```
def single_ES(data, alpha):
  if not 0<alpha<=1:
    raise ValueError('alpha must be between 0 and 1')

  smoothed_forecasting = [data[0]]

  for i in range(1, len(data)+1):
    new_forecasting = alpha * data[i-1] + (1-alpha) * smoothed_forecasting[i-1]
    smoothed_forecasting.append(new_forecasting)

  return smoothed_forecasting
```

```
def double_ES(data, alpha, beta):
  n = len(data)
  if n<2:
    raise ValueError("data must have at least two periods' data")
  if not 0<alpha<=1:
    raise ValueError('alpha must be between 0 and 1')
  if not 0<beta<=1:
    raise ValueError('beta must be between 0 and 1')

  I = [data[0]]
  S = [data[1] - data[0]]
  y = ["N/A"]*2

  for t in range(1, n):
    new_I = alpha * data[t] + (1-alpha) * (I[t-1] + S[t-1])
    I.append(new_I)
    new_S = beta * (I[t]-I[t-1]) + (1-beta) * S[t-1]
    S.append(new_S)
    y.append(new_I + new_S)

  return I, S, y
```

```
def triple_ES(D, alpha, beta, gamma, N):
  S = ["N/A"] * (2*N-1)
```

```
  S.append((sum(D[N:2*N]) - sum(D[0:N]))/N**2) #Initilize  trend for the period 2N, i.e., S_2N
  #Intialize seasonality for the second season of N periods, i.e., c_N+1, c_N+2, ..., c_2N
  c = ["N/A"]*N
  for t in range(N):
    new_c = D[t]/(sum(D[0:N])/N)+D[N+t]/(sum(D[N:2*N])/N)
    new_c/=2
    c.append(new_c)
  #Intialize level for the period 2N, i.e., I_2N
  I = ["N/A"] * (2*N-1)
  I.append(D[2*N-1] / c[2*N-1])
  #Intialize forecasting for the period 2N+1, i.e., y_2N+1
  y = ["N/A"] * (2*N)
  y.append((I[2*N-1] + S[2*N-1]) * c[N])


  #Recursively # Recursively generating the forecasting for level I, trend S, seasonality c, and demand y
  for t in range(2*N, len(D)):
    I.append(alpha * D[t] / c[t-N] + (1-alpha) * (I[t-1] + S[t-1]))
    S.append(beta * (I[t]-I[t-1]) +( 1-beta) * S[t-1])
    c.append(gamma * D[t] / I[t] + (1-gamma) * c[t-N])
    y.append((I[t] + S[t]) * c[t+1-N])

  return I, S, c, y
```

```
import numpy as np
from scipy.stats import linregress

def LR(independent_variable, dependent_variable):
  x = np.array(independent_variable)
  y = np.array(dependent_variable)

  # Perform linear regression
  slope, intercept, r_value, p_value, stderr = linregress(x, y)

  # Create the fitting line
  fitting_line = slope * x + intercept
```

```
  return fitting_line, slope, intercept
```

```
def performance_measures(forecasting, observation):
  n = len(observation)
  sum_AD, sum_SE, sum_APE = 0.0, 0.0, 0.0

  for i in range(n):
    if forecasting[i] == "N/A":
      n-=1 #n-=1 means n=n-1
    else:
      e_t = forecasting[i]-observation[i]
      sum_AD += abs(e_t) #sum_AD += abs(e_t) means sum_AD = sum_AD+abs(e_t)
      sum_SE += e_t**2
      sum_APE += abs(e_t)/observation[i]


  MAD=sum_AD / n; MSE=sum_SE / n; MAPE=sum_APE / N


  return MAD, MSE, MAPE
```

```
import pandas as pd
import matplotlib.pyplot as plt

df=pd.read_excel("HW 5 - ES and LR.xlsx")

month=df['Month'].tolist()
sales=df['Demand'].tolist()

new_df=pd.DataFrame()

period=[]
for i in range(len(month) + 1):
  period.append(i+1)
period += ["MAD", "MSE", "MAPE"]

print(period)
```

```
print(period)


new_df['Month']=period


forecasting_LR, slope, intercept = LR(month, sales)
forecasting_LR=np.append(forecasting_LR, intercept+slope*(len(month)+1))
MAD, MSE, MAPE = performance_measures(forecasting_LR, sales)
forecasting_LR=np.append(forecasting_LR, MAD)
forecasting_LR=np.append(forecasting_LR, MSE)
forecasting_LR=np.append(forecasting_LR, MAPE)
new_df['LR']=forecasting_LR

alpha = 0.5
forecasting_SES = single_ES(sales, alpha)
results = performance_measures(forecasting_SES, sales)
measures = list(results)
forecasting_SES += measures
new_df['Single ES']=forecasting_SES


alpha = 0.5; beta = 0.35
I, S, forecasting_DES = double_ES(sales, alpha, beta)
results = performance_measures(forecasting_DES, sales)
measures = list(results)
forecasting_DES += measures
new_df['Double ES']=forecasting_DES


alpha = 0.5; beta = 0.35; gamma = 0.6
N = 4
I, S, c, forecasting_TES = triple_ES(sales, alpha, beta, gamma, N)
results = performance_measures(forecasting_TES, sales)
measures = list(results)
forecasting_TES += measures
new_df['Triple ES']=forecasting_TES


new_df.to_excel("Results.xlsx", index = False)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 'MAD', 'MSE', 'MAPE']
```