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


np.random.seed(88)
pd.set_option('display.max_rows',420)
pd.set_option('display.max_columns',20)

n=100
ID = np.arange(1, n+1)
age= np.random.randint(20,70,size=n)
sex=np.random.choice(['male','female'],size = n)
height_cm=np.clip (np.random.normal(165,10,size=n),150, 170).round(1)
weight_kg=np.clip (np.random.normal(90,10,size=n),80,None).round(1)
muscle_kg=(weight_kg*np.random.uniform(0.3 ,0.4 , size=n)).round(1)
fat_kg=(weight_kg-muscle_kg).round(1)
activity= np.random.choice(['low' , 'medium' ,'high'],size=n ,p=[0.3,0.5,0.2])
protein_day_kg=np.random.uniform(0.8,1.6,size=n).round(2)
height_cm_sq=(height_cm*height_cm)
BMI= ((weight_kg/height_cm_sq)*10000).round(2)


Df=pd.DataFrame({"ID":ID,
"Age":age,
"sex":sex,
"height_cm":height_cm,
"weight_kg":weight_kg,
"muscle_kg":muscle_kg,
"fat_kg":fat_kg,
"activity level":activity,
"protein per day":protein_day_kg,
 "BMI":BMI})
```

```
def adj_muscle_pct(base_muscle_pct,activity, protein_per_day):
    adjustment=0

    if activity == "high":
        adjustment -= 0.95
    elif activity == "medium":
        adjustment -= 0.5

    if protein_per_day >= 1.3:
        adjustment -=0.15
    elif 1.0 < protein_per_day <= 1.2:
        adjustment -= 0.10
    else:
        adjustment += 0.05

    final_pct=base_muscle_pct+adjustment

    return max(0.10,min(final_pct,0.60))

def simulation_func(Df,weight_loss_pct,base_muscle_pct):
    out=Df.copy()

    out['total_loss_kg']=Df['weight_kg']*weight_loss_pct
    out['muscle_fraction']=out.apply(
        lambda row : adj_muscle_pct(
            base_muscle_pct,
            row['activity level'],
            row['protein per day']
        ),
        axis = 1
    )
    out['muscle_loss_kg']=out['total_loss_kg']*out['muscle_fraction']
    out['fat_loss_kg']=out['total_loss_kg']-out['muscle_loss_kg']
    out['new_muscle_kg']=Df['muscle_kg']-out['muscle_loss_kg']
```

```
    out['new_weight']=Df['weight_kg']-out['muscle_loss_kg']
    out['muscle_loss_pct_of_original']=(out['muscle_loss_kg']/Df['muscle_kg'])*100
    out['age_decline_years'] = np.where(out['sex']==
'male',out['muscle_loss_pct_of_original']/0.47,out['muscle_loss_pct_of_original']/0.37).round(1)

    return out

scenarios = [
    (0.09,0.40),
    (0.13,0.40),
    (0.16,0.40),
    (0.175,0.40)
]

results=[]

for pct_loss,base_pct in scenarios:
    simulated_values=
simulation_func(Df,weight_loss_pct=pct_loss,base_muscle_pct=base_pct)
    simulated_values['scenario']=f'{int (pct_loss*100)}% Weight loss'
    results.append(simulated_values)

all_sim_values = pd.concat(results, ignore_index=True)
```