

Assignment -2
Python Programming

Assignment Date	29 September 2022
Student Name	Mr. Sukumar R
Student Roll Number	812419104309
Maximum Marks	2 Marks

Assignment 2

Question-1:

Load the dataset

Data Visualization & Pre-processing

Loading Dataset

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

```
df=pd.read_csv(r'/content/Churn_Modelling.csv')
```

```
df.shape
```

```
(10000, 14)
```

```
df.columns
```

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

```
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()

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	\
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

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	10000.000000	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

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
```

Question-2:

Perform Below Visualizations.

- **Univariate Analysis**
- **Bi - Variate Analysis**
- **Multi - Variate Analysis**

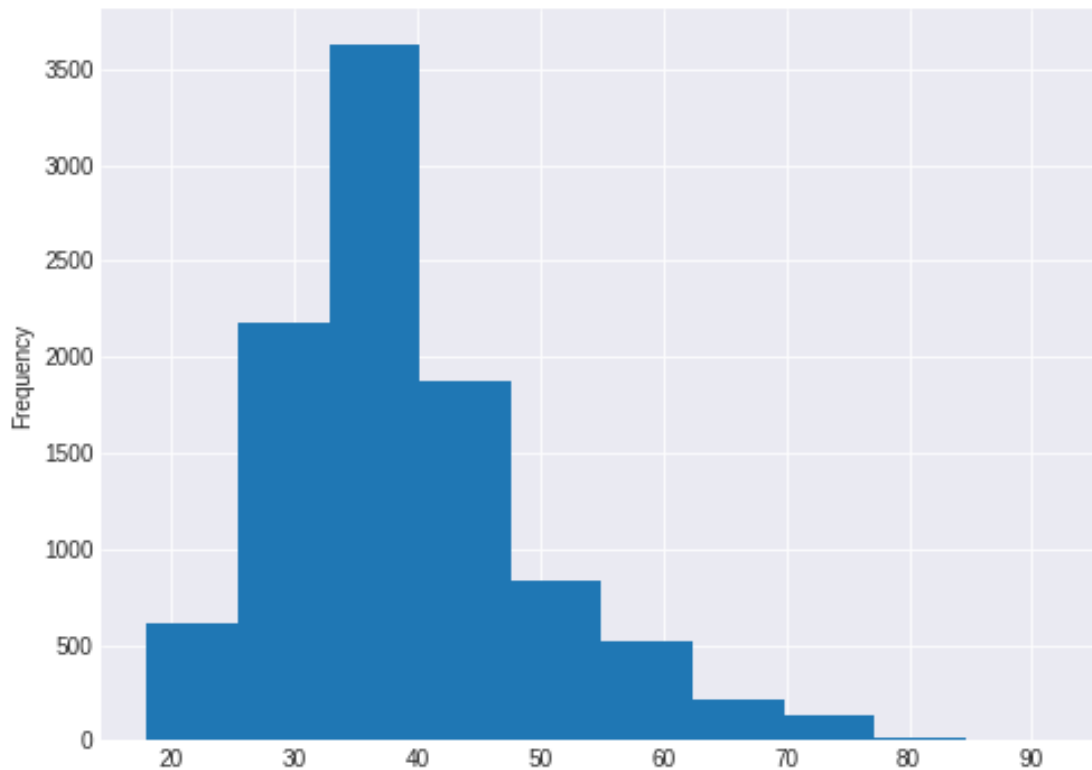
VISUALIZATION

1. UNI-VARIATE ANALYSIS

```
from matplotlib import pyplot as plt  
plt.style.use('seaborn-darkgrid')
```

```
plt.figure(figsize=(8,6))  
df.Age.plot(kind='hist')
```

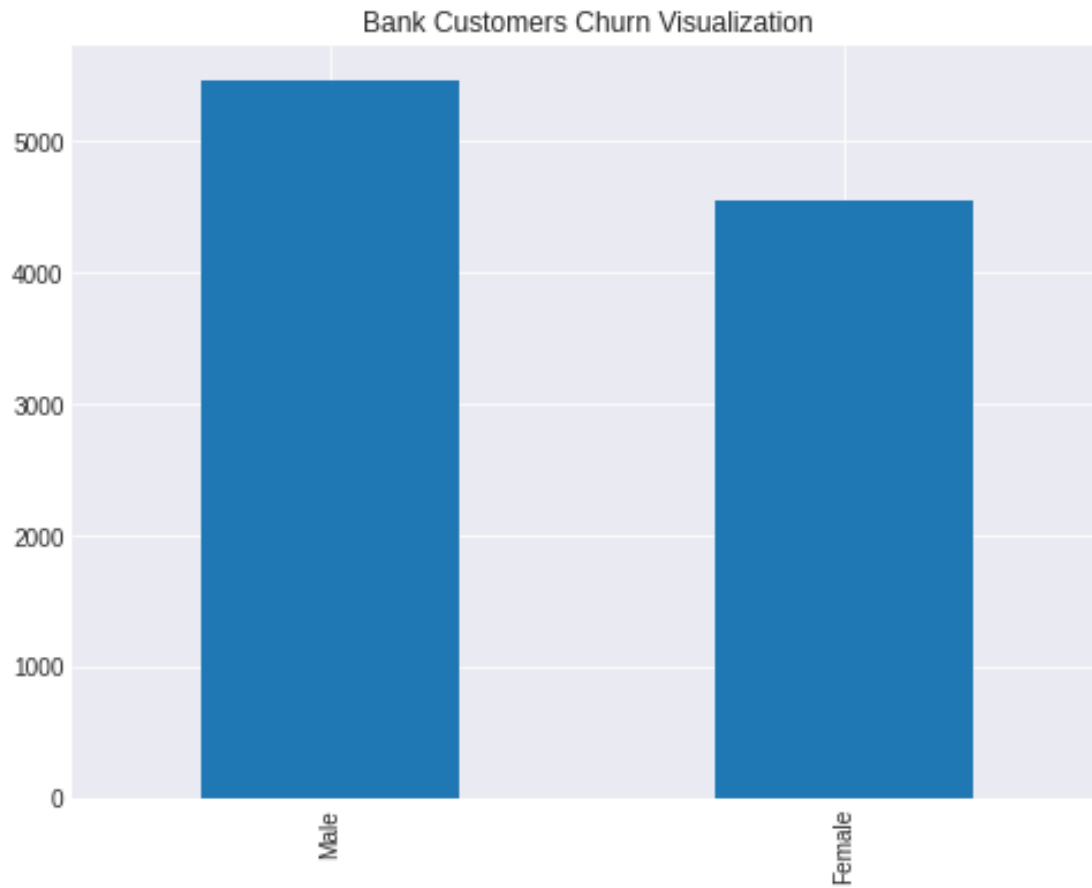
<matplotlib.axes._subplots.AxesSubplot at 0x7f8acb57b9d0>



```
plt.figure(figsize=(8,6))  
df.NumOfProducts.value_counts().plot(kind='bar')
```

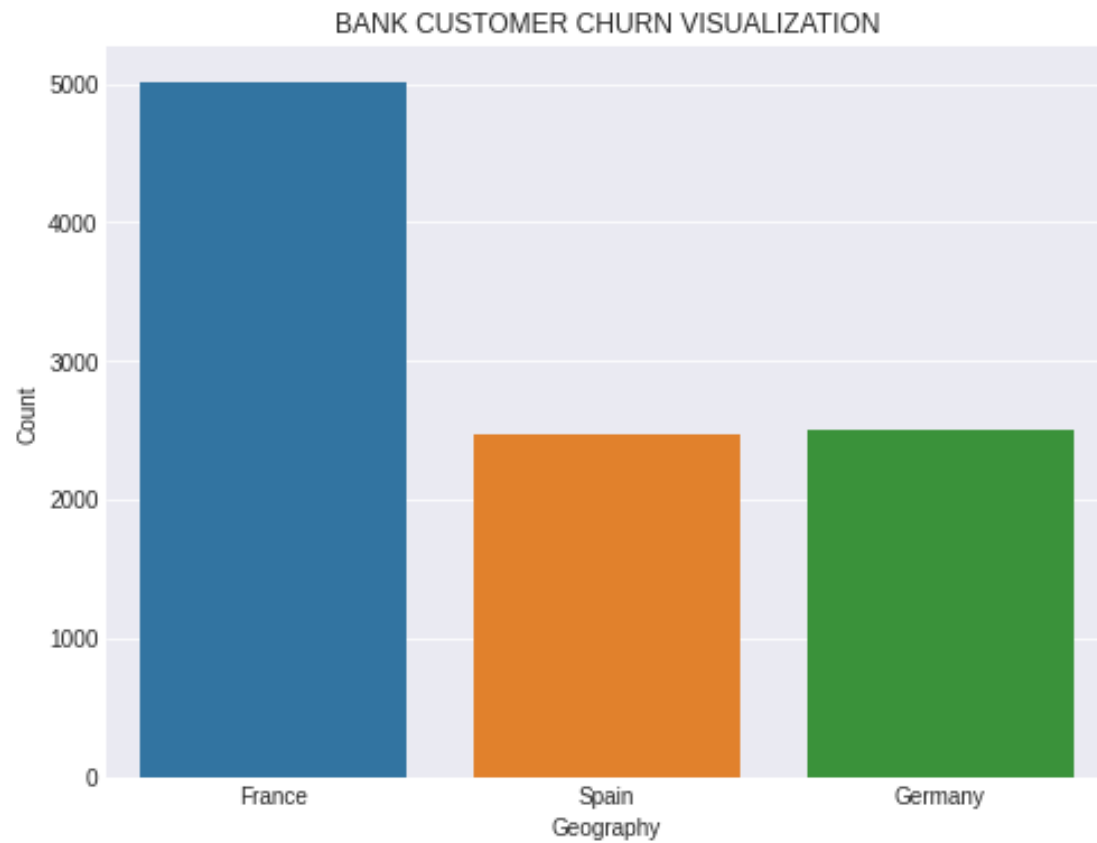
```
plt.figure(figsize=(8,6))  
df.Gender.value_counts().plot(kind='bar')  
plt.title('Bank Customers Churn Visualization')
```

```
Text(0.5, 1.0, 'Bank Customers Churn Visualization')
```

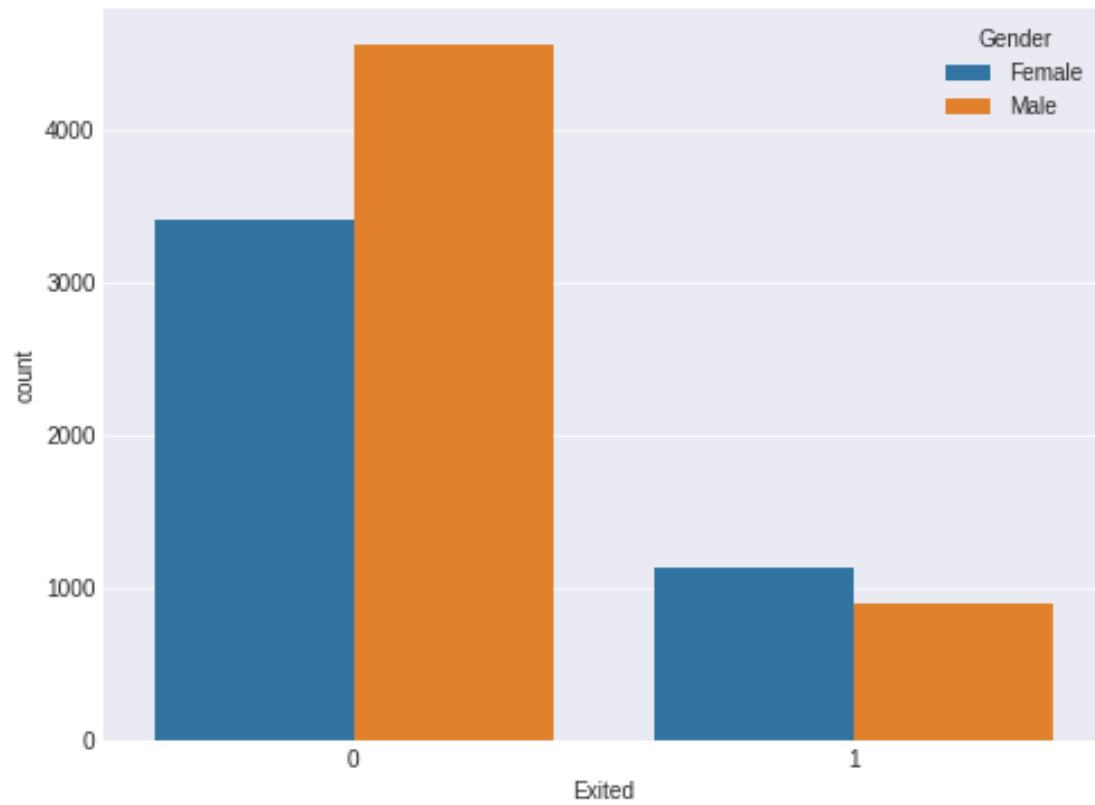


```
#COUNT PLOT
plt.figure(figsize=(8,6))
sns.countplot(x='Geography',data=df)
plt.xlabel('Geography')
plt.ylabel('Count')
plt.title('BANK CUSTOMER CHURN VISUALIZATION')

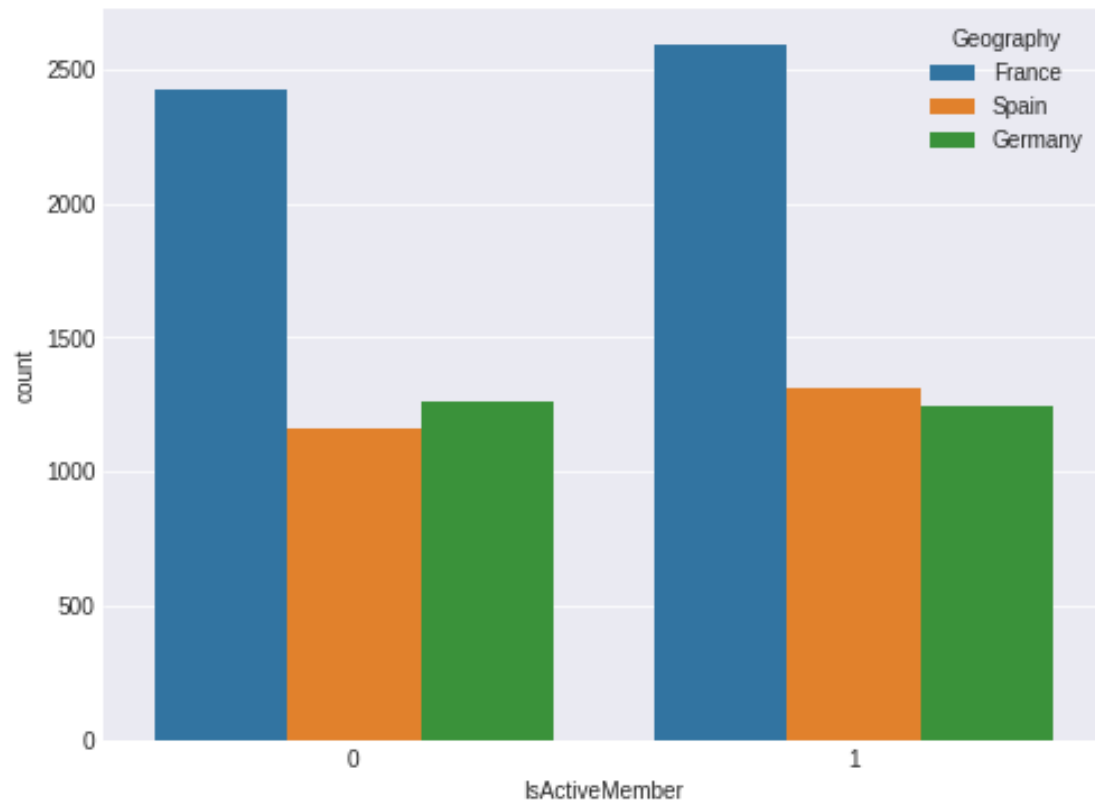
Text(0.5, 1.0, 'BANK CUSTOMER CHURN VISUALIZATION')
```



```
plt.figure(figsize=(8,6))  
sns.countplot(x=df.Exited,hue=df.Gender)  
<matplotlib.axes._subplots.AxesSubplot at 0x7f8acadc8090>
```

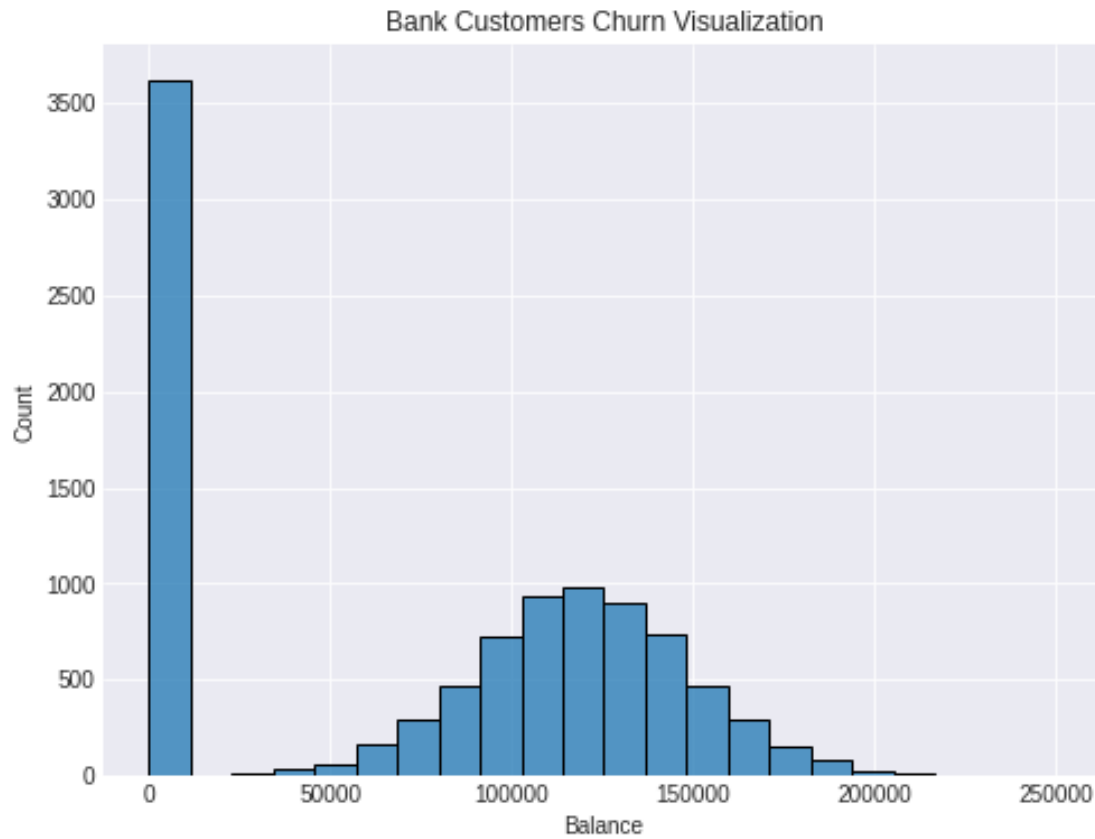


```
plt.figure(figsize=(8,6))
sns.countplot(x=df.IsActiveMember,hue=df.Geography)
<matplotlib.axes._subplots.AxesSubplot at 0x7f8acad8810>
```

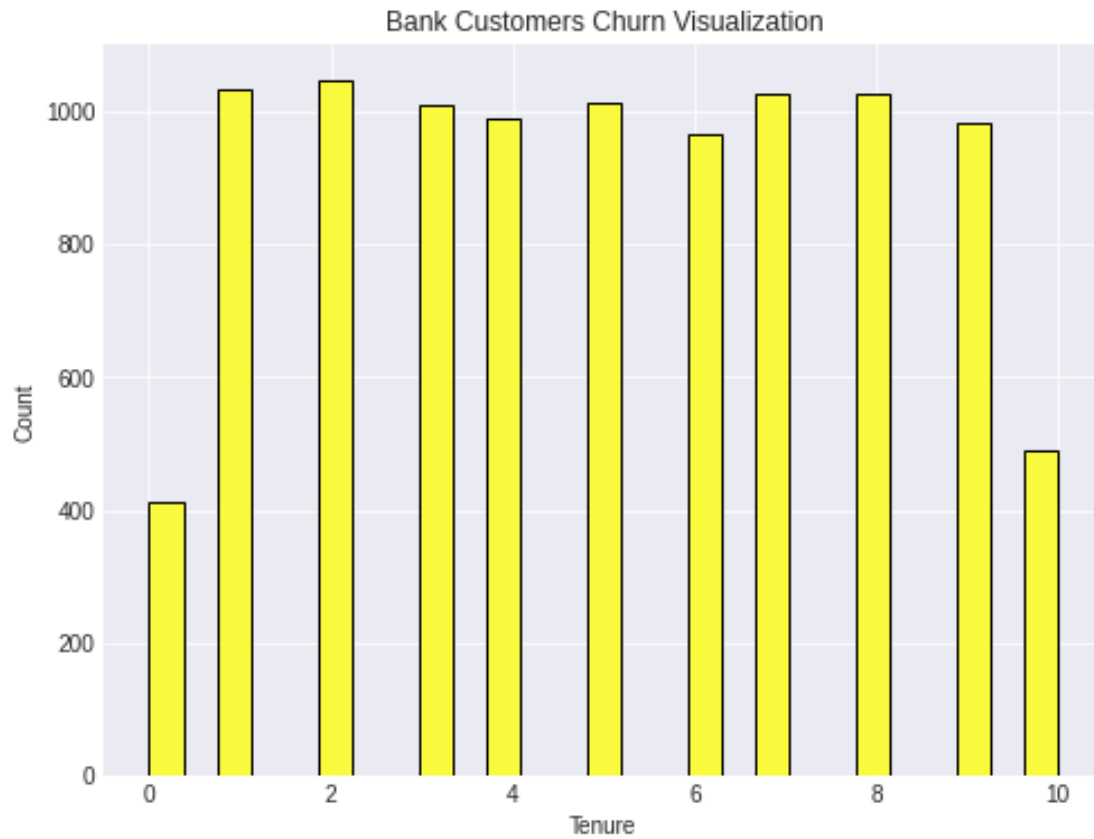


```
#HISTOGRAM
plt.figure(figsize=(8,6))
sns.histplot(df.Balance)
plt.xlabel('Balance')
plt.ylabel('Count')
plt.title('Bank Customers Churn Visualization')

Text(0.5, 1.0, 'Bank Customers Churn Visualization')
```

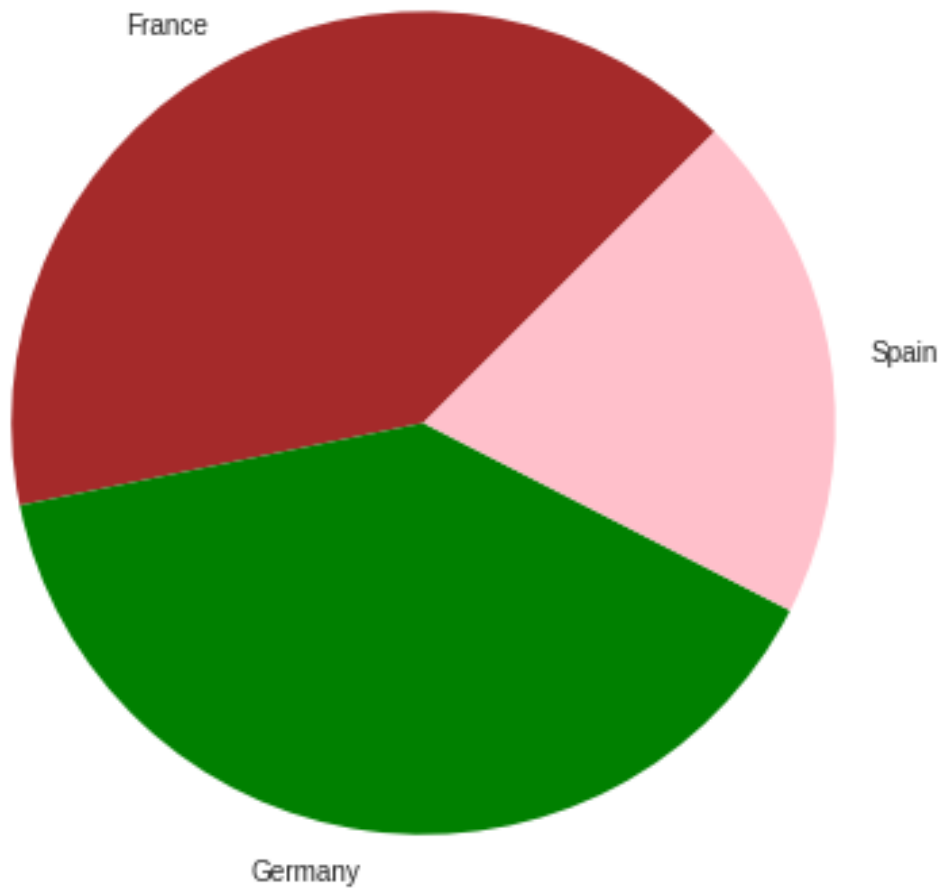



```
plt.figure(figsize=(8,6))
sns.histplot(df['Tenure'],kde=False,color='Yellow')
plt.xlabel('Tenure')
plt.ylabel('Count')
plt.title('Bank Customers Churn Visualization')
Text(0.5, 1.0, 'Bank Customers Churn Visualization')
```



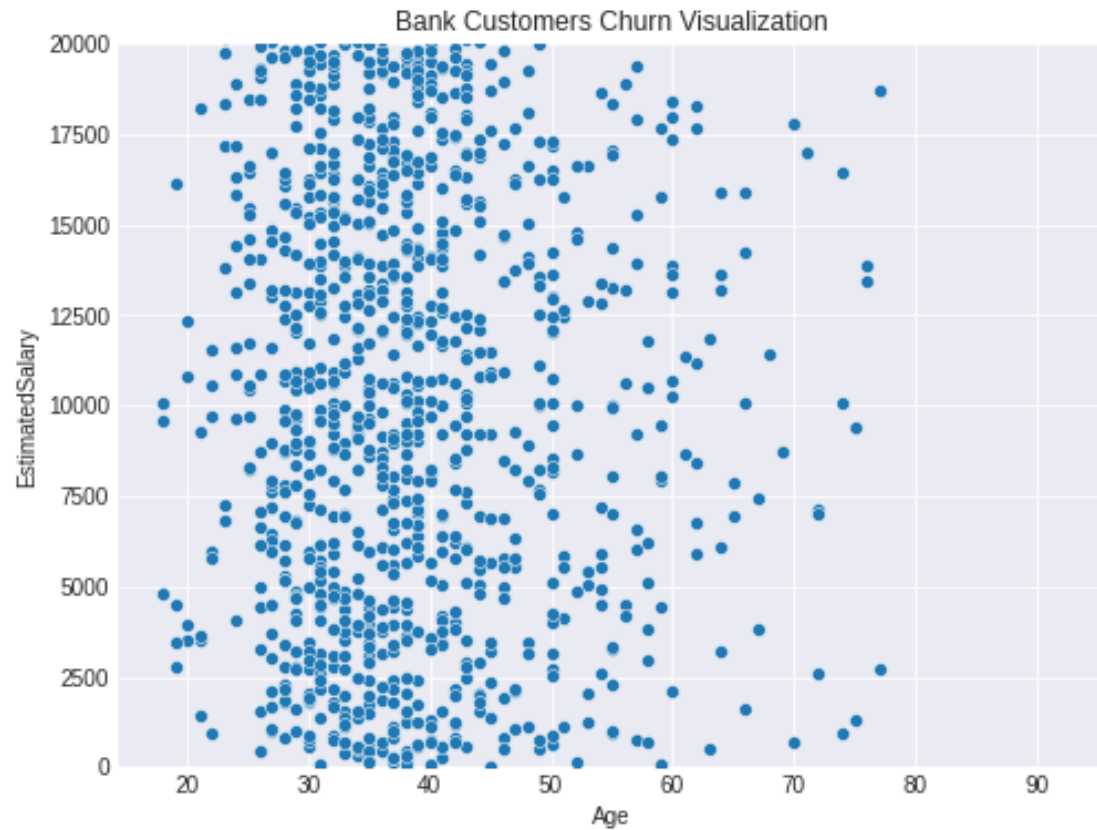
```
labels = 'France', 'Germany', 'Spain'  
colors = ['brown', 'green', 'pink']  
area = [311, 300, 153]  
plt.figure(figsize =(8, 7))  
plt.pie(area, colors = colors, labels = labels,startangle=45)  
plt.title('Bank Customers Churn Visualization')  
  
Text(0.5, 1.0, 'Bank Customers Churn Visualization')
```

Bank Customers Churn Visualization



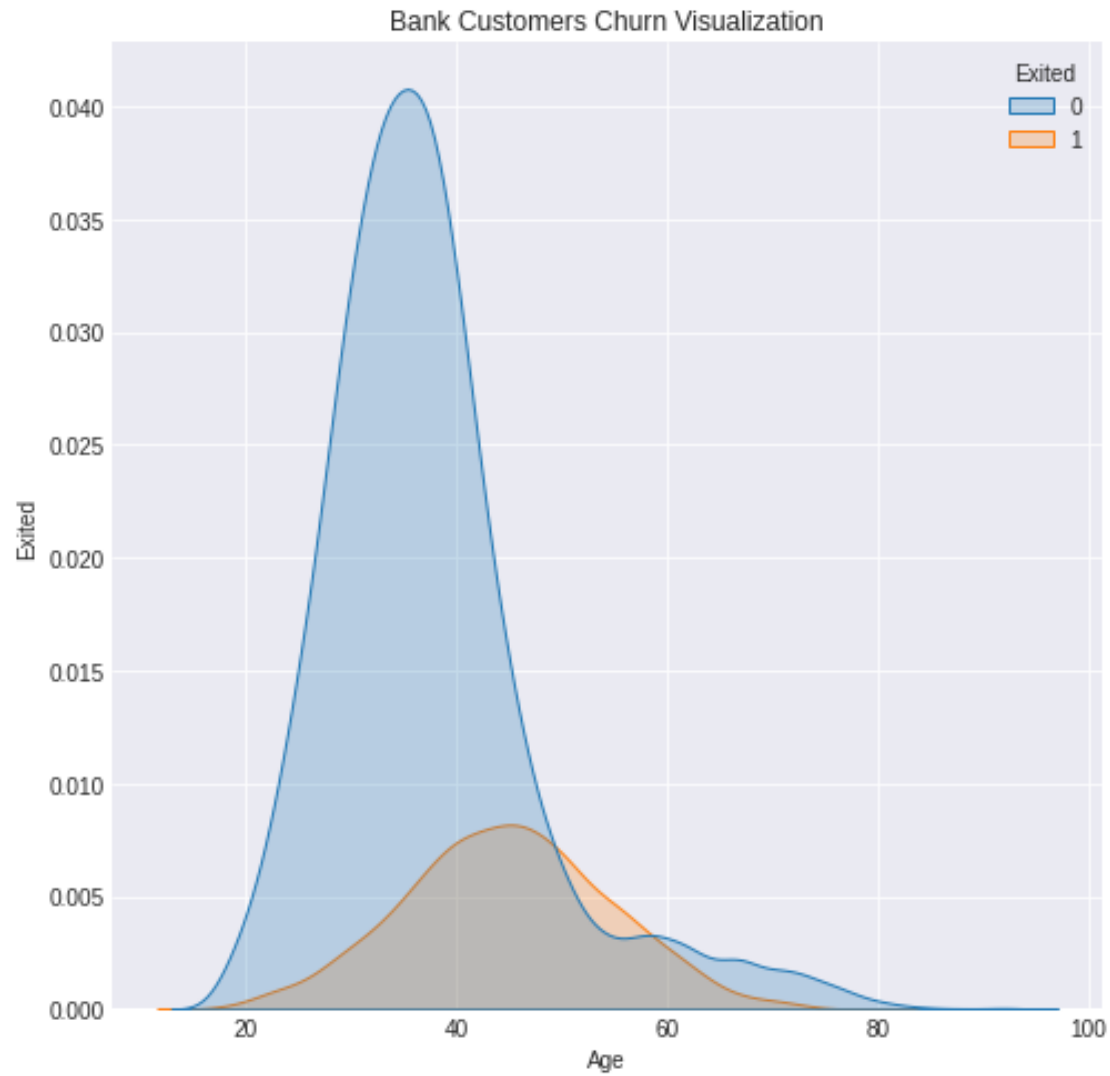
2.BI-VARIATE ANALYSIS

```
plt.figure(figsize=(8,6))
sns.scatterplot(x=df.Age,y=df.EstimatedSalary)
plt.ylim(0,20000)
plt.title('Bank Customers Churn Visualization')
Text(0.5, 1.0, 'Bank Customers Churn Visualization')
```



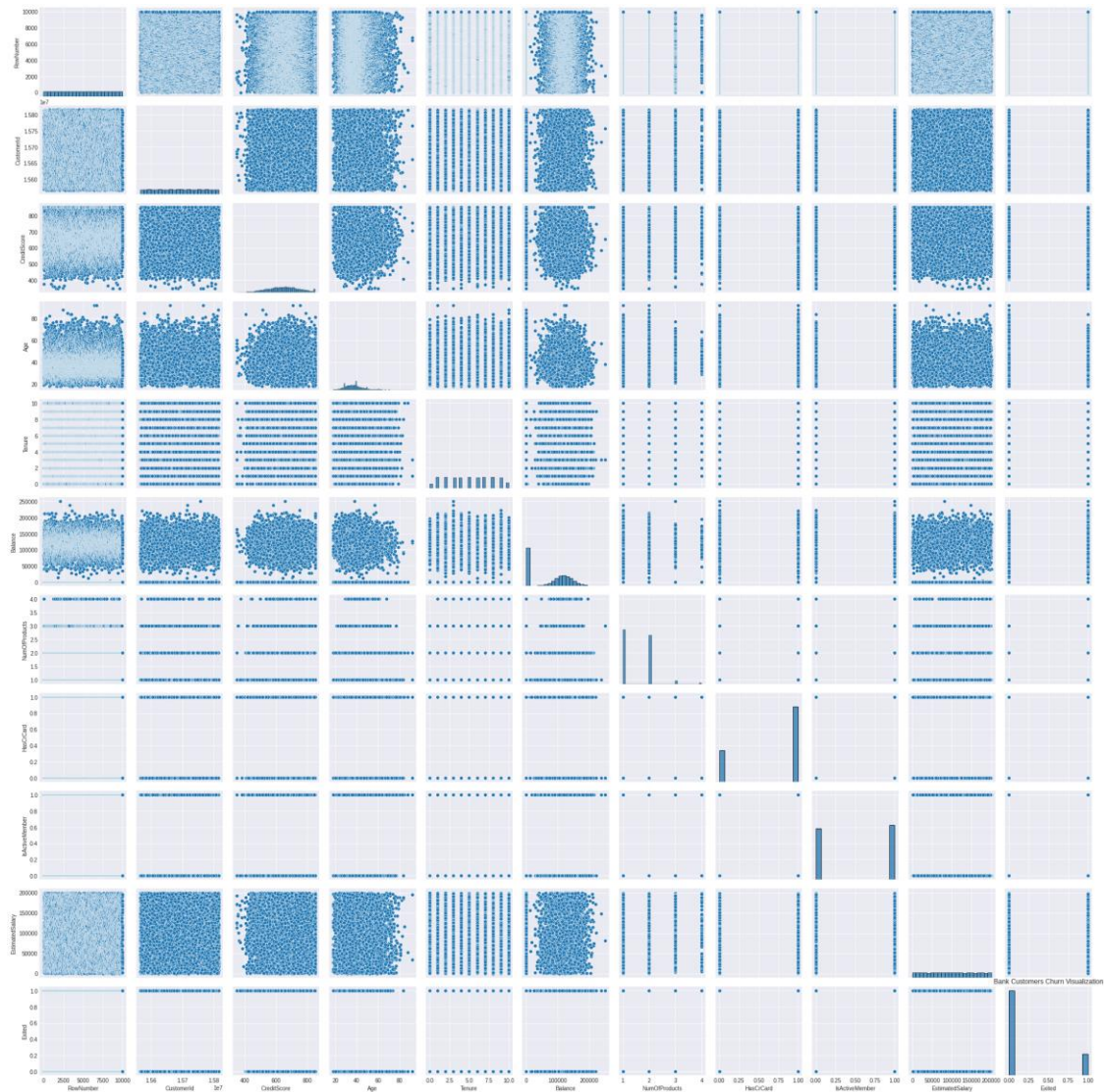
```
plt.figure(figsize=(8,8))
sns.kdeplot(data=df,x='Age',hue='Exited',fill=True)
plt.xlabel('Age')
plt.ylabel('Exited')
plt.title('Bank Customers Churn Visualization')

Text(0.5, 1.0, 'Bank Customers Churn Visualization')
```



3.MULTI-VARITE ANALYSIS

```
sns.pairplot(df)
plt.title('Bank Customers Churn Visualization')
Text(0.5, 1.0, 'Bank Customers Churn Visualization')
```



Question-3:

Perform descriptive statistics on the dataset

DESCRIPTIVE STATISTICS

```
import statistics as sts
sts.mean(df.EstimatedSalary)
```

100090.239881

```
sts.median(df.CreditScore)
```

652.0

```
sts.mode(df.Geography)
```

```
{"type":"string"}
```

```
sts.variance(df.Age)
```

```
109.99408416841685
```

```
sts.stdev(df.Tenure)
```

```
2.8921743770496837
```

```
#RANGE
```

```
max(df.Balance)-min(df.Balance)
```

```
250898.09
```

Question-4:

HANDLING THE MISSING VALUES

```
print(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
```

```
df.columns
```

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

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

```
df.columns
```

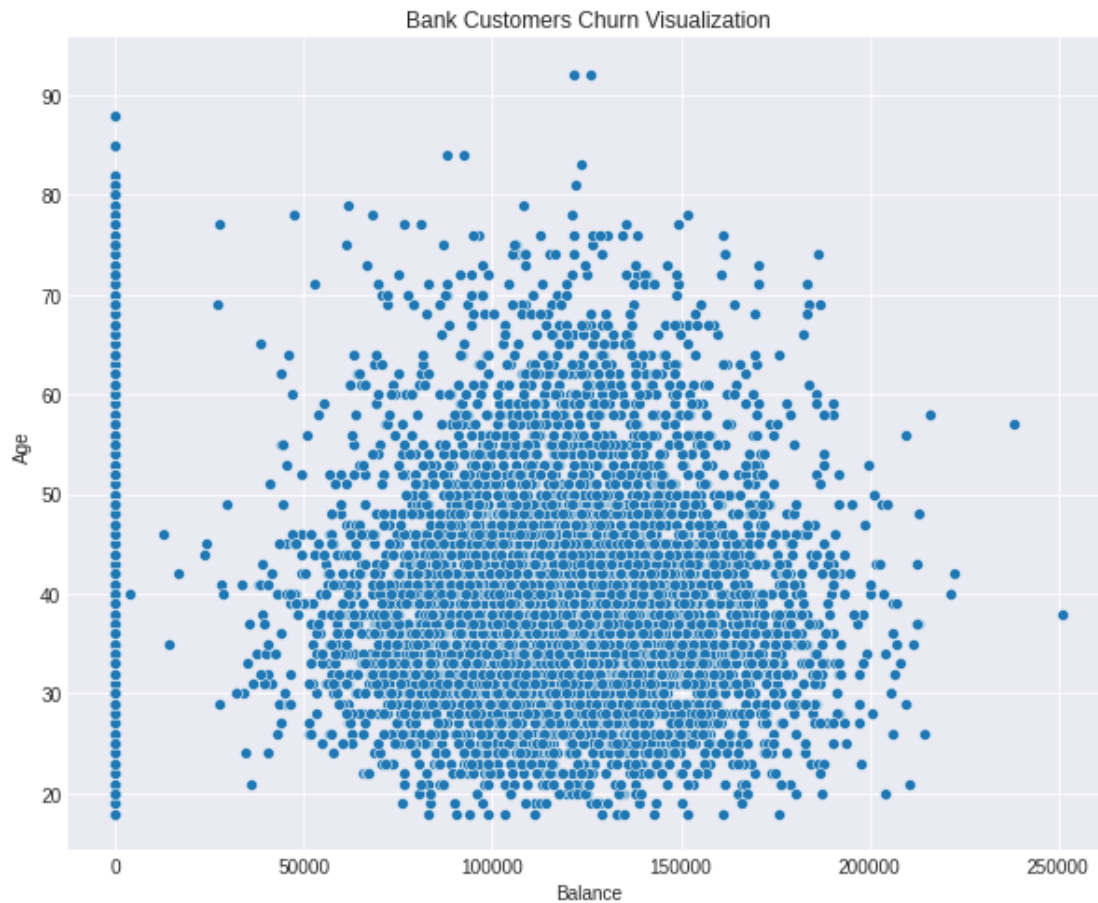
```
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance',  
      'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary',  
      'Exited'],  
      dtype='object')
```

Question-5:

FIND THE OUTLIERS & REPLACE THE OUTLIERS

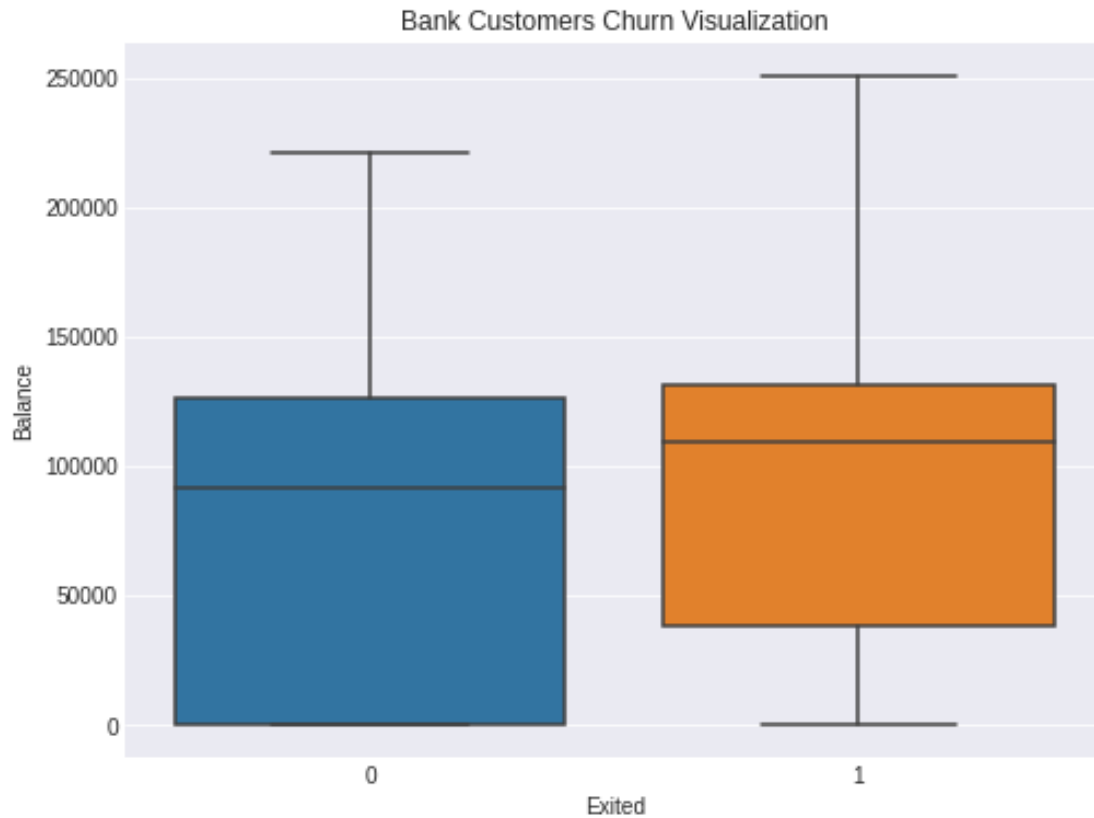
```
plt.figure(figsize=(10,8))  
sns.scatterplot(x=df.Balance,y=df.Age)  
plt.title('Bank Customers Churn Visualization')
```

```
Text(0.5, 1.0, 'Bank Customers Churn Visualization')
```



```
plt.figure(figsize=(8,6))  
sns.boxplot(x=df.Exited,y=df.Balance)  
plt.title('Bank Customers Churn Visualization')
```

```
Text(0.5, 1.0, 'Bank Customers Churn Visualization')
```

```
df.Age.describe()
```

```
count      10000.000000
mean         38.921800
std          10.487806
min          18.000000
25%          32.000000
50%          37.000000
75%          44.000000
max          92.000000
Name: Age, dtype: float64
```

```
median = df.loc[df['Age'] < 50, 'Age'].median()
df.loc[df.Age > 50, 'Age'] = np.nan
df.fillna(median, inplace=True)
df.Age.describe()
```

```
count      10000.000000
mean         35.948900
std           6.454739
min          18.000000
25%          32.000000
50%          36.000000
75%          40.000000
```

```
max          50.000000
Name: Age, dtype: float64
```

Question-6:

CHECK FOR CATEGORICAL COLUMNS & PERFORM ENCODING

```
df.columns
```

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

#LABEL ENCODING

```
from sklearn.preprocessing import LabelEncoder
from collections import Counter as count
le=LabelEncoder()
print('Before label Encoding: ',count(df['Geography']))
df['Geography']=le.fit_transform(df["Geography"])
print('After label Encoding: ',count(df["Geography"]))
```

```
Before label Encoding:  Counter({'France': 5014, 'Germany': 2509, 'Spain':
2477})
```

```
After label Encoding:  Counter({0: 5014, 1: 2509, 2: 2477})
```

```
print('Before Replace: ',count(df["Exited"]))
df['Exited']=df['Exited'].replace([0,1],['No','Yes'])
print('After Replace: ',count(df['Exited']))
```

```
Before Replace:  Counter({0: 7963, 1: 2037})
```

```
After Replace:  Counter({'No': 7963, 'Yes': 2037})
```

```
df.shape
```

```
(10000, 11)
```

#One Hot Encoder

```
from sklearn.preprocessing import OneHotEncoder
df1=OneHotEncoder()
df2=df1.fit_transform(df[['Gender','Tenure']])
```

```
df2.shape
```

```
(10000, 13)
```

Question-7:

SPLIT THE DATA INTO DEPENDENT & INDEPENDENT VARIABLES

#INDEPENDENT VARIABLES

```
x=df.iloc[:,0:10]
print(x)
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts
\							
0	619	0	Female	42.0	2	0.00	1
1	608	2	Female	41.0	1	83807.86	1
2	502	0	Female	42.0	8	159660.80	3
3	699	0	Female	39.0	1	0.00	2
4	850	2	Female	43.0	2	125510.82	1
...
9995	771	0	Male	39.0	5	0.00	2
9996	516	0	Male	35.0	10	57369.61	1
9997	709	0	Female	36.0	7	0.00	1
9998	772	1	Male	42.0	3	75075.31	2
9999	792	0	Female	28.0	4	130142.79	1

	HasCrCard	IsActiveMember	EstimatedSalary
0	1	1	101348.88
1	0	1	112542.58
2	1	0	113931.57
3	0	0	93826.63
4	1	1	79084.10
...
9995	1	0	96270.64
9996	1	1	101699.77
9997	0	1	42085.58
9998	1	0	92888.52
9999	1	0	38190.78

[10000 rows x 10 columns]

#DEPENDENT VARIABLES

```
y=df.iloc[:,10]
print(y)
```

0	Yes
1	No
2	Yes
3	No
4	No
...	
9995	No
9996	No
9997	Yes
9998	Yes
9999	No

Name: Exited, Length: 10000, dtype: object

Question-8:

SCALE THE INDEPENDENT VARIABLES

```
x = pd.get_dummies(x)
x.head()
```

	CreditScore	Geography	Age	Tenure	Balance	NumOfProducts	HasCrCard
0	619	0	42.0	2	0.00	1	1
1	608	2	41.0	1	83807.86	1	0
2	502	0	42.0	8	159660.80	3	1
3	699	0	39.0	1	0.00	2	0
4	850	2	43.0	2	125510.82	1	1

	IsActiveMember	EstimatedSalary	Gender_Female	Gender_Male
0	1	101348.88	1	0
1	1	112542.58	1	0
2	0	113931.57	1	0
3	0	93826.63	1	0
4	1	79084.10	1	0

```
x.shape
```

```
(10000, 11)
```

```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

```
x_train = pd.DataFrame(x_train)
x_train.head()
```

	0	1	2	3	4	5	6	\
0	-0.735507	0.312661	0.480615	0.008860	0.673160	2.535034	-1.553624	
1	1.024427	-0.892353	-0.600877	0.008860	-1.207724	0.804242	0.643657	
2	0.808295	1.517675	-0.291879	1.393293	-0.356937	0.804242	0.643657	
3	0.396614	1.517675	0.326116	0.008860	-0.009356	-0.926551	0.643657	
4	-0.467915	-0.892353	0.017118	0.701077	-1.207724	0.804242	0.643657	

	7	8	9	10
0	-1.034460	-1.640810	1.087261	-1.087261
1	-1.034460	-0.079272	1.087261	-1.087261
2	0.966688	-0.996840	1.087261	-1.087261
3	0.966688	-1.591746	-0.919743	0.919743
4	0.966688	1.283302	-0.919743	0.919743

Question-9:

SPLIT THE DATA INTO TRAINING & TESTING

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

```
print('x_train size: {}'.format(x_train.shape))
print('y_train size: {}'.format(y_train.shape))
print('x_test size: {}'.format(x_test.shape))
print('y_test size: {}'.format(y_test.shape))
```

```
x_train size: (7500, 11)
y_train size: (7500,)
x_test size: (2500, 11)
y_test size: (2500,)
```