Problem Statement:

Forecast the CocaCola prices. 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.

1. Import Neccesary Libraries

In [1]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
import seaborn as sns
from matplotlib import pyplot as plt
```

2. Import Data

In [5]:
```
Coco_Cola_Data = pd.read_csv('CocaCola_Sales_Rawdata.csv')
Coco_Cola_Data
```

Out[5]:

	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

3. Data Understanding

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

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

In [10]:
```
Coco_Cola_Data.isna().sum()
```

Out[10]:
```
Quarter    0
Sales      0
dtype: int64
```

In [11]:
```
Coco_Cola_Data.describe()
```

Out[11]:

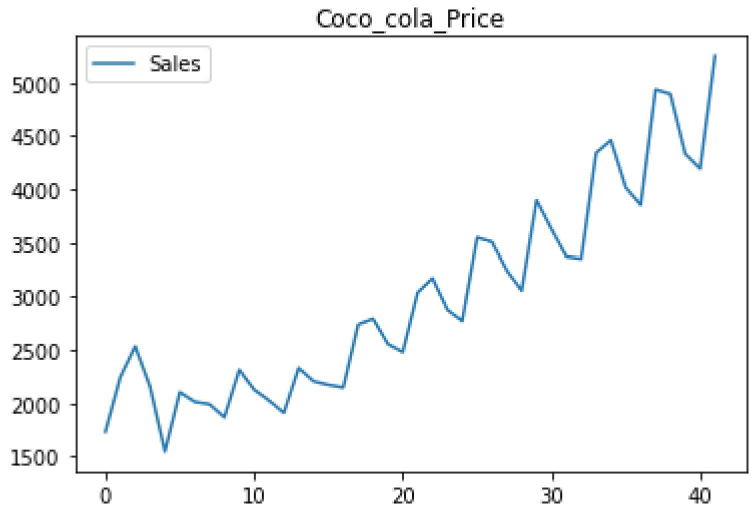
	Sales
count	42.000000
mean	2994.353308
std	977.930896
min	1547.818996
25%	2159.714247
50%	2782.376999
75%	3609.250000
max	5253.000000

In [12]:
```
Coco_Cola_Data.dtypes
```

Out[12]:
```
Quarter     object
Sales      float64
dtype: object
```

4. Data Understanding

In [13]:
```
Coco_Cola_Data.plot()
plt.title('Coco_cola_Price')
plt.show()
```



Creating dummy variables

In [15]:
```
quarter=['Q1','Q2','Q3','Q4']
n=Coco_Cola_Data['Quarter'][0]
n[0:2]
```

Out[15]:
```
'Q1'
```

In [16]:
```
Coco_Cola_Data['quarter']=0
Coco_Cola_Data['quarter']
```

Out[16]:
```
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
11    0
12    0
13    0
14    0
15    0
16    0
17    0
18    0
19    0
20    0
21    0
22    0
23    0
24    0
25    0
26    0
27    0
28    0
29    0
30    0
31    0
32    0
33    0
34    0
35    0
36    0
37    0
38    0
39    0
40    0
41    0
Name: quarter, dtype: int64
```

In [17]:
```
for i in range(42):
    n=Coco_Cola_Data['Quarter'][i]
    Coco_Cola_Data['quarter'][i]=n[0:2]
Coco_Cola_Data['quarter']
```

```
<ipython-input-17-c6255bda0493>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  Coco_Cola_Data['quarter'][i]=n[0:2]
C:\Users\nandini\anaconda3\lib\site-packages\pandas\core\indexing.py:1637: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self._setitem_single_block(indexer, value, name)
```

Out[17]:
```
0     Q1
1     Q2
2     Q3
3     Q4
4     Q1
5     Q2
6     Q3
7     Q4
8     Q1
9     Q2
10    Q3
11    Q4
12    Q1
13    Q2
14    Q3
15    Q4
16    Q1
17    Q2
18    Q3
19    Q4
20    Q1
21    Q2
22    Q3
23    Q4
24    Q1
25    Q2
26    Q3
27    Q4
28    Q1
29    Q2
30    Q3
31    Q4
32    Q1
```

```
33    Q2
34    Q3
35    Q4
36    Q1
37    Q2
38    Q3
39    Q4
40    Q1
41    Q2
Name: quarter, dtype: object
```

In [18]:
```
dummy=pd.DataFrame(pd.get_dummies(Coco_Cola_Data['quarter']))
dummy
```

Out[18]:

	Q1	Q2	Q3	Q4
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0
5	0	1	0	0
6	0	0	1	0
7	0	0	0	1
8	1	0	0	0
9	0	1	0	0
10	0	0	1	0
11	0	0	0	1
12	1	0	0	0
13	0	1	0	0
14	0	0	1	0
15	0	0	0	1
16	1	0	0	0
17	0	1	0	0
18	0	0	1	0
19	0	0	0	1
20	1	0	0	0
21	0	1	0	0
22	0	0	1	0
23	0	0	0	1
24	1	0	0	0
25	0	1	0	0
26	0	0	1	0
27	0	0	0	1
28	1	0	0	0
29	0	1	0	0
30	0	0	1	0
31	0	0	0	1
32	1	0	0	0

	Q1	Q2	Q3	Q4
33	0	1	0	0
34	0	0	1	0
35	0	0	0	1
36	1	0	0	0
37	0	1	0	0
38	0	0	1	0
39	0	0	0	1
40	1	0	0	0
41	0	1	0	0

In [20]:
```
Coco_Cola_Data_New=pd.concat((Coco_Cola_Data,dummy),axis=1)
t= np.arange(1,43)
Coco_Cola_Data_New['t']=t
Coco_Cola_Data_New['t_square']=Coco_Cola_Data_New['t']*Coco_Cola_Data_New['t']
Coco_Cola_Data_New
```

Out[20]:

	Quarter	Sales	quarter	Q1	Q2	Q3	Q4	t	t_square
0	Q1_86	1734.827000	Q1	1	0	0	0	1	1
1	Q2_86	2244.960999	Q2	0	1	0	0	2	4
2	Q3_86	2533.804993	Q3	0	0	1	0	3	9
3	Q4_86	2154.962997	Q4	0	0	0	1	4	16
4	Q1_87	1547.818996	Q1	1	0	0	0	5	25
5	Q2_87	2104.411995	Q2	0	1	0	0	6	36
6	Q3_87	2014.362999	Q3	0	0	1	0	7	49
7	Q4_87	1991.746998	Q4	0	0	0	1	8	64
8	Q1_88	1869.049999	Q1	1	0	0	0	9	81
9	Q2_88	2313.631996	Q2	0	1	0	0	10	100
10	Q3_88	2128.320000	Q3	0	0	1	0	11	121
11	Q4_88	2026.828999	Q4	0	0	0	1	12	144
12	Q1_89	1910.603996	Q1	1	0	0	0	13	169
13	Q2_89	2331.164993	Q2	0	1	0	0	14	196
14	Q3_89	2206.549995	Q3	0	0	1	0	15	225
15	Q4_89	2173.967995	Q4	0	0	0	1	16	256
16	Q1_90	2148.278000	Q1	1	0	0	0	17	289
17	Q2_90	2739.307999	Q2	0	1	0	0	18	324
18	Q3_90	2792.753998	Q3	0	0	1	0	19	361
19	Q4_90	2556.009995	Q4	0	0	0	1	20	400
20	Q1_91	2480.973999	Q1	1	0	0	0	21	441
21	Q2_91	3039.522995	Q2	0	1	0	0	22	484
22	Q3_91	3172.115997	Q3	0	0	1	0	23	529
23	Q4_91	2879.000999	Q4	0	0	0	1	24	576
24	Q1_92	2772.000000	Q1	1	0	0	0	25	625
25	Q2_92	3550.000000	Q2	0	1	0	0	26	676
26	Q3_92	3508.000000	Q3	0	0	1	0	27	729
27	Q4_92	3243.859993	Q4	0	0	0	1	28	784
28	Q1_93	3056.000000	Q1	1	0	0	0	29	841
29	Q2_93	3899.000000	Q2	0	1	0	0	30	900
30	Q3_93	3629.000000	Q3	0	0	1	0	31	961

	Quarter	Sales	quarter	Q1	Q2	Q3	Q4	t	t_square
31	Q4_93	3373.000000	Q4	0	0	0	1	32	1024
32	Q1_94	3352.000000	Q1	1	0	0	0	33	1089
33	Q2_94	4342.000000	Q2	0	1	0	0	34	1156
34	Q3_94	4461.000000	Q3	0	0	1	0	35	1225
35	Q4_94	4017.000000	Q4	0	0	0	1	36	1296
36	Q1_95	3854.000000	Q1	1	0	0	0	37	1369
37	Q2_95	4936.000000	Q2	0	1	0	0	38	1444
38	Q3_95	4895.000000	Q3	0	0	1	0	39	1521
39	Q4_95	4333.000000	Q4	0	0	0	1	40	1600
40	Q1_96	4194.000000	Q1	1	0	0	0	41	1681
41	Q2_96	5253.000000	Q2	0	1	0	0	42	1764

In [21]:
```
log_Sales=np.log(Coco_Cola_Data_New['Sales'])
Coco_Cola_Data_New['log_Sales']=log_Sales
Coco_Cola_Data_New
```

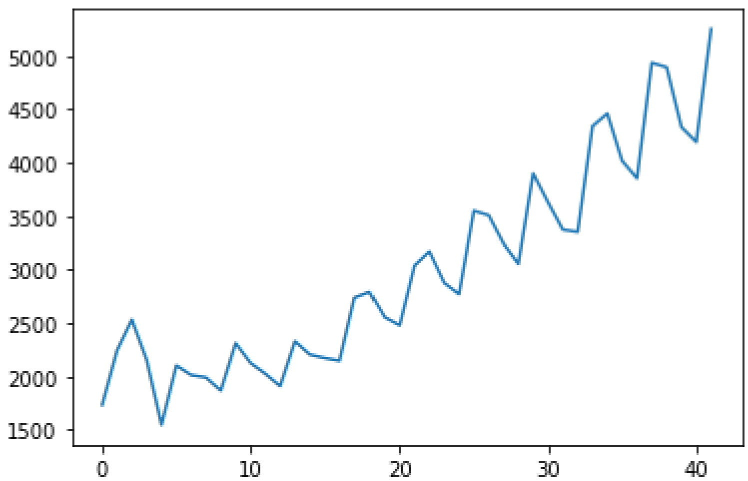
Out[21]:

	Quarter	Sales	quarter	Q1	Q2	Q3	Q4	t	t_square	log_Sales
0	Q1_86	1734.827000	Q1	1	0	0	0	1	1	7.458663
1	Q2_86	2244.960999	Q2	0	1	0	0	2	4	7.716443
2	Q3_86	2533.804993	Q3	0	0	1	0	3	9	7.837477
3	Q4_86	2154.962997	Q4	0	0	0	1	4	16	7.675529
4	Q1_87	1547.818996	Q1	1	0	0	0	5	25	7.344602
5	Q2_87	2104.411995	Q2	0	1	0	0	6	36	7.651791
6	Q3_87	2014.362999	Q3	0	0	1	0	7	49	7.608058
7	Q4_87	1991.746998	Q4	0	0	0	1	8	64	7.596767
8	Q1_88	1869.049999	Q1	1	0	0	0	9	81	7.533186
9	Q2_88	2313.631996	Q2	0	1	0	0	10	100	7.746574
10	Q3_88	2128.320000	Q3	0	0	1	0	11	121	7.663088
11	Q4_88	2026.828999	Q4	0	0	0	1	12	144	7.614228
12	Q1_89	1910.603996	Q1	1	0	0	0	13	169	7.555175
13	Q2_89	2331.164993	Q2	0	1	0	0	14	196	7.754123
14	Q3_89	2206.549995	Q3	0	0	1	0	15	225	7.699185
15	Q4_89	2173.967995	Q4	0	0	0	1	16	256	7.684309
16	Q1_90	2148.278000	Q1	1	0	0	0	17	289	7.672422
17	Q2_90	2739.307999	Q2	0	1	0	0	18	324	7.915461
18	Q3_90	2792.753998	Q3	0	0	1	0	19	361	7.934783
19	Q4_90	2556.009995	Q4	0	0	0	1	20	400	7.846203
20	Q1_91	2480.973999	Q1	1	0	0	0	21	441	7.816407
21	Q2_91	3039.522995	Q2	0	1	0	0	22	484	8.019456
22	Q3_91	3172.115997	Q3	0	0	1	0	23	529	8.062154
23	Q4_91	2879.000999	Q4	0	0	0	1	24	576	7.965199
24	Q1_92	2772.000000	Q1	1	0	0	0	25	625	7.927324
25	Q2_92	3550.000000	Q2	0	1	0	0	26	676	8.174703
26	Q3_92	3508.000000	Q3	0	0	1	0	27	729	8.162801
27	Q4_92	3243.859993	Q4	0	0	0	1	28	784	8.084519
28	Q1_93	3056.000000	Q1	1	0	0	0	29	841	8.024862
29	Q2_93	3899.000000	Q2	0	1	0	0	30	900	8.268475
30	Q3_93	3629.000000	Q3	0	0	1	0	31	961	8.196712
31	Q4_93	3373.000000	Q4	0	0	0	1	32	1024	8.123558
32	Q1_94	3352.000000	Q1	1	0	0	0	33	1089	8.117312

	Quarter	Sales	quarter	Q1	Q2	Q3	Q4	t	t_square	log_Sales
33	Q2_94	4342.000000	Q2	0	1	0	0	34	1156	8.376090
34	Q3_94	4461.000000	Q3	0	0	1	0	35	1225	8.403128
35	Q4_94	4017.000000	Q4	0	0	0	1	36	1296	8.298291
36	Q1_95	3854.000000	Q1	1	0	0	0	37	1369	8.256867
37	Q2_95	4936.000000	Q2	0	1	0	0	38	1444	8.504311
38	Q3_95	4895.000000	Q3	0	0	1	0	39	1521	8.495970
39	Q4_95	4333.000000	Q4	0	0	0	1	40	1600	8.374015
40	Q1_96	4194.000000	Q1	1	0	0	0	41	1681	8.341410
41	Q2_96	5253.000000	Q2	0	1	0	0	42	1764	8.566555

Train test split

In [23]:
```
train= Coco_Cola_Data_New.head(38)
test=Coco_Cola_Data_New.tail(4)
Coco_Cola_Data_New.Sales.plot()
plt.show()
```



5. Model building, Training & Testing

In [24]:
```
import statsmodels.formula.api as smf
```

1. Model Based Forecasting Techniques

1. Linear Model

In [25]:
```
linear= smf.ols('Sales~t',data=train).fit()
predlin=pd.Series(linear.predict(pd.DataFrame(test['t'])))
rmselin=np.sqrt((np.mean(np.array(test['Sales'])-np.array(predlin))**2))
rmselin
```

Out[25]: 421.1787876367787

2. Exponential Model

In [26]:
```
expo=smf.ols('log_Sales~t',data=train).fit()
predexp=pd.Series(expo.predict(pd.DataFrame(test['t'])))
rmseexpo=np.sqrt(np.mean((np.array(test['Sales'])-np.array(np.exp(predexp)))**2))
rmseexpo
```

Out[26]: 466.2479731321065

3. Quadratic Model

In [27]:
```
quad=smf.ols('Sales~t+t_square',data=train).fit()
predquad=pd.Series(quad.predict(pd.DataFrame(test[['t','t_square']])))
rmsequad=np.sqrt(np.mean((np.array(test['Sales'])-np.array(predquad))**2))
rmsequad
```

Out[27]: 475.56183519820195

4. Additive Seasonality

In [28]:
```
additive= smf.ols('Sales~ Q1+Q2+Q3+Q4',data=train).fit()
predadd=pd.Series(additive.predict(pd.DataFrame(test[['Q1','Q2','Q3','Q4']])))
rmseadd=np.sqrt(np.mean((np.array(test['Sales'])-np.array(predadd))**2))
rmseadd
```

Out[28]: 1860.0238154374442

5. Additive Seasonality With Quadratic Trend

In [29]:
```
addquad=smf.ols('Sales~t+t_square+Q1+Q2+Q3+Q4',data=train).fit()
predaddquad=pd.Series(addquad.predict(pd.DataFrame(test[['t','t_square','Q1','Q2'
rmseaddquad=np.sqrt(np.mean((np.array(test['Sales'])-np.array(predaddquad))**2))
rmseaddquad
```

Out[29]: 301.73800721461606

6. Multiplicative Seasonality

In [30]:
```
mulsea=smf.ols('log_Sales~Q1+Q2+Q3+Q4',data=train).fit()
predmul= pd.Series(mulsea.predict(pd.DataFrame(test[['Q1','Q2','Q3','Q4']])))
rmsemul= np.sqrt(np.mean((np.array(test['Sales'])-np.array(np.exp(predmul)))**2))
rmsemul
```

Out[30]:
```
1963.38964005634
```

7. Multiplicative Seasonality With Linear Trend

In [31]:
```
mullin= smf.ols('log_Sales~t+Q1+Q2+Q3+Q4',data=train).fit()
predmullin= pd.Series(mullin.predict(pd.DataFrame(test[['t','Q1','Q2','Q3','Q4']]
rmsemulin=np.sqrt(np.mean((np.array(test['Sales'])-np.array(np.exp(predmullin)))*
rmsemulin
```

Out[31]:
```
225.52439056169976
```

Tabulating RMSE values

In [36]:
```
data={'Model':pd.Series(['rmseadd','rmseaddquad','rmseexpo','rmselin','rmsemul','
data
```

Out[36]:
```
{'Model': 0        rmseadd
 1    rmseaddquad
 2       rmseexpo
 3        rmselin
 4        rmsemul
 5      rmsemulin
 6       rmsequad
 dtype: object,
 'Values': 0    1860.023815
 1     301.738007
 2     466.247973
 3     421.178788
 4    1963.389640
 5     225.524391
 6     475.561835
 dtype: float64}
```

In [37]:
```
Rmse=pd.DataFrame(data)
Rmse
```

Out[37]:

	Model	Values
0	rmseadd	1860.023815
1	rmseaddquad	301.738007
2	rmseexpo	466.247973
3	rmselin	421.178788
4	rmsemul	1963.389640
5	rmsemulin	225.524391
6	rmsequad	475.561835

Conclusion: - Multiplicative Seasonality With Linear Trend is the best model as comapred to all other models based on the forecasting technique

In [38]:
```
Coco_Cola_Data_New
```

Out[38]:

	Quarter	Sales	quarter	Q1	Q2	Q3	Q4	t	t_square	log_Sales
0	Q1_86	1734.827000	Q1	1	0	0	0	1	1	7.458663
1	Q2_86	2244.960999	Q2	0	1	0	0	2	4	7.716443
2	Q3_86	2533.804993	Q3	0	0	1	0	3	9	7.837477
3	Q4_86	2154.962997	Q4	0	0	0	1	4	16	7.675529
4	Q1_87	1547.818996	Q1	1	0	0	0	5	25	7.344602
5	Q2_87	2104.411995	Q2	0	1	0	0	6	36	7.651791
6	Q3_87	2014.362999	Q3	0	0	1	0	7	49	7.608058
7	Q4_87	1991.746998	Q4	0	0	0	1	8	64	7.596767
8	Q1_88	1869.049999	Q1	1	0	0	0	9	81	7.533186
9	Q2_88	2313.631996	Q2	0	1	0	0	10	100	7.746574
10	Q3_88	2128.320000	Q3	0	0	1	0	11	121	7.663088
11	Q4_88	2026.828999	Q4	0	0	0	1	12	144	7.614228
12	Q1_89	1910.603996	Q1	1	0	0	0	13	169	7.555175
13	Q2_89	2331.164993	Q2	0	1	0	0	14	196	7.754123
14	Q3_89	2206.549995	Q3	0	0	1	0	15	225	7.699185
15	Q4_89	2173.967995	Q4	0	0	0	1	16	256	7.684309
16	Q1_90	2148.278000	Q1	1	0	0	0	17	289	7.672422
17	Q2_90	2739.307999	Q2	0	1	0	0	18	324	7.915461
18	Q3_90	2792.753998	Q3	0	0	1	0	19	361	7.934783
19	Q4_90	2556.009995	Q4	0	0	0	1	20	400	7.846203
20	Q1_91	2480.973999	Q1	1	0	0	0	21	441	7.816407
21	Q2_91	3039.522995	Q2	0	1	0	0	22	484	8.019456
22	Q3_91	3172.115997	Q3	0	0	1	0	23	529	8.062154
23	Q4_91	2879.000999	Q4	0	0	0	1	24	576	7.965199
24	Q1_92	2772.000000	Q1	1	0	0	0	25	625	7.927324
25	Q2_92	3550.000000	Q2	0	1	0	0	26	676	8.174703
26	Q3_92	3508.000000	Q3	0	0	1	0	27	729	8.162801
27	Q4_92	3243.859993	Q4	0	0	0	1	28	784	8.084519
28	Q1_93	3056.000000	Q1	1	0	0	0	29	841	8.024862
29	Q2_93	3899.000000	Q2	0	1	0	0	30	900	8.268475
30	Q3_93	3629.000000	Q3	0	0	1	0	31	961	8.196712
31	Q4_93	3373.000000	Q4	0	0	0	1	32	1024	8.123558
32	Q1_94	3352.000000	Q1	1	0	0	0	33	1089	8.117312
33	Q2_94	4342.000000	Q2	0	1	0	0	34	1156	8.376090

	Quarter	Sales	quarter	Q1	Q2	Q3	Q4	t	t_square	log_Sales
34	Q3_94	4461.000000	Q3	0	0	1	0	35	1225	8.403128
35	Q4_94	4017.000000	Q4	0	0	0	1	36	1296	8.298291
36	Q1_95	3854.000000	Q1	1	0	0	0	37	1369	8.256867
37	Q2_95	4936.000000	Q2	0	1	0	0	38	1444	8.504311
38	Q3_95	4895.000000	Q3	0	0	1	0	39	1521	8.495970
39	Q4_95	4333.000000	Q4	0	0	0	1	40	1600	8.374015
40	Q1_96	4194.000000	Q1	1	0	0	0	41	1681	8.341410
41	Q2_96	5253.000000	Q2	0	1	0	0	42	1764	8.566555

6. Building model by using entire data set of Multiplicative seasonality with linear trend

In [39]:
```
Model_full =smf.ols('Sales~t',data=Coco_Cola_Data).fit()
```

In [40]:
```
Pred_new = Model_full.predict(Coco_Cola_Data_New)
Pred_new
```

Out[40]:
```
0     1492.151553
1     1565.429688
2     1638.707822
3     1711.985956
4     1785.264091
5     1858.542225
6     1931.820360
7     2005.098494
8     2078.376628
9     2151.654763
10    2224.932897
11    2298.211031
12    2371.489166
13    2444.767300
14    2518.045434
15    2591.323569
16    2664.601703
17    2737.879837
18    2811.157972
19    2884.436106
20    2957.714241
21    3030.992375
22    3104.270509
23    3177.548644
24    3250.826778
25    3324.104912
26    3397.383047
27    3470.661181
28    3543.939315
29    3617.217450
30    3690.495584
31    3763.773719
32    3837.051853
33    3910.329987
34    3983.608122
35    4056.886256
36    4130.164390
37    4203.442525
38    4276.720659
39    4349.998793
40    4423.276928
41    4496.555062
dtype: float64
```

In [41]:
```
Coco_Cola_Data_New["Forcasted_Coco_cola_Data"] = Pred_new
Coco_Cola_Data_New
```

Out[41]:

	Quarter	Sales	quarter	Q1	Q2	Q3	Q4	t	t_square	log_Sales	Forcasted_Coco_cola
0	Q1_86	1734.827000	Q1	1	0	0	0	1	1	7.458663	1492.1
1	Q2_86	2244.960999	Q2	0	1	0	0	2	4	7.716443	1565.4
2	Q3_86	2533.804993	Q3	0	0	1	0	3	9	7.837477	1638.7
3	Q4_86	2154.962997	Q4	0	0	0	1	4	16	7.675529	1711.9
4	Q1_87	1547.818996	Q1	1	0	0	0	5	25	7.344602	1785.2
5	Q2_87	2104.411995	Q2	0	1	0	0	6	36	7.651791	1858.5
6	Q3_87	2014.362999	Q3	0	0	1	0	7	49	7.608058	1931.8
7	Q4_87	1991.746998	Q4	0	0	0	1	8	64	7.596767	2005.0
8	Q1_88	1869.049999	Q1	1	0	0	0	9	81	7.533186	2078.3
9	Q2_88	2313.631996	Q2	0	1	0	0	10	100	7.746574	2151.6
10	Q3_88	2128.320000	Q3	0	0	1	0	11	121	7.663088	2224.9
11	Q4_88	2026.828999	Q4	0	0	0	1	12	144	7.614228	2298.2
12	Q1_89	1910.603996	Q1	1	0	0	0	13	169	7.555175	2371.4
13	Q2_89	2331.164993	Q2	0	1	0	0	14	196	7.754123	2444.7
14	Q3_89	2206.549995	Q3	0	0	1	0	15	225	7.699185	2518.0
15	Q4_89	2173.967995	Q4	0	0	0	1	16	256	7.684309	2591.3
16	Q1_90	2148.278000	Q1	1	0	0	0	17	289	7.672422	2664.6
17	Q2_90	2739.307999	Q2	0	1	0	0	18	324	7.915461	2737.8
18	Q3_90	2792.753998	Q3	0	0	1	0	19	361	7.934783	2811.1
19	Q4_90	2556.009995	Q4	0	0	0	1	20	400	7.846203	2884.4
20	Q1_91	2480.973999	Q1	1	0	0	0	21	441	7.816407	2957.7
21	Q2_91	3039.522995	Q2	0	1	0	0	22	484	8.019456	3030.9
22	Q3_91	3172.115997	Q3	0	0	1	0	23	529	8.062154	3104.2
23	Q4_91	2879.000999	Q4	0	0	0	1	24	576	7.965199	3177.5
24	Q1_92	2772.000000	Q1	1	0	0	0	25	625	7.927324	3250.8
25	Q2_92	3550.000000	Q2	0	1	0	0	26	676	8.174703	3324.1
26	Q3_92	3508.000000	Q3	0	0	1	0	27	729	8.162801	3397.3
27	Q4_92	3243.859993	Q4	0	0	0	1	28	784	8.084519	3470.6
28	Q1_93	3056.000000	Q1	1	0	0	0	29	841	8.024862	3543.9
29	Q2_93	3899.000000	Q2	0	1	0	0	30	900	8.268475	3617.2
30	Q3_93	3629.000000	Q3	0	0	1	0	31	961	8.196712	3690.4
31	Q4_93	3373.000000	Q4	0	0	0	1	32	1024	8.123558	3763.7
32	Q1_94	3352.000000	Q1	1	0	0	0	33	1089	8.117312	3837.0

	Quarter	Sales	quarter	Q1	Q2	Q3	Q4	t	t_square	log_Sales	Forcasted_Coco_cola
33	Q2_94	4342.000000	Q2	0	1	0	0	34	1156	8.376090	3910.3
34	Q3_94	4461.000000	Q3	0	0	1	0	35	1225	8.403128	3983.6
35	Q4_94	4017.000000	Q4	0	0	0	1	36	1296	8.298291	4056.8
36	Q1_95	3854.000000	Q1	1	0	0	0	37	1369	8.256867	4130.1
37	Q2_95	4936.000000	Q2	0	1	0	0	38	1444	8.504311	4203.4
38	Q3_95	4895.000000	Q3	0	0	1	0	39	1521	8.495970	4276.7
39	Q4_95	4333.000000	Q4	0	0	0	1	40	1600	8.374015	4349.9
40	Q1_96	4194.000000	Q1	1	0	0	0	41	1681	8.341410	4423.2
41	Q2_96	5253.000000	Q2	0	1	0	0	42	1764	8.566555	4496.5

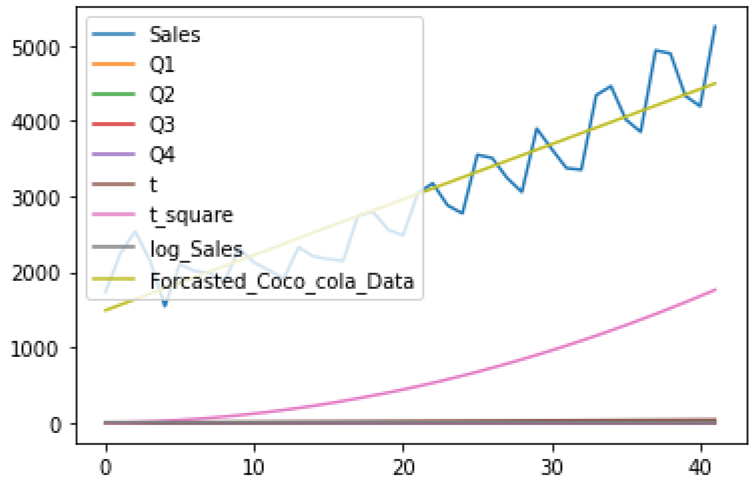
In [42]:
```python
new_var = pd.concat([Coco_Cola_Data,Coco_Cola_Data_New])
new_var
```

Out[42]:

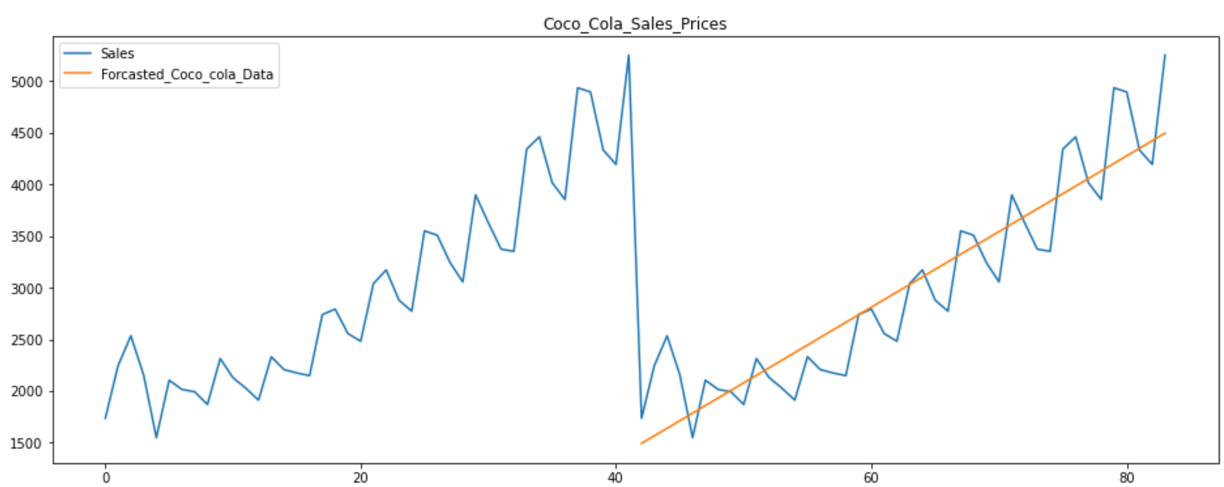
	Quarter	Sales	quarter	Q1	Q2	Q3	Q4	t	t_square	log_Sales	Forcasted_Co
0	Q1_86	1734.827000	Q1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	Q2_86	2244.960999	Q2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	Q3_86	2533.804993	Q3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	Q4_86	2154.962997	Q4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	Q1_87	1547.818996	Q1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	
37	Q2_95	4936.000000	Q2	0.0	1.0	0.0	0.0	38.0	1444.0	8.504311	
38	Q3_95	4895.000000	Q3	0.0	0.0	1.0	0.0	39.0	1521.0	8.495970	
39	Q4_95	4333.000000	Q4	0.0	0.0	0.0	1.0	40.0	1600.0	8.374015	
40	Q1_96	4194.000000	Q1	1.0	0.0	0.0	0.0	41.0	1681.0	8.341410	
41	Q2_96	5253.000000	Q2	0.0	1.0	0.0	0.0	42.0	1764.0	8.566555	

84 rows × 11 columns

In [44]:
```
Coco_Cola_Data_New.plot()
plt.show()
```



In [46]:
```
new_var[['Sales','Forcasted_Coco_cola_Data']].reset_index(drop=True).plot(figsize
plt.title('Coco_Cola_Sales_Prices')
plt.show()
```



*****

2. Data Driven Forecasting Techniques

1. Import Neccesary Libraries

In [47]:
```
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import Holt
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
import statsmodels.graphics.tsaplots as tsa_plots
import statsmodels.tsa.statespace as tm_models
from datetime import datetime, time
```

2. Import Data

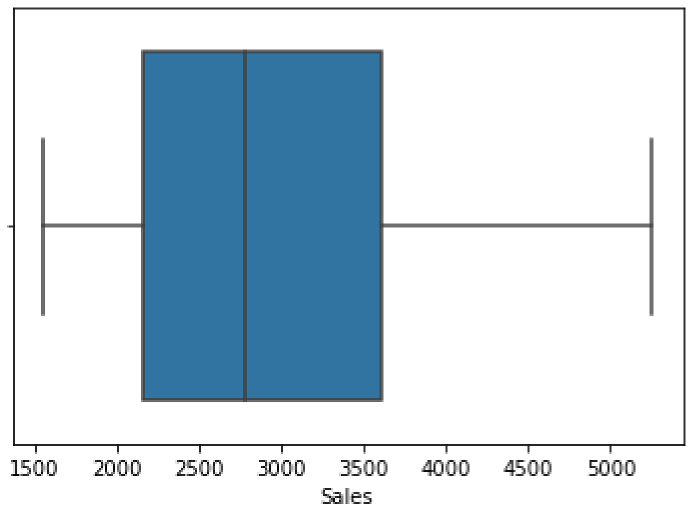
In [48]:
```
Data_Driven_Coco = pd.read_csv('CocaCola_Sales_Rawdata.csv')
Data_Driven_Coco
```

Out[48]:

	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

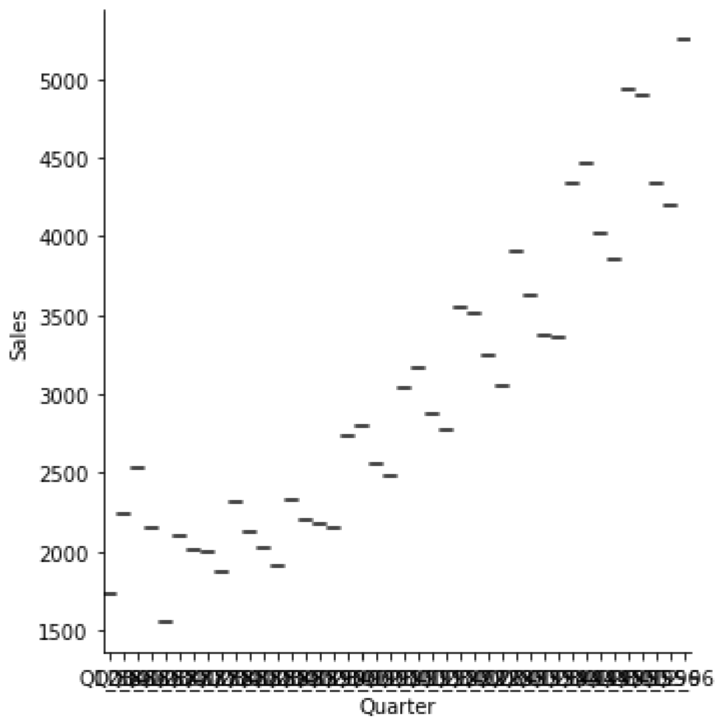
In [50]:
```
sns.boxplot('Sales', data = Data_Driven_Coco)
plt.show()
```

```
C:\Users\nandini\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments with
out an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



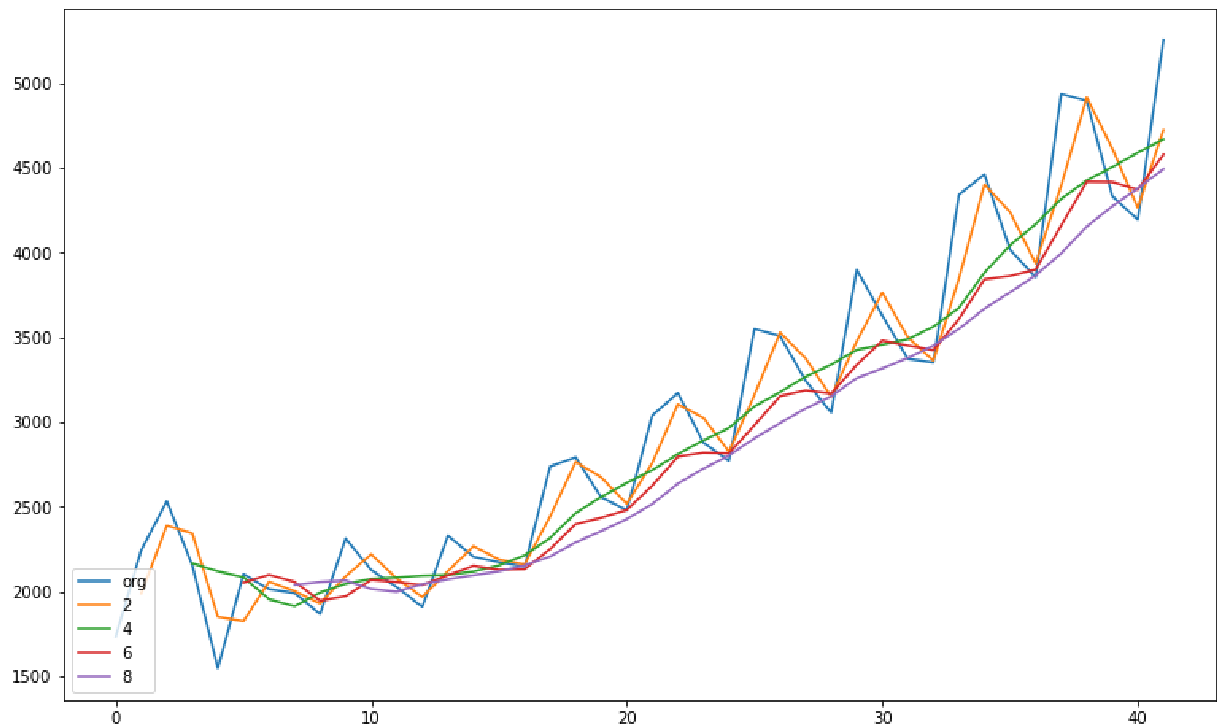
In [53]:
```
sns.factorplot('Quarter', 'Sales', data = Data_Driven_Coco, kind="box")
plt.show()
```

```
C:\Users\nandini\anaconda3\lib\site-packages\seaborn\categorical.py:3714: UserW
arning: The `factorplot` function has been renamed to `catplot`. The original n
ame will be removed in a future release. Please update your code. Note that the
default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\Users\nandini\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



1. Moving Average

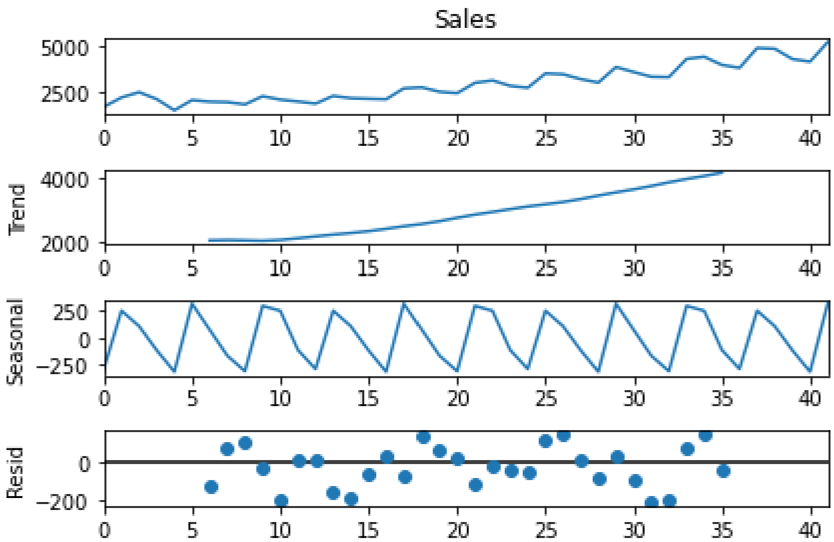
In [65]:
```
plt.figure(figsize=(13,8))
Data_Driven_Coco.Sales.plot(label="org")
for i in range(2,10,2):
    Data_Driven_Coco["Sales"].rolling(i).mean().plot(label=str(i))
plt.legend(loc=3)
plt.show()
```

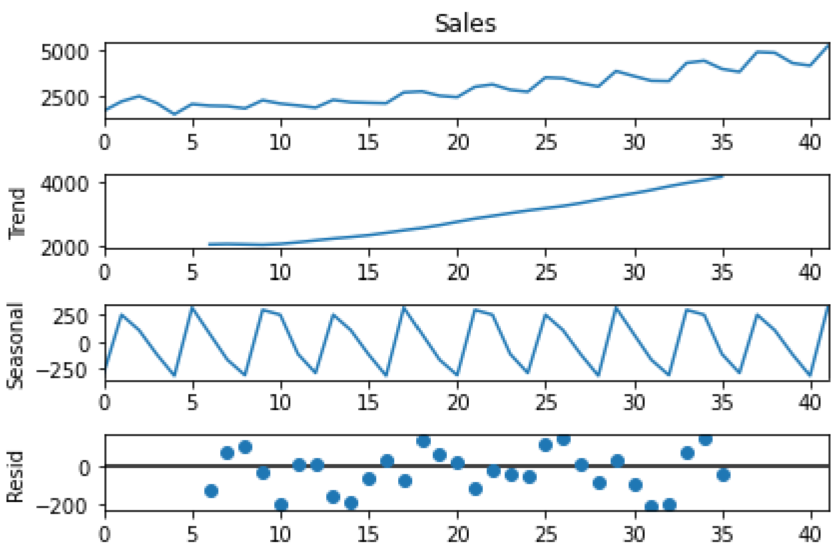


2. Time series decomposition plot

In [67]:
```
Decompse_ts_ad = seasonal_decompose(Data_Driven_Coco.Sales, period=12)
Decompse_ts_ad.plot()
```

Out[67]:

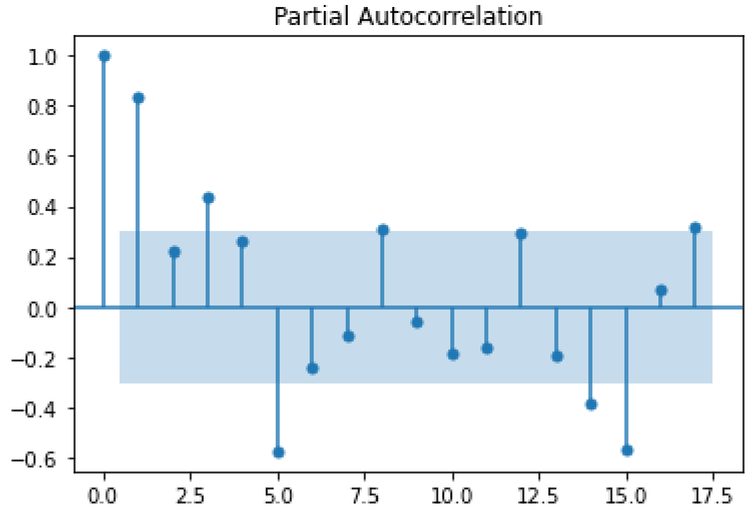


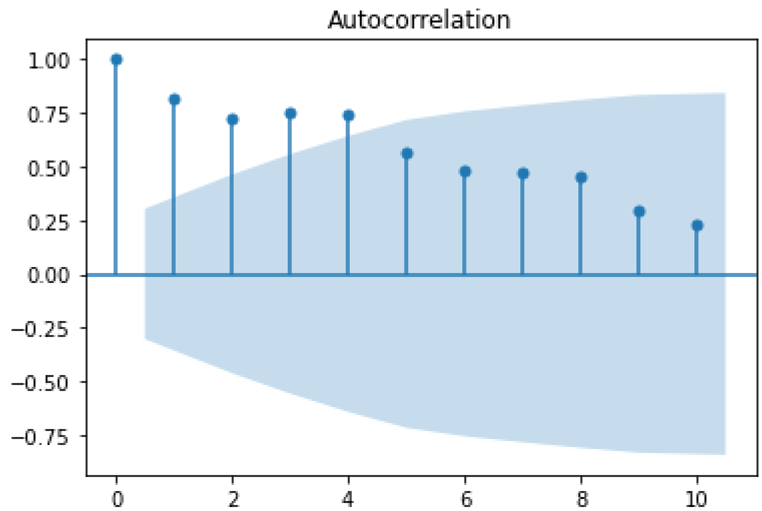


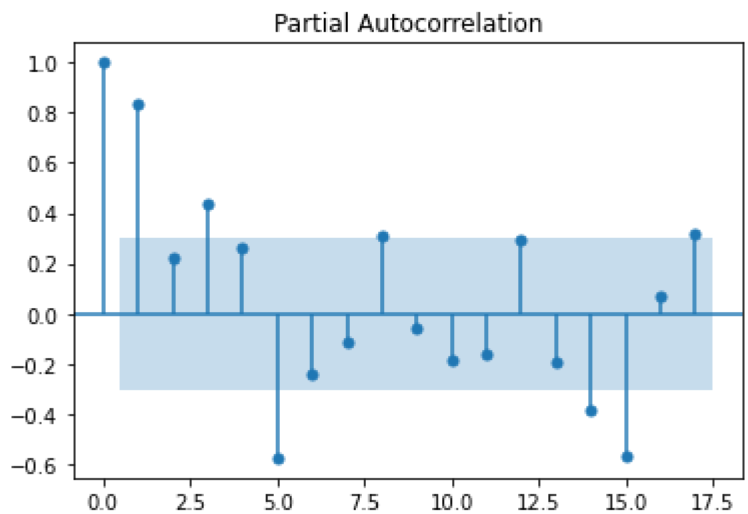
3. ACF & PACF plots on original data sets

In [72]:
```
tsa_plots.plot_acf(Data_Driven_Coco.Sales,lags=10)
tsa_plots.plot_pacf(Data_Driven_Coco.Sales)
```

Out[72]:







Train test split

In [73]:
```
train = Data_Driven_Coco.head(100)
test = Data_Driven_Coco.tail(20)
```

Creating function to calculate MAPE values for the test data

In [75]:
```
def MAPE(pred, org):
    temp = abs((pred-org))*100/org
    return np.mean(temp)
```

Data Driven Forecasting Methods

1. Simple Exponential Smoothing Method

In [77]:
```
ses_model = SimpleExpSmoothing(train['Sales']).fit()
pred_ses = ses_model.predict(start = test.index[0], end = test.index[-1])
MAPE(pred_ses,test.Sales)
```

Out[77]:
```
9.70054258539694
```

2. Holts Method

In [80]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [81]:
```
Holts_model = Holt(train['Sales']).fit()
pred_Holts = Holts_model.predict(start = test.index[0], end = test.index[-1])
MAPE(pred_Holts,test.Sales)
```

Out[81]:
```
10.986623183911579
```

3. Holts Winter Exponential Smoothing

In [82]:
```
Holts_Winter_model = ExponentialSmoothing(train['Sales']).fit()
pred_Holts_Winter = Holts_Winter_model.predict(start = test.index[0], end = test.
MAPE(pred_Holts_Winter,test.Sales)
```

Out[82]:
```
9.70054258539694
```

4. Holts Winter Exponential Smoothing With Multiplicative Seasonality & Additive Trend

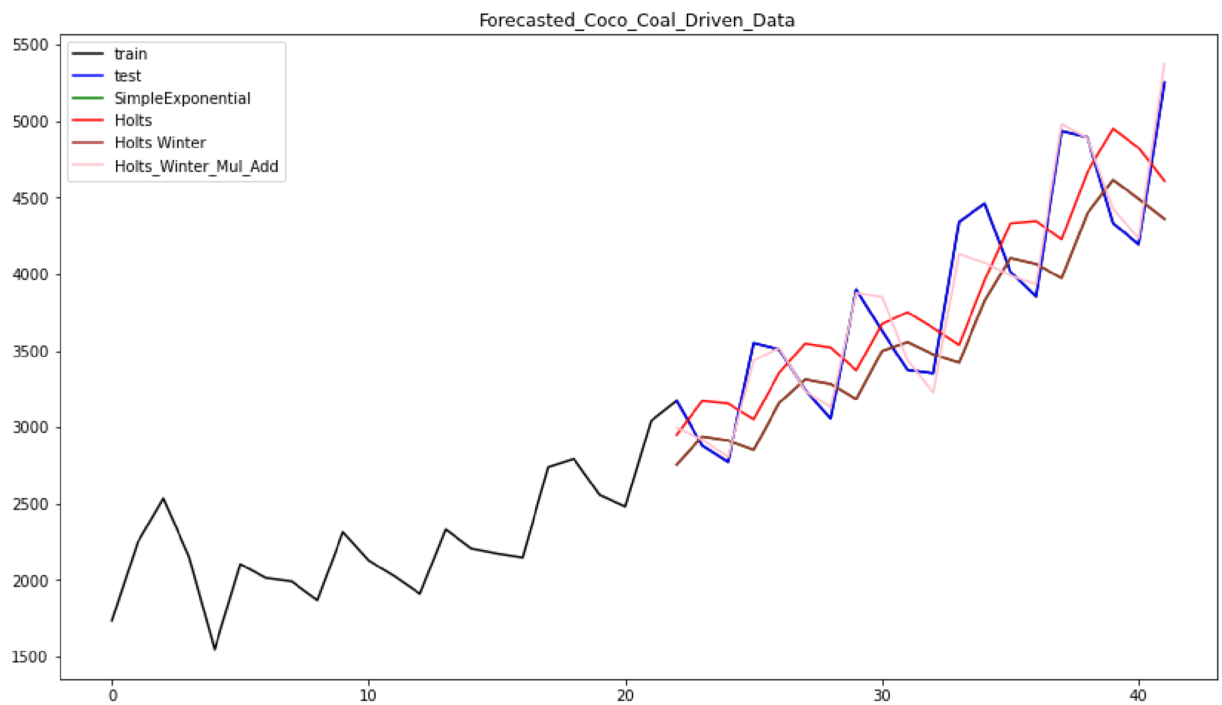
In [88]:
```
Holts_Winter_Mul_Add_model = ExponentialSmoothing(train["Sales"], seasonal = "Mul
pred_Holts_Winter_Mul_Add = Holts_Winter_Mul_Add_model.predict(start = test.index
MAPE(pred_Holts_Winter_Mul_Add,test.Sales)
```

Out[88]: 2.4669537850086103

Conclusion:- Holts Winter Exponential Smoothing With Multiplicative Seasonality & Additive Trend is best model compare to all other methods based on data driven forecasting technique

Visualization of forecasted values of test data using different methods

In [97]:
```
plt.figure(figsize=(14,8))
plt.plot(train.index, train["Sales"], label='train', color = "black")
plt.plot(test.index, test["Sales"], label='test', color = "blue")
plt.plot(pred_ses.index,pred_ses, label='SimpleExponential', color = "Green")
plt.plot(pred_Holts.index,pred_Holts, label='Holts', color = "Red")
plt.plot(pred_Holts_Winter.index,pred_Holts_Winter, label='Holts Winter', color =
plt.plot(pred_Holts_Winter_Mul_Add.index,pred_Holts_Winter_Mul_Add, label='Holts_
plt.legend(loc='best')
plt.title("Forecasted_Coco_Coal_Driven_Data")
plt.show()
```



The END