Problem Statement:

Forecast Airlines Passengers data set. Prepare a document for each model explaining how many dummy variables you have created and RMSE value for each model. Finally which model you will use for Forecasting.

1. Import Necessary Libraries

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

2. Import Data

In [3]:
```
Airline_forecast = pd.read_csv('Airlines+Data.csv')
Airline_forecast
```

Out[3]:

	Month	Passengers
0	Jan-95	112
1	Feb-95	118
2	Mar-95	132
3	Apr-95	129
4	May-95	121
...	...	...
91	Aug-02	405
92	Sep-02	355
93	Oct-02	306
94	Nov-02	271
95	Dec-02	306

96 rows × 2 columns

3. Data Understanding

In [4]:
```
Airline_forecast.shape
```

Out[4]: (96, 2)

In [5]:
```
Airline_forecast.dtypes
```

Out[5]:
```
Month         object
Passengers     int64
dtype: object
```

In [6]:
```
Airline_forecast.describe()
```

Out[6]:

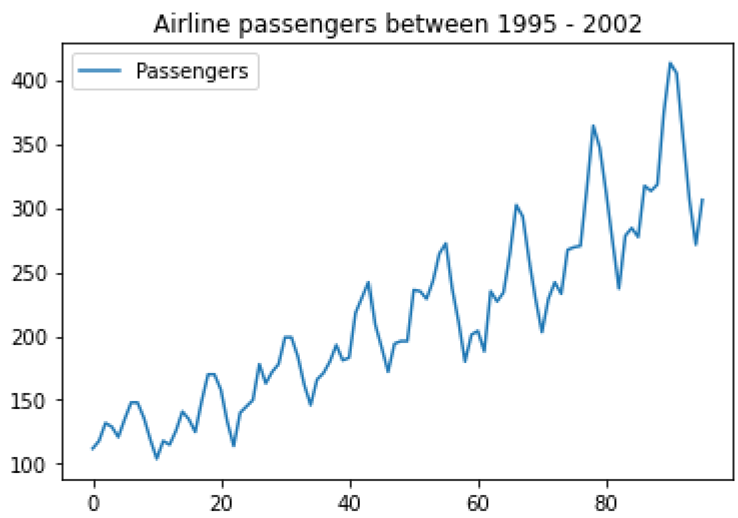
	Passengers
count	96.000000
mean	213.708333
std	71.918216
min	104.000000
25%	156.000000
50%	200.000000
75%	264.750000
max	413.000000

In [7]:
```
Airline_forecast.isna().sum()
```

Out[7]:
```
Month         0
Passengers    0
dtype: int64
```

4. Data Preparation

In [8]:
```
Airline_forecast.plot()
plt.title('Airline passengers between 1995 - 2002')
plt.show()
```



**** Creating dummy variables****

In [9]:
```
Months = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'
n = Airline_forecast['Month'][0]
n[0:3]
```

Out[9]:
```
'Jan'
```

In [10]:
```
Airline_forecast['Months'] = 0
Airline_forecast['Months']
```

Out[10]:
```
0     0
1     0
2     0
3     0
4     0
     ..
91    0
92    0
93    0
94    0
95    0
Name: Months, Length: 96, dtype: int64
```

In [11]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [12]:
```
for i in range(96):
    n=Airline_forecast['Month'][i]
    Airline_forecast['Months'][i]=n[0:3]
Airline_forecast['Months']
```

Out[12]:
```
0     Jan
1     Feb
2     Mar
3     Apr
4     May
     ...
91    Aug
92    Sep
93    Oct
94    Nov
95    Dec
Name: Months, Length: 96, dtype: object
```

In [13]:
```
dummy = pd.DataFrame(pd.get_dummies(Airline_forecast['Months']))
dummy
```

Out[13]:

	Apr	Aug	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov	Oct	Sep
0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
91	0	1	0	0	0	0	0	0	0	0	0	0
92	0	0	0	0	0	0	0	0	0	0	0	1
93	0	0	0	0	0	0	0	0	0	0	1	0
94	0	0	0	0	0	0	0	0	0	1	0	0
95	0	0	1	0	0	0	0	0	0	0	0	0

96 rows × 12 columns

In [14]:
```
print(dummy.columns)
```

```
Index(['Apr', 'Aug', 'Dec', 'Feb', 'Jan', 'Jul', 'Jun', 'Mar', 'May', 'Nov',
       'Oct', 'Sep'],
      dtype='object')
```

In [15]:
```
Airline_forecast.dtypes
```

Out[15]:
```
Month          object
Passengers      int64
Months         object
dtype: object
```

In [16]:
```
Airlines = pd.concat((Airline_forecast,dummy), axis = 1)
t = np.arange(1,97)
Airlines['t'] = t
Airlines['t_square'] = Airlines['t']*Airlines['t']
Airlines
```

Out[16]:

	Month	Passengers	Months	Apr	Aug	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov	Oct	Sep
0	Jan-95	112	Jan	0	0	0	0	1	0	0	0	0	0	0	0
1	Feb-95	118	Feb	0	0	0	1	0	0	0	0	0	0	0	0
2	Mar-95	132	Mar	0	0	0	0	0	0	0	1	0	0	0	0
3	Apr-95	129	Apr	1	0	0	0	0	0	0	0	0	0	0	0
4	May-95	121	May	0	0	0	0	0	0	0	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
91	Aug-02	405	Aug	0	1	0	0	0	0	0	0	0	0	0	0
92	Sep-02	355	Sep	0	0	0	0	0	0	0	0	0	0	0	1
93	Oct-02	306	Oct	0	0	0	0	0	0	0	0	0	0	1	0
94	Nov-02	271	Nov	0	0	0	0	0	0	0	0	0	1	0	0
95	Dec-02	306	Dec	0	0	1	0	0	0	0	0	0	0	0	0

96 rows × 17 columns

In [17]:
```
Airlines.dtypes
```

Out[17]:
```
Month          object
Passengers      int64
Months         object
Apr             uint8
Aug             uint8
Dec             uint8
Feb             uint8
Jan             uint8
Jul             uint8
Jun             uint8
Mar             uint8
May             uint8
Nov             uint8
Oct             uint8
Sep             uint8
t               int32
t_square        int32
dtype: object
```

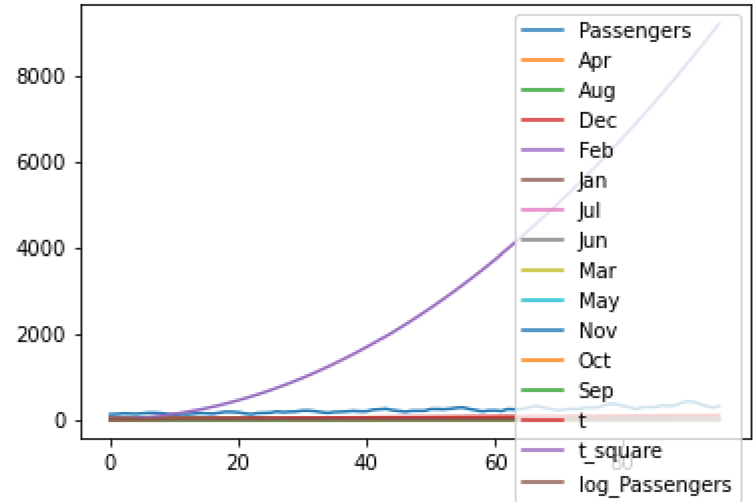
In [18]:
```
log_Passengers = np.log(Airlines['Passengers'])
Airlines['log_Passengers'] = log_Passengers
Airlines
```

Out[18]:

	Month	Passengers	Months	Apr	Aug	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov	Oct	Sep
0	Jan-95	112	Jan	0	0	0	0	1	0	0	0	0	0	0	0
1	Feb-95	118	Feb	0	0	0	1	0	0	0	0	0	0	0	0
2	Mar-95	132	Mar	0	0	0	0	0	0	0	1	0	0	0	0
3	Apr-95	129	Apr	1	0	0	0	0	0	0	0	0	0	0	0
4	May-95	121	May	0	0	0	0	0	0	0	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
91	Aug-02	405	Aug	0	1	0	0	0	0	0	0	0	0	0	0
92	Sep-02	355	Sep	0	0	0	0	0	0	0	0	0	0	0	1
93	Oct-02	306	Oct	0	0	0	0	0	0	0	0	0	0	1	0
94	Nov-02	271	Nov	0	0	0	0	0	0	0	0	0	1	0	0
95	Dec-02	306	Dec	0	0	1	0	0	0	0	0	0	0	0	0

96 rows × 18 columns

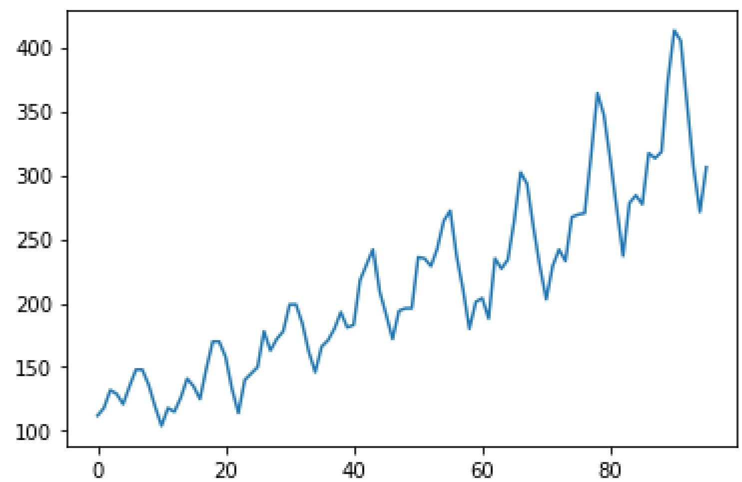
In [19]:
```
Airlines.plot()
plt.show()
```



5. Model Building - Train Test Split

In [20]:
```
train = Airlines.head(92)
test = Airlines.tail(4)
Airlines.Passengers.plot()
```

Out[20]: <AxesSubplot:>



6. Model Training, Testing & Evaluation

In [21]:
```
import statsmodels.formula.api as smf
```

1. Model Based Forecasting Techniques

1. For Linear Model

1. For Linear Model

In [22]:
```
linear= smf.ols('Passengers~t',data=train).fit()
predlin=pd.Series(linear.predict(pd.DataFrame(test['t'])))
rmselin=np.sqrt((np.mean(np.array(test['Passengers'])-np.array(predlin))**2))
rmselin
```

Out[22]: 13.834024320700323

2. For Exponential Model

In [23]:
```
expo=smf.ols('log_Passengers~t',data=train).fit()
predexp=pd.Series(expo.predict(pd.DataFrame(test['t'])))
predexp
rmseexpo=np.sqrt(np.mean((np.array(test['Passengers'])-np.array(np.exp(predexp))
rmseexpo
```

Out[23]: 47.528807018553685

3.For Quadratic Model

In [24]:
```
quad=smf.ols('Passengers~t+t_square',data=train).fit()
predquad=pd.Series(quad.predict(pd.DataFrame(test[['t','t_square']])))
rmsequad=np.sqrt(np.mean((np.array(test['Passengers'])-np.array(predquad))**2))
rmsequad
```

Out[24]: 51.289266550795745

4. For Additive Seasonality

In [25]:
```
additive= smf.ols('Passengers~Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+Nov+Dec', d
predadd=pd.Series(additive.predict(pd.DataFrame(test[['Jan','Feb','Mar','Apr','Ma
rmseadd = np.sqrt(np.mean((np.array(test['Passengers'])-np.array(predadd))**2))
rmseadd
```

Out[25]: 121.057542397938

5. For Additive Seasonality With Quadratic Trend

In [26]:
```
additivequad= smf.ols('Passengers~t+t_square+Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+
predaddquad=pd.Series(additivequad.predict(pd.DataFrame(test[['t', 't_square', 'J
rmseaddquad = np.sqrt(np.mean((np.array(test['Passengers'])-np.array(predaddquad)
rmseaddquad
```

Out[26]: 20.02636369707501

6. For Multiplicative Seasonality

In [27]:
```
Multiplicative= smf.ols('log_Passengers~Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+N
predmul=pd.Series(Multiplicative.predict(pd.DataFrame(test[['Jan','Feb','Mar','Ap
rmsemul = np.sqrt(np.mean((np.array(test['Passengers'])-np.array(predmul))**2))
rmsemul
```

Out[27]:
```
305.75515002909657
```

7. Tabulating the RMSE values|

In [28]:
```
data = {'Model':pd.Series(['rmselin', 'rmseexpo', 'rmsequad', 'rmseadd', 'rmseadd
data
```

Out[28]:
```
{'Model': 0        rmselin
 1       rmseexpo
 2       rmsequad
 3        rmseadd
 4    rmseaddquad
 5        rmsemul
 dtype: object,
 'Values': 0     13.834024
 1     47.528807
 2     51.289267
 3    121.057542
 4     20.026364
 5    305.755150
 dtype: float64}
```

In [29]:
```
RMSE = pd.DataFrame(data)
data
```

Out[29]:
```
{'Model': 0        rmselin
 1       rmseexpo
 2       rmsequad
 3        rmseadd
 4    rmseaddquad
 5        rmsemul
 dtype: object,
 'Values': 0     13.834024
 1     47.528807
 2     51.289267
 3    121.057542
 4     20.026364
 5    305.755150
 dtype: float64}
```

CONCLUSION:

Linear Model is the Best model compared to all the other Model based Forecasting Techniques.

In [30]:
```
Airlines
```

Out[30]:

	Month	Passengers	Months	Apr	Aug	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov	Oct	Sep
0	Jan-95	112	Jan	0	0	0	0	1	0	0	0	0	0	0	0
1	Feb-95	118	Feb	0	0	0	1	0	0	0	0	0	0	0	0
2	Mar-95	132	Mar	0	0	0	0	0	0	0	1	0	0	0	0
3	Apr-95	129	Apr	1	0	0	0	0	0	0	0	0	0	0	0
4	May-95	121	May	0	0	0	0	0	0	0	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
91	Aug-02	405	Aug	0	1	0	0	0	0	0	0	0	0	0	0
92	Sep-02	355	Sep	0	0	0	0	0	0	0	0	0	0	0	1
93	Oct-02	306	Oct	0	0	0	0	0	0	0	0	0	0	1	0
94	Nov-02	271	Nov	0	0	0	0	0	0	0	0	0	1	0	0
95	Dec-02	306	Dec	0	0	1	0	0	0	0	0	0	0	0	0

96 rows × 18 columns

8. Building model by using entire data of Linear model

In [31]:
```
Model_full =smf.ols('Passengers~t',data=Airline_forecast).fit()
```

In [32]:
```
Pred_new = Model_full.predict(Airlines)
Pred_new
```

Out[32]:
```
0      102.809493
1      105.144206
2      107.478918
3      109.813630
4      112.148343
          ...
91     315.268324
92     317.603036
93     319.937749
94     322.272461
95     324.607174
Length: 96, dtype: float64
```

In [33]:
```
Airlines["Forcasted_Airlines_Data"] = Pred_new
Airlines
```

Out[33]:

	Month	Passengers	Months	Apr	Aug	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov	Oct	Sep
0	Jan-95	112	Jan	0	0	0	0	1	0	0	0	0	0	0	0
1	Feb-95	118	Feb	0	0	0	1	0	0	0	0	0	0	0	0
2	Mar-95	132	Mar	0	0	0	0	0	0	0	1	0	0	0	0
3	Apr-95	129	Apr	1	0	0	0	0	0	0	0	0	0	0	0
4	May-95	121	May	0	0	0	0	0	0	0	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
91	Aug-02	405	Aug	0	1	0	0	0	0	0	0	0	0	0	0
92	Sep-02	355	Sep	0	0	0	0	0	0	0	0	0	0	0	1
93	Oct-02	306	Oct	0	0	0	0	0	0	0	0	0	0	1	0
94	Nov-02	271	Nov	0	0	0	0	0	0	0	0	0	1	0	0
95	Dec-02	306	Dec	0	0	1	0	0	0	0	0	0	0	0	0

96 rows × 19 columns

In [34]:
```
new_var = pd.concat([Airline_forecast,Airlines])
new_var
```

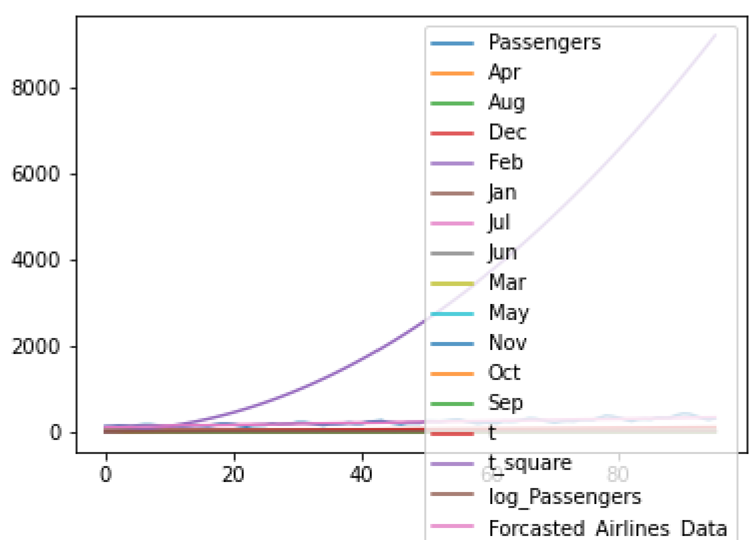
Out[34]:

	Month	Passengers	Months	Apr	Aug	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov	Oct
0	Jan-95	112	Jan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Feb-95	118	Feb	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Mar-95	132	Mar	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Apr-95	129	Apr	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	May-95	121	May	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
91	Aug-02	405	Aug	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
92	Sep-02	355	Sep	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
93	Oct-02	306	Oct	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
94	Nov-02	271	Nov	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
95	Dec-02	306	Dec	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

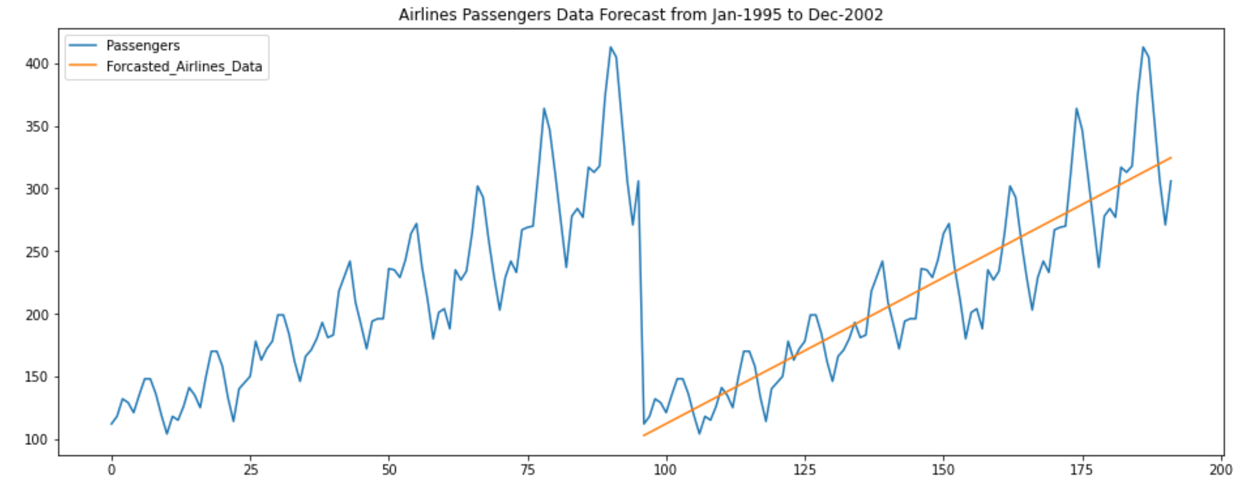
192 rows × 19 columns

In [35]:
```
Airlines.plot()
```

Out[35]: <AxesSubplot:>



In [36]:
```
new_var[['Passengers', 'Forcasted_Airlines_Data']].reset_index(drop=True).plot(fi
plt.title('Airlines Passengers Data Forecast from Jan-1995 to Dec-2002')
plt.show()
```



2. Data Driven Forecasting Technique

1. Import Necessary Libraries

In [37]:
```
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import Holt
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
import statsmodels.graphics.tsaplots as tsa_plots
import statsmodels.tsa.statespace as tm_models
from datetime import datetime,time
```

2. Import Data

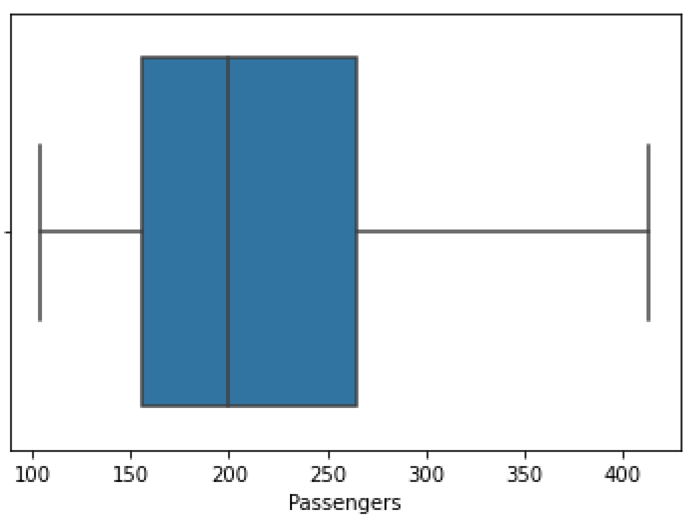
In [38]:
```
DD_Airlines = pd.read_csv('Airlines+Data.csv')
DD_Airlines
```

Out[38]:

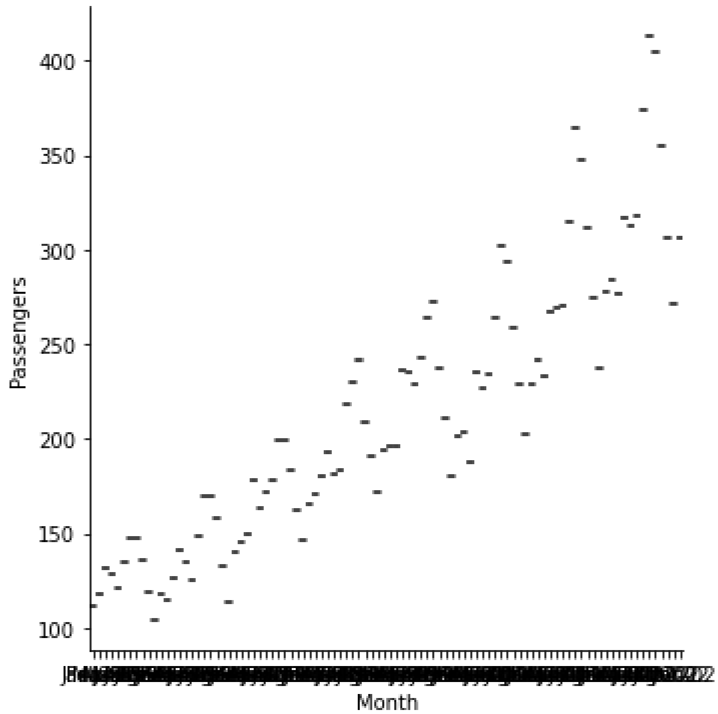
	Month	Passengers
0	Jan-95	112
1	Feb-95	118
2	Mar-95	132
3	Apr-95	129
4	May-95	121
...	...	...
91	Aug-02	405
92	Sep-02	355
93	Oct-02	306
94	Nov-02	271
95	Dec-02	306

96 rows × 2 columns

In [39]:
```
sns.boxplot( 'Passengers', data=DD_Airlines)
plt.show()
```

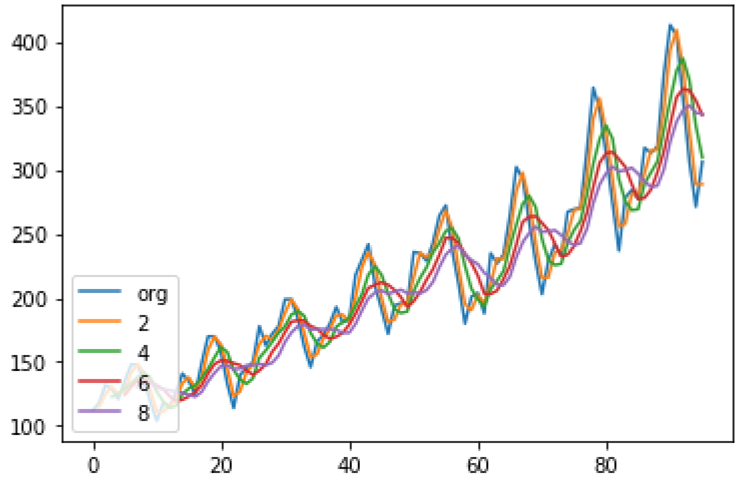


In [40]:
```
sns.factorplot('Month', 'Passengers', data = DD_Airlines, kind = "box")
plt.show()
```



Moving Average

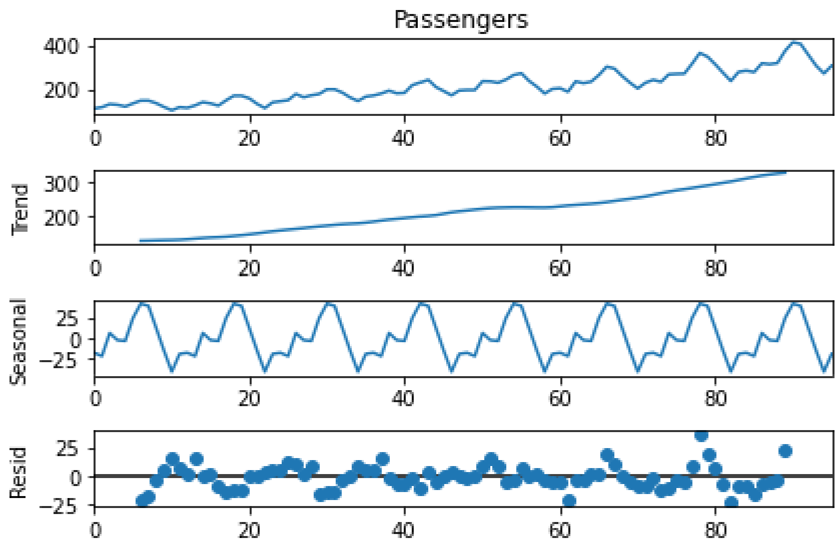
In [41]:
```
DD_Airlines.Passengers.plot(label="org")
for i in range(2,10,2):
    DD_Airlines["Passengers"].rolling(i).mean().plot(label=str(i))
plt.legend(loc=3)
plt.show()
```

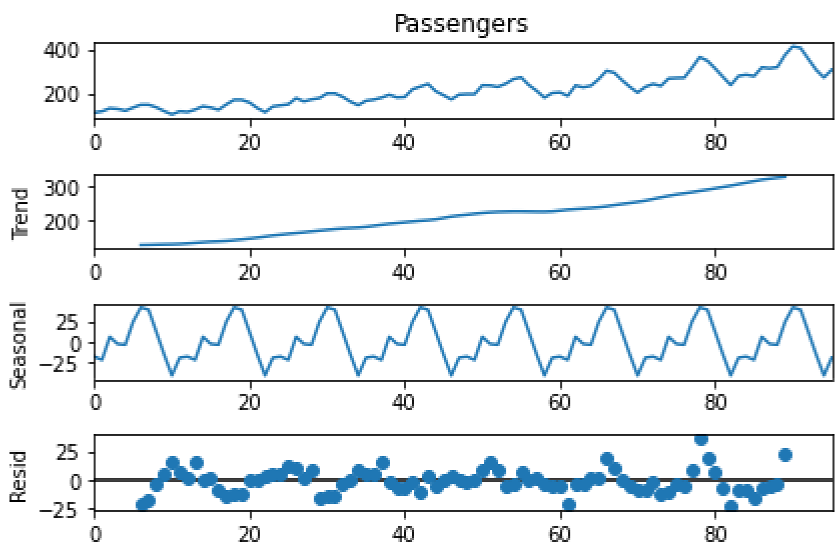


Time series decomposition plot

In [42]:
```
Decompose_ts_ad = seasonal_decompose(DD_Airlines.Passengers, period=12)
Decompose_ts_ad.plot()
```

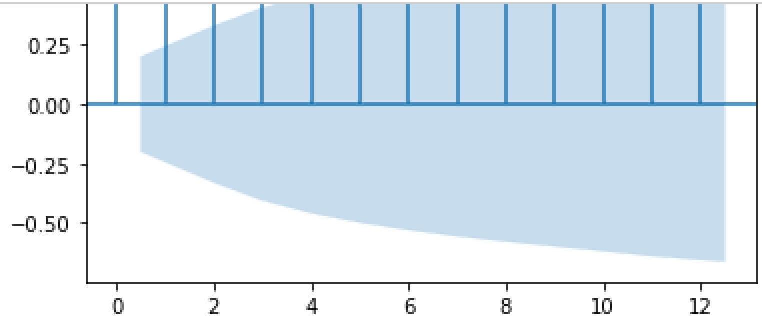
Out[42]:

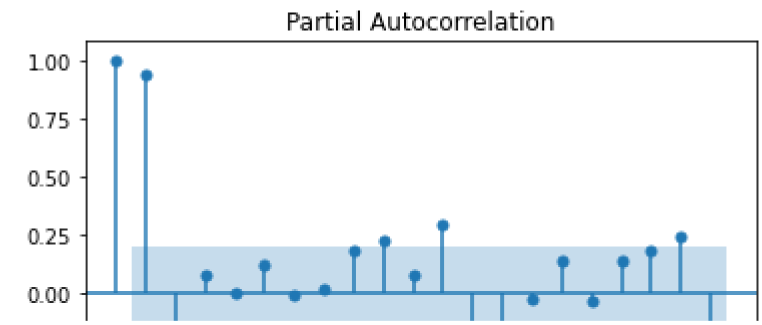




ACF plots & PACF plots on original data sets

In [43]:
```
tsa_plots.plot_acf(DD_Airlines.Passengers, lags=12)
tsa_plots.plot_pacf(DD_Airlines.Passengers)
```





Train test split

In [44]:
```
train = DD_Airlines.head(100)
test = DD_Airlines.tail(20)
```

Creating function to calculate the MAPE vales for the test data

In [45]:
```
def MAPE(pred,org):
    temp = np.abs((pred-org))*100/org
    return np.mean(temp)
```

Data Driven Forecasting Methods

1. Simple exponential method

In [46]:
```
ses_model = SimpleExpSmoothing(train['Passengers']).fit()
pred_ses = ses_model.predict(start=test.index[0], end = test.index[-1])
MAPE(pred_ses,test.Passengers)
```

Out[46]: 9.470697707516285

2. Holts method

In [47]:
```
hw_model = Holt(train['Passengers']).fit()
pred_hw = hw_model.predict(start=test.index[0], end = test.index[-1])
MAPE(pred_hw,test.Passengers)
```

Out[47]: 9.525737135381483

3. Holts winter exponential smoothing

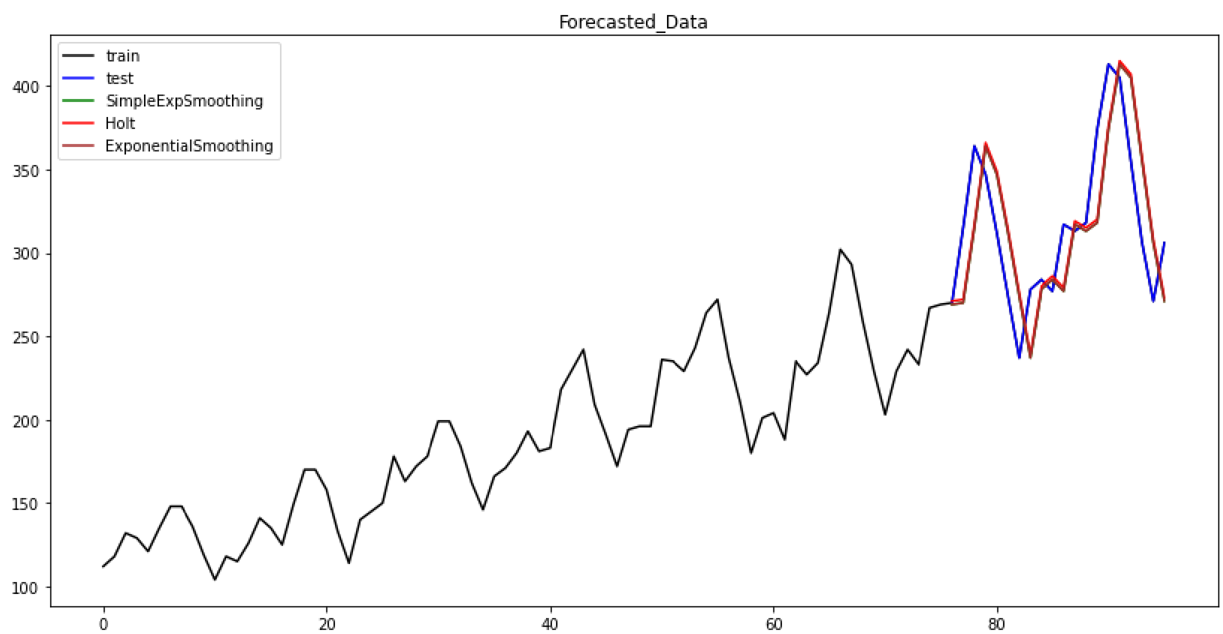
In [50]:
```
holts_w_model = ExponentialSmoothing(train['Passengers']).fit()
pred_holts_w = holts_w_model.predict(start=test.index[0], end = test.index[-1])
MAPE(pred_holts_w,test.Passengers)
```

Out[50]: 9.470697707516285

Conclusion:- Simple and Holts winter exponential smoothing is best model compare to holts method

Visualization of forecasted values for test data set using different methods

In [51]:
```
plt.figure(figsize=(14,7))
plt.plot(train.index, train["Passengers"], label='train',color="black")
plt.plot(test.index, test["Passengers"], label='test',color="blue")
plt.plot(pred_ses.index, pred_ses, label='SimpleExpSmoothing',color="green")
plt.plot(pred_hw.index, pred_hw, label='Holt',color="red")
plt.plot(pred_holts_w.index,pred_holts_w,label="ExponentialSmoothing",color="brow
plt.legend(loc='best')
plt.title("Forecasted_Data")
plt.show()
```



THE END

In [ ]: