In [1]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\admin\anaconda3\lib\site-packages (1.
6.0)
Requirement already satisfied: scipy in c:\users\admin\anaconda3\lib\site-packages (from
xgboost) (1.7.1)
Requirement already satisfied: numpy in c:\users\admin\anaconda3\lib\site-packages (from
xgboost) (1.20.3)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import numpy as np
import pandas as pd, datetime
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
from time import time
import os
from math import sqrt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import itertools
import statsmodels.api as sm
from statsmodels.tsa.stattools import acf,pacf
from statsmodels.tsa.arima_model import  ARIMA
from sklearn import model_selection
from sklearn.metrics import mean_squared_error, r2_score
from pandas import DataFrame
import xgboost as xgb
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
store = pd.read_excel("C:\\Users\\Admin\\Downloads\\assignment 10\\CocaCola_Sales_Rawda
```

In [4]:
```
store.head()
```

Out[4]:

	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 [5]:
```
quarter=['Q1','Q2','Q3','Q4']
n=store['Quarter'][0]
n[0:2]

store['quarter']=0
```

In [6]:
```
for i in range(42):
    n=store['Quarter'][i]
    store['quarter'][i]=n[0:2]
```

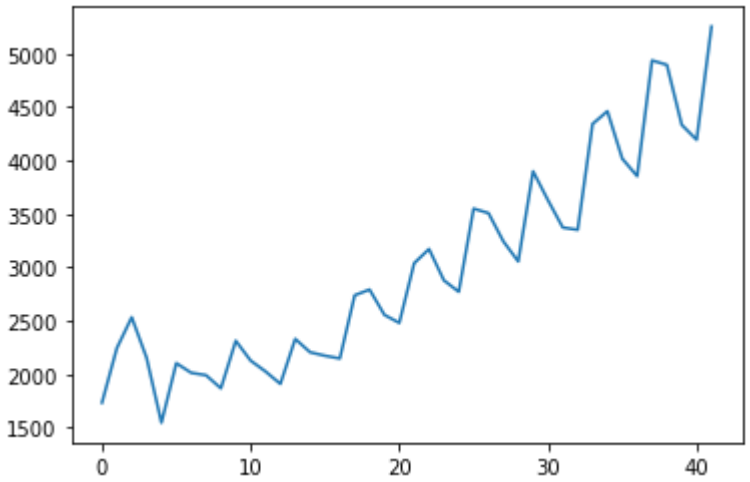
In [7]:
```
dummy=pd.DataFrame(pd.get_dummies(store['quarter']))
```

In [8]:
```
coco=pd.concat((store,dummy),axis=1)
t= np.arange(1,43)
coco['t']=t
coco['t_square']=coco['t']*coco['t']
```

In [9]:
```
log_Sales=np.log(coco['Sales'])
coco['log_Sales']=log_Sales
```

In [10]:
```
train= coco.head(38)
test=coco.tail(4)
coco.Sales.plot()
```

Out[10]:
```
<AxesSubplot:>
```



In [11]:
```
import statsmodels.formula.api as smf


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[11]:
```
421.17878760022813
```

In [12]:
```
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[12]: 475.56183518315095

In [13]:
```
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[13]: 466.24797310672346

In [14]:
```
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[14]: 1860.0238154547283

In [15]:
```
addlinear= smf.ols('Sales~t+Q1+Q2+Q3+Q4',data=train).fit()
predaddlinear=pd.Series(addlinear.predict(pd.DataFrame(test[['t','Q1','Q2','Q3','Q4']]))
predaddlinear

rmseaddlinear=np.sqrt(np.mean((np.array(test['Sales'])-np.array(predaddlinear))**2))
rmseaddlinear
```

Out[15]: 464.98290239822427

In [16]:
```
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','Q3',
rmseaddquad=np.sqrt(np.mean((np.array(test['Sales'])-np.array(predaddquad))**2))
rmseaddquad
```

Out[16]: 301.73800719352977

In [17]:
```
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[17]: 1963.3896400779709

In [18]:
```
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)))**2))
rmsemulin
```

Out[18]: 225.5243904982721

In [19]:
```
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_quad)))
rmse_mul_quad
```

Out[19]:
```
581.8457187971785
```

In [20]:
```
data={'Model':pd.Series(['rmse_mul_quad','rmseadd','rmseaddlinear','rmseaddquad','rmsee
data
```

Out[20]:
```
{'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}
```

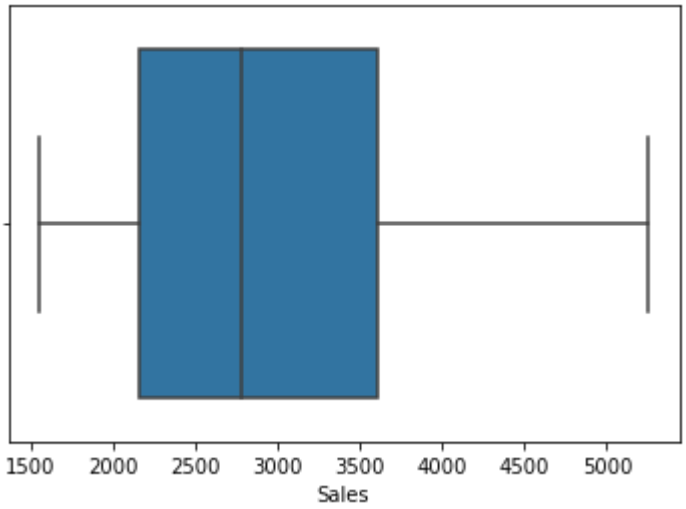
In [21]:
```
Rmse=pd.DataFrame(data)
Rmse
```

Out[21]:

	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

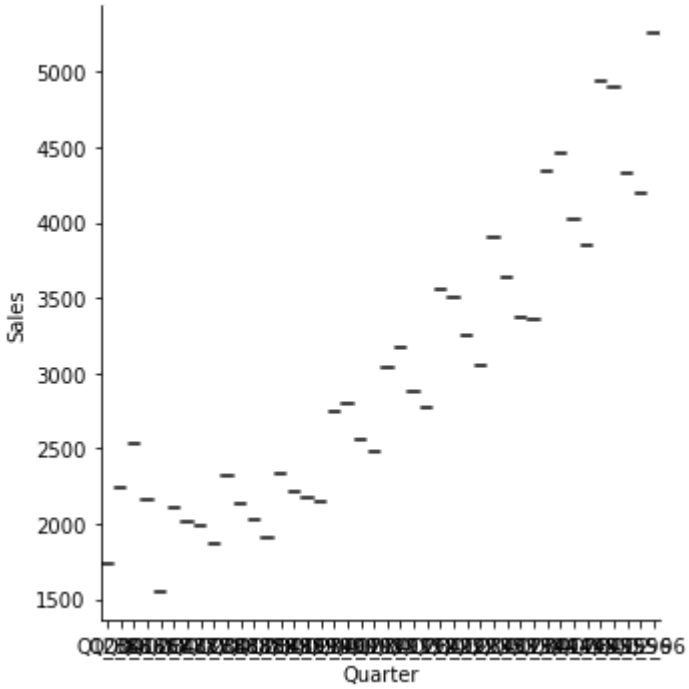
In [22]:
```
sns.boxplot("Sales",data=store)
```

Out[22]:
```
<AxesSubplot:xlabel='Sales'>
```



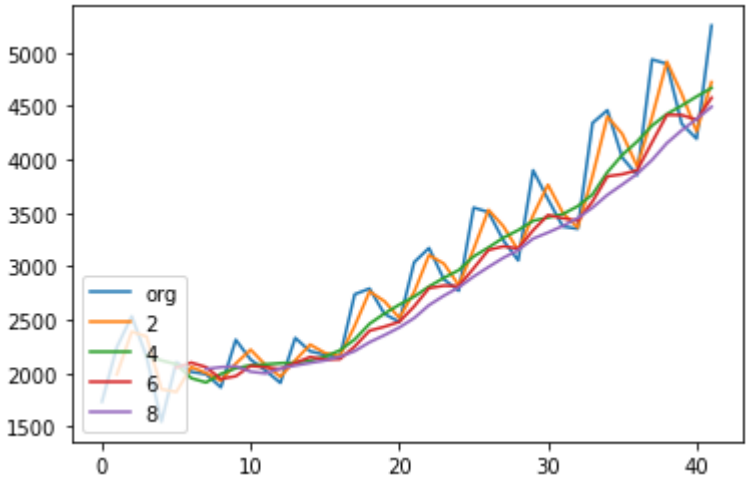
In [23]:
```
sns.factorplot("Quarter","Sales",data=store,kind="box")
```

Out[23]:
```
<seaborn.axisgrid.FacetGrid at 0x21408197640>
```



In [24]:
```
store.Sales.plot(label="org")
for i in range(2,10,2):
    store["Sales"].rolling(i).mean().plot(label=str(i))
plt.legend(loc=3)
```

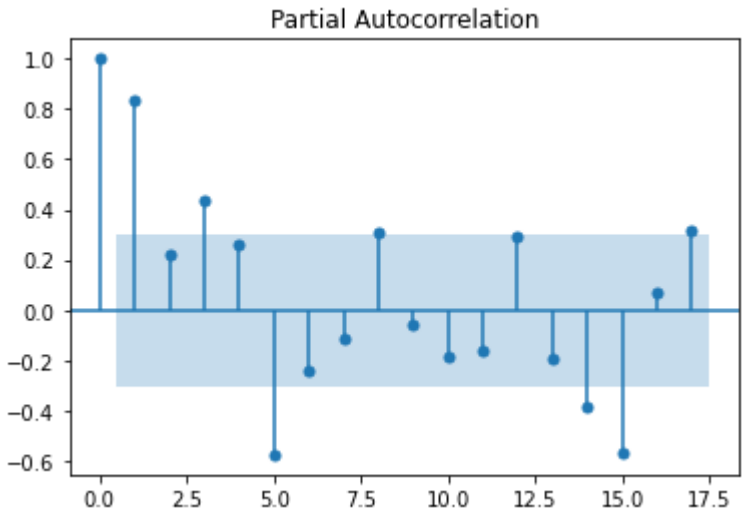
Out[24]:
```
<matplotlib.legend.Legend at 0x21408610ac0>
```

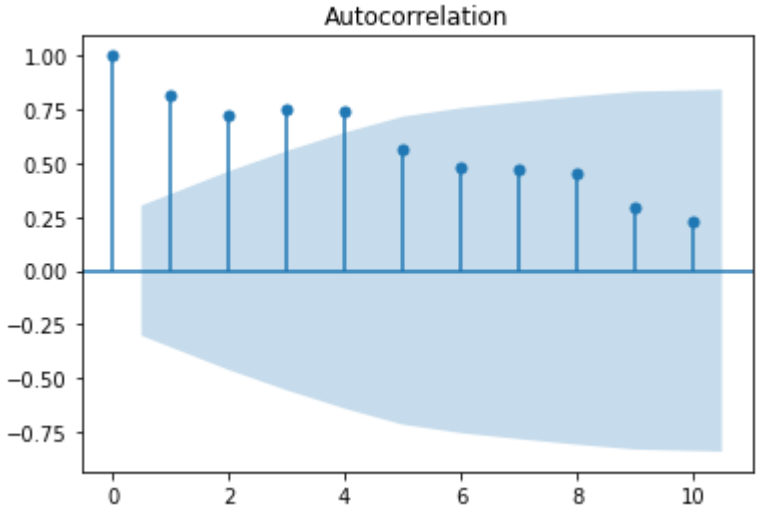


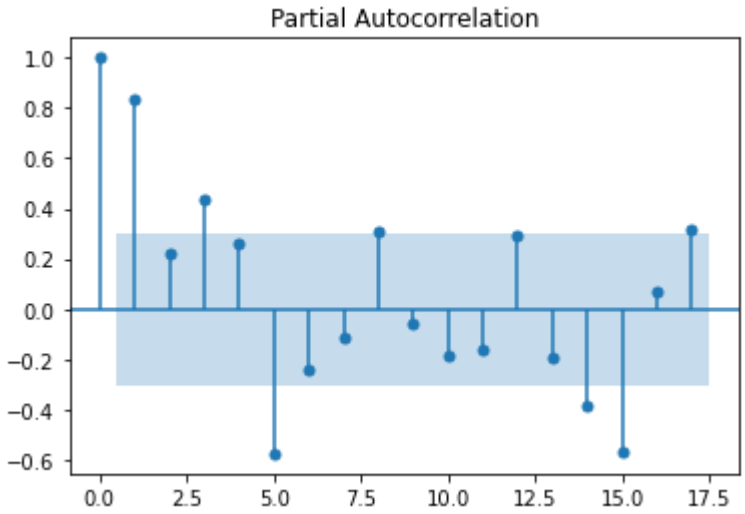
In [25]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
from statsmodels.tsa.holtwinters import Holt
from statsmodels.tsa.holtwinters import ExponentialSmoothing
import statsmodels.graphics.tsaplots as tsa_plots
import statsmodels.tsa.statespace as tm_models
from datetime import datetime,time
```

In [26]:
```
tsa_plots.plot_acf(store.Sales,lags=10)
tsa_plots.plot_pacf(store.Sales)
```

Out[26]:







In [27]:
```
Train = store.head(48)
Test =store.tail(12)
```

In [28]:
```
def MAPE(pred,org):
    temp = np.abs((pred-org))*100/org
    return np.mean(temp)
```

In [29]:
```
ses_model = SimpleExpSmoothing(Train["Sales"]).fit()
pred_ses = ses_model.predict(start = Test.index[0],end = Test.index[-1])
MAPE(pred_ses,Test.Sales) # 9.76
```

Out[29]:
```
9.765094286822912
```

In [30]:
```
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[30]:
```
11.025182503719508
```

In [31]:
```
hwe_model_add_add = ExponentialSmoothing(Train["Sales"],seasonal="add",trend="add",seas
```

```
pred_hwe_add_add = hwe_model_add_add.predict(start = Test.index[0],end = Test.index[-1]
MAPE(pred_hwe_add_add,Test.Sales)
```

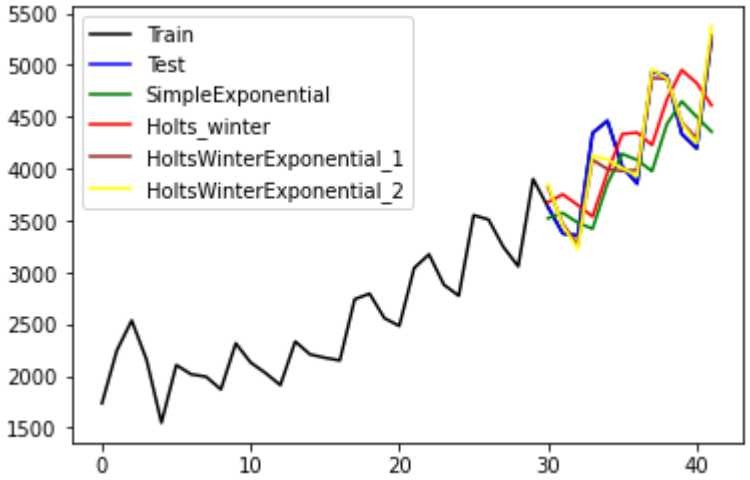
Out[31]:
```
3.245837019668915
```

In [32]:
```
hwe_model_mul_add = ExponentialSmoothing(Train["Sales"],seasonal="mul",trend="add",seas
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[32]:
```
2.853976992162947
```

In [33]:
```
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",color
plt.plot(pred_hwe_mul_add.index,pred_hwe_mul_add,label="HoltsWinterExponential_2",color
plt.legend(loc='best')
```

Out[33]:
```
<matplotlib.legend.Legend at 0x21408c6adc0>
```



In [ ]: