In [1]:

```
# load libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [3]:

```
# Import dataset
airline = pd.read_excel('Airlines+Data.xlsx')
airline.head()
```

Out[3]:

	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 [4]:

```
airline.isna().sum()
```

Out[4]:

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

In [5]:

```
airline.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96 entries, 0 to 95
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   Month       96 non-null     datetime64[ns]
 1   Passengers  96 non-null     int64         
dtypes: datetime64[ns](1), int64(1)
memory usage: 1.6 KB
```

In [6]:

```
airline['Month'].unique()
```

Out[6]:

```
array(['1995-01-01T00:00:00.000000000', '1995-02-01T00:00:00.000000000',
       '1995-03-01T00:00:00.000000000', '1995-04-01T00:00:00.000000000',
       '1995-05-01T00:00:00.000000000', '1995-06-01T00:00:00.000000000',
       '1995-07-01T00:00:00.000000000', '1995-08-01T00:00:00.000000000',
       '1995-09-01T00:00:00.000000000', '1995-10-01T00:00:00.000000000',
       '1995-11-01T00:00:00.000000000', '1995-12-01T00:00:00.000000000',
       '1996-01-01T00:00:00.000000000', '1996-02-01T00:00:00.000000000',
       '1996-03-01T00:00:00.000000000', '1996-04-01T00:00:00.000000000',
       '1996-05-01T00:00:00.000000000', '1996-06-01T00:00:00.000000000',
       '1996-07-01T00:00:00.000000000', '1996-08-01T00:00:00.000000000',
       '1996-09-01T00:00:00.000000000', '1996-10-01T00:00:00.000000000',
       '1996-11-01T00:00:00.000000000', '1996-12-01T00:00:00.000000000',
       '1997-01-01T00:00:00.000000000', '1997-02-01T00:00:00.000000000',
       '1997-03-01T00:00:00.000000000', '1997-04-01T00:00:00.000000000',
       '1997-05-01T00:00:00.000000000', '1997-06-01T00:00:00.000000000',
       '1997-07-01T00:00:00.000000000', '1997-08-01T00:00:00.000000000',
       '1997-09-01T00:00:00.000000000', '1997-10-01T00:00:00.000000000',
       '1997-11-01T00:00:00.000000000', '1997-12-01T00:00:00.000000000',
       '1998-01-01T00:00:00.000000000', '1998-02-01T00:00:00.000000000',
       '1998-03-01T00:00:00.000000000', '1998-04-01T00:00:00.000000000',
       '1998-05-01T00:00:00.000000000', '1998-06-01T00:00:00.000000000',
       '1998-07-01T00:00:00.000000000', '1998-08-01T00:00:00.000000000',
       '1998-09-01T00:00:00.000000000', '1998-10-01T00:00:00.000000000',
       '1998-11-01T00:00:00.000000000', '1998-12-01T00:00:00.000000000',
       '1999-01-01T00:00:00.000000000', '1999-02-01T00:00:00.000000000',
       '1999-03-01T00:00:00.000000000', '1999-04-01T00:00:00.000000000',
       '1999-05-01T00:00:00.000000000', '1999-06-01T00:00:00.000000000',
       '1999-07-01T00:00:00.000000000', '1999-08-01T00:00:00.000000000',
       '1999-09-01T00:00:00.000000000', '1999-10-01T00:00:00.000000000',
       '1999-11-01T00:00:00.000000000', '1999-12-01T00:00:00.000000000',
       '2000-01-01T00:00:00.000000000', '2000-02-01T00:00:00.000000000',
       '2000-03-01T00:00:00.000000000', '2000-04-01T00:00:00.000000000',
       '2000-05-01T00:00:00.000000000', '2000-06-01T00:00:00.000000000',
       '2000-07-01T00:00:00.000000000', '2000-08-01T00:00:00.000000000',
       '2000-09-01T00:00:00.000000000', '2000-10-01T00:00:00.000000000',
       '2000-11-01T00:00:00.000000000', '2000-12-01T00:00:00.000000000',
       '2001-01-01T00:00:00.000000000', '2001-02-01T00:00:00.000000000',
       '2001-03-01T00:00:00.000000000', '2001-04-01T00:00:00.000000000',
       '2001-05-01T00:00:00.000000000', '2001-06-01T00:00:00.000000000',
       '2001-07-01T00:00:00.000000000', '2001-08-01T00:00:00.000000000',
       '2001-09-01T00:00:00.000000000', '2001-10-01T00:00:00.000000000',
       '2001-11-01T00:00:00.000000000', '2001-12-01T00:00:00.000000000',
       '2002-01-01T00:00:00.000000000', '2002-02-01T00:00:00.000000000',
       '2002-03-01T00:00:00.000000000', '2002-04-01T00:00:00.000000000',
       '2002-05-01T00:00:00.000000000', '2002-06-01T00:00:00.000000000',
       '2002-07-01T00:00:00.000000000', '2002-08-01T00:00:00.000000000',
       '2002-09-01T00:00:00.000000000', '2002-10-01T00:00:00.000000000',
       '2002-11-01T00:00:00.000000000', '2002-12-01T00:00:00.000000000'],
      dtype='datetime64[ns]')
```

In [7]:

```
airline.describe()
```

Out[7]:

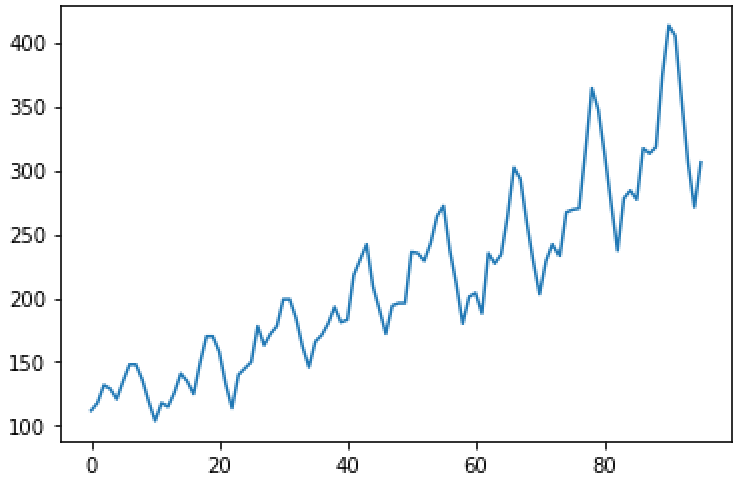
	Passengers
count	96.000000
mean	213.708333
std	71.918216
min	104.000000
25%	156.000000
50%	200.000000
75%	264.750000
max	413.000000

In [8]:

```
airline.Passengers.plot()
```

Out[8]:

<AxesSubplot:>



In [9]:

```
airline['Date']= pd.to_datetime(airline.Month,format='%b-%y')
airline['Months']= airline.Date.dt.strftime('%b')
airline['Year'] = airline.Date.dt.strftime('%Y')
```

In [10]:

```
airline
```

Out[10]:

	Month	Passengers	Date	Months	Year
0	1995-01-01	112	1995-01-01	Jan	1995
1	1995-02-01	118	1995-02-01	Feb	1995
2	1995-03-01	132	1995-03-01	Mar	1995
3	1995-04-01	129	1995-04-01	Apr	1995
4	1995-05-01	121	1995-05-01	May	1995
...	...	...	...	...	...
91	2002-08-01	405	2002-08-01	Aug	2002
92	2002-09-01	355	2002-09-01	Sep	2002
93	2002-10-01	306	2002-10-01	Oct	2002
94	2002-11-01	271	2002-11-01	Nov	2002
95	2002-12-01	306	2002-12-01	Dec	2002

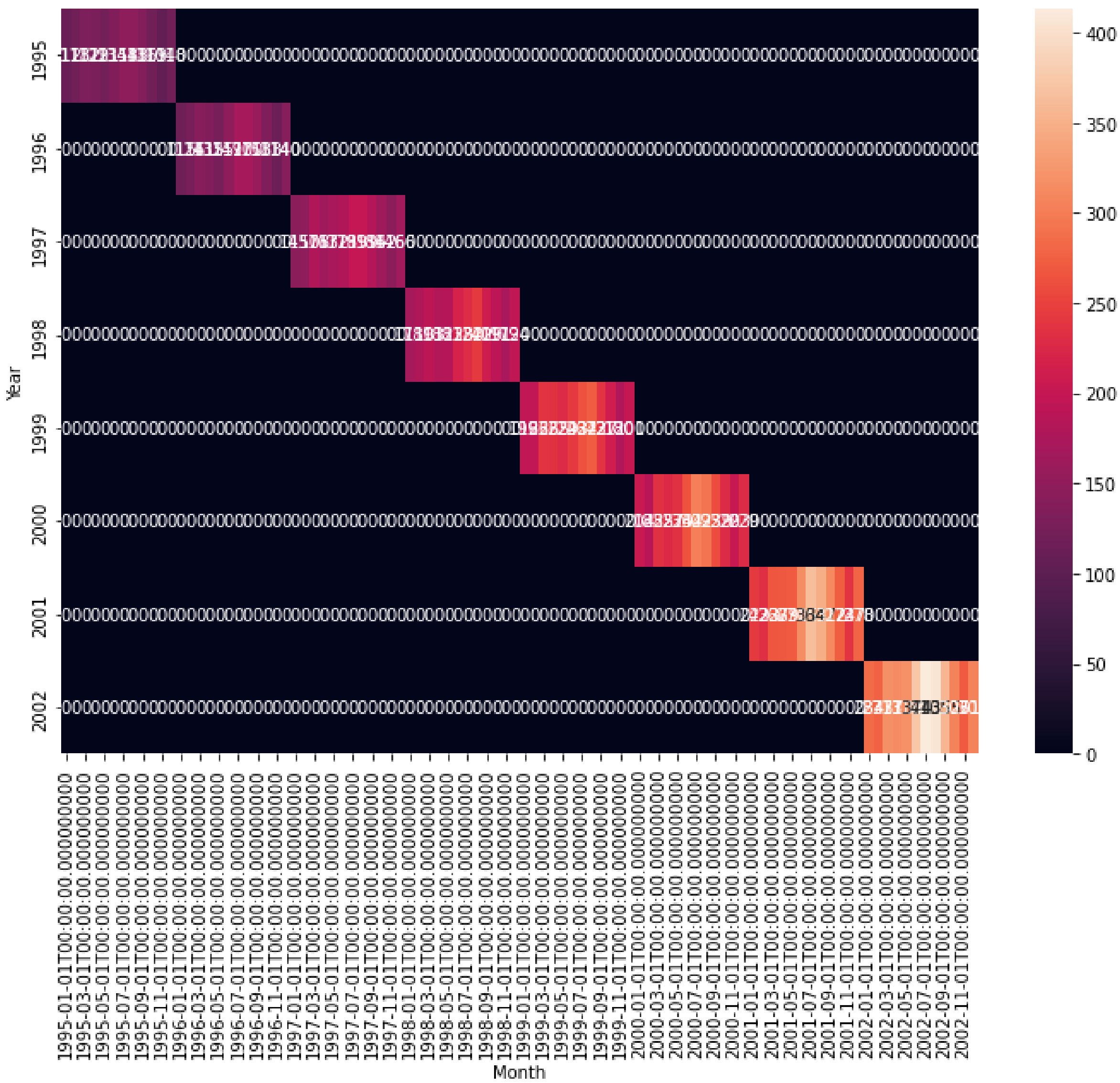
96 rows × 5 columns

In [11]:

```
# Heatmap
plt.figure(figsize=(12,8))
heatmap_y_month = pd.pivot_table(data=airline,values='Passengers',index='Year',columns='Mon
sns.heatmap(heatmap_y_month,annot=True,fmt='g') # fmt is format of the grid values
```

Out[11]:

```
<AxesSubplot:xlabel='Month', ylabel='Year'>
```

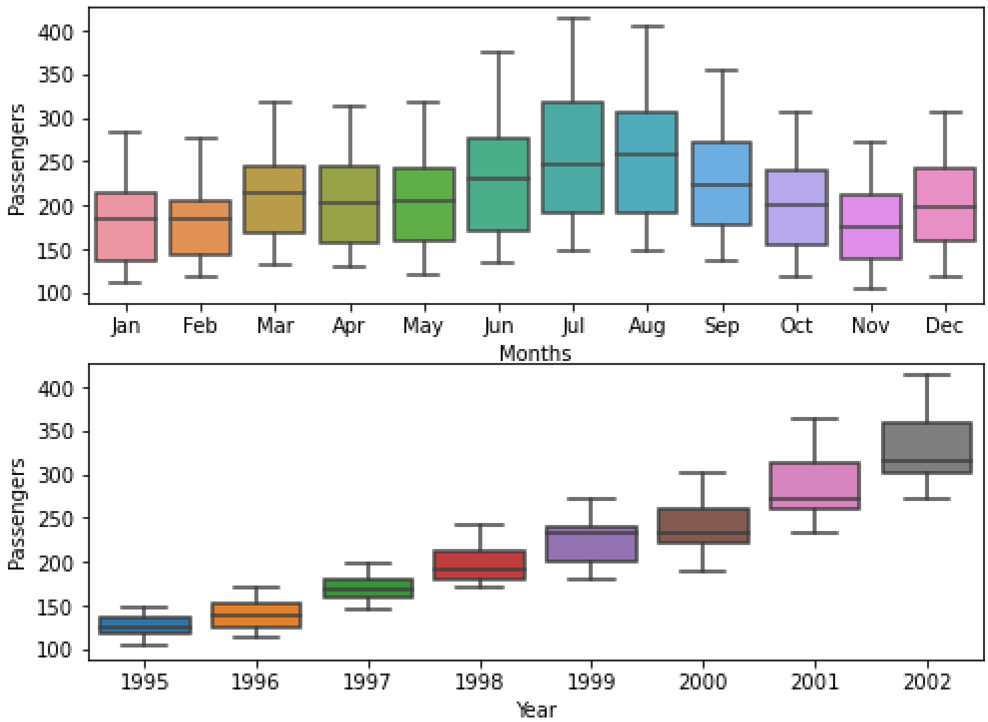


In [12]:

```
# Boxplot
plt.figure(figsize=(8,6))
plt.subplot(211)
sns.boxplot(x='Months',y='Passengers',data=airline)
plt.subplot(212)
sns.boxplot(x='Year',y='Passengers', data=airline)
```

Out[12]:

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



In [13]:

```
#Preparing dummies
Month_Dummies= pd.DataFrame(pd.get_dummies(airline['Months']))
airline1 = pd.concat([airline,Month_Dummies],axis =1)
```

In [14]:

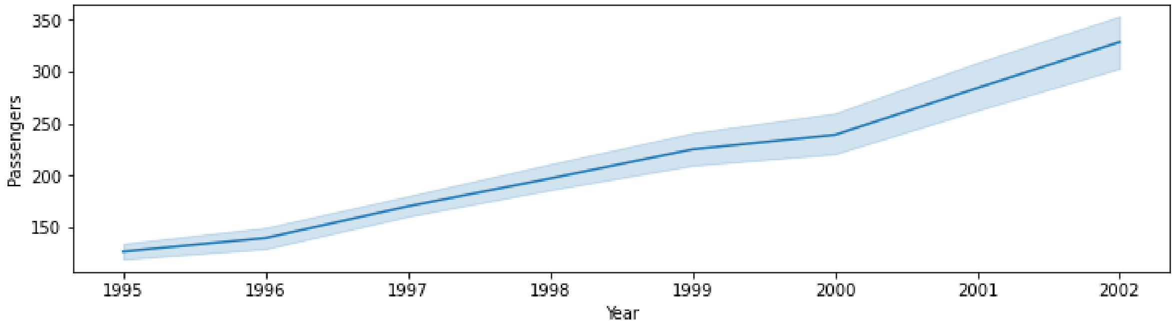
```
airline1["t"] = np.arange(1,97)
airline1["t_squared"] = airline1["t"] * airline1["t"]
airline1["Log_Passengers"] = np.log(airline1["Passengers"])
```

In [15]:

```
plt.figure(figsize=(12,3))
sns.lineplot(x='Year', y='Passengers', data=airline)
```

Out[15]:

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



In [16]:

```
#Splitting data
Train = airline1.head(80)
Test = airline1.tail(16)
```

In [17]:

```
#Linear Model
import statsmodels.formula.api as smf

linear_model = smf.ols('Passengers~t', data=Train).fit()
pred_linear = pd.Series(linear_model.predict(pd.DataFrame(Test['t'])))
rmse_linear = np.sqrt(np.mean((np.array(Test['Passengers'])- np.array(pred_linear))**2))
rmse_linear
```

Out[17]:

```
47.542624067726734
```

In [18]:

```
#Exponential Model
Exp = smf.ols('Log_Passengers~t', data = Train).fit()
pred_Exp = pd.Series(Exp.predict(pd.DataFrame(Test['t'])))
rmse_Exp = np.sqrt(np.mean((np.array(Test['Passengers'])- np.array(np.exp(pred_Exp)))**2))
rmse_Exp
```

Out[18]:

```
43.79373939334317
```

In [19]:

```
#Quadratic Model
Quad = smf.ols('Passengers~t+t_squared',data=Train).fit()
pred_Quad = pd.Series(Quad.predict(Test[['t','t_squared']]))
rmse_Quad = np.sqrt(np.mean((np.array(Test['Passengers'])-np.array(pred_Quad))**2))
rmse_Quad
```

Out[19]:

43.65440369584248

In [20]:

```
#Additive seasonality
add_sea = smf.ols('Passengers~Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+Nov', data=Train).fit
pred_add_sea = pd.Series(add_sea.predict(Test[['Jan','Feb','Mar','Apr','May','Jun','Jul','A
rmse_add_sea = np.sqrt(np.mean((np.array(Test['Passengers'])- np.array(pred_add_sea))**2))
rmse_add_sea
```

Out[20]:

129.26647641443313

In [21]:

```
#Additive Seasonality quadratic
add_sea_Quad = smf.ols('Passengers~t+t_squared+Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+Nov'
pred_add_sea_quad = pd.Series(add_sea_Quad.predict(Test[['Jan','Feb','Mar','Apr','May','Jun
rmse_add_sea_quad = np.sqrt(np.mean((np.array(Test['Passengers'])-np.array(pred_add_sea_qua
rmse_add_sea_quad
```

Out[21]:

23.91098357010659

In [22]:

```
#Multiplicative Seasonality
Mul_sea = smf.ols('Log_Passengers~Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+Nov', data=Train)
pred_Mult_sea = pd.Series(Mul_sea.predict(Test))
rmse_Mult_sea = np.sqrt(np.mean((np.array(Test['Passengers'])-np.array(np.exp(pred_Mult_sea
rmse_Mult_sea
```

Out[22]:

135.3264841462111

In [23]:

```
#Multiplicative Additive Seasonality
Mul_Add_sea = smf.ols('Log_Passengers~t+Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+Nov',data=T
pred_Mult_add_sea = pd.Series(Mul_Add_sea.predict(Test))
rmse_Mult_add_sea = np.sqrt(np.mean((np.array(Test['Passengers'])-np.array(np.exp(pred_Mult
rmse_Mult_add_sea
```

Out[23]:

9.469000230303608

In [24]:

```
#Tabulating the rmse values
data= {'Model':pd.Series(['rmse_linear','rmse_Exp','rmse_Quad','rmse_add_sea','rmse_add_sea
table_rmse = pd.DataFrame(data)
table_rmse.sort_values(['RMSE_Values'])
```

Out[24]:

	Model	RMSE_Values
6	rmse_Mult_add_sea	9.469000
4	rmse_add_sea_quad	23.910984
2	rmse_Quad	43.654404
1	rmse_Exp	43.793739
0	rmse_linear	47.542624
3	rmse_add_sea	129.266476
5	rmse_Mult_sea	135.326484

Conclusion:- From the above rmse values (rmse_Mult_ADD_sea - 9.469) is the best fit model

In [25]:

```
# Forecasting using Multiplicative Additive Seasonality Model
# Forecasting for next 12 months
data = [['2003-01-01','Jan'],['2003-02-01','Feb'],['2003-03-01','Mar'],['2003-04-01','Apr']
# Print(data)
forecast = pd.DataFrame(data,columns = ['Date','Months'])
forecast
```

Out[25]:

	Date	Months
0	2003-01-01	Jan
1	2003-02-01	Feb
2	2003-03-01	Mar
3	2003-04-01	Apr
4	2003-05-01	May
5	2003-06-01	Jun
6	2003-07-01	Jul
7	2003-08-01	Aug
8	2003-09-01	Sep
9	2003-10-01	Oct
10	2003-11-01	Nov
11	2003-12-01	Dec

In [26]:

```
# Create dummies and T and T-Squared columns

dummies = pd.DataFrame(pd.get_dummies(forecast['Months']))
forecast1 = pd.concat([forecast, dummies], axis =1)
print('After dummy\n',forecast1.head())

forecast1['t'] = np.arange(1,13)
forecast1['t_squared'] = forecast1['t'] * forecast1['t']
print('\nAfter T and T-Squared\n', forecast1.head())
```

```
After dummy
         Date Months  Apr  Aug  Dec  Feb  Jan  Jul  Jun  Mar  May  Nov  Oct  \
0 2003-01-01    Jan    0    0    0    0    1    0    0    0    0    0    0
1 2003-02-01    Feb    0    0    0    1    0    0    0    0    0    0    0
2 2003-03-01    Mar    0    0    0    0    0    0    0    1    0    0    0
3 2003-04-01    Apr    1    0    0    0    0    0    0    0    0    0    0
4 2003-05-01    May    0    0    0    0    0    0    0    0    1    0    0

   Sep
0    0
1    0
2    0
3    0
4    0

After T and T-Squared
         Date Months  Apr  Aug  Dec  Feb  Jan  Jul  Jun  Mar  May  Nov  Oct  \
0 2003-01-01    Jan    0    0    0    0    1    0    0    0    0    0    0
1 2003-02-01    Feb    0    0    0    1    0    0    0    0    0    0    0
2 2003-03-01    Mar    0    0    0    0    0    0    0    1    0    0    0
3 2003-04-01    Apr    1    0    0    0    0    0    0    0    0    0    0
4 2003-05-01    May    0    0    0    0    0    0    0    0    1    0    0

   Sep  t  t_squared
0    0  1          1
1    0  2          4
2    0  3          9
3    0  4         16
4    0  5         25
```

In [27]:

```
# Forecasting using Multiplicative Additive Seasonality Model

model_full = smf.ols('Log_Passengers~t+Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+Nov+Dec',dat
pred_new  = pd.Series(model_full.predict(forecast1))
pred_new

forecast1["Forecasted_log"] = pd.Series(pred_new)
forecast1['Forecasted_Passengers'] = np.exp(forecast1['Forecasted_log'])
```

In [28]:

```
# Final Prediction

Final_predict = forecast1.loc[:, ['Months','Forecasted_Passengers']]
Final_predict
```

Out[28]:

	Months	Forecasted_Passengers
0	Jan	109.176148
1	Feb	110.331245
2	Mar	127.315234
3	Apr	123.200587
4	May	122.399578
5	Jun	138.536397
6	Jul	154.066959
7	Aug	153.741209
8	Sep	137.693733
9	Oct	120.894736
10	Nov	106.109309
11	Dec	121.633998

In [ ]:

```

```