stationary or not convert into stationary

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:

```
df=pd.read_csv('D:/Dixant/CDAC/Machine Learning/29-12-21/Train.csv')
```

In [13]:

```
df.head(25)
```

Out[13]:

Datetime	Count
2012-08-25 00:00:00	8
2012-08-25 01:00:00	2
2012-08-25 02:00:00	6
2012-08-25 03:00:00	2
2012-08-25 04:00:00	2
2012-08-25 05:00:00	2
2012-08-25 06:00:00	2
2012-08-25 07:00:00	2
2012-08-25 08:00:00	6
2012-08-25 09:00:00	2
2012-08-25 10:00:00	2
2012-08-25 11:00:00	6
2012-08-25 12:00:00	4
2012-08-25 13:00:00	2
2012-08-25 14:00:00	6
2012-08-25 15:00:00	2
2012-08-25 16:00:00	2
2012-08-25 17:00:00	2
2012-08-25 18:00:00	2
2012-08-25 19:00:00	2
2012-08-25 20:00:00	2
2012-08-25 21:00:00	6
2012-08-25 22:00:00	2
2012-08-25 23:00:00	2
2012-08-26 00:00:00	4

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18288 entries, 0 to 18287
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   ID        18288 non-null  int64 
 1   Datetime  18288 non-null  object
 2   Count     18288 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 428.8+ KB
```

In [7]:

```
df.drop(['ID'],axis=1,inplace=True)
```

In [9]:

```
df['Datetime']=pd.to_datetime(df['Datetime'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18288 entries, 0 to 18287
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype         
---  ------    --------------  -----         
 0   Datetime  18288 non-null  datetime64[ns]
 1   Count     18288 non-null  int64         
dtypes: datetime64[ns](1), int64(1)
memory usage: 285.9 KB
```

In [10]:

```
df.set_index('Datetime',inplace=True)
df.head()
```

Out[10]:

	Count
Datetime	
2012-08-25 00:00:00	8
2012-08-25 01:00:00	2
2012-08-25 02:00:00	6
2012-08-25 03:00:00	2
2012-08-25 04:00:00	2

In [11]:

```
df.describe()
```

Out[11]:

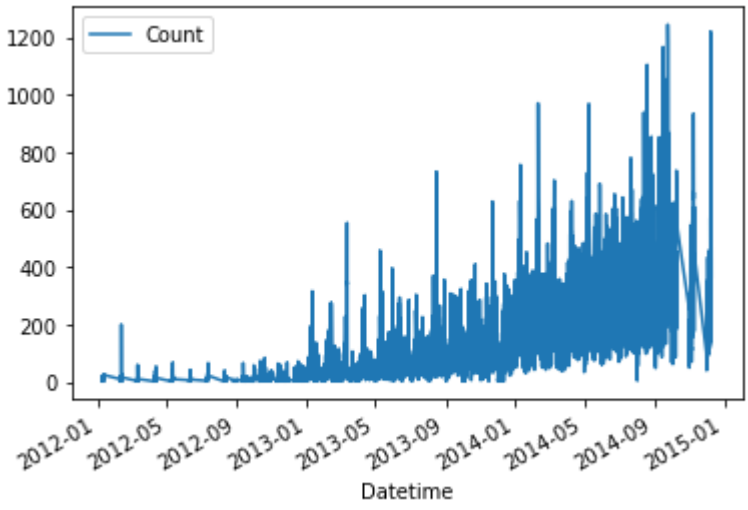
	Count
count	18288.000000
mean	138.958115
std	153.467461
min	2.000000
25%	22.000000
50%	84.000000
75%	210.000000
max	1244.000000

In [12]:

```
df.plot()
```

Out[12]:

```
<AxesSubplot:xlabel='Datetime'>
```



Moving Average

In [15]:

```
# SMA over a period of 12 and 24 hours
#min_period = min value to start calculation
df['SMA_12'] = df.Count.rolling(12, min_periods=1).mean()
df['SMA_24'] = df.Count.rolling(24, min_periods=1).mean()
df.head(25)
```

Out[15]:

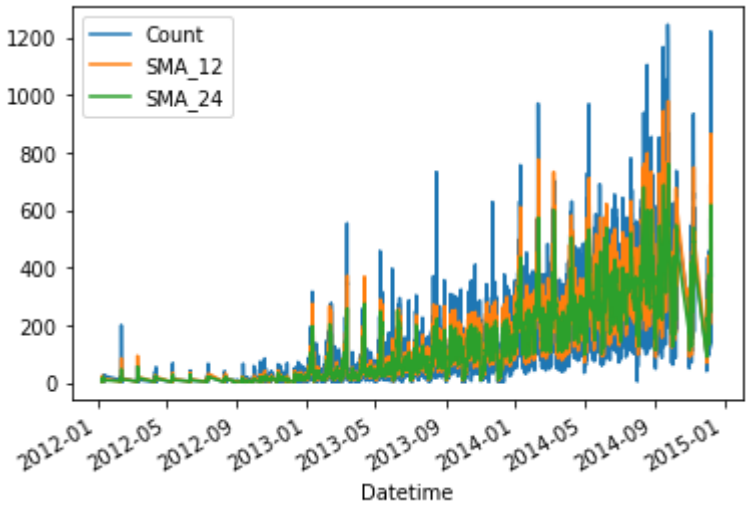
Datetime	Count	SMA_12	SMA_24
2012-08-25 00:00:00	8	8.000000	8.000000
2012-08-25 01:00:00	2	5.000000	5.000000
2012-08-25 02:00:00	6	5.333333	5.333333
2012-08-25 03:00:00	2	4.500000	4.500000
2012-08-25 04:00:00	2	4.000000	4.000000
2012-08-25 05:00:00	2	3.666667	3.666667
2012-08-25 06:00:00	2	3.428571	3.428571
2012-08-25 07:00:00	2	3.250000	3.250000
2012-08-25 08:00:00	6	3.555556	3.555556
2012-08-25 09:00:00	2	3.400000	3.400000
2012-08-25 10:00:00	2	3.272727	3.272727
2012-08-25 11:00:00	6	3.500000	3.500000
2012-08-25 12:00:00	4	3.166667	3.538462
2012-08-25 13:00:00	2	3.166667	3.428571
2012-08-25 14:00:00	6	3.166667	3.600000
2012-08-25 15:00:00	2	3.166667	3.500000
2012-08-25 16:00:00	2	3.166667	3.411765
2012-08-25 17:00:00	2	3.166667	3.333333
2012-08-25 18:00:00	2	3.166667	3.263158
2012-08-25 19:00:00	2	3.166667	3.200000
2012-08-25 20:00:00	2	2.833333	3.142857
2012-08-25 21:00:00	6	3.166667	3.272727
2012-08-25 22:00:00	2	3.166667	3.217391
2012-08-25 23:00:00	2	2.833333	3.166667
2012-08-26 00:00:00	4	2.833333	3.000000

In [16]:

```
df.plot()
```

Out[16]:

```
<AxesSubplot:xlabel='Datetime'>
```



In [17]:

```
df['CMA']= df.Count.expanding(min_periods=1).mean() # cummulative moving average the cumul
df.head(25)
```

Out[17]:

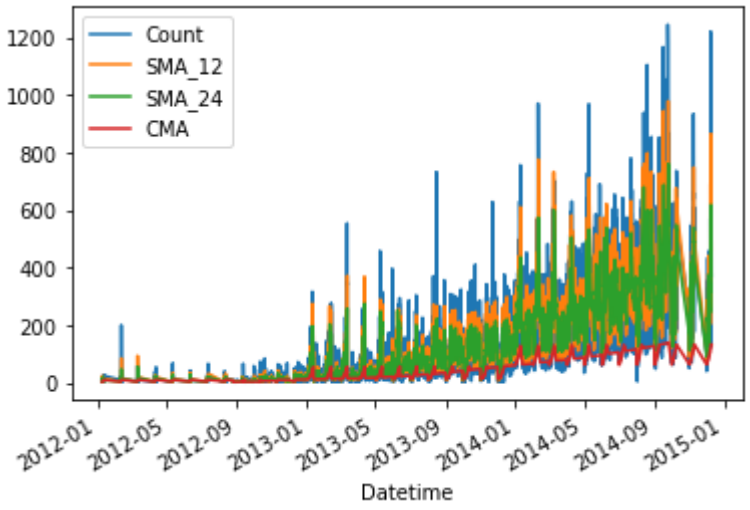
Datetime	Count	SMA_12	SMA_24	CMA
2012-08-25 00:00:00	8	8.000000	8.000000	8.000000
2012-08-25 01:00:00	2	5.000000	5.000000	5.000000
2012-08-25 02:00:00	6	5.333333	5.333333	5.333333
2012-08-25 03:00:00	2	4.500000	4.500000	4.500000
2012-08-25 04:00:00	2	4.000000	4.000000	4.000000
2012-08-25 05:00:00	2	3.666667	3.666667	3.666667
2012-08-25 06:00:00	2	3.428571	3.428571	3.428571
2012-08-25 07:00:00	2	3.250000	3.250000	3.250000
2012-08-25 08:00:00	6	3.555556	3.555556	3.555556
2012-08-25 09:00:00	2	3.400000	3.400000	3.400000
2012-08-25 10:00:00	2	3.272727	3.272727	3.272727
2012-08-25 11:00:00	6	3.500000	3.500000	3.500000
2012-08-25 12:00:00	4	3.166667	3.538462	3.538462
2012-08-25 13:00:00	2	3.166667	3.428571	3.428571
2012-08-25 14:00:00	6	3.166667	3.600000	3.600000
2012-08-25 15:00:00	2	3.166667	3.500000	3.500000
2012-08-25 16:00:00	2	3.166667	3.411765	3.411765
2012-08-25 17:00:00	2	3.166667	3.333333	3.333333
2012-08-25 18:00:00	2	3.166667	3.263158	3.263158
2012-08-25 19:00:00	2	3.166667	3.200000	3.200000
2012-08-25 20:00:00	2	2.833333	3.142857	3.142857
2012-08-25 21:00:00	6	3.166667	3.272727	3.272727
2012-08-25 22:00:00	2	3.166667	3.217391	3.217391
2012-08-25 23:00:00	2	2.833333	3.166667	3.166667
2012-08-26 00:00:00	4	2.833333	3.000000	3.200000

In [18]:

```
df.plot()
```

Out[18]:

```
<AxesSubplot:xlabel='Datetime'>
```



In [19]:

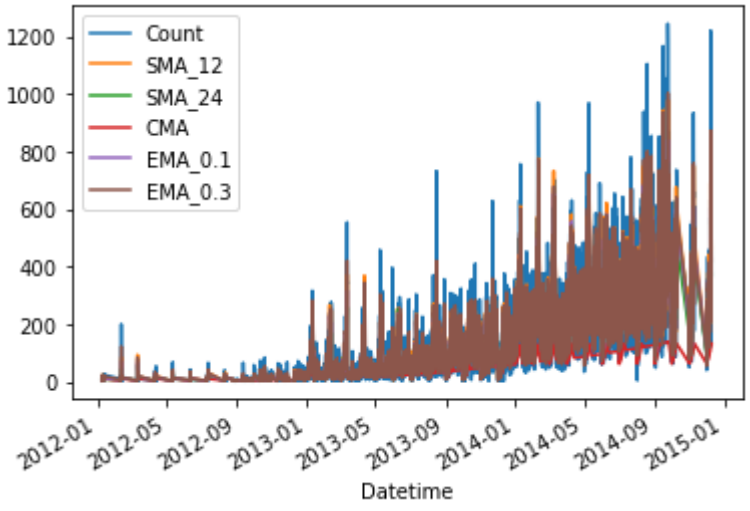
```
df['EMA_0.1'] = df.Count.ewm(alpha=0.1, adjust=False).mean()
df['EMA_0.3'] = df.Count.ewm(alpha=0.3, adjust=False).mean()
```

In [20]:

```
df.plot()
```

Out[20]:

```
<AxesSubplot:xlabel='Datetime'>
```

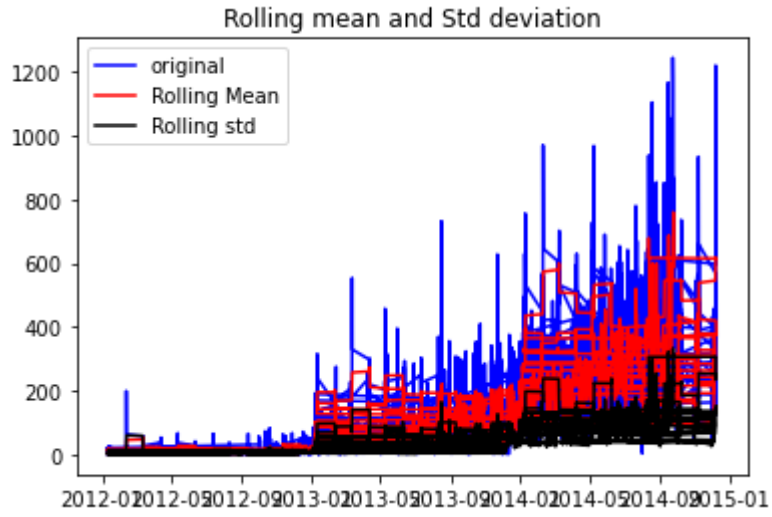


In [21]:

```
rolmean = df.Count.rolling(window=24).mean()
rolstd = df.Count.rolling(window=24).std()
```

In [22]:

```
orig = plt.plot(df.Count,color='blue',label='original')
mean = plt.plot(rolmean,color='red',label='Rolling Mean')
std = plt.plot(rolstd,color='black',label='Rolling std')
plt.legend()
plt.title('Rolling mean and Std deviation')
plt.show()
```



In [23]:

```
from statsmodels.tsa.stattools import adfuller
```

In [24]:

```
test_result=adfuller(df['Count'])
test_result
```

Out[24]:

```
(-4.456560536856799,
 0.00023540466467667786,
 45,
 18242,
 {'1%': -3.430708525404171,
  '5%': -2.861698454786869,
  '10%': -2.5668543412994906},
 181026.8337109476)
```

In [27]:

```
def adfuller_test(sales):
    result=adfuller(sales)
    #print(result)
    labels = ['ADF Test Statistic','p-value','Lags Used','Number of Observations Used']
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )
    if result[1] <= 0.05:
        print("strong evidence against the null hypothesis(Ho), reject the null hypothesis.
    else:
        print("weak evidence against null hypothesis, indicating it is non-stationary ")
```

In [28]:

```
adfuller_test(df['Count'])
```

```
ADF Test Statistic : -4.456560536856799
p-value : 0.00023540466467667786
Lags Used : 45
Number of Observations Used : 18242
strong evidence against the null hypothesis(Ho), reject the null hypothesis.
```

In [ ]:

```
```