Importing dataset

In [1]:

```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
from statsmodels.tsa.holtwinters import Holt
from statsmodels.tsa.holtwinters import ExponentialSmoothing
import statsmodels.graphics.tsaplots as tsa_plots
import statsmodels.tsa.statespace as tm_models
from datetime import datetime,time
from pylab import rcParams
from statsmodels.tsa.arima_model import ARIMA
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
import statsmodels.formula.api as smf
```

Importing Dataset

In [2]:

```
coca_data=pd.read_excel("CocaCola_Sales_Rawdata.xlsx")
coca_data
```

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

	Quarter	Sales
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

Business Problem.

Forecast the CocaCola prices data set. Prepare a document for each model explaining how many dummy variables you have created and RMSE value for each model. Finally which model you will use for Forecasting

Performing initila analysis

In [3]:

```
coca_data.isna().sum()
```

Out[3]:

```
Quarter    0
Sales      0
dtype: int64
```

In [4]:

```
coca_data.shape
```

Out[4]:

```
(42, 2)
```

In [5]:

```
coca_data.dtypes
```

Out[5]:

```
Quarter     object
Sales      float64
dtype: object
```

In [6]:

```
coca_data.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
Sales	42.0	2994.353308	977.930896	1547.818996	2159.714247	2782.376999	3609.25	5253.0

Data Preprocessing

In [7]:

```
replace = coca_data.Quarter.str.replace(r'(Q\d)_(\d+)', r'19\2-\1')
```

In [8]:

```
coca_data['quarter']=pd.to_datetime(replace).dt.strftime('%b-%Y')
```

In [9]:

```
coca_data
```

Out[9]:

	Quarter	Sales	quarter
0	Q1_86	1734.827000	Jan-1986
1	Q2_86	2244.960999	Apr-1986
2	Q3_86	2533.804993	Jul-1986
3	Q4_86	2154.962997	Oct-1986
4	Q1_87	1547.818996	Jan-1987
5	Q2_87	2104.411995	Apr-1987
6	Q3_87	2014.362999	Jul-1987
7	Q4_87	1991.746998	Oct-1987
8	Q1_88	1869.049999	Jan-1988
9	Q2_88	2313.631996	Apr-1988
10	Q3_88	2128.320000	Jul-1988
11	Q4_88	2026.828999	Oct-1988
12	Q1_89	1910.603996	Jan-1989
13	Q2_89	2331.164993	Apr-1989
14	Q3_89	2206.549995	Jul-1989
15	Q4_89	2173.967995	Oct-1989
16	Q1_90	2148.278000	Jan-1990
17	Q2_90	2739.307999	Apr-1990
18	Q3_90	2792.753998	Jul-1990
19	Q4_90	2556.009995	Oct-1990
20	Q1_91	2480.973999	Jan-1991
21	Q2_91	3039.522995	Apr-1991
22	Q3_91	3172.115997	Jul-1991
23	Q4_91	2879.000999	Oct-1991
24	Q1_92	2772.000000	Jan-1992
25	Q2_92	3550.000000	Apr-1992
26	Q3_92	3508.000000	Jul-1992
27	Q4_92	3243.859993	Oct-1992
28	Q1_93	3056.000000	Jan-1993
29	Q2_93	3899.000000	Apr-1993
30	Q3_93	3629.000000	Jul-1993
31	Q4_93	3373.000000	Oct-1993
32	Q1_94	3352.000000	Jan-1994
33	Q2_94	4342.000000	Apr-1994

	Quarter	Sales	quarter
34	Q3_94	4461.000000	Jul-1994
35	Q4_94	4017.000000	Oct-1994
36	Q1_95	3854.000000	Jan-1995
37	Q2_95	4936.000000	Apr-1995
38	Q3_95	4895.000000	Jul-1995
39	Q4_95	4333.000000	Oct-1995
40	Q1_96	4194.000000	Jan-1996
41	Q2_96	5253.000000	Apr-1996

In [10]:

```
#drop the Quarter column, because we get quarter columns on the basis of it.
coca_data=coca_data.drop(['Quarter'],axis=1)
```

In [11]:

```
coca_data.reset_index(inplace=True)
```

In [12]:

```
coca_data.dtypes
```

Out[12]:

```
index         int64
Sales       float64
quarter      object
dtype: object
```

In [13]:

```
coca_data['quarter']=pd.to_datetime(coca_data['quarter'])
```

In [14]:

```
coca_data = coca_data.set_index('quarter')
```

In [15]:

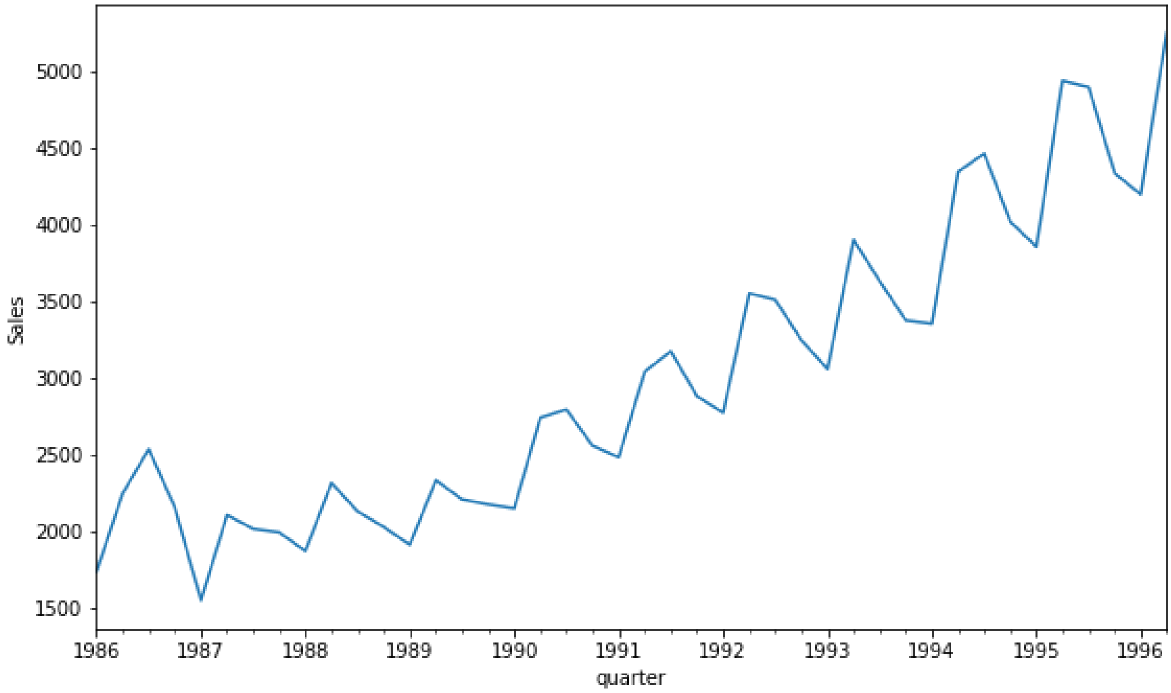
```
coca_data.head()
```

Out[15]:

quarter	index	Sales
1986-01-01	0	1734.827000
1986-04-01	1	2244.960999
1986-07-01	2	2533.804993
1986-10-01	3	2154.962997
1987-01-01	4	1547.818996

In [16]:

```
coca_data['Sales'].plot(figsize=(10,6))
plt.xlabel("quarter")
plt.ylabel("Sales")
plt.show()
```

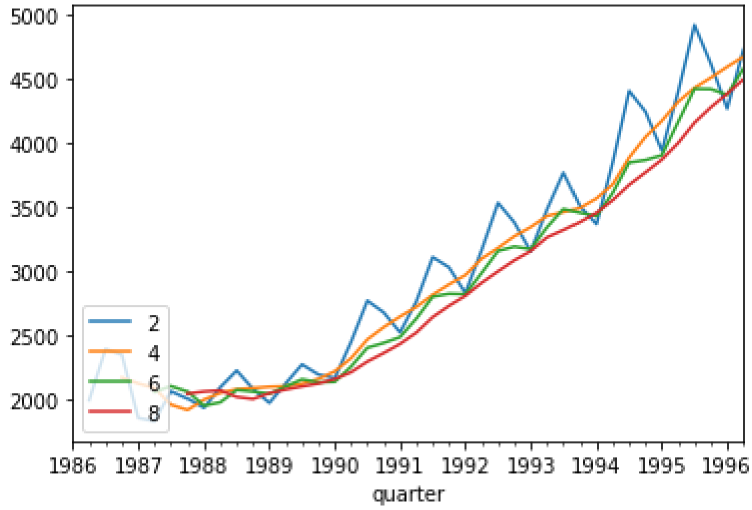


In [17]:

```
for i in range(2,10,2):
    coca_data["Sales"].rolling(i).mean().plot(label=str(i))
plt.legend(loc=3)
```

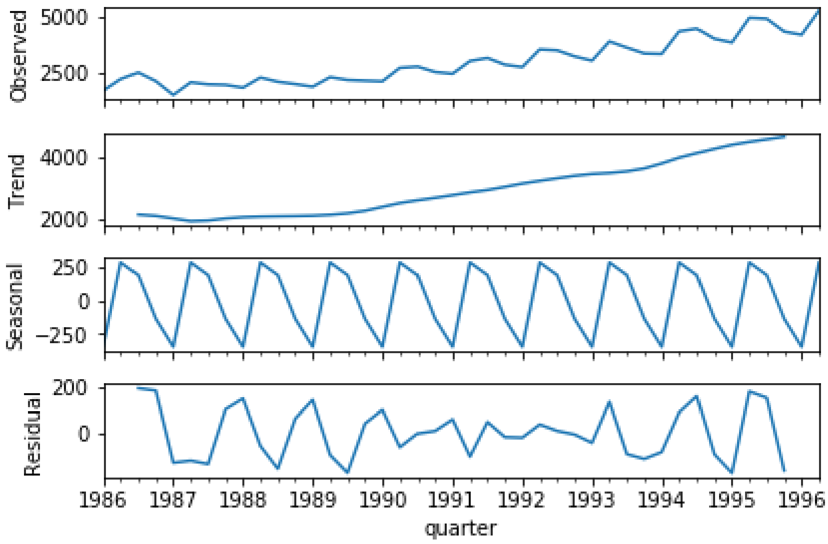
Out[17]:

<matplotlib.legend.Legend at 0x1caf7000bc8>



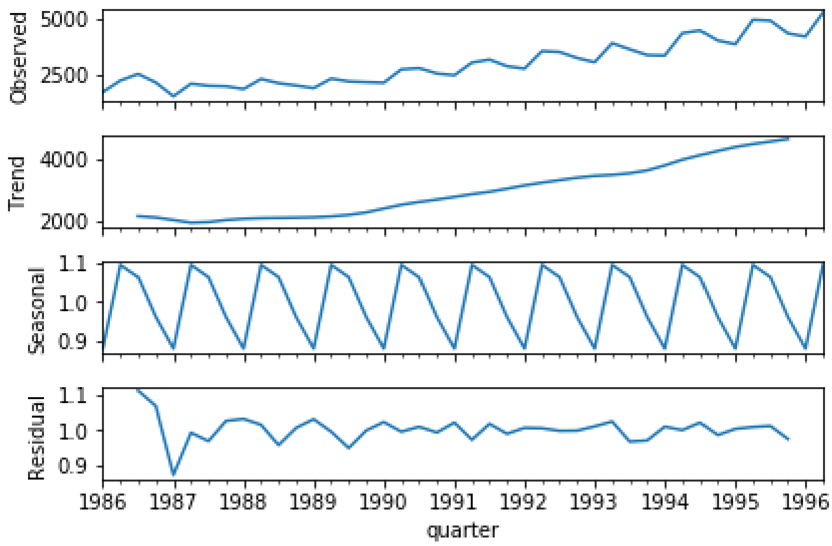
In [18]:

```
ts_add = seasonal_decompose(coca_data.Sales,model="additive")
fig = ts_add.plot()
plt.show()
```



In [19]:

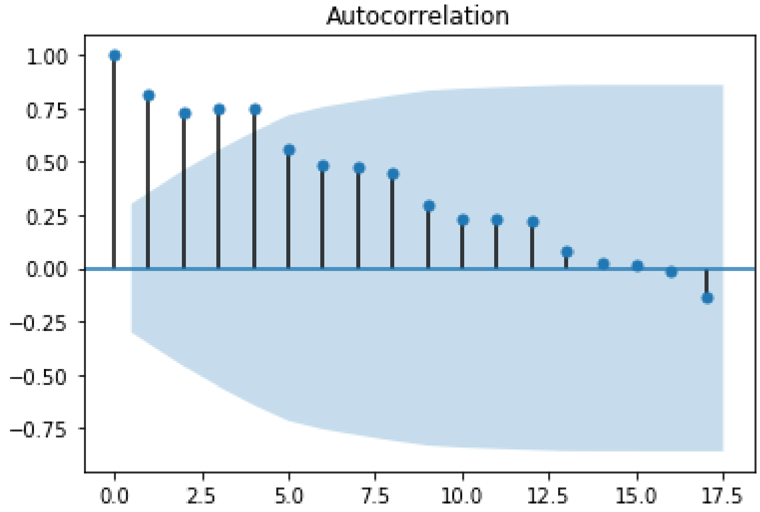
```
ts_mul = seasonal_decompose(coca_data.Sales,model="multiplicative")
fig = ts_mul.plot()
plt.show()
```

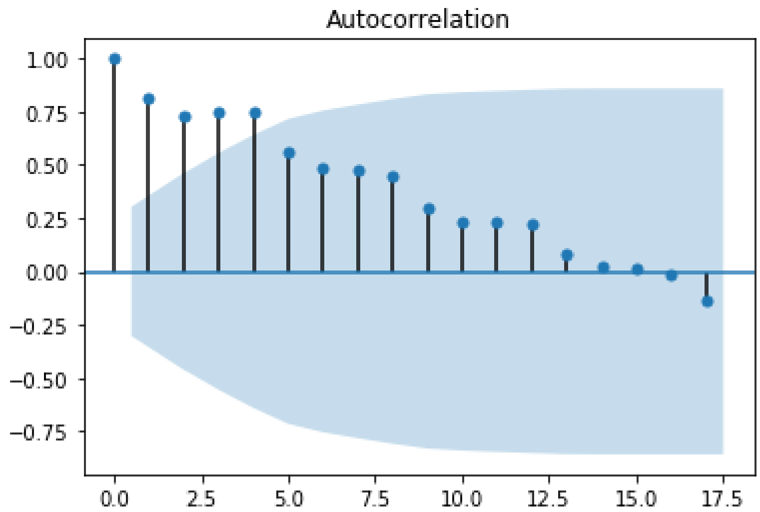


In [20]:

```
tsa_plots.plot_acf(coca_data.Sales)
```

Out[20]:





Building the Time series Forecasting with ARIMA

In [21]:

```
X=coca_data['Sales'].values
X
```

Out[21]:

```
array([1734.82699966, 2244.96099854, 2533.80499268, 2154.96299744,
       1547.81899643, 2104.41199493, 2014.36299896, 1991.74699783,
       1869.04999924, 2313.63199615, 2128.31999969, 2026.82899857,
       1910.60399628, 2331.16499329, 2206.54999542, 2173.96799469,
       2148.27799988, 2739.30799866, 2792.7539978 , 2556.00999451,
       2480.97399902, 3039.522995  , 3172.11599731, 2879.00099945,
       2772.        , 3550.        , 3508.        , 3243.85999298,
       3056.        , 3899.        , 3629.        , 3373.        ,
       3352.        , 4342.        , 4461.        , 4017.        ,
       3854.        , 4936.        , 4895.        , 4333.        ,
       4194.        , 5253.        ])
```

In [22]:

```
size = int(len(X) * 0.68)
```

In [23]:

```
train, test = X[0:size], X[size:len(X)]
```

In [24]:

```
train
```

Out[24]:

