**Assignment -2**

|  |  |
| --- | --- |
| Team ID | PNT2022TMID03497 |
| Maximum Marks | 2 Marks |

**Question-1:**

1. Download the dataset: Dataset

**Solution:**

# Downloaded succesfully

**Question-2:**

1. Load the dataset.

**Solution:**

import

pandas

as

pd

import

numpy

as

np

file

=

pd.read\_csv

(

"/content/Churn\_Modelling (1).csv"

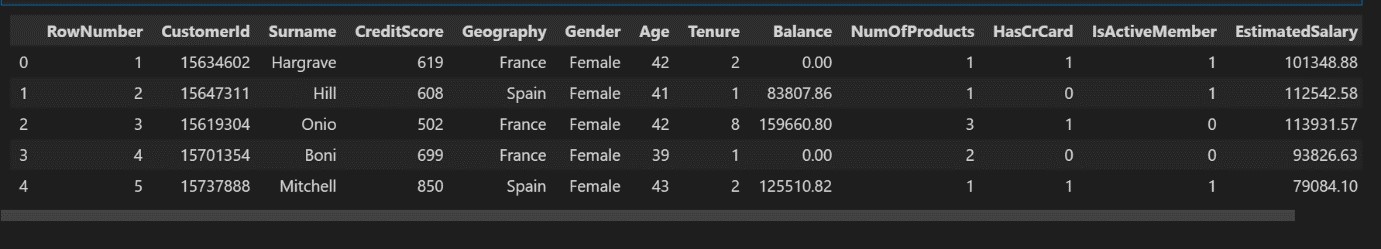
)

df=pd.DataFrame(

file

)

df.head()

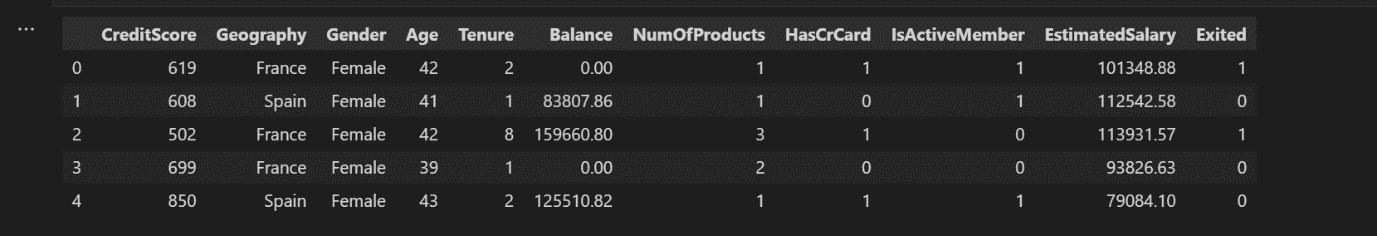


df['HasCrCard'] = df['HasCrCard'].astype('category')

df['IsActiveMember'] = df['IsActiveMember'].astype('category') df['Exited'] = df['Exited'].astype('category')

df = df.drop(columns=['RowNumber', 'CustomerId', 'Surname'])

df.head()



**Question 3:**

1. Perform Below Visualizations:

Univariate Analysis, Bi - Variate Analysis, Multi - Variate Analysis

**Solution:**

import seaborn as sns density = df['Exited'].value\_counts(normalize=True).reset\_index() sns.barplot(data=density, x='index', y='Exited', ); density

|  |  |  |
| --- | --- | --- |
|  | index | Exited |
| 0 | 0 | 0.7963 |
| 1 | 1 | 0.2037 |



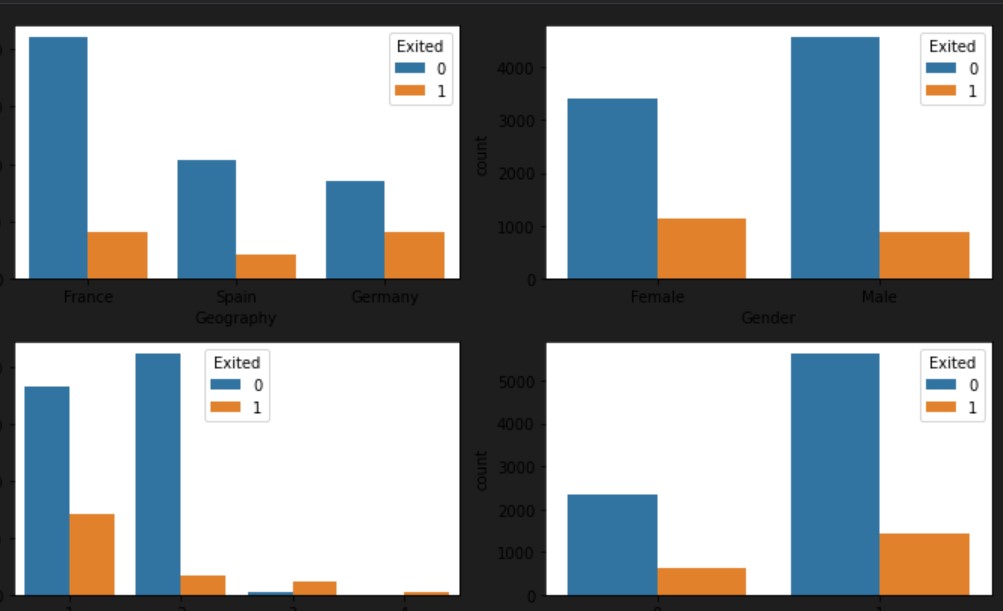
# The data is significantly imbalanced

import matplotlib.pyplot as plt

categorical = df.drop(columns=['CreditScore', 'Age', 'Tenure', 'Balance',

'EstimatedSalary'])

|  |
| --- |
| rows = int(np.ceil(categorical.shape[1] / 2)) - 1  # create sub-plots anf title them fig, axes = plt.subplots(nrows=rows, ncols=2, figsize=(10,6)) axes = axes.flatten()  for row in range(rows):  cols = min(2, categorical.shape[1] - row\*2) for col in range(cols):  col\_name = categorical.columns[2 \* row + col] ax = axes[row\*2 + col]  sns.countplot(data=categorical, x=col\_name, hue="Exited", ax=ax);  plt.tight\_layout() |



**Question 4:**

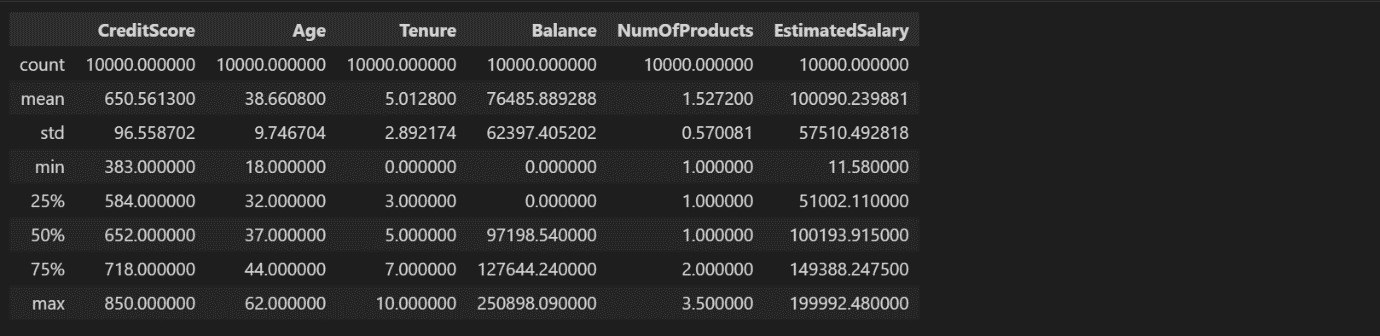
1. Perform descriptive statistics on the dataset.

**Solution:**

df.info()



df.describe()



**Question 5:**

1. Handle the Missing values.

**Solution:**

df.isna().sum()



There is no missing values in dataset

|  |
| --- |
| for i in df: if df[i].dtype=='object' or df[i].dtype=='category':  print("unique of "+i+" is "+str(len(set(df[i])))+" they are "+str(set(df[i]))) |

unique of Geography is 3 they are {‘France’,’Germany’,’Spain’} unique of Gender is 2 they are {‘Male’,’Female’} unique of Has CrCard is 2 they are {0,1} unique of Is Active Member is 2 they are {0,1} unique of Exited is 2 they are {0,1}

**Question 6:**

1. Find the outliers and replace the outliers.

**Solution:**

Checking for outliers

def box\_scatter(data, x, y):

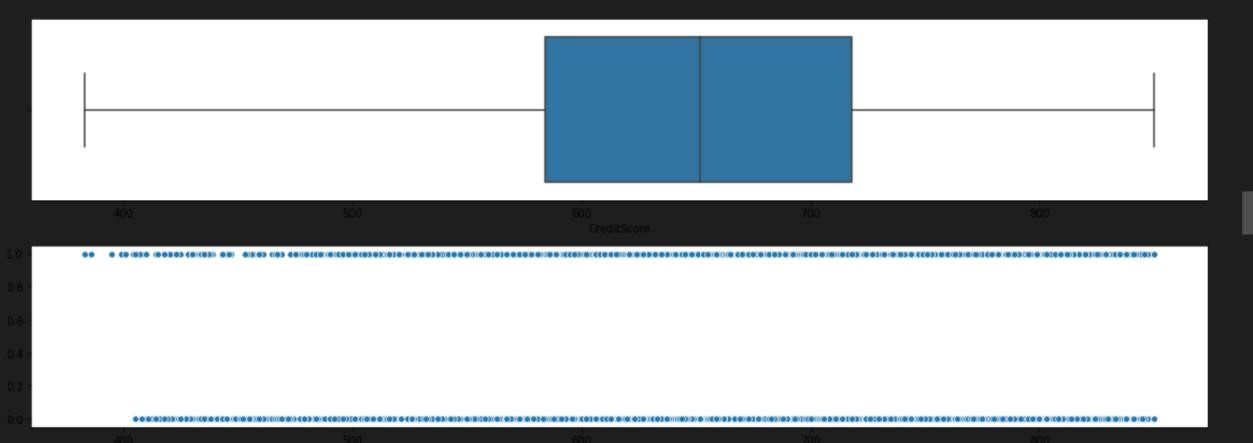
fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(16,6)) sns.boxplot(data=data, x=x, ax=ax1)

sns.scatterplot(data=data, x=x,y=y,ax=ax2)

box\_scatter(df,'CreditScore','Exited'); plt.tight\_layout()

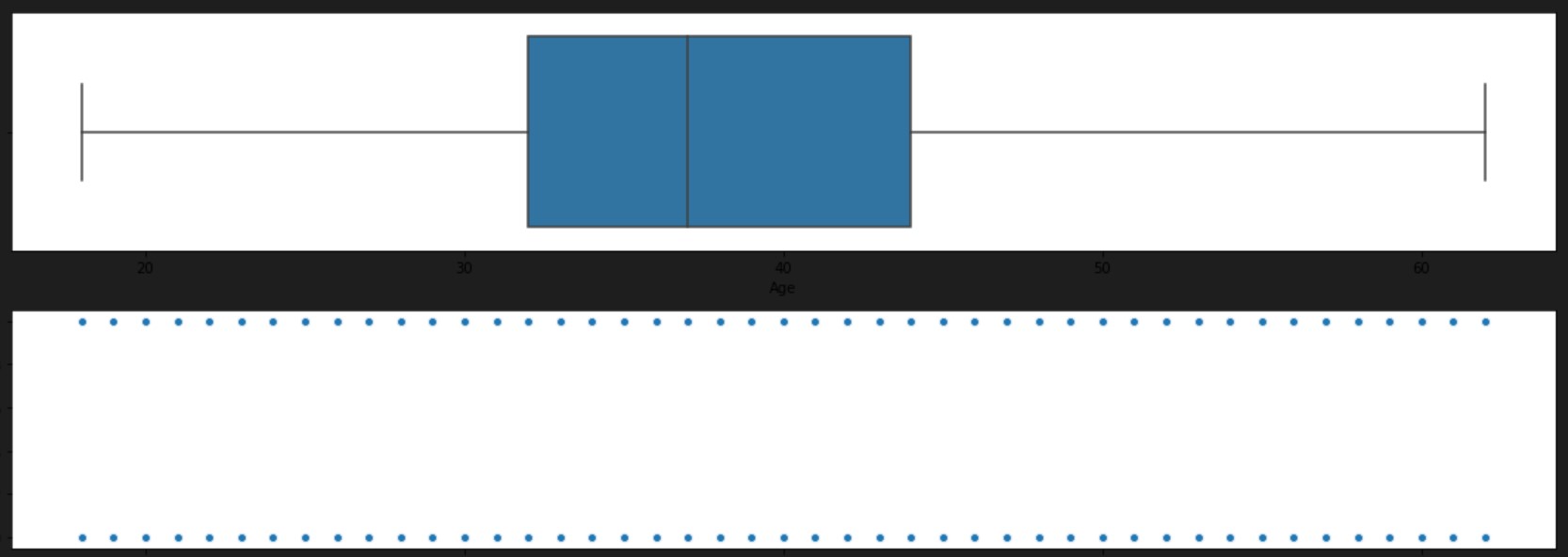
print(f"# of Bivariate Outliers: {len(df.loc[df['CreditScore'] < 400])}")

# # of bivariate Outliers:19



box\_scatter(df,'Age','Exited'); plt.tight\_layout() print(f"# of Bivariate Outliers: {len(df.loc[df['Age'] > 87])}")

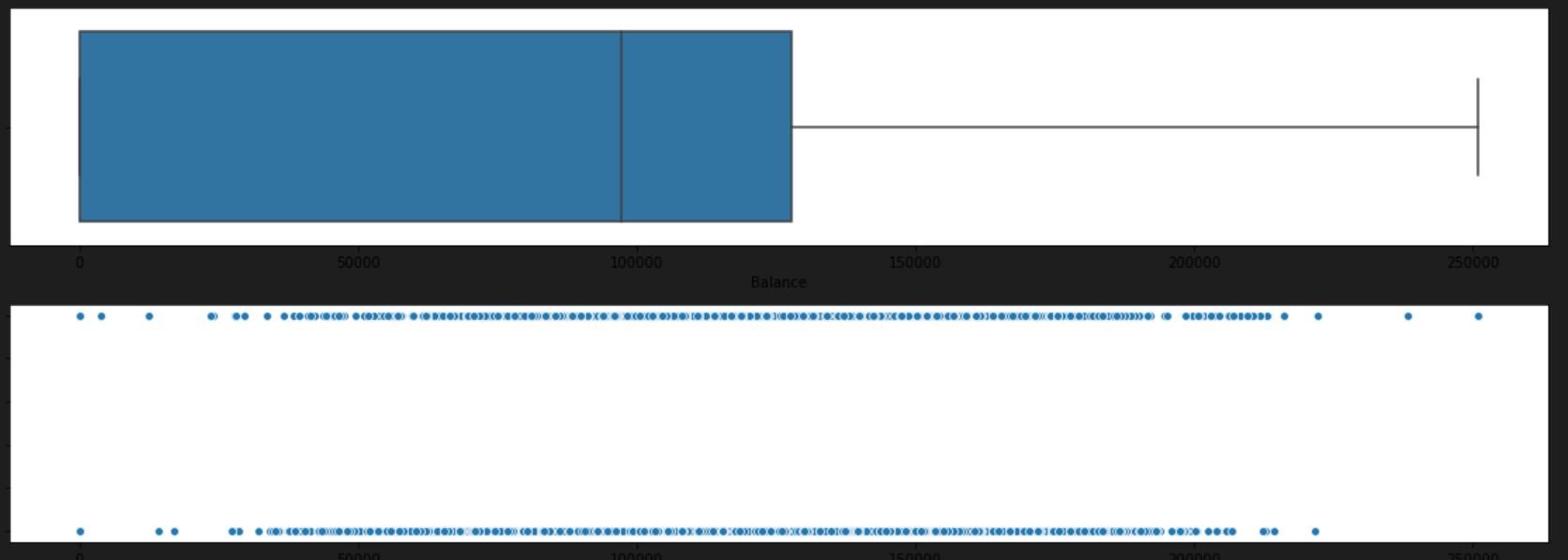
# # of bivariate Outliers:0



box\_scatter(df,'Balance','Exited'); plt.tight\_layout()

print(f"# of Bivariate Outliers: {len(df.loc[df['Balance'] > 220000])}")

# # of bivariate Outliers:4



box\_scatter(df,'EstimatedSalary','Exited'); plt.tight\_layout()



# Removing Outliers

|  |
| --- |
| for i in df: if df[i].dtype=='int64' or df[i].dtypes=='float64':  q1=df[i].quantile(0.25) q3=df[i].quantile(0.75) iqr=q3-q1 upper=q3+1.5\*iqr lower=q1-1.5\*iqr df[i]=np.where(df[i] >upper, upper, df[i]) df[i]=np.where(df[i] <lower, lower, df[i]) |

# After removing outliers,boxplot will be like

box\_scatter(df,'CreditScore','Exited'); plt.tight\_layout()

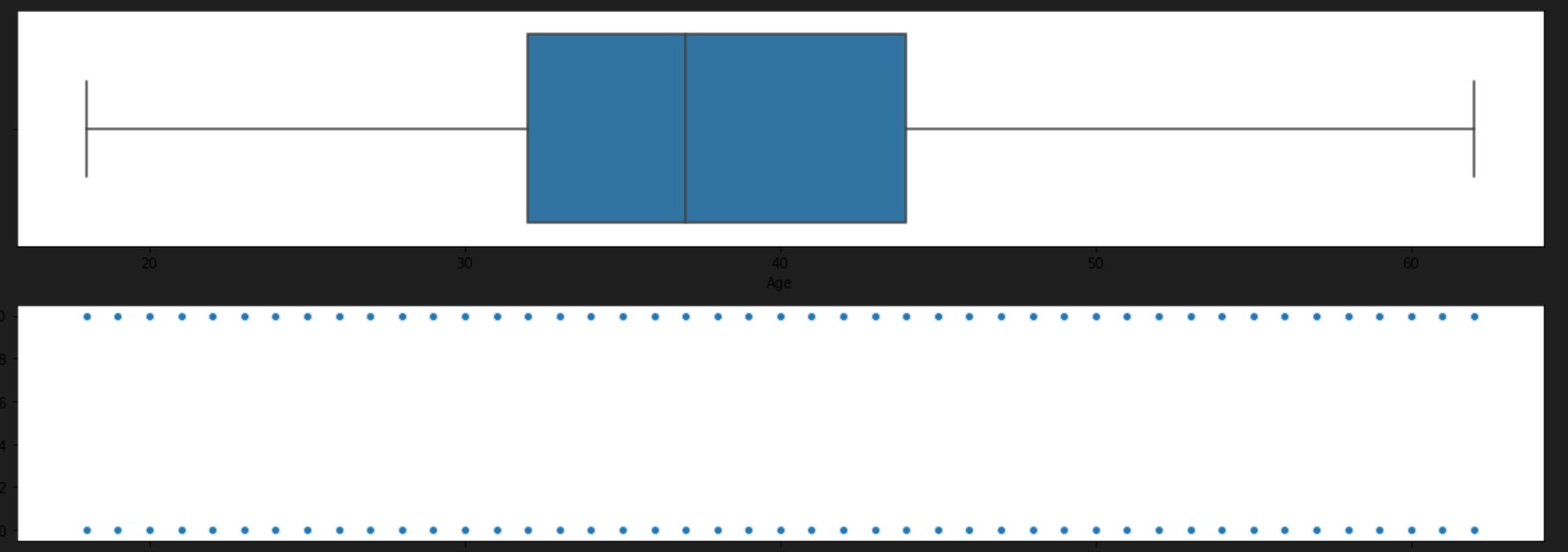
print(f"# of Bivariate Outliers: {len(df.loc[df['CreditScore'] < 400])}")

# # of bivariate Outliers:19



box\_scatter(df,'Age','Exited'); plt.tight\_layout() print(f"# of Bivariate Outliers: {len(df.loc[df['Age'] > 87])}")

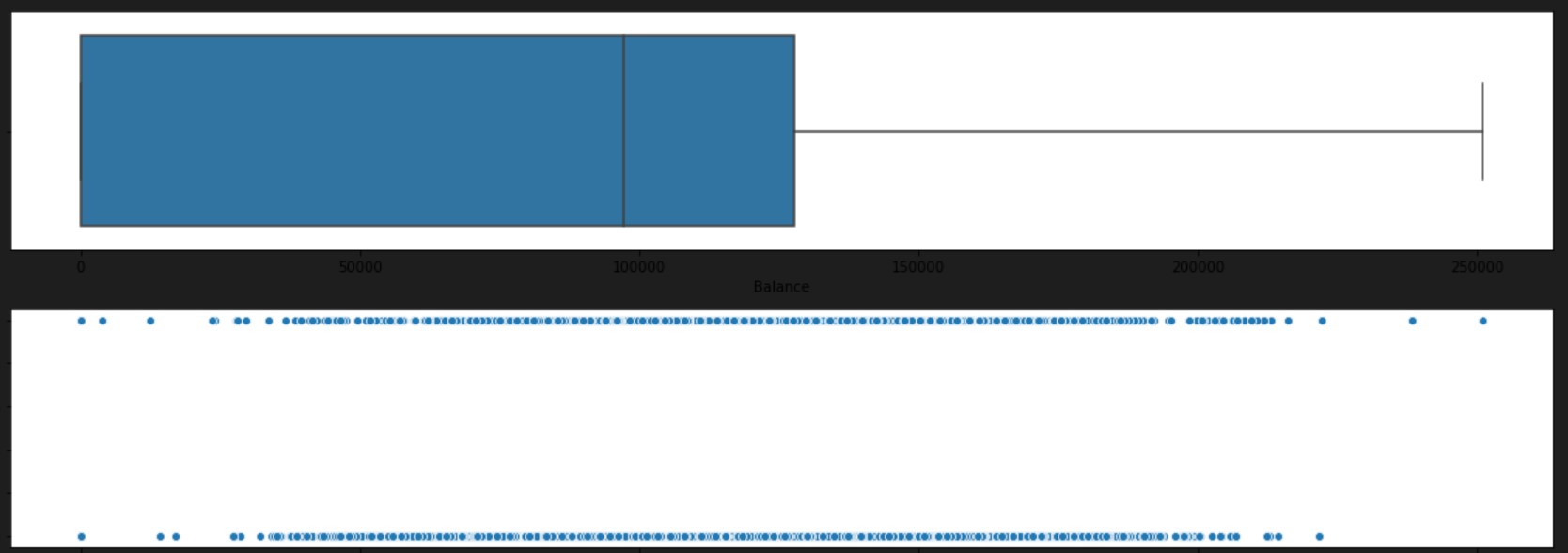
# # of bivariate Outliers:0



box\_scatter(df,'Balance','Exited'); plt.tight\_layout()

print(f"# of Bivariate Outliers: {len(df.loc[df['Balance'] > 220000])}")

# # of bivariate Outliers:4



**Question 7:**

1. Check for Categorical columns and perform encoding.

**Solution:**

from sklearn.preprocessing import LabelEncoder encoder=LabelEncoder() for i in df: if df[i].dtype=='object' or df[i].dtype=='category':

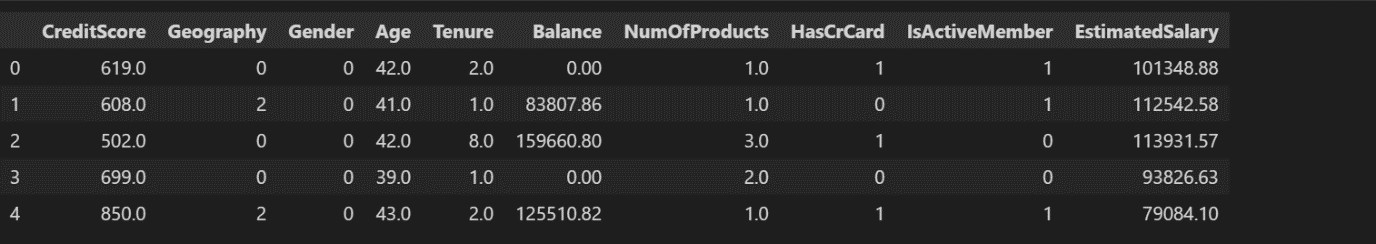
df[i]=encoder.fit\_transform(df[i])

**Question 8:**

1. Split the data into dependent and independent variables.

**Solution:**

x=df.iloc[:,:-1] x.head()



y=df.iloc[:,-1] y.head()



**Question 9:**

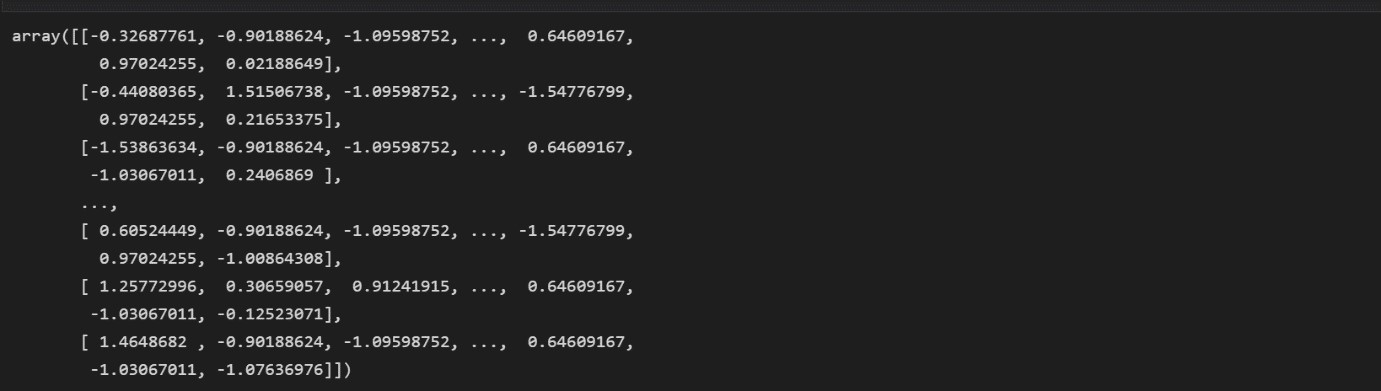
1. Scale the independent variables

**Solution:**

from sklearn.preprocessing import StandardScaler scaler=StandardScaler()

x=scaler.fit\_transform(x)

x



**Question 10:**

1. Split the data 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,y,test\_size=0.33)

