**Assignment - 2**

**Data Visualization and Data Preprocessing**

|  |  |
| --- | --- |
| Assignment Date | 21.09.2022 |
| Student Name | Aravind J |
| Student Roll Number | 2019115017 |
| Maximum Marks | 2 Marks |

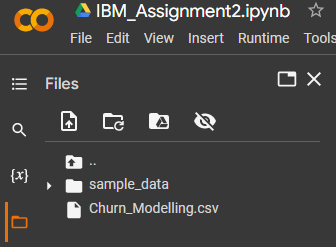
**Task-1:**

Download the dataset.



**Task-2:**

Load the dataset.



**Solution:**

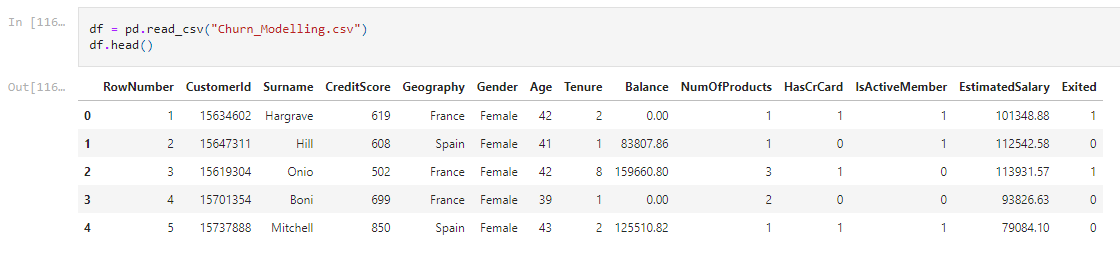
df = pd.read\_csv("Churn\_Modelling.csv")

df.head()

#-----------------#

#-----------------#

**Screenshot:**



**Task-3:**

Perform Univariate, Bivariate, Multivariate analysis.

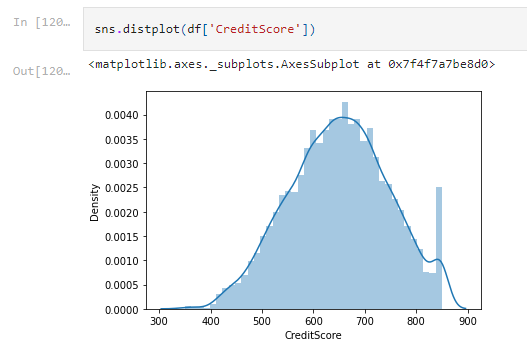
**Univariate analysis solution:**

sns.distplot(df['CreditScore'])

#-----------------#

#-----------------#

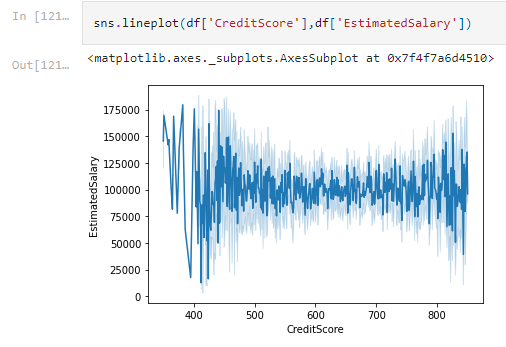
**Screenshot:**



**Bivariate analysis solution:**

sns.lineplot(df['CreditScore'],df['EstimatedSalary'])

**Screenshot:**



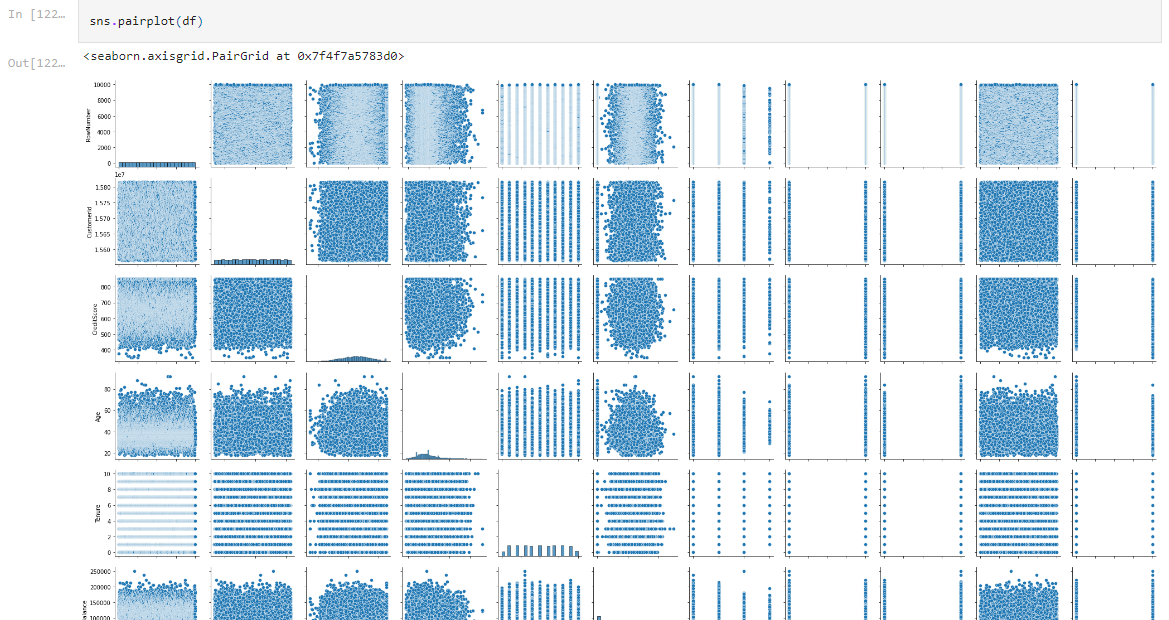
**Multivariate analysis solution:**

sns.pairplot(df)

#-----------------#

#-----------------#

**Screenshot:**



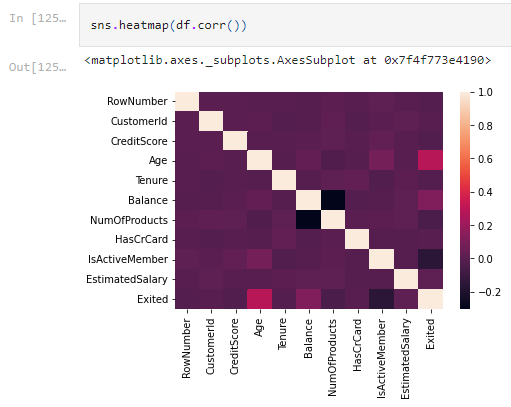
**Heatmap solution:**

sns.heatmap(df.corr())

#-----------------#

#-----------------#

**Screenshot:**



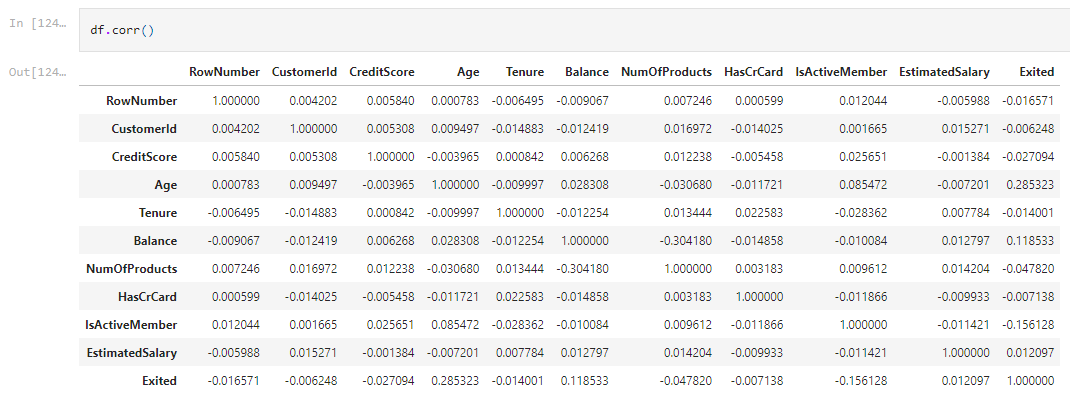
**Correlation solution:**

df.corr()

#-----------------#

#-----------------#

**Screenshot:**



**Task-4:**

Perform descriptive statistics on the dataset

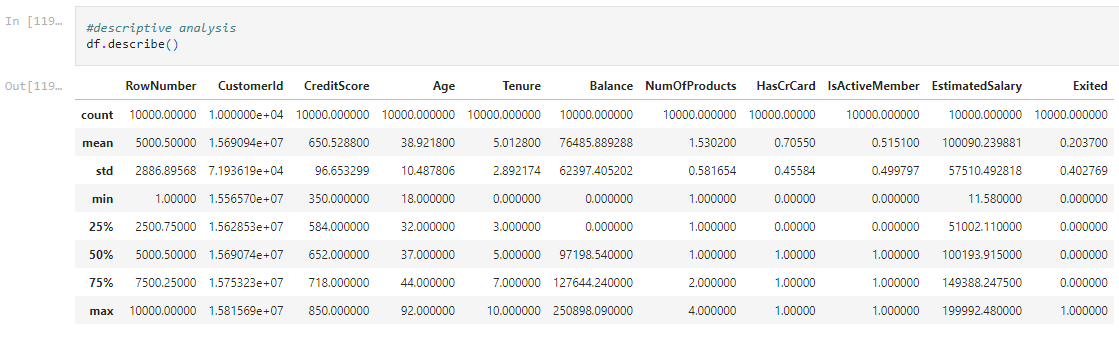
**Solution:**

df.describe()

#-----------------#

#-----------------#

**Screenshot:**



**Task-5:**

Handle the Missing values.

**Solution:**

#check for null values

df.isnull().any()

df['CreditScore'].fillna(df['CreditScore'].mean(),inplace=True)

df['Age'].fillna(df['Age'].median(),inplace=True)

df['Tenure'].fillna(df['Tenure'].median(),inplace=True)

df['Balance'].fillna(df['Balance'].median(),inplace=True)

df['CreditScore'].fillna(df['CreditScore'].median(),inplace=True)

df['NumOfProducts'].fillna(df['NumOfProducts'].median(),inplace=True)

df['HasCrCard'].fillna(0,inplace=True)

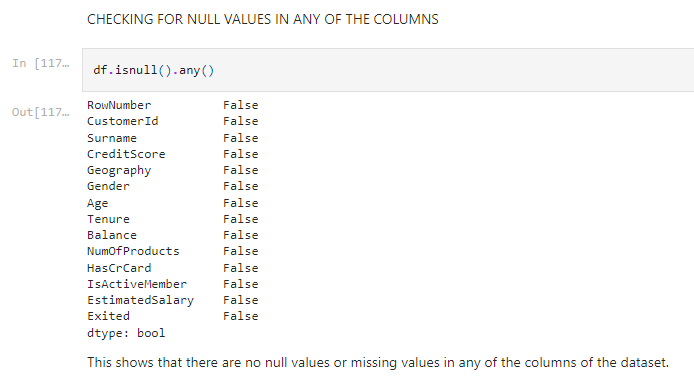
df['IsActiveMember'].fillna(0, inplace=True)

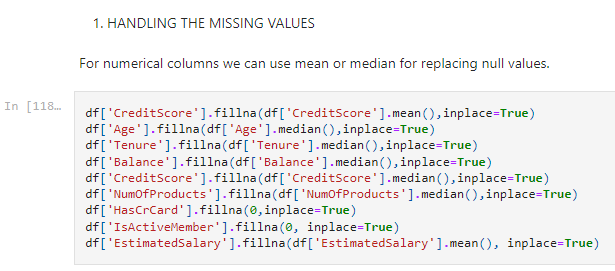
df['EstimatedSalary'].fillna(df['EstimatedSalary'].mean(), inplace=True)

#-----------------#

#-----------------#

**Screenshot:**





**Task-6:**

Find the outliers and replace the outliers

**Solution:**

#detecting outliers

sns.boxplot(df['Age'])

#replacing outliers

Q1= df['Age'].quantile(0.25)

Q3=df['Age'].quantile(0.75)

IQR=Q3-Q1

upper\_limit =Q3 + 1.5\*IQR

lower\_limit =Q1 - 1.5\*IQR

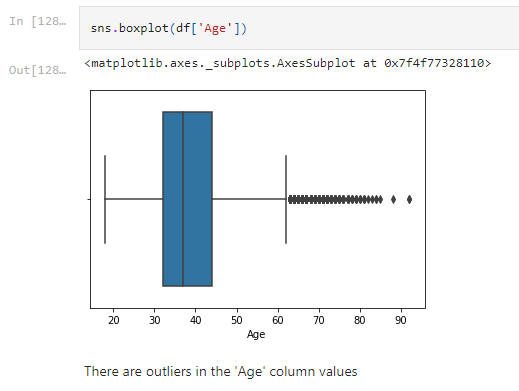
# df=df[df['Age']<upper\_limit]

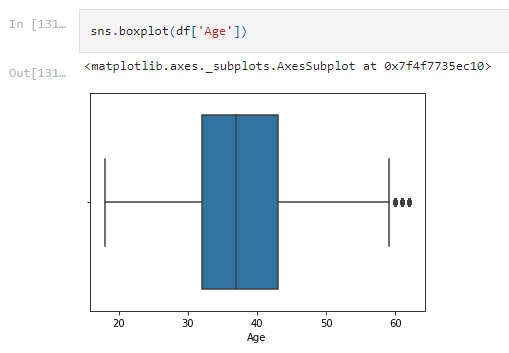
df['Age'] = np.where(df['Age']>upper\_limit,37,df['Age']) #median 37

#-----------------#

#-----------------#

**Screenshot:**





**Task-7:**

Check for Categorical columns and perform encoding.

**Solution:**

**#check for categorical columns**

textualColumns = [x for x in df.columns if df[x].dtype == np.dtype('O')]

print(textualColumns)

#perform label encoding to gender column

from sklearn.preprocessing import LabelEncoder

lbEnc=LabelEncoder()

df['Gender'] = lbEnc.fit\_transform(df['Gender'])

#perform one hot encoding to Geography column

df\_main=pd.get\_dummies(df,columns=['Geography'])

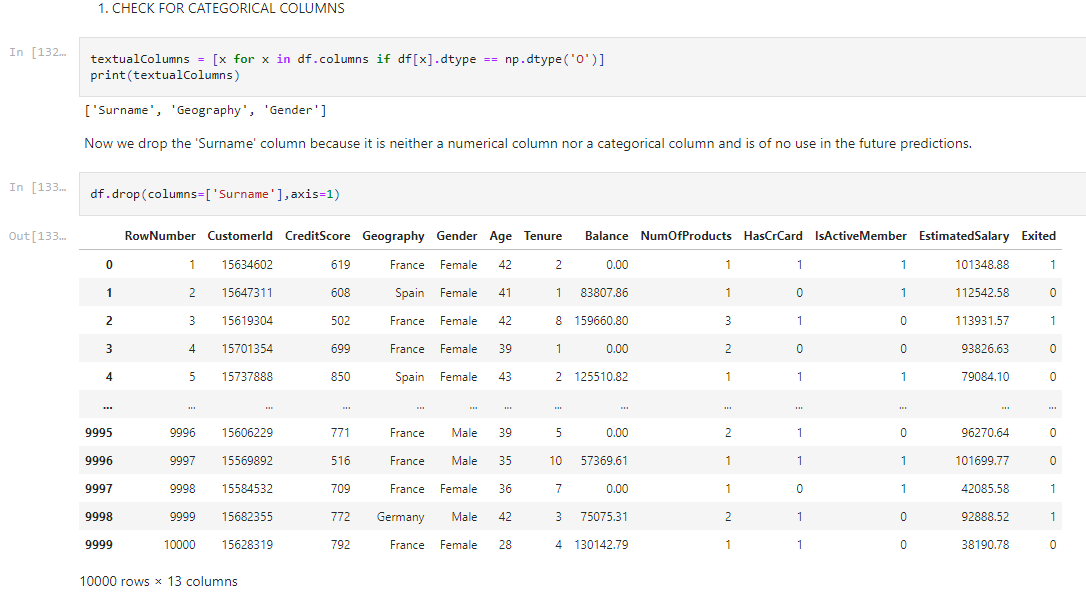
df\_main\_main=df\_main.drop(columns=['Surname'], axis=1)

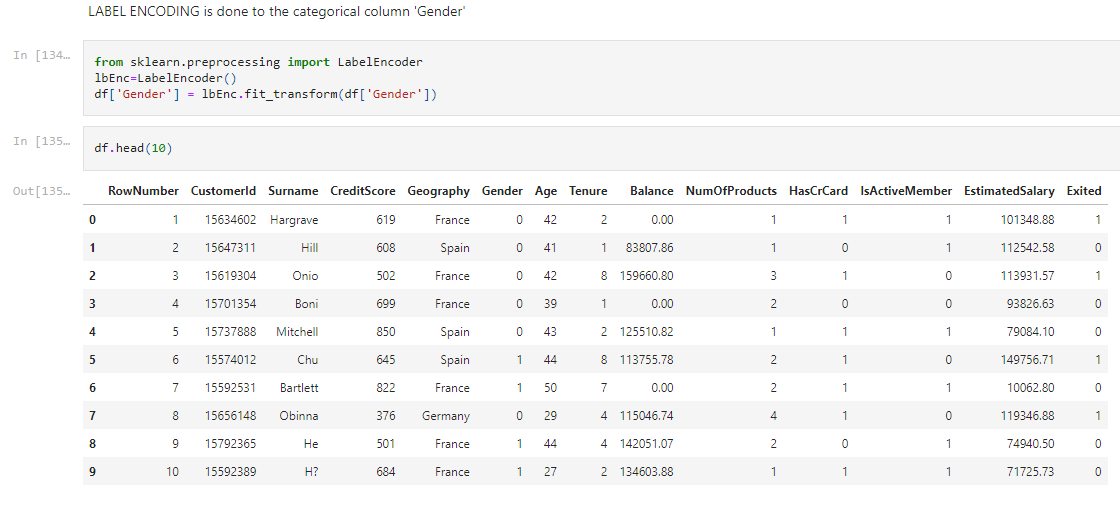
df\_main\_main.head(10)

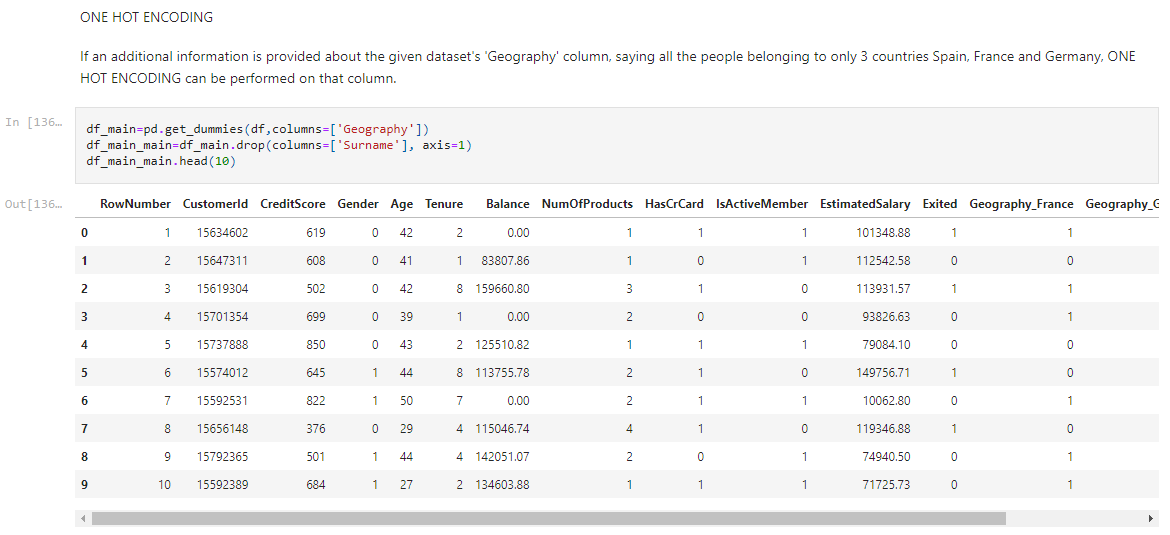
#-----------------#

#-----------------#

**Screenshots:**







**Task-8:**

Split the data into dependent and independent variables.

**Solution:**

X=df\_main\_main.drop(columns=['EstimatedSalary'],axis=1)

X.head()

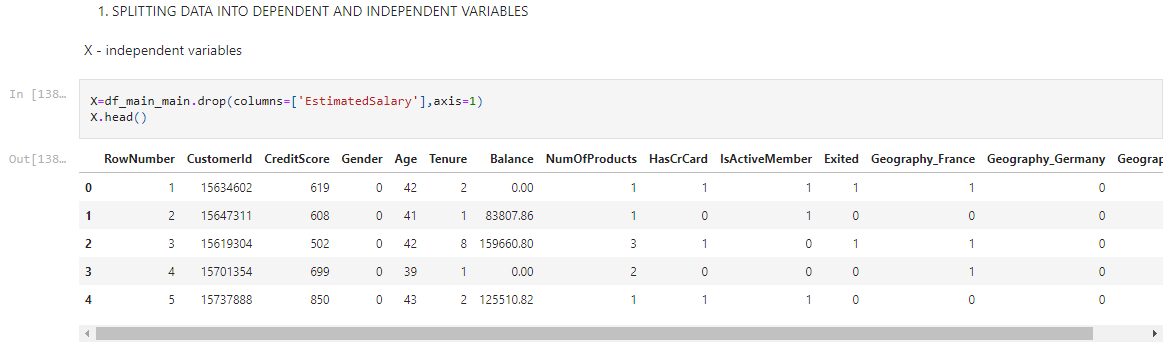
Y=df\_main\_main['EstimatedSalary']

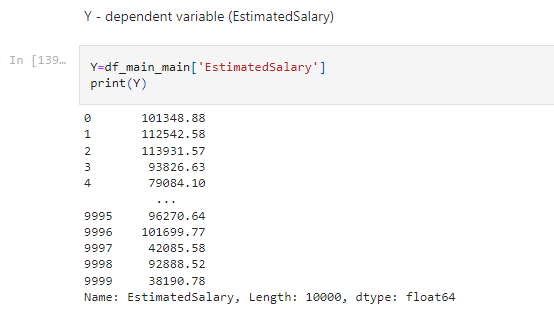
print(Y)

#-----------------#

#-----------------#

**Screenshots:**





**Task-9:**

Scale the independent variables

**Solution:**

from sklearn.preprocessing import scale

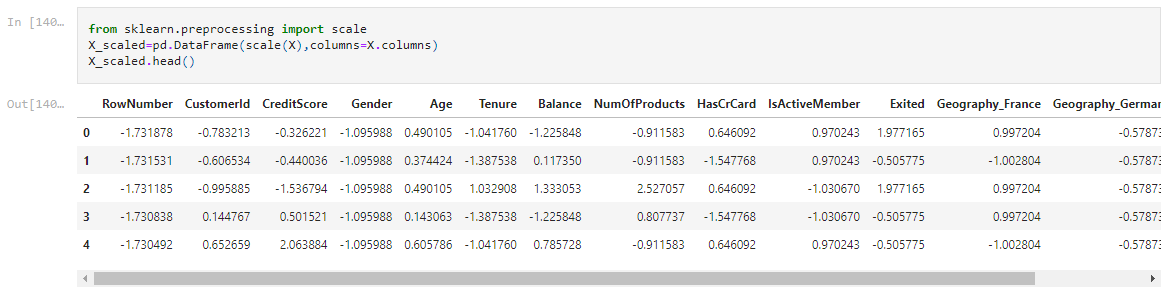
X\_scaled=pd.DataFrame(scale(X),columns=X.columns)

X\_scaled.head()

#-----------------#

#-----------------#

**Screenshot:**



**Task-10:**

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\_scaled,y, test\_size=0.3,random\_state=0)

print(X\_train.shape)

X\_train

#-----------------#

#-----------------#

**Screenshot:**

