In [2]:
```python
import pandas as pd
import numpy as np
import math
import os
from datetime import datetime as dt
from statsmodels.tsa.stattools import acf,pacf,adfuller
from statsmodels.tsa.arima_model import ARIMA
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib.pylab import rcParams
rcParams['figure.figsize']=(15,6)
import warnings
warnings.filterwarnings('ignore')
os.chdir('C:\\Users\\utpala mohapatra\\Documents\\python_folder\\python datasets')
```

In [7]:
```python
zinc_df = pd.read_csv('zinc_prices_IMF.csv')
zinc_df.head()
```

Out[7]:

	Date	Price
0	1-Jan-80	773.82
1	1-Feb-80	868.62
2	1-Mar-80	740.75
3	1-Apr-80	707.68
4	1-May-80	701.07

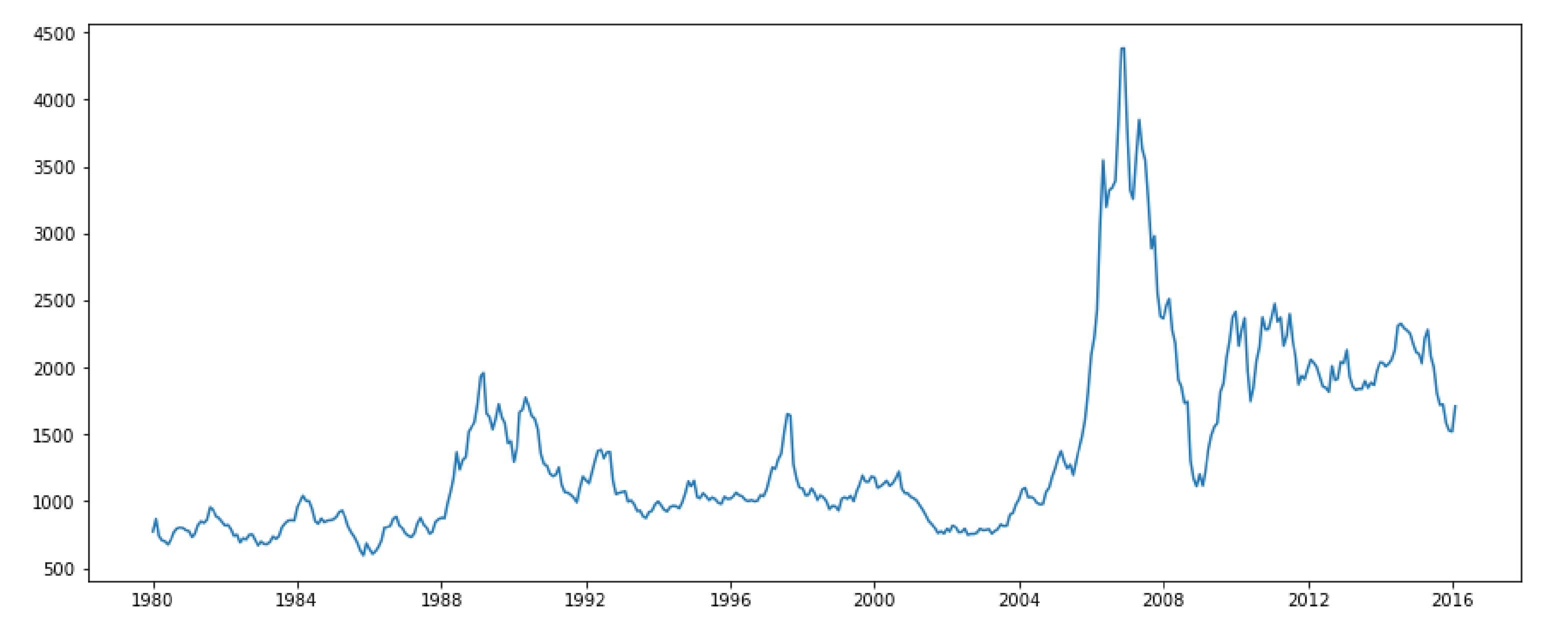
In [8]:
```python
zinc_df['Date']= pd.to_datetime(zinc_df.Date)
zinc_df.set_index('Date',inplace=True)

zinc_df.head()
```

Out[8]:

Date	Price
1980-01-01	773.82
1980-02-01	868.62
1980-03-01	740.75
1980-04-01	707.68
1980-05-01	701.07

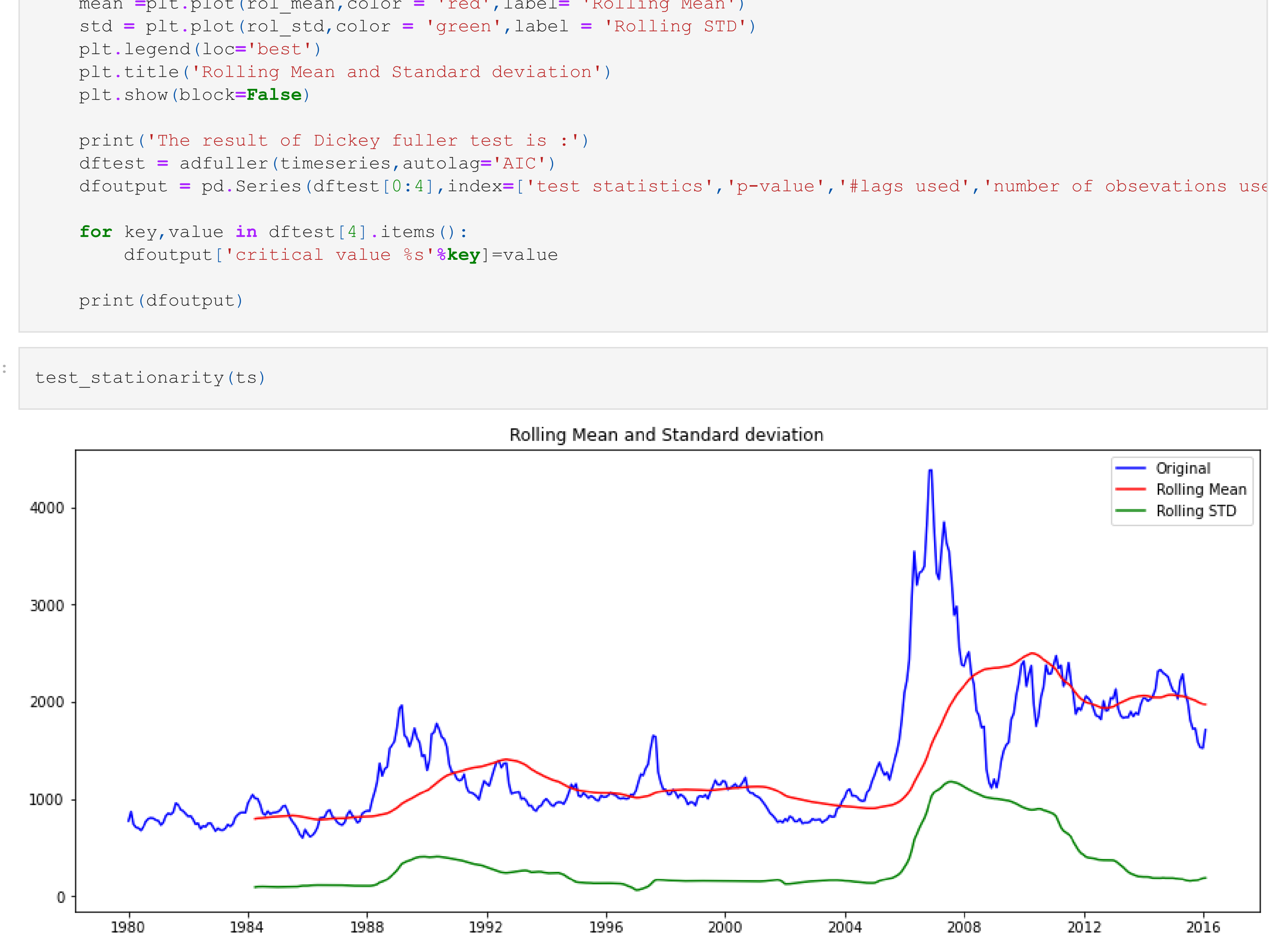
In [9]:
```python
ts = zinc_df['Price']
plt.plot(ts)
```

Out[9]: [<matplotlib.lines.Line2D at 0x2c320b0ec10>]



In [10]:
```python
def test_stationarity(timeseries):
    rol_mean = timeseries.rolling(window = 52,center = False).mean()
    rol_std = timeseries.rolling(window = 52,center = False).std()

    orig = plt.plot(timeseries,color = 'blue',label='Original')
    mean =plt.plot(rol_mean,color = 'red',label= 'Rolling Mean')
    std = plt.plot(rol_std,color = 'green',label = 'Rolling STD')
    plt.legend(loc='best')
    plt.title('Rolling Mean and Standard deviation')
    plt.show(block=False)

    print('The result of Dickey fuller test is :')
    dftest = adfuller(timeseries,autolag='AIC')
    dfoutput = pd.Series(dftest[0:4],index=['test statistics','p-value','#lags used','number of obsevations use

    for key,value in dftest[4].items():
        dfoutput['critical value %s'%key]=value

    print(dfoutput)
```

In [11]:
```python
test_stationarity(ts)
```

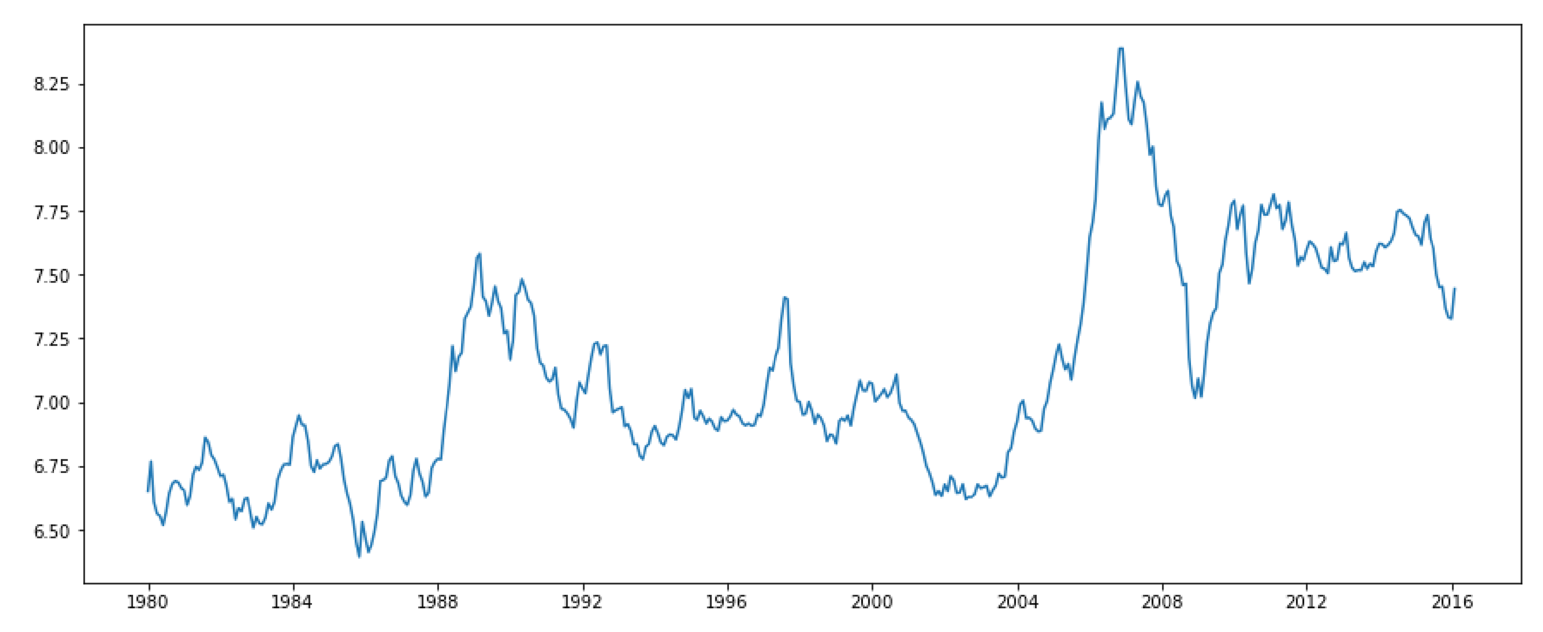



```
The result of Dickey fuller test is :
test statistics                -3.139601
p-value                         0.023758
#lags used                      7.000000
number of obsevations used    426.000000
critical value 1%              -3.445794
critical value 5%              -2.868349
critical value 10%             -2.570397
dtype: float64
```

In [ ]:
```python
# as the test statistic > critical value(%1) so cant reject null hypothesis: moving average is not constant
```

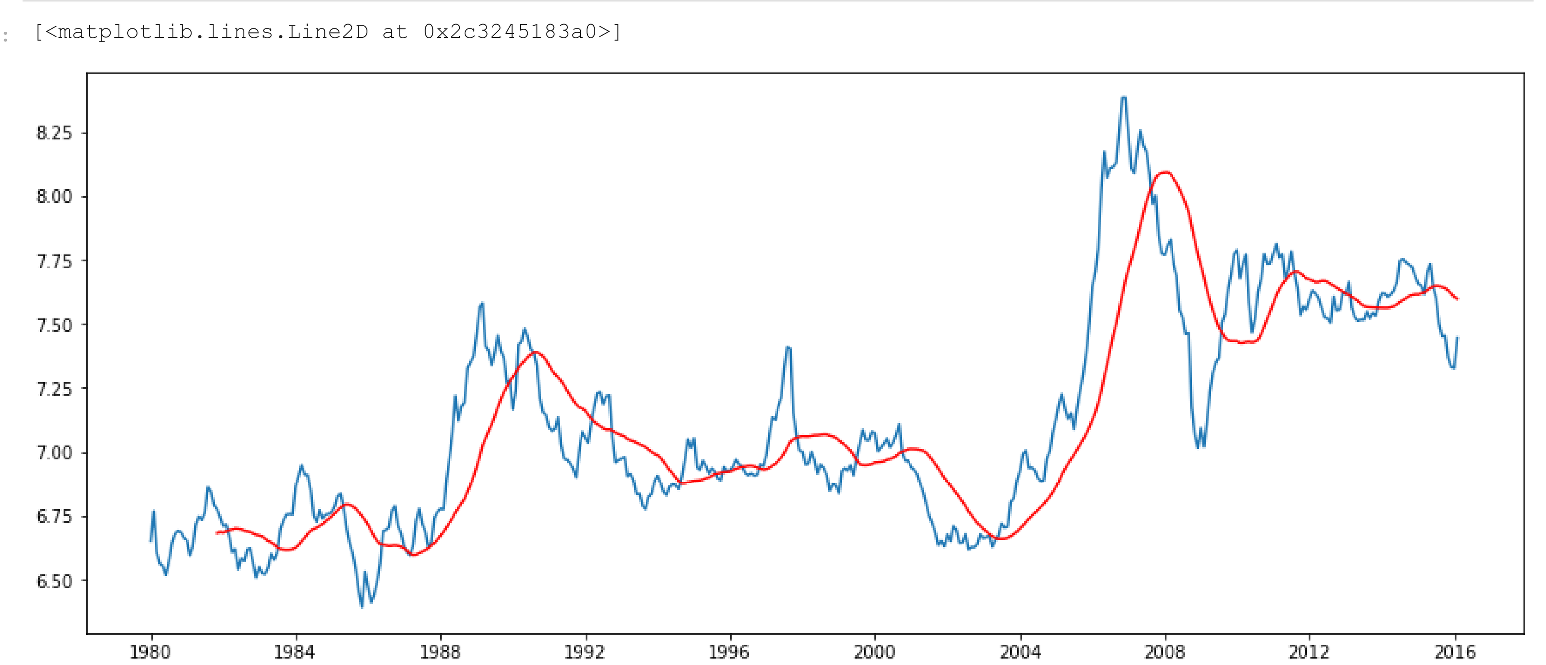
In [12]:
```python
# apply decomposition
ts_log = np.log(ts)
plt.plot(ts_log)
```

Out[12]: [<matplotlib.lines.Line2D at 0x2c324db4f10>]



In [14]:
```python
moving_avg = ts_log.rolling(window=23).mean()
moving_std = ts_log.rolling(window=23).std()

plt.plot(ts_log)
plt.plot(moving_avg,color = 'red')
```

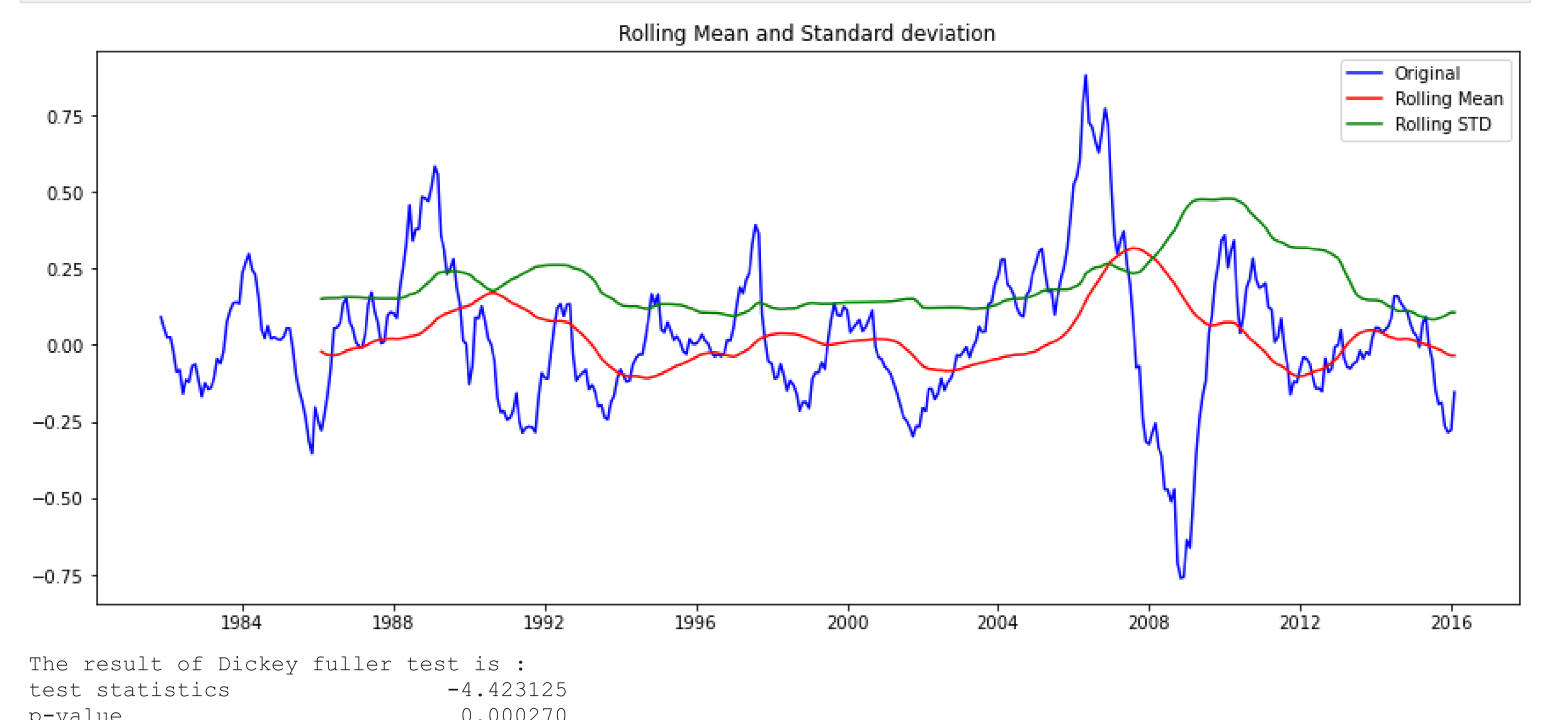
Out[14]: [<matplotlib.lines.Line2D at 0x2c3245183a0>]



In [15]:
```python
ts_log_mv_diff = ts_log - moving_avg
ts_log_mv_diff.dropna(inplace=True)
ts_log_mv_diff.head()
```

Out[15]:
```
Date
1981-11-01    0.092255
1981-12-01    0.057548
1982-01-01    0.025697
1982-02-01    0.026401
1982-03-01   -0.019398
Name: Price, dtype: float64
```

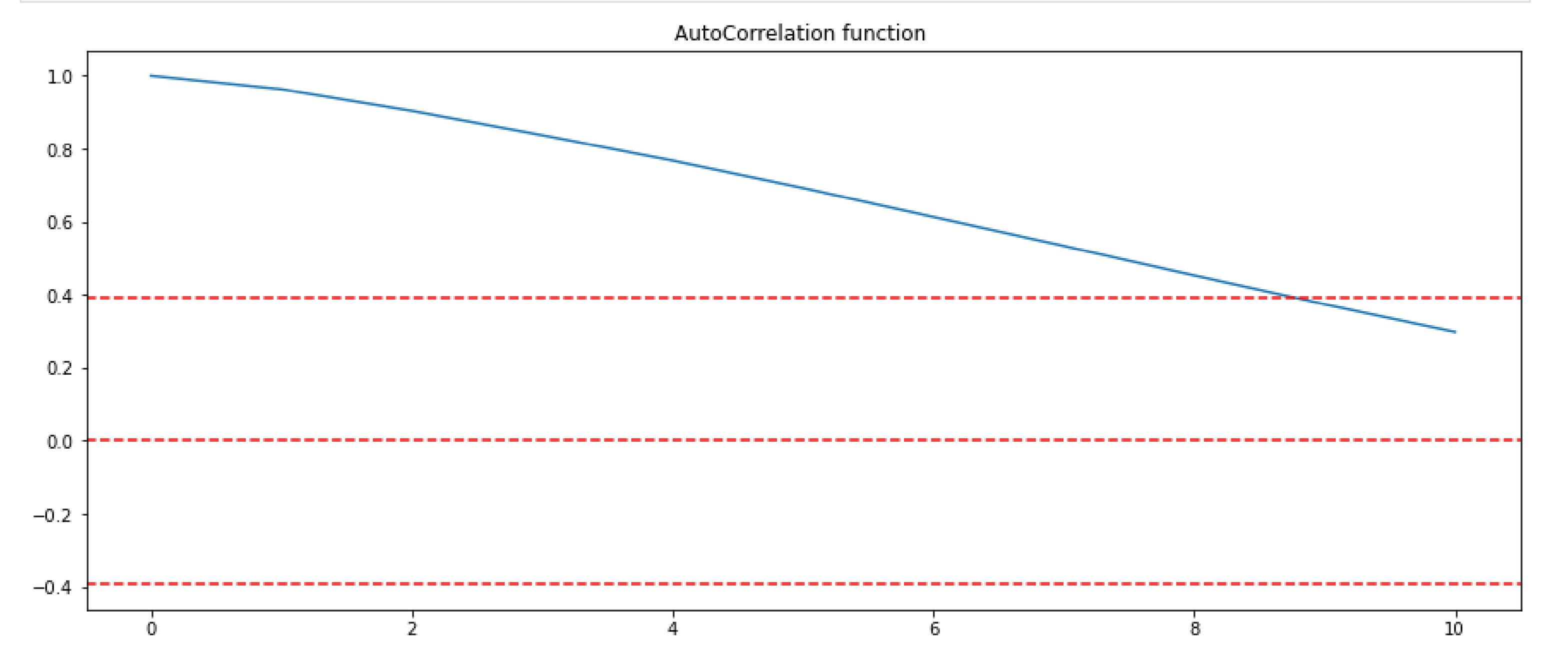
In [16]:
```python
test_stationarity(ts_log_mv_diff)
```



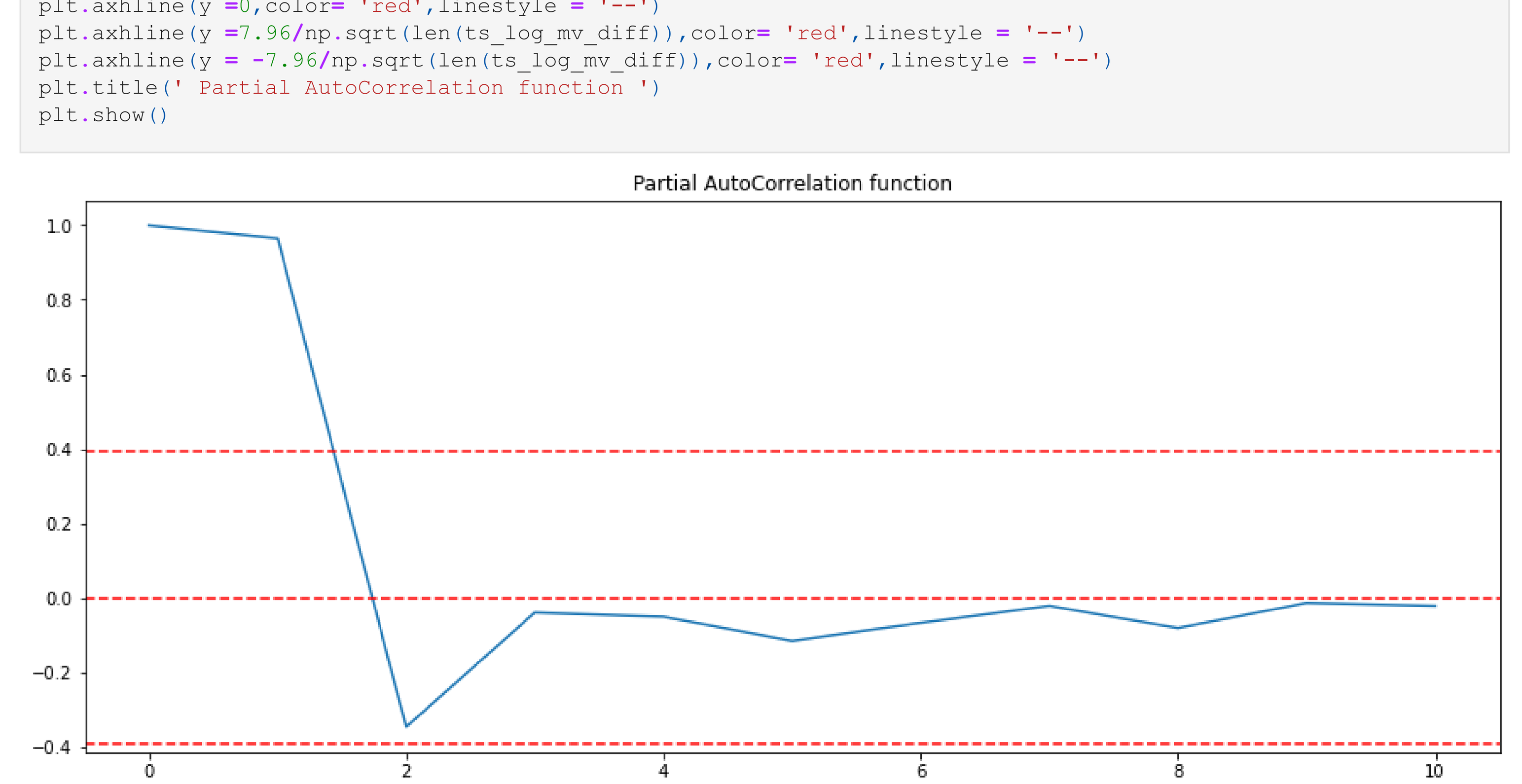

```
The result of Dickey fuller test is :
test statistics                -4.423125
p-value                         0.000270
#lags used                      4.000000
number of obsevations used    407.000000
critical value 1%              -3.446520
critical value 5%              -2.868668
critical value 10%             -2.570567
dtype: float64
```

In [ ]:
```python
# now the test statistic < critical value: data is little more stationary
```

In [19]:
```python
# plot acf
plt.plot(np.arange(0,11),acf(ts_log_mv_diff,nlags=10))
plt.axhline(y =0,color= 'red',linestyle = '--')
plt.axhline(y =7.96/np.sqrt(len(ts_log_mv_diff)),color= 'red',linestyle = '--')
plt.axhline(y = -7.96/np.sqrt(len(ts_log_mv_diff)),color= 'red',linestyle = '--')
plt.title('AutoCorrelation function ')
plt.show()
```



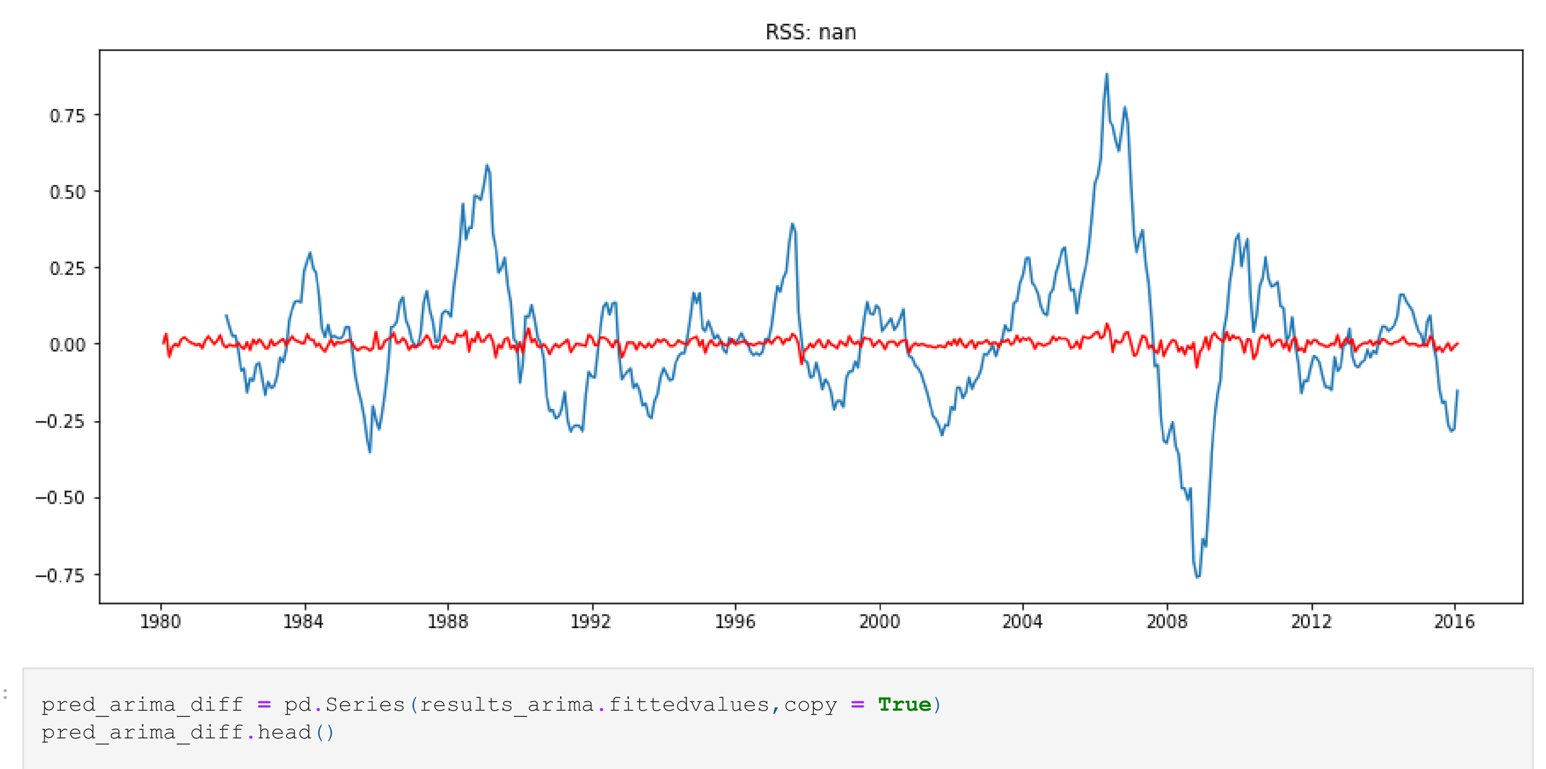
In [20]:
```python
# plot pacf
plt.plot(np.arange(0,11),pacf(ts_log_mv_diff,nlags=10))
plt.axhline(y =0,color= 'red',linestyle = '--')
plt.axhline(y =7.96/np.sqrt(len(ts_log_mv_diff)),color= 'red',linestyle = '--')
plt.axhline(y = -7.96/np.sqrt(len(ts_log_mv_diff)),color= 'red',linestyle = '--')
plt.title(' Partial AutoCorrelation function ')
plt.show()
```



In [ ]:
```python
# The pcaf curve drops to zero between lag values 1 and 2 so the p value for ARIMA madel can be 1 or 2
```

In [25]:
```python
model = ARIMA(ts_log,order=(1,1,0))
results_arima = model.fit(disp=-1)

plt.plot(ts_log_mv_diff)
plt.plot(results_arima.fittedvalues,color = 'red')

plt.title('RSS: %.4f'%sum((results_arima.fittedvalues[1:] - ts_log_mv_diff)**2))
```

Out[25]: Text(0.5, 1.0, 'RSS: nan')



In [26]:
```python
pred_arima_diff = pd.Series(results_arima.fittedvalues,copy = True)
pred_arima_diff.head()
```

Out[26]:
```
Date
1980-02-01    0.002030
1980-03-01    0.033049
1980-04-01   -0.042031
1980-05-01   -0.011002
1980-06-01   -0.001089
dtype: float64
```

In [27]:
```python
pred_arima_diff_cumsum = pred_arima_diff.cumsum()
pred_arima_diff_cumsum.head()
```

Out[27]:
```
Date
1980-02-01    0.002030
1980-03-01    0.035079
1980-04-01   -0.006952
1980-05-01   -0.017955
1980-06-01   -0.019043
dtype: float64
```

In [28]:
```python
pred_arima_log = pd.Series(ts_log,index= ts_log.index)
pred_arima_log = pred_arima_log.add(pred_arima_diff_cumsum,fill_value = 0)
pred_arima_log.head()
```

Out[28]:
```
Date
1980-01-01    6.651339
1980-02-01    6.768936
1980-03-01    6.642742
1980-04-01    6.555040
1980-05-01    6.534653
dtype: float64
```

In [30]:
```python
pred_arima = np.exp(pred_arima_log)
plt.plot(ts)
plt.plot(pred_arima,color = 'red')
plt.title('RMSE: %.4f'%np.sqrt(sum((pred_arima - ts)**2)/len(ts)))
```

Out[30]: Text(0.5, 1.0, 'RMSE: 1718.5568')