In [97]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [56]:
```
airlines = pd.read_csv('Airlines+Data.csv')
airlines
```

Out[56]:

	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 [79]:
```
import numpy as np
months=['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
n=airlines['Month'][0]
n[0:3]
```

Out[79]:
```
'Jan'
```

In [80]:
```
airlines['months']=0
airlines['months']
```

Out[80]:
```
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 [82]:
```
for i in range(96):
    n=airlines['Month'][i]
    airlines['months'][i]=n[0:3]
airlines['months']
```

```
<ipython-input-82-f28ad2f72dcd>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  airlines['months'][i]=n[0:3]
```

Out[82]:
```
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 [83]:
```
dummy = pd.DataFrame(pd.get_dummies(airlines['months']))
dummy
```

Out[83]:

	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 [84]:
```
print(dummy.columns)
```

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

In [85]:
```
air=pd.concat((airlines,dummy),axis=1)
t= np.arange(1,97)
air['t']=t
air['t_square']=air['t']*air['t']
air
```

Out[85]:

	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 [86]:
```
log_Passengers=np.log(air['Passengers'])
air['log_Passengers']=log_Passengers
air
```

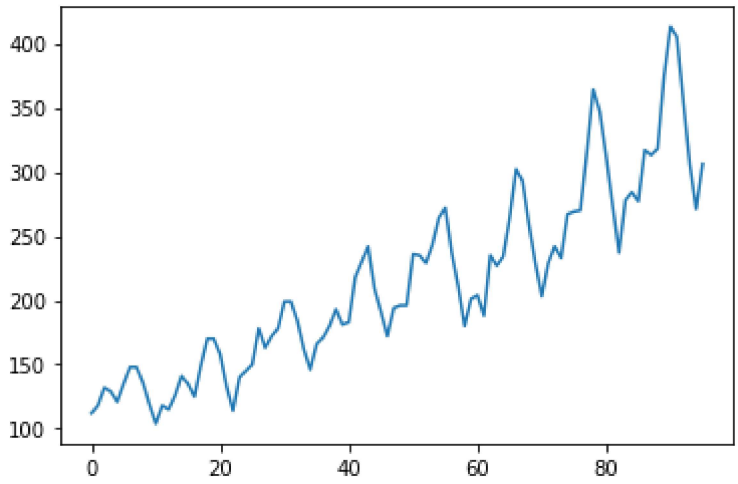
Out[86]:

	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 [98]:
```
train= air.head(92)
test=air.tail(4)
air.Passengers.plot()
```

Out[98]: <AxesSubplot:>



In [88]:
```
import statsmodels.formula.api as smf
```

In [89]:
```
#linear model
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[89]: 13.834024320700223

In [90]:
```
#quadratic model
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[90]: 51.289266550796974

In [91]:
```
#exponential model
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[91]: 47.52880701855483

In [93]:
```
additive seasonality
dditive= smf.ols('Passengers~Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+Nov+Dec', dat
redadd=pd.Series(additive.predict(pd.DataFrame(test[['Jan','Feb','Mar','Apr','May
redadd
mseadd = np.sqrt(np.mean((np.array(test['Passengers'])-np.array(predadd))**2))
mseadd
```

Out[93]: 121.05754239793797

In [94]:
```
asonality with linear trend
mf.ols('Passengers~t+Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+Nov+Dec', data=train)
=pd.Series(addlinear.predict(pd.DataFrame(test[['t','Jan','Feb','Mar','Apr','May'
 = np.sqrt(np.mean((np.array(test['Passengers'])-np.array(predaddlinear))**2))
```

Out[94]: 15.588971216767016

In [96]:
```
#additive seasonality with quadratic trend
addquad= smf.ols('Passengers~t+t_square+Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+N
predaddquad=pd.Series(addquad.predict(pd.DataFrame(test[['t','t_square','Jan','Fe
predaddlinear
rmseaddquad = np.sqrt(np.mean((np.array(test['Passengers'])-np.array(predaddquad)
rmseaddquad
```

Out[96]: 20.026363697070188

In [99]:
```
#multiplicative seasonality
mulsea=smf.ols('log_Passengers~Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+Nov+Dec',da
predmul= pd.Series(mulsea.predict(pd.DataFrame(test[['Jan','Feb','Mar','Apr','May
rmsemul= np.sqrt(np.mean((np.array(test['Passengers'])-np.array(np.exp(predmul)))*
rmsemul
```

Out[99]: 127.84380162494922

In [100]:
```
#multiplicative seasonality with linear trend
mullin= smf.ols('log_Passengers~t+Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+Nov+Dec
predmullin= pd.Series(mullin.predict(pd.DataFrame(test[['t','Jan','Feb','Mar','Ap
rmsemulin=np.sqrt(np.mean((np.array(test['Passengers'])-np.array(np.exp(predmulli
rmsemulin
```

Out[100]: 4.8541384948963895

In [102]:
```
#multiplicative seasonality with quadratic trend
mul_quad= smf.ols('log_Passengers~t+t_square+Jan+Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+
pred_mul_quad= pd.Series(mul_quad.predict(test[['t','t_square','Jan','Feb','Mar',
rmse_mul_quad=np.sqrt(np.mean((np.array(test['Passengers'])-np.array(np.exp(pred_
rmse_mul_quad
```

Out[102]:
```
3.660865095653222
```

In [103]:
```
#tabulating the rmse values

data={'Model':pd.Series(['rmse_mul_quad','rmseadd','rmseaddlinear','rmseaddquad',
data
```

Out[103]:
```
{'Model': 0    rmse_mul_quad
 1          rmseadd
 2    rmseaddlinear
 3      rmseaddquad
 4         rmseexpo
 5          rmselin
 6          rmsemul
 7        rmsemulin
 8         rmsequad
 dtype: object,
 'Values': 0      3.660865
 1    121.057542
 2     15.588971
 3     20.026364
 4     47.528807
 5     13.834024
 6    127.843802
 7      4.854138
 8     51.289267
 dtype: float64}
```

In [104]:
```
Rmse=pd.DataFrame(data)
Rmse
```

Out[104]:

	Model	Values
0	rmse_mul_quad	3.660865
1	rmseadd	121.057542
2	rmseaddlinear	15.588971
3	rmseaddquad	20.026364
4	rmseexpo	47.528807
5	rmselin	13.834024
6	rmsemul	127.843802
7	rmsemulin	4.854138
8	rmsequad	51.289267

Airlines Data-Driven

In [105]:
```
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing # SES
from statsmodels.tsa.holtwinters import Holt # Holts Exponential Smoothing
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
```

In [118]:
```
airlines_1 = pd.read_csv('Airlines+Data.csv')
airlines_1
```

Out[118]:

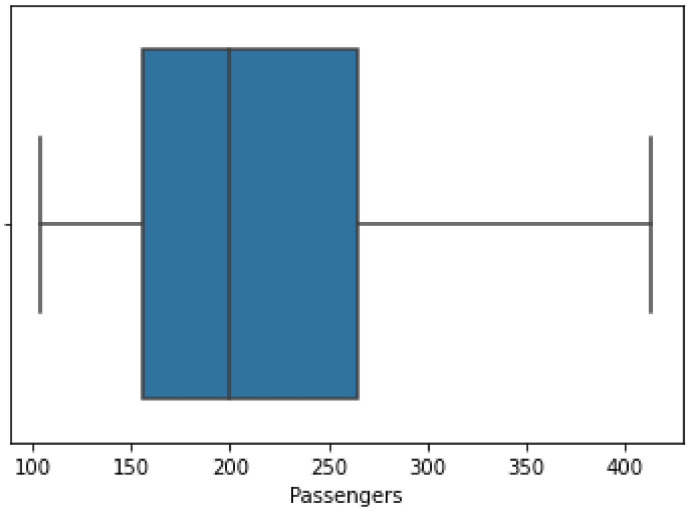
	Month	Passengers
0	Jan-95	112
1	Feb-95	118
2	Mar-95	132
3	Apr-95	129
4	May-95	121
...	...	...
91	Aug-02	405
92	Sep-02	355
93	Oct-02	306
94	Nov-02	271
95	Dec-02	306

96 rows × 2 columns

In [116]:
```
sns.boxplot('Passengers', data=airlines_1)
```

```
C:\Users\Hp\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

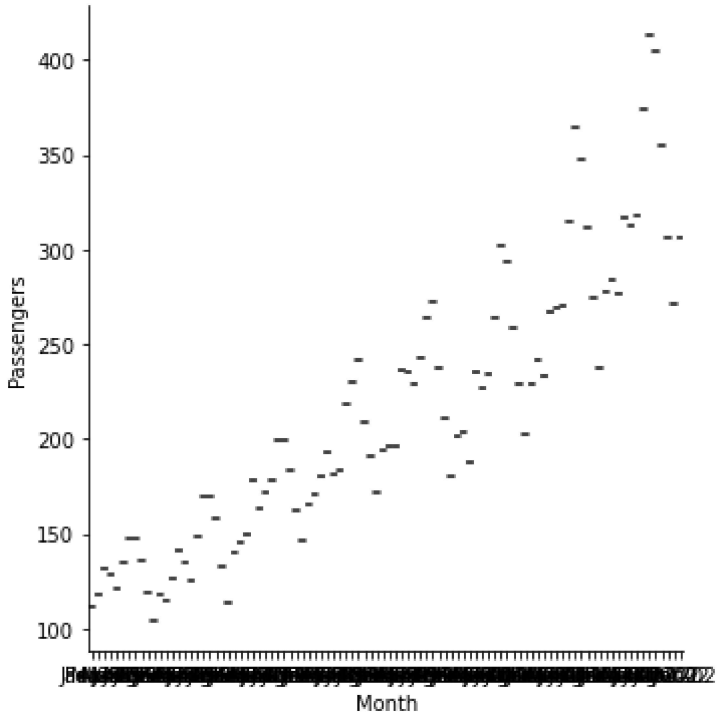
Out[116]:
```
<AxesSubplot:xlabel='Passengers'>
```



In [119]:
```
sns.factorplot("Month","Passengers",data=airlines_1,kind="box")
```

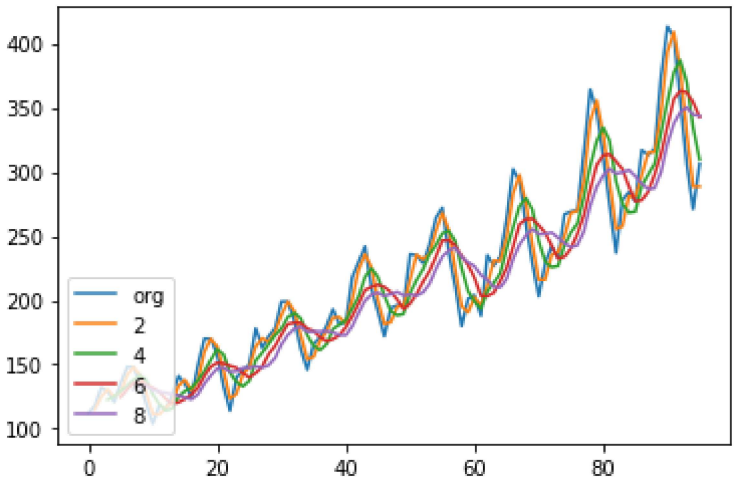
```
C:\Users\Hp\anaconda3\lib\site-packages\seaborn\categorical.py:3714: UserWarnin
g: The `factorplot` function has been renamed to `catplot`. The original name w
ill be removed in a future release. Please update your code. Note that the defa
ult `kind` in `factorplot` (`'point'`) has changed ``'strip'`` in `catplot`.
  warnings.warn(msg)
C:\Users\Hp\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[119]: <seaborn.axisgrid.FacetGrid at 0xb125928>



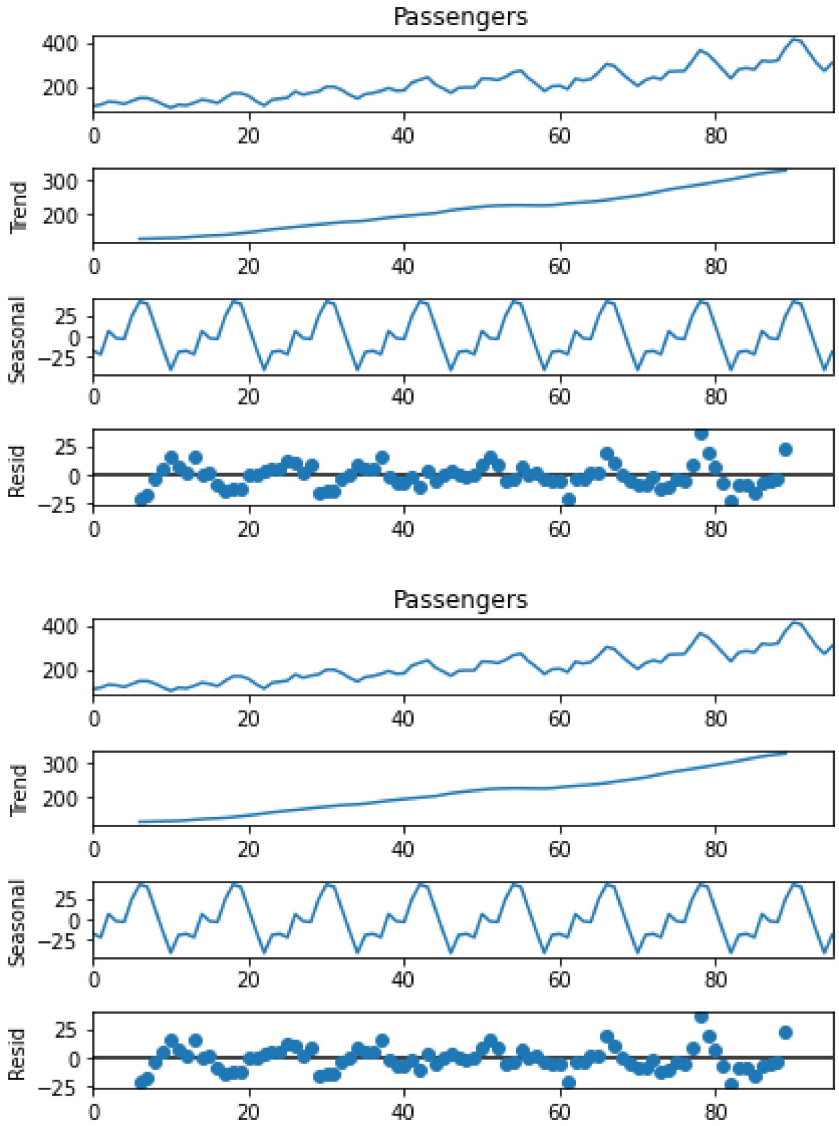
In [120]:
```
# moving average for the time series to understand better about the trend charact
airlines_1.Passengers.plot(label="org")
for i in range(2,10,2):
    airlines_1["Passengers"].rolling(i).mean().plot(label=str(i))
plt.legend(loc=3)
```

Out[120]: <matplotlib.legend.Legend at 0xb235778>



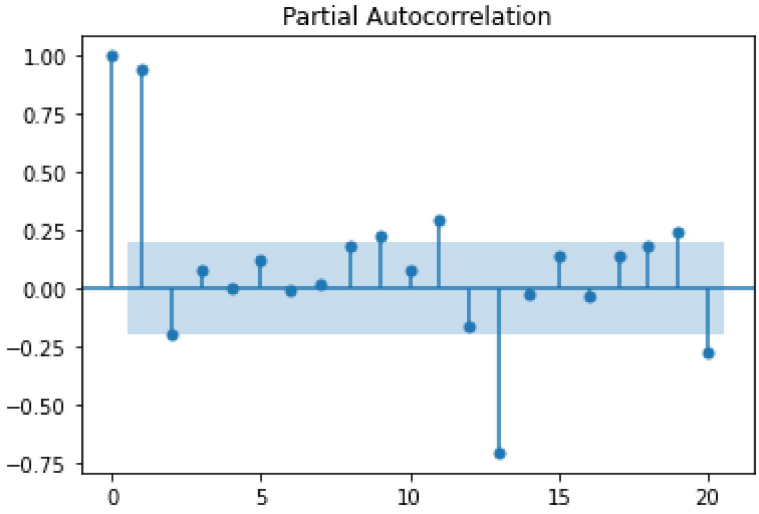
In [121]:
```
# Time series decomposition plot
decompose_ts_add = seasonal_decompose(airlines_1.Passengers,period =12)
decompose_ts_add.plot()
```

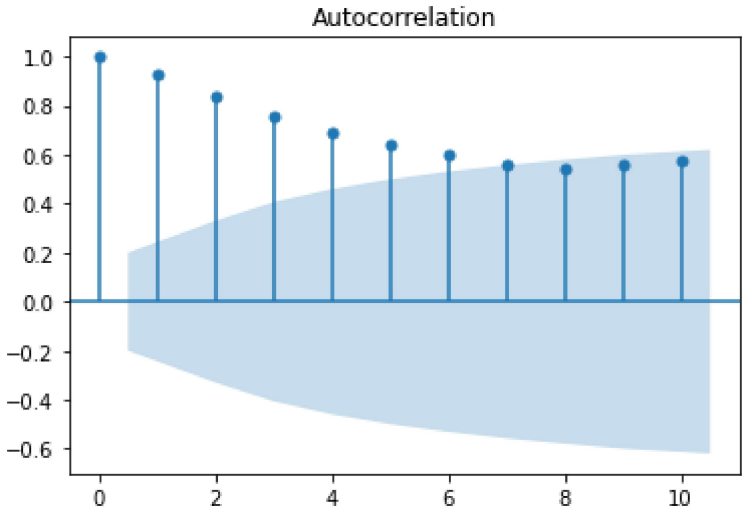
Out[121]:

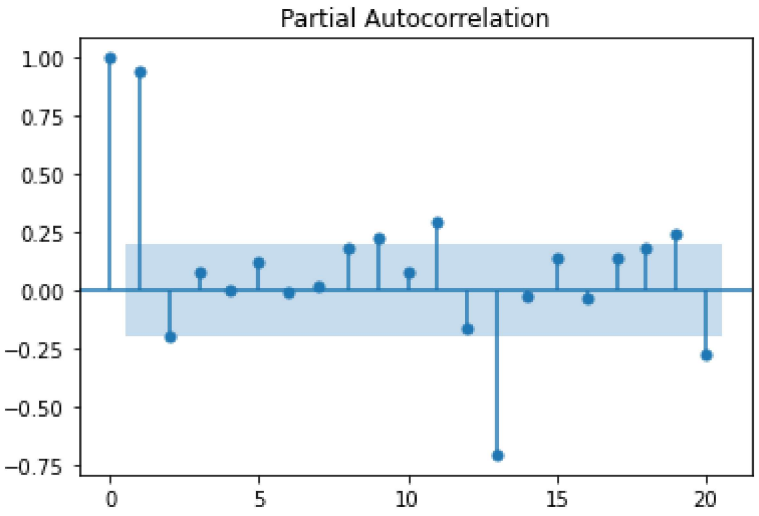


In [122]:
```
# ACF plots and PACF plots on Original data sets
tsa_plots.plot_acf(airlines_1.Passengers,lags=10)
tsa_plots.plot_pacf(airlines_1.Passengers)
```

Out[122]:







In [158]:
```
Train = airlines_1.head(100)
Test = airlines_1.tail(20)
```

In [159]:
```
# Creating a function to calculate the MAPE value for test data
def MAPE(pred,org):
    temp = np.abs((pred-org))*100/org
    return np.mean(temp)
```

In [160]:
```
# Simple Exponential Method
ses_model = SimpleExpSmoothing(Train["Passengers"]).fit()
pred_ses = ses_model.predict(start = Test.index[0],end = Test.index[-1])
MAPE(pred_ses,Test.Passengers)
```

```
C:\Users\Hp\anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.py:42
7: FutureWarning: After 0.13 initialization must be handled at model creation
  warnings.warn(
```

Out[160]: 9.470697707516285

In [161]:
```
# Holt method
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[161]: 9.526783804397628

In [162]:
```
# Holts winter exponential smoothing with additive seasonality and additive trend
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.ind
MAPE(pred_hwe_add_add,Test.Passengers)
```

```
<ipython-input-162-583fe044e1dc>:2: FutureWarning: the 'damped'' keyword is dep
recated, use 'damped_trend' instead
  hwe_model_add_add = ExponentialSmoothing(Train["Passengers"],seasonal="add",t
rend="add",seasonal_periods=4,damped=True).fit()
```

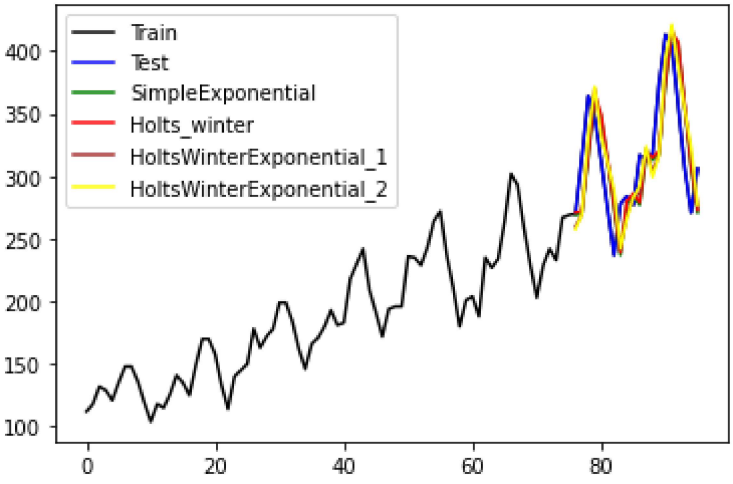
Out[162]: 9.683101766252062

In [163]:
```
# Holts winter exponential smoothing with multiplicative seasonality and additive
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.ind
MAPE(pred_hwe_mul_add,Test.Passengers)
```

Out[163]:
```
9.596787079249053
```

In [164]:
```
# Visualization of Forecasted values for Test data set using different methods
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='SimpleExponential',color="green")
plt.plot(pred_hw.index, pred_hw, label='Holts_winter',color="red")
plt.plot(pred_hwe_add_add.index,pred_hwe_add_add,label="HoltsWinterExponential_1"
plt.plot(pred_hwe_mul_add.index,pred_hwe_mul_add,label="HoltsWinterExponential_2"
plt.legend(loc='best')
```

Out[164]:
```
<matplotlib.legend.Legend at 0xd0af0b8>
```



In [ ]: