In [1]:
```
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
import statsmodels.formula.api as smf
```

```
Matplotlib is building the font cache; this may take a moment.
```

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

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

	Quarter	Sales
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]:
```
coca1 = coca.copy()
```

In [4]:
```
coca1.head().T
```

Out[4]:

	0	1	2	3	4
Quarter	Q1_86	Q2_86	Q3_86	Q4_86	Q1_87
Sales	1734.827	2244.960999	2533.804993	2154.962997	1547.818996

In [5]:
```
coca1.isnull().sum()
```

Out[5]:
```
Quarter    0
Sales      0
dtype: int64
```

In [6]:
```
coca1.shape
```

Out[6]:
```
(42, 2)
```

In [7]:
```
coca1.describe()
```

Out[7]:

	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 [8]:
```
coca1.dtypes
```
Out[8]:
```
Quarter      object
Sales       float64
dtype: object
```

In [9]:
```
coca1.describe().T
```
Out[9]:

	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

In [10]:
```
temp = coca1.Quarter.str.replace(r'(Q\d)_(\d+)', r'19\2-\1')
```

In [11]:
```
coca1['quater'] = pd.to_datetime(temp).dt.strftime('%b-%Y')
```

In [12]:
```
coca1.head()
```
Out[12]:

	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 [13]:
```
coca1 = coca1.drop(['Quarter'], axis=1)
```

In [14]:
```
coca1.reset_index(inplace=True)
```

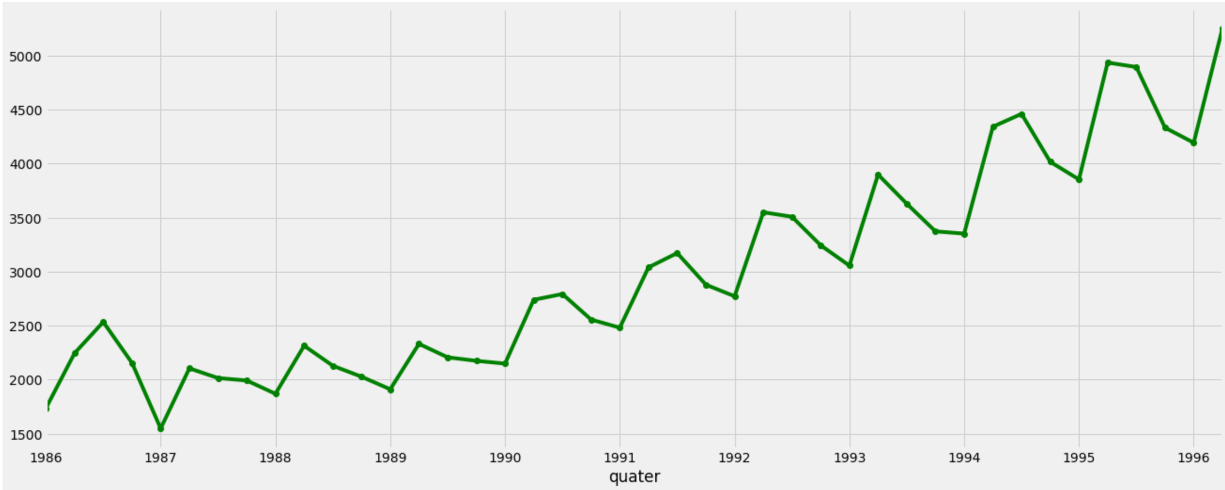
In [15]:
```
coca1['quater'] = pd.to_datetime(coca1['quater'])
```

In [16]:
```
coca1 = coca1.set_index('quater')
```

In [17]:
```
coca1.head()
```
Out[17]:

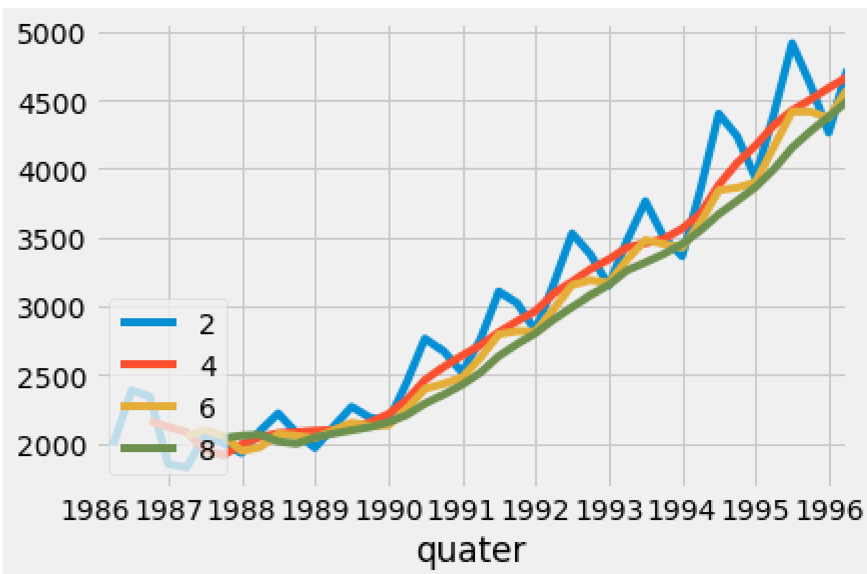
	index	Sales
quater		
1986-01-01	0	1734.827000
1986-04-01	1	2244.960999
1986-07-01	2	2533.804993
1986-10-01	3	2154.962997
1987-01-01	4	1547.818996

In [18]:
```
coca1['Sales'].plot(figsize=(20, 8),color='green',marker='o')
plt.show()
```

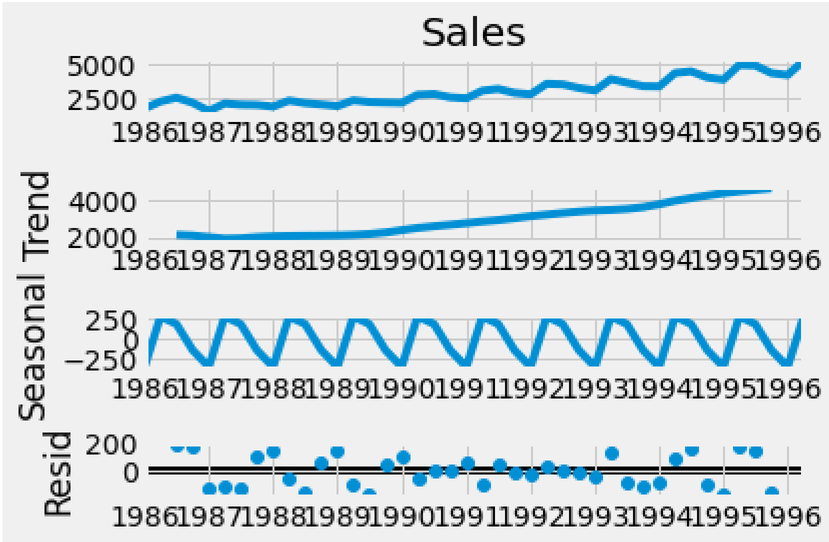


In [19]:
```
for i in range(2,10,2):
    coca1["Sales"].rolling(i).mean().plot(label=str(i))
plt.legend(loc=3)
```

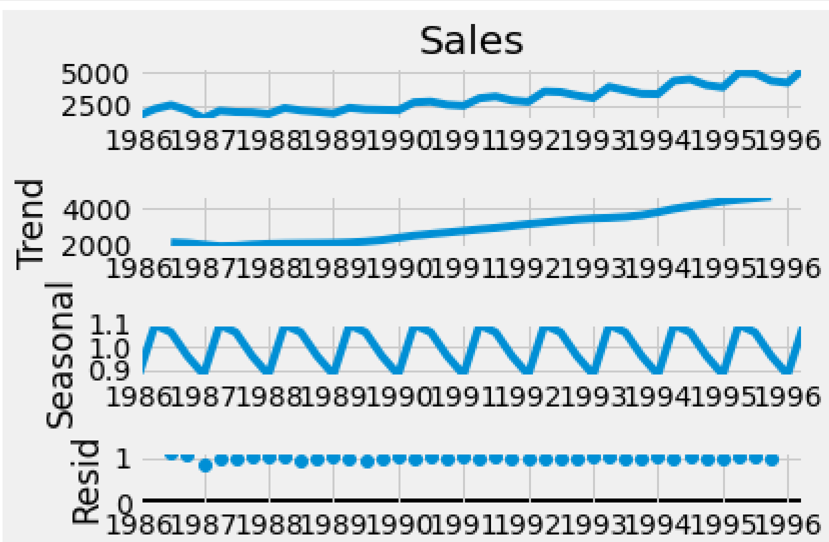
Out[19]: <matplotlib.legend.Legend at 0x40284d76a0>



In [20]:
```
ts_add = seasonal_decompose(coca1.Sales,model="additive")
fig = ts_add.plot()
plt.show()
```

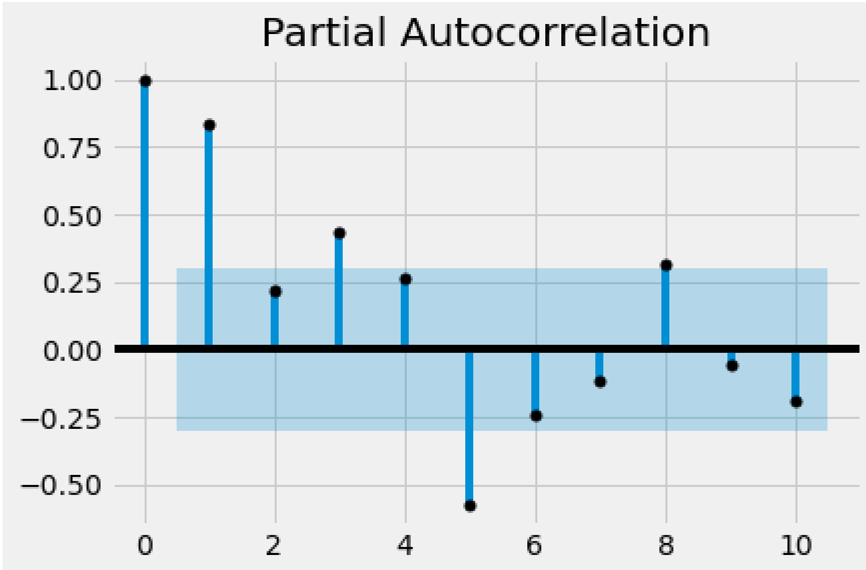


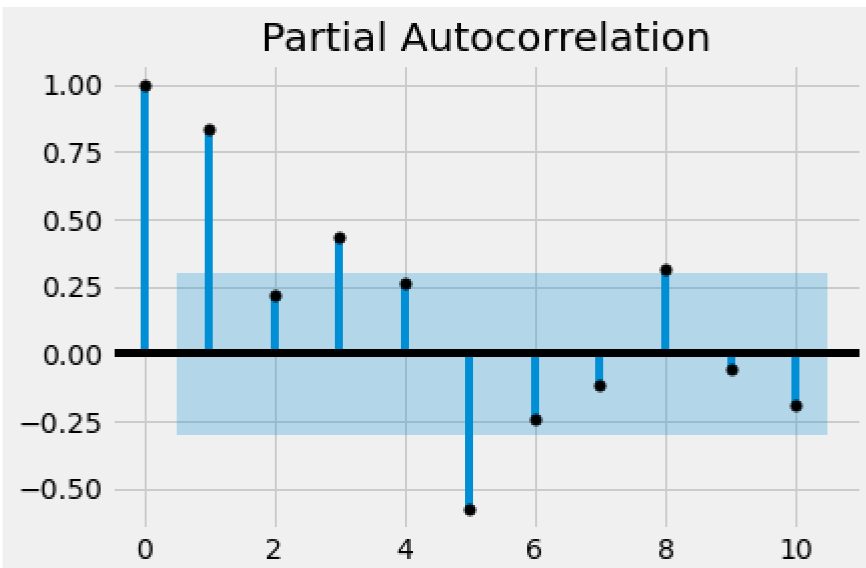
In [21]:
```
ts_mul = seasonal_decompose(coca1.Sales,model="multiplicative")
fig = ts_mul.plot()
plt.show()
```



In [22]:
```
tsa_plots.plot_pacf(coca1.Sales, lags=10,color='black')
```

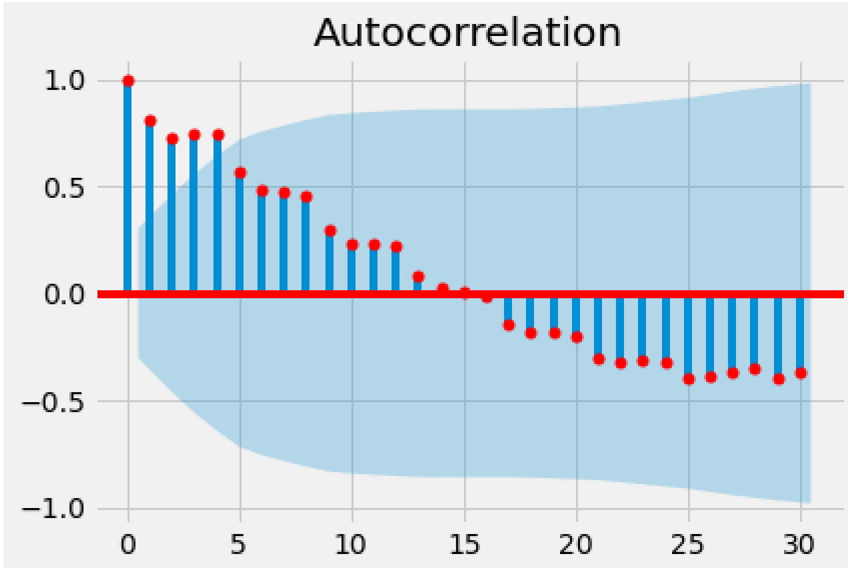
Out[22]:

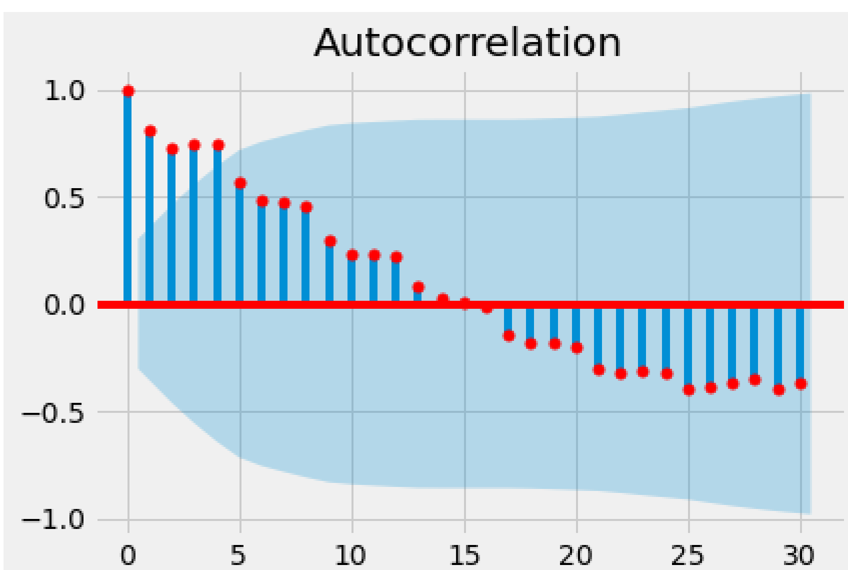




In [23]:
```
tsa_plots.plot_acf(coca1.Sales, lags=30,color='red')
```

Out[23]:





Building Time series forecasting with ARIMA

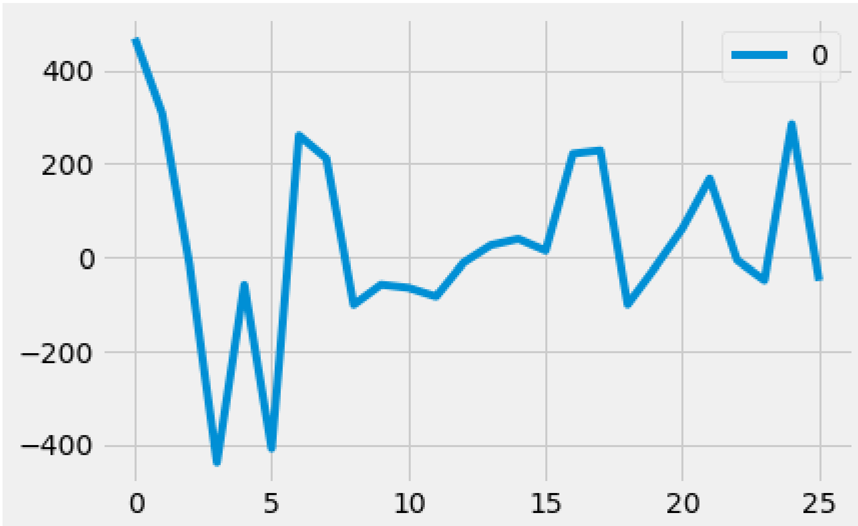
In [24]:
```
X = coca1['Sales'].values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
model = ARIMA(train, order=(5,1,0))
model_fit = model.fit(disp=0)
print(model_fit.summary())
```

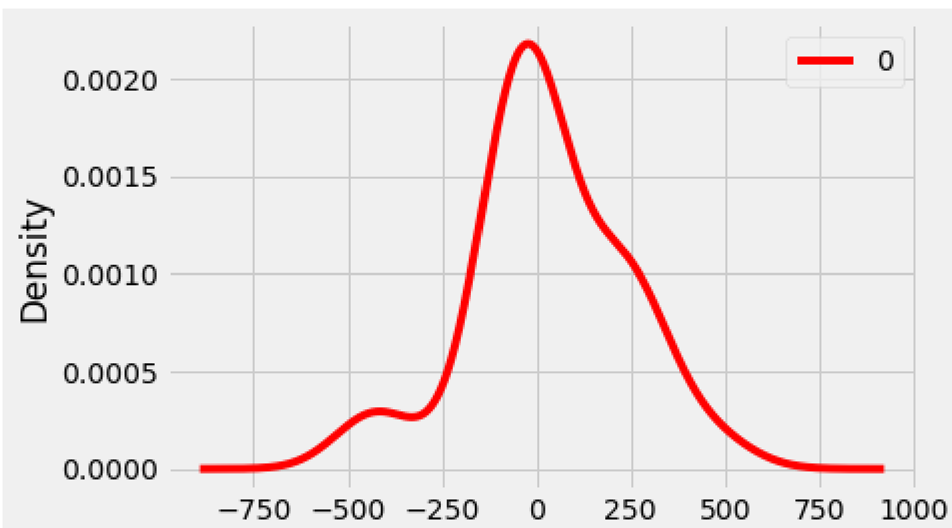
```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                    D.y   No. Observations:                   26
Model:                 ARIMA(5, 1, 0)   Log Likelihood                -172.036
Method:                       css-mle   S.D. of innovations            163.191
Date:                Sun, 30 Jan 2022   AIC                            358.071
Time:                        20:24:36   BIC                            366.878
Sample:                             1   HQIC                           360.607

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         41.8440     26.509      1.579      0.114     -10.112      93.800
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 [25]:
```
residuals = pd.DataFrame(model_fit.resid)
residuals.plot()
pyplot.show()
residuals.plot(kind='kde',color='red')
pyplot.show()
print(residuals.describe())
```






```
                0
count   26.000000
mean    31.325143
std    202.029834
min   -438.906210
25%    -58.603725
50%     -9.191026
75%    200.235650
max    468.290007
```

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

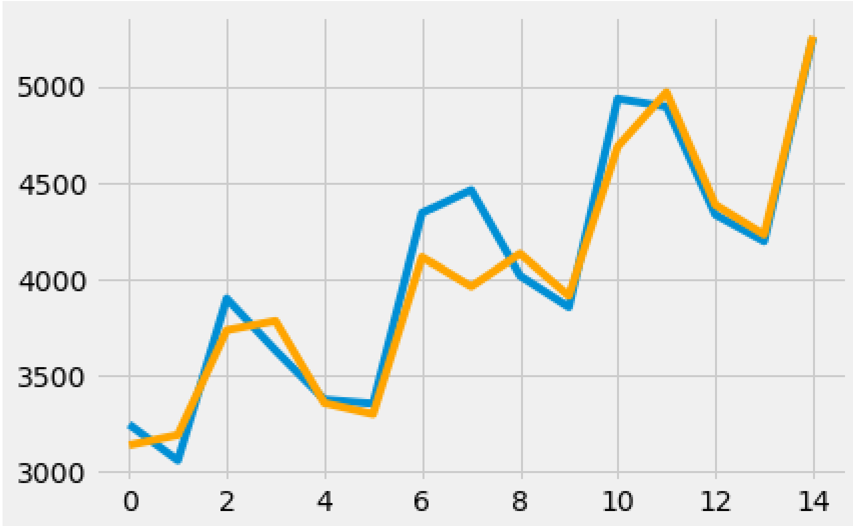
In [26]:
```
history = [x for x in train]
predictions = list()
for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit(disp=0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))
```

```
predicted=3135.586995, expected=3243.859993
predicted=3188.846687, expected=3056.000000
predicted=3734.223342, expected=3899.000000
predicted=3782.622445, expected=3629.000000
predicted=3355.124415, expected=3373.000000
predicted=3297.218016, expected=3352.000000
predicted=4112.813791, expected=4342.000000
predicted=3961.043871, expected=4461.000000
predicted=4130.787500, expected=4017.000000
predicted=3912.795280, expected=3854.000000
predicted=4687.044803, expected=4936.000000
predicted=4970.519023, expected=4895.000000
predicted=4384.040798, expected=4333.000000
predicted=4229.063937, expected=4194.000000
predicted=5261.673205, expected=5253.000000
```

In [27]:
```
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)
```

```
Test MSE: 31594.780
```

In [28]:
```
pyplot.plot(test)
pyplot.plot(predictions, color='orange')
pyplot.show()
```



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 [29]:
```
coca2 = pd.get_dummies(coca, columns = ['Quarter'])
```

In [30]:
```
coca2.columns = ['Sales','Q1','Q1','Q1','Q1','Q1','Q1','Q1','Q1','Q1','Q1','Q1','
coca2
```

Out[30]:

	Sales	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	...	Q4	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	0
1	2244.960999	0	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	0
3	2154.962997	0	0	0	0	0	0	0	0	0	...	1	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	0
5	2104.411995	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
6	2014.362999	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
7	1991.746998	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0
8	1869.049999	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
9	2313.631996	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
10	2128.320000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
11	2026.828999	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0
12	1910.603996	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0
13	2331.164993	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
14	2206.549995	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
15	2173.967995	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0
16	2148.278000	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0
17	2739.307999	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
18	2792.753998	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
19	2556.009995	0	0	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0
20	2480.973999	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0
21	3039.522995	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
22	3172.115997	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
23	2879.000999	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0
24	2772.000000	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0
25	3550.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
26	3508.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
27	3243.859993	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0
28	3056.000000	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0
29	3899.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
30	3629.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
31	3373.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1
32	3352.000000	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0

	Sales	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	...	Q4	Q4	Q4	Q4	Q4	Q4	Q4	Q4
33	4342.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
34	4461.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
35	4017.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
36	3854.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
37	4936.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
38	4895.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
39	4333.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
40	4194.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
41	5253.000000	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

42 rows × 43 columns

In [32]:
```
coca2.head().T
```

Out[32]:

	0	1	2	3	4
Sales	1734.827	2244.960999	2533.804993	2154.962997	1547.818996
Q1	1.000	0.000000	0.000000	0.000000	0.000000
Q1	0.000	0.000000	0.000000	0.000000	1.000000
Q1	0.000	0.000000	0.000000	0.000000	0.000000
Q1	0.000	0.000000	0.000000	0.000000	0.000000
Q1	0.000	0.000000	0.000000	0.000000	0.000000
Q1	0.000	0.000000	0.000000	0.000000	0.000000
Q1	0.000	0.000000	0.000000	0.000000	0.000000
Q1	0.000	0.000000	0.000000	0.000000	0.000000
Q1	0.000	0.000000	0.000000	0.000000	0.000000
Q1	0.000	0.000000	0.000000	0.000000	0.000000
Q1	0.000	0.000000	0.000000	0.000000	0.000000
Q2	0.000	1.000000	0.000000	0.000000	0.000000
Q2	0.000	0.000000	0.000000	0.000000	0.000000
Q2	0.000	0.000000	0.000000	0.000000	0.000000
Q2	0.000	0.000000	0.000000	0.000000	0.000000
Q2	0.000	0.000000	0.000000	0.000000	0.000000
Q2	0.000	0.000000	0.000000	0.000000	0.000000
Q2	0.000	0.000000	0.000000	0.000000	0.000000
Q2	0.000	0.000000	0.000000	0.000000	0.000000
Q2	0.000	0.000000	0.000000	0.000000	0.000000
Q2	0.000	0.000000	0.000000	0.000000	0.000000
Q2	0.000	0.000000	0.000000	0.000000	0.000000
Q3	0.000	0.000000	1.000000	0.000000	0.000000
Q3	0.000	0.000000	0.000000	0.000000	0.000000
Q3	0.000	0.000000	0.000000	0.000000	0.000000
Q3	0.000	0.000000	0.000000	0.000000	0.000000
Q3	0.000	0.000000	0.000000	0.000000	0.000000
Q3	0.000	0.000000	0.000000	0.000000	0.000000
Q3	0.000	0.000000	0.000000	0.000000	0.000000
Q3	0.000	0.000000	0.000000	0.000000	0.000000
Q3	0.000	0.000000	0.000000	0.000000	0.000000
Q3	0.000	0.000000	0.000000	0.000000	0.000000
Q4	0.000	0.000000	0.000000	1.000000	0.000000

	0	1	2	3	4
Q4	0.000	0.000000	0.000000	0.000000	0.000000
Q4	0.000	0.000000	0.000000	0.000000	0.000000
Q4	0.000	0.000000	0.000000	0.000000	0.000000
Q4	0.000	0.000000	0.000000	0.000000	0.000000
Q4	0.000	0.000000	0.000000	0.000000	0.000000
Q4	0.000	0.000000	0.000000	0.000000	0.000000
Q4	0.000	0.000000	0.000000	0.000000	0.000000
Q4	0.000	0.000000	0.000000	0.000000	0.000000
Q4	0.000	0.000000	0.000000	0.000000	0.000000

In [33]:
```
t= np.arange(1,43)
coca2['t'] = t
coca2['t_sq'] = coca2['t']*coca2['t']
log_Sales=np.log(coca2['Sales'])
coca2['log_Sales']=log_Sales
```

In [34]:
```
coca2.head().T
```

Out[34]:

	0	1	2	3	4
Sales	1734.827000	2244.960999	2533.804993	2154.962997	1547.818996
Q1	1.000000	0.000000	0.000000	0.000000	0.000000
Q1	0.000000	0.000000	0.000000	0.000000	1.000000
Q1	0.000000	0.000000	0.000000	0.000000	0.000000
Q1	0.000000	0.000000	0.000000	0.000000	0.000000
Q1	0.000000	0.000000	0.000000	0.000000	0.000000
Q1	0.000000	0.000000	0.000000	0.000000	0.000000
Q1	0.000000	0.000000	0.000000	0.000000	0.000000
Q1	0.000000	0.000000	0.000000	0.000000	0.000000
Q1	0.000000	0.000000	0.000000	0.000000	0.000000
Q1	0.000000	0.000000	0.000000	0.000000	0.000000
Q1	0.000000	0.000000	0.000000	0.000000	0.000000
Q2	0.000000	1.000000	0.000000	0.000000	0.000000
Q2	0.000000	0.000000	0.000000	0.000000	0.000000
Q2	0.000000	0.000000	0.000000	0.000000	0.000000
Q2	0.000000	0.000000	0.000000	0.000000	0.000000
Q2	0.000000	0.000000	0.000000	0.000000	0.000000
Q2	0.000000	0.000000	0.000000	0.000000	0.000000
Q2	0.000000	0.000000	0.000000	0.000000	0.000000
Q2	0.000000	0.000000	0.000000	0.000000	0.000000
Q2	0.000000	0.000000	0.000000	0.000000	0.000000
Q2	0.000000	0.000000	0.000000	0.000000	0.000000
Q2	0.000000	0.000000	0.000000	0.000000	0.000000
Q3	0.000000	0.000000	1.000000	0.000000	0.000000
Q3	0.000000	0.000000	0.000000	0.000000	0.000000
Q3	0.000000	0.000000	0.000000	0.000000	0.000000
Q3	0.000000	0.000000	0.000000	0.000000	0.000000
Q3	0.000000	0.000000	0.000000	0.000000	0.000000
Q3	0.000000	0.000000	0.000000	0.000000	0.000000
Q3	0.000000	0.000000	0.000000	0.000000	0.000000
Q3	0.000000	0.000000	0.000000	0.000000	0.000000
Q3	0.000000	0.000000	0.000000	0.000000	0.000000
Q3	0.000000	0.000000	0.000000	0.000000	0.000000
Q4	0.000000	0.000000	0.000000	1.000000	0.000000

	0	1	2	3	4
Q4	0.000000	0.000000	0.000000	0.000000	0.000000
Q4	0.000000	0.000000	0.000000	0.000000	0.000000
Q4	0.000000	0.000000	0.000000	0.000000	0.000000
Q4	0.000000	0.000000	0.000000	0.000000	0.000000
Q4	0.000000	0.000000	0.000000	0.000000	0.000000
Q4	0.000000	0.000000	0.000000	0.000000	0.000000
Q4	0.000000	0.000000	0.000000	0.000000	0.000000
Q4	0.000000	0.000000	0.000000	0.000000	0.000000
Q4	0.000000	0.000000	0.000000	0.000000	0.000000
t	1.000000	2.000000	3.000000	4.000000	5.000000
t_sq	1.000000	4.000000	9.000000	16.000000	25.000000
log_Sales	7.458663	7.716443	7.837477	7.675529	7.344602

In [35]:
```
train1, test1 = np.split(coca2, [int(.67 *len(coca2))])
```

In [36]:
```
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[36]: 580.1224130918627

In [37]:
```
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[37]: 783.7297975037421

In [38]:
```
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[38]: 588.1405104900215

In [39]:
```
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[39]: 1869.718820918695

In [40]:
```
addlinear= smf.ols('Sales~t+Q1+Q2+Q3+Q4',data=train1).fit()
predaddlinear=pd.Series(addlinear.predict(pd.DataFrame(test1[['t','Q1','Q2','Q3',
rmseaddlinear=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(predaddlinear))*
rmseaddlinear
```

Out[40]: 596.152628237252

In [41]:
```
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','Q
rmseaddquad=np.sqrt(np.mean((np.array(test1['Sales'])-np.array(predaddquad))**2))
rmseaddquad
```

Out[41]: 412.1144436054246

In [42]:
```
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[42]: 2374.9194407954424

In [43]:
```
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)))
rmsemulin
```

Out[43]: 5359.687911931707

In [44]:
```
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_
rmse_mul_quad
```

Out[44]: 3630.5619467350475

Conclusion

In [45]:
```
output = {'Model':pd.Series(['rmse_mul_quad','rmseadd','rmseaddlinear','rmseaddqu
          'Values':pd.Series([rmse_mul_quad,rmseadd,rmseaddlinear,rmseaddquad,rms
```

In [46]:
```
rmse=pd.DataFrame(output)
```

In [47]:
```
print(rmse)
```

```
             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 [ ]: