Assignment-2

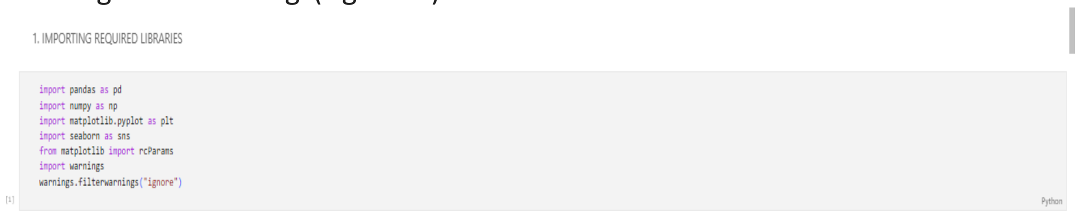
Assignment Date	19 September 2022
Student Name	Vijay S
Student Roll Number	2019115120
Maximum Marks	2 Marks

Question-1:

IMPORTING REQUIRED LIBRARIES

Solution:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
import warnings
warnings.filterwarnings("ignore")
```


1. IMPORTING REQUIRED LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
import warnings
warnings.filterwarnings("ignore")
```

Python

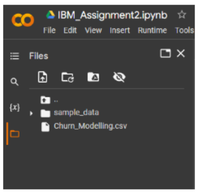
Question-2:

2.1.LOADING AND DISPLAYING THE DATASET

Solution:

```
df = pd.read_csv("Churn_Modelling.csv")

df.head()
```

2. LOADING AND DISPLAYING THE DATASET




```
df = pd.read_csv("Churn_Modelling.csv")
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

2.2.CHECKING FOR NULL VALUES IN ANY OF THE COLUMNS

Solution:

df.isnull().any()

CHECKING FOR NULL VALUES IN ANY OF THE COLUMNS

```
df.isnull().any()
```

```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

This shows that there are no null values or missing values in any of the columns of the dataset.

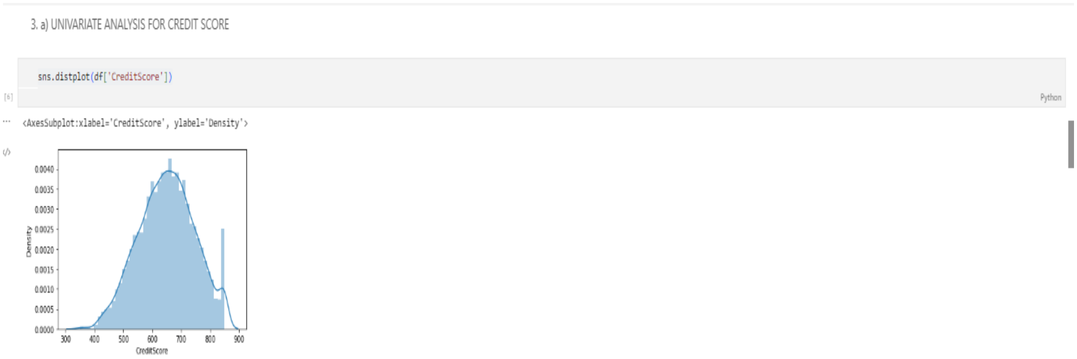
Question-3:

CHECKING FOR NULL VALUES IN ANY OF THE COLUMNS

3.1.UNIVARIATE ANALYSIS FOR CREDIT SCORE

Solution:

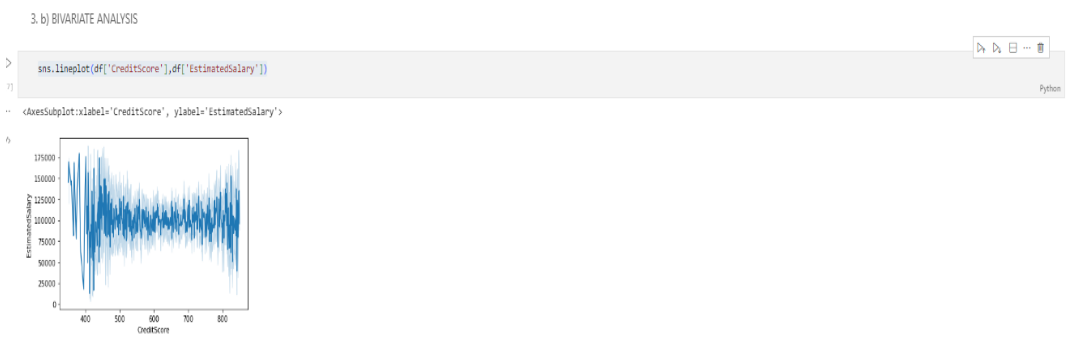
sns.distplot(df['CreditScore'])



3.2.BIVARIATE ANALYSIS

Solution:

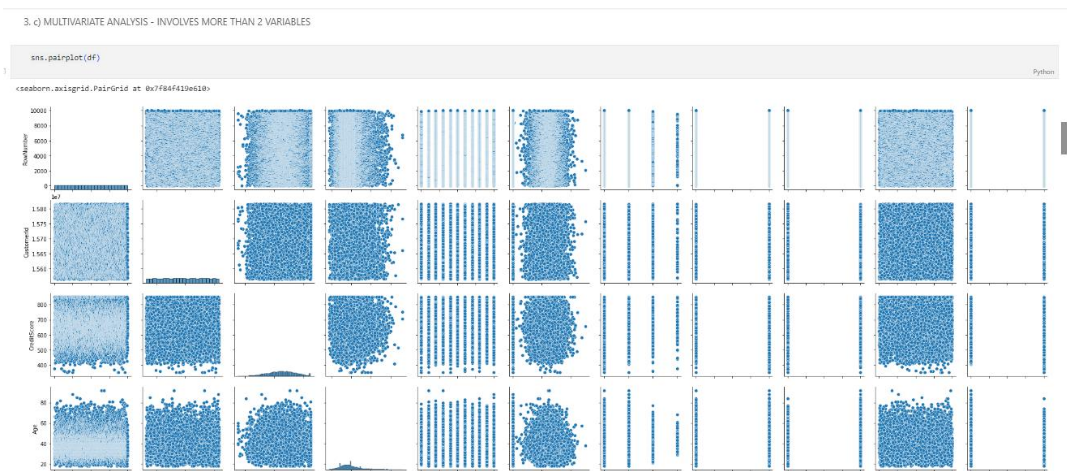
sns.lineplot(df['CreditScore'],df['EstimatedSalary'])

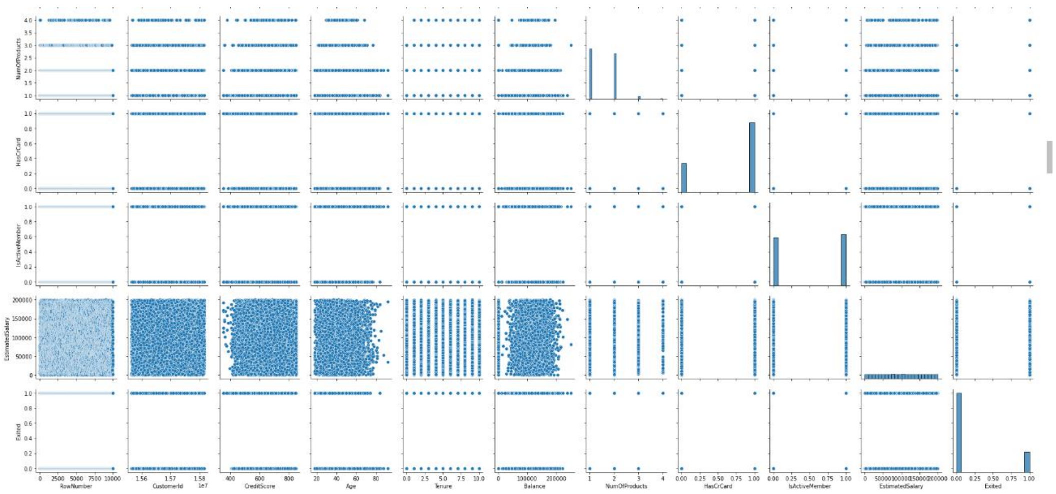


3. 3. MULTIVARIATE ANALYSIS - INVOLVES MORE THAN 2 VARIABLES

Solution:

sns.pairplot(df)





Question-4:
DESCRIPTIVE STATISTICS OF THE DATASET

Solution:
#descriptive analysis

df.describe()

4. DESCRIPTIVE STATISTICS OF THE DATASET

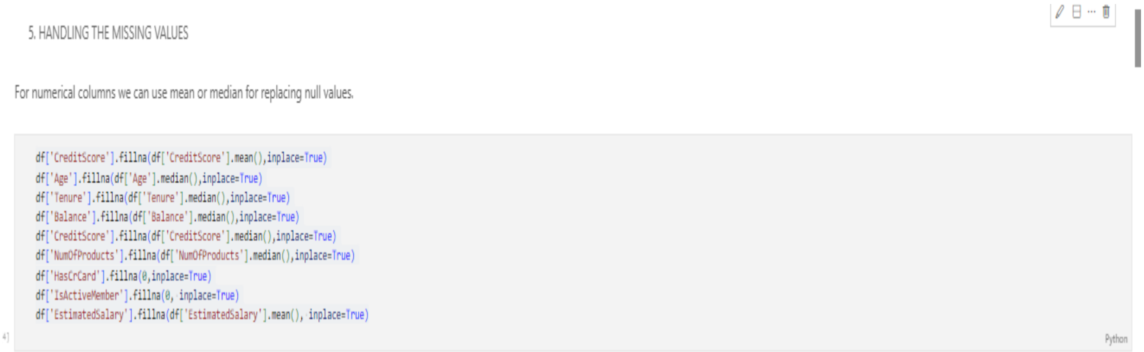
```
#descriptive analysis
df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

Question-5:
5.1.HANDLING THE MISSING VALUES
Solution:

```
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)
```



5. HANDLING THE MISSING VALUES

For numerical columns we can use mean or median for replacing null values.

```
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)
```

4]

Python

5.2.LINEAR REGRESSION BETWEEN BALANCE AND ESTIMATED SALARY

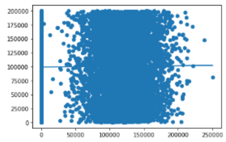
Solution:

```
from scipy import stats

x = df['Balance'].values

y = df['EstimatedSalary'].values


slope, intercept, r, p, std_err = stats.linregress(x, y)

print("B0 = ",intercept)

print("B1 = ",slope)

print("STD ERROR : ",std_err)


def myfunc(x):

  return slope * x + intercept


mymodel = list(map(myfunc, x))

print("Linear Regression model between balance and estimated salary \n")

plt.scatter(x, y)

plt.plot(x, mymodel)

plt.show()

df.corr()

sns.heatmap(df.corr())
```

LINEAR REGRESSION BETWEEN BALANCE AND ESTIMATED SALARY

```
from scipy import stats
x = df['Balance'].values
y = df['EstimatedSalary'].values

slope, intercept, r, p, std_err = stats.linregress(x, y)
print("B0 = ",intercept)
print("B1 = ",slope)
print("STD ERROR : ",std_err)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))
print("Linear Regression model between balance and estimated salary \n")
plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```

```
B0 =  99188.07297108804
B1 =  0.011795207172331332
STD ERROR :  0.009216975363422659
Linear Regression model between balance and estimated salary
```

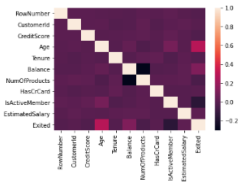



```
df.corr()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495	-0.009067	0.007246	0.000599	0.012044	-0.005988	-0.016571
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883	-0.012419	0.016972	-0.014025	0.001665	0.015271	-0.006248
CreditScore	0.005840	0.005308	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
Age	0.000783	0.009497	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
Tenure	-0.006495	-0.014883	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
Balance	-0.009067	-0.012419	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
NumOfProducts	0.007246	0.016972	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
HasCrCard	0.000599	-0.014025	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
IsActiveMember	0.012044	0.001665	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
EstimatedSalary	-0.005988	0.015271	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097
Exited	-0.016571	-0.006248	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.000000

```
sns.heatmap(df.corr())
```

```
<AxesSubplot:>
```



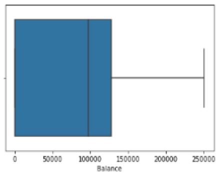
Question-6:
6.1.DETECTING OUTLIERS

Solution:

```
sns.boxplot(df.Balance)

sns.boxplot(df['EstimatedSalary'])

sns.boxplot(df['Age'])

df['Age'].median()
```

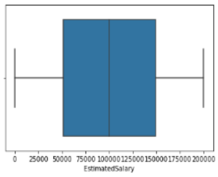
```
sns.boxplot(df.Balance)
```

```
<AxesSubplot:xlabel='Balance'>
```



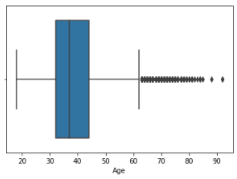

```
sns.boxplot(df['EstimatedSalary'])
```

```
<AxesSubplot:xlabel='EstimatedSalary'>
```




```
sns.boxplot(df['Age'])
```

```
<AxesSubplot:xlabel='Age'>
```



There are outliers in the 'Age' column values

```
df['Age'].median()
```

```
37.0
```

6.2.REPLACING THE OUTLIERS

Solution:

```
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

sns.boxplot(df['Age'])
```

REPLACING THE 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
```

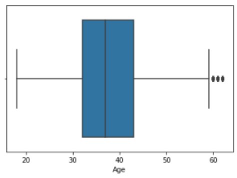
Python

Here we have replaced the outliers present in the 'Age' column by the median of the column (37).

```
sns.boxplot(df['Age'])
```

Python

<AxesSubplot:xlabel='Age'>



Solution:

Question-7:

7.1.CHECK FOR CATEGORICAL COLUMNS

Solution:

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

print(textualColumns)

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

7. CHECK FOR CATEGORICAL COLUMNS

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

Python

['Surname', 'Geography', 'Gender']

Now we drop the 'Surname' column because it is neither a numerical column nor a categorical column and is of no use in the future predictions.

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

Python

	RowNumber	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

7.2.LABEL ENCODING is done to the categorical column 'Gender'

Solution:

from sklearn.preprocessing import LabelEncoder

lbEnc=LabelEncoder()

df['Gender'] = lbEnc.fit_transform(df['Gender']) df.head(10)

LABEL ENCODING is done to the categorical column 'Gender'

```
from sklearn.preprocessing import LabelEncoder
lbEnc=LabelEncoder()
df['Gender'] = lbEnc.fit_transform(df['Gender'])
```

Python

```
df.head(10)
```

Python

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	0	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	0	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	0	43	2	125510.82	1	1	1	79084.10	0
5	6	15574012	Chu	645	Spain	1	44	8	113755.78	2	1	0	149756.71	1
6	7	15592531	Bartlett	822	France	1	50	7	0.00	2	1	1	10062.80	0
7	8	15656148	Obinna	376	Germany	0	29	4	115046.74	4	1	0	119346.88	1
8	9	15792365	He	501	France	1	44	4	142051.07	2	0	1	74940.50	0
9	10	15592389	H?	684	France	1	27	2	134603.88	1	1	1	71725.73	0

7.3.ONE HOT ENCODING

Solution:

df_main=pd.get_dummies(df,columns=['Geography'])

df_main_main=df_main.drop(columns=['Surname'], axis=1)

df_main_main.head(10)

ONE HOT ENCODING

```
df_main=pd.get_dummies(df,columns=['Geography'])
df_main_main=df_main.drop(columns=['Surname'], axis=1)
df_main_main.head(10)
```

Python

	RowNumber	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_France	Geography_Germany	Geography_Spain
0	1	15634602	619	0	42	2	0.00	1	1	1	101348.88	1	1	0	0
1	2	15647311	608	0	41	1	83807.86	1	0	1	112542.58	0	0	0	1
2	3	15619304	502	0	42	8	159660.80	3	1	0	113931.57	1	1	0	0
3	4	15701354	699	0	39	1	0.00	2	0	0	93826.63	0	1	0	0
4	5	15737888	850	0	43	2	125510.82	1	1	1	79084.10	0	0	0	1
5	6	15574012	645	1	44	8	113755.78	2	1	0	149756.71	1	0	0	1
6	7	15592531	822	1	50	7	0.00	2	1	1	10062.80	0	1	0	0
7	8	15656148	376	0	29	4	115046.74	4	1	0	119346.88	1	0	1	0
8	9	15792365	501	1	44	4	142051.07	2	0	1	74940.50	0	1	0	0
9	10	15592389	684	1	27	2	134603.88	1	1	1	71725.73	0	1	0	0

```
df_main_main.describe()
```

Python

	RowNumber	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_France	Geogra
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	
mean	5000.50000	1.569094e+07	650.528800	0.545700	37.763300	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700	0.501400	
std	2886.89568	7.193619e+04	96.653299	0.497932	8.644903	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769	0.500023	
min	1.00000	1.556570e+07	350.000000	0.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000	0.000000	
25%	2500.75000	1.562853e+07	584.000000	0.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000	0.000000	
50%	5000.50000	1.569074e+07	652.000000	1.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000	1.000000	
75%	7500.25000	1.575323e+07	718.000000	1.000000	43.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000	1.000000	
max	10000.00000	1.581569e+07	850.000000	1.000000	62.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000	1.000000	

Question-8:

SPLITTING DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

Solution:

X INDEPENDENT VARIABLES

X=df_main_main.drop(columns=['EstimatedSalary'],axis=1)

X.head()

Y DEPENDENT VARIABLES

Y=df_main_main['EstimatedSalary']

print(Y)

8. SPLITTING DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

X - independent variables

```
X=df_main_main.drop(columns=['EstimatedSalary'],axis=1)
X.head()
```

Python

	RowNumber	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Exited	Geography_France	Geography_Germany	Geography_Spain
0	1	15634602	619	0	42	2	0.00	1	1	1	1	1	0	0
1	2	15647311	608	0	41	1	83807.86	1	0	1	0	0	0	1
2	3	15619304	502	0	42	8	159660.80	3	1	0	1	1	0	0
3	4	15701354	699	0	39	1	0.00	2	0	0	0	1	0	0
4	5	15737888	850	0	43	2	125510.82	1	1	1	0	0	0	1

Y - dependent variable (EstimatedSalary)

```
Y=df_main_main['EstimatedSalary']
print(Y)
```

Python

```
0       101348.88
1       112542.58
2       113931.57
3        93826.63
4        79084.10
          ...
9995     96270.64
9996    101699.77
9997     42085.58
9998     92888.52
9999     38190.78
Name: EstimatedSalary, Length: 10000, dtype: float64
```

Question-9:
SCALING THE INDEPENDENT VARIABLES

Solution:
from sklearn.preprocessing import scale

X_scaled=pd.DataFrame(scale(X),columns=X.columns)

X_scaled.head()

9. SCALING THE INDEPENDENT VARIABLES

We do scaling for making data points generalized so that the distance between them will be lower

```
from sklearn.preprocessing import scale
X_scaled=pd.DataFrame(scale(X),columns=X.columns)
X_scaled.head()
```

Python

	RowNumber	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Exited	Geography_France	Geography_Germany	Geography_Spain
0	-1.731878	-0.783213	-0.326221	-1.095988	0.490105	-1.041760	-1.225848	-0.911583	0.646092	0.970243	1.977165	0.997204	-0.578736	-0.573809
1	-1.731531	-0.606534	-0.440036	-1.095988	0.374424	-1.387538	0.117350	-0.911583	-1.547768	0.970243	-0.505775	-1.002804	-0.578736	1.742740
2	-1.731185	-0.995885	-1.536794	-1.095988	0.490105	1.032908	1.333053	2.527057	0.646092	-1.030670	1.977165	0.997204	-0.578736	-0.573809
3	-1.730838	0.144767	0.501521	-1.095988	0.143063	-1.387538	-1.225848	0.807737	-1.547768	-1.030670	-0.505775	0.997204	-0.578736	-0.573809
4	-1.730492	0.652659	2.063884	-1.095988	0.605786	-1.041760	0.785728	-0.911583	0.646092	0.970243	-0.505775	-1.002804	-0.578736	1.742740

Question-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

10. SPLIT THE DATA INTO TRAINING AND TESTING

```python
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)
```

Python

```python
print(X_train.shape)
X_train
```

Python

```
(7000, 14)
```

	RowNumber	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Exited	Geography_France	Geography_Germany	Geography_Spain
7681	0.928899	-0.797032	-0.098592	0.912419	-0.551023	-1.041760	1.117213	0.807737	0.646092	0.970243	1.977165	0.997204	-0.578736	-0.573809
9031	1.396553	0.714314	-1.133270	0.912419	0.143063	0.687130	-1.225848	0.807737	0.646092	-1.030670	-0.505775	0.997204	-0.578736	-0.573809
3691	-0.453278	0.963450	-0.626278	-1.095988	-0.088299	-0.004426	1.354191	-0.911583	-1.547768	0.970243	-0.505775	0.997204	-0.578736	-0.573809
202	-1.661903	-1.250707	-1.391939	0.912419	1.415552	-0.004426	-1.225848	-0.911583	-1.547768	0.970243	1.977165	-1.002804	-0.578736	1.742740
5625	0.216680	-0.385174	-1.474714	-1.095988	2.572361	0.687130	1.070229	-0.911583	0.646092	0.970243	-0.505775	0.997204	-0.578736	-0.573809
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9225	1.463756	-1.473777	-0.584891	-1.095988	-0.666704	-0.350204	0.698607	0.807737	0.646092	0.970243	-0.505775	-1.002804	1.727904	-0.573809
4859	-0.048671	-0.609314	1.484464	-1.095988	-1.823512	-0.350204	0.608299	-0.911583	0.646092	0.970243	-0.505775	-1.002804	-0.578736	1.742740
3264	-0.601195	-1.620525	0.905045	0.912419	-0.319661	-0.004426	1.358909	0.807737	0.646092	-1.030670	-0.505775	0.997204	-0.578736	-0.573809
9845	1.678530	-0.374039	-0.626278	-1.095988	0.027382	1.378686	-1.225848	0.807737	0.646092	0.970243	-0.505775	-1.002804	-0.578736	1.742740
2732	-0.785485	-1.364118	-0.284834	-1.095988	1.184190	-1.387538	0.506303	-0.911583	0.646092	-1.030670	1.977165	-1.002804	1.727904	-0.573809

7000 rows × 14 columns