Import neccessery libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
from statsmodels.tsa.holtwinters import Holt
from statsmodels.tsa.holtwinters import ExponentialSmoothing
import statsmodels.graphics.tsaplots as tsa_plots
import statsmodels.tsa.statespace as tm_models
from datetime import datetime,time
import warnings
import itertools
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
import pandas as pd
import statsmodels.api as sm
import matplotlib
from pylab import rcParams
from statsmodels.tsa.arima_model import ARIMA
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
import statsmodels.formula.api as smf
```

Problem

Forecast the airlines 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

Import data

In [2]:

```
airline = pd.read_excel('Airlines+Data.xlsx')
airline
```

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

	Month	Passengers
92	2002-09-01	355
93	2002-10-01	306
94	2002-11-01	271
95	2002-12-01	306

In [3]:
```
airline_data=airline.copy()
```

In [14]:
```
airline_data.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

Data understanding

In [15]:
```
airline_data.shape
```

Out[15]:
```
(96, 2)
```

In [6]:
```
airline_data.isnull().sum()
```

Out[6]:
```
Month         0
Passengers    0
dtype: int64
```

In [7]:
```
airline_data.dtypes
```

Out[7]:
```
Month         datetime64[ns]
Passengers             int64
dtype: object
```

In [8]:
```
airline_data.describe().T
```

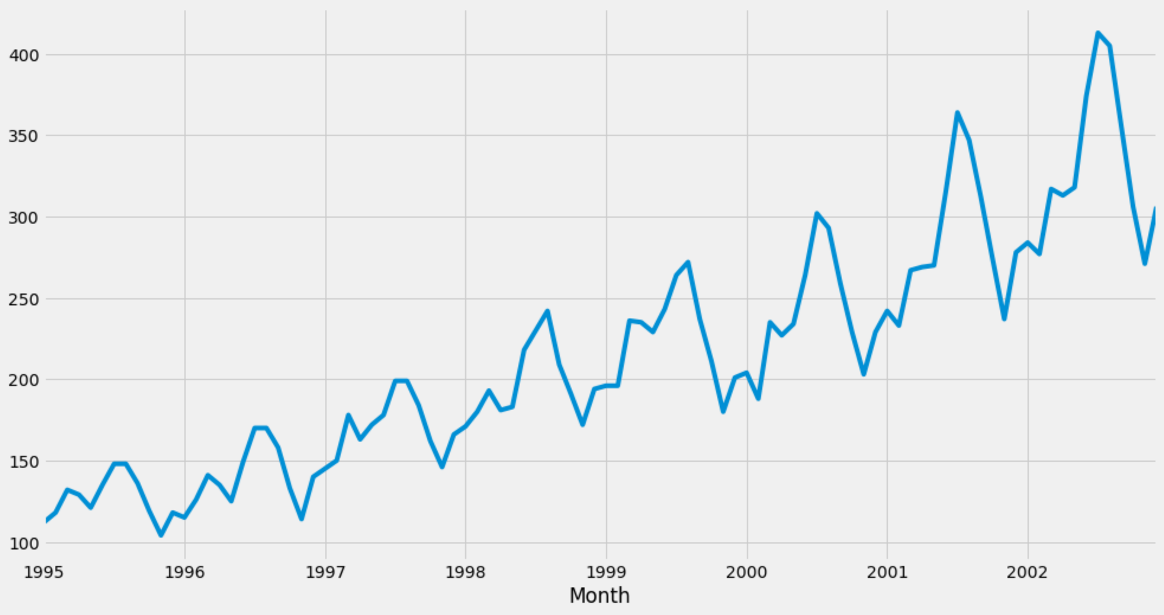
Out[8]:

	count	mean	std	min	25%	50%	75%	max
Passengers	96.0	213.708333	71.918216	104.0	156.0	200.0	264.75	413.0

In [9]:
```
airline_data_2 = airline_data.set_index('Month')
```

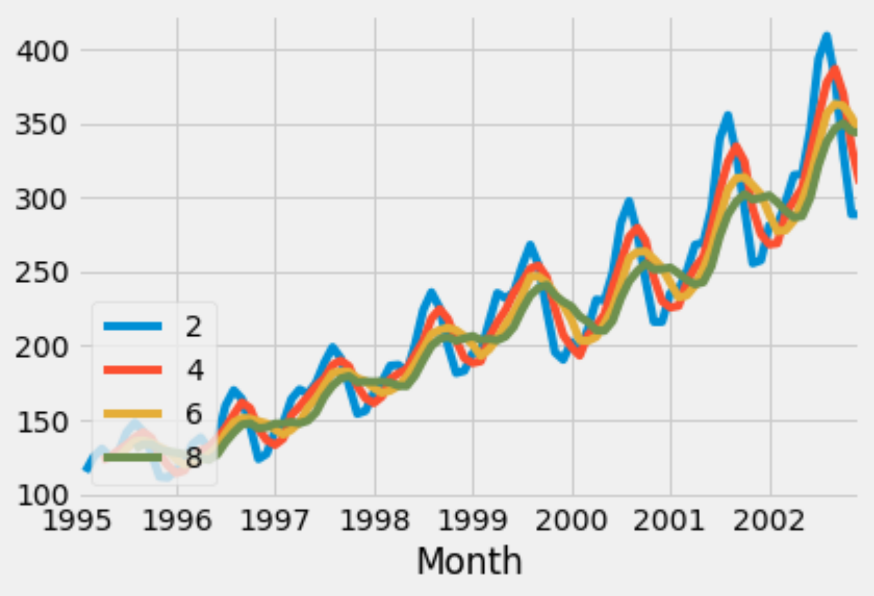
In [72]:

```
airline_data_2['Passengers'].plot(figsize=(15,8))
plt.show()
```



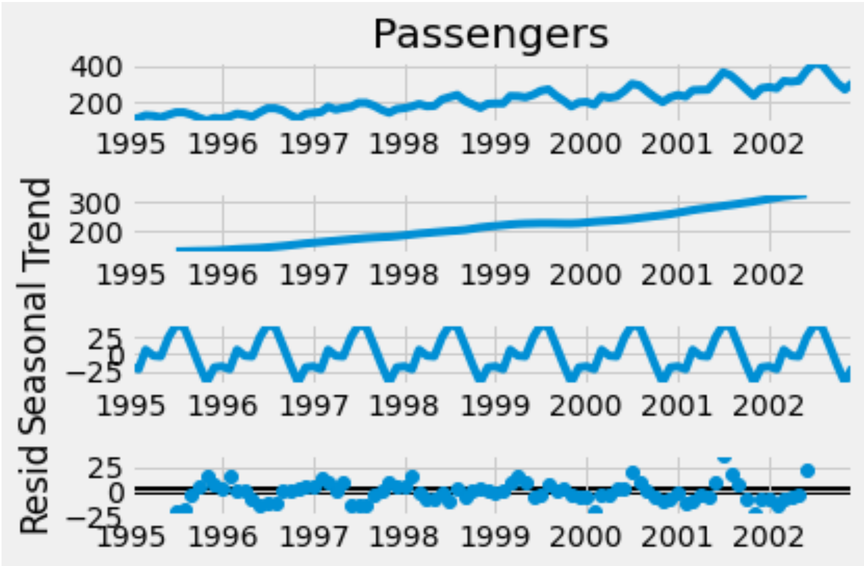
In [73]:

```
for i in range(2,10,2):
    airline_data_2['Passengers'].rolling(i).mean().plot(label=str(i))
    plt.legend(loc =3)
```



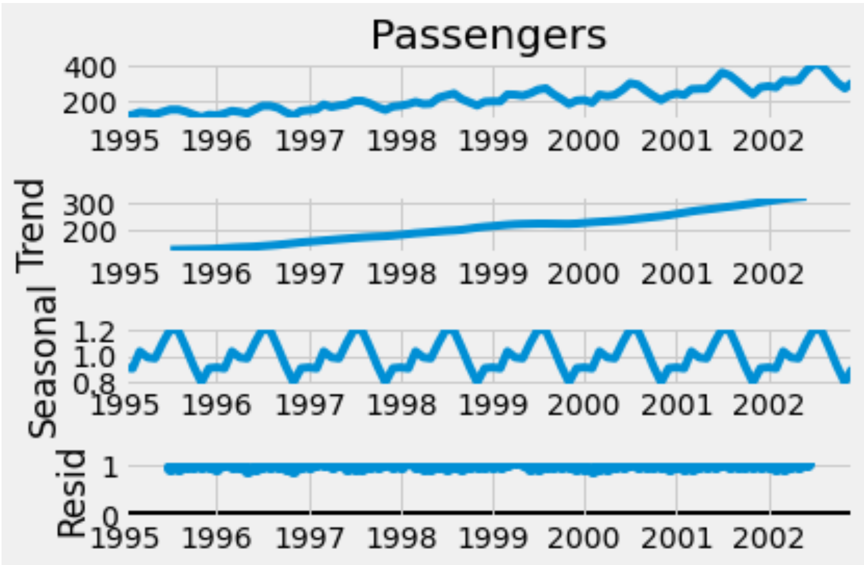
In [74]:

```
ts_add = seasonal_decompose(airline_data_2['Passengers'],model="additive")
fig = ts_add.plot()
plt.show()
```



In [75]:

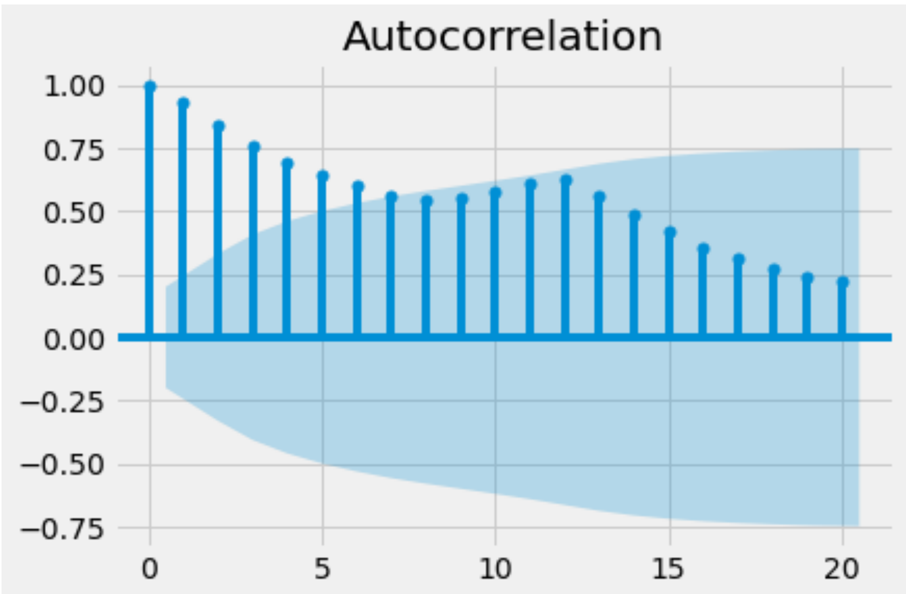
```
ts_mul = seasonal_decompose(airline_data_2.Passengers,model="multiplicative")
fig = ts_mul.plot()
plt.show()
```

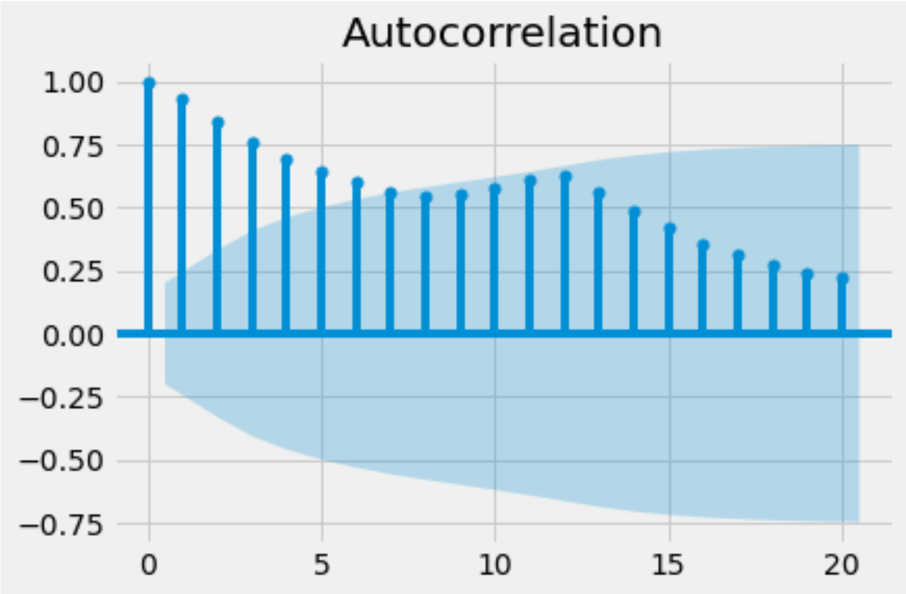


In [76]:

```
tsa_plots.plot_acf(airline_data_2['Passengers'])
```

Out[76]:





Building Time series forecasting with ARIMA

In [77]:
```
X = airline_data_2['Passengers'].values
```

In [78]:
```
size = int(len(X) * 0.66)
```

In [79]:
```
train, test = X[0:size], X[size:len(X)]
```

In [80]:
```
model = ARIMA(train, order=(5,1,0))
```

In [81]:
```
model_fit = model.fit(disp=0)
```

In [82]:
```
print(model_fit.summary())
```

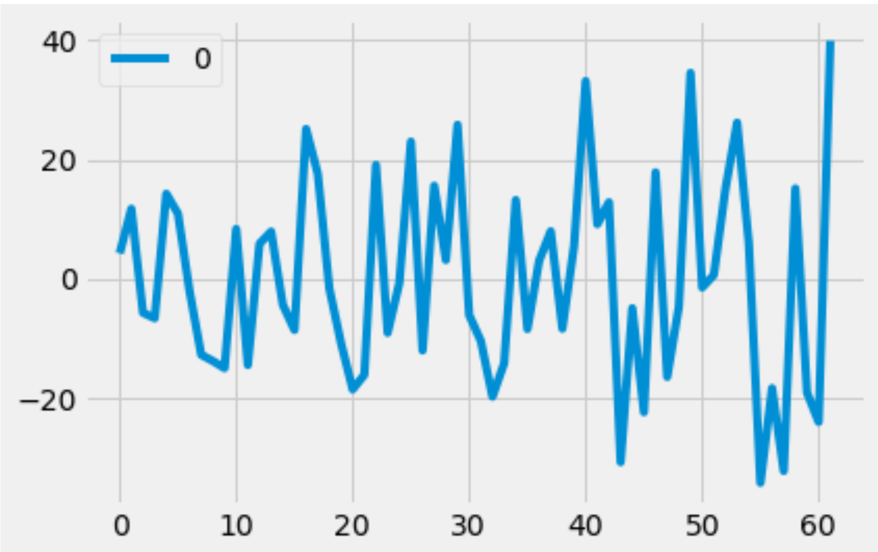
```
                              ARIMA Model Results
==============================================================================
=
Dep. Variable:                    D.y   No. Observations:                    6
2
Model:                 ARIMA(5, 1, 0)   Log Likelihood                 -262.90
9
Method:                       css-mle   S.D. of innovations              16.74
8
Date:                Sat, 26 Mar 2022   AIC                             539.81
7
Time:                        11:01:36   BIC                             554.70
7
Sample:                             1   HQIC                            545.66
3

==============================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
```

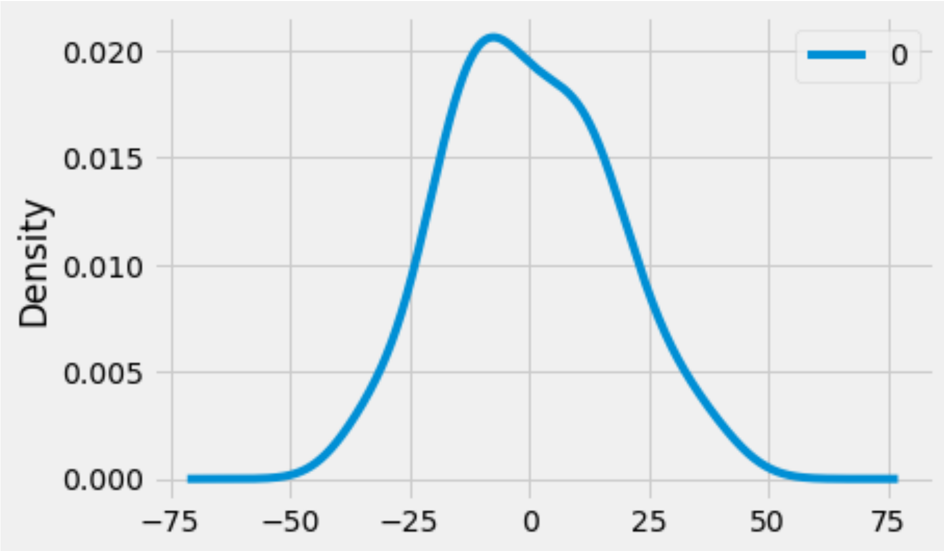
```
-------------------------------------------------------------------------------
-
const          1.7497      1.477      1.185      0.236      -1.145      4.64
4
ar.L1.D.y      0.0905      0.134      0.677      0.498      -0.171      0.35
2
ar.L2.D.y     -0.2096      0.135     -1.549      0.121      -0.475      0.05
6
ar.L3.D.y     -0.0829      0.133     -0.623      0.533      -0.344      0.17
8
ar.L4.D.y     -0.2598      0.133     -1.953      0.051      -0.521      0.00
1
ar.L5.D.y     -0.0090      0.139     -0.065      0.948      -0.282      0.26
4
                                    Roots
==============================================================================
                  Real          Imaginary           Modulus         Frequency
------------------------------------------------------------------------------
AR.1            0.8182          -1.0121j            1.3015           -0.1418
AR.2            0.8182          +1.0121j            1.3015            0.1418
AR.3           -0.9648          -1.1683j            1.5152           -0.3599
AR.4           -0.9648          +1.1683j            1.5152            0.3599
AR.5          -28.5048          -0.0000j           28.5048           -0.5000
------------------------------------------------------------------------------
```

This summarizes the coefficient values used as well as the skill of the fit on the on the in-sample observations

In [83]:

```
residuals = pd.DataFrame(model_fit.resid)
residuals.plot()
pyplot.show()
residuals.plot(kind='kde')
pyplot.show()
print(residuals.describe())
```






```
                0
count   62.000000
mean     0.057356
```

The plot of the residual errors suggests that there may still be some trend information not captured by the model

The results show that there is no a bias in the prediction (a zero mean in the residuals)

Rolling Forecast ARIMA Model

In [84]:

```
history = [x for x in train]
```

In [85]:

```
predictions = list()
```

In [86]:

```
for t in range(len(test)):
        model = ARIMA(history, order=(5,1,0))
        model_fit = model.fit(disp=0)
        output = model_fit.forecast()
        yhat = output[0]
        predictions.append(yhat)
        obs = test[t]
        history.append(obs)
        print('predicted=%f, expected=%f' % (yhat, obs))
```

```
predicted=239.755179, expected=227.000000
predicted=220.737300, expected=234.000000
predicted=237.815008, expected=264.000000
predicted=252.750592, expected=302.000000
predicted=306.715794, expected=293.000000
predicted=285.374643, expected=259.000000
predicted=250.264004, expected=229.000000
predicted=227.093124, expected=203.000000
predicted=211.011455, expected=229.000000
predicted=253.260281, expected=242.000000
predicted=252.490682, expected=233.000000
predicted=234.042132, expected=267.000000
predicted=268.773632, expected=269.000000
```

```
predicted=261.782257, expected=270.000000
predicted=271.798054, expected=315.000000
predicted=314.422115, expected=364.000000
predicted=368.637742, expected=347.000000
predicted=334.957878, expected=312.000000
predicted=301.161818, expected=274.000000
predicted=265.936481, expected=237.000000
predicted=244.037181, expected=278.000000
predicted=312.961792, expected=284.000000
predicted=291.748167, expected=277.000000
predicted=284.551868, expected=317.000000
predicted=316.501195, expected=313.000000
predicted=303.218148, expected=318.000000
predicted=324.834619, expected=374.000000
predicted=373.140656, expected=413.000000
predicted=415.007200, expected=405.000000
predicted=397.508453, expected=355.000000
predicted=332.087112, expected=306.000000
predicted=299.452956, expected=271.000000
predicted=279.908341, expected=306.000000
```

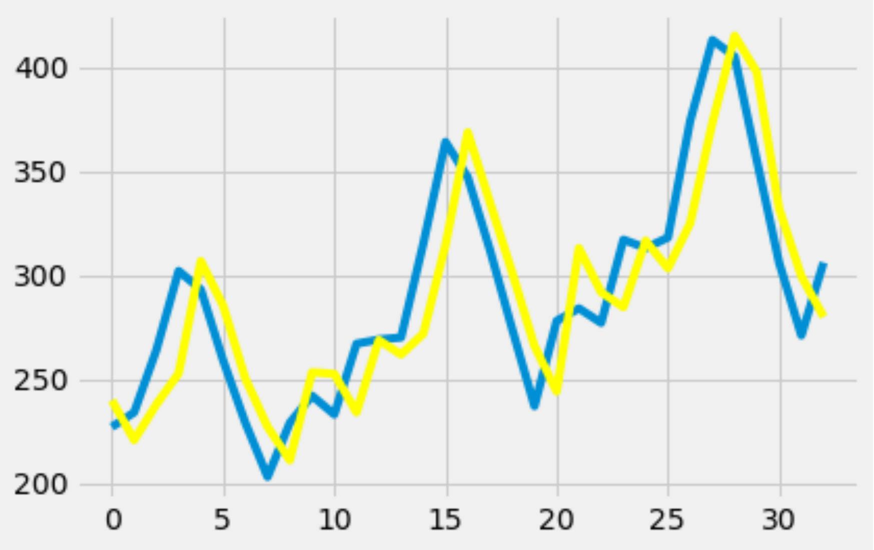
In [87]:

```
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)
```

```
Test MSE: 782.495
```

In [88]:

```
pyplot.plot(test)
pyplot.plot(predictions, color='yellow')
pyplot.show()
```



A line plot is created showing the expected values (blue) compared to the rolling forecast predictions (yellow). We can see the values show some trend and are in the correct scale

Comparing Multiple Models

In [51]:

```
airline_data_3 = airline.copy()
```

In [52]:
```
airline_data_3 = pd.get_dummies(airline_data_3, columns = ['Month'])
```

In [ ]:
```
airline_data_3.columns['Passengers','month_1','month_2','month_3','month_4','
```

In [69]:
```
airline_data_3.head()
```

Out[69]:

	Passengers	Month_1995-01-01 00:00:00	Month_1995-02-01 00:00:00	Month_1995-03-01 00:00:00	Month_1995-04-0 00:00:0
0	112	1	0	0	
1	118	0	1	0	
2	132	0	0	1	
3	129	0	0	0	
4	121	0	0	0	

5 rows × 97 columns

In [20]:
```
airline_data_3.shape
```

Out[20]:
```
(96, 97)
```

In [21]:
```
t= np.arange(1,97)
```

In [22]:
```
airline_data_3['t'] = t
```

In [23]:
```
airline_data_3['t_sq'] = airline_data_3['t']*airline_data_3['t']
```

In [33]:
```
passengers1= np.log(airline_data_3['Passengers'])
```

In [34]:
```
airline_data_3['Passengers1']=passengers1
```

In [35]:
```
airline_data_3.head()
```

Out[35]:

	Passengers	Month_1995-01-01 00:00:00	Month_1995-02-01 00:00:00	Month_1995-03-01 00:00:00	Month_1995-04-0 00:00:0
0	4.718499	1	0	0	
1	4.770685	0	1	0	
2	4.882802	0	0	1	
3	4.859812	0	0	0	

Passengers	Month_1995-01-01 00:00:00	Month_1995-02-01 00:00:00	Month_1995-03-01 00:00:00	Month_1995-04-0 00:00:0

In [36]:
```
train1, test1 = np.split(airline_data_3, [int(.67 *len(airline_data_3))])
```

In [37]:
```
linear= smf.ols('Passengers ~ t',data=train1).fit()
predlin=pd.Series(linear.predict(pd.DataFrame(test1['t'])))
rmselin=np.sqrt((np.mean(np.array(test1['Passengers'])-np.array(predlin))**2)
rmselin
```

Out[37]:
```
0.011446730996560794
```

In [38]:
```
quad=smf.ols('Passengers~t+t_sq',data=train1).fit()
predquad=pd.Series(quad.predict(pd.DataFrame(test1[['t','t_sq']])))
rmsequad=np.sqrt(np.mean((np.array(test1['Passengers'])-np.array(predquad))**
rmsequad
```

Out[38]:
```
0.17833090054825948
```

In [39]:
```
expo=smf.ols('Passengers~t',data=train1).fit()
predexp=pd.Series(expo.predict(pd.DataFrame(test1['t'])))
rmseexpo=np.sqrt(np.mean((np.array(test1['Passengers'])-np.array(np.exp(prede
rmseexpo
```

Out[39]:
```
289.1236843586758
```

Conclusion

In [66]:
```
output = {'Model':pd.Series(['rmseexpo','rmselin','rmsequad']),
          'Values':pd.Series(['rmseexpo','rmselin','rmsequad'])}
```

In [67]:
```
rmse=pd.DataFrame(output)
```

In [68]:
```
print(rmse)
```

```
      Model    Values
0  rmseexpo  rmseexpo
1   rmselin   rmselin
2  rmsequad  rmsequad
```

In [ ]: