Problem Statement:

Pick up the following stocks and generate forecasts accordingly Stocks:

- 1. NASDAQ.AAPL
- 2. NASDAQ.ADP
- 3. NASDAQ.CBOE
- 4. NASDAQ.CSCO
- 5. NASDAQ.EBAY

Importing all necesarry libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#from pandas.tools.plotting import autocorrelation_plot
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.arima_model import ARIMA, ARMAResults
import datetime
import sys
import itertools
import warnings
from sklearn.metrics import mean_squared_error
import seaborn as sns
import statsmodels
import statsmodels.stats.diagnostic as diag
from statsmodels.tsa.stattools import adfuller
from scipy.stats.mstats import normaltest
from matplotlib.pyplot import acorr
plt.style.use('fivethirtyeight')
%matplotlib inline
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files data_stocks.csv

- **data_stocks.csv**(application/vnd.ms-excel) - 134290457 bytes, last modified: 1/29/2021 - 100% done

```
Saving data_stocks.csv to data_stocks.csv
```

```
import io
df = pd.read_csv(io.BytesIO(uploaded['data_stocks.csv']))
df.head()
```

	DATE	SP500	NASDAQ.AAL	NASDAQ.AAPL	NASDAQ.ADBE	NASDAQ.ADI	NASDAQ.ADP
0	1491226200	2363.6101	42.3300	143.6800	129.6300	82.040	102.2300
1	1491226260	2364.1001	42.3600	143.7000	130.3200	82.080	102.1400
2	1491226320	2362.6799	42.3100	143.6901	130.2250	82.030	102.2125
3	1491226380	2364.3101	42.3700	143.6400	130.0729	82.000	102.1400
4	1491226440	2364.8501	42.5378	143.6600	129.8800	82.035	102.0600

```
df["DATE"].dtypes
```

```
dtype('int64')
```

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

```
df['DATE'].tail()
```

```
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]
```

```
df.index = df['DATE']
```

```
df.drop('DATE',axis = 1,inplace=True)
```

```
df.tail()
```

DATE	SP500	NASDAQ.AAL	NASDAQ.AAPL	NASDAQ.ADBE	NASDAQ.ADI	NASDAQ.ADP	NASDAQ.A
2017-08-31 19:58:00	2472.22	44.72	164.11	155.090	83.67	106.565	11

NASDAQ.AAPL

```
df_AAPL = df[['NASDAQ.AAPL']].copy()
```

```
df_AAPL.tail()
```

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

```
df_AAPL.count()
```

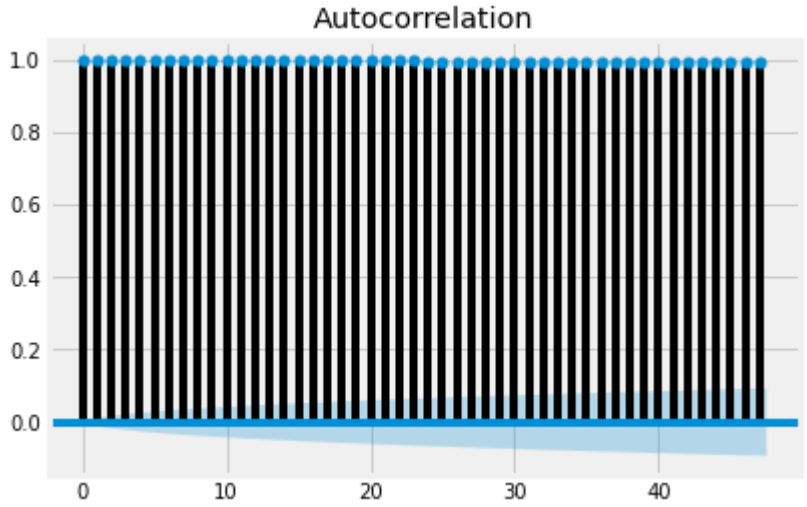
```
NASDAQ.AAPL    41266
dtype: int64
```

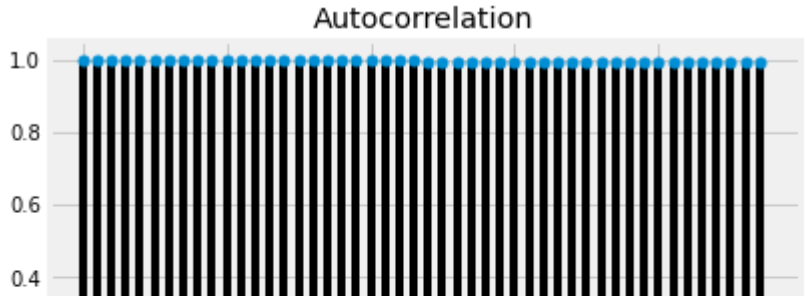
```
df_AAPL.plot()
```

```
NASDAQ.AAPL    41266
dtype: int64
```

Stationary means mean,variance and covariance is constant over periods.

```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(df_AAPL)
```





Converting series to stationary


```
df_AAPL.shift(1)
```

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

```
X = df_AAPL.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
```

ARIMA model

```
p=d=q=range(0,6)
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
```

```
(2, 0, 5) -64715.502681032
(2, 1, 0) -64717.93086809867
(2, 1, 1) -64725.40319961721
(2, 2, 0) -56350.17877202308
(2, 2, 1) -64659.769804128315
(2, 2, 2) -64677.502365500564
(2, 2, 3) -64691.46246876808
(2, 2, 4) -64685.84817785518
(2, 2, 5) -64693.66249286859
(3, 0, 0) -64712.21547313593
(3, 0, 1) -64719.72556096892
(3, 0, 2) -64717.73653537771
(3, 0, 3) -64716.71699425264
(3, 0, 4) -64716.65501111015
(3, 0, 5) -64715.34579769155
(3, 1, 0) -64719.31896994381
(3, 1, 1) -64723.40583150582
(3, 1, 2) -64722.548049394085
(3, 1, 3) -64730.23498554829
(3, 2, 0) -58063.13991574173
(3, 2, 1) -64689.83897493665
(3, 2, 2) -64694.61297205488
(3, 2, 3) -64683.629782719
(3, 2, 4) -64690.80892789339
(3, 2, 5) -64691.410751982694
(4, 0, 0) -64713.65383302767
(4, 0, 1) -64717.729736205656
(4, 0, 2) -64716.80279567305
(4, 0, 3) -64724.48324530764
```

```
(4, 0, 3) -64724.48524956764
(4, 0, 4) -64740.39799828685
(4, 0, 5) -64736.72723943158
(4, 1, 0) -64724.018654585685
(4, 1, 1) -64723.51089426236
(4, 1, 2) -64722.38263266637
(4, 1, 3) -64733.81297745615
(4, 1, 4) -64744.94385837171
(4, 2, 0) -59269.36850049677
(4, 2, 1) -64692.28391601001
(4, 2, 2) -64639.31493635605
(4, 2, 3) -64687.60041243113
(4, 2, 4) -64688.01446272283
(4, 2, 5) -64696.683434018705
(5, 0, 0) -64718.28359458002
(5, 0, 1) -64716.73288922297
(5, 0, 2) -64716.61860438483
(5, 0, 3) -64726.15379558882
(5, 0, 4) -64731.59334920652
(5, 0, 5) -64736.748883241
(5, 1, 0) -64722.574194261004
(5, 1, 1) -64721.560227816764
(5, 1, 2) -64721.03844522101
(5, 1, 3) -64726.319806849395
(5, 1, 4) -64742.966680323254
(5, 2, 0) -60006.91481505723
(5, 2, 1) -64703.69781624562
(5, 2, 2) -64691.2139680719
(5, 2, 3) -64625.62119157975
(5, 2, 4) -64646.7733992742
(5, 2, 5) -64674.03015284707
```

```
#p,d,q
#p -> Periods taken for auto regressive model
#d -> Integrated order, difference
#q -> Periods in moving average model
model_arima = ARIMA(train, order=(4,1,4))
model_arima_fit = model_arima.fit()
print(model_arima_fit.aic)
```

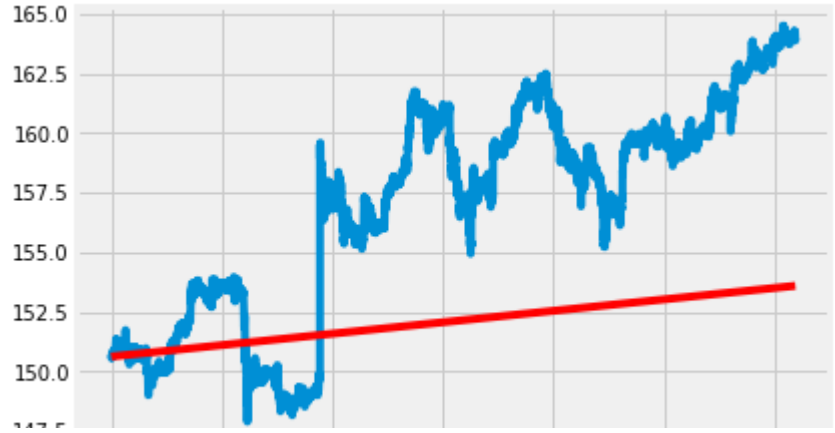
```
-64744.94385837171
```

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

```
array([150.61361413, 150.61507159, 150.61484591, ..., 153.58384484,
       153.58408475, 153.58432466])
```

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

```
[<matplotlib.lines.Line2D at 0x7f502a80e780>]
```




```
res1 = mean_squared_error(test,predictions)
```

```
res1 = round(res1,2)
```

```
37.55273431612137
```

NASDAQ.ADP

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

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

```
df_ADP.count()
```

```
NASDAQ.ADP    41266
dtype: int64
```

```
df_ADP.plot()
```