TEAM:
Nimisha - PES1201700083
Shashank - PES1201700667
Suhail - PES1201701420
Riya - PES1201701814

ASSIGNMENT 5

Problem statement:
On a chosen dataset, perform the time series model functions by calculating the moving average, single and double exponential smoothing.

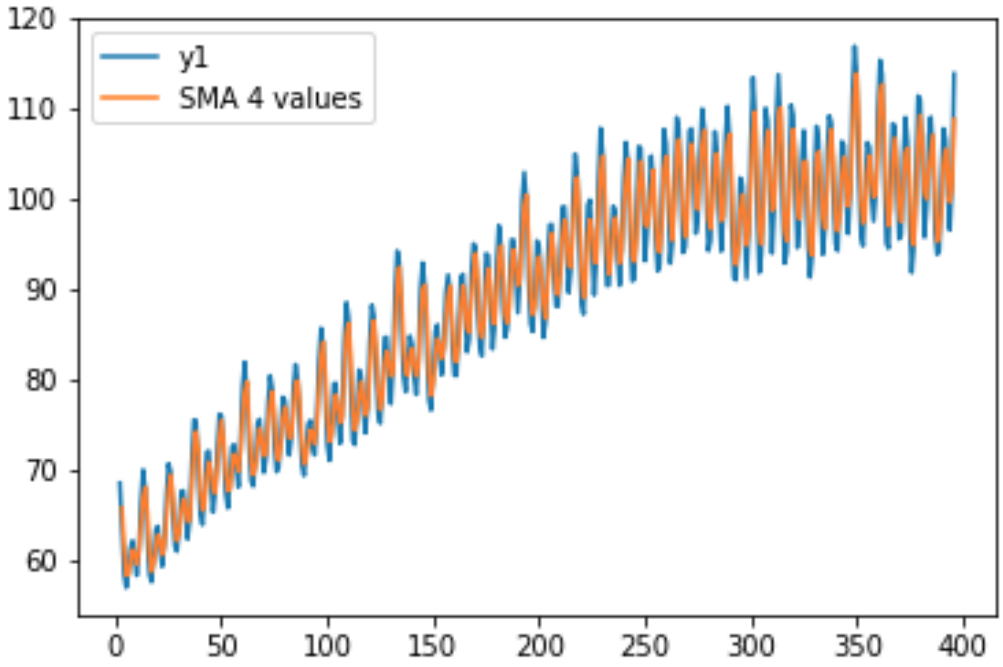
Dataset: Electric_Productions

1)Moving average:

In [8]:
```
#moving average
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

r=pd.read_csv("Electric_Production.csv")
y1=r['IPG2211A2N']
rolling_mean = y1.rolling(window=3).mean()
rolling_mean1 = y1.rolling(window=4).mean()
display(y1)
display(rolling_mean1)
plt.plot( rolling_mean ,label='y1')
plt.plot( rolling_mean1 ,label='SMA 4 values')
plt.legend()
```

Out[8]: <matplotlib.legend.Legend at 0x103e58d68>



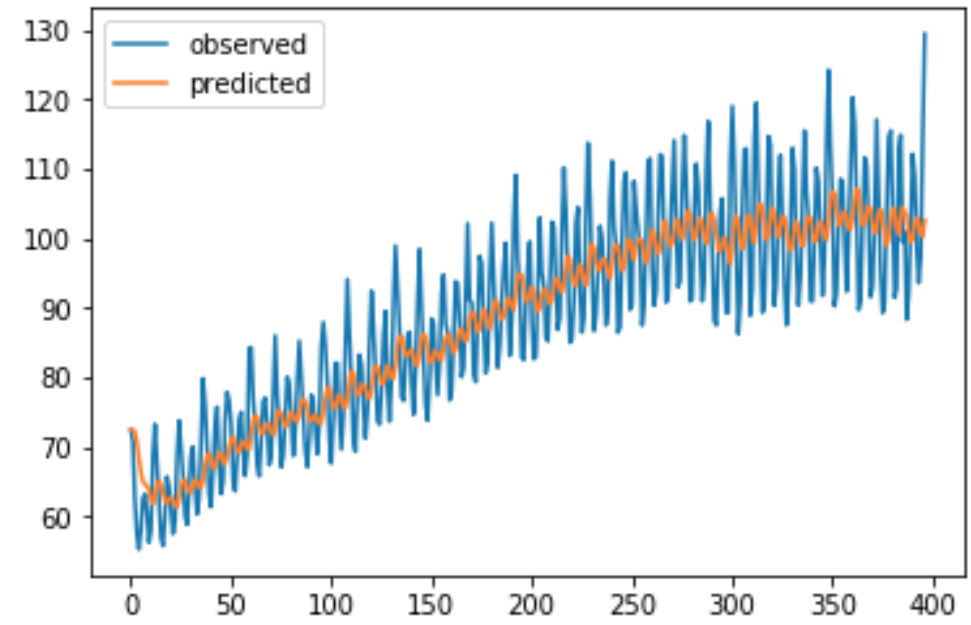
2) Single exponential smoothing:

In [9]:
```
#single exponential smoothing
from statsmodels.tsa.api import ExponentialSmoothing,SimpleExpSmoothing, Holt
import pandas as pd
import matplotlib.pyplot as plt
import numpy
import math

from statsmodels.tsa.holtwinters import ExponentialSmoothing
# prepare data
r = pd.read_csv("Electric_Production.csv")
# create class
data=list(r['IPG2211A2N'])
model = SimpleExpSmoothing(data)
# fit model
model_fit = model.fit(smoothing_level=0.160360045,optimized=False)
# make prediction
yhat_se = model_fit.predict(0,396)
plt.plot(data,label="observed")
plt.plot(yhat_se,label="predicted")
plt.legend()
plt.show()

def Rms(f_t):
    mae=0#mean absolute error
    mape=0#mean absolute percentage error
    mse=0#mean square error
    rmse=0#root mean square error
    for i in range(len(f_t)):
        mae=mae+(abs(data[i]-f_t[i]))
        mape=mape+(abs(data[i]-f_t[i]))/f_t[i]
        mse=mse+((abs(data[i]-f_t[i])**2))
    mae=mae/(len(f_t))
    mape=(mape/len(f_t))*100
    mse=mse/(len(f_t))
    rmse=math.sqrt(mse)
    print("Root Mean square error",rmse)

Rms(yhat_se)
```




```
Root Mean square error 8.394076617276836
```

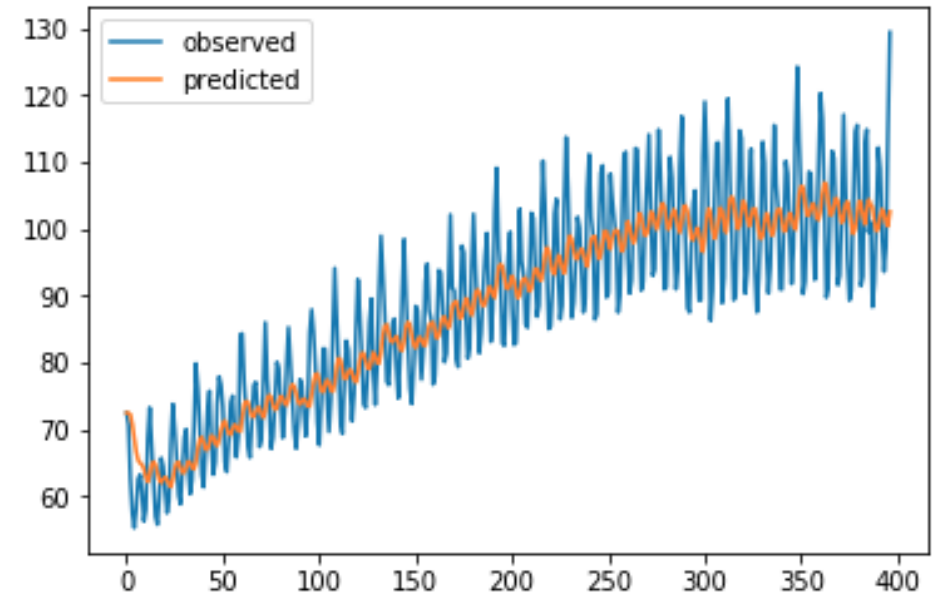
3) Double exponential smoothing:

In [60]:

```
#double exponential smoothing
from statsmodels.tsa.api import ExponentialSmoothing,SimpleExpSmoothing, Holt
import pandas as pd
import matplotlib.pyplot as plt
import numpy
import math

from statsmodels.tsa.holtwinters import ExponentialSmoothing
# prepare data
r = pd.read_csv("Electric_Production.csv")
# create class
data=list(r['IPG2211A2N'])
model = Holt(data)
model_fit = model.fit(smoothing_level=0.288049603,smoothing_slope=0.02009562,optimized=False)
# make prediction
yhat_de = model_fit.predict(0,396)
plt.plot(data,label="observed")
plt.plot(yhat_de,label="predicted")
plt.legend()
plt.show()

def Rms(f_t):
    mae=0#mean absolute error
    mape=0#mean absolute percentage error
    mse=0#mean square error
    rmse=0#root mean square error
    for i in range(len(f_t)):
        mae=mae+(abs(data[i]-f_t[i]))
        mape=mape+(abs(data[i]-f_t[i]))/f_t[i]
        mse=mse+((abs(data[i]-f_t[i])**2))
    mae=mae/(len(f_t))
    mape=(mape/len(f_t))*100
    mse=mse/(len(f_t))
    rmse=math.sqrt(mse)
    print("Root Mean square error",rmse)
Rms(yhat_de)
```




```
Root Mean square error 8.375966644763022
```

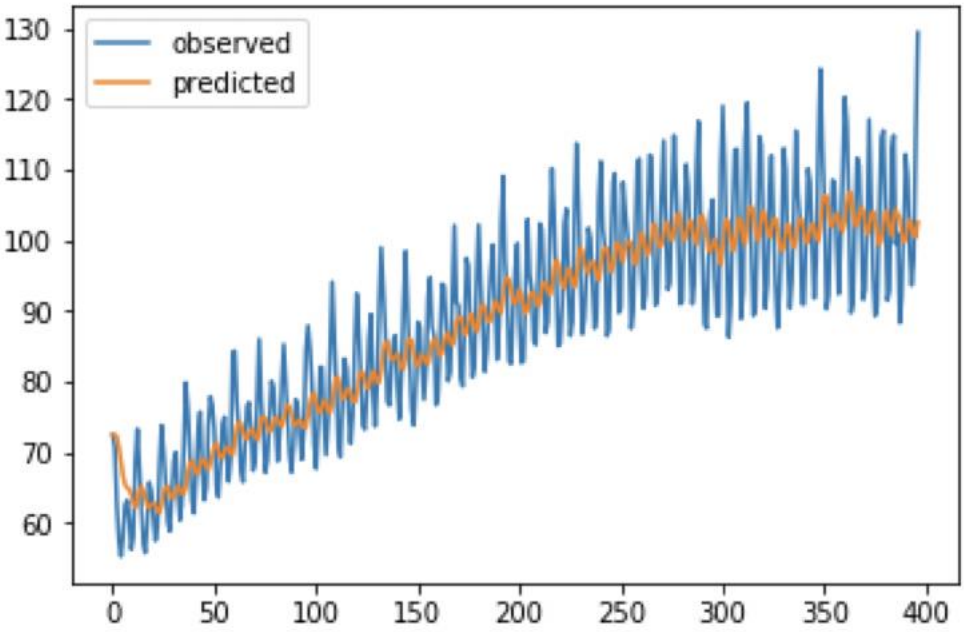
4)Triple Exponential Smoothing

In [17]:
```
#triple exponential smoothing
from statsmodels.tsa.api import ExponentialSmoothing,SimpleExpSmoothing, Holt
import pandas as pd
import matplotlib.pyplot as plt
import numpy
import math

from statsmodels.tsa.holtwinters import ExponentialSmoothing
# prepare data
r = pd.read_csv("Electric_Production.csv")
# create class
data=list(r['IPG2211A2N'])
model = ExponentialSmoothing(data)
model_fit = model.fit(smoothing_level=0.15,smoothing_slope=0.002,smoothing_seasonal=10,optimized=F
alse)
# make prediction
yhat_te = model_fit.predict(0,396)
plt.plot(data,label="observed")
plt.plot(yhat_te,label="predicted")
plt.legend()
plt.show()

def Rms(f_t):
    mae=0#mean absolute error
    mape=0#mean absolute percentage error
    mse=0#mean square error
    rmse=0#root mean square error
    for i in range(len(f_t)):
        mae=mae+(abs(data[i]-f_t[i]))
        mape=mape+(abs(data[i]-f_t[i]))/f_t[i]
        mse=mse+((abs(data[i]-f_t[i])**2))
    mae=mae/(len(f_t))
    mape=(mape/len(f_t))*100
    mse=mse/(len(f_t))
    rmse=math.sqrt(mse)
    print("Root Mean square error",rmse)

Rms(yhat_te)
```




```
Root Mean square error 8.375966644763022
```