Importing necessary python libraries

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

```
df = pd.read_csv('/content/Bank-Records.csv')
```

```
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	8380
2	3	15619304	Onio	502	France	Female	42	8	15966
3	4	15701354	Boni	699	France	Female	39	1	

Next steps: Generate code with df View recommended plots

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 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
 14  Complain            10000 non-null  int64
 15  Satisfaction Score  10000 non-null  int64
 16  Card Type           10000 non-null  object
 17  Point Earned        10000 non-null  int64
dtypes: float64(2), int64(12), object(4)
memory usage: 1.4+ MB
```

```
obj = ['Geography', 'Gender', 'Card Type']
for i in obj:
  print(f'The {i} column has {df[i].unique()} unique values.')
```

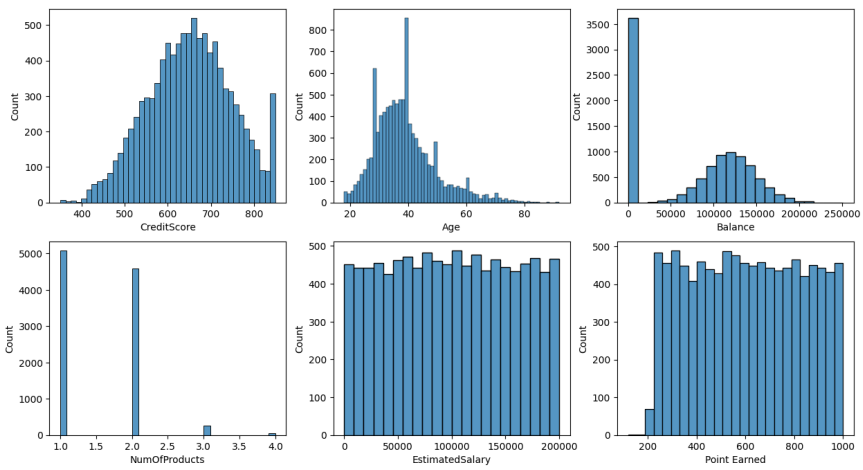
```
The Geography column has ['France' 'Spain' 'Germany'] unique values.
The Gender column has ['Female' 'Male'] unique values.
The Card Type column has ['DIAMOND' 'GOLD' 'SILVER' 'PLATINUM'] unique values.
```

```
num = ['CreditScore', 'Age', 'Balance', 'NumOfProducts', 'EstimatedSalary', 'Point Earned']

for i in num:
  print(f'The mean value of {i} is {df[i].mean()}.')
  print(f'The median value of {i} is {df[i].median()}.')
  print(f'The mode value of {i} is {df[i].mode()}.')
  print('-'*50)
```

```
The mean value of CreditScore is 650.5288.
The median value of CreditScore is 652.0.
The mode value of CreditScore is 0    850
Name: CreditScore, dtype: int64.
--------------------------------------------------
The mean value of Age is 38.9218.
The median value of Age is 37.0.
The mode value of Age is 0    37
Name: Age, dtype: int64.
--------------------------------------------------
The mean value of Balance is 76485.889288.
```

```
The median value of Balance is 97198.54000000001.
The mode value of Balance is 0    0.0
Name: Balance, dtype: float64.
--------------------------------------------------
The mean value of NumOfProducts is 1.5302.
The median value of NumOfProducts is 1.0.
The mode value of NumOfProducts is 0    1
Name: NumOfProducts, dtype: int64.
--------------------------------------------------
The mean value of EstimatedSalary is 100090.239881.
The median value of EstimatedSalary is 100193.915.
The mode value of EstimatedSalary is 0    24924.92
Name: EstimatedSalary, dtype: float64.
--------------------------------------------------
The mean value of Point Earned is 606.5151.
The median value of Point Earned is 605.0.
The mode value of Point Earned is 0    408
Name: Point Earned, dtype: int64.
--------------------------------------------------
```

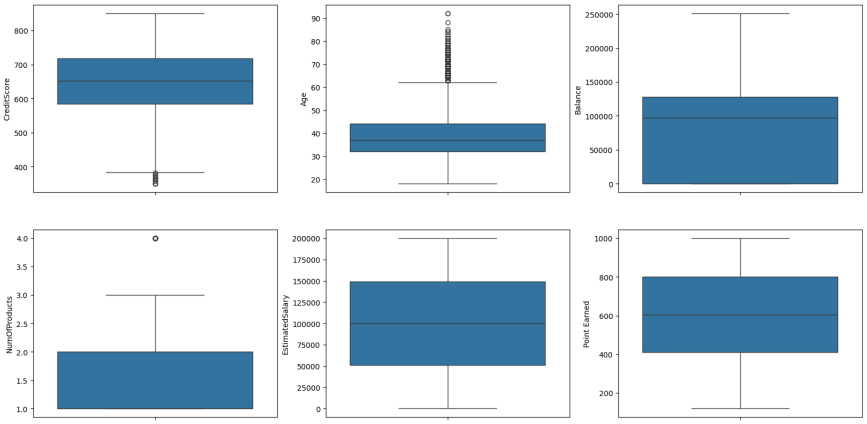
```
plt.figure(figsize=(15, 8))

plt.subplot(2, 3, 1)
sns.histplot(df['CreditScore'])

plt.subplot(2, 3, 2)
sns.histplot(df['Age'])

plt.subplot(2, 3, 3)
sns.histplot(df['Balance'])

plt.subplot(2, 3, 4)
sns.histplot(df['NumOfProducts'])

plt.subplot(2, 3, 5)
sns.histplot(df['EstimatedSalary'])

plt.subplot(2, 3, 6)
sns.histplot(df['Point Earned'])

plt.show()
```



```
# Outlier calculation
for i in num:
  q1 = np.quantile(df[i], 0.25)
  q3 = np.quantile(df[i], 0.75)
  iqr = q3 - q1
  lcv = q1 - 1.5 * iqr
  ucv = q3 + 1.5 * iqr
  print(f'The first quartile value for {i} is {q1} and second quartile value for {i} is {q3}.')
  print(f'The IQR for {i} is {iqr}.')
  print(f'The LCV and UCV are {lcv} and {ucv} respectively.')
  print('-'*50)
```

```
The first quartile value for CreditScore is 584.0 and second quartile value for CreditScore is 718.0.
The IQR for CreditScore is 134.0.
The LCV and UCV are 383.0 and 919.0 respectively.
--------------------------------------------------
The first quartile value for Age is 32.0 and second quartile value for Age is 44.0.
The IQR for Age is 12.0.
The LCV and UCV are 14.0 and 62.0 respectively.
--------------------------------------------------
The first quartile value for Balance is 0.0 and second quartile value for Balance is 127644.24.
The IQR for Balance is 127644.24.
The LCV and UCV are -191466.36000000002 and 319110.60000000003 respectively.
--------------------------------------------------
The first quartile value for NumOfProducts is 1.0 and second quartile value for NumOfProducts is 2.0.
The IQR for NumOfProducts is 1.0.
The LCV and UCV are -0.5 and 3.5 respectively.
--------------------------------------------------
The first quartile value for EstimatedSalary is 51002.11 and second quartile value for EstimatedSalary is 149388.2475.
The IQR for EstimatedSalary is 98386.1375.
The LCV and UCV are -96577.09624999999 and 296967.45375 respectively.
--------------------------------------------------
The first quartile value for Point Earned is 410.0 and second quartile value for Point Earned is 801.0.
The IQR for Point Earned is 391.0.
The LCV and UCV are -176.5 and 1387.5 respectively.
--------------------------------------------------
```

```
# Box plot
plt.figure(figsize=(20, 10))

plt.subplot(2, 3, 1)
sns.boxplot(df['CreditScore'])

plt.subplot(2, 3, 2)
sns.boxplot(df['Age'])

plt.subplot(2, 3, 3)
sns.boxplot(df['Balance'])

plt.subplot(2, 3, 4)
sns.boxplot(df['NumOfProducts'])

plt.subplot(2, 3, 5)
sns.boxplot(df['EstimatedSalary'])

plt.subplot(2, 3, 6)
sns.boxplot(df['Point Earned'])

plt.show()
```




```
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	8380
2	3	15619304	Onio	502	France	Female	42	8	15966
3	4	15701354	Boni	699	France	Female	39	1	

Next steps: Generate code with df View recommended plots

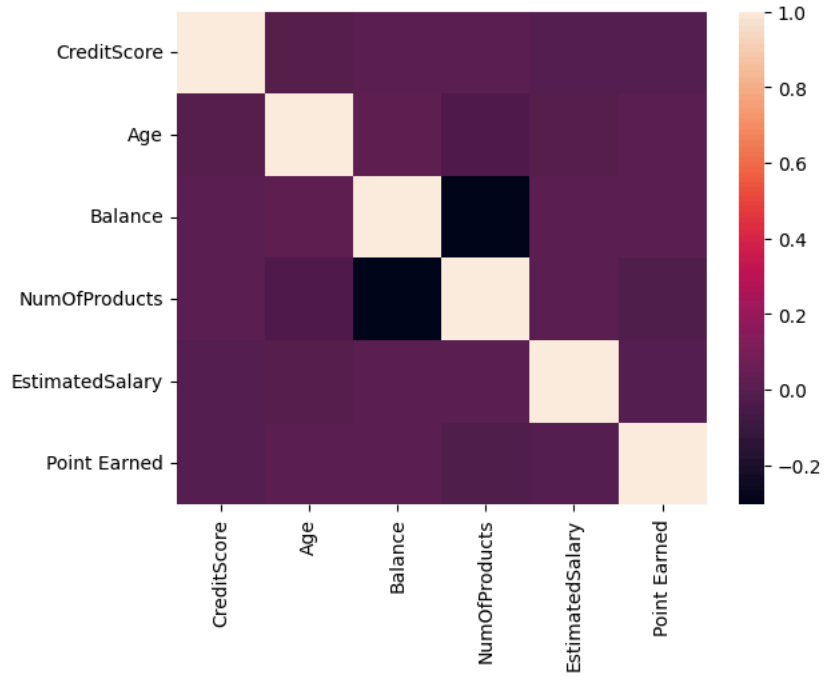
```
df_num = df[['CreditScore', 'Age', 'Balance', 'NumOfProducts', 'EstimatedSalary', 'Point Earned']]

df_cor = df_num.corr()
df_cor
```

	CreditScore	Age	Balance	NumOfProducts	EstimatedSalary	E
CreditScore	1.000000	-0.003965	0.006268	0.012238	-0.001384	0.0
Age	-0.003965	1.000000	0.028308	-0.030680	-0.007201	0.0
Balance	0.006268	0.028308	1.000000	-0.304180	0.012797	0.0
NumOfProducts	0.012238	-0.030680	-0.304180	1.000000	0.014204	-0.0
EstimatedSalary	-0.001384	-0.007201	0.012797	0.014204	1.000000	-0.0

Next steps: Generate code with df_cor View recommended plots

```
sns.heatmap(df_cor)
```

```
<Axes: >
```

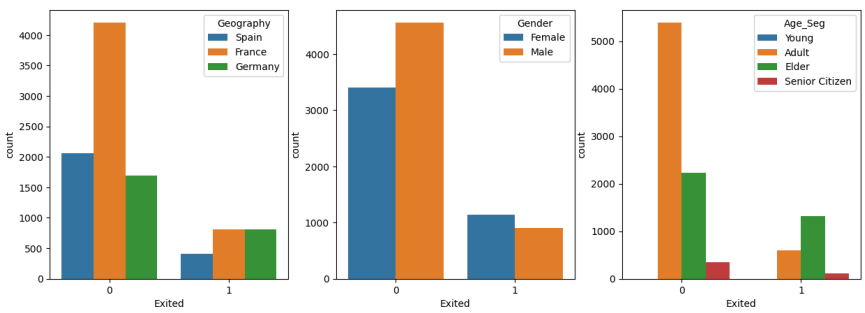



```
for i in num:
  print(f'Below is the correlation between {i} and Exited.')
  print(df[[i, 'Exited']].corr())
  print('-'*50)
```

```
Below is the correlation between CreditScore and Exited.
             CreditScore    Exited
CreditScore     1.000000 -0.026771
Exited         -0.026771  1.000000
--------------------------------------------------
Below is the correlation between Age and Exited.
             Age    Exited
Age     1.000000  0.285296
Exited  0.285296  1.000000
--------------------------------------------------
Below is the correlation between Balance and Exited.
          Balance    Exited
Balance  1.000000  0.118577
Exited   0.118577  1.000000
--------------------------------------------------
Below is the correlation between NumOfProducts and Exited.
               NumOfProducts    Exited
NumOfProducts       1.000000 -0.047611
Exited             -0.047611  1.000000
--------------------------------------------------
Below is the correlation between EstimatedSalary and Exited.
                 EstimatedSalary   Exited
EstimatedSalary          1.00000  0.01249
Exited                   0.01249  1.00000
--------------------------------------------------
Below is the correlation between Point Earned and Exited.
              Point Earned    Exited
Point Earned      1.000000 -0.004628
Exited           -0.004628  1.000000
--------------------------------------------------
```

```
bins= [0,18,40,61,110]
labels = ['Young','Adult','Elder','Senior Citizen']
df['Age_Seg'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)
```

```
demo = ['Geography',    'Gender',   'Age_Seg']
plt.figure(figsize=(15, 5))

plt.subplot(1,3,1)
sns.countplot(data = df, x = 'Exited', hue = 'Geography')

plt.subplot(1,3,2)
sns.countplot(data = df, x = 'Exited', hue = 'Gender')

plt.subplot(1,3,3)
sns.countplot(data = df, x = 'Exited', hue = 'Age_Seg')

plt.show()
```




```
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	8380
2	3	15619304	Onio	502	France	Female	42	8	15966
3	4	15701354	Boni	699	France	Female	39	1	

Next steps: Generate code with df View recommended plots

```
df_geo = df.groupby('Geography')
df_geo_churn = df_geo['Exited'].value_counts().reset_index()
df_geo_churn
```

	Geography	Exited	count
0	France	0	4203
1	France	1	811
2	Germany	0	1695
3	Germany	1	814
4	Spain	0	2064
5	Spain	1	413

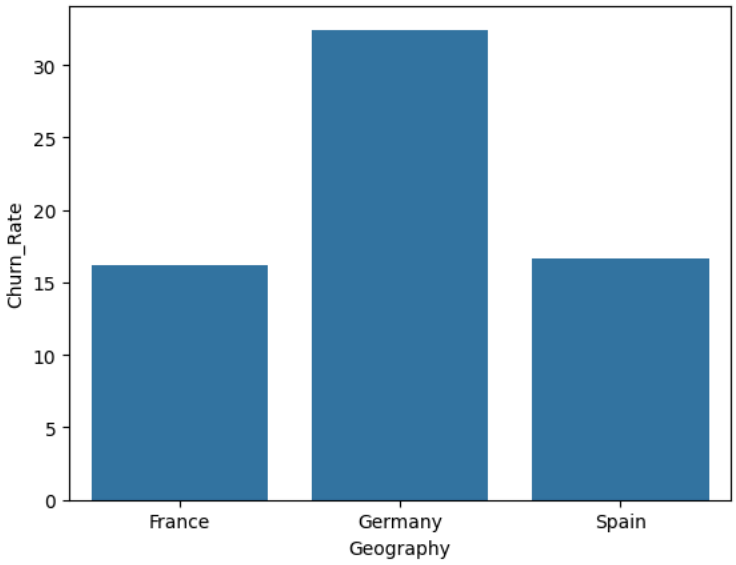
Next steps: Generate code with df_geo_churn View recommended plots

```
df_geo_count = df_geo['Geography'].value_counts().reset_index()
df_geo_new = pd.merge(df_geo_churn, df_geo_count, on = 'Geography')
df_geo_new['Churn_Rate'] = (df_geo_new['count_x']/df_geo_new['count_y'])*100
df_geo_new[df_geo_new['Exited'] == 1]
```

	Geography	Exited	count_x	count_y	Churn_Rate
1	France	1	811	5014	16.174711
3	Germany	1	814	2509	32.443204
5	Spain	1	413	2477	16.673395

```
sns.barplot(y = df_geo_new[df_geo_new['Exited'] == 1]['Churn_Rate'], x = df_geo_new['Geography'])
```

```
<Axes: xlabel='Geography', ylabel='Churn_Rate'>
```




```
df_gen = df.groupby('Gender')
df_gen_churn = df_gen['Exited'].value_counts().reset_index()
df_gen_churn
```

	Gender	Exited	count
0	Female	0	3404
1	Female	1	1139
2	Male	0	4558
3	Male	1	899

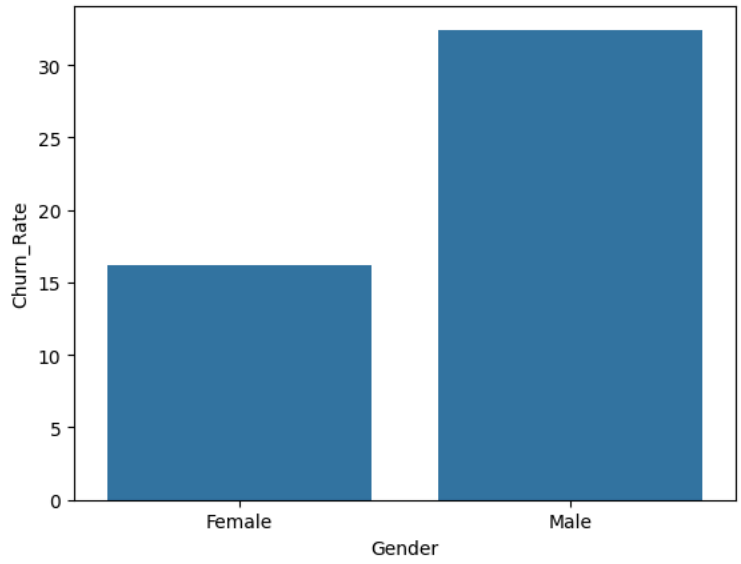
Next steps: Generate code with df_gen_churn View recommended plots

```
df_gen_count = df_gen['Gender'].value_counts().reset_index()
df_gen_new = pd.merge(df_gen_churn, df_gen_count, on = 'Gender')
df_gen_new['Churn_Rate'] = (df_geo_new['count_x']/df_geo_new['count_y'])*100
df_gen_new[df_gen_new['Exited'] == 1]
```

	Gender	Exited	count_x	count_y	Churn_Rate
1	Female	1	1139	4543	16.174711
3	Male	1	899	5457	32.443204

```
sns.barplot(y = df_gen_new[df_gen_new['Exited'] == 1]['Churn_Rate'], x = df_gen_new['Gender'])
```

```
<Axes: xlabel='Gender', ylabel='Churn_Rate'>
```




```
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	8380
2	3	15619304	Onio	502	France	Female	42	8	15966
3	4	15701354	Boni	699	France	Female	39	1	

Next steps: Generate code with df View recommended plots

```
df['Customer_Exited'] = df['Exited'].apply(lambda x: 'Left' if x == 1 else 'Remain' )
```

```
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	8380
2	3	15619304	Onio	502	France	Female	42	8	15966
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	12551

Next steps: Generate code with df View recommended plots

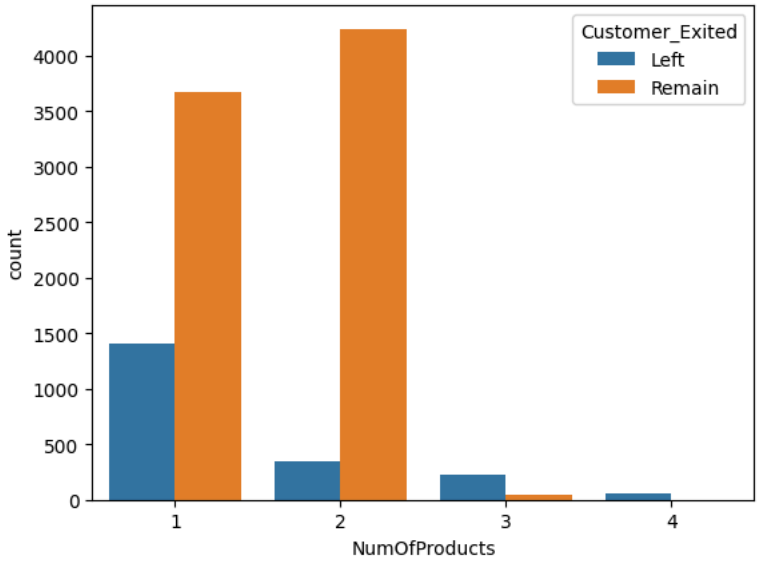
```
df[['Customer_Exited', 'NumOfProducts']]
```

	Customer_Exited	NumOfProducts
0	Left	1
1	Remain	1
2	Left	3
3	Remain	2
4	Remain	1
...	...	...
9995	Remain	2
9996	Remain	1
9997	Left	1
9998	Left	2
9999	Remain	1

10000 rows × 2 columns

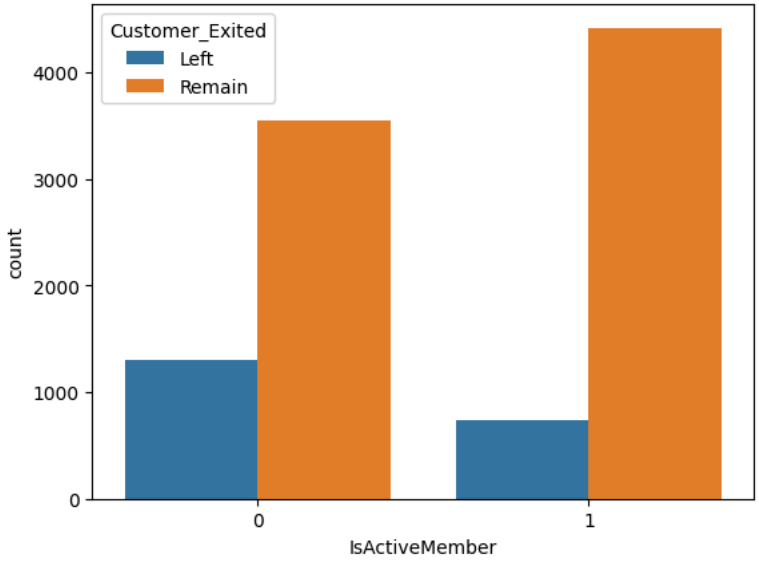
```
sns.countplot(data = df[['Customer_Exited', 'NumOfProducts']], x = 'NumOfProducts', hue = 'Customer_Exited')
```

```
<Axes: xlabel='NumOfProducts', ylabel='count'>
```




```
sns.countplot(data = df[['Customer_Exited', 'IsActiveMember']], x = 'IsActiveMember', hue = 'Customer_Exited')
```

```
<Axes: xlabel='IsActiveMember', ylabel='count'>
```




```
df[['Balance', 'Customer_Exited']]
```

	Balance	Customer_Exited
0	0.00	Left
1	83807.86	Remain
2	159660.80	Left
3	0.00	Remain
4	125510.82	Remain
...	...	...
9995	0.00	Remain
9996	57369.61	Remain
9997	0.00	Left
9998	75075.31	Left
9999	130142.79	Remain

10000 rows × 2 columns

```
df['Balance'].describe()
```

```
count     10000.000000
mean      76485.889288
std       62397.405202
min           0.000000
25%           0.000000
50%       97198.540000
75%      127644.240000
max      250898.090000
Name: Balance, dtype: float64
```

```
df['Fin_Level'] = df['Balance'].apply(lambda x: 'Very very high' if x > 200000 else ('Very High' if x > 100000 and x <= 200000 else ('Hig
```

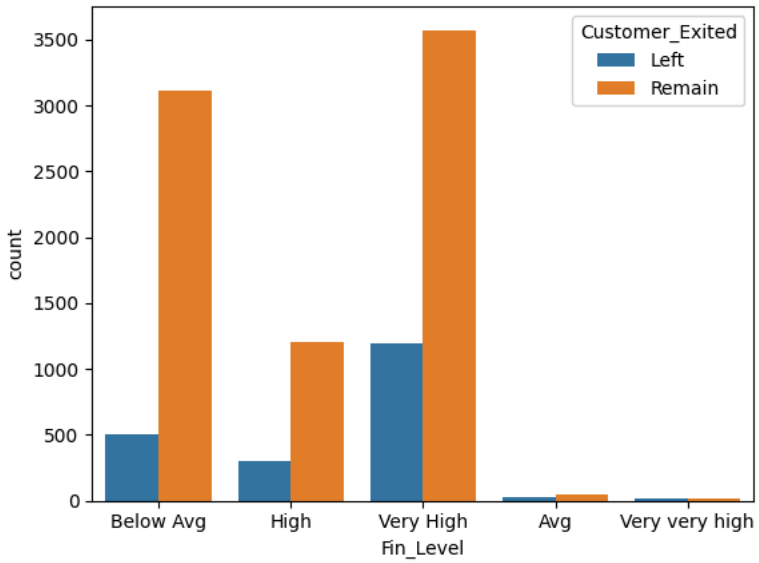
```
df.head()
```

	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	8380
2	3	15619304	Onio	502	France	Female	42	8	15966
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	12551

5 rows × 21 columns

```
sns.countplot(data = df[['Customer_Exited', 'Fin_Level']], x = 'Fin_Level', hue = 'Customer_Exited')
```

```
<Axes: xlabel='Fin_Level', ylabel='count'>
```

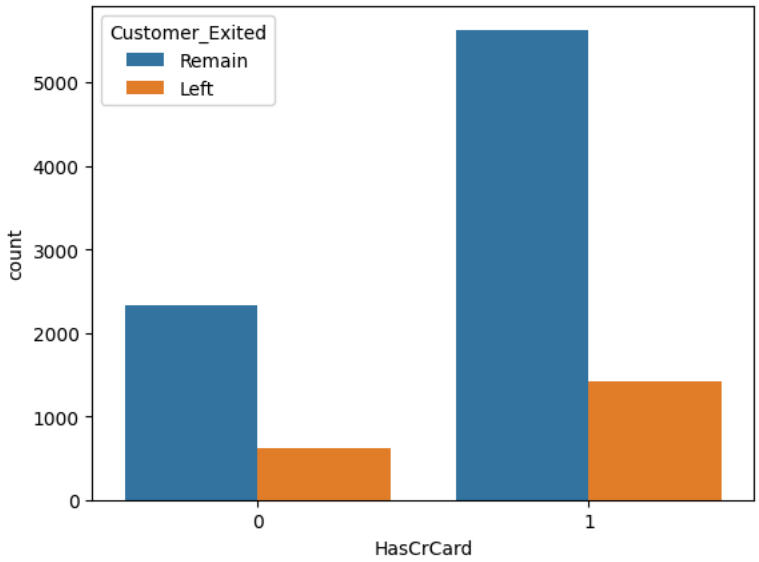


```
df[['HasCrCard', 'Customer_Exited', 'Card Type']].head()
```

	HasCrCard	Customer_Exited	Card Type
0	1	Left	DIAMOND
1	0	Remain	DIAMOND
2	1	Left	DIAMOND
3	0	Remain	GOLD
4	1	Remain	GOLD

```
sns.countplot(data = df[['Customer_Exited', 'HasCrCard']], x = 'HasCrCard', hue = 'Customer_Exited')
```

```
<Axes: xlabel='HasCrCard', ylabel='count'>
```




```
data = pd.crosstab(df['Customer_Exited'], df['HasCrCard'])


# H0: Customer churn and Has credit card are independent.
# H1: Both are dependent.
stat, p, dof, expected = stats.chi2_contingency(data)


# interpret p-value
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (H0 holds true)')
```

```
p value is 0.5026181509009862
Independent (H0 holds true)
```

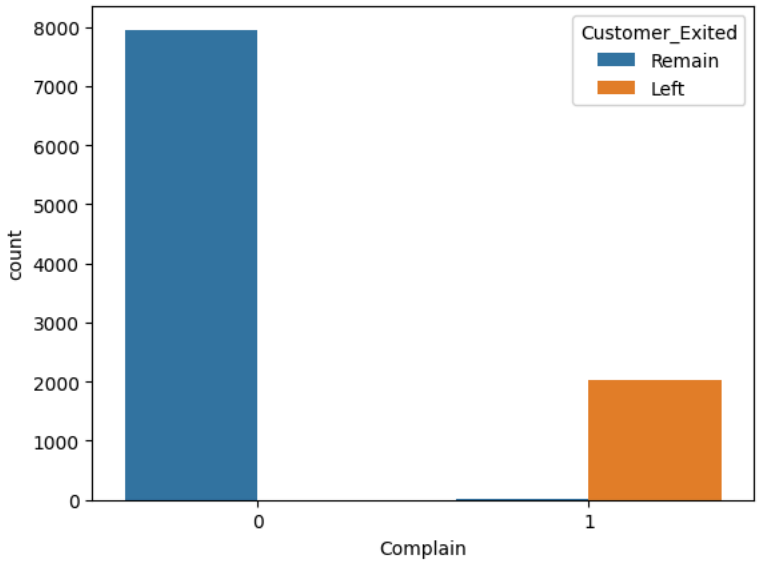
```
df.head()
```

	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	8380
2	3	15619304	Onio	502	France	Female	42	8	15966
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	12551

5 rows × 21 columns

```
sns.countplot(data = df[['Customer_Exited', 'Complain']], x = 'Complain', hue = 'Customer_Exited')
```

```
<Axes: xlabel='Complain', ylabel='count'>
```

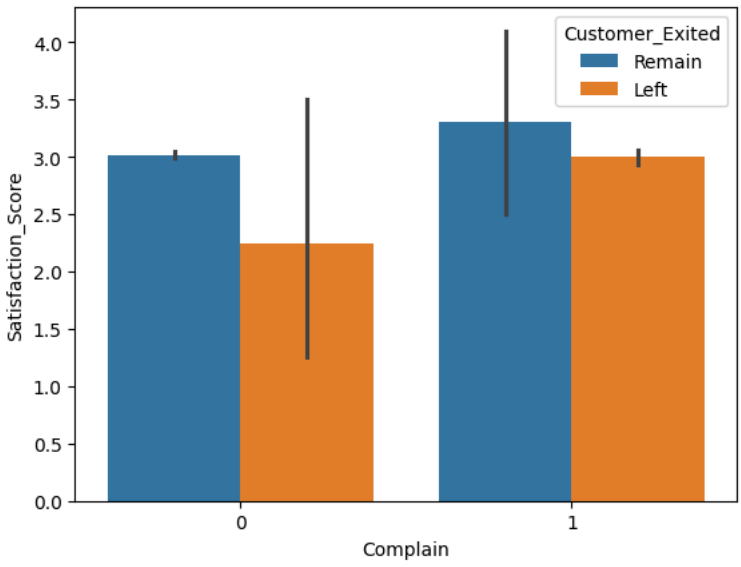



```
# H0: Customer churn and complain raised are independent.
# H1: Both are dependent.
data = pd.crosstab(df['Customer_Exited'], df['Complain'])
stat, p, dof, expected = stats.chi2_contingency(data)

# interpret p-value
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (H0 holds true)')
```

```
p value is 0.0
Dependent (reject H0)
```

```
sns.barplot(data = df[['Customer_Exited', 'Complain', 'Satisfaction_Score']], x = 'Complain', y = 'Satisfaction_Score', hue = 'Customer_
```

```
<Axes: xlabel='Complain', ylabel='Satisfaction_Score'>
```



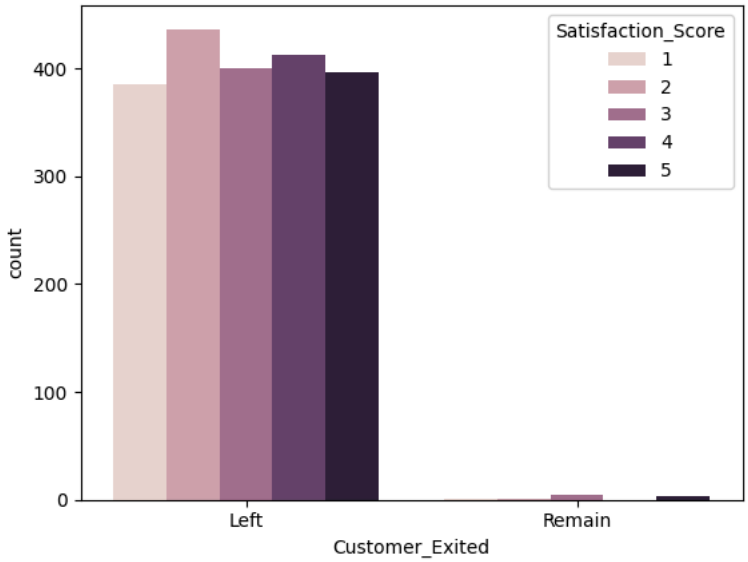

```
import statsmodels.api as sm
from statsmodels.formula.api import ols

model = ols('Exited ~ C(Complain) * C(Satisfaction_Score) + C(Complain):C(Satisfaction_Score)', data=df).fit()
result = sm.stats.anova_lm(model, type=2)

# Print the result
print(result)
```

```
                                    df       sum_sq      mean_sq  \
C(Complain)                        1.0  1608.706535  1608.706535
C(Satisfaction_Score)              4.0     0.014756     0.003689
C(Complain):C(Satisfaction_Score)  4.0     0.027065     0.006766
```

```
Residual                          9990.0    13.907244     0.001392

                                           F    PR(>F)
C(Complain)                     1.155583e+06  0.000000
C(Satisfaction_Score)           2.649905e+00  0.031512
C(Complain):C(Satisfaction_Score)  4.860418e+00  0.000648
Residual                                 NaN       NaN
```

```
df_com = df[df['Complain'] == 1]
sns.countplot(data = df_com[['Customer_Exited', 'Satisfaction_Score']], hue = 'Satisfaction_Score', x = 'Customer_Exited')
```

```
<Axes: xlabel='Customer_Exited', ylabel='count'>
```




```
# H0: Customer churn and Satisfaction_Score are independent.
# H1: Both are dependent.
data = pd.crosstab(df_com['Customer_Exited'], df['Satisfaction_Score'])
stat, p, dof, expected = stats.chi2_contingency(data)

# interpret p-value
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (H0 holds true)')
```

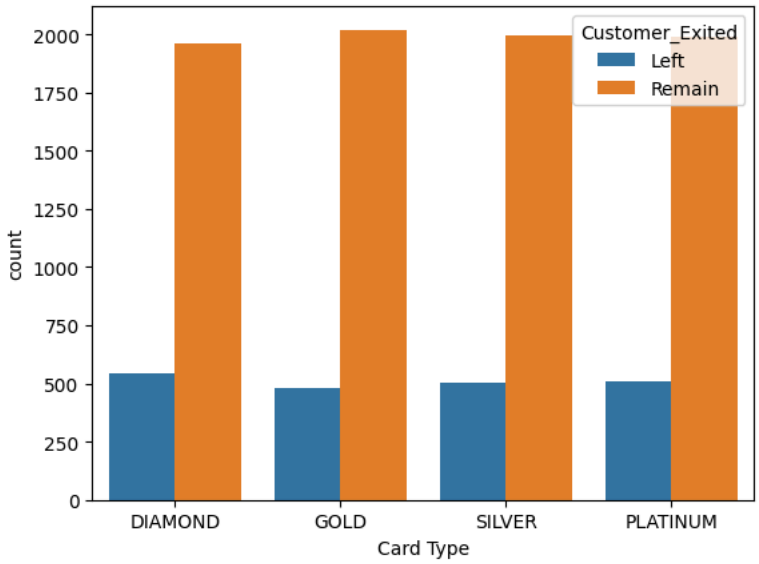
```
p value is 0.08383511492614232
Independent (H0 holds true)
```

```
# H0: Customer churn and card type are independent.
# H1: Both are dependent.
data = pd.crosstab(df['Customer_Exited'], df['Card Type'])
stat, p, dof, expected = stats.chi2_contingency(data)

# interpret p-value
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (H0 holds true)')
```

```
p value is 0.16794112067810177
Independent (H0 holds true)
```

```
sns.countplot(data = df[['Customer_Exited', 'Card Type']], x = 'Card Type', hue = 'Customer_Exited')
```

```
<Axes: xlabel='Card Type', ylabel='count'>
```

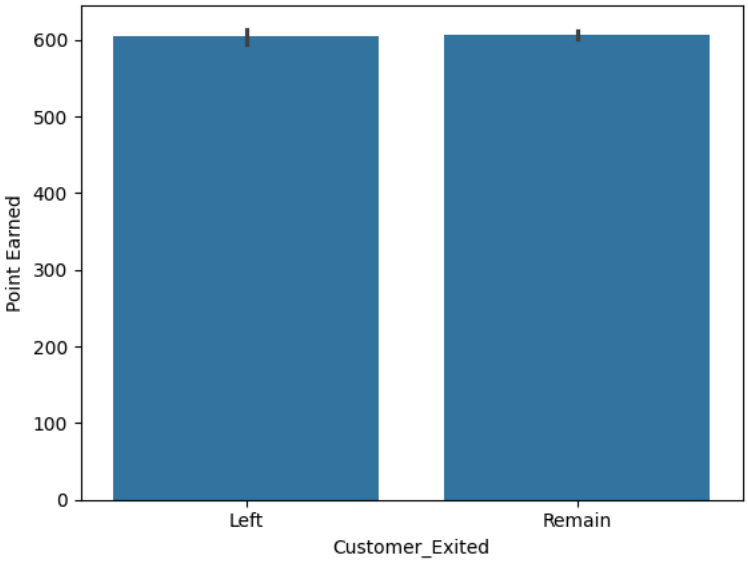



```
df.columns
```

```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited', 'Complain',
       'Satisfaction_Score', 'Card Type', 'Point Earned', 'Age_Seg',
       'Customer_Exited', 'Fin_Level'],
      dtype='object')
```

```
sns.barplot(data = df, x = 'Customer_Exited', y = 'Point Earned')
```

```
<Axes: xlabel='Customer_Exited', ylabel='Point Earned'>
```

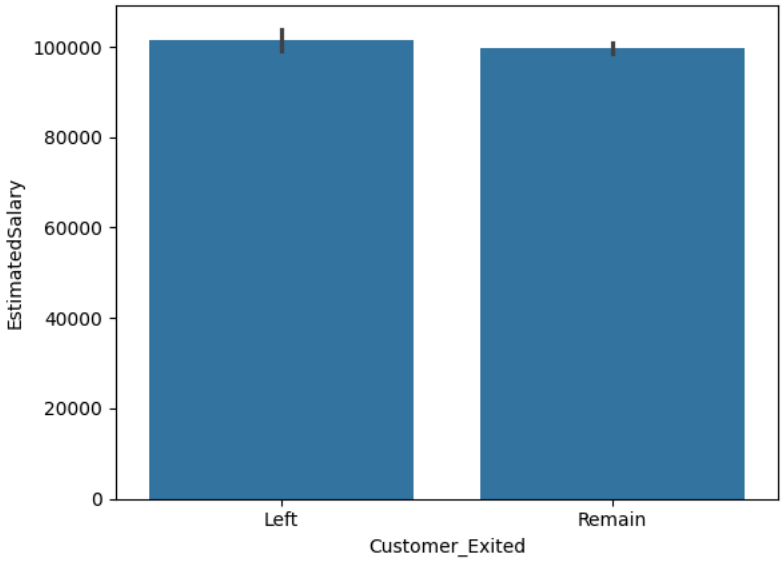



```
# H0: The mean value of loyality point for loyal and churned customer is same.
# H1: The mean value of loyality point for loyal and churned customer is different.
df_remain = df[df['Customer_Exited'] == 'Remain']['Point Earned']
df_left = df[df['Customer_Exited'] == 'Left']['Point Earned']
stat, p = stats.ttest_ind(df_remain, df_left)

alpha = 0.05
print('The p value is ', p)
if p > alpha:
  print('Churn does not depends on loyality point.')
else:
  print('Churn depends on loyality point.')
```

```
The p value is  0.6435350184288993
Churn does not depends on loyality point.
```

```
sns.barplot(data = df, x = 'Customer_Exited', y = 'EstimatedSalary')
```

```
<Axes: xlabel='Customer_Exited', ylabel='EstimatedSalary'>
```




```
# H0: The mean value of EstimatedSalary for loyal and churned customer is same.
# H1: The mean value of EstimatedSalary for loyal and churned customer is different.
df_remain = df[df['Customer_Exited'] == 'Remain']['EstimatedSalary']
df_left = df[df['Customer_Exited'] == 'Left']['EstimatedSalary']
stat, p = stats.ttest_ind(df_remain, df_left)

alpha = 0.05
print('The p value is ', p)
if p > alpha:
  print('Churn does not depends on EstimatedSalary.')
else:
  print('Churn depends on EstimatedSalary.')
```