2, 4, 6, 9 of **813.401, 1156.590, 1283.927, and 1346.278 respectively** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

We can see that the RMSE score for 2 point moving average is lowest owing to which we will consider as the most accurate predictor of sales for the test dataset time period as also seen in the below plots the accuracy of the prediction.

Figure 83. Moving Average Plot (Train Dataset)

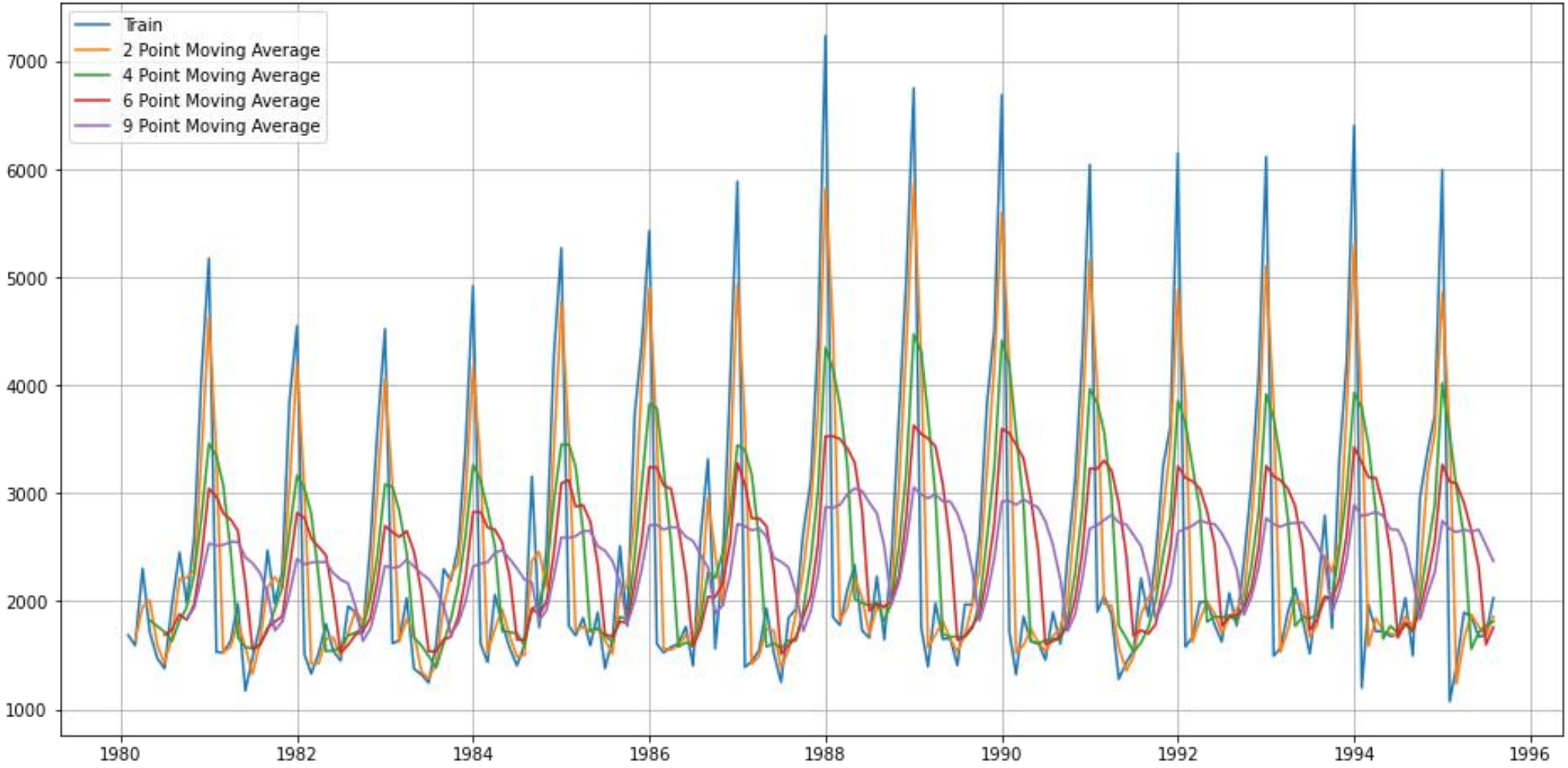
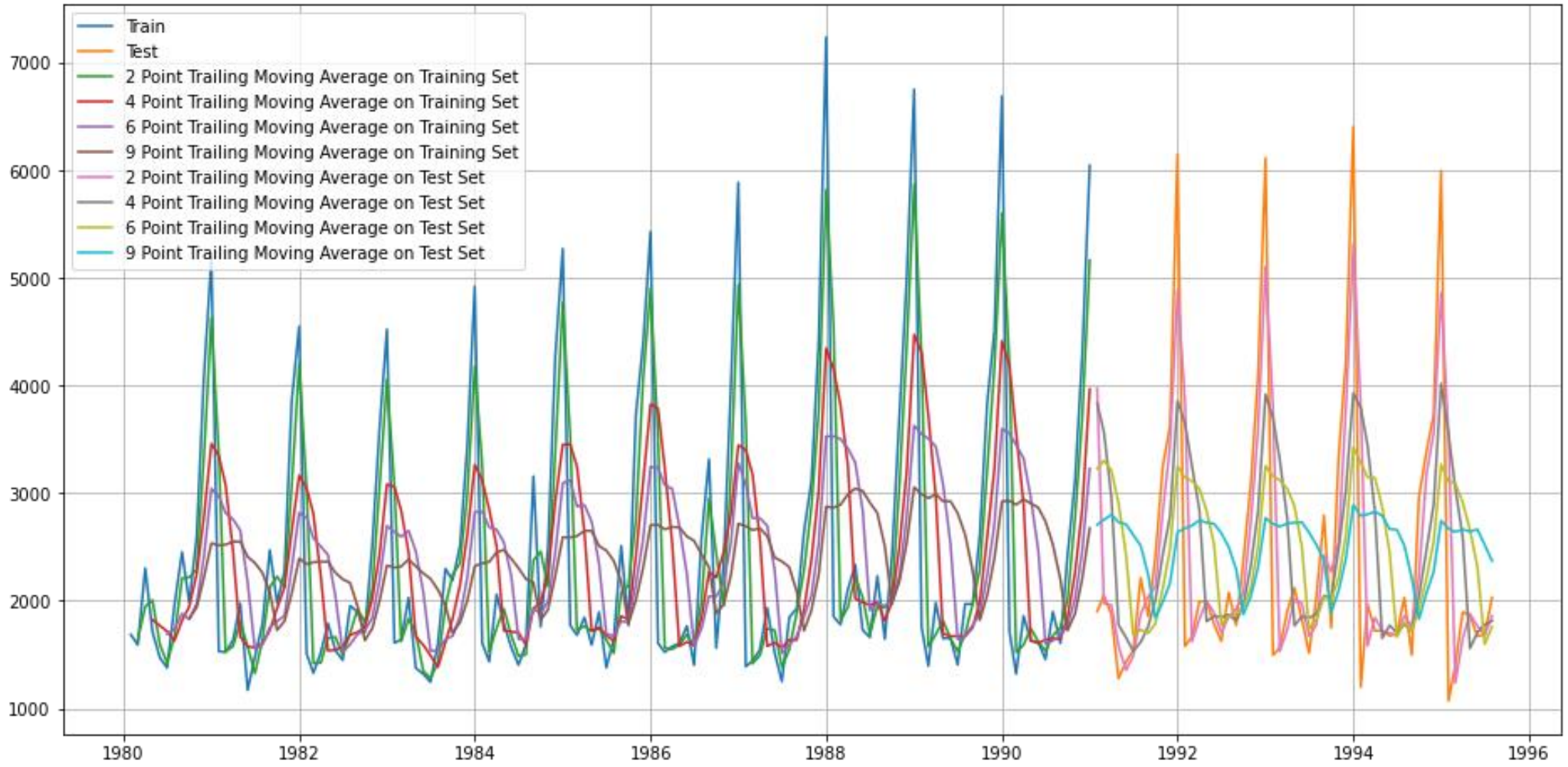
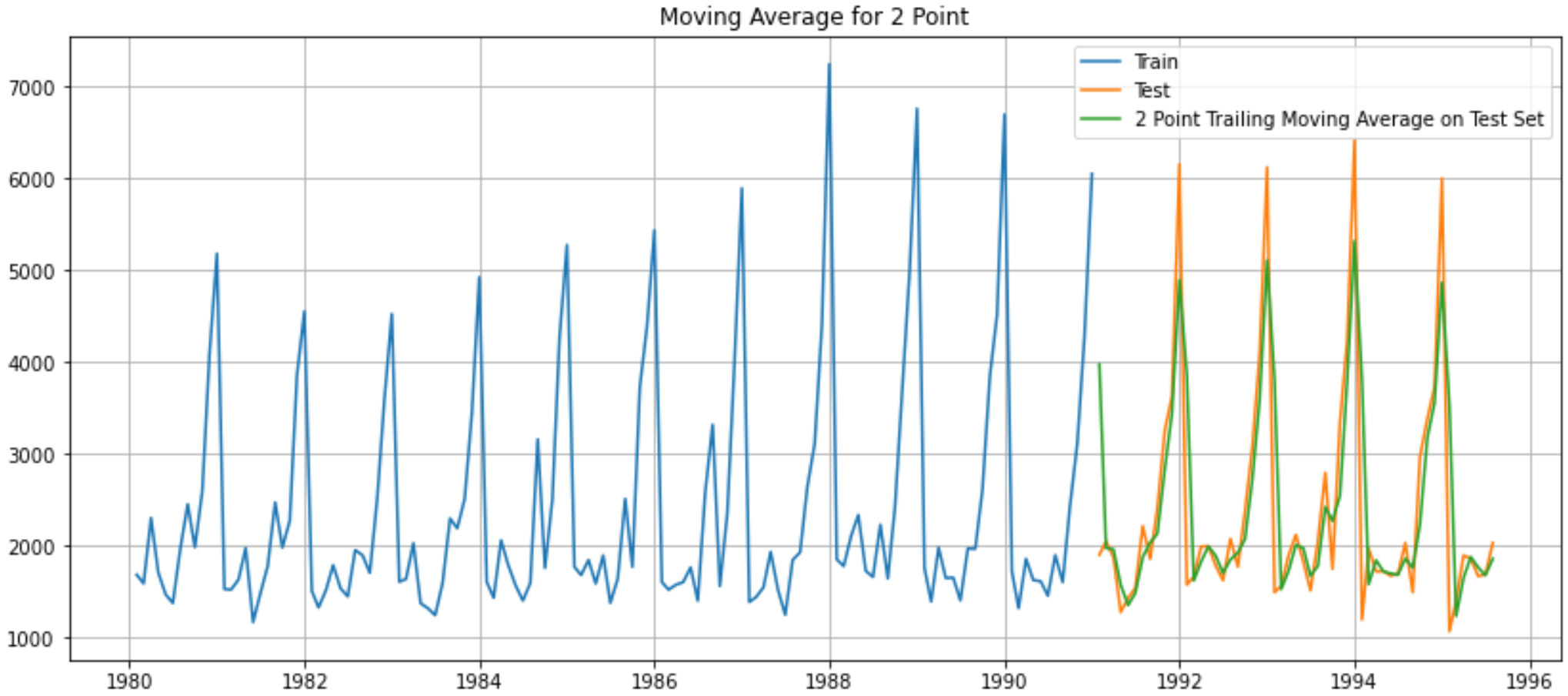


Figure 84. Moving Average Plot (Train & Test Dataset)



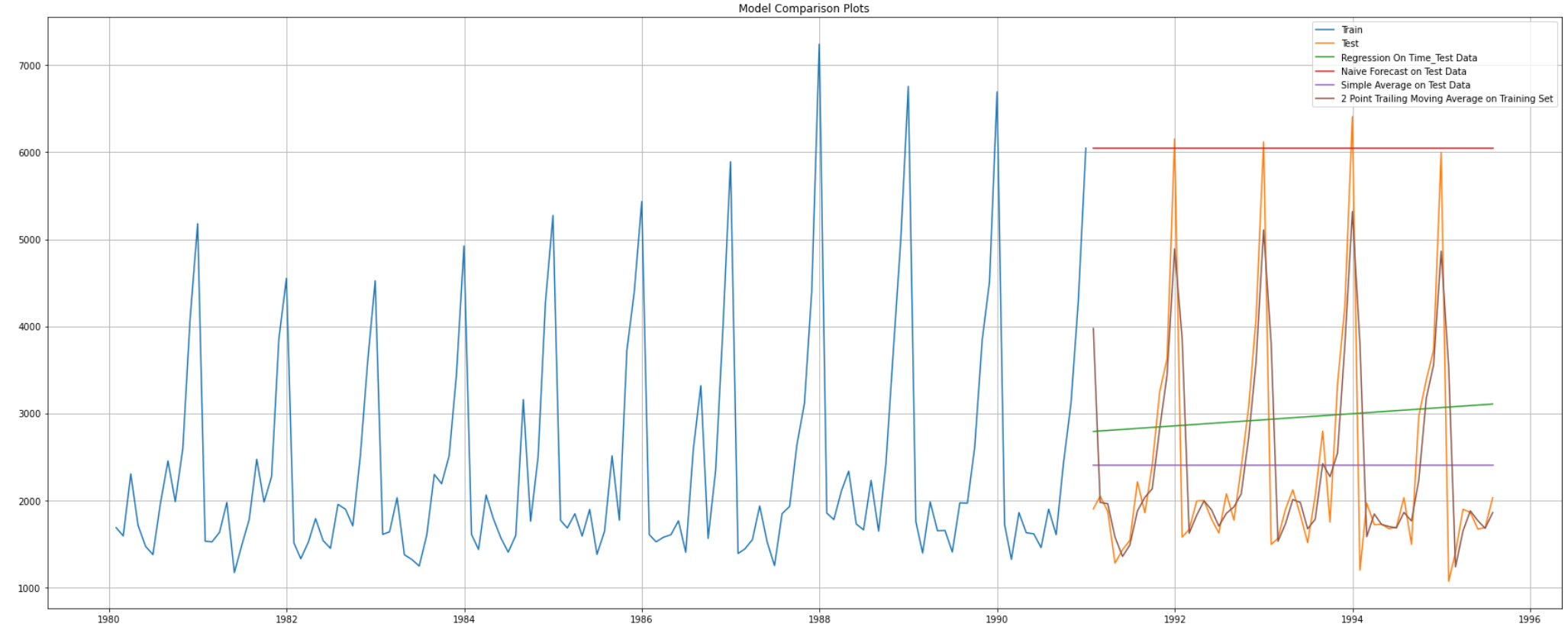
Back to Top

Figure 85. Moving Average Plot (2 Point)



Back to Top

Figure 86. Prediction Plot for Regression, Naïve, Simple Average, and Moving Average (2 Point)



Back to Top

3.4.5 Simple Exponential Smoothing

Simple Exponential Smoothing: (Please refer to the Jupyter notebook)

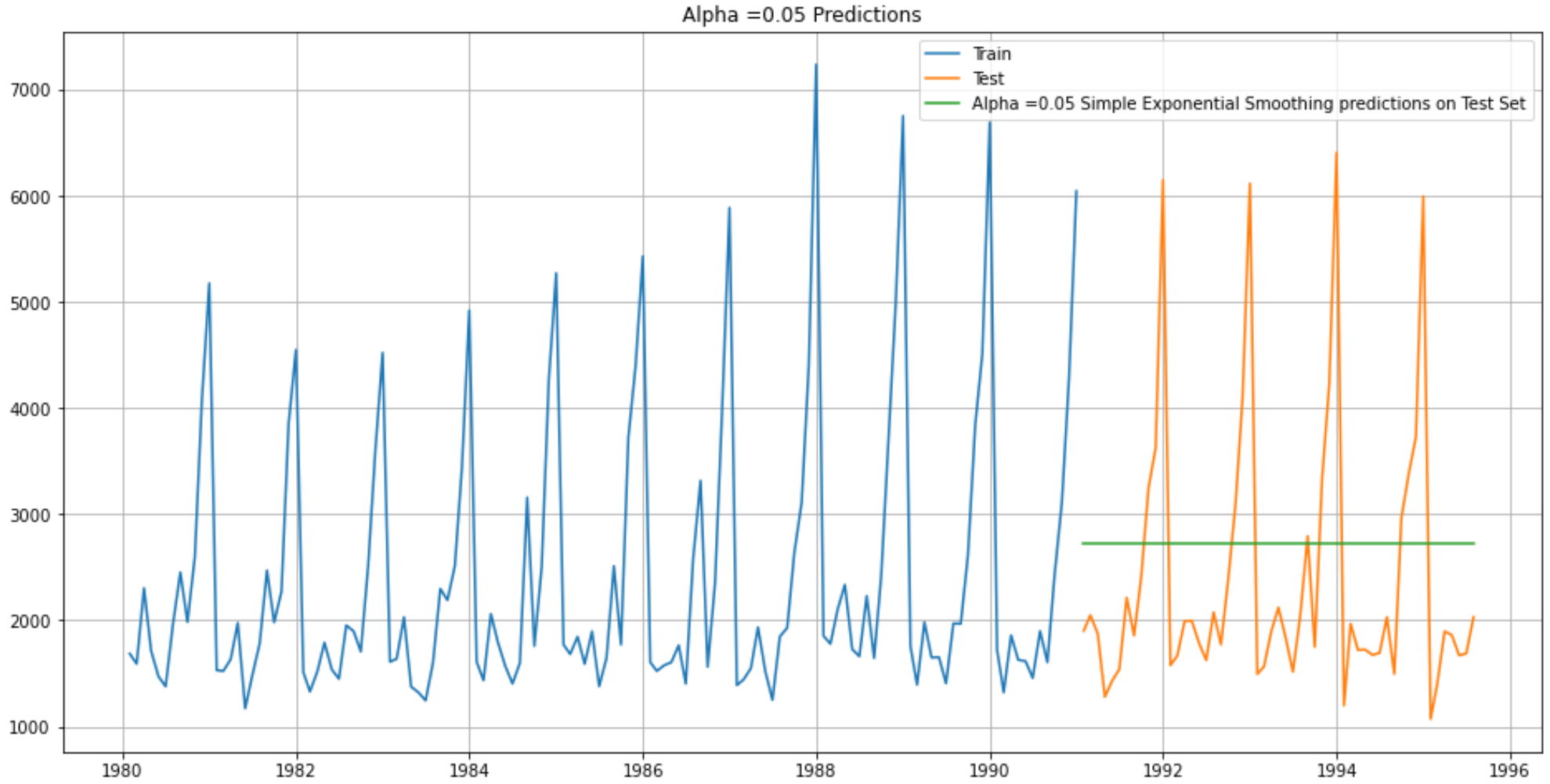
- We have used the SimpleExpSmoothing function and the hyperparameters used for the same is given in the below figure 86
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as **1316.035** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Figure 87. Auto Hyperparameters

```
{'smoothing_level': 0.04960736058186293 6,
 '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}
```

Figure 88. Simple Exponential Smoothing Plot



3.4.6 Double Exponential Smoothing

Double Exponential Smoothing: (Please refer to the Jupyter notebook)

- We have used the Holt's function and the hyperparameters used for the same is given in the below figure 89
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as **5291.880** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Figure 89. Prediction on Test Dataset

```
Time_Stamp
1991-01-31    5401.733026
1991-02-28    5476.005230
1991-03-31    5550.277433
1991-04-30    5624.549637
1991-05-31    5698.821840
Name: predict, dtype: float64
```

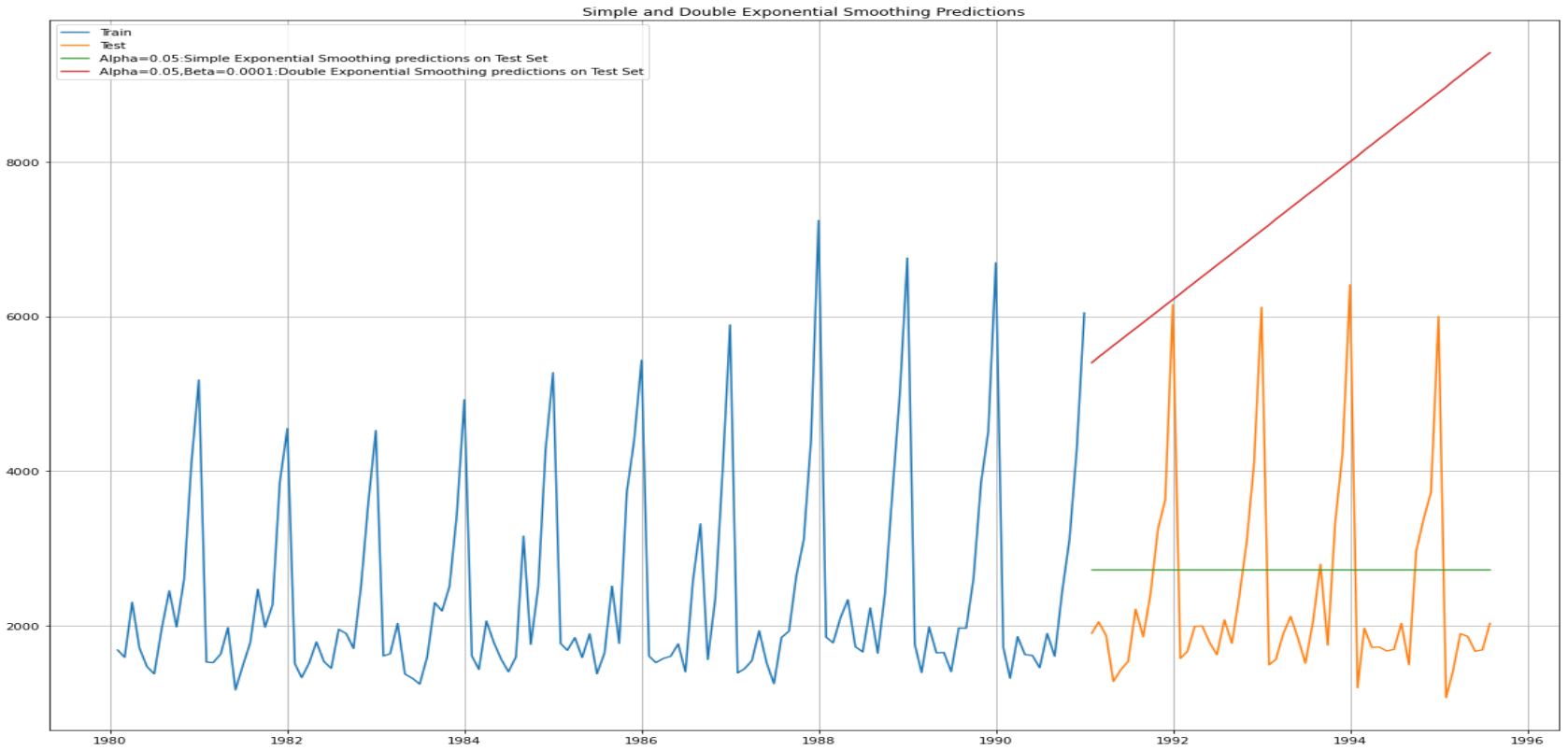
While plotting the graph for the sparkling dataset, we could see that there was no trend in the data. As a result, we can say that double exponential smoothing will not be a good predictor; however, just to observe the result, we have built a model using double exponential smoothing on the training dataset.

We can also see that the calculated RMSE is very high, which can also help us understand that this model is not of any help for us and we cannot rely on the accuracy of the model built using the same smoothing function.

Figure 90. Holt's Hyperparameters

```
==Holt model Exponential Smoothing Estimated Parameters ==

{'smoothing_level': 0.6649999999999999, 'smoothing_trend': 0.0001, 'smoothing_seasonal': nan, 'damping_trend': nan, 'initial_le
vel': 1502.1999999999991, 'initial_trend': 74.87272727272739, 'initial_seasons': array([], dtype=float64), 'use_boxcox': False,
'lamda': None, 'remove_bias': False}
```

Figure 91. Double & Simple Exponential Smoothing Plot



3.4.7 Triple Exponential Smoothing

Triple Exponential Smoothing: (Please refer to the Jupyter notebook)

- We have used the ExponentialSmoothing function and the hyperparameters used for the same is given in the below figure 91 with trend set as additive and seasonality set as multiplicative
- The training and test datasets are then predicted
- The next step would be to plot the predicted sales on test dataset:

We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) as **469.768** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

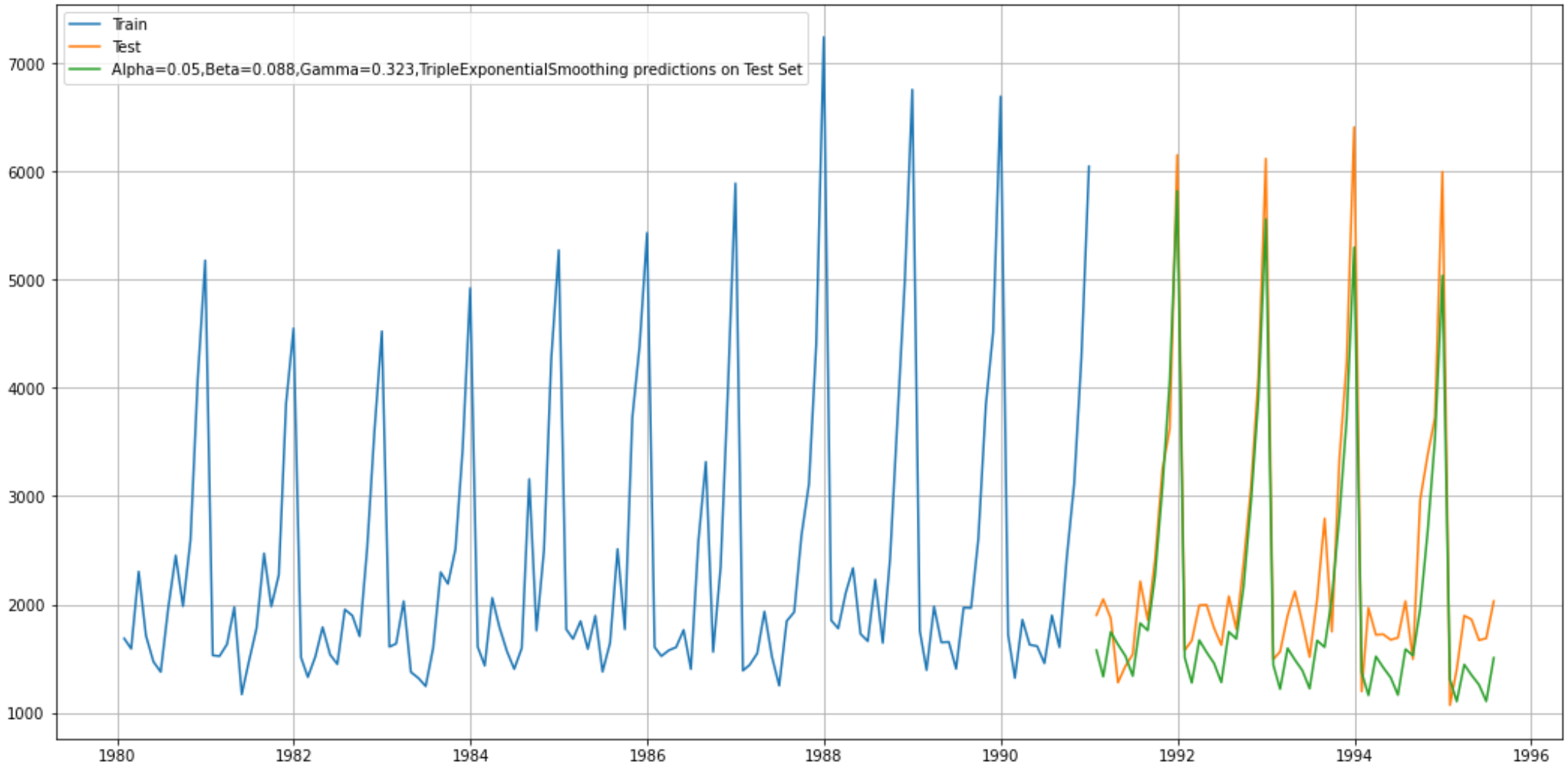
Figure 92. Auto Hyperparameters

```
{'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}
```

Figure 93. Auto Prediction on Test Dataset

	Sparkling	auto_predict
Time_Stamp		
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

Figure 94. Triple Exponential Smoothing Auto Plot



We also given hyperparameters manually for the following variables,

- Smoothing Level: 0.3, 1.1, 0.1
- Smoothing Trend: 0.3, 1.1, 0.1
- Smoothing Seasonality: 0.3, 1.1, 0.1

We run these aforementioned iterations and get multiple combinations with RMSE scores on train and test datasets. We have got 512 combinations as follows with top and bottom 5 combinations:

Figure 95. Manual Prediction Combinations

	Alpha Values	Beta Values	Gamma Values	Train RMSE	Test RMSE
0	0.3	0.3	0.3	404.513320	3.927862e+02
1	0.3	0.3	0.4	402.088628	9.513202e+02
2	0.3	0.3	0.5	408.282432	1.470487e+03
3	0.3	0.3	0.6	428.631668	2.181724e+03
4	0.3	0.3	0.7	468.958530	3.513351e+03
...	...	...	...	...	...
507	1.0	1.0	0.6	153394.791826	7.989790e+05
508	1.0	1.0	0.7	94040.964958	1.074413e+06
509	1.0	1.0	0.8	102196.953755	5.010607e+06
510	1.0	1.0	0.9	77924.294413	4.318265e+05
511	1.0	1.0	1.0	239917.432847	1.254280e+05

Figure 96. Combinations with sort function on RMSE

	Alpha Values	Beta Values	Gamma Values	Train RMSE	Test RMSE
0	0.3	0.3	0.3	404.513320	392.786198
8	0.3	0.4	0.3	424.828055	410.854547
65	0.4	0.3	0.4	435.553595	421.409170
296	0.7	0.8	0.3	700.317756	518.188752
130	0.5	0.3	0.5	498.239915	542.175497

We have also calculated the Root Mean Squared Error (RMSE) (Please refer to the Jupyter notebook) for manual hyperparameters **392.786** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Figure 97. Triple Exponential Smoothing Plot with Best Parameters

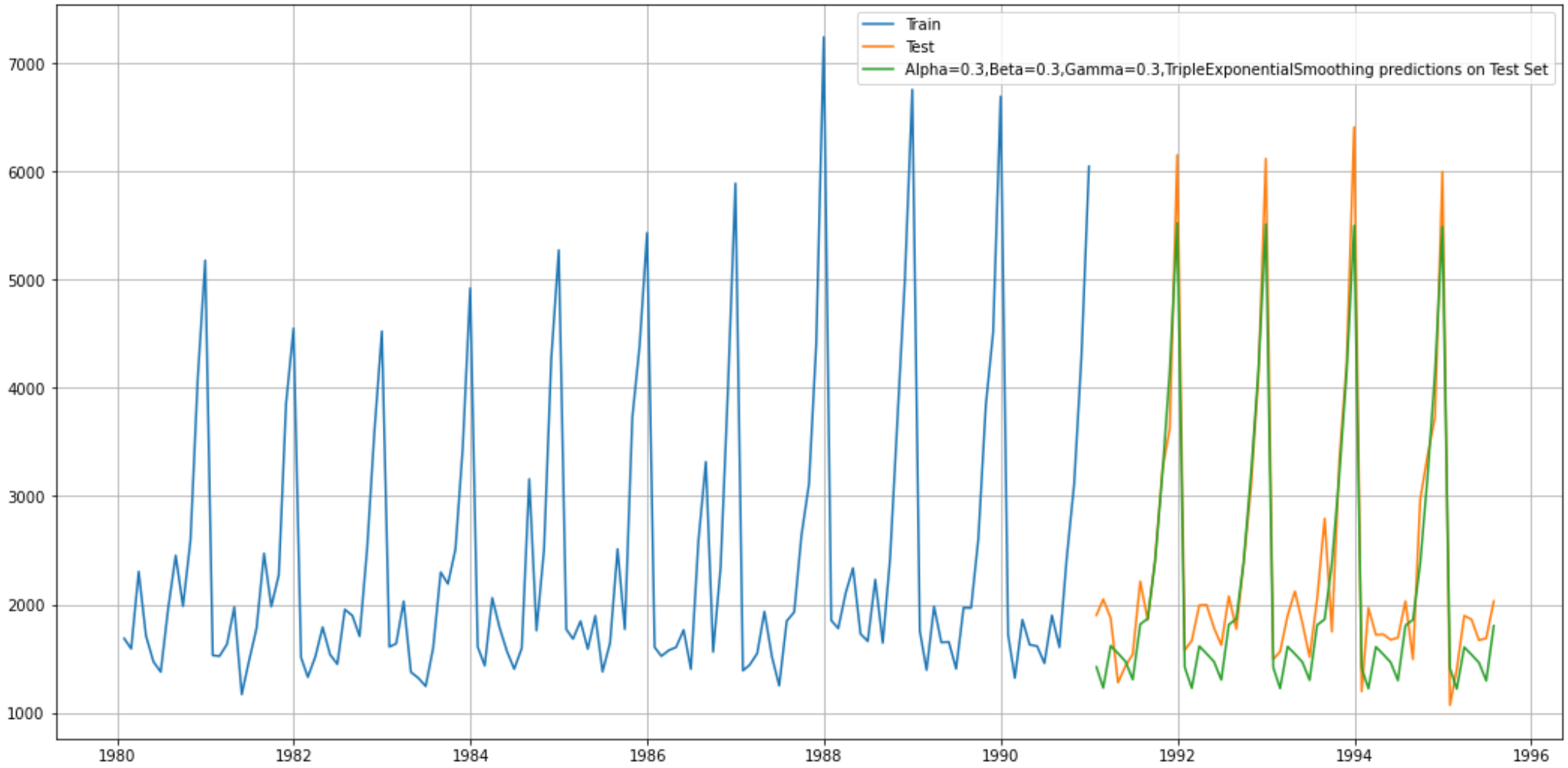
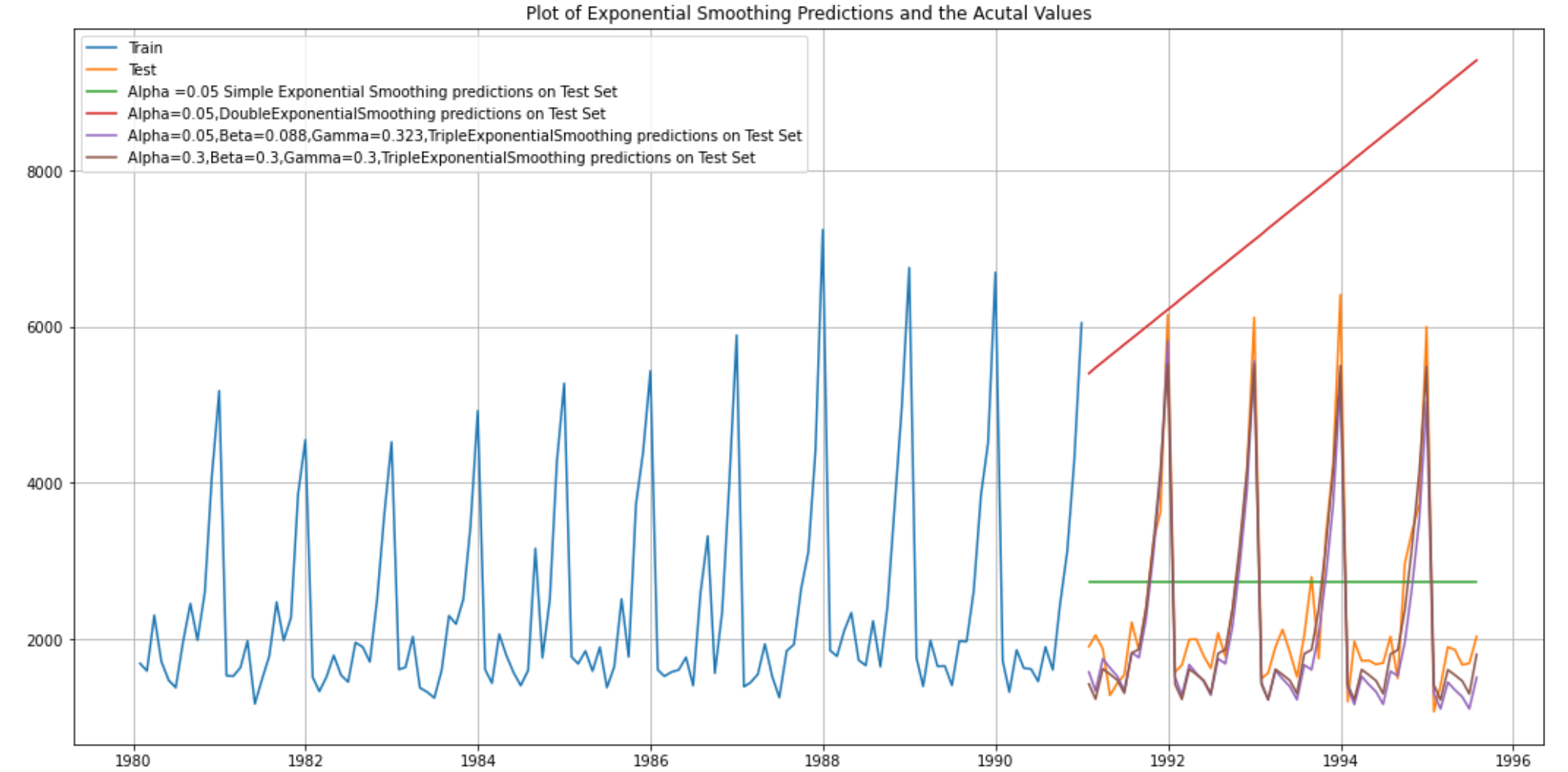


Figure 98. Simple, Double, & Triple Smoothing Plot



Back to Top

3.5 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.

The Augmented Dickey-Fuller test is a unit root test which determines whether there is a unit root and subsequently whether the series is non-stationary.

The hypothesis in a simple form for the ADF test is:

- H_0 : The Time Series has a unit root and is thus non-stationary.
- H_1 : The Time Series does not have a unit root and is thus stationary.

We would want the series to be stationary for building ARIMA models and thus we would want the p-value of this test to be less than the alpha value.

We have used the augmented Dickey Fuller test to check the stationarity of the dataset and we can see that as the p-value is much higher than 0.05 or 95% confidence level as seen below:

Figure 99. ADF Test of Stationarity

```
DF test statistic is -1.798
DF test p-value is 0.7055958459932417
Number of lags used 12
```

Figure 100. ADF Test of Stationarity with difference = 1

```
DF test statistic is -44.912
DF test p-value is 0.0
Number of lags used 10
```

When we take difference as 1, we can see that the data is stationary and there doesn't seem to be any kind of trend involved. As a result, we can use the data for building models such as ARIMA and SARIMA models.

Figure 101. Stationarity Plot

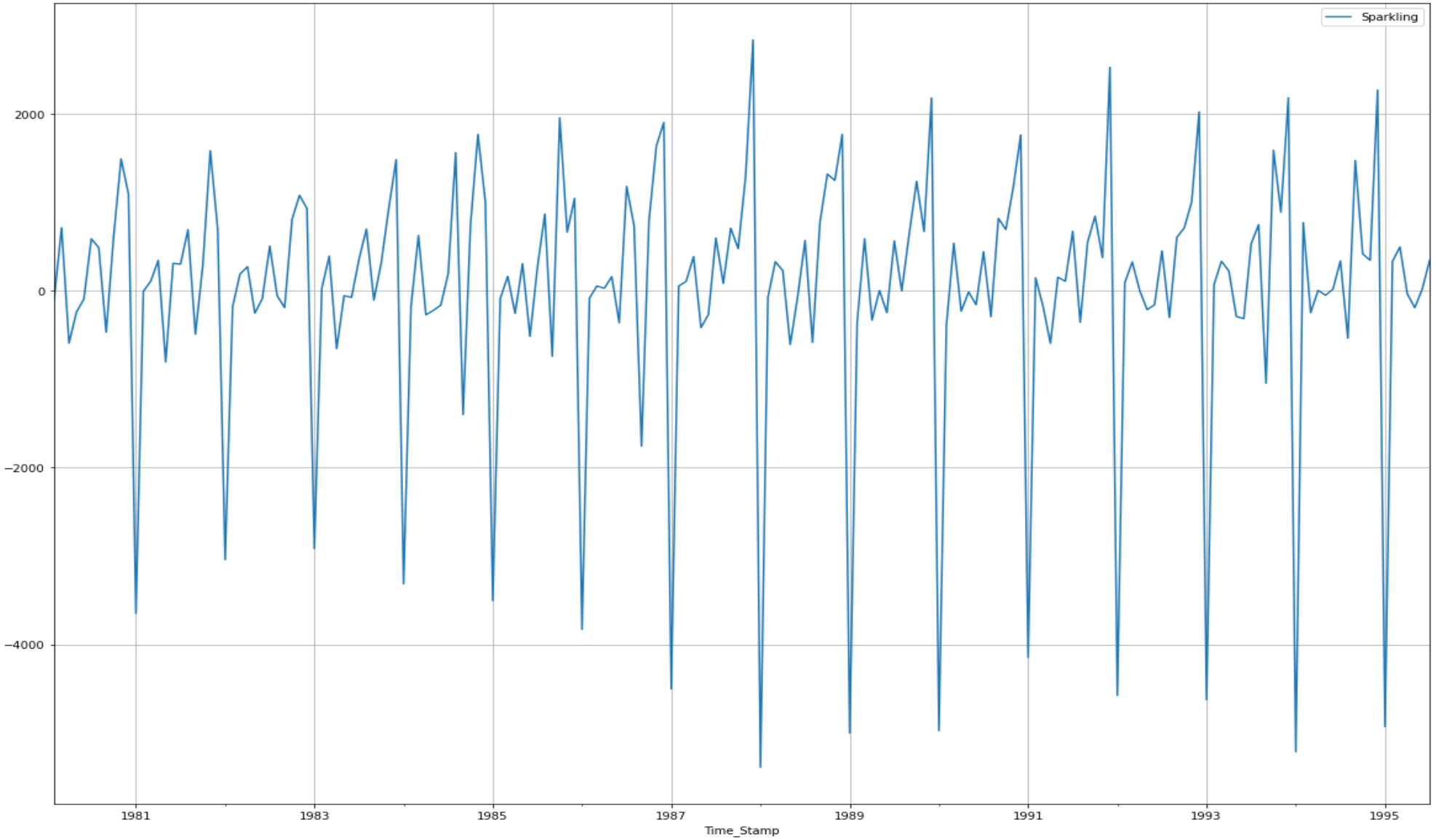
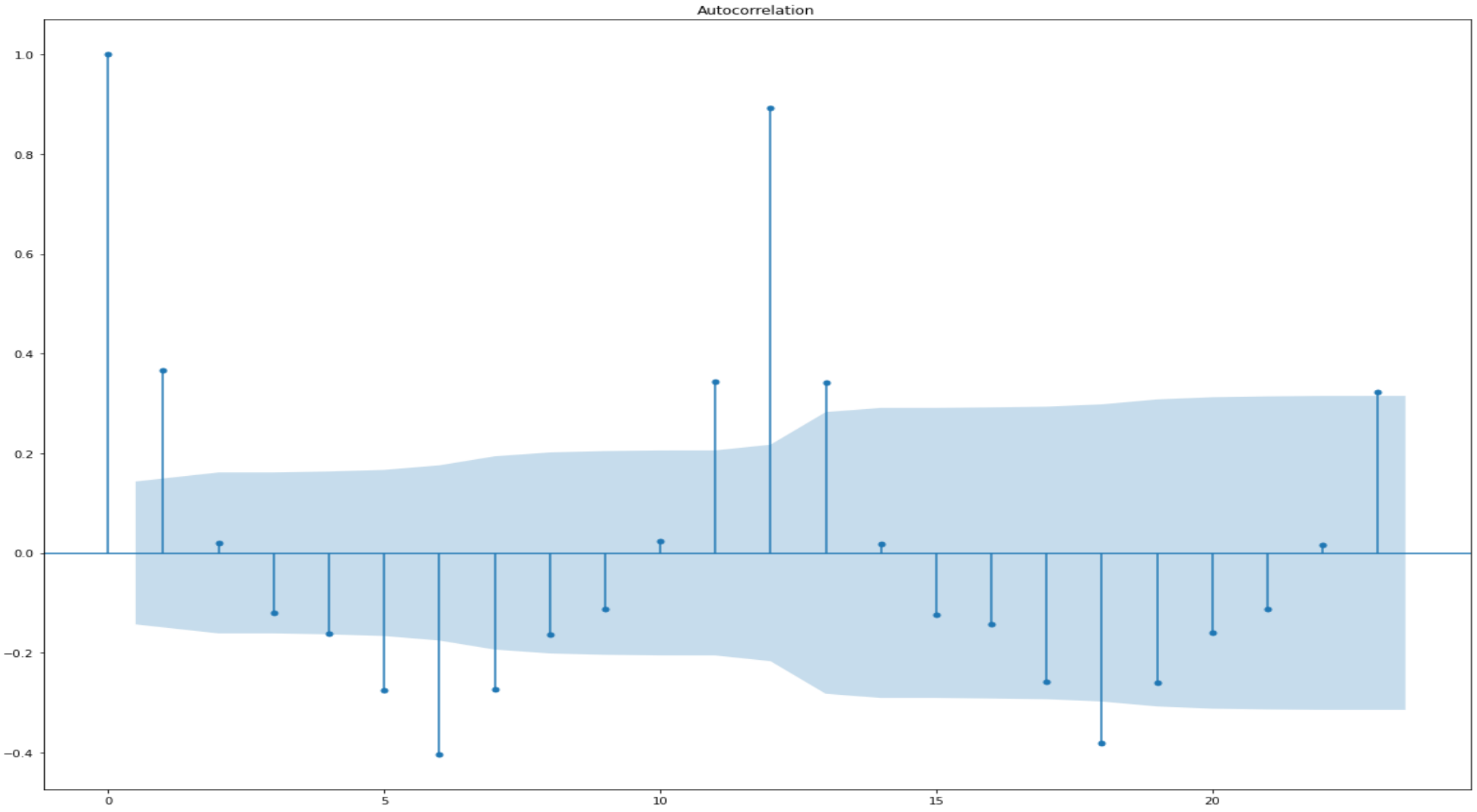
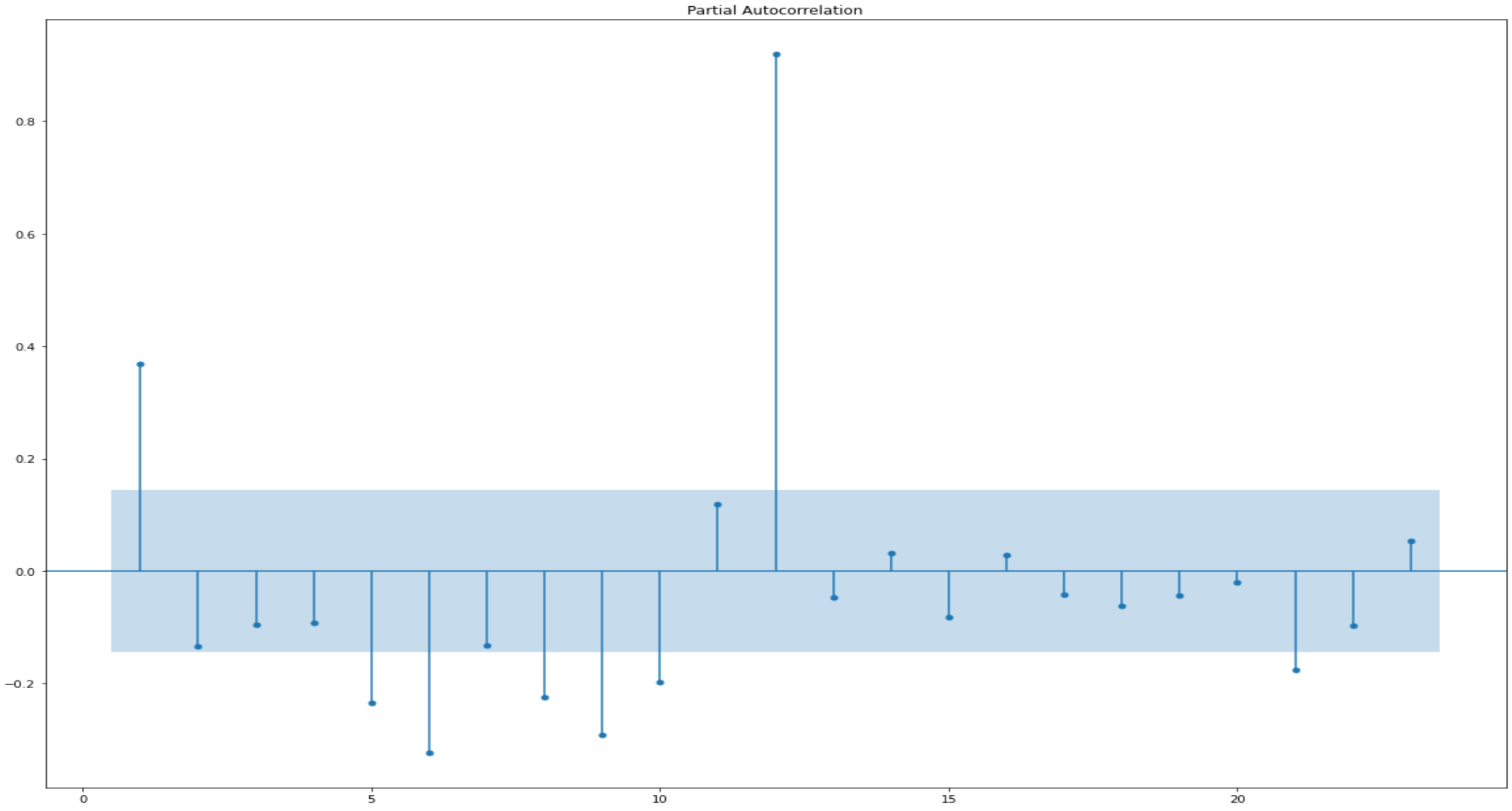


Figure 102. ACF Plot



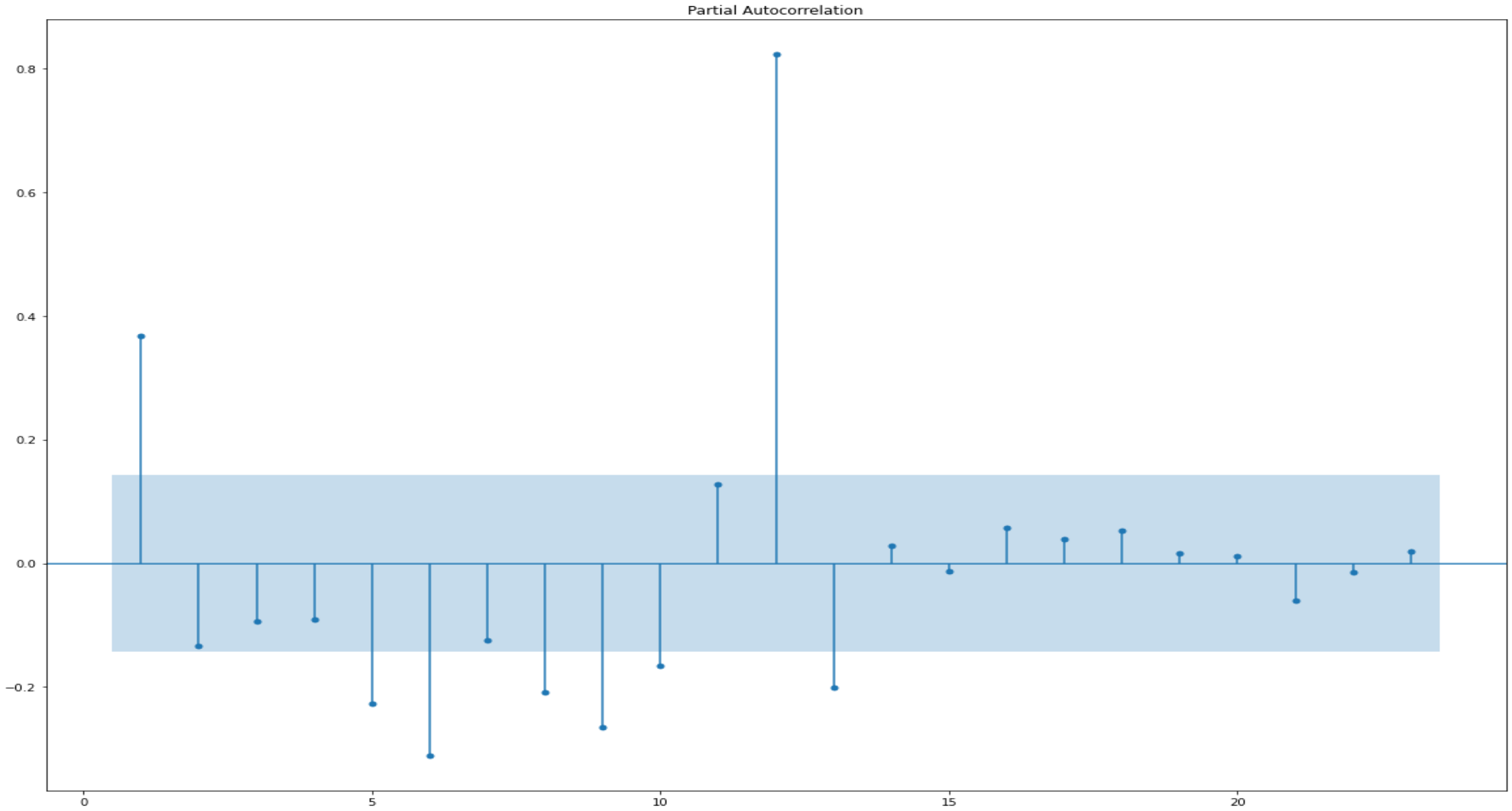
Back to Top

Figure 103. Partial ACF Plot



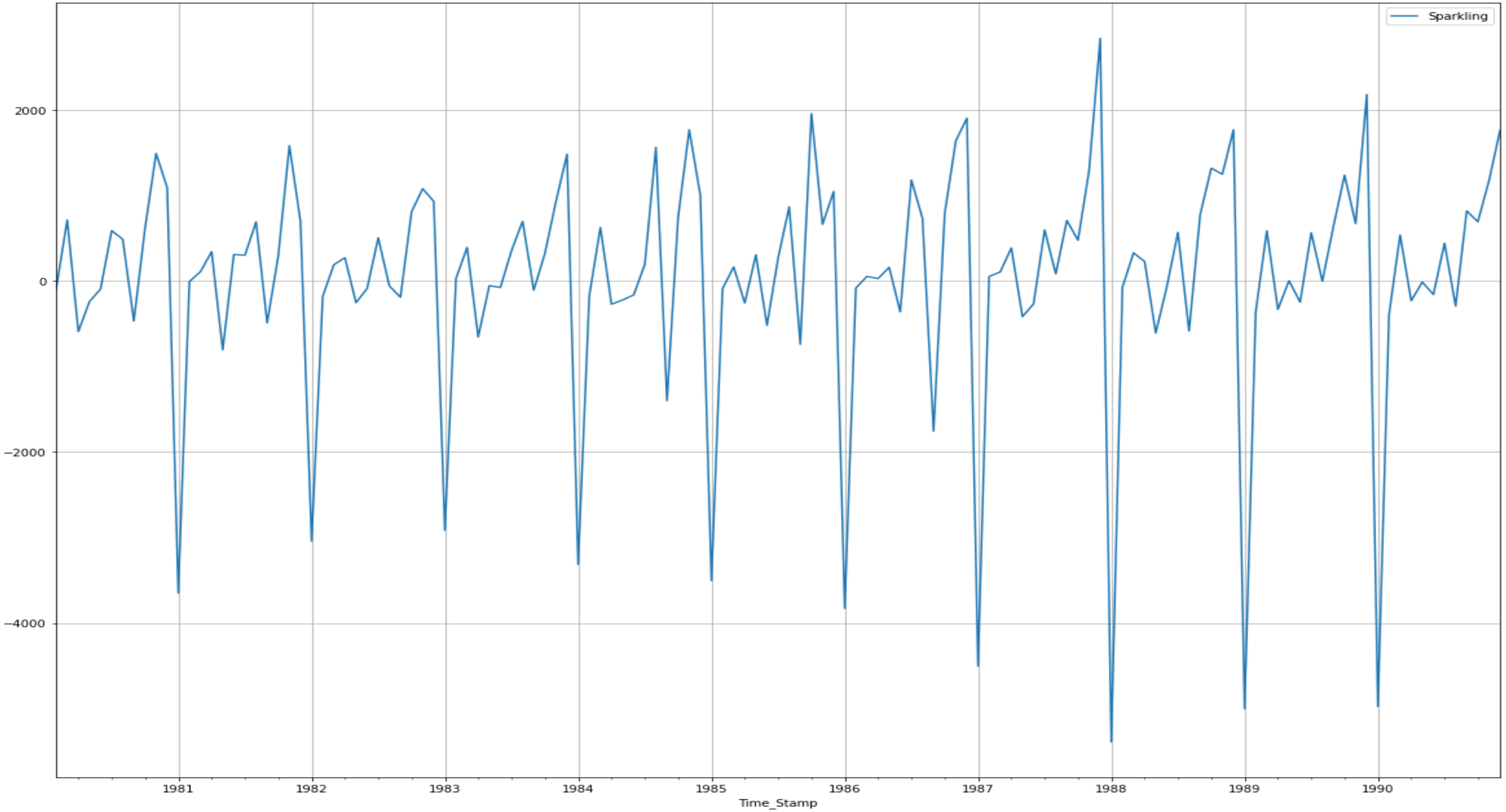
Back to Top

Figure 104. Partial ACF Plot with MLE Method



Back to Top

Figure 105. Train Plot with Stationarity



Back to Top

3.6 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.

3.7 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.

(Note: Combining two questions together 6 & 7)

3.7.1 ARIMA Model

ARIMA Model: (Please refer to the Jupyter notebook)

- We have used iteration tools to try combinations with p, d, q parameters, where p is the number of autoregressive terms, d is the number of nonseasonal differences needed for stationarity, and. q is the number of lagged forecast errors in the prediction equation
- For the above combinations, we have calculated the Akaike Information Criterion (AIC) score to determine which is the best combination with lowest AIC score
- Based on the figure 107, we can see that combination (2,1,2) has the lowest AIC score and we will build a model on the training dataset for the same combination and test the accuracy using by building a summary report, diagnostics, and RMSE score which has come to **1299.980,** and saved it in the dataframe 'results' with the RMSE score to analyze the same for all the models we build.

Figure 106. Example Parameter Combinations

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

Figure 107. Parameters with AIC scores

```
ARIMA(0, 1, 0) - AIC:2267.6630357855465
ARIMA(0, 1, 1) - AIC:2263.060015591336
ARIMA(0, 1, 2) - AIC:2234.408323131676
ARIMA(0, 1, 3) - AIC:2233.9948577793975
ARIMA(1, 1, 0) - AIC:2266.6085393190087
ARIMA(1, 1, 1) - AIC:2235.755094673383
ARIMA(1, 1, 2) - AIC:2234.5272004508324
ARIMA(1, 1, 3) - AIC:2235.6078073353247
ARIMA(2, 1, 0) - AIC:2260.365743968086
ARIMA(2, 1, 1) - AIC:2233.7776263084434
ARIMA(2, 1, 2) - AIC:2213.5092122831566
ARIMA(2, 1, 3) - AIC:2232.9370761971877
ARIMA(3, 1, 0) - AIC:2257.72337899794
ARIMA(3, 1, 1) - AIC:2235.498924009065
ARIMA(3, 1, 2) - AIC:2230.952332594816
ARIMA(3, 1, 3) - AIC:2221.458954306803
```

Figure 108. Parameters with AIC scores (ascending function)

	param	AIC
10	(2, 1, 2)	2213.509212
15	(3, 1, 3)	2221.458954
14	(3, 1, 2)	2230.952333
11	(2, 1, 3)	2232.937076
9	(2, 1, 1)	2233.777626

Figure 109. ARIMA: Summary Report

SARIMAX Results

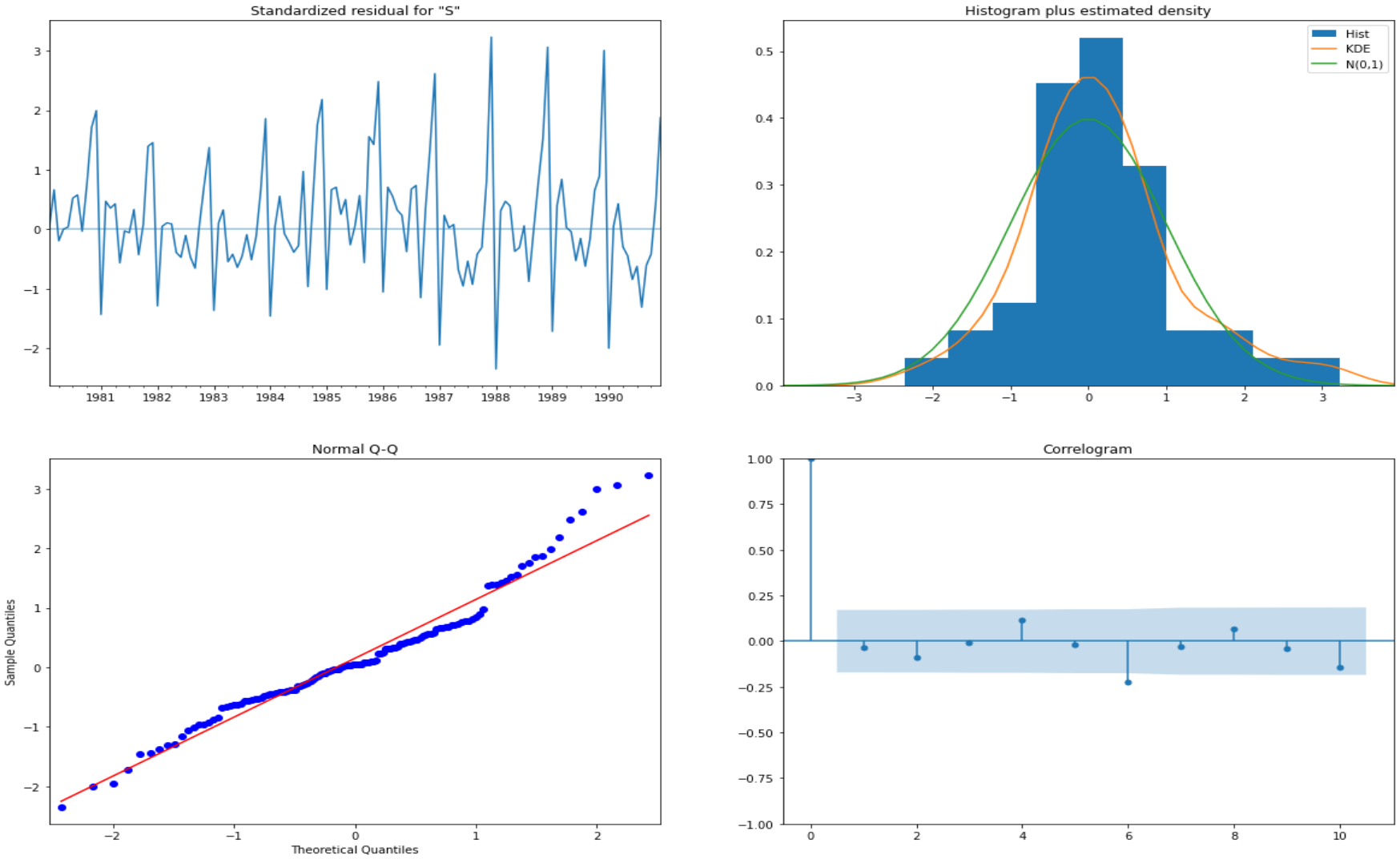
Dep. Variable:	Sparkling	No. Observations:	132
Model:	ARIMA(2, 1, 2)	Log Likelihood	-1101.755
Date:	Fri, 07 Oct 2022	AIC	2213.509
Time:	23:06:42	BIC	2227.885
Sample:	01-31-1980	HQIC	2219.351
	- 12-31-1990		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	1.3121	0.046	28.781	0.000	1.223	1.401
ar.L2	-0.5593	0.072	-7.741	0.000	-0.701	-0.418
ma.L1	-1.9917	0.109	-18.218	0.000	-2.206	-1.777
ma.L2	0.9999	0.110	9.109	0.000	0.785	1.215
sigma2	1.099e+06	1.99e-07	5.51e+12	0.000	1.1e+06	1.1e+06

Ljung-Box (L1) (Q):	0.19	Jarque-Bera (JB):	14.46
Prob(Q):	0.67	Prob(JB):	0.00
Heteroskedasticity (H):	2.43	Skew:	0.61
Prob(H) (two-sided):	0.00	Kurtosis:	4.08

Back to Top

Figure 110. ARIMA: Diagnostics



We use Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) to figure out the order of ARIMA model:

Figure 111. ARIMA Model: Autocorrelation Function (ACF)

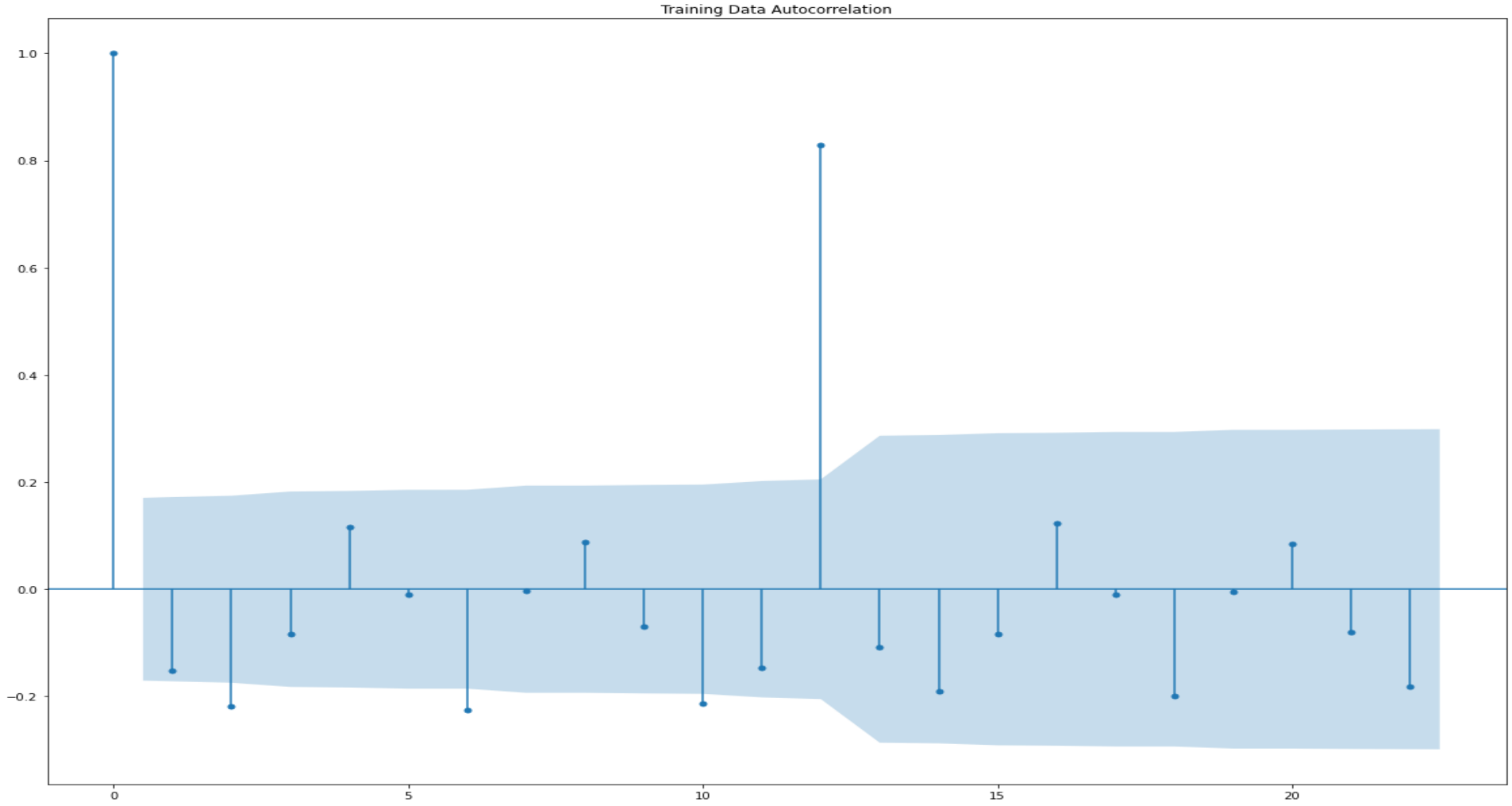
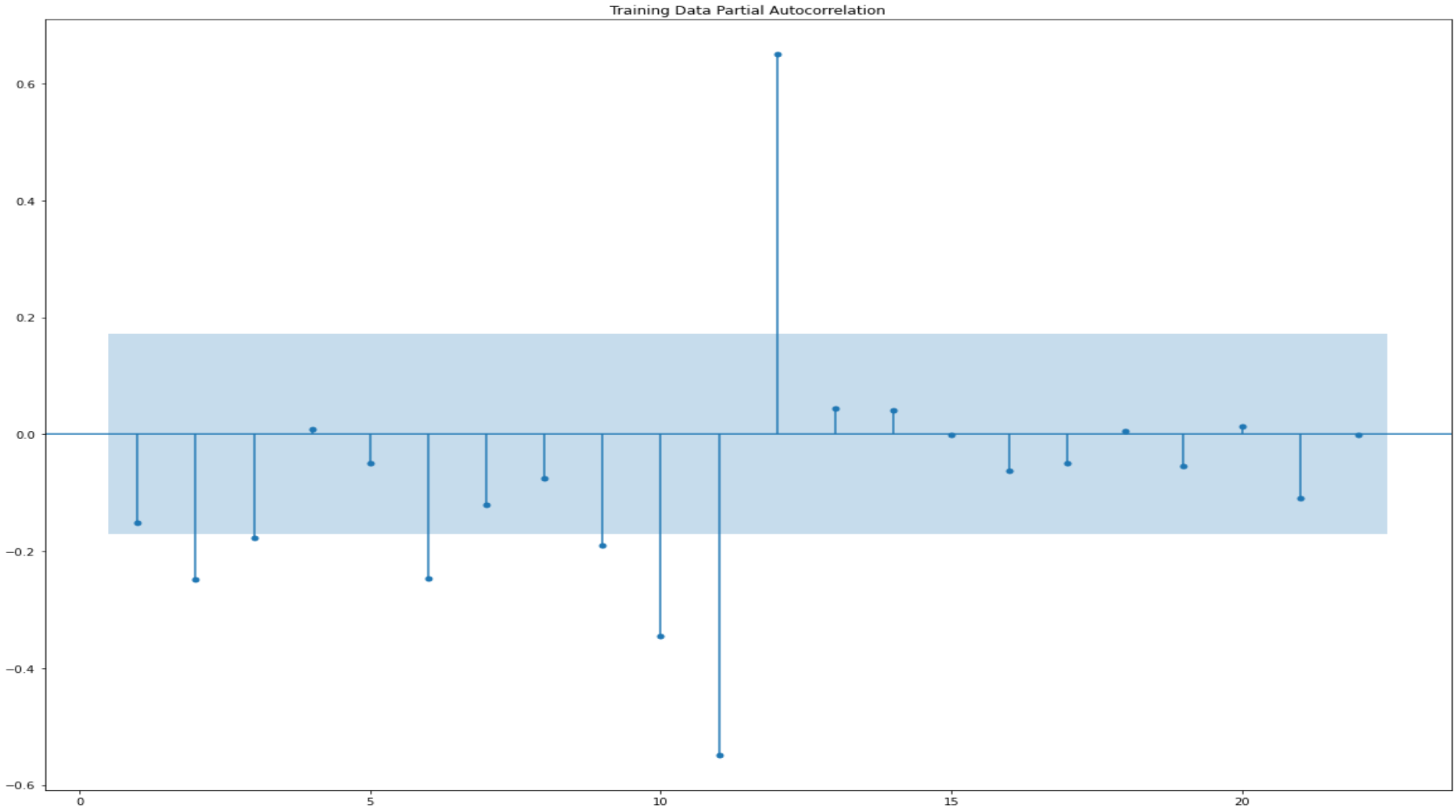


Figure 112. ARIMA Model: Partial Autocorrelation Function (PACF)



Back to Top

We see that the best combination based on the above ACF and PACF plots is (1,1,4) and below is the summary report, diagnostic plots, and manual RMSE score, which is **1263.471**, and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Figure 113. ARIMA Model: Partial Autocorrelation Function (PACF)

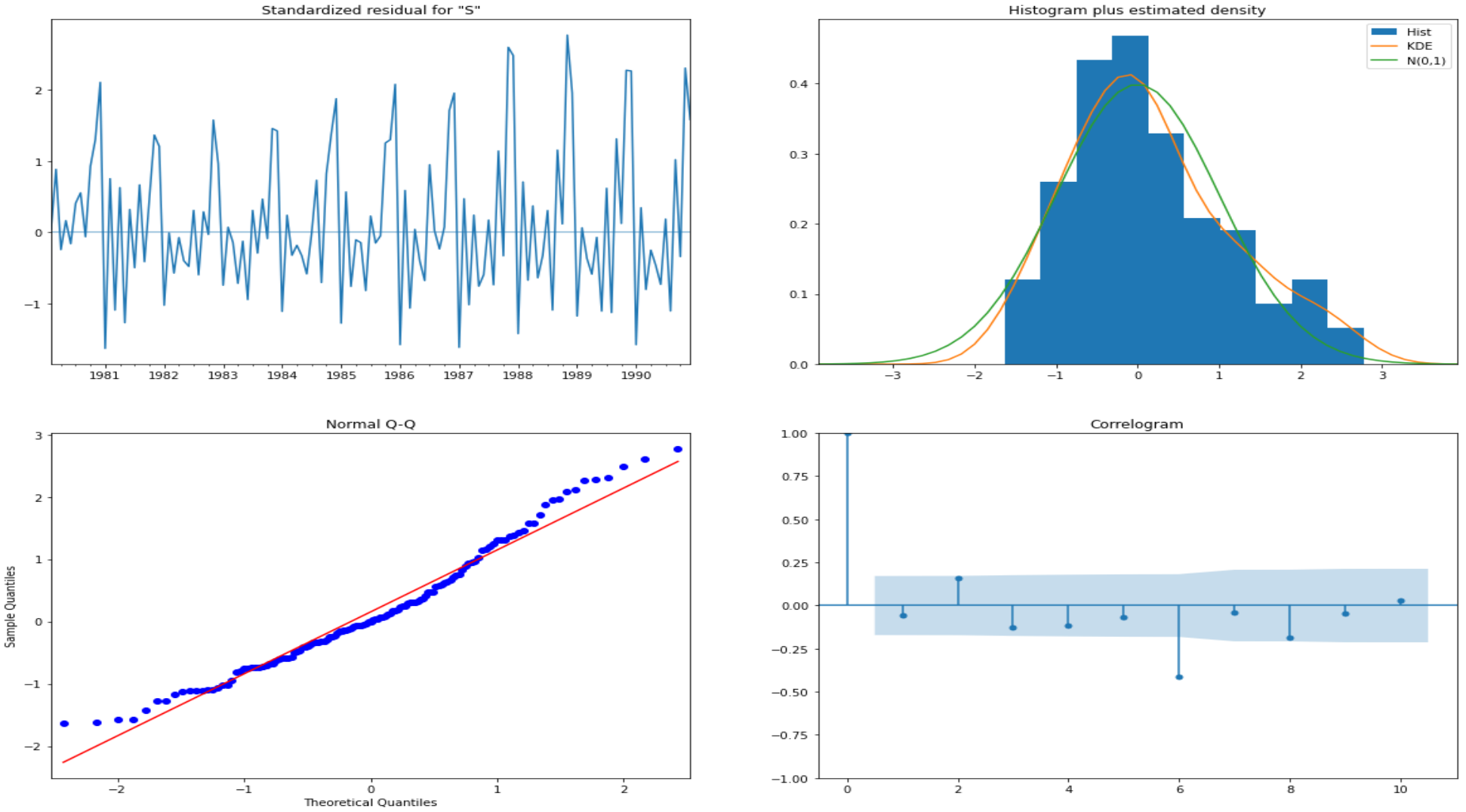
SARIMAX Results

Dep. Variable:	Sparkling	No. Observations:	132
Model:	ARIMA(1, 1, 4)	Log Likelihood	-1107.868
Date:	Fri, 07 Oct 2022	AIC	2227.737
Time:	23:06:44	BIC	2244.988
Sample:	01-31-1980	HQIC	2234.747
	- 12-31-1990		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.9959	0.009	-116.351	0.000	-1.013	-0.979
ma.L1	0.6628	0.162	4.103	0.000	0.346	0.979
ma.L2	-1.1650	0.242	-4.813	0.000	-1.639	-0.691
ma.L3	-0.6764	0.074	-9.169	0.000	-0.821	-0.532
ma.L4	0.1787	0.080	2.222	0.026	0.021	0.336
sigma2	1.218e+06	4.81e-07	2.53e+12	0.000	1.22e+06	1.22e+06

Ljung-Box (L1) (Q):	0.47	Jarque-Bera (JB):	6.95
Prob(Q):	0.49	Prob(JB):	0.03
Heteroskedasticity (H):	2.36	Skew:	0.55
Prob(H) (two-sided):	0.01	Kurtosis:	2.80

Figure 114. Manual ARIMA: Diagnostics



Back to Top

3.7.2 SARIMA Model

SARIMA Model: (Please refer to the Jupyter notebook)

- We have used iteration tools to try combinations with p, d, q and P, D, Q parameters as performed
- For the above combinations, we have calculated the Akaike Information Criterion (AIC) score to determine which is the best combination with lowest AIC score
- Based on the figure 115, we can see that combination (2, 1, 3) x (2, 0, 3, 6) has the lowest AIC score and we will build a model on the training dataset for the same combination and test the accuracy using by building a summary report, diagnostics, and RMSE score which has come to **824.024,** and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Figure 115. Example Parameter Combinations

```
Examples of the parameter combinations for the Model are
Model: (0, 1, 1)(0, 0, 1, 6)
Model: (0, 1, 2)(0, 0, 2, 6)
Model: (0, 1, 3)(0, 0, 3, 6)
Model: (1, 1, 0)(1, 0, 0, 6)
Model: (1, 1, 1)(1, 0, 1, 6)
Model: (1, 1, 2)(1, 0, 2, 6)
Model: (1, 1, 3)(1, 0, 3, 6)
Model: (2, 1, 0)(2, 0, 0, 6)
Model: (2, 1, 1)(2, 0, 1, 6)
Model: (2, 1, 2)(2, 0, 2, 6)
Model: (2, 1, 3)(2, 0, 3, 6)
Model: (3, 1, 0)(3, 0, 0, 6)
Model: (3, 1, 1)(3, 0, 1, 6)
Model: (3, 1, 2)(3, 0, 2, 6)
Model: (3, 1, 3)(3, 0, 3, 6)
```

Figure 116. Parameters with AIC scores (ascending function)

	param	seasonal	AIC
187	(2, 1, 3)	(2, 0, 3, 6)	1629.05271
251	(3, 1, 3)	(2, 0, 3, 6)	1631.005209
59	(0, 1, 3)	(2, 0, 3, 6)	1633.327863
63	(0, 1, 3)	(3, 0, 3, 6)	1634.986909
123	(1, 1, 3)	(2, 0, 3, 6)	1635.42744

Figure 117. SARIMA Model: Summary Report

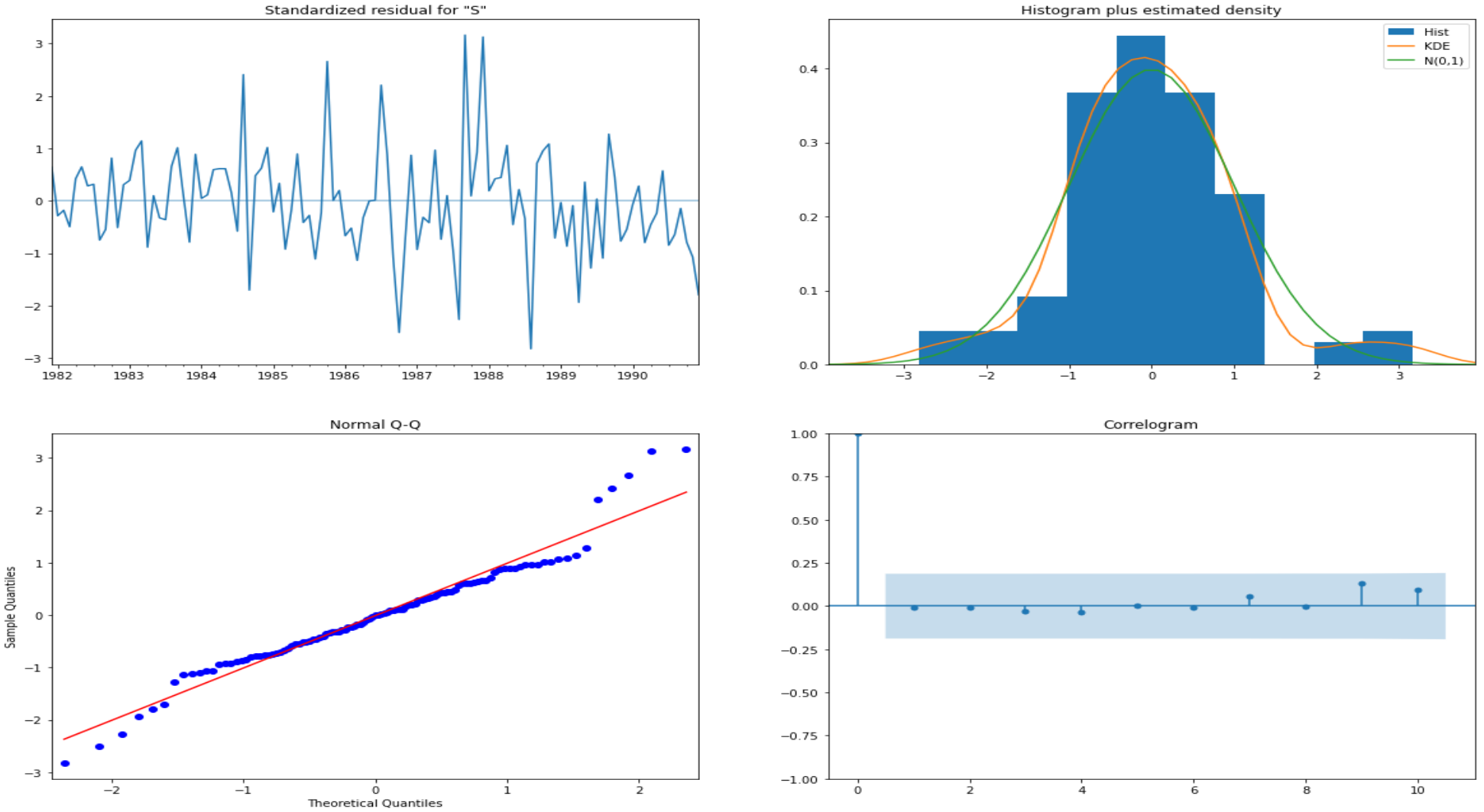
SARIMAX Results

Dep. Variable:	Sparkling	No. Observations:	132
Model:	SARIMAX(2, 1, 3)x(2, 0, 3, 6)	Log Likelihood	-803.526
Date:	Fri, 07 Oct 2022	AIC	1629.053
Time:	23:12:50	BIC	1658.658
Sample:	01-31-1980 - 12-31-1990	HQIC	1641.059
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-1.7438	0.063	-27.602	0.000	-1.868	-1.620
ar.L2	-0.7862	0.068	-11.595	0.000	-0.919	-0.653
ma.L1	1.0836	0.291	3.727	0.000	0.514	1.653
ma.L2	-0.7533	0.122	-6.151	0.000	-0.993	-0.513
ma.L3	-0.8893	0.257	-3.462	0.001	-1.393	-0.386
ar.S.L6	-0.0115	0.029	-0.391	0.696	-0.069	0.046
ar.S.L12	1.0382	0.022	47.576	0.000	0.995	1.081
ma.S.L6	0.3761	0.261	1.443	0.149	-0.135	0.887
ma.S.L12	-0.7589	0.200	-3.788	0.000	-1.151	-0.366
ma.S.L18	0.1110	0.159	0.700	0.484	-0.200	0.422
sigma2	1.026e+05	5e-06	2.05e+10	0.000	1.03e+05	1.03e+05

Ljung-Box (L1) (Q):	0.01	Jarque-Bera (JB):	14.94
Prob(Q):	0.93	Prob(JB):	0.00
Heteroskedasticity (H):	1.49	Skew:	0.38
Prob(H) (two-sided):	0.23	Kurtosis:	4.65

Figure 118. SARIMA Model: Diagnostic



Back to Top

We use Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) to figure out the order of SARIMA model:

Figure 119. SARIMA Model: Autocorrelation Function (ACF)

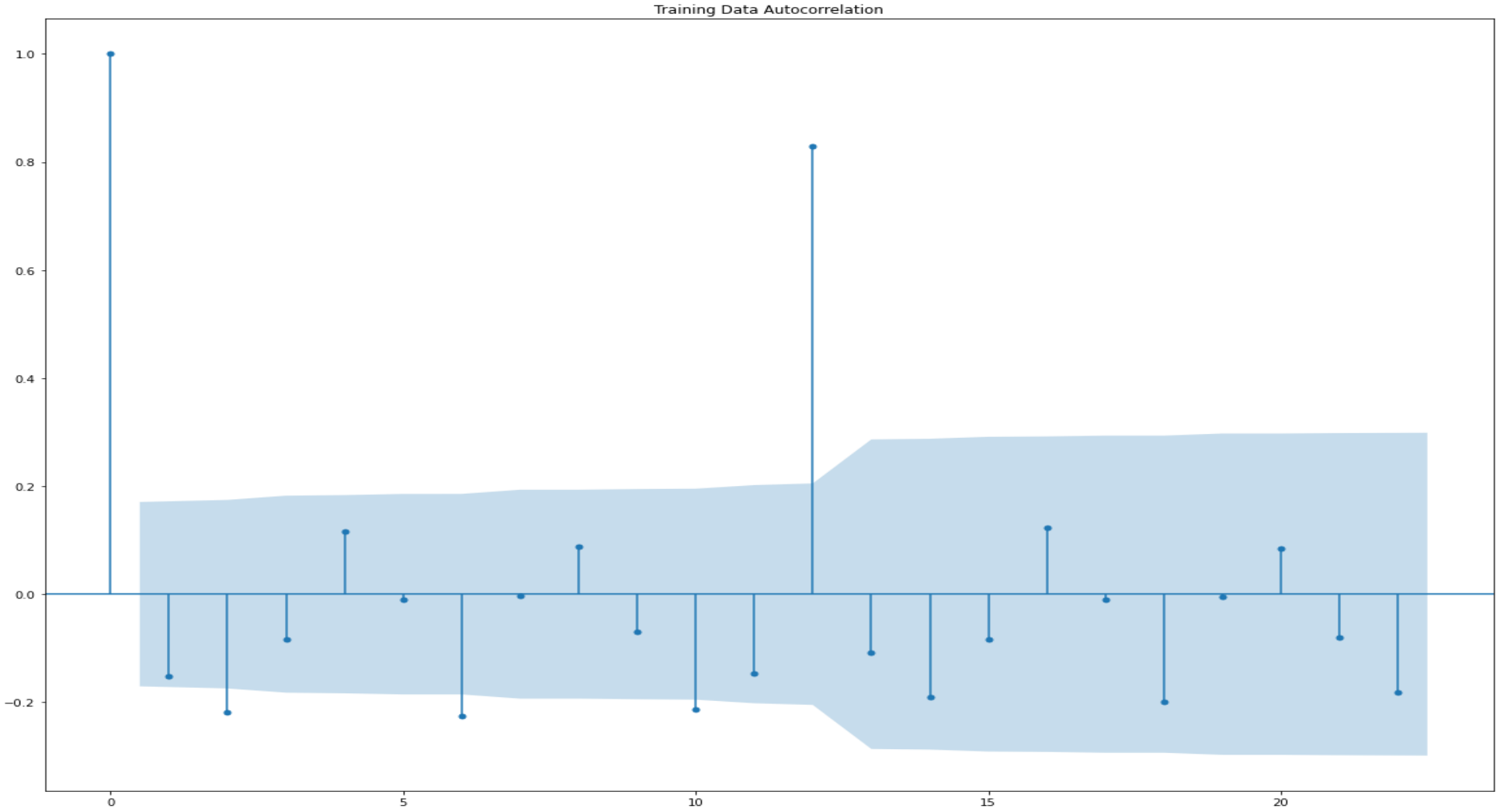
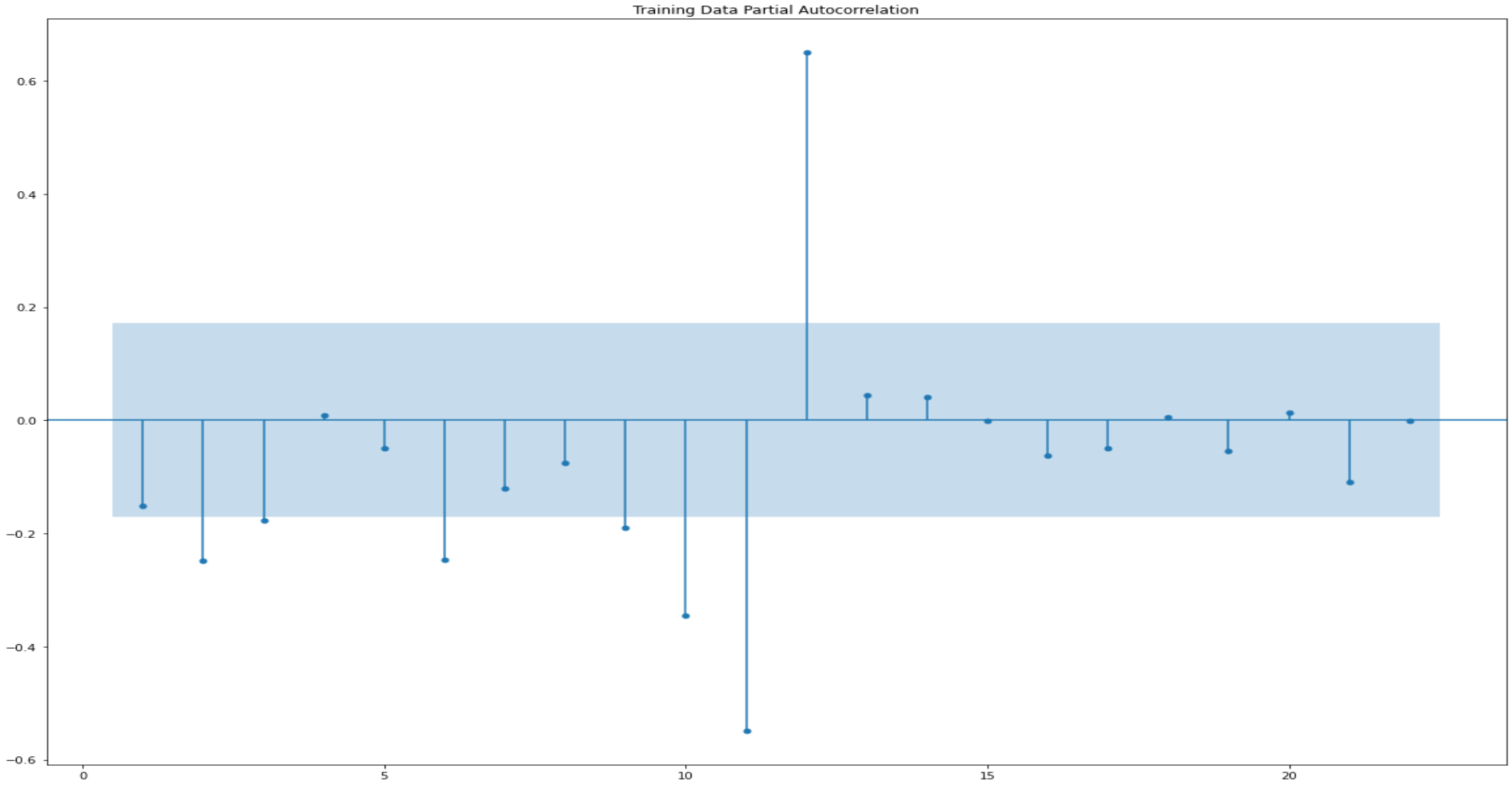


Figure 120. SARIMA Model: Partial Autocorrelation Function (PACF)



We see that the best combination based on the above ACF and PACF plots is (1,1,2) x (2,0,3,6) and below is the summary report, diagnostic plots, and manual RMSE score, which is **768.759**, and saved it in the dataframe **'results'** with the RMSE score to analyze the same for all the models we build.

Back to Top

Figure 121. Manual SARIMA: Summary Report

SARIMAX Results

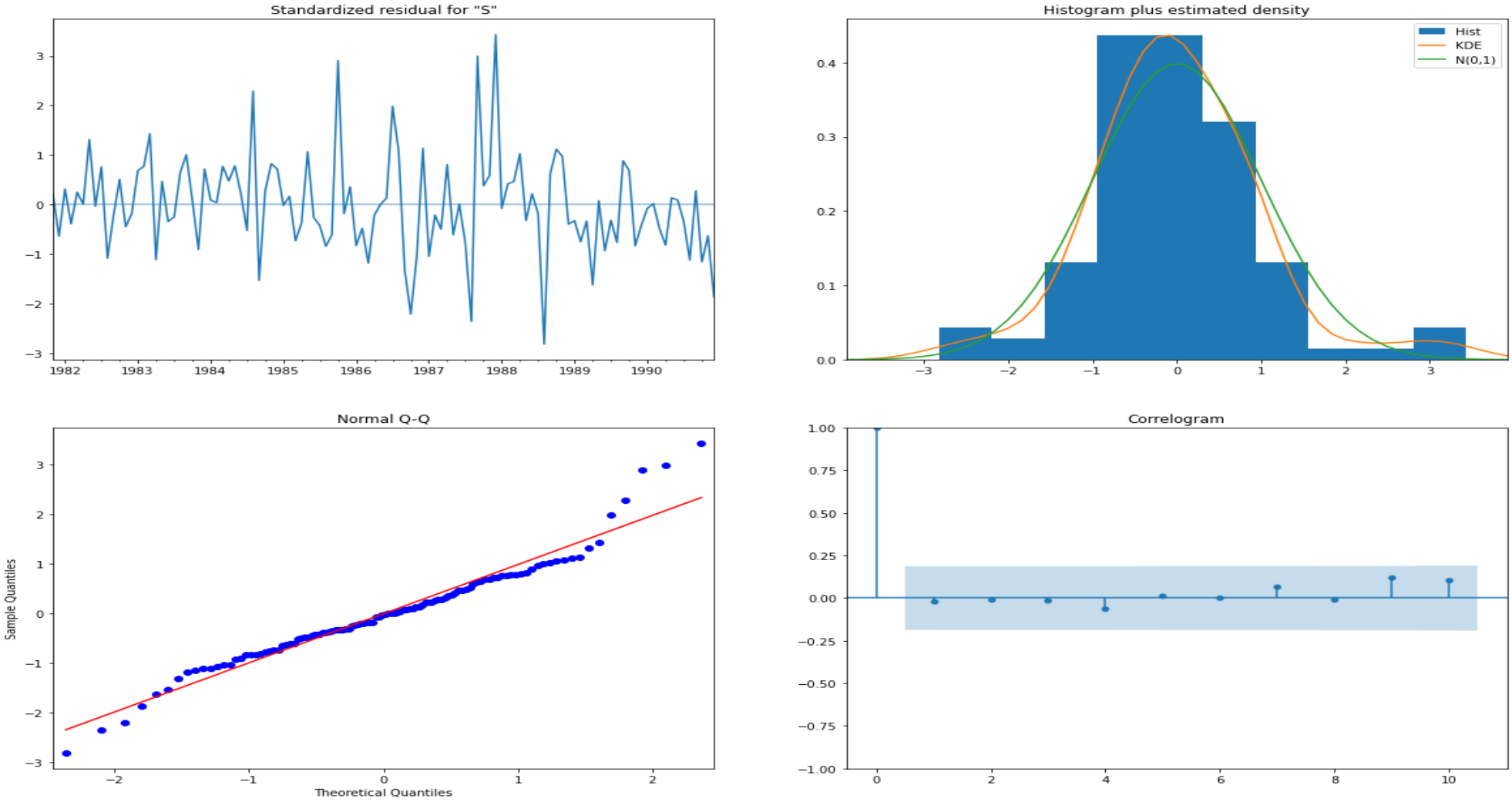
Dep. Variable:	Sparkling	No. Observations:	132
Model:	SARIMAX(1, 1, 2)x(2, 0, [1, 2, 3], 6)	Log Likelihood	-813.744
Date:	Fri, 07 Oct 2022	AIC	1645.488
Time:	23:12:54	BIC	1669.793
Sample:	01-31-1980	HQIC	1655.346
	- 12-31-1990		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.7153	0.274	-2.608	0.009	-1.253	-0.178
ma.L1	-0.0428	0.239	-0.179	0.858	-0.510	0.425
ma.L2	-0.7535	0.173	-4.368	0.000	-1.092	-0.415
ar.S.L6	-0.0151	0.033	-0.453	0.650	-0.080	0.050
ar.S.L12	1.0294	0.024	42.992	0.000	0.982	1.076
ma.S.L6	-1.0932	1.258	-0.869	0.385	-3.559	1.372
ma.S.L12	-1.3484	0.531	-2.541	0.011	-2.389	-0.308
ma.S.L18	0.3272	0.812	0.403	0.687	-1.264	1.918
sigma2	4.862e+04	5.62e+04	0.865	0.387	-6.15e+04	1.59e+05

Ljung-Box (L1) (Q):	0.04	Jarque-Bera (JB):	20.68
Prob(Q):	0.84	Prob(JB):	0.00
Heteroskedasticity (H):	1.73	Skew:	0.50
Prob(H) (two-sided):	0.10	Kurtosis:	4.87

Back to Top

Figure 122. Manual SARIMA: Diagnostic



Back to Top

3.8 Build a table with all the models built along with their corresponding parameters and the respective RMSE values on the test data.

We have built a new dataframe **'Final_results'** with RMSE scores of all the models built to check the accuracy for further forecast for next twelve months

Table 17 Final Results Dataframe with RMSE Scores

	Test RMSE
RegressionOnTime	1389.135175
NaiveModel	3864.279352
SimpleAverageModel	1275.081804
2pointTrailingMovingAverage	813.400684
4pointTrailingMovingAverage	1156.589694
6pointTrailingMovingAverage	1283.927428
9pointTrailingMovingAverage	1346.278315
Alpha=0.05,SimpleExponentialSmoothing	1316.035487
Alpha=0.05,DoubleExponentialSmoothing	5291.879833
Alpha=0.03,Beta=0.03,Gamma=0.3,TripleExponentialSmoothing	392.786198
ARIMA(2,1,2)	1299.979640
Manual ARIMA(1,1,4)	1263.471289
SARIMA (2, 1, 3) (2, 0, 3, 6)	824.024016
Manual SARIMA (1, 1, 2) (2, 0, 3, 6)	768.759649

In line with the aforementioned table, we can see that manual triple exponential smoothing model is most accurate as its RMSE score is lowest among all the models built for future prediction.

3.9 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.

As manual triple exponential smoothing model gave us the lowest RMSE, we will build the model on the full data for forecasting rose wine sales for next the 12 months, which will give us the most accurate prediction for the same. In the manual triple exponential smoothing model, the following parameters were considered to give most accurate prediction, smoothing level: 0.3, smoothing trend: 0.3, smoothing seasonality: 0.3. We got the RMSE score using the aforementioned parameters on the complete dataset as **421.473**.

Figure 123. 12 Months Prediction Plot

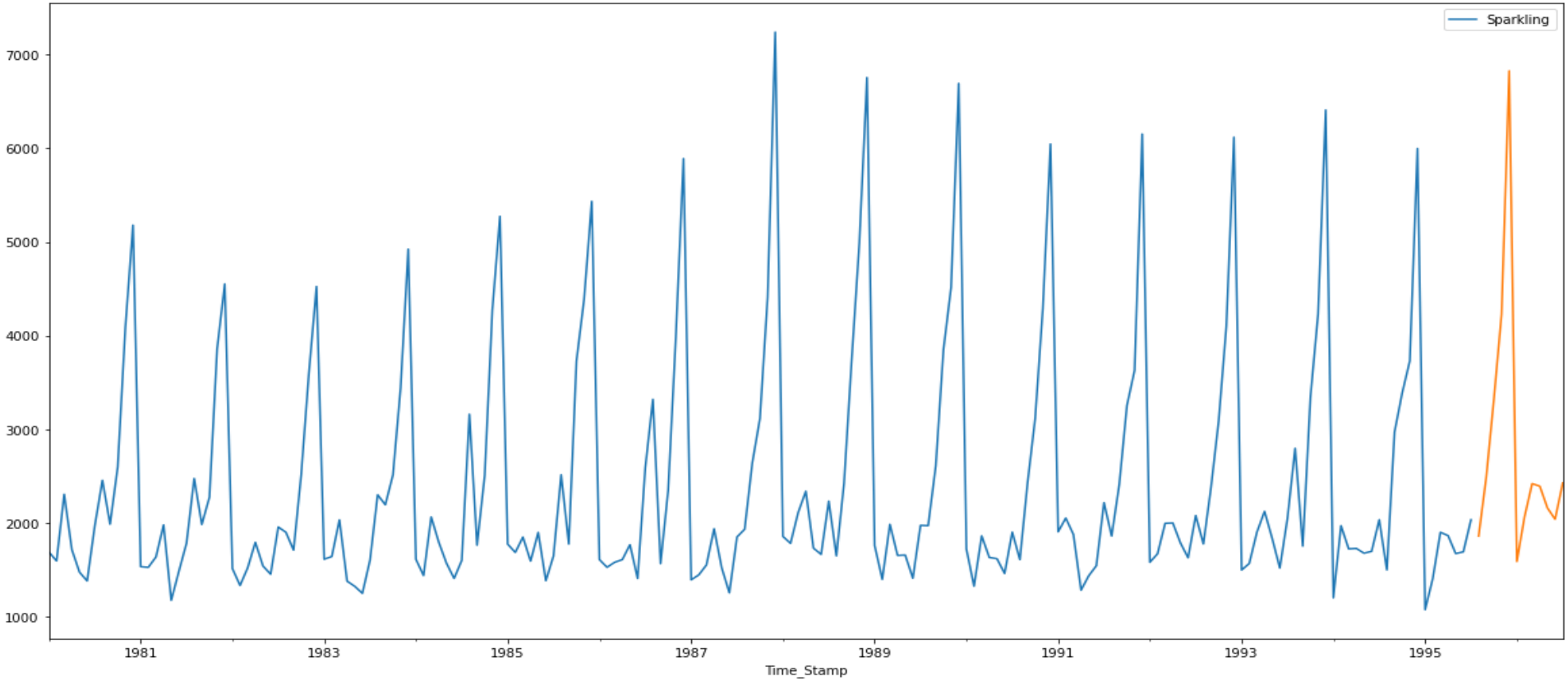


Table 18 Final Results Dataframe with RMSE Scores (including for complete dataset)

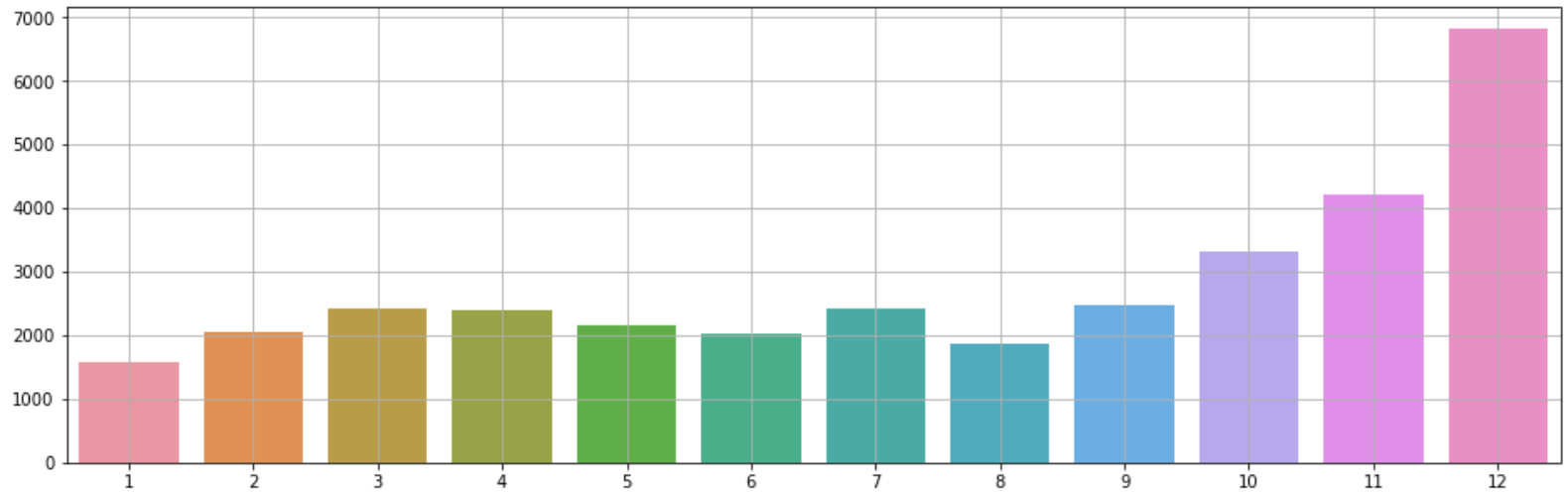
	Test RMSE
RegressionOnTime	1389.135175
NaiveModel	3864.279352
SimpleAverageModel	1275.081804
2pointTrailingMovingAverage	813.400684
4pointTrailingMovingAverage	1156.589694
6pointTrailingMovingAverage	1283.927428
9pointTrailingMovingAverage	1346.278315
Alpha=0.05,SimpleExponentialSmoothing	1316.035487
Alpha=0.05,DoubleExponentialSmoothing	5291.879833
Alpha=0.05,Beta=0.088,Gamma=0.323,TripleExponentialSmoothing	469.767970
Alpha=0.03,Beta=0.03,Gamma=0.3,TripleExponentialSmoothing	392.786198
ARIMA(2,1,2)	1299.979640
Manual ARIMA(1,1,4)	1263.471289
SARIMA (2, 1, 3) (2, 0, 3, 6)	824.024016
Manual SARIMA (1, 1, 2) (2, 0, 3, 6)	768.759649
Full Data Triple Exponential Smoothing Alpha=0.3,Beta=0.3,Gamma=0.3	421.473091

3.10 Comment on the model thus built and report your findings and suggest the measures that the company should be taking for future sales.

3.10.1 Comments on the data and model built

- We see that manual triple exponential smoothing model gives us the lowest RMSE and has most accurately predicts or forecasts for the next 12 months
- There is a clear seasonality in the data, however, there is no trend in the sales data and the we cannot determine if the sales are going up or down, they don't follow any pattern
- In addition, there is an obvious upward trend in sales when we see the box plot for monthly sales from January to December. The sparkling wine sales pick up by the end of the year and specifically the sales increase tremendously from July and August.
- There is clear seasonality in the data and its constant on a yearly basis. We can see that June denotes lowest sales while December records higher number of sales as compared to other months.
- The parameters used to build a triple exponential smoothing model on train data is smoothing level as 0.3, smoothing trend as 0.3, and smoothing seasonality is 0.3, which gives highest accuracy. In addition, we can see that the model built on training data is not so accurate when to predict the dips in the sales, however, it resembles with the peaks in sales.
- The revenue sales are estimated to total around **33,791.36** for next 12 months with an average of **2,815.95**

Figure 124. 12 Months Bar Plot on Forecast Data, August 1995 - July 1996



Back to Top

3.10.2 Measures that the company should be taking for future sales

- The sparkling wine sales do not show an upward or downward trend. However, the seasonality is highly evident with increasing trend at month-on-month level and there is a sudden surge in the sales of sparkling wines from July-August time of the year. The company can observe the reasons for the same and try to optimize the sales opportunities and maximize the sparkling wine sales during the last quarter of the year.
- The company can have wine tasting events twice or thrice a year at their estate to popularize their wine collections and get feedback from its customers
- Offering food dishes which complement rose wines could prove helpful in attracting customers and improve sales for the rose wines
- We know sparkling wine is associated with festivals as it is served with food during holiday season and it is also bought as a gift during Christmas and New Year's. As a result, the company can improve their customer reach during the last quarter of the year. The company can also introduce new wines during holiday season to augment their revenues.

The End