```
array([1734.82699966, 2244.96099854, 2533.80499268, 2154.96299744,
       1547.81899643, 2104.41199493, 2014.36299896, 1991.74699783,
       1869.04999924, 2313.63199615, 2128.31999969, 2026.82899857,
       1910.60399628, 2331.16499329, 2206.54999542, 2173.96799469,
       2148.27799988, 2739.30799866, 2792.7539978 , 2556.00999451,
       2480.97399902, 3039.522995  , 3172.11599731, 2879.00099945,
       2772.        , 3550.        , 3508.        , 3243.85999298])
```

In [25]:

```
model = ARIMA(train, order=(5,1,0))
```

In [26]:

```
model_fit=model.fit(disp=0)
```

In [27]:

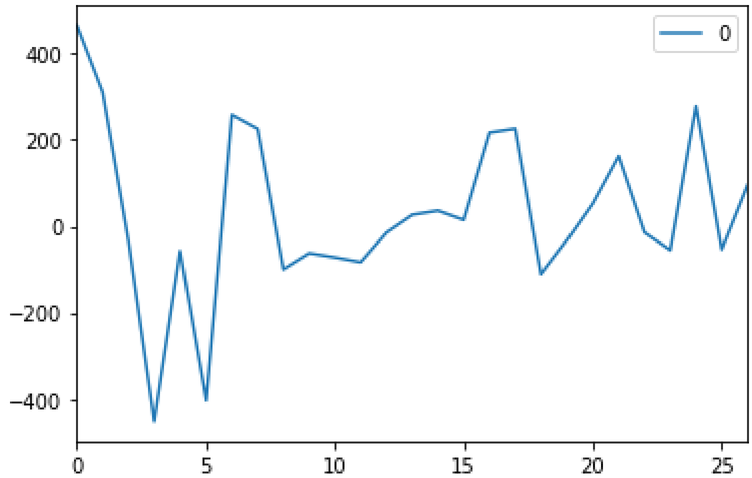
```
print(model_fit.summary())
```

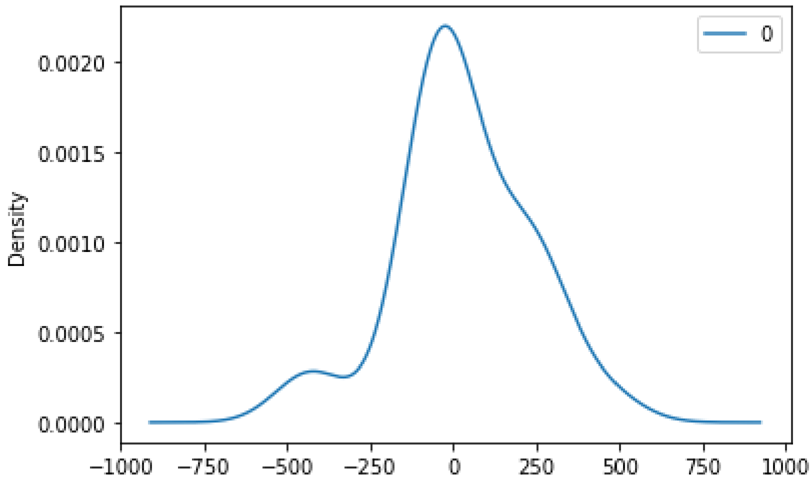
```
                              ARIMA Model Results
==============================================================================
==
Dep. Variable:                    D.y   No. Observations:
27
Model:                 ARIMA(5, 1, 0)   Log Likelihood                -178.2
39
Method:                       css-mle   S.D. of innovations            161.4
62
Date:                Thu, 23 Sep 2021   AIC                            370.4
79
Time:                        23:00:30   BIC                            379.5
49
Sample:                             1   HQIC                           373.1
76

==============================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const         44.7936     27.213      1.646      0.115      -8.543      98.1
30
ar.L1.D.y     -0.1524      0.193     -0.792      0.437      -0.530       0.2
25
ar.L2.D.y     -0.2894      0.150     -1.934      0.067      -0.583       0.0
04
ar.L3.D.y     -0.1733      0.168     -1.030      0.315      -0.503       0.1
57
ar.L4.D.y      0.6438      0.163      3.956      0.001       0.325       0.9
63
ar.L5.D.y     -0.1562      0.218     -0.716      0.482      -0.584       0.2
71
                                    Roots
==============================================================================
=
                  Real          Imaginary           Modulus         Frequenc
y
------------------------------------------------------------------------------
-
AR.1           -1.0442           -0.0000j            1.0442           -0.500
0
AR.2           -0.0427           -1.0172j            1.0181           -0.256
7
AR.3           -0.0427           +1.0172j            1.0181            0.256
7
AR.4            1.6364           -0.0000j            1.6364           -0.000
0
AR.5            3.6157           -0.0000j            3.6157           -0.000
0
------------------------------------------------------------------------------
-
```

In [28]:

```
residuals = pd.DataFrame(model_fit.resid)
residuals.plot()
pyplot.show()
residuals.plot(kind='kde')
pyplot.show()
print(residuals.describe())
```






```
                0
count   27.000000
mean    30.410359
std    199.847571
min   -452.041792
25%    -60.067411
50%    -13.994281
75%    189.526131
max    465.340373
```

Rolling Forecast ARIMA Model

In [29]:

```
history = [x for x in train]
```

In [30]:

```
predictions = list()
```

In [31]:

```
for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit(disp=0)
    output = model_fit.forecast()
    ypred = output[0]
    predictions.append(ypred)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (ypred, obs))
```

```
predicted=3188.845937, expected=3056.000000
predicted=3734.223958, expected=3899.000000
predicted=3782.622364, expected=3629.000000
predicted=3355.125380, expected=3373.000000
predicted=3297.216822, expected=3352.000000
predicted=4112.814175, expected=4342.000000

C:\Users\Tejas\Anaconda3\lib\site-packages\statsmodels\base\model.py:492: He
ssianInversionWarning: Inverting hessian failed, no bse or cov_params availa
ble
  'available', HessianInversionWarning)
C:\Users\Tejas\Anaconda3\lib\site-packages\statsmodels\base\model.py:512: Co
nvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

predicted=3961.108729, expected=4461.000000
predicted=4130.787183, expected=4017.000000
predicted=3912.795825, expected=3854.000000
predicted=4687.043782, expected=4936.000000
predicted=4970.520619, expected=4895.000000
predicted=4384.040096, expected=4333.000000

C:\Users\Tejas\Anaconda3\lib\site-packages\statsmodels\base\model.py:492: He
ssianInversionWarning: Inverting hessian failed, no bse or cov_params availa
ble
  'available', HessianInversionWarning)
C:\Users\Tejas\Anaconda3\lib\site-packages\statsmodels\base\model.py:512: Co
nvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

predicted=4229.064556, expected=4194.000000
predicted=5261.673785, expected=5253.000000

C:\Users\Tejas\Anaconda3\lib\site-packages\statsmodels\base\model.py:492: He
ssianInversionWarning: Inverting hessian failed, no bse or cov_params availa
ble
  'available', HessianInversionWarning)
C:\Users\Tejas\Anaconda3\lib\site-packages\statsmodels\base\model.py:512: Co
nvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)
```

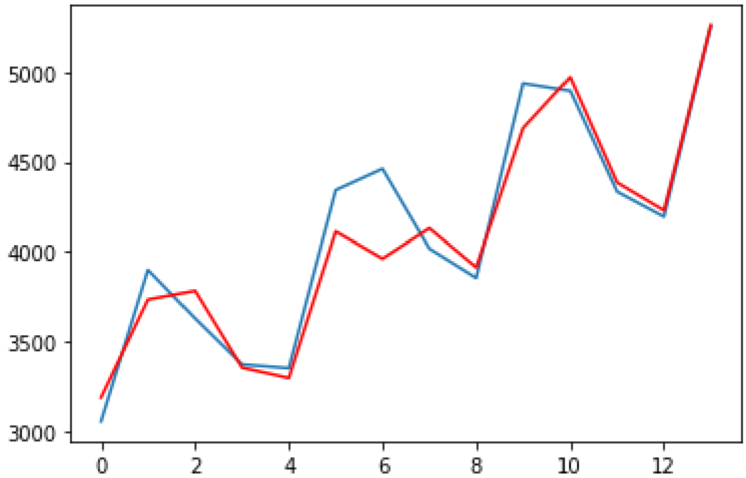
In [32]:

```
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)
```

Test MSE: 33009.573

In [33]:

```
pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
```



The value shows some trend are in correct sales.

Comparing Multiple Models

In [34]:

```
coca_data=pd.read_excel("CocaCola_Sales_Rawdata.xlsx")
coca_data
```

Out[34]:

	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

	Quarter	Sales
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 [35]:

```
coca_data1= pd.get_dummies(coca_data, columns = ['Quarter'])
```

In [36]:

```
coca_data1
```

Out[36]:

	Sales	Quarter_Q1_86	Quarter_Q1_87	Quarter_Q1_88	Quarter_Q1_89	Quarter_Q1
0	1734.827000	1	0	0	0	
1	2244.960999	0	0	0	0	
2	2533.804993	0	0	0	0	
3	2154.962997	0	0	0	0	
4	1547.818996	0	1	0	0	
5	2104.411995	0	0	0	0	
6	2014.362999	0	0	0	0	
7	1991.746998	0	0	0	0	
8	1869.049999	0	0	1	0	
9	2313.631996	0	0	0	0	
10	2128.320000	0	0	0	0	
11	2026.828999	0	0	0	0	
12	1910.603996	0	0	0	1	
13	2331.164993	0	0	0	0	
14	2206.549995	0	0	0	0	
15	2173.967995	0	0	0	0	
16	2148.278000	0	0	0	0	
17	2739.307999	0	0	0	0	
18	2792.753998	0	0	0	0	
19	2556.009995	0	0	0	0	
20	2480.973999	0	0	0	0	
21	3039.522995	0	0	0	0	
22	3172.115997	0	0	0	0	
23	2879.000999	0	0	0	0	
24	2772.000000	0	0	0	0	
25	3550.000000	0	0	0	0	
26	3508.000000	0	0	0	0	
27	3243.859993	0	0	0	0	
28	3056.000000	0	0	0	0	
29	3899.000000	0	0	0	0	
30	3629.000000	0	0	0	0	
31	3373.000000	0	0	0	0	
32	3352.000000	0	0	0	0	
33	4342.000000	0	0	0	0	

	Sales	Quarter_Q1_86	Quarter_Q1_87	Quarter_Q1_88	Quarter_Q1_89	Quarter_Q1
34	4461.000000	0	0	0	0	
35	4017.000000	0	0	0	0	
36	3854.000000	0	0	0	0	
37	4936.000000	0	0	0	0	
38	4895.000000	0	0	0	0	
39	4333.000000	0	0	0	0	
40	4194.000000	0	0	0	0	
41	5253.000000	0	0	0	0	

42 rows × 43 columns

In [37]:

```
,'Q3','Q3','Q3','Q3','Q3','Q3','Q3','Q3','Q3','Q3','Q4','Q4','Q4','Q4','Q4','Q4','Q4','Q4','
```

In [38]:

```
t= np.arange(1,43)
```

In [39]:

```
coca_data1['t'] = t
```

In [40]:

```
coca_data1['t_sq'] = coca_data1['t']*coca_data1['t']
```

In [41]:

```
log_Sales=np.log(coca_data1['Sales'])
```

In [42]:

```
coca_data1['log_Sales']=log_Sales
```

In [43]:

```
coca_data1.head()
```

Out[43]:

	Sales	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	...	Q4	Q4	Q4	Q4	Q4	Q4	Q
0	1734.827000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
1	2244.960999	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
2	2533.804993	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
3	2154.962997	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
4	1547.818996	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	

5 rows × 46 columns

In [44]:

```
train1, test1 = np.split(coca_data1, [int(.68 *len(coca_data1))])
```

In [45]:

```
# Linear Model
linear= smf.ols('Sales ~ t',data=train1).fit()
predlin=pd.Series(linear.predict(pd.DataFrame(test1['t'])))
rmselin=np.sqrt((np.mean(np.array(test1['Sales'])-np.array(predlin))**2))
rmselin
```

Out[45]:

580.1224130918637

In [46]:

```
#Quadratic Model
quad=smf.ols('Sales~t+t_sq',data=train1).fit()
predquad=pd.Series(quad.predict(pd.DataFrame(test1[['t','t_sq']])))
rmsequad=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(predquad))**2))
rmsequad
```

Out[46]:

783.7297975037425

In [47]:

```
#Exponential Model
expo=smf.ols('log_Sales~t',data=train1).fit()
predexp=pd.Series(expo.predict(pd.DataFrame(test1['t'])))
rmseexpo=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(np.exp(predexp)))**2))
rmseexpo
```

Out[47]:

588.1405104900183

In [48]:

```
#Additive Model
additive= smf.ols('Sales~ Q1+Q2+Q3+Q4',data=train1).fit()
predadd=pd.Series(additive.predict(pd.DataFrame(test1[['Q1','Q2','Q3','Q4']])))
rmseadd=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(predadd))**2))
rmseadd
```

Out[48]:

```
1869.7188209186943
```

In [49]:

```
#Additive Linear
addlinear= smf.ols('Sales~t+Q1+Q2+Q3+Q4',data=train1).fit()
predaddlinear=pd.Series(addlinear.predict(pd.DataFrame(test1[['t','Q1','Q2','Q3','Q4']])))
rmseaddlinear=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(predaddlinear))**2))
rmseaddlinear
```

Out[49]:

```
596.1526282372248
```

In [50]:

```
#Additive sesonaly Quadratic
addquad=smf.ols('Sales~t+t_sq+Q1+Q2+Q3+Q4',data=train1).fit()
predaddquad=pd.Series(addquad.predict(pd.DataFrame(test1[['t','t_sq','Q1','Q2','Q3','Q4']])
rmseaddquad=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(predaddquad))**2))
rmseaddquad
```

Out[50]:

```
412.1144436052239
```

In [51]:

```
#Multiplicative seasonaly
mulsea=smf.ols('log_Sales~Q1+Q2+Q3+Q4',data=train1).fit()
predmul= pd.Series(mulsea.predict(pd.DataFrame(test1[['Q1','Q2','Q3','Q4']])))
rmsemul= np.sqrt(np.mean((np.array(test1['Sales'])-np.array(np.exp(predmul)))**2))
rmsemul
```

Out[51]:

```
2374.919440795434
```

In [52]:

```
#Multiplicative Linear
mullin= smf.ols('log_Sales~t+Q1+Q2+Q3+Q4',data=train1).fit()
predmullin= pd.Series(mullin.predict(pd.DataFrame(test1[['t','Q1','Q2','Q3','Q4']])))
rmsemulin=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(np.exp(predmullin)))**2))
rmsemulin
```

Out[52]:

```
5359.687911933539
```

In [53]:

```
#Multiplicative Quadratic
mul_quad= smf.ols('log_Sales~t+t_sq+Q1+Q2+Q3+Q4',data=train1).fit()
pred_mul_quad= pd.Series(mul_quad.predict(test1[['t','t_sq','Q1','Q2','Q3','Q4']]))
rmse_mul_quad=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(np.exp(pred_mul_quad)))**2
rmse_mul_quad
```

Out[53]:

```
3630.5619467339175
```

Checking each value of model and declare the value which is lowest than rest of others, we called it is best RMSE value.

In [54]:

```
'Model':pd.Series(['rmse_mul_quad','rmseadd','rmseaddlinear','rmseaddquad','rmseexpo','rmse
'Values':pd.Series([rmse_mul_quad,rmseadd,rmseaddlinear,rmseaddquad,rmseexpo,rmselin,rmsemu
```

In [55]:

```
rmse=pd.DataFrame(output)
```

In [56]:

```
print(output)
```

```
{'Model': 0     rmse_mul_quad
1           rmseadd
2     rmseaddlinear
3       rmseaddquad
4          rmseexpo
5           rmselin
6           rmsemul
7         rmsemulin
8          rmsequad
dtype: object, 'Values': 0    3630.561947
1     1869.718821
2      596.152628
3      412.114444
4      588.140510
5      580.122413
6     2374.919441
7     5359.687912
8      783.729798
dtype: float64}
```

Solution of Business problem: we have created dummy variables of quarter.

Additive seasonaly quadratic has a best RMSE value.