Assignment -2

Python Programming

Assignment Date	25 September 2022
Student Name	Earnest Wesley S
Student Roll Number	511919104005
Maximum Marks	2 Marks

Data Visualization and pre-processing

Question-1:

1. Download dataset

Solution:

Dataset has been downloaded successfully

Question-2:

2. Load the dataset

Solution:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
df = pd.read_csv("Churn_Modelling.csv")
df
```

Output:

	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:

3. Perform below visualizations.

Univarient Analysis

Solution-1:

```
df.head()
```

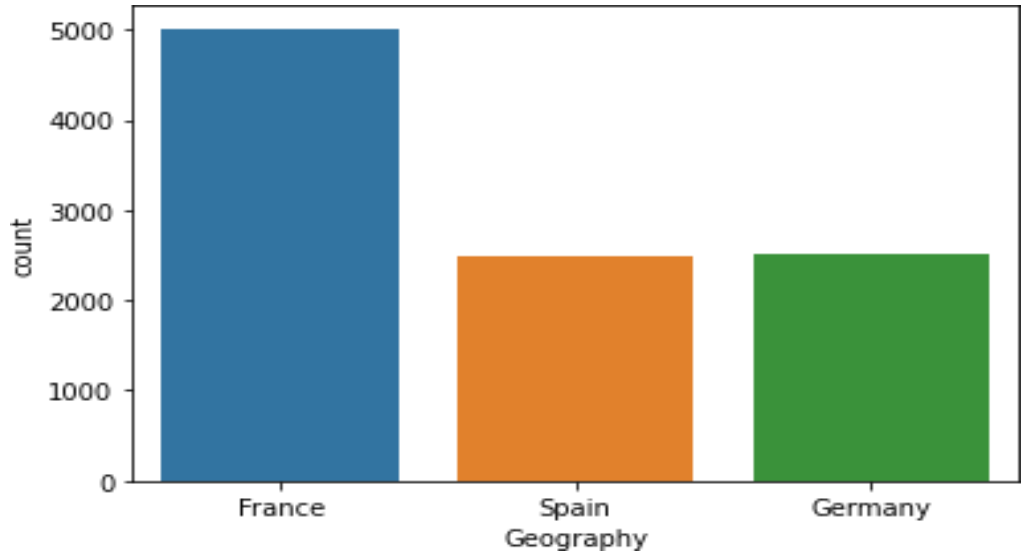
Output:

	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

Solution-2:

```
sns.countplot(x='Geography',data=df)ggfddg
```

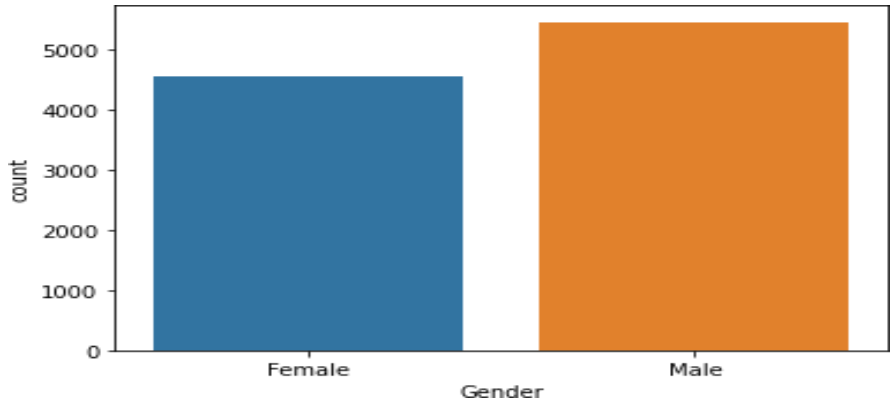
Output:



Solution-3:

```
sns.countplot(x='Gender',data=df)
```

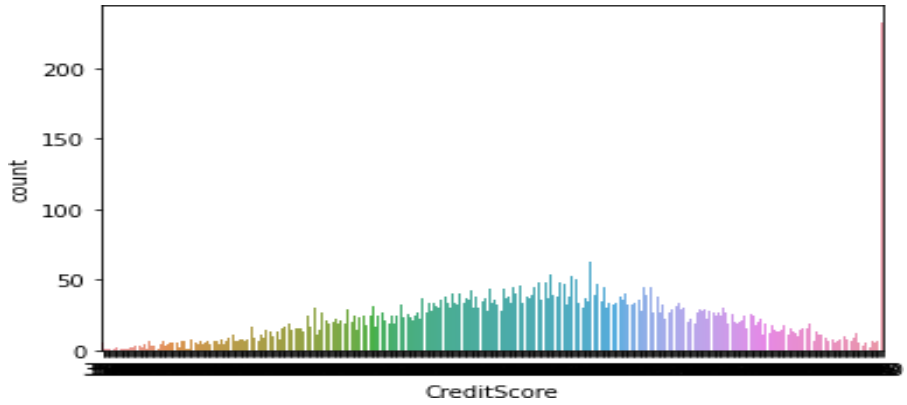
Output:



Solution-4:

```
sns.countplot(x='CreditScore',data=df)
```

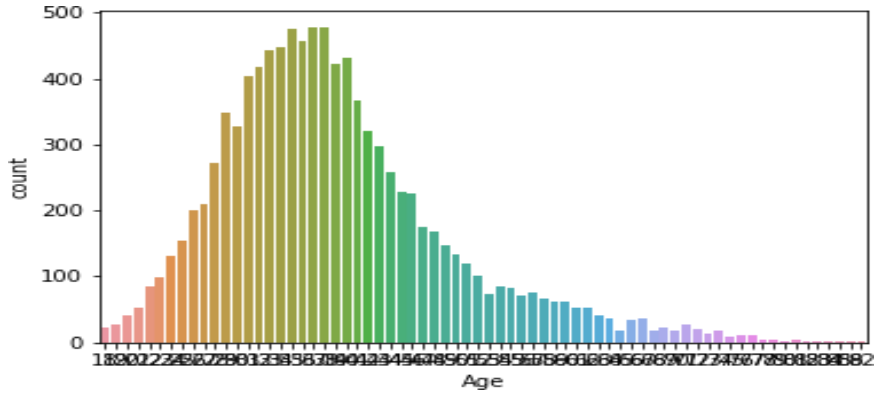
Output:



Solution-5:

```
sns.countplot(x='Age',data=df)
```

Output:

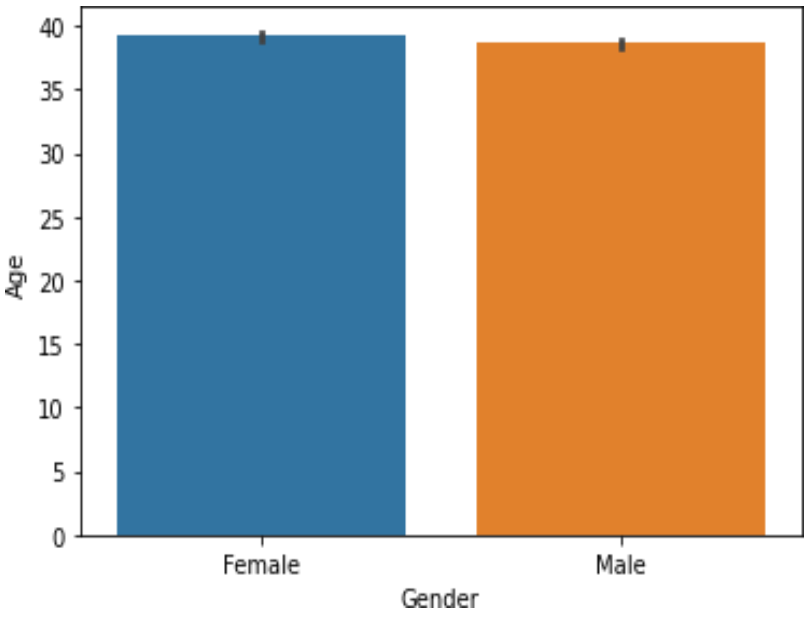


2.2.Bi- variant Analysis

Solution-1:

```
sns.barplot(x='Gender',y='Age',data=df)
```

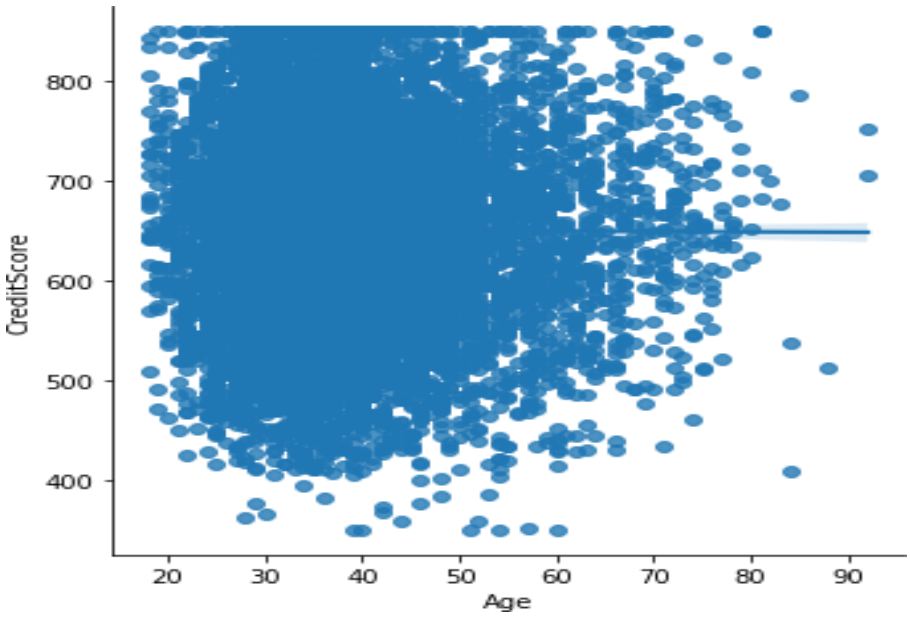
Output:



Solution-2:

```
sns.lmplot(x='Age',y='CreditScore',data=df)
```

Output:



Solution-3:

