Import neccessery libraries

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
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
import warnings
import itertools
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
import pandas as pd
import statsmodels.api as sm
import matplotlib
from pylab import rcParams
from statsmodels.tsa.arima_model import ARIMA
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
```

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

Import data

In [6]:

```python
coca_cola = pd.read_excel('CocaCola_Sales_Rawdata.xlsx')
```

In [7]:

In [8]:

Out[8]:

	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

Data understanding

In [9]:

Out[9]:

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

In [11]:

Out[11]:

```
(42, 2)
```

In [12]:

Out[12]:

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

In [13]:

Out[13]:

	count	mean	std	min	25%	50%	75%	ma
Sales	42.0	2994.353308	977.930896	1547.818996	2159.714247	2782.376999	3609.25	5253

In [14]:

In [15]:

In [16]:

Out[16]:

	Quarter	Sales	quater
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

In [17]:

In [18]:

In [20]:

In [21]:

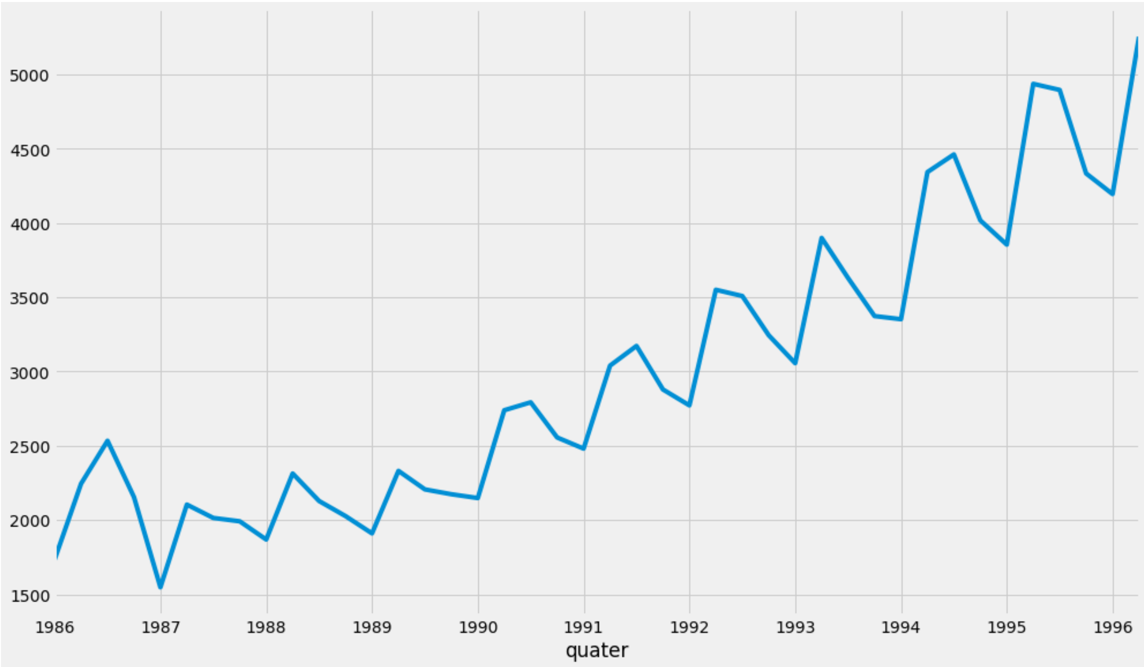
In [22]:

Out[22]:

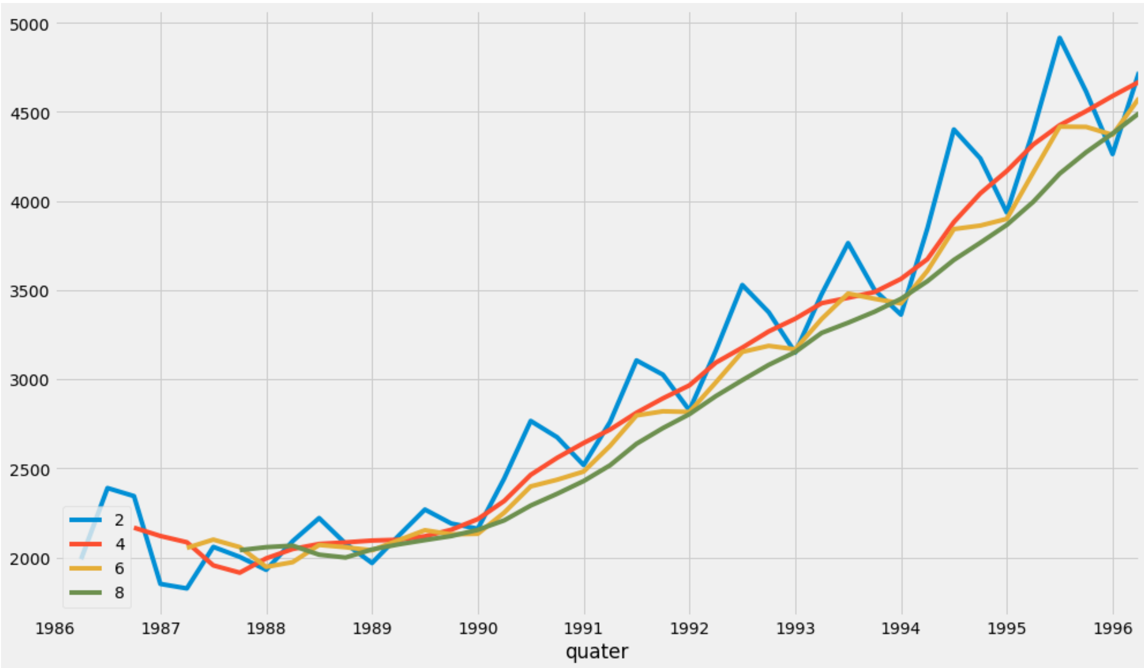
	index	Sales
quater		
1986-01-01	0	1734.827000

	index	Sales
quater		
1986-04-01	1	2244.960999
1986-07-01	2	2533.804993

In [24]:
```
cola_data['Sales'].plot(figsize=(15, 9))
plt.show()
```

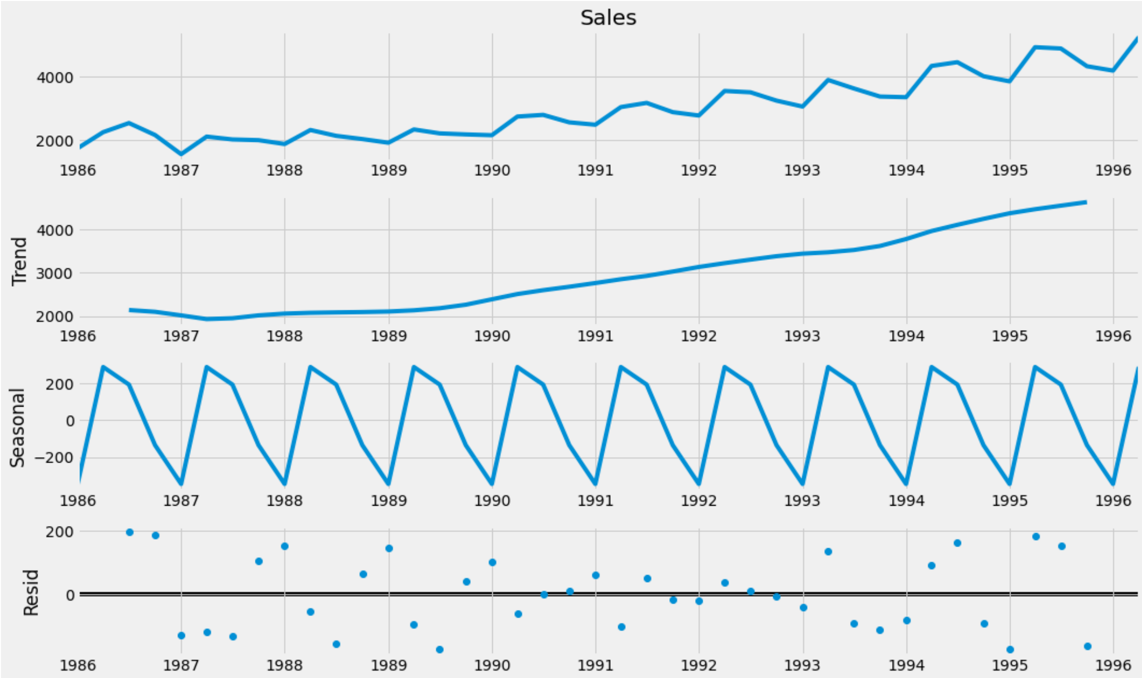


In [31]:
```
for i in range(2,10,2):
    cola_data["Sales"].rolling(i).mean().plot(figsize=(15, 9),label=str
plt.legend(loc=3)
plt.show()
```

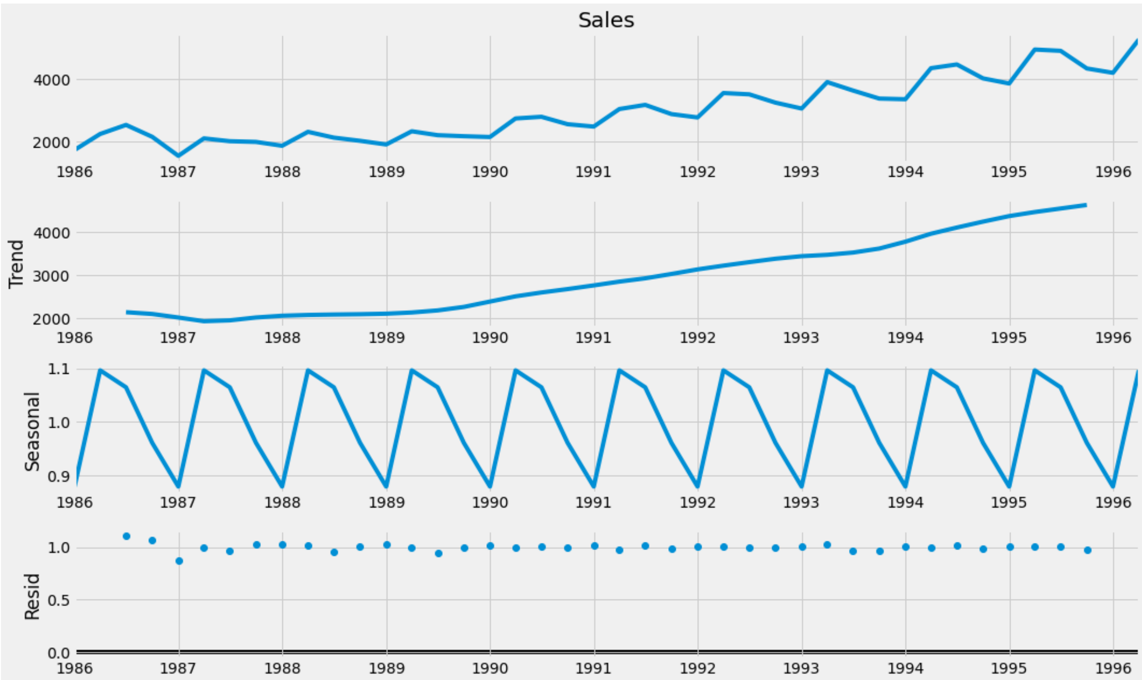


In [47]:
```
rcParams['figure.figsize'] = 15,9
```

In [48]:
```
ts_add = seasonal_decompose(cola_data.Sales,model="additive")
fig = ts_add.plot()
plt.show()
```

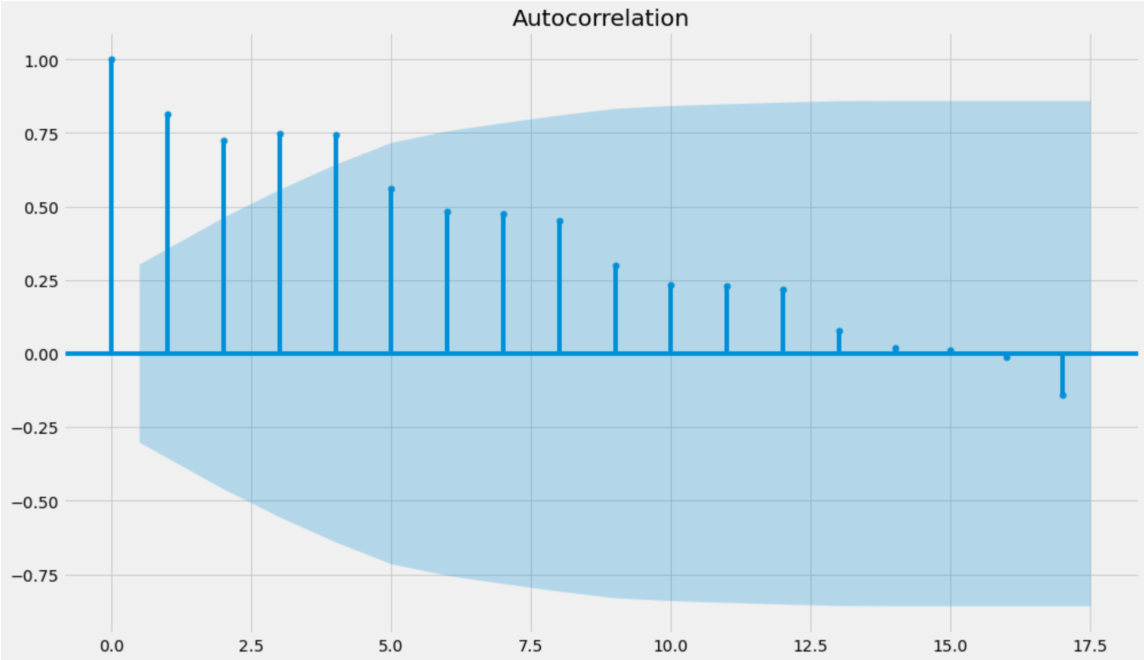


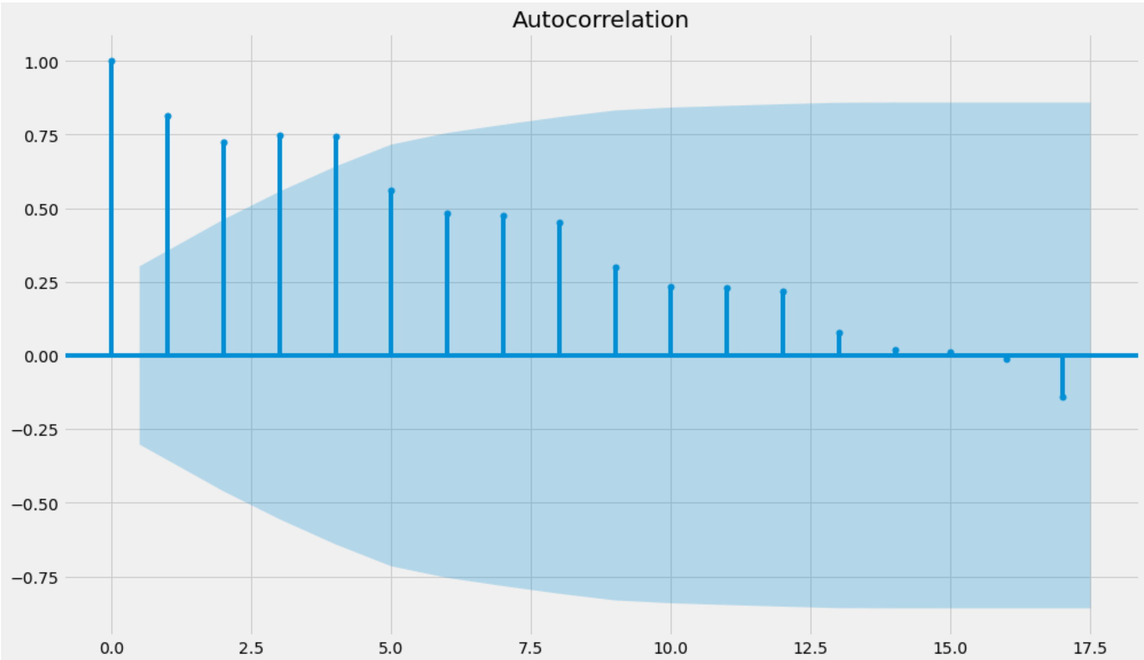
In [49]:
```
ts_mul = seasonal_decompose(cola_data.Sales,model="multiplicative")
fig = ts_mul.plot()
plt.show()
```



In [51]:

Out[51]:





Building Time series forecasting with ARIMA

In [52]:

In [53]:

In [54]:

In [55]:

In [56]:

In [57]:

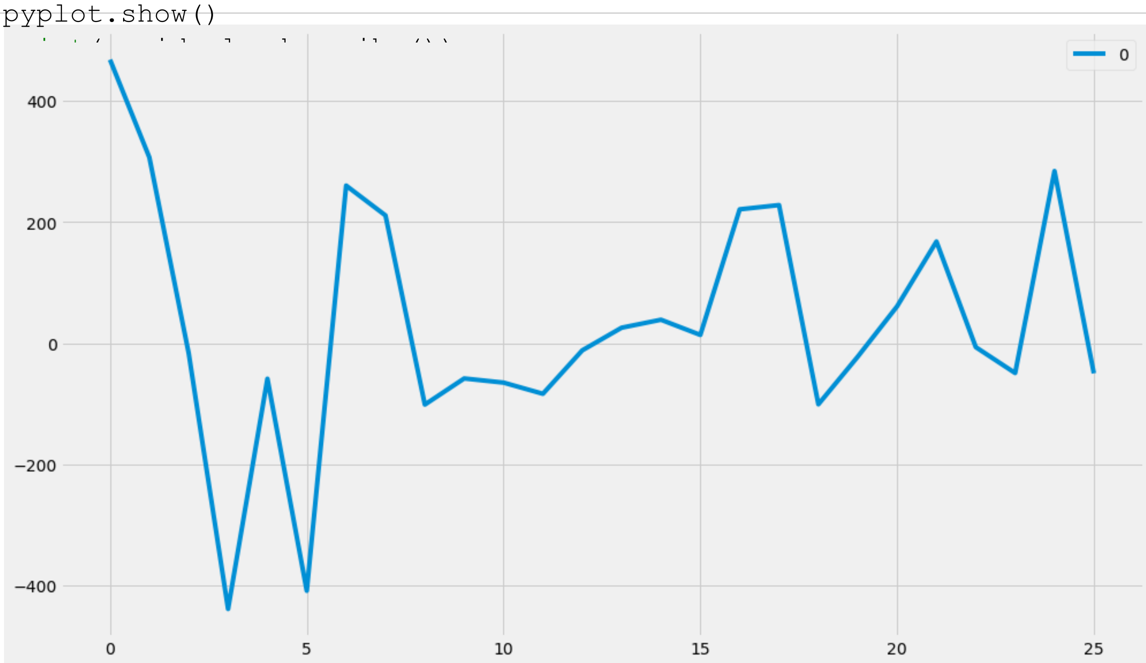
```
print(model_fit.summary())
```

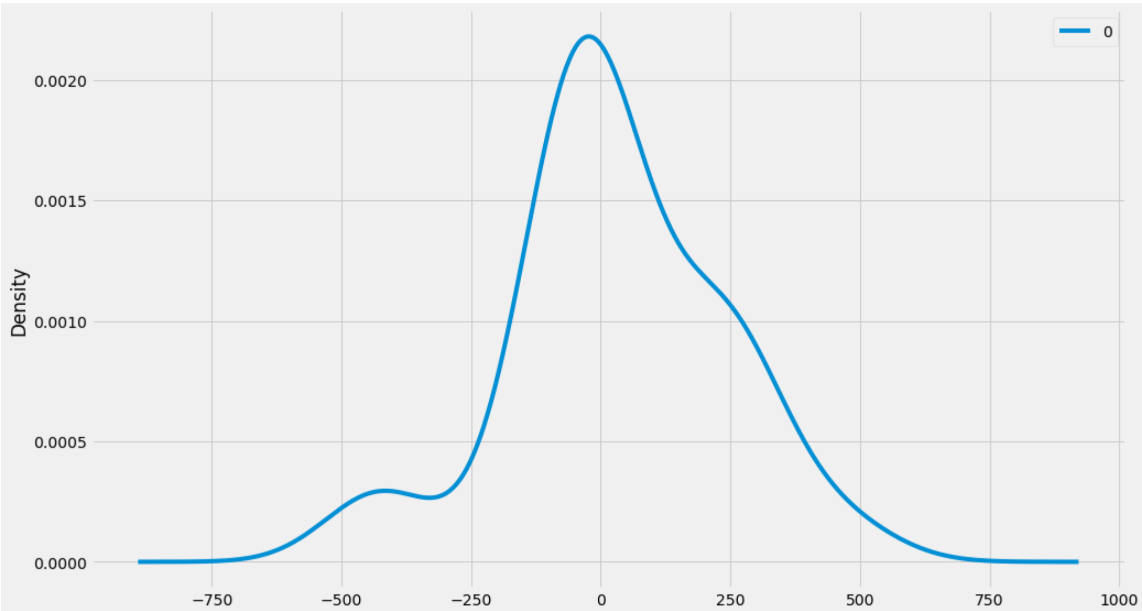
```
                              ARIMA Model Results
==============================================================================
Dep. Variable:                    D.y   No. Observations:
26
Model:                 ARIMA(5, 1, 0)   Log Likelihood
-172.036
Method:                       css-mle   S.D. of innovations
163.191
Date:                Sat, 26 Mar 2022   AIC
358.071
Time:                        12:57:07   BIC
366.878
Sample:                             1   HQIC
360.607

==============================================================================
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
const         41.8434     26.509      1.578      0.114     -10.113
93.799
ar.L1.D.y     -0.1479      0.195     -0.758      0.448      -0.530
0.234
ar.L2.D.y     -0.3127      0.157     -1.996      0.046      -0.620
-0.006
ar.L3.D.y     -0.1881      0.173     -1.090      0.276      -0.526
0.150
ar.L4.D.y      0.6222      0.167      3.716      0.000       0.294
0.950
ar.L5.D.y     -0.1766      0.220     -0.804      0.422      -0.607
0.254
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus
Frequency
-----------------------------------------------------------------------------
AR.1           -1.0476           -0.0000j            1.0476
-0.5000
AR.2           -0.0437           -1.0161j            1.0170
-0.2568
AR.3           -0.0437           +1.0161j            1.0170
0.2568
AR.4            1.8835           -0.0000j            1.8835
-0.0000
AR.5            2.7754           -0.0000j            2.7754
-0.0000
-----------------------------------------------------------------------------
```

This summarizes the coefficient values used as well as the skill of the fit on the on the in-sample observations

In [58]:
```
residuals = pd.DataFrame(model_fit.resid)
```

```
residuals.plot()
pyplot.show()
residuals.plot(kind='kde')
pyplot.show()
```






```
                0
count   26.000000
mean    31.325878
std    202.029702
min   -438.904389
25%    -58.603106
50%     -9.190229
75%    200.236018
max    468.290585
```

**The plot of the residual errors suggests that there may still be some trend information not captured by the model

The results show that indeed there is a bias in the prediction (a non-zero mean in the residuals)

Rolling Forecast ARIMA Model

In [59]:

In [60]:

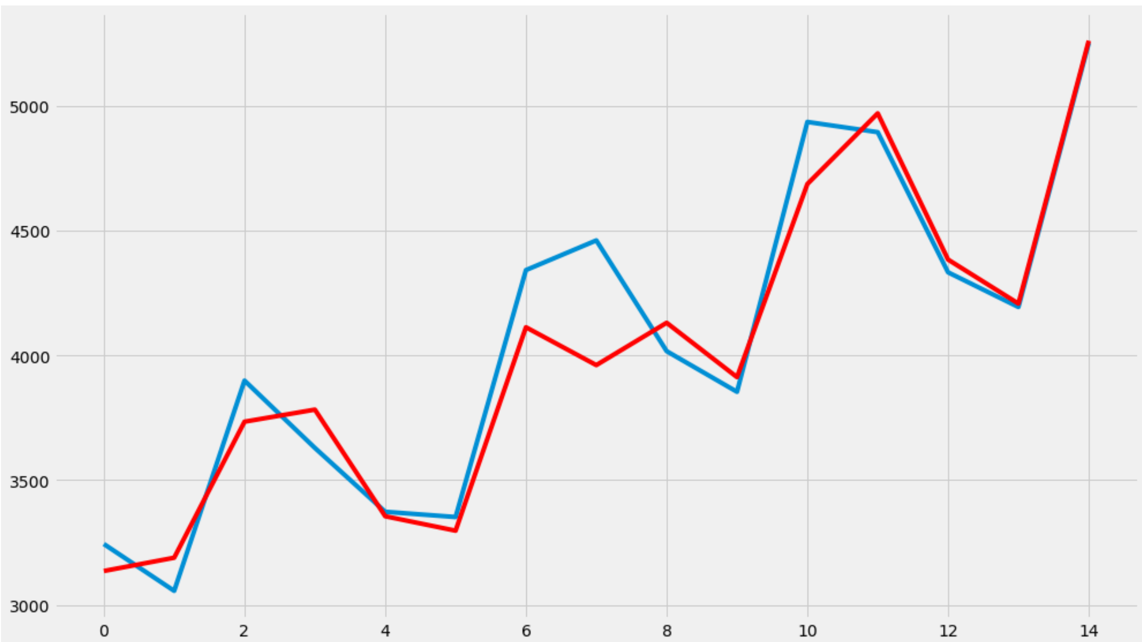
In [61]:
```
for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit(disp=0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
```

```
predicted=3135.586029, expected=3243.859993
predicted=3188.847068, expected=3056.000000
predicted=3734.224502, expected=3899.000000
predicted=3782.620891, expected=3629.000000
predicted=3355.125969, expected=3373.000000
predicted=3297.218120, expected=3352.000000
predicted=4112.813891, expected=4342.000000
predicted=3961.043678, expected=4461.000000
predicted=4130.787225, expected=4017.000000
predicted=3912.794182, expected=3854.000000
predicted=4687.043733, expected=4936.000000
predicted=4970.516924, expected=4895.000000
predicted=4384.040534, expected=4333.000000
predicted=4207.687405, expected=4194.000000
predicted=5261.673040, expected=5253.000000
```

In [62]:
```
error = mean_squared_error(test, predictions)
```

```
Test MSE: 31525.273
```

In [63]:
```
pyplot.plot(test)
pyplot.plot(predictions, color='red')
```



A line plot is created showing the expected values (blue) compared to the rolling forecast predictions (red). We can see the values show some trend and are in the correct scale

Comparing Multiple Models

In [64]:

In [68]:

```
data.columns = ['Sales','Q1','Q1','Q1','Q1','Q1','Q1','Q1','Q1','Q1','Q
```

In [69]:

Out[69]:

	Sales	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	...	Q3	Q3	Q3	Q4	Q4	Q4	Q4
0	1734.827000	1	0	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	0
2	2533.804993	0	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	...	1	0	0	0	0	0	0
4	1547.818996	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

5 rows × 43 columns

In [70]:

In [72]:

In [73]:

In [74]:

In [75]:

In [76]:

Out[76]:

	Sales	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	...	Q4	Q4	Q4	Q4	Q4	Q4	Q4
0	1734.827000	1	0	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	0
2	2533.804993	0	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	0
4	1547.818996	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

5 rows × 46 columns

In [77]:

In [78]:
```
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
```
Out[78]: 580.1224130918641

In [79]:
```
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))*
```
Out[79]: 783.7297975037103

In [80]:
```
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(pred
rmseexpo
```
Out[80]: 588.1405104900134

In [81]:
```
additive= smf.ols('Sales~ Q1+Q2+Q3+Q4',data=train1).fit()
predadd=pd.Series(additive.predict(pd.DataFrame(test1[['Q1','Q2','Q3',
rmseadd=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(predadd))**2
rmseadd
```
Out[81]: 1869.7188209186947

In [82]:
```
addlinear= smf.ols('Sales~t+Q1+Q2+Q3+Q4',data=train1).fit()
predaddlinear=pd.Series(addlinear.predict(pd.DataFrame(test1[['t','Q1',
rmseaddlinear=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(predad
rmseaddlinear
```
Out[82]: 596.1526282372472

In [83]:
```
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','Q
rmseaddquad=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(predaddq
rmseaddquad
```
Out[83]: 412.1144436053775

In [84]:
```
mulsea=smf.ols('log_Sales~Q1+Q2+Q3+Q4',data=train1).fit()
predmul= pd.Series(mulsea.predict(pd.DataFrame(test1[['Q1','Q2','Q3','Q
rmsemul= np.sqrt(np.mean((np.array(test1['Sales'])-np.array(np.exp(pred
rmsemul
```
Out[84]: 2374.9194407954374

In [85]:
```
mullin= smf.ols('log_Sales~t+Q1+Q2+Q3+Q4',data=train1).fit()
predmullin= pd.Series(mullin.predict(pd.DataFrame(test1[['t','Q1','Q2',
rmsemulin=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(np.exp(pre
rmsemulin
```
Out[85]: 5359.687911932085

In [86]:
```
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',
rmse_mul_quad=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(np.exp
```

```
rmse_mul_quad
```

Out[86]: 3630.5619467347524

Conclusion

In [87]:

```
output = {'Model':pd.Series(['rmse_mul_quad','rmseadd','rmseaddlinear',
          'Values':pd.Series([rmse_mul_quad,rmseadd,rmseaddlinear,rmsea
```

In [88]:

```
```

In [89]:

```
```

	Model	Values
0	rmse_mul_quad	3630.561947
1	rmseadd	1869.718821
2	rmseaddlinear	596.152628
3	rmseaddquad	412.114444
4	rmseexpo	588.140510
5	rmselin	580.122413
6	rmsemul	2374.919441
7	rmsemulin	5359.687912
8	rmsequad	783.729798

Additive seasonality with quadratic trend has the best RMSE value

In [ ]: