In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima_model import ARIMA
import datetime
import itertools
import warnings
from sklearn.metrics import mean_squared_error
import seaborn as sns
import statsmodels
plt.style.use('fivethirtyeight')
%matplotlib inline
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [3]:
```
df = pd.read_csv("/content/drive/MyDrive/Ineuron/data_stocks.csv")
df.head()
```

Out[3]:

	DATE	SP500	NASDAQ.AAL	NASDAQ.AAPL	NASDAQ.ADBE	NASDAQ.ADI	NASDAQ.A
0	1491226200	2363.6101	42.3300	143.6800	129.6300	82.040	102.2
1	1491226260	2364.1001	42.3600	143.7000	130.3200	82.080	102.1
2	1491226320	2362.6799	42.3100	143.6901	130.2250	82.030	102.2
3	1491226380	2364.3101	42.3700	143.6400	130.0729	82.000	102.1
4	1491226440	2364.8501	42.5378	143.6600	129.8800	82.035	102.0

5 rows × 502 columns

In [4]:
```
df["DATE"].dtypes
```

Out[4]:
```
dtype('int64')
```

In [5]:
```
df['DATE'] = pd.to_datetime(df['DATE'], unit='s')
```

In [6]:
```
df['DATE'].tail()
```

Out[6]:
```
41261   2017-08-31 19:56:00
41262   2017-08-31 19:57:00
41263   2017-08-31 19:58:00
41264   2017-08-31 19:59:00
41265   2017-08-31 20:00:00
Name: DATE, dtype: datetime64[ns]
```

In [7]:
```
df.index = df['DATE']
```

In [8]:
```
df.drop('DATE',axis = 1,inplace=True)
```

In [9]:
```
df.tail()
```

Out[9]:

DATE	SP500	NASDAQ.AAL	NASDAQ.AAPL	NASDAQ.ADBE	NASDAQ.ADI	NASDAQ.ADP	NAS
2017-08-31 19:56:00	2472.22	44.72	164.11	155.090	83.67	106.565	
2017-08-31 19:57:00	2471.77	44.73	164.12	155.160	83.65	106.590	
2017-08-31 19:58:00	2470.03	44.74	164.01	155.065	83.62	106.520	
2017-08-31 19:59:00	2471.49	44.71	163.88	154.960	83.58	106.400	
2017-08-31 20:00:00	2471.49	44.74	163.98	155.160	83.69	106.470	

5 rows × 501 columns

NASDAQ.AAPL

In [66]:
```
df_AAPL = df[["NASDAQ.AAPL"]].copy()
df_AAPL.tail()
```

Out[66]:

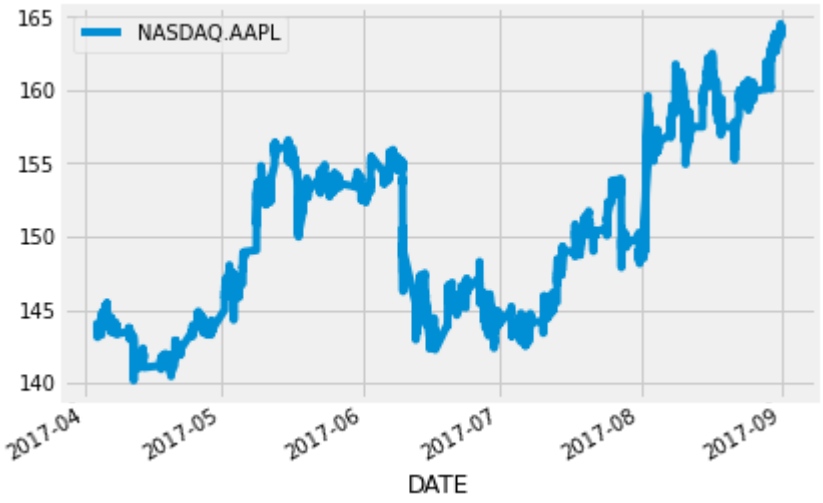
DATE	NASDAQ.AAPL
2017-08-31 19:56:00	164.11
2017-08-31 19:57:00	164.12
2017-08-31 19:58:00	164.01
2017-08-31 19:59:00	163.88
2017-08-31 20:00:00	163.98

In [67]:
```
df_AAPL.count()
```

Out[67]:
```
NASDAQ.AAPL    41266
dtype: int64
```

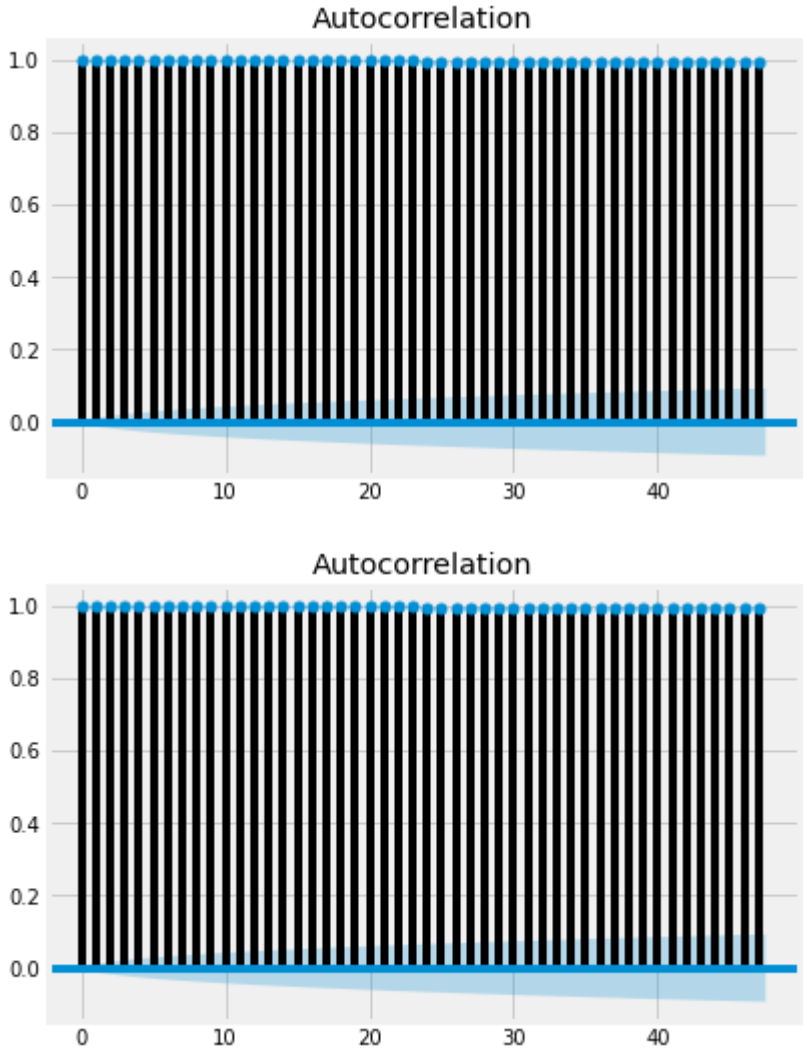
In [68]:
```
df_AAPL.plot()
```

Out[68]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f98410597f0>
```



In [69]:
```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(df_AAPL)
```

Out[69]:



In [70]:
```
##Converting series to stationary
df_AAPL.shift(1)
```

Out[70]:

DATE	NASDAQ.AAPL
2017-04-03 13:30:00	NaN
2017-04-03 13:31:00	143.6800
2017-04-03 13:32:00	143.7000
2017-04-03 13:33:00	143.6901
2017-04-03 13:34:00	143.6400
...	...
2017-08-31 19:56:00	164.1400
2017-08-31 19:57:00	164.1100
2017-08-31 19:58:00	164.1200
2017-08-31 19:59:00	164.0100
2017-08-31 20:00:00	163.8800

41266 rows × 1 columns

In [74]:
```
X0 = df_AAPL.values
train0 = X0[0:28886] # 27 data as train data
test0 = X0[28886:] # 9 data as test data
print(train0.size)
print(test0.size)
predictions0 = []
```

```
28886
12380
```

In [75]:
```
p0=d0=q0=range(0,2)
pdq0=list(itertools.product(p0,d0,q0))

warnings.filterwarnings('ignore')
for param in pdq0:
    try:
        model_arima0 = ARIMA(train0, order=param)
        model_arima_fit0 = model_arima0.fit()
        print(param,model_arima_fit0.aic)
    except:
        continue
```

```
(0, 0, 0) 170326.9720446082
(0, 0, 1) 131018.03599865251
(0, 1, 0) -64708.712006361384
(0, 1, 1) -64706.8773153409
(1, 0, 0) -64703.106010175194
(1, 0, 1) -64701.25996864913
(1, 1, 0) -64706.87029558887
(1, 1, 1) -64714.63157446154
```

In [77]:
```
from statsmodels.tsa.arima_model import ARIMA
model_arima0 = ARIMA(train0, order=(4,1,4))
model_arima_fit0 = model_arima0.fit()
```

In [78]:
```
#p0,d0,q0
#p0 -> Periods taken for auto regressive model
#d0 -> Integrated order, difference
#q0 -> Periods in moving average model
from statsmodels.tsa.arima_model import ARIMA
model_arima0 = ARIMA(train0, order=(3,1,3))
model_arima_fit0 = model_arima0.fit()
print(model_arima_fit0.aic)
```

```
-64730.23498583691
```

In [80]:
```
predictions0 = model_arima_fit0.forecast(steps=12380)[0]
predictions0
```

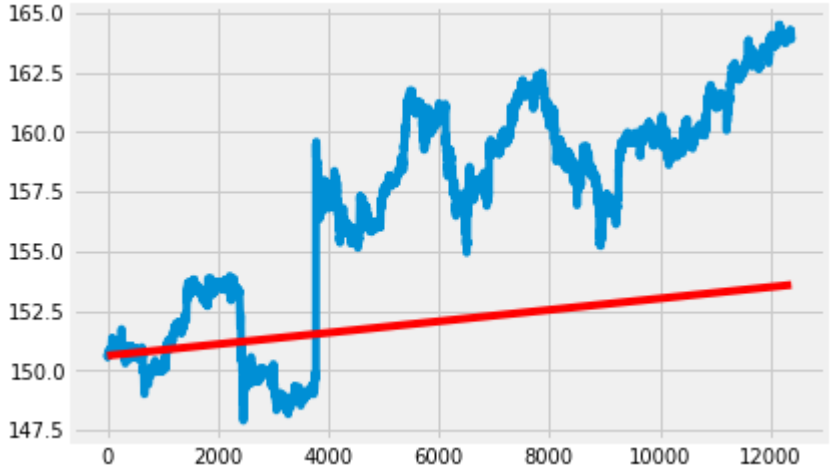
Out[80]:
```
array([150.61066006, 150.61168229, 150.61211528, ..., 153.58143875,
       153.58167866, 153.58191858])
```

In [81]:
```
res0 = round(mean_squared_error(test0,predictions0))
res0
```

Out[81]:
```
38
```

In [82]:
```
plt.plot(test0)
plt.plot(predictions0, color='red')
```

Out[82]:
```
[<matplotlib.lines.Line2D at 0x7f9840e3c438>]
```



NASDAQ.ADP

In [ ]:
```
df_ADP = df[['NASDAQ.ADP']].copy()
df_ADP.tail()
```

Out[10]:

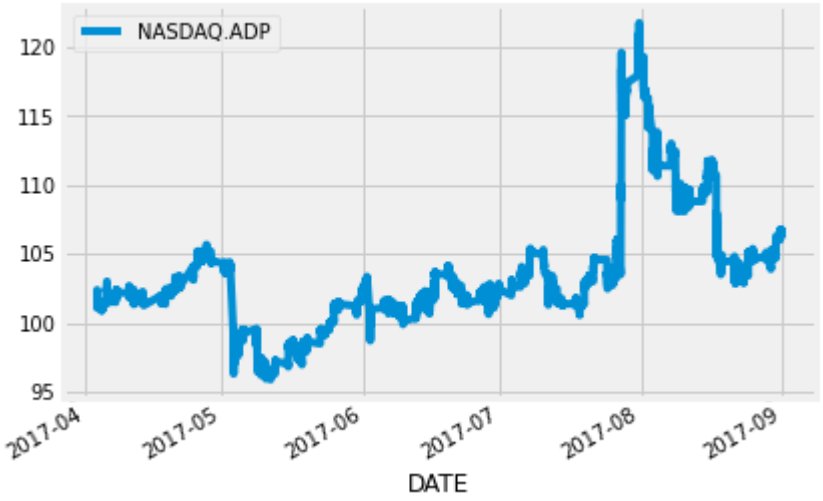
DATE	NASDAQ.ADP
2017-08-31 19:56:00	106.565
2017-08-31 19:57:00	106.590
2017-08-31 19:58:00	106.520
2017-08-31 19:59:00	106.400
2017-08-31 20:00:00	106.470

In [ ]:
```
df_ADP.count()
```

Out[11]:
```
NASDAQ.ADP    41266
dtype: int64
```

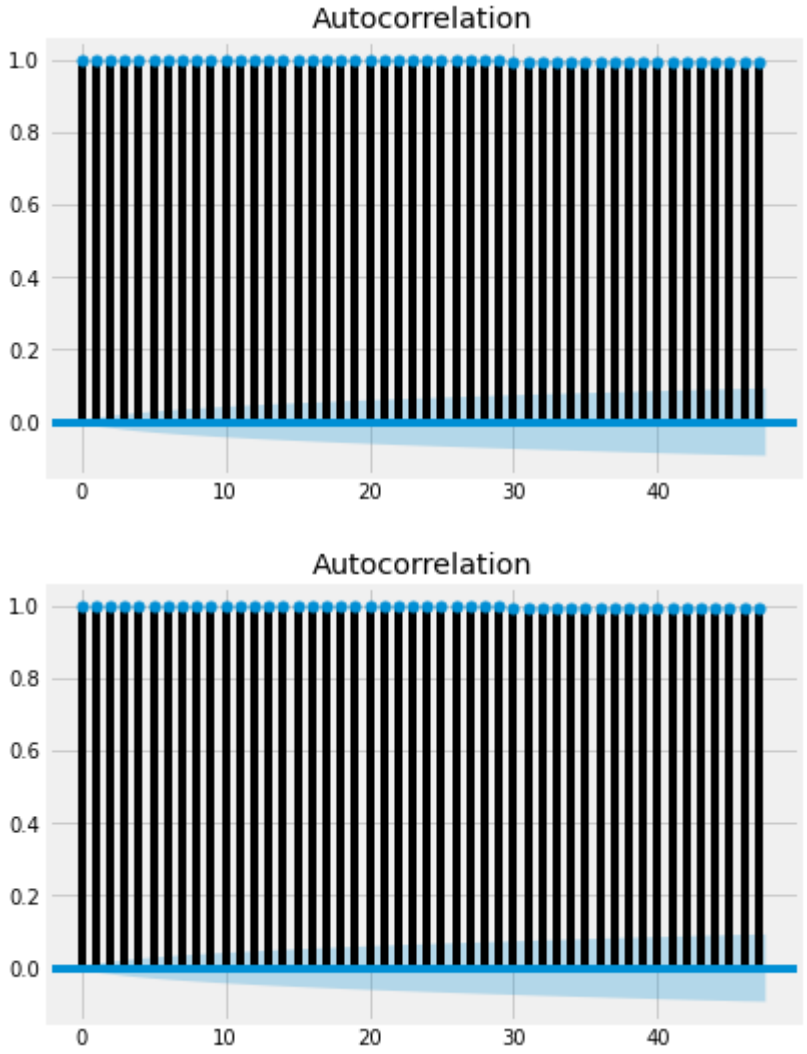
In [ ]:
```
df_ADP.plot()
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f96167444a8>
```



In [ ]:
```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(df_ADP)
```

Out[13]:



In [ ]:
```
##Converting series to stationary
df_ADP.shift(1)
```

Out[14]:

DATE	NASDAQ.ADP
2017-04-03 13:30:00	NaN
2017-04-03 13:31:00	102.2300
2017-04-03 13:32:00	102.1400
2017-04-03 13:33:00	102.2125
2017-04-03 13:34:00	102.1400
...	...
2017-08-31 19:56:00	106.6300
2017-08-31 19:57:00	106.5650
2017-08-31 19:58:00	106.5900
2017-08-31 19:59:00	106.5200
2017-08-31 20:00:00	106.4000

41266 rows × 1 columns

In [ ]:
```
X = df_ADP.values
train = X[0:28886] # 27 data as train data
test = X[28886:] # 9 data as test data
print(train.size)
print(test.size)
predictions = []
```

```
28886
12380
```

In [ ]:
```
p=d=q=range(0,2)
pdq=list(itertools.product(p,d,q))

warnings.filterwarnings('ignore')
for param in pdq:
    try:
        model_arima = ARIMA(train, order=param)
        model_arima_fit = model_arima.fit()
        print(param,model_arima_fit.aic)
    except:
        continue
```

```
(0, 0, 0) 124317.93290534396
(0, 0, 1) 85271.48908067068
(0, 1, 0) -80762.52187440016
(0, 1, 1) -81075.63405539667
(1, 0, 0) -80762.97956376115
(1, 0, 1) -81077.32276388479
(1, 1, 0) -81067.89180999548
(1, 1, 1) -81073.99649148404
```

In [ ]:
```
from statsmodels.tsa.arima_model import ARIMA
model_arima = ARIMA(train, order=(2,1,2))
model_arima_fit = model_arima.fit()
```

In [ ]:
```
#p,d,q
#p -> Periods taken for auto regressive model
#d -> Integrated order, difference
#q -> Periods in moving average model
from statsmodels.tsa.arima_model import ARIMA
model_arima = ARIMA(train, order=(3,1,3))
model_arima_fit = model_arima.fit()
print(model_arima_fit.aic)
```

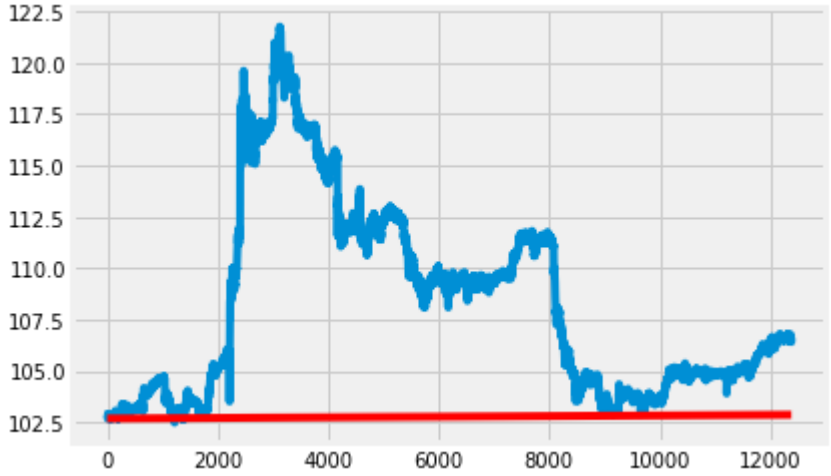
```
-81073.99649148404
```

In [ ]:
```
predictions = model_arima_fit.forecast(steps=12380)[0]
predictions
```

Out[23]:
```
array([102.67200781, 102.67196666, 102.67198318, ..., 102.85692098,
       102.85693592, 102.85695086])
```

In [ ]:
```
plt.plot(test)
plt.plot(predictions, color='red')
```

Out[24]: [<matplotlib.lines.Line2D at 0x7f960d6d5fd0>]



In [ ]:
```
res =round(mean_squared_error(test,predictions),2)
```

In [ ]:
```
res
```

Out[29]: 52.8

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

NADBAQ.CBOE

In [42]:
```
df_CBOE = df[["NASDAQ.CBOE"]].copy()
df_CBOE.tail()
```

Out[42]:

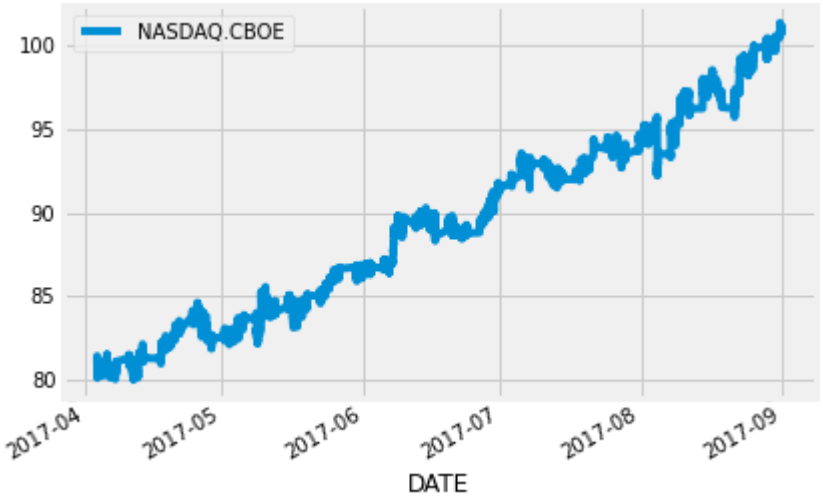
DATE	NASDAQ.CBOE
2017-08-31 19:56:00	100.89
2017-08-31 19:57:00	100.88
2017-08-31 19:58:00	100.86
2017-08-31 19:59:00	100.83
2017-08-31 20:00:00	100.89

In [43]:
```
df_CBOE.count()
```

Out[43]:
```
NASDAQ.CBOE    41266
dtype: int64
```

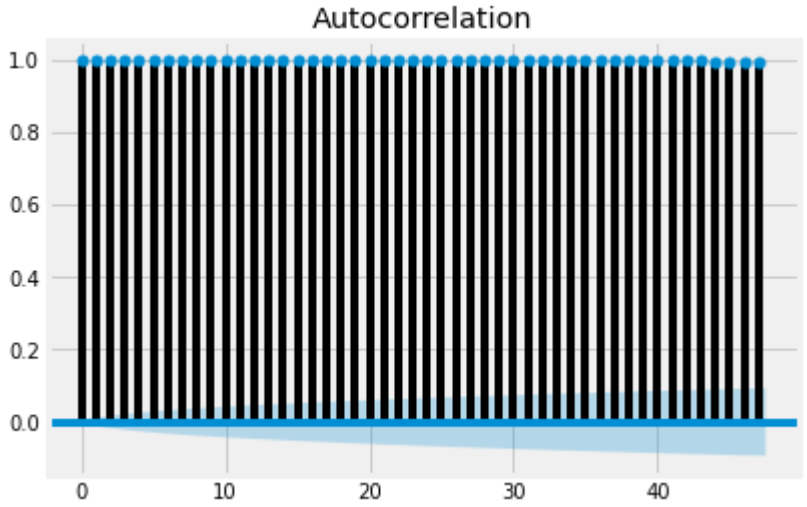
In [44]:
```
df_CBOE.plot()
```

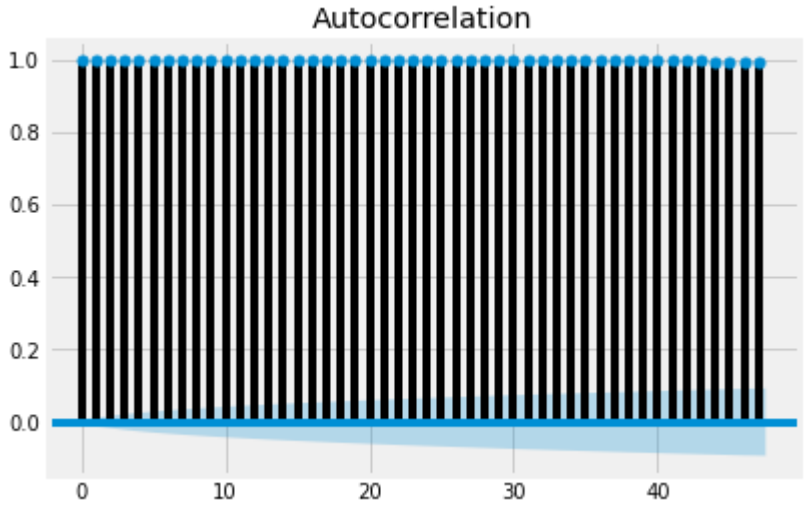
Out[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f98416fcf60>
```



In [45]:
```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(df_CBOE)
```

Out[45]:





In [46]:
```
##Converting series to stationary
df_CBOE.shift(1)
```

Out[46]:

DATE	NASDAQ.CBOE
2017-04-03 13:30:00	NaN
2017-04-03 13:31:00	81.0300
2017-04-03 13:32:00	81.2100
2017-04-03 13:33:00	81.2100
2017-04-03 13:34:00	81.1300
...	...
2017-08-31 19:56:00	100.8899
2017-08-31 19:57:00	100.8900
2017-08-31 19:58:00	100.8800
2017-08-31 19:59:00	100.8600
2017-08-31 20:00:00	100.8300

41266 rows × 1 columns

In [47]:
```
X3 = df_CBOE.values
train3 = X3[0:28886] # 27 data as train data
test3 = X3[28886:] # 9 data as test data
print(train3.size)
print(test3.size)
predictions3 = []
```

```
28886
12380
```

In [48]:
```
p3=d3=q3=range(0,2)
pdq3=list(itertools.product(p3,d3,q3))

warnings.filterwarnings('ignore')
for param in pdq3:
    try:
        model_arima3 = ARIMA(train3, order=param)
        model_arima_fit3 = model_arima3.fit()
        print(param,model_arima_fit3.aic)
    except:
        continue
```

```
(0, 0, 0) 160441.4526311847
(0, 0, 1) 120929.59019310356
(0, 1, 0) -96706.67408377743
(0, 1, 1) -96892.8965002238
(1, 0, 0) -96696.92223399912
(1, 0, 1) -96882.56874919325
(1, 1, 0) -96878.58101700693
(1, 1, 1) -97008.39440809148
```

In [49]:
```
from statsmodels.tsa.arima_model import ARIMA
model_arima3 = ARIMA(train3, order=(3,1,3))
model_arima_fit3 = model_arima3.fit()
```

In [50]:
```
predictions3 = model_arima_fit3.forecast(steps=12380)[0]
predictions3
```

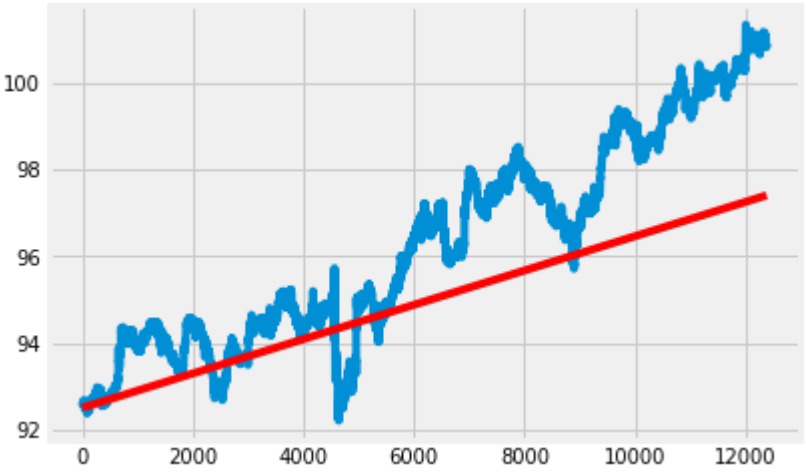
Out[50]:
```
array([92.5017572 , 92.50262953, 92.5037775 , ..., 97.40717353,
       97.40756968, 97.40796582])
```

In [51]:
```
res3 = round(mean_squared_error(test3,predictions3))
res3
```

Out[51]:
```
3
```

In [52]:
```
plt.plot(test3)
plt.plot(predictions3, color='red')
```

Out[52]: [<matplotlib.lines.Line2D at 0x7f98416f6748>]



In [52]:

In [52]:

In [52]:

#NASDAQ.CSCO

In [53]:
```
df_CSCO = df[["NASDAQ.CSCO"]].copy()
df_CSCO.tail()
```

Out[53]:

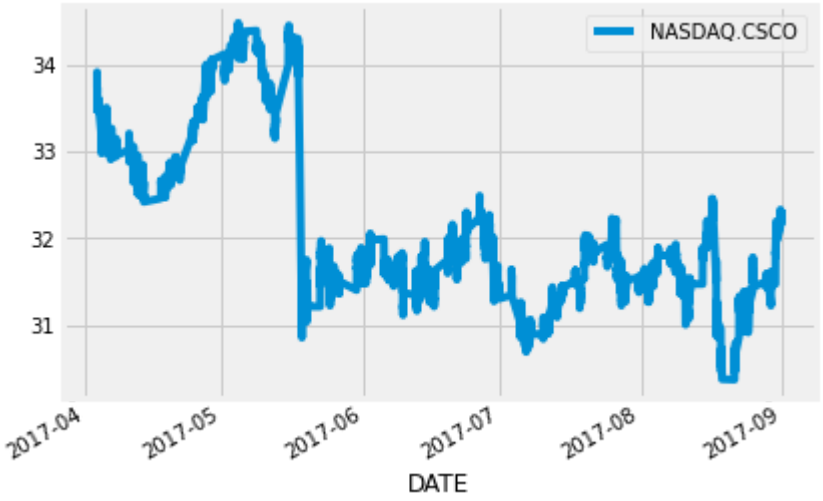
DATE	NASDAQ.CSCO
2017-08-31 19:56:00	32.185
2017-08-31 19:57:00	32.200
2017-08-31 19:58:00	32.200
2017-08-31 19:59:00	32.195
2017-08-31 20:00:00	32.225

In [54]:
```
df_CSCO.count()
```

Out[54]:
```
NASDAQ.CSCO    41266
dtype: int64
```

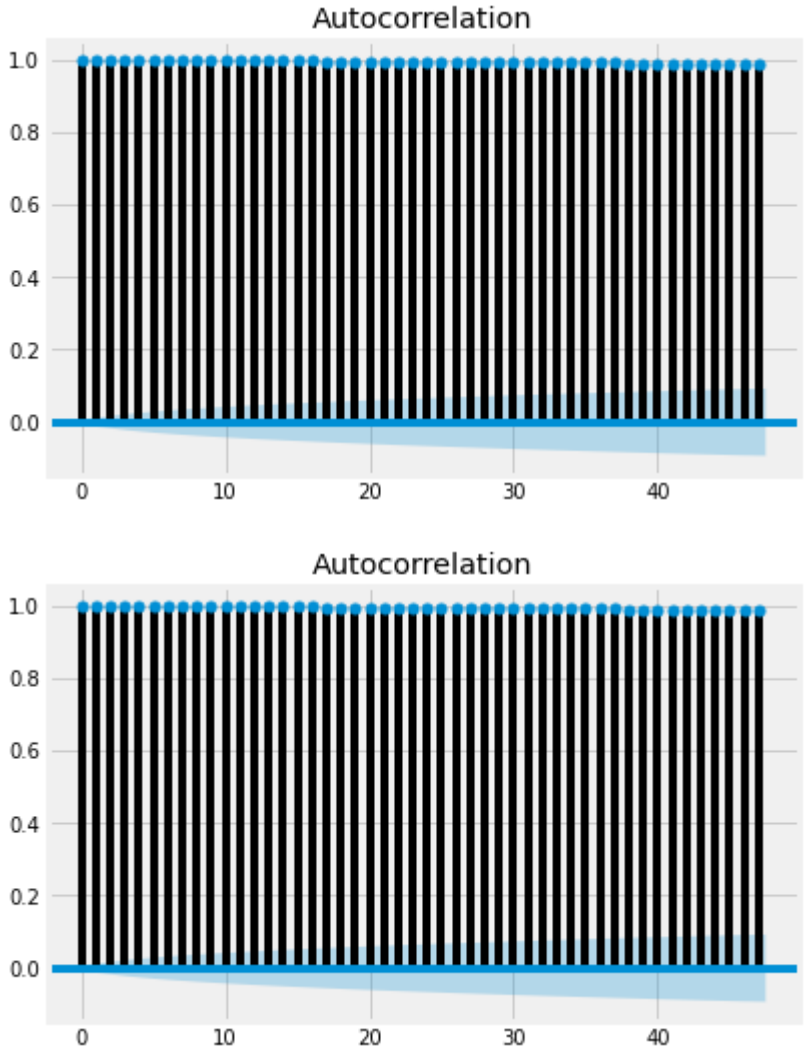
In [55]:
```
df_CSCO.plot()
```

Out[55]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f98413a5978>
```



In [56]:
```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(df_CSCO)
```

Out[56]:



In [57]:
```
##Converting series to stationary
df_CSCO.shift(1)
```

Out[57]:

DATE	NASDAQ.CSCO
2017-04-03 13:30:00	NaN
2017-04-03 13:31:00	33.7400
2017-04-03 13:32:00	33.8800
2017-04-03 13:33:00	33.9000
2017-04-03 13:34:00	33.8499
...	...
2017-08-31 19:56:00	32.1700
2017-08-31 19:57:00	32.1850
2017-08-31 19:58:00	32.2000
2017-08-31 19:59:00	32.2000
2017-08-31 20:00:00	32.1950

41266 rows × 1 columns

In [59]:
```
X4 = df_CSCO.values
train4 = X4[0:28886] # 27 data as train data
test4 = X4[28886:] # 9 data as test data
print(train4.size)
print(test4.size)
predictions4 = []
```

```
28886
12380
```

In [60]:
```
p4=d4=q4=range(0,2)
pdq4=list(itertools.product(p4,d4,q4))

warnings.filterwarnings('ignore')
for param in pdq4:
    try:
        model_arima4 = ARIMA(train4, order=param)
        model_arima_fit4 = model_arima4.fit()
        print(param,model_arima_fit4.aic)
    except:
        continue
```

```
(0, 0, 0) 85353.81423993816
(0, 0, 1) 46052.34616741124
(0, 1, 0) -135890.52761203377
(0, 1, 1) -135922.78479698166
(1, 0, 0) -135888.85266493712
(1, 0, 1) -135921.37291295695
(1, 1, 0) -135920.50823736849
(1, 1, 1) -136015.15611626126
```

In [61]:
```
from statsmodels.tsa.arima_model import ARIMA
model_arima4 = ARIMA(train4, order=(4,1,4))
model_arima_fit4 = model_arima4.fit()
```

In [62]:
```
predictions4 = model_arima_fit4.forecast(steps=12380)[0]
predictions4
```

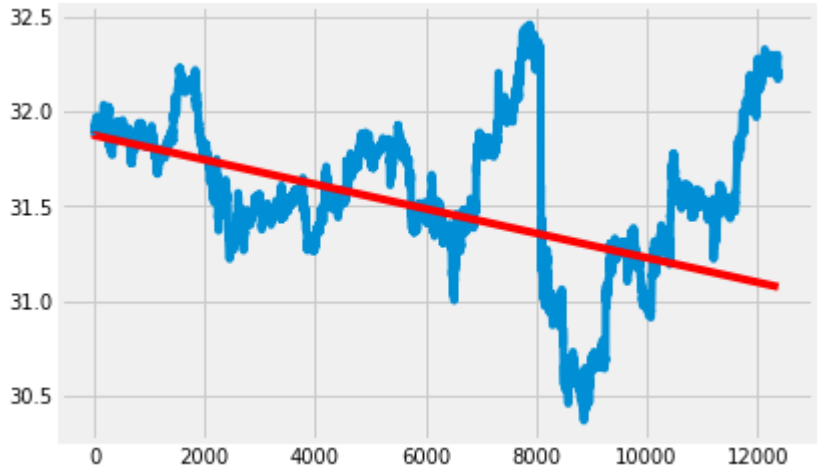
Out[62]:
```
array([31.87458762, 31.87343597, 31.87461045, ..., 31.07262991,
       31.07256513, 31.07250035])
```

In [64]:
```
res4 = round(mean_squared_error(test4,predictions4))
res4
```

Out[64]:
```
0
```

In [63]:
```
plt.plot(test4)
plt.plot(predictions4, color='red')
```

Out[63]: [<matplotlib.lines.Line2D at 0x7f98410d01d0>]



In [ ]:

In [ ]:

NASDAQ.EBAY

In [10]:
```
df_EBAY = df[["NASDAQ.EBAY"]].copy()
```

In [12]:
```
df_EBAY.tail()
```

Out[12]:

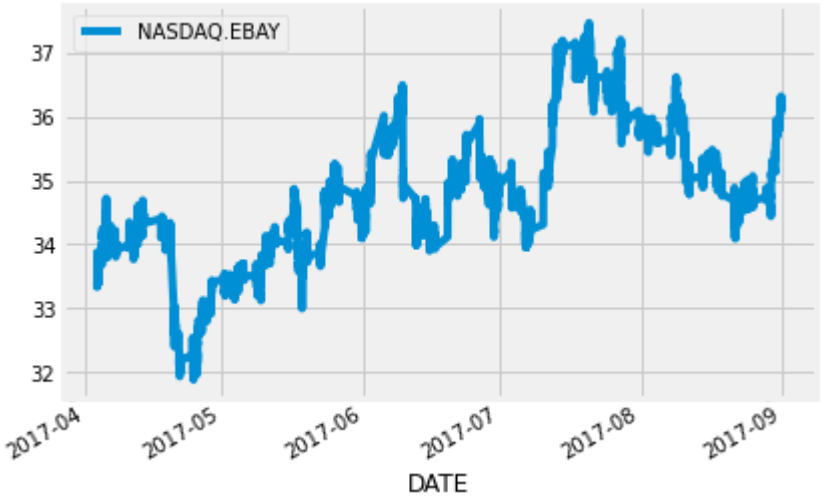
DATE	NASDAQ.EBAY
2017-08-31 19:56:00	36.135
2017-08-31 19:57:00	36.130
2017-08-31 19:58:00	36.130
2017-08-31 19:59:00	36.120
2017-08-31 20:00:00	36.130

In [13]:
```
df_EBAY.count()
```

Out[13]:
```
NASDAQ.EBAY    41266
dtype: int64
```

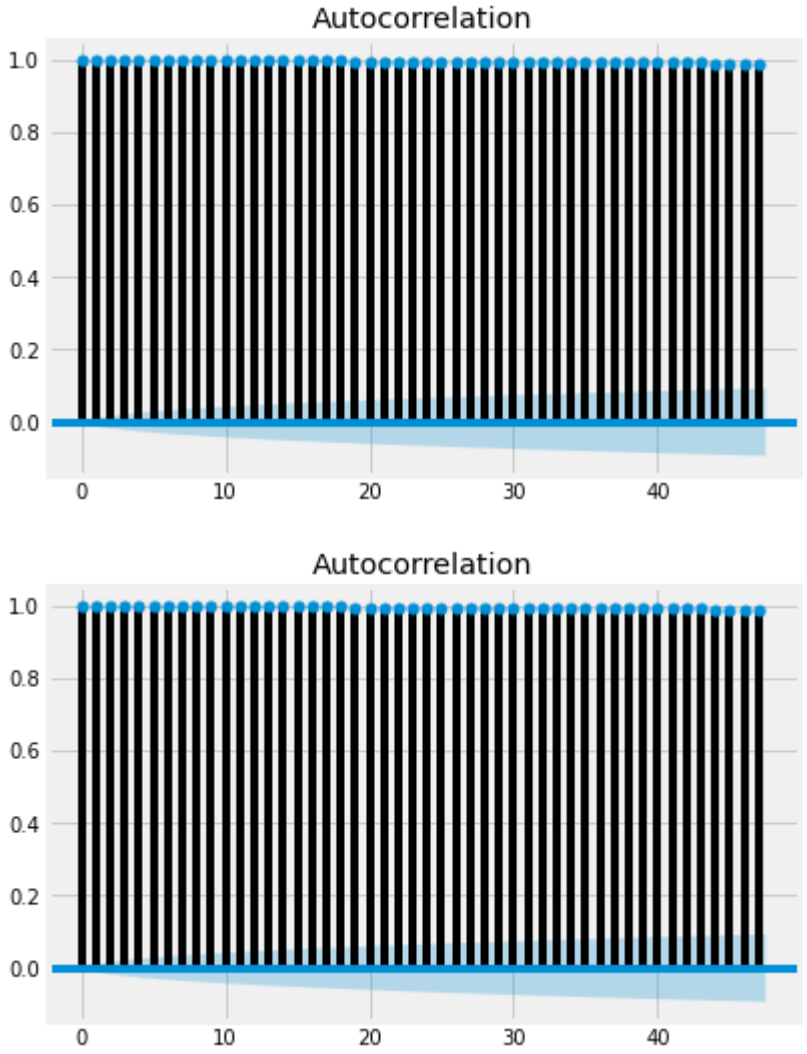
In [14]:
```
df_EBAY.plot()
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f984aa24e48>
```



In [15]:
```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(df_EBAY)
```

Out[15]:



In [16]:
```
##Converting series to stationary
df_EBAY.shift(1)
```

Out[16]:

DATE	NASDAQ.EBAY
2017-04-03 13:30:00	NaN
2017-04-03 13:31:00	33.3975
2017-04-03 13:32:00	33.3950
2017-04-03 13:33:00	33.4100
2017-04-03 13:34:00	33.3350
...	...
2017-08-31 19:56:00	36.1300
2017-08-31 19:57:00	36.1350
2017-08-31 19:58:00	36.1300
2017-08-31 19:59:00	36.1300
2017-08-31 20:00:00	36.1200

41266 rows × 1 columns

In [18]:
```
X5 = df_EBAY.values
train5 = X5[0:28886] # 27 data as train data
test5 = X5[28886:] # 9 data as test data
print(train5.size)
print(test5.size)
predictions5 = []
```

```
28886
12380
```

In [19]:
```
p5=d5=q5=range(0,2)
pdq5=list(itertools.product(p5,d5,q5))

warnings.filterwarnings('ignore')
for param in pdq5:
    try:
        model_arima5 = ARIMA(train5, order=param)
        model_arima_fit5 = model_arima5.fit()
        print(param,model_arima_fit5.aic)
    except:
        continue
```

```
(0, 0, 0) 83955.30612486275
(0, 0, 1) 44870.471253968884
(0, 1, 0) -135861.67620322717
(0, 1, 1) -135872.0650324924
(1, 0, 0) -135857.4009523872
(1, 0, 1) -135867.62575643833
(1, 1, 0) -135872.54671056976
(1, 1, 1) -135876.14912516676
```

In [21]:
```
from statsmodels.tsa.arima_model import ARIMA
model_arima5 = ARIMA(train5, order=(4,1,4))
model_arima_fit5 = model_arima5.fit()
```

In [22]:
```
predictions5 = model_arima_fit5.forecast(steps=12380)[0]
predictions5
```

Out[22]:
```
array([37.01556121, 37.01587908, 37.01520994, ..., 38.55472764,
       38.55485247, 38.5549773 ])
```

In [25]:
```
res5 = round(mean_squared_error(test5,predictions5))
res5
```

Out[25]:
```
6
```

In [23]:
```
plt.plot(test5)
plt.plot(predictions5, color='red')
```

Out[23]:
```
[<matplotlib.lines.Line2D at 0x7f9841a9c630>]
```