In [1]:

```
import pandas as pd
```

In [2]:

```
airline_data=pd.read_excel('Airlines+Data.xlsx')
airline_data['Month']
```

Out[2]:

```
0    1995-01-01
1    1995-02-01
2    1995-03-01
3    1995-04-01
4    1995-05-01
        ...
91   2002-08-01
92   2002-09-01
93   2002-10-01
94   2002-11-01
95   2002-12-01
Name: Month, Length: 96, dtype: datetime64[ns]
```

Initial investigation

In [3]:

```
airline_data.shape
```

Out[3]:

```
(96, 2)
```

In [3]:

```
airline_data.dtypes
```

Out[3]:

```
Month         datetime64[ns]
Passengers             int64
dtype: object
```

In [4]:

```
airline_data.isnull().sum()
```

Out[4]:

```
Month         0
Passengers    0
dtype: int64
```

The data have 2 features and 96 records

The datatype of the features are assigned coorectly and there is no null values

In [11]:

```
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from numpy import sqrt
import matplotlib.pyplot as plt
```

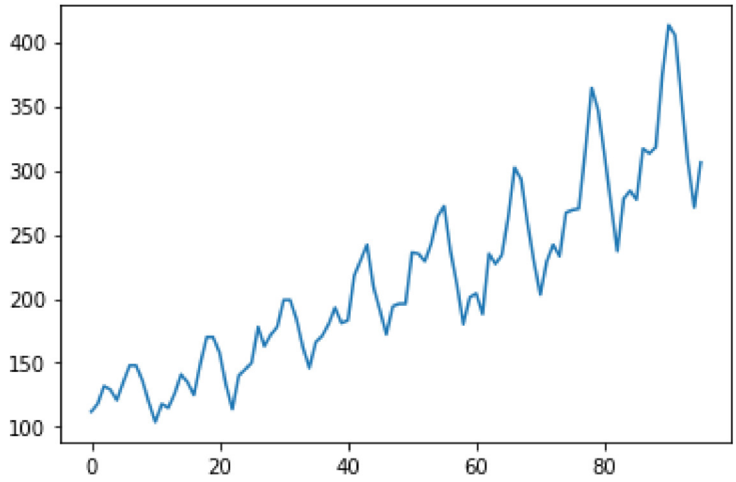
Data visualization

In [7]:

```
airline_data['Passengers'].plot()
```

Out[7]:

<AxesSubplot:>

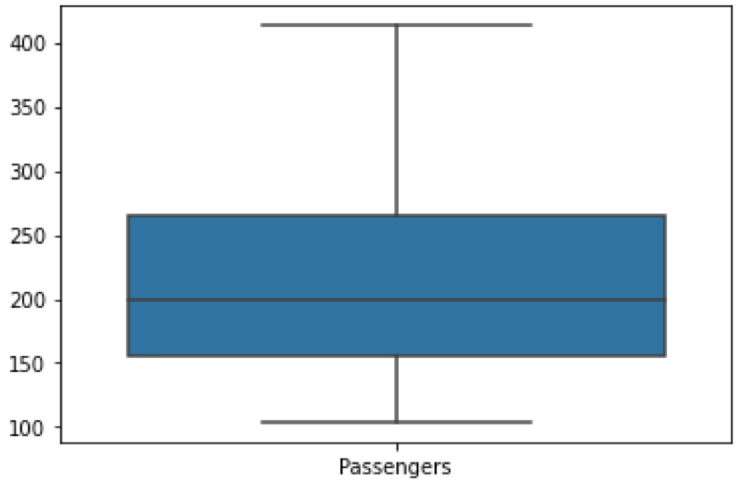


In [6]:

```
sns.boxplot(data=airline_data)
```

Out[6]:

<AxesSubplot:>

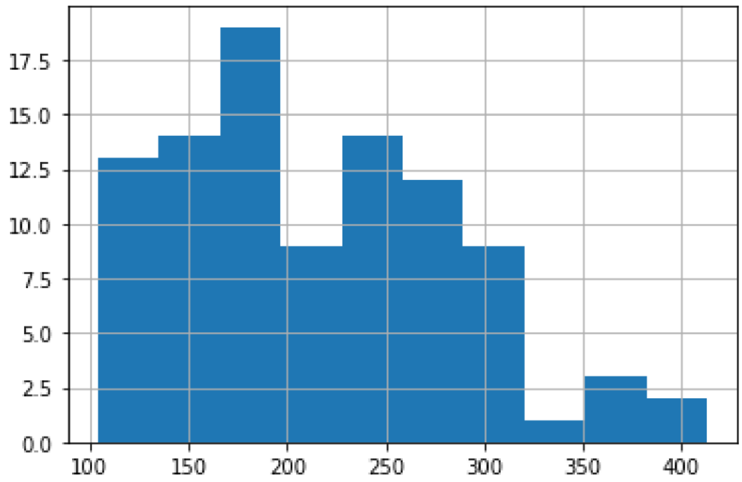


In [8]:

```
airline_data['Passengers'].hist()
```

Out[8]:

<AxesSubplot:>

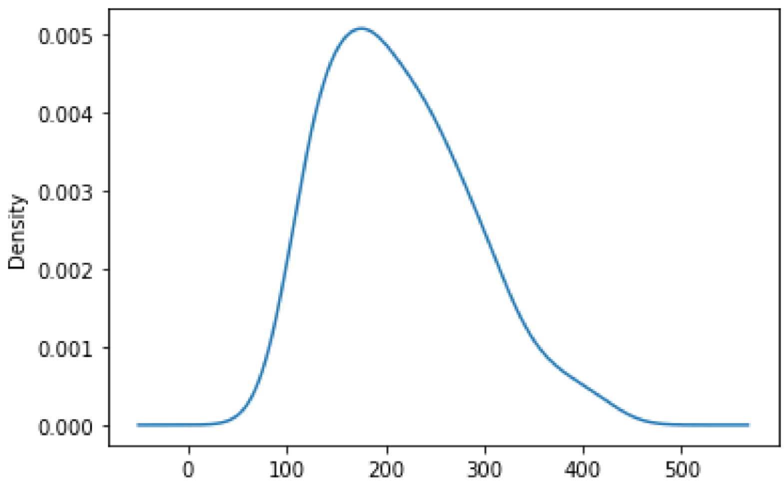


In [9]:

```
airline_data['Passengers'].plot(kind='kde')
```

Out[9]:

<AxesSubplot:ylabel='Density'>

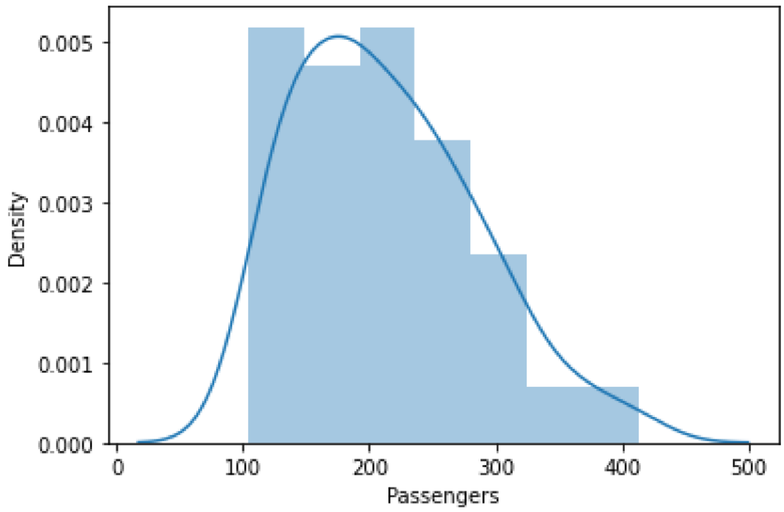


In [12]:

```
sns.distplot(airline_data['Passengers'])
```

Out[12]:

```
<AxesSubplot:xlabel='Passengers', ylabel='Density'>
```

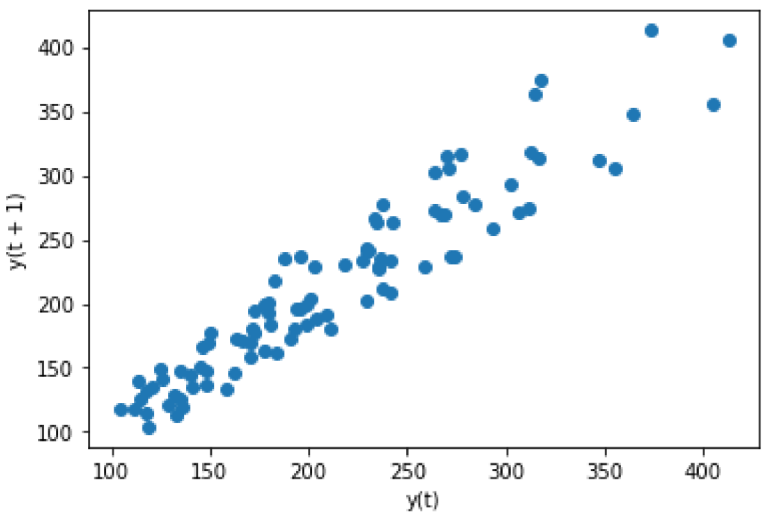


In [13]:

```
from pandas.plotting import lag_plot

lag_plot(airline_data['Passengers'])
```

Out[13]:

```
<AxesSubplot:xlabel='y(t)', ylabel='y(t + 1)'>
```



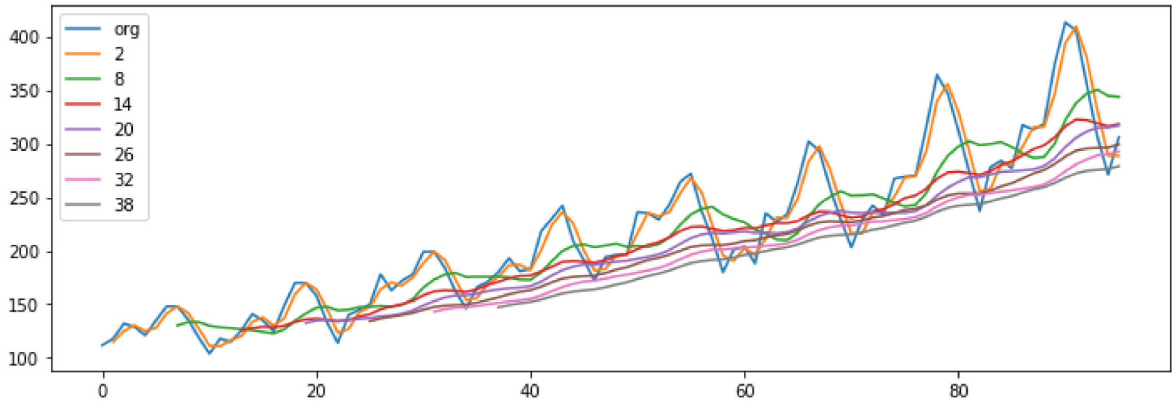
Data driven modelling

In [14]:

```
plt.figure(figsize=(12,4))
airline_data['Passengers'].plot(label="org")
for i in range(2,40,6):
    airline_data['Passengers'].rolling(i).mean().plot(label=str(i))
plt.legend(loc='best')
```

Out[14]:

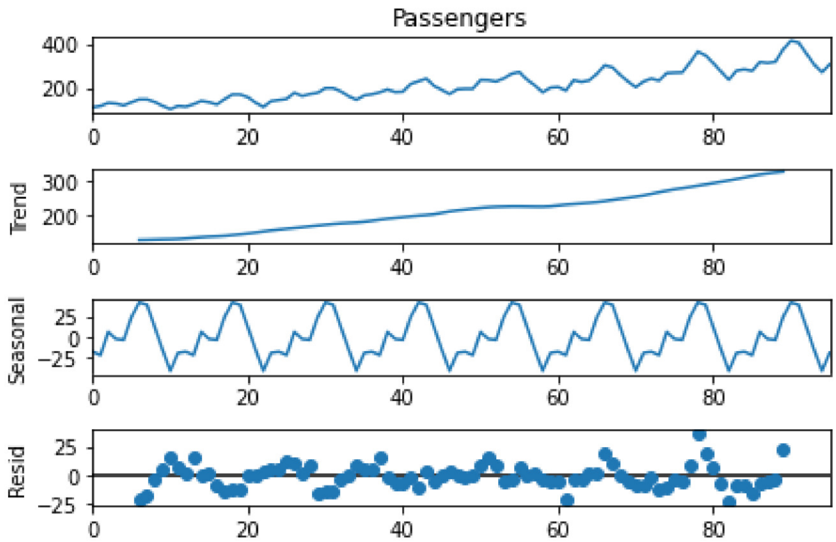
```
<matplotlib.legend.Legend at 0x16126ad3610>
```



Time series decomposition plot

In [15]:

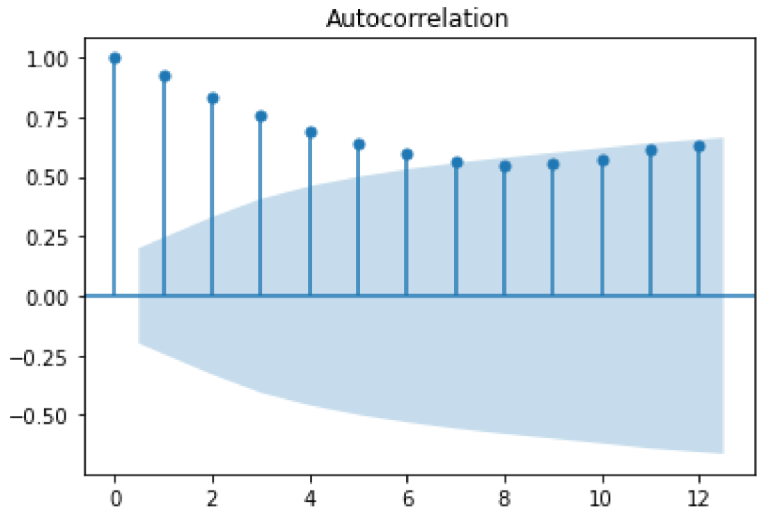
```
from statsmodels.tsa.seasonal import seasonal_decompose

decompose_ts_add = seasonal_decompose(airline_data['Passengers'],period=12)
decompose_ts_add.plot()
plt.show()
```

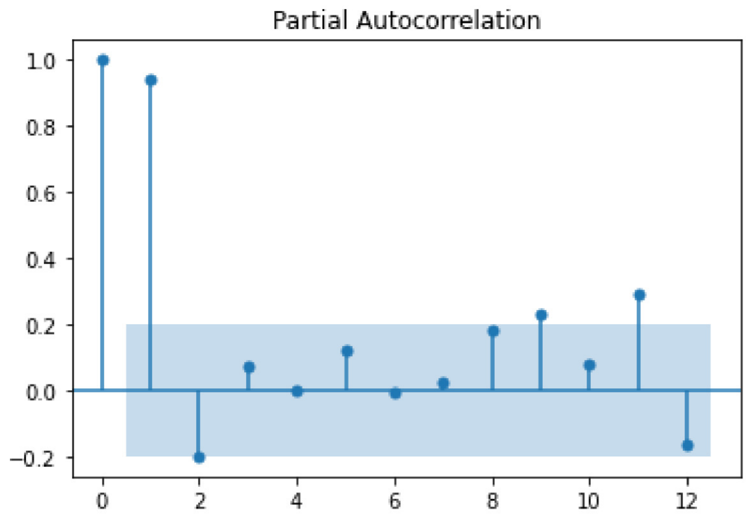


ACF Plot and PACF Plot

In [16]:

```
import statsmodels.graphics.tsaplots as tsa_plots
tsa_plots.plot_acf(airline_data['Passengers'],lags=12)
tsa_plots.plot_pacf(airline_data['Passengers'],lags=12)
plt.show()
```





Simple exponential method

In [17]:

```
train=airline_data.head(80)
test=airline_data.tail(15)
```

In [18]:

```
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
import numpy as np
from sklearn.metrics import mean_absolute_percentage_error

ses_model = SimpleExpSmoothing(train["Passengers"]).fit(smoothing_level=0.2)
pred_ses = ses_model.predict(start = test.index[0],end = test.index[-1])
ses_rms=mean_absolute_percentage_error(pred_ses,test['Passengers'])*100
ses_rms
```

Out[18]:

```
13.307089401087206
```

Holt method

In [19]:

```
from statsmodels.tsa.holtwinters import Holt

hw_model = Holt(train["Passengers"]).fit(smoothing_level=0.8, smoothing_slope=0.2)
pred_hw = hw_model.predict(start = test.index[0],end = test.index[-1])
hw_rms=mean_absolute_percentage_error(pred_hw,test['Passengers'])*100
hw_rms
```

Out[19]:

```
33.89294255735539
```

Holts winter exponential smoothing with multiplicative seasonality and additive trend

In [20]:

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

hwe_model_mul_add = ExponentialSmoothing(train["Passengers"],seasonal="mul",trend="add",sea
pred_hwe_mul_add = hwe_model_mul_add.predict(start = test.index[0],end = test.index[-1])

hw_ma_rms=mean_absolute_percentage_error(pred_hwe_mul_add,test['Passengers'])*100
hw_ma_rms
```

Out[20]:

```
3.5722884040046288
```

Holts winter exponential smoothing with additive seasonality and additive trend

In [21]:

```
hwe_model_add_add = ExponentialSmoothing(train["Passengers"],seasonal="add",trend="add",sea
pred_hwe_add_add = hwe_model_add_add.predict(start = test.index[0],end = test.index[-1])

hw_aa_rms=mean_absolute_percentage_error(pred_hwe_add_add,test['Passengers'])*100
hw_aa_rms
```

Out[21]:

```
7.414943343844639
```

ARMA Model

In [22]:

```
from statsmodels.tsa.arima_model import ARMA
```

In [23]:

```
ARMAmodel = ARMA(train['Passengers'], order=(1, 1)) #model with AR=0 and MA=1
ARMAmodel_fit = ARMAmodel.fit()

ARMA_pred = ARMAmodel_fit.predict(0,14)
ARMA_pred

arma_rms=mean_absolute_percentage_error(ARMA_pred,test['Passengers'])*100
arma_rms
```

Out[23]:

```
139.1255068537556
```

ARIMA Model

In [24]:

```
from statsmodels.tsa.arima_model import ARIMA
```

In [26]:

```
ARIMAmodel = ARIMA(train['Passengers'], order=(1, 1, 2)) #notice p,d and q value here
ARIMA_model_fit = ARIMAmodel.fit()

ARIMA_pred = ARIMA_model_fit.predict(1,15,typ='levels')

arima_rms=mean_absolute_percentage_error(ARIMA_pred,test['Passengers'])*100
arima_rms
```

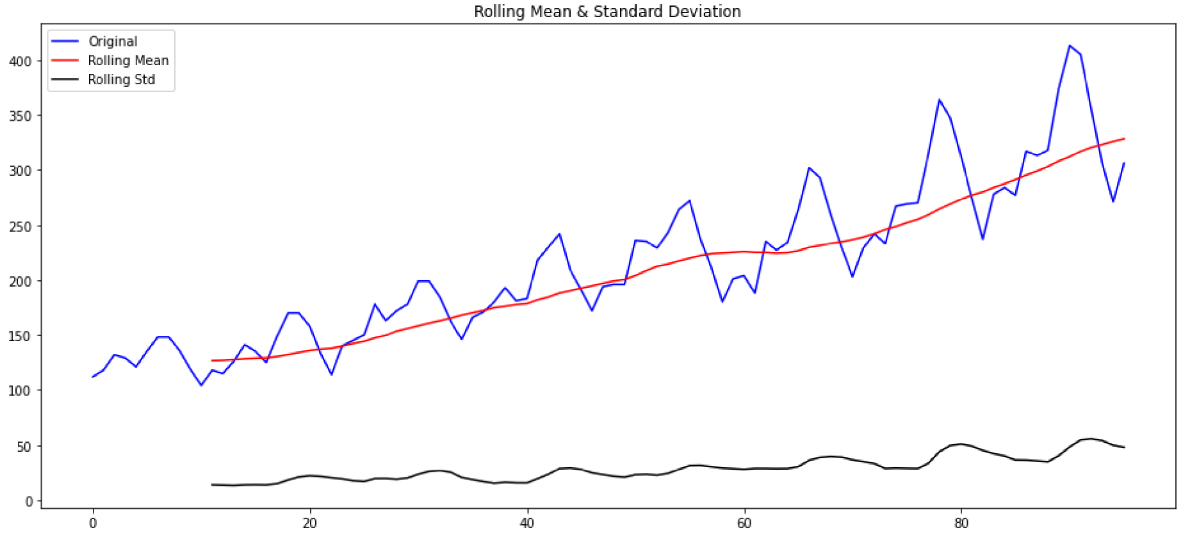
Out[26]:

```
143.7587075852369
```

Converting non stationary data to stationary data to improve ARIMA model

In [27]:

```
rolLmean = airline_data['Passengers'].rolling(12).mean() # 12 entries
rolLstd = airline_data['Passengers'].rolling(12).std()

plt.figure(figsize=(16,7))
fig = plt.figure(1)

#Plot rolling statistics:
orig = plt.plot(airline_data['Passengers'], color='blue',label='Original')
mean = plt.plot(rolLmean, color='red', label='Rolling Mean')
std = plt.plot(rolLstd, color='black', label = 'Rolling Std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard Deviation')
plt.show(block=False)
```



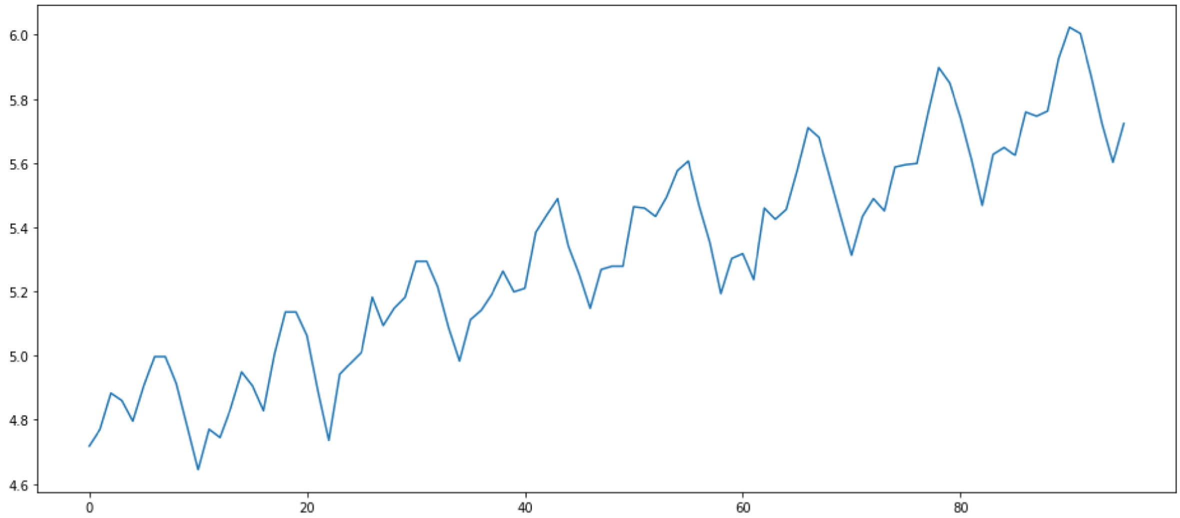
Making stationary

In [28]:

```
plt.figure(figsize=(16,7))
fig = plt.figure(1)

import numpy as np
ts_log = np.log(airline_data['Passengers'])#to transform to stationary from non-stationary
plt.plot(ts_log)
```

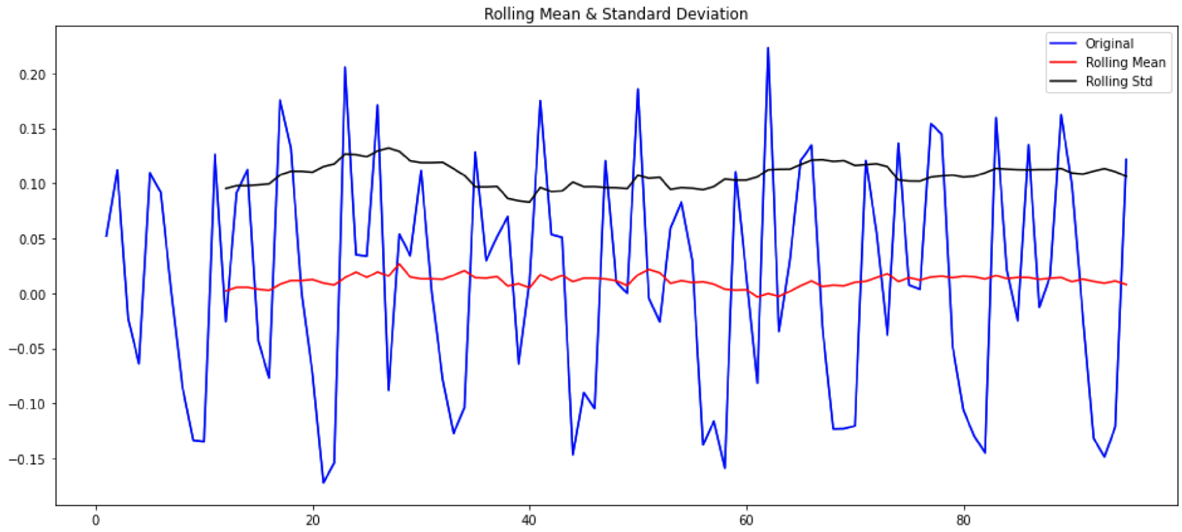
Out[28]:

```
[<matplotlib.lines.Line2D at 0x1612bfc0070>]
```



In [29]:

```
plt.figure(figsize=(16,7))
fig = plt.figure(1)
ts_log_diff = ts_log - ts_log.shift() # I will shift the time series by 1 and subtract from
plt.plot(ts_log_diff)

#Determing rolling statistics
rolLmean = ts_log_diff.rolling(12).mean()
rolLstd = ts_log_diff.rolling(12).std()



#Plot rolling statistics:
orig = plt.plot(ts_log_diff, color='blue',label='Original')
mean = plt.plot(rolLmean, color='red', label='Rolling Mean')
std = plt.plot(rolLstd, color='black', label = 'Rolling Std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard Deviation')
plt.show(block=False)
```



In [30]:

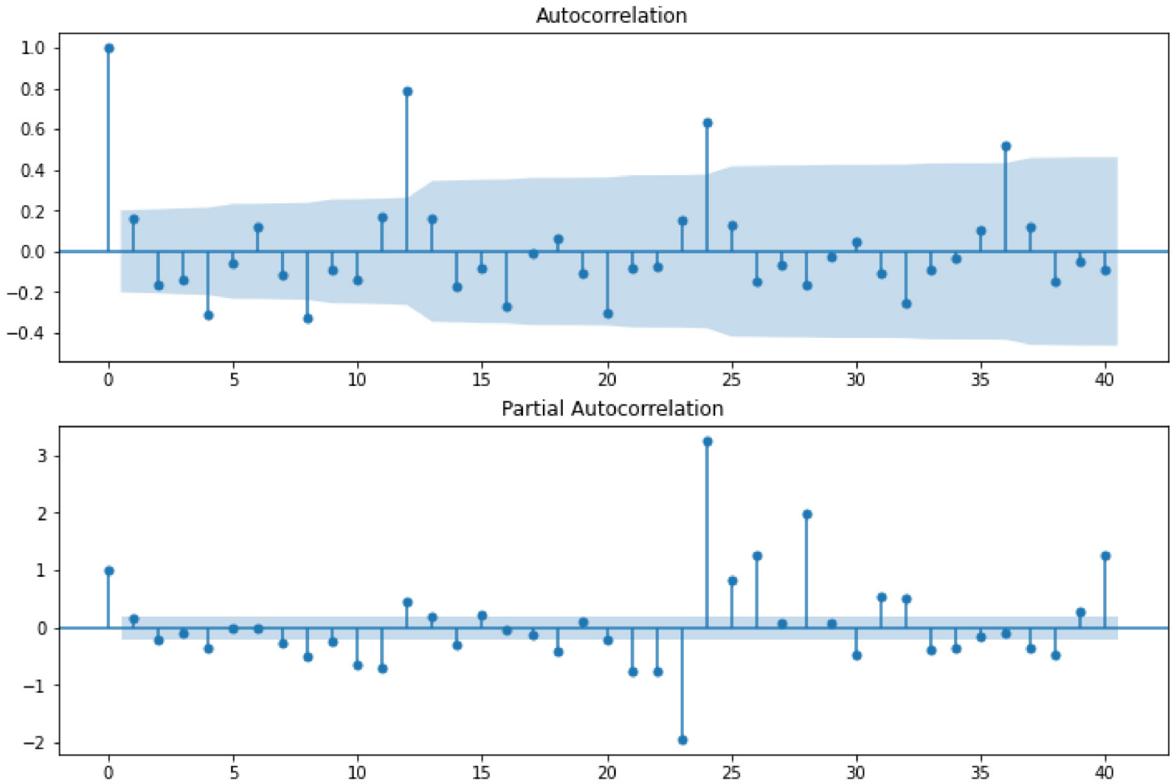
```
#after differencing, there is no pattern in the mean. No upward trend.No standard.
```

In [31]:

```
from statsmodels.tsa.stattools import acf, pacf
lag_acf = acf(ts_log_diff, nlags=12)
lag_pacf = pacf(ts_log_diff, nlags=12)
```

In [32]:

```
import statsmodels.api as sm
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(ts_log_diff.dropna(),lags=40,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(ts_log_diff.dropna(),lags=40,ax=ax2)
```



In [33]:

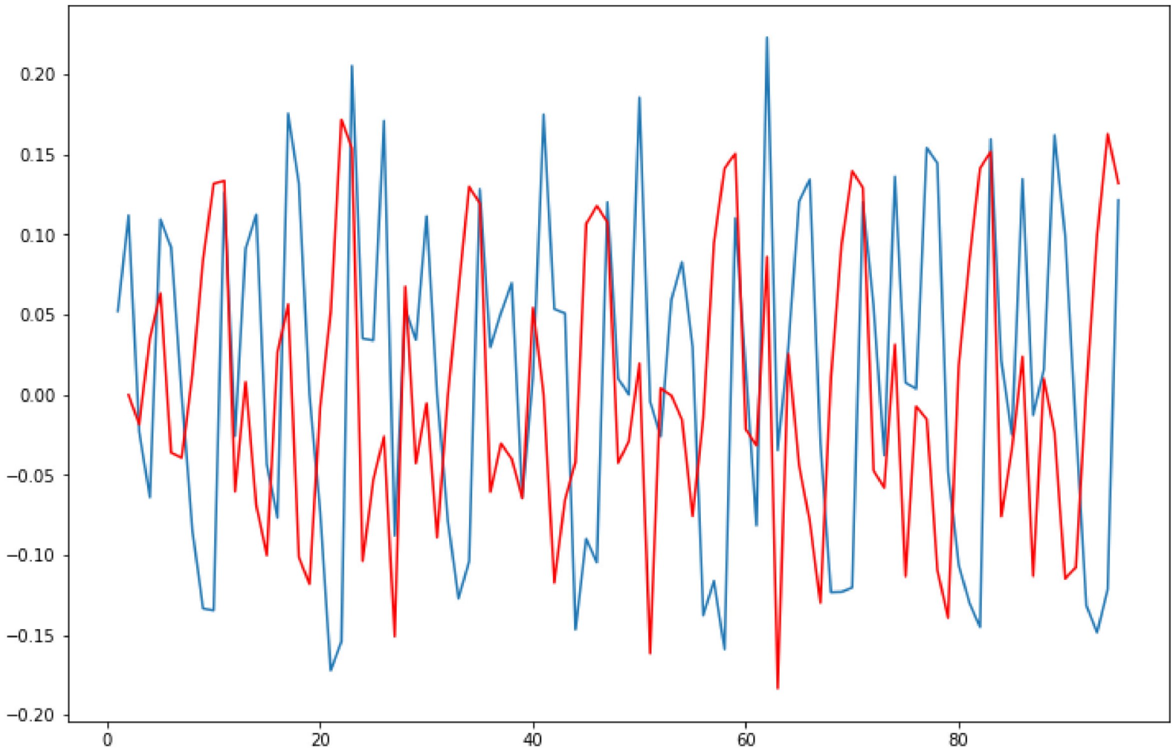
```
ts_log_diff = ts_log_diff[~ts_log_diff.isnull()]
```

In [116]:

```
plt.figure(figsize=(12,8))
ts_log_diff.dropna(inplace=True)
model = ARIMA(ts_log_diff, order=(4,1,2))
results_ARIMA = model.fit()
plt.plot(ts_log_diff)
plt.plot(results_ARIMA.fittedvalues, color='red')
```

Out[116]:

```
[<matplotlib.lines.Line2D at 0x1e6af082400>]
```



Model based Forcasting

In [34]:

```
airline_data_model=airline_data
```

In [35]:

```
airline_data_model.head()
```

Out[35]:

	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 [36]:

```
airline_data_model['Month']=airline_data_model['Month'].astype('str')
```

In [37]:

```
airline_data_model['Year']=0
for i in range(len(airline_data_model)):
    p=airline_data_model['Month'][i]
    airline_data_model['Year'][i]=p[0:4]
```

In [38]:

```
airline_data_model['Months']=0
for i in range(len(airline_data_model)):
    p=airline_data_model['Month'][i]
    airline_data_model['Months'][i]=p[5:7]
```

In [39]:

```
airline_data_model
```

Out[39]:

	Month	Passengers	Year	Months
0	1995-01-01	112	1995	1
1	1995-02-01	118	1995	2
2	1995-03-01	132	1995	3
3	1995-04-01	129	1995	4
4	1995-05-01	121	1995	5
...	...	...	...	...
91	2002-08-01	405	2002	8
92	2002-09-01	355	2002	9
93	2002-10-01	306	2002	10
94	2002-11-01	271	2002	11
95	2002-12-01	306	2002	12

96 rows × 4 columns

In [40]:

```
df_dummies = pd.DataFrame(pd.get_dummies(airline_data_model['Months']))
```

In [41]:

```
airlines_df =pd.concat([airline_data_model,df_dummies],axis= 1)
airlines_df
```

Out[41]:

	Month	Passengers	Year	Months	1	2	3	4	5	6	7	8	9	10	11	12
0	1995-01-01	112	1995	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1995-02-01	118	1995	2	0	1	0	0	0	0	0	0	0	0	0	0
2	1995-03-01	132	1995	3	0	0	1	0	0	0	0	0	0	0	0	0
3	1995-04-01	129	1995	4	0	0	0	1	0	0	0	0	0	0	0	0
4	1995-05-01	121	1995	5	0	0	0	0	1	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
91	2002-08-01	405	2002	8	0	0	0	0	0	0	0	1	0	0	0	0
92	2002-09-01	355	2002	9	0	0	0	0	0	0	0	0	1	0	0	0
93	2002-10-01	306	2002	10	0	0	0	0	0	0	0	0	0	1	0	0
94	2002-11-01	271	2002	11	0	0	0	0	0	0	0	0	0	0	1	0
95	2002-12-01	306	2002	12	0	0	0	0	0	0	0	0	0	0	0	1

96 rows × 16 columns

In [42]:

```
airlines_df['tsquare']=airlines_df['Months'].apply(lambda x:x**2)
```

In [43]:

```
from numpy import log
airlines_df['log_passengers']=airlines_df['Passengers'].apply(lambda x:log(x))
```

In [44]:

```
airlines_df
```

Out[44]:

	Month	Passengers	Year	Months	1	2	3	4	5	6	7	8	9	10	11	12	tsquare	l
0	1995-01-01	112	1995	1	1	0	0	0	0	0	0	0	0	0	0	0	1	
1	1995-02-01	118	1995	2	0	1	0	0	0	0	0	0	0	0	0	0	4	
2	1995-03-01	132	1995	3	0	0	1	0	0	0	0	0	0	0	0	0	9	
3	1995-04-01	129	1995	4	0	0	0	1	0	0	0	0	0	0	0	0	16	
4	1995-05-01	121	1995	5	0	0	0	0	1	0	0	0	0	0	0	0	25	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
91	2002-08-01	405	2002	8	0	0	0	0	0	0	0	1	0	0	0	0	64	
92	2002-09-01	355	2002	9	0	0	0	0	0	0	0	0	1	0	0	0	81	
93	2002-10-01	306	2002	10	0	0	0	0	0	0	0	0	0	1	0	0	100	
94	2002-11-01	271	2002	11	0	0	0	0	0	0	0	0	0	0	1	0	121	
95	2002-12-01	306	2002	12	0	0	0	0	0	0	0	0	0	0	0	1	144	

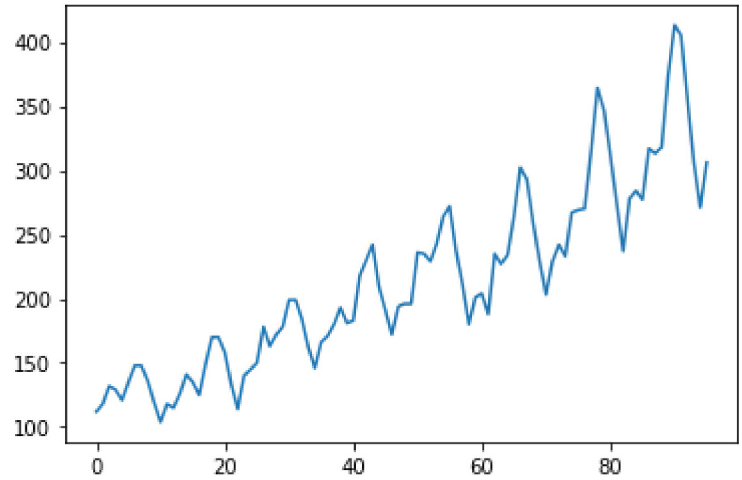
96 rows × 18 columns

In [45]:

```
import matplotlib.pyplot as plt
plt.plot(airlines_df['Passengers'])
```

Out[45]:

```
[<matplotlib.lines.Line2D at 0x1612c20aee0>]
```



In [63]:

```
airlines_df=airlines_df.rename(columns={1:'jan',2:'feb',3:'mar',4:'apr',5:'may',6:'jun',7:'
```

In [64]:

```
train_model=airlines_df.head(80)
test_model=airlines_df.tail(15)
```

Linear model

In [47]:

```
import statsmodels.formula.api as smf
from sklearn.metrics import mean_absolute_percentage_error

linear_model = smf.ols('Passengers~Months',data=train_model).fit()
linear_pred=linear_model.predict(test_model['Months'])

linear_rms=mean_absolute_percentage_error(test_model['Passengers'],linear_pred)*100
linear_rms
```

Out[47]:

```
37.13562686329391
```

Exponential model

In [48]:

```
exponential_model = smf.ols('log_passengers~Months',data=train_model).fit()
exponential_pred=exponential_model.predict(test_model['Months'])

exp_rms=mean_absolute_percentage_error(test_model['Passengers'],exponential_pred)*100
exp_rms
```

Out[48]:

98.30419643524579

In [49]:

```
train_model
```

Out[49]:

	Month	Passengers	Year	Months	1	2	3	4	5	6	7	8	9	10	11	12	tsquare	l
0	1995-01-01	112	1995	1	1	0	0	0	0	0	0	0	0	0	0	0	1	
1	1995-02-01	118	1995	2	0	1	0	0	0	0	0	0	0	0	0	0	4	
2	1995-03-01	132	1995	3	0	0	1	0	0	0	0	0	0	0	0	0	9	
3	1995-04-01	129	1995	4	0	0	0	1	0	0	0	0	0	0	0	0	16	
4	1995-05-01	121	1995	5	0	0	0	0	1	0	0	0	0	0	0	0	25	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
75	2001-04-01	269	2001	4	0	0	0	1	0	0	0	0	0	0	0	0	16	
76	2001-05-01	270	2001	5	0	0	0	0	1	0	0	0	0	0	0	0	25	
77	2001-06-01	315	2001	6	0	0	0	0	0	1	0	0	0	0	0	0	36	
78	2001-07-01	364	2001	7	0	0	0	0	0	0	1	0	0	0	0	0	49	
79	2001-08-01	347	2001	8	0	0	0	0	0	0	0	1	0	0	0	0	64	

80 rows × 18 columns

In [50]:

```
quaratic_model = smf.ols('Passengers~(Months+tsquare)',data=train_model).fit()
#quaratic_model.fit()
quaratic_pred=quaratic_model.predict(test_model[['Months','tsquare']])

qua_rms=mean_absolute_percentage_error(test_model['Passengers'],quaratic_pred)*100
```

In [65]:

```
add_sea = smf.ols("Passengers~jan+feb+mar+apr+may+jun+july+aug+sep+oct+nov+dec",data=train_
add_pred=add_sea.predict(test_model[['jan','feb','mar','apr','may','jun','july','aug','sep'

add_rms=mean_absolute_percentage_error(test_model['Passengers'],add_pred)*100
add_rms
```

Out[65]:

40.28314165835847

In [66]:

```
add_qua_model=smf.ols("Passengers~Months+tsquare+jan+feb+mar+apr+may+jun+july+aug+sep+oct+n
add_qua_pred=add_qua_model.predict(test_model[['Months','tsquare','jan','feb','mar','apr','

add_qua_rms=mean_absolute_percentage_error(test_model['Passengers'],add_qua_pred)*100
add_qua_rms
```

Out[66]:

40.28314165837216

In [67]:

```
mul_sea = smf.ols("log_passengers~jan+feb+mar+apr+may+jun+july+aug+sep+oct+nov+dec",data=tr
mul_pred=mul_sea.predict(test_model[['jan','feb','mar','apr','may','jun','july','aug','sep'

mul_rms=mean_absolute_percentage_error(test_model['Passengers'],mul_pred)*100
mul_rms
```

Out[67]:

98.31984707397673

In [68]:

```
mul_add_model=smf.ols("Passengers~Months+jan+feb+mar+apr+may+jun+july+aug+sep+oct+nov+dec",
mul_add_pred=mul_add_model.predict(test_model[['Months','jan','feb','mar','apr','may','jun'

mul_add_rms=mean_absolute_percentage_error(test_model['Passengers'],mul_add_pred)*100
mul_add_rms
```

Out[68]:

40.283141658358495

In [69]:

```
data = {"MODEL":pd.Series(["rmse_linear","rmse_Exp","rmse_Quad","rmse_add_sea","rmse_add_se
        "RMSE_Values":pd.Series([linear_rms,exp_rms,qua_rms,add_rms,add_qua_rms,mul_rms,mul
table_rmse=pd.DataFrame(data)
table_rmse.sort_values(['RMSE_Values'])
```

Out[69]:

	MODEL	RMSE_Values
9	rmse_holt_ma	3.572288
10	rmse_holt_aa	7.414943
7	rmse_ses	13.307089
8	rmse_holt	33.892943
0	rmse_linear	37.135627
2	rmse_Quad	39.211566
3	rmse_add_sea	40.283142
6	rmse_Mult_add_sea	40.283142
4	rmse_add_sea_quad	40.283142
1	rmse_Exp	98.304196
5	rmse_Mult_sea	98.319847
11	rmse_arma	139.125507
12	rmse_arima	143.758708

Inference

The mean absolute percentage error of Holts winter exponential smoothing with multiplicative seasonality and additive trend model is comparitively low compared to all other model

Hence, the forecast model can be built by Holts winter exponential smoothing with multiplicative seasonality and additive trend

Final model

In [70]:

```
hwe_model_mul_add = ExponentialSmoothing(train["Passengers"],seasonal="mul",trend="add",sea
```

In [71]:

```
hwe_model_mul_add.forecast(10)
```

Out[71]:

```
80    310.168225
81    273.379578
82    239.340177
83    270.784249
84    276.723737
85    274.061717
86    317.646102
87    307.800973
88    306.489927
89    343.885684
dtype: float64
```