Assignment -2
Data Visualization and Preprocessing

Assignment Date	19 September 2022
Student Name	LOKESH.P
Student Roll Number	211419104151
Maximum Marks	2 Marks

Question-1:

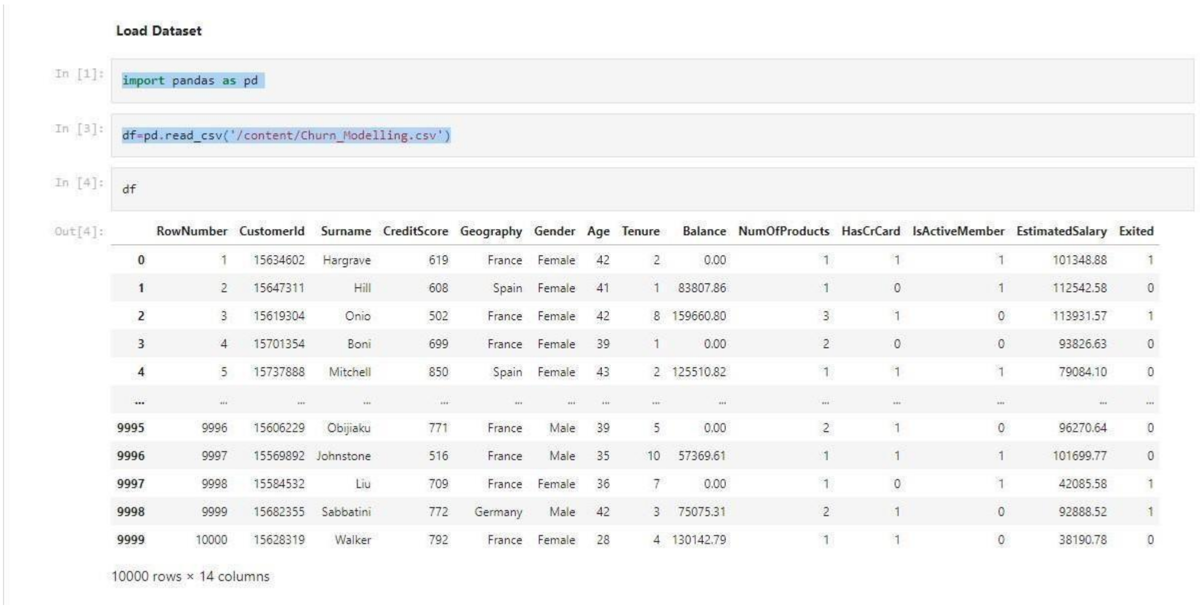
Download the dataset:

Question-2:

Load the dataset.

Solution:

```
import pandas as pd
df=pd.read_csv('/content/Churn_Modelling.csv')
```



Load Dataset

```
In [1]: import pandas as pd
```

```
In [3]: df=pd.read_csv('/content/Churn_Modelling.csv')
```

```
In [4]: df
```

Out[4]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 14 columns

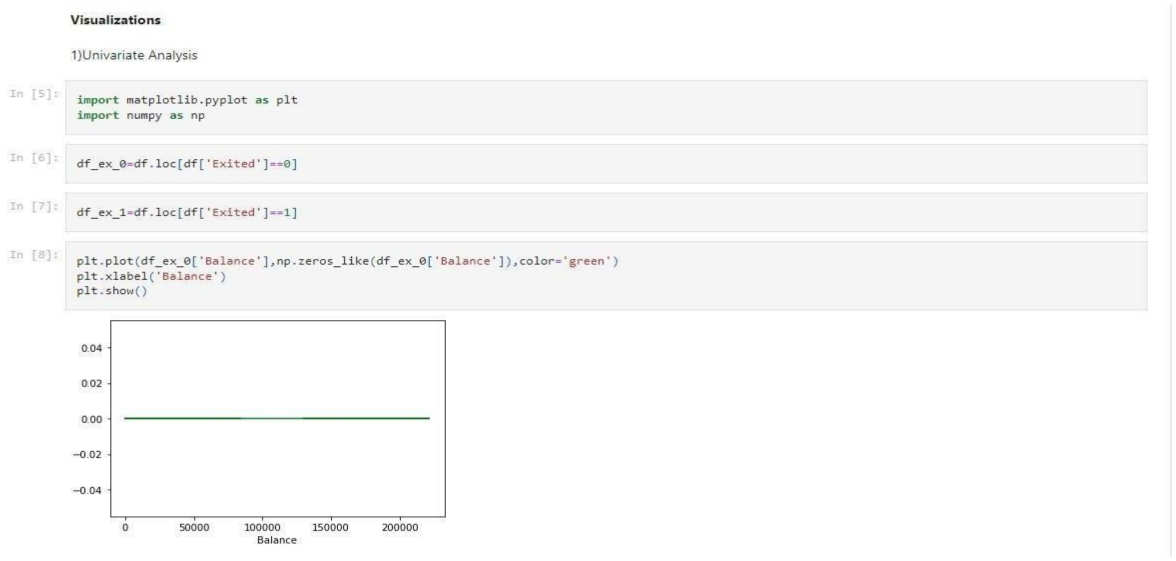
Question-3:

Perform Below Visualizations.

1)Univariate Analysis

Solution:

import matplotlib.pyplot **as** plt import
numpy **as** np
df_ex_0=df.loc[df['Exited']==0]
df_ex_1=df.loc[df['Exited']==1]
plt.plot(df_ex_0['Balance'],np.zeros_like(df_ex_0['Balance']),color='green'
) plt.xlabel('Balance') plt.show()



Visualizations

1)Univariate Analysis

```
In [5]: import matplotlib.pyplot as plt
        import numpy as np

In [6]: df_ex_0=df.loc[df['Exited']==0]

In [7]: df_ex_1=df.loc[df['Exited']==1]

In [8]: plt.plot(df_ex_0['Balance'],np.zeros_like(df_ex_0['Balance']),color='green')
        plt.xlabel('Balance')
        plt.show()
```

2)Bi - Variate Analysis

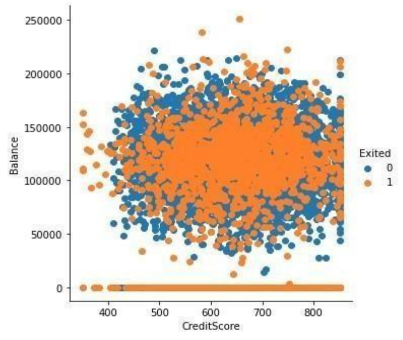
Solution:

import seaborn as sns
sns.FacetGrid(df,hue='Exited',size=5).map(plt.scatter,'CreditScore','Balance').add_legend()

2)Bivariate Analysis

```
In [10]: import seaborn as sns
         sns.FacetGrid(df,hue='Exited',size=5).map(plt.scatter,'CreditScore','Balance').add_legend()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has been renamed to `height`; please update your cod
e.
  warnings.warn(msg, UserWarning)
Out[10]: <seaborn.axisgrid.FacetGrid at 0x7f77b612a7d0>
```



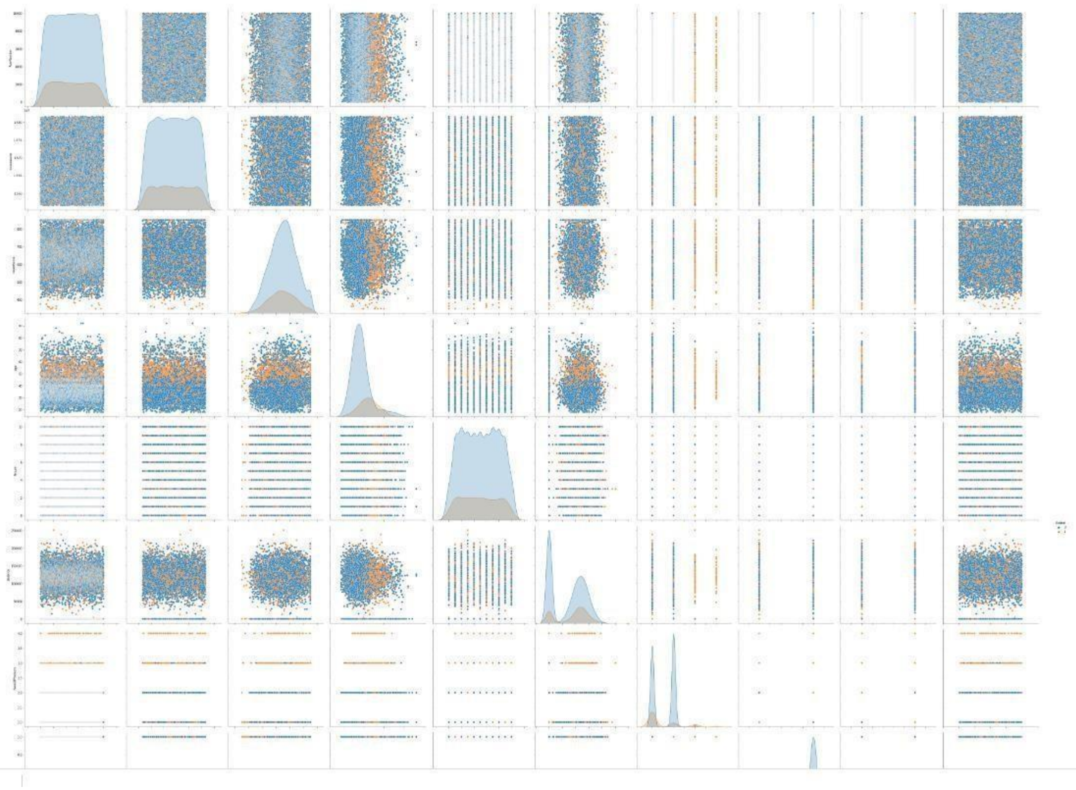
1)Multivariate Analysis

Solution:
sns.pairplot(df,hue='Exited',height=5)

3)Multivariate Analysis

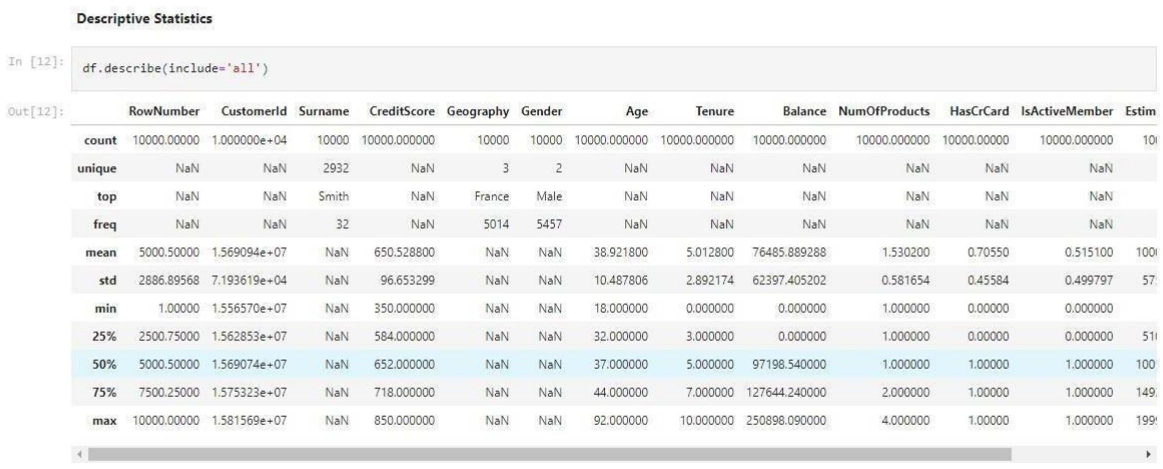
```
In [11]: sns.pairplot(df,hue='Exited',height=5)
```

```
Out[11]: <seaborn.axisgrid.PairGrid at 0x7f77a74e3910>
```



Question-4:

Perform descriptive statistics on the dataset.
Solution: df.describe(include='all')


Descriptive Statistics

```
In [12]: df.describe(include='all')
```

Out[12]:

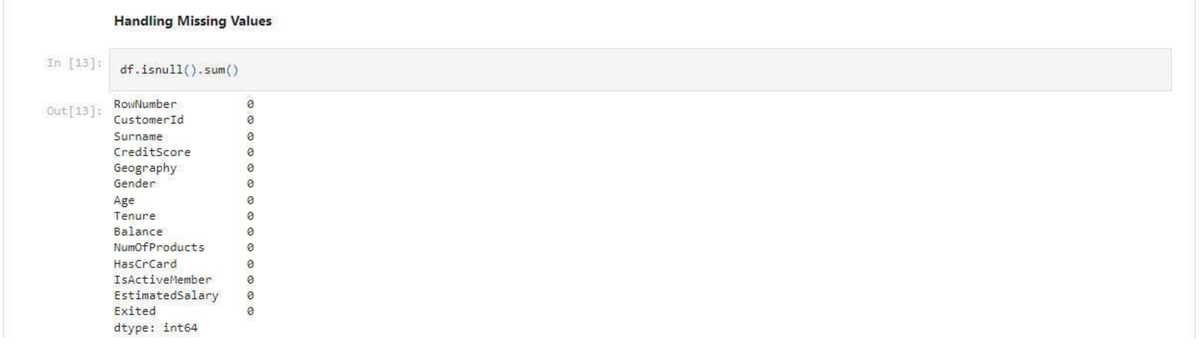
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estim
count	10000.00000	1.000000e+04	10000	10000.000000	10000	10000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10
unique	NaN	NaN	2932	NaN	3	2	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	Smith	NaN	France	Male	NaN	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	32	NaN	5014	5457	NaN	NaN	NaN	NaN	NaN	NaN	
mean	5000.50000	1.569094e+07	NaN	650.528800	NaN	NaN	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100
std	2886.89568	7.193619e+04	NaN	96.653299	NaN	NaN	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57
min	1.00000	1.556570e+07	NaN	350.000000	NaN	NaN	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	
25%	2500.75000	1.562853e+07	NaN	584.000000	NaN	NaN	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51
50%	5000.50000	1.569074e+07	NaN	652.000000	NaN	NaN	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100
75%	7500.25000	1.575323e+07	NaN	718.000000	NaN	NaN	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149
max	10000.00000	1.581569e+07	NaN	850.000000	NaN	NaN	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199

Question-5:

Handle the Missing values.

Solution:

df.isnull().sum()


Handling Missing Values

```
In [13]: df.isnull().sum()
```

```
Out[13]: RowNumber          0
         CustomerId         0
         Surname            0
         CreditScore        0
         Geography          0
         Gender             0
         Age                0
         Tenure             0
         Balance            0
         NumOfProducts      0
         HasCrCard          0
         IsActiveMember     0
         EstimatedSalary    0
         Exited             0
         dtype: int64
```

Question-6:

Find the outliers and replace the outliers

Solution:

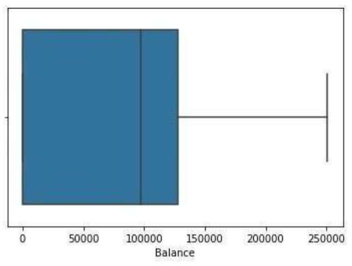
```
import seaborn as sns
sns.boxplot(df['Balance']
)
```

Find and Replace Outliers

```
In [14]: import seaborn as sns
```

```
In [15]: sns.boxplot(df['Balance'])
```

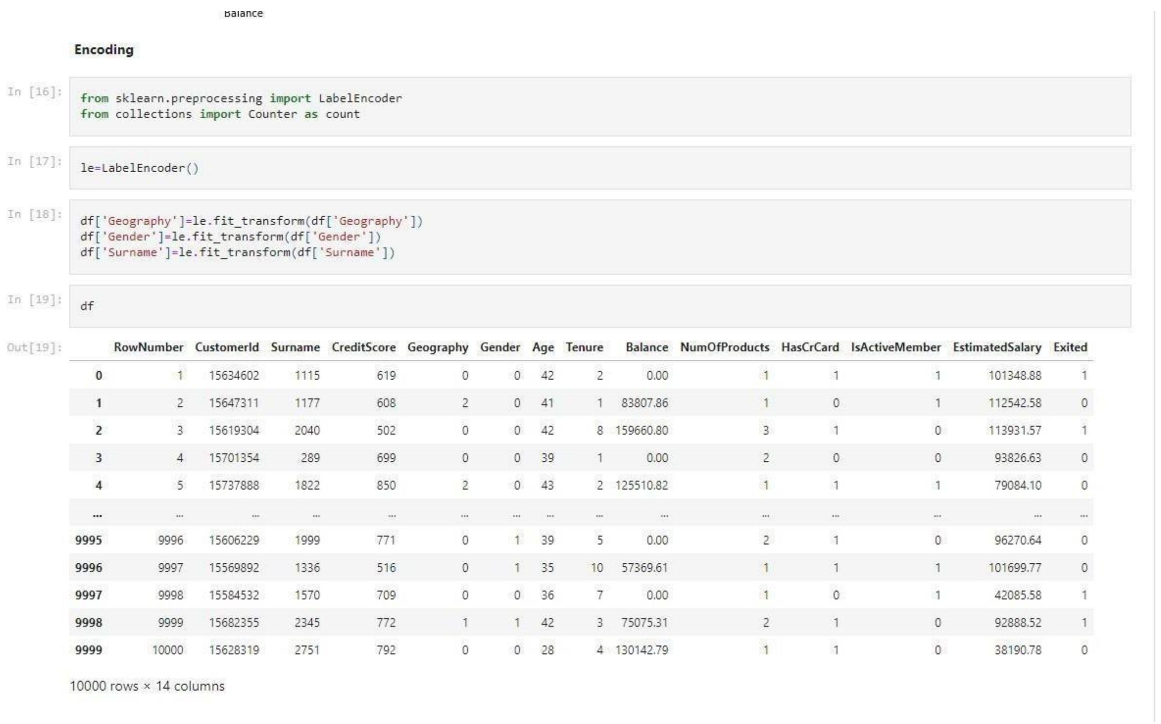
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretatio
n.
  FutureWarning
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f77a4b64550>
```



Question-7:

Check for Categorical columns and perform encoding.

Solution:
from sklearn.preprocessing **import** LabelEncoder **from** collections **import** Counter **as** count le**=**LabelEncoder()
df['Geography']**=**le**.**fit_transform(df['Geography'])
df['Gender']**=**le**.**fit_transform(df['Gender'])
df['Surname']**=**le**.**fit_transform(df['Surname'])



Balance

Encoding

```
In [16]: from sklearn.preprocessing import LabelEncoder
         from collections import Counter as count
```

```
In [17]: le=LabelEncoder()
```

```
In [18]: df['Geography']=le.fit_transform(df['Geography'])
         df['Gender']=le.fit_transform(df['Gender'])
         df['Surname']=le.fit_transform(df['Surname'])
```

```
In [19]: df
```

Out[19]:

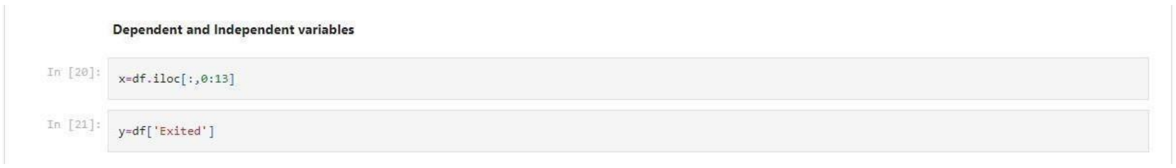
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	1115	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	1177	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	2040	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	289	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	1822	850	2	0	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	1999	771	0	1	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	1336	516	0	1	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	1570	709	0	0	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	2345	772	1	1	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	2751	792	0	0	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 14 columns

Question-8:

Split the data into dependent and independent variables.

Solution:
x=df.iloc[:,0:13]
y=df['Exited']



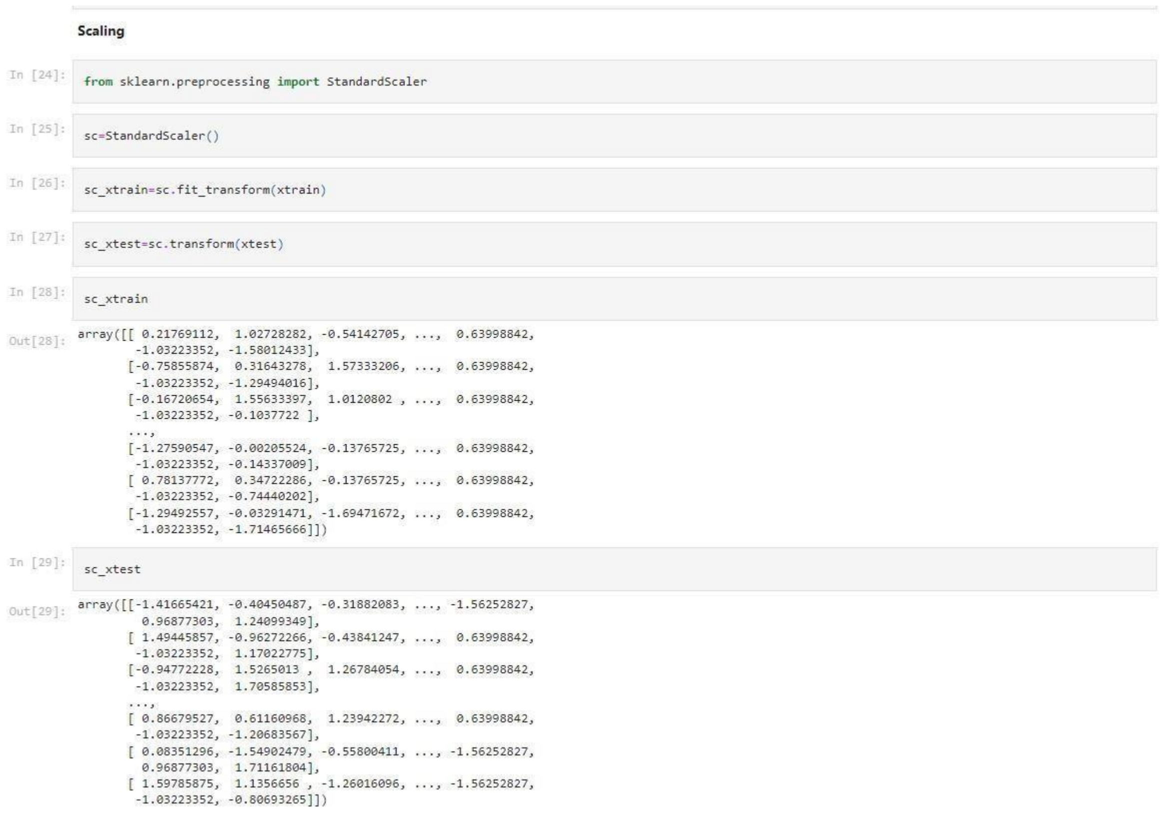
Dependent and Independent variables

```
In [20]: x=df.iloc[:,0:13]

In [21]: y=df['Exited']
```

Question-9:

Scale the independent variables

Solution: from sklearn.preprocessing import
StandardScaler sc=StandardScaler()
sc_xtrain=sc.fit_transform(xtrain)
sc_xtest=sc.transform(xtest)



Scaling

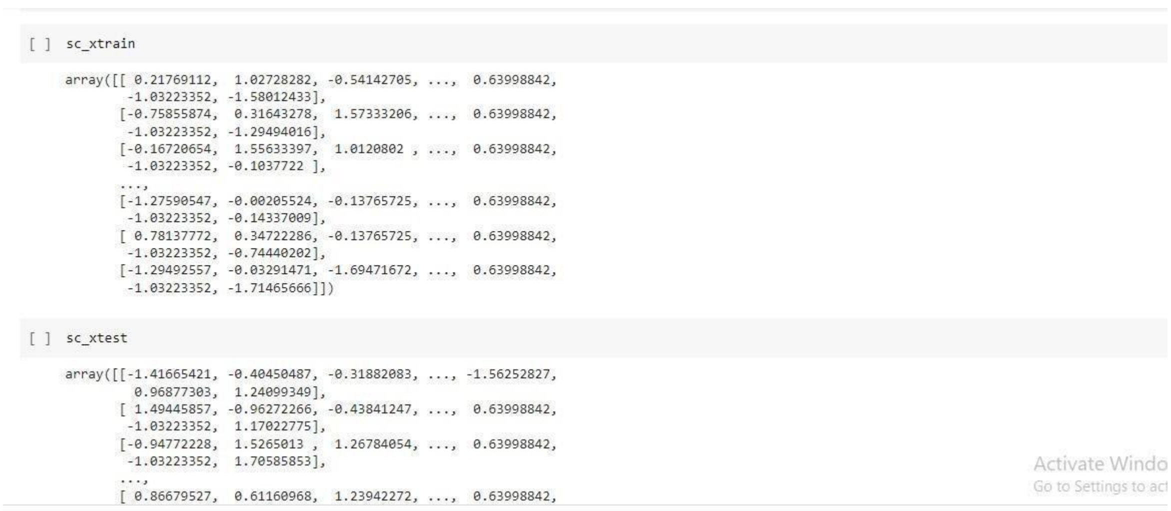
```
In [24]: from sklearn.preprocessing import StandardScaler

In [25]: sc=StandardScaler()

In [26]: sc_xtrain=sc.fit_transform(xtrain)

In [27]: sc_xtest=sc.transform(xtest)

In [28]: sc_xtrain

Out[28]: array([[ 0.21769112,  1.02728282, -0.54142705, ...,  0.63998842,
                 -1.03223352, -1.58012433],
                [-0.75855874,  0.31643278,  1.57333206, ...,  0.63998842,
                 -1.03223352, -1.29494016],
                [-0.16720654,  1.55633397,  1.0120802 , ...,  0.63998842,
                 -1.03223352, -0.1037722 ],
                ...,
                [-1.27590547, -0.00205524, -0.13765725, ...,  0.63998842,
                 -1.03223352, -0.14337009],
                [ 0.78137772,  0.34722286, -0.13765725, ...,  0.63998842,
                 -1.03223352, -0.74440202],
                [-1.29492557, -0.03291471, -1.69471672, ...,  0.63998842,
                 -1.03223352, -1.71465666]])

In [29]: sc_xtest

Out[29]: array([[-1.41665421, -0.40450487, -0.31882083, ..., -1.56252827,
                  0.96877303,  1.24099349],
                [ 1.49445857, -0.96272266, -0.43841247, ...,  0.63998842,
                 -1.03223352,  1.17022775],
                [-0.94772228,  1.5265013 ,  1.26784054, ...,  0.63998842,
                 -1.03223352,  1.70585853],
                ...,
                [ 0.86679527,  0.61160968,  1.23942272, ...,  0.63998842,
                 -1.03223352, -1.20683567],
                [ 0.08351296, -1.54902479, -0.55800411, ..., -1.56252827,
                  0.96877303,  1.71161804],
                [ 1.59785875,  1.1356656 , -1.26016096, ..., -1.56252827,
                 -1.03223352, -0.80693265]])
```

Question-10:

Testing and training data

Solution: from sklearn.model_selection import
train_test_split

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=10)



```
[ ]  sc_xtrain

     array([[ 0.21769112,  1.02728282, -0.54142705, ...,  0.63998842,
             -1.03223352, -1.58012433],
            [-0.75855874,  0.31643278,  1.57333206, ...,  0.63998842,
             -1.03223352, -1.29494016],
            [-0.16720654,  1.55633397,  1.0120802 , ...,  0.63998842,
             -1.03223352, -0.1037722 ],
            ...,
            [-1.27590547, -0.00205524, -0.13765725, ...,  0.63998842,
             -1.03223352, -0.14337009],
            [ 0.78137772,  0.34722286, -0.13765725, ...,  0.63998842,
             -1.03223352, -0.74440202],
            [-1.29492557, -0.03291471, -1.69471672, ...,  0.63998842,
             -1.03223352, -1.71465666]])

[ ]  sc_xtest

     array([[-1.41665421, -0.40450487, -0.31882083, ..., -1.56252827,
              0.96877303,  1.24099349],
            [ 1.49445857, -0.96272266, -0.43841247, ...,  0.63998842,
             -1.03223352,  1.17022775],
            [-0.94772228,  1.5265013 ,  1.26784054, ...,  0.63998842,
             -1.03223352,  1.70585853],
            ...,
            [ 0.86679527,  0.61160968,  1.23942272, ...,  0.63998842,
```