In [22]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import pyplot
import seaborn as sns
import statsmodels.api as smf
```

In [23]:
```
df=pd.read_excel('CocaCola_Sales_Rawdata.xlsx')
```

In [24]:
```

df.head()
```

Out[24]:

	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

In [25]:
```
df.describe()
```

Out[25]:

	Sales
count	42.000000
mean	2994.353308
std	977.930896
min	1547.818996
25%	2159.714247
50%	2782.376999
75%	3609.250000
max	5253.000000

In [26]:
```
df.info()
```

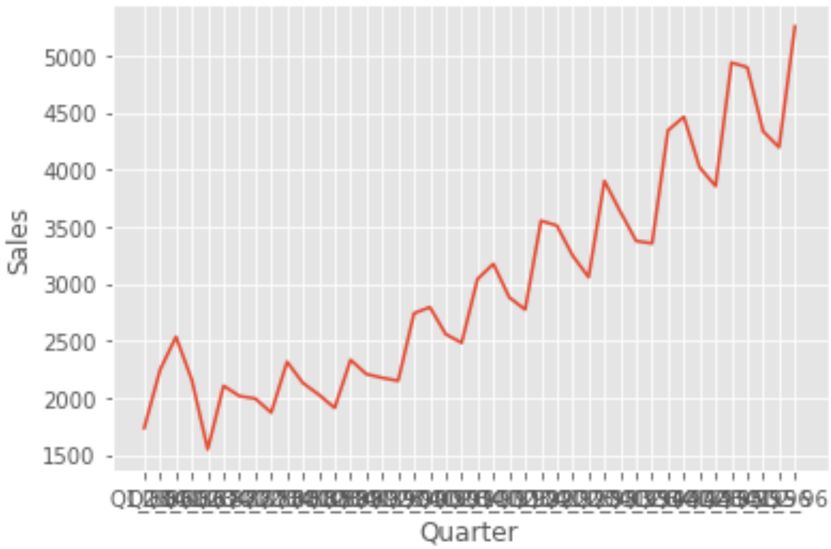
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42 entries, 0 to 41
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Quarter  42 non-null     object 
 1   Sales    42 non-null     float64
dtypes: float64(1), object(1)
memory usage: 800.0+ bytes
```

In [27]:
```
df.isnull().sum()
```

Out[27]:
```
Quarter    0
Sales      0
dtype: int64
```

In [28]:
```
sns.lineplot(x="Quarter",y="Sales",data=df)
```

Out[28]:
```
<AxesSubplot:xlabel='Quarter', ylabel='Sales'>
```

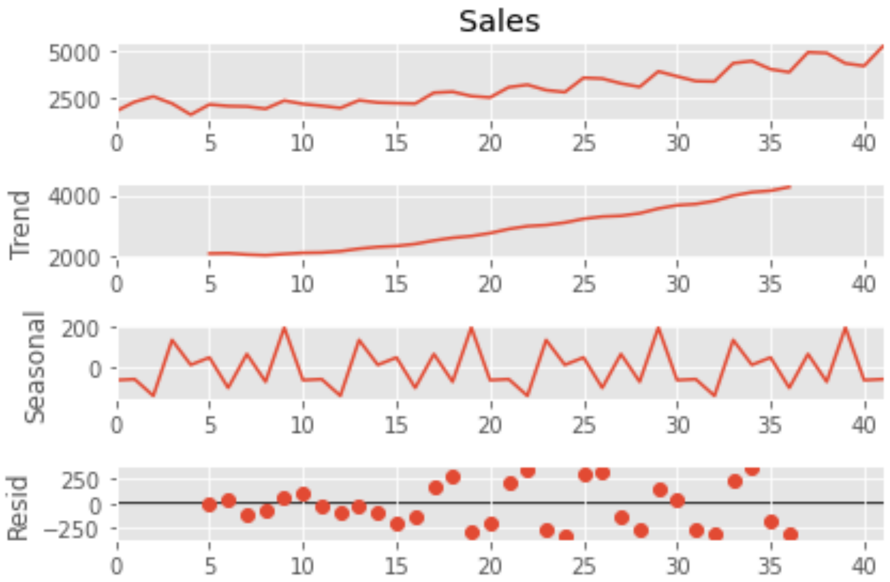


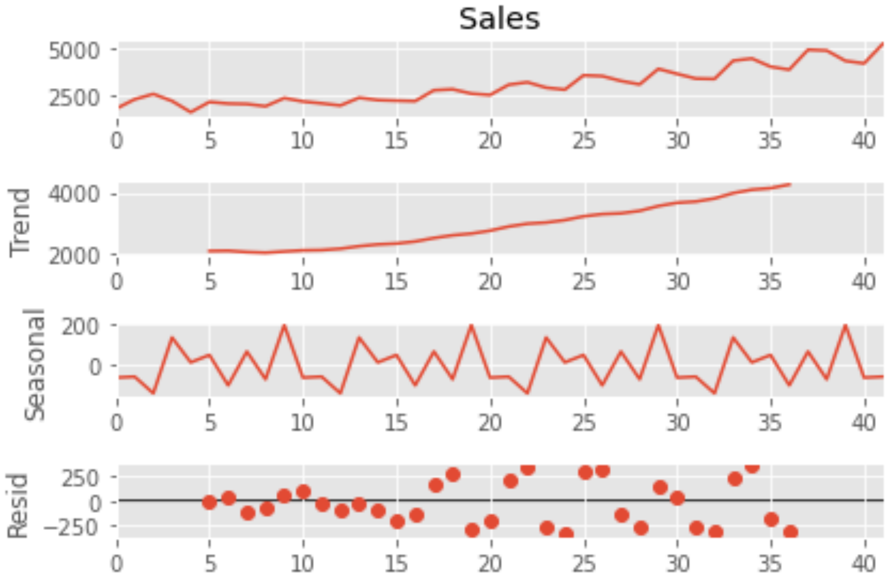
In [29]:

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

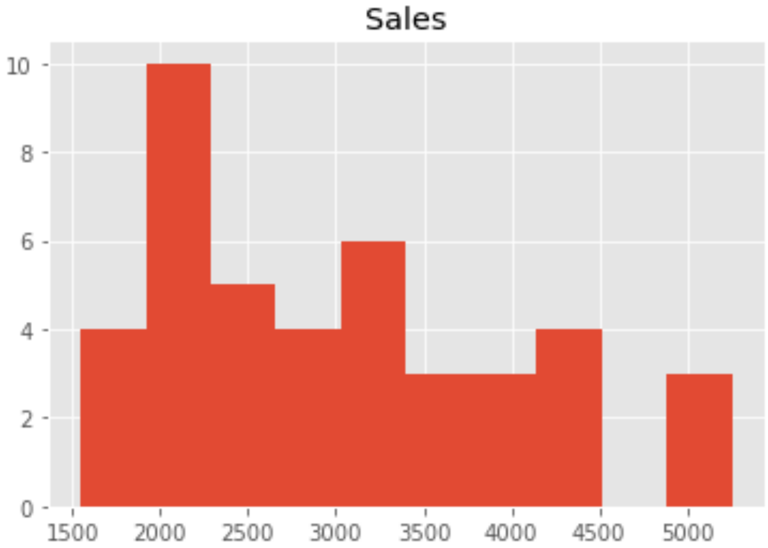
```
/var/folders/9_/ckpgdd3s4qzg3w1zytsfvsmh0000gn/T/ipykernel_21859/4
189873342.py:1: FutureWarning: the 'freq'' keyword is deprecated,
use 'period' instead
  seasonal_ts_add=smf.tsa.seasonal_decompose(df["Sales"],freq=10)
```

Out[29]:



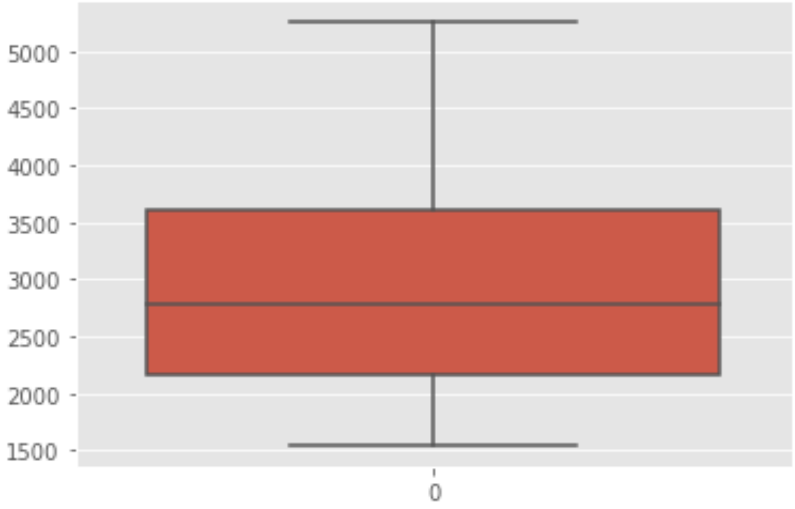


In [30]:
```
df.hist()
pyplot.show()
```



In [31]:
```
sns.boxplot(data=df['Sales'])
```

Out[31]:
```
<AxesSubplot:>
```



In [32]:
```
quarter =['Q1','Q2','Q3','Q4']
```

In [33]:

```
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()
```

```
/var/folders/9_/ckpgdd3s4qzg3w1zytsfvsmh0000gn/T/ipykernel_21859/2
825882259.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/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy)
  df['quarter'][i]= p[0:2]
/opt/anaconda3/lib/python3.9/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/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy)
  self._setitem_single_block(indexer, value, name)
```

Out[33]:

	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 [34]:

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

Out[34]:

	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 [35]:

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

Out[35]:

	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 [36]:

```
Train = df1.head(30)
Test = df1.tail(10)
```

In [37]:

```
# 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(pre
print("RMSE Linear: ",rmse_linear)
```

```
RMSE Linear:  777.6287139221071
```

In [38]:

```
# 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
print("RMSE Exponential: ",rmse_Exp)
```

```
RMSE Exponential:  600.0893693739741
```

In [39]:

```
# 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_
print("RMSE Quadratic: ",rmse_Quad)
```

```
RMSE Quadratic:  680.2527854192889
```

In [40]:

```
 Additive seasonality

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

```
RMSE Additive seasonality:  1898.3504804157517
```

In [41]:

```
#Additive Seasonality Quadratic

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

```
RMSE Additive Seasonality Quadratic: 607.8520720183552
```

In [42]:

```
# 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(n
print("RMSE Multiplicative Seasonality:",rmse_Mult_sea)
```

```
RMSE Multiplicative Seasonality: 1951.0349399697623
```

In [43]:

```
# 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.arr
print("RMSE Multiplicative Additive Seasonality:",rmse_Mult_add_sea
```

```
RMSE Multiplicative Additive Seasonality: 449.40354891629227
```

In [44]:

```
# Testing

data1 = {"MODEL":pd.Series(["rmse_linear","rmse_Exp","rmse_Quad","r
        "RMSE_Values":pd.Series([rmse_linear,rmse_Exp,rmse_Quad,rms
table_rmse=pd.DataFrame(data1)
table_rmse
```

Out[44]:

	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 [45]:

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

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

Out[45]:

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

In [46]:

```
# 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 [47]:

```
# 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 [48]:

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

Out[48]:

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

In [ ]: