In [1]:
```
import pandas as pd
```

In [49]:
```
cocacola = pd.read_csv('CocaCola_Sales_Rawdata.csv')
cocacola
```

Out[49]:

	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 [51]:
```
import numpy as np
quarter=['Q1','Q2','Q3','Q4']
n=cocacola['Quarter'][0]
n[0:2]
```

Out[51]: 'Q1'

In [52]:
```
cocacola['quarter']=0
```

In [53]:
```
for i in range(42):
    n=cocacola['Quarter'][i]
    cocacola['quarter'][i]=n[0:2]
```

```
<ipython-input-53-081269711e9b>: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)
  cocacola['quarter'][i]=n[0:2]
C:\Users\Hp\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)
```

In [54]:
```
dummy=pd.DataFrame(pd.get_dummies(cocacola['quarter']))
dummy
```

Out[54]:

	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 [55]:
```
coca=pd.concat((cocacola,dummy),axis=1)
t= np.arange(1,43)
coca['t']=t
coca['t_square']=coca['t']*coca['t']
coca
```

Out[55]:

	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 [56]:
```
log_Sales=np.log(coca['Sales'])
coca['log_Sales']=log_Sales
coca
```

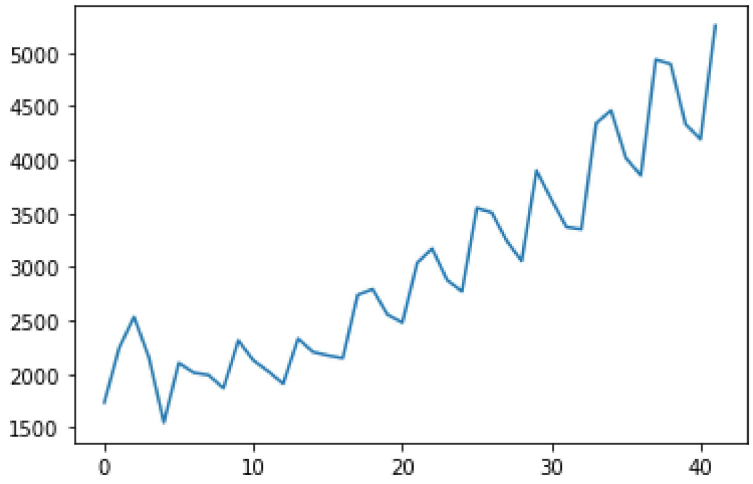
Out[56]:

	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

In [57]:
```
train= coca.head(38)
test=coca.tail(4)
coca.Sales.plot()
```

Out[57]: <AxesSubplot:>



In [58]:
```
import statsmodels.formula.api as smf
```

In [59]:
```
#Linear model
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[59]: 421.17878760022745

In [60]:
```
#quadratic model
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[60]: 475.56183518316163

In [61]:
```
#exponential model
expo=smf.ols('log_Sales~t',data=train).fit()
predexp=pd.Series(expo.predict(pd.DataFrame(test['t'])))
predexp
rmseexpo=np.sqrt(np.mean((np.array(test['Sales'])-np.array(np.exp(predexp)))**2))
rmseexpo
```

Out[61]: 466.2479731067161

In [62]:
```
#additive seasonality
additive= smf.ols('Sales~ Q1+Q2+Q3+Q4',data=train).fit()
predadd=pd.Series(additive.predict(pd.DataFrame(test[['Q1','Q2','Q3','Q4']])))
predadd
rmseadd=np.sqrt(np.mean((np.array(test['Sales'])-np.array(predadd))**2))
rmseadd
```

Out[62]: 1860.0238154547276

In [63]:
```
#additive seasonality with linear trend
addlinear= smf.ols('Sales~t+Q1+Q2+Q3+Q4',data=train).fit()
predaddlinear=pd.Series(addlinear.predict(pd.DataFrame(test[['t','Q1','Q2','Q3',
predaddlinear

rmseaddlinear=np.sqrt(np.mean((np.array(test['Sales'])-np.array(predaddlinear))**
rmseaddlinear
```

Out[63]: 464.98290239822535

In [64]:
```
#additive seasonality with quadratic trend
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[64]: 301.7380071934848

In [65]:
```
#multiplicative seasonality
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[65]: 1963.3896400779704

In [66]:
```
#multiplicative seasonality with linear trend
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[66]: 225.52439049825057

In [67]:
```
#multiplicative seasonality with quadratic trend
mul_quad= smf.ols('log_Sales~t+t_square+Q1+Q2+Q3+Q4',data=train).fit()
pred_mul_quad= pd.Series(mul_quad.predict(test[['t','t_square','Q1','Q2','Q3','Q4
rmse_mul_quad=np.sqrt(np.mean((np.array(test['Sales'])-np.array(np.exp(pred_mul_q
rmse_mul_quad
```

Out[67]:
```
581.8457187961241
```

In [68]:
```
#tabulating the rmse values

data={'Model':pd.Series(['rmse_mul_quad','rmseadd','rmseaddlinear','rmseaddquad',
data
```

Out[68]:
```
{'Model': 0    rmse_mul_quad
 1          rmseadd
 2    rmseaddlinear
 3      rmseaddquad
 4         rmseexpo
 5          rmselin
 6          rmsemul
 7        rmsemulin
 8         rmsequad
 dtype: object,
 'Values': 0     581.845719
 1    1860.023815
 2     464.982902
 3     301.738007
 4     466.247973
 5     421.178788
 6    1963.389640
 7     225.524390
 8     475.561835
 dtype: float64}
```

In [69]:
```
Rmse=pd.DataFrame(data)
Rmse
```

Out[69]:

	Model	Values
0	rmse_mul_quad	581.845719
1	rmseadd	1860.023815
2	rmseaddlinear	464.982902
3	rmseaddquad	301.738007
4	rmseexpo	466.247973
5	rmselin	421.178788
6	rmsemul	1963.389640
7	rmsemulin	225.524390
8	rmsequad	475.561835

CocaCola Data-Driven

In [77]:
```
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing # SES
from statsmodels.tsa.holtwinters import Holt # Holts Exponential Smoothing
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
```

In [78]:
```
cocacola_1 = pd.read_csv('CocaCola_Sales_Rawdata.csv')
cocacola_1
```

Out[78]:

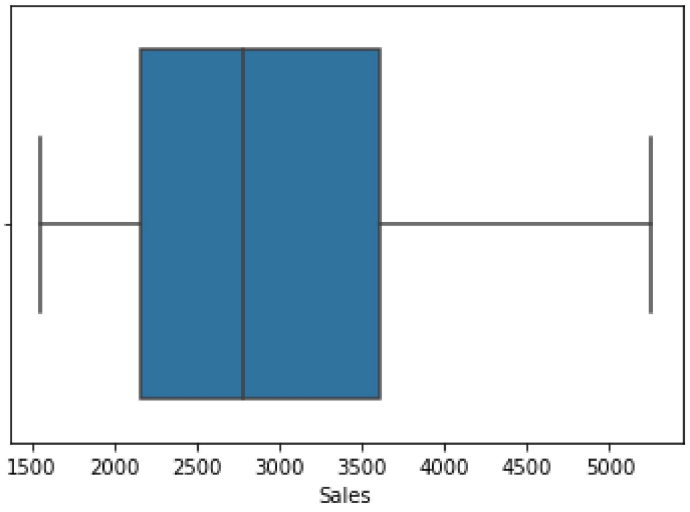
	Quarter	Sales
0	Q1_86	1734.827000
1	Q2_86	2244.960999
2	Q3_86	2533.804993
3	Q4_86	2154.962997
4	Q1_87	1547.818996
5	Q2_87	2104.411995
6	Q3_87	2014.362999
7	Q4_87	1991.746998
8	Q1_88	1869.049999
9	Q2_88	2313.631996
10	Q3_88	2128.320000
11	Q4_88	2026.828999
12	Q1_89	1910.603996
13	Q2_89	2331.164993
14	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 [79]:
```
sns.boxplot('Sales',data=cocacola_1)
```

```
C:\Users\Hp\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: 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 without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

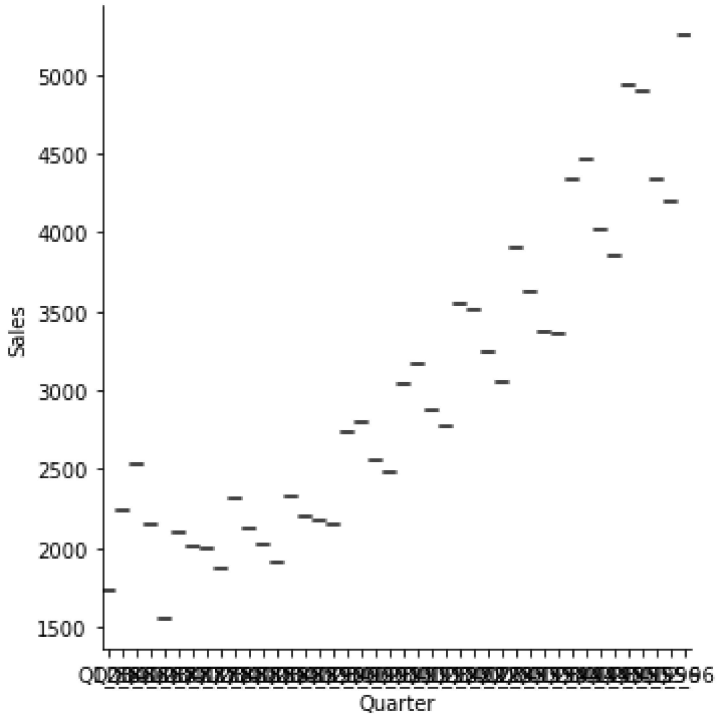
Out[79]: <AxesSubplot:xlabel='Sales'>



In [80]:
```
sns.factorplot("Quarter","Sales",data=cocacola_1,kind="box")
```

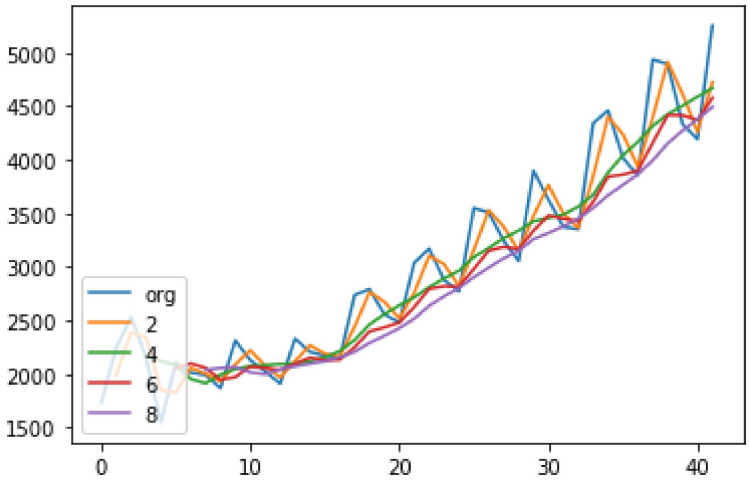
```
C:\Users\Hp\anaconda3\lib\site-packages\seaborn\categorical.py:3714: UserWarnin
g: The `factorplot` function has been renamed to `catplot`. The original name w
ill be removed in a future release. Please update your code. Note that the defa
ult `kind` in `factorplot` (`'point'`) has changed ``'strip'`` in `catplot`.
  warnings.warn(msg)
C:\Users\Hp\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[80]: <seaborn.axisgrid.FacetGrid at 0xace7058>



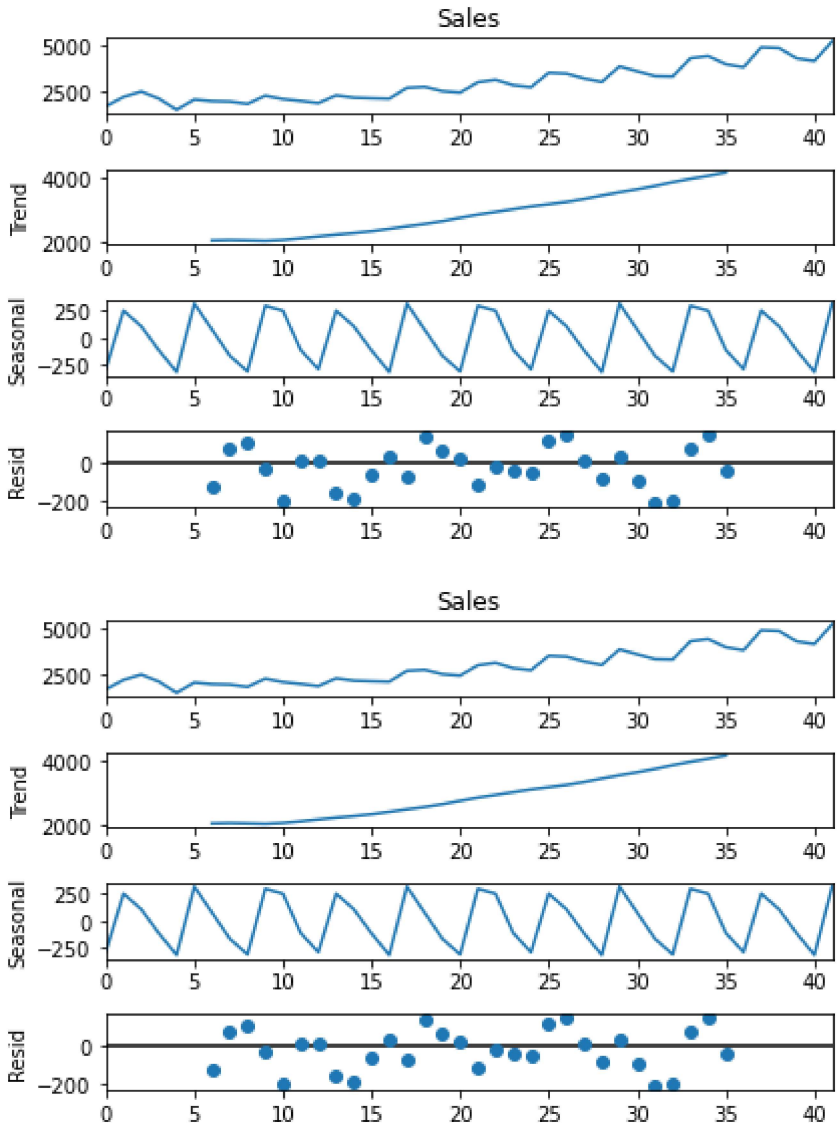
In [81]:
```
# moving average for the time series to understand better about the trend charact
cocacola_1.Sales.plot(label="org")
for i in range(2,10,2):
    cocacola_1["Sales"].rolling(i).mean().plot(label=str(i))
plt.legend(loc=3)
```

Out[81]: <matplotlib.legend.Legend at 0xab9eaa8>



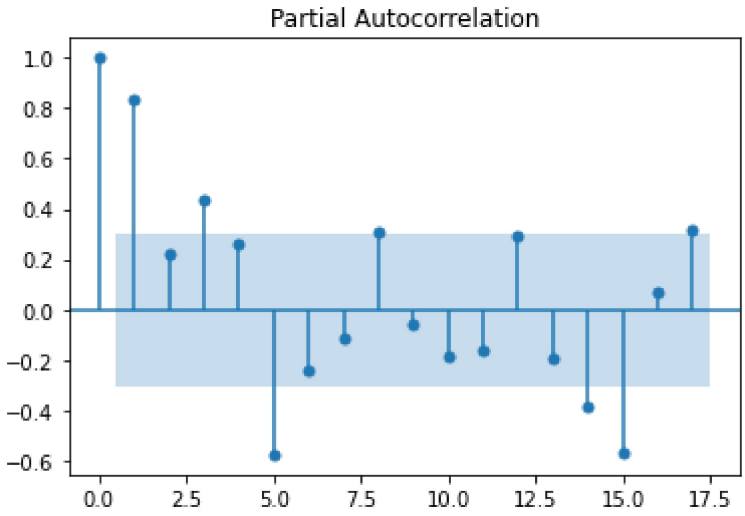
In [83]:
```
# Time series decomposition plot
decompose_ts_add = seasonal_decompose(cocacola_1.Sales,period =12)
decompose_ts_add.plot()
```

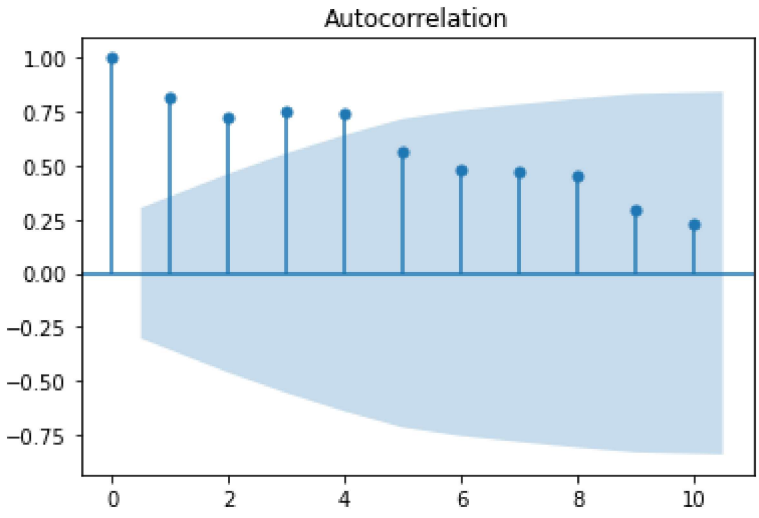
Out[83]:

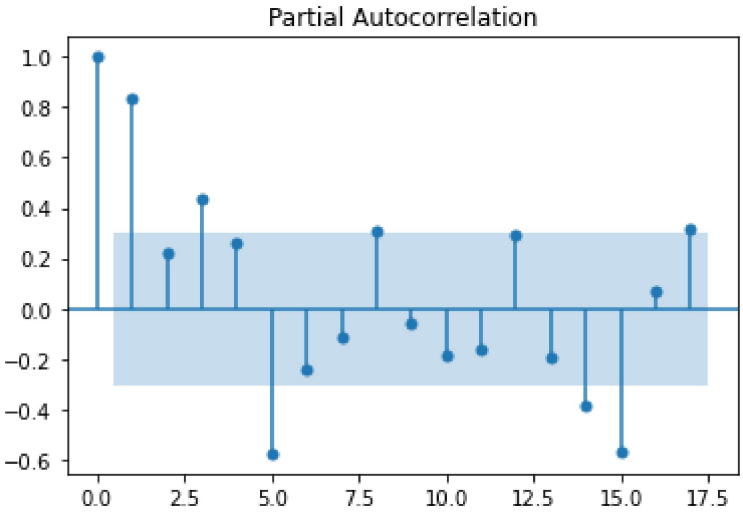


In [84]:
```
# ACF plots and PACF plots on Original data sets
tsa_plots.plot_acf(cocacola_1.Sales,lags=10)
tsa_plots.plot_pacf(cocacola_1.Sales)
```

Out[84]:







In [85]:
```
Train = cocacola_1.head(48)
Test =cocacola_1.tail(12)
```

In [86]:
```
# Creating a function to calculate the MAPE value for test data
def MAPE(pred,org):
    temp = np.abs((pred-org))*100/org
    return np.mean(temp)
```

In [88]:
```
# Simple Exponential Method
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[88]: 9.765093559919109

In [90]:
```
# Holt method
hw_model = Holt(Train["Sales"]).fit()
pred_hw = hw_model.predict(start = Test.index[0],end = Test.index[-1])
MAPE(pred_hw,Test.Sales)
```

Out[90]: 11.025181091221222

In [91]:
```
# Holts winter exponential smoothing with additive seasonality and additive trend
hwe_model_add_add = ExponentialSmoothing(Train["Sales"],seasonal="add",trend="add
pred_hwe_add_add = hwe_model_add_add.predict(start = Test.index[0],end = Test.ind
MAPE(pred_hwe_add_add,Test.Sales)
```

```
<ipython-input-91-0da59f3c1cca>:2: FutureWarning: the 'damped'' keyword is depr
ecated, use 'damped_trend' instead
  hwe_model_add_add = ExponentialSmoothing(Train["Sales"],seasonal="add",trend
="add",seasonal_periods=4,damped=True).fit()
```

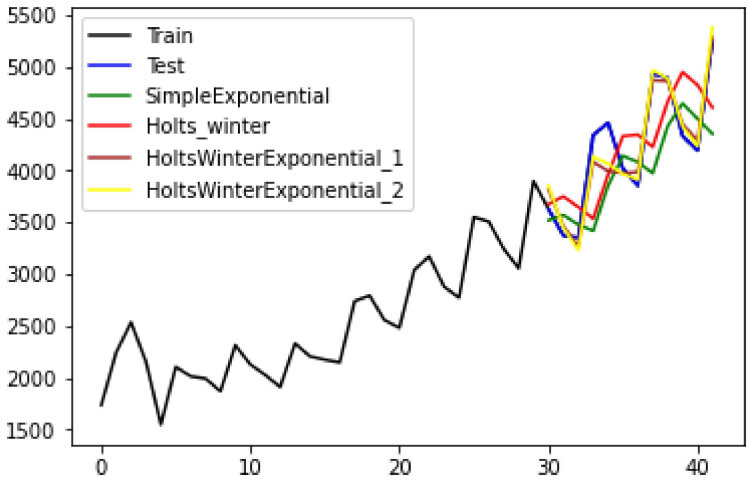
Out[91]:
```
3.2478075162123416
```

In [92]:
```
# Holts winter exponential smoothing with multiplicative seasonality and additive
hwe_model_mul_add = ExponentialSmoothing(Train["Sales"],seasonal="mul",trend="add
pred_hwe_mul_add = hwe_model_mul_add.predict(start = Test.index[0],end = Test.ind
MAPE(pred_hwe_mul_add,Test.Sales)
```

Out[92]:
```
2.8845566683394424
```

In [93]:
```
# Visualization of Forecasted values for Test data set using different methods
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_hw.index, pred_hw, label='Holts_winter',color="red")
plt.plot(pred_hwe_add_add.index,pred_hwe_add_add,label="HoltsWinterExponential_1"
plt.plot(pred_hwe_mul_add.index,pred_hwe_mul_add,label="HoltsWinterExponential_2"
plt.legend(loc='best')
```

Out[93]:
```
<matplotlib.legend.Legend at 0xad07fa0>
```



In [ ]:
```

```