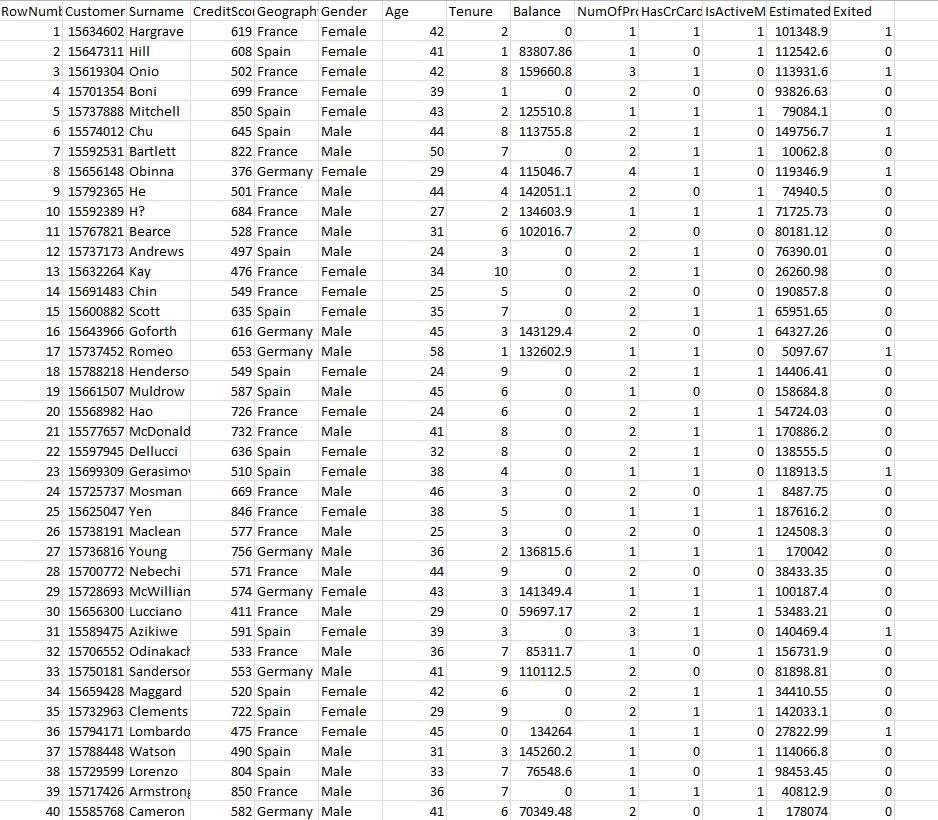
Data Visualization and Pre-processing Assignment -2

|  |  |
| --- | --- |
| Assignment Date | 26 September 2022 |
| Team ID | PNT2022TMID07052 |
| Project Name | AI BASED DISCOURSE FOR BANKING INDUSTRY |
| Student Name | C.Girish |
| Student Roll Number | 130719104027 |
| Maximum Marks | 2 Marks |

**Question-1.**Download dataset

Solution:



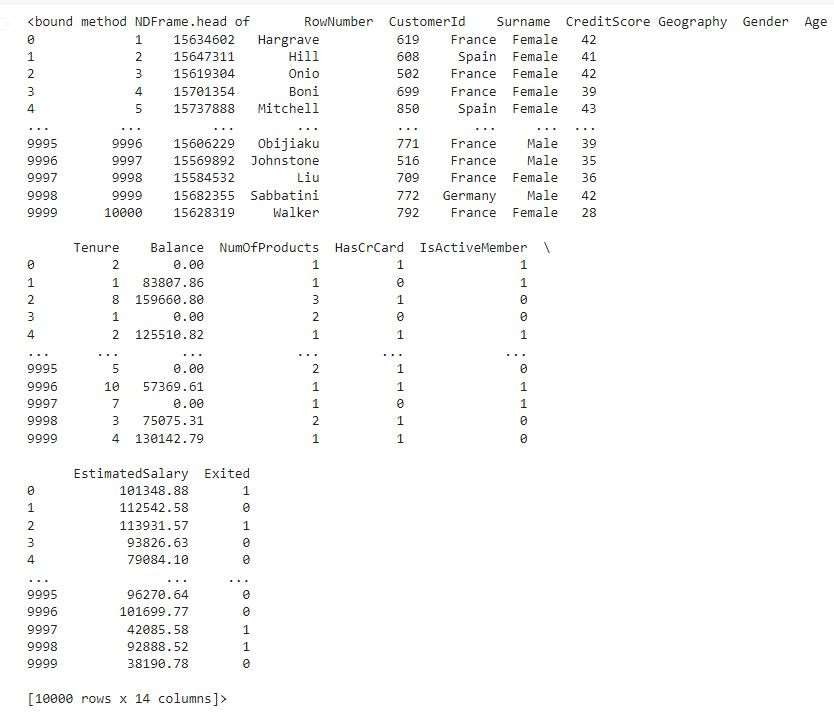
**Question-2.**Load the dataset

Solution:

import numpy as np import pandas as pd import seaborn as sns

importmatplotlib.pyplot as plt import sklearn

data = pd.read\_csv(r'Churn\_Modelling.csv') df.head

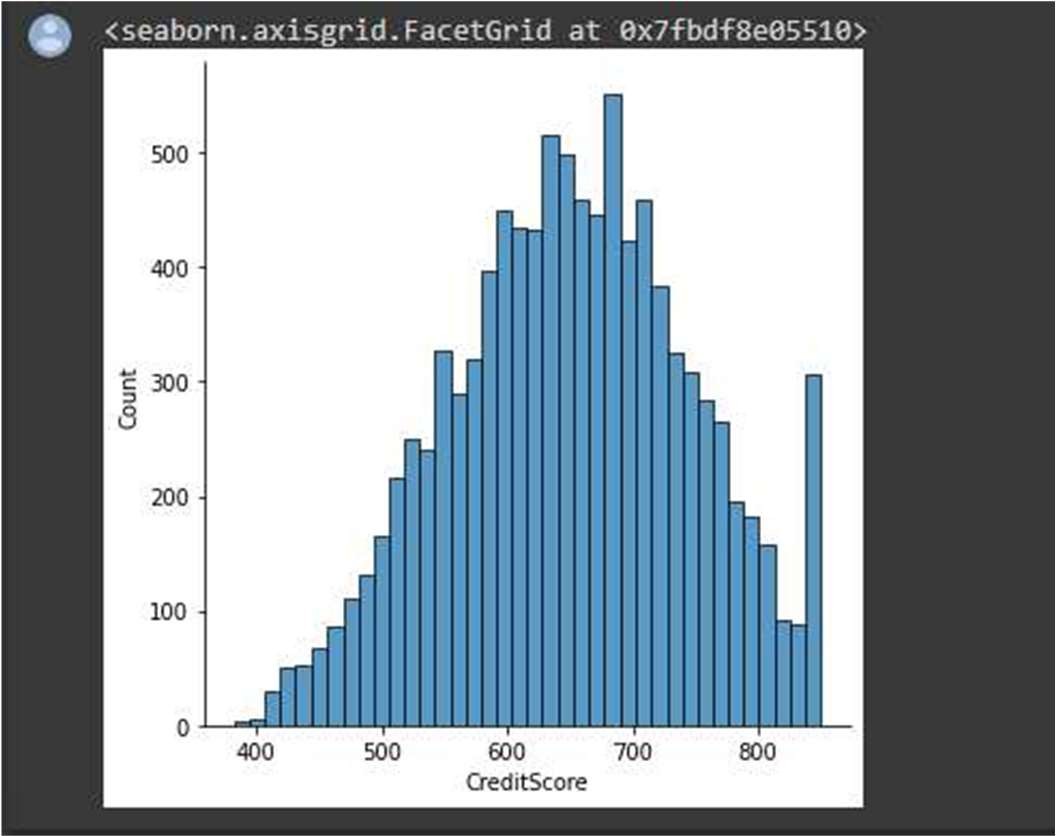


**Question-3.**Perform Below Visualizations.

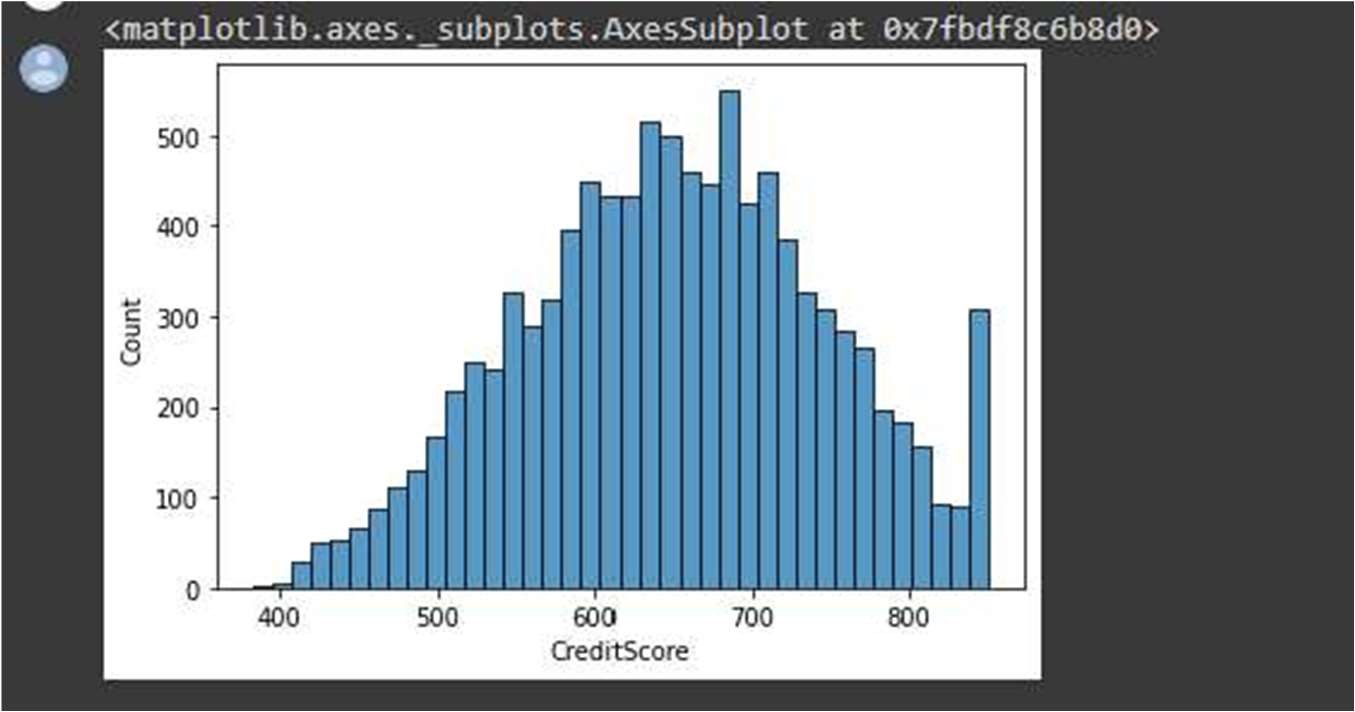
* 1. ***Univariate Analysis***

Solution:

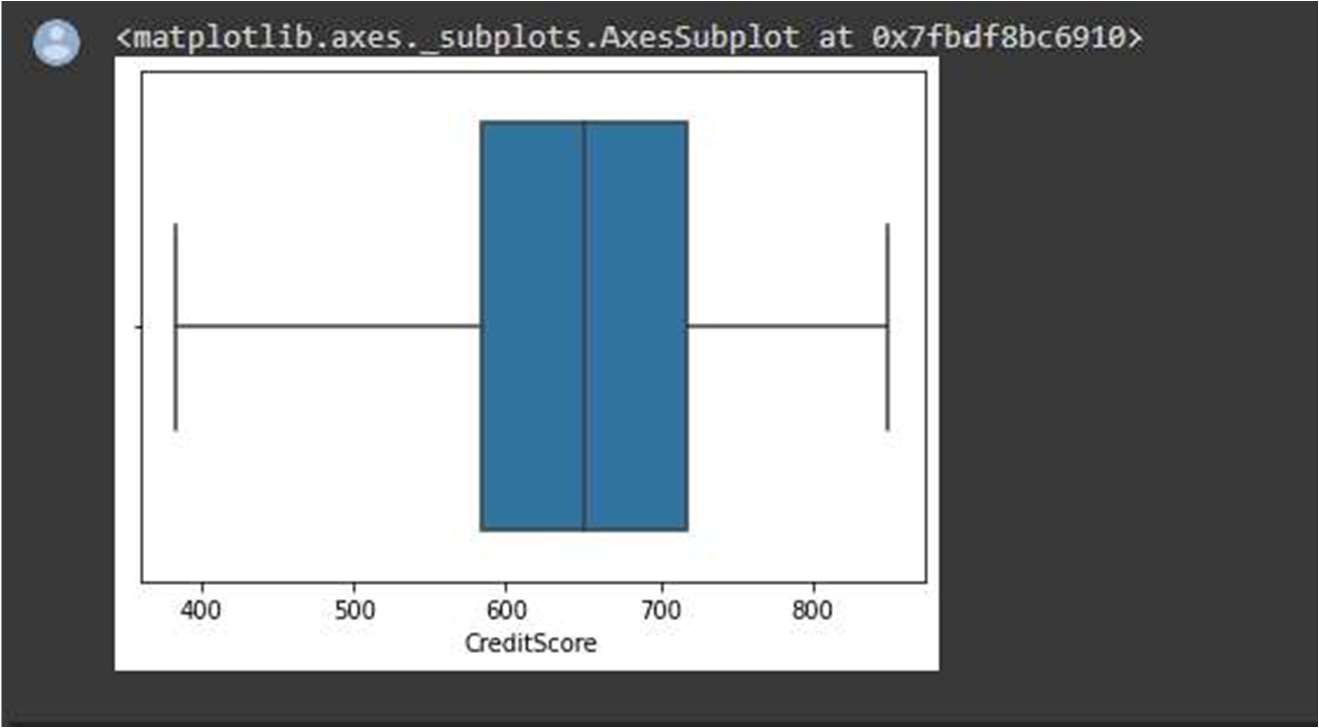
sns.displot(data['CreditScore'])



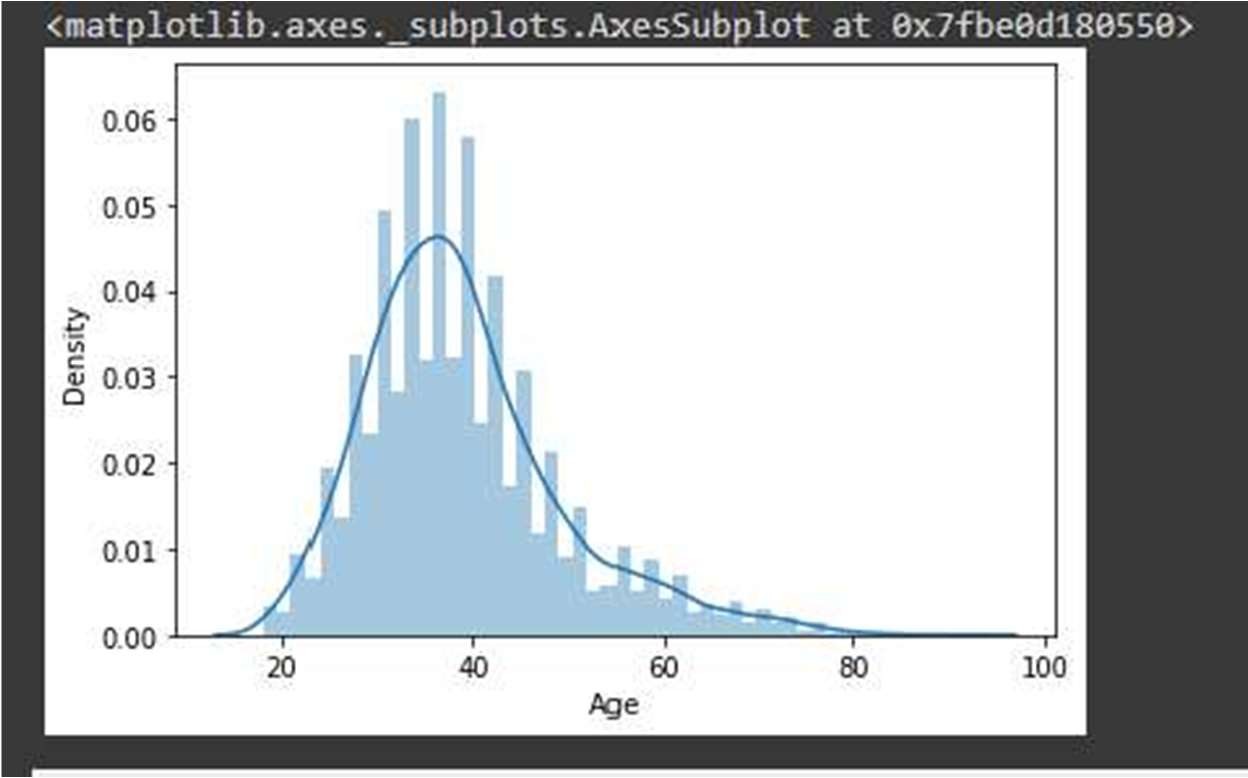
sns.histplot(data['CreditScore'])



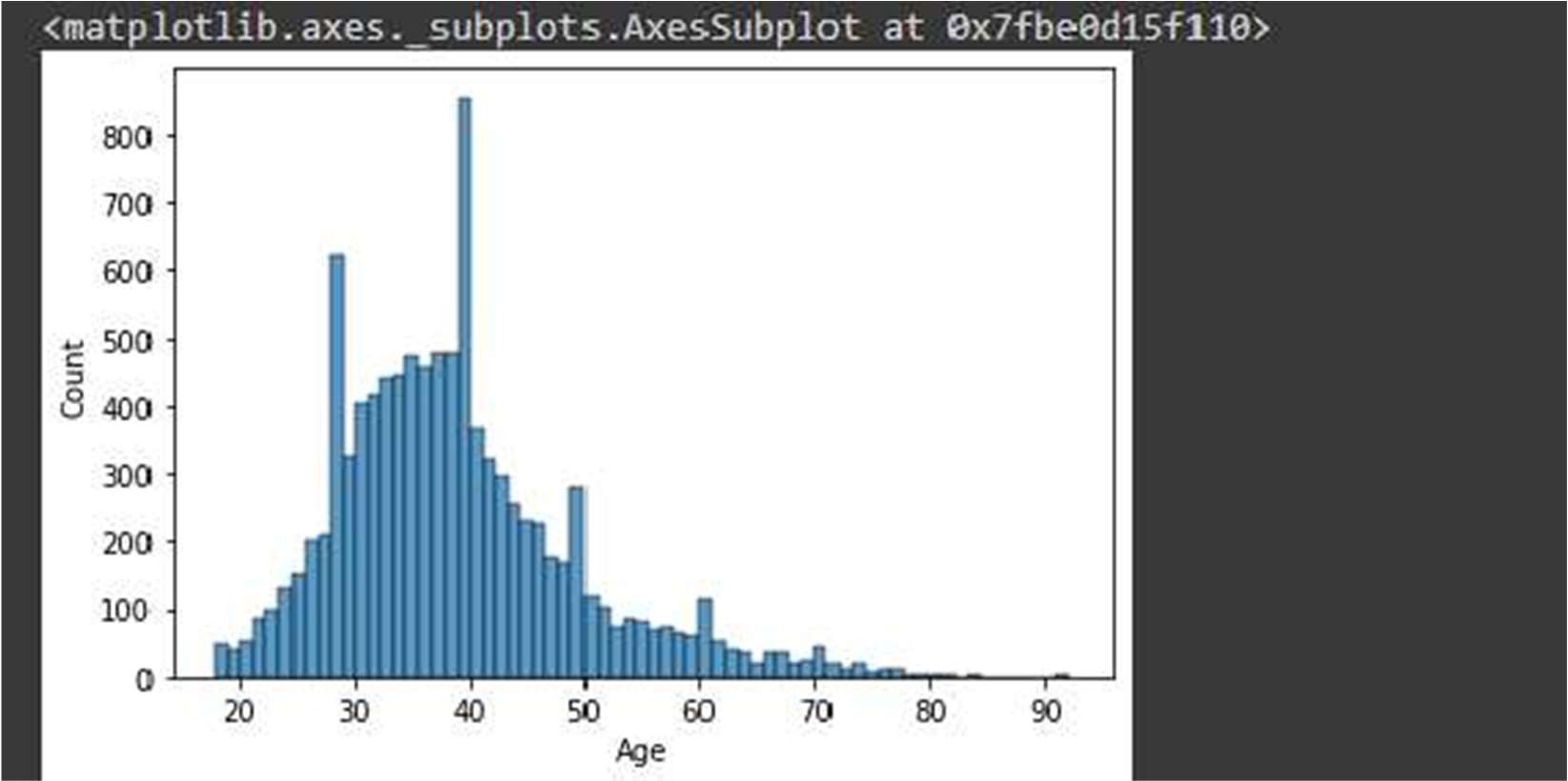
sns.boxplot(x = data['CreditScore'])



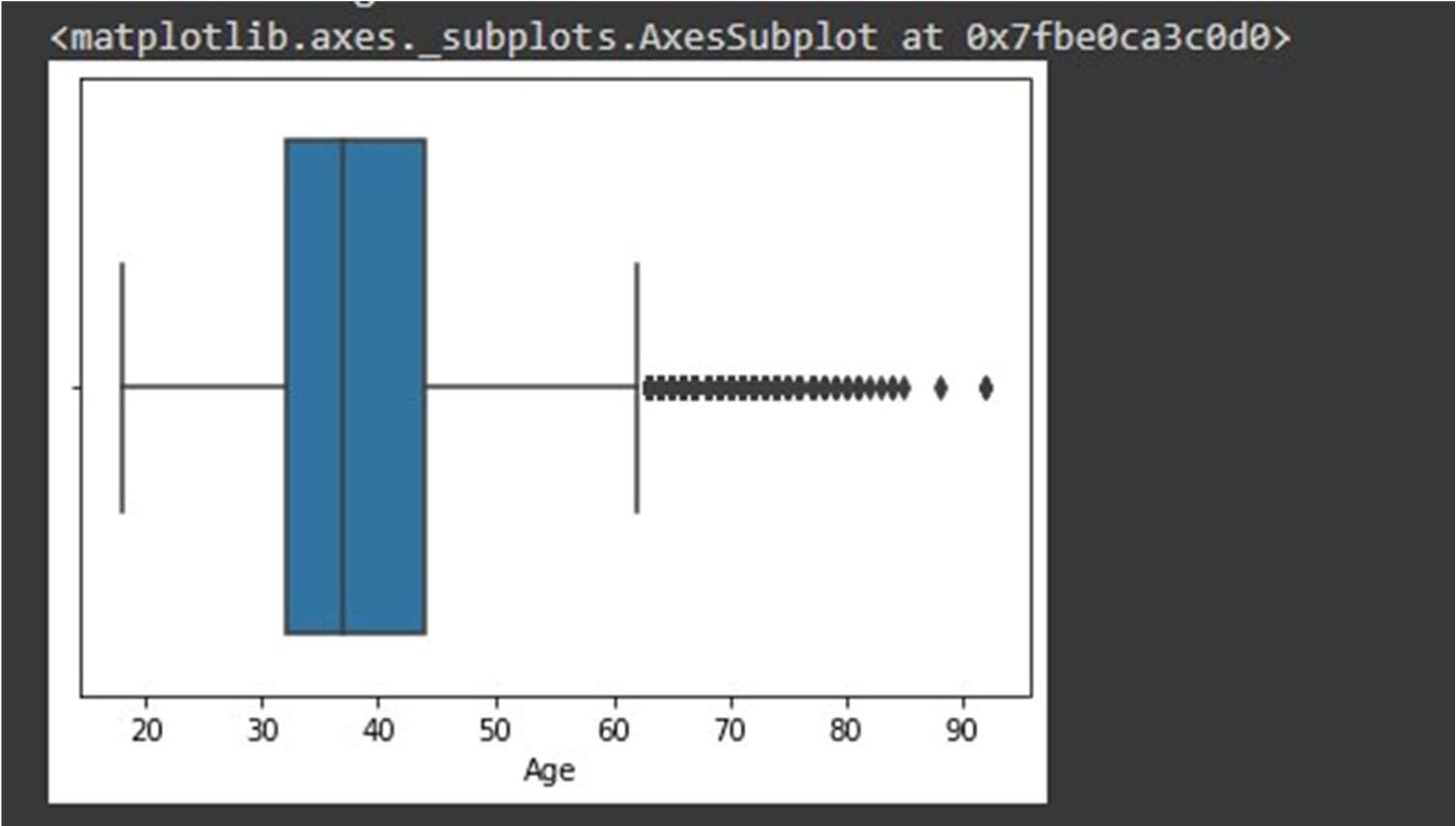
sns.distplot(data['Age'])



sns.histplot(data['Age'])



sns.boxplot(data['Age'])

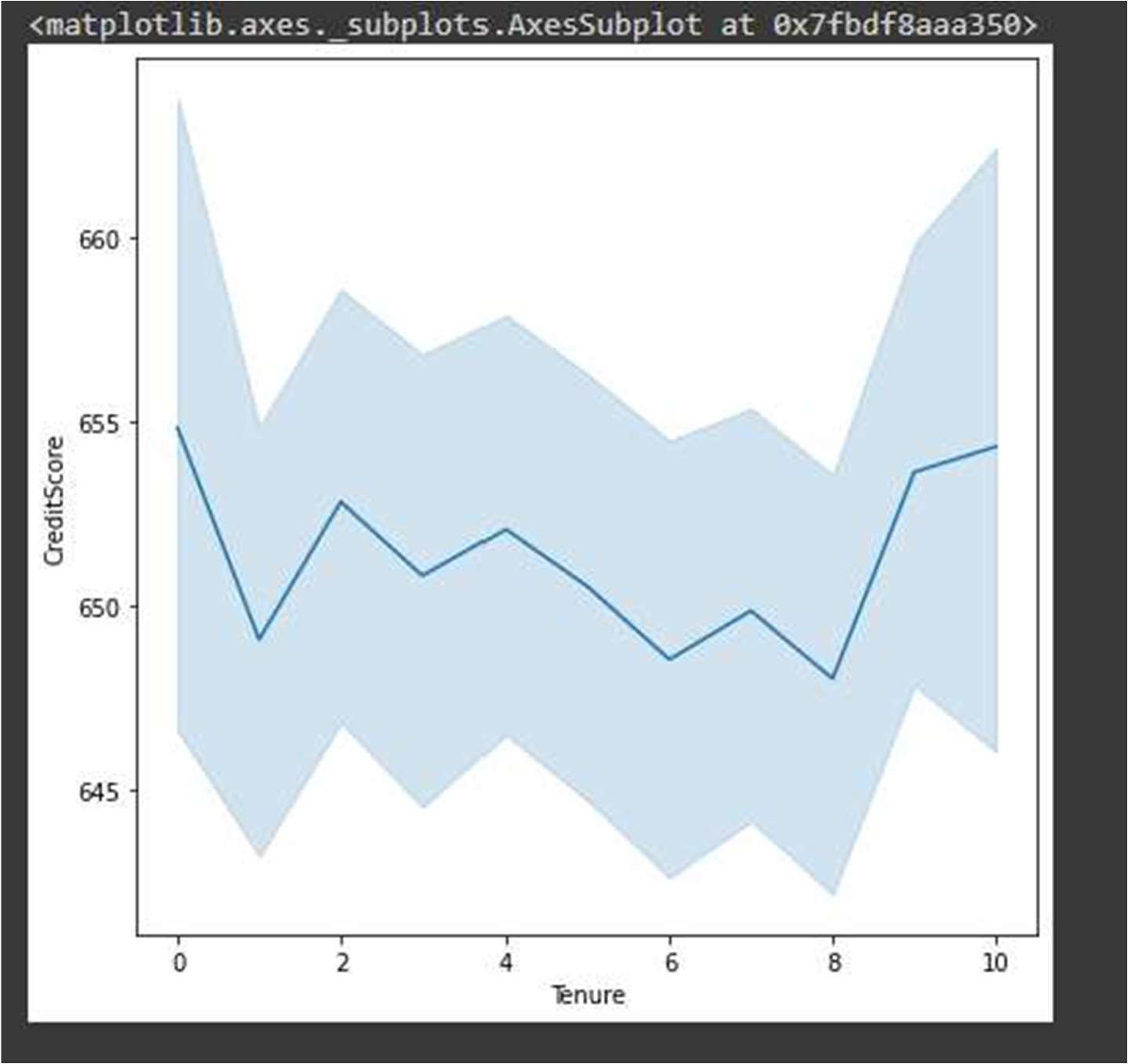


* 1. ***Bivariate Analysis***

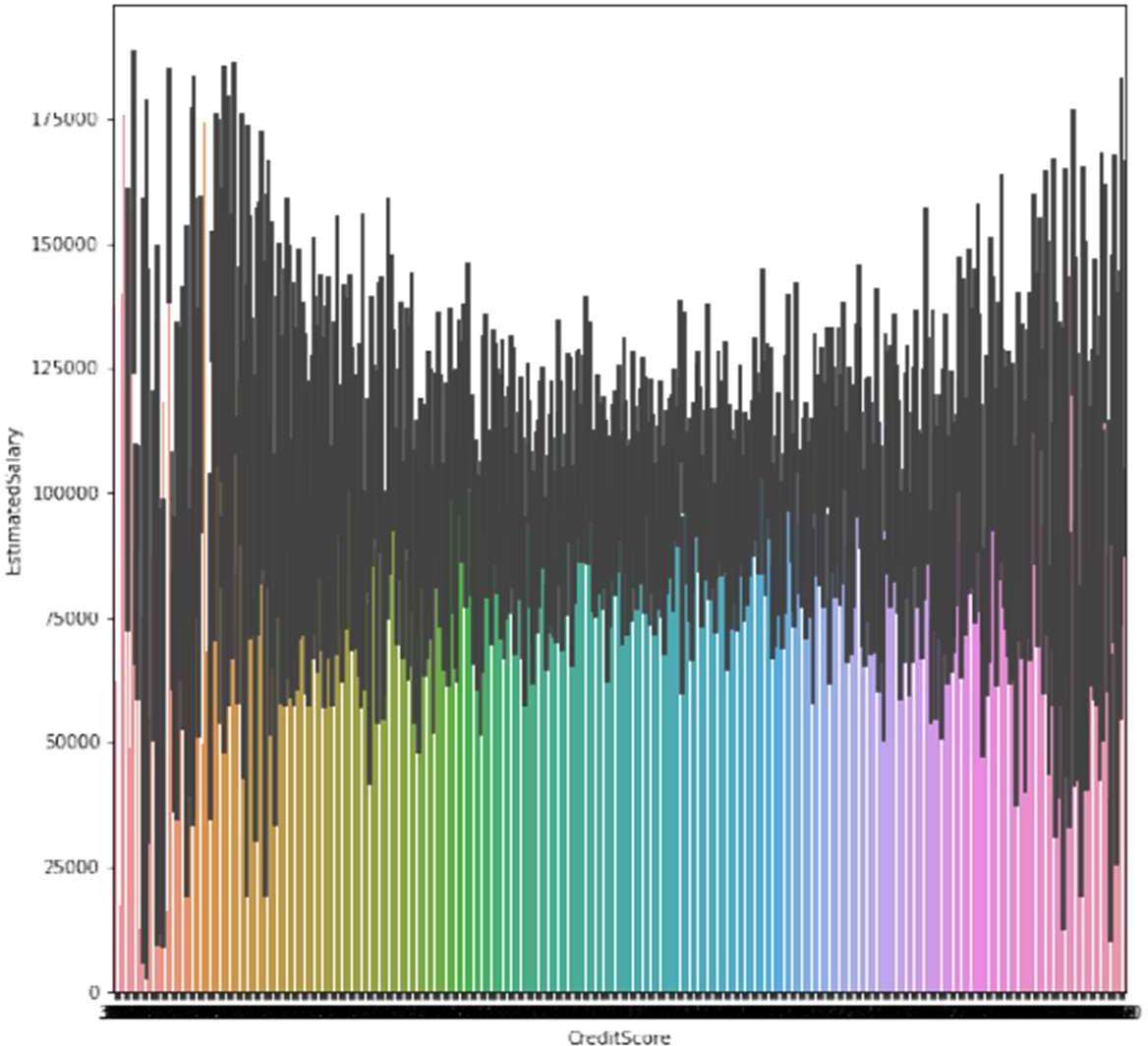
Solution:

plt.figure(figsize=(7,7))

sns.lineplot(data = data, x = 'Tenure', y = 'CreditScore')

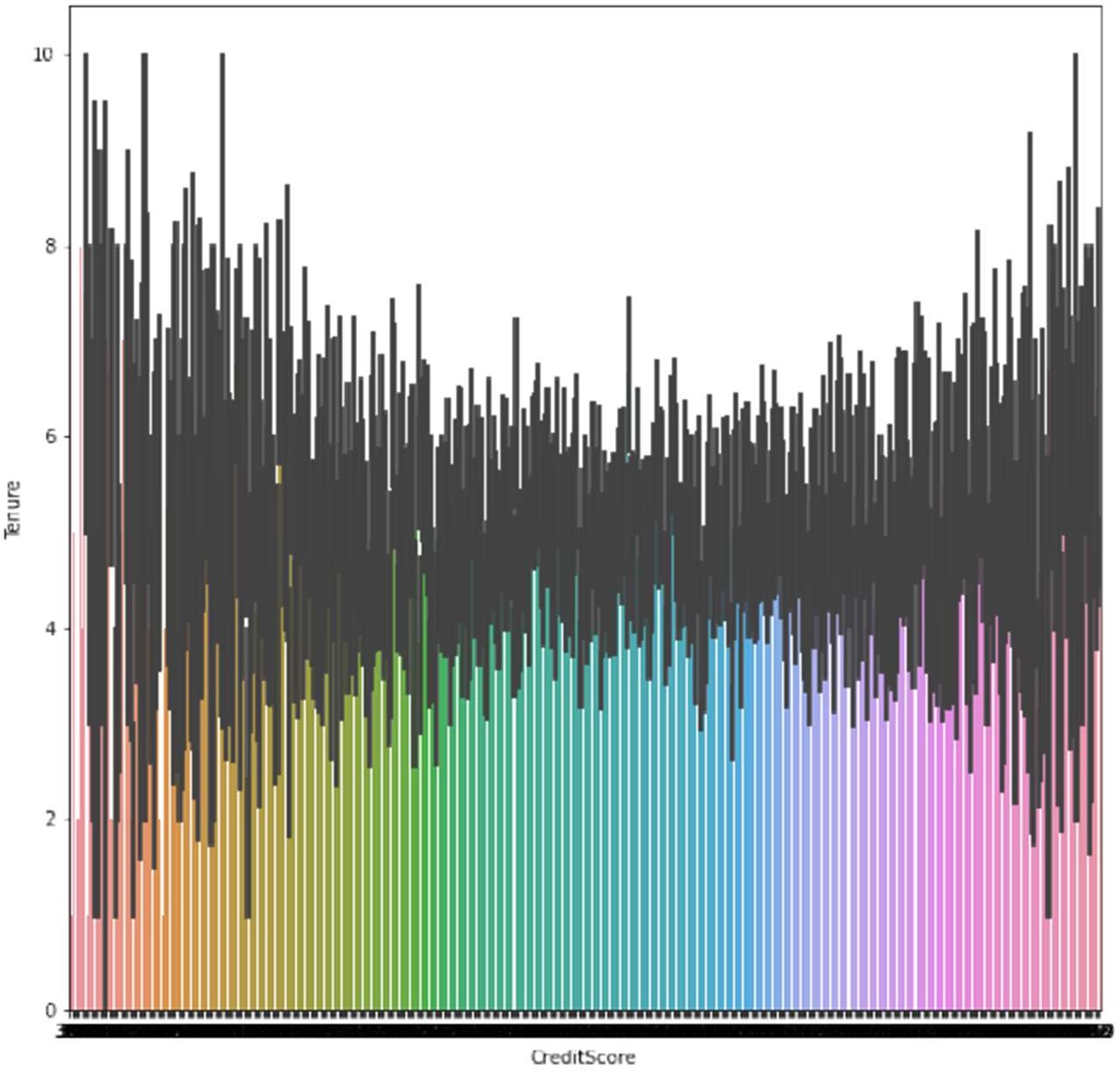


plt.figure(figsize=(10,10))

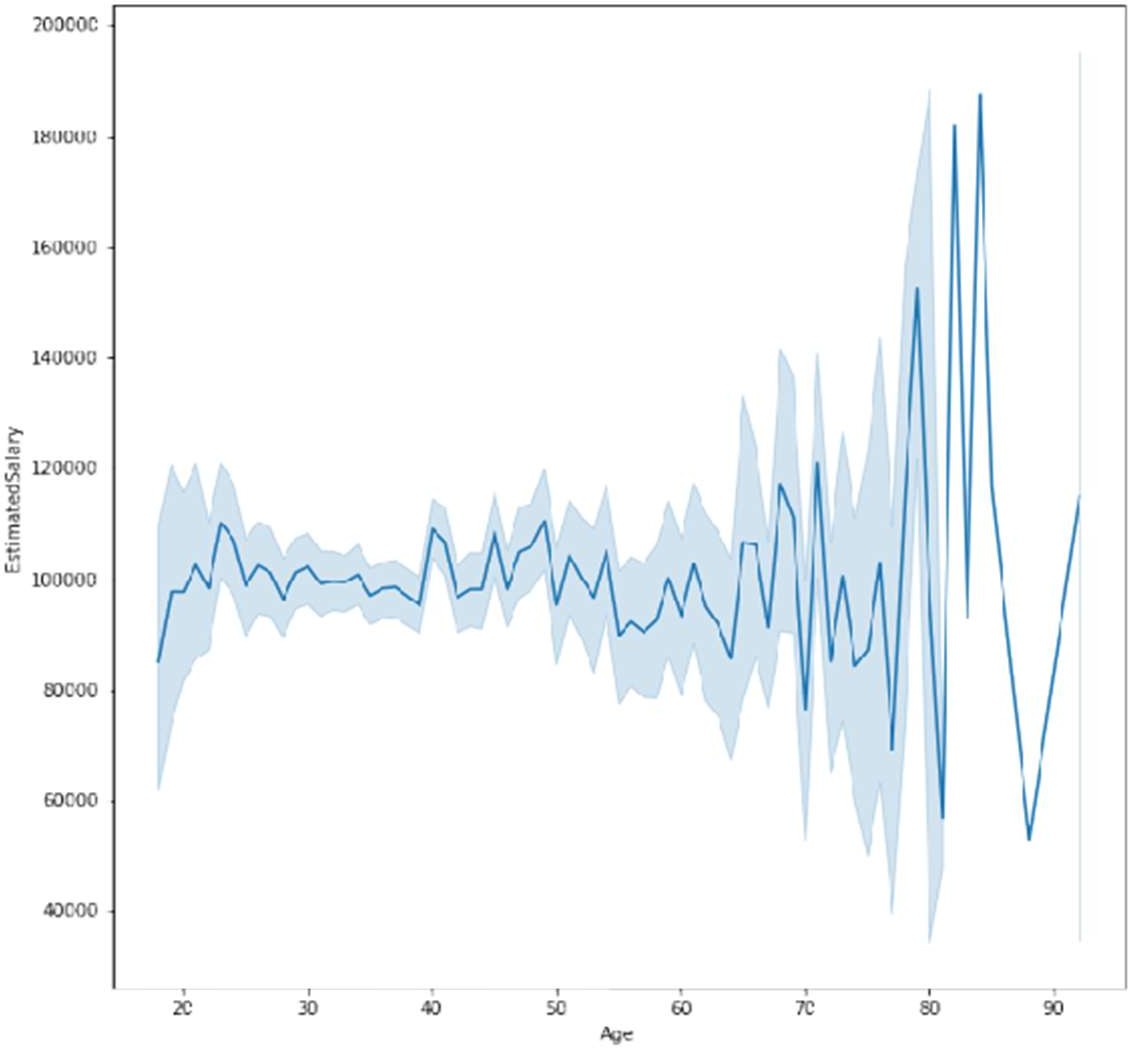
sns.barplot(data = data, x = 'CreditScore', y ='EstimatedSalary')

plt.figure(figsize=(10,10))

sns.barplot(data = data, x = 'CreditScore', y = 'Tenure')

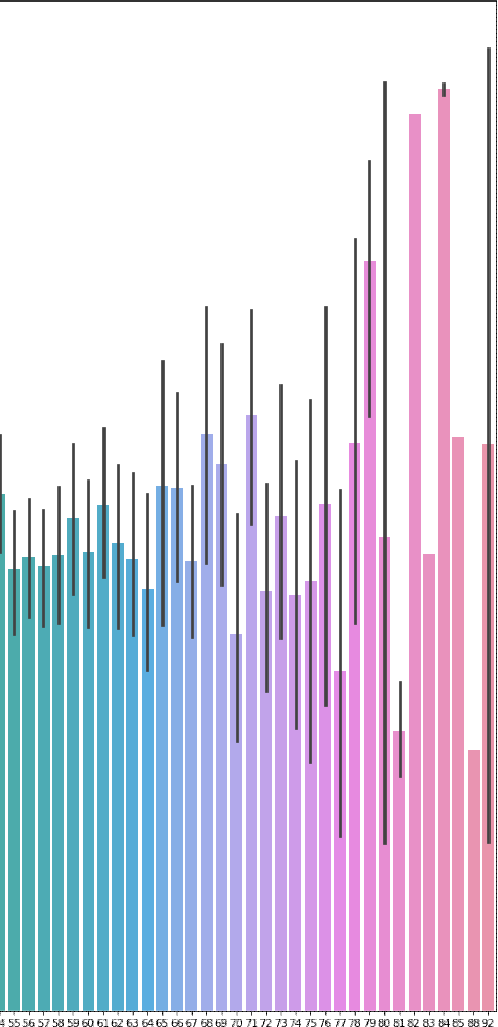
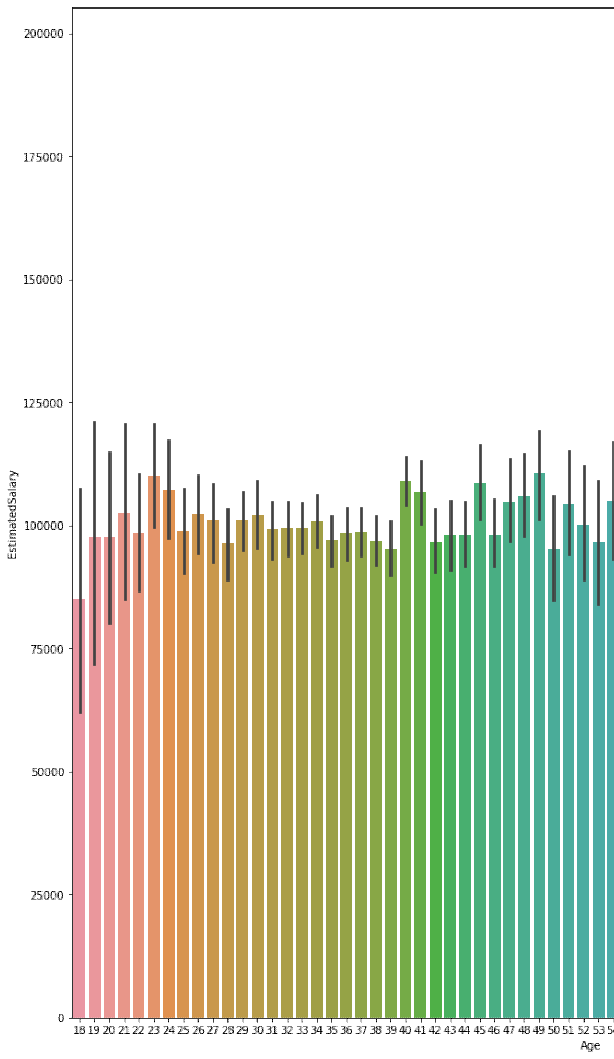


plt.figure(figsize=(10,10))

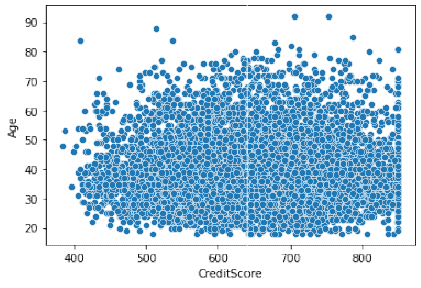
sns.lineplot(data['Age'], data['EstimatedSalary'])

plt.figure(figsize=(17,17))

sns.barplot(data['Age'], data['EstimatedSalary'])



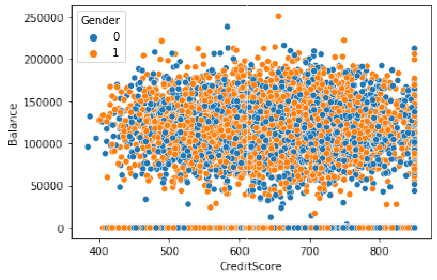
sns.scatterplot(data = data, x = 'CreditScore', y = 'Age')



* 1. ***Multivariate Analysis***

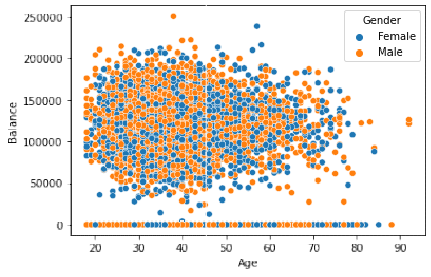
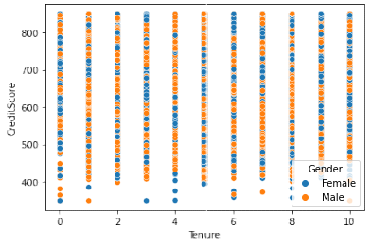
Solution:

sns.scatterplot(data = data, x = 'CreditScore', y = 'Balance', hue = 'Gender')



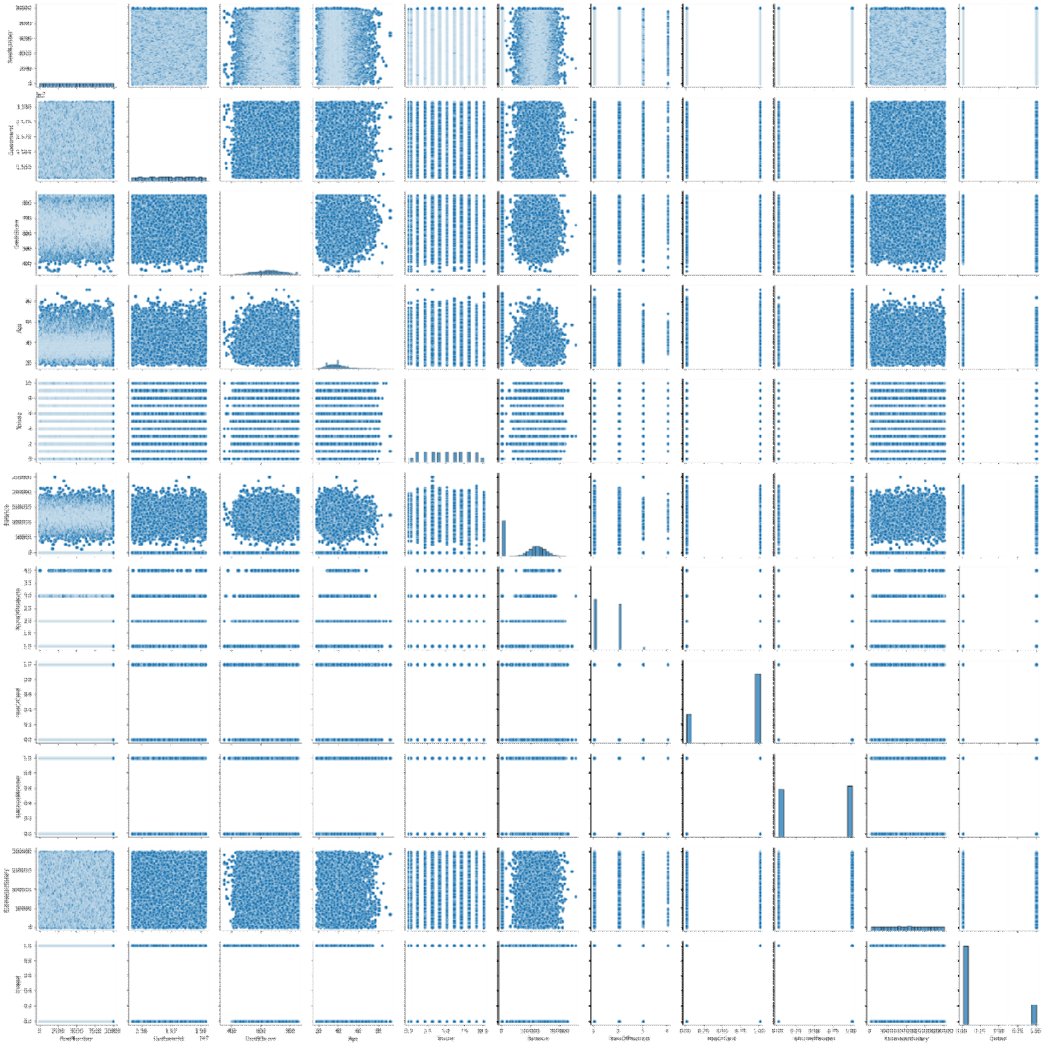
sns.scatterplot(data['Tenure'], data['CreditScore'], hue = data['Gender'])

B

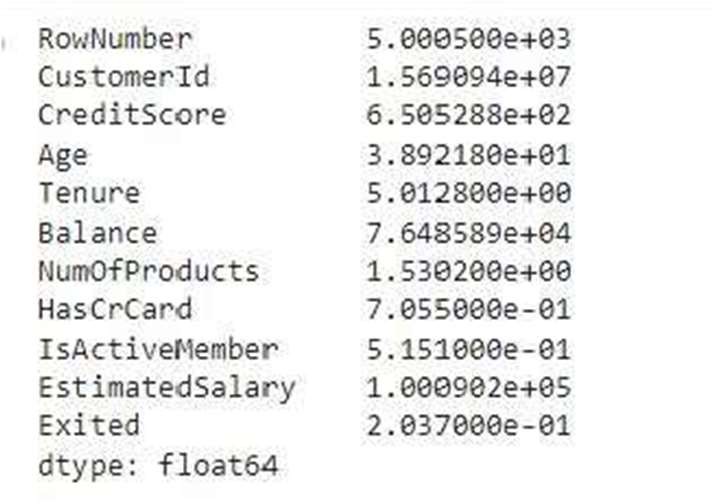


**sns.scatterplot(data['Age'], data[' alance'], hue = data['Gender'])**

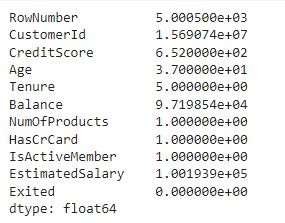
sns.pairplot(data)



**Question-4.**Perform descriptive statistics on the dataset.

Solution: data.mean(numeric\_only = True)

data.median(numeric\_only = True)



data['CreditScore'].mode()

data['EstimatedSalary'].mode()



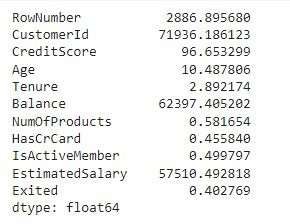
data['HasCrCard'].unique()



data['Tenure'].unique()



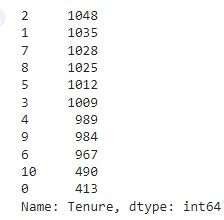
data.std(numeric\_only=True)



data.describe()



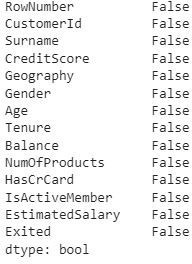
data['Tenure'].value\_counts()



**Question-5.**Handle the Missing values.

Solution:

data.isnull().any()



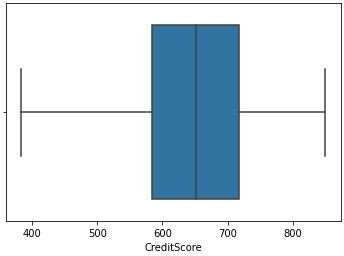
data.isnull().sum()



**Question-6.**Find the outliers and replace the outliers

Solution:

sns.boxplot(data['CreditScore'])#Outlier detection - box plot



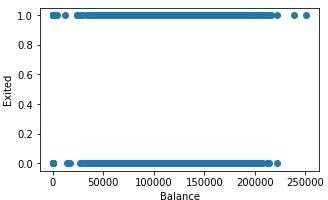
fig, ax = plt.subplots(figsize = (5,3)) #Outlier detection - Scatter plot ax.scatter(data['Balance'], data['Exited'])

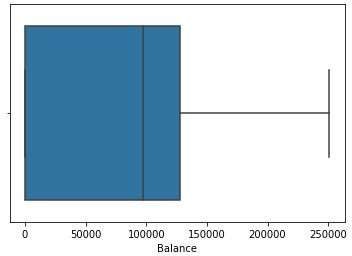
# x-axis label

ax.set\_xlabel('Balance')

# y-axis label ax.set\_ylabel('Exited') plt.show()

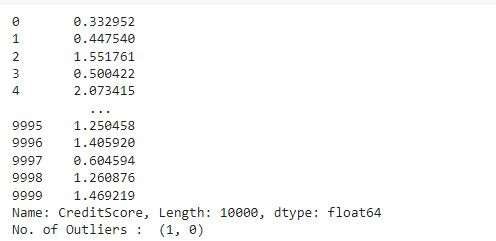
sns.boxplot(x=data['Balance'])



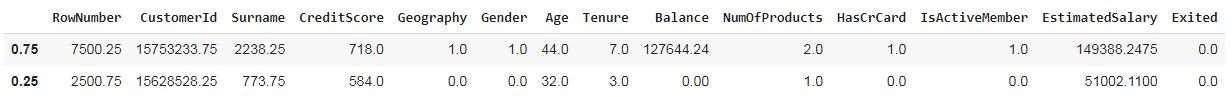


from scipy import stats #Outlier detection – zscore zscore = np.abs(stats.zscore(data['CreditScore'])) print(zscore)

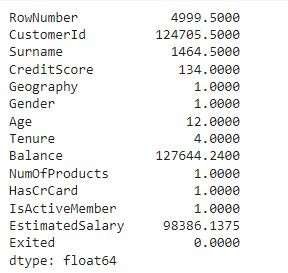
print('No. of Outliers : ', np.shape(np.where(zscore>3)))

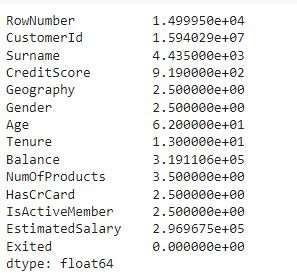


q = data.quantile([0.75,0.25])

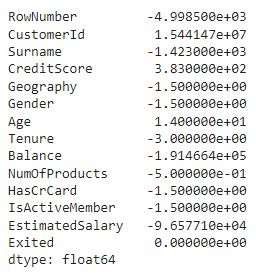
q

iqr = q.iloc[0] - q.iloc[1] iqr



u = q.iloc[0] + (1.5\*iqr) u

l = q.iloc[1] - (1.5\*iqr) l



Q1 = data['EstimatedSalary'].quantile(0.25) #Outlier detection - IQR Q3 = data['EstimatedSalary'].quantile(0.75)

iqr = Q3 - Q1 print(iqr)

upper=Q3 + 1.5 \* iqr lower=Q1 - 1.5 \* iqr

count = np.size(np.where(data['EstimatedSalary'] >upper))

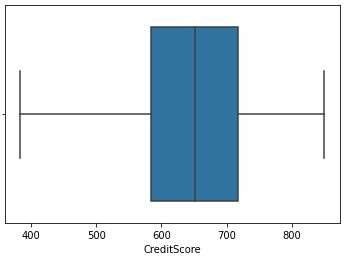
count = count + np.size(np.where(data['EstimatedSalary'] <lower))

print('No. of outliers : ', count)



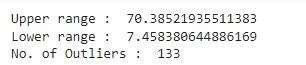
data['CreditScore'] = np.where(np.logical\_or(data['CreditScore']>900, data['CreditScore']<383), 65

0, data['CreditScore']) sns.boxplot(data['CreditScore'])



upper = data.Age.mean() + (3 \* data.Age.std()) #Outlier detection - 3 sigma lower = data.Age.mean() - (3 \* data.Age.std())

columns = data[ ( data['Age'] > upper ) | ( data['Age']<lower ) ] print('Upper range : ', upper)

print('Lower range : ', lower) print('No. of Outliers : ', len(columns))

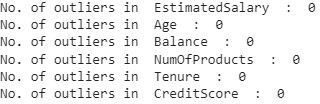
columns = ['EstimatedSalary', 'Age', 'Balance', 'NumOfProducts', 'Tenure', 'CreditScore'] #After outl ier removal

for i in columns:

Q1 = data[i].quantile(0.25) Q3 = data[i].quantile(0.75) iqr = Q3 - Q1

upper=Q3 + 1.5 \* iqr lower=Q1 - 1.5 \* iqr

count = np.size(np.where(data[i] >upper))

count = count + np.size(np.where(data[i] <lower)) print('No. of outliers in ', i, ' : ', count)

**Question-7.** Check for Categorical columns and perform encoding

Solution:

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

le = LabelEncoder() oneh = OneHotEncoder()

data['Surname'] = le.fit\_transform(data['Surname']) data['Gender'] = le.fit\_transform(data['Gender']) data['Geography'] = le.fit\_transform(data['Geography']) data.head()



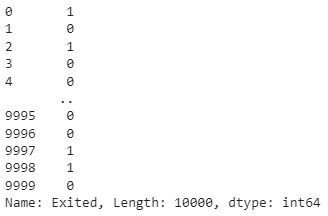
**Question-8.**Split the data into dependent and independent variables split the data in X and Y

Solution:

x # independent values ( inputs) x = data.iloc[:, 0:13]



y # dependent values (output) y = data['Exited']

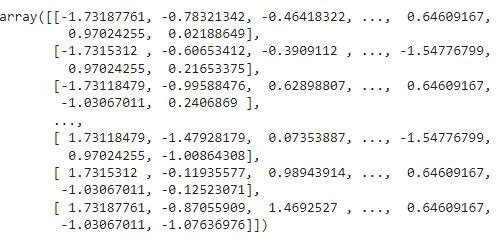


**Question-9.**Scale the independent variables

Solution:

from sklearn.preprocessing import StandardScaler, MinMaxScaler

sc = StandardScaler() x\_scaled = sc.fit\_transform(x) x\_scaled



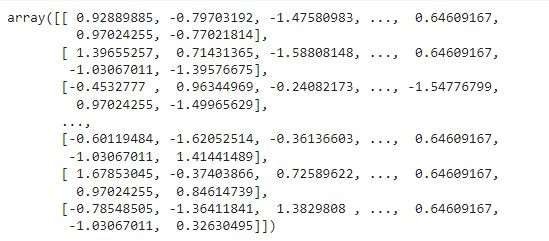
**Question-10.**Split x and y into Training and Testing

Solution:

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_scaled, y, test\_size = 0.3, random\_state = 0)

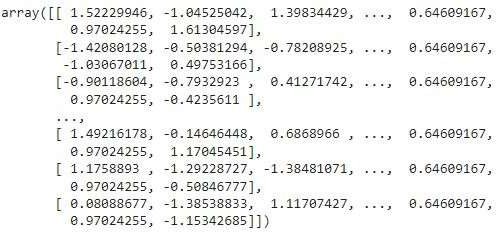
x\_train



x\_train.shape



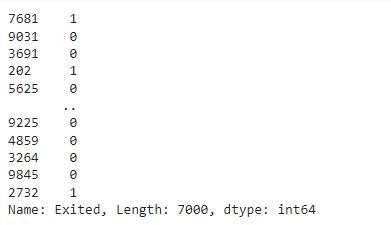
x\_test



x\_test.shape



y\_train



y\_test

