Assignment -2

Python Programming

Assignment Date	26 September 2022
Student Name	Ms S.Ragavi
Student Roll Number	510119104016
Maximum Marks	2 Marks

Question 1: Load the dataset.

Solution:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
dataset1=pd.read_csv("/content/drive/MyDrive/ibm/dataset.csv")
dataset1.head()
```

OUTPUT:

```
[1] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
```

```
[3] dataset1=pd.read_csv("/content/drive/MyDrive/ibm/dataset.csv")
```

```
dataset1.head()
```

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619.000000	0	42.0	2	0.00	1	1	1	101348.88	1
1	608.000000	0	41.0	1	83807.86	1	0	1	112542.58	0
2	502.000000	0	42.0	8	159660.80	3	1	0	113931.57	1
3	699.000000	0	39.0	1	0.00	2	0	0	93826.63	0
4	628.808847	0	43.0	2	125510.82	1	1	1	79084.10	0

```
dataset1.tail()
dataset1.shape
```

OUTPUT:

```
[ ] dataset1.tail()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
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

```
[ ] dataset1.shape
```

```
(10000, 14)
```

Question 2: Perform Univariate,Bivariate and Multivariate Analysis.

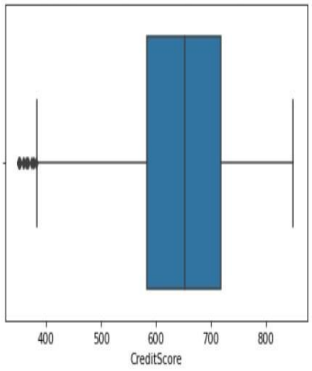
Solution:

```
sns.boxplot(dataset1['CreditScore'])
sns.boxplot(dataset1['Age'])
sns.boxplot(dataset1['Tenure'])
sns.boxplot(dataset1['Balance'])
sns.boxplot(dataset1['EstimatedSalary'])
sns.heatmap(dataset1.corr(),annot=True)
```

OUTPUT:

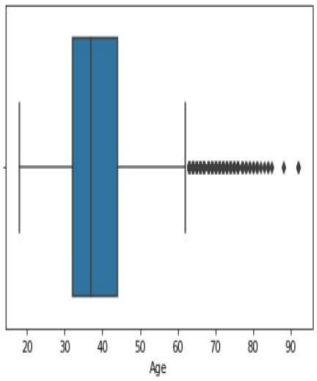
```
[10] sns.boxplot(dataset1['CreditScore'])
```

```
/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, the only valid positional argument
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f92ac9dad90>
```



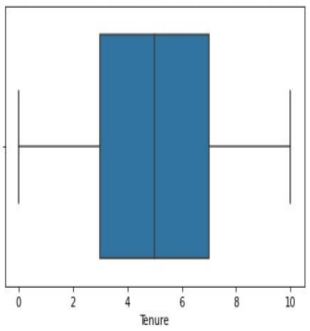

```
[11] sns.boxplot(dataset1['Age'])
```

```
/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, the only valid positional argument
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f92aca163d0>
```



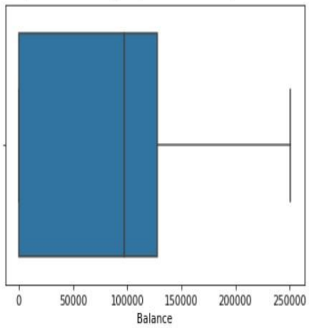
```
[13] sns.boxplot(dataset1['Tenure'])
```

```
/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, the only valid positional argument
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f92ac1be1d0>
```



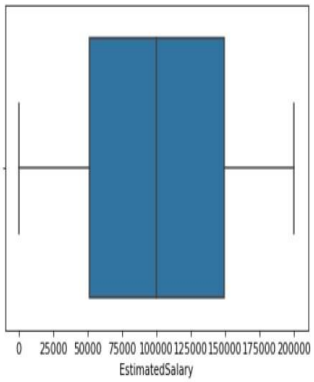

```
[14] sns.boxplot(dataset1['Balance'])
```

```
/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, the only valid positional argument
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f92ac13d7d0>
```



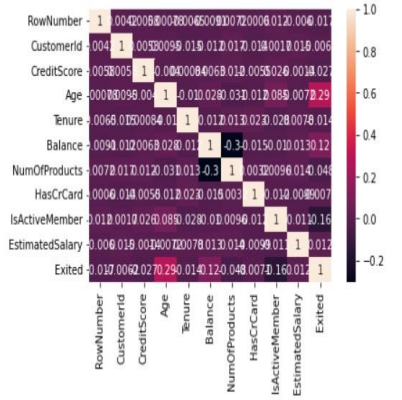

```
[15] sns.boxplot(dataset1['EstimatedSalary'])
```

```
/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, the only valid positional argument
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f92ac101890>
```



```
[17] sns.heatmap(dataset1.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f92aa259550>
```



Question 3: Perform descriptive ststistics on the dataset.

Solution:

```
dataset1.describe()
```

OUTPUT:

```
dataset1.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 4: Handle the missing values.

Solution:

```
dataset1.isnull().sum()
dataset1.duplicated().sum()
dataset1.isna().sum()
dataset1.nunique()
dataset1.info()
```

OUTPUT:

```
[18] dataset1.isnull().sum()
```

```
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
```

```
#Handling Missing values
dataset1.duplicated().sum()
```

```
0
```

```
[22] dataset1.isna().sum()
```

```
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
```

```
[23] dataset1.nunique()

RowNumber          10000
CustomerId         10000
Surname             2932
CreditScore          460
Geography              3
Gender                 2
Age                   70
Tenure                11
Balance             6382
NumOfProducts          4
HasCrCard              2
IsActiveMember         2
EstimatedSalary     9999
Exited                 2
dtype: int64
```

```
[24] dataset1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Question 5: Find the outliers and replace the outliers

Solution:

```
out=dataset1.drop(columns=['Gender','Tenure','HasCrCard','IsActiveMembe
r','NumOfProducts','Exited']).quantile(q=[0.25,0.50])

out
```

```
Q1=out.iloc[0]
Q3=out.iloc[1]
iqr=Q3-Q1
iqr
```

OUTPUT:

```
[26] #Find the outliers
     out=dataset1.drop(columns=['Gender','Tenure','HasCrCard','IsActiveMember','NumOfProducts','Exited']).quantile(q=[0.25,0.50])
     out
```

	RowNumber	CustomerId	CreditScore	Age	Balance	EstimatedSalary
0.25	2500.75	15628528.25	584.0	32.0	0.00	51002.110
0.50	5000.50	15690738.00	652.0	37.0	97198.54	100193.915

```
Q1=out.iloc[0]
Q3=out.iloc[1]
iqr=Q3-Q1
iqr
```

```
RowNumber          2499.750
CustomerId        62209.750
CreditScore          68.000
Age                   5.000
Balance           97198.540
EstimatedSalary   49191.805
dtype: float64
```

```
upper=out.iloc[1]+1.5*iqr
upper

lower=out.iloc[0]-1.5*iqr
lower
```

OUTPUT:

```
[28] upper=out.iloc[1]+1.5*iqr
     upper
```

```
RowNumber          8.750125e+03
CustomerId         1.578405e+07
CreditScore        7.540000e+02
Age                4.450000e+01
Balance            2.429964e+05
EstimatedSalary    1.739816e+05
dtype: float64
```

```
[29] lower=out.iloc[0]-1.5*iqr
     lower
```

```
RowNumber         -1.248875e+03
CustomerId         1.553521e+07
CreditScore        4.820000e+02
Age                2.450000e+01
Balance           -1.457978e+05
EstimatedSalary   -2.278560e+04
dtype: float64
```

```
#Replace outliers
dataset['Balance']=np.where(dataset1['Balance']>75000,dataset['Balance'
].mean(),dataset1['Balance'])
dataset1[dataset1['Balance']>75000]
```

