NAME : MOHANAVEL K
REGISTER NO : 611219106046
DATE : 24/09/2022

Assignment 2

1. Download the dataset from the source here.

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 the churn 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 **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. Load the dataset

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

df.tail()

```
      RowNumbe  CustomerId    Surname   CreditScore Geography   Gender
Age   r
  \
9995      9996    15606229   Obijiaku           771    France     Male
39
9996      9997    15569892  Johnstone           516    France     Male
35
9997      9998    15584532        Liu           709    France   Female
36
9998      9999    15682355  Sabbatini           772   Germany     Male
42
9999     10000    15628319     Walker           792    France   Female
28

      Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember   \
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0

      EstimatedSalary   Exited
9995         96270.64        0
9996        101699.77        0
9997         42085.58        1
9998         92888.52        1
9999         38190.78        0
```

3 a). 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

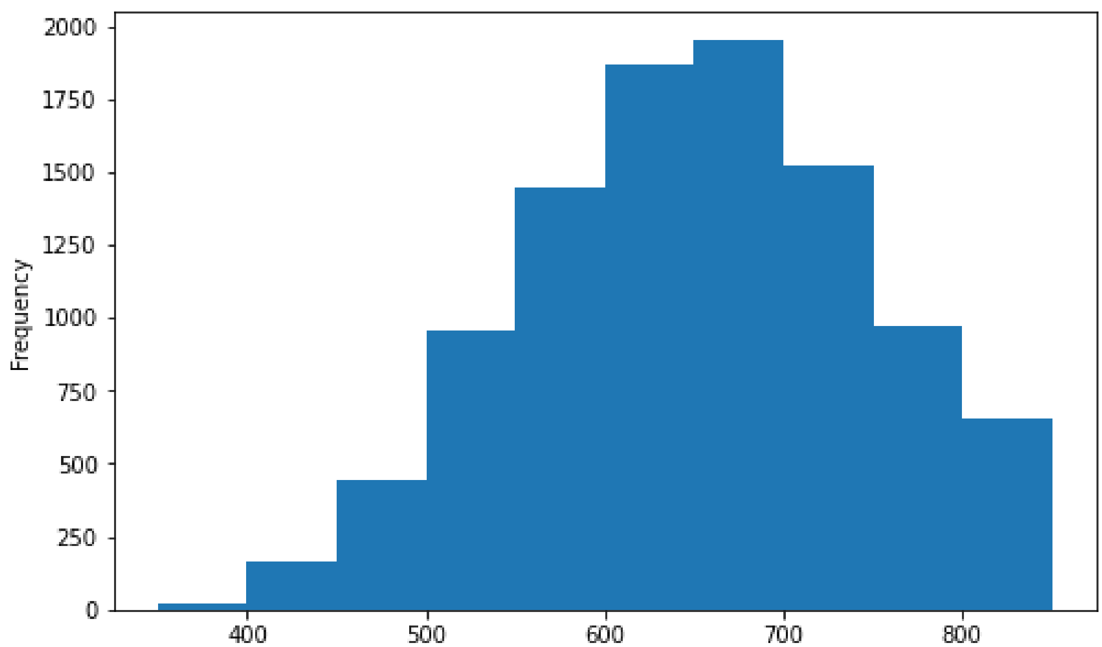
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore',
'Geography',
'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
'IsActiveMember', 'EstimatedSalary', 'Exited'], dtype='object')

df.shape

(10000, 14)

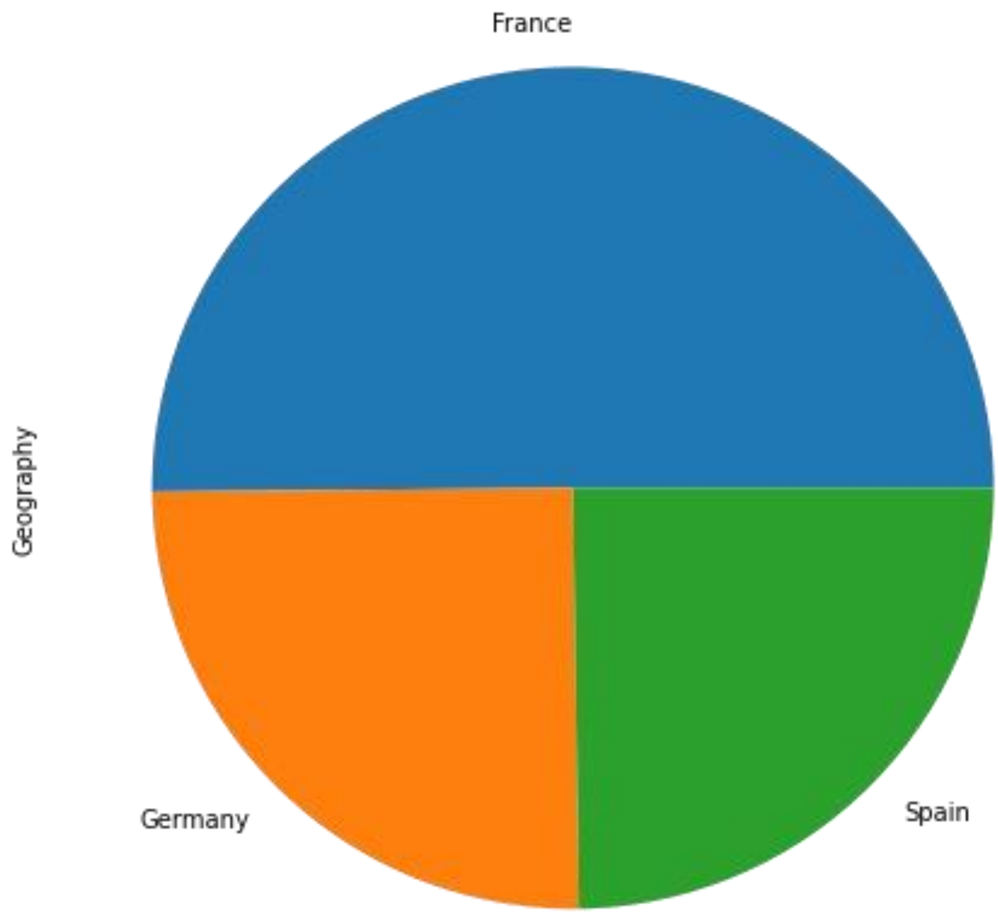
credit = df['CreditScore']
credit.plot(kind="hist",figsize=(8,5))

<matplotlib.axes._subplots.AxesSubplot at 0x7f5aff578ad0>



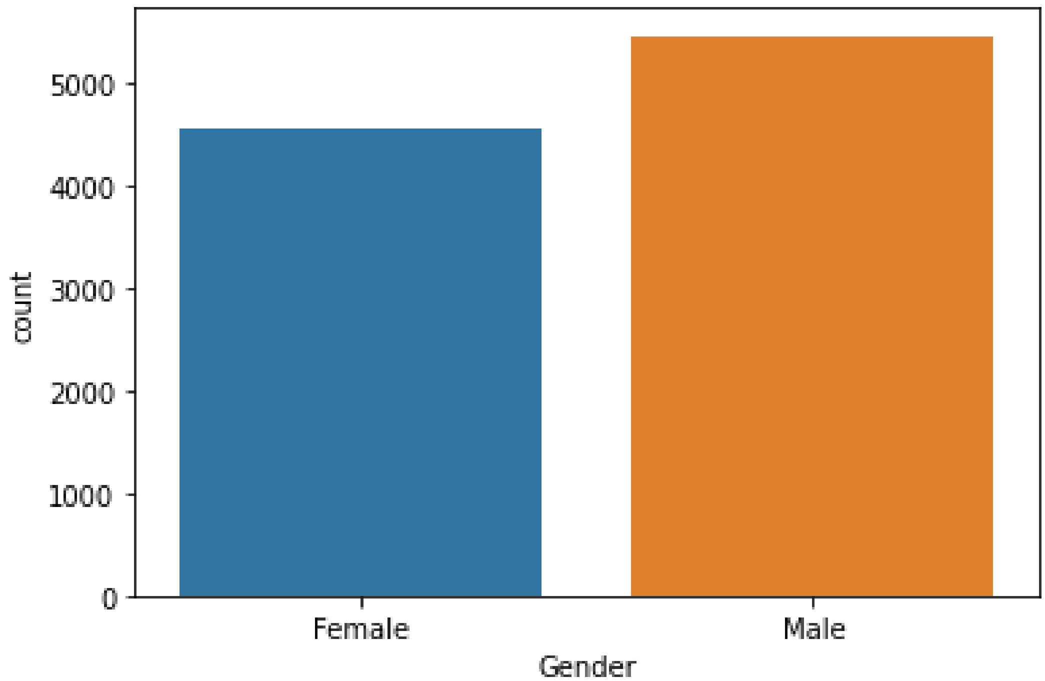
geo = df['Geography'].value_counts()
geo.plot(kind="pie",figsize=(10,8))

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af91209d0>



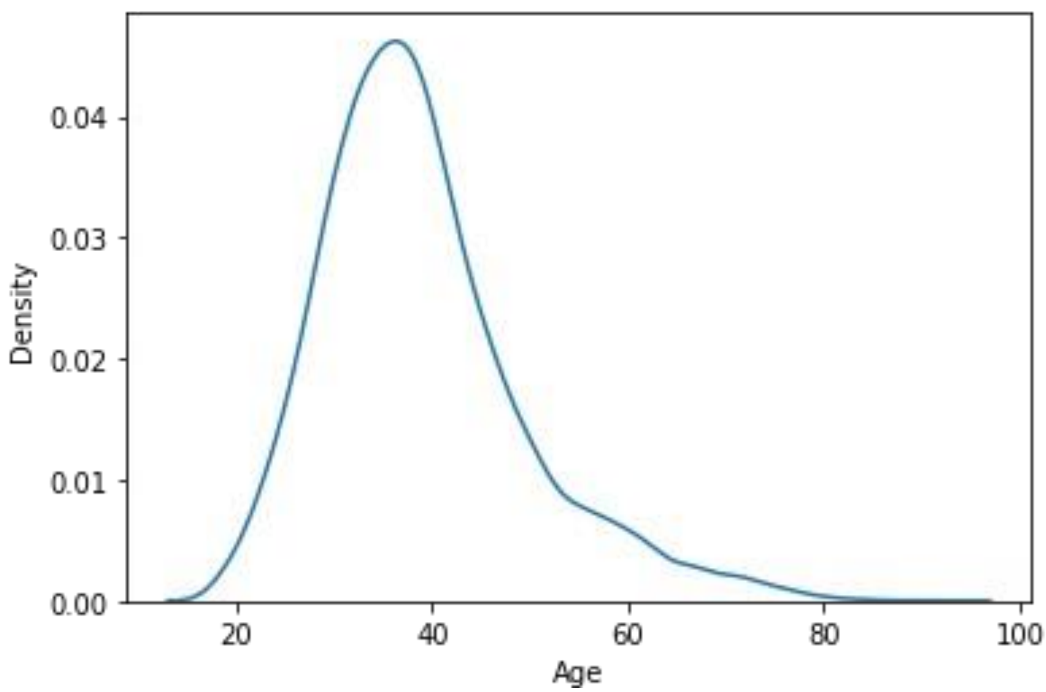

```
sns.countplot(df['Gender'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af8ff5f10>



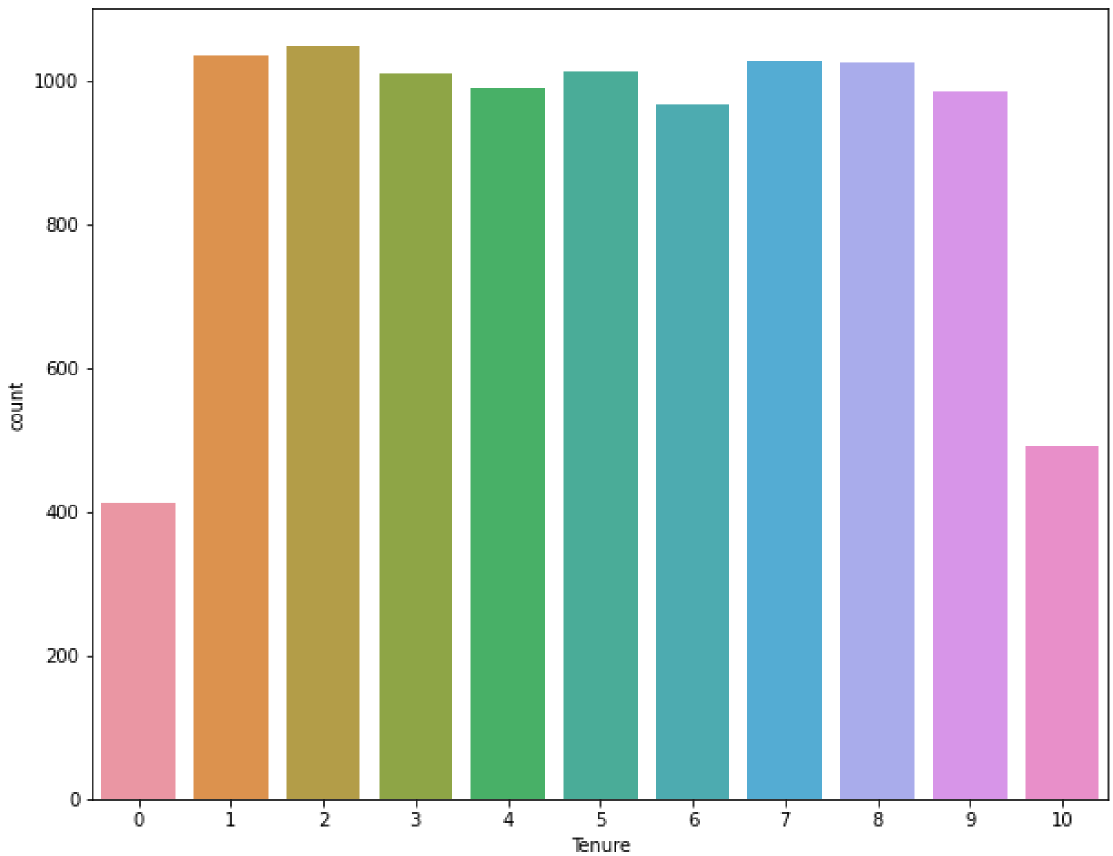

```
sns.distplot(df['Age'],hist=False)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af8b08450>



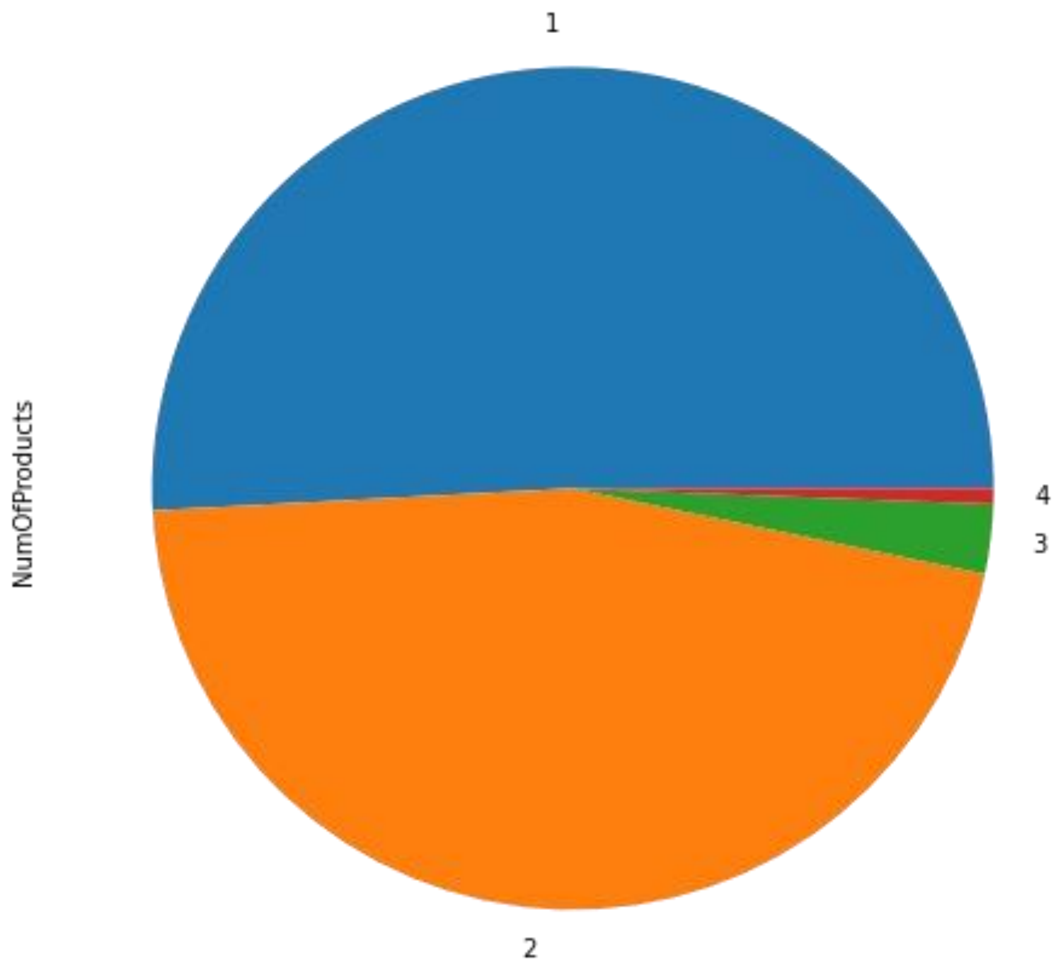

```
plt.figure(figsize=(10,8))
sns.countplot(df['Tenure'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af8a5fbd0>



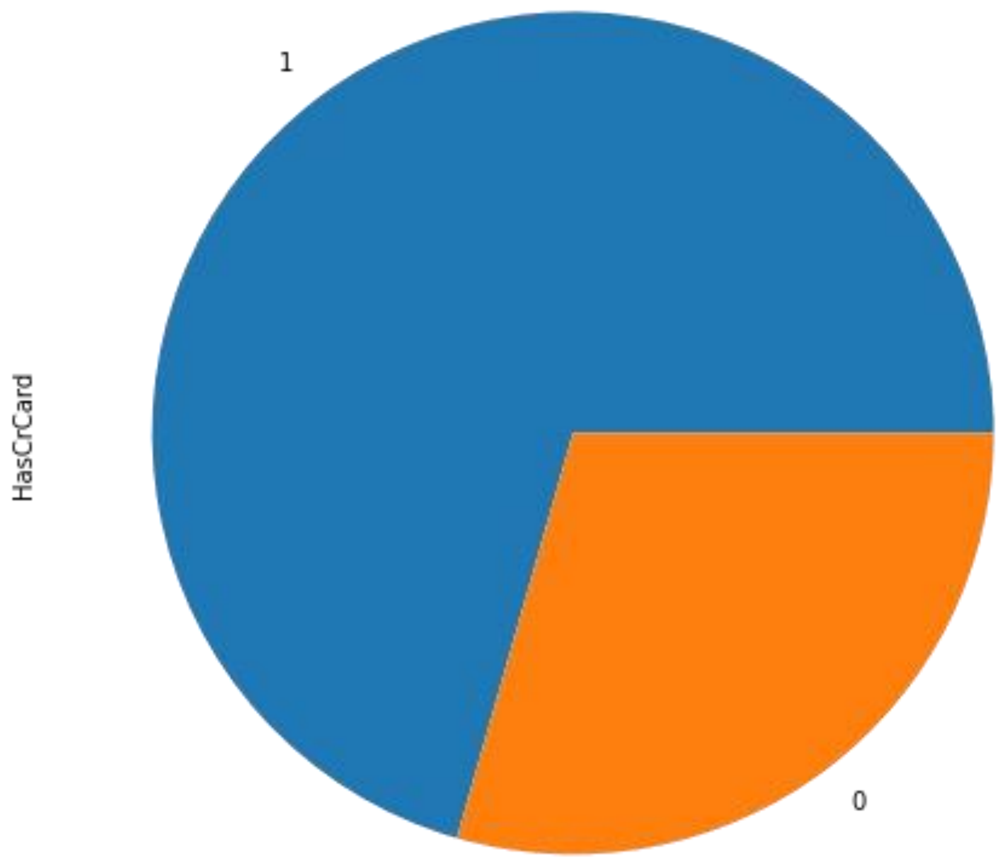

```
product = df['NumOfProducts'].value_counts()
product.plot(kind="pie",figsize=(10,8))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5af89fe350>
```



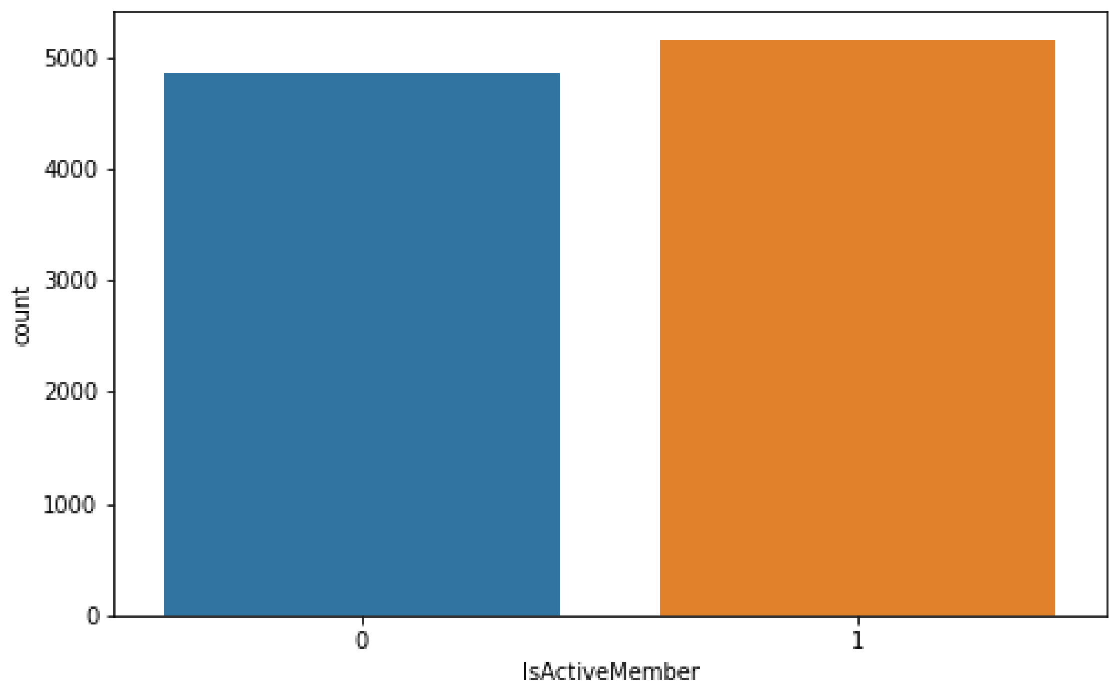

```
cr = df['HasCrCard'].value_counts()
cr.plot(kind="pie",figsize=(10,8))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5af8969490>
```



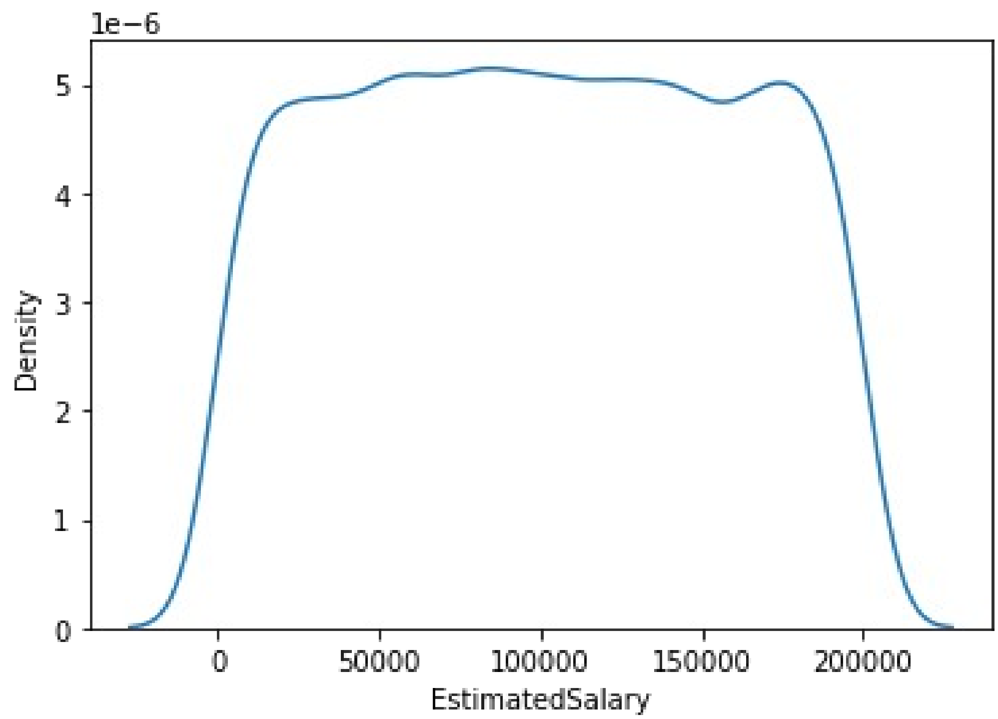

```
plt.figure(figsize=(8,5))
sns.countplot(df['IsActiveMember'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af891f2d0>



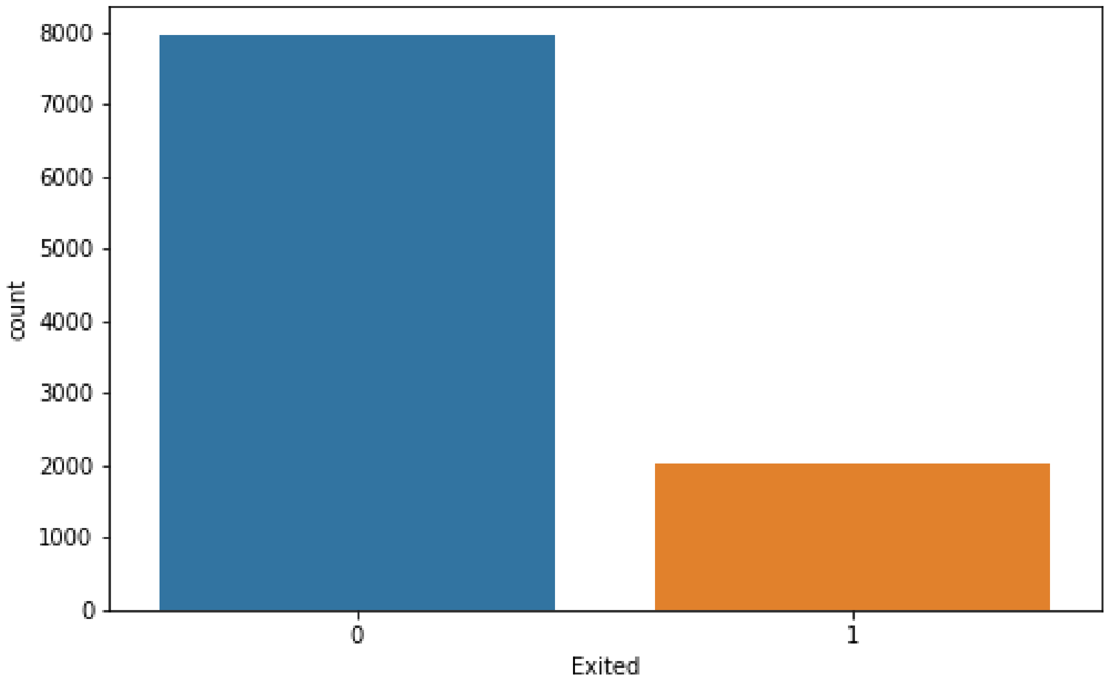

```
sns.distplot(df['EstimatedSalary'],hist=False)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af892a550>




```
plt.figure(figsize=(8,5))
sns.countplot(df['Exited'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af880ff90>



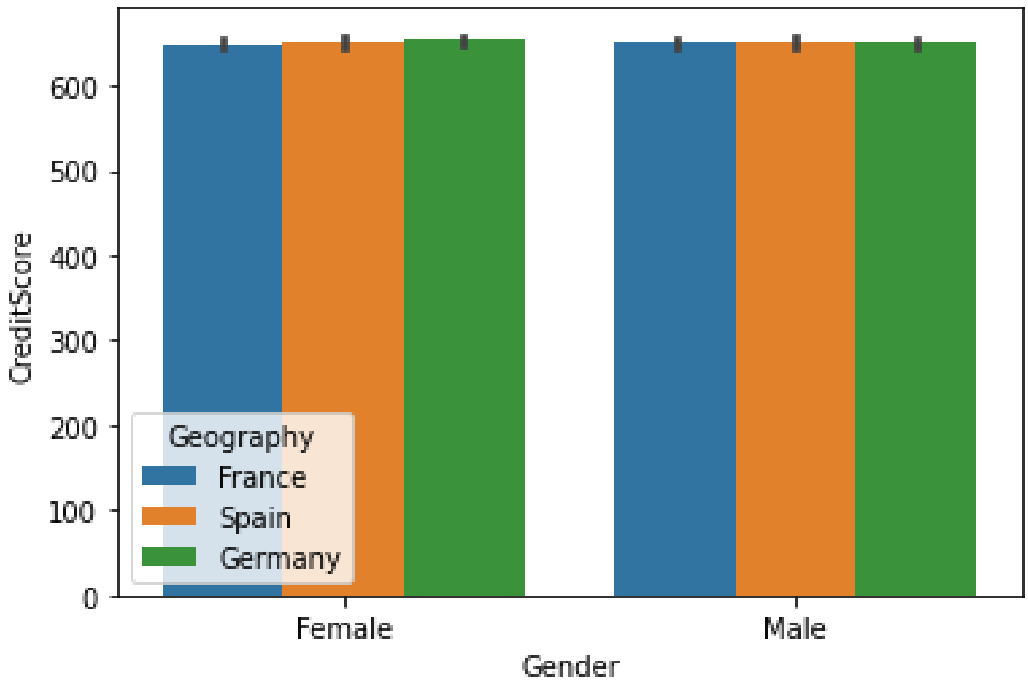
Inference:

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 creditscore greater than 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.**

3 b). Bivariate analysis

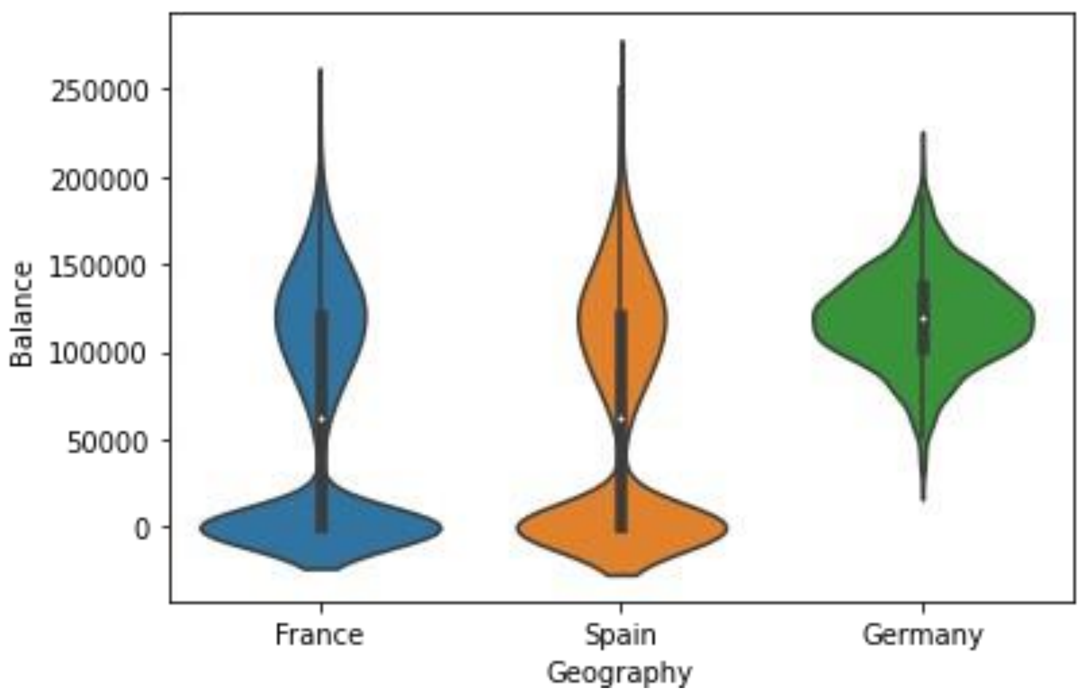
```
sns.barplot(x='Gender',y='CreditScore',hue='Geography',data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af87e39d0>



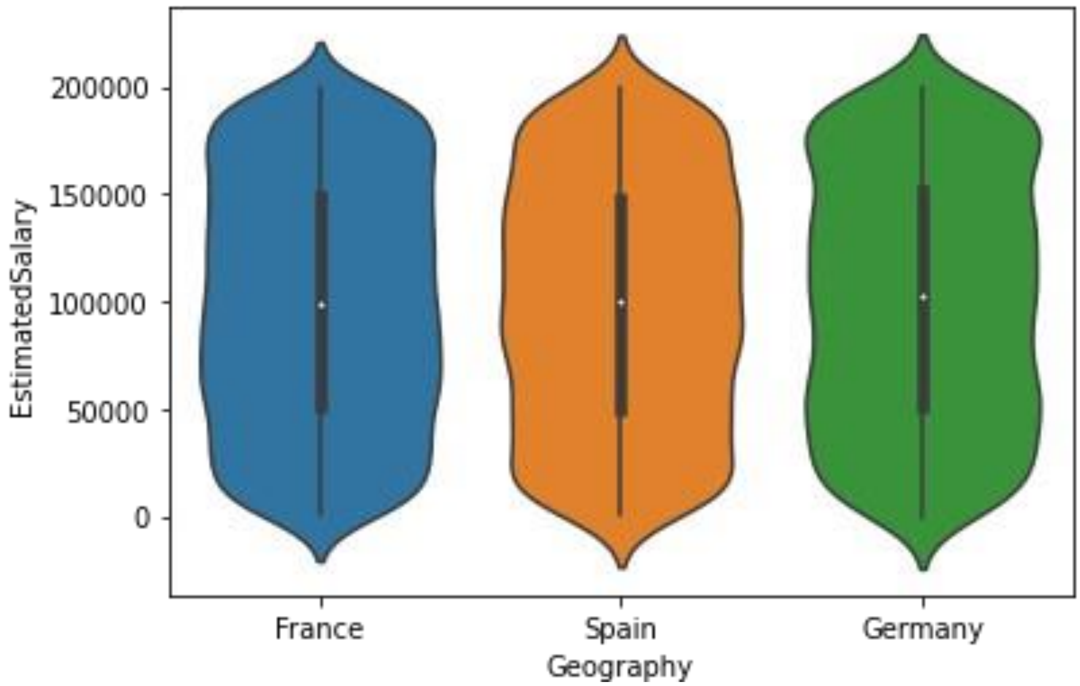

```
sns.violinplot(x='Geography',y='Balance',data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af8700650>



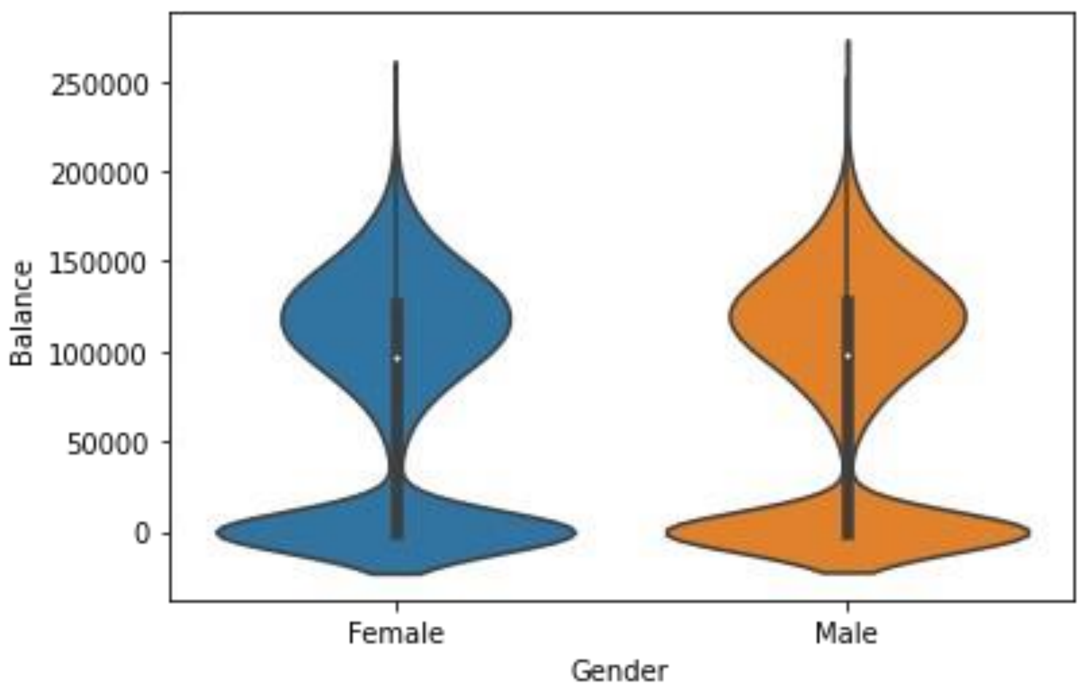

```
sns.violinplot(x='Geography',y='EstimatedSalary',data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af868dcd0>



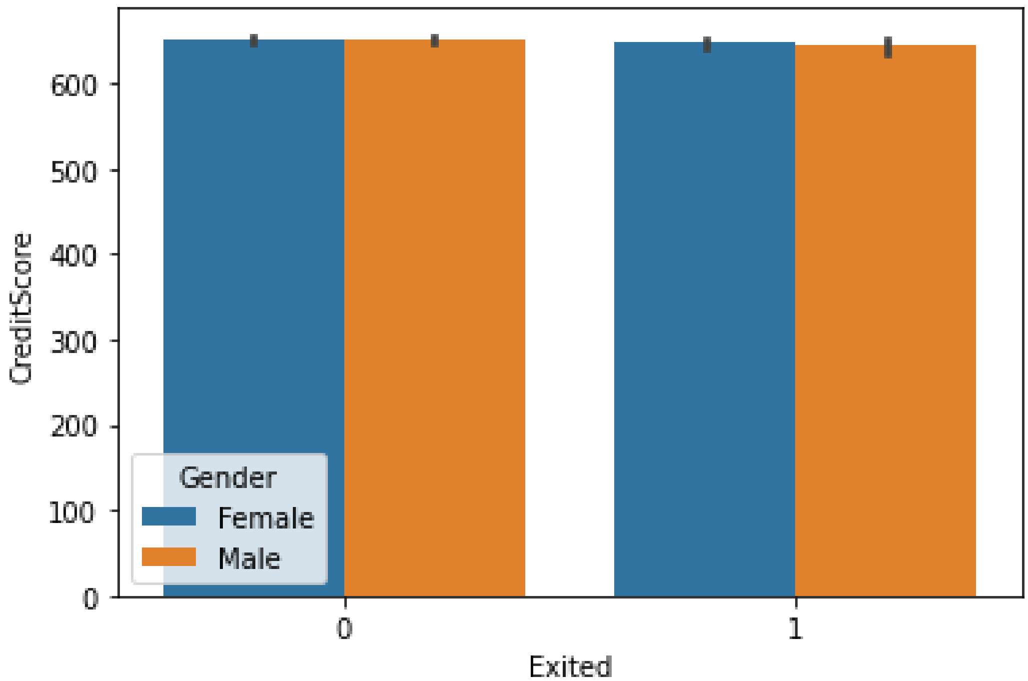
sns.violinplot(x='Gender',y='Balance',data=df)

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af8608290>



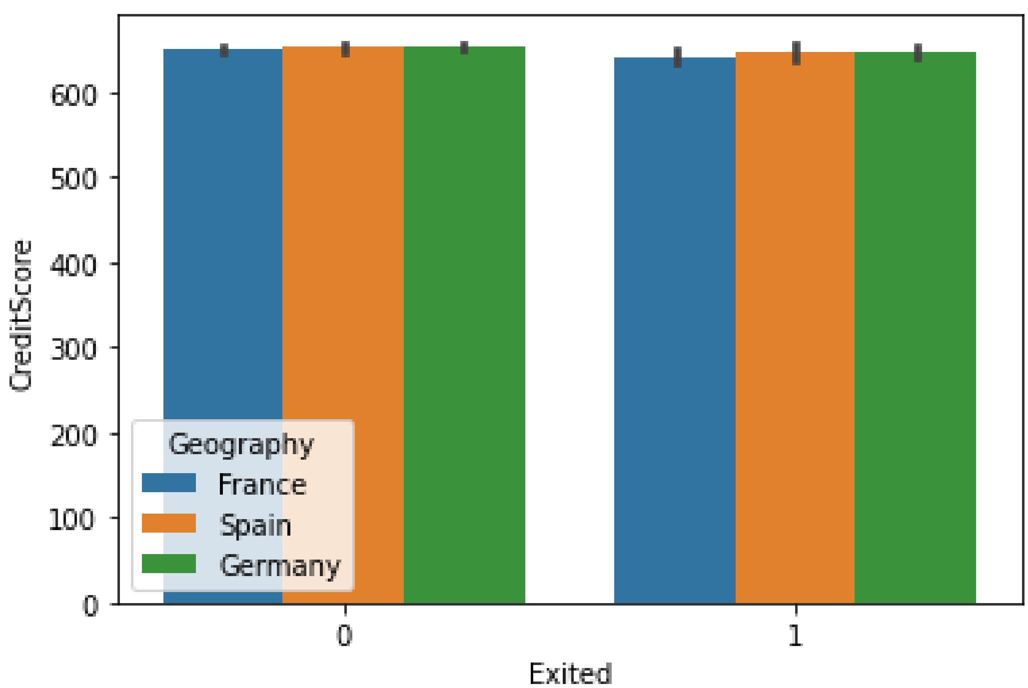
sns.barplot(x='Exited',y='CreditScore',hue='Gender',data=df)

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af85f9b90>



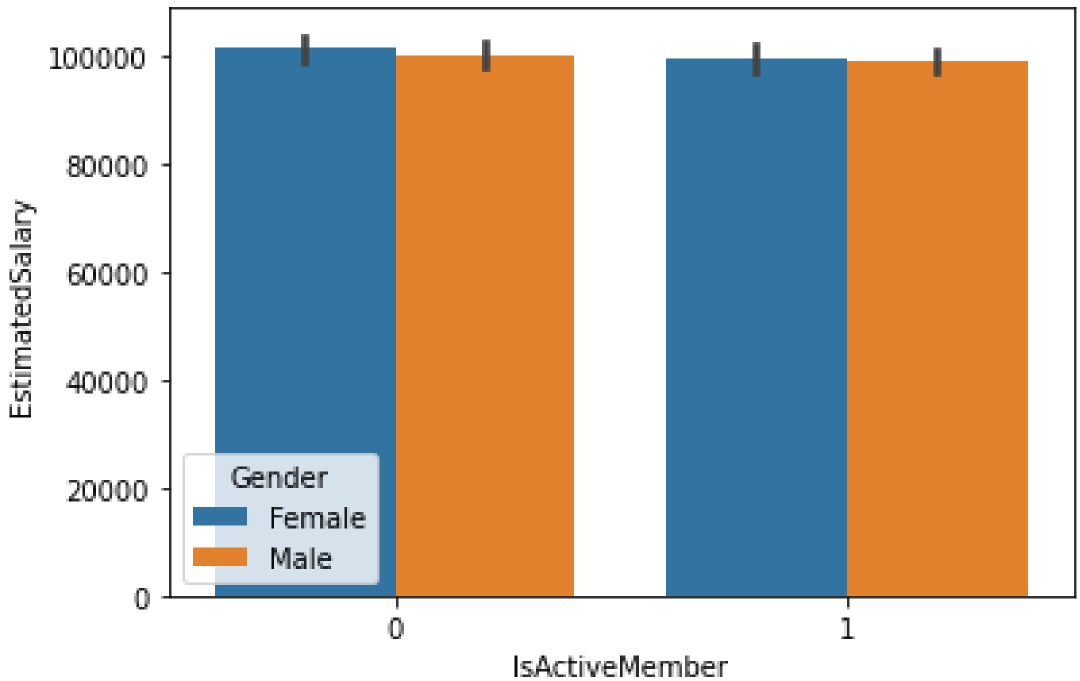
sns.barplot(x='Exited',y='CreditScore',hue='Geography',data=df)

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af85627d0>



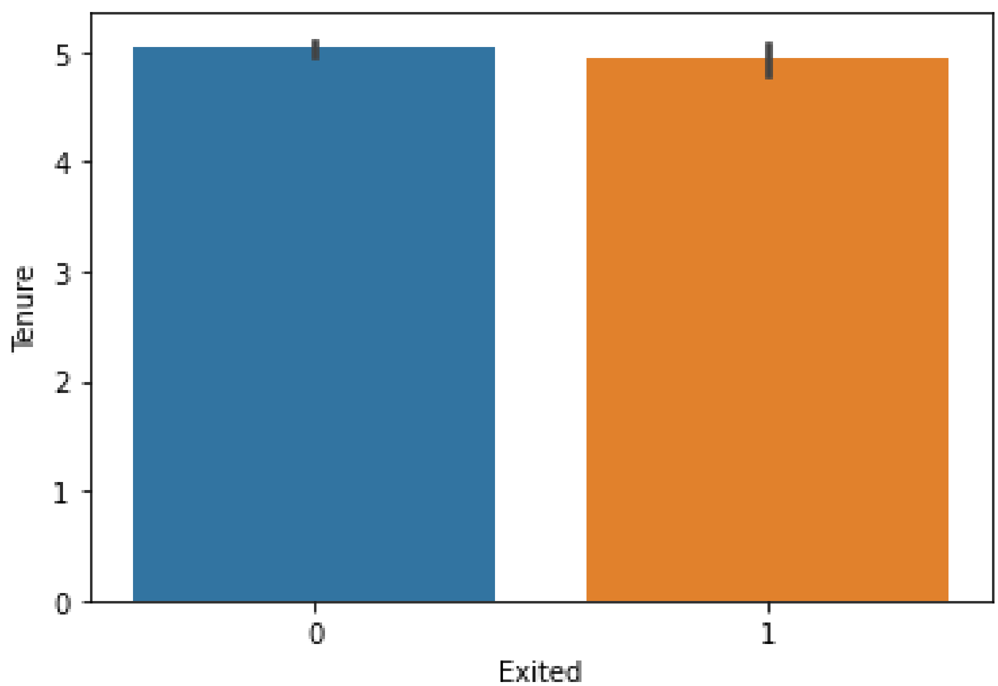
sns.barplot(x='IsActiveMember',y='EstimatedSalary',hue='Gender',data=d f)

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af88c0190>




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

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af87d2550>



Inference:

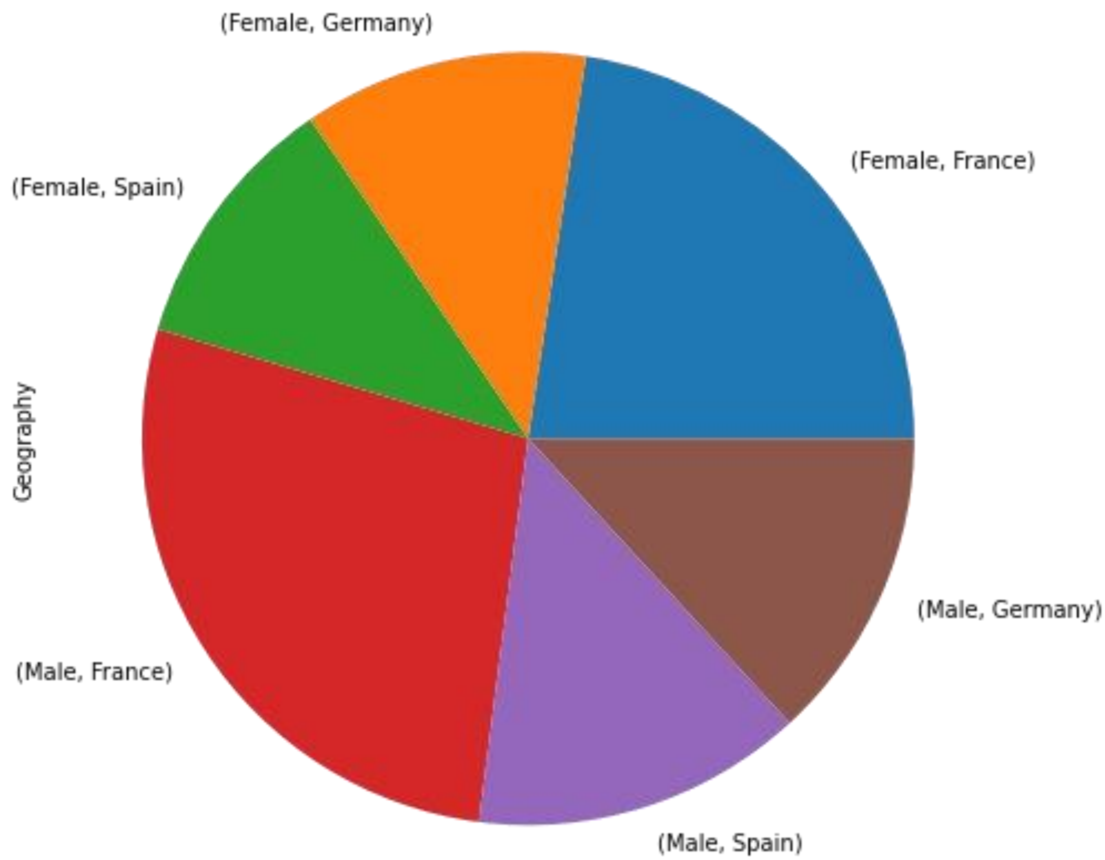
1. Credit score for Male is higher in Spain.

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.

3 c). Multivariate 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('Geography')
['HasCrCard','IsActiveMember'].value_counts()
gp4.plot(kind="bar",figsize=(8,5))  print(gp4)

-------------------------------------------------------------------------------
------
AttributeError                                             Traceback (most recent call
last)
<ipython-input-29-869056562afd> in <module>
----> 1 gp4 = df.groupby('Geography')
['HasCrCard','IsActiveMember'].value_counts()
      2  gp4.plot(kind="bar",figsize=(8,5))
      3  print(gp4)

/usr/local/lib/python3.7/dist-packages/pandas/core/groupby/groupby.py in_
______getattr____(self, attr)
    910
    911             raise AttributeError(
--> 912                   f"'{type(self).______name__}' object has no attribute
'{attr}'"
    913             )
    914

AttributeError: 'DataFrameGroupBy' object has no attribute
'value_counts'

gp5 = df.groupby(['Gender','HasCrCard','IsActiveMember'])
['EstimatedSalary'].mean() gp5.plot(kind="line",figsize=(10,8))
print(gp5)

gp6 = df.groupby(['Gender','IsActiveMember'])['Exited'].value_counts()
gp6.plot(kind='bar',figsize=(10,8))
print(gp6)

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

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

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 to those 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. Descriptive statistics

df.describe().T

	count	mean	std	min
RowNumber	10000.0	5.000500e+03	2886.895680	1.00
CustomerId	10000.0	1.569094e+07	71936.186123	15565701.00
CreditScore	10000.0	6.505288e+02	96.653299	350.00
Age	10000.0	3.892180e+01	10.487806	18.00
Tenure	10000.0	5.012800e+00	2.892174	0.00
Balance	10000.0	7.648589e+04	62397.405202	0.00
NumOfProducts	10000.0	1.530200e+00	0.581654	1.00
HasCrCard	10000.0	7.055000e-01	0.455840	0.00
IsActiveMember	10000.0	5.151000e-01	0.499797	0.00
EstimatedSalary	10000.0	1.000902e+05	57510.492818	11.58
Exited	10000.0	2.037000e-01	0.402769	0.00

\

	25%	50%	75%	max
RowNumber	2500.75	5.000500e+03	7.500250e+03	10000.00
CustomerId	15628528.25	1.569074e+07	1.575323e+07	15815690.00
CreditScore	584.00	6.520000e+02	7.180000e+02	850.00
Age	32.00	3.700000e+01	4.400000e+01	92.00
Tenure	3.00	5.000000e+00	7.000000e+00	10.00
Balance	0.00	9.719854e+04	1.276442e+05	250898.09
NumOfProducts	1.00	1.000000e+00	2.000000e+00	4.00
HasCrCard	0.00	1.000000e+00	1.000000e+00	1.00
IsActiveMember	0.00	1.000000e+00	1.000000e+00	1.00
EstimatedSalary	51002.11	1.001939e+05	1.493882e+05	199992.48
Exited	0.00	0.000000e+00	0.000000e+00	1.00

5. Handling the missing values

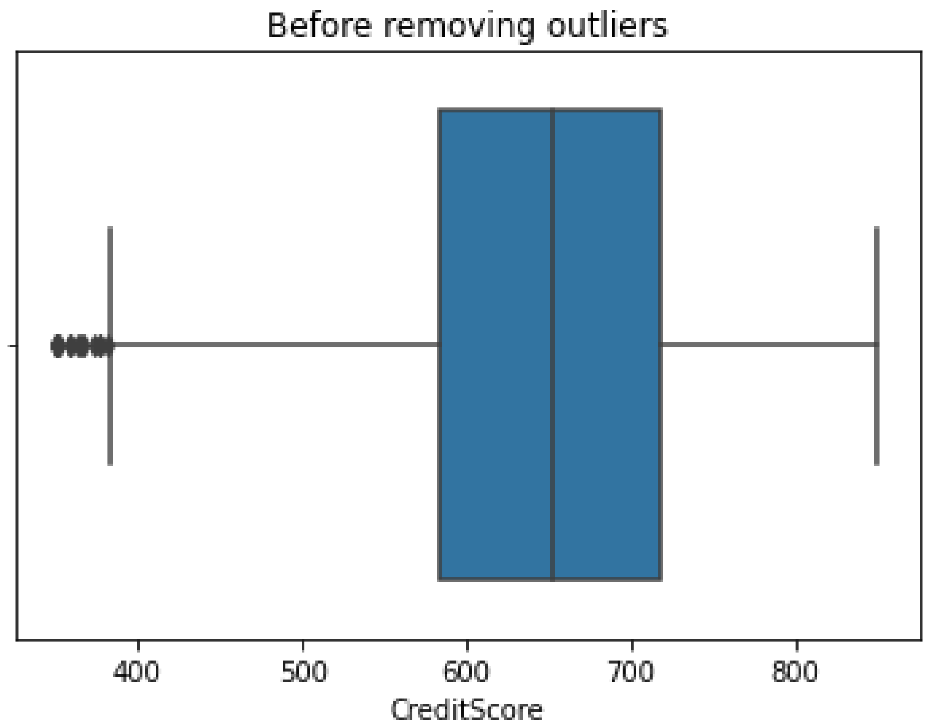
```
df.isnull().sum()
```

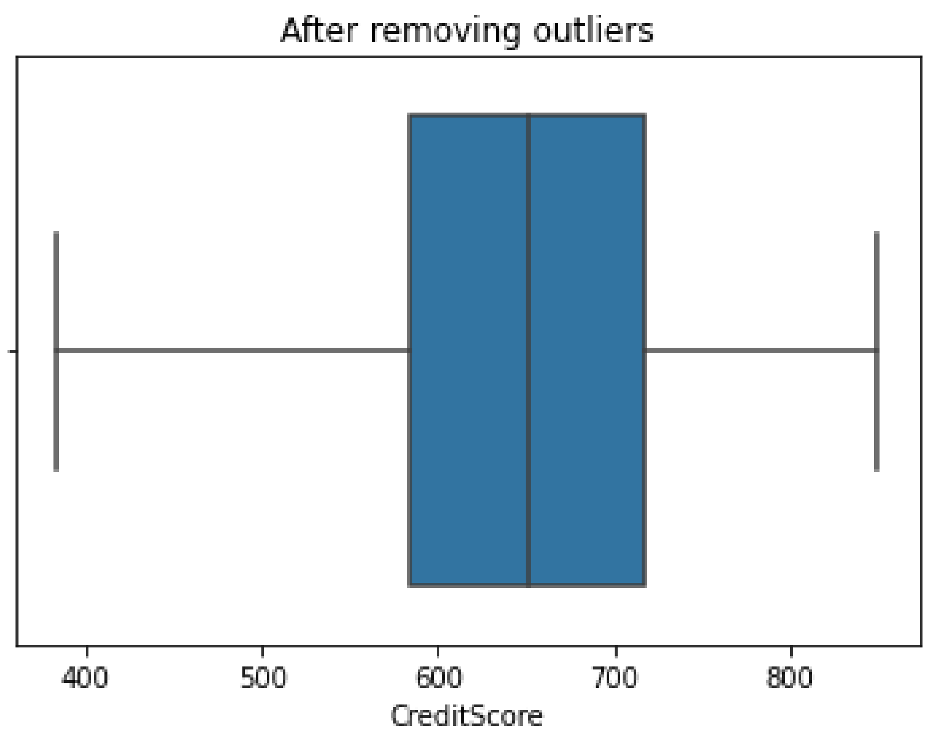
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: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

There is no missing value in the dataset

6. Finding outliers

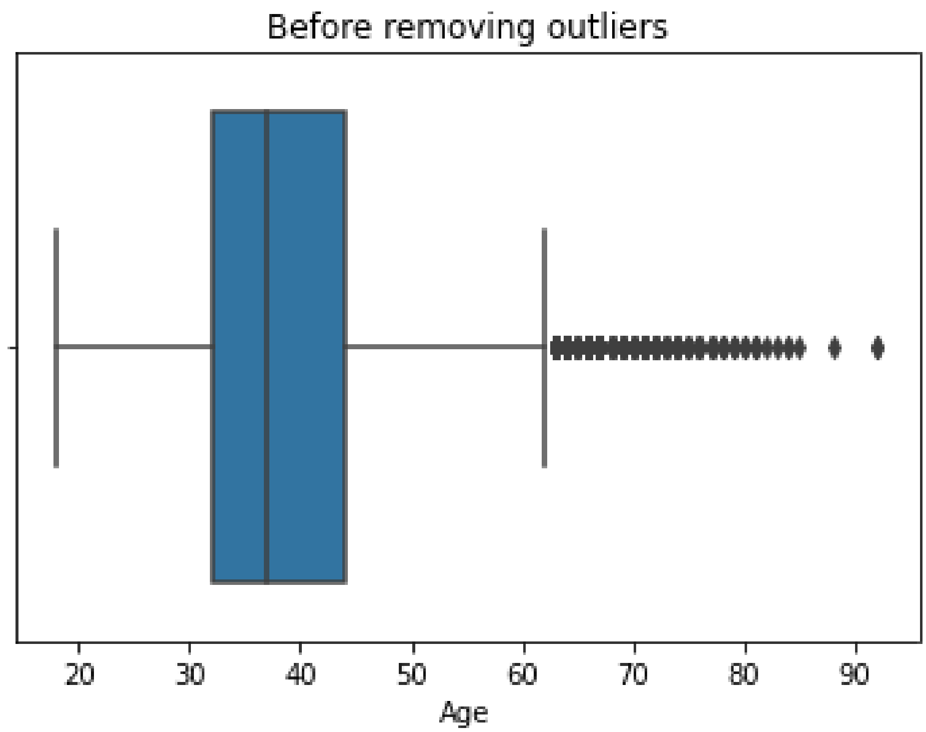
```
def replace_outliers(df, field_name):
    Q1 = np.percentile(df[field_name],25,interpolation='midpoint') Q3 =
    np.percentile(df[field_name],75,interpolation='midpoint')
    IQR = Q3-Q1
    maxi =
    Q3+1.5*IQR mini =
    Q1-1.5*IQR
    df[field_name]=df[field_name].mask(df[field_name]>maxi,maxi)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

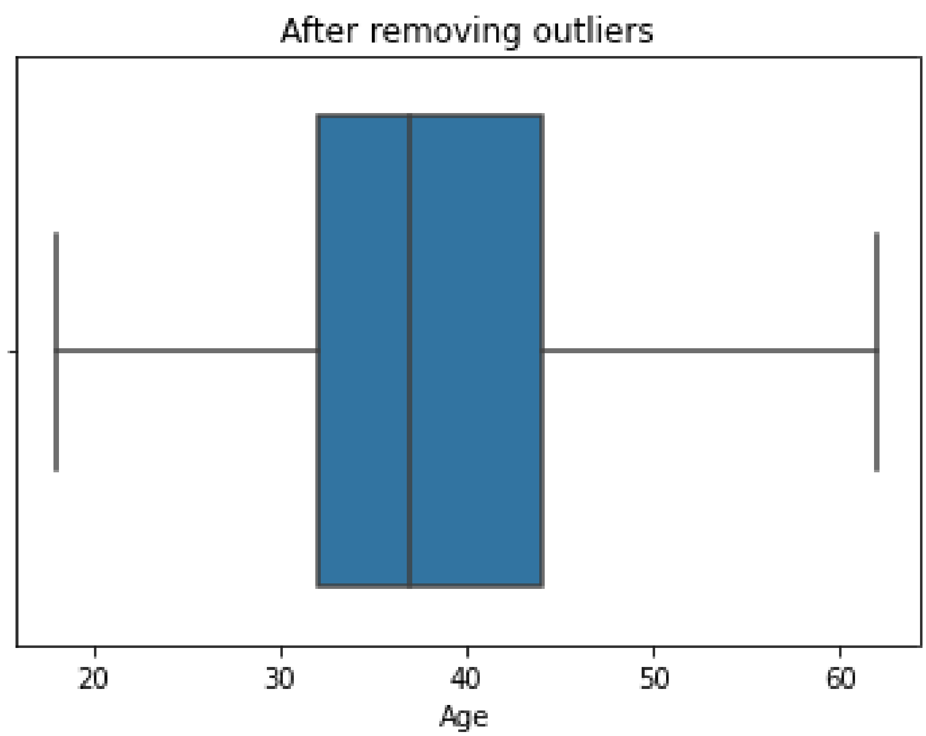





```
plt.title("Before removing outliers")
sns.boxplot(df['Age'])
plt.show()
plt.title("After removing outliers")
replace_outliers(df, 'Age')
```

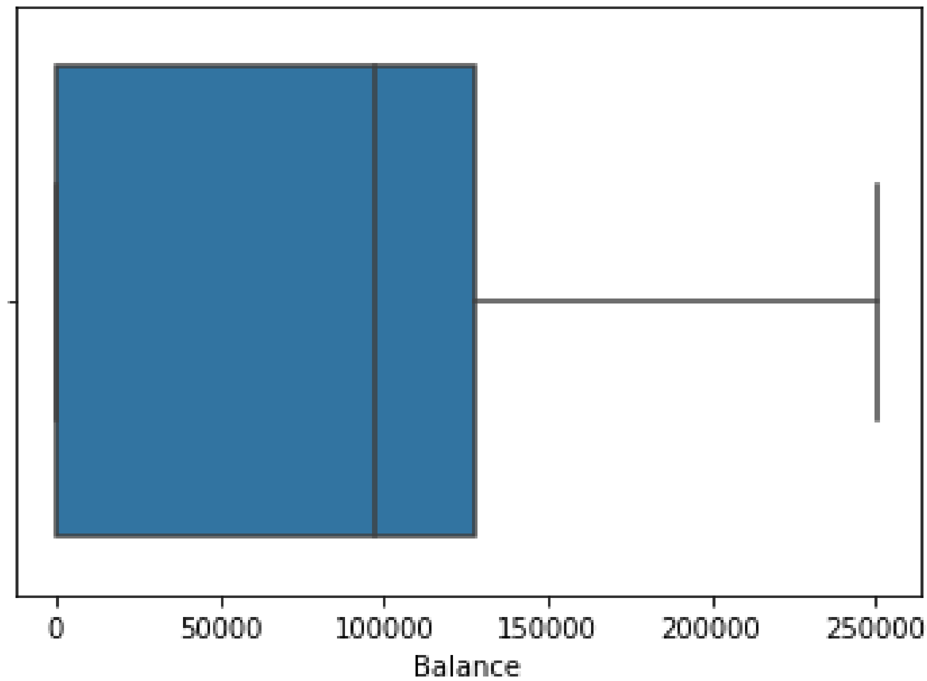
```
sns.boxplot(df['Age'])
plt.show()
```





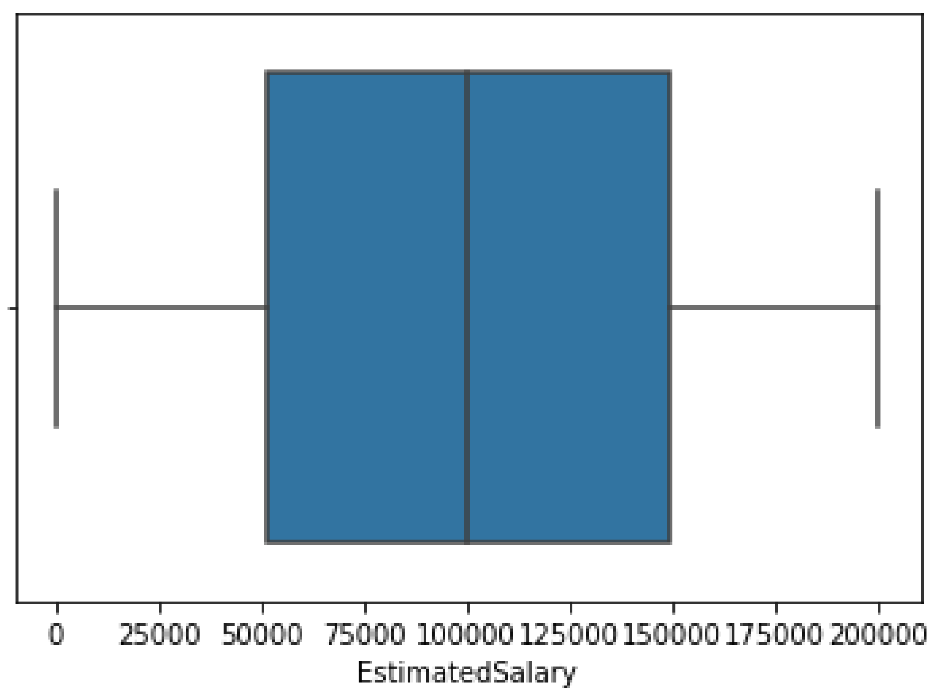

```
sns.boxplot(df['Balance'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af810e490>



sns.boxplot(df['EstimatedSalary'])

<matplotlib.axes._subplots.AxesSubplot at 0x7f5af807c810>



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   \
0           1    15634602   Hargrave        619.0            0        0
42.0
1           2    15647311       Hill        608.0            2        0
41.0
2           3    15619304       Onio        502.0            0        0
42.0
3           4    15701354       Boni        699.0            0        0
39.0
4           5    15737888   Mitchell        850.0            2        0
43.0

   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
```

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)

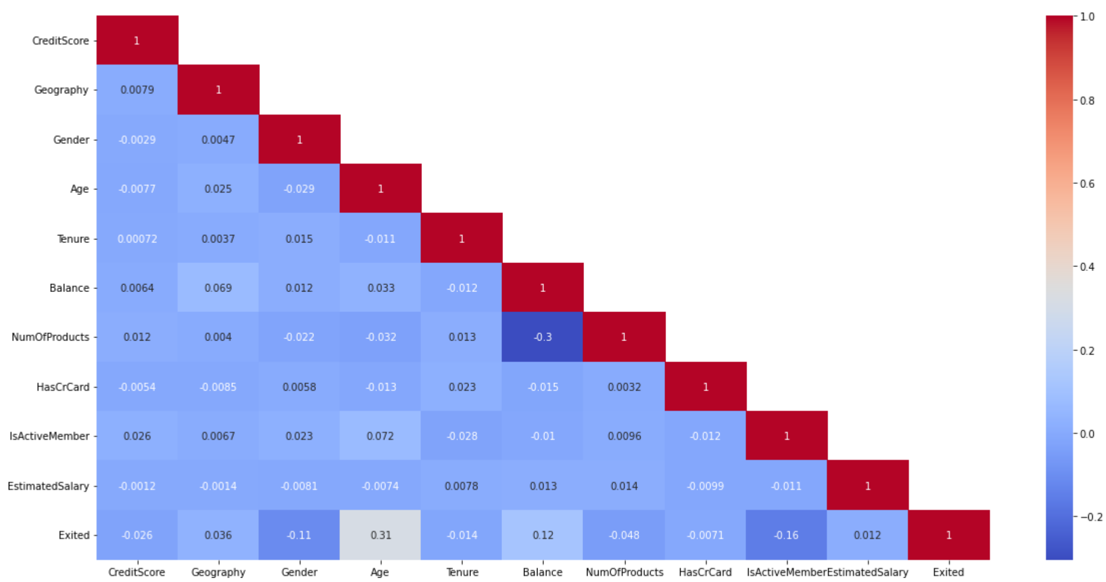
df.head()

```
   CreditScore    Geography   Gender    Age   Tenure      Balance
NumOfProducts   \
0        619.0            0        0   42.0        2         0.00
1
1        608.0            2        0   41.0        1     83807.86
1
```

```
2          502.0          0          0    42.0          8    159660.80
3
3          699.0          0          0    39.0          1         0.00
2
4          850.0          2          0    43.0          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
```

```
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 0x7f5af85d82d0>
```



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

8. Data Splitting

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

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

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

9. Scaling the independent values

```
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   \
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

    HasCrCard   IsActiveMember   EstimatedSalary
0           1                1          0.021886
1           0                1          0.216534
2           1                0          0.240687
3           0                0         -0.108918
4           1                1         -0.365276
```

10. Train test split

```
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=101)

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

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

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

Conclusion:

1. The model is scaled using StandarScaler method.
2. The train and test split ratio is 15:5.
3. As it is a classification problem, basic algorithms can be used to build ML models.