MOUNTAINS OF THE MOON UNIVERSITY
FACULTY OF SCIENCE TECHNOLOGY AND INNOVATION
BACHELOR OF SCIENCE IN SOFTWARE ENG.
ARTIFICIAL INTELLIGENCE
NAME: Davis Itaagi
REG.NO: 2023/U/MMU/BSSE/01134
LECTURER: MR OCEN SAMUEL

February 22, 2024

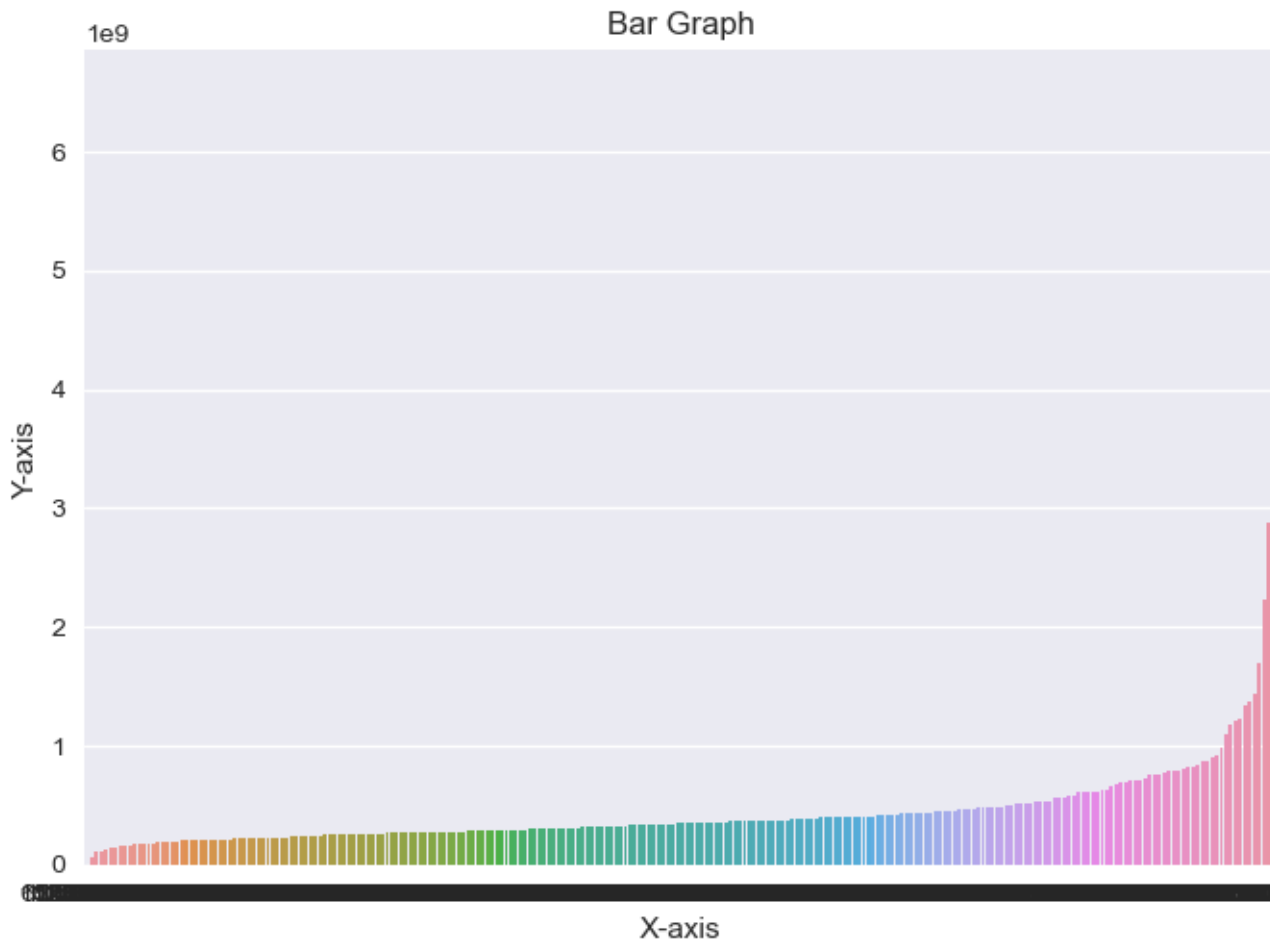
1 Introduction

```
import pandas as pd
dt = pd.read_csv("C:\\Users\\DAVIS\\Desktop\\Pandas\\loan_amount.csv")
dt.head()
Gender ApplicantIncome CoapplicantIncome LoanAmount
0 Male 5720 0 110.0
1 Male 3076 1500 126.0
2 Male 5000 1800 208.0
3 Male 2340 2546 100.0
4 Male 3276 0 78.0
dt.tail()
Gender ApplicantIncome CoapplicantIncome LoanAmount
362 Male 1202700 533100 33900.0
363 Male 1247400 212700 34500.0
364 Male 975000 597900 37800.0
365 Male 1500000 717900 47400.0
366 Male 2760000 0 29400.0
loan_amount = dt[dt['Gender'] == 'Male'][: 300]
```

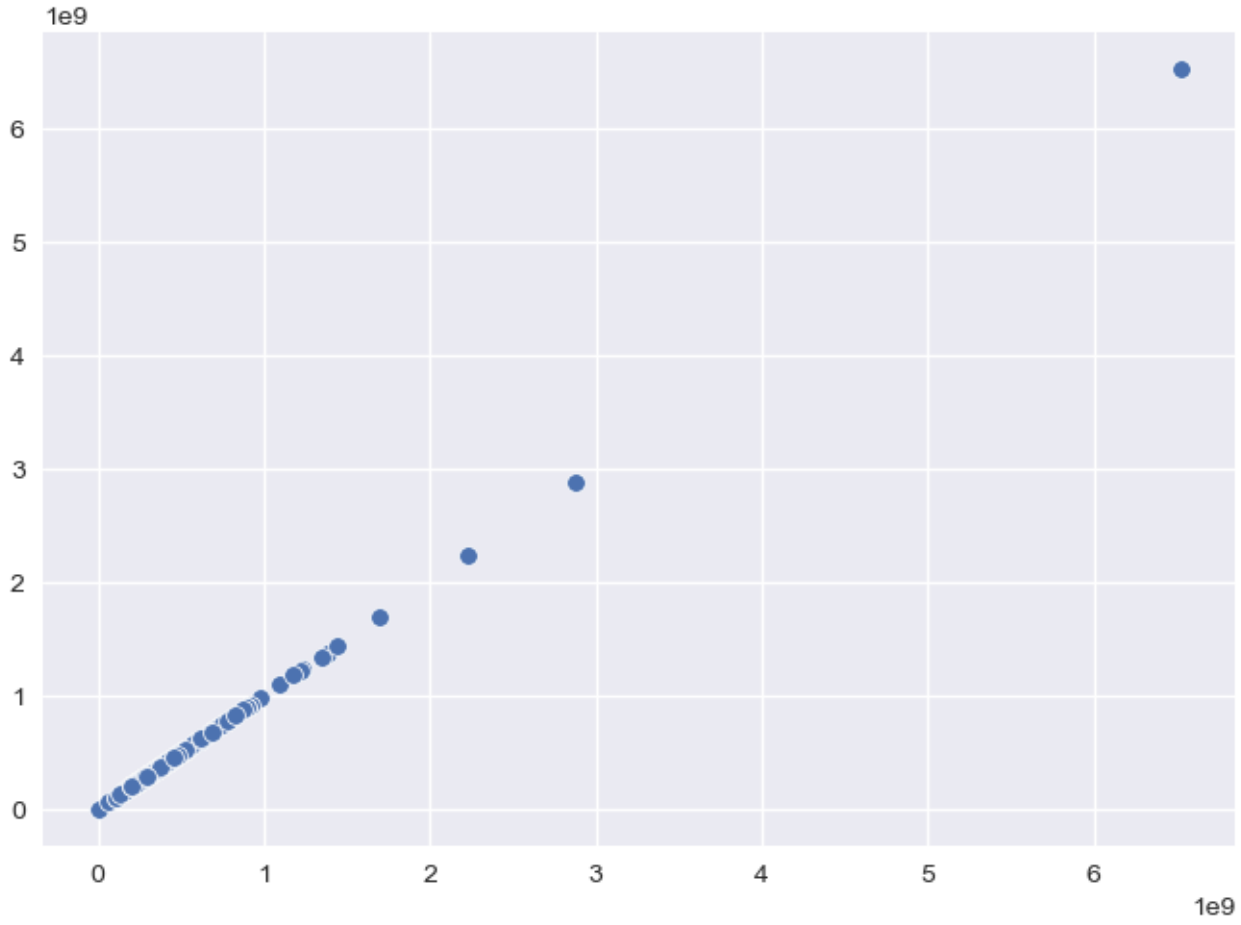
```
loan_amount
Gender ApplicantIncome CoapplicantIncome LoanAmount
0 Male 514800000 0 9900000.0
1 Male 276840000 135000000 11340000.0
2 Male 450000000 162000000 18720000.0
3 Male 210600000 229140000 9000000.0
4 Male 294840000 0 7020000.0
... ... ... ... ...
362 Male 360810000 159930000 10170000.0
363 Male 374220000 63810000 10350000.0
364 Male 292500000 179370000 11340000.0
365 Male 450000000 215370000 14220000.0
366 Male 828000000 0 8820000.0
286 rows × 4 columns

#All the money iis in $,
#$1= Ugx300

dt['ApplicantIncome'] = dt['ApplicantIncome'].apply(lambda x: x*300)
dt['LoanAmount'] = dt['LoanAmount'].apply(lambda x: x*300)
dt['CoapplicantIncome'] = dt['CoapplicantIncome'].apply(lambda x: x*300)
dt
Gender ApplicantIncome CoapplicantIncome LoanAmount
0 Male 154440000000 0 2.970000e+09
1 Male 83052000000 40500000000 3.402000e+09
2 Male 135000000000 48600000000 5.616000e+09
3 Male 63180000000 68742000000 2.700000e+09
4 Male 88452000000 0 2.106000e+09
... ... ... ... ...
362 Male 108243000000 47979000000 3.051000e+09
363 Male 112266000000 19143000000 3.105000e+09
364 Male 87750000000 53811000000 3.402000e+09
365 Male 135000000000 64611000000 4.266000e+09
366 Male 248400000000 0 2.646000e+09
367 rows × 4 columns

#We convert the data to lists ansd use it to plot the graph
app_list = loan_amount['ApplicantIncome'].tolist()
loan_list = loan_amount['LoanAmount'].tolist()

app_list[:10]
[514800000,
 276840000,
 450000000,
 210600000,
 294840000,
```

```
 194850000,
 349290000,
 1226970000,
 216000000,
 278190000]
#To plot the graph of Applicant Icome  against Loan amount, wer need to define a function th
#of Apllicant income and loan amount and return their respective values.

def line_of_best_fit(xs,ys):
    slope = (((mean(xs)*mean(ys)) - mean(xs*ys))/(mean(xs)*mean(xs) - mean(xs*xs)))
    y_intercept = mean(ys) - slope*mean(xs)
    return slope, y_intercept

#import numpy
import numpy as np
xs = np.array(app_list, dtype = np.float64)
ys = np.array(app_list, dtype = np.float64)
ys
array([5.14800e+08, 2.76840e+08, 4.50000e+08, 2.10600e+08, 2.94840e+08,
       1.94850e+08, 3.49290e+08, 1.22697e+09, 2.16000e+08, 2.78190e+08,
       1.96650e+08, 3.74940e+08, 1.09557e+09, 5.10030e+08, 4.12470e+08,
       3.40740e+08, 8.30340e+08, 1.17000e+08, 1.69920e+08, 4.86000e+08,
       0.00000e+00, 3.92670e+08, 6.75000e+08, 3.39480e+08, 2.64780e+08,
       5.62500e+08, 2.94120e+08, 2.50470e+08, 2.46600e+08, 2.83500e+08,
       6.61500e+08, 2.04030e+08, 5.24970e+08, 3.27870e+08, 5.06610e+08,
       1.57500e+08, 5.85000e+08, 3.83400e+08, 3.74670e+08, 2.12040e+08,
       6.11280e+08, 7.20000e+08, 2.17710e+08, 3.15000e+08, 3.70440e+08,
       4.76370e+08, 2.47500e+08, 3.25170e+08, 4.24800e+08, 2.17350e+08,
       6.31440e+08, 4.04100e+08, 2.62530e+08, 4.23000e+08, 3.10050e+08,
       6.89940e+08, 2.21220e+08, 4.01670e+08, 3.67470e+08, 3.51000e+08,
       4.27500e+08, 3.22470e+08, 2.87010e+08, 5.72040e+08, 3.07170e+08,
       3.44610e+08, 6.52761e+09, 3.72240e+08, 7.60410e+08, 4.01040e+08,
       4.17150e+08, 3.21390e+08, 2.75940e+08, 2.91150e+08, 2.86920e+08,
       1.21662e+09, 3.92760e+08, 4.28940e+08, 4.14810e+08, 2.99970e+08,
       3.15000e+08, 8.74710e+08, 6.15150e+08, 4.00680e+08, 3.51090e+08,
       2.41830e+08, 2.01870e+08, 4.06170e+08, 4.05000e+08, 4.07070e+08,
       3.67380e+08, 3.75030e+08, 3.81870e+08, 1.36440e+08, 6.51600e+07,
       2.81250e+08, 2.09970e+08, 3.01500e+08, 2.25000e+08, 4.20030e+08,
       5.85000e+08, 6.75000e+08, 2.76570e+08, 2.98890e+08, 2.99970e+08,
       3.05190e+08, 3.00870e+08, 3.60000e+08, 3.83220e+08, 4.05000e+08,
       1.81260e+08, 4.25430e+08, 2.78010e+08, 6.11460e+08, 2.88000e+09,
       9.80100e+08, 2.94840e+08, 7.83270e+08, 4.26780e+08, 5.31000e+08,
       2.76390e+08, 2.50470e+08, 4.50000e+08, 2.21670e+08, 4.36950e+08,
       1.43910e+08, 3.82140e+08, 3.89970e+08, 5.24070e+08, 7.10550e+08,
       3.73500e+08, 2.66760e+08, 5.02470e+08, 2.86200e+08, 2.04120e+08,
       1.02690e+08, 2.73780e+08, 4.03470e+08, 4.70250e+08, 2.71530e+08,
```

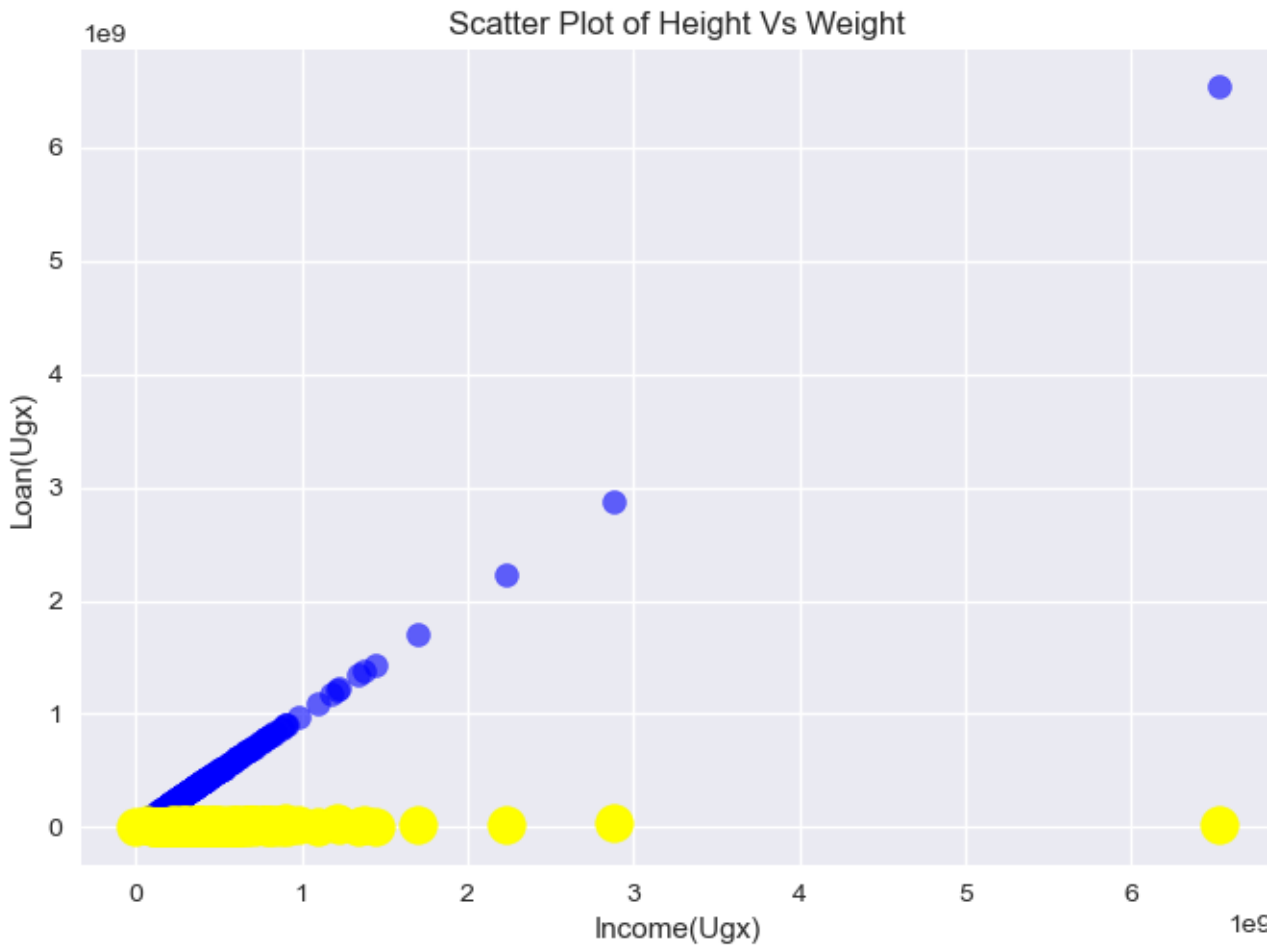
```
        2.18790e+08, 4.42080e+08, 2.25000e+08, 4.61520e+08, 1.37808e+09,
        3.56220e+08, 3.90060e+08, 3.92220e+08, 9.14940e+08, 4.03470e+08,
        4.06890e+08, 8.25030e+08, 1.17747e+09, 7.08660e+08, 3.67470e+08,
        3.40650e+08, 2.38860e+08, 9.00000e+08, 5.24970e+08, 4.31640e+08,
        1.80000e+08, 2.28600e+08, 1.71000e+08, 7.83540e+08, 2.56950e+08,
        2.71440e+08, 2.35170e+08, 4.46400e+08, 2.76660e+08, 4.79970e+08,
        4.84560e+08, 5.13720e+08, 3.37860e+08, 2.62260e+08, 2.47230e+08,
        7.04700e+08, 3.15630e+08, 3.37230e+08, 1.94940e+08, 3.15000e+08,
        2.60640e+08, 2.29770e+08, 3.09240e+08, 2.17080e+08, 4.66200e+08,
        1.34199e+09, 1.05570e+08, 2.00790e+08, 5.54940e+08, 2.26170e+08,
        3.89970e+08, 3.45960e+08, 3.49830e+08, 3.15900e+08, 2.28510e+08,
        4.50000e+08, 9.00000e+08, 3.54870e+08, 2.63250e+08, 2.91780e+08,
        3.47670e+08, 3.60900e+08, 2.57220e+08, 2.99970e+08, 2.70630e+08,
        2.51280e+08, 2.68380e+08, 3.07530e+08, 1.69560e+09, 2.69550e+08,
        3.22110e+08, 5.24970e+08, 3.15720e+08, 2.23173e+09, 5.10030e+08,
        2.49570e+08, 5.85000e+08, 3.27060e+08, 1.94940e+08, 4.50000e+08,
        8.10000e+08, 7.87500e+08, 1.94130e+08, 1.77480e+08, 4.48470e+08,
        7.49970e+08, 3.30030e+08, 2.84940e+08, 2.94390e+08, 1.61280e+08,
        3.42720e+08, 3.70800e+08, 6.75000e+08, 5.67000e+08, 2.34000e+08,
        2.45970e+08, 6.75000e+08, 3.47310e+08, 6.14250e+08, 3.33720e+08,
        4.78260e+08, 1.85940e+08, 4.50000e+08, 1.53000e+08, 2.77470e+08,
        2.40030e+08, 3.06000e+08, 1.44000e+09, 4.79970e+08, 2.58750e+08,
        2.34000e+08, 4.53690e+08, 6.26220e+08, 3.15000e+08, 4.95810e+08,
        8.72910e+08, 1.36440e+08, 2.16000e+08, 2.06280e+08, 2.12400e+08,
        2.36070e+08, 3.57480e+08, 3.16980e+08, 4.23000e+08, 6.17220e+08,
        7.50060e+08, 3.05190e+08, 2.58120e+08, 3.07620e+08, 2.25000e+08,
        7.80030e+08, 2.05470e+08, 5.23530e+08, 4.60710e+08, 4.78440e+08,
        6.84270e+08, 3.41190e+08, 2.25000e+08, 2.81880e+08, 3.60000e+08,
        2.04210e+08, 3.60810e+08, 3.74220e+08, 2.92500e+08, 4.50000e+08,
        8.28000e+08])


import seaborn as sns
import matplotlib.pyplot as plt

# Create a bar graph using Seaborn
sns.barplot(x=xs, y=ys)

# Customize the plot (optional)
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Bar Graph")
plt.show()
```



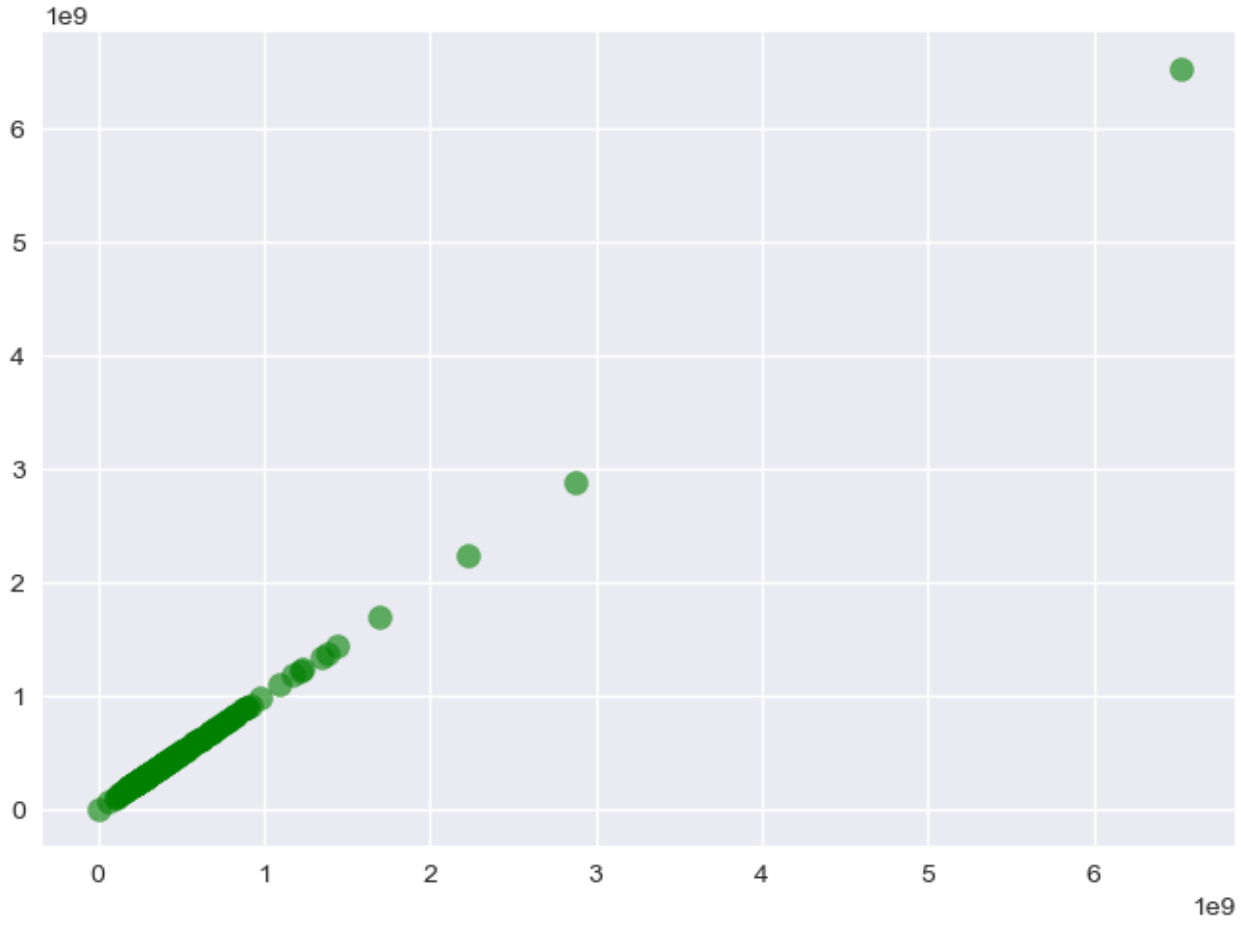

```
import matplotlib.pyplot as plt
from matplotlib import style
from statistics import mean

    import seaborn as sns

# Create the scatter plot with Seaborn style
sns.scatterplot(x=xs, y=ys)

# Show the plot
plt.show()
```




```
    style.use('seaborn')
plt.scatter(xs,ys, label = 'Data Points', alpha = 0.6, color = 'blue', s = 75)
plt.scatter(app_list, loan_list, label = 'Predicted loan amount', color ='yellow', s = 200)
plt.xlabel('Income(Ugx)')
plt.ylabel('Loan(Ugx)')
plt.title("Scatter Plot of Height Vs Weight")
```




```
   style.use('seaborn')
plt.scatter(xs,ys, label = 'Data Points', alpha = 0.6, color = 'blue', s = 75)
plt.scatter(app_list, loan_list, label = 'Predicted loan amount', color ='yellow', s = 200)
plt.xlabel('Income(Ugx)')
plt.ylabel('Loan(Ugx)')
plt.title("Scatter Plot of Height Vs Weight")
```


1e9
6
5
4
3
2
1
0
0
1
2
3
4
5
6
1e9