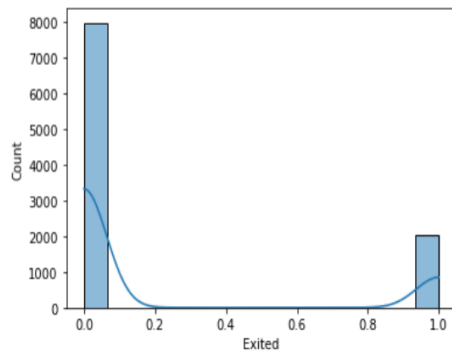


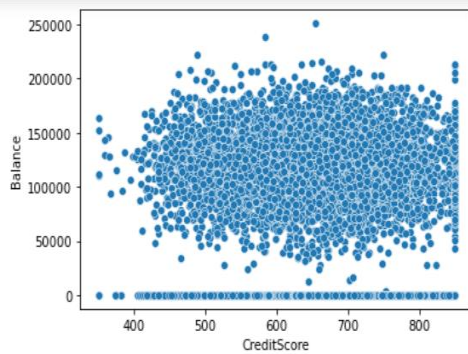

```
In [5]: # Step - 3 : Visualizations
```

```
In [6]: import seaborn  
from matplotlib import pyplot
```

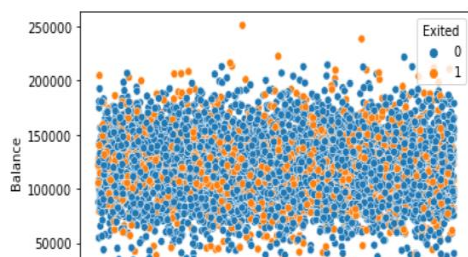
```
In [7]: # Univariate Analysis  
seaborn.histplot(data = data_frame['Exited'], kde = True)  
pyplot.show()
```



```
In [8]: # Bivariate Analysis  
seaborn.scatterplot(data = data_frame, x = 'CreditScore', y = 'Balance')  
pyplot.show()
```



```
In [11]: # Multivariate analysis  
seaborn.scatterplot(data = data_frame, x = 'EstimatedSalary', y = 'Balance', hue = 'Exited')  
pyplot.show()
```



```
In [12]: # Step - 4 : Handle missing values
data_frame.isnull().sum()
```

```
Out[12]: 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
```

```
In [13]: # Step - 5 : checking for categorical columns and performing encoding
data_frame.head(1)
```

```
Out[13]:
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	42	2	0.0	1	1	1	101348.88

```
In [14]: input_data = data_frame.iloc[:, 3:13]
result_data = data_frame.iloc[:, 13:14]
```

```
In [15]: input_data
```

```
Out[15]:
```

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

10000 rows x 10 columns

```
In [16]: result_data
```

Out[16]:

Exited	
0	1
1	0
2	1
3	0
4	0
...	...
9995	0
9996	0
9997	1
9998	1
9999	0

10000 rows × 1 columns

In [17]: *# converting dataframe into arrays*

```
input_data = input_data.values  
result_data = result_data.values
```

In [18]: input_data

Out[18]: array([[619, 'France', 'Female', ..., 1, 1, 101348.88],
[608, 'Spain', 'Female', ..., 0, 1, 112542.58],
[502, 'France', 'Female', ..., 1, 0, 113931.57],
...,
[709, 'France', 'Female', ..., 0, 1, 42085.58],
[772, 'Germany', 'Male', ..., 1, 0, 92888.52],
[792, 'France', 'Female', ..., 1, 0, 38190.78]], dtype=object)

In [19]: result_data

Out[19]: array([[1],
[0],
[1],
...,
[1],
[1],
[0]], dtype=int64)

In [20]: *# finding out unique categorical values*

```
data_frame['Gender'].unique()
```

Out[20]: array(['Female', 'Male'], dtype=object)

In [21]: data_frame['Geography'].unique()

Out[21]: array(['France', 'Spain', 'Germany'], dtype=object)

In [25]: *#Step-6: Applying encoding*

```
In [22]: from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

```
In [23]: ct = ColumnTransformer([("oh",OneHotEncoder(),[1,2])],remainder = "passthrough")
input_data = ct.fit_transform(input_data)
```

```
In [26]: input_data[0:10, 1:2]
```

```
Out[26]: array([[0.0],
               [0.0],
               [0.0],
               [0.0],
               [0.0],
               [0.0],
               [1.0],
               [0.0],
               [0.0]], dtype=object)
```

```
In [27]: input_data.shape
```

```
Out[27]: (10000, 13)
```

```
In [36]: result_data.shape
```

```
Out[36]: (10000, 1)
```

```
In [37]: #Step-7: Splitting data into train and test data
```

```
In [29]: from sklearn.model_selection import train_test_split
input_data_train,input_data_test,result_data_train,result_data_test = train_test_split(input_data,result_data,test_size = 0.3,ran
```

```
In [30]: input_data_train
```

```
Out[30]: array([[1.0, 0.0, 0.0, ..., 1, 1, 55796.83],
               [1.0, 0.0, 0.0, ..., 1, 0, 19823.02],
               [1.0, 0.0, 0.0, ..., 0, 1, 13848.58],
               ...,
               [1.0, 0.0, 0.0, ..., 1, 0, 181429.87],
               [0.0, 0.0, 1.0, ..., 1, 1, 148750.16],
               [0.0, 1.0, 0.0, ..., 1, 0, 118855.26]], dtype=object)
```

```
In [34]: input_data_train.shape
```

```
Out[34]: (7000, 13)
```

```
In [32]: input_data_test
```

```
Out[32]: array([[0.0, 1.0, 0.0, ..., 1, 1, 192852.67],
               [1.0, 0.0, 0.0, ..., 1, 0, 128702.1],
               [0.0, 0.0, 1.0, ..., 1, 1, 75732.25],
               ...,
               [1.0, 0.0, 0.0, ..., 1, 1, 167400.29],
               [1.0, 0.0, 0.0, ..., 1, 1, 70849.47],
               [0.0, 1.0, 0.0, ..., 1, 1, 33759.41]], dtype=object)
```

```
In [33]: input_data_test.shape
```

```
Out[33]: (3000, 13)
```

```
In [38]: result_data_train
```

```
Out[38]: array([[1],
               [0],
               [0],
               ...,
               [0],
               [0],
               [1]], dtype=int64)
```

```
In [39]: result_data_train.shape
```

```
Out[39]: (7000, 1)
```

```
In [40]: result_data_test
```

```
Out[40]: array([[0],
               [1],
               [0],
               ...,
               [0],
               [0],
               [1]], dtype=int64)
```

```
In [41]: result_data_test.shape
```

```
Out[41]: (3000, 1)
```