In [1]:

```
# Load libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [3]:

```
# Import dataset
coke =pd.read_excel('CocaCola_Sales_Rawdata.xlsx')
coke.head()
```

Out[3]:

	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 [4]:

```
coke.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 [5]:

```
coke.describe()
```

Out[5]:

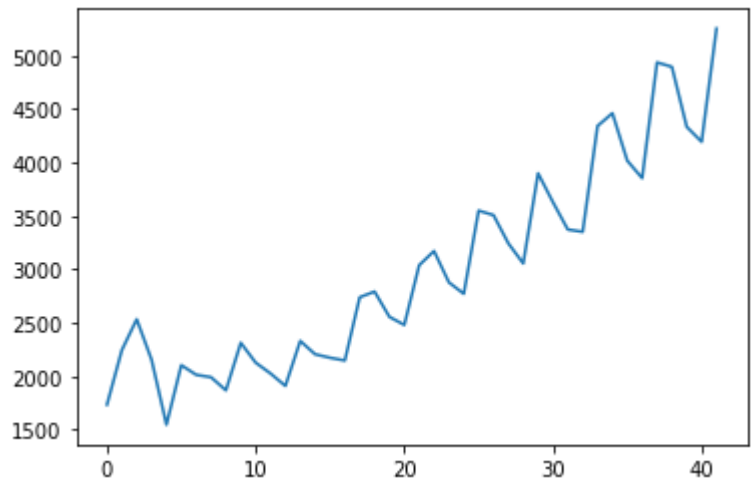
	Sales
count	42.000000
mean	2994.353308
std	977.930896
min	1547.818996
25%	2159.714247
50%	2782.376999
75%	3609.250000
max	5253.000000

In [6]:

```
coke.Sales.plot()
```

Out[6]:

```
<AxesSubplot:>
```



In [7]:

```
coke['Quarters'] = 0
coke['Year'] = 0
for i in range(42):
    p = coke['Quarter'][i]
    coke['Quarters'][i] = p[0:2]
    coke['Year'][i] = p[3:5]
```

```
C:\Users\SOWMYA~1\AppData\Local\Temp/ipykernel_18240/1645036308.py:5: Settin
gWithCopyWarning:
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 (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  coke['Quarters'][i] = p[0:2]
C:\Users\sowmya sandeep\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 (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  self._setitem_single_block(indexer, value, name)
C:\Users\SOWMYA~1\AppData\Local\Temp/ipykernel_18240/1645036308.py:6: Settin
gWithCopyWarning:
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 (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  coke['Year'][i] = p[3:5]
```

In [8]:

```
# Preparing summies

Quarters_Dummies =pd.DataFrame(pd.get_dummies(coke['Quarters']))
coke1 = pd.concat([coke,Quarters_Dummies], axis = 1)
```

In [9]:

```
coke1["t"]=np.arange(1,43)

coke1["t_squared"] = coke1["t"]*coke1["t"]


coke1["Log_Sales"]=np.log(coke1["Sales"])
coke1.columns
```

Out[9]:

```
Index(['Quarter', 'Sales', 'Quarters', 'Year', 'Q1', 'Q2', 'Q3', 'Q4', 't',
       't_squared', 'Log_Sales'],
      dtype='object')
```

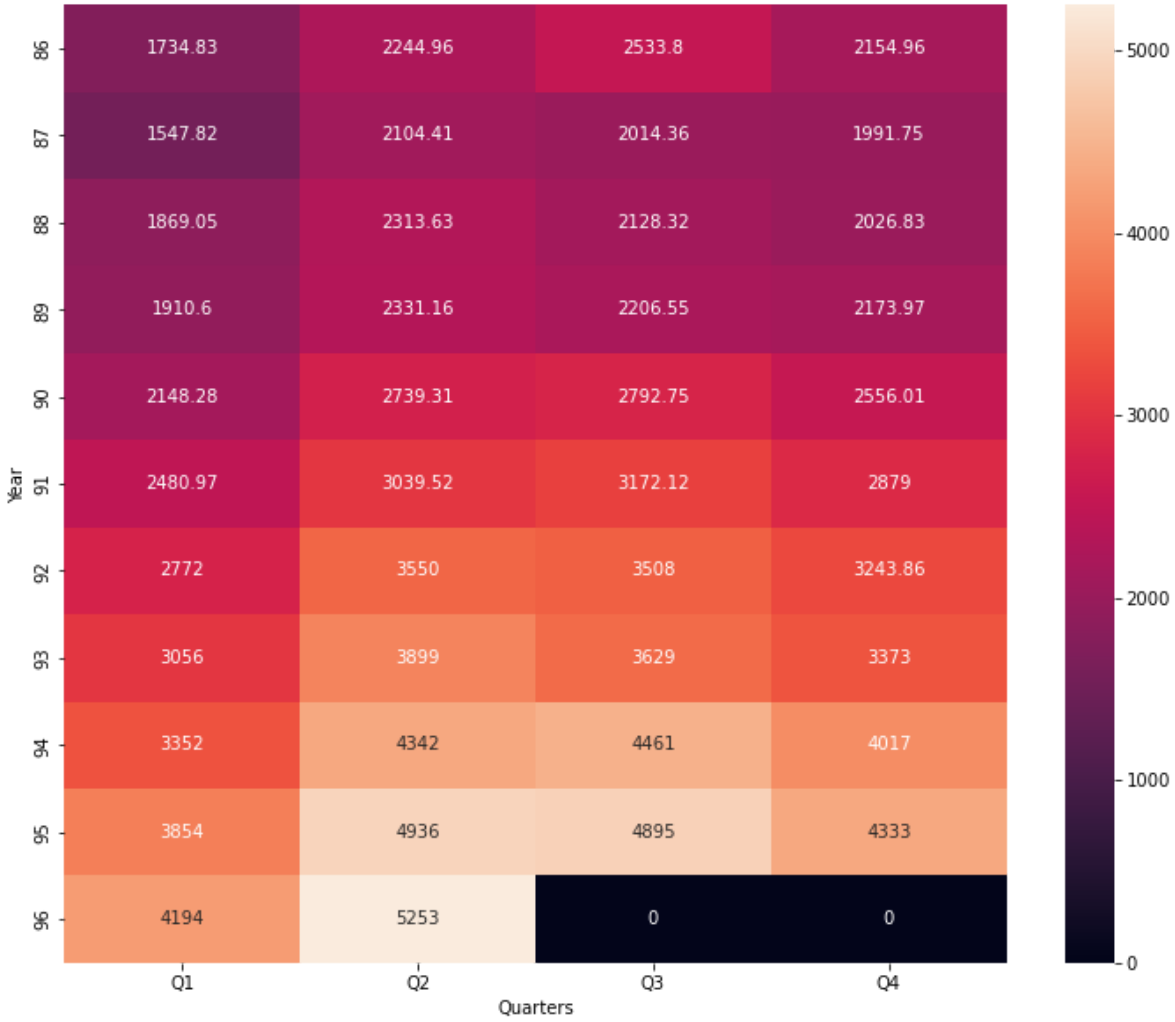
In [10]:

```
# Visualize the data

plt.figure(figsize=(12,10))
plot_month_y = pd.pivot_table(data = coke, values='Sales',index='Year', columns='Quarters',
sns.heatmap(plot_month_y, annot= True, fmt = 'g')
```

Out[10]:

```
<AxesSubplot:xlabel='Quarters', ylabel='Year'>
```

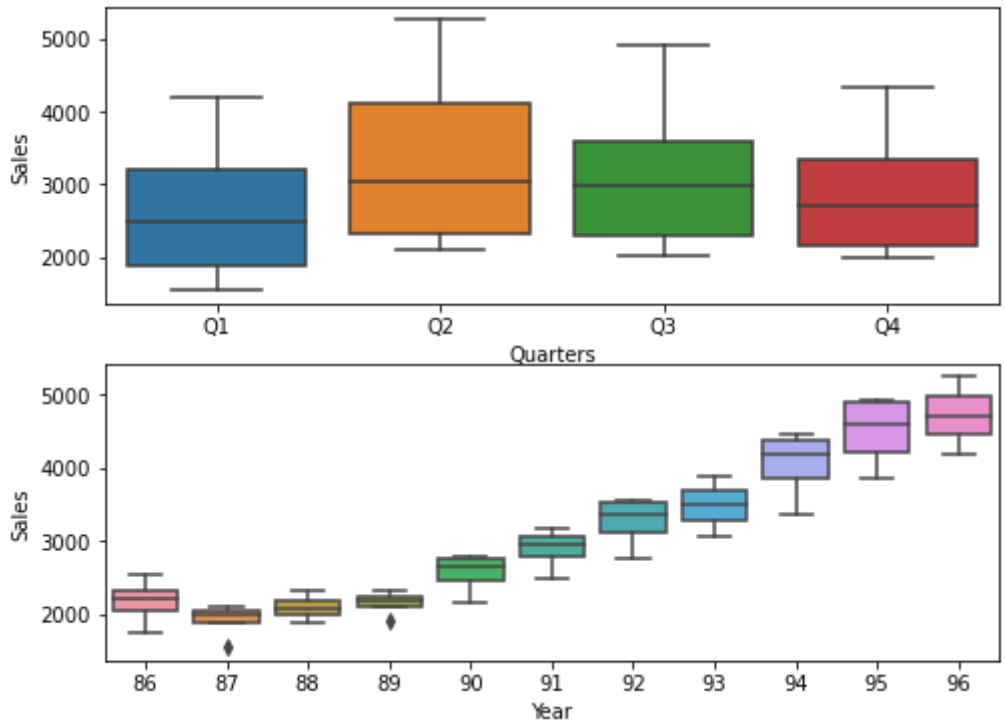


In [11]:

```
# Boxplot
plt.figure(figsize=(8,6))
plt.subplot(211)
sns.boxplot(x="Quarters",y="Sales",data=coke1)
plt.subplot(212)
sns.boxplot(x="Year",y="Sales",data=coke1)
```

Out[11]:

```
<AxesSubplot:xlabel='Year', ylabel='Sales'>
```



In [12]:

```
#Split data in Train and Test
Train = coke1.head(38)
Test = coke1.tail(4)
```

In [13]:

```
#Linear Model
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))
rmse_linear
```

Out[13]:

```
591.553295722396
```

In [14]:

```
#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))
rmse_Exp
```

Out[14]:

```
466.24797310672255
```

In [15]:

```
#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))
rmse_Quad
```

Out[15]:

```
475.56183518315254
```

In [16]:

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

Out[16]:

```
1860.0238154547276
```

In [17]:

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

Out[17]:

```
301.73800719346673
```

In [18]:

```
#Multiplicative Seasonality
Mul_sea = smf.ols('Log_Sales~Q1+Q2+Q3', 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)))**
rmse_Mult_sea
```

Out[18]:

```
1963.3896400779686
```

In [19]:

```
#Multiplicative Additive Seasonality
Mul_Add_sea = smf.ols('Log_Sales~t+Q1+Q2+Q3',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_
rmse_Mult_add_sea
```

Out[19]:

```
225.52439049817488
```

In [20]:

```
#Tabulating the rmse values
data = {'MODEL':pd.Series(['rmse_linear','rmse_Exp','rmse_Quad','rmse_add_sea','rmse_add_se
table_rmse = pd.DataFrame(data)
table_rmse.sort_values(['RMSE_Values'])
```

Out[20]:

	MODEL	RMSE_Values
6	rmse_Mult_add_sea	225.524390
4	rmse_add_sea_quad	301.738007
1	rmse_Exp	466.247973
2	rmse_Quad	475.561835
0	rmse_linear	591.553296
3	rmse_add_sea	1860.023815
5	rmse_Mult_sea	1963.389640

Conclustion:- From the above rmse values (rmse_Mult_add_sea 225.524390) is the best fit model

In [21]:

```
#Forecast for next 4 Quarters
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[21]:

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

In [22]:

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

dummies  = pd.DataFrame(pd.get_dummies(forecast['quarter']))
forecast1 = pd.concat([forecast, dummies], axis =1)
print('After dummy\n', forecast1.head())

forecast1['t'] = np.arange(1,5)
forecast1['t_squared'] = forecast1['t']* forecast1['t']
print('\nAfter T and T-Squared\n', forecast1.head())
```

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

After T and T-Squared
   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 [23]:

```
# Forecasting using Additive Seasonality Quadratic Model

model_full = smf.ols('Sales~ t+t_squared+Q1+Q2+Q3+Q4', data=coke1).fit()
pred_new = pd.Series(model_full.predict(forecast1))
pred_new

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

In [24]:

```
# Final sales prediction

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

Out[24]:

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

In [ ]:

```

```