```
df.info()
```

Output:

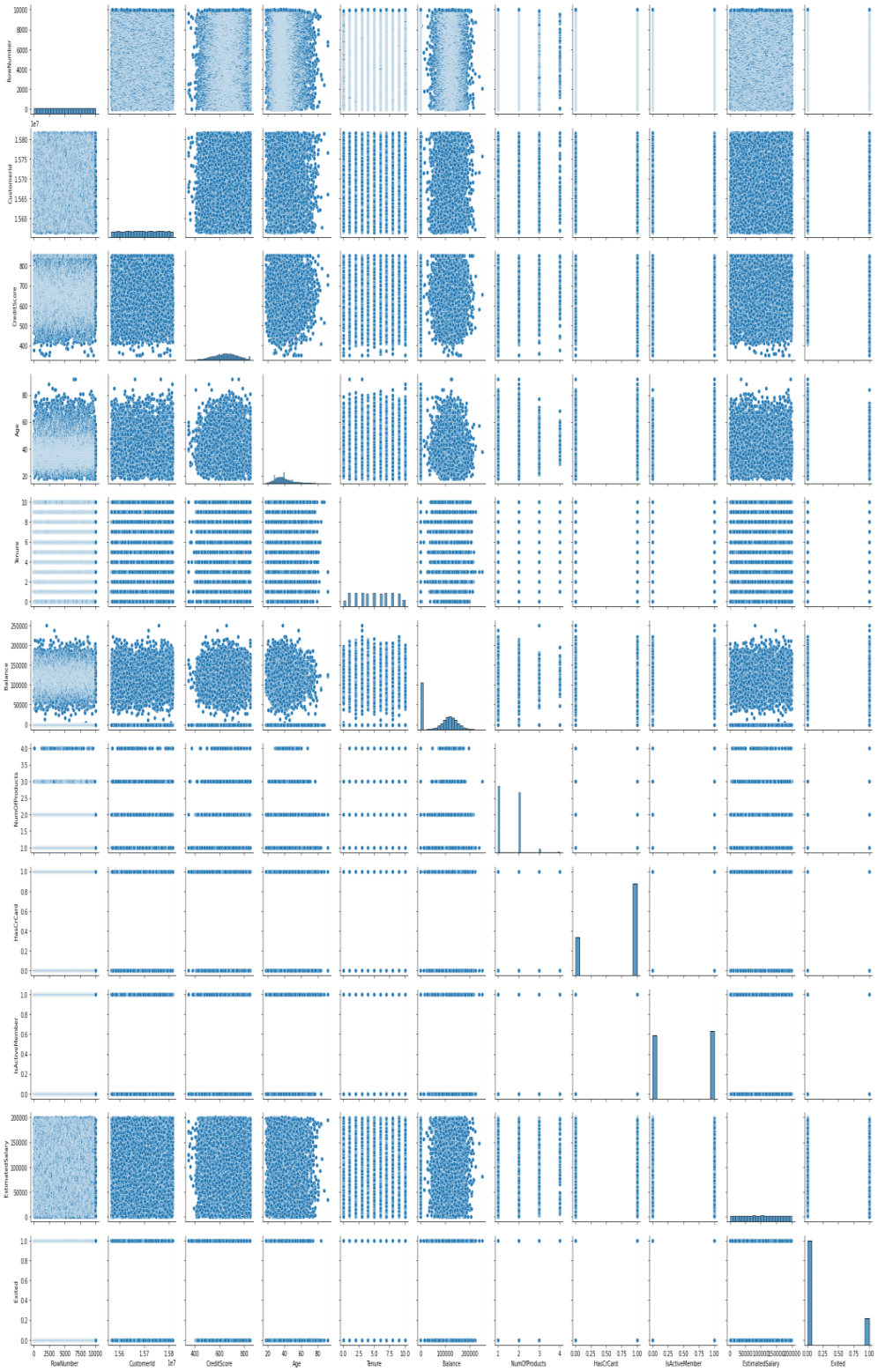
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          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
```

3.3.Multi-variate Analysis

Solution:

```
sns.pairplot(df)
```

Output:



Question-4:

4. Perform descriptive statistics on the dataset

Solution-1:

```
df.describe()
```

Output:

	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

Solution-2:

```
df. describe(include='all')
```

Output:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000	10000.000000	10000	10000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
unique	NaN	NaN	2932	NaN	3	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	NaN	Smith	NaN	France	Male	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	NaN	32	NaN	5014	5457	NaN	NaN	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	100090.239881	0.203700
std	2886.89568	7.193619e+04	NaN	96.653299	NaN	NaN	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	NaN	350.000000	NaN	NaN	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	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	51002.110000	0.000000
50%	5000.50000	1.569074e+07	NaN	652.000000	NaN	NaN	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	NaN	718.000000	NaN	NaN	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	NaN	850.000000	NaN	NaN	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

Question-5:

5. Handle the missing values

Solution-1:

df.isnull()

Output:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9996	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9997	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9998	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9999	False	False	False	False	False	False	False	False	False	False	False	False	False	False

10000 rows × 14 columns

Solution-2:

df.notnull()

Output:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	True	True	True	True	True	True	True	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True	True	True	True	True	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	True	True	True	True	True	True	True	True	True	True	True	True	True	True
9996	True	True	True	True	True	True	True	True	True	True	True	True	True	True
9997	True	True	True	True	True	True	True	True	True	True	True	True	True	True
9998	True	True	True	True	True	True	True	True	True	True	True	True	True	True
9999	True	True	True	True	True	True	True	True	True	True	True	True	True	True

10000 rows × 14 columns

Solution-3:

df.dropna()

Output:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	4	130142.79	1	1	0	38190.78	0

10000 rows × 13 columns

Question-6:

6. Find the outliers and replace the outliers

Solution-1:

```
df.interpolate(method ='linear', limit_direction ='backward', limit = 1)
```

Output:

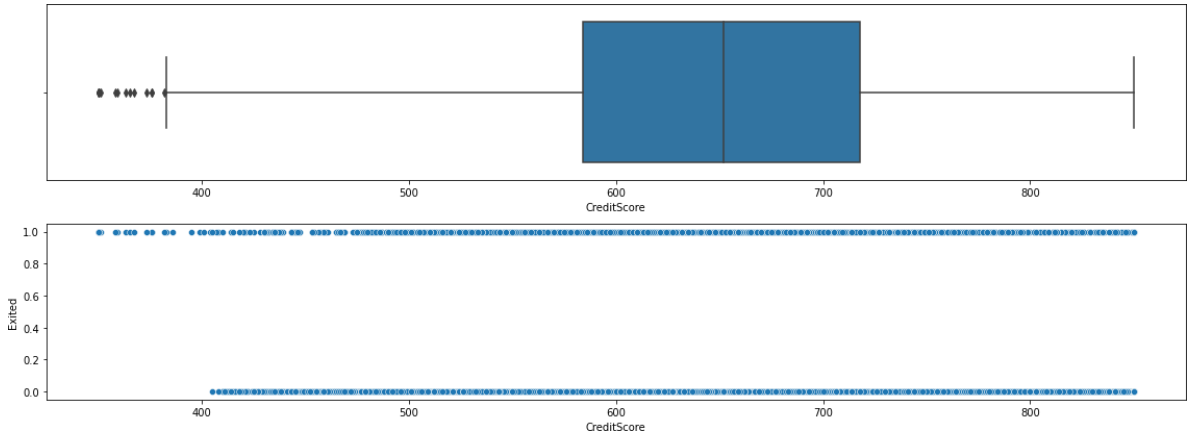
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	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

Solution-2:

```
def box_scatter(data, x, y):
    fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(16,6))
    sns.boxplot(data=data, x=x, ax=ax1)
    sns.scatterplot(data=data, x=x,y=y,ax=ax2)
box_scatter(df,'CreditScore','Exited');
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(df.loc[df['CreditScore'] < 400])}")
```

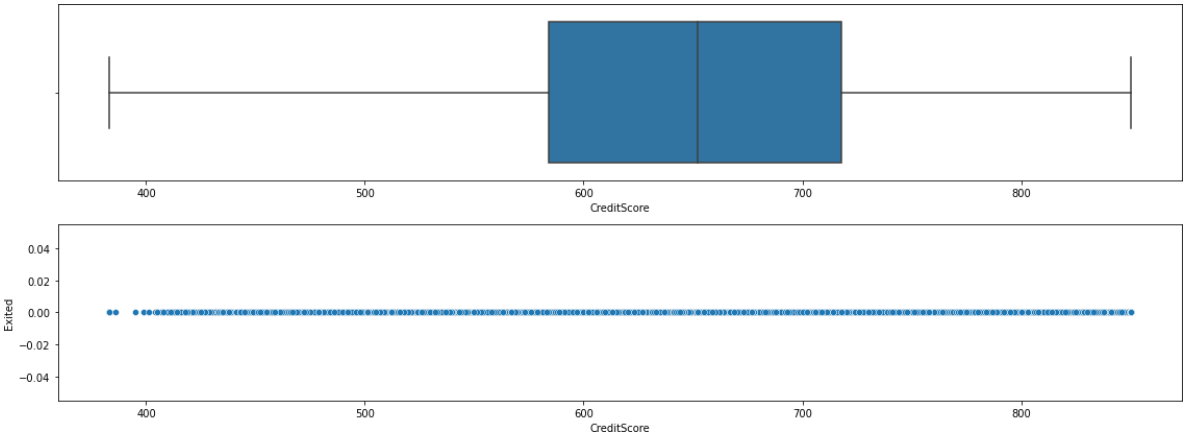
Output:



Solution-3:

```
for i in df:
    if df[i].dtype=='int64' or df[i].dtypes=='float64':
        q1=df[i].quantile(0.25)
        q3=df[i].quantile(0.75)
        iqr=q3-q1
        upper=q3+1.5*iqr
        lower=q1-1.5*iqr
        df[i]=np.where(df[i] >upper, upper, df[i])
        df[i]=np.where(df[i] <lower, lower, df[i])
box_scatter(df,'CreditScore','Exited');
plt.tight_layout()
print(f"# of Bivariate Outliers: {len(df.loc[df['CreditScore'] < 400])}")
```

Output:



Question-7:

7. Check for Categorical columns and perform encoding.

Solution:

```
df.head()
```

Output:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1.0	15634602.0	1115	619.0	0	0	2.0	0.00	1.0	1.0	1.0	101348.88	0.0
1	2.0	15647311.0	1177	608.0	2	0	1.0	83807.86	1.0	0.0	1.0	112542.58	0.0
2	3.0	15619304.0	2040	502.0	0	0	8.0	159660.80	3.0	1.0	0.0	113931.57	0.0
3	4.0	15701354.0	289	699.0	0	0	1.0	0.00	2.0	0.0	0.0	93826.63	0.0
4	5.0	15737888.0	1822	850.0	2	0	2.0	125510.82	1.0	1.0	1.0	79084.10	0.0

Question-8:

8. Split the data into dependent and independent variables.

Solution-1:

```
x=df.iloc[:,:6]
```

```
x.head()
```

Output:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender
0	1.0	15634602.0	1115	619.0	0	0
1	2.0	15647311.0	1177	608.0	2	0
2	3.0	15619304.0	2040	502.0	0	0
3	4.0	15701354.0	289	699.0	0	0
4	5.0	15737888.0	1822	850.0	2	0

Solution-2:

```
Y = df.iloc[:, 6]
print(Y)
```

Output:

```
0        2.0
1        1.0
2        8.0
3        1.0
4        2.0
       ...
9995     5.0
9996    10.0
9997     7.0
9998     3.0
9999     4.0
Name: Tenure, Length: 10000, dtype: float64
```

Solution-3:

```
df.count(0)
```

Output:

```
RowNumber          10000
CustomerId         10000
Surname            10000
CreditScore        10000
Geography          10000
Gender             10000
Tenure             10000
Balance            10000
NumOfProducts      10000
HasCrCard          10000
IsActiveMember     10000
EstimatedSalary    10000
Exited             10000
dtype: int64
```

Question-9:

9. Scale the independent variables

Solution:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x=scaler.fit_transform(x)
print(x)
```

Output:

```
[[-1.73187761 -0.78321342 -0.46418322 -0.32687761 -0.90188624 -1.09598752]
 [-1.7315312  -0.60653412 -0.3909112  -0.44080365  1.51506738 -1.09598752]
 [-1.73118479 -0.99588476  0.62898807 -1.53863634 -0.90188624 -1.09598752]
 ...
 [ 1.73118479 -1.47928179  0.07353887  0.60524449 -0.90188624 -1.09598752]
 [ 1.7315312  -0.11935577  0.98943914  1.25772996  0.30659057  0.91241915]
 [ 1.73187761 -0.87055909  1.4692527   1.4648682  -0.90188624 -1.09598752]]
```

Question-10:

10. Split the data into training and testing

Solution-1:

```
training_data = df.sample(frac=0.8, random_state=25)
testing_data = df.drop(training_data.index)
print(f"No. of training examples: {training_data.shape[0]}")
print(f"No. of testing examples: {testing_data.shape[0]}")
```

Output:

```
No. of training examples: 8000
No. of testing examples: 2000
```

Solution-2:

```
from sklearn.model_selection import train_test_split
training_data, testing_data = train_test_split(df, test_size=0.2, random_state=25)
print(f"No. of training examples: {training_data.shape[0]}")
print(f"No. of testing examples: {testing_data.shape[0]}")
```

Output:

```
No. of training examples: 8000
No. of testing examples: 2000
```