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

1. Load the dataset

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

```
(10000, 14)
```

```
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  10000.00000    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

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

# Data Cleaning
df.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1, inplace=True)
df.head(5)

   CreditScore Geography  Gender  Age  Tenure    Balance
NumOfProducts  \
0          619    France  Female   42       2       0.00
1
```

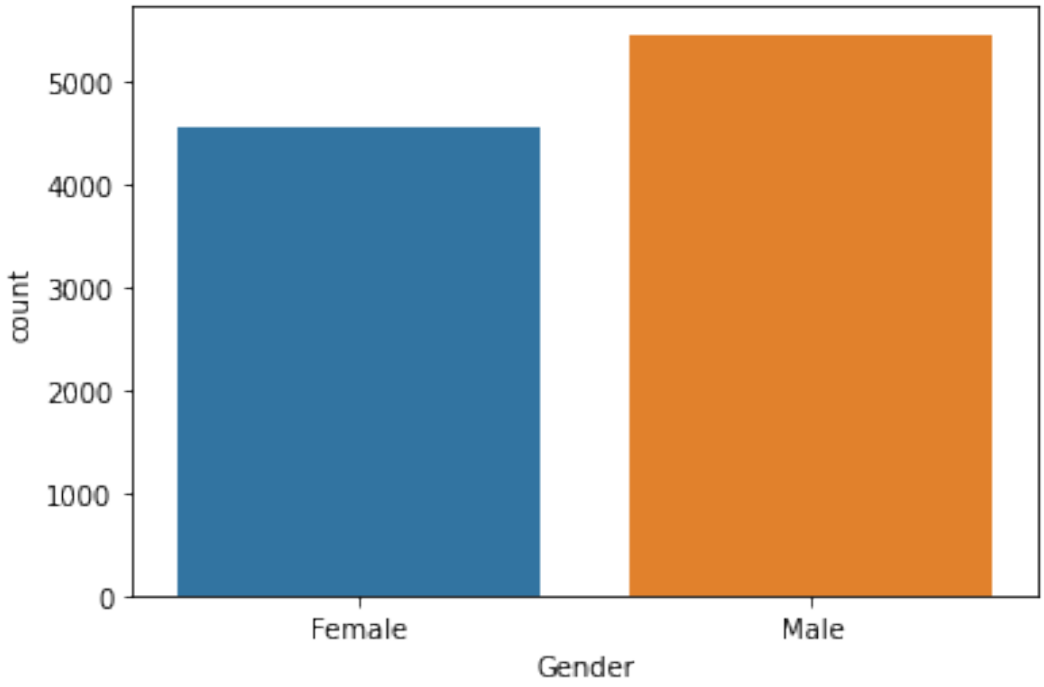
```
1          608     Spain  Female   41       1   83807.86
1
2          502    France  Female   42       8  159660.80
3
3          699    France  Female   39       1       0.00
2
4          850     Spain  Female   43       2  125510.82
1
```

	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	1	101348.88	1
1	0	1	112542.58	0
2	1	0	113931.57	1
3	0	0	93826.63	0
4	1	1	79084.10	0

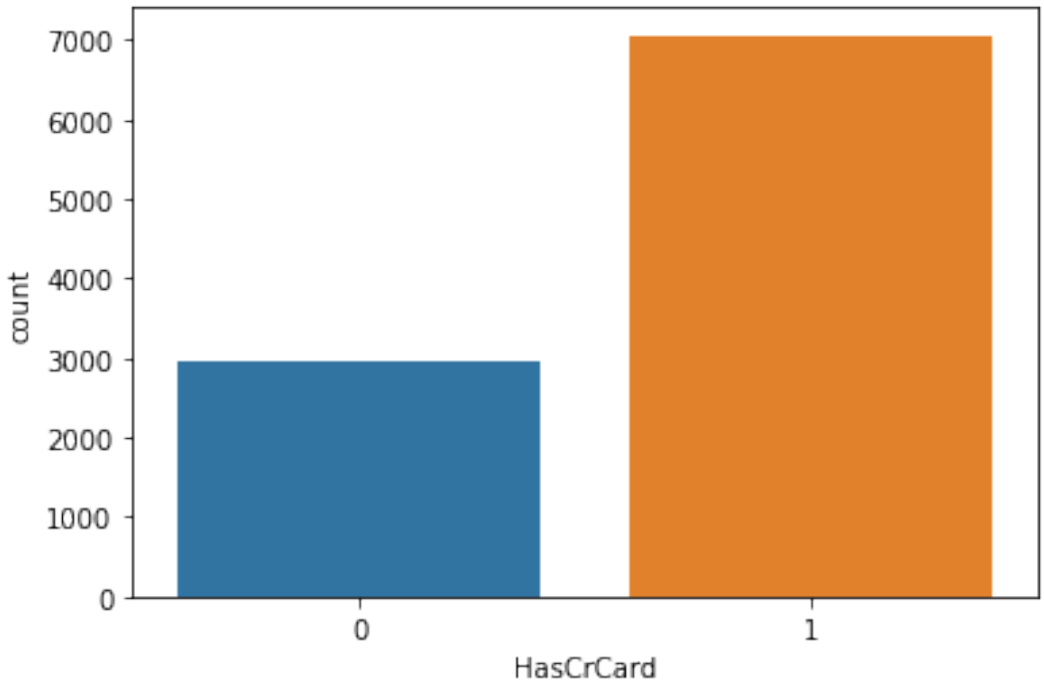
2. Univariate Analysis

```
# Categorical Data

# Gender
sns.countplot(df.Gender)
plt.show()
# HasCrCard
sns.countplot(df.HasCrCard)
plt.show()
# IsActiveMember
sns.countplot(df.IsActiveMember)
plt.show()
# Geography
sns.countplot(df.Geography)
plt.show()
```

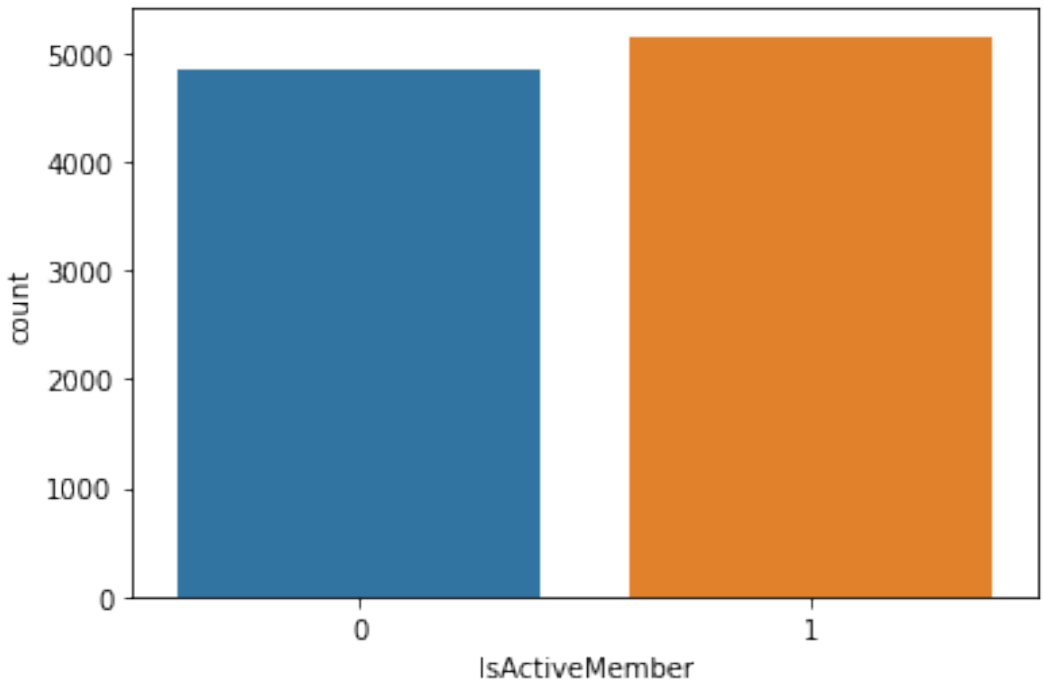
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.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



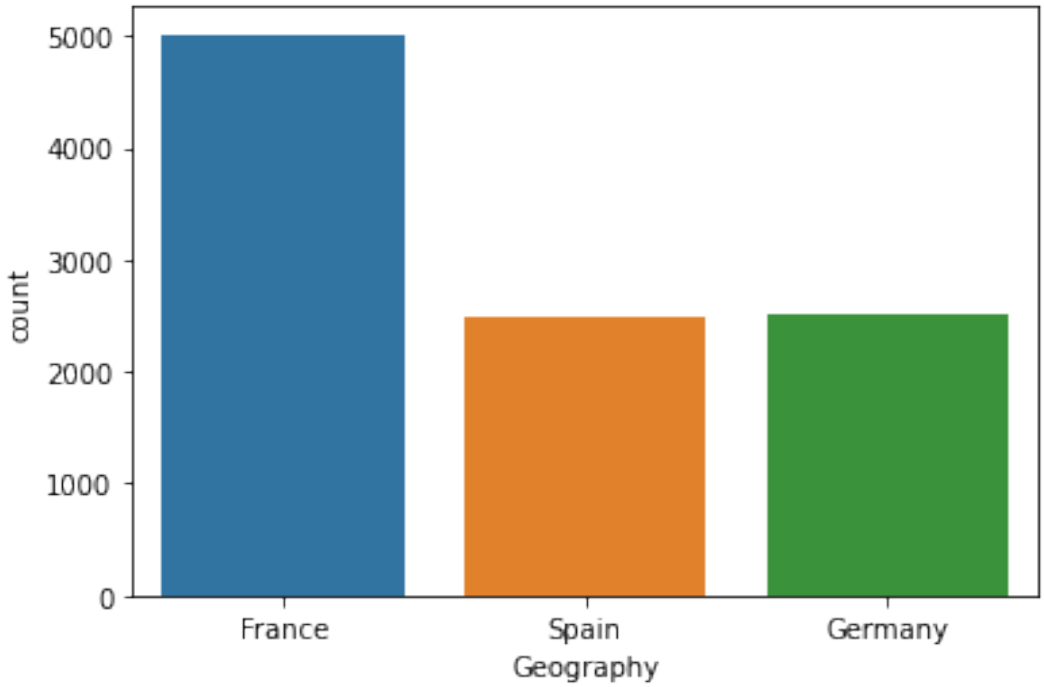

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.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



```
/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
```



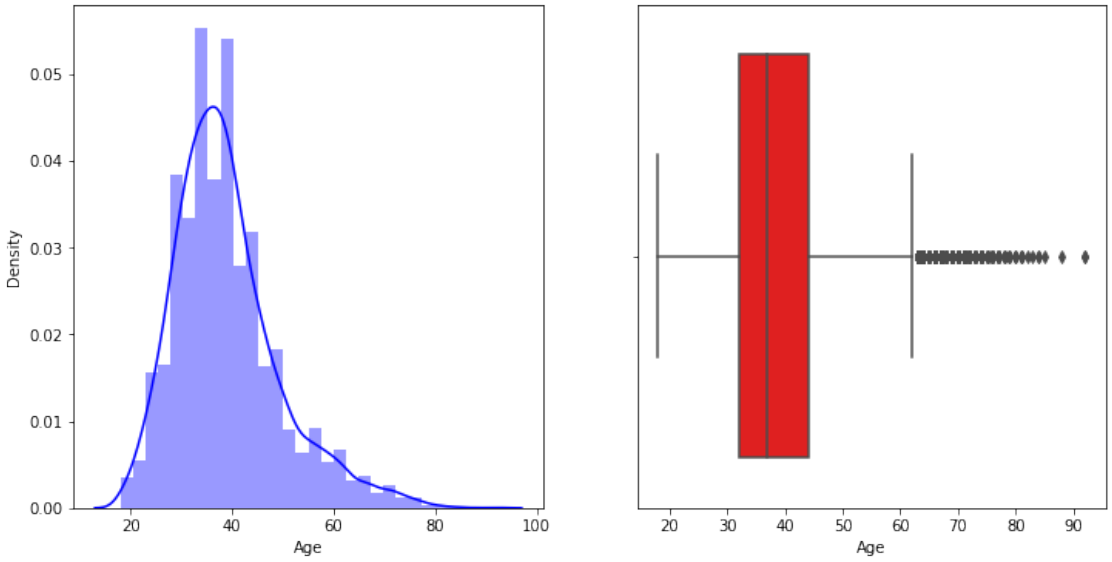

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.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




```
# Numerical Data

# Age
fig1, axes1 = plt.subplots(1, 2, figsize=(12,6))
sns.distplot(df.Age, hist=True, kde=True, color="b", bins=30,
ax=axes1[0])
sns.boxplot(df.Age, color='r', ax=axes1[1])
plt.show()
# CreditScore
fig2, axes2 = plt.subplots(1, 2, figsize=(12,6))
sns.distplot(df.CreditScore, hist=True, kde=True, color="b", bins=30,
ax=axes2[0])
sns.boxplot(df.CreditScore, color='r', ax=axes2[1])
plt.show()
# Tenure
fig3, axes3 = plt.subplots(1, 2, figsize=(12,6))
sns.distplot(df.Tenure, hist=True, kde=True, color="b", bins=30,
ax=axes3[0])
sns.boxplot(df.Tenure, color='r', ax=axes3[1])
plt.show()
# Balance
fig4, axes4 = plt.subplots(1, 2, figsize=(12,6))
sns.distplot(df.Balance, hist=True, kde=True, color="b", bins=30,
ax=axes4[0])
sns.boxplot(df.Balance, color='r', ax=axes4[1])
plt.show()
# NumOfProducts
fig5, axes5 = plt.subplots(1, 2, figsize=(12,6))
sns.distplot(df.NumOfProducts, hist=True, kde=True, color="b",
```

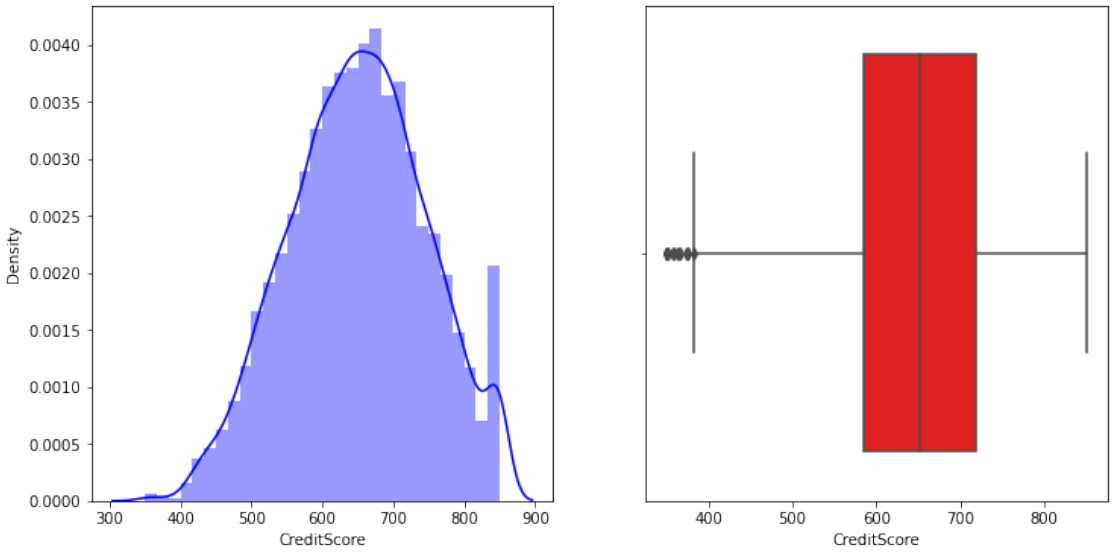
```
bins=30, ax=axes5[0])
sns.boxplot(df.NumOfProducts, color='r', ax=axes5[1])
plt.show()
# EstimatedSalary
fig6, axes6 = plt.subplots(1, 2, figsize=(12,6))
sns.distplot(df.EstimatedSalary, hist=True, kde=True, color="b",
bins=30, ax=axes6[0])
sns.boxplot(df.EstimatedSalary, color='r', ax=axes6[1])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/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
```

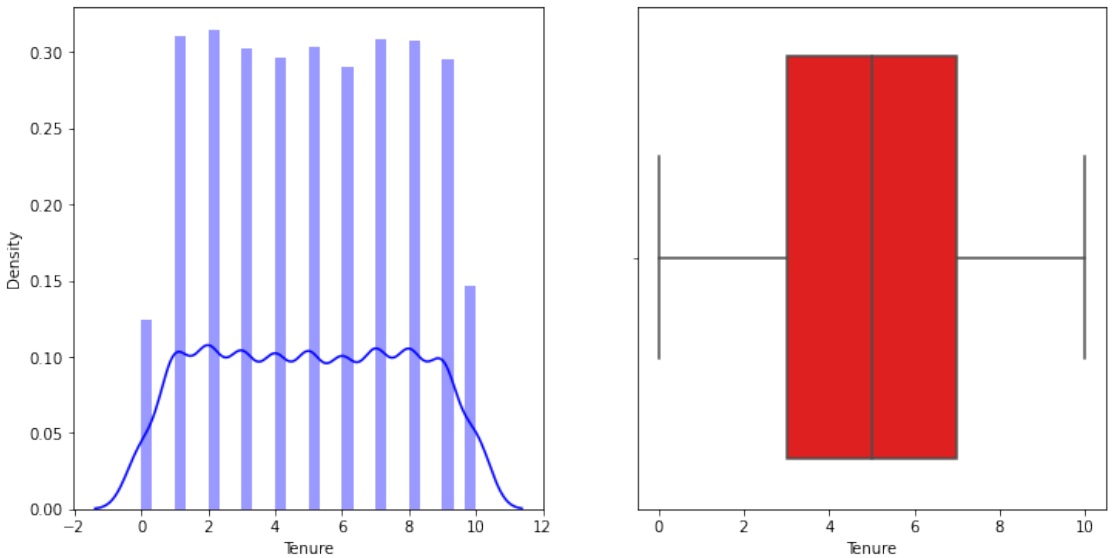



```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/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
```

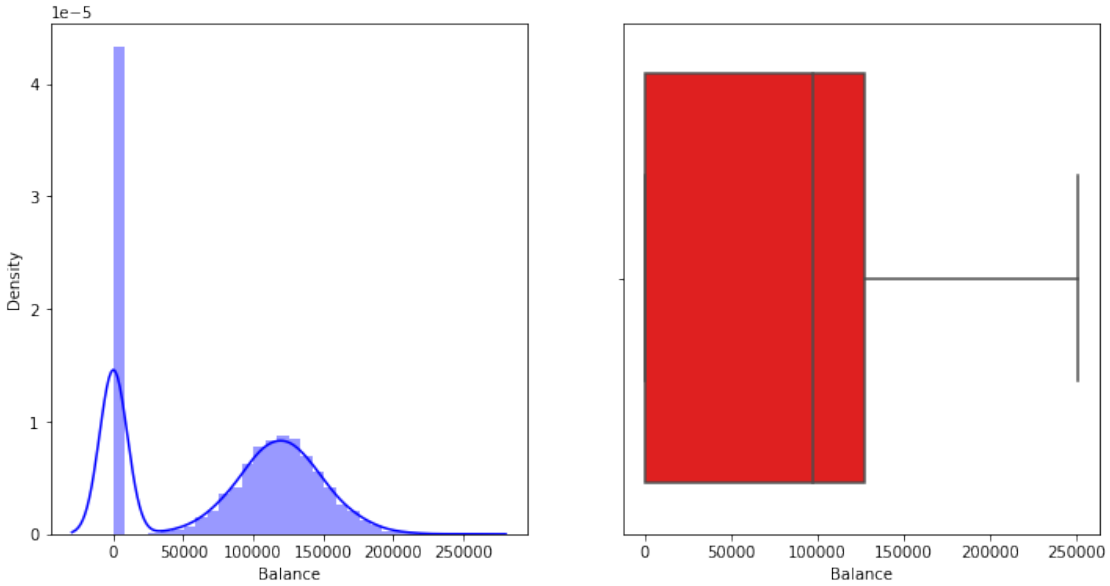
```
error or misinterpretation.
  FutureWarning
```



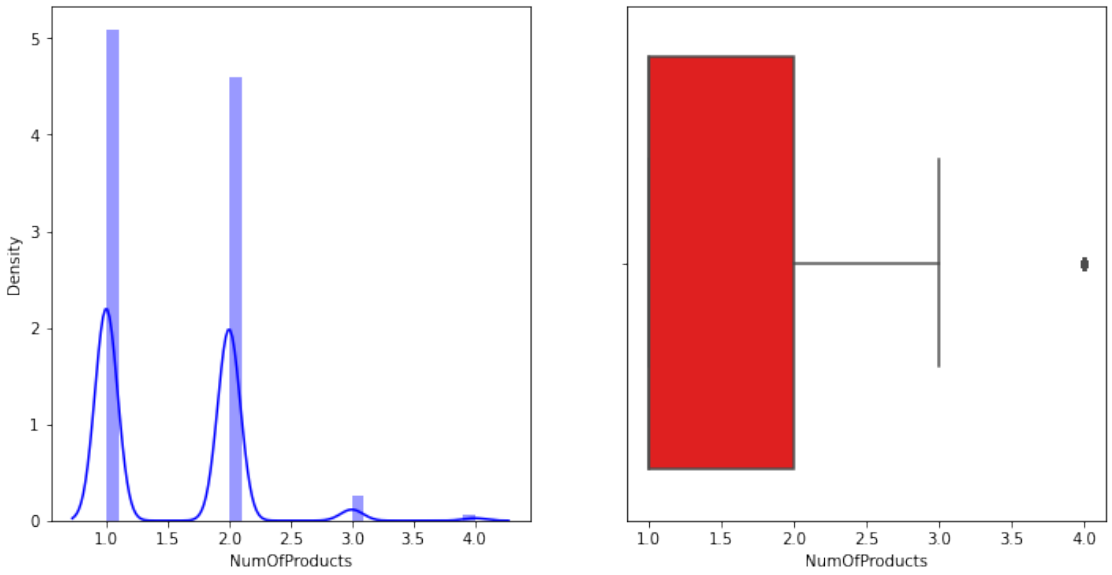

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/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
```



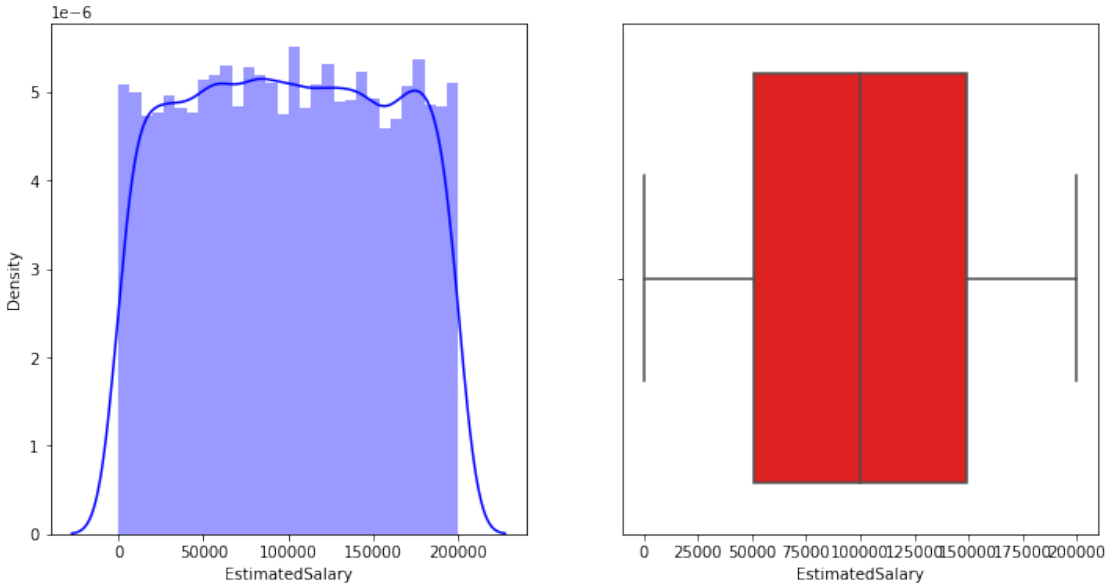
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/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
```




```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/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
```




```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/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
```



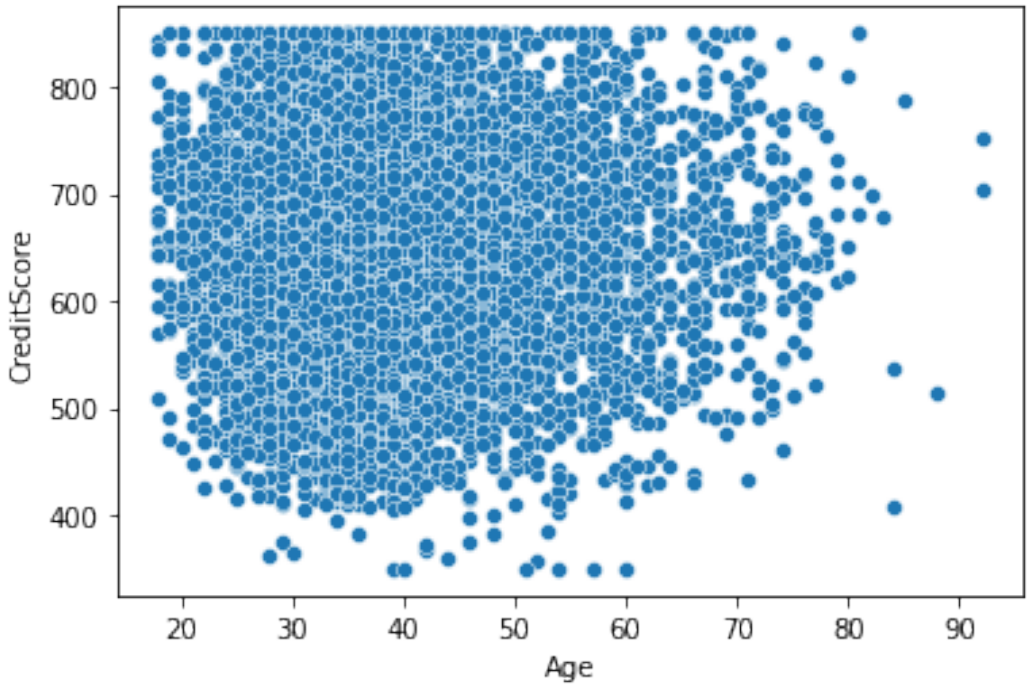
Bivariate Analysis

```
# Between COntinuous Variables

df[['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts',
'EstimatedSalary']].corr()
```