OUTPUT:

```
#Replace outliers
dataset['Balance']=np.where(dataset1['Balance']>75000,dataset['Balance'].mean(),dataset1['Balance'])
dataset1[dataset1['Balance']>75000]
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	2	15647311	Hill	608.0000	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502.0000	France	Female	42	8	159660.80	3	1	0	113931.57	1
4	5	15737888	Mitchell	650.5288	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
5	6	15574012	Chu	645.0000	Spain	Male	44	8	113755.78	2	1	0	149756.71	1
7	8	15656148	Obinna	376.0000	Germany	Female	29	4	115046.74	4	1	0	119346.88	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9987	9988	15588839	Mancini	606.0000	Spain	Male	30	8	180307.73	2	1	1	1914.41	0
9991	9992	15769959	Ajuluchukwu	597.0000	France	Female	53	4	88381.21	1	1	0	69384.71	1
9993	9994	15569266	Rahman	644.0000	France	Male	28	7	155060.41	1	1	0	29179.52	0
9998	9999	15682355	Sabbatini	650.5288	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	650.5288	France	Female	28	4	130142.79	1	1	0	38190.78	0

5959 rows × 14 columns

Question 6: Check for categorical columns and perform encoding

Solution:

```
#categorical column and encoding

dataset1['Gender'].replace({'Male':1,'Female':0},inplace=True)
dataset1.head(10)
```

OUTPUT:

```
[72] dataset1['Gender'].replace({'Male':1,'Female':0},inplace=True)
     dataset1.head(10)
```

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619.000000	0	42.0	2	0.00	1	1	1	101348.88	1
1	608.000000	0	41.0	1	83807.86	1	0	1	112542.58	0
2	502.000000	0	42.0	8	159660.80	3	1	0	113931.57	1
3	699.000000	0	39.0	1	0.00	2	0	0	93826.63	0
4	628.808847	0	43.0	2	125510.82	1	1	1	79084.10	0
5	645.000000	1	44.0	8	113755.78	2	1	0	149756.71	1
6	628.808847	1	50.0	7	0.00	2	1	1	10062.80	0
7	376.000000	0	29.0	4	115046.74	4	1	0	119346.88	1
8	501.000000	1	44.0	4	142051.07	2	0	1	74940.50	0
9	684.000000	1	27.0	2	134603.88	1	1	1	71725.73	0

```
pd.get_dummies(dataset1,columns=['Geography'])
```

OUTPUT:

```
#categorical column and encoding
pd.get_dummies(dataset1,columns=['Geography'])
```

	RowNumber	CustomerId	Surname	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_0	Geography_1	Geograph
0	1	15634602	Hargrave	619.000000	0	42.0	2	0.00	1	1	1	101348.88	1	1	0	
1	2	15647311	Hill	608.000000	0	41.0	1	83807.86	1	0	1	112542.58	0	0	0	
2	3	15619304	Onio	502.000000	0	42.0	8	159660.80	3	1	0	113931.57	1	1	0	
3	4	15701354	Boni	699.000000	0	39.0	1	0.00	2	0	0	93826.63	0	1	0	
4	5	15737888	Mitchell	628.808847	0	43.0	2	125510.82	1	1	1	79084.10	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	628.808847	1	39.0	5	0.00	2	1	0	96270.64	0	1	0	
9996	9997	15569892	Johnstone	516.000000	1	35.0	10	57369.61	1	1	1	101699.77	0	1	0	
9997	9998	15584532	Liu	709.000000	0	36.0	7	0.00	1	0	1	42085.58	1	1	0	
9998	9999	15682355	Sabbatini	628.808847	1	42.0	3	75075.31	2	1	0	92888.52	1	0	1	
9999	10000	15628319	Walker	628.808847	0	28.0	4	130142.79	1	1	0	38190.78	0	1	0	

10000 rows × 16 columns

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dataset1['Geography']=le.fit_transform(dataset['Geography'])
dataset1.head()
```

OUTPUT:

```
[47] from sklearn.preprocessing import LabelEncoder
     le=LabelEncoder()
     dataset1['Geography']=le.fit_transform(dataset['Geography'])
     dataset1.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619.000000	0	0	42.0	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608.000000	2	0	41.0	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502.000000	0	0	42.0	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699.000000	0	0	39.0	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	628.808847	2	0	43.0	2	125510.82	1	1	1	79084.10	0

Question 7: Split the data into dependent and independent variables.
Solution:

