In [3]:
```
import pandas as pd
import numpy as np
import requests
from matplotlib.pylab import rcParams
from pandas import DataFrame
from io import StringIO
import time
import json
from datetime import date
from statsmodels.tsa.stattools import adfuller, acf, pacf
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import mean_squared_error
import matplotlib.pylab as plt
get_ipython().run_line_magic('matplotlib', 'inline')
rcParams['figure.figsize'] = 15, 6
```

In [4]:
```
data = pd.read_csv("SeaPlaneTravel.csv")
data.head()
```

Out[4]:

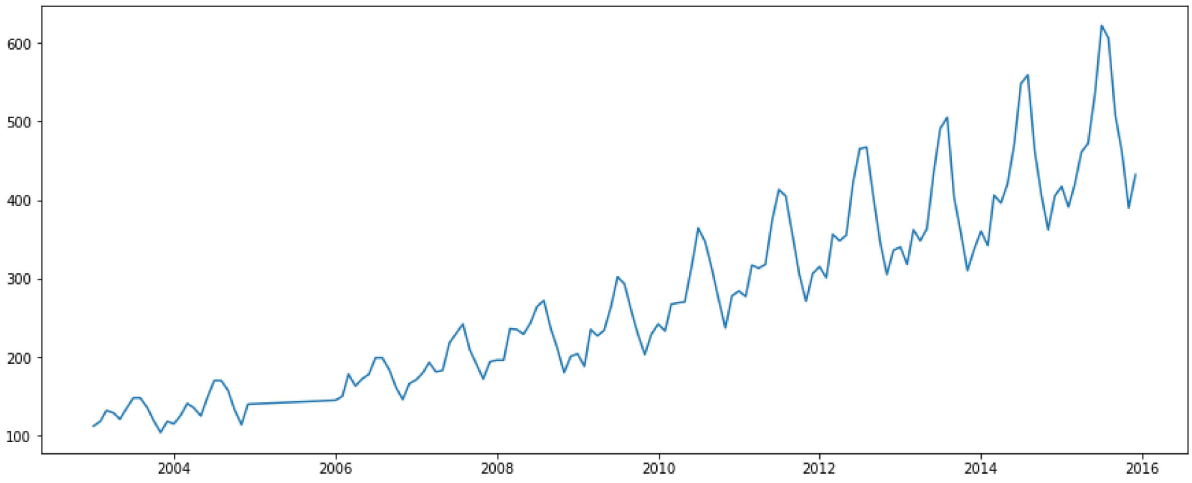
	Month	#Passengers
0	2003-01	112
1	2003-02	118
2	2003-03	132
3	2003-04	129
4	2003-05	121

In [7]:
```
data['Month'] = pd.to_datetime(data['Month'])
indexed_df = data.set_index('Month')
ts = indexed_df['#Passengers']
ts.head(5)
```

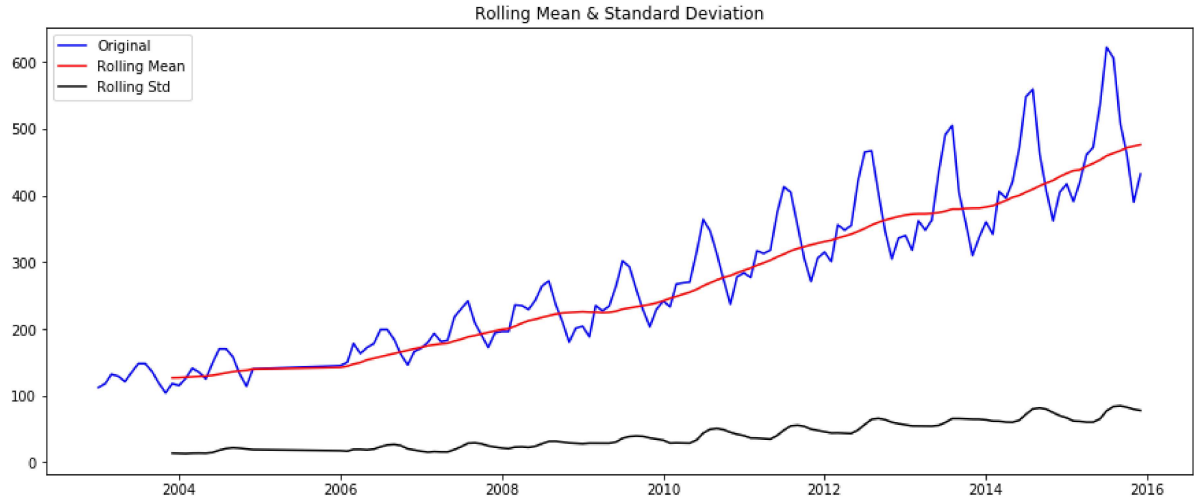
Out[7]:
```
Month
2003-01-01    112
2003-02-01    118
2003-03-01    132
2003-04-01    129
2003-05-01    121
Name: #Passengers, dtype: int64
```

In [10]:
```
plt.plot(ts)
```

Out[10]:
```
[<matplotlib.lines.Line2D at 0x22c64265b08>]
```

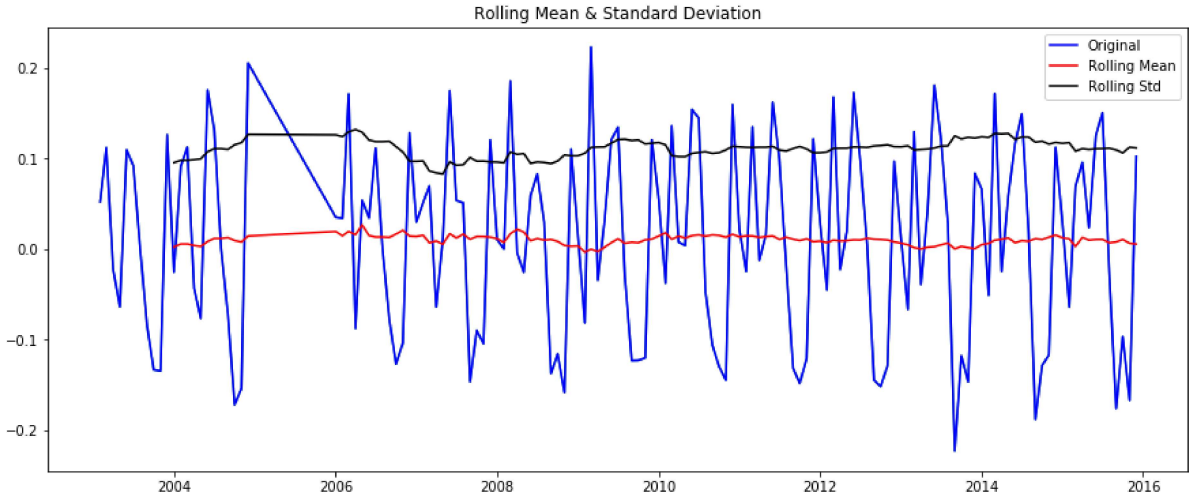


In [13]:
```
def test_stationarity(timeseries):
    #Determining rolling statistics
    rolmean = timeseries.rolling(window=12, center=False).mean()
    rolstd = timeseries.rolling(window=12, center=False).std()
    #Plot Rolling Statistics
    orig = plt.plot(timeseries, color='blue', label='Original')
    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)
    #Perform Dickey-Fuller Test
    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistics', 'p-value', '#L
ag Used', 'Number of Observations used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)' % key] = value
    print(dfoutput)

test_stationarity(ts)
```



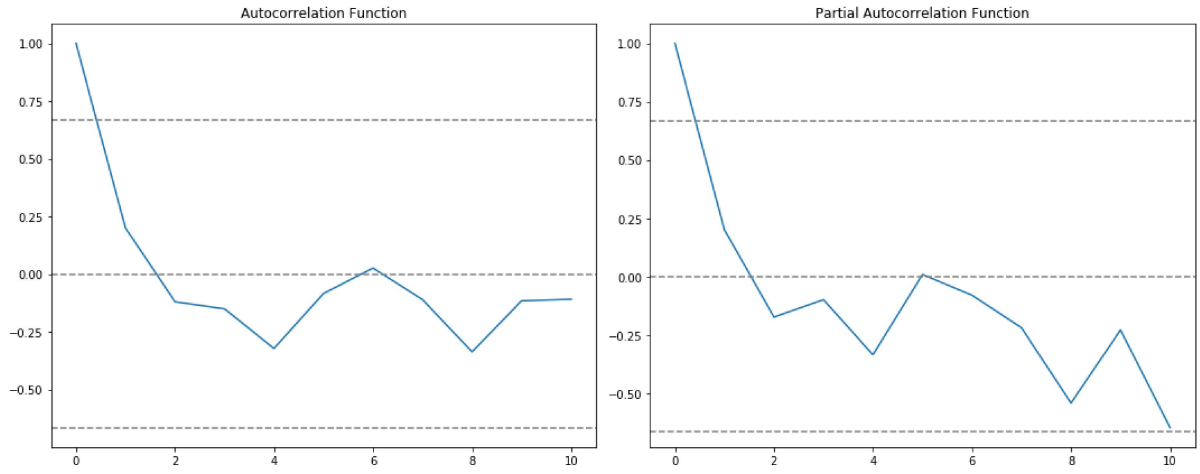

```
Results of Dickey-Fuller Test:
Test Statistics                  0.815369
p-value                          0.991880
#Lag Used                       13.000000
Number of Observations used    130.000000
Critical Value (1%)             -3.481682
Critical Value (5%)             -2.884042
Critical Value (10%)            -2.578770
dtype: float64
```

In [14]:
```
ts_log = np.log(ts)
ts_log_diff = ts_log - ts_log.shift()
plt.plot(ts_log_diff)

ts_log_diff.dropna(inplace=True)
test_stationarity(ts_log_diff)
```



In [17]:
```
#acf
lag_acf = acf(ts_log_diff, nlags=10)
lag_pacf = pacf(ts_log_diff, nlags=10, method='ols')

plt.subplot(121)
plt.plot(lag_acf)
plt.axhline(y=0, linestyle='--', color='gray')
plt.axhline(y=-7.96/np.sqrt(len(ts_log_diff)), linestyle='--', color='gray')
plt.axhline(y=7.96/np.sqrt(len(ts_log_diff)), linestyle='--', color='gray')
plt.title('Autocorrelation Function')

plt.subplot(122)
plt.plot(lag_pacf)
plt.axhline(y=0, linestyle='--', color='gray')
plt.axhline(y=-7.96/np.sqrt(len(ts_log_diff)), linestyle='--', color='gray')
plt.axhline(y=7.96/np.sqrt(len(ts_log_diff)), linestyle='--', color='gray')
plt.title('Partial Autocorrelation Function')
plt.tight_layout()
```

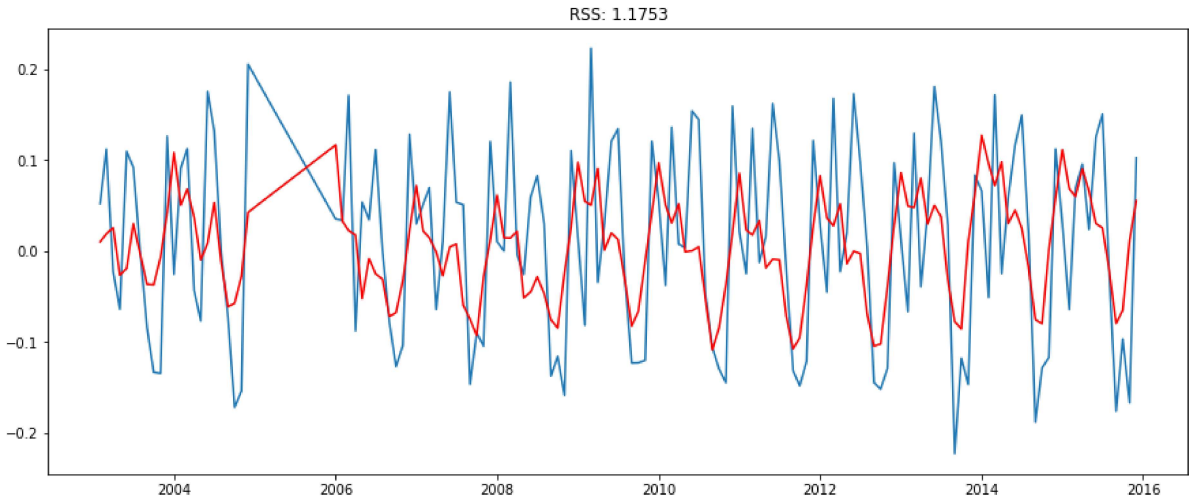
```
C:\Users\hp\anaconda3\lib\site-packages\statsmodels\tsa\stattools.py:572: Fut
ureWarning: fft=True will become the default in a future version of statsmode
ls. To suppress this warning, explicitly set fft=False.
  FutureWarning
```



In [18]:
```
model = ARIMA(ts_log, order=(2,1,1))
results_ARIMA = model.fit(disp=-1)
plt.plot(ts_log_diff)
plt.plot(results_ARIMA.fittedvalues, color='red')
plt.title('RSS: %.4f' % sum((results_ARIMA.fittedvalues-ts_log_diff)**2))
```

```
C:\Users\hp\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:21
8: ValueWarning: A date index has been provided, but it has no associated fre
quency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\hp\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:21
8: ValueWarning: A date index has been provided, but it has no associated fre
quency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
```

Out[18]:
```
Text(0.5, 1.0, 'RSS: 1.1753')
```



In [19]:

```
print(results_ARIMA.summary())
# plot residual errors
residuals = DataFrame(results_ARIMA.resid)
residuals.plot(kind='kde')
print(residuals.describe())
```

ARIMA Model Results

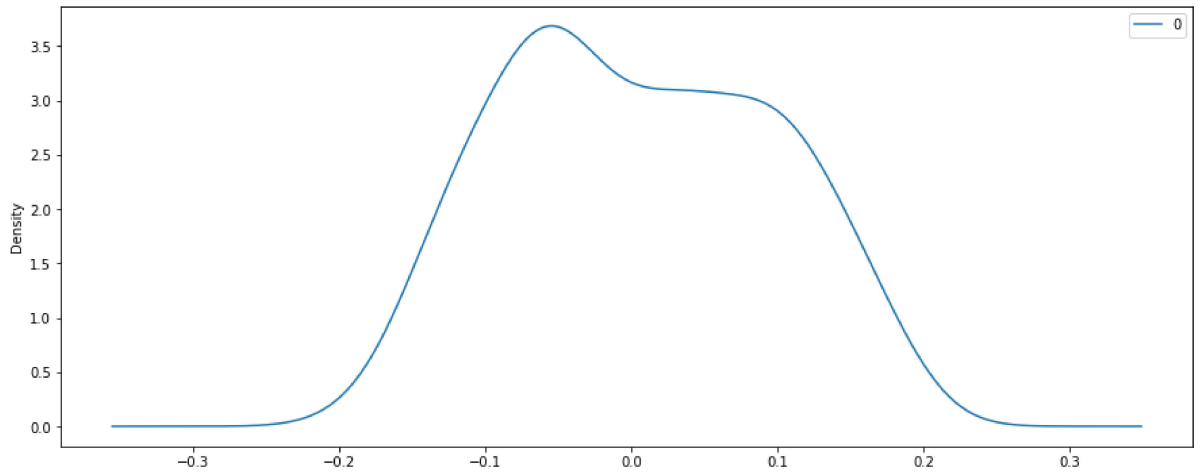
Dep. Variable:	D.#Passengers	No. Observations:	143
Model:	ARIMA(2, 1, 1)	Log Likelihood	140.076
Method:	css-mle	S.D. of innovations	0.090
Date:	Sun, 30 Jan 2022	AIC	-270.151
Time:	18:01:15	BIC	-255.337
Sample:	1	HQIC	-264.131

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0101	0.000	23.509	0.000	0.009	0.011
ar.L1.D.#Passengers	0.9982	0.076	13.162	0.000	0.850	1.147
ar.L2.D.#Passengers	-0.4134	0.077	-5.384	0.000	-0.564	-0.263
ma.L1.D.#Passengers	-0.9999	0.028	-35.273	0.000	-1.055	-0.944

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.2073	-0.9805j	1.5553	-0.1086
AR.2	1.2073	+0.9805j	1.5553	0.1086
MA.1	1.0001	+0.0000j	1.0001	0.0000

	0
count	143.000000
mean	0.005157
std	0.090830
min	-0.179044
25%	-0.065519
50%	0.003244
75%	0.082554
max	0.173153



In [20]:
```
predictions_ARIMA_diff = pd.Series(results_ARIMA.fittedvalues, copy=True)
print(predictions_ARIMA_diff.head())
```

```
Month
2003-02-01    0.010077
2003-03-01    0.018744
2003-04-01    0.025561
2003-05-01   -0.026626
2003-06-01   -0.019243
dtype: float64
```

In [26]:

```
predictions_ARIMA_diff_cumsum = predictions_ARIMA_diff.cumsum()
predictions_ARIMA_log = pd.Series(ts_log.ix[0], index=ts_log.index)
predictions_ARIMA_log = predictions_ARIMA_log.add(
    predictions_ARIMA_diff_cumsum, fill_value=0)
predictions_ARIMA = np.exp(predictions_ARIMA_log)
plt.plot(ts)
plt.plot(predictions_ARIMA)
plt.title('RMSE: %.4f' % np.sqrt(sum((predictions_ARIMA-ts)**2)/len(ts)))

size = int(len(ts_log) - 15)
train, test = ts_log[0:size], ts_log[size:len(ts_log)]
history = [x for x in train]
predictions = list()


size = int(len(ts_log) - 15)
train, test = ts_log[0:size], ts_log[size:len(ts_log)]
history = [x for x in train]
predictions = list()
print('Printing Predicted vs Expected Values...')
print('\n')
for t in range(len(test)):
    model = ARIMA(history, order=(2, 1, 1))
    model_fit = model.fit(disp=0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(float(yhat))
    obs = test[t]
    history.append(obs)
print('predicted=%f, expected=%f' % (np.exp(yhat), np.exp(obs)))


error = mean_squared_error(test, predictions)
print('\n')
print('Printing Mean Squared Error of Predictions...')
print('Test MSE: %.6f' % error)
predictions_series = pd.Series(predictions, index=test.index)


fig, ax = plt.subplots()
ax.set(title='Spot Exchange Rate, Euro into USD',
       xlabel='Date', ylabel='Euro into USD')
ax.plot(ts[-60:], 'o', label='observed')
ax.plot(np.exp(predictions_series), 'g',
        label='rolling one-step out-of-sample forecast')
legend = ax.legend(loc='upper left')
legend.get_frame().set_facecolor('w')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-26-17ff3dfdb496> in <module>
      1 predictions_ARIMA_diff_cumsum = predictions_ARIMA_diff.cumsum()
----> 2 predictions_ARIMA_log = pd.Series(ts_log.ix[0], index=ts_log.index)
      3 predictions_ARIMA_log = predictions_ARIMA_log.add(
      4     predictions_ARIMA_diff_cumsum, fill_value=0)
      5 predictions_ARIMA = np.exp(predictions_ARIMA_log)

~\anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, nam
e)
   5272             if self._info_axis._can_hold_identifiers_and_holds_name(n
ame):
   5273                 return self[name]
-> 5274             return object.__getattribute__(self, name)
   5275
   5276     def __setattr__(self, name: str, value) -> None:

AttributeError: 'Series' object has no attribute 'ix'
```

In [ ]: