```
!pip install kaggle
```

```
    Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.5
    Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (
    Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (fr
    Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-pack
    Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (f
    Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from
    Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packa
    Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (fr
    Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (fro
    Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-package
    Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-
    Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/
    Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-package
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
from logging import warning
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/kaggle_dataset/customer_churn/Chu
```

```
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bal
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

```
df.columns
```

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

```
print('df:', df.shape)
```

```
df: (10000, 14)
```

```
df.duplicated().sum()
```

```
0
```

```
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.info()
```

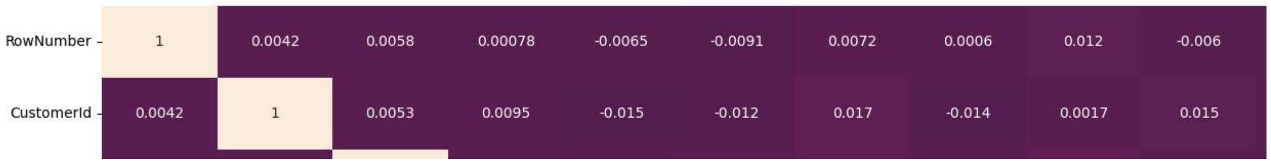
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   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
```

```
df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balan
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.0000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.8892
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.4052
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.0000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.0000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.5400
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.2400
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.0900

```
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(), annot = True)
```

```
<Axes: >
```



```
col = []
for i in df.columns:
  num = len(df[i].unique())
  print(i,':', str(num) + str(' Distinct values'))
  #append < 10 distinct values columns in list
  if num < 10:
    col.append(i)
```

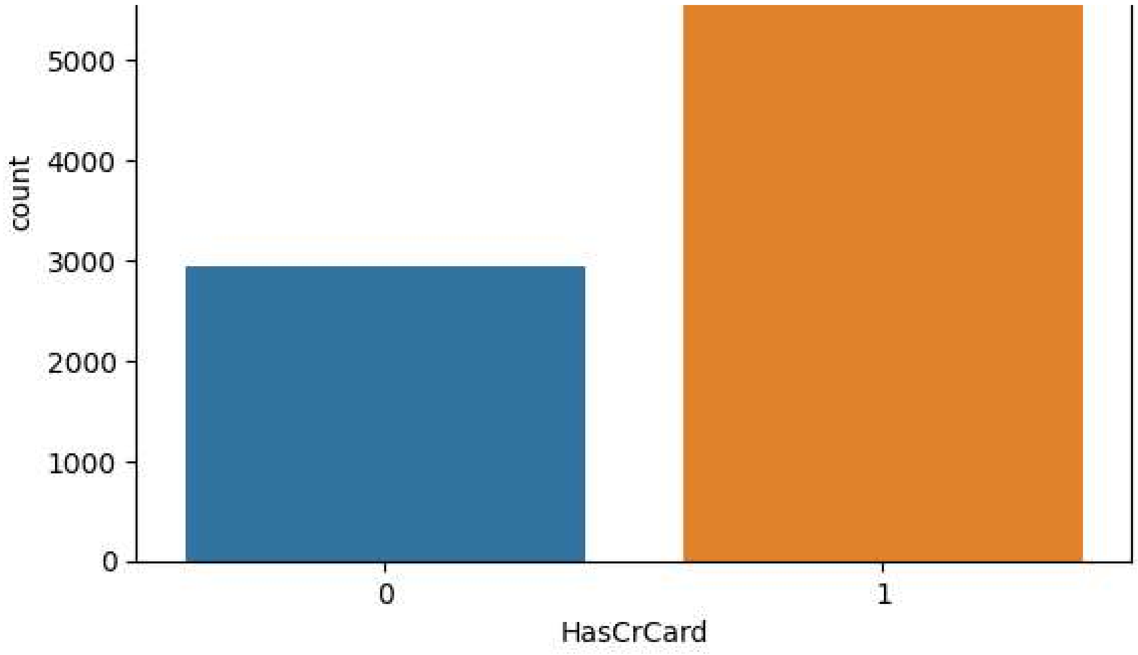
```
RowNumber : 10000 Distinct values
CustomerId : 10000 Distinct values
Surname : 2932 Distinct values
CreditScore : 460 Distinct values
Geography : 3 Distinct values
Gender : 2 Distinct values
Age : 70 Distinct values
Tenure : 11 Distinct values
Balance : 6382 Distinct values
NumOfProducts : 4 Distinct values
HasCrCard : 2 Distinct values
IsActiveMember : 2 Distinct values
EstimatedSalary : 9999 Distinct values
Exited : 2 Distinct values
```

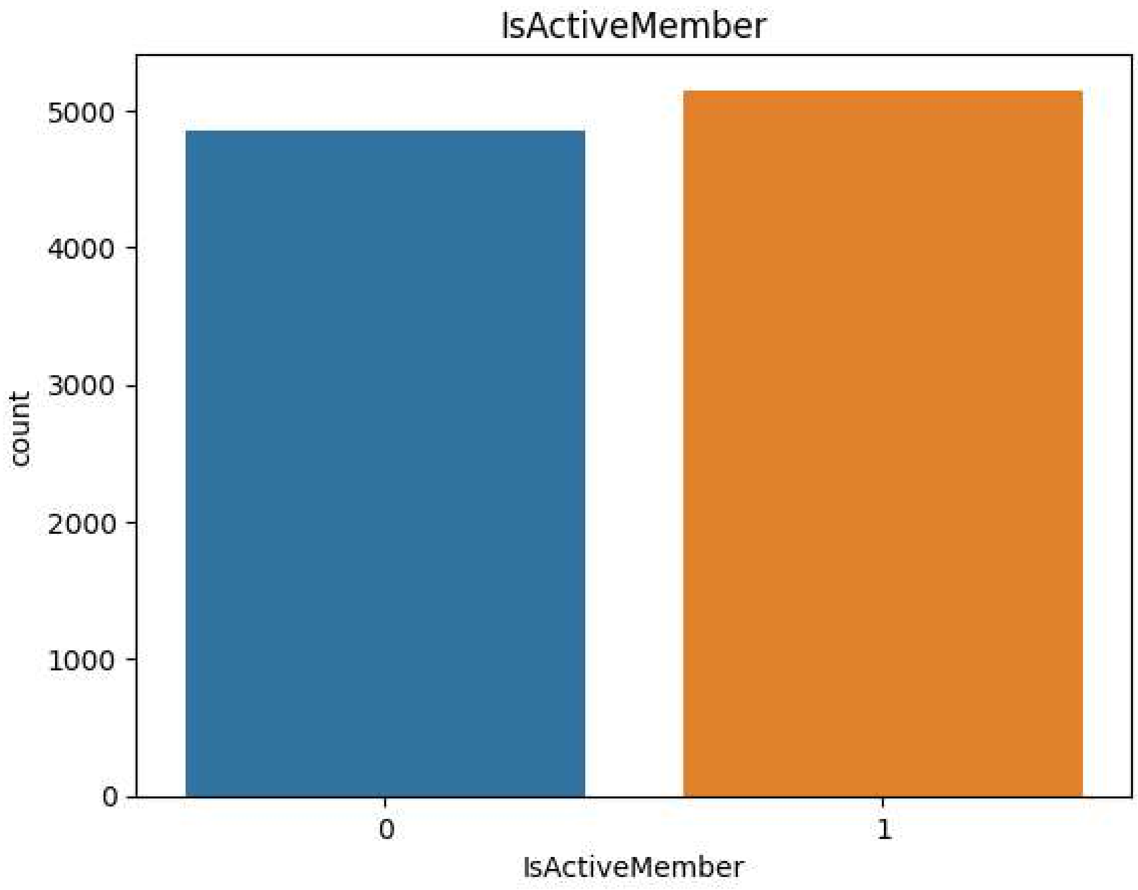
```
for i in col:
  print(df[i].value_counts(), '\n')
```

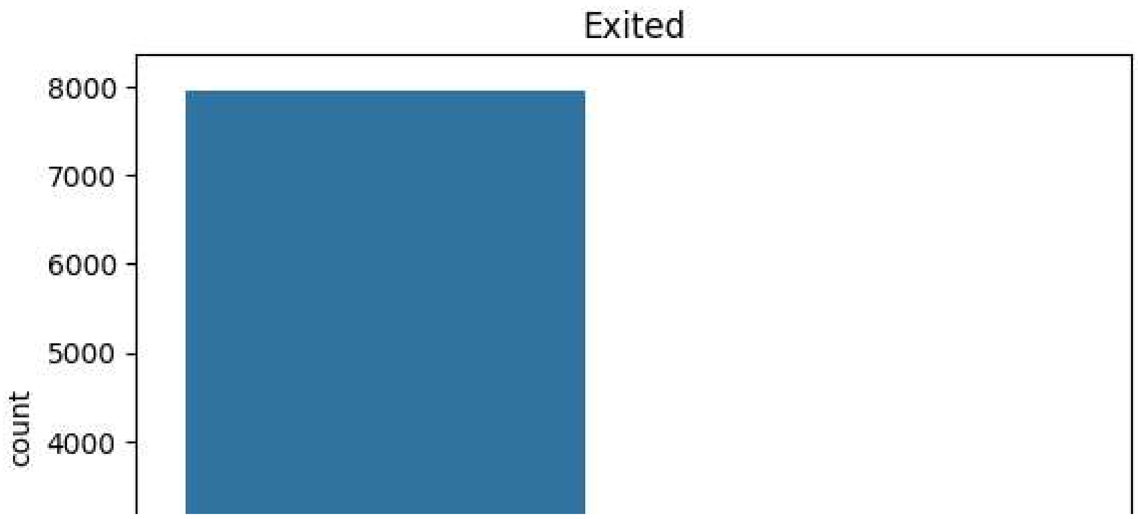
```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64

Male      5457
Female    4543
Name: Gender, dtype: int64

1    5084
2    4590
3     266
4      60
Name: NumOfProducts, dtype: int64

1    7055
0    2945
Name: HasCrCard, dtype: int64

1    5151
0    4849
Name: IsActiveMember, dtype: int64

0    7963
1    2037
```

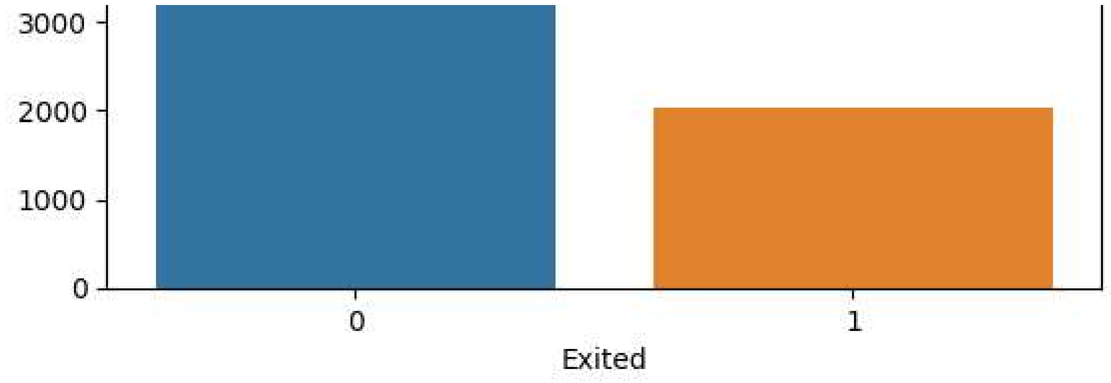
```
    Name: Exited, dtype: int64
```

```
for i in col:
  sns.countplot(x=i, data=df)
  plt.title(i)
  plt.show()
```

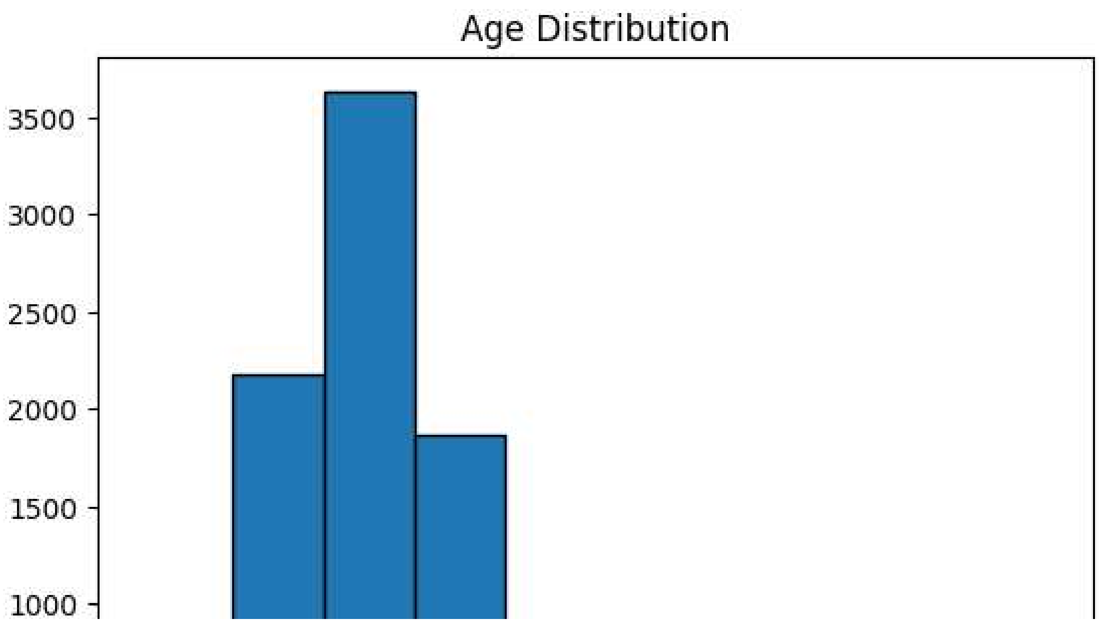

count
0
1000
2000
3000
4000
5000
0
1
HasCrCard


IsActiveMember
count
0
1000
2000
3000
4000
5000
0
1
IsActiveMember

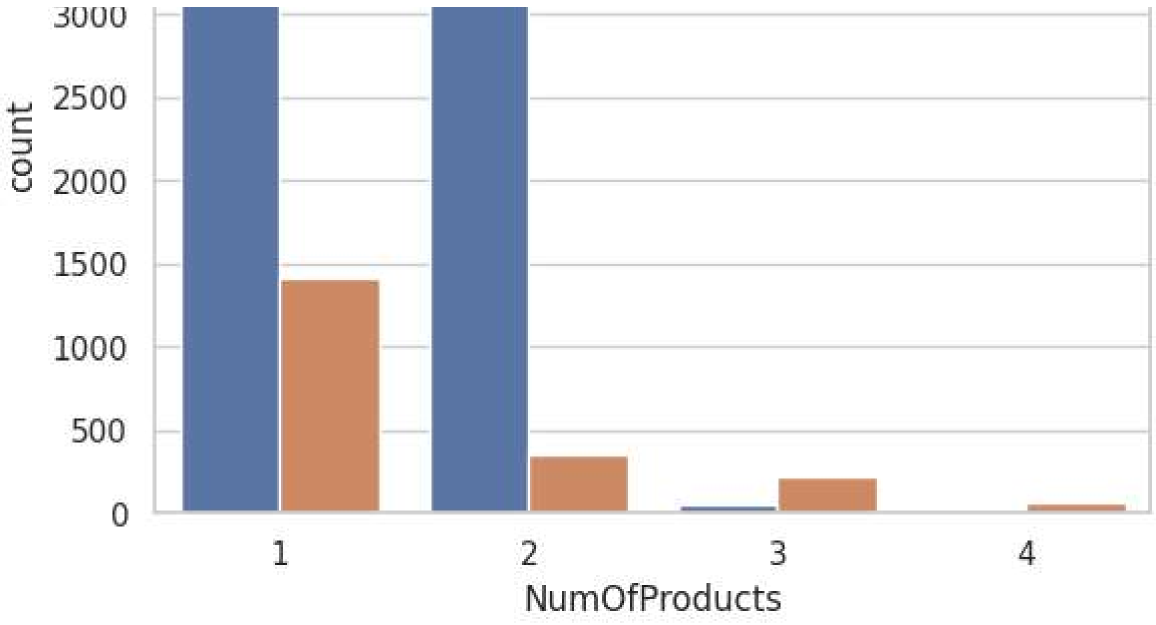

Exited
count
4000
5000
6000
7000
8000

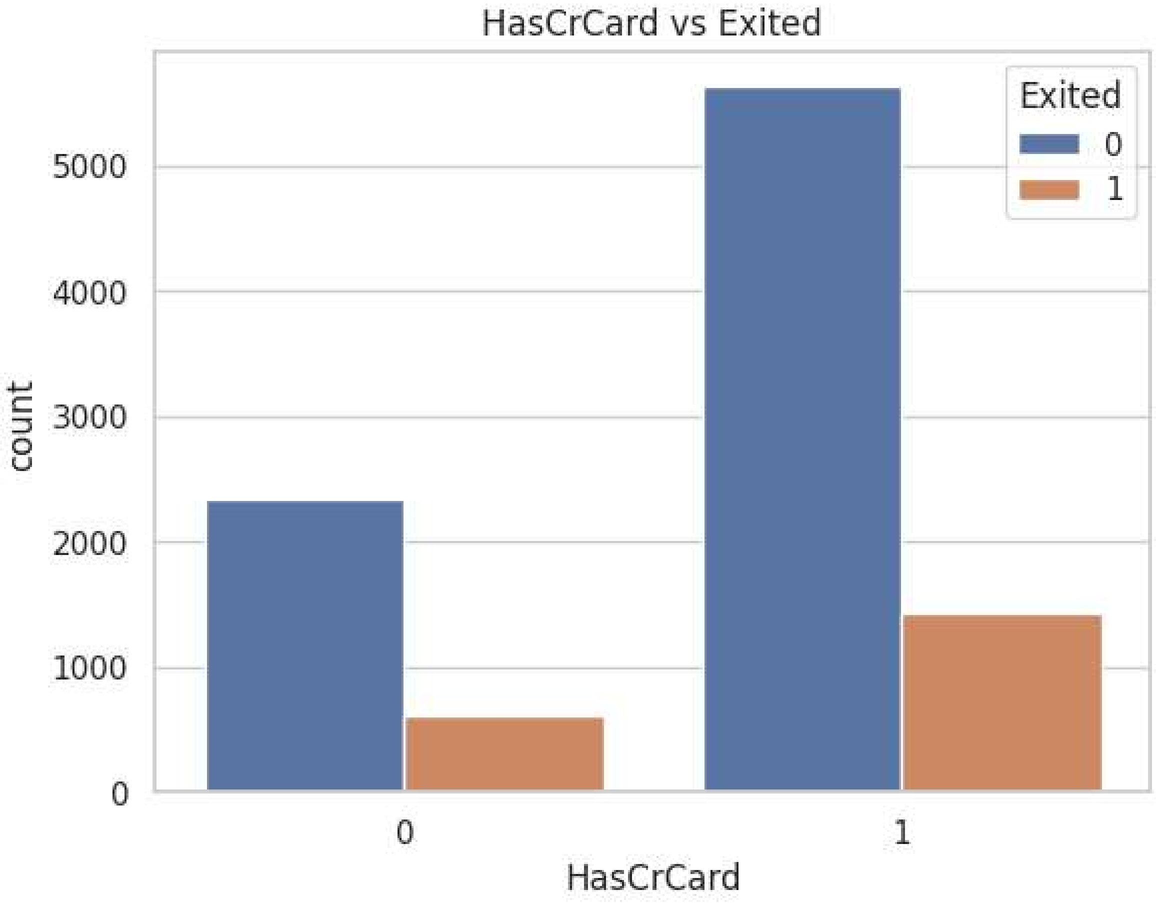


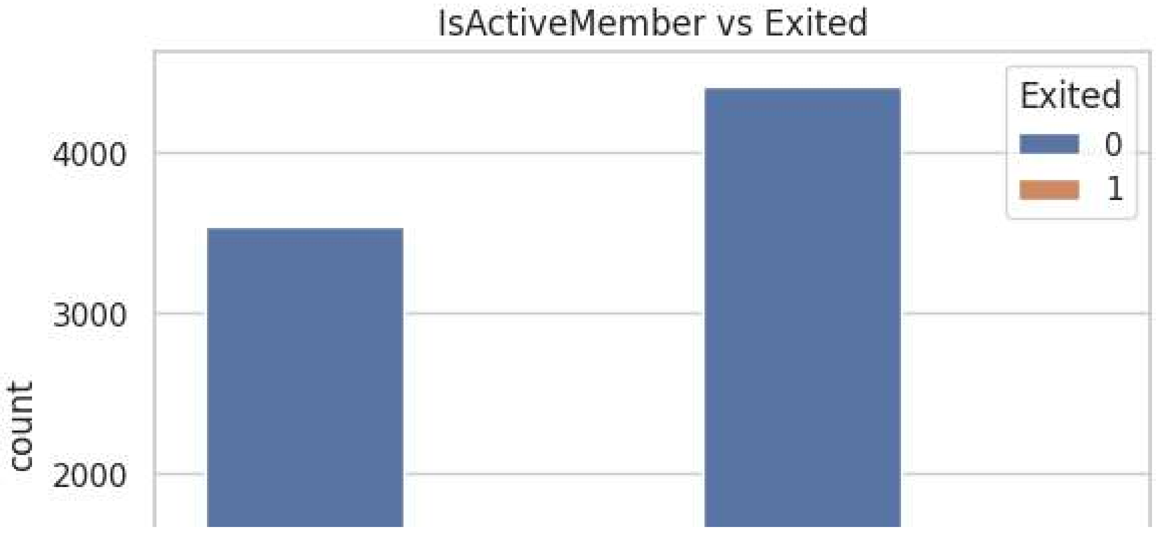
```
plt.hist(df['Age'], edgecolor='black')
plt.title('Age Distribution')
plt.show()
```

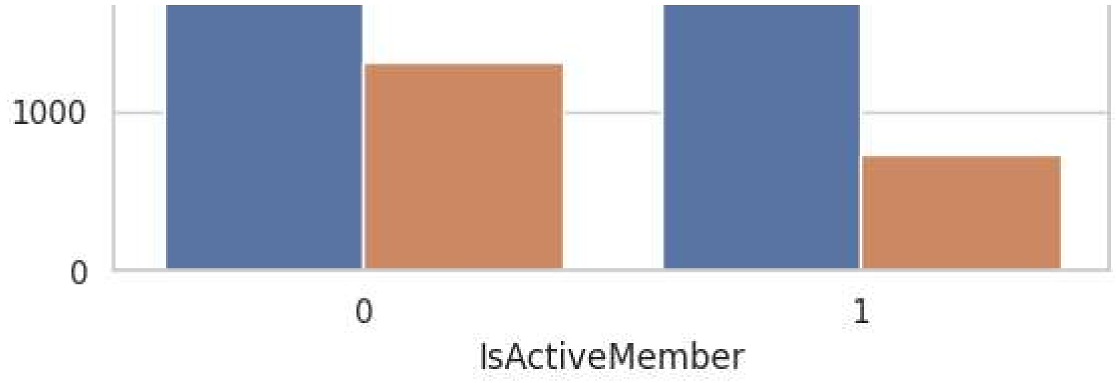


```
sns.set(style="whitegrid")
j = col.pop()
for i in col:
  sns.countplot(x=i, hue=j, data=df)
  plt.title( i + ' vs ' + j)
  plt.show()
```


count
3000
2500
2000
1500
1000
500
0
1
2
3
4
NumOfProducts


HasCrCard vs Exited
Exited
0
1
count
5000
4000
3000
2000
1000
0
0
1
HasCrCard


IsActiveMember vs Exited
Exited
0
1
count
4000
3000
2000



```
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bal
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

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

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

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Gender'] = le.fit_transform(df['Gender'])
```

```
df = pd.get_dummies(data = df, columns=['Geography'], drop_first=True)
```

```
df.head()
```

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

```
a = df.iloc[:,-3:-2]
```

```
a.head()
```

	Exited
0	1
1	0
2	1
3	0
4	0

```
df.drop(columns=['Exited'], inplace = True)
```

```
df.head()
```

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

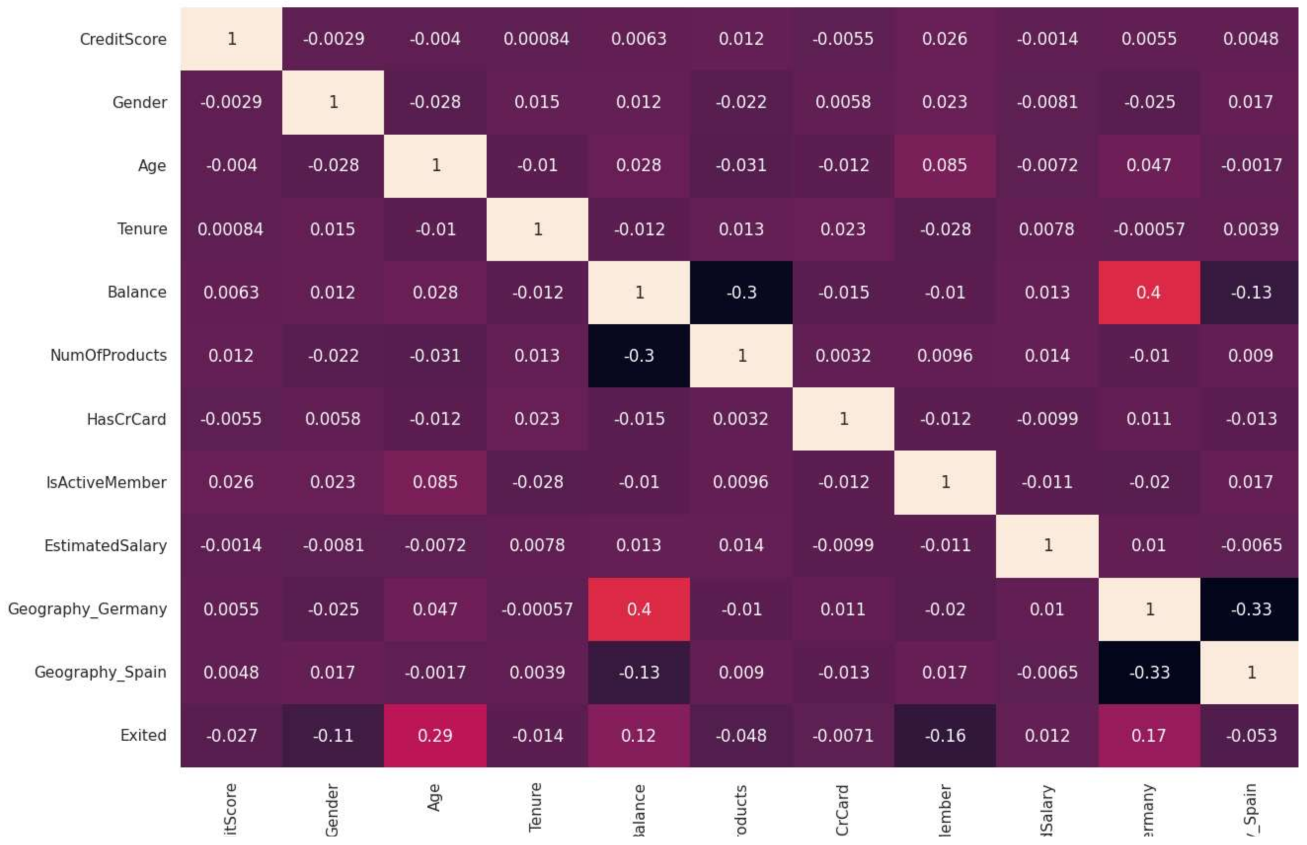
```
df = pd.concat([df, a], axis=1)
```

```
df.head()
```

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

```
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(), annot = True)
```

```
<Axes: >
```



```
x = df.iloc[:, :-1].values
y = df.iloc[:,-1].values
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=
```

```
print(x_train.shape, x_test.shape)
print(y_train.shape, y_test.shape)

     (8000, 11) (2000, 11)
     (8000,) (2000,)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

```
from sklearn.linear_model import LogisticRegression
log = LogisticRegression(random_state = 42)
log.fit(x_train, y_train)
```

```
▾         LogisticRegression
LogisticRegression(random_state=42)
```

```
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
R = RandomForestClassifier(n_estimators=5 ,random_state = 42)
```

```
R.fit(x_train, y_train)
```

```
▾                RandomForestClassifier
RandomForestClassifier(n_estimators=5, random_state=42)
```

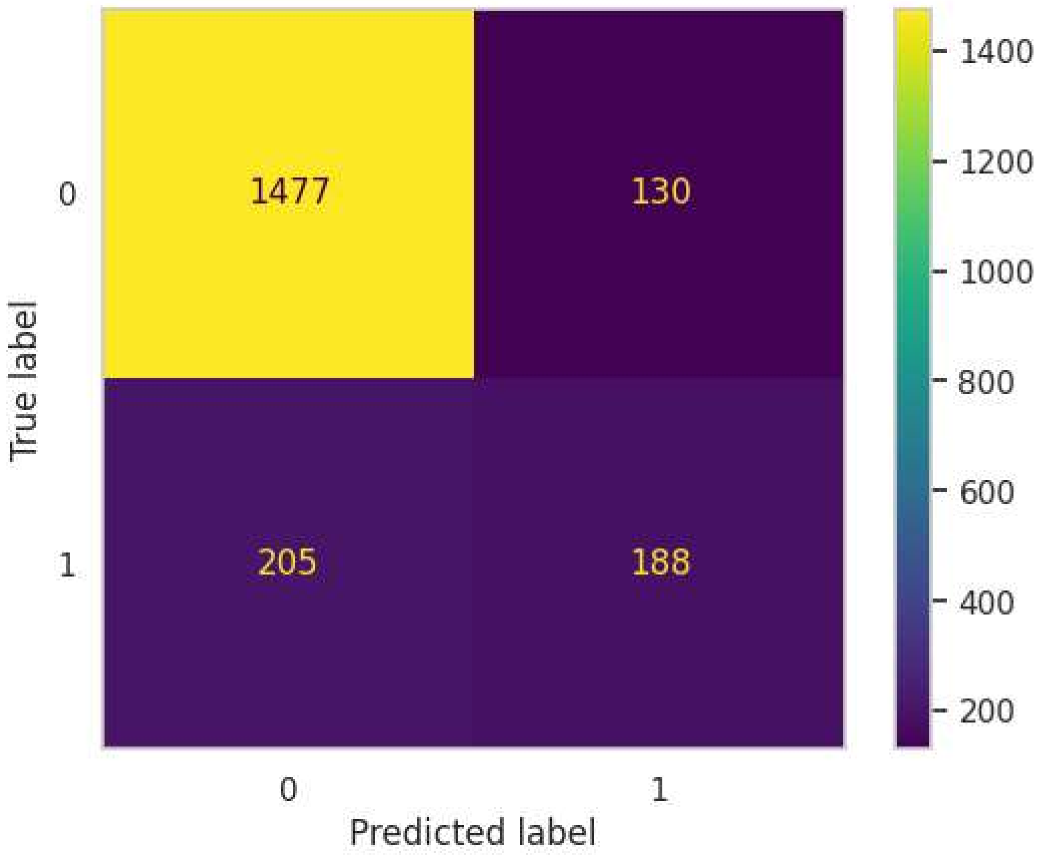
```
clf = GradientBoostingClassifier(n_estimators=10, learning_rate=1.0, random_state=42)
clf.fit(x_train, y_train)
```

```
▾                          GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=1.0, n_estimators=10, random_state=42)
```

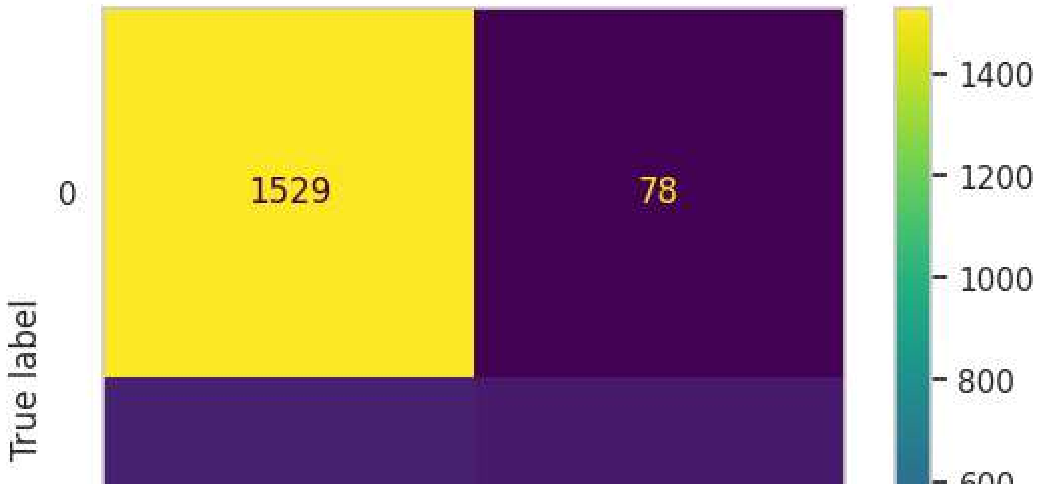
```
classifier = [log, R, clf]
model = ['Logistic Regression', 'Random Forest Classifier', 'Gradient Boosting Classifier'
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, Confu
for i in range(len(classifier)):
  y_pred = classifier[i].predict(x_test)
  cm = confusion_matrix(y_test, y_pred)
  accuracy = accuracy_score(y_test, y_pred)*100
  print('\nfor ' + str(model[i]) + ':\n')
  disp = ConfusionMatrixDisplay(confusion_matrix=cm)
  plt.rcParams['axes.grid'] = False
  disp.plot()
  print(accuracy)
  print(classification_report(y_test, y_pred))
  plt.show()
```

```
           0       0.88      0.92      0.90      1607
           1       0.59      0.48      0.53       393

    accuracy                           0.83      2000
   macro avg       0.73      0.70      0.71      2000
weighted avg       0.82      0.83      0.83      2000
```



```
for Gradient Boosting Classifier:

85.65
              precision    recall  f1-score   support

           0       0.88      0.95      0.91      1607
           1       0.70      0.47      0.56       393

    accuracy                           0.86      2000
   macro avg       0.79      0.71      0.74      2000
weighted avg       0.84      0.86      0.84      2000
```


True label