Forecasting - 02

In [1]:
```
# Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
passengers_data = pd.read_excel('Airlines+Data.xlsx')
passengers_data
```

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
...	...	...
91	2002-08-01	405
92	2002-09-01	355
93	2002-10-01	306
94	2002-11-01	271
95	2002-12-01	306

96 rows × 2 columns

In [3]:
```
passengers_data.shape
```

Out[3]:
```
(96, 2)
```

In [4]:
```
passengers_data.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
```

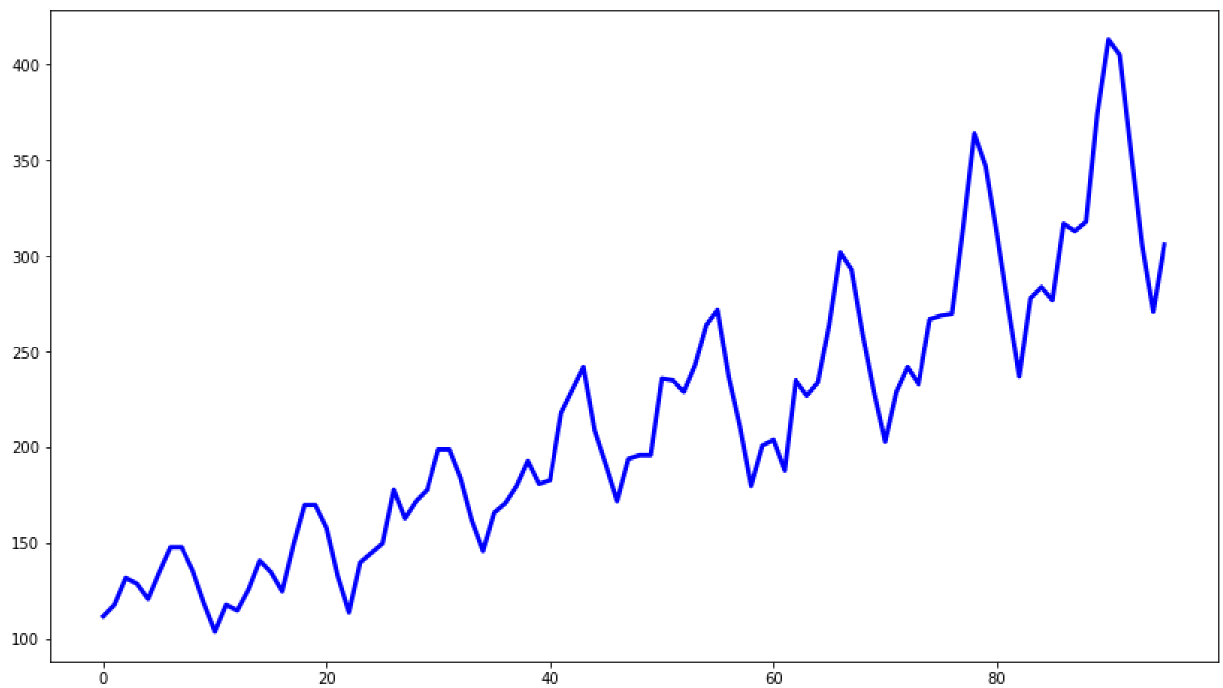
In [5]:
```
# Getting dummy variables
Months_Dummies = pd.DataFrame(pd.get_dummies(passengers_data['Month']))
passengers_data_dm = pd.concat([passengers_data,Months_Dummies],axis = 1)
passengers_data_dm.head()
```

Out[5]:

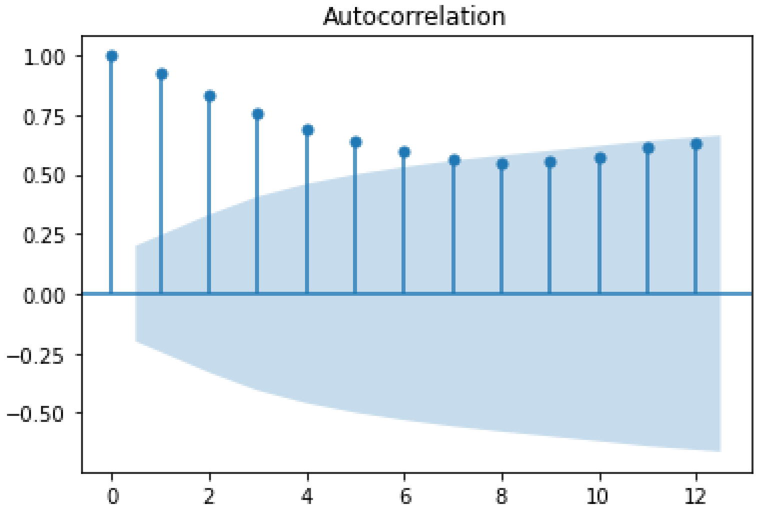
	Month	Passengers	1995-01-01 00:00:00	1995-02-01 00:00:00	1995-03-01 00:00:00	1995-04-01 00:00:00	1995-05-01 00:00:00	1995-06-01 00:00:00	1995-07-01 00:00:00	1995-0(00:00:(
0	1995-01-01	112	1	0	0	0	0	0	0	
1	1995-02-01	118	0	1	0	0	0	0	0	
2	1995-03-01	132	0	0	1	0	0	0	0	
3	1995-04-01	129	0	0	0	1	0	0	0	
4	1995-05-01	121	0	0	0	0	1	0	0	

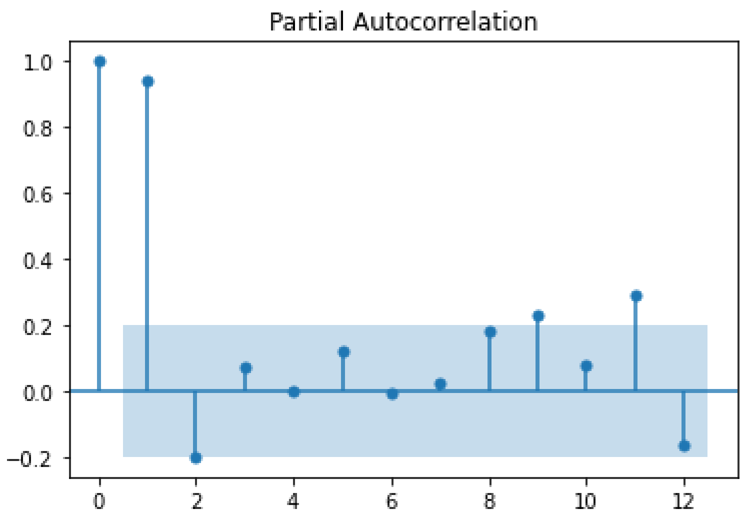
5 rows × 98 columns

In [6]:
```
# Lineplot for passengers
plt.figure(figsize=(14,8))
plt.plot(passengers_data['Passengers'],color = 'blue' , linewidth = 3)
plt.show()
```



In [7]:
```
import statsmodels.graphics.tsaplots as tsa_plots
tsa_plots.plot_acf(passengers_data.Passengers,lags=12)
tsa_plots.plot_pacf(passengers_data.Passengers,lags =12)
plt.show()
```





Data Driven Forcasting Methods

In [11]:
```
from statsmodels.tsa.holtwinters import SimpleExpSmoothing #SES
from statsmodels.tsa.holtwinters import Holt # Holts Exponential Smoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

In [12]:
```
# Spliting data into train and test
Train = passengers_data.head(84)
Test  =passengers_data.tail(12)
```

In [14]:
```
Train.head()
```

Out[14]:

	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

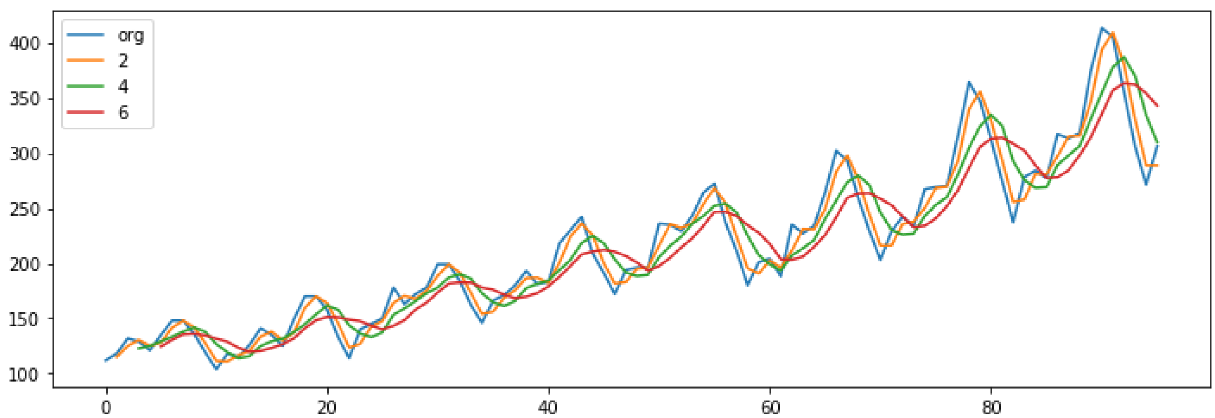
In [15]:
```
Test.head()
```

Out[15]:

	Month	Passengers
84	2002-01-01	284
85	2002-02-01	277
86	2002-03-01	317
87	2002-04-01	313
88	2002-05-01	318

Moving Average Method

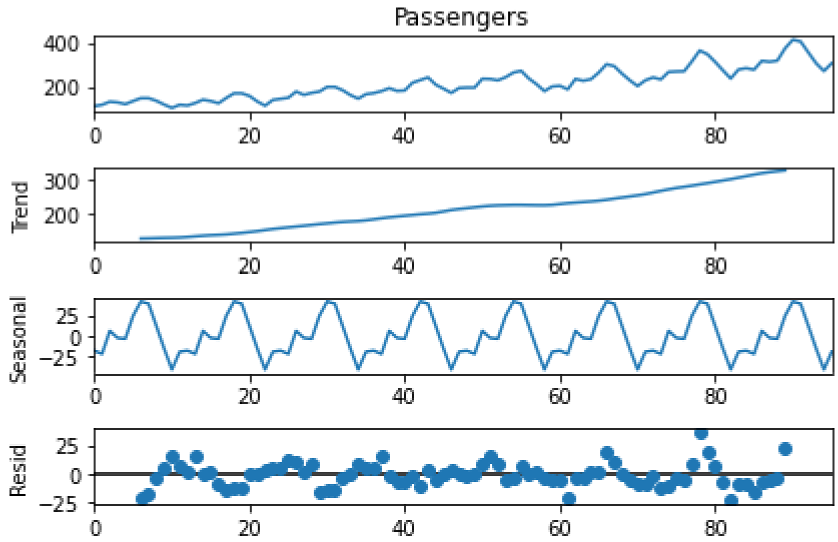
In [16]:
```
plt.figure(figsize=(12,4))
passengers_data.Passengers.plot(label="org")
for i in range(2,8,2):
    passengers_data["Passengers"].rolling(i).mean().plot(label=str(i))
plt.legend(loc='best')
plt.show()
```



Time Series Decomposition Plot

In [17]:
```
from statsmodels.tsa.seasonal import seasonal_decompose

decompose_ts_add = seasonal_decompose(passengers_data.Passengers,period=12)
decompose_ts_add.plot()
plt.show()
```



Evaluation Metric RMSE

In [18]:
```
def RMSE(org, pred):
    rmse=np.sqrt(np.mean((np.array(org)-np.array(pred))**2))
    return rmse
```

Simple Exponential Method

In [19]:
```
ses_model = SimpleExpSmoothing(Train["Passengers"]).fit()
pred_ses = ses_model.predict(start = Test.index[0],end = Test.index[-1])
rmse_ses = RMSE(Test.Passengers, pred_ses)
rmse_ses
```

Out[19]: 68.00674031349644

Holt Method

In [20]:
```
hw_model = Holt(Train["Passengers"]).fit()
pred_hw = hw_model.predict(start = Test.index[0],end = Test.index[-1])
rmse_hw = RMSE(Test.Passengers, pred_hw)
rmse_hw
```

Out[20]: 58.57384693071804

Holts winter exponential smoothing with additive seasonality and additive trend

In [21]:
```
hwe_model_add_add = ExponentialSmoothing(Train["Passengers"],seasonal="add",trend
pred_hwe_add_add = hwe_model_add_add.predict(start = Test.index[0],end = Test.inc
rmse_hwe_add_add = RMSE(Test.Passengers, pred_hwe_add_add)
rmse_hwe_add_add
```

Out[21]: 62.71406428068746

Holts winter exponential smoothing with additive seasonality and additive trend

In [22]:
```
hwe_model_mul_add = ExponentialSmoothing(Train["Passengers"],seasonal="mul",trend
pred_hwe_mul_add = hwe_model_mul_add.predict(start = Test.index[0],end = Test.inc
rmse_hwe_mul_add = RMSE(Test.Passengers, pred_hwe_mul_add)
rmse_hwe_mul_add
```

Out[22]: 64.77748540879074

Model based Forecasting Methods

In [23]:
```
# Data preprocessing for models
passengers_data["t"] = np.arange(1,97)
passengers_data["t_squared"] = passengers_data["t"]*passengers_data["t"]

passengers_data["log_psngr"] = np.log(passengers_data["Passengers"])

passengers_data.head()
```

Out[23]:

	Month	Passengers	t	t_squared	log_psngr
0	1995-01-01	112	1	1	4.718499
1	1995-02-01	118	2	4	4.770685
2	1995-03-01	132	3	9	4.882802
3	1995-04-01	129	4	16	4.859812
4	1995-05-01	121	5	25	4.795791

In [24]:
```
# Splitting data into Train and Test (77/33)
Train = passengers_data.head(84)
Test = passengers_data.tail(12)
```

In [25]:
```
Train.head()
```

Out[25]:

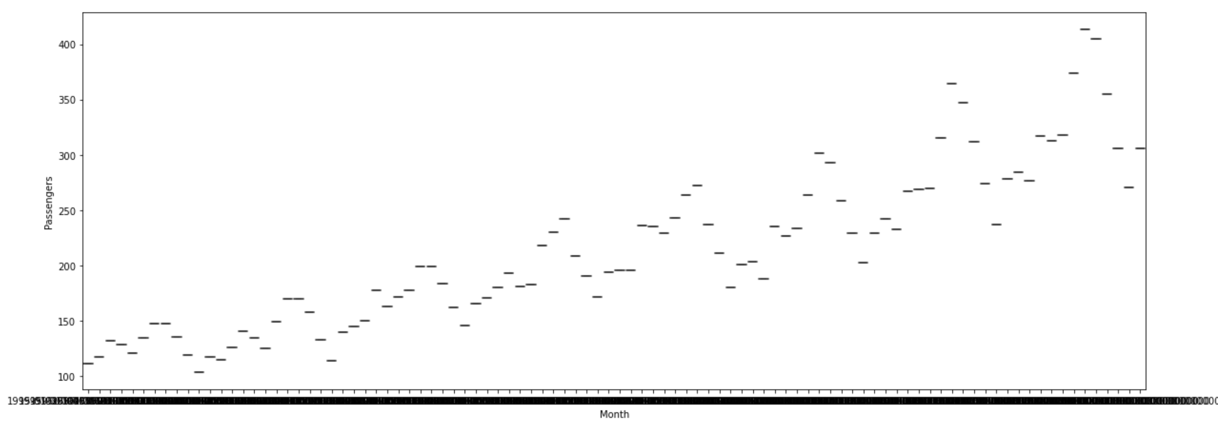
	Month	Passengers	t	t_squared	log_psngr
0	1995-01-01	112	1	1	4.718499
1	1995-02-01	118	2	4	4.770685
2	1995-03-01	132	3	9	4.882802
3	1995-04-01	129	4	16	4.859812
4	1995-05-01	121	5	25	4.795791

In [26]:
```
Test.head()
```

Out[26]:

	Month	Passengers	t	t_squared	log_psngr
84	2002-01-01	284	85	7225	5.648974
85	2002-02-01	277	86	7396	5.624018
86	2002-03-01	317	87	7569	5.758902
87	2002-04-01	313	88	7744	5.746203
88	2002-05-01	318	89	7921	5.762051

In [15]:
```
plt.figure(figsize=(20,16))
plt.subplot(2,1,1)
sns.boxplot(x="Month",y="Passengers",data=passengers_data)
None
```



Linear Model

In [27]:
```
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 = RMSE(Test['Passengers'], pred_linear)
rmse_linear
```

Out[27]: 53.199236534802715

Exponential Model

In [28]:
```
Exp = smf.ols('log_psngr~t',data=Train).fit()
pred_Exp = pd.Series(Exp.predict(pd.DataFrame(Test['t'])))
rmse_Exp = RMSE(Test['Passengers'], np.exp(pred_Exp))
rmse_Exp
```

Out[28]:
```
46.0573611031562
```

Quadratic Model

In [29]:
```
Quad = smf.ols('Passengers~t+t_squared',data=Train).fit()
pred_Quad = pd.Series(Quad.predict(Test[["t","t_squared"]]))
rmse_quad_model = RMSE(Test['Passengers'], pred_Quad)
rmse_quad_model
```

Out[29]:
```
48.051888979330975
```

In [30]:
```
list = [['Simple Exponential Method',rmse_ses], ['Holt method',rmse_hw],
          ['HW exp smoothing add',rmse_hwe_add_add],['HW exp smoothing mult',rmse
          ['Linear Mode',rmse_linear],['Exp model',rmse_Exp],['Quad model',rmse_q
```

In [31]:
```
dframe = pd.DataFrame(list, columns =['Model', 'RMSE_Value'])
dframe
```

Out[31]:

	Model	RMSE_Value
0	Simple Exponential Method	68.006740
1	Holt method	58.573847
2	HW exp smoothing add	62.714064
3	HW exp smoothing mult	64.777485
4	Linear Mode	53.199237
5	Exp model	46.057361
6	Quad model	48.051889

Building final model with least RMSE value

In [32]:
```
passengers_data.head()
```

Out[32]:

	Month	Passengers	t	t_squared	log_psngr
0	1995-01-01	112	1	1	4.718499
1	1995-02-01	118	2	4	4.770685
2	1995-03-01	132	3	9	4.882802
3	1995-04-01	129	4	16	4.859812
4	1995-05-01	121	5	25	4.795791

In [34]:
```
final_model = smf.ols('Passengers~t+t_squared',data=passengers_data).fit()
pred_final = pd.Series(final_model.predict(passengers_data[['t','t_squared']]))
rmse_final_model = RMSE(passengers_data['Passengers'], pred_final)
rmse_final_model
```

Out[34]:
```
29.59097162530025
```

In [36]:
```
pred_df = pd.DataFrame({'Actual' : passengers_data.Passengers, 'Predicted' : pred
pred_df
```

Out[36]:

	Actual	Predicted
0	112	119.158137
1	118	120.460303
2	132	121.784439
3	129	123.130544
4	121	124.498617
...	...	...
91	405	327.618598
92	355	330.919950
93	306	334.243270
94	271	337.588559
95	306	340.955817

96 rows × 2 columns