Project 5 - Time Series Model

In [2]:

```
# Importing Module and aliasing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pandas.plotting import autocorrelation_plot
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.arima_model import ARIMA, ARMAResults
import datetime
import sys
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
import warnings
warnings.filterwarnings('ignore')
import datetime as dt
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
```

In [3]:

```
#Read CSV (comma-separated) file into DataFrame
df = pd.read_csv('data_stocks.csv')
```

In [4]:

```
#The summary statistics of the 'df' dataframe
df.describe()
```

Out[4]:

	DATE	SP500	NASDAQ.AAL	NASDAQ.AAPL	NASDAQ.ADBE	NASDAQ.AD
count	4.126600e+04	41266.000000	41266.000000	41266.000000	41266.00000	41266.00000
mean	1.497749e+09	2421.537882	47.708346	150.453566	141.31793	79.44687
std	3.822211e+06	39.557135	3.259377	6.236826	6.91674	2.00028
min	1.491226e+09	2329.139900	40.830000	140.160000	128.24000	74.80000
25%	1.494432e+09	2390.860100	44.945400	144.640000	135.19500	78.03000
50%	1.497638e+09	2430.149900	48.360000	149.945000	142.26000	79.41000
75%	1.501090e+09	2448.820100	50.180000	155.065000	147.10000	80.58000
max	1.504210e+09	2490.649900	54.475000	164.510000	155.33000	90.44000

8 rows × 502 columns

In [5]:

```
#Check for any NA’s in the dataframe.
df.isnull().values.any()
```

Out[5]:

```
False
```

1. NASDAQ.AAPL

In [6]:

```
#Makes a copy of df dataframe.
df1 = df.copy()
```

In [7]:

```
#Creating a column 'AAPL_LOG' with the log values of 'NASDAQ.AAPL' column data
df1["AAPL_LOG"] = df1["NASDAQ.AAPL"].apply(lambda x:np.log(x))
```

In [8]:

```
#Returns the first 5 rows of df1 dataframe
df1.head()
```

Out[8]:

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

5 rows × 503 columns

In [9]:

```
#Type of the data in 'DATE' column
type(df1["DATE"][0])
```

Out[9]:

```
numpy.int64
```

In [10]:

```
#Creating a new column 'DATE_NEW' with formatted timestamp
df1["DATE_NEW"] = df1["DATE"].apply(lambda x:dt.datetime.fromtimestamp(x).strftime("%Y-%m-%
```

In [11]:

```
#Returns the first 5 rows of df1 dataframe
df1.head()
```

Out[11]:

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

5 rows × 504 columns

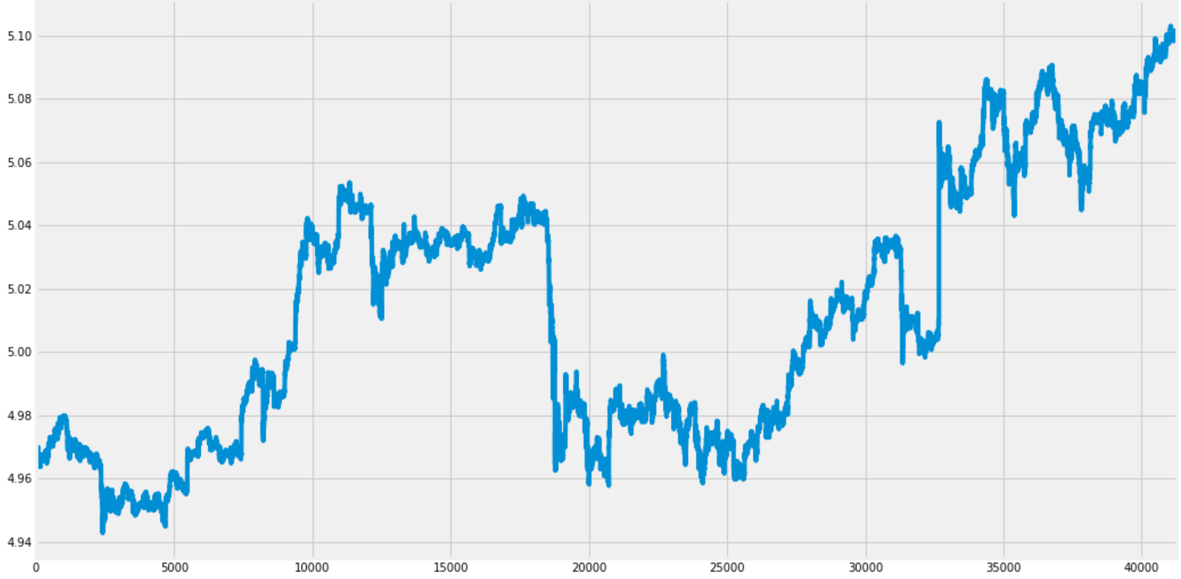
In [12]:

```
#Prints Durbin-Watson statistic of data in "AAPL_LOG" column of 'df1'.
print("Durbin-Watson statistic:",sm.stats.durbin_watson(df1["AAPL_LOG"]))
```

```
Durbin-Watson statistic: 1.5195875753588083e-08
```

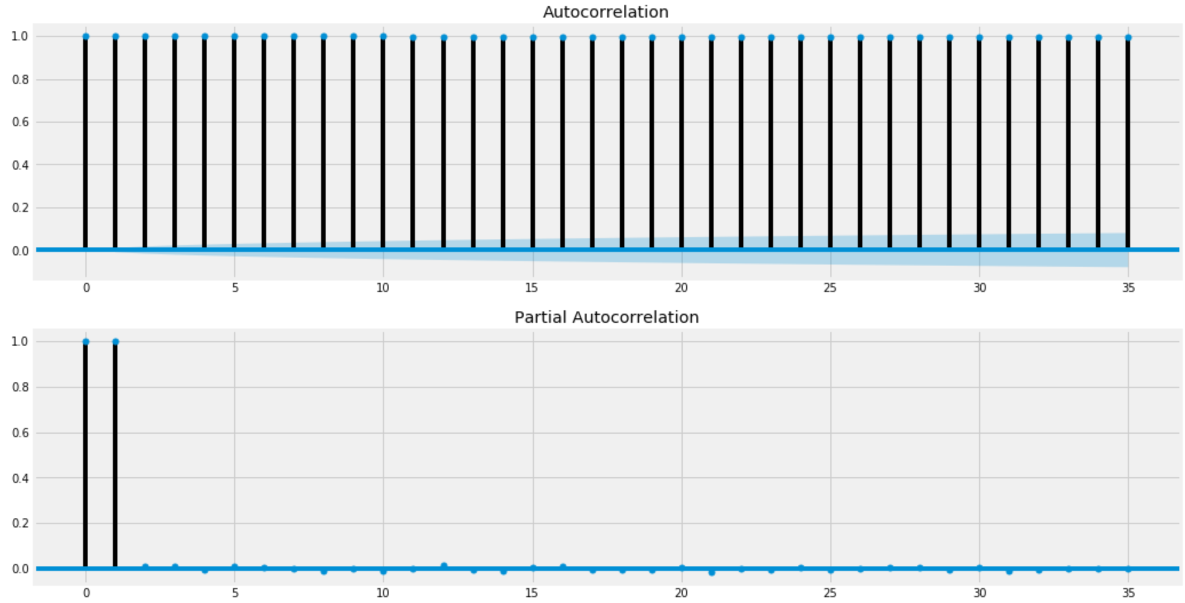
In [13]:

```
#Series Plot
df1["AAPL_LOG"].plot(figsize=(16,9))
plt.show()
```



In [14]:

```
#Autocorrelation Plot
fig = plt.figure(figsize=(16,9))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df1["AAPL_LOG"].values.squeeze(), lags=35, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df1["AAPL_LOG"], lags=35, ax=ax2)
```



In [15]:

```
#Getting the 'AAPL_LOG' column values as array with dropping NaN values
array1 = (df1["AAPL_LOG"].dropna().as_matrix())
```

In [16]:

```
#Creating a column 'AAPL_LOG_DIFF' with data as difference of 'AAPL_LOG' column current row
df1["AAPL_LOG_DIFF"] = df1["AAPL_LOG"] - df1["AAPL_LOG"].shift(periods=-1)
```

In [17]:

```
#Creating ARMA Model
model1 = sm.tsa.ARMA(array1,(2,0)).fit()
#Prints model parameter
print(model1.params)
```

```
[5.02083888 0.99073775 0.0091842 ]
```

In [18]:

```
#Printing Model's AIC, BIC and HQIC values
print(model1.aic, model1.bic, model1.hqic)
```

```
-492715.6402172709 -492681.1290404895 -492704.7324359643
```

In [19]:

```
#Finding the best values for ARIMA model parameter
aic=999999
a,b,c = 0,0,0

for p in range(3):
    for q in range(1,3):
        for r in range(3):
            try:
                model= ARIMA(array1,(p,q,r)).fit()
                if(aic > model1.aic):
                    aic = model1.aic
                    a,b,c = p,q,r
            except:
                pass

print(a,b,c)
```

```
C:\Users\mallikarjuna.m\AppData\Local\Continuum\anaconda\lib\site-packages\s
tatsmodels\base\model.py:508: ConvergenceWarning: Maximum Likelihood optimiz
ation failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
C:\Users\mallikarjuna.m\AppData\Local\Continuum\anaconda\lib\site-packages\s
tatsmodels\base\model.py:508: ConvergenceWarning: Maximum Likelihood optimiz
ation failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
C:\Users\mallikarjuna.m\AppData\Local\Continuum\anaconda\lib\site-packages\s
tatsmodels\base\model.py:508: ConvergenceWarning: Maximum Likelihood optimiz
ation failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)

0 1 0

C:\Users\mallikarjuna.m\AppData\Local\Continuum\anaconda\lib\site-packages\s
tatsmodels\base\model.py:508: ConvergenceWarning: Maximum Likelihood optimiz
ation failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
```

In [20]:

```
#Creating and fitting ARIMA model
model1_arima = ARIMA(array1,(0, 1, 0)).fit()
```

In [21]:

```
#Prints Durbin-Watson statistic of model1_arima.
print("Durbin-Watson statistic:",sm.stats.durbin_watson(model1_arima.resid))
```

```
Durbin-Watson statistic: 2.01849625374405
```

In [22]:

```
#Predicting the values using ARIMA Model
pred1 = model1_arima.predict()
pred1
```

Out[22]:

```
array([3.20258375e-06, 3.20258375e-06, 3.20258375e-06, ...,
       3.20258375e-06, 3.20258375e-06, 3.20258375e-06])
```

In [23]:

```
#Printing RMSE value for the model
print("RMSE for Model1=",np.sqrt(mean_squared_error(pred1,df1["AAPL_LOG_DIFF"][:-1])))
```

```
RMSE for Model1= 0.0006179891020655316
```

2. NASDAQ.ADP

In [24]:

```
#Makes a copy of df dataframe.
df2 = df.copy()
```

In [25]:

```
#Creating a column 'ADP_LOG' with the log values of 'NASDAQ.ADP' column data
df2["ADP_LOG"] = df2["NASDAQ.ADP"].apply(lambda x:np.log(x))
```

In [26]:

```
#Returns the first 5 rows of df2 dataframe
df2.head()
```

Out[26]:

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

5 rows × 503 columns

In [27]:

```
#Creating a new column 'DATE_NEW' with formatted timestamp
df2["DATE_NEW"] = df2["DATE"].apply(lambda x:dt.datetime.fromtimestamp(x).strftime("%Y-%m-%
```

In [28]:

```
#Returns the first 5 rows of df2 dataframe
df2.head()
```

Out[28]:

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

5 rows × 504 columns

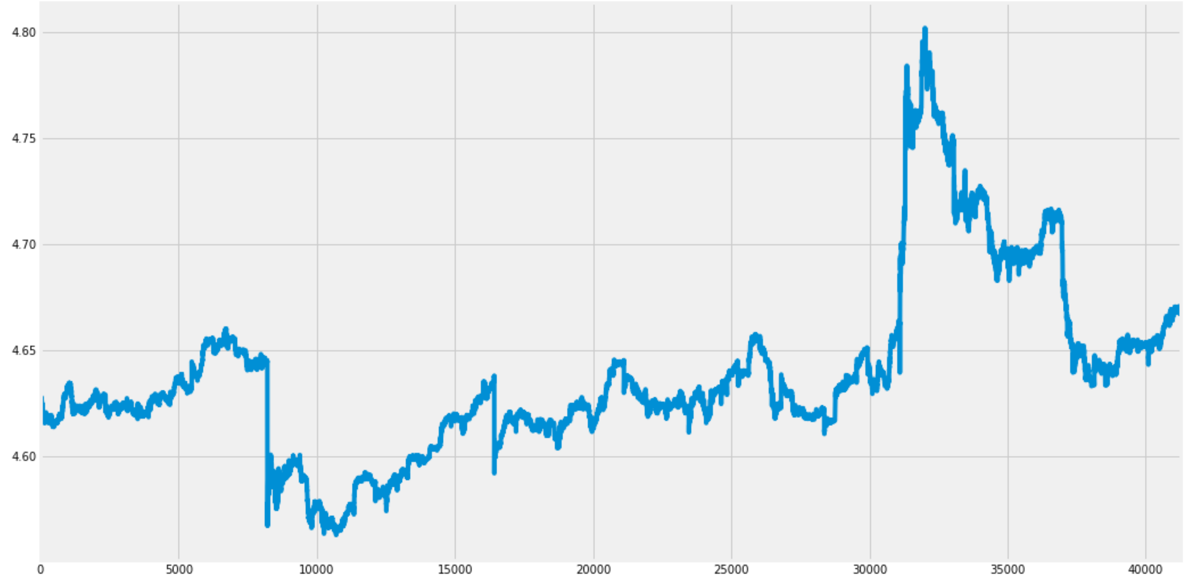
In [29]:

```
#Prints Durbin-Watson statistic of "ADP_LOG" column of 'df2'.
print("Durbin-Watson statistic:",sm.stats.durbin_watson(df2["ADP_LOG"]))
```

```
Durbin-Watson statistic: 2.270798861744159e-08
```

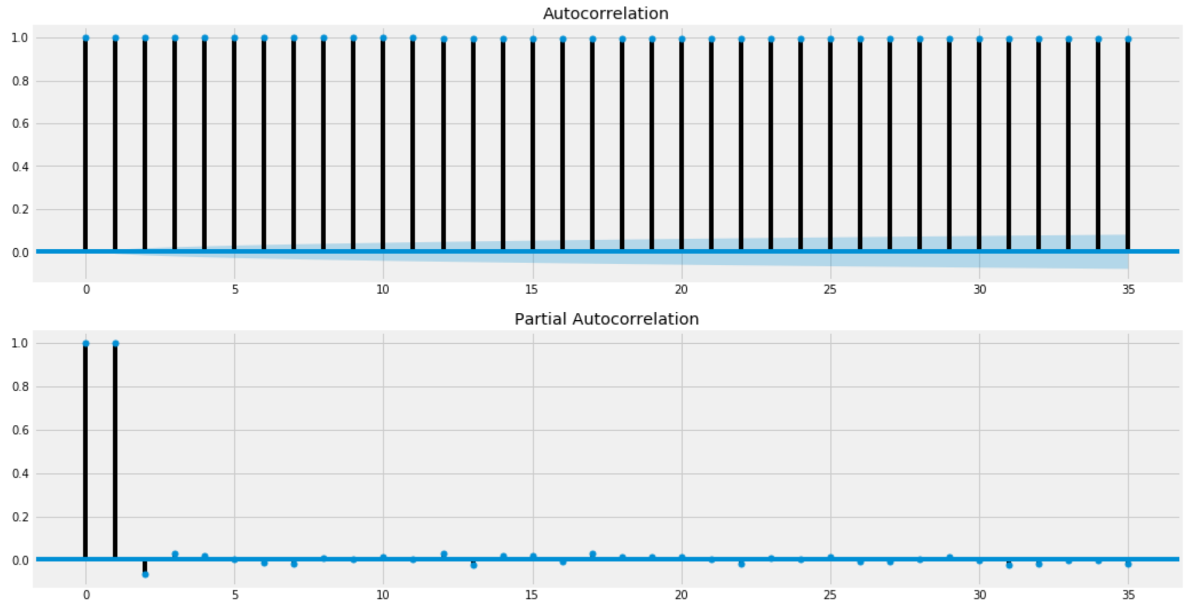
In [30]:

```
#Series Plot
df2["ADP_LOG"].plot(figsize=(16,9))
plt.show()
```



In [31]:

```
#Autocorrelation Plot
fig = plt.figure(figsize=(16,9))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df2["ADP_LOG"].values.squeeze(), lags=35, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df2["ADP_LOG"], lags=35, ax=ax2)
```



In [32]:

```
#Getting the 'ADP_LOG' column values as array with dropping NaN values
array2 = (df2["ADP_LOG"].dropna().as_matrix())
```

In [33]:

```
#Creating a column 'ADP_LOG_DIFF' with data as difference of 'ADP_LOG' column current row a
df2["ADP_LOG_DIFF"] = df2["ADP_LOG"] - df2["ADP_LOG"].shift(periods=-1)
```

In [34]:

```
#Creating ARMA Model
model2 = sm.tsa.ARMA(array2,(2,0)).fit()
#Prints model parameter
print(model2.params)
```

```
[ 4.64047764  1.05961551 -0.05977954]
```

In [35]:

```
#Printing Model2's AIC, BIC and HQIC values
print(model2.aic, model2.bic, model2.hqic)
```

```
-482690.94953447237 -482656.438357691 -482680.0417531658
```

In [36]:

```
#Finding the best values for ARIMA model parameter
aic=999999
a,b,c = 0,0,0

for p in range(3):
    for q in range(1,3):
        for r in range(3):
            try:
                model= ARIMA(array2,(p,q,r)).fit()
                if(aic > model2.aic):
                    aic = model2.aic
                    a,b,c = p,q,r
            except:
                pass

print(a,b,c)
```

```
C:\Users\mallikarjuna.m\AppData\Local\Continuum\anaconda\lib\site-packages\s
tatsmodels\base\model.py:508: ConvergenceWarning: Maximum Likelihood optimiz
ation failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)

0 1 0
```

In [37]:

```
#Creating and fitting ARIMA model
model2_arima = ARIMA(array2,(0, 1, 0)).fit()
```

In [38]:

```
#Prints Durbin-Watson statistic of model2_arima.
print("Durbin-Watson statistic:",sm.stats.durbin_watson(model2_arima.resid))
```

```
Durbin-Watson statistic: 1.8805348562321806
```

In [39]:

```
#Predicting the values using ARIMA Mode2
pred2 = model2_arima.predict()
pred2
```

Out[39]:

```
array([9.84773475e-07, 9.84773475e-07, 9.84773475e-07, ...,
       9.84773475e-07, 9.84773475e-07, 9.84773475e-07])
```

In [40]:

```
#Printing RMSE value for the mode2
print("RMSE for Model-2=",np.sqrt(mean_squared_error(pred2,df2["ADP_LOG_DIFF"][:-1])))
```

```
RMSE for Model-2= 0.0006990223369080944
```

3. NASDAQ.CBOE

In [41]:

```
#Makes a copy of df dataframe.
df3 = df.copy()
```

In [42]:

```
#Creating a column 'CBOE_LOG' with the log values of 'NASDAQ.CBOE' column data
df3["CBOE_LOG"] = df3["NASDAQ.CBOE"].apply(lambda x:np.log(x))
```

In [43]:

```
#Returns the first 5 rows of df3 dataframe
df3.head()
```

Out[43]:

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

5 rows × 503 columns

In [44]:

```
#Creating a new column 'DATE_NEW' with formatted timestamp
df3["DATE_NEW"] = df3["DATE"].apply(lambda x:dt.datetime.fromtimestamp(x).strftime("%Y-%m-%
```

In [45]:

```
#Returns the first 5 rows of df3 dataframe
df3.head()
```

Out[45]:

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

5 rows × 504 columns

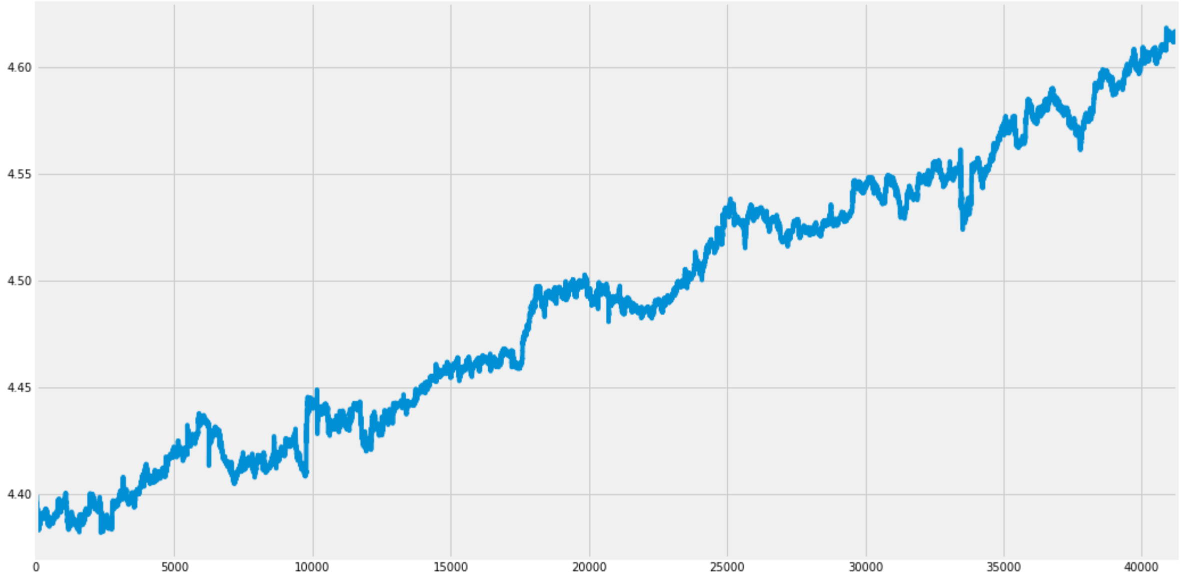
In [46]:

```
#Prints Durbin-Watson statistic of given data.
print("Durbin-Watson statistic:",sm.stats.durbin_watson(df3["CBOE_LOG"]))
```

```
Durbin-Watson statistic: 1.3696573056329881e-08
```

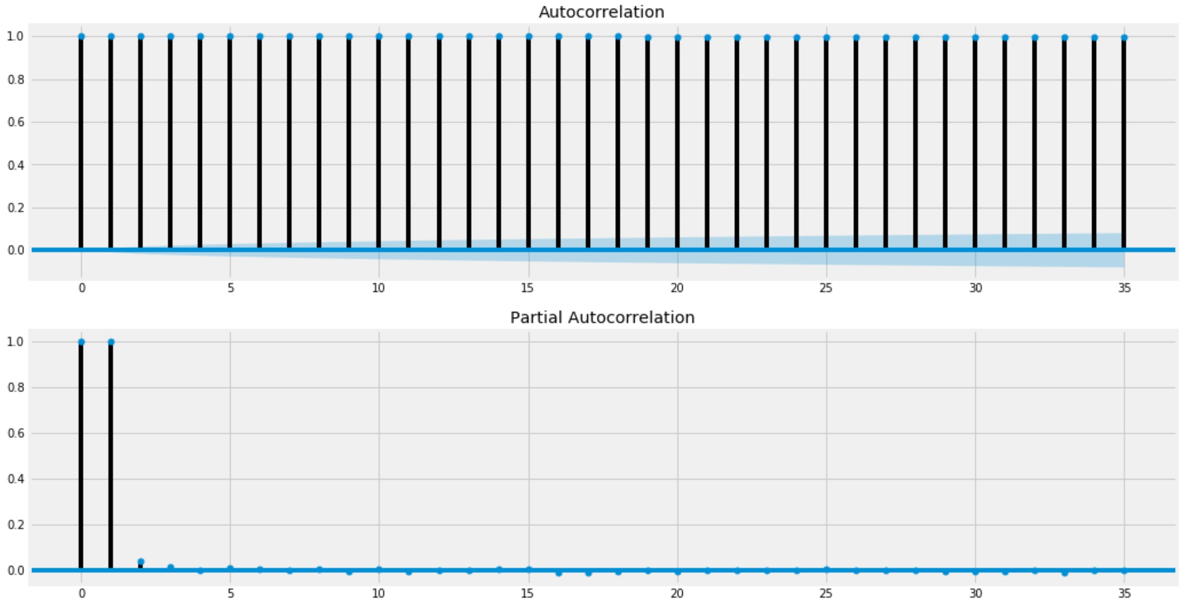
In [47]:

```
#Series Plot
df3["CBOE_LOG"].plot(figsize=(16,9))
plt.show()
```



In [48]:

```
#Autocorrelation Plot
fig = plt.figure(figsize=(16,9))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df3["CBOE_LOG"].values.squeeze(), lags=35, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df3["CBOE_LOG"], lags=35, ax=ax2)
```



In [49]:

```
#Getting the 'CBOE_LOG' column values as array with dropping NaN values
array3 = (df3["CBOE_LOG"].dropna().as_matrix())
```

In [50]:

```
#Creating a column 'CBOE_LOG_DIFF' with data as difference of 'CBOE_LOG' column current row
df3["CBOE_LOG_DIFF"] = df3["CBOE_LOG"] - df3["CBOE_LOG"].shift(periods=-1)
```

In [51]:

```
#Creating ARMA Model
model3 = sm.tsa.ARMA(array3,(2,0)).fit()
#Prints model parameter
print(model3.params)
```

```
[4.50153597 0.92316431 0.07682208]
```

In [52]:

```
#Printing Model's AIC, BIC and HQIC values
print(model3.aic, model3.bic, model3.hqic)
```

```
-506320.7421279051 -506286.2309511237 -506309.8343465985
```

In [53]:

```
#Finding the best values for ARIMA model parameter
aic=999999
a,b,c = 0,0,0

for p in range(3):
    for q in range(1,3):
        for r in range(3):
            try:
                model= ARIMA(array3,(p,q,r)).fit()
                if(aic > model3.aic):
                    aic = model3.aic
                    a,b,c = p,q,r
            except:
                pass

print(a,b,c)
```

```
C:\Users\mallikarjuna.m\AppData\Local\Continuum\anaconda\lib\site-packages\s
tatsmodels\base\model.py:508: ConvergenceWarning: Maximum Likelihood optimiz
ation failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)

0 1 0
```

In [54]:

```
#Creating and fitting ARIMA mode3
model3_arima = ARIMA(array3,(0, 1, 0)).fit()
```

In [55]:

```
#Prints Durbin-Watson statistic of given data.
print("Durbin-Watson statistic:",sm.stats.durbin_watson(model3_arima.resid))
```

```
Durbin-Watson statistic: 2.153351702869301
```

In [56]:

```
#Predicting the values using ARIMA Mode3
pred3 = model3_arima.predict()
pred3
```

Out[56]:

```
array([5.31227345e-06, 5.31227345e-06, 5.31227345e-06, ...,
       5.31227345e-06, 5.31227345e-06, 5.31227345e-06])
```

In [57]:

```
#Printing RMSE value for the mode3
print("RMSE for Model-3=",np.sqrt(mean_squared_error(pred3,df3["CBOE_LOG_DIFF"][:-1])))
```

```
RMSE for Model-3= 0.0005256421962450771
```

4. NASDAQ.CSCO

In [58]:

```
#Makes a copy of df dataframe.
df4 = df.copy()
```

In [59]:

```
#Creating a column 'CSCO_LOG' with the log values of 'NASDAQ.CSCO' column data
df4["CSCO_LOG"] = df4["NASDAQ.CSCO"].apply(lambda x:np.log(x))
```

In [60]:

```
#Returns the first 5 rows of df4 dataframe
df4.head()
```

Out[60]:

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

5 rows × 503 columns

In [61]:

```
#Creating a new column 'DATE_NEW' with formatted timestamp
df4["DATE_NEW"] = df4["DATE"].apply(lambda x:dt.datetime.fromtimestamp(x).strftime("%Y-%m-%
```

In [62]:

```
#Returns the first 5 rows of df_4 dataframe
df4.head()
```

Out[62]:

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

5 rows × 504 columns

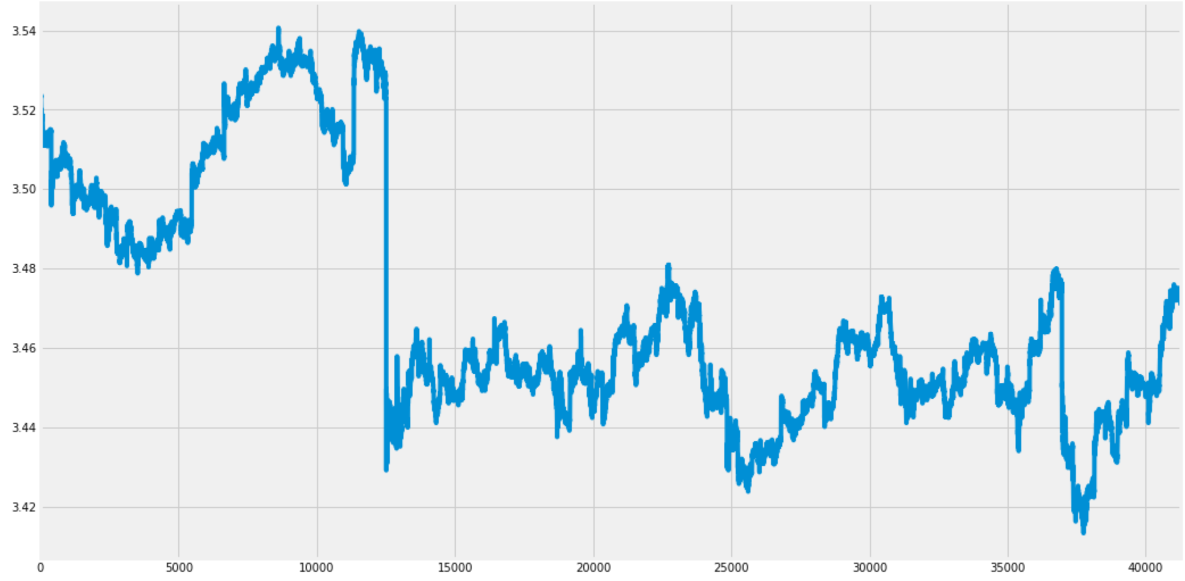
In [63]:

```
#Prints Durbin-Watson statistic of given data.
print("Durbin-Watson statistic:",sm.stats.durbin_watson(df4["CSCO_LOG"]))
```

```
Durbin-Watson statistic: 3.654769389312727e-08
```

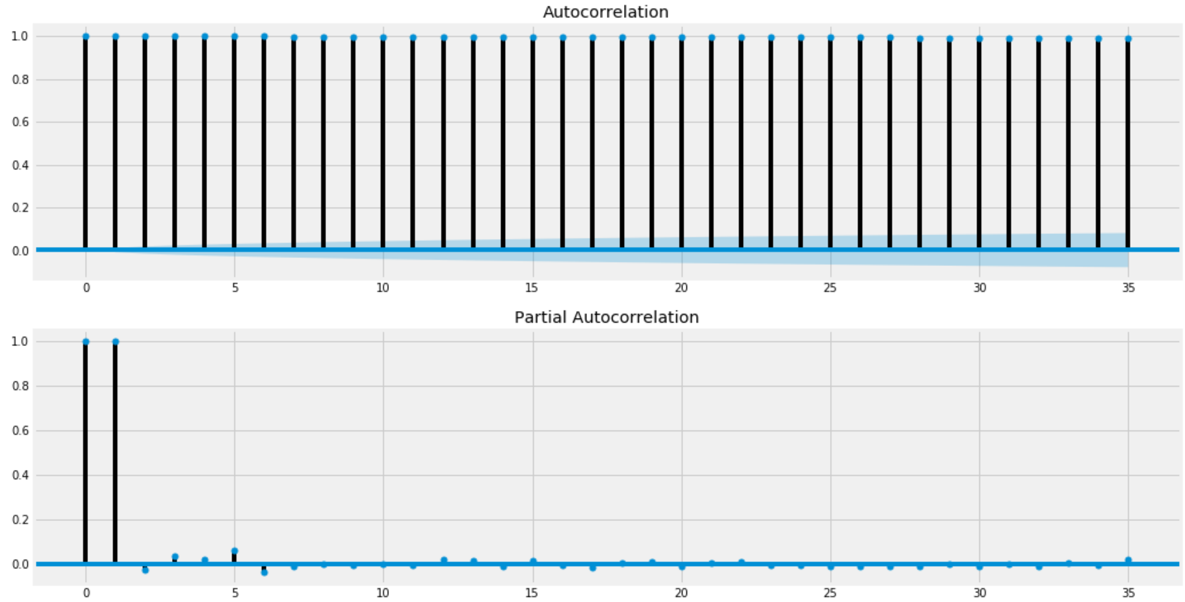
In [64]:

```
#Series Plot
df4["CSCO_LOG"].plot(figsize=(16,9))
plt.show()
```



In [65]:

```
#Autocorrelation Plot
fig = plt.figure(figsize=(16,9))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df4["CSCO_LOG"].values.squeeze(), lags=35, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df4["CSCO_LOG"], lags=35, ax=ax2)
```


Autocorrelation

Partial Autocorrelation

In [66]:

```
#Getting the 'CSCO_LOG' column values as array with dropping NaN values
array4 = (df4["CSCO_LOG"].dropna().as_matrix())
```

In [67]:

```
#Creating a column 'AAPL_LOG_DIFF' with data as difference of 'AAPL_LOG' column current row
df4["CSCO_LOG_DIFF"] = df4["CSCO_LOG"] - df4["CSCO_LOG"].shift(periods=-1)
```

In [68]:

```
#Creating ARMA Model4
model4 = sm.tsa.ARMA(array4,(2,0)).fit()
#Prints model4 parameter
print(model4.params)
```

```
[ 3.47398615  1.01601786 -0.01625508]
```

In [69]:

```
#Printing Model's AIC, BIC and HQIC values
print(model4.aic, model4.bic, model4.hqic)
```

```
-486880.2585551347 -486845.7473783533 -486869.3507738281
```

In [70]:

```
#Finding the best values for ARIMA model parameter
aic=999999
a,b,c = 0,0,0

for p in range(3):
    for q in range(1,3):
        for r in range(3):
            try:
                model= ARIMA(array4,(p,q,r)).fit()
                if(aic > model4.aic):
                    aic = model4.aic
                    a,b,c = p,q,r
            except:
                pass

print(a,b,c)
```

```
0 1 0
```

In [71]:

```
#Creating and fitting ARIMA model4
model4_arima = ARIMA(array4,(0, 1, 0)).fit()
```

In [72]:

```
#Prints Durbin-Watson statistic of given data.
print("Durbin-Watson statistic:",sm.stats.durbin_watson(model4_arima.resid))
```

```
Durbin-Watson statistic: 1.9667794687094717
```

In [73]:

```
#Predicting the values using ARIMA Model4
pred4 = model4_arima.predict()
pred4
```

Out[73]:

```
array([-1.11336651e-06, -1.11336651e-06, -1.11336651e-06, ...,
       -1.11336651e-06, -1.11336651e-06, -1.11336651e-06])
```

In [74]:

```
#Printing RMSE value for the model4
print("RMSE for Model-4=",np.sqrt(mean_squared_error(pred4,df4["CSCO_LOG_DIFF"][:-1])))
```

```
RMSE for Model-4= 0.0006633386742358213
```

5. NASDAQ.EBAY

In [75]:

```
#Makes a copy of df dataframe.
df5 = df.copy()
```

In [76]:

```
#Creating a column 'EBAY_LOG' with the log values of 'NASDAQ.EBAY' column data
df5["EBAY_LOG"] = df5["NASDAQ.EBAY"].apply(lambda x:np.log(x))
```

In [77]:

```
#Returns the first 5 rows of df5 dataframe
df5.head()
```

Out[77]:

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

5 rows × 503 columns

In [78]:

```
#Creating a new column 'DATE_NEW' with formatted timestamp
df5["DATE_NEW"] = df5["DATE"].apply(lambda x:dt.datetime.fromtimestamp(x).strftime("%Y-%m-%
```

In [79]:

```
#Returns the first 5 rows of df5 dataframe
df5.head()
```

Out[79]:

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

5 rows × 504 columns

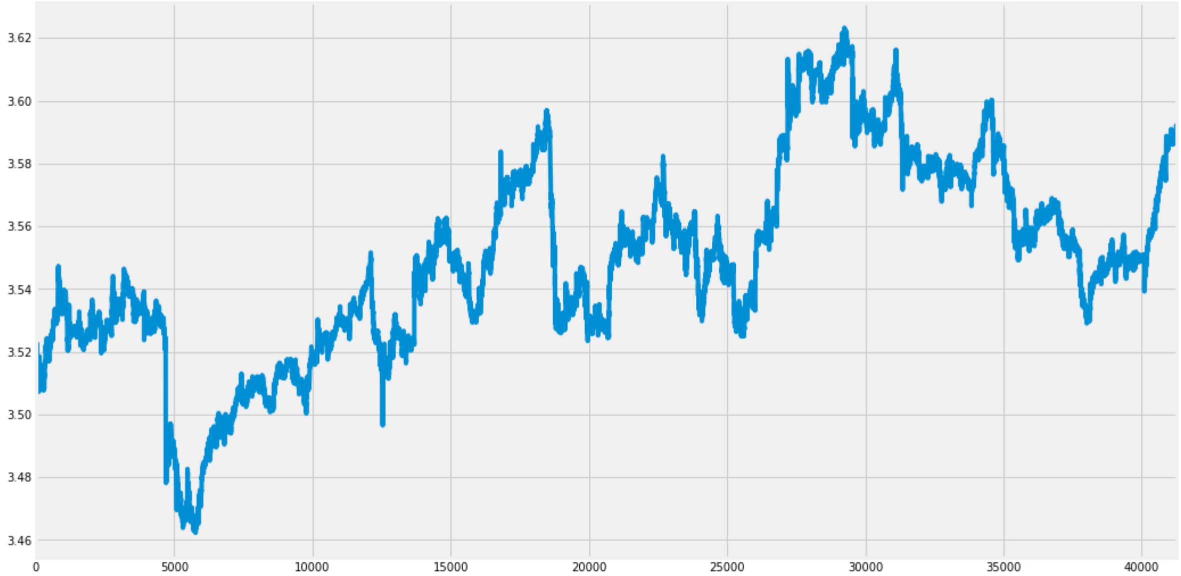
In [80]:

```
#Prints Durbin-Watson statistic of given data.
print("Durbin-Watson statistic:",sm.stats.durbin_watson(df5["EBAY_LOG"]))
```

```
Durbin-Watson statistic: 3.5208792726754005e-08
```

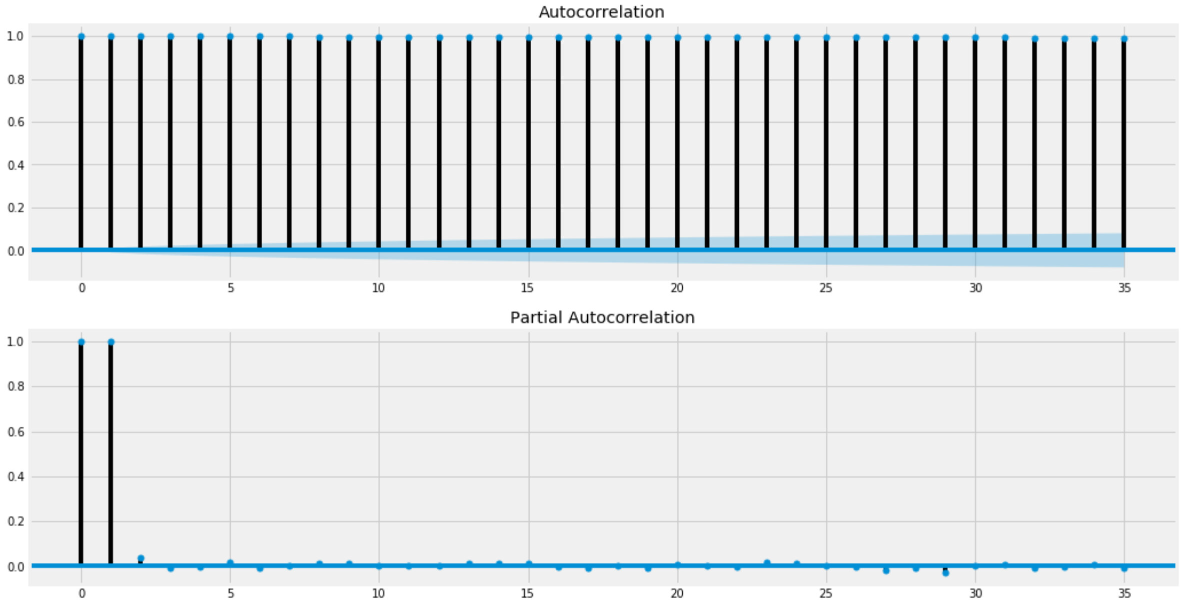
In [81]:

```
#Series Plot
df5["EBAY_LOG"].plot(figsize=(16,9))
plt.show()
```



In [82]:

```
#Autocorrelation Plot
fig = plt.figure(figsize=(16,9))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df5["EBAY_LOG"].values.squeeze(), lags=35, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df5["EBAY_LOG"], lags=35, ax=ax2)
```



In [83]:

```
#Getting the 'EBAY_LOG' column values as array with dropping NaN values
array5 = (df5["EBAY_LOG"].dropna().as_matrix())
```

In [84]:

```
#Creating a column 'EBAY_LOG_DIFF' with data as difference of 'EBAY_LOG' column row and pre
df5["EBAY_LOG_DIFF"] = df5["EBAY_LOG"] - df5["EBAY_LOG"].shift(periods=-1)
```

In [85]:

```
#Creating ARMA Model
model5 = sm.tsa.ARMA(array5,(2,0)).fit()
#Prints model parameter
print(model5.params)
```

```
[3.54872696 0.95983135 0.03996809]
```

In [86]:

```
#Printing Model's AIC, BIC and HQIC values
print(model5.aic, model5.bic, model5.hqic)
```

```
-486608.4996361916 -486573.9884594102 -486597.591854885
```

In [87]:

```
#Finding the best values for ARIMA model parameter
aic=999999
a,b,c = 0,0,0

for p in range(3):
    for q in range(1,3):
        for r in range(3):
            try:
                model= ARIMA(array5,(p,q,r)).fit()
                if(aic > model5.aic):
                    aic = model5.aic
                    a,b,c = p,q,r
            except:
                pass

print(a,b,c)
```

```
C:\Users\mallikarjuna.m\AppData\Local\Continuum\anaconda\lib\site-packages\s
tatsmodels\base\model.py:508: ConvergenceWarning: Maximum Likelihood optimiz
ation failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
C:\Users\mallikarjuna.m\AppData\Local\Continuum\anaconda\lib\site-packages\s
tatsmodels\base\model.py:508: ConvergenceWarning: Maximum Likelihood optimiz
ation failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)

0 1 0

C:\Users\mallikarjuna.m\AppData\Local\Continuum\anaconda\lib\site-packages\s
tatsmodels\base\model.py:508: ConvergenceWarning: Maximum Likelihood optimiz
ation failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
```

In [88]:

```
#Creating and fitting ARIMA model5
model5_arima = ARIMA(array5,(0, 1, 0)).fit()
```

In [89]:

```
#Prints Durbin-Watson statistic of given data.
print("Durbin-Watson statistic:",sm.stats.durbin_watson(model5_arima.resid))
```

```
Durbin-Watson statistic: 2.080160689877867
```

In [90]:

```
#Predicting the values using ARIMA Model5
pred5 = model5_arima.predict()
pred5
```

Out[90]:

```
array([1.90577558e-06, 1.90577558e-06, 1.90577558e-06, ...,
       1.90577558e-06, 1.90577558e-06, 1.90577558e-06])
```

In [91]:

```
#Printing RMSE value for the model
print("RMSE for Model-5=",np.sqrt(mean_squared_error(pred5,df5["EBAY_LOG_DIFF"][:-1])))
```

```
RMSE for Model-5= 0.0006659697472260219
```