ASSIGNMENT DATE	24 SEPTEMBER 2022
STUDENT NAME	PRIYADHARSHAN S
STUDENT REGISTER NUMBER	2019504564
MAXIMUM MARKS	2 MARKS

Assignment 2: Data Visualization and Pre-processing

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

2. Load the data set

In [ ]:
```
df = pd.read_csv('Churn_Modelling.csv')
df.head()
```

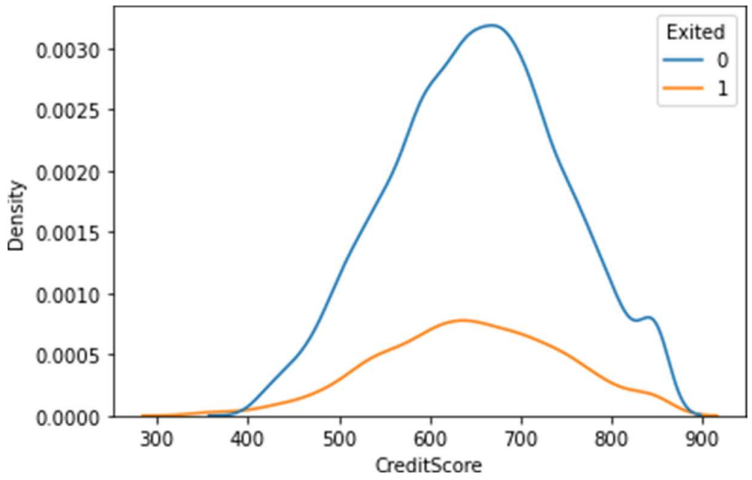
Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estima
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	

3. Data Visualizations

3.1. Univariate Analysis

In [ ]:
```
sns.kdeplot(x='CreditScore', data = df , hue = 'Exited')
plt.show()
```

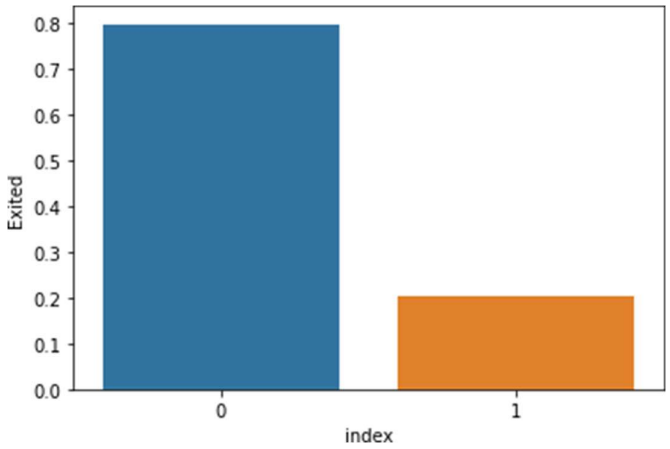


3.2. Bi - Variate Analysis

In [ ]:
```
density  =  df['Exited'].value_counts(normalize=True).reset_index()
sns.barplot(data=density, x='index', y='Exited', );
density
```

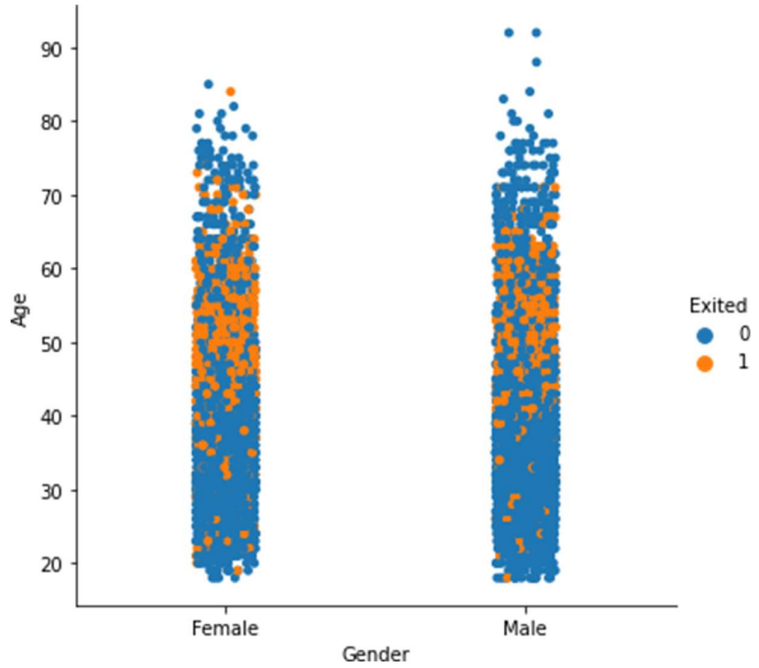
Out[ ]:

	index	Exited
0	0	0.7963
1	1	0.2037



In [ ]:
```
sns.catplot(x='Gender', y='Age', hue='Exited', data=df)
```

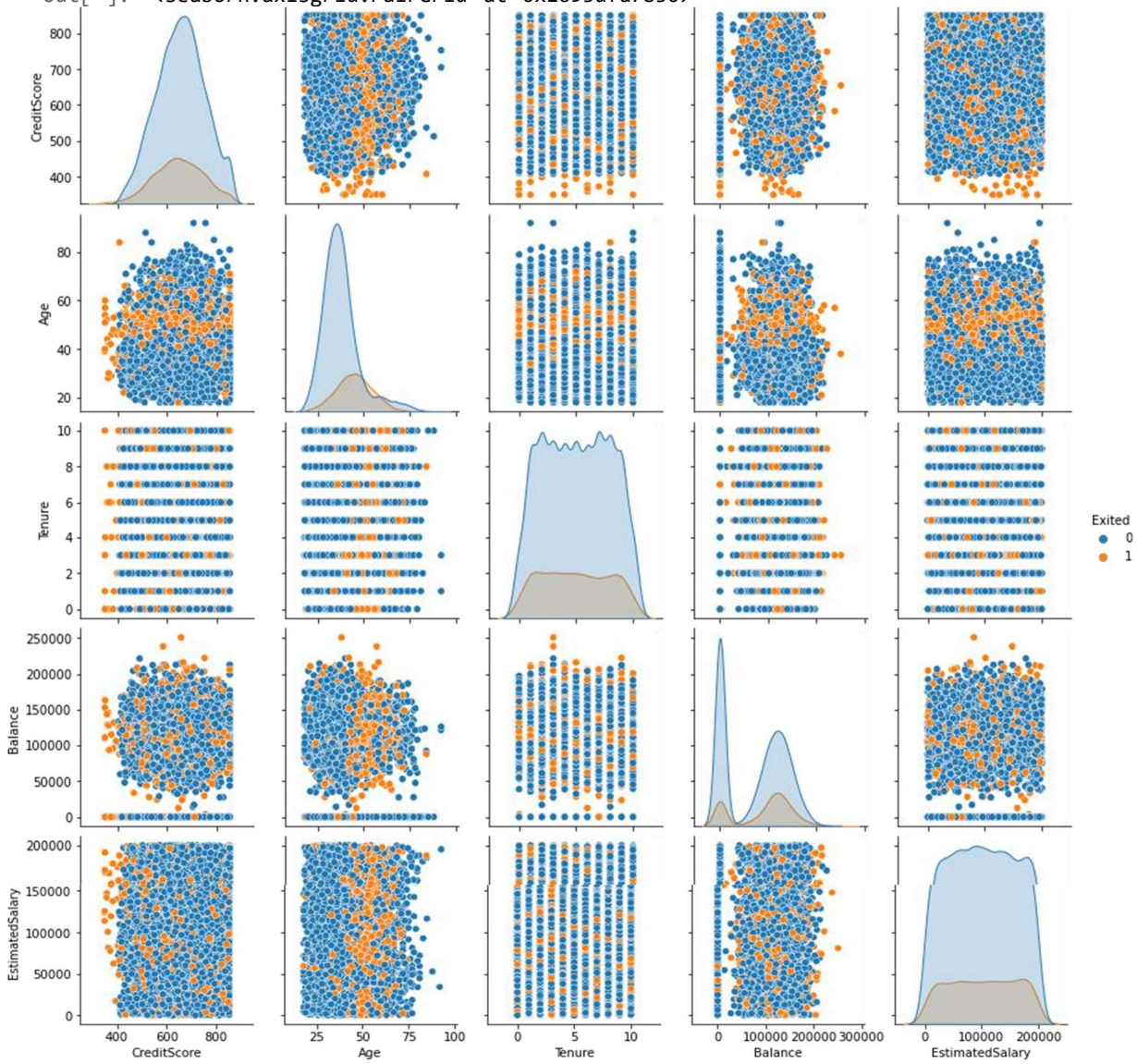
Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1695df8bdc0>
```



3.3. Multi - Variate Analysis

In [ ]:
```
sns.pairplot(data=df[['CreditScore','Age','Tenure','Balance','EstimatedSalary','Exited']],hue='Exited')
```

Out[ ]: <seaborn.axisgrid.PairGrid at 0x1695dfa7850>



4. Descriptive Statistics

In [ ]:
```
df.head()
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estima
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	

In [ ]:
```
df.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
```

In [ ]:
```
df.describe()
```

Out[ ]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedS
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.0
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.2
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.4
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.5
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.1
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.9
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.2
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.4

5. Handle the Missing values

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

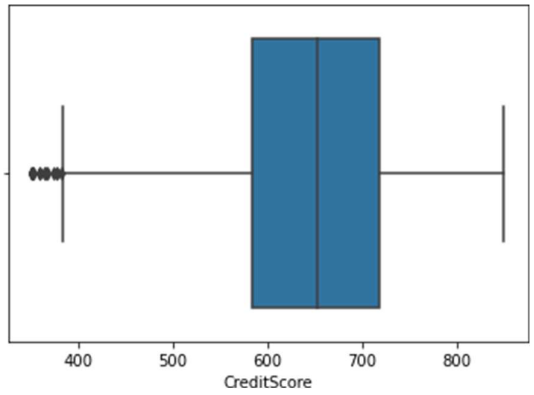
Out[ ]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited         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

The datset does not any missing values, So no need for null value handling!!!

6. Find the outliers and replace the outliers

In [ ]:
```
sns.boxplot(x='CreditScore',data=df)
```

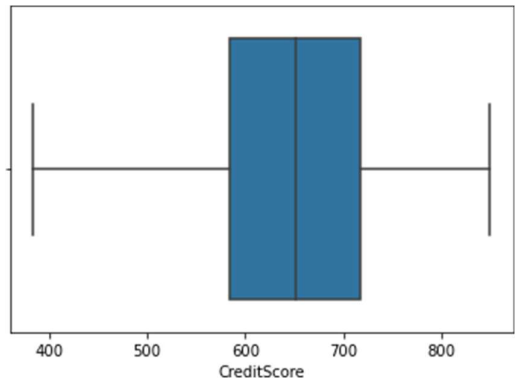
Out[ ]:
```
<AxesSubplot:xlabel='CreditScore'>
```



In [ ]:
```
Q1 = df['CreditScore'].quantile(0.25)
Q3 = df['CreditScore'].quantile(0.75)
IQR = Q3 - Q1
whisker_width = 1.5
lower_whisker = Q1 - (whisker_width*IQR)
upper_whisker = Q3 + (whisker_width*IQR)
df['CreditScore']=np.where(df['CreditScore']>upper_whisker,upper_whisker,np.where(df['CreditScore']<lower_whisker,lower_whisker,
```

In [ ]:
```
sns.boxplot(x='CreditScore',data=df)
```

Out[ ]:
```
<AxesSubplot:xlabel='CreditScore'>
```



7. Check for Categorical columns and perform encoding

In [ ]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
df['Geography'].unique()
ct = ColumnTransformer([('oh', OneHotEncoder(), [4])], remainder="passthrough")
```

8. Split the data into dependent and independent variables.

In [ ]:
```
x=df.iloc[:,0:12].values
x.shape
```

Out[ ]: (10000, 12)

In [ ]:
```
x=df.iloc[:,0:12]
x.shape
x.head()
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
0	1	15634602	Hargrave	619.0	France	Female	42	2	0.00	1	1	1
1	2	15647311	Hill	608.0	Spain	Female	41	1	83807.86	1	0	1
2	3	15619304	Onio	502.0	France	Female	42	8	159660.80	3	1	0
3	4	15701354	Boni	699.0	France	Female	39	1	0.00	2	0	0
4	5	15737888	Mitchell	850.0	Spain	Female	43	2	125510.82	1	1	1

In [ ]:
```
y=df.iloc[:,12:14].values
y.shape
```

Out[ ]:
```
(10000, 2)
```

In [ ]:
```
y=df.iloc[:,12:14]
y.shape
y.head()
```

Out[ ]:

	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1
3	93826.63	0
4	79084.10	0

In [ ]:
```
x=ct.fit_transform(x)
```

In [ ]:
```
print(x.shape)
print(y.shape)
```

```
(10000, 14)
(10000, 2)
```

9. Scale the independent variables

```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
x[:,8:12]=sc.fit_transform(x[:,8:12])
```

10. Split the data into training and testing

In [ ]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25, random_state=0)
```

In [ ]:
```
print(x_train.shape)
print(x_test.shape)
x_train
```

```
(7500, 14)
(2500, 14)
```

Out[ ]:
```
array([[0.0, 1.0, 0.0, ..., 2.5270566192762067, 0, 0],
       [1.0, 0.0, 0.0, ..., 0.8077365626180174, 1, 0],
       [0.0, 0.0, 1.0, ..., 0.8077365626180174,  1, 1],
       ...,
       [1.0, 0.0, 0.0, ..., 0.8077365626180174,  1, 0],
       [0.0, 0.0, 1.0, ..., 0.8077365626180174, 1, 1],
       [0.0, 1.0, 0.0, ..., -0.911583494040172, 1, 0]], dtype=object)
```

In [ ]:
```
print(y_train.shape)
print(y_test.shape)
y_test
```

```
(7500, 2)
(2500, 2)
```

Out[ ]:

	EstimatedSalary	Exited
9394	192852.67	0
898	128702.10	1
2398	75732.25	0
5906	89368.59	0
2343	135662.17	0
...	...	...
8764	86701.40	0
4359	108398.63	0
2041	84487.62	0
1108	46522.68	0
3332	72927.68	0

2500 rows × 2 columns