```
x=dataset1.iloc[:,1:6]
y=dataset1.iloc[:,6]
dataset1.iloc[1:8]
```

OUTPUT:

```
[53] x=dataset1.iloc[:,1:6]
     y=dataset1.iloc[:,6]
     dataset1.iloc[1:8]
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	2	15647311	Hill	608.000000	2	0	41.0	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502.000000	0	0	42.0	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699.000000	0	0	39.0	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	628.808847	2	0	43.0	2	125510.82	1	1	1	79084.10	0
5	6	15574012	Chu	645.000000	2	1	44.0	8	113755.78	2	1	0	149756.71	1
6	7	15592531	Bartlett	628.808847	0	1	50.0	7	0.00	2	1	1	10062.80	0
7	8	15656148	Obinna	376.000000	1	0	29.0	4	115046.74	4	1	0	119346.88	1

```
dataset1=dataset1.drop(columns=['RowNumber','CustomerId','Surname','Geography'])
dataset1.head(10)
```

OUTPUT:

```
dataset1=dataset1.drop(columns=['RowNumber','CustomerId','Surname','Geography'])
dataset1.head(10)
```

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619.000000	0	42.0	2	0.00	1	1	1	101348.88	1
1	608.000000	0	41.0	1	83807.86	1	0	1	112542.58	0
2	502.000000	0	42.0	8	159660.80	3	1	0	113931.57	1
3	699.000000	0	39.0	1	0.00	2	0	0	93826.63	0
4	628.808847	0	43.0	2	125510.82	1	1	1	79084.10	0
5	645.000000	1	44.0	8	113755.78	2	1	0	149756.71	1
6	628.808847	1	50.0	7	0.00	2	1	1	10062.80	0
7	376.000000	0	29.0	4	115046.74	4	1	0	119346.88	1
8	501.000000	1	44.0	4	142051.07	2	0	1	74940.50	0
9	684.000000	1	27.0	2	134603.88	1	1	1	71725.73	0

```
x=dataset1.iloc[:,:-1]
x.head()

y=dataset1.iloc[:,-1]
y.head()
```

OUTPUT:

```
x=dataset1.iloc[:,:-1]
x.head()
```

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619.000000	0	42.0	2	0.00	1	1	1	101348.88
1	608.000000	0	41.0	1	83807.86	1	0	1	112542.58
2	502.000000	0	42.0	8	159660.80	3	1	0	113931.57
3	699.000000	0	39.0	1	0.00	2	0	0	93826.63
4	628.808847	0	43.0	2	125510.82	1	1	1	79084.10

```
[61] y=dataset1.iloc[:,-1]
     y.head()

     0    1
     1    0
     2    1
     3    0
     4    0
     Name: Exited, dtype: int64
```

Question 8: Scale the independent variables

Solution:

```
#scale the independent variables
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x=ss.fit_transform(x)
x
```

OUTPUT:

```
#scale the independent variables
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x=ss.fit_transform(x)
x

array([[-0.09113993, -1.09598752,  0.48205148, ...,  0.64609167,
         0.97024255,  0.02188649],
       [-0.24119429, -1.09598752,  0.36638802, ..., -1.54776799,
         0.97024255,  0.21653375],
       [-1.68717266, -1.09598752,  0.48205148, ...,  0.64609167,
        -1.03067011,  0.2406869 ],
       ...,
       [ 1.13657755, -1.09598752, -0.21192932, ..., -1.54776799,
         0.97024255, -1.00864308],
       [ 0.04266555,  0.91241915,  0.48205148, ...,  0.64609167,
        -1.03067011, -0.12523071],
       [ 0.04266555, -1.09598752, -1.13723705, ...,  0.64609167,
        -1.03067011, -1.07636976]])
```

Question 9: Split the data into training and testing

Solution:

```
#split and train the dataset
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

x_train.shape

x_test.shape

y_train.shape

y_test.shape
```

OUTPUT: