In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pylab as plt
%matplotlib inline
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 10, 6
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
dataset=pd.read_excel("C:\\Users\\Admin\\Downloads\\assignment 10\\Airlines+Data.xlsx")
dataset.head()
```

Out[2]:

	Month	Passengers
0	1995-01-01	112
1	1995-02-01	118
2	1995-03-01	132
3	1995-04-01	129
4	1995-05-01	121

In [3]:

```
dataset['Month'] = pd.to_datetime(dataset['Month'], infer_datetime_format=True)
indexedDataset = dataset.set_index(['Month'])
```

In [4]:

```
from datetime import datetime
indexedDataset['1995-03']
indexedDataset['1995-03':'1995-06']
indexedDataset['1995']
```

Out[4]:

	Passengers
Month	
1995-01-01	112
1995-02-01	118
1995-03-01	132
1995-04-01	129
1995-05-01	121
1995-06-01	135
1995-07-01	148
1995-08-01	148
1995-09-01	136
1995-10-01	119
1995-11-01	104

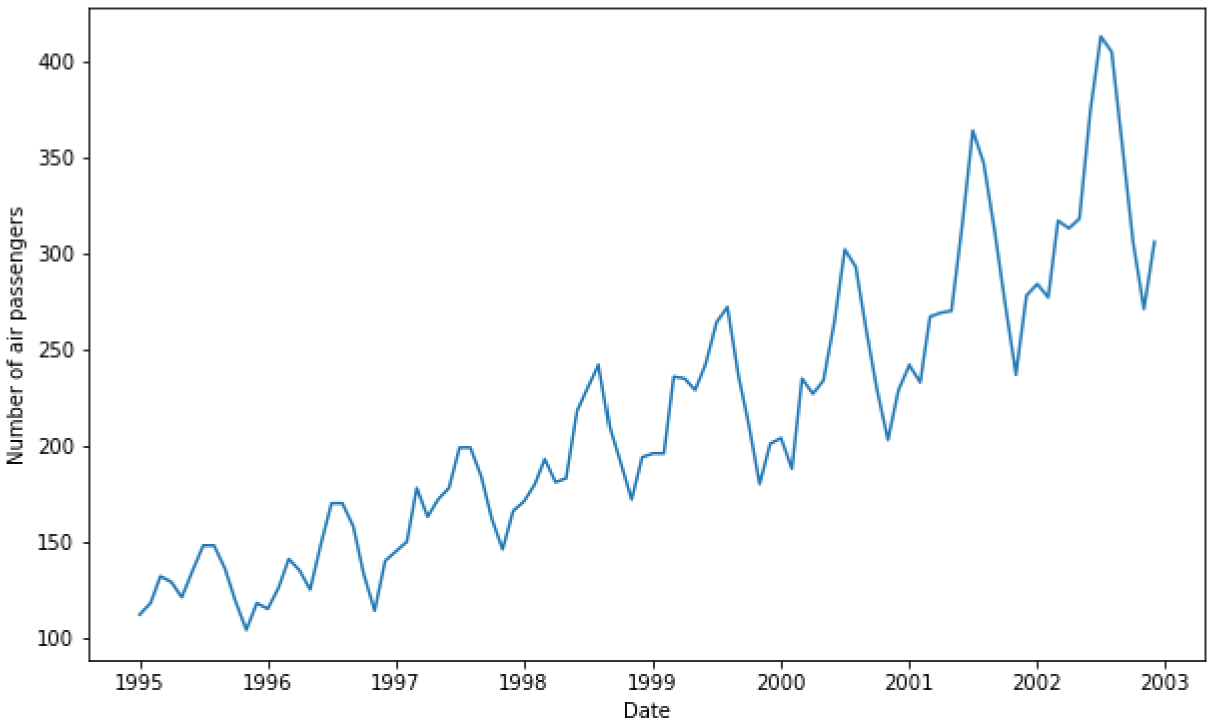
	Passengers
Month	
1995-12-01	118

In [5]:

```
plt.xlabel("Date")
plt.ylabel("Number of air passengers")
plt.plot(indexedDataset)
```

Out[5]:

```
[<matplotlib.lines.Line2D at 0x203ba0d13d0>]
```



In [6]:

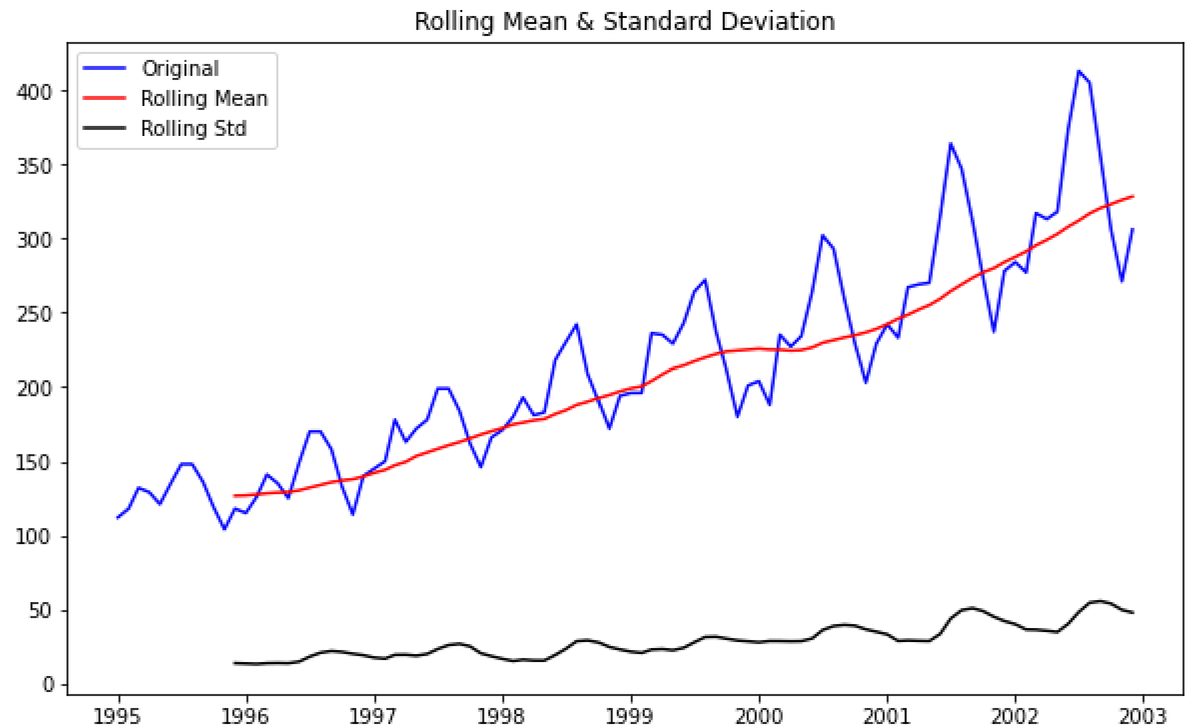
```
rolmean = indexedDataset.rolling(window=12).mean()
rolstd = indexedDataset.rolling(window=12).std()
print(rolmean, rolstd)
```

```
             Passengers
Month
1995-01-01          NaN
1995-02-01          NaN
1995-03-01          NaN
1995-04-01          NaN
1995-05-01          NaN
...                 ...
2002-08-01   316.833333
2002-09-01   320.416667
2002-10-01   323.083333
2002-11-01   325.916667
2002-12-01   328.250000

[96 rows x 1 columns]              Passengers
Month
1995-01-01          NaN
1995-02-01          NaN
```

```
1995-03-01         NaN
1995-04-01         NaN
1995-05-01         NaN
...                ...
2002-08-01   54.530781
2002-09-01   55.586883
2002-10-01   53.899668
2002-11-01   49.692616
2002-12-01   47.861780

[96 rows x 1 columns]
```

In [7]:

```
orig = plt.plot(indexedDataset, 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)
```



In [8]:

```
from statsmodels.tsa.stattools import adfuller

print ('Results of Dickey-Fuller Test:')
dftest = adfuller(indexedDataset['Passengers'], autolag='AIC')
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Numbe
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)
```

```
Results of Dickey-Fuller Test:
Test Statistic                  1.340248
p-value                         0.996825
#Lags Used                     12.000000
Number of Observations Used    83.000000
Critical Value (1%)            -3.511712
```

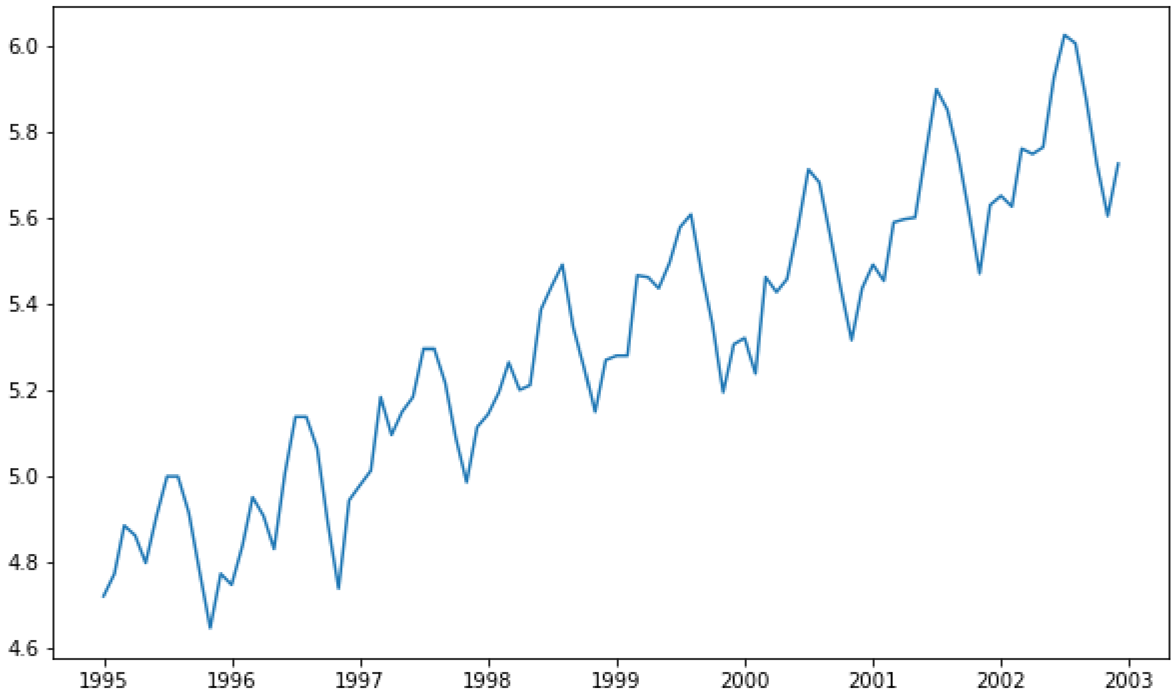
```
Critical Value (5%)           -2.897048
Critical Value (10%)          -2.585713
dtype: float64
```

In [9]:

```
indexedDataset_logScale = np.log(indexedDataset)
plt.plot(indexedDataset_logScale)
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x203bcf002b0>]
```

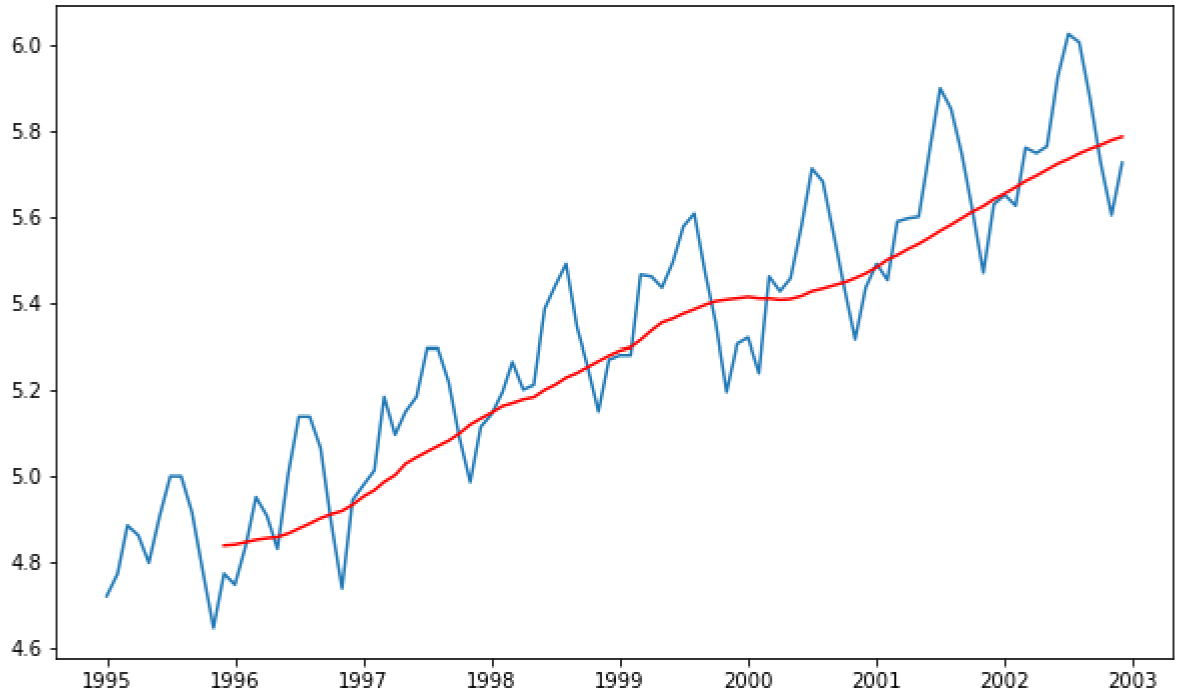


In [10]:

```
movingAverage = indexedDataset_logScale.rolling(window=12).mean()
movingSTD = indexedDataset_logScale.rolling(window=12).std()
plt.plot(indexedDataset_logScale)
plt.plot(movingAverage, color='red')
```

Out[10]:

```
[<matplotlib.lines.Line2D at 0x203bd51d190>]
```



In [11]:

```
datasetLogScaleMinusMovingAverage = indexedDataset_logScale - movingAverage
datasetLogScaleMinusMovingAverage.head(12)
#Remove Nan Values
datasetLogScaleMinusMovingAverage.dropna(inplace=True)
datasetLogScaleMinusMovingAverage.head(10)
```

Out[11]:

	Passengers
Month	
1995-12-01	-0.065494
1996-01-01	-0.093449
1996-02-01	-0.007566
1996-03-01	0.099416
1996-04-01	0.052142
1996-05-01	-0.027529
1996-06-01	0.139881
1996-07-01	0.260184
1996-08-01	0.248635
1996-09-01	0.162937

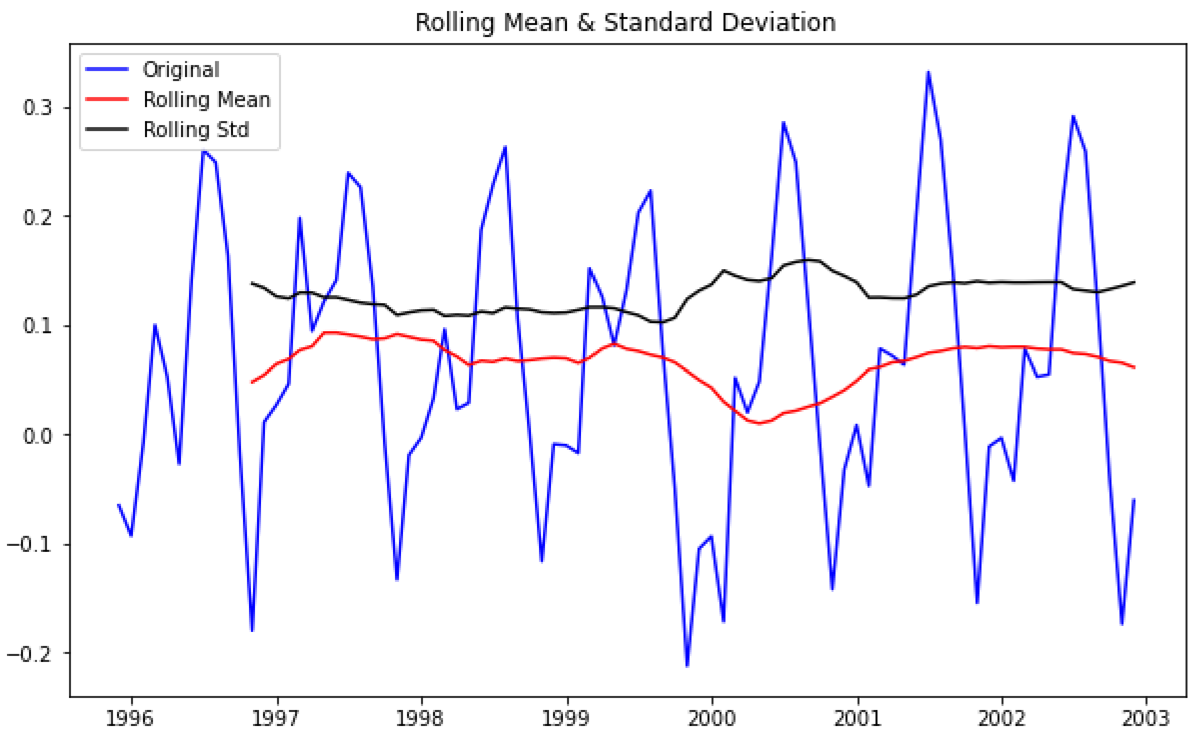
In [12]:

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

    #Determing rolling statistics
    movingAverage = timeseries.rolling(window=12).mean()
    movingSTD = timeseries.rolling(window=12).std()
```

```
    #Plot rolling statistics:
    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)

    #Perform Dickey-Fuller test:
    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries['Passengers'], autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','N
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print(dfoutput)
```

In [13]:

```
test_stationarity(datasetLogScaleMinusMovingAverage)
```



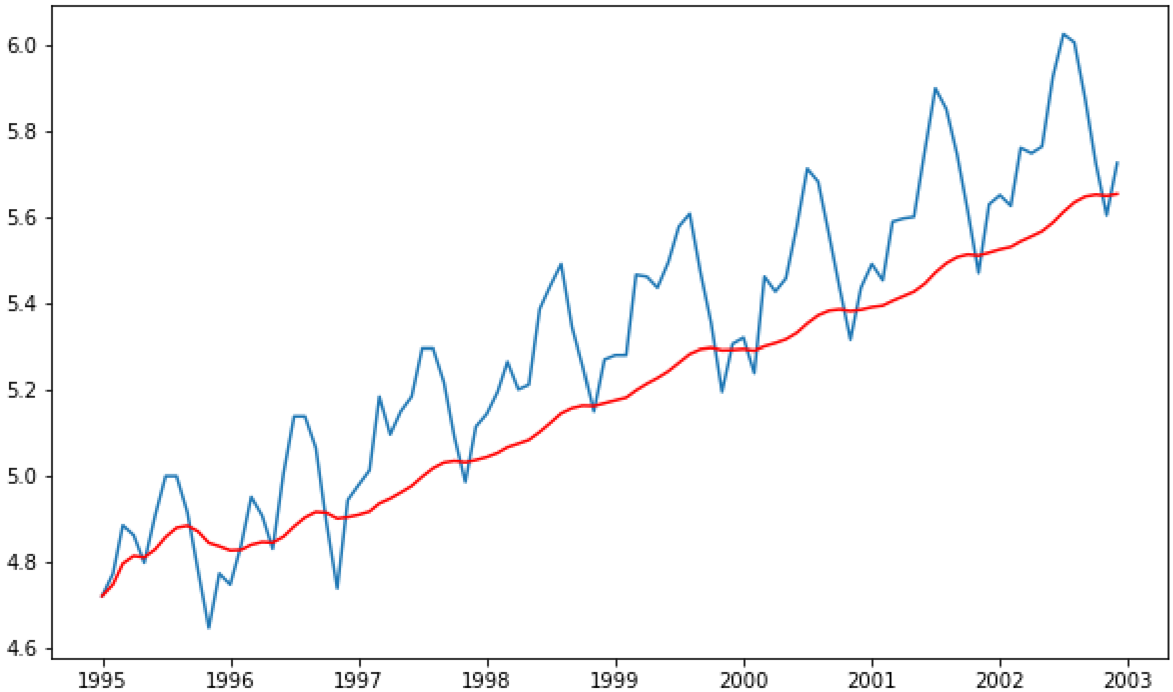

```
Results of Dickey-Fuller Test:
Test Statistic                 -1.910930
p-value                         0.326937
#Lags Used                     12.000000
Number of Observations Used    72.000000
Critical Value (1%)            -3.524624
Critical Value (5%)            -2.902607
Critical Value (10%)           -2.588679
dtype: float64
```

In [14]:

```
exponentialDecayWeightedAverage = indexedDataset_logScale.ewm(halflife=12, min_periods=
plt.plot(indexedDataset_logScale)
plt.plot(exponentialDecayWeightedAverage, color='red')
```

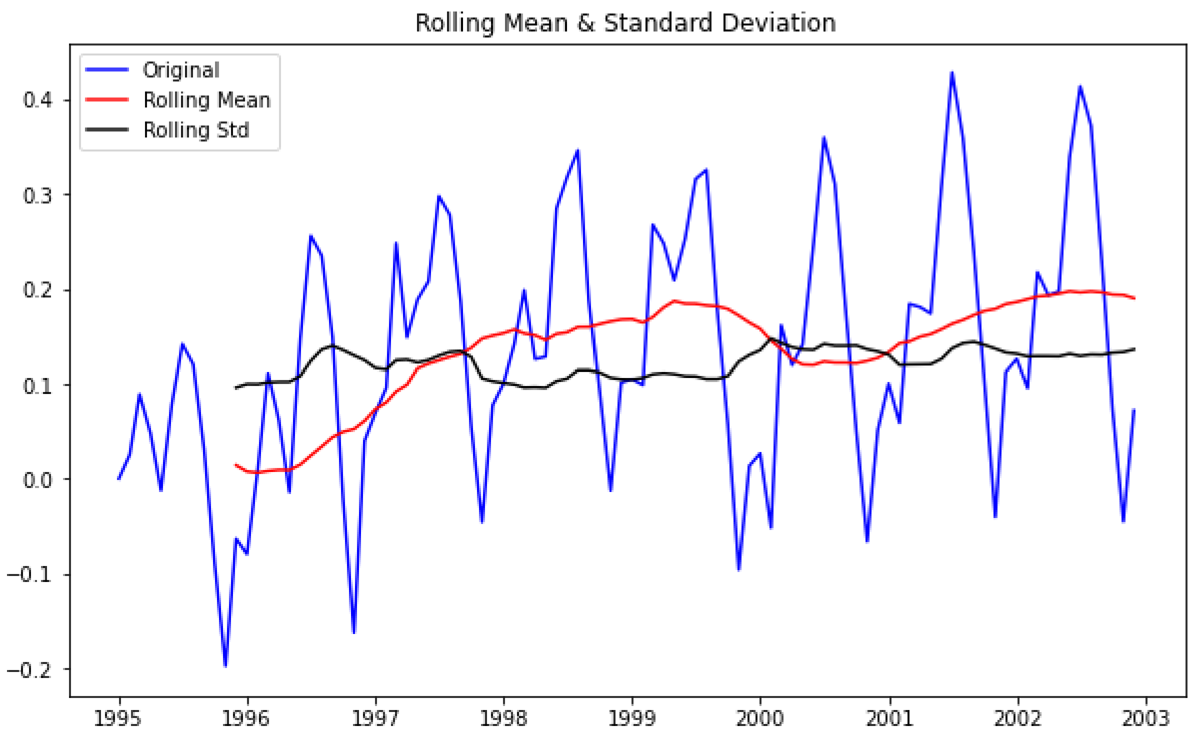
Out[14]:

```
[<matplotlib.lines.Line2D at 0x203bd498f70>]
```



In [15]:

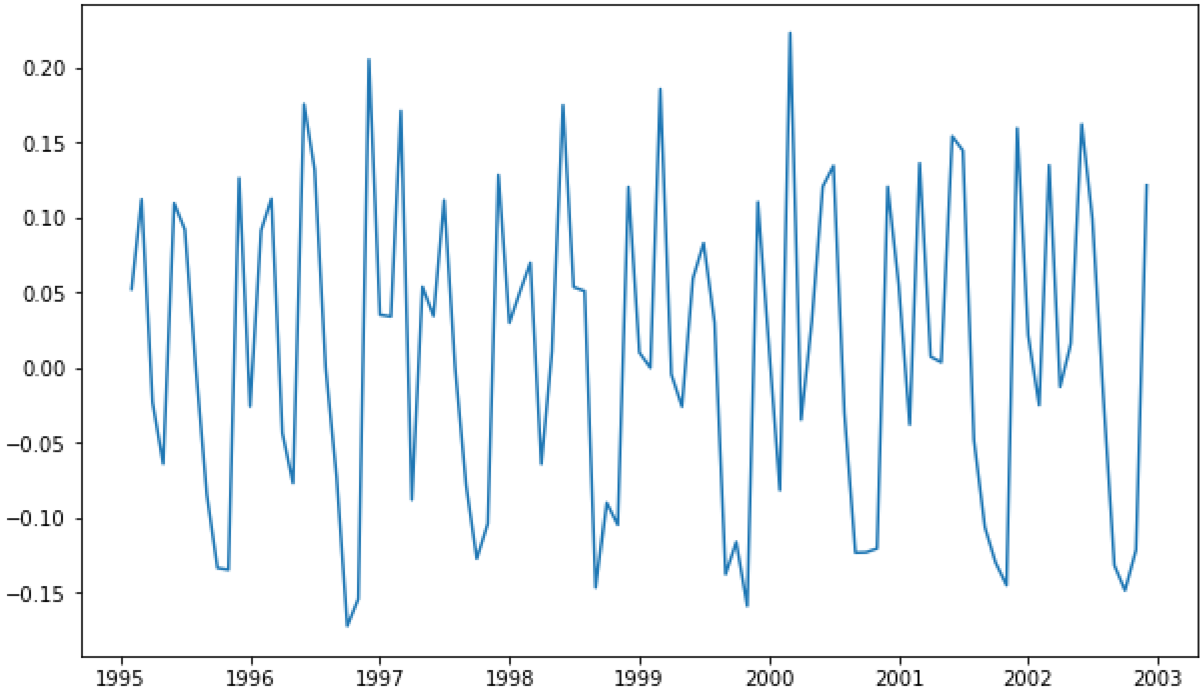
```
datasetLogScaleMinusMovingExponentialDecayAverage = indexedDataset_logScale - exponenti
test_stationarity(datasetLogScaleMinusMovingExponentialDecayAverage)
```



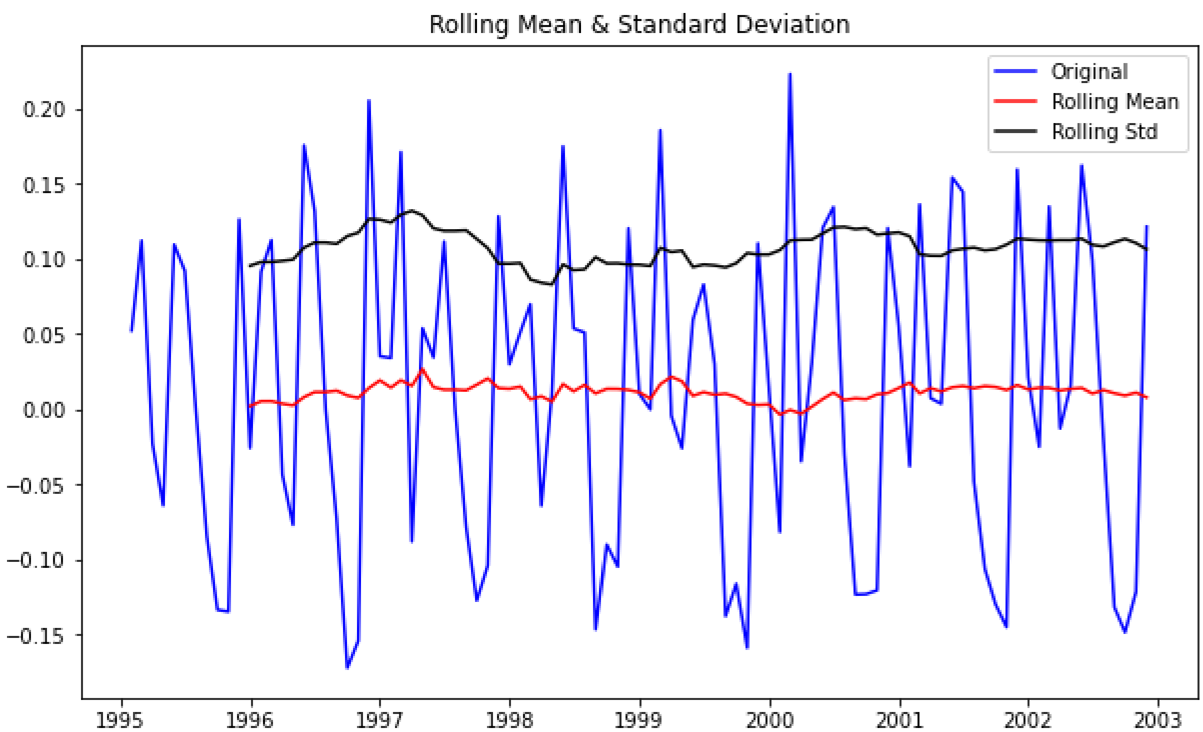

```
Results of Dickey-Fuller Test:
Test Statistic                 -2.835036
p-value                         0.053441
#Lags Used                     12.000000
Number of Observations Used    83.000000
Critical Value (1%)            -3.511712
Critical Value (5%)            -2.897048
Critical Value (10%)           -2.585713
dtype: float64
```

In [16]:
```
datasetLogDiffShifting = indexedDataset_logScale - indexedDataset_logScale.shift()
plt.plot(datasetLogDiffShifting)
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x203bd5714f0>]
```



In [17]:
```
datasetLogDiffShifting.dropna(inplace=True)
test_stationarity(datasetLogDiffShifting)
```

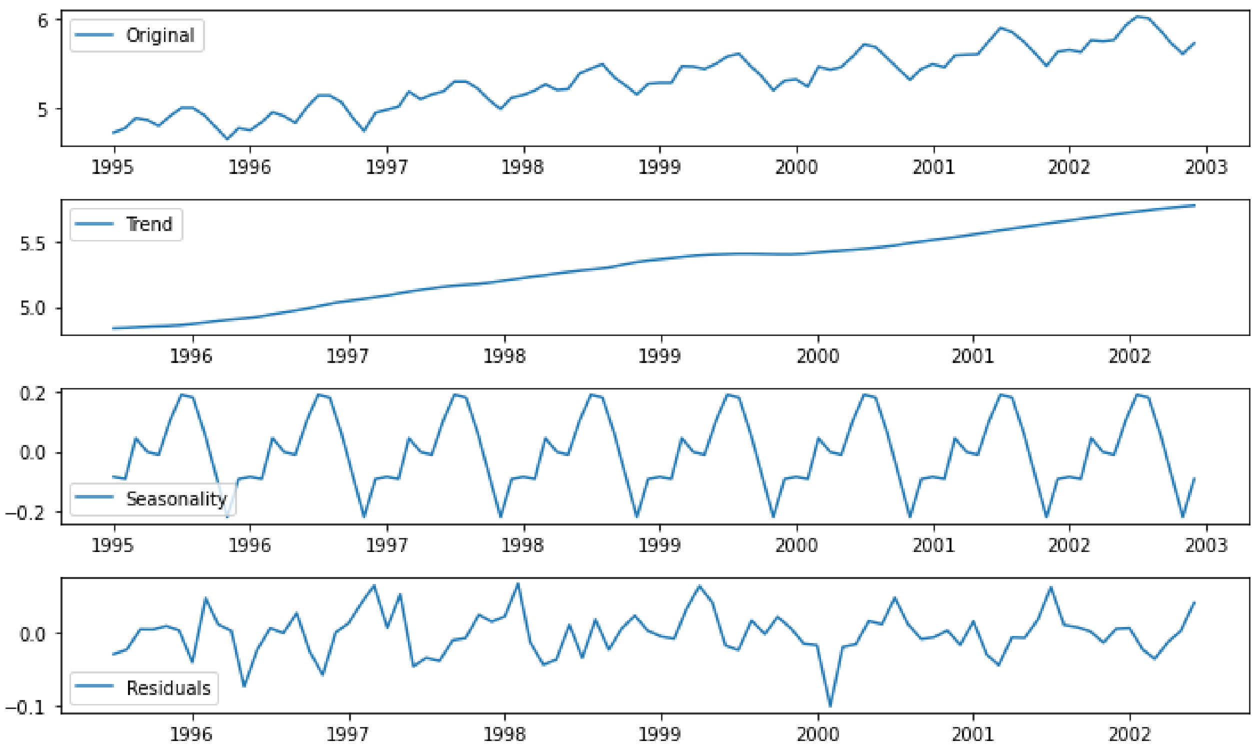



```
Results of Dickey-Fuller Test:
Test Statistic                 -2.670823
p-value                         0.079225
#Lags Used                     12.000000
Number of Observations Used    82.000000
```

```
Critical Value (1%)           -3.512738
Critical Value (5%)           -2.897490
Critical Value (10%)          -2.585949
dtype: float64
```

In [18]:

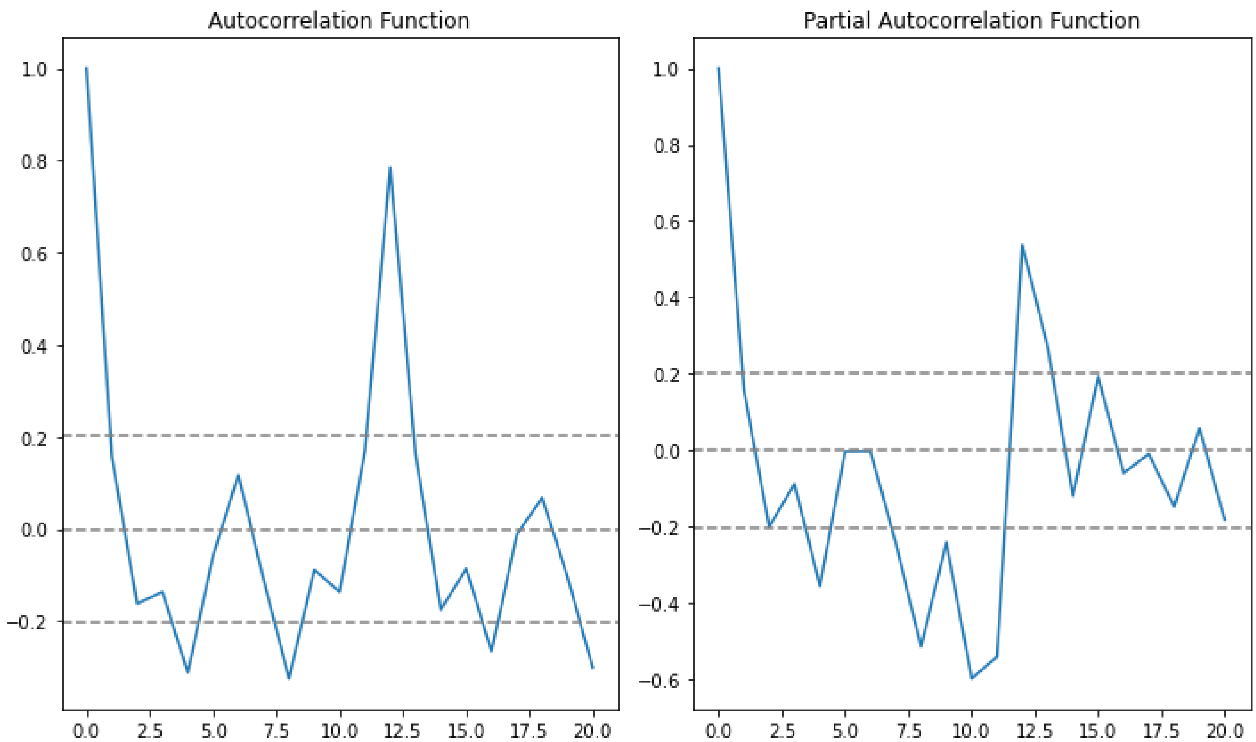
```
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(indexedDataset_logScale)

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

plt.subplot(411)
plt.plot(indexedDataset_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()
```



In [19]:

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

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


#Plot ACF:
plt.subplot(121)
```

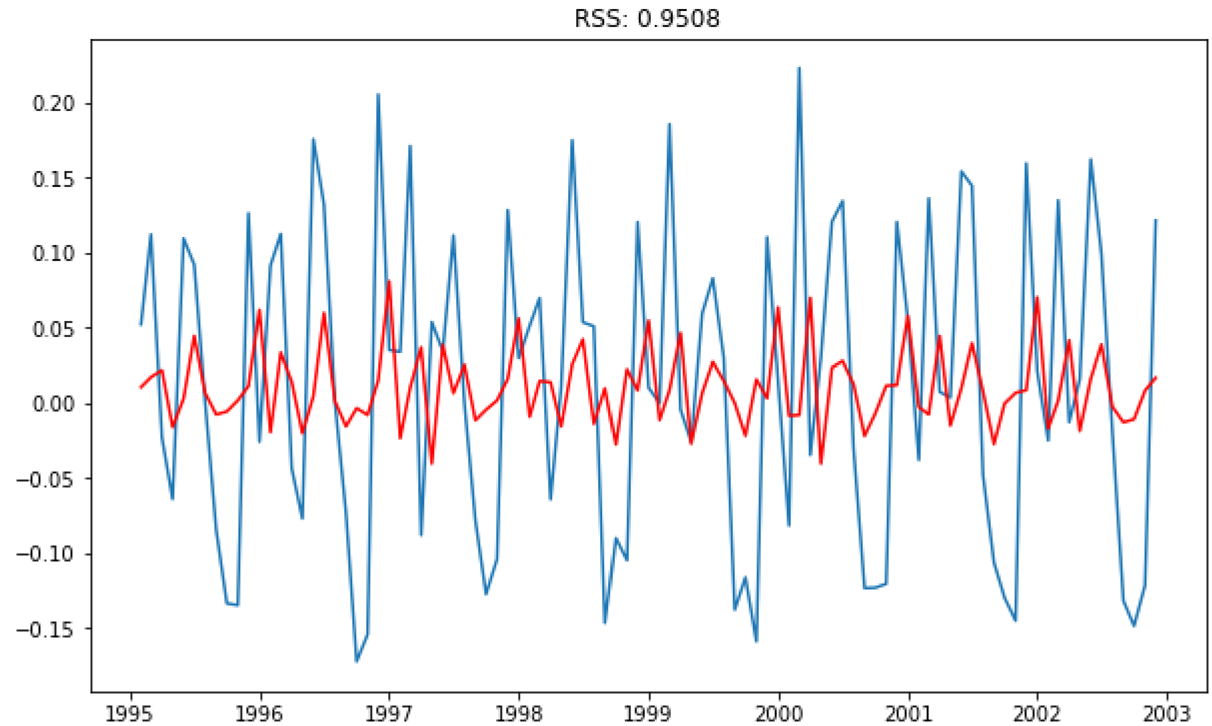
```
plt.plot(lag_acf)
plt.axhline(y=0,linestyle='--',color='gray')
plt.axhline(y=-1.96/np.sqrt(len(datasetLogDiffShifting)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(datasetLogDiffShifting)),linestyle='--',color='gray')
plt.title('Autocorrelation Function')

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



In [20]:

```
from statsmodels.tsa.arima_model import ARIMA

#AR MODEL
model = ARIMA(indexedDataset_logScale, order=(2, 1, 0))
results_AR = model.fit(disp=-1)
plt.plot(datasetLogDiffShifting)
plt.plot(results_AR.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_AR.fittedvalues-datasetLogDiffShifting["Passengers"
print('Plotting AR model')
```

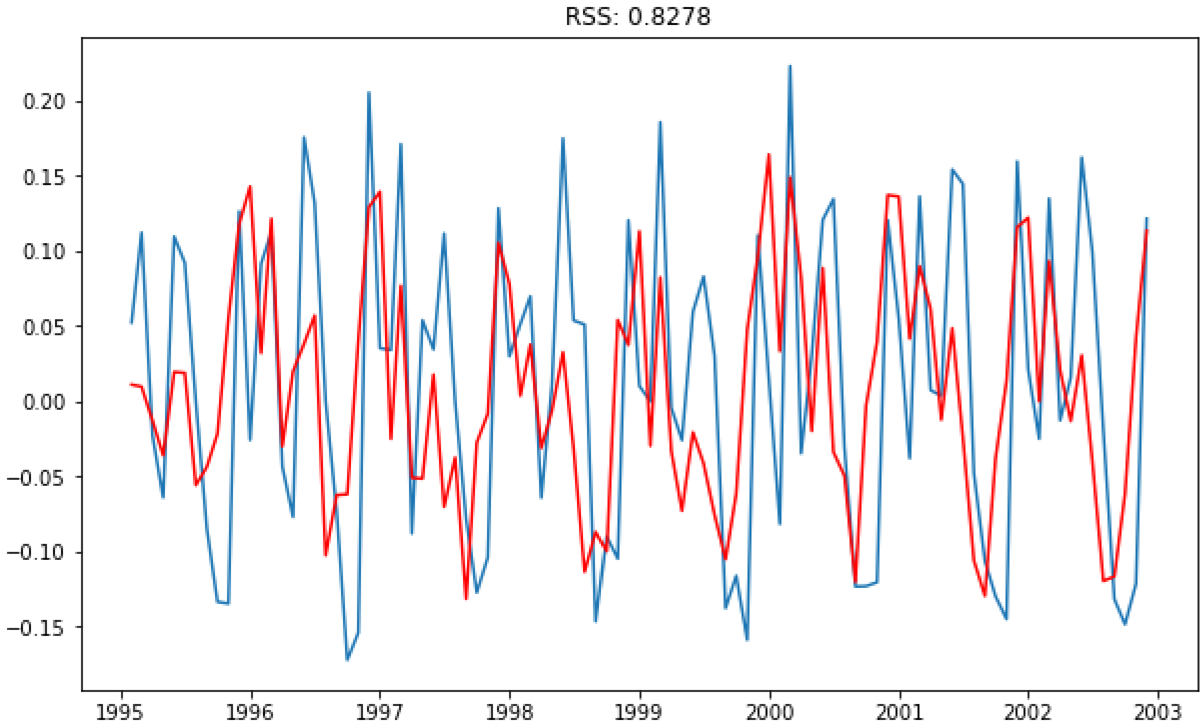
```
C:\Users\Admin\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueW
arning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\Admin\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueW
arning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
Plotting AR model
```



In [21]:

```
model = ARIMA(indexedDataset_logScale, order=(0, 1, 2))
results_MA = model.fit(disp=-1)
plt.plot(datasetLogDiffShifting)
plt.plot(results_MA.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_MA.fittedvalues-datasetLogDiffShifting["Passengers"
print('Plotting AR model')
```

```
C:\Users\Admin\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueW
arning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\Admin\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueW
arning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
Plotting AR model
```



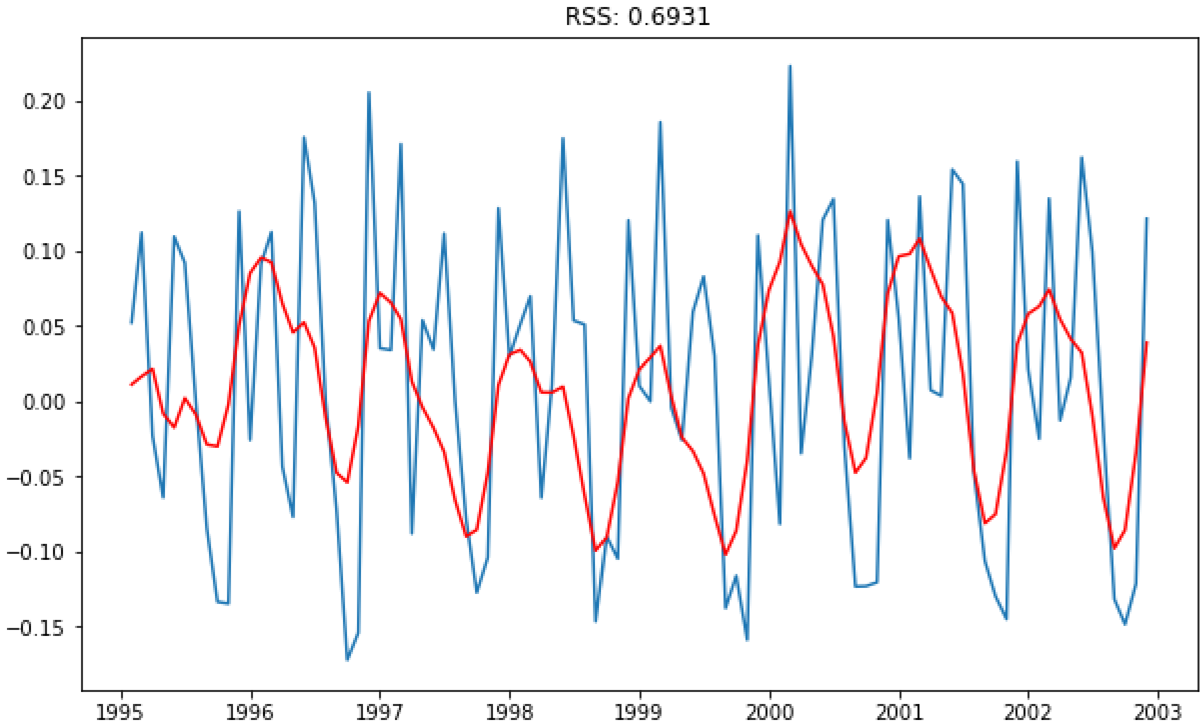
In [22]:

```
model = ARIMA(indexedDataset_logScale, order=(2, 1, 2))
results_ARIMA = model.fit(disp=-1)
plt.plot(datasetLogDiffShifting)
plt.plot(results_ARIMA.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_ARIMA.fittedvalues-datasetLogDiffShifting["Passenge
```

```
C:\Users\Admin\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueW
arning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\Admin\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueW
arning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

Out[22]:

```
Text(0.5, 1.0, 'RSS: 0.6931')
```



In [23]:

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

```
Month
1995-02-01    0.011261
1995-03-01    0.016602
1995-04-01    0.021663
1995-05-01   -0.008096
1995-06-01   -0.017395
dtype: float64
```

In [24]:

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

```
Month
1995-02-01    0.011261
1995-03-01    0.027863
1995-04-01    0.049526
1995-05-01    0.041430
1995-06-01    0.024035
dtype: float64
```

In [25]:

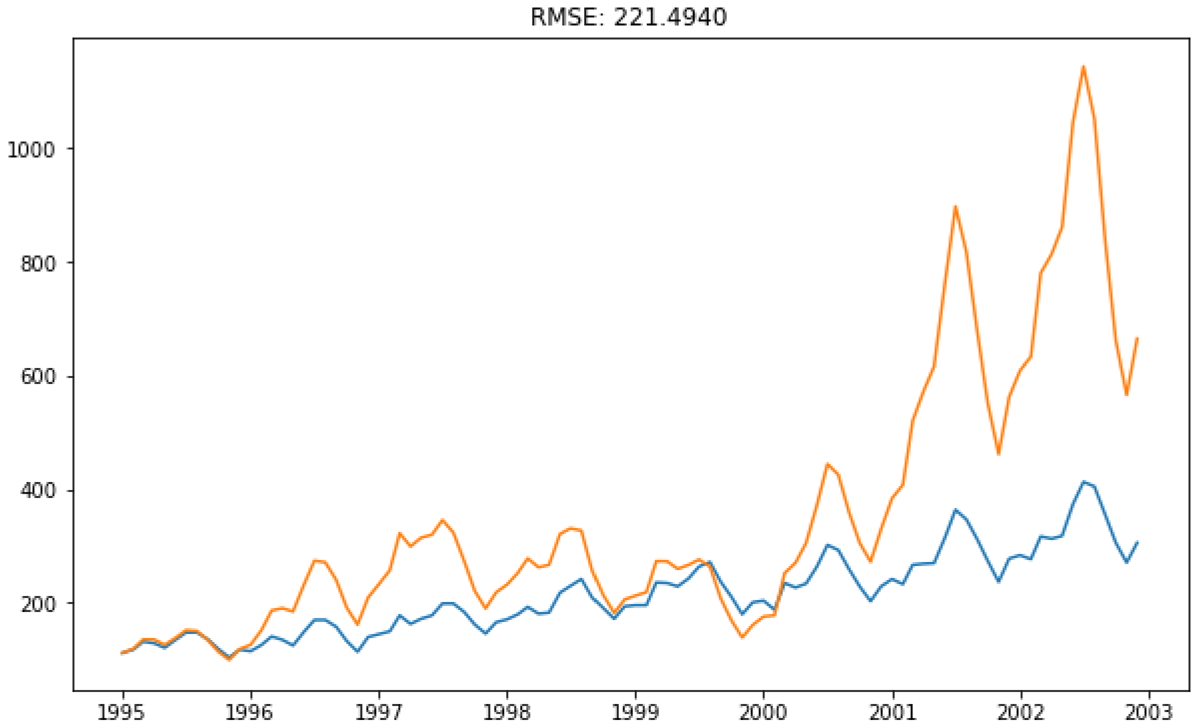
```
predictions_ARIMA_log = pd.Series(indexedDataset_logScale['Passengers'], index=indexedD
predictions_ARIMA_log = predictions_ARIMA_log.add(predictions_ARIMA_diff_cumsum,fill_va
predictions_ARIMA_log.head()
```

Out[25]:

```
Month
1995-01-01    4.718499
1995-02-01    4.781946
1995-03-01    4.910665
1995-04-01    4.909338
1995-05-01    4.837220
dtype: float64
```

In [26]:
```
predictions_ARIMA = np.exp(predictions_ARIMA_log)
plt.plot(indexedDataset)
plt.plot(predictions_ARIMA)
plt.title('RMSE: %.4f'% np.sqrt(sum((predictions_ARIMA-indexedDataset["Passengers"])**2
```

Out[26]:
```
Text(0.5, 1.0, 'RMSE: 221.4940')
```



In [ ]: