Problem Statement: -

The dataset consists of quarterly sales data of Coca-Cola from 1986 to 1996. Predict sales for the next two years by using time series forecasting and prepare a document for each model explaining how many dummy variables you have created and also include the RMSE value for each model.

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
```

In [ ]:

```

```

In [2]:

```
co = pd.read_excel("D:/DataScience/Class/assignment working/Forcasting/CocaCola_Sales_RawData.xlsx")
```

In [3]:

```
co.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]:

```
co.tail()
```

Out[4]:

	Quarter	Sales
37	Q2_95	4936.0
38	Q3_95	4895.0
39	Q4_95	4333.0
40	Q1_96	4194.0
41	Q2_96	5253.0

In [5]:

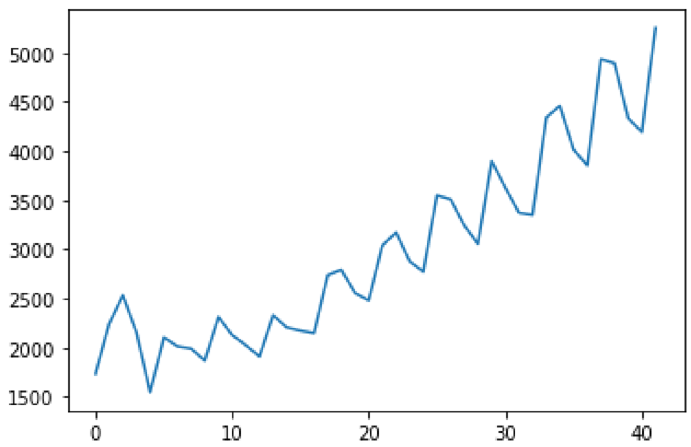
```
#checking missing values
co.isna().sum()
```

Out[5]:

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

In [6]:

```
co.Sales.plot()
plt.show()
```



In [7]:

```
#checking stationarity of data
```

In [8]:

```
from statsmodels.tsa.stattools import adfuller
```

In [9]:

```
test = adfuller(co.Sales)
```

```
test
```

Out[9]:

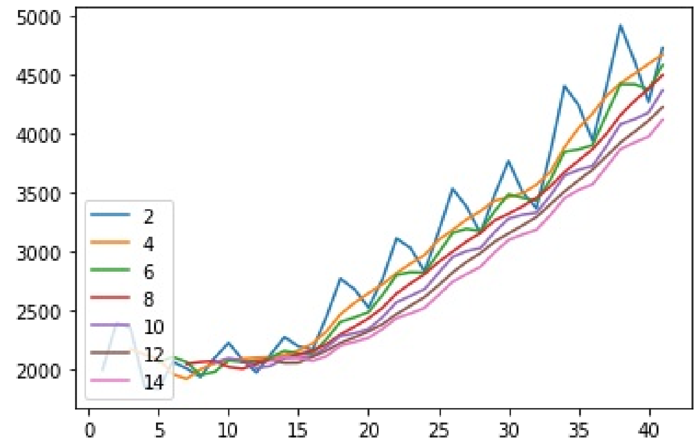
```
(1.3094210153268144,
 0.9966611673930905,
 7,
 34,
 {'1%': -3.639224104416853,
  '5%': -2.9512301791166293,
  '10%': -2.614446989619377},
 395.6639212829265)
```

HO - Data is not stationary
Ha - Data is stationary
p value is very high so we can not reject null hypothesis , data is not stationary

removing seasonality and trend

In [10]:

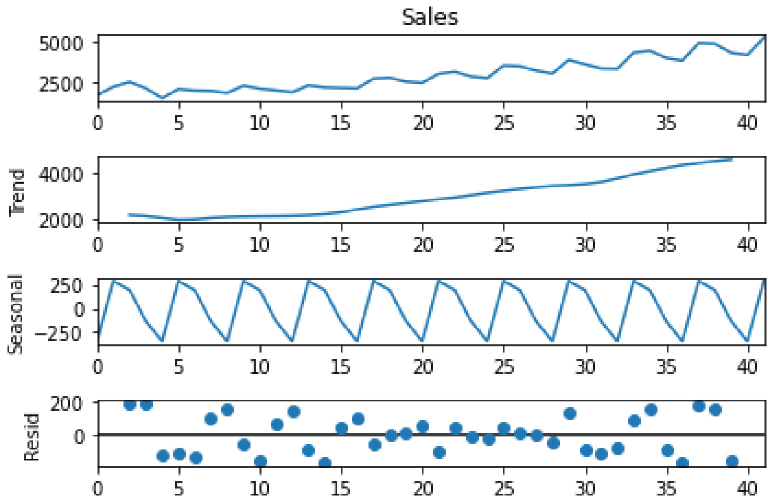
```
# Centering moving average for the time series
for i in range (2,15,2):
    co.Sales.rolling(i).mean().plot(label=str(i))
plt.legend(loc=3)
plt.show()
```

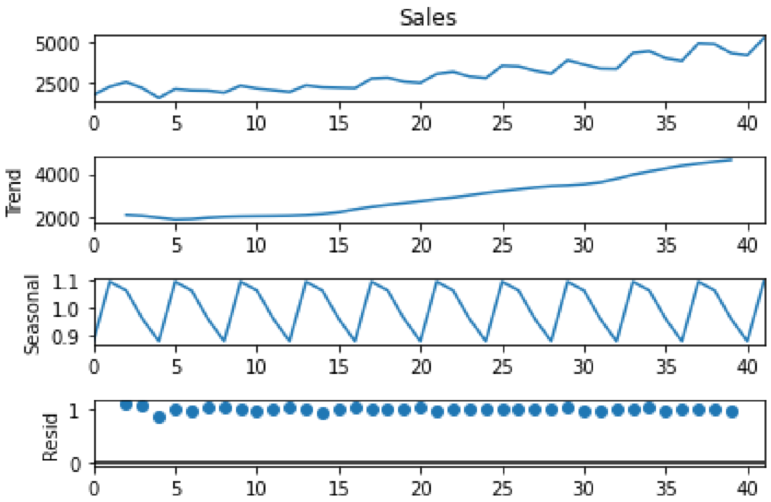


at lag =4 we are getting good smooth curve

In [11]:

```
# Time series decomposition plot
from statsmodels.tsa.seasonal import seasonal_decompose
decompose_ts_add = seasonal_decompose(co.Sales, model = "additive", period = 4)
decompose_ts_add.plot()
decompose_ts_mul = seasonal_decompose(co.Sales, model = "multiplicative", period = 4)
decompose_ts_mul.plot()
plt.show()
```





In [12]:

```
co["diff_4"]=co.Sales-co.Sales.shift(4)
co["diff_8"]=co.Sales-co.Sales.shift(8)
```

In [13]:

```
#again checking stationarity with dickey fuller test
```

```
test_4 = adfuller(co.diff_4.dropna())
test_4
```

Out[13]:

```
(-2.6101117490935724,
 0.09092533196680103,
 0,
 37,
 {'1%': -3.6209175221605827,
  '5%': -2.9435394610388332,
  '10%': -2.6104002410518627},
 341.7453372896339)
```

In [14]:

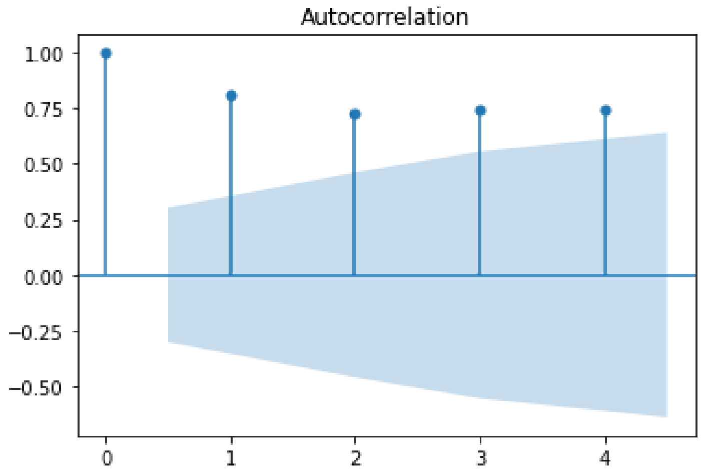
```
test_8 = adfuller(co.diff_8.dropna())
test_8
```

Out[14]:

```
(-1.8599907449358082,
 0.35111032474604853,
 0,
 33,
 {'1%': -3.6461350877925254,
  '5%': -2.954126991123355,
  '10%': -2.6159676124885216},
 303.9377386823995)
```

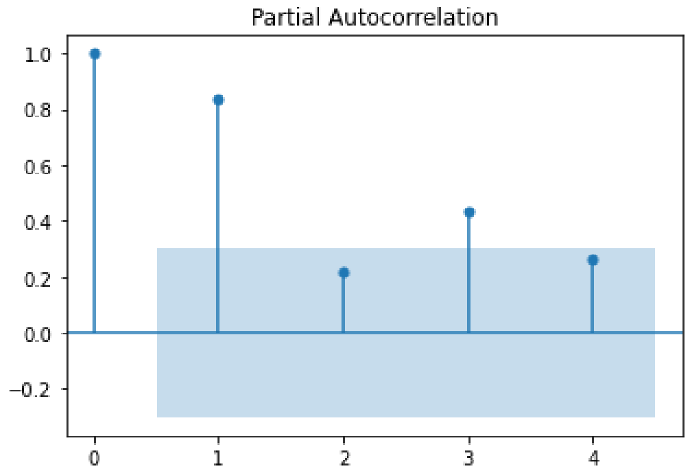
In [17]:

```
# ACF plot on Original data sets
import statsmodels.graphics.tsaplots as tsa_plots
tsa_plots.plot_acf(co.Sales, lags = 4)
plt.show()
```



In [18]:

```
tsa_plots.plot_pacf(co.Sales, lags=4)
plt.show()
```



In [19]:

```
#splitting Data
```

In [20]:

```
len(co.Sales)
```

Out[20]:

```
42
```

In [21]:

```
Train = co.head(34)
Test=co.tail(8)
```

In [22]:

```
# Creating a function to calculate the MAPE value for test data
def MAPE(pred,org):
    temp = np.abs((pred-org)/org)*100
    return np.mean(temp)
```

In [66]:

```
# Simple Exponential Method
```

```python
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
ses_model = SimpleExpSmoothing(Train.Sales).fit(smoothing_level=1)
pred_ses = ses_model.predict(start = Test.index[0], end = Test.index[-1])
MAPE(pred_ses, Test.Sales)
```

Out[66]:

```
8.478831890219737
```

In [75]:

```python
# Holt method
from statsmodels.tsa.holtwinters import Holt
hw_model = Holt(Train.Sales).fit(smoothing_level=0.3)
pred_hw = hw_model.predict(start = Test.index[0], end = Test.index[-1])
MAPE(pred_hw, Test.Sales)
```

Out[75]:

```
8.352488545988642
```

In [84]:

```python
# Holts winter exponential smoothing with additive seasonality and additive trend
from statsmodels.tsa.holtwinters import ExponentialSmoothing
hwe_model_add_add = ExponentialSmoothing(Train.Sales, seasonal = "add", trend = "add", seasonal_periods = 4).fit(smoothing_level=
0.4)
pred_hwe_add_add = hwe_model_add_add.predict(start = Test.index[0], end = Test.index[-1])
MAPE(pred_hwe_add_add,Test.Sales)
```

Out[84]:

```
6.4391281601664305
```

In [85]:

```python
# Holts winter exponential smoothing with multiplicative seasonality and additive trend
hwe_model_mul_add = ExponentialSmoothing(Train.Sales, seasonal = "mul", trend = "add", seasonal_periods = 4).fit()
pred_hwe_mul_add = hwe_model_mul_add.predict(start = Test.index[0], end = Test.index[-1])
MAPE(pred_hwe_mul_add, Test.Sales)
```

Out[85]:

```
6.85805895700624
```

Holts winter exponential smoothing with additive seasonality and additive trend is giving best accuracy so choosing it as final model

Final model

In [90]:

```python
#preparing new data to store predictions
pred = pd.read_excel("D:/DataScience/Class/assignment working/Forcasting/pred_CocaCola_Sales_RawData.xlsx")
```

In [95]:

```python
pred.tail(10)
```

Out[95]:

	Quarter	Sales
40	Q1_96	4194.0
41	Q2_96	5253.0
42	Q3_96	NaN
43	Q4_96	NaN
44	Q1_97	NaN
45	Q2_97	NaN
46	Q3_97	NaN
47	Q4_97	NaN
48	Q1_98	NaN
49	Q2_98	NaN

In [91]:

```python
final_model = ExponentialSmoothing(co.Sales, seasonal = "add", trend = "add", seasonal_periods = 4).fit(smoothing_level=0.4)
```

In [108]:

```python
final_pred = final_model.predict(start = pred.index[0], end = pred.index[-1])
```

In [109]:

```python
final_pred
```

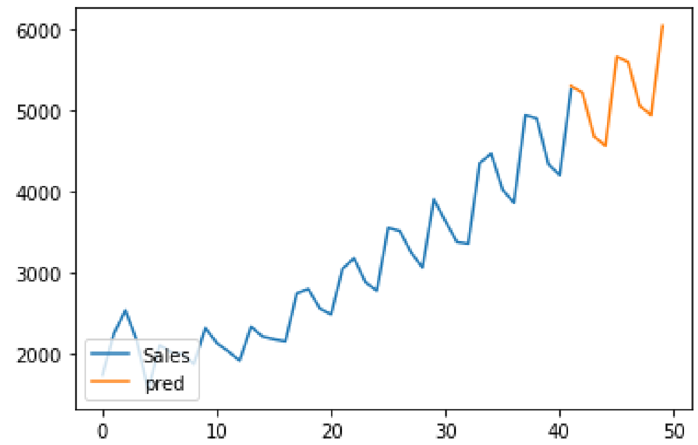
Out[109]:

```
0      1734.651836
1      2205.405183
2      2488.127623
3      2352.664713
4      1716.738058
5      2125.482785
6      2365.750135
7      1878.157846
8      1356.653128
```

```
9     2142.850444
10    2290.158259
11    2197.985368
12    1824.267033
13    2230.217680
14    2140.699362
15    2139.255025
16    2020.792987
17    2473.013714
18    2460.197460
19    2595.434358
20    2546.919794
21    3036.619890
22    2972.922487
23    2837.812866
24    2816.974158
25    3369.057442
26    3518.423781
27    3215.329294
28    3147.585864
29    3823.448544
30    3815.971210
31    3454.469925
32    3252.459462
33    4099.083896
34    4005.773319
35    4010.265478
36    4003.598700
37    4872.111893
38    4858.643820
39    4423.766247
40    4275.336229
41    5294.649992
42    5211.100140
43    4671.733686
44    4557.401205
45    5655.387002
46    5588.497147
47    5049.130692
48    4934.798211
49    6032.784008
dtype: float64
```

In [157]:

```
co.Sales.plot()
final_pred.iloc[41:].plot(label="pred")
plt.legend(loc=3)
plt.show()
```



In [107]:

```
final_pred
```

Out[107]:

```
42    5211.100140
43    4671.733686
44    4557.401205
45    5655.387002
46    5588.497147
47    5049.130692
48    4934.798211
49    6032.784008
dtype: float64
```

In [113]:

```
MAPE(final_pred.iloc[:43], co.Sales)
```

Out[113]:

```
4.803231889345973
```

In [115]:

```
rmse = np.sqrt(np.mean((final_pred.iloc[:43]- co.Sales)**2))
rmse
```

Out[115]:

```
171.51053481517508
```

Summary and inference

final model looks more reliable and has good predictions

In [ ]:

```

```