Yes Bank Forecate for 123 days (Started from 31 Aug 2019 to 31 Dec 2019)

Using Time Series Analysis

In [ ]:
```
import numpy as np
import pandas as pd
```

Importing File

In [46]:
```
YB_df=pd.read_csv(r'C:\Users\admin\Desktop\Yesbank.csv',header=0)
```

In [47]:
```
YB_df.head()
```

Out[47]:

	Date	Open Price	High Price	Low Price	Close Price	WAP	No.of Shares	No. of Trades	Total Turnover (Rs.)	Deliverable Quantity	% D Qty Trad (
0	12-Jul-05	65.00	70.00	60.15	12.42	62.113290	26141270	53078	1623720286	7123756.0	27
1	13-Jul-05	61.45	62.35	57.40	12.03	60.126246	5756848	13173	346137657	1799410.0	31
2	14-Jul-05	59.40	59.80	57.00	11.61	58.062454	1362144	4998	79089423	481702.0	35
3	15-Jul-05	57.80	64.40	56.25	12.31	61.528808	7338637	18200	451537589	1725030.0	23
4	18-Jul-05	62.80	65.65	62.35	12.88	64.378865	6084805	14645	391732838	1717998.0	28

In [48]:
```
YB_df.dtypes
```

Out[48]:
```
Date                          object
Open Price                   float64
High Price                   float64
Low Price                    float64
Close Price                  float64
WAP                          float64
No.of Shares                   int64
No. of Trades                  int64
Total Turnover (Rs.)           int64
Deliverable Quantity         float64
% Deli. Qty to Traded Qty    float64
Spread High-Low              float64
Spread Close-Open            float64
dtype: object
```

In [49]:
```
YB_df.shape
```

Out[49]:
```
(3504, 13)
```

In [50]:
```
YB_df.isnull().sum()
```

Out[50]:
```
Date                         0
Open Price                   0
High Price                   0
Low Price                    0
Close Price                  0
WAP                          0
No.of Shares                 0
No. of Trades                0
Total Turnover (Rs.)         0
Deliverable Quantity         1
% Deli. Qty to Traded Qty    1
Spread High-Low              0
Spread Close-Open            0
dtype: int64
```

Use Drop function

In [51]:

```
YB_df = YB_df.drop(['Open Price','High Price','Low Price','WAP','No.of Shares'
,'No. of Trades',
                   'Total Turnover (Rs.)', 'Deliverable Quantity', '% Deli. Qt
y to Traded Qty', 'Spread High-Low',
                   'Spread Close-Open'],axis=1)
print(YB_df)
```

```
           Date  Close Price
0     12-Jul-05        12.42
1     13-Jul-05        12.03
2     14-Jul-05        11.61
3     15-Jul-05        12.31
4     18-Jul-05        12.88
5     19-Jul-05        12.79
6     20-Jul-05        12.73
7     21-Jul-05        12.62
8     22-Jul-05        12.50
9     25-Jul-05        12.36
10    26-Jul-05        12.51
11    27-Jul-05        12.70
12    29-Jul-05        12.60
13     1-Aug-05        12.70
14     2-Aug-05        14.05
15     3-Aug-05        14.13
16     4-Aug-05        14.41
17     5-Aug-05        14.37
18     8-Aug-05        14.26
19     9-Aug-05        13.95
20    10-Aug-05        14.39
21    11-Aug-05        14.42
22    12-Aug-05        14.23
23    16-Aug-05        13.99
24    17-Aug-05        13.95
25    18-Aug-05        13.80
26    19-Aug-05        13.63
27    22-Aug-05        13.38
28    23-Aug-05        12.99
29    24-Aug-05        12.90
...         ...          ...
3474  18-Jul-19        85.80
3475  19-Jul-19        83.25
3476  22-Jul-19        91.15
3477  23-Jul-19        90.70
3478  24-Jul-19        89.15
3479  25-Jul-19        87.65
3480  26-Jul-19        96.10
3481  29-Jul-19        94.75
3482  30-Jul-19        86.10
3483  31-Jul-19        91.30
3484   1-Aug-19        88.40
3485   2-Aug-19        88.30
3486   5-Aug-19        81.10
3487   6-Aug-19        85.40
3488   7-Aug-19        86.85
3489   8-Aug-19        89.15
3490   9-Aug-19        82.10
3491  13-Aug-19        73.60
3492  14-Aug-19        76.55
3493  16-Aug-19        79.45
3494  19-Aug-19        76.70
3495  20-Aug-19        71.25
3496  21-Aug-19        65.40
3497  22-Aug-19        56.30
3498  23-Aug-19        59.25
```

```
3499  26-Aug-19        63.00
3500  27-Aug-19        64.30
3501  28-Aug-19        59.50
3502  29-Aug-19        57.35
3503  30-Aug-19        59.50

[3504 rows x 2 columns]
```

In [52]:
```
YB_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3504 entries, 0 to 3503
Data columns (total 2 columns):
Date           3504 non-null object
Close Price    3504 non-null float64
dtypes: float64(1), object(1)
memory usage: 54.8+ KB
```

Importing matplot, seaborn & datatime for data visualisation

In [53]:
```
from datetime import datetime

YB_df['Date'] = pd.to_datetime(YB_df['Date'], infer_datetime_format=True)
YB_df = YB_df.set_index(['Date'])
```

In [135]:
```
YB_df.head()
```

Out[135]:

	Close Price
Date	
2005-07-12	12.42
2005-07-13	12.03
2005-07-14	11.61
2005-07-15	12.31
2005-07-18	12.88

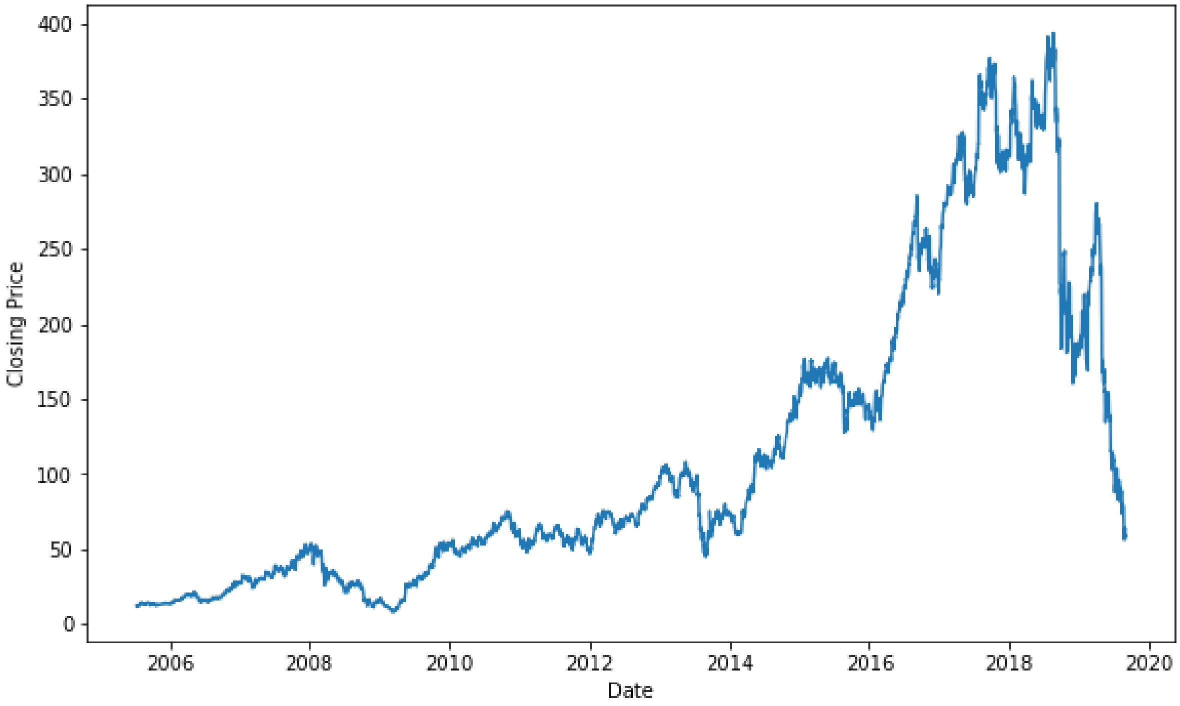
In [56]:
```
YB_df.tail()
```

Out[56]:

	Close Price
Date	
2019-08-26	63.00
2019-08-27	64.30
2019-08-28	59.50
2019-08-29	57.35
2019-08-30	59.50

In [57]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.plot(YB_df)
```

Out[57]:
```
[<matplotlib.lines.Line2D at 0xbe7c668>]
```



Using Rolling Mean & Rolling STD, to find out Stationarity

In [59]:

```
rolmean = YB_df.rolling(window=365).mean()
rolstd = YB_df.rolling(window=365).std()
print(rolmean, rolstd)
```

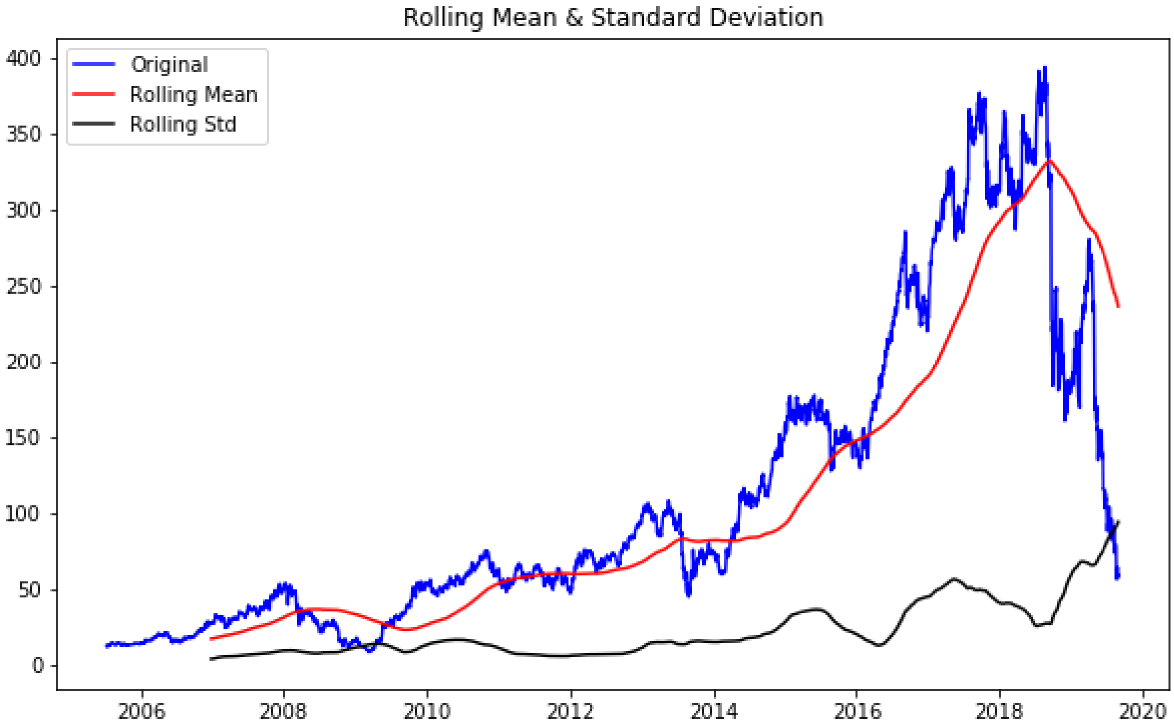
```
            Close Price
Date
2005-07-12          NaN
2005-07-13          NaN
2005-07-14          NaN
2005-07-15          NaN
2005-07-18          NaN
2005-07-19          NaN
2005-07-20          NaN
2005-07-21          NaN
2005-07-22          NaN
2005-07-25          NaN
2005-07-26          NaN
2005-07-27          NaN
2005-07-29          NaN
2005-08-01          NaN
2005-08-02          NaN
2005-08-03          NaN
2005-08-04          NaN
2005-08-05          NaN
2005-08-08          NaN
2005-08-09          NaN
2005-08-10          NaN
2005-08-11          NaN
2005-08-12          NaN
2005-08-16          NaN
2005-08-17          NaN
2005-08-18          NaN
2005-08-19          NaN
2005-08-22          NaN
2005-08-23          NaN
2005-08-24          NaN
...                 ...
2019-07-18   256.341507
2019-07-19   255.570137
2019-07-22   254.823973
2019-07-23   254.092877
2019-07-24   253.368767
2019-07-25   252.637808
2019-07-26   251.917123
2019-07-29   251.217808
2019-07-30   250.510959
2019-07-31   249.833562
2019-08-01   249.162877
2019-08-02   248.486849
2019-08-05   247.817123
2019-08-06   247.133425
2019-08-07   246.494110
2019-08-08   245.861781
2019-08-09   245.232192
2019-08-13   244.577397
2019-08-14   243.939726
2019-08-16   243.302192
2019-08-19   242.645479
2019-08-20   241.954110
2019-08-21   241.241781
2019-08-22   240.500000
```

```
2019-08-23   239.779452
2019-08-26   239.073288
2019-08-27   238.390959
2019-08-28   237.696438
2019-08-29   236.996301
2019-08-30   236.316027

[3504 rows x 1 columns]             Close Price
Date
2005-07-12          NaN
2005-07-13          NaN
2005-07-14          NaN
2005-07-15          NaN
2005-07-18          NaN
2005-07-19          NaN
2005-07-20          NaN
2005-07-21          NaN
2005-07-22          NaN
2005-07-25          NaN
2005-07-26          NaN
2005-07-27          NaN
2005-07-29          NaN
2005-08-01          NaN
2005-08-02          NaN
2005-08-03          NaN
2005-08-04          NaN
2005-08-05          NaN
2005-08-08          NaN
2005-08-09          NaN
2005-08-10          NaN
2005-08-11          NaN
2005-08-12          NaN
2005-08-16          NaN
2005-08-17          NaN
2005-08-18          NaN
2005-08-19          NaN
2005-08-22          NaN
2005-08-23          NaN
2005-08-24          NaN
...                 ...
2019-07-18    84.745187
2019-07-19    85.036128
2019-07-22    85.281015
2019-07-23    85.541374
2019-07-24    85.816214
2019-07-25    86.088699
2019-07-26    86.295394
2019-07-29    86.532287
2019-07-30    86.824741
2019-07-31    87.100055
2019-08-01    87.398346
2019-08-02    87.686016
2019-08-05    88.028666
2019-08-06    88.318687
2019-08-07    88.632346
2019-08-08    88.929649
2019-08-09    89.273623
```

```
2019-08-13    89.653829
2019-08-14    90.019203
2019-08-16    90.358173
2019-08-19    90.695871
2019-08-20    91.038243
2019-08-21    91.399993
2019-08-22    91.799480
2019-08-23    92.187392
2019-08-26    92.551920
2019-08-27    92.920133
2019-08-28    93.307515
2019-08-29    93.699613
2019-08-30    94.084712

[3504 rows x 1 columns]
```

In [87]:

```
orig = plt.plot(YB_df, color='blue', label='Original')
mean = plt.plot(rolmean, color='red', label='Rolling Mean')
std = plt.plot(rolstd, color='black', label='Rolling Std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard Deviation')
plt.show(block=False)
```



Dickey Fuller Test for Stationarity

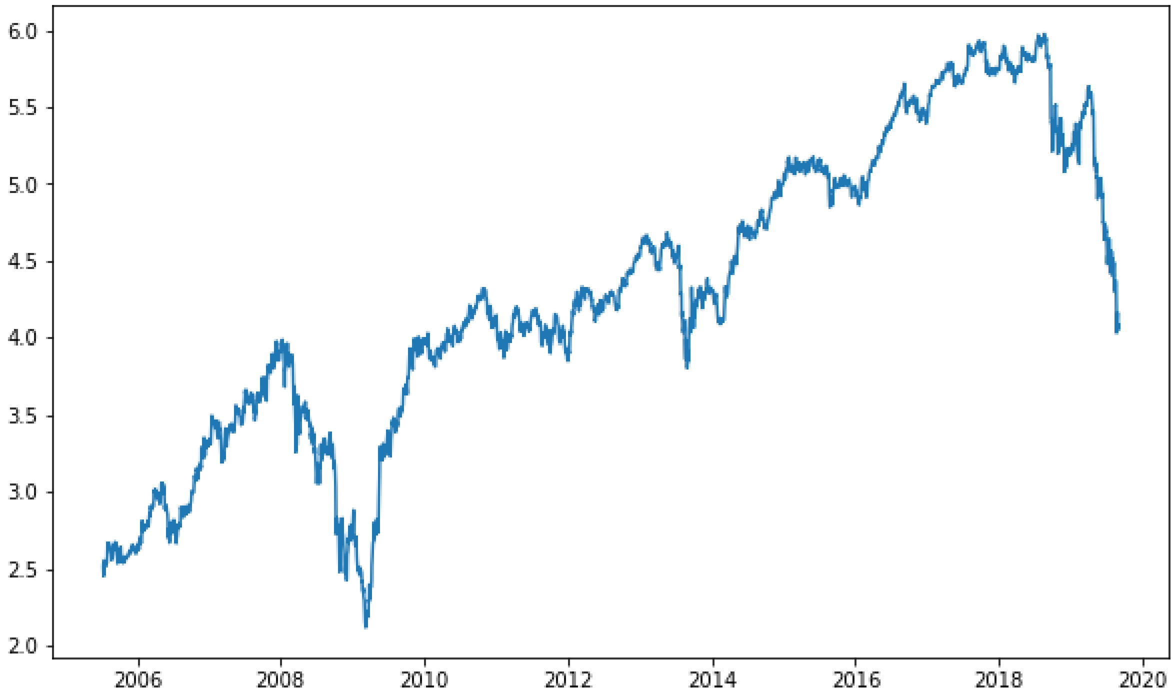
In [88]:
```
from statsmodels.tsa.stattools import adfuller

print ('Results of Dickey-Fuller Test:')
Dickey_fuller_test = adfuller(YB_df['Close Price'], autolag='AIC')

dfoutput = pd.Series(Dickey_fuller_test[0:4], index=['Test Statistics', 'p-value', '#Lags Used', 'Number of Observation Used'])
for key,value in Dickey_fuller_test[4].items():
    dfoutput['Critical Value (%s)'%key] = value

print(dfoutput)
```

```
Results of Dickey-Fuller Test:
Test Statistics                  -1.597308
p-value                           0.484946
#Lags Used                       25.000000
Number of Observation Used     3478.000000
Critical Value (1%)              -3.432232
Critical Value (5%)              -2.862371
Critical Value (10%)             -2.567213
dtype: float64
```

For Estimating trend using Log-Scale

In [89]:
```
YB_df_logscale = np.log(YB_df)
plt.plot(YB_df_logscale)
```

Out[89]:
```
[<matplotlib.lines.Line2D at 0x10453080>]
```

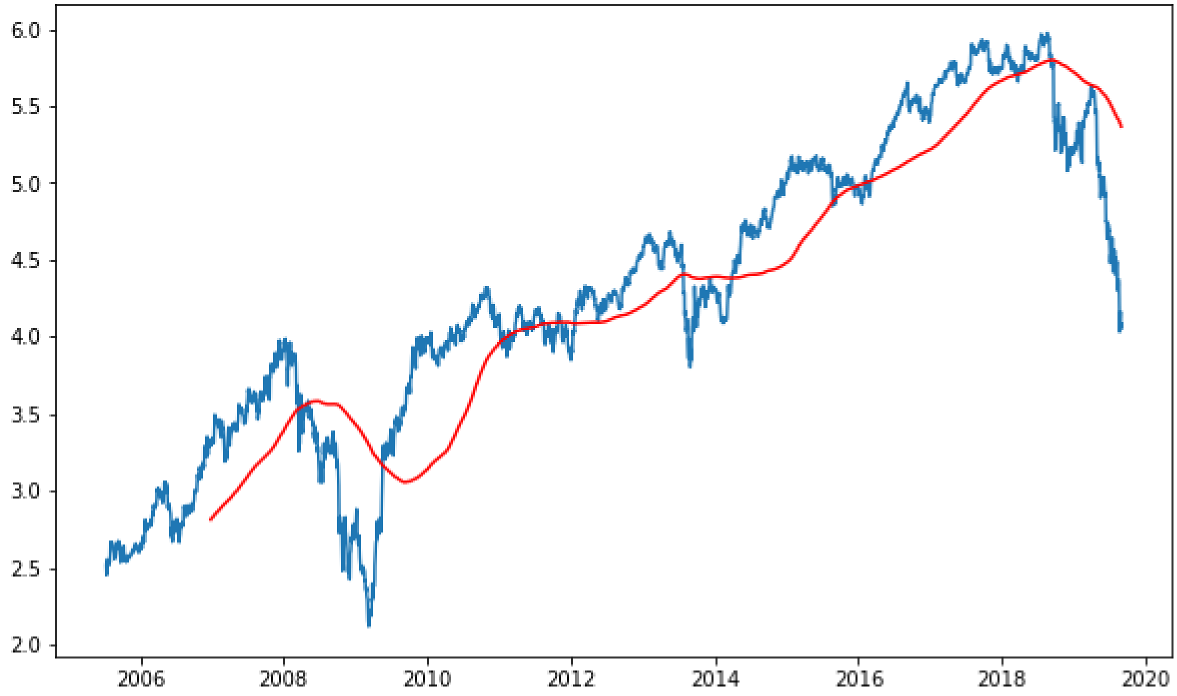


In [90]:

```
movingAverage = YB_df_logscale.rolling(window=365).mean()
movingStd = YB_df_logscale.rolling(window=365).std()
plt.plot(YB_df_logscale)
plt.plot(movingAverage, color='red')
```

Out[90]:

```
[<matplotlib.lines.Line2D at 0x104e56a0>]
```



In [91]:

```
YB_Logscale_minus_movingaverage = YB_df_logscale - movingAverage
YB_Logscale_minus_movingaverage.head(12)
```

Out[91]:

	Close Price
Date	
2005-07-12	NaN
2005-07-13	NaN
2005-07-14	NaN
2005-07-15	NaN
2005-07-18	NaN
2005-07-19	NaN
2005-07-20	NaN
2005-07-21	NaN
2005-07-22	NaN
2005-07-25	NaN
2005-07-26	NaN
2005-07-27	NaN

In [92]:

```
YB_Logscale_minus_movingaverage.dropna(inplace=True)
YB_Logscale_minus_movingaverage.head(12)
```

Out[92]:

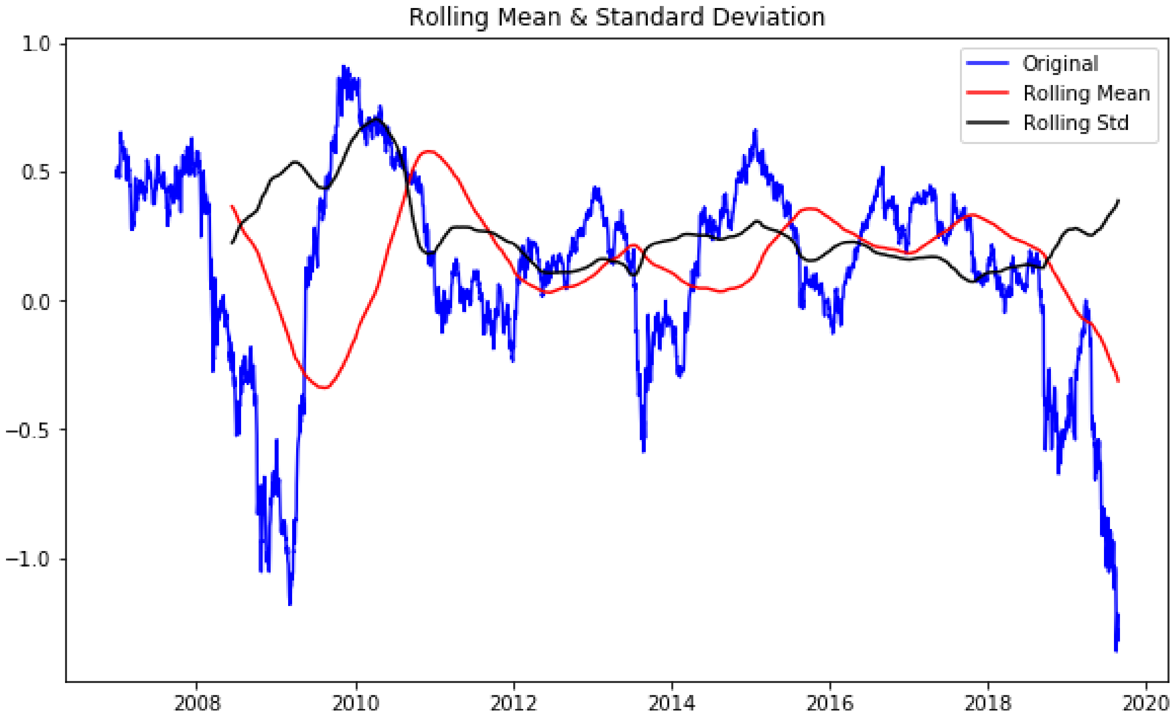
	Close Price
Date	
2006-12-26	0.494501
2006-12-27	0.502135
2006-12-28	0.478693
2006-12-29	0.476746
2007-01-02	0.520696
2007-01-03	0.503641
2007-01-04	0.500431
2007-01-05	0.506477
2007-01-08	0.490349
2007-01-09	0.491041
2007-01-10	0.475095
2007-01-11	0.475125

Using def function, where call come together in Dickey Fuller

In [93]:

```
from statsmodels.tsa.stattools import adfuller

def test_Stationarity(timeseries):

    movingAverage = timeseries.rolling(window=365).mean()
    movingSTD = timeseries.rolling(window=365).std()

    orig = plt.plot(timeseries, color='blue', label='Original')
    mean = plt.plot(movingAverage, color='red', label='Rolling Mean')
    std = plt.plot(movingSTD, color='black', label='Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    print ('Results of Dickey-Fuller Test:')
    Dickey_fuller_test = adfuller(timeseries['Close Price'], autolag='AIC')

    dfoutput = pd.Series(Dickey_fuller_test[0:4], index=['Test Statistics', 'p-value', '#Lags Used', 'Number of Observation Used'])
    for key,value in Dickey_fuller_test[4].items():
        dfoutput['Critical Value (%s)'%key] = value

    print(dfoutput)
```

```
In [94]: test_Stationarity(YB_Logscale_minus_movingaverage)
```

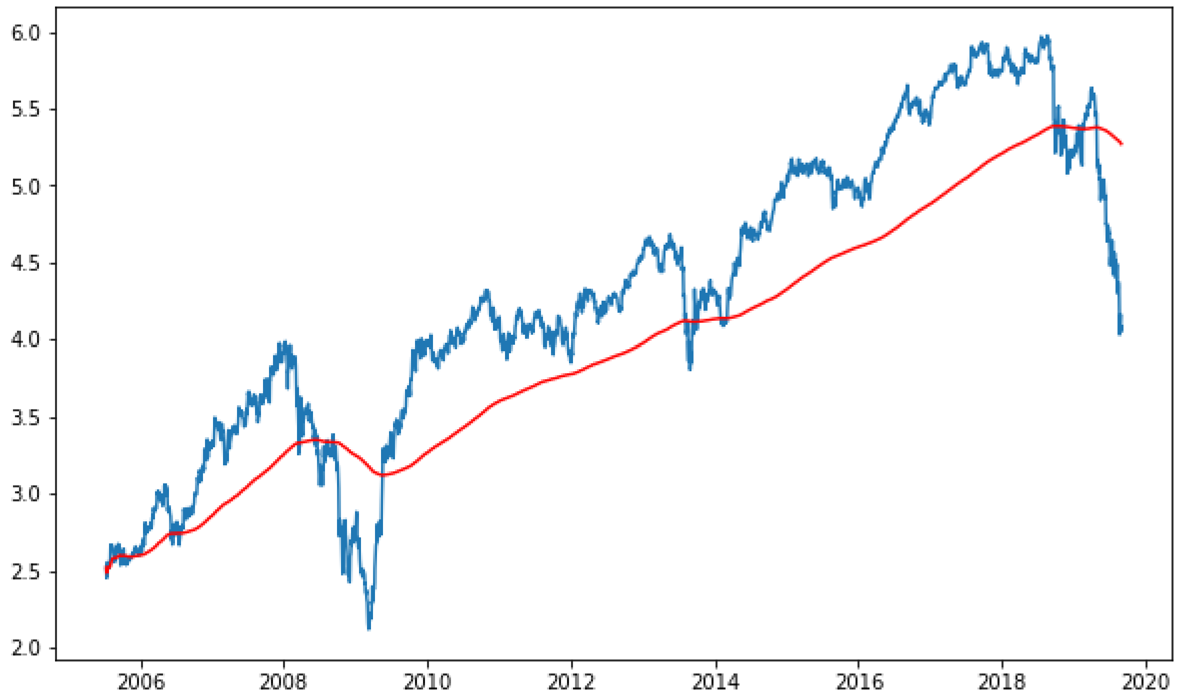


```
Results of Dickey-Fuller Test:
Test Statistics                  -1.132803
p-value                           0.701815
#Lags Used                        2.000000
Number of Observation Used     3137.000000
Critical Value (1%)              -3.432436
Critical Value (5%)              -2.862462
Critical Value (10%)             -2.567261
dtype: float64
```

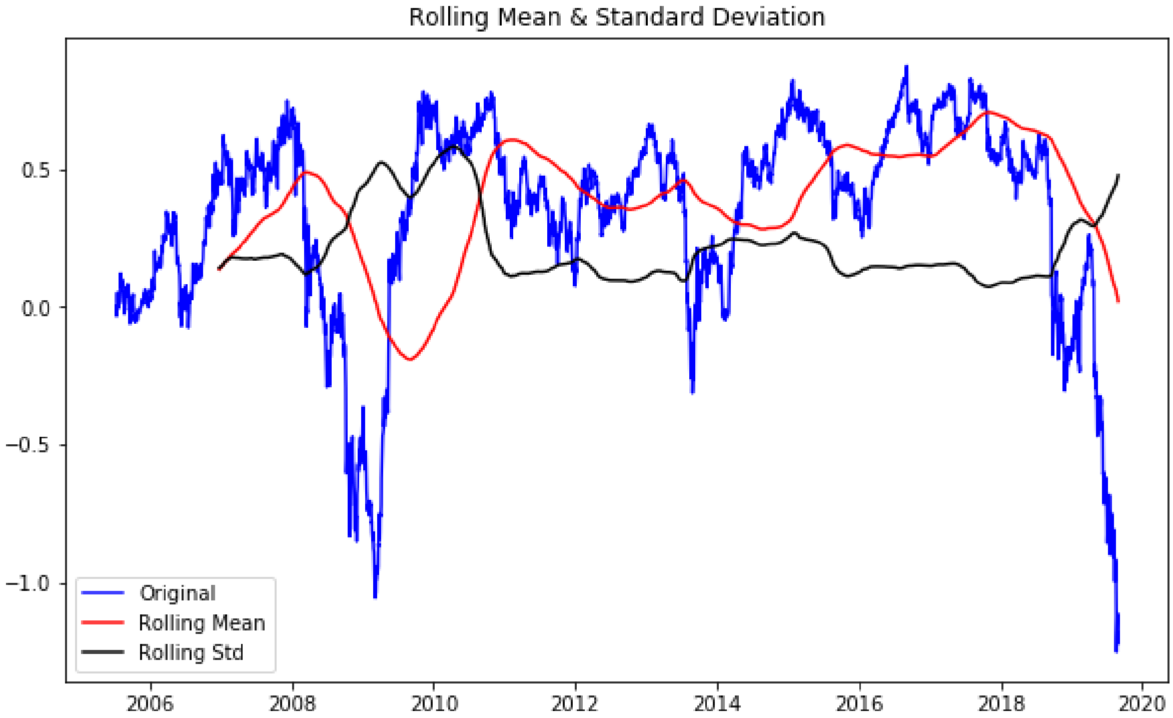
Using EWA to find trend

In [95]:
```
EWA = YB_df_logscale.ewm(halflife=365, min_periods=0, adjust=True).mean()
plt.plot(YB_df_logscale)
plt.plot(EWA, color='red')
```

Out[95]:
```
[<matplotlib.lines.Line2D at 0x115ba710>]
```



```
In [96]: LEWA = YB_df_logscale - EWA
         test_Stationarity(LEWA)
```

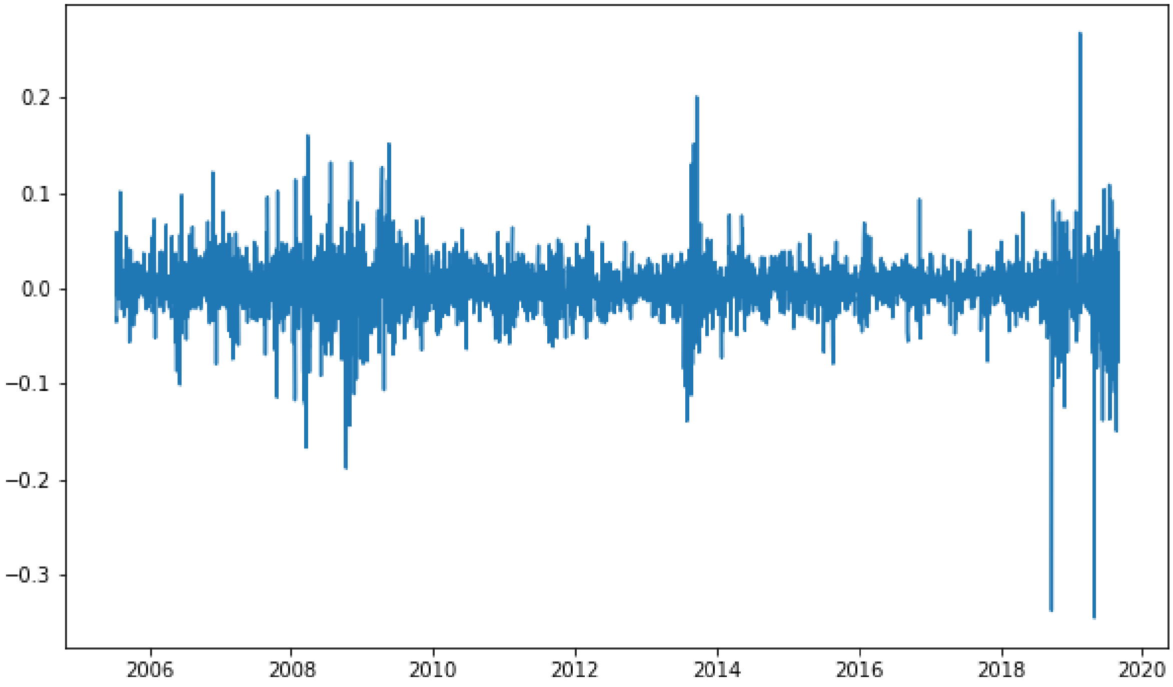


```
Results of Dickey-Fuller Test:
Test Statistics                 -1.030771
p-value                          0.741876
#Lags Used                       7.000000
Number of Observation Used    3496.000000
Critical Value (1%)             -3.432222
Critical Value (5%)             -2.862367
Critical Value (10%)            -2.567210
dtype: float64
```

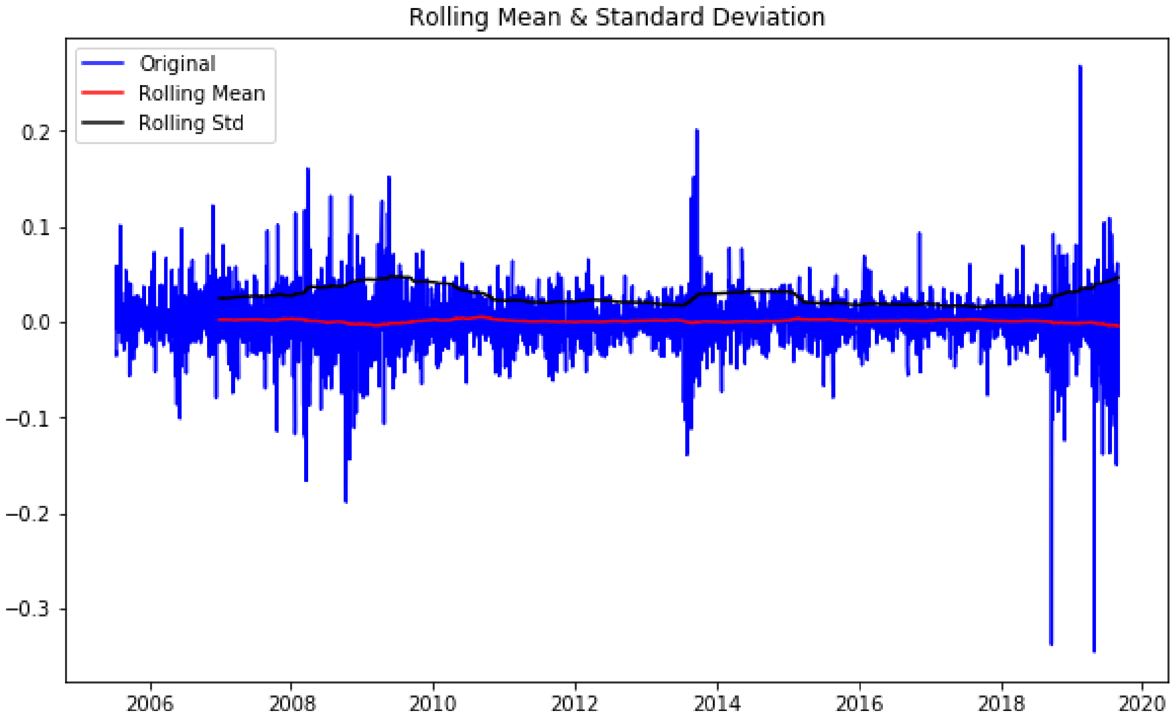
Log diff is nonthing but (d)

In [97]:
```
log_diff_shifting = YB_df_logscale - YB_df_logscale.shift()
plt.plot(log_diff_shifting)
```

Out[97]:
```
[<matplotlib.lines.Line2D at 0x116447f0>]
```



```
In [102]: log_diff_shifting.dropna(inplace=True)
          test_Stationarity(log_diff_shifting)
```

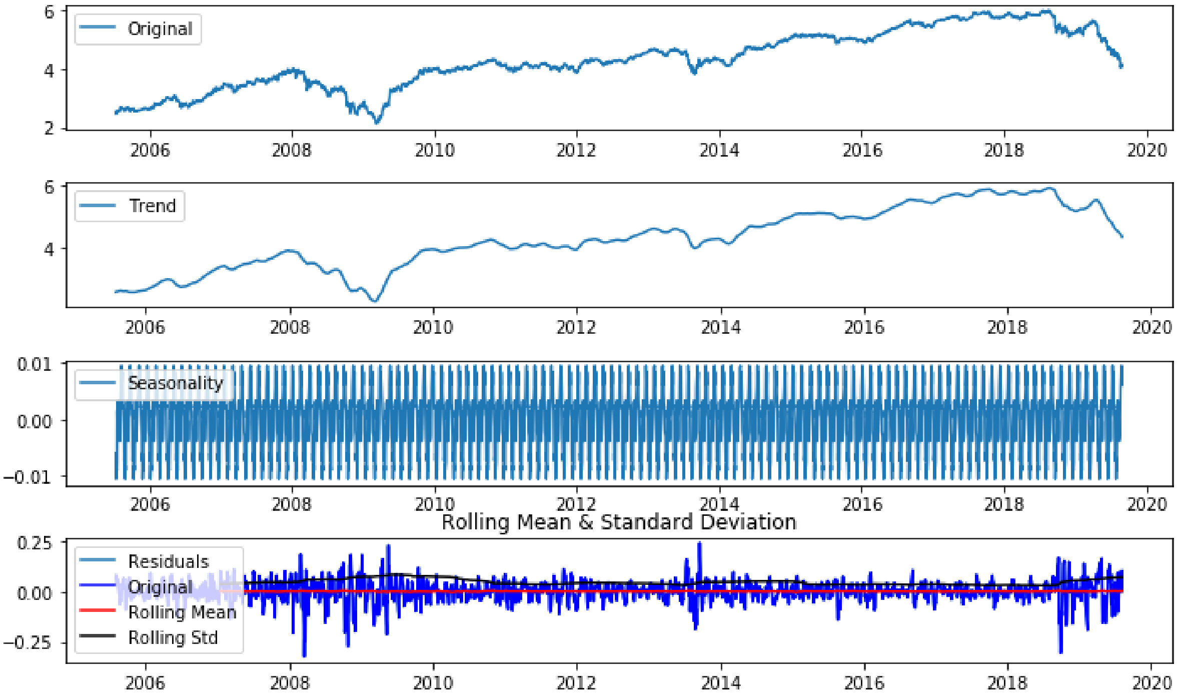


```
Results of Dickey-Fuller Test:
Test Statistics                 -21.898607
p-value                           0.000000
#Lags Used                        6.000000
Number of Observation Used     3496.000000
Critical Value (1%)              -3.432222
Critical Value (5%)              -2.862367
Critical Value (10%)             -2.567210
dtype: float64
```

Using Decompose function to find out Trend, Seasonal & Residual

In [111]:

```
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(YB_df_logscale, freq=30)

trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

plt.subplot(411)
plt.plot(YB_df_logscale, label='Original')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(trend, label='Trend')
plt.legend(loc='best')
plt.subplot(413)
plt.plot(seasonal, label='Seasonality')
plt.legend(loc='best')
plt.subplot(414)
plt.plot(residual, label='Residuals')
plt.legend(loc='best')
plt.tight_layout()

decomposedlogdata = residual
decomposedlogdata.dropna(inplace=True)
test_Stationarity(decomposedlogdata)
```

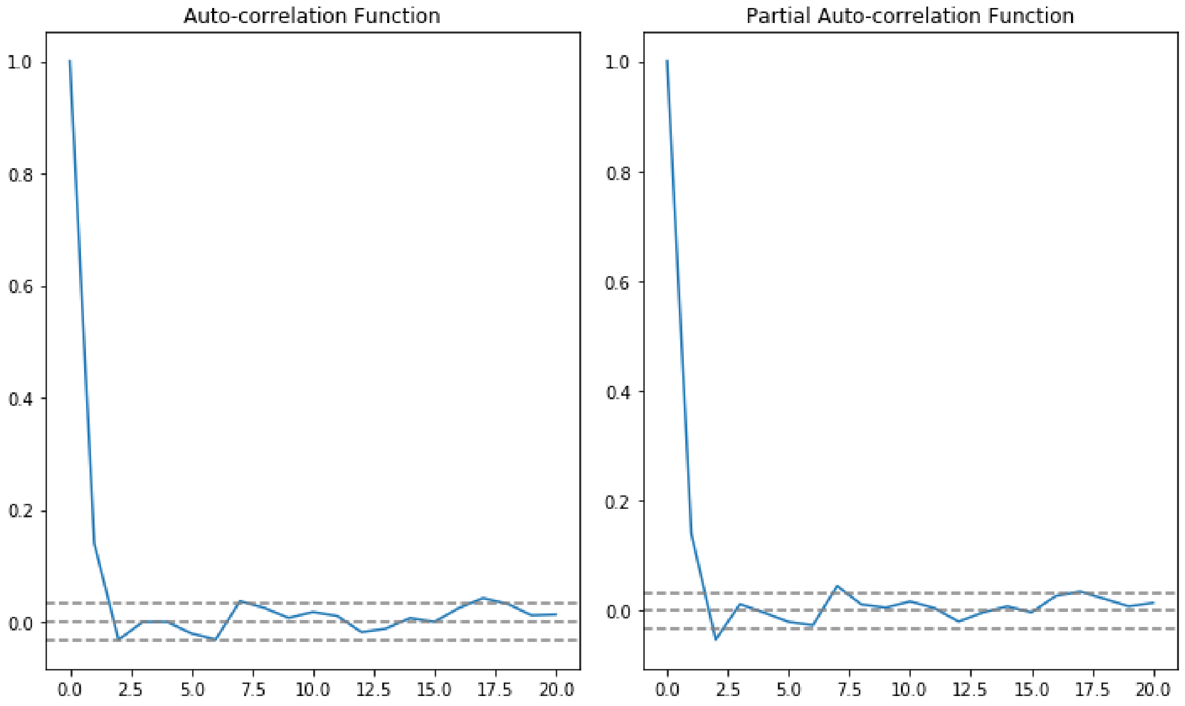


```
Results of Dickey-Fuller Test:
Test Statistics                 -1.535076e+01
p-value                          3.722924e-28
#Lags Used                       2.500000e+01
Number of Observation Used       3.448000e+03
Critical Value (1%)             -3.432248e+00
Critical Value (5%)             -2.862379e+00
Critical Value (10%)            -2.567216e+00
dtype: float64
```

ACF And PACF { Is nonthing but (p) & (q) }

In [113]:

```
from statsmodels.tsa.stattools import acf, pacf

lag_acf = acf(log_diff_shifting, nlags=20)
lag_pacf = pacf(log_diff_shifting, nlags=20, method='ols')

#plot ACF
plt.subplot(121)
plt.plot(lag_acf)
plt.axhline(y=0,linestyle='--',color='gray')
plt.axhline(y=-1.96/np.sqrt(len(log_diff_shifting)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(log_diff_shifting)),linestyle='--',color='gray')
plt.title('Auto-correlation Function')

#plot PACF
plt.subplot(122)
plt.plot(lag_pacf)
plt.axhline(y=0,linestyle='--',color='gray')
plt.axhline(y=-1.96/np.sqrt(len(log_diff_shifting)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(log_diff_shifting)),linestyle='--',color='gray')
plt.title('Partial Auto-correlation Function')
plt.tight_layout()
```

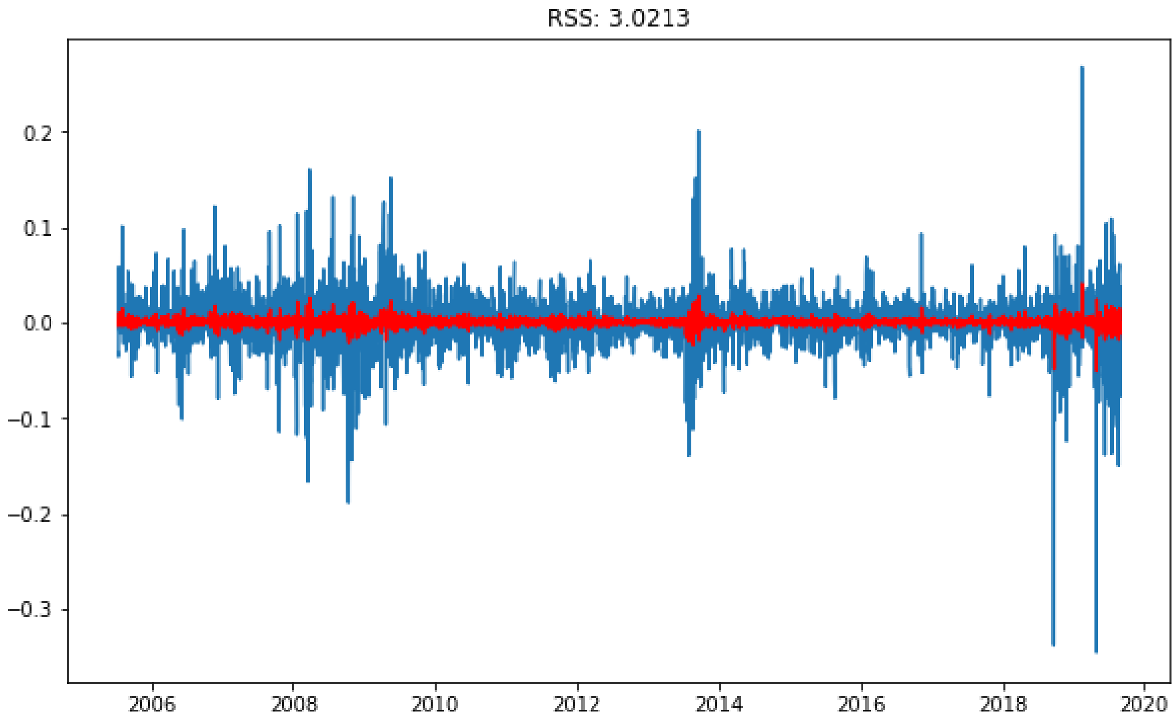


Finally Using ARIMA Model to Predict the 123 days

In [121]:

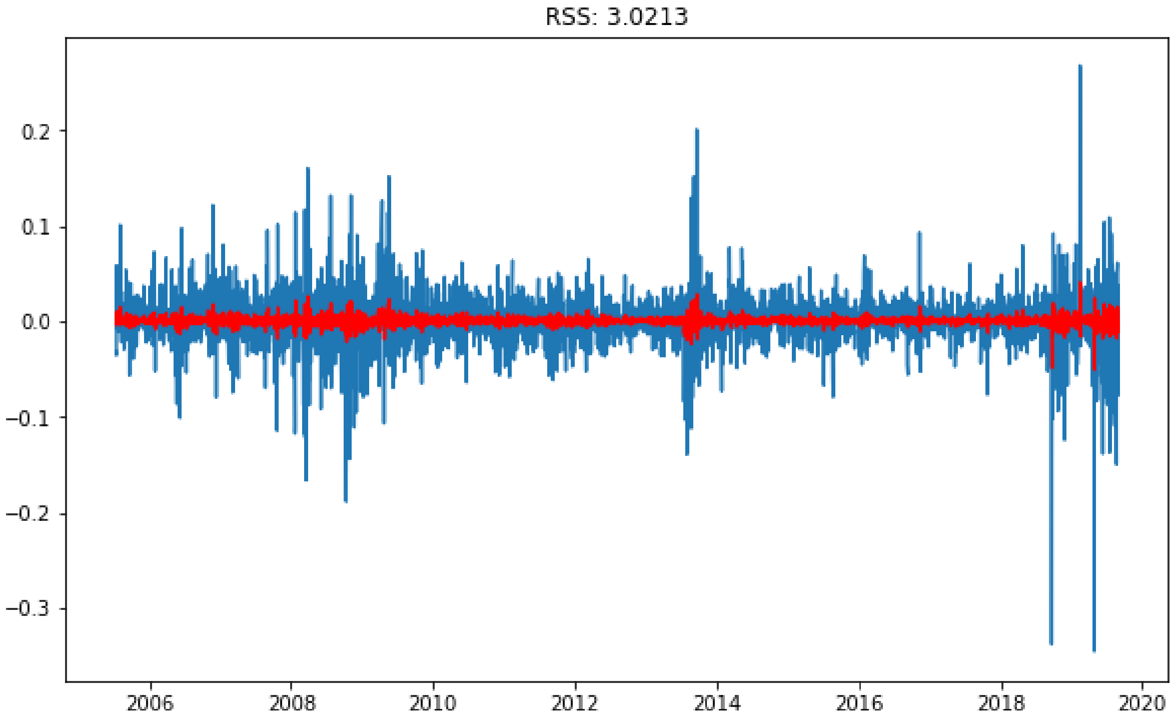
```
from statsmodels.tsa.arima_model import ARIMA

#AR model
model = ARIMA(YB_df_logscale, order=(2,1,2))
results_AR = model.fit(disp=-1)

plt.plot(log_diff_shifting)
plt.plot(results_AR.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_AR.fittedvalues-log_diff_shifting["Close P
rice"])**2)) # Residual sum of sq (RSS)
print('Plotting AR model')
```

```
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:
225: ValueWarning: A date index has been provided, but it has no associated f
requency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:
225: ValueWarning: A date index has been provided, but it has no associated f
requency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)

Plotting AR model
```



In [122]:
```
#MA model

model = ARIMA(YB_df_logscale, order=(2,1,2))
results_MA = model.fit(disp=-1)

plt.plot(log_diff_shifting)
plt.plot(results_MA.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_MA.fittedvalues-log_diff_shifting["Close P
rice"])**2)) # Residual sum of sq (RSS)
print('Plotting MA model')
```

```
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:
225: ValueWarning: A date index has been provided, but it has no associated f
requency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:
225: ValueWarning: A date index has been provided, but it has no associated f
requency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)

Plotting MA model
```



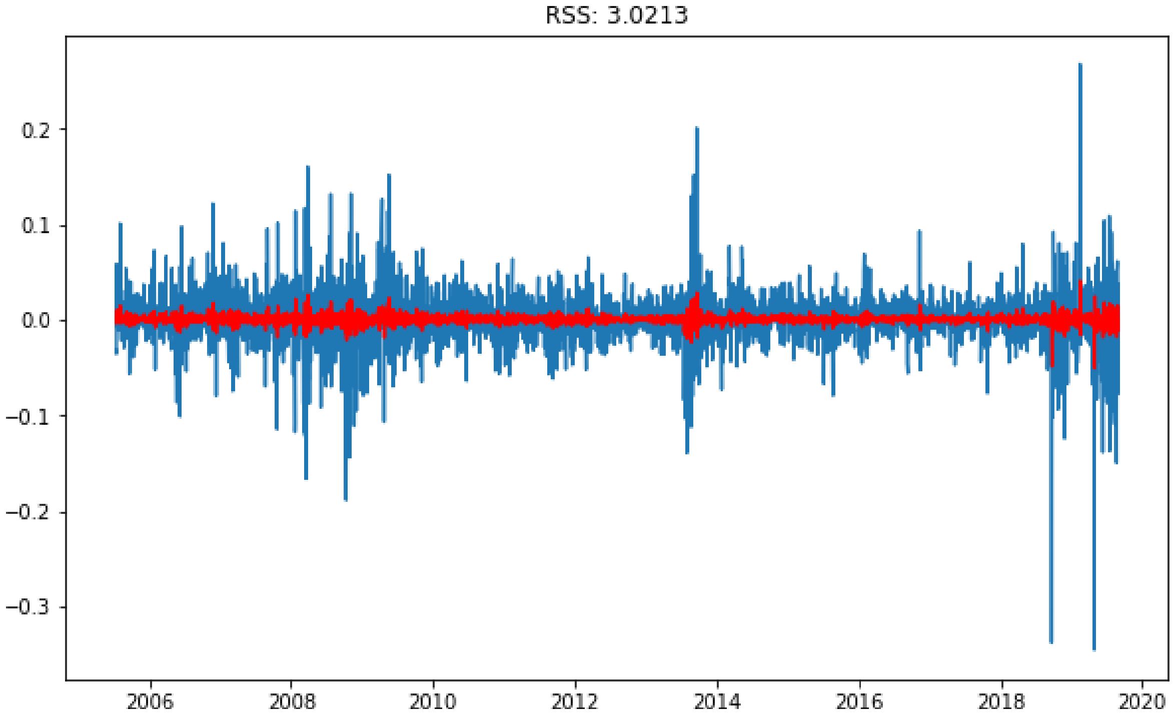
In [123]:

```
model = ARIMA(YB_df_logscale, order=(2,1,2))
results_ARIMA = model.fit(disp=-1)

plt.plot(log_diff_shifting)
plt.plot(results_ARIMA.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_ARIMA.fittedvalues-log_diff_shifting["Clos
e Price"])**2)) # Residual sum of sq (RSS)
```

```
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:
225: ValueWarning: A date index has been provided, but it has no associated f
requency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:
225: ValueWarning: A date index has been provided, but it has no associated f
requency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
```

Out[123]:

```
Text(0.5, 1.0, 'RSS: 3.0213')
```



In [126]:

```
# Now joining all together

predictions_ARIMA_diff = pd.Series(results_ARIMA.fittedvalues, copy=True)
print(predictions_ARIMA_diff.head())
```

```
Date
2005-07-13    0.000448
2005-07-14   -0.004095
2005-07-15   -0.003149
2005-07-18    0.010713
2005-07-19    0.003538
dtype: float64
```

In [128]:

```
# Convert to cumalative sum

predictions_ARIMA_diff_cumsum = predictions_ARIMA_diff.cumsum()
print(predictions_ARIMA_diff_cumsum.head())
```

```
Date
2005-07-13    0.000448
2005-07-14   -0.003647
2005-07-15   -0.006796
2005-07-18    0.003917
2005-07-19    0.007455
dtype: float64
```

In [129]:

```
predictions_ARIMA_log = pd.Series(YB_df_logscale['Close Price'].ix[0], index=Y
B_df_logscale.index)
predictions_ARIMA_log = predictions_ARIMA_log.add(predictions_ARIMA_diff_cumsu
m,fill_value=0)
predictions_ARIMA_log.head()
```

```
C:\Users\admin\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Deprecati
onWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-depre
cated
  """Entry point for launching an IPython kernel.
```

Out[129]:

```
Date
2005-07-12    2.519308
2005-07-13    2.519756
2005-07-14    2.515661
2005-07-15    2.512512
2005-07-18    2.523225
dtype: float64
```

In [ ]:

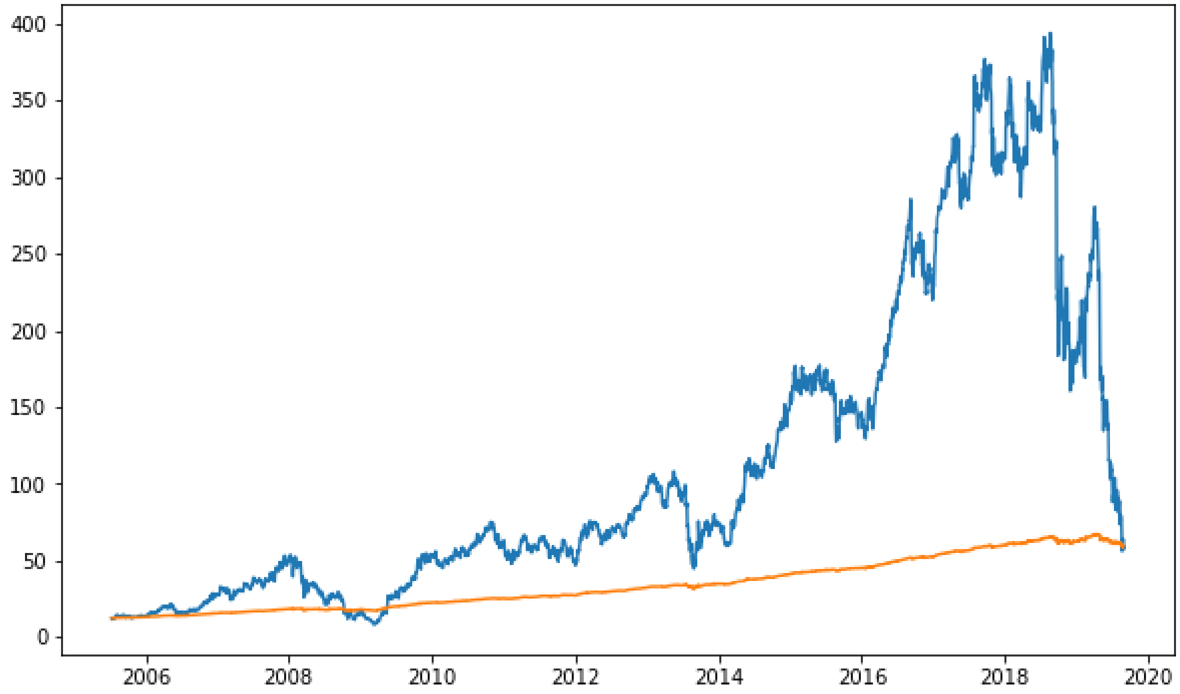
```
# converting data into original form
```

In [130]:

```
prediction_ARIMA = np.exp(predictions_ARIMA_log)
plt.plot(YB_df)
plt.plot(prediction_ARIMA)
```

Out[130]:

```
[<matplotlib.lines.Line2D at 0x14730da0>]
```



In [136]:

```
YB_df
```

Out[136]:

	Close Price
Date	
2005-07-12	12.42
2005-07-13	12.03
2005-07-14	11.61
2005-07-15	12.31
2005-07-18	12.88
2005-07-19	12.79
2005-07-20	12.73
2005-07-21	12.62
2005-07-22	12.50
2005-07-25	12.36
2005-07-26	12.51
2005-07-27	12.70
2005-07-29	12.60
2005-08-01	12.70
2005-08-02	14.05
2005-08-03	14.13
2005-08-04	14.41
2005-08-05	14.37
2005-08-08	14.26
2005-08-09	13.95
2005-08-10	14.39
2005-08-11	14.42
2005-08-12	14.23
2005-08-16	13.99
2005-08-17	13.95
2005-08-18	13.80
2005-08-19	13.63
2005-08-22	13.38
2005-08-23	12.99
2005-08-24	12.90
...	...
2019-07-18	85.80
2019-07-19	83.25
2019-07-22	91.15

	Close Price
Date	
2019-07-23	90.70
2019-07-24	89.15
2019-07-25	87.65
2019-07-26	96.10
2019-07-29	94.75
2019-07-30	86.10
2019-07-31	91.30
2019-08-01	88.40
2019-08-02	88.30
2019-08-05	81.10
2019-08-06	85.40
2019-08-07	86.85
2019-08-08	89.15
2019-08-09	82.10
2019-08-13	73.60
2019-08-14	76.55
2019-08-16	79.45
2019-08-19	76.70
2019-08-20	71.25
2019-08-21	65.40
2019-08-22	56.30
2019-08-23	59.25
2019-08-26	63.00
2019-08-27	64.30
2019-08-28	59.50
2019-08-29	57.35
2019-08-30	59.50

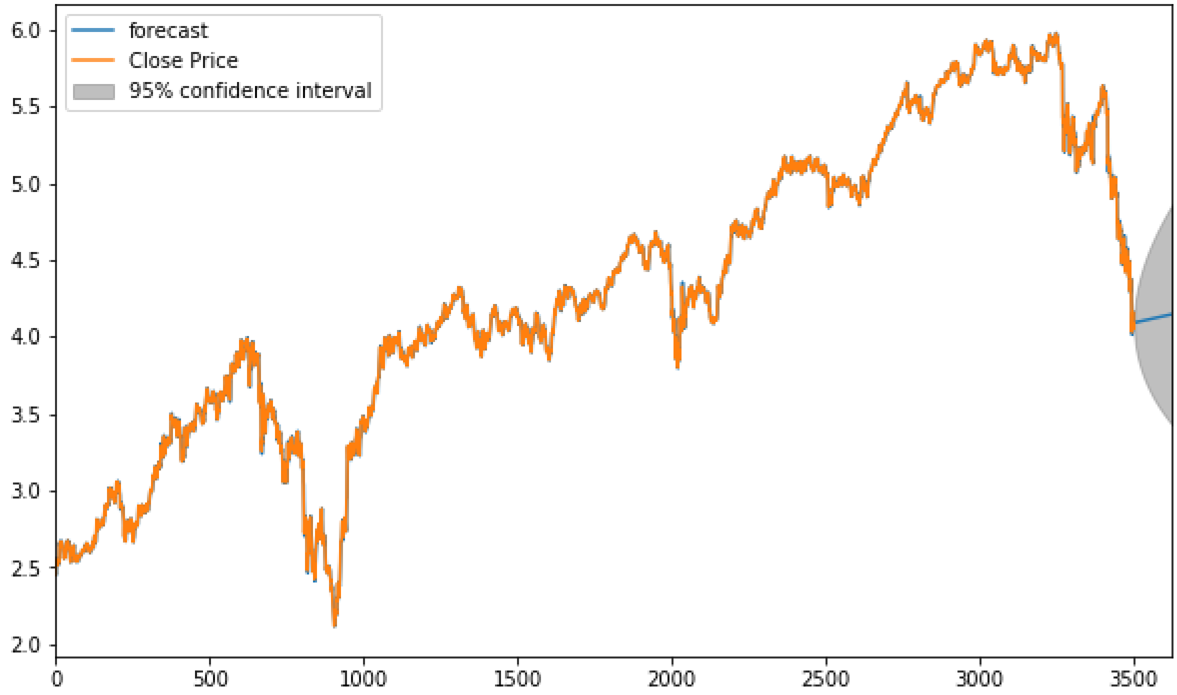
3504 rows × 1 columns

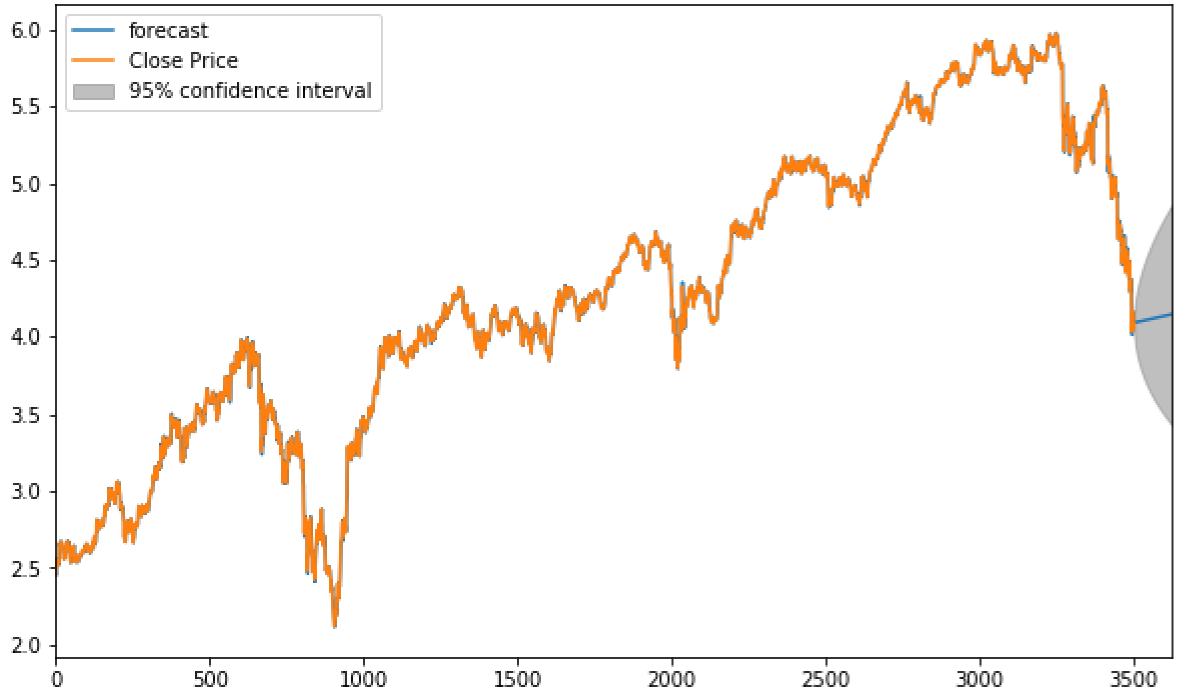
In [137]:

```
results_ARIMA.plot_predict(1,3627)
```

```
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:
531: ValueWarning: No supported index is available. Prediction results will b
e given with an integer index beginning at `start`.
  ValueWarning)
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:
531: ValueWarning: No supported index is available. Prediction results will b
e given with an integer index beginning at `start`.
  ValueWarning)
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:
531: ValueWarning: No supported index is available. Prediction results will b
e given with an integer index beginning at `start`.
  ValueWarning)
```

Out[137]:





In [134]:

```
results_ARIMA.forecast(steps=123)
```

```
Out[134]: (array([4.09303664, 4.09206857, 4.09238572, 4.09297463, 4.09339644,
                  4.09384104, 4.09429232, 4.09474016, 4.09518838, 4.09563677,
                  4.09608508, 4.0965334 , 4.09698172, 4.09743005, 4.09787837,
                  4.09832669, 4.09877502, 4.09922334, 4.09967166, 4.10011998,
                  4.10056831, 4.10101663, 4.10146495, 4.10191327, 4.1023616 ,
                  4.10280992, 4.10325824, 4.10370656, 4.10415489, 4.10460321,
                  4.10505153, 4.10549985, 4.10594818, 4.1063965 , 4.10684482,
                  4.10729314, 4.10774147, 4.10818979, 4.10863811, 4.10908643,
                  4.10953476, 4.10998308, 4.1104314 , 4.11087973, 4.11132805,
                  4.11177637, 4.11222469, 4.11267302, 4.11312134, 4.11356966,
                  4.11401798, 4.11446631, 4.11491463, 4.11536295, 4.11581127,
                  4.1162596 , 4.11670792, 4.11715624, 4.11760456, 4.11805289,
                  4.11850121, 4.11894953, 4.11939785, 4.11984618, 4.1202945 ,
                  4.12074282, 4.12119114, 4.12163947, 4.12208779, 4.12253611,
                  4.12298443, 4.12343276, 4.12388108, 4.1243294 , 4.12477773,
                  4.12522605, 4.12567437, 4.12612269, 4.12657102, 4.12701934,
                  4.12746766, 4.12791598, 4.12836431, 4.12881263, 4.12926095,
                  4.12970927, 4.1301576 , 4.13060592, 4.13105424, 4.13150256,
                  4.13195089, 4.13239921, 4.13284753, 4.13329585, 4.13374418,
                  4.1341925 , 4.13464082, 4.13508914, 4.13553747, 4.13598579,
                  4.13643411, 4.13688244, 4.13733076, 4.13777908, 4.1382274 ,
                  4.13867573, 4.13912405, 4.13957237, 4.14002069, 4.14046902,
                  4.14091734, 4.14136566, 4.14181398, 4.14226231, 4.14271063,
                  4.14315895, 4.14360727, 4.1440556 , 4.14450392, 4.14495224,
                  4.14540056, 4.14584889, 4.14629721]),
           array([0.02936799, 0.04472246, 0.05544031, 0.06435836, 0.07222504,
                  0.07930779, 0.08580701, 0.09184819, 0.09751571, 0.10287145,
                  0.10796183, 0.11282278, 0.11748278, 0.12196486, 0.12628796,
                  0.13046789, 0.13451801, 0.13844969, 0.14227276, 0.14599576,
                  0.14962615, 0.15317052, 0.1566347 , 0.16002391, 0.16334282,
                  0.16659561, 0.1697861 , 0.17291774, 0.17599366, 0.17901673,
                  0.18198959, 0.18491467, 0.18779419, 0.19063022, 0.19342467,
                  0.19617932, 0.19889582, 0.20157572, 0.20422045, 0.20683137,
                  0.20940974, 0.21195675, 0.21447351, 0.21696107, 0.21942044,
                  0.22185255, 0.22425828, 0.22663848, 0.22899393, 0.23132541,
                  0.23363362, 0.23591924, 0.23818294, 0.24042532, 0.24264698,
                  0.24484849, 0.24703037, 0.24919315, 0.25133732, 0.25346335,
                  0.2555717 , 0.2576628 , 0.25973706, 0.26179489, 0.26383666,
                  0.26586276, 0.26787354, 0.26986933, 0.27185047, 0.27381727,
                  0.27577005, 0.2777091 , 0.2796347 , 0.28154714, 0.28344667,
                  0.28533355, 0.28720804, 0.28907037, 0.29092079, 0.2927595 ,
                  0.29458674, 0.29640272, 0.29820764, 0.3000017 , 0.30178509,
                  0.30355801, 0.30532063, 0.30707313, 0.30881569, 0.31054847,
                  0.31227164, 0.31398535, 0.31568976, 0.31738501, 0.31907126,
                  0.32074864, 0.3224173 , 0.32407736, 0.32572896, 0.32737224,
                  0.3290073 , 0.33063428, 0.33225329, 0.33386445, 0.33546787,
                  0.33706367, 0.33865195, 0.34023281, 0.34180636, 0.3433727 ,
                  0.34493192, 0.34648413, 0.34802942, 0.34956788, 0.35109959,
                  0.35262465, 0.35414315, 0.35565516, 0.35716077, 0.35866006,
                  0.3601531 , 0.36163999, 0.36312078]),
           array([[4.03547644, 4.15059685],
                  [4.00441416, 4.17972298],
                  [3.98372471, 4.20104672],
                  [3.96683455, 4.21911471],
                  [3.95183796, 4.23495491],
                  [3.93840063, 4.24928145],
                  [3.92611367, 4.26247097],
                  [3.91472102, 4.2747593 ],
                  [3.9040611 , 4.28631567],
                  [3.89401244, 4.2972611 ],
                  [3.88448377, 4.30768639],
                  [3.87540481, 4.31766199],
                  [3.86671971, 4.32724374],
                  [3.85838332, 4.33647677],
                  [3.85035852, 4.34539822],
                  [3.84261432, 4.35403907],
                  [3.83512457, 4.36242546],
                  [3.82786693, 4.37057974],
                  [3.82082217, 4.37852115],
                  [3.81397355, 4.38626642],
                  [3.80730644, 4.39383017],
                  [3.80080793, 4.40122533],
                  [3.79446657, 4.40846333],
                  [3.78827216, 4.41555438],
                  [3.78221556, 4.42250763],
                  [3.77628851, 4.42933132],
                  [3.77048359, 4.43603289],
                  [3.76479402, 4.4426191 ],
                  [3.75921366, 4.44909611],
                  [3.75373687, 4.45546955],
                  [3.74835848, 4.46174458],
                  [3.74307376, 4.46792595],
                  [3.73787833, 4.47401803],
                  [3.73276813, 4.48002486],
                  [3.72773943, 4.48595021],
                  [3.72278874, 4.49179755],
                  [3.71791282, 4.49757012],
                  [3.71310864, 4.50327094],
                  [3.70837338, 4.50890285],
                  [3.70370439, 4.51446848],
                  [3.69909921, 4.51997031],
                  [3.69455549, 4.52541067],
                  [3.69007105, 4.53079175],
                  [3.68564383, 4.53611562],
                  [3.68127188, 4.54138421],
                  [3.67695336, 4.54659938],
                  [3.67268654, 4.55176284],
                  [3.66846976, 4.55687627],
                  [3.66430148, 4.5619412 ],
                  [3.66018019, 4.56695913],
                  [3.65610451, 4.57193146],
                  [3.65207308, 4.57685953],
                  [3.64808465, 4.58174461],
                  [3.64413798, 4.58658792],
                  [3.64023193, 4.59139062],
                  [3.63636538, 4.59615381],
                  [3.63253729, 4.60087855],
                  [3.62874664, 4.60556584],
                  [3.62499247, 4.61021666],
                  [3.62127384, 4.61483193],
                  [3.61758988, 4.61941254],
                  [3.61393973, 4.62395933],
                  [3.61032257, 4.62847314],
                  [3.60673763, 4.63295473],
                  [3.60318414, 4.63740486],
                  [3.59966138, 4.64182426],
                  [3.59616866, 4.64621363],
                  [3.59270531, 4.65057363],
                  [3.58927067, 4.65490491],
                  [3.58586412, 4.65920811],
                  [3.58248506, 4.66348381],
                  [3.57913292, 4.66773259],
                  [3.57580713, 4.67195503],
                  [3.57250716, 4.67615165],
                  [3.56923247, 4.68032298],
                  [3.56598256, 4.68446953],
                  [3.56275696, 4.68859178],
                  [3.55955517, 4.69269021],
                  [3.55637675, 4.69676528],
                  [3.55322126, 4.70081742],
                  [3.55008825, 4.70484707],
                  [3.54697733, 4.70885464],
                  [3.54388808, 4.71284053],
                  [3.54082011, 4.71680515],
                  [3.53777305, 4.72074886],
                  [3.53474652, 4.72467203],
                  [3.53174016, 4.72857503],
                  [3.52875364, 4.7324582 ],
                  [3.52578661, 4.73632187],
                  [3.52283874, 4.74016639],
                  [3.51990972, 4.74399205],
                  [3.51699923, 4.74779918],
                  [3.51410698, 4.75158808],
                  [3.51123267, 4.75535904],
                  [3.508376  , 4.75911235],
                  [3.50553672, 4.76284828],
                  [3.50271453, 4.76656711],
                  [3.49990919, 4.7702691 ],
                  [3.49712043, 4.77395451],
                  [3.494348  , 4.77762358],
                  [3.49159165, 4.78127657],
                  [3.48885116, 4.78491371],
                  [3.48612627, 4.78853524],
                  [3.48341678, 4.79214138],
                  [3.48072245, 4.79573235],
                  [3.47804307, 4.79930838],
                  [3.47537843, 4.80286966],
                  [3.47272832, 4.80641642],
                  [3.47009254, 4.80994884],
                  [3.4674709 , 4.81346713],
                  [3.46486319, 4.81697148],
                  [3.46226924, 4.82046208],
                  [3.45968886, 4.82393911],
                  [3.45712186, 4.82740275],
                  [3.45456808, 4.83085318],
                  [3.45202733, 4.83429057],
                  [3.44949946, 4.83771508],
                  [3.4469843 , 4.84112689],
                  [3.44448168, 4.84452616],
                  [3.44199145, 4.84791303],
                  [3.43951345, 4.85128768],
                  [3.43704753, 4.85465024],
                  [3.43459355, 4.85800087]]))
```

```
In [ ]:
```