Date: 25.09.2022

Name:Nandhini B

Roll No:737819ITR050

Assignment-2 Data Visualization and Preprocessing

1) Download the dataset from the source link

About the dataset: This dataset is all about churn modelling of a credit company. It has the details about the end user who are using credit card and also it has some variables to depicit thechurn of the customer.

- **RowNumber** - Serial number of the rows
- **CustomerId** - Unique identification of customer
- **Surname** - Name of the customer
- **CreditScore** - Cipil score of the customer
- **Geography** - Location of the bank **Gender** -
- Sex of the customer
- **Age** - Age of the customer
- **Tenure** - Repayment period for the credit amount
- **Balance** - Current balance in thier creidt card
- **NumOfProducts** - Products owned by the customer from the company
- **HasCrCard** - Has credit card or not (0 - no , 1 - yes)
- **IsactiveMember** - Is a active member or not List item
- **EstimatedSalary** - Salary of the customer
- **Exited** - Churn of the customer

```
import warnings
warnings.filterwarnings("ignore")
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

2) Loading the dataset

```
df = pd.read_csv("Churn_Modelling.csv")
```

```
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Ba
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	838
2	3	15619304	Onio	502	France	Female	42	8	1596
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	1255

3) Performing Visualizations

Univariate Analysis

```
#checking for categorical variables
category = df.select_dtypes(include=[np.object])
print("Categorical Variables: ",category.shape[1])

#checking for numerical variables
numerical = df.select_dtypes(include=[np.int64,np.float64])
print("Numerical Variables: ",numerical.shape[1])
```

```
Categorical Variables:  3
Numerical Variables:  11
```

```
df.columns
```

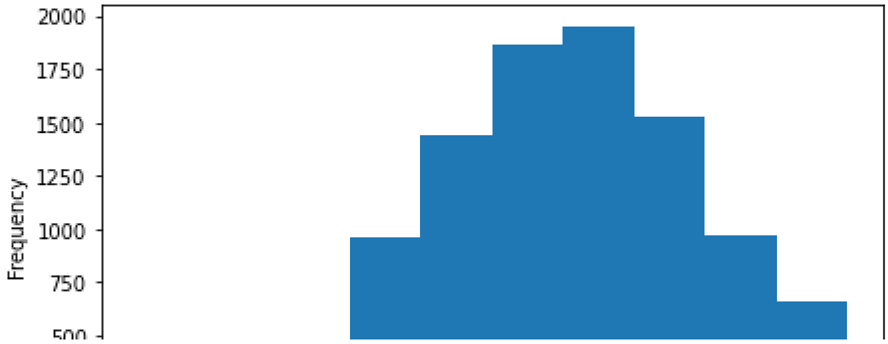
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

```
df.shape
```

```
(10000, 14)
```

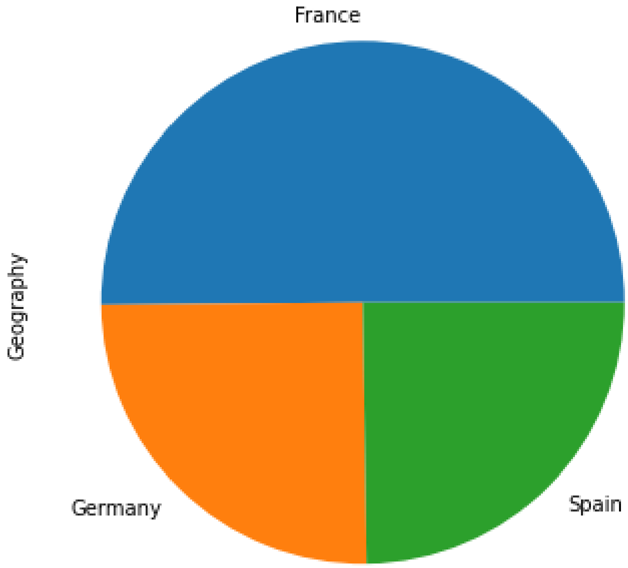
```
credit = df['CreditScore']
credit.plot(kind="hist",figsize=(7,4))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc976a55210>
```



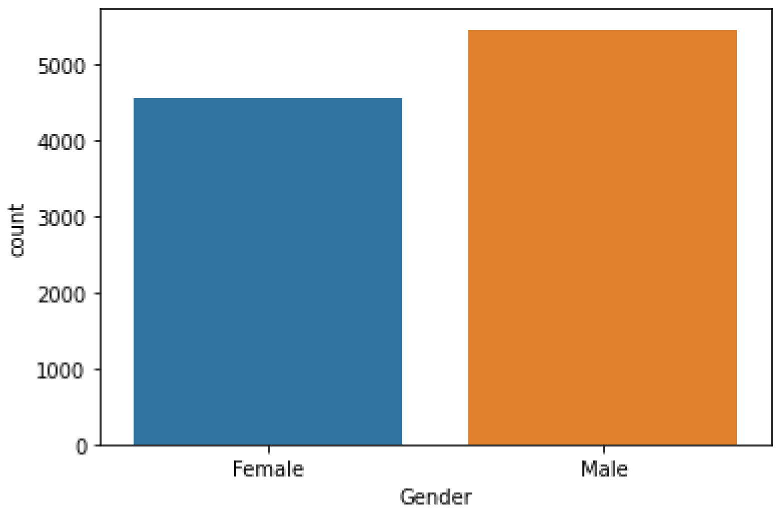

```
geo = df['Geography'].value_counts()
geo.plot(kind="pie",figsize=(8,6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc97696e110>
```



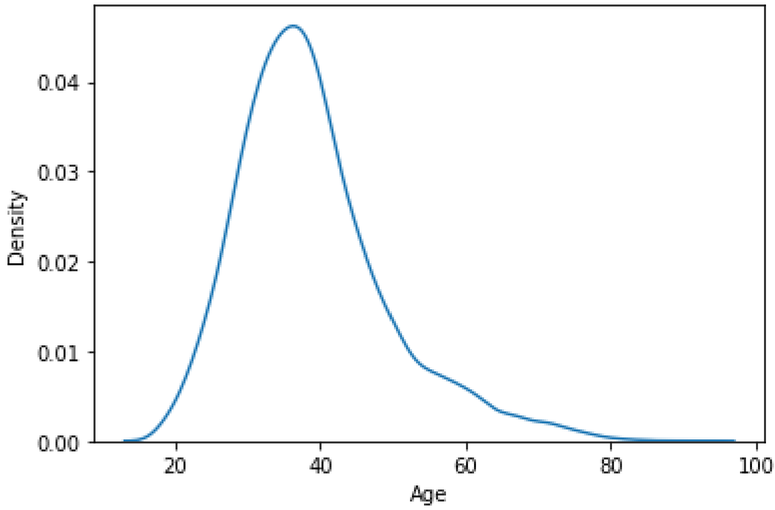

```
sns.countplot(df['Gender'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9769e2dd0>
```



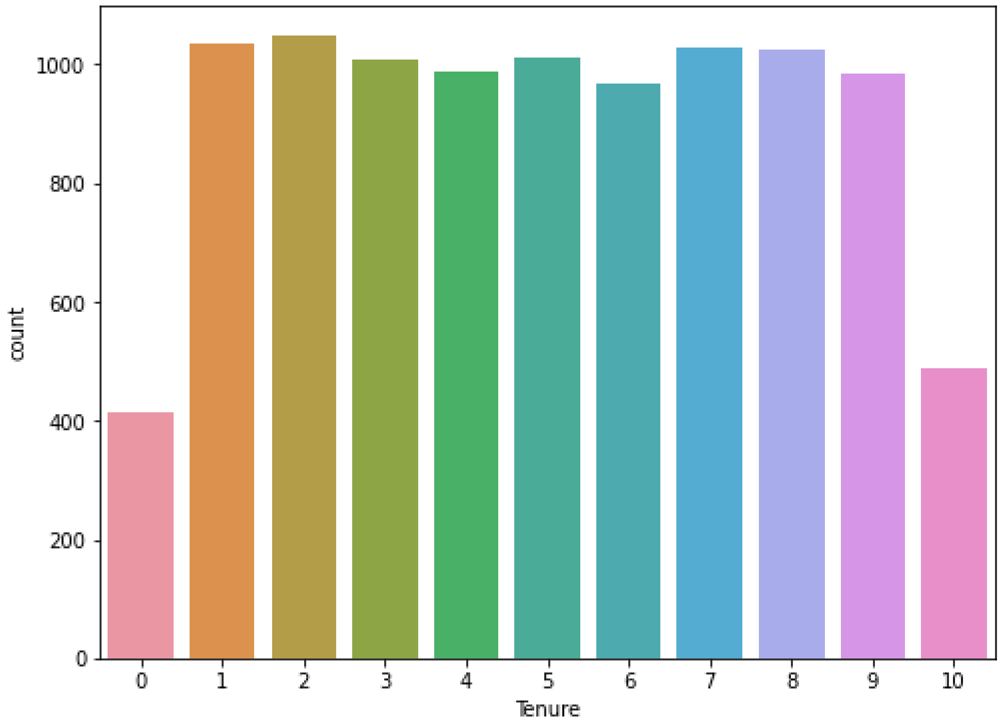

```
sns.distplot(df['Age'],hist=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc97692be90>
```



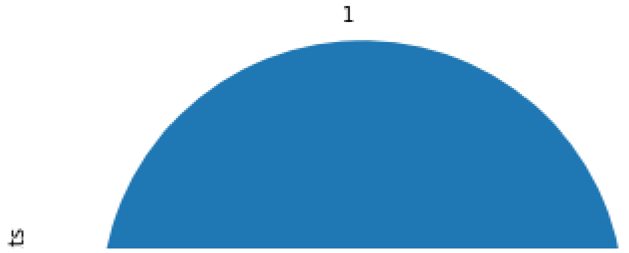

```
plt.figure(figsize=(8,6))
sns.countplot(df['Tenure'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9767ab590>
```



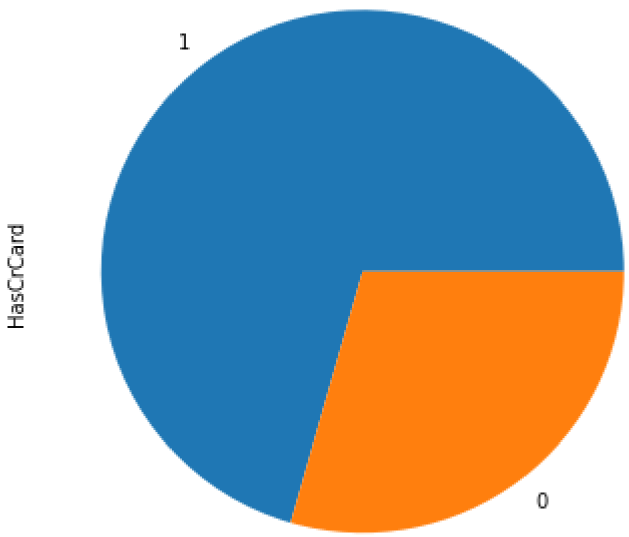

```
product = df['NumOfProducts'].value_counts()
product.plot(kind="pie",figsize=(8,6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9767abf50>
```



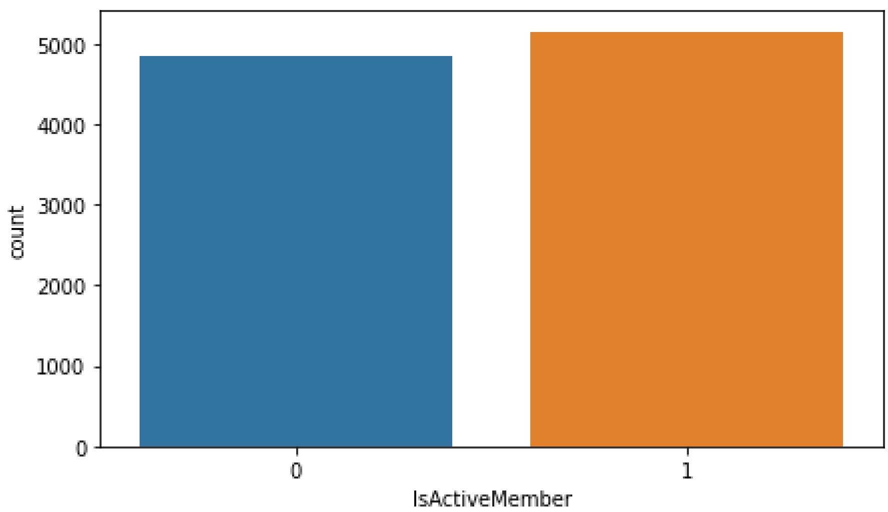

```
cr = df['HasCrCard'].value_counts()
cr.plot(kind="pie",figsize=(8,6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc97671b3d0>
```



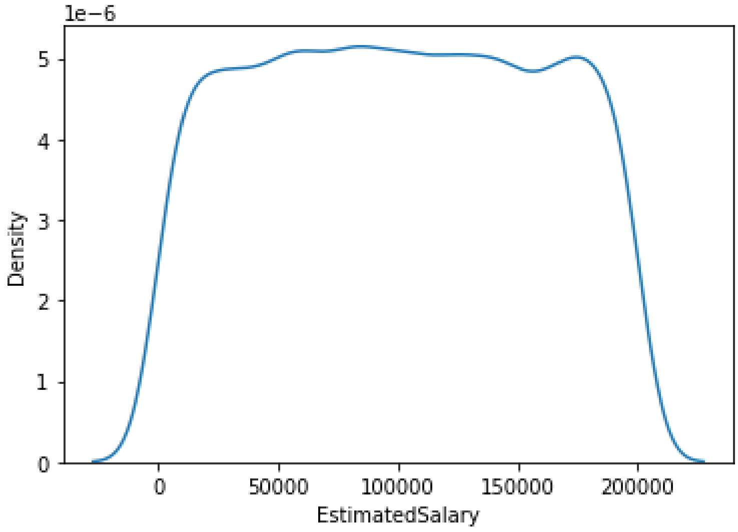

```
plt.figure(figsize=(7,4))
sns.countplot(df['IsActiveMember'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc976746d90>
```



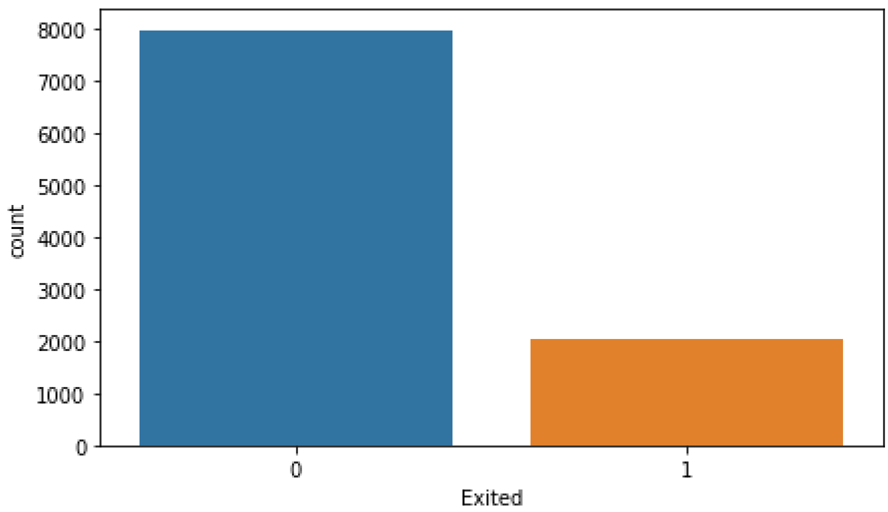

```
sns.distplot(df['EstimatedSalary'],hist=False)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc976642510>




```
plt.figure(figsize=(7,4))
sns.countplot(df['Exited'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc97691ae50>



Inference:

1. The data has 11 numerical variables and 3 categorical variables.
2. It has 10000 rows and 14 columns
3. The normalized credit score is around 700, More than 500 people have credit score greaterthan 800.
4. France occupies 50% of customers, where as Germany and Spain shared equal.
5. Dataset is dominated by Male Customers.
6. Median age is around 40 to 45.
7. Highest number of customer has thier tenure period for 2 years.
8. Credit company has maximum customers, who uses single product.
9. Most of the customer has credit card.

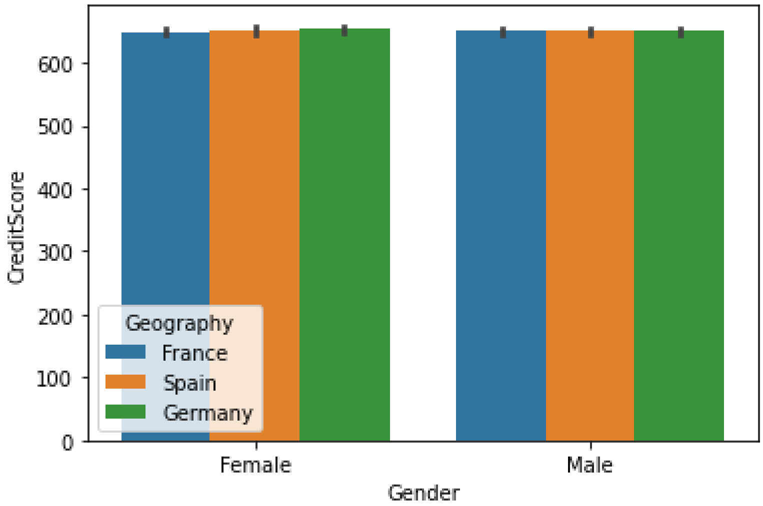
10. More than 40% of the population is not an active member.

11. The Churn is less compared to the satisfaction. Dataset is imbalanced.

Bi-Variate Analysis

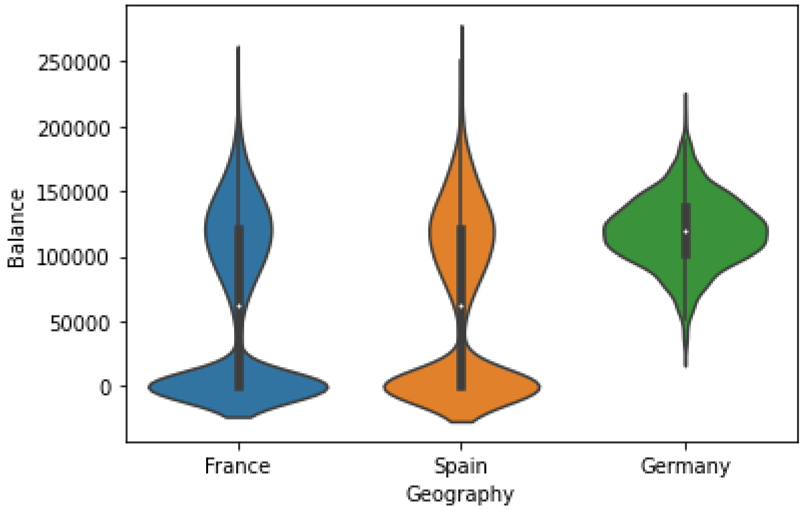
```
sns.barplot(x='Gender',y='CreditScore',hue='Geography',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc97674d0d0>
```



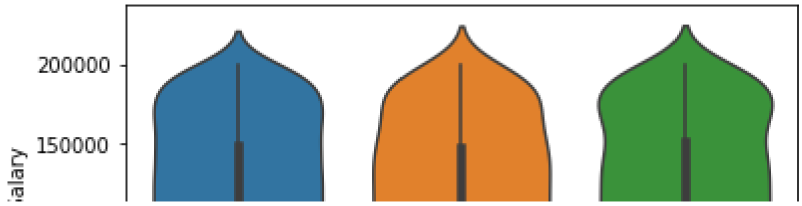

```
sns.violinplot(x='Geography',y='Balance',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc976558550>
```



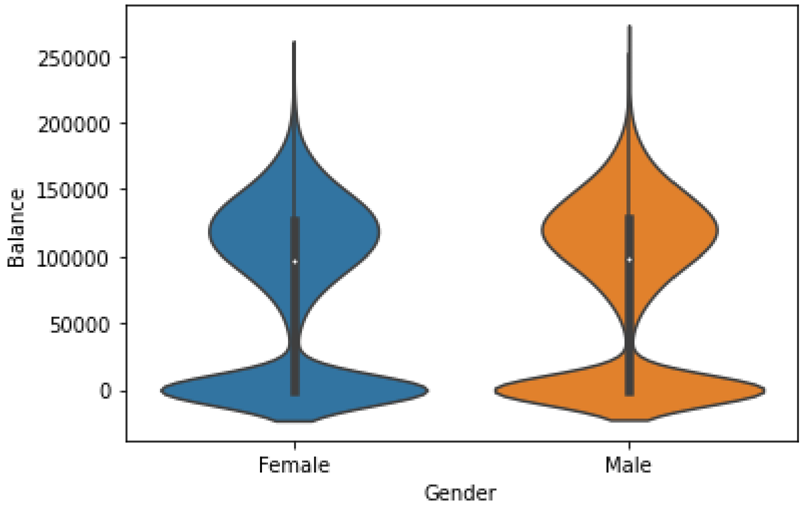

```
sns.violinplot(x='Geography',y='EstimatedSalary',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc97647be50>
```



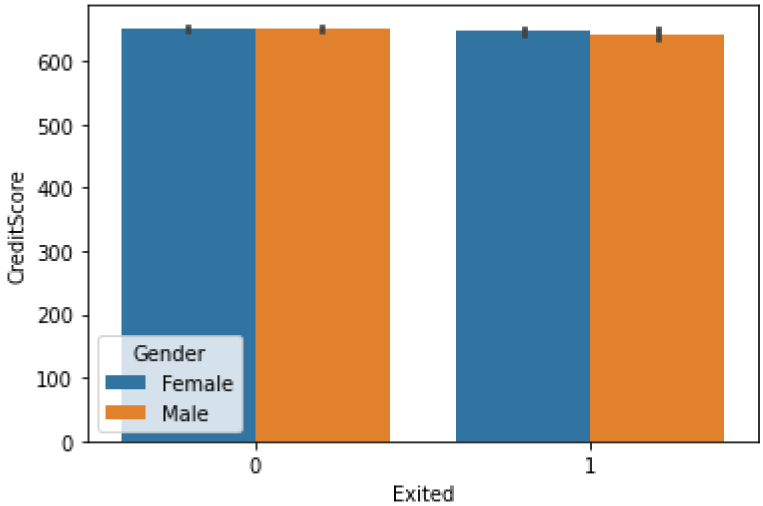

```
sns.violinplot(x='Gender',y='Balance',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9763f41d0>
```



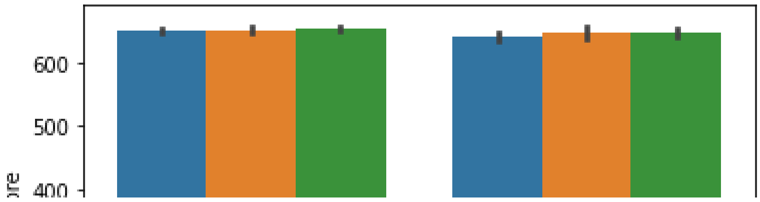

```
sns.barplot(x='Exited',y='CreditScore',hue='Gender',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc976403e90>
```



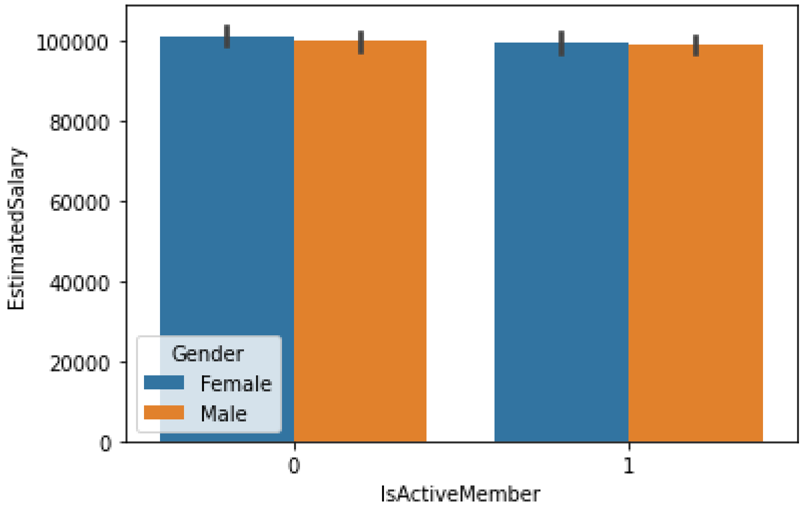

```
sns.barplot(x='Exited',y='CreditScore',hue='Geography',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9763d1090>
```



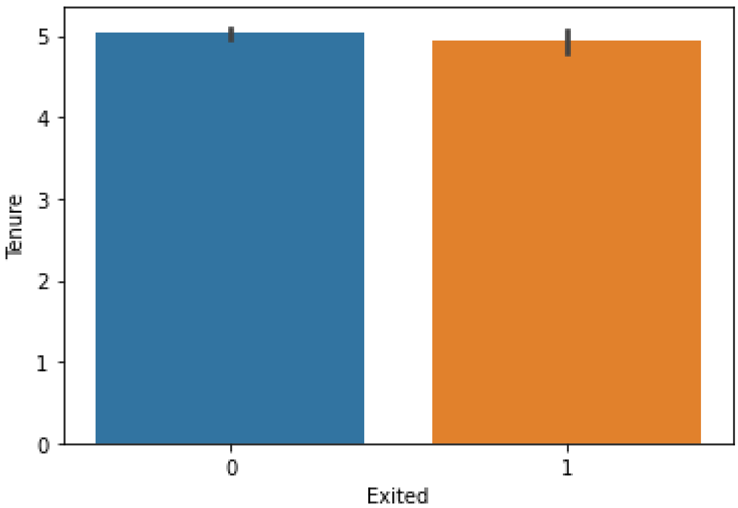

```
sns.barplot(x='IsActiveMember',y='EstimatedSalary',hue='Gender',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc97626f4d0>
```




```
sns.barplot(x='Exited',y='Tenure',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9761f6d10>
```



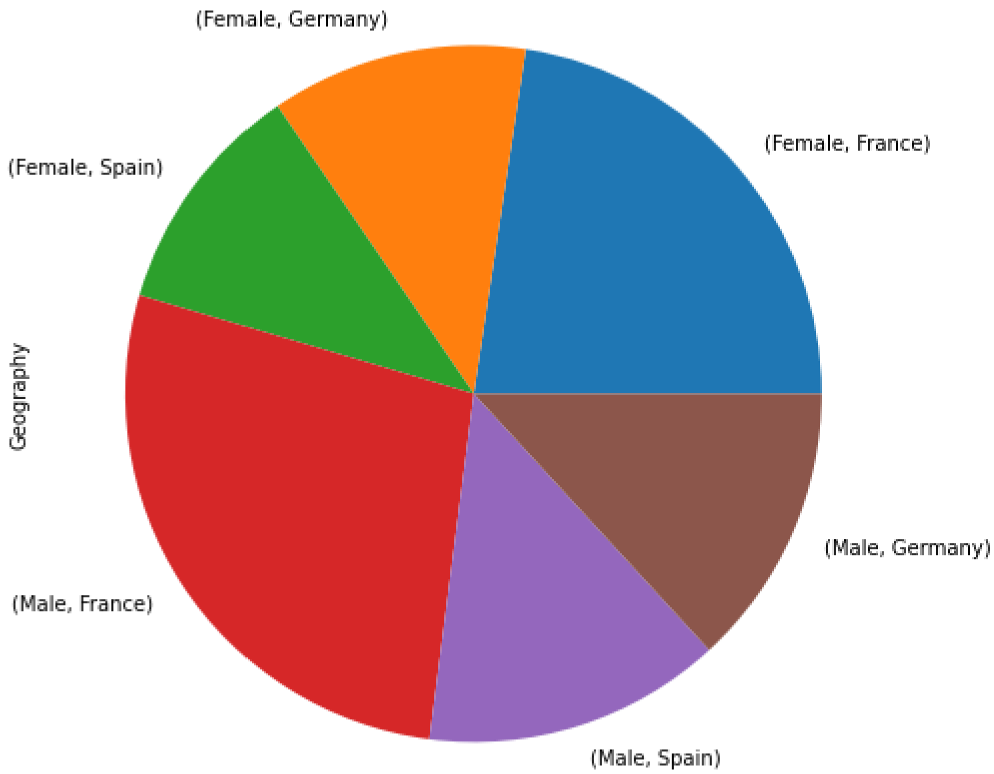
Inference:

1. Credit score for Male is higher in Spain.
2. Average bank salary lies in the range of 100k to 150k.
3. Estimated salary is normalized and same for all country.
4. Credit score for churn is low.
5. Churn in Germany is higher compared to other countries.
6. Exited people tenure period is around 6 years.

Multi-Variate Analysis

```
gp1 = df.groupby('Gender')['Geography'].value_counts()
gp1.plot(kind='pie',figsize=(10,8))
print(gp1)
```

```
Gender   Geography
Female   France       2261
         Germany      1193
         Spain        1089
Male     France       2753
         Spain        1388
         Germany      1316
Name: Geography, dtype: int64
```




```
gp2 = df.groupby('Gender')['Age'].mean()
print(gp2)
```

```
Gender
Female    39.238389
Male      38.658237
Name: Age, dtype: float64
```

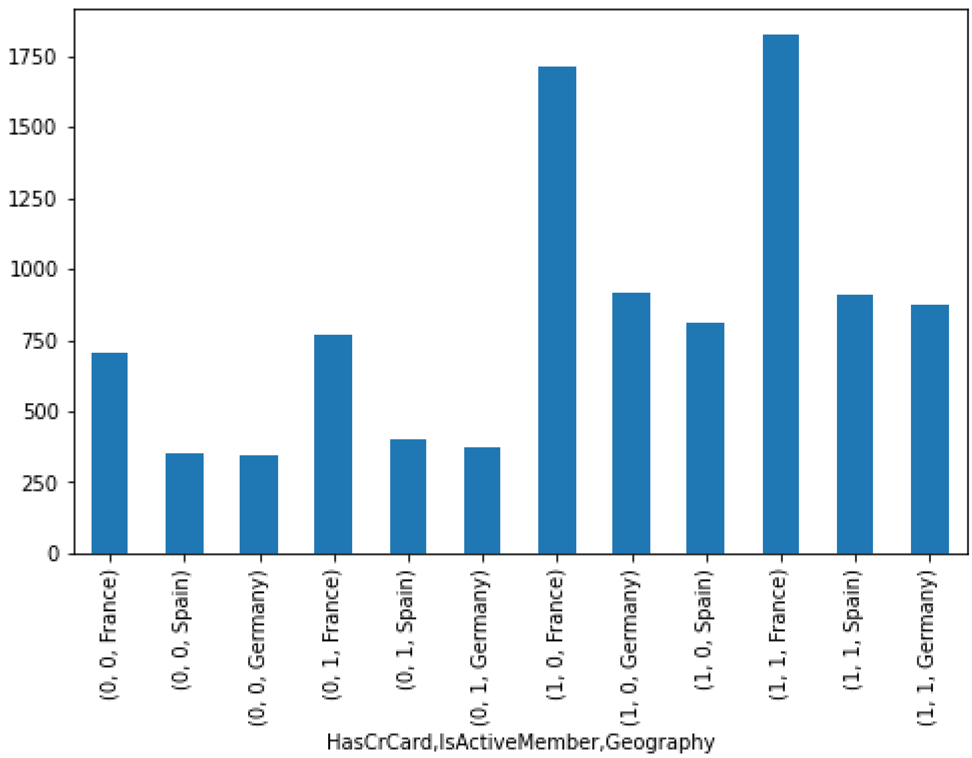
```
gp3 = df.groupby(['Gender','Geography'])['Tenure'].mean()
print(gp3)
```

```
Gender   Geography
Female   France       4.950022
         Germany      4.965633
```

```
        Spain        5.000000
Male    France       5.049401
        Germany      5.050152
        Spain        5.057637
Name: Tenure, dtype: float64
```

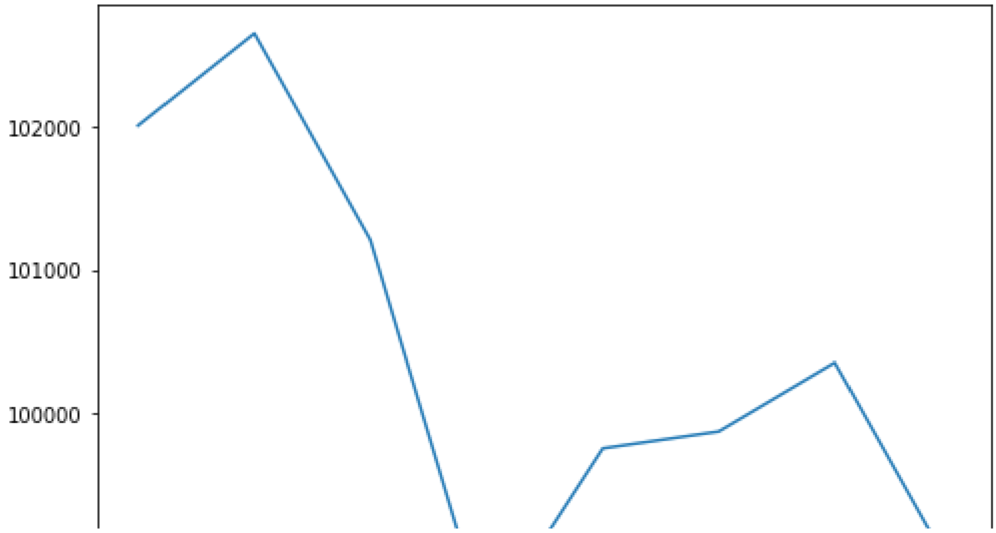
```
gp4 = df.groupby(['HasCrCard','IsActiveMember'])['Geography'].value_counts()
gp4.plot(kind="bar",figsize=(8,5))
print(gp4)
```

```
HasCrCard  IsActiveMember  Geography
0          0               France        706
                           Spain         352
                           Germany       343
           1               France        765
                           Spain         404
                           Germany       375
1          0               France       1717
                           Germany       918
                           Spain         813
           1               France       1826
                           Spain         908
                           Germany       873
Name: Geography, dtype: int64
```




```
gp5 = df.groupby(['Gender','HasCrCard','IsActiveMember'])['EstimatedSalary'].mean()
gp5.plot(kind="line",figsize=(8,6))
print(gp5)
```

```
Gender  HasCrCard  IsActiveMember
Female  0          0                 102006.080352
                   1                 102648.996944
        1          0                 101208.014567
                   1                  98510.152300
Male    0          0                  99756.431151
                   1                  99873.931251
        1          0                 100353.378996
                   1                  98914.378703
Name: EstimatedSalary, dtype: float64
```




```
gp6 = df.groupby(['Gender','IsActiveMember'])['Exited'].value_counts()
gp6.plot(kind='bar',figsize=(8,6))
print(gp6)
```

```
Gender  IsActiveMember  Exited
Female  0               0         1534
                        1          725
        1               0         1870
                        1          414
Male    0               0         2013
```

```
gp7 = df.groupby('Exited')['Balance','EstimatedSalary'].mean()
print(gp7)
```

```
              Balance  EstimatedSalary
Exited
0        72745.296779     99738.391772
1        91108.539337    101465.677531
```


```
gp8 = df.groupby(['Geography','Exited'])['Gender'].value_counts()
gp8.plot(kind='bar',figsize=(10,8))
print (gp8)
```

```
Geography  Exited  Gender
France     0       Male      2403
                   Female    1801
           1       Female     460
                   Male       350
Germany    0       Male       950
                   Female     745
           1       Female     448
                   Male       366
```

Inference:

1. Germany has more female customers compared to male customers.
2. Average age of Male is 38, whereas average age of Female is 39.
3. Tenure period for both male and female is high in Spain.
4. It is observed that, those who have credit card are very active member in the company.
5. The estimated salary for a person who is not having credit card is high when compared tothose having them.
6. Churn for inactive member is high compared to active member.
7. Those who churn has thier estimated salary very low.
8. France has the more churn rate.

4) Performing descriptive statistics on dataset

```
df.describe().T
```

	count	mean	std	min	25%	
RowNumber	10000.0	5.000500e+03	2886.895680	1.00	2500.75	5.00050
CustomerId	10000.0	1.569094e+07	71936.186123	15565701.00	15628528.25	1.56907
CreditScore	10000.0	6.505288e+02	96.653299	350.00	584.00	6.52000
Age	10000.0	3.892180e+01	10.487806	18.00	32.00	3.70000
Tenure	10000.0	5.012800e+00	2.892174	0.00	3.00	5.00000
Balance	10000.0	7.648589e+04	62397.405202	0.00	0.00	9.71985
NumOfProducts	10000.0	1.530200e+00	0.581654	1.00	1.00	1.00000
HasCrCard	10000.0	7.055000e-01	0.455840	0.00	0.00	1.00000
IsActiveMember	10000.0	5.151000e-01	0.499797	0.00	0.00	1.00000
EstimatedSalary	10000.0	1.000902e+05	57510.492818	11.58	51002.11	1.00193
Exited	10000.0	2.037000e-01	0.402769	0.00	0.00	0.00000

5) Handle the Missing values

```
df.isnull().sum()
```

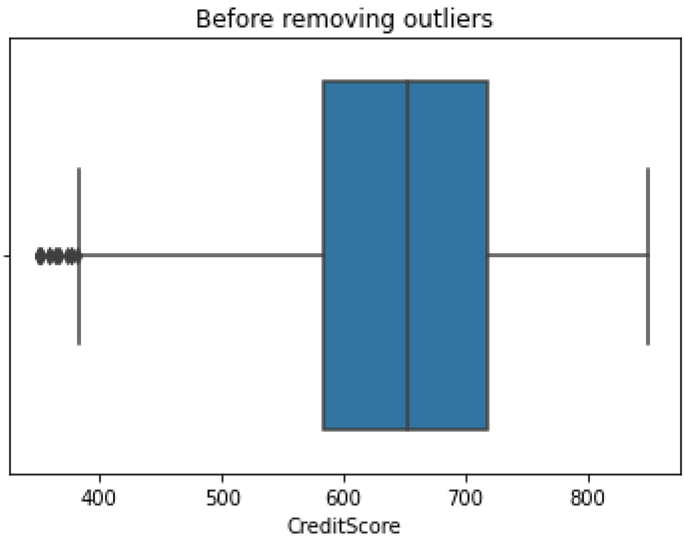
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: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

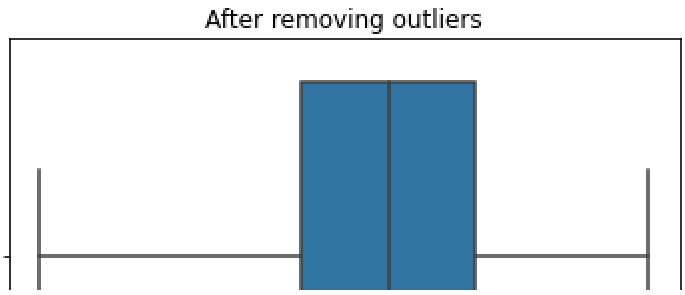
There is no missing value in dataset

6) Finding the outliers and replacing it

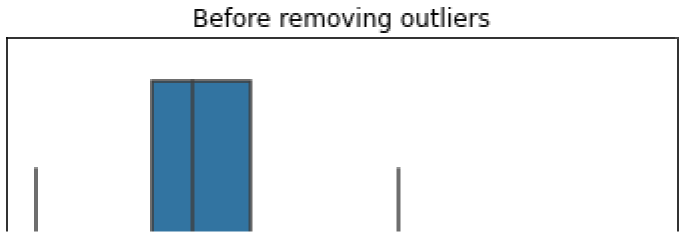
```
def replace_outliers(df, field_name):
    Q1 = np.percentile(df[field_name],25,interpolation='midpoint')
    Q3 = np.percentile(df[field_name],75,interpolation='midpoint')
    IQR = Q3-Q1
    maxi = Q3+1.5*IQR
    mini = Q1-1.5*IQR
    df[field_name]=df[field_name].mask(df[field_name]>maxi,maxi)
    df[field_name]=df[field_name].mask(df[field_name]<mini,mini)
```

```
plt.title("Before removing outliers")
sns.boxplot(df['CreditScore'])
plt.show()
plt.title("After removing outliers")
replace_outliers(df, 'CreditScore')
sns.boxplot(df['CreditScore'])
plt.show()
```



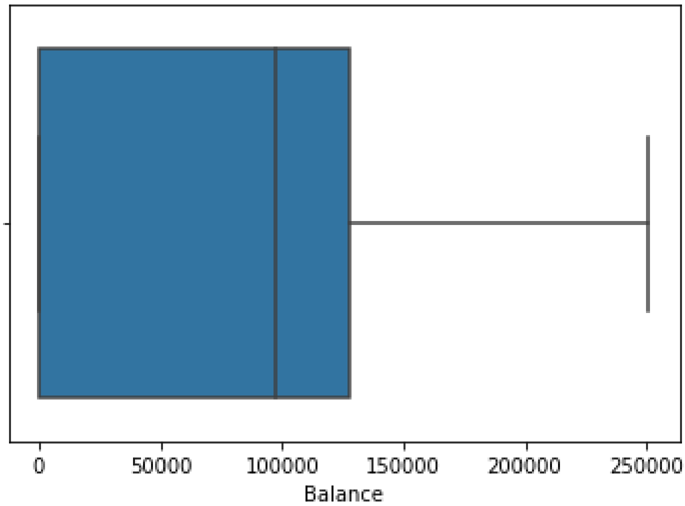



```
plt.title("Before removing outliers")
sns.boxplot(df['Age'])
plt.show()
plt.title("After removing outliers")
replace_outliers(df, 'Age')
sns.boxplot(df['Age'])
plt.show()
```



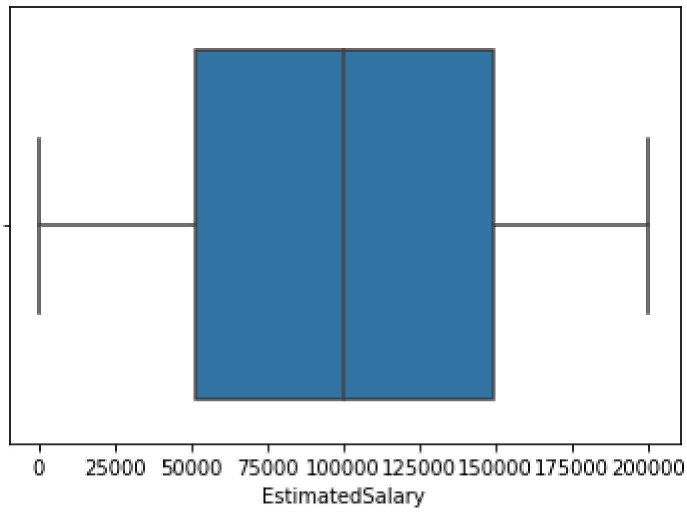

```
sns.boxplot(df['Balance'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc95ba3cf50>



```
sns.boxplot(df['EstimatedSalary'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc95b94a790>



The Outliers from Age and Credit Score columns are removed.

7) Check for categorical column and perform encoding

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
df['Gender'] = le.fit_transform(df['Gender'])
df['Geography'] = le.fit_transform(df['Geography'])
```

```
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Ba
0	1	15634602	Hargrave	619.0	0	0	42.0	2	
1	2	15647311	Hill	608.0	2	0	41.0	1	83
2	3	15619304	Onio	502.0	0	0	42.0	8	159
3	4	15701354	Boni	699.0	0	0	39.0	1	
4	5	15737888	Mitchell	850.0	2	0	43.0	2	125

Only two columns Gender and Geography is label encoded.

Removing unwanted columns and checking for feature importance

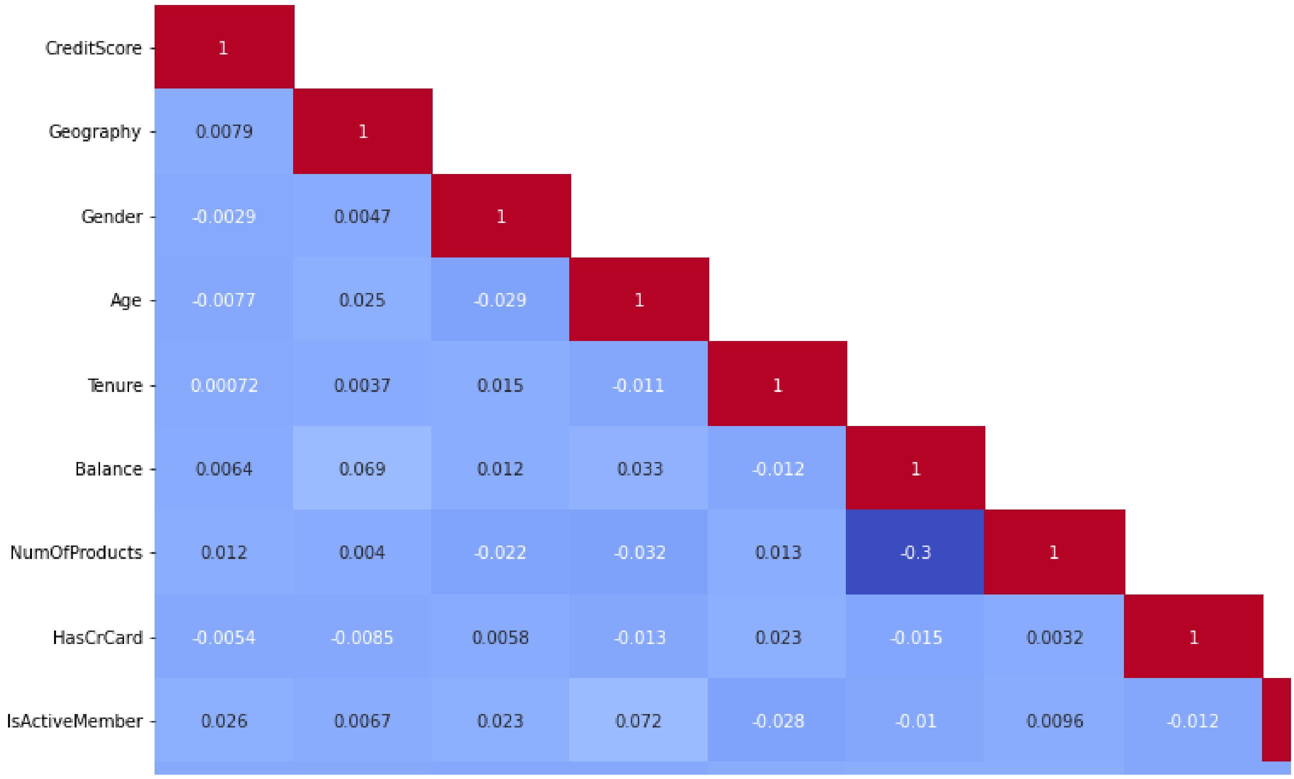
```
df = df.drop(['RowNumber','CustomerId','Surname'],axis=1)
```

```
df.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard
0	619.0	0	0	42.0	2	0.00	1	1
1	608.0	2	0	41.0	1	83807.86	1	0
2	502.0	0	0	42.0	8	159660.80	3	1
3	699.0	0	0	39.0	1	0.00	2	0
4	850.0	2	0	43.0	2	125510.82	1	1

```
plt.figure(figsize=(20,10))
df_lt = df.corr(method = "pearson")
df_lt1 = df_lt.where(np.tril(np.ones(df_lt.shape)).astype(np.bool))
sns.heatmap(df_lt1,annot=True,cmap="coolwarm")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc95b0fefd0>
```



1. The Removed columns are nothing to do with model building.
2. Feature importance also checked using pearson correlation.

8) Splitting the data into dependent and independent variables

```
target = df['Exited']
data = df.drop(['Exited'],axis=1)
```

```
print(data.shape)
print(target.shape)
```

```
(10000, 10)
(10000,)
```

9) Scaling the independent variables

```
from sklearn.preprocessing import StandardScaler
se = StandardScaler()
```

```
data['CreditScore'] = se.fit_transform(pd.DataFrame(data['CreditScore']))
data['Age'] = se.fit_transform(pd.DataFrame(data['Age']))
```

```
data['Balance'] = se.fit_transform(pd.DataFrame(data['Balance']))
data['EstimatedSalary'] = se.fit_transform(pd.DataFrame(data['EstimatedSalary']))

data.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrC
0	-0.326878	0	0	0.342615	2	-1.225848	1	
1	-0.440804	2	0	0.240011	1	0.117350	1	
2	-1.538636	0	0	0.342615	8	1.333053	3	
3	0.501675	0	0	0.034803	1	-1.225848	2	
4	2.065569	2	0	0.445219	2	0.785728	1	

10) Splitting 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(data,target,test_size=0.25,random_state=1
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(7500, 10)
(2500, 10)
(7500,)
(2500,)
```

Conclusion: The model is scaled using StandarScaler method. The train and test split ratio is15:5. As it is a classification problem, basic algorithms can be used to build ML models