Assignment -2

Data Visualization and Pre-processing

Assignment Date	19 September 2022
Student Name	Vasu deva krishna rayan K
Student Roll Number	953719104058
Maximum Marks	2 Marks

QUESTION 1: Load the dataset:- **SOLUTION:**

```
import pandas as pd import
seaborn as sns
df=pd.read_csv("/content/Churn_Modelling.csv")
df.dtypes
```

OUTPUT:

Python

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSala
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.
...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.

10000 rows × 14 columns

QUESTION 2:

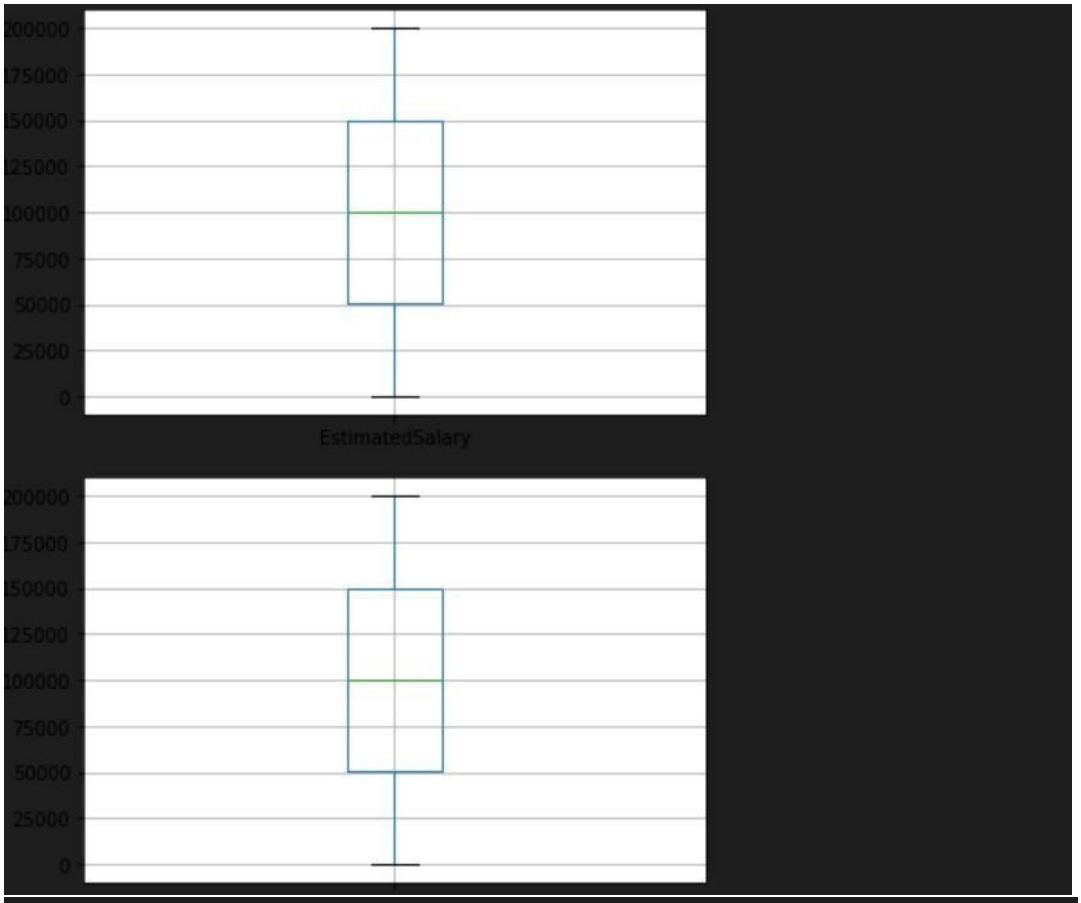
Perform Below Visualizations.

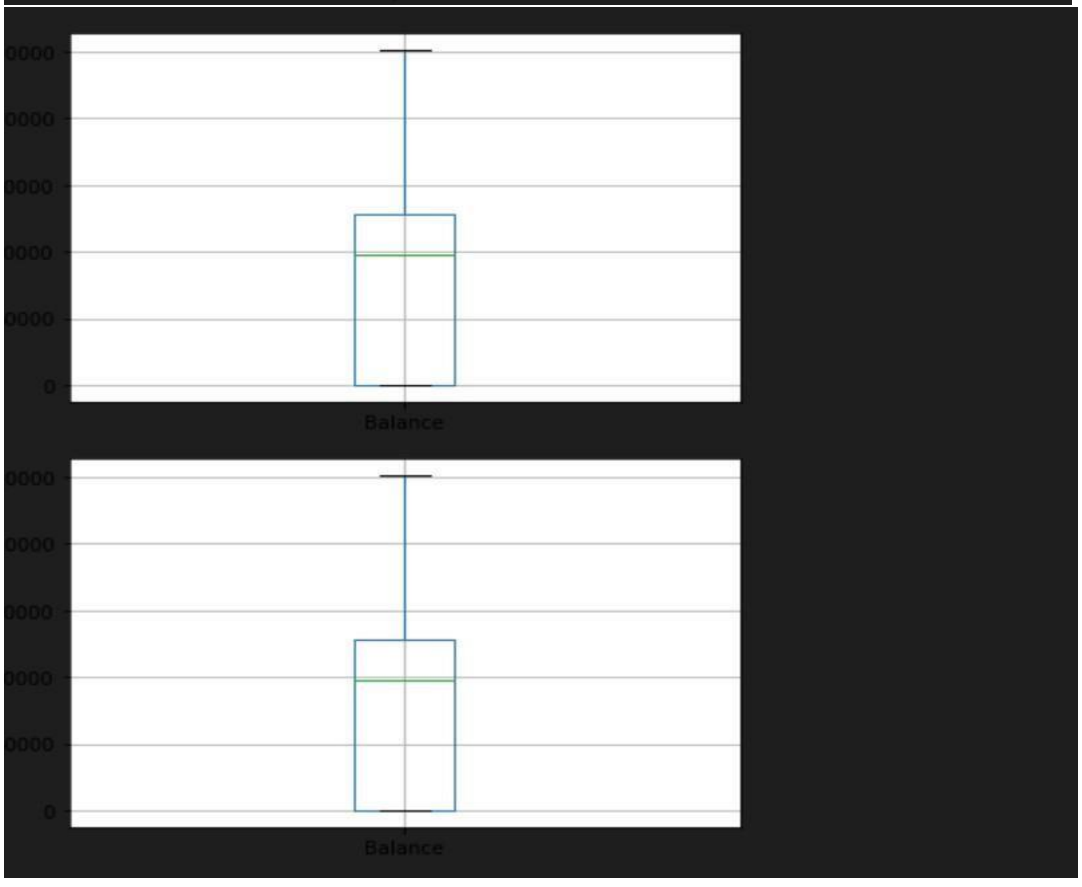
- Univariate Analysis
- Bi - Variate Analysis
- Multi - Variate Analysis

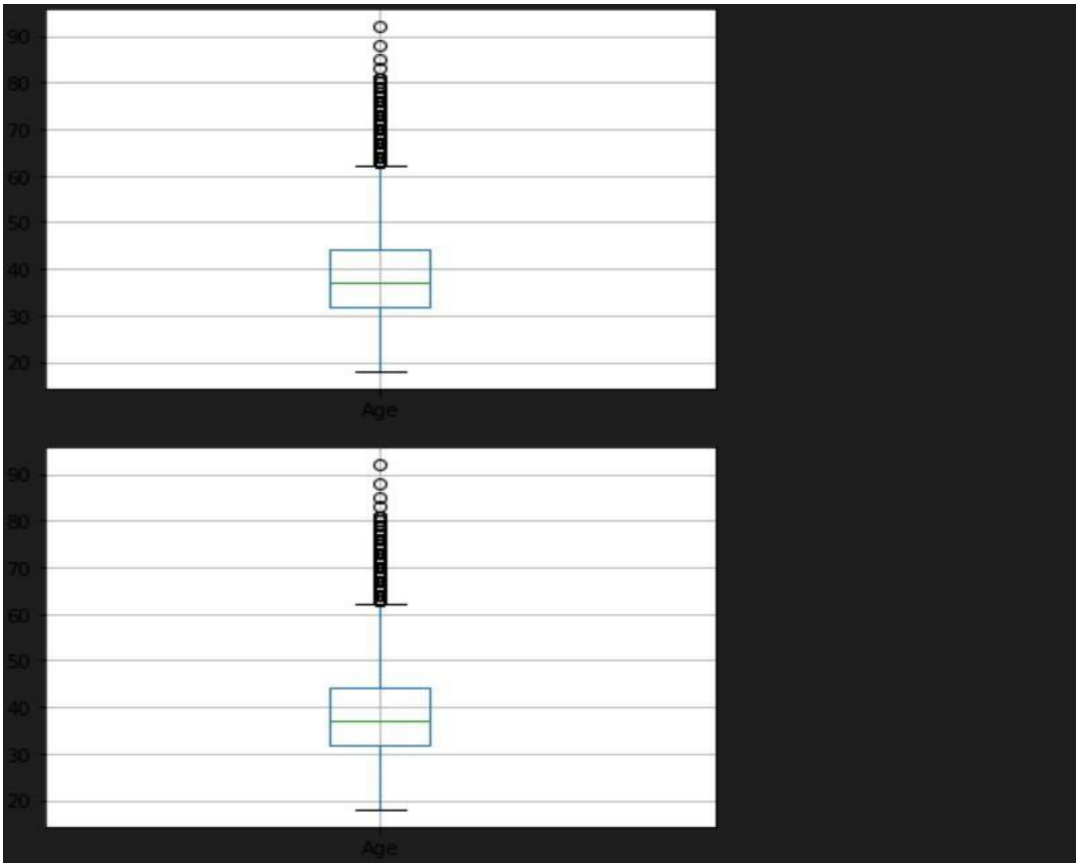
SOLUTION:

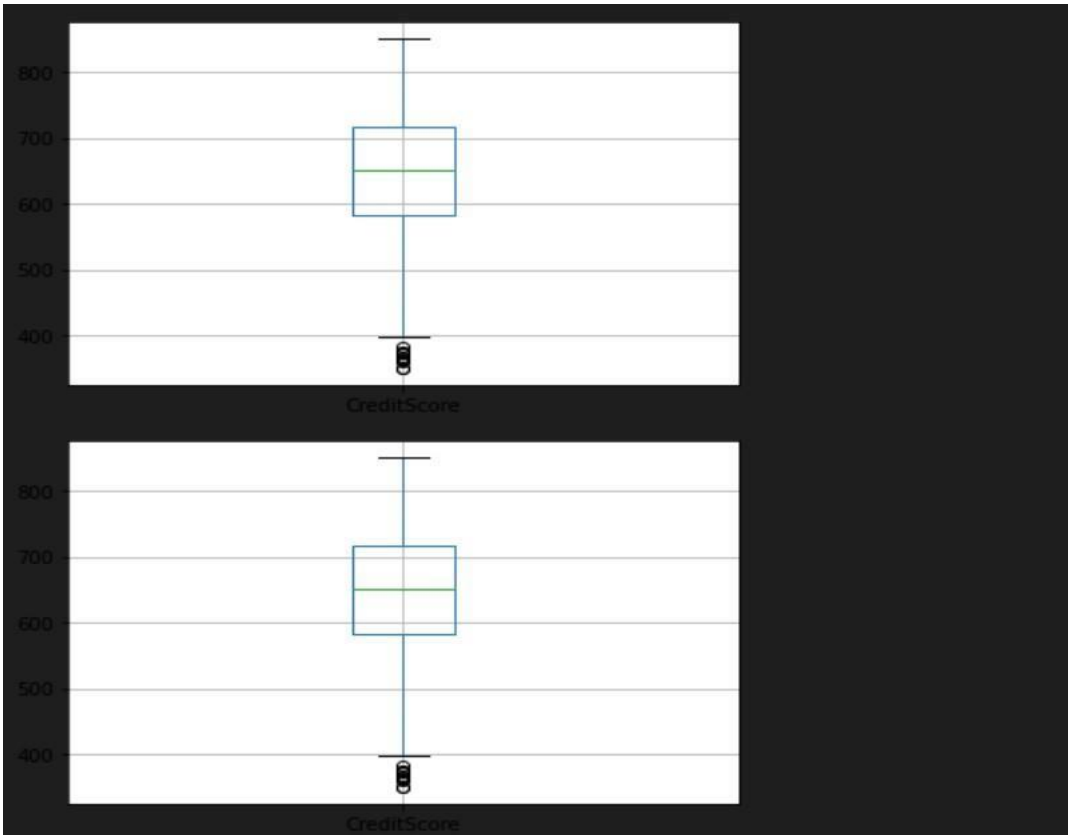
```
df.boxplot(column="EstimatedSalary")
df.boxplot(column="Balance")
df.boxplot(column="Age")
df.boxplot(column="CreditScore")
```

OUTPUT:


200000
175000
150000
125000
100000
75000
50000
25000
0
EstimatedSalary
200000
175000
150000
125000
100000
75000
50000
25000
0


0000
0000
0000
0000
0000
0
Balance
0000
0000
0000
0000
0000
0
Balance


90
80
70
60
50
40
30
20
Age
90
80
70
60
50
40
30
20
Age


800
700
600
500
400
CreditScore
800
700
600
500
400
CreditScore

QUESTION 3:

Perform descriptive statistics on the dataset

SOLUTION:

```
df.describe()
```

OUTPUT:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	1000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	

QUESTION 4:

Handle the Missing values.

SOLUTION:

```
df.isnull().any() df.isna().sum()
df.skew()
```

OUTPUT:

```
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
```

```
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
```

```
RowNumber          0.000000
CustomerId         0.001149
CreditScore       -0.071607
Age                1.011320
Tenure             0.010991
Balance           -0.141109
NumOfProducts      0.745568
HasCrCard         -0.901812
IsActiveMember    -0.060437
EstimatedSalary    0.002085
Exited             1.471611
dtype: float64
```

QUESTION 5:

Find the outliers and replace the outliers **SOLUTION:**

```
out =
df.drop(columns=['Gender','Tenure','HasCrCard,'IsActiveMember','NumOfProducts'
,'Exited'].quantile([q=0.25,0.50]))
Q1=out.iloc[0]
Q3=out.iloc[1] iqr=Q3-
Q1
```

OUTPUT:

```
RowNumber          0.000000
CustomerId         0.001149
CreditScore       -0.071607
Age                1.011320
Tenure             0.010991
Balance           -0.141109
NumOfProducts      0.745568
HasCrCard         -0.901812
IsActiveMember    -0.060437
EstimatedSalary    0.002085
Exited             1.471611
dtype: float64
```

QUESTION 6:

Scale the independent variables **SOLUTION:**

```
ct =
ColumnTransformer([("oh",OneHotEncoder(),[1,2])],remainder="passthrough")
feature_onehot= ct.fit_transform(feature)  feature_onehot
```

OUTPUT:

```
array([[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 1.0134888e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 1.1254258e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
        0.0000000e+00, 9.3826630e+04],
       ...,
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 1.0169977e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 4.2085580e+04],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 9.2888520e+04]])
```

QUESTION 7:

Split the data into training and testing **SOLUTION:**

```
TrainX testX
trainY testY
testX_scale
trainY
```

OUTPUT:

```
array([[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 8.4300480e+04],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 1.4203307e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 1.6737626e+05],
       ...,
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 3.8270470e+04],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 1.1812088e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 9.7755290e+04]])
```

```
array([[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 1.1045799e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 6.3981370e+04],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 1.1343608e+05],
       ...,
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 2.6450570e+04],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 5.4947510e+04],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 1.6318162e+05]])
```

```
Name: Exited, Length: 3994, dtype: int64
```

```
Name: Exited, Length: 999, dtype: int64
```

```
Name: Exited, Length: 3994, dtype: int64
```

```
Name: Exited, Length: 999, dtype: int64
```