In [1]:
```
# Import libraries
from pandas import read_csv
from matplotlib import pyplot
from numpy import sqrt
import warnings
import itertools
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

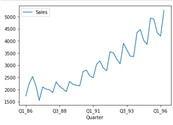
In [2]:
```
import pandas as pd
series = pd.read_excel("/Users/SAURABH/Saurabh patil/DATA SCIENCE/SVM/CocaCola_Sales_Rawdata.xlsx", header=0, index_col=0, parse_dates=True)
```

In [3]:
```
series
```

Out[3]:

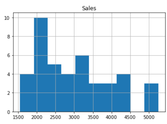
Quarter	Sales
Q1_86	1734.827000
Q2_86	2244.960999
Q3_86	2533.804993
Q4_86	2154.962997
Q1_87	1547.818996
Q2_87	2104.411995
Q3_87	2014.362999
Q4_87	1991.746998
Q1_88	1869.049999
Q2_88	2313.631996
Q3_88	2128.320000
Q4_88	2026.828999
Q1_89	1910.603996
Q2_89	2331.164993
Q3_89	2206.549995
Q4_89	2173.967995
Q1_90	2148.278000
Q2_90	2739.307999
Q3_90	2792.753998
Q4_90	2556.009995
Q1_91	2480.973999
Q2_91	3039.522995
Q3_91	3172.115997
Q4_91	2879.000999
Q1_92	2772.000000
Q2_92	3550.000000
Q3_92	3508.000000
Q4_92	3243.859993
Q1_93	3056.000000
Q2_93	3899.000000
Q3_93	3629.000000
Q4_93	3373.000000
Q1_94	3352.000000
Q2_94	4342.000000
Q3_94	4461.000000
Q4_94	4017.000000
Q1_95	3854.000000
Q2_95	4936.000000
Q3_95	4895.000000
Q4_95	4333.000000
Q1_96	4194.000000
Q2_96	5253.000000

In [4]:
```
from pandas import read_csv
from matplotlib import pyplot
series.plot()
pyplot.show()
```



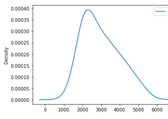
In [5]:
```
series.hist()
```

Out[5]:
```
array([[<AxesSubplot:title={'center':'Sales'}>]], dtype=object)
```



In [6]:
```
series.plot(kind='kde')
```

Out[6]:
```
<AxesSubplot:ylabel='Density'>
```



In [7]:
```
from pandas import read_csv
from sklearn.metrics import mean_squared_error
from math import sqrt
# load data
train = pd.read_excel('/Users/SAURABH/Saurabh patil/DATA SCIENCE/SVM/CocaCola_Sales_Rawdata.xlsx', header=0, index_col=0, parse_dates=True)
# prepare data
X = train.values
X = X.astype('float32')
train_size = int(len(X) * 0.50)
train, test = X[0:train_size], X[train_size:]
```

In [8]:
```
train
```

Out[8]:
```
array([[1734.827],
       [2244.961],
       [2533.805],
       [2154.963],
       [1547.819],
       [2104.412],
       [2014.363],
       [1991.747],
       [1869.05 ],
       [2313.632],
       [2128.32 ],
       [2026.829],
       [1910.604],
       [2331.165],
       [2206.55 ],
       [2173.968],
       [2148.278],
       [2739.308],
       [2792.754],
       [2556.01 ],
       [2480.974]], dtype=float32)
```

In [9]:
```
history = [x for x in train]
predictions = list()
for i in range(len(test)):
    yhat = history[-1]
    predictions.append(yhat)
# observation
    obs = test[i]
    history.append(obs)
    print('>Predicted=%.3f, Expected=%.3f' % (yhat, obs))
# report performance
rmse = sqrt(mean_squared_error(test, predictions))
print('RMSE: %.3f' % rmse)
```

```
>Predicted=2480.974, Expected=3039.523
>Predicted=3039.523, Expected=3172.116
>Predicted=3172.116, Expected=2879.001
>Predicted=2879.001, Expected=2772.000
>Predicted=2772.000, Expected=3550.000
>Predicted=3550.000, Expected=3508.000
>Predicted=3508.000, Expected=3243.860
>Predicted=3243.860, Expected=3056.000
>Predicted=3056.000, Expected=3899.000
>Predicted=3899.000, Expected=3629.000
>Predicted=3629.000, Expected=3373.000
>Predicted=3373.000, Expected=3352.000
>Predicted=3352.000, Expected=4342.000
>Predicted=4342.000, Expected=4461.000
>Predicted=4461.000, Expected=4017.000
>Predicted=4017.000, Expected=3854.000
>Predicted=3854.000, Expected=4936.000
>Predicted=4936.000, Expected=4895.000
>Predicted=4895.000, Expected=4333.000
>Predicted=4333.000, Expected=4194.000
>Predicted=4194.000, Expected=5253.000
RMSE: 527.148
```

In [11]:
```
import warnings
from pandas import read_csv
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error
from math import sqrt


# evaluate an ARIMA model for a given order (p,d,q) and return RMSE
def evaluate_arima_model(X, arima_order):
# prepare training dataset
    X = X.astype('float32')
    train_size = int(len(X) * 0.50)
    train, test = X[0:train_size], X[train_size:]
    history = [x for x in train]
# make predictions
    predictions = list()
    for t in range(len(test)):
        model = ARIMA(history, order=arima_order)
# model_fit = model.fit(disp=0)
        model_fit = model.fit(disp=0)
        yhat = model_fit.forecast()[0]
        predictions.append(yhat)
        history.append(test[t])
# calculate out of sample error
    rmse = sqrt(mean_squared_error(test, predictions))
    return rmse
```

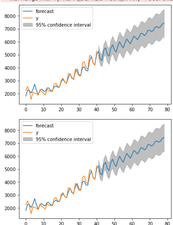
In [12]:
```
# evaluate combinations of p, d and q values for an ARIMA model
def evaluate_models(dataset, p_values, d_values, q_values):
    dataset = dataset.astype('float32')
    best_score, best_cfg = float('inf'), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                try:
                    rmse = evaluate_arima_model(train, order)
                    if rmse < best_score:
                        best_score, best_cfg = rmse, order
                        print('ARIMA%s RMSE=%.3f' % (order,rmse))
                except:
                    continue
    print('Best ARIMA%s RMSE=%.3f' % (best_cfg, best_score))
```

In [13]:
```
train = pd.read_excel('/Users/SAURABH/Saurabh patil/DATA SCIENCE/SVM/CocaCola_Sales_Rawdata.xlsx', header=0, index_col=0, parse_dates=True)
X = train.values
X = X.astype('float32')
```

In [14]:
```
# fit model
model = ARIMA(X, order=(3,1,0))
model_fit = model.fit()
forecast=model_fit.forecast(steps=10)[0]
model_fit.plot_predict(1, 79)
```

```
C:\Users\SAURABH\anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py:472: FutureWarning:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
```

Out[14]:



In [ ]: