DATA ANALYTICS
Assignment 4

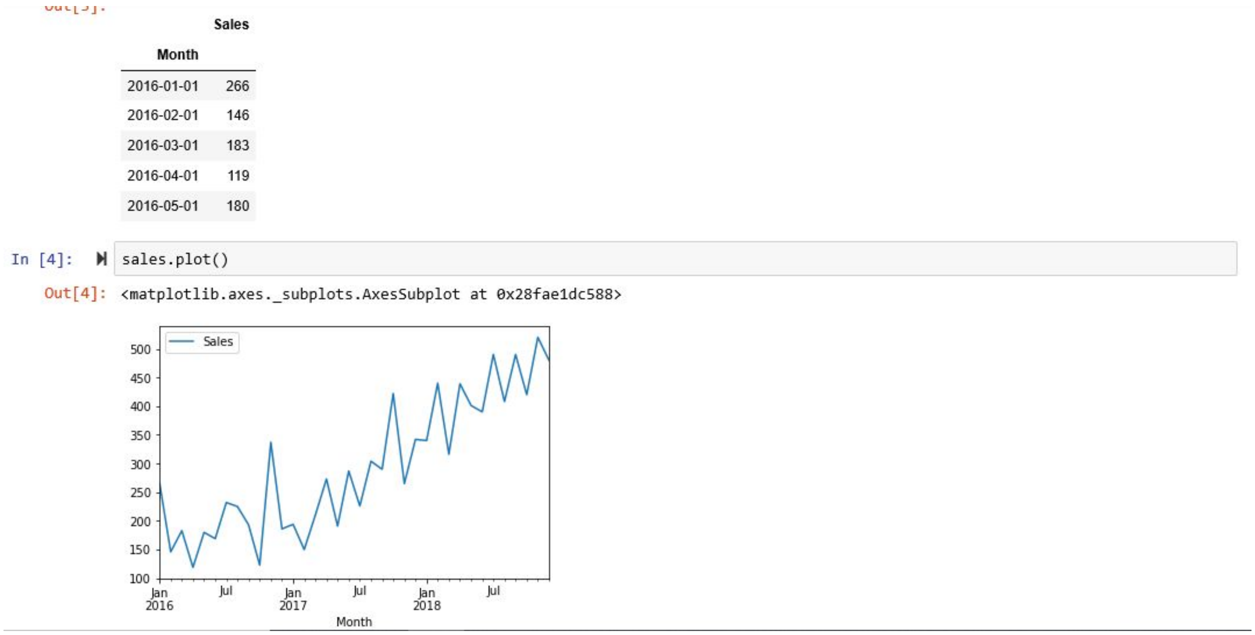
Amrutha S - USN : PES120700829

Yoshitha - USN : PES1201701744

Dhruv - USN : PES1201700122

Swathi - USN : PES1201701826

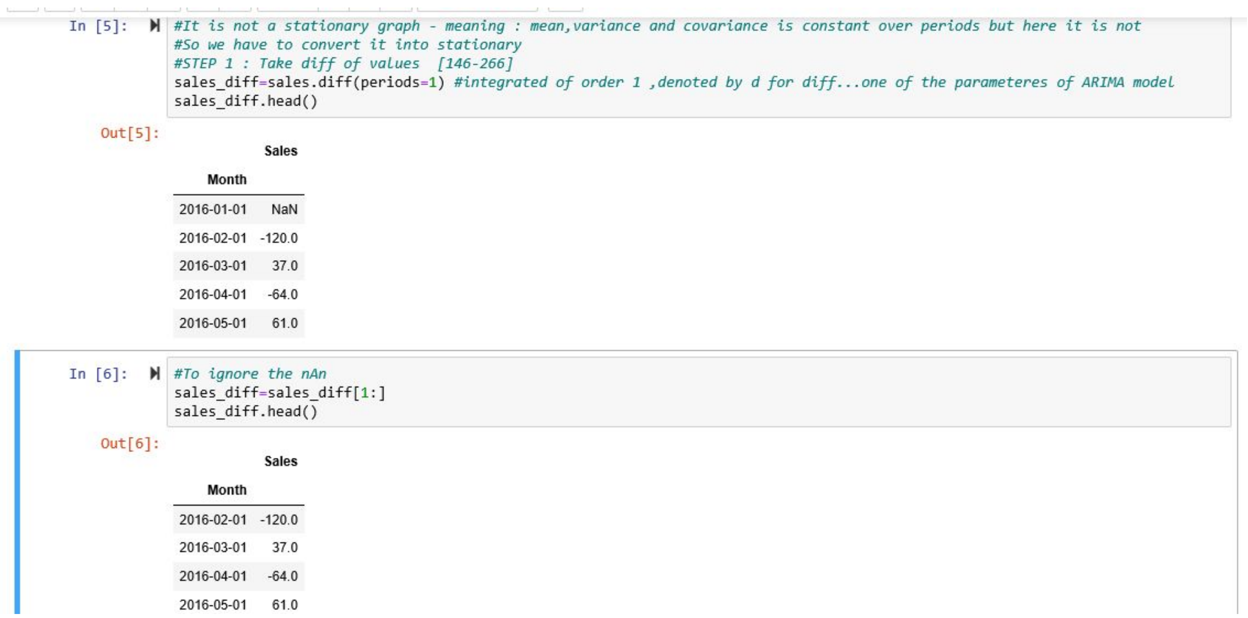
PROBLEM STATEMENT :
ARMA and ARIMA



Out[3]:

	Sales
Month	
2016-01-01	266
2016-02-01	146
2016-03-01	183
2016-04-01	119
2016-05-01	180

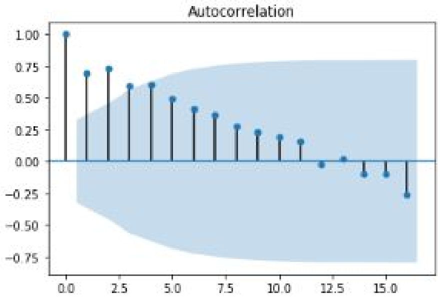
```
In [4]:  sales.plot()
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x28fae1dc588>
```

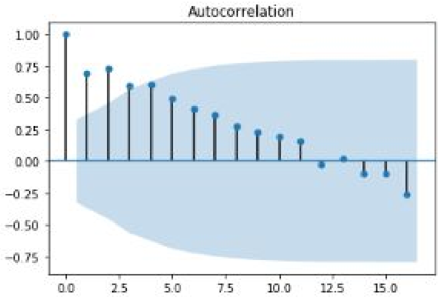


```
In [5]:  #It is not a stationary graph - meaning : mean,variance and covariance is constant over periods but here it is not
         #So we have to convert it into stationary
         #STEP 1 : Take diff of values  [146-266]
         sales_diff=sales.diff(periods=1) #integrated of order 1 ,denoted by d for diff...one of the parameteres of ARIMA model
         sales_diff.head()

Out[5]:
                  Sales
         Month
         2016-01-01   NaN
         2016-02-01 -120.0
         2016-03-01   37.0
         2016-04-01  -64.0
         2016-05-01   61.0
```

```
In [6]:  #To ignore the nAn
         sales_diff=sales_diff[1:]
         sales_diff.head()

Out[6]:
                  Sales
         Month
         2016-02-01 -120.0
         2016-03-01   37.0
         2016-04-01  -64.0
         2016-05-01   61.0
```

```
In [8]:  #Another way to check if its stationary - acf plots - auto correlation between sales & sales.shift(1)
         from statsmodels.graphics.tsaplots import plot_acf
         plot_acf(sales)
         #NOT STATIONARY

Out[8]:
```





In [22]:

```
sales_diff = sales.diff(periods=2)
# integrated of order 1, denoted by d (for diff), one of the parameter of ARIMA model
```

In [23]:

```
sales_diff = sales_diff[1:]
sales_diff.head()
```

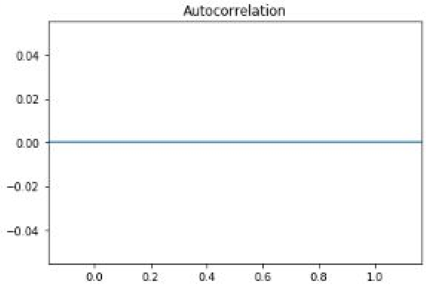
Out[23]:

	Sales
Month	
2016-02-01	NaN
2016-03-01	-83.0
2016-04-01	-27.0
2016-05-01	-3.0
2016-06-01	50.0

In [24]:

```
plot_acf(sales_diff)
#STATIONARY - Converted
```

Out[24]:



In [62]:

```
#similar to AR instead of predict we u se forcase
predictions= model_arima_fit.forecast(steps=10)[0]
predictions
```

Out[62]:

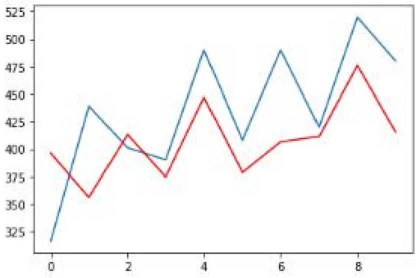
```
array([396.0576284 , 355.94957155, 413.10872345, 374.52333107,
       446.95926654, 378.64333011, 406.40593148, 411.41810008,
       475.9198063 , 415.42608995])
```

In [63]:

```
plt.plot(test)
plt.plot(predictions,color='red')
```

Out[63]:

```
[<matplotlib.lines.Line2D at 0x1682142ad88>]
```



In [64]:

```
mean_squared_error(test,predictions)
```

Out[64]:

```
2958.153543147542
```

In [38]:

```
rmse=0
total_rows=len(test)
for i in range(9):
    value=(test[i]-predictions[i])**2
    rmse=rmse+value
rmse=(rmse/total_rows)**0.5
print(rmse)
```

Out[3]:

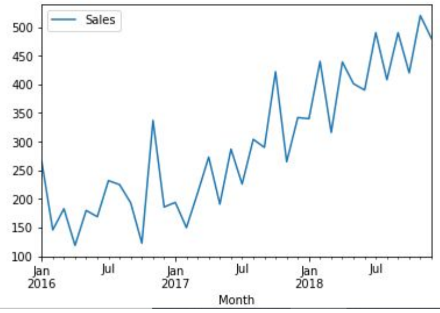
	Sales
Month	
2016-01-01	266
2016-02-01	146
2016-03-01	183
2016-04-01	119
2016-05-01	180

In [4]:

```
sales.plot()
```

Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28fae1dc588>
```



In [62]:

```
#similar to AR instead of predict we u se forcase
predictions= model_arima_fit.forecast(steps=10)[0]
predictions
```

Out[62]:

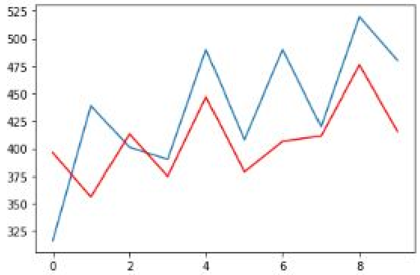
```
array([396.0576284 , 355.94957155, 413.10872345, 374.52333107,
       446.95926654, 378.64333011, 406.40593148, 411.41810008,
       475.9198063 , 415.42608995])
```

In [63]:

```
plt.plot(test)
plt.plot(predictions,color='red')
```

Out[63]:

```
[<matplotlib.lines.Line2D at 0x1682142ad88>]
```



In [64]:

```
mean_squared_error(test,predictions)
```

Out[64]:

```
2958.153543147542
```

In [38]:

```
rmse=0
total_rows=len(test)
for i in range(9):
    value=(test[i]-predictions[i])**2
    rmse=rmse+value
rmse=(rmse/total_rows)**0.5
print(rmse)
```