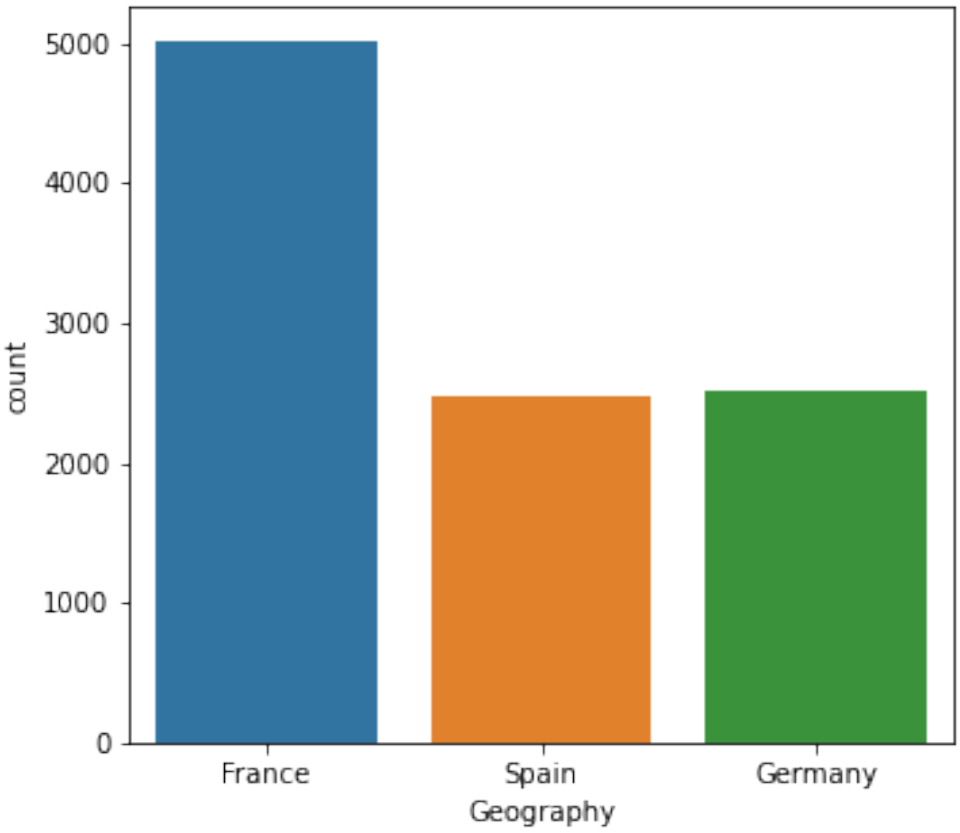
```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from google.colab import drive

drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd
data=pd.read_csv('/content/drive/MyDrive/IBM/Churn_Modelling.csv')
```

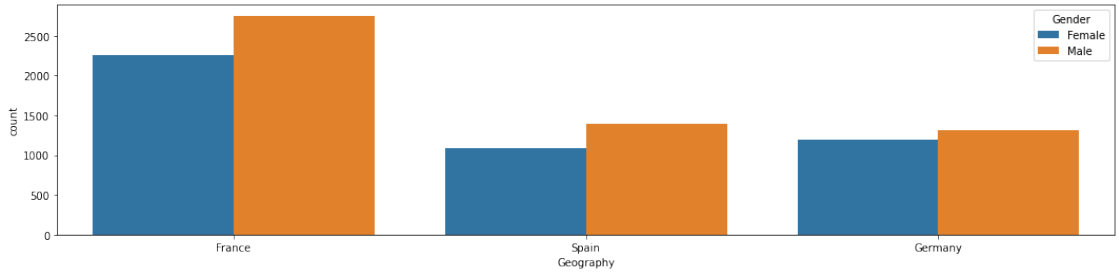
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.countplot(data['Geography'])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```




```
plt.figure(figsize=(18,4))
plt.plot()
sns.countplot(data['Geography'],hue=data['Gender'])
plt.show()
```

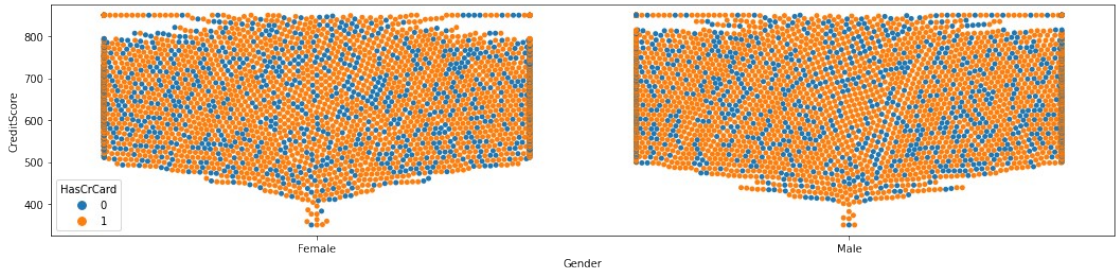
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```




```
plt.figure(figsize=(18,4))
plt.plot()
sns.swarmplot(data['Gender'], data['CreditScore'], hue =
```

```
data['HasCrCard'])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296:
UserWarning: 46.6% of the points cannot be placed; you may want to
decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296:
UserWarning: 53.9% of the points cannot be placed; you may want to
decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```




```
data.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  10000.00000    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

data.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

data.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
```

```
data.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  10000.00000    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
```

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

```
print(x.shape)
print(y.shape)

print(x.columns)
```

```
(10000, 13)
(10000,)
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore',
'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts',
'HasCrCard',
       'IsActiveMember', 'EstimatedSalary'],
      dtype='object')
```

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

```
   RowNumber  CustomerId  CreditScore  Age  Tenure    Balance
NumOfProducts  \
0          1    15634602          619   42       2       0.00
1
1          2    15647311          608   41       1   83807.86
1
2          3    15619304          502   42       8  159660.80
3
3          4    15701354          699   39       1       0.00
2
4          5    15737888          850   43       2  125510.82
1

   HasCrCard  IsActiveMember  EstimatedSalary  ...  Surname_Zubarev  \
0          1               1        101348.88  ...                0
1          0               1        112542.58  ...                0
2          1               0        113931.57  ...                0
3          0               0         93826.63  ...                0
4          1               1         79084.10  ...                0

   Surname_Zubareva  Surname_Zuev  Surname_Zuyev  Surname_Zuyeva  \
0                 0             0              0               0
1                 0             0              0               0
2                 0             0              0               0
3                 0             0              0               0
4                 0             0              0               0

   Geography_France  Geography_Germany  Geography_Spain  Gender_Female
\
0                 1                  0                0              1

1                 0                  0                1              1
```

```
2                 1                 0                 0              1

3                 1                 0                 0              1

4                 0                 0                 1              1


   Gender_Male
0            0
1            0
2            0
3            0
4            0

[5 rows x 2947 columns]
```

```
x.shape
```

```
(10000, 2947)
```

```
from sklearn import model_selection
x_train,x_test,y_train,y_test=model_selection.train_test_split(x,y,
test_size=0.2, random_state=0)

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

```
(8000, 2947)
(8000,)
(2000, 2947)
(2000,)
```

```
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  0.831470 -0.202167  0.169582 -0.464608  0.006661 -1.215717
0.809503
1  1.483423  0.807044 -2.304559  0.301026 -1.377440 -0.006312 -
0.921591
2 -0.687448 -1.519081 -1.191196 -0.943129 -1.031415  0.579935 -
```

```
0.921591
3  0.114875  1.241115  0.035566  0.109617  0.006661  0.473128 -
0.921591
4 -1.599698 -1.110869  2.056114  1.736588  1.044737  0.810193
0.809503

       7         8         9     ...      2937      2938      2939
2940  \
0  0.642595 -1.032270  1.106432  ... -0.015813 -0.011181 -0.011181 -
0.015813
1  0.642595  0.968738 -0.748664  ... -0.015813 -0.011181 -0.011181 -
0.015813
2  0.642595 -1.032270  1.485335  ... -0.015813 -0.011181 -0.011181 -
0.015813
3  0.642595 -1.032270  1.276528  ... -0.015813 -0.011181 -0.011181 -
0.015813
4  0.642595  0.968738  0.558378  ... -0.015813 -0.011181 -0.011181 -
0.015813

       2941      2942      2943      2944      2945      2946
0 -0.015813 -1.014607 -0.569844  1.743090  1.091687 -1.091687
1 -0.015813 -1.014607  1.754865 -0.573694 -0.916013  0.916013
2 -0.015813  0.985604 -0.569844 -0.573694  1.091687 -1.091687
3 -0.015813 -1.014607 -0.569844  1.743090 -0.916013  0.916013
4 -0.015813 -1.014607 -0.569844  1.743090  1.091687 -1.091687

[5 rows x 2947 columns]
```