In [2]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
import matplotlib
```

In [14]:
```
Amazon_dataset = pd.read_csv('Amazon.csv',index_col=[0], parse_dates=[0])
Amazon_dataset.head()
```

Out[14]:

Date	Open	High	Low	Close	AdjClose	Volume
1997-05-15	2.437500	2.500000	1.927083	1.958333	1.958333	72156000
1997-05-16	1.968750	1.979167	1.708333	1.729167	1.729167	14700000
1997-05-19	1.760417	1.770833	1.625000	1.708333	1.708333	6106800
1997-05-20	1.729167	1.750000	1.635417	1.635417	1.635417	5467200
1997-05-21	1.635417	1.645833	1.375000	1.427083	1.427083	18853200

In [15]:
```
Amazon_dataset.tail()
```

Out[15]:

Date	Open	High	Low	Close	AdjClose	Volume
2020-07-27	3062.00000	3098.000000	3015.77002	3055.209961	3055.209961	4170500
2020-07-28	3054.27002	3077.090088	2995.76001	3000.330078	3000.330078	3126700
2020-07-29	3030.98999	3039.159912	2996.77002	3033.530029	3033.530029	2974100
2020-07-30	3014.00000	3092.000000	3005.00000	3051.879883	3051.879883	6128300
2020-07-31	3244.00000	3246.820068	3151.00000	3164.679932	3164.679932	8085500

In [16]:
```
Amazon_dataset.describe()
```

Out[16]:

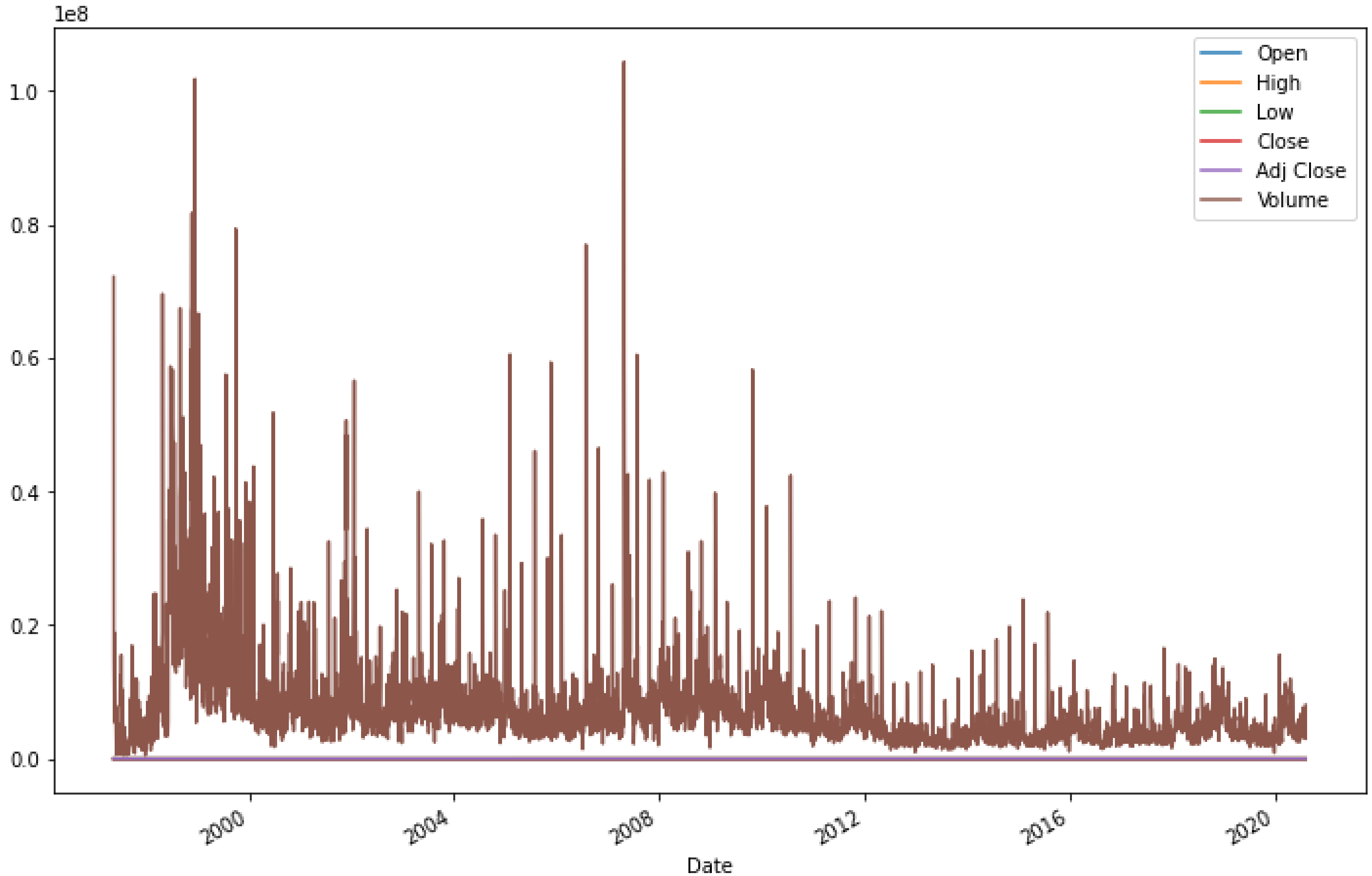
	Open	High	Low	Close	AdjClose	Volume
count	5842.000000	5842.000000	5842.000000	5842.000000	5842.000000	5.842000e+03
mean	372.707174	376.921392	368.114569	372.746660	372.746660	7.519048e+06
std	585.571802	591.766458	578.660700	585.607655	585.607655	7.282683e+06
min	1.406250	1.447917	1.312500	1.395833	1.395833	4.872000e+05
25%	37.955001	38.547501	37.207500	37.927499	37.927499	3.684900e+06
50%	83.428749	84.945000	81.656250	83.459999	83.459999	5.657200e+06
75%	359.729988	363.439987	356.280006	360.047501	360.047501	8.533400e+06
max	3251.060059	3344.290039	3151.000000	3200.000000	3200.000000	1.043292e+08

In [5]:

```
Amazon_dataset.plot(figsize=(12,8))
```

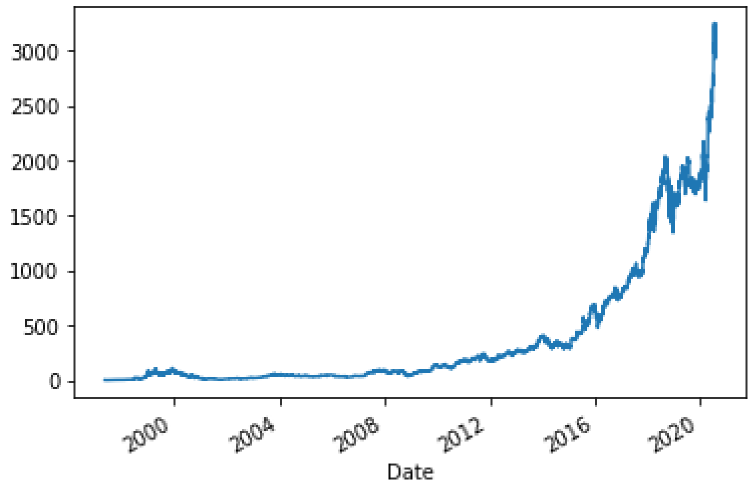
Out[5]:

```
<AxesSubplot:xlabel='Date'>
```

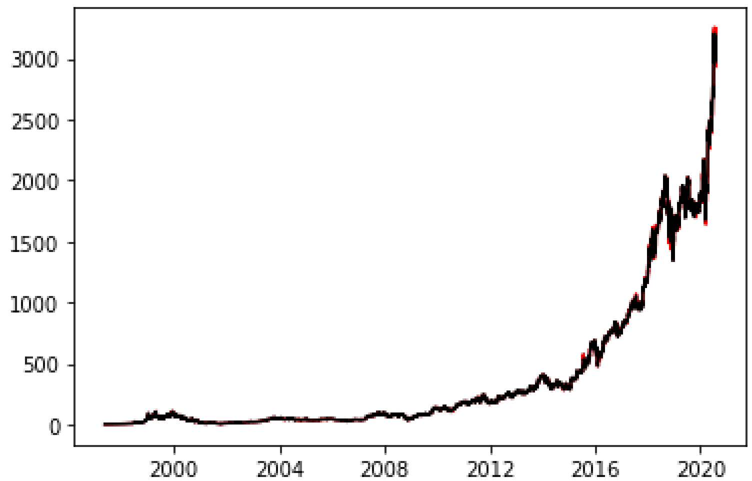


In [7]:
```
Amazon_dataset.Open.plot()
```

Out[7]: <AxesSubplot:xlabel='Date'>



In [9]:
```
plt.plot(Amazon_dataset.Open, label='OPEN', color='red')
plt.plot(Amazon_dataset.Close, label ='CLOSE', color='black')
plt.show()
```



In [17]:
```
amazon_close_df = Amazon_dataset.drop(['Open','High','Low','AdjClose','Volume'], axis=1)
amazon_close_df.head()
```

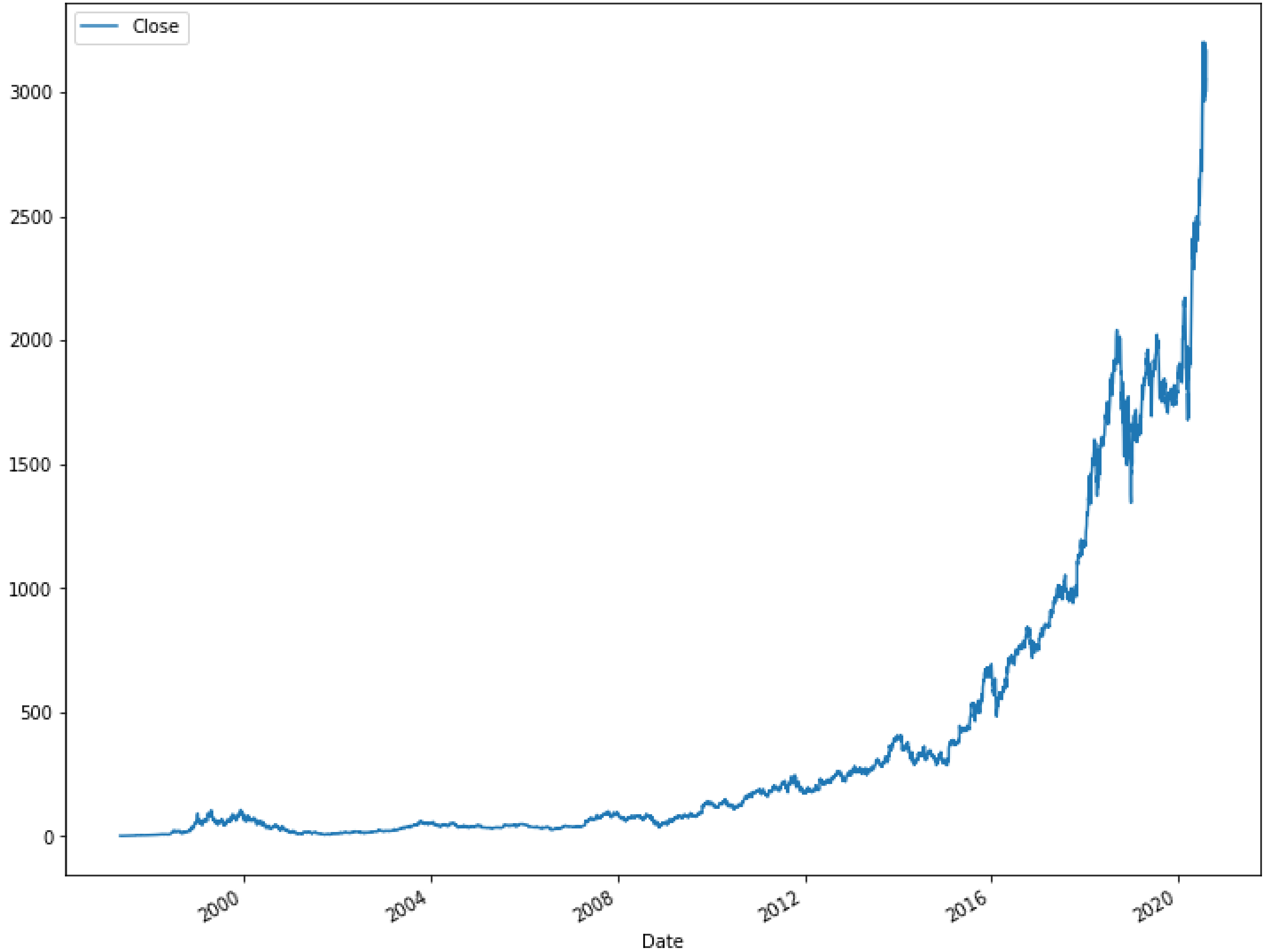
Out[17]:

Close

	Close
Date	
1997-05-15	1.958333
1997-05-16	1.729167
1997-05-19	1.708333
1997-05-20	1.635417
1997-05-21	1.427083

In [18]:
```
amazon_close_df.plot(figsize=(12,10))
```

Out[18]: <AxesSubplot:xlabel='Date'>



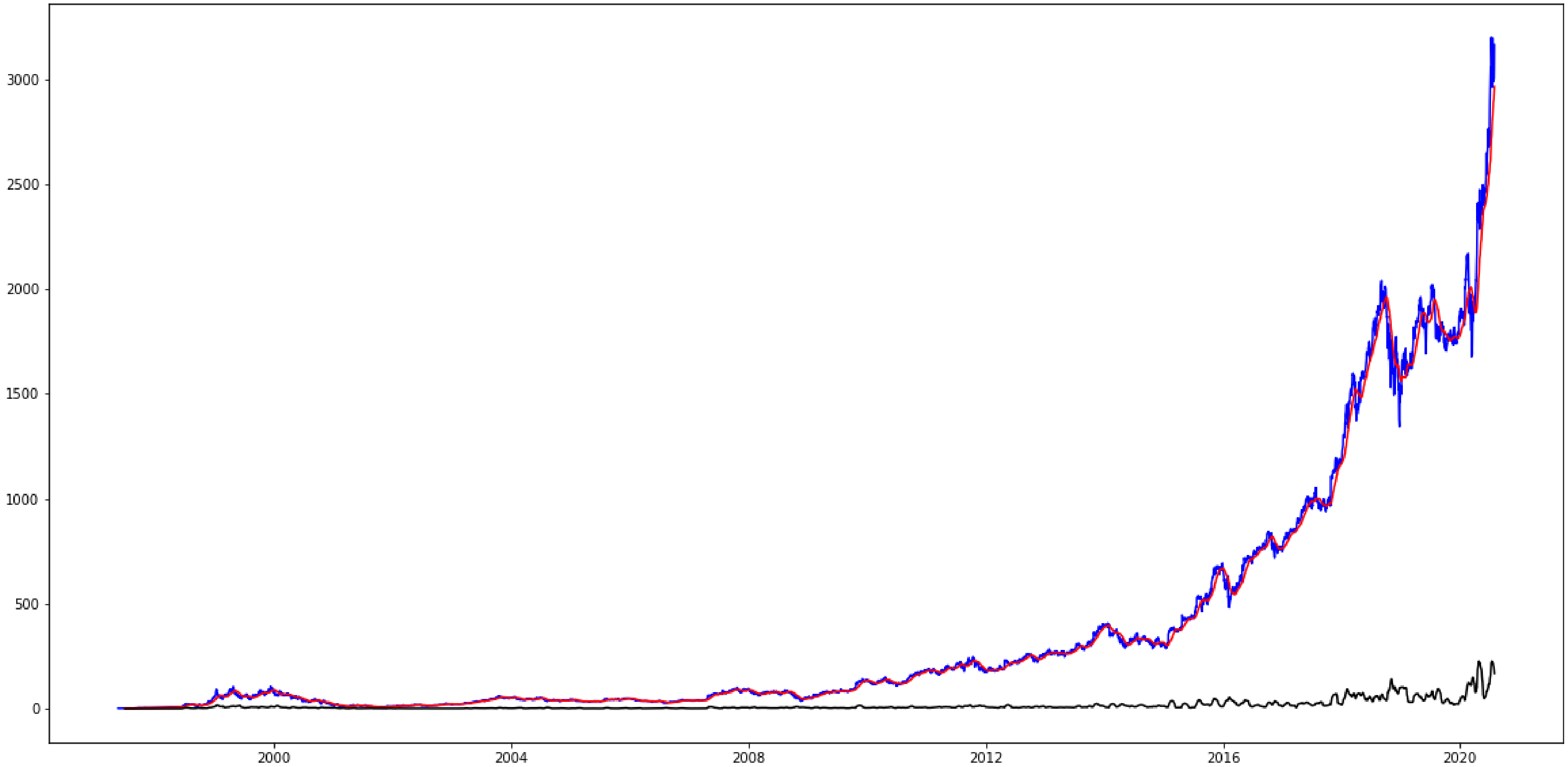
In [20]:

```
rol_mean = amazon_close_df.rolling(window=30).mean()
rol_std = amazon_close_df.rolling(window=30).std()
```

In [27]:

```
plt.figure(figsize=(20,10))
plt.plot(amazon_close_df, color='blue',label='Close Value')
plt.plot(rol_mean,color='red',label='Moving Average')
```

```
plt.plot(rol_std,color='black',label='Moving STD')
plt.show()
```



In [31]:

```
from statsmodels.tsa.stattools import adfuller

print('RESULTS OF ADF TEST')
dftest = adfuller(amazon_close_df.Close, autolag='AIC')

dfout = pd.Series(dftest[0:4], index=['TEST STATISTIC','P-VALUE','LAGS USED', 'NUMBER OF OBS'])
for key,value in dftest[4].items():
    dfout['CRITICAL VALUE %s'%key] = value
```

```
print(dfout)
```

```
RESULTS OF ADF TEST
TEST STATISTIC            5.618547
P-VALUE                   1.000000
LAGS USED                34.000000
NUMBER OF OBS          5807.000000
CRITICAL VALUE 1%        -3.431477
CRITICAL VALUE 5%        -2.862038
CRITICAL VALUE 10%       -2.567035
dtype: float64
```

In [32]:

```
#taking the log
df_log = np.log(amazon_close_df)
df_log.head()
```

Out[32]:

Date	Close
1997-05-15	0.672094
1997-05-16	0.547640
1997-05-19	0.535518
1997-05-20	0.491898
1997-05-21	0.355633

In [34]:

```
df_log.tail(6)
```

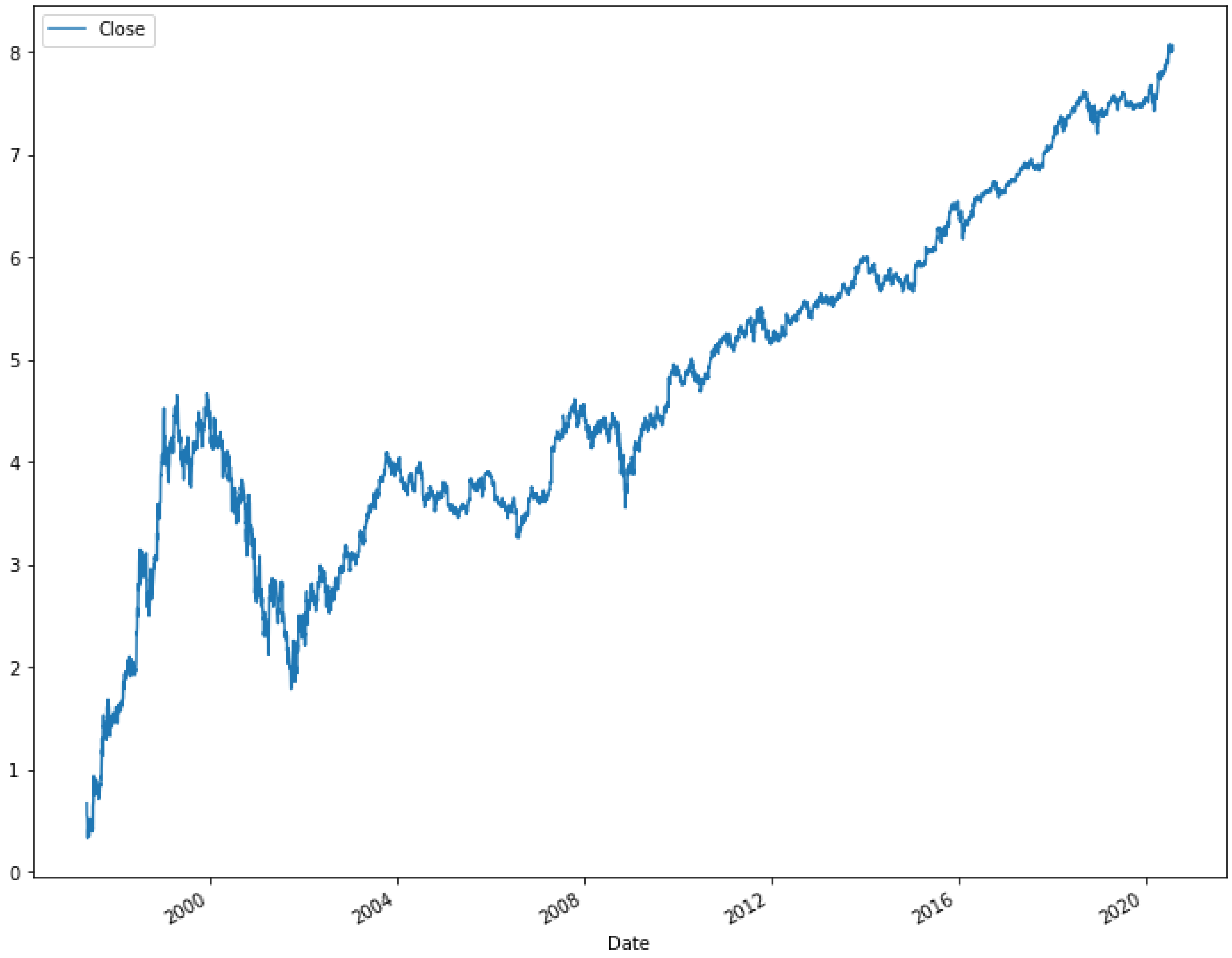
Out[34]:

Date	Close
2020-07-24	8.009333
2020-07-27	8.024604
2020-07-28	8.006478
2020-07-29	8.017482
2020-07-30	8.023513

	Close
Date	
2020-07-31	8.059807

In [36]:
```
df_log.plot(figsize=(12,10))
```

Out[36]: <AxesSubplot:xlabel='Date'>



In [37]:
```
series_log =df_log.values
series_log
```

Out[37]:
```
array([[0.67209379],
       [0.54763957],
       [0.53551826],
       ...,
```

```
       [8.01748225],
       [8.02351303],
       [8.0598072 ]])
```

In [38]:
```
value_df = pd.DataFrame(series_log)
value_df.head()
```

Out[38]:

	0
0	0.672094
1	0.547640
2	0.535518
3	0.491898
4	0.355633

In [39]:
```
df_base = pd.concat([value_df, value_df.shift(1)], axis=1)
df_base.head()
```

Out[39]:

	0	0
0	0.672094	NaN
1	0.547640	0.672094
2	0.535518	0.547640
3	0.491898	0.535518
4	0.355633	0.491898

In [40]:
```
df_base.columns=['Actual','Forecast']
df_base.head()
```

Out[40]:

	Actual	Forecast
0	0.672094	NaN
1	0.547640	0.672094
2	0.535518	0.547640

	Actual	Forecast
3	0.491898	0.535518
4	0.355633	0.491898

In [41]:
```
df_base = df_base[1:]
df_base.head()
```

Out[41]:

	Actual	Forecast
1	0.547640	0.672094
2	0.535518	0.547640
3	0.491898	0.535518
4	0.355633	0.491898
5	0.333492	0.355633

In [42]:
```
from sklearn.metrics import mean_squared_error
import numpy as np
```

In [43]:
```
base_error = mean_squared_error(df_base.Actual, df_base.Forecast)
base_error
```

Out[43]: 0.0013476612979921394

In [44]:
```
np.sqrt(base_error)
```

Out[44]: 0.03671050664308707

In [45]:
```
#Checking on the log transformed dataset
mov_mean = df_log.rolling(window=30).mean()
mov_std = df_log.rolling(window=30).std()

plt.figure(figsize=(20,10))
plt.plot(df_log, color='blue',label='Close Value')
plt.plot(mov_mean,color='red',label='Moving Average')
```

```
plt.plot(mov_std,color='black',label='Moving STD')
plt.show()
```

In [46]:

```
dflog2 = df_log - mov_mean

dflog2.dropna(inplace=True)
dflog2.head(10)
```

Out[46]:

	Close
Date	

	Close
Date	
1997-06-26	-0.028650
1997-06-27	-0.033419
1997-06-30	0.004775
1997-07-01	-0.008272
1997-07-02	0.039686
1997-07-03	0.214995
1997-07-07	0.248287
1997-07-08	0.374676
1997-07-09	0.370909
1997-07-10	0.451905

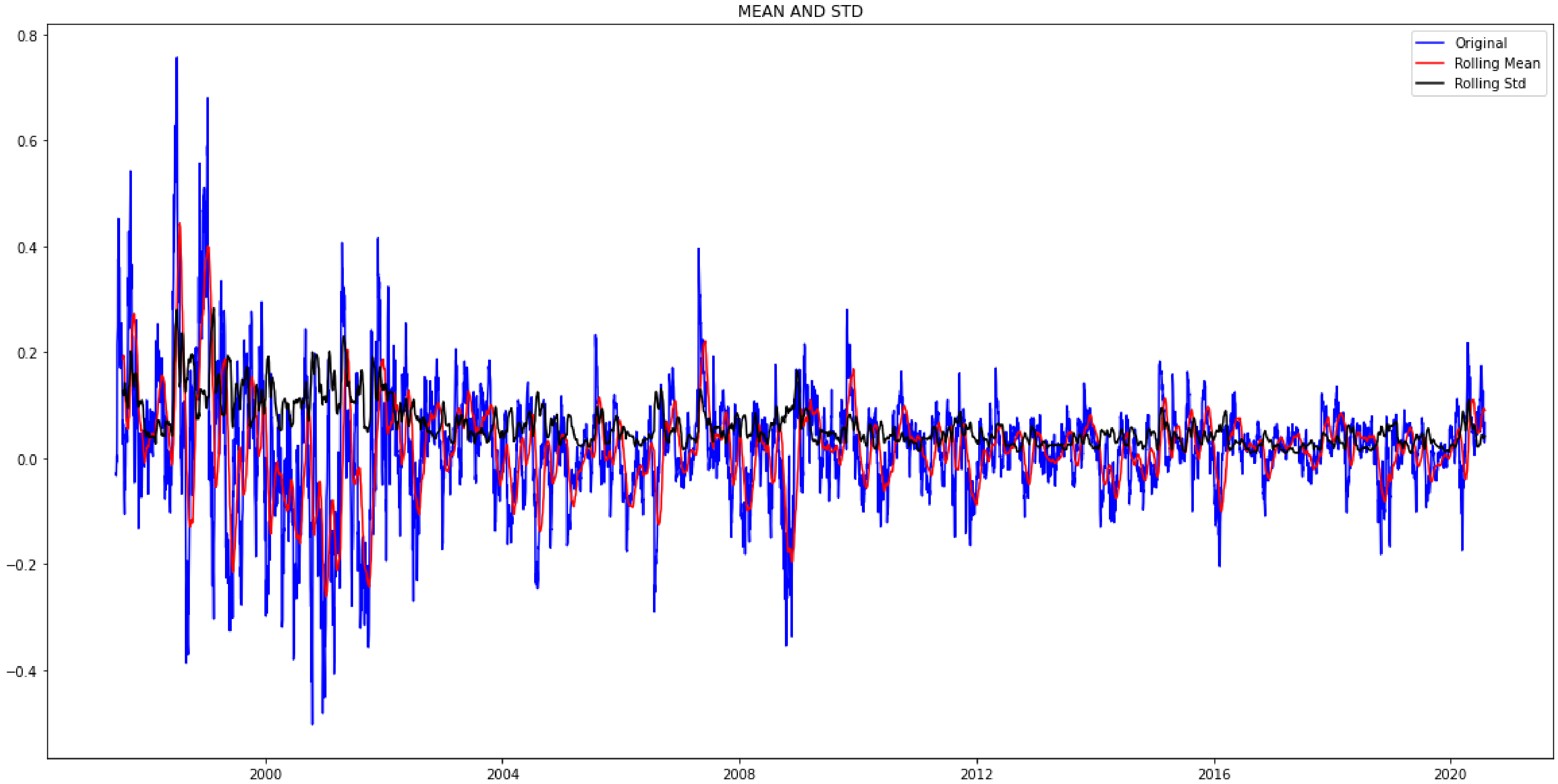
In [52]:

```
from statsmodels.tsa.stattools import adfuller
def test_stationary(timeseries):

    #determinging rolling statistics
    movingAv = timeseries.rolling(window = 30).mean()
    movingStd = timeseries.rolling(window = 30).std()

    #plotting rolling statistics
    plt.figure(figsize=(20,10))
    original = plt.plot(timeseries, color = 'blue', label = 'Original')
    mean = plt.plot(movingAv, color = 'red', label = 'Rolling Mean')
    std = plt.plot(movingStd, color = 'black', label= 'Rolling Std')
    plt.legend(loc= 'best')
    plt.title('MEAN AND STD')
    plt.show(block=False)

    #ADF test
    print('Results of ADF Test')
    dftest = adfuller(timeseries['Close'], autolag='AIC')
    dfout = pd.Series(dftest[0:4], index = ['Test Statistic', 'P-Value', 'Lags Used', 'Number of Observations'])
    for key,value in dftest[4].items():
```

```
        dfout['Crictical Values (%s)'%key] = value
    print(dfout)
```

In [53]:

```
test_stationary(dflog2)
```




```
Results of ADF Test
Test Statistic                -1.004452e+01
P-Value                        1.466400e-17
Lags Used                      3.400000e+01
Number of Observations         5.778000e+03
Crictical Values (1%)         -3.431482e+00
Crictical Values (5%)         -2.862040e+00
```

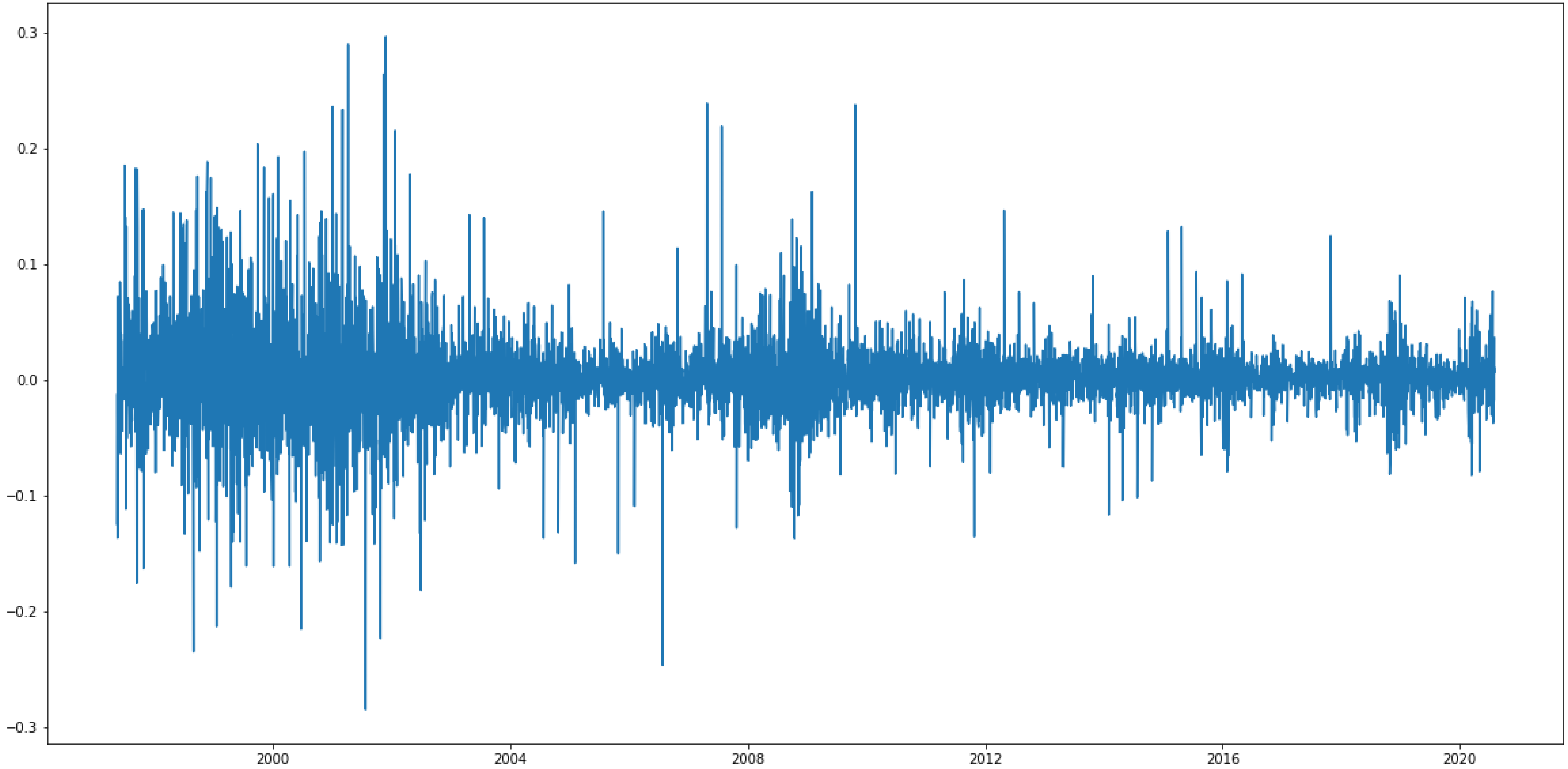
```
Crictical Values (10%)    -2.567036e+00
dtype: float64
```

In [55]:

```
logshift_df = df_log - df_log.shift()
plt.figure(figsize=(20,10))
plt.plot(logshift_df)
```

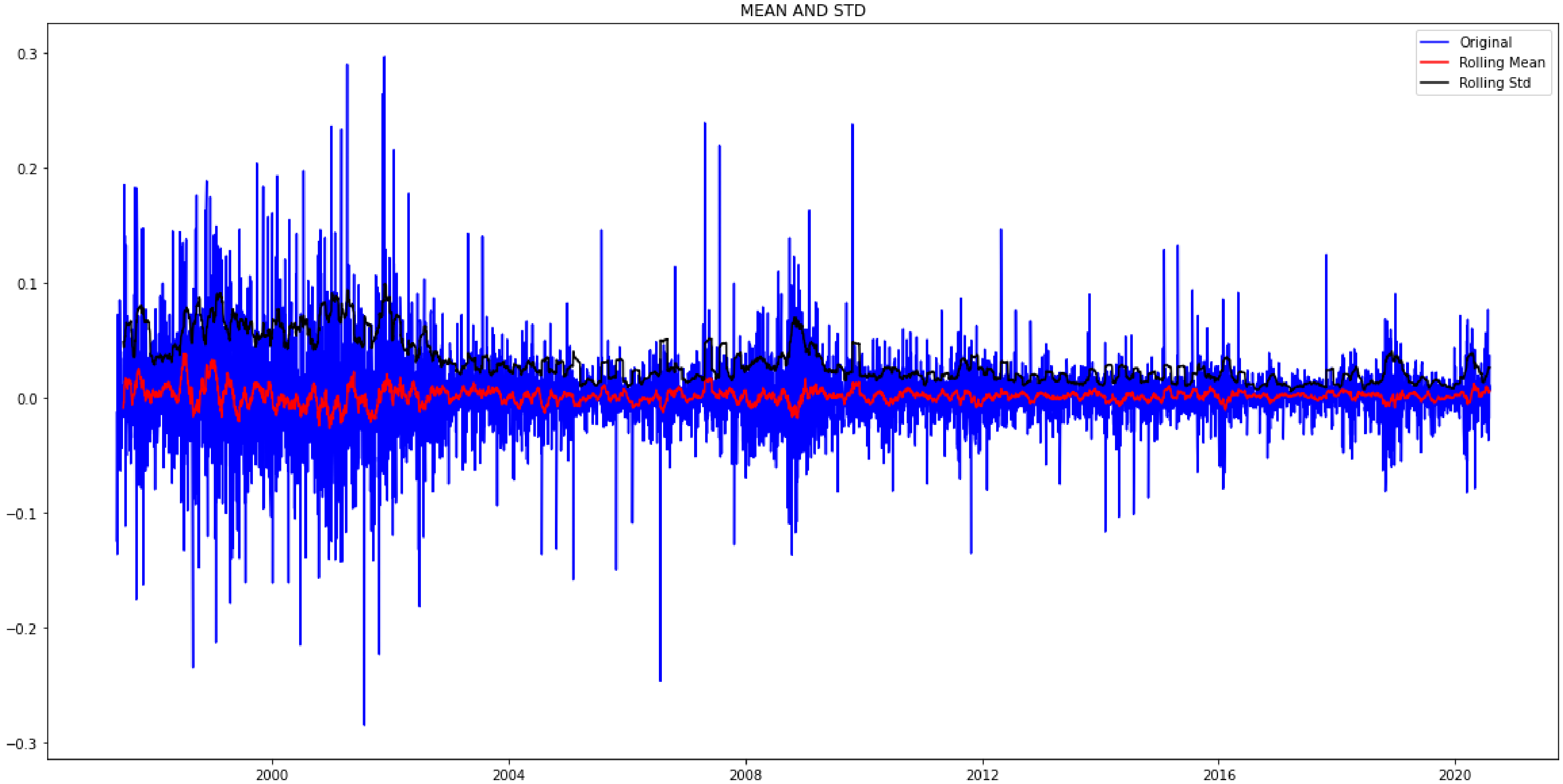
Out[55]:

```
[<matplotlib.lines.Line2D at 0x2341d7299c8>]
```



In [56]:

```
logshift_df.dropna(inplace=True)
test_stationary(logshift_df)
```



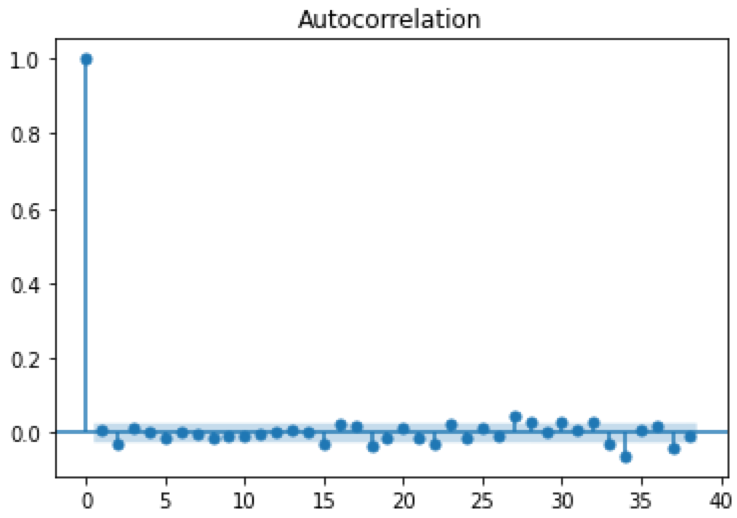

```
Results of ADF Test
Test Statistic              -1.301784e+01
P-Value                      2.489450e-24
Lags Used                    3.300000e+01
Number of Observations       5.807000e+03
Crictical Values (1%)       -3.431477e+00
Crictical Values (5%)       -2.862038e+00
Crictical Values (10%)      -2.567035e+00
dtype: float64
```

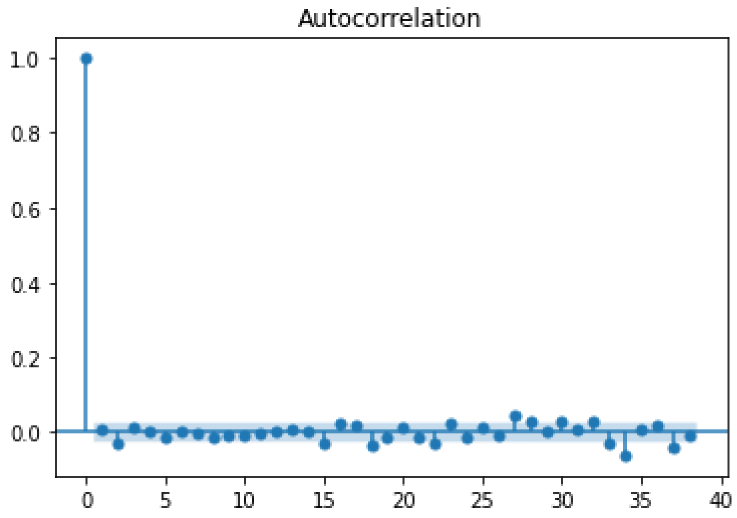
In [57]:

```
#ACR AND PACF TESTS
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

In [64]:
```
plot_acf(logshift_df)
```

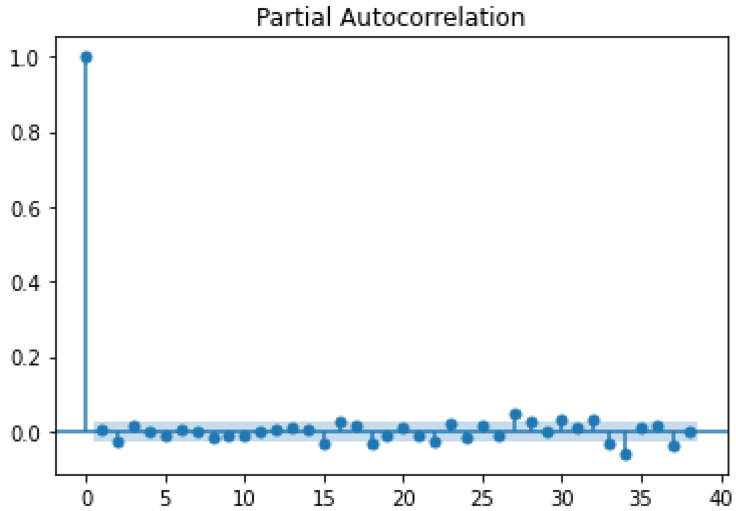
Out[64]:

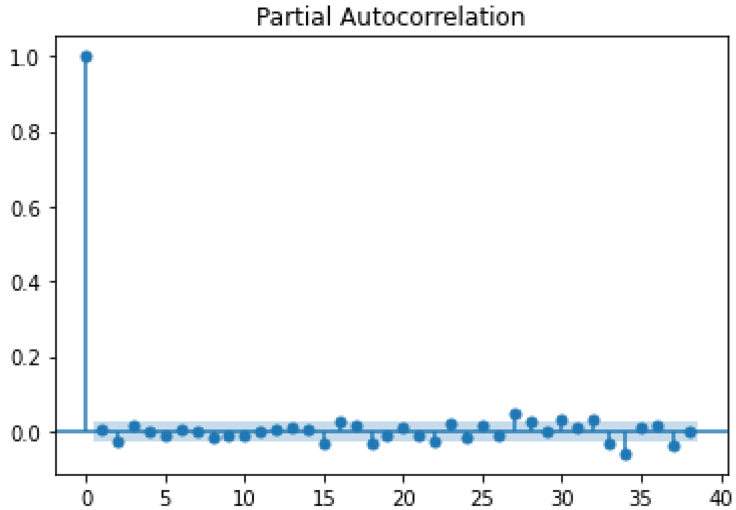




In [65]:
```
plot_pacf(logshift_df)
```

Out[65]:





In [66]:
```
logshift_df.shape
```

Out[66]: (5841, 1)

In [67]:
```
train_data = logshift_df[0:5300]
test_data = logshift_df[5301:]
```

In [68]:
```
from pandas.plotting import autocorrelation_plot
from matplotlib import pyplot
%matplotlib inline
```

In [69]:
```
from statsmodels.tsa.arima_model import ARIMA
```

In [70]:
```
model = ARIMA(train_data, order=(2,1,2))
```

```
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\statsmodels\tsa\arima_model.py:472: FutureWarn
ing:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\statsmodels\tsa\base\tsa_model.py:583: ValueWa
rning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.
g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\statsmodels\tsa\base\tsa_model.py:583: ValueWa
rning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.
g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
```

In [71]:
```
model_fit = model.fit()
```

```
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\statsmodels\base\model.py:548: HessianInversio
nWarning: Inverting hessian failed, no bse or cov_params available
  'available', HessianInversionWarning)
c:\users\user\appdata\local\programs\python\python37\lib\site-packages\statsmodels\tsa\arima_model.py:472: FutureWarn
```

```
ing:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
```

In [72]:
```
test_data.shape
```

Out[72]:
```
(540, 1)
```

In [73]:
```
model_fit.aic
```

Out[73]:
```
-19623.912676607644
```

In [75]:
```
model_forecast = model_fit.forecast(steps=540)[0]
model_forecast
```

Out[75]:
```
array([0.001638  , 0.00141455, 0.00151647, 0.00144339, 0.00149493,
       0.0014571 , 0.00148337, 0.00146366, 0.00147694, 0.00146655,
       0.00147314, 0.00146754, 0.00147069, 0.00146757, 0.00146894,
       0.00146709, 0.00146756, 0.00146636, 0.00146635, 0.00146549,
       0.00146525, 0.00146456, 0.00146419, 0.00146359, 0.00146315,
       0.0014626 , 0.00146213, 0.00146161, 0.00146112, 0.0014606 ,
       0.00146011, 0.0014596 , 0.0014591 , 0.00145859, 0.00145809,
       0.00145759, 0.00145709, 0.00145658, 0.00145608, 0.00145557,
       0.00145507, 0.00145457, 0.00145406, 0.00145356, 0.00145306,
       0.00145255, 0.00145205, 0.00145155, 0.00145104, 0.00145054,
       0.00145004, 0.00144953, 0.00144903, 0.00144853, 0.00144802,
       0.00144752, 0.00144702, 0.00144651, 0.00144601, 0.00144551,
```

```
0.001445  , 0.0014445 , 0.001444  , 0.00144349, 0.00144299,
0.00144249, 0.00144198, 0.00144148, 0.00144098, 0.00144047,
0.00143997, 0.00143947, 0.00143896, 0.00143846, 0.00143796,
0.00143745, 0.00143695, 0.00143645, 0.00143594, 0.00143544,
0.00143494, 0.00143443, 0.00143393, 0.00143342, 0.00143292,
0.00143242, 0.00143191, 0.00143141, 0.00143091, 0.0014304 ,
0.0014299 , 0.0014294 , 0.00142889, 0.00142839, 0.00142789,
0.00142738, 0.00142688, 0.00142638, 0.00142587, 0.00142537,
0.00142487, 0.00142436, 0.00142386, 0.00142336, 0.00142285,
0.00142235, 0.00142185, 0.00142134, 0.00142084, 0.00142034,
0.00141983, 0.00141933, 0.00141883, 0.00141832, 0.00141782,
0.00141732, 0.00141681, 0.00141631, 0.00141581, 0.0014153 ,
0.0014148 , 0.0014143 , 0.00141379, 0.00141329, 0.00141279,
0.00141228, 0.00141178, 0.00141128, 0.00141077, 0.00141027,
0.00140976, 0.00140926, 0.00140876, 0.00140825, 0.00140775,
0.00140725, 0.00140674, 0.00140624, 0.00140574, 0.00140523,
0.00140473, 0.00140423, 0.00140372, 0.00140322, 0.00140272,
0.00140221, 0.00140171, 0.00140121, 0.0014007 , 0.0014002 ,
0.0013997 , 0.00139919, 0.00139869, 0.00139819, 0.00139768,
0.00139718, 0.00139668, 0.00139617, 0.00139567, 0.00139517,
0.00139466, 0.00139416, 0.00139366, 0.00139315, 0.00139265,
0.00139215, 0.00139164, 0.00139114, 0.00139064, 0.00139013,
0.00138963, 0.00138913, 0.00138862, 0.00138812, 0.00138762,
0.00138711, 0.00138661, 0.0013861 , 0.0013856 , 0.0013851 ,
0.00138459, 0.00138409, 0.00138359, 0.00138308, 0.00138258,
0.00138208, 0.00138157, 0.00138107, 0.00138057, 0.00138006,
0.00137956, 0.00137906, 0.00137855, 0.00137805, 0.00137755,
0.00137704, 0.00137654, 0.00137604, 0.00137553, 0.00137503,
0.00137453, 0.00137402, 0.00137352, 0.00137302, 0.00137251,
0.00137201, 0.00137151, 0.001371  , 0.0013705 , 0.00137   ,
0.00136949, 0.00136899, 0.00136849, 0.00136798, 0.00136748,
0.00136698, 0.00136647, 0.00136597, 0.00136547, 0.00136496,
0.00136446, 0.00136396, 0.00136345, 0.00136295, 0.00136244,
0.00136194, 0.00136144, 0.00136093, 0.00136043, 0.00135993,
0.00135942, 0.00135892, 0.00135842, 0.00135791, 0.00135741,
0.00135691, 0.0013564 , 0.0013559 , 0.0013554 , 0.00135489,
0.00135439, 0.00135389, 0.00135338, 0.00135288, 0.00135238,
0.00135187, 0.00135137, 0.00135087, 0.00135036, 0.00134986,
0.00134936, 0.00134885, 0.00134835, 0.00134785, 0.00134734,
0.00134684, 0.00134634, 0.00134583, 0.00134533, 0.00134483,
0.00134432, 0.00134382, 0.00134332, 0.00134281, 0.00134231,
0.00134181, 0.0013413 , 0.0013408 , 0.0013403 , 0.00133979,
0.00133929, 0.00133878, 0.00133828, 0.00133778, 0.00133727,
0.00133677, 0.00133627, 0.00133576, 0.00133526, 0.00133476,
0.00133425, 0.00133375, 0.00133325, 0.00133274, 0.00133224,
```

```
0.00133174, 0.00133123, 0.00133073, 0.00133023, 0.00132972,
0.00132922, 0.00132872, 0.00132821, 0.00132771, 0.00132721,
0.0013267 , 0.0013262 , 0.0013257 , 0.00132519, 0.00132469,
0.00132419, 0.00132368, 0.00132318, 0.00132268, 0.00132217,
0.00132167, 0.00132117, 0.00132066, 0.00132016, 0.00131966,
0.00131915, 0.00131865, 0.00131815, 0.00131764, 0.00131714,
0.00131664, 0.00131613, 0.00131563, 0.00131512, 0.00131462,
0.00131412, 0.00131361, 0.00131311, 0.00131261, 0.0013121 ,
0.0013116 , 0.0013111 , 0.00131059, 0.00131009, 0.00130959,
0.00130908, 0.00130858, 0.00130808, 0.00130757, 0.00130707,
0.00130657, 0.00130606, 0.00130556, 0.00130506, 0.00130455,
0.00130405, 0.00130355, 0.00130304, 0.00130254, 0.00130204,
0.00130153, 0.00130103, 0.00130053, 0.00130002, 0.00129952,
0.00129902, 0.00129851, 0.00129801, 0.00129751, 0.001297  ,
0.0012965 , 0.001296  , 0.00129549, 0.00129499, 0.00129449,
0.00129398, 0.00129348, 0.00129298, 0.00129247, 0.00129197,
0.00129146, 0.00129096, 0.00129046, 0.00128995, 0.00128945,
0.00128895, 0.00128844, 0.00128794, 0.00128744, 0.00128693,
0.00128643, 0.00128593, 0.00128542, 0.00128492, 0.00128442,
0.00128391, 0.00128341, 0.00128291, 0.0012824 , 0.0012819 ,
0.0012814 , 0.00128089, 0.00128039, 0.00127989, 0.00127938,
0.00127888, 0.00127838, 0.00127787, 0.00127737, 0.00127687,
0.00127636, 0.00127586, 0.00127536, 0.00127485, 0.00127435,
0.00127385, 0.00127334, 0.00127284, 0.00127234, 0.00127183,
0.00127133, 0.00127083, 0.00127032, 0.00126982, 0.00126932,
0.00126881, 0.00126831, 0.0012678 , 0.0012673 , 0.0012668 ,
0.00126629, 0.00126579, 0.00126529, 0.00126478, 0.00126428,
0.00126378, 0.00126327, 0.00126277, 0.00126227, 0.00126176,
0.00126126, 0.00126076, 0.00126025, 0.00125975, 0.00125925,
0.00125874, 0.00125824, 0.00125774, 0.00125723, 0.00125673,
0.00125623, 0.00125572, 0.00125522, 0.00125472, 0.00125421,
0.00125371, 0.00125321, 0.0012527 , 0.0012522 , 0.0012517 ,
0.00125119, 0.00125069, 0.00125019, 0.00124968, 0.00124918,
0.00124868, 0.00124817, 0.00124767, 0.00124717, 0.00124666,
0.00124616, 0.00124566, 0.00124515, 0.00124465, 0.00124414,
0.00124364, 0.00124314, 0.00124263, 0.00124213, 0.00124163,
0.00124112, 0.00124062, 0.00124012, 0.00123961, 0.00123911,
0.00123861, 0.0012381 , 0.0012376 , 0.0012371 , 0.00123659,
0.00123609, 0.00123559, 0.00123508, 0.00123458, 0.00123408,
0.00123357, 0.00123307, 0.00123257, 0.00123206, 0.00123156,
0.00123106, 0.00123055, 0.00123005, 0.00122955, 0.00122904,
0.00122854, 0.00122804, 0.00122753, 0.00122703, 0.00122653,
0.00122602, 0.00122552, 0.00122502, 0.00122451, 0.00122401,
0.00122351, 0.001223  , 0.0012225 , 0.001222  , 0.00122149,
0.00122099, 0.00122048, 0.00121998, 0.00121948, 0.00121897,
```

```
       0.00121847, 0.00121797, 0.00121746, 0.00121696, 0.00121646,
       0.00121595, 0.00121545, 0.00121495, 0.00121444, 0.00121394,
       0.00121344, 0.00121293, 0.00121243, 0.00121193, 0.00121142,
       0.00121092, 0.00121042, 0.00120991, 0.00120941, 0.00120891,
       0.0012084 , 0.0012079 , 0.0012074 , 0.00120689, 0.00120639,
       0.00120589, 0.00120538, 0.00120488, 0.00120438, 0.00120387])
```

In [80]:
```
test_data.head()
```

Out[80]:

	Close
Date	
2018-06-11	0.003042
2018-06-12	0.005685
2018-06-13	0.003590
2018-06-14	0.011083
2018-06-15	-0.004587

In [78]:
```
np.sqrt(base_error)
```

Out[78]:
```
0.036710506643087O7
```

In [79]:
```
np.sqrt(mean_squared_error(test_data,model_forecast))
```

Out[79]:
```
0.02120737821276121
```