2448513_Deshmukh_Pratik_Bhushanrao_ML Lab-1

Name: Deshmukh Pratik Bhushanrao Roll no.: 2448513 Subject: Machine Learning Subject code: MAI171

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

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

Out[ ]:

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

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	Bal
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.00
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.88
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.40
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.00
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.00
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.54
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.24
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.09

In [ ]:

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

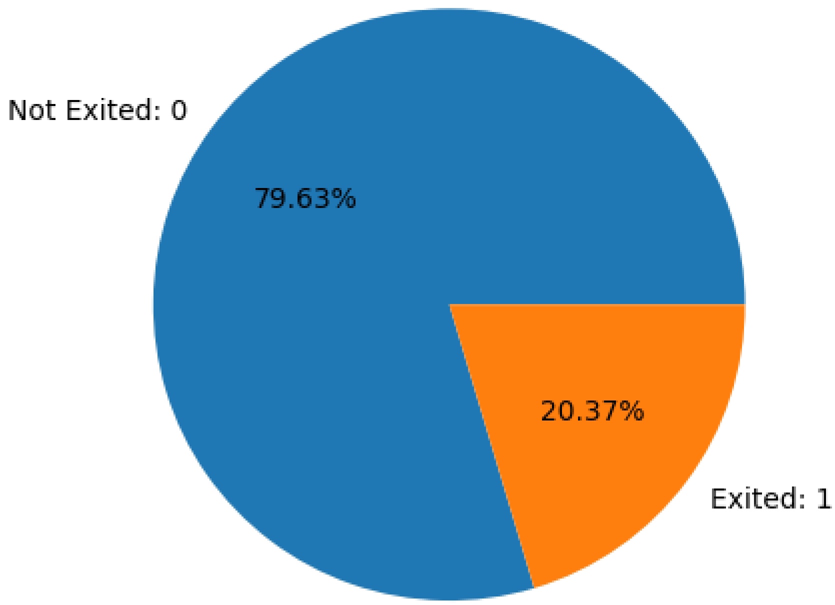
Out[ ]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCa
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	

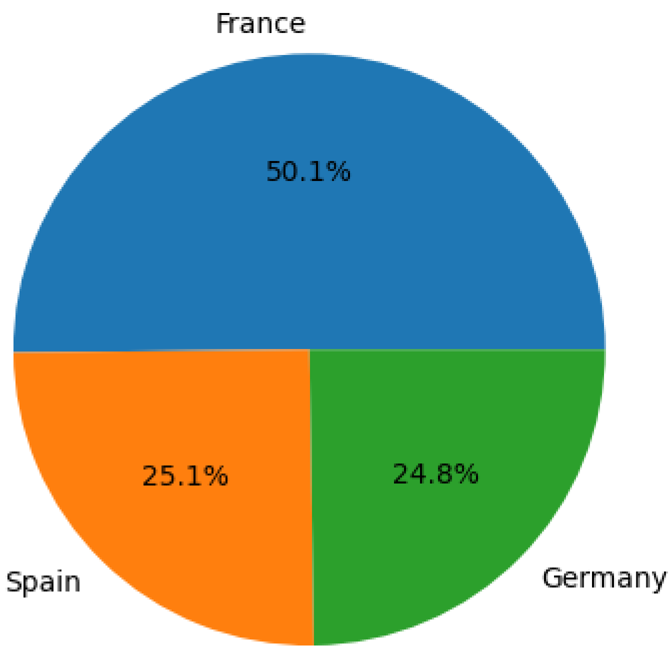
EDA

In [ ]:

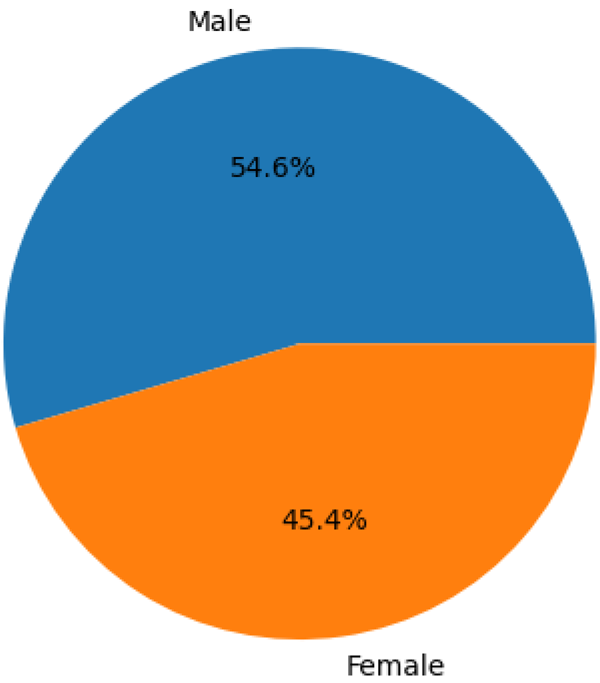
```
# Let's explore our target variable first
plt.pie(df.Exited.value_counts(), autopct='%1.2f%%', labels=['Not Exited: 0', 'E
plt.show()
```



In [ ]:
```
plt.pie(df.Geography.value_counts(), autopct='%1.1f%%', labels=['France', 'Spain
plt.show()
```



In [ ]:
```
# Let's explore Unique values in Gender
plt.pie(df.Gender.value_counts(), autopct='%1.1f%%', labels=['Male', 'Female'])
plt.show()
```



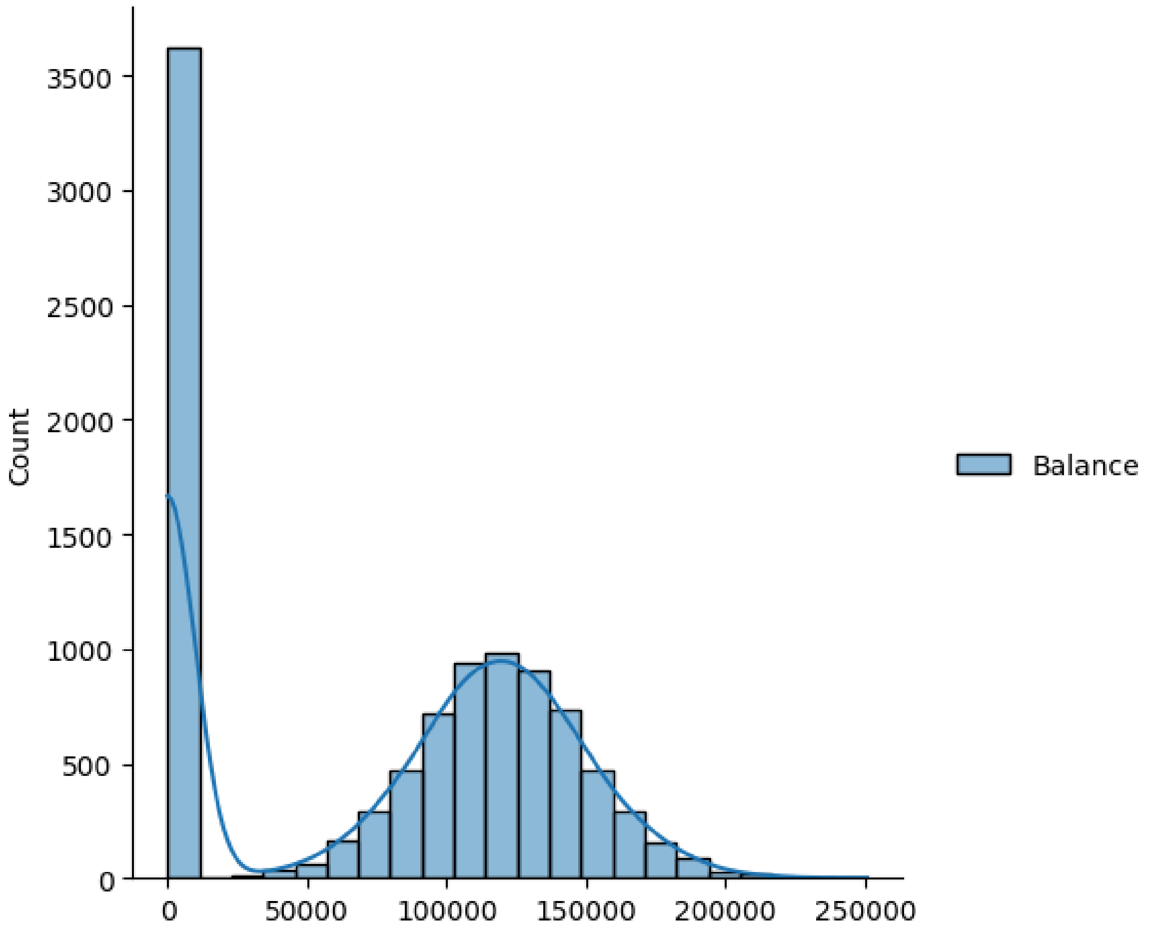
In [ ]:
```
df.head()
```

Out[ ]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCa
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	

In [ ]:
```
sns.displot([df.Balance], kde=True)
```

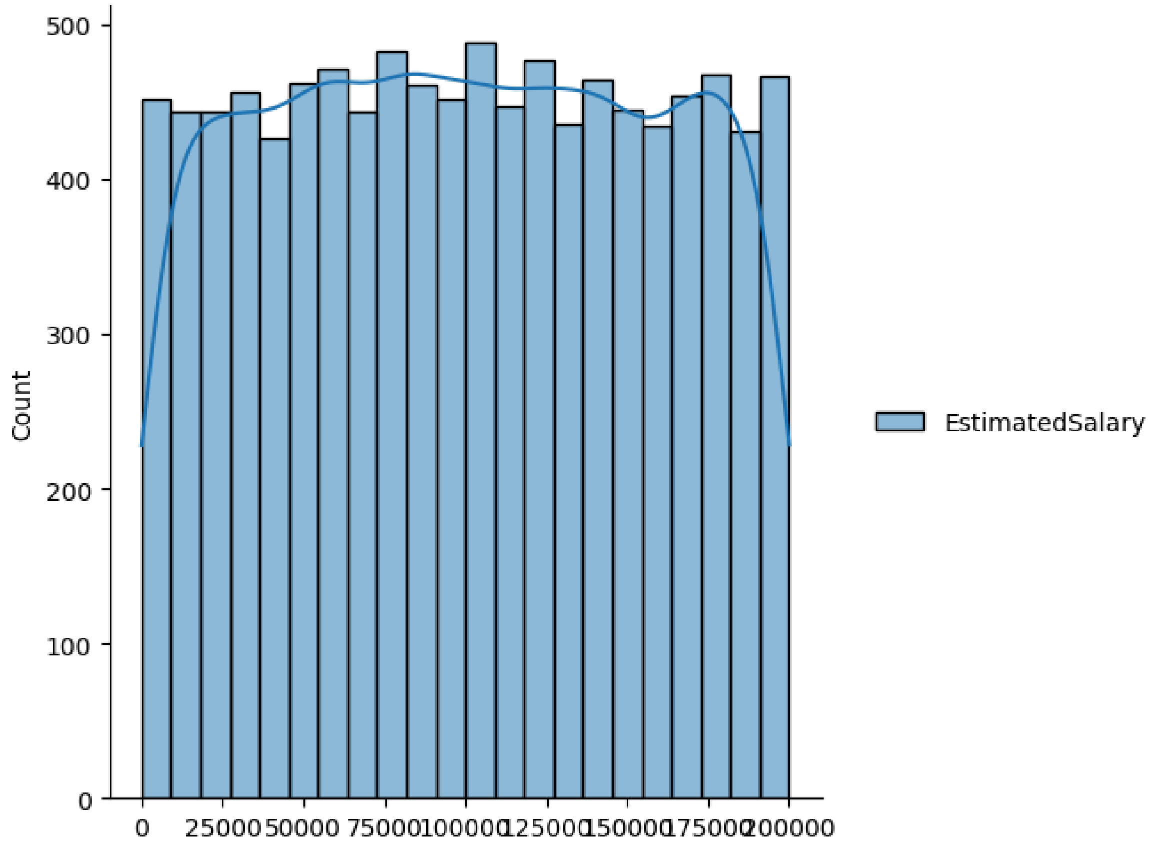
Out[ ]: <seaborn.axisgrid.FacetGrid at 0x1fdaaea3910>



It looks like most of the accounts has zero balance

In [ ]:
```
sns.displot([df.EstimatedSalary], kde=True)
```

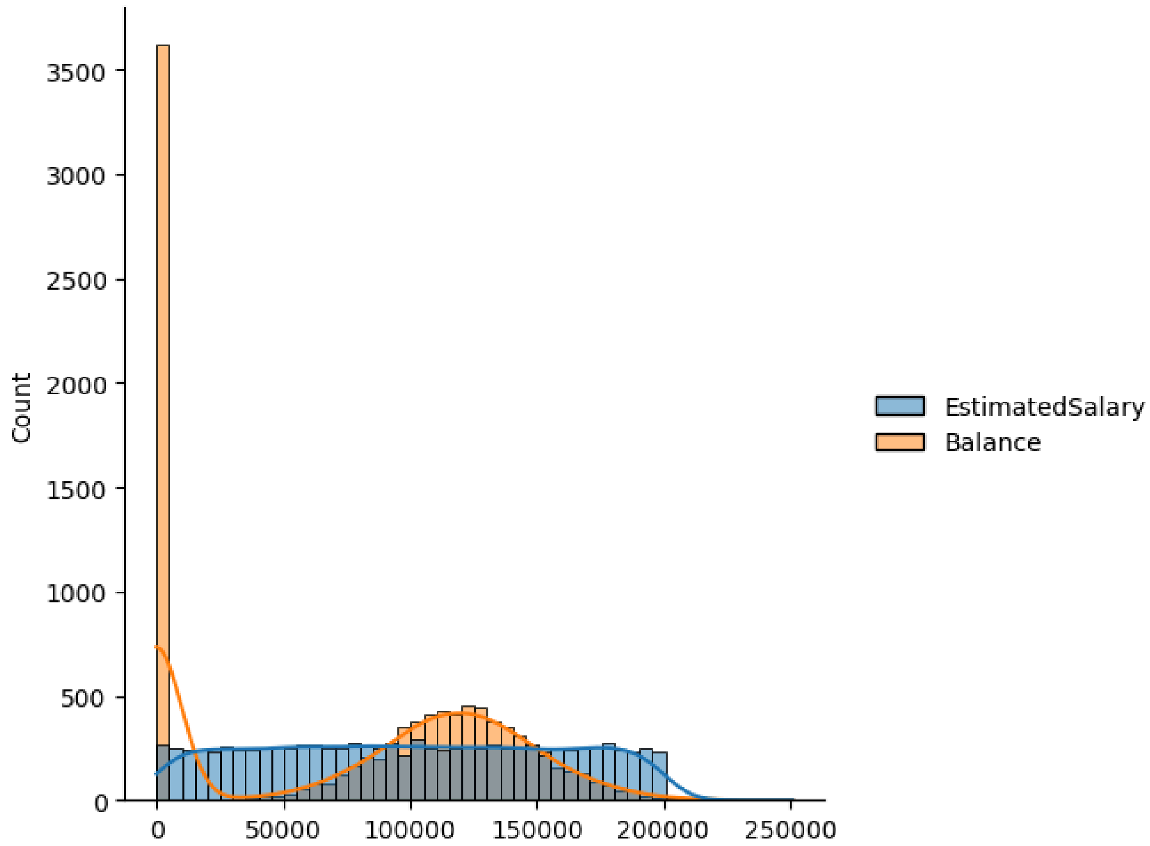
Out[ ]: <seaborn.axisgrid.FacetGrid at 0x1fdaaf61190>



Let's find relationship between Balance and Estimated salary

In [ ]:
```
sns.displot([df.EstimatedSalary, df.Balance], kde=True, bins=50)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x1fdab01cc90>



It look like people having no or less salary have less or zero balance

In [ ]:
```
# Let's find relationship between zero balance and churn
```

In [ ]:
```
df_zerobalance_count = df[df.Balance == 0].shape[0]
total_count = df.shape[0]
percentage = (df_zerobalance_count/total_count)*100
print(f" The total of {df[df.Balance == 0].shape[0]} customers have a balance of
```

```
 The total of 3617 customers have a balance of 0 which is 36.17% of our entire da
taset
```

In [ ]:
```
# Let's find relationship between zero Balance and age
zero_balance_age = df[df.Balance == 0].Age.value_counts()
zero_balance_age.describe()
```

Out[ ]:
```
count     67.000000
mean      53.985075
std       58.541488
min        1.000000
25%        8.000000
50%       25.000000
75%       87.000000
max      178.000000
Name: count, dtype: float64
```

In [ ]:
```
# Let's find relationship between zero Balance and geography
zero_balance_geography = df[df.Balance == 0].Geography.value_counts()
zero_balance_geography
```

Out[ ]:
```
Geography
France    2418
Spain     1199
Name: count, dtype: int64
```

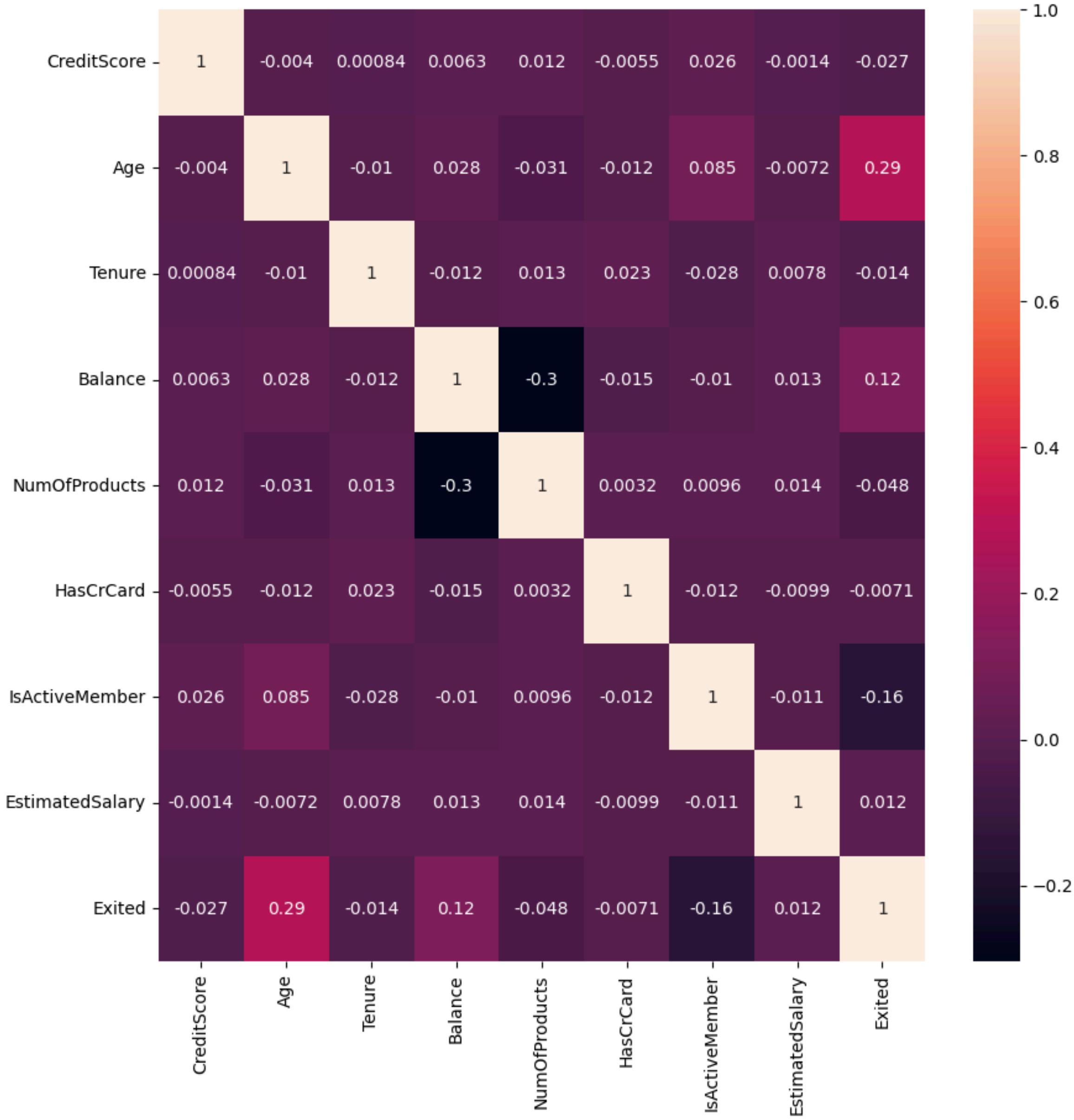
Observations:

In [ ]:
```
print(f"-There are people from France and spain who has zero balance in there ac
```

```
-There are people from France and spain who has zero balance in there account
-In total 36.17% people who have zero balance in their account and left the bank
```

In [ ]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.drop(['Geography', 'Gender'], axis=True).corr(), annot=True)
```

Out[ ]:
```
<Axes: >
```

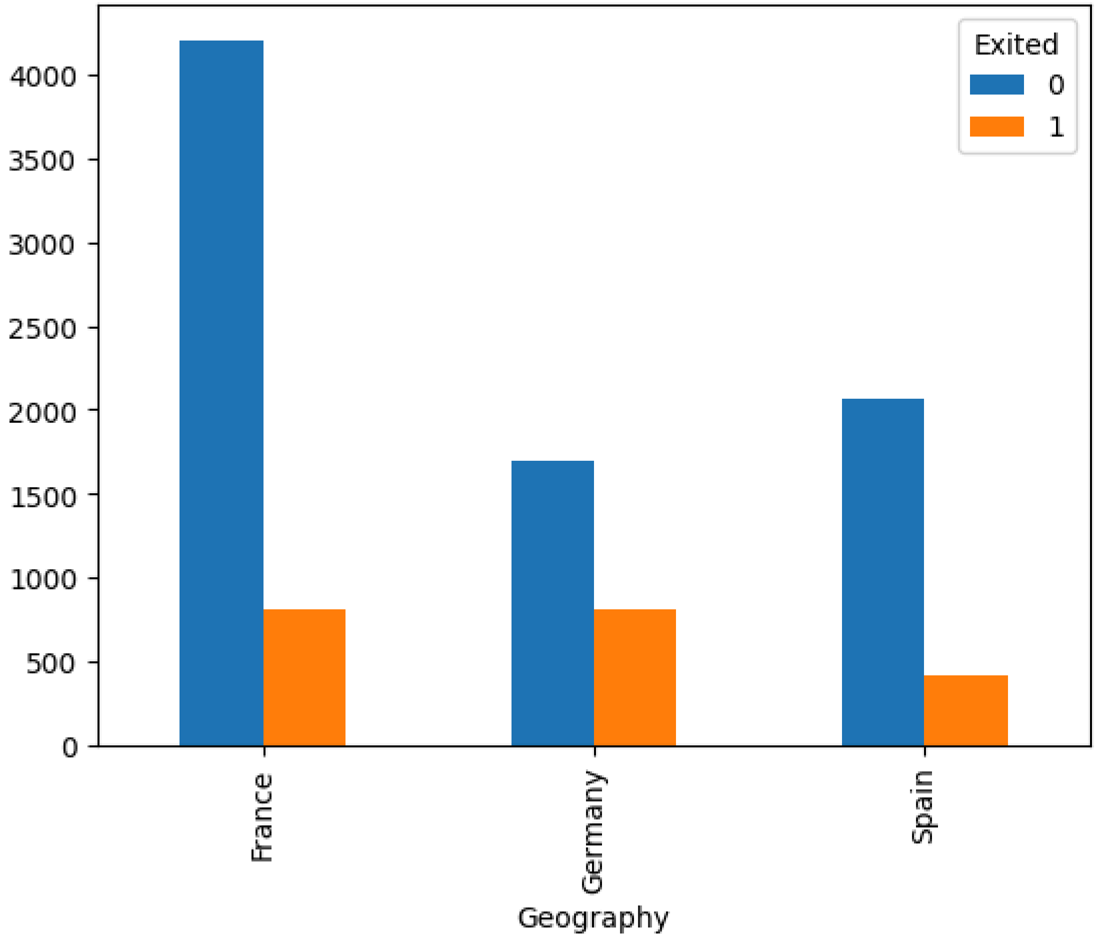


Data Cleaning

Dropping unnecessory columns of data

In [ ]:
```
relationship = pd.crosstab(df['Geography'], df['Exited'])
print(relationship)
relationship.plot(kind='bar', stacked=False, legend=True)
plt.show()
```

```
Exited        0    1
Geography
France     4204  810
Germany    1695  814
Spain      2064  413
```



In [ ]:
```
from scipy.stats import chi2_contingency
chi2_contingency(pd.crosstab(df.Geography, df.Exited))[1]
```

Out[ ]: np.float64(3.8303176053541544e-66)

Since there's no relationship between our target variable with Geography, we can safely remove Geography column

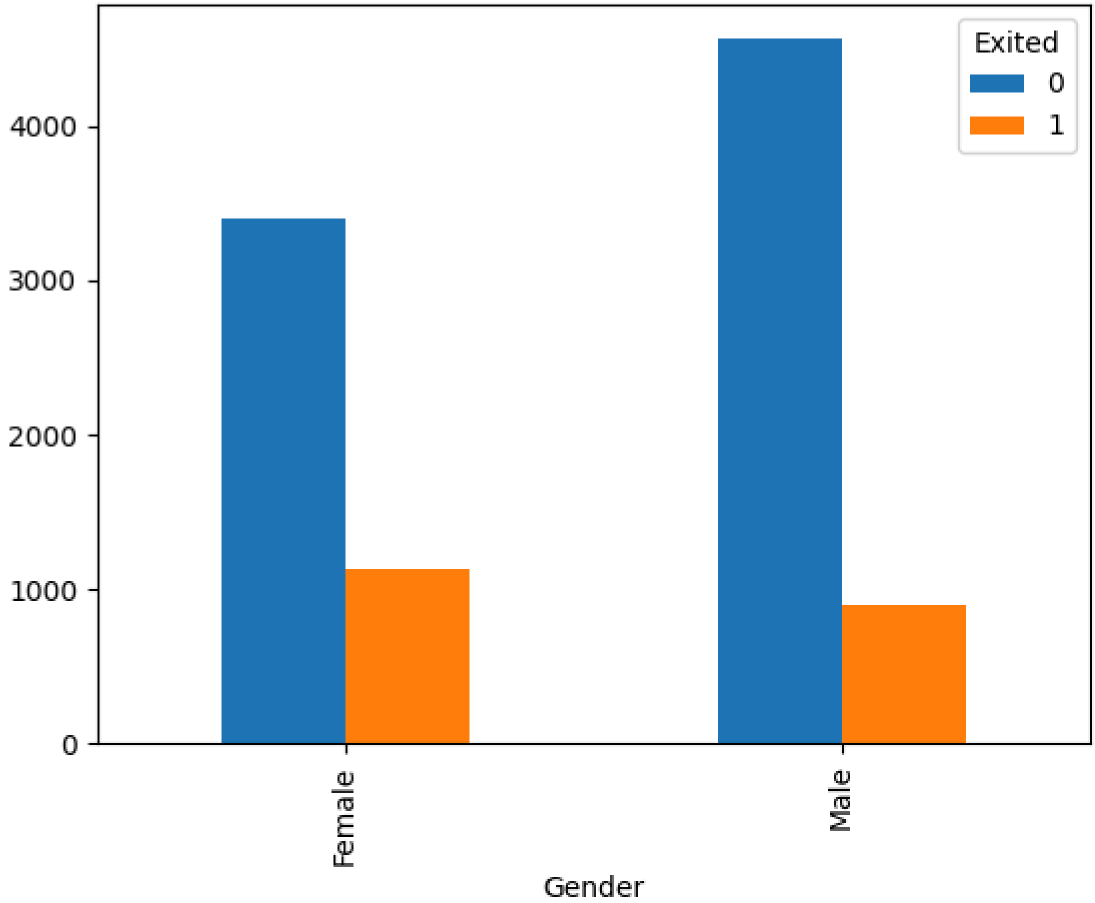
In [ ]:
```
df.drop(['Geography'], axis=1, inplace=True)
df.head()
```

Out[ ]:

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveM
0	619	Female	42	2	0.00	1	1	
1	608	Female	41	1	83807.86	1	0	
2	502	Female	42	8	159660.80	3	1	
3	699	Female	39	1	0.00	2	0	
4	850	Female	43	2	125510.82	1	1	

In [ ]:
```
relationship = pd.crosstab(df['Gender'], df['Exited'])
relationship.plot(kind='bar', stacked=False, )
print(relationship)
```

```
Exited     0     1
Gender
Female  3404  1139
Male    4559   898
```



In [ ]:
```
# Performing chi-squared test
chi2_contingency(pd.crosstab(df.Gender, df.Exited))[1]
```

Out[ ]:
```
np.float64(2.2482100097131755e-26)
```

Since there's no relationship between our target variable with Gender, we can safely remove Gender column

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

Out[ ]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
0	619	42	2	0.00	1	1	1
1	608	41	1	83807.86	1	0	1
2	502	42	8	159660.80	3	1	0
3	699	39	1	0.00	2	0	0
4	850	43	2	125510.82	1	1	1

In [ ]:
```
df.NumOfProducts.value_counts()
```

Out[ ]:
```
NumOfProducts
1    5084
2    4590
3     266
4      60
Name: count, dtype: int64
```

The people who uses the total of 4 services is very less so its better for us to remove it

In [ ]:
```
df = df[df.NumOfProducts < 3]
df.head()
```

Out[ ]:

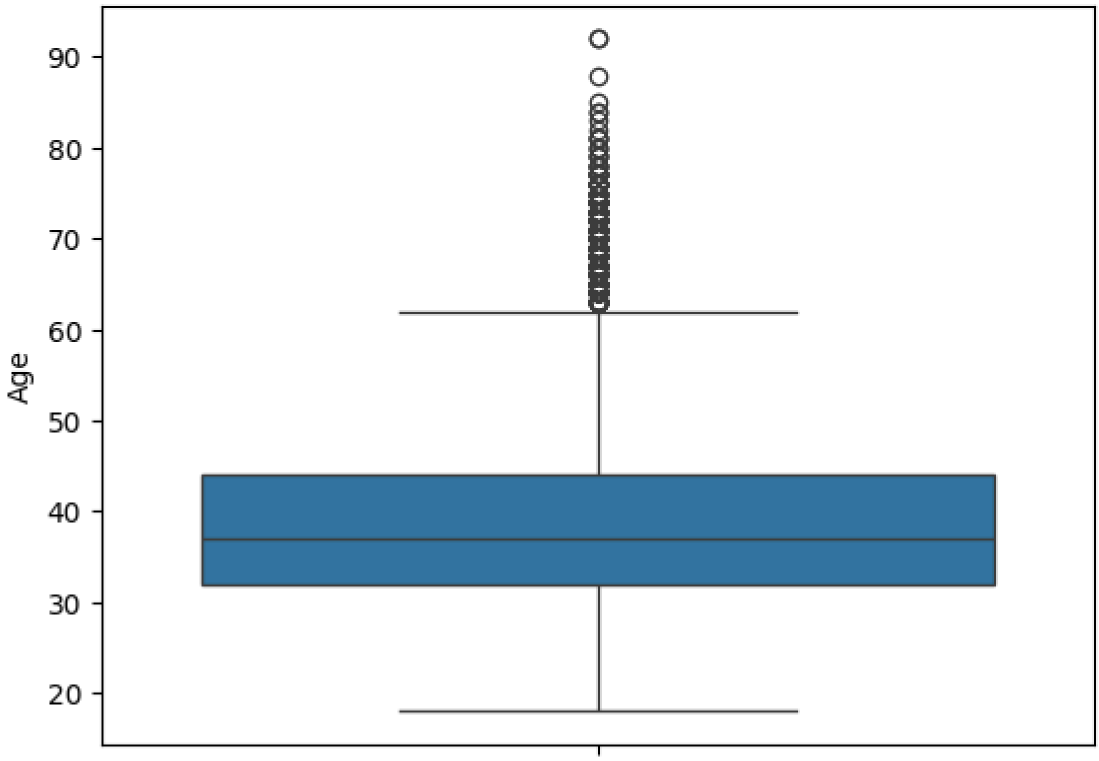
	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
0	619	42	2	0.00	1	1	1
1	608	41	1	83807.86	1	0	1
3	699	39	1	0.00	2	0	0
4	850	43	2	125510.82	1	1	1
5	645	44	8	113755.78	2	1	0

In [ ]:
```
df.reset_index(drop=True, inplace=True)
```

Working with outliers

In [ ]:
```
sns.boxplot(df.Age)
```

Out[ ]: <Axes: ylabel='Age'>



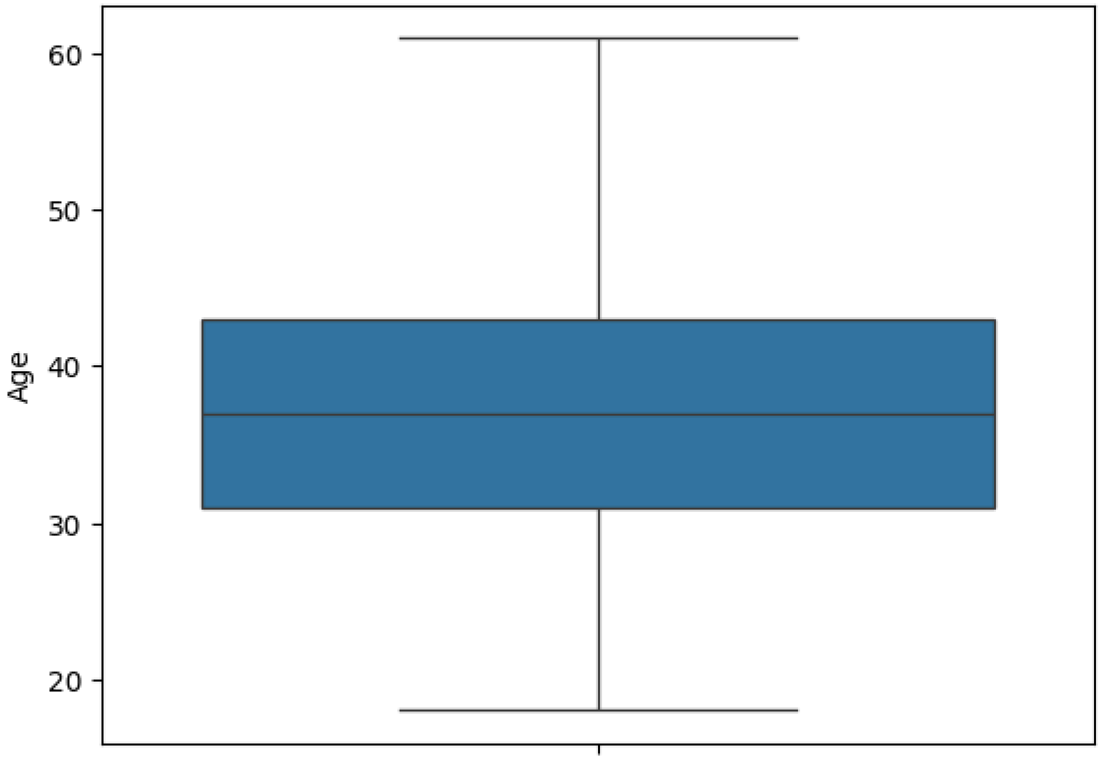
Let's remove ourliers from Age column first

In [ ]:
```
q1_age = df.Age.quantile(.25)
median_age = df.Age.median()
q3_age = df.Age.quantile(.75)
# finding IQR
upperLimit_age= q3_age + 1.5*(q3_age - q1_age)
lowerLimit_age = q1_age - 1.5*(q3_age - q1_age)

print(f"The upper limit of the age: {upperLimit_age} and the lower limit of the
```

```
The upper limit of the age: 62.0 and the lower limit of the age: 14.0
```

In [ ]:
```
df = df[df.Age < upperLimit_age]
sns.boxplot(df.Age)
```

Out[ ]: <Axes: ylabel='Age'>



Let's remove ourliers from CreditScore column

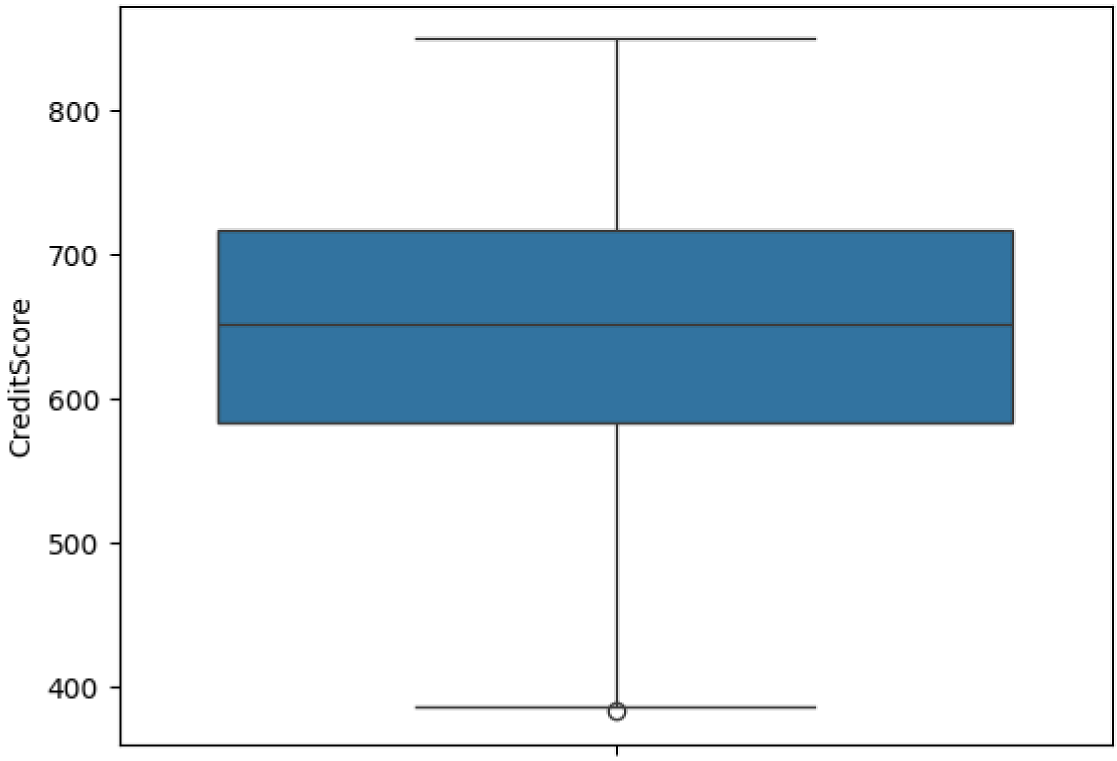
In [ ]:
```
q1_CreditScore = df.CreditScore.quantile(.25)
median_CreditScore = df.CreditScore.median()
q3_CreditScore = df.CreditScore.quantile(.75)
# finding IQR
upperLimit_CreditScore= q3_CreditScore + 1.5*(q3_CreditScore - q1_CreditScore)
lowerLimit_CreditScore = q1_CreditScore - 1.5*(q3_CreditScore - q1_CreditScore)

print(f"The upper limit of the CreditScore: {upperLimit_CreditScore} and the low
```

```
The upper limit of the CreditScore: 918.0 and the lower limit of the CreditScore:
382.0
```

In [ ]:
```
df = df[df.CreditScore > lowerLimit_CreditScore]
```

```
sns.boxplot(df.CreditScore)
```

Out[ ]: <Axes: ylabel='CreditScore'>



Feature Scaling

In [ ]:
```
from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler()
data_scaled = pd.DataFrame(sc.fit_transform(df.drop(['HasCrCard', 'IsActiveMembe
data_scaled.head()
```

Out[ ]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary
0	0.505353	0.558140	0.2	0.000000	0.0	0.506735
1	0.481799	0.534884	0.1	0.351561	0.0	0.562709
2	0.676660	0.488372	0.1	0.000000	1.0	0.469120
3	1.000000	0.581395	0.2	0.526499	0.0	0.395400
4	0.561028	0.604651	0.8	0.477188	1.0	0.748797

In [ ]:
```
df.reset_index(drop=True, inplace=True)
```

In [ ]:
```
df = pd.concat([data_scaled, df.HasCrCard, df.IsActiveMember, df.Exited], axis=1
df.head()
```

Out[ ]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	HasCrCar
0	0.505353	0.558140	0.2	0.000000	0.0	0.506735	
1	0.481799	0.534884	0.1	0.351561	0.0	0.562709	
2	0.676660	0.488372	0.1	0.000000	1.0	0.469120	
3	1.000000	0.581395	0.2	0.526499	0.0	0.395400	
4	0.561028	0.604651	0.8	0.477188	1.0	0.748797	

In [ ]:

```
df
```

Out[ ]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	HasCr
0	0.505353	0.558140	0.2	0.000000	0.0	0.506735	
1	0.481799	0.534884	0.1	0.351561	0.0	0.562709	
2	0.676660	0.488372	0.1	0.000000	1.0	0.469120	
3	1.000000	0.581395	0.2	0.526499	0.0	0.395400	
4	0.561028	0.604651	0.8	0.477188	1.0	0.748797	
...	...	...	...	...	...	...	
9263	0.830835	0.488372	0.5	0.000000	1.0	0.481341	
9264	0.284797	0.395349	1.0	0.240657	0.0	0.508490	
9265	0.698073	0.418605	0.7	0.000000	0.0	0.210390	
9266	0.832976	0.558140	0.3	0.314930	1.0	0.464429	
9267	0.875803	0.232558	0.4	0.545929	0.0	0.190914	

9268 rows × 9 columns

Feature Engineering

In [ ]:

```
df.describe()
```

Out[ ]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedS
count	9268.000000	9268.000000	9268.000000	9268.000000	9268.000000	9268.0(
mean	0.573279	0.453313	0.501554	0.320777	0.476155	0.5(
std	0.205919	0.199735	0.288785	0.261847	0.499458	0.2{
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.0(
25%	0.430407	0.302326	0.300000	0.000000	0.000000	0.2!
50%	0.576017	0.441860	0.500000	0.407224	0.000000	0.5(
75%	0.715203	0.581395	0.700000	0.535348	1.000000	0.7/
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.0(

From the above observation we can say the columns and "NumOfProducts" need feature engineering

In [ ]:
```
df = pd.get_dummies(df, columns=['NumOfProducts'], dtype=int)
df.reset_index(drop=True, inplace=True)
df.head()
```

Out[ ]:

	CreditScore	Age	Tenure	Balance	EstimatedSalary	HasCrCard	IsActiveMembe
0	0.505353	0.558140	0.2	0.000000	0.506735	1	
1	0.481799	0.534884	0.1	0.351561	0.562709	0	
2	0.676660	0.488372	0.1	0.000000	0.469120	0	
3	1.000000	0.581395	0.2	0.526499	0.395400	1	
4	0.561028	0.604651	0.8	0.477188	0.748797	1	

In [ ]:
```
df = df.rename(columns={'NumOfProducts_0.0':'NumOfProducts_1', 'NumOfProducts_0.
df.head()
```

Out[ ]:

	CreditScore	Age	Tenure	Balance	EstimatedSalary	HasCrCard	IsActiveMembe
0	0.505353	0.558140	0.2	0.000000	0.506735	1	
1	0.481799	0.534884	0.1	0.351561	0.562709	0	
2	0.676660	0.488372	0.1	0.000000	0.469120	0	
3	1.000000	0.581395	0.2	0.526499	0.395400	1	
4	0.561028	0.604651	0.8	0.477188	0.748797	1	

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

Out[ ]: (9268, 10)

Handling Data Imbalance

In [ ]:
```
df.Exited.value_counts()
```

Out[ ]:
```
Exited
0    7604
1    1664
Name: count, dtype: int64
```

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

Out[ ]:
```
Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'EstimatedSalary',
       'HasCrCard', 'IsActiveMember', 'Exited', 'NumOfProducts_1',
       'NumOfProducts_1.0'],
      dtype='object')
```

It looks like our dataset is extreamly imbalanced. So let's make it balanced by using SMOTE method

In [ ]:
```
from imblearn.over_sampling import SMOTE
sm = SMOTE()
x, y = sm.fit_resample(df.drop(['Exited'], axis=1), df.Exited)

df_balanced = pd.DataFrame(x, columns= ['CreditScore','Age', 'Tenure', 'Balance'
df_balanced['Exited'] = y
```

In [ ]:
```
df_balanced.Exited.value_counts()
```

Out[ ]:
```
Exited
1    7604
0    7604
Name: count, dtype: int64
```

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

Out[ ]:

	CreditScore	Age	Tenure	Balance	EstimatedSalary	HasC
CreditScore	1.000000	-0.011430	0.000881	0.006623	0.003393	-0.0
Age	-0.011430	1.000000	-0.014453	0.041543	-0.005612	-0.0
Tenure	0.000881	-0.014453	1.000000	-0.017085	0.010656	0.0
Balance	0.006623	0.041543	-0.017085	1.000000	0.009381	-0.0
EstimatedSalary	0.003393	-0.005612	0.010656	0.009381	1.000000	-0.0
HasCrCard	-0.003122	-0.011923	0.020608	-0.014401	-0.013159	1.0
IsActiveMember	0.021880	0.016354	-0.030989	-0.005491	-0.010217	-0.0
Exited	-0.015295	0.346740	-0.015465	0.113509	0.004935	-0.0
NumOfProducts_1	-0.011977	0.108527	-0.018063	0.373621	-0.006127	-0.0
NumOfProducts_1.0	0.011977	-0.108527	0.018063	-0.373621	0.006127	0.0

Dimensionality reduction

In [ ]:

```
sample = df_balanced.sample(10)
```

In [ ]:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=9)
principalComponents = pca.fit_transform(sample.drop(['Exited'], axis=1))
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['PCA 1', 'PCA 2', 'PCA 3', 'PCA 4', 'PCA 5', 'PCA 6',
principalDf
```

Out[ ]:

	PCA 1	PCA 2	PCA 3	PCA 4	PCA 5	PCA 6	PCA 7	PCA 8
0	-0.539927	-0.303986	0.475473	0.212445	0.318477	0.101628	0.022936	0.013850
1	-0.777634	-0.399645	-0.827473	0.261080	0.052585	-0.080196	-0.000487	0.000644
2	1.014358	0.438309	0.062581	0.176591	-0.000060	-0.188576	-0.130689	-0.010095
3	0.990014	0.413844	-0.147753	0.176727	-0.139885	0.152756	0.016335	0.027022
4	-0.613833	-0.201908	0.188803	-0.018175	-0.104891	0.164459	-0.198324	-0.012200
5	-0.473228	0.739126	-0.249897	-0.291764	-0.090969	0.162761	0.063001	-0.010469
6	-0.389880	0.694851	0.166942	-0.040130	0.302605	-0.132112	0.066751	-0.006150
7	-0.753053	-0.153093	0.323491	-0.114353	-0.405941	-0.196474	0.038869	0.014266
8	0.783155	-0.641217	-0.154737	-0.514379	0.194685	-0.038968	-0.028315	0.008152
9	0.760028	-0.586282	0.162570	0.151958	-0.126605	0.054721	0.149922	-0.025020

In [ ]: