In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df= pd.read_csv("C:\\Users\\ASHISH TIWARI\\Downloads\\Churn_Modelling.csv")
df
```

Out[2]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProduct
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	
...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	

10000 rows × 14 columns

In [3]:
```
df.head()
```

Out[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	H
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	

In [4]:
```
df.tail()
```

Out[4]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProduct
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	

In [5]:
```
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 [6]:
```
df.dtypes
```

Out[6]:
```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

In [7]:
```
df.describe(include='all')
```

Out[7]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	I
count	10000.00000	1.000000e+04	10000	10000.000000	10000	10000	10000.000000	10000.000000	10000
unique	NaN	NaN	2932	NaN	3	2	NaN	NaN	
top	NaN	NaN	Smith	NaN	France	Male	NaN	NaN	
freq	NaN	NaN	32	NaN	5014	5457	NaN	NaN	
mean	5000.50000	1.569094e+07	NaN	650.528800	NaN	NaN	38.921800	5.012800	76485
std	2886.89568	7.193619e+04	NaN	96.653299	NaN	NaN	10.487806	2.892174	62397
min	1.00000	1.556570e+07	NaN	350.000000	NaN	NaN	18.000000	0.000000	0
25%	2500.75000	1.562853e+07	NaN	584.000000	NaN	NaN	32.000000	3.000000	0
50%	5000.50000	1.569074e+07	NaN	652.000000	NaN	NaN	37.000000	5.000000	97198
75%	7500.25000	1.575323e+07	NaN	718.000000	NaN	NaN	44.000000	7.000000	127644
max	10000.00000	1.581569e+07	NaN	850.000000	NaN	NaN	92.000000	10.000000	250898

In [8]:
```
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 [9]:
```
df.isnull().sum()
```

Out[9]:
```
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
```

In [10]:
```
df.columns
```

Out[10]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [11]:
```
df = df.drop (['RowNumber', 'CustomerId', 'Surname'], axis= 1)
```

In [12]:
```
df.head()
```

Out[12]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimated
0	619	France	Female	42	2	0.00	1	1	1	101
1	608	Spain	Female	41	1	83807.86	1	0	1	112
2	502	France	Female	42	8	159660.80	3	1	0	113
3	699	France	Female	39	1	0.00	2	0	0	93
4	850	Spain	Female	43	2	125510.82	1	1	1	79

In [13]:
```
df.columns
```

Out[13]:
```
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary',
       'Exited'],
      dtype='object')
```

In [14]:
```
df['Geography'].unique()
```

Out[14]:
```
array(['France', 'Spain', 'Germany'], dtype=object)
```

In [15]:
```
df= pd.get_dummies(df,drop_first=True)
df.head()
```

Out[15]:

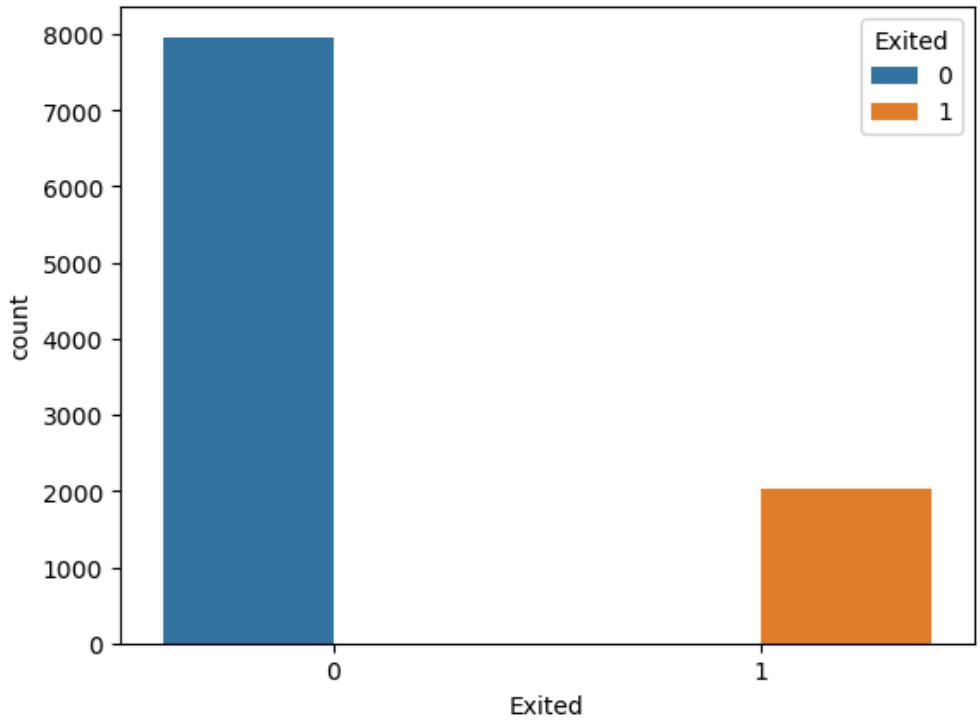
	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geogr
0	619	42	2	0.00	1	1	1	101348.88	1	
1	608	41	1	83807.86	1	0	1	112542.58	0	
2	502	42	8	159660.80	3	1	0	113931.57	1	
3	699	39	1	0.00	2	0	0	93826.63	0	
4	850	43	2	125510.82	1	1	1	79084.10	0	

In [16]:
```
df['Exited'].value_counts()
```

Out[16]:
```
Exited
0    7963
1    2037
Name: count, dtype: int64
```

In [24]:
```
import seaborn as sns
sns.countplot(x='Exited', hue='Exited',data=df)
```

Out[24]:
```
<Axes: xlabel='Exited', ylabel='count'>
```



In [29]:
```
X =df.drop('Exited',axis = 1)
Y = df['Exited']
```

In [30]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.20,random_state= 51,stratify=Y)
```

In [31]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_train
```

Out[31]:
```
array([[-0.78860057, -1.03742686,  0.6965047 , ..., -0.57619557,
         1.73726261, -1.09582175],
       [-0.68522026, -0.56111692, -0.68966051, ...,  1.73552185,
        -0.57561821,  0.91255717],
       [-1.03671334, -0.08480698,  1.3895873 , ..., -0.57619557,
        -0.57561821,  0.91255717],
       ...,
       [ 1.34103398, -0.46585493,  0.0034221 , ...,  1.73552185,
        -0.57561821,  0.91255717],
       [-0.20967079,  0.96307488, -1.38274311, ..., -0.57619557,
         1.73726261, -1.09582175],
       [ 1.12393531,  0.010455  ,  0.3499634 , ..., -0.57619557,
        -0.57561821, -1.09582175]])
```

In [32]:
```
from sklearn.linear_model import LogisticRegression
log = LogisticRegression()
log.fit(X_train,Y_train)
```

Out[32]:
```
LogisticRegression()
```

In [33]:
```
Y_pred1 = log.predict(X_test)
```

In [34]:
```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,Y_pred1)
```

Out[34]:
```
0.81
```

In [35]:
```
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier()
knn.fit(X_train, Y_train)
KNeighborsClassifier()
```

Out[35]:
```
KNeighborsClassifier()
```

In [36]:
```
Y_pred2 = knn.predict(X_test)
```

In [37]:
```
accuracy_score(Y_test, Y_pred2)
```

Out[37]:
```
0.827
```

In [38]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, Y_train)
```

Out[38]:
```
RandomForestClassifier()
```

In [39]:
```
Y_pred3 = rf.predict(X_test)
```

In [40]:
```
accuracy_score(Y_test, Y_pred3)
```

Out[40]:
```
0.8635
```

In [41]:
```
from sklearn.ensemble import GradientBoostingClassifier
gradient_booster = GradientBoostingClassifier()
gradient_booster.fit(X_train,Y_train)
```

Out[41]:
```
GradientBoostingClassifier()
```

In [42]:
```
Y_pred4 = gradient_booster.predict(X_test)
```

In [43]:
```
accuracy_score(Y_test, Y_pred4)
```

Out[43]:
```
0.8615
```

In [ ]:
```

```