In [1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
```

In [2]:
```
df=pd.read_excel("CocaCola_Sales_Rawdata.xlsx")
df
```

Out[2]:

	Quarter	Sales
0	Q1_86	1734.827000
1	Q2_86	2244.960999
2	Q3_86	2533.804993
3	Q4_86	2154.962997
4	Q1_87	1547.818996
5	Q2_87	2104.411995
6	Q3_87	2014.362999
7	Q4_87	1991.746998
8	Q1_88	1869.049999
9	Q2_88	2313.631996
10	Q3_88	2128.320000
11	Q4_88	2026.828999
12	Q1_89	1910.603996
13	Q2_89	2331.164993
14	Q3_89	2206.549995
15	Q4_89	2173.967995
16	Q1_90	2148.278000
17	Q2_90	2739.307999
18	Q3_90	2792.753998
19	Q4_90	2556.009995
20	Q1_91	2480.973999
21	Q2_91	3039.522995
22	Q3_91	3172.115997
23	Q4_91	2879.000999
24	Q1_92	2772.000000
25	Q2_92	3550.000000
26	Q3_92	3508.000000
27	Q4_92	3243.859993
28	Q1_93	3056.000000
29	Q2_93	3899.000000
30	Q3_93	3629.000000
31	Q4_93	3373.000000
32	Q1_94	3352.000000
33	Q2_94	4342.000000
34	Q3_94	4461.000000
35	Q4_94	4017.000000
36	Q1_95	3854.000000
37	Q2_95	4936.000000
38	Q3_95	4895.000000
39	Q4_95	4333.000000
40	Q1_96	4194.000000
41	Q2_96	5253.000000

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42 entries, 0 to 41
```

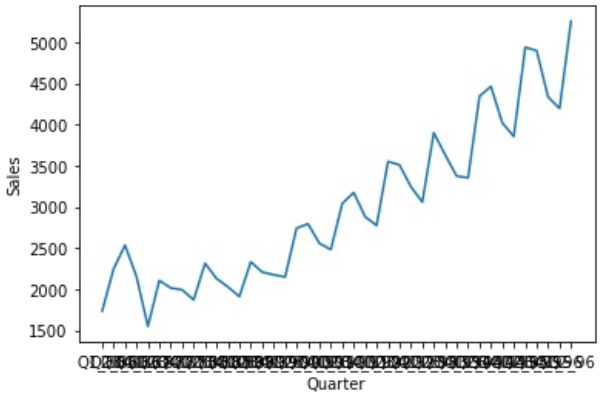
```
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Quarter  42 non-null     object
 1   Sales    42 non-null     float64
dtypes: float64(1), object(1)
memory usage: 800.0+ bytes
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Quarter    0
Sales      0
dtype: int64
```

In [5]:
```
import seaborn as sns
sns.lineplot(x="Quarter",y="Sales",data=df)
```

Out[5]:
```
<AxesSubplot:xlabel='Quarter', ylabel='Sales'>
```

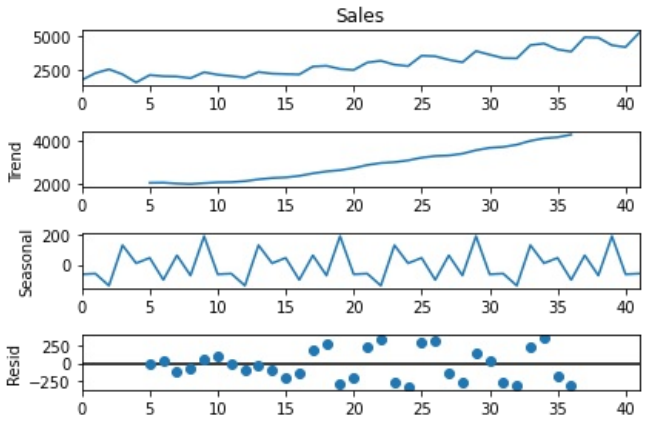


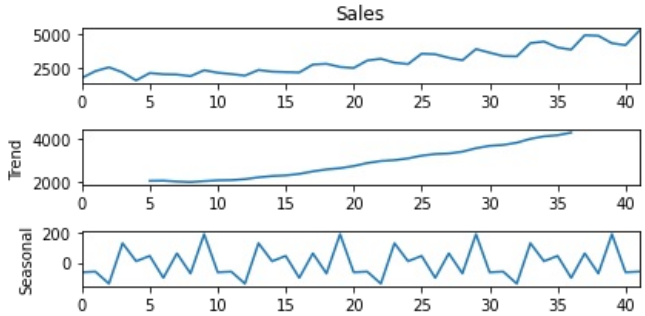
In [6]:
```
import statsmodels.api as smf
```

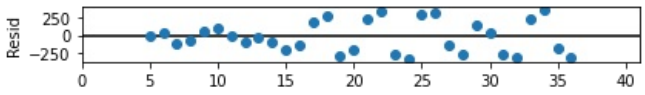
In [7]:
```
seasonal_ts_add=smf.tsa.seasonal_decompose(df["Sales"],freq=10)
seasonal_ts_add.plot()
```

```
C:\Users\rajesh\AppData\Local\Temp/ipykernel_15828/3913548348.py:1: FutureWarning: the 'freq'' keyword is depreca
ted, use 'period' instead
  seasonal_ts_add=smf.tsa.seasonal_decompose(df["Sales"],freq=10)
```

Out[7]:

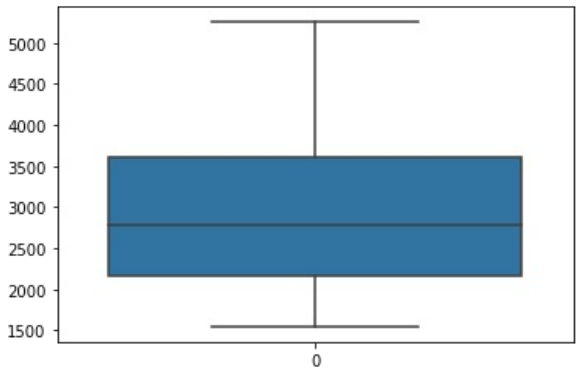






In [8]:
```
sns.boxplot(data=df['Sales'])
```

Out[8]:
```
<AxesSubplot:>
```



In [9]:
```
quarter =['Q1','Q2','Q3','Q4']
```

In [10]:
```
p = df["Quarter"][0]
p[0:2]
df['quarter']= 0

for i in range(42):
    p = df["Quarter"][i]
    df['quarter'][i]= p[0:2]

df.head()
```

```
C:\Users\rajesh\AppData\Local\Temp/ipykernel_15828/2922281831.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['quarter'][i]= p[0:2]
C:\Users\rajesh\anaconda3\lib\site-packages\pandas\core\indexing.py:1732: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```

Out[10]:

	Quarter	Sales	quarter
0	Q1_86	1734.827000	Q1
1	Q2_86	2244.960999	Q2
2	Q3_86	2533.804993	Q3
3	Q4_86	2154.962997	Q4
4	Q1_87	1547.818996	Q1

In [11]:
```
quarter_dummies = pd.DataFrame(pd.get_dummies(df['quarter']))
df1 = pd.concat([df,quarter_dummies],axis = 1)
df1.head()
```

Out[11]:

	Quarter	Sales	quarter	Q1	Q2	Q3	Q4
0	Q1_86	1734.827000	Q1	1	0	0	0
1	Q2_86	2244.960999	Q2	0	1	0	0
2	Q3_86	2533.804993	Q3	0	0	1	0
3	Q4_86	2154.962997	Q4	0	0	0	1
4	Q1_87	1547.818996	Q1	1	0	0	0

In [12]:
```
df1["t"] = np.arange(1,43)
df1["t_squared"] = df1["t"]*df1["t"]
df1["log_Sales"] = np.log(df1["Sales"])
df1.head()
```

Out[12]:

	Quarter	Sales	quarter	Q1	Q2	Q3	Q4	t	t_squared	log_Sales
0	Q1_86	1734.827000	Q1	1	0	0	0	1	1	7.458663
1	Q2_86	2244.960999	Q2	0	1	0	0	2	4	7.716443
2	Q3_86	2533.804993	Q3	0	0	1	0	3	9	7.837477
3	Q4_86	2154.962997	Q4	0	0	0	1	4	16	7.675529
4	Q1_87	1547.818996	Q1	1	0	0	0	5	25	7.344602

In [13]:
```
Train = df1.head(30)
Test = df1.tail(10)
```

In [14]:
```
# L I N E A R
import statsmodels.formula.api as smf

linear_model = smf.ols('Sales~t',data=Train).fit()
pred_linear =  pd.Series(linear_model.predict(pd.DataFrame(Test['t'])))
rmse_linear = np.sqrt(np.mean((np.array(Test['Sales'])-np.array(pred_linear))**2))
print("RMSE Linear: ",rmse_linear)
```

```
RMSE Linear:  777.6287139221073
```

In [15]:
```
# Exponential

Exp = smf.ols('log_Sales~t',data=Train).fit()
pred_Exp = pd.Series(Exp.predict(pd.DataFrame(Test['t'])))
rmse_Exp = np.sqrt(np.mean((np.array(Test['Sales'])-np.array(np.exp(pred_Exp)))**2))
print("RMSE Exponential: ",rmse_Exp)
```

```
RMSE Exponential:  600.089369373966
```

In [16]:
```
# Quadratic

Quad = smf.ols('Sales~t+t_squared',data=Train).fit()
pred_Quad = pd.Series(Quad.predict(Test[["t","t_squared"]]))
rmse_Quad = np.sqrt(np.mean((np.array(Test['Sales'])-np.array(pred_Quad))**2))
print("RMSE Quadratic: ",rmse_Quad)
```

```
RMSE Quadratic:  680.2527854193611
```

In [17]:
```
# Additive seasonality

add_sea = smf.ols('Sales~Q1+Q2+Q3+Q4',data=Train).fit()
pred_add_sea = pd.Series(add_sea.predict(Test[['Q1', 'Q2', 'Q3', 'Q4']]))
rmse_add_sea = np.sqrt(np.mean((np.array(Test['Sales'])-np.array(pred_add_sea))**2))
print("RMSE Additive seasonality: ",rmse_add_sea)
```

```
RMSE Additive seasonality:  1898.350480415752
```

In [18]:
```
#Additive Seasonality Quadratic

add_sea_Quad = smf.ols('Sales~t+t_squared+Q1+Q2+Q3+Q4',data=Train).fit()
pred_add_sea_quad = pd.Series(add_sea_Quad.predict(Test[['Q1', 'Q2', 'Q3', 'Q4','t','t_squared']]))
rmse_add_sea_quad = np.sqrt(np.mean((np.array(Test['Sales'])-np.array(pred_add_sea_quad))**2))
print("RMSE Additive Seasonality Quadratic:",rmse_add_sea_quad )
```

```
RMSE Additive Seasonality Quadratic: 607.8520720183915
```

In [19]:
```
# Multiplicative Seasonality
```

```
# Multiplicative Seasonality

Mul_sea = smf.ols('log_Sales~Q1+Q2+Q3+Q4',data = Train).fit()
pred_Mult_sea = pd.Series(Mul_sea.predict(Test))
rmse_Mult_sea = np.sqrt(np.mean((np.array(Test['Sales'])-np.array(np.exp(pred_Mult_sea)))**2))
print("RMSE Multiplicative Seasonality:",rmse_Mult_sea)
```

```
RMSE Multiplicative Seasonality: 1951.034939969767
```

In [20]:

```
# Multiplicative Additive Seasonality

Mul_Add_sea = smf.ols('log_Sales~t+Q1+Q2+Q3+Q4',data = Train).fit()
pred_Mult_add_sea = pd.Series(Mul_Add_sea.predict(Test))
rmse_Mult_add_sea = np.sqrt(np.mean((np.array(Test['Sales'])-np.array(np.exp(pred_Mult_add_sea)))**2))
print("RMSE Multiplicative Additive Seasonality:",rmse_Mult_add_sea )
```

```
RMSE Multiplicative Additive Seasonality: 449.4035489163217
```

In [21]:

```
# Testing

data1 = {"MODEL":pd.Series(["rmse_linear","rmse_Exp","rmse_Quad","rmse_add_sea","rmse_add_sea_quad","rmse_Mult_se
        "RMSE_Values":pd.Series([rmse_linear,rmse_Exp,rmse_Quad,rmse_add_sea,rmse_add_sea_quad,rmse_Mult_sea,rmse
table_rmse=pd.DataFrame(data1)
table_rmse
```

Out[21]:

	MODEL	RMSE_Values
0	rmse_linear	777.628714
1	rmse_Exp	600.089369
2	rmse_Quad	680.252785
3	rmse_add_sea	1898.350480
4	rmse_add_sea_quad	607.852072
5	rmse_Mult_sea	1951.034940
6	rmse_Mult_add_sea	449.403549

In [22]:

```
data = [['Q3_96', 'Q3'], ['Q4_96', 'Q4'], ['Q1_97', 'Q1'],['Q2_97', 'Q2']]
print(data)
forecast = pd.DataFrame(data, columns = ['Quarter', 'quarter'])
forecast
```

```
[['Q3_96', 'Q3'], ['Q4_96', 'Q4'], ['Q1_97', 'Q1'], ['Q2_97', 'Q2']]
```

Out[22]:

	Quarter	quarter
0	Q3_96	Q3
1	Q4_96	Q4
2	Q1_97	Q1
3	Q2_97	Q2

In [23]:

```
# Create dummies and T and T-Squared columns

dummies = pd.DataFrame(pd.get_dummies(forecast['quarter']))
forecast1 = pd.concat([forecast,dummies],axis = 1)

forecast1["t"] = np.arange(1,5)
forecast1["t_squared"] = forecast1["t"]*forecast1["t"]
print("\nAfter Dummy, T and T-Square\n\n",forecast1.head())
```

```
After Dummy, T and T-Square

   Quarter quarter  Q1  Q2  Q3  Q4  t  t_squared
0   Q3_96      Q3   0   0   1   0  1          1
1   Q4_96      Q4   0   0   0   1  2          4
2   Q1_97      Q1   1   0   0   0  3          9
3   Q2_97      Q2   0   1   0   0  4         16
```

In [24]:

```
# Forecasting using Additive Seasonality Quadratic Model

model_full = smf.ols('Sales~t+t_squared+Q1+Q2+Q3+Q4',data=df1).fit()
pred_new  = pd.Series(model_full.predict(forecast1))
pred_new

forecast1["forecasted_sales"] = pd.Series(pred_new)
```

In [25]:

```
# Final Prediction for next 4 Quarters

Final_predict = forecast1.loc[:, ['Quarter', 'forecasted_sales']]
Final_predict
```

Out[25]:

	Quarter	forecasted_sales
0	Q3_96	2180.858824
1	Q4_96	1851.383709
2	Q1_97	1635.419724
3	Q2_97	2284.261547

In [ ]: