Import Necessary Libraries

In [1]:
```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
import statsmodels.graphics.tsaplots as tsa_plots
from statsmodels.tsa.arima_model import ARIMA
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
import numpy as np
import statsmodels.formula.api as smf
```

Data Collection

In [77]:
```
df=pd.read_excel('CocaCola_Sales_Rawdata.xlsx')
```

In [81]:
```
cola=df.copy()
cola.head()
```

Out[81]:

	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 [82]:
```
cola.shape
```

Out[82]:
```
(42, 2)
```

In [83]:
```
cola.isna().sum()
```

Out[83]:
```
Quarter    0
Sales      0
dtype: int64
```

In [84]:
```
cola.dtypes
```

Out[84]:
```
Quarter     object
Sales      float64
dtype: object
```

In [85]:
```
cola.describe().T
```

Out[85]:

	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 [86]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [87]:
```
temp = cola.Quarter.str.replace(r'(Q\d)_(\d+)', r'19\2-\1')
```

In [88]:
```
cola['quater'] = pd.to_datetime(temp).dt.strftime('%b-%Y')
```

In [89]:
```
cola.head()
```

Out[89]:

	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 [90]:
```
cola = cola.drop(['Quarter'], axis=1)
```

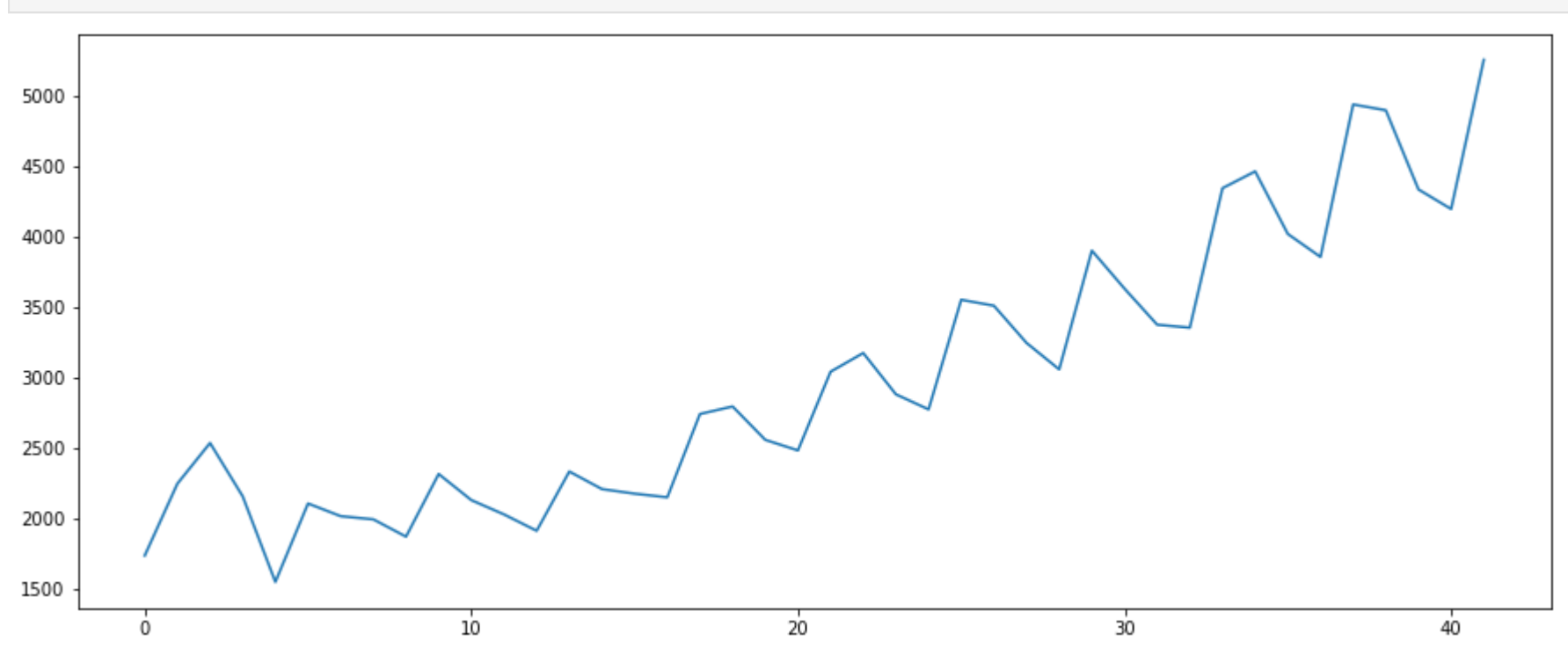
In [91]:
```
cola.reset_index(inplace=True)
```

In [92]:
```
cola.head()
```

Out[92]:

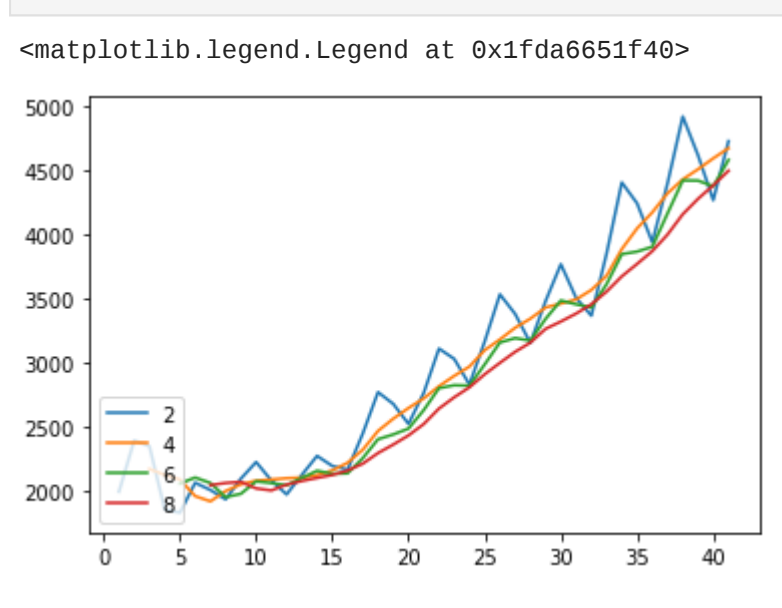
	index	Sales	quater
0	0	1734.827000	Jan-1986
1	1	2244.960999	Apr-1986
2	2	2533.804993	Jul-1986
3	3	2154.962997	Oct-1986
4	4	1547.818996	Jan-1987

In [93]:
```
cola['Sales'].plot(figsize=(15, 6))
plt.show()
```

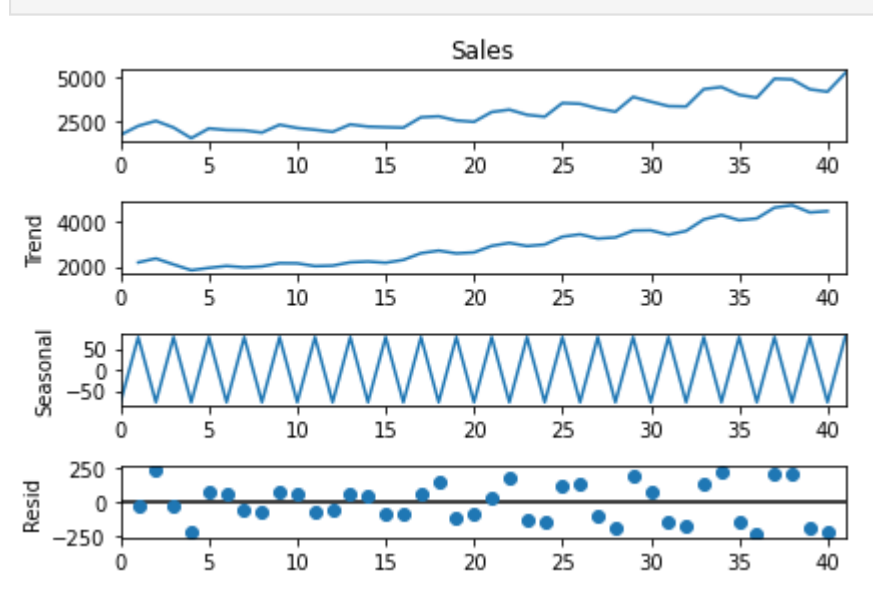


In [94]:
```
for i in range(2,10,2):
    cola["Sales"].rolling(i).mean().plot(label=str(i))
plt.legend(loc=3)
```

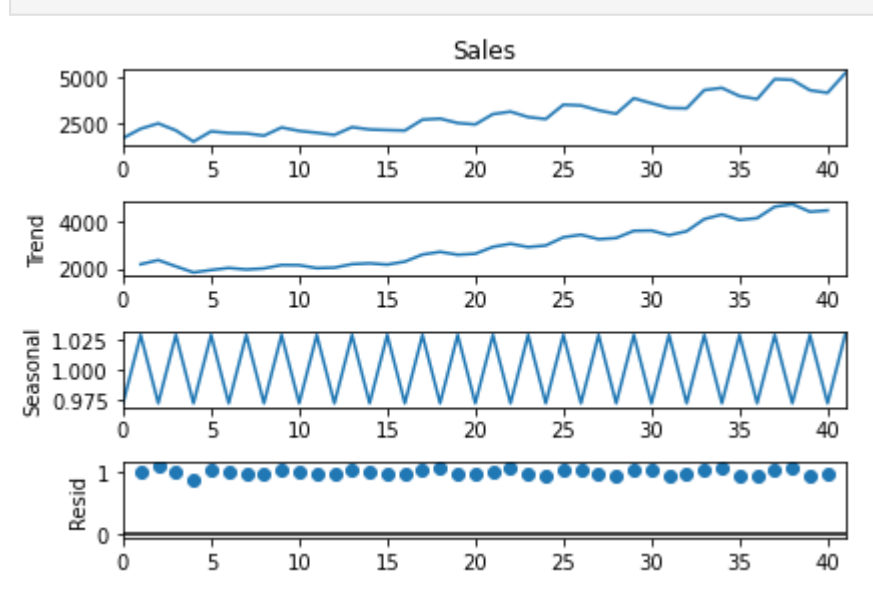
Out[94]:
```
<matplotlib.legend.Legend at 0x1fda6651f40>
```



In [95]:
```
ts_add = seasonal_decompose(cola.Sales,model="additive", period=2)
fig = ts_add.plot()
plt.show()
```

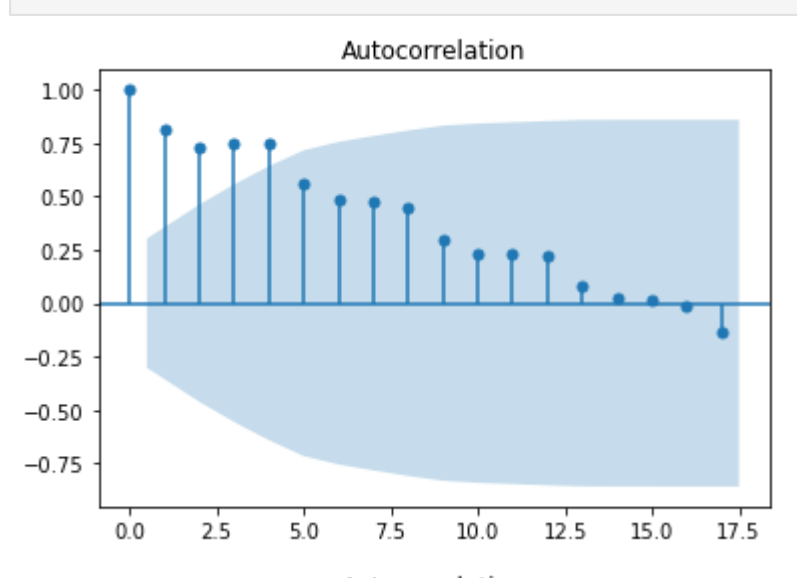


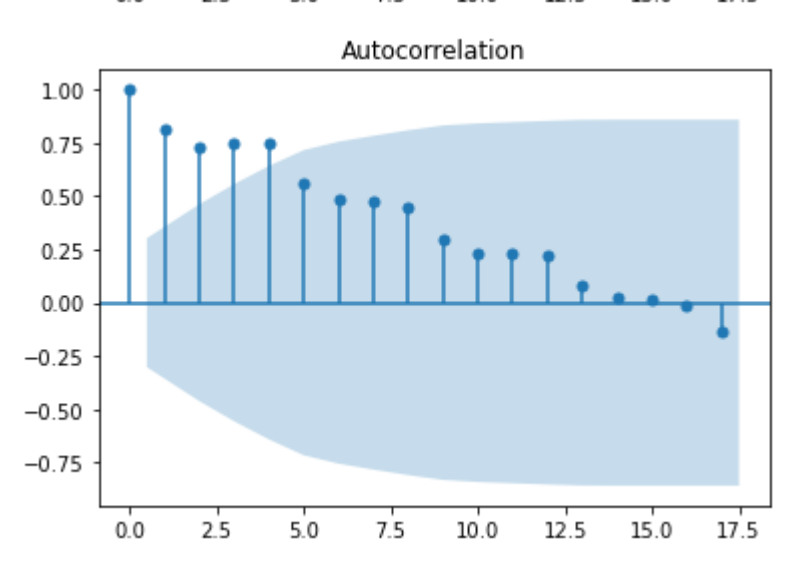
In [96]:
```
ts_mul = seasonal_decompose(cola.Sales,model="multiplicative", period=2)
fig = ts_mul.plot()
plt.show()
```



In [97]:
```
tsa_plots.plot_acf(cola.Sales)
```

Out[97]:





Building Time Series Forecasting

In [98]:
```
X = cola['Sales'].values
```

In [99]:
```
size = int(len(X) * 0.66)
```

In [100…
```
train, test = X[0:size], X[size:len(X)]
```

In [101…
```
model = ARIMA(train, order=(5,1,0))
```

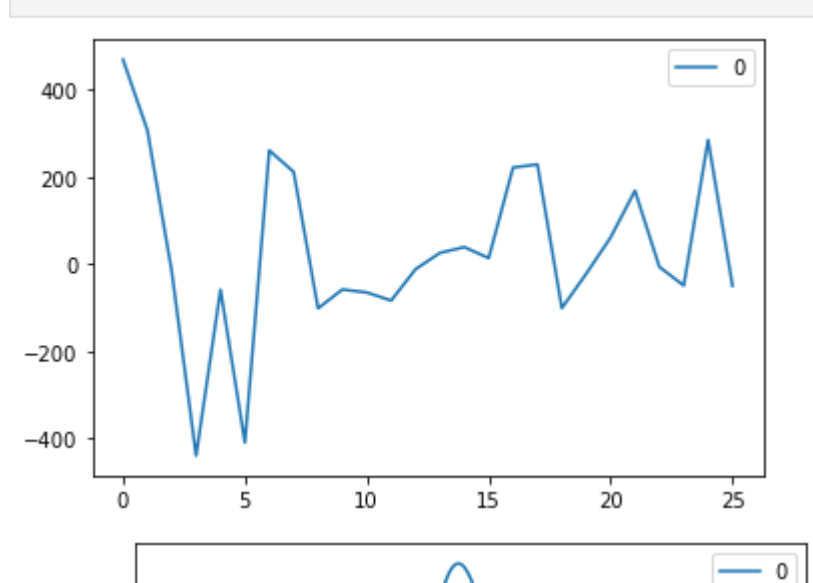
In [102…
```
model_fit = model.fit(disp=0)
```

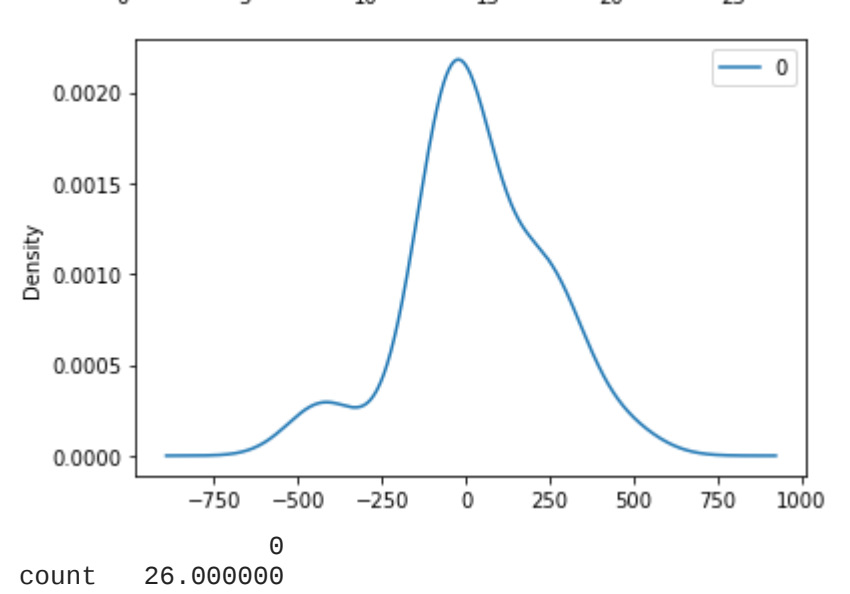
In [103…
```
print(model_fit.summary())
```

```
                             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, 25 Dec 2021   AIC                            358.071
Time:                        15:10:06   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
-----------------------------------------------------------------------------
```

In [104…
```
residuals = pd.DataFrame(model_fit.resid)
residuals.plot()
pyplot.show()
residuals.plot(kind='kde')
pyplot.show()
print(residuals.describe())
```






```
                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
```

Forecast ARIMA Model

In [105…
```
history = [x for x in train]
```

In [106…
```
predictions = list()
```

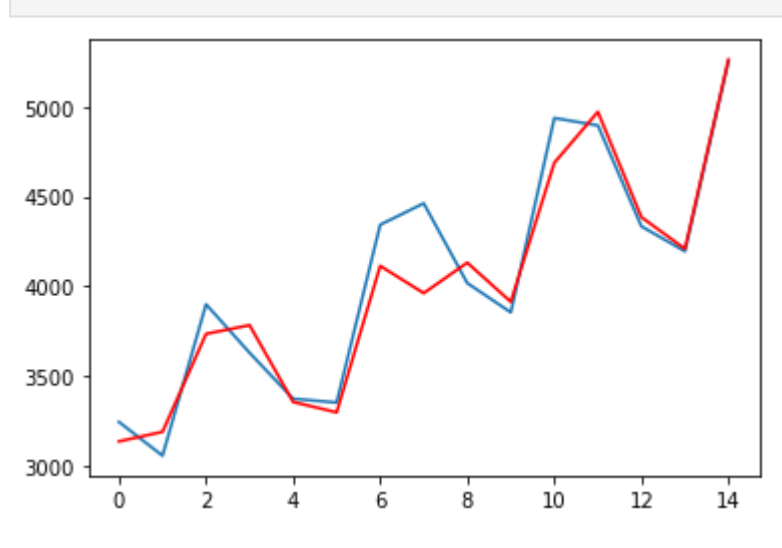
In [107…
```
for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit(disp=0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))
```

```
predicted=3135.586029, 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 [108…
```
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)
```

```
Test MSE: 31525.273
```

In [109…
```
pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
```



Comparing Multiple Models

In [116…
```
cola2 = pd.get_dummies(df, columns = ['Quarter'])
```

In [117…
```
cola2.columns = ['Sales','Q1','Q1','Q1','Q1','Q1','Q1','Q1','Q1','Q1','Q1','Q2','Q2','Q2','Q2','Q2','Q2','Q2','Q2','Q2','Q2','Q3','Q3','Q3','Q3
```

In [118…
```
cola2.head()
```

Out[118…

	Sales	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	...	Q4	Q4	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	0	0
1	2244.960999	0	0	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	0	0
3	2154.962997	0	0	0	0	0	0	0	0	0	...	1	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	0	0	0

5 rows × 43 columns

In [119…
```
t= np.arange(1,43)
```

In [120…
```
cola2['t'] = t
```

In [121…
```
cola2['t_sq'] = cola2['t']*cola2['t']
```

In [124…
```
log_Sales=np.log(cola2['Sales'])
```

In [125…
```
cola2['log_Sales']=log_Sales
```

In [126…
```
cola2.head()
```

Out[126…

	Sales	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	Q1	...	Q4	Q4	Q4	Q4	Q4	Q4	Q4	t	t_sq	log_Sales
0	1734.827000	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	1	7.458663
1	2244.960999	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	2	4	7.716443
2	2533.804993	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	3	9	7.837477
3	2154.962997	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	4	16	7.675529
4	1547.818996	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	5	25	7.344602

5 rows × 46 columns

In [127…
```
train1, test1 = np.split(cola2, [int(.67 *len(cola2))])
```

In [128…
```
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[128…
```
580.1224130918641
```

In [129…
```
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[129…
```
783.7297975037103
```

In [130…
```
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[130…
```
588.1405104900134
```

In [131…
```
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[131…
```
1869.7188209186947
```

In [132…
```
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[132…
```
596.1526282372472
```

In [133…
```
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[133…
```
412.1144436053775
```

In [134…
```
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[134…
```
2374.9194407954374
```

In [135…
```
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[135…
```
5359.687911932085
```

In [136…
```
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[136…
```
3630.5619467347524
```

In [137…
```
output = {'Model':pd.Series(['rmse_mul_quad','rmseadd','rmseaddlinear','rmseaddquad','rmseexpo','rmselin','rmsemul','rmsemulin','rmsequad']),
          'Values':pd.Series([rmse_mul_quad,rmseadd,rmseaddlinear,rmseaddquad,rmseexpo,rmselin,rmsemul,rmsemulin,rmsequad])}
```

In [138…
```
rmse=pd.DataFrame(output)
```

In [139…
```
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
```