## Data Visualization and Pre-processing

##Load the Dataset

```
import numpy as np
import pandas as pd

df=pd.read_csv("/content/Churn_Modelling.csv")

df.shape

(10000, 14)

df.head()
```

```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age
\
0          1    15634602  Hargrave          619    France  Female   42

1          2    15647311      Hill          608     Spain  Female   41

2          3    15619304      Onio          502    France  Female   42

3          4    15701354      Boni          699    France  Female   39

4          5    15737888  Mitchell          850     Spain  Female   43


   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

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

##Perform the Visualizations

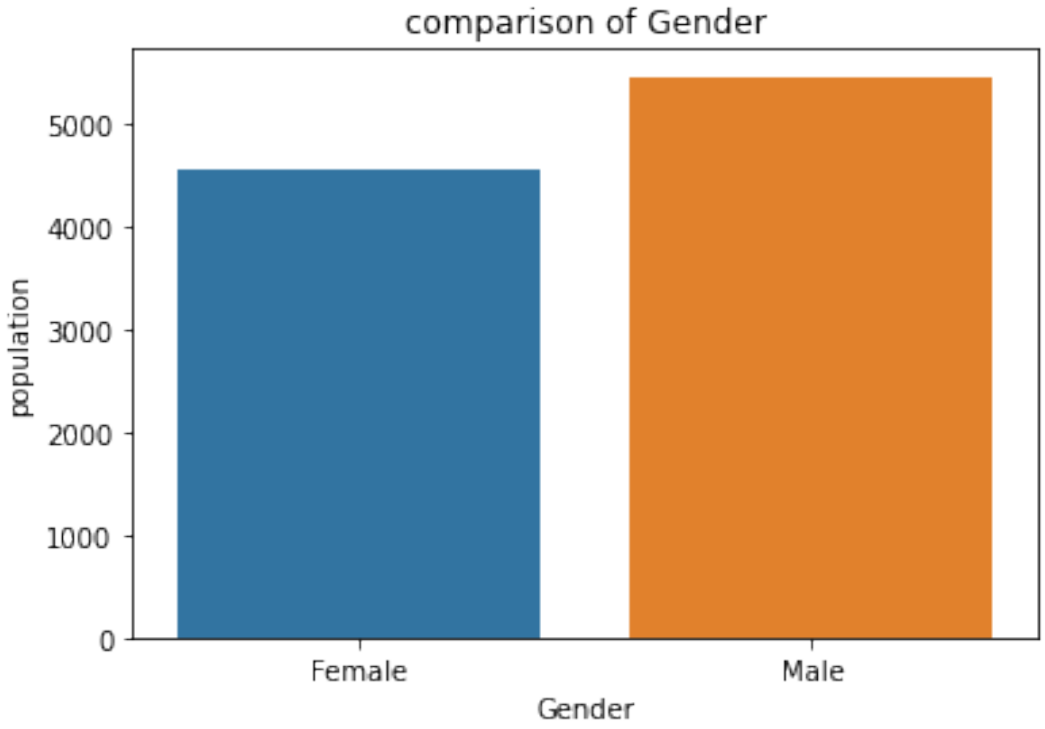
```
#importing essential libraries for visualization
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

###1)Univariate Analysis

```
df['Gender'].value_counts()
```

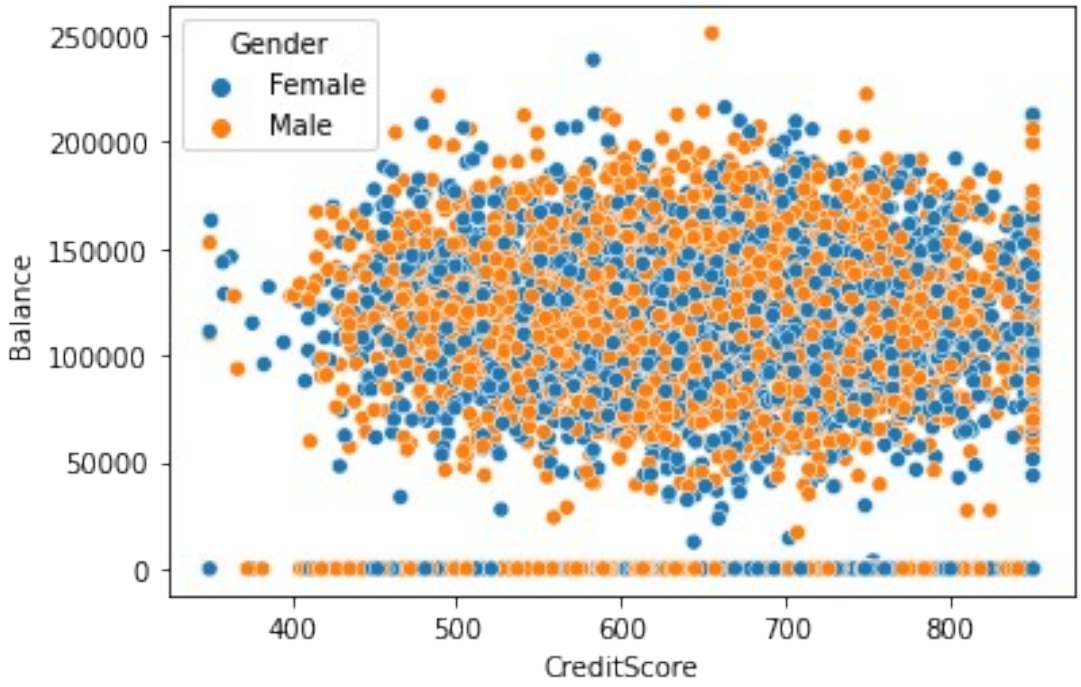
```
Male      5457
Female    4543
Name: Gender, dtype: int64
```

```
sns.countplot(x='Gender',data=df)
plt.title('comparison of Gender')
plt.xlabel('Gender')
plt.ylabel('population')
plt.show()
```

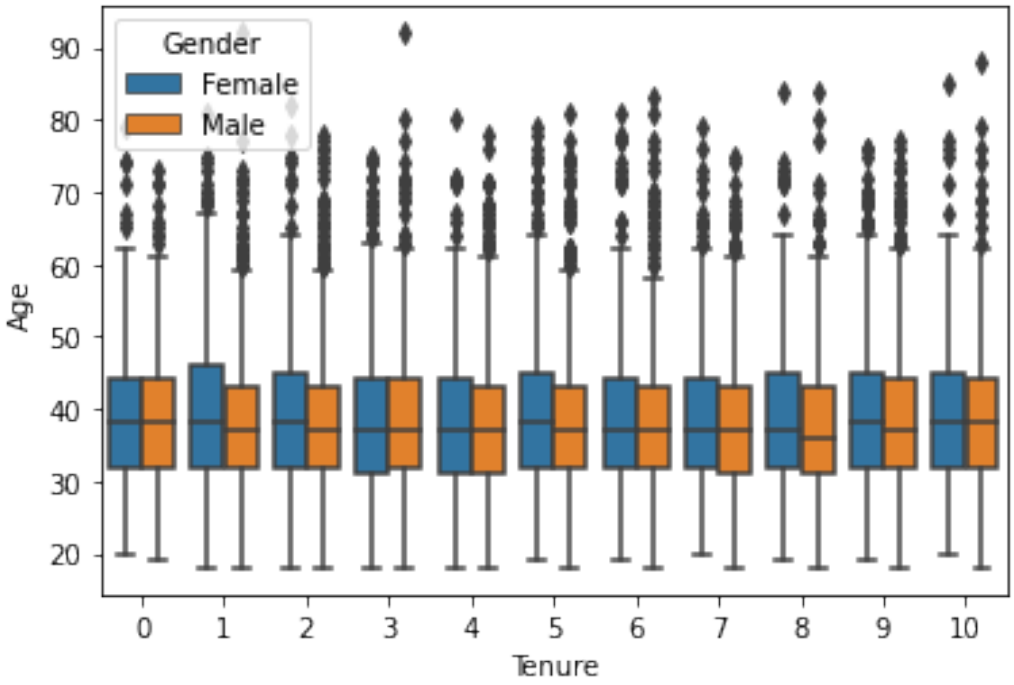



###2)Bi-variate Analysis

```
sns.scatterplot(x='CreditScore',y='Balance',data=df,hue='Gender')
plt.show()
```




```
sns.boxplot(x='Tenure',y='Age',data=df,hue='Gender')
plt.show()
```




###3)Multi-Variate Analysis

```
sns.pairplot(df,hue='Exited')
plt.show()
```


##Descriptive Statistics

```
df.describe()
```

```
         RowNumber    CustomerId   CreditScore           Age
Tenure  \
count  10000.00000  1.000000e+04  10000.000000  10000.000000
10000.000000
mean    5000.50000  1.569094e+07    650.528800     38.921800
5.012800
std     2886.89568  7.193619e+04     96.653299     10.487806
2.892174
min        1.00000  1.556570e+07    350.000000     18.000000
0.000000
25%     2500.75000  1.562853e+07    584.000000     32.000000
3.000000
50%     5000.50000  1.569074e+07    652.000000     37.000000
5.000000
```

```
75%     7500.25000  1.575323e+07    718.000000     44.000000
7.000000
max    10000.00000  1.581569e+07    850.000000     92.000000
10.000000

             Balance  NumOfProducts    HasCrCard  IsActiveMember  \
count   10000.000000   10000.000000  10000.00000    10000.000000
mean    76485.889288       1.530200      0.70550        0.515100
std     62397.405202       0.581654      0.45584        0.499797
min         0.000000       1.000000      0.00000        0.000000
25%         0.000000       1.000000      0.00000        0.000000
50%     97198.540000       1.000000      1.00000        1.000000
75%    127644.240000       2.000000      1.00000        1.000000
max    250898.090000       4.000000      1.00000        1.000000

       EstimatedSalary        Exited
count     10000.000000  10000.000000
mean     100090.239881      0.203700
std       57510.492818      0.402769
min          11.580000      0.000000
25%       51002.110000      0.000000
50%      100193.915000      0.000000
75%      149388.247500      0.000000
max      199992.480000      1.000000
```

##Handling Missing Values

```
df.isna().any()
```

```
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
```

##Find the outliers and replace the outliers

```
df.drop(['RowNumber','CustomerId','Surname'],axis=1,inplace=True)
df.columns
```

```
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure',
'Balance',
       'NumOfProducts', 'HasCrCard', 'IsActiveMember',
'EstimatedSalary',
       'Exited'],
      dtype='object')
```

```
df.head()
```

```
   CreditScore Geography  Gender  Age  Tenure    Balance
NumOfProducts  \
0          619    France  Female   42       2       0.00
1
1          608     Spain  Female   41       1   83807.86
1
2          502    France  Female   42       8  159660.80
3
3          699    France  Female   39       1       0.00
2
4          850     Spain  Female   43       2  125510.82
1

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

## Check for Categorical columns and perform encoding

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.preprocessing import MinMaxScaler

geography=pd.get_dummies(df['Geography'],drop_first=False)
gender=pd.get_dummies(df['Gender'],drop_first=False)

df=pd.concat([df,geography,gender],axis=1)
df.columns
```

```
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure',
'Balance',
       'NumOfProducts', 'HasCrCard', 'IsActiveMember',
'EstimatedSalary',
       'Exited', 'France', 'Germany', 'Spain', 'Female', 'Male'],
      dtype='object')
```

```
df.drop(['Geography','Gender'],axis=1 , inplace=True)

df.head()
```

```
   CreditScore  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0          619   42       2       0.00              1          1
```

```
1          608   41       1   83807.86              1          0
2          502   42       8  159660.80              3          1
3          699   39       1       0.00              2          0
4          850   43       2  125510.82              1          1

   IsActiveMember  EstimatedSalary  Exited  France  Germany  Spain
Female  \
0               1        101348.88       1       1        0      0
1
1               1        112542.58       0       0        0      1
1
2               0        113931.57       1       1        0      0
1
3               0         93826.63       0       1        0      0
1
4               1         79084.10       0       0        0      1
1

   Male
0     0
1     0
2     0
3     0
4     0
```

##Split the data into dependent and independent variables

```
X = df.iloc[:, 0:8].values
Y = df.iloc[:,-6:].values

print(X)
#Independent Features
```

```
[[6.1900000e+02 4.2000000e+01 2.0000000e+00 ... 1.0000000e+00
  1.0000000e+00 1.0134888e+05]
 [6.0800000e+02 4.1000000e+01 1.0000000e+00 ... 0.0000000e+00
  1.0000000e+00 1.1254258e+05]
 [5.0200000e+02 4.2000000e+01 8.0000000e+00 ... 1.0000000e+00
  0.0000000e+00 1.1393157e+05]
 ...
 [7.0900000e+02 3.6000000e+01 7.0000000e+00 ... 0.0000000e+00
  1.0000000e+00 4.2085580e+04]
 [7.7200000e+02 4.2000000e+01 3.0000000e+00 ... 1.0000000e+00
  0.0000000e+00 9.2888520e+04]
 [7.9200000e+02 2.8000000e+01 4.0000000e+00 ... 1.0000000e+00
  0.0000000e+00 3.8190780e+04]]
```

```
print(Y)
#Dependent features
```

```
[[1 1 0 0 1 0]
 [0 0 0 1 1 0]
 [1 1 0 0 1 0]
 ...
 [1 1 0 0 1 0]
 [1 0 1 0 0 1]
 [0 1 0 0 1 0]]
```

##Scale the independent variables

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
print(X)
```

```
[[-0.32622142  0.29351742 -1.04175968 ...  0.64609167  0.97024255
   0.02188649]
 [-0.44003595  0.19816383 -1.38753759 ... -1.54776799  0.97024255
   0.21653375]
 [-1.53679418  0.29351742  1.03290776 ...  0.64609167 -1.03067011
   0.2406869 ]
 ...
 [ 0.60498839 -0.27860412  0.68712986 ... -1.54776799  0.97024255
  -1.00864308]
 [ 1.25683526  0.29351742 -0.69598177 ...  0.64609167 -1.03067011
  -0.12523071]
 [ 1.46377078 -1.04143285 -0.35020386 ...  0.64609167 -1.03067011
  -1.07636976]]
```

##Split the data into training and testing

```
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, random_state = 0)

print('X_train size:{},X-test size:
{}'.format(X_train.shape,X_test.shape))
print('Y_train size:{},Y-test size:
{}'.format(Y_train.shape,Y_test.shape))
```

```
X_train size:(8000, 8),X-test size:(2000, 8)
Y_train size:(8000, 6),Y-test size:(2000, 6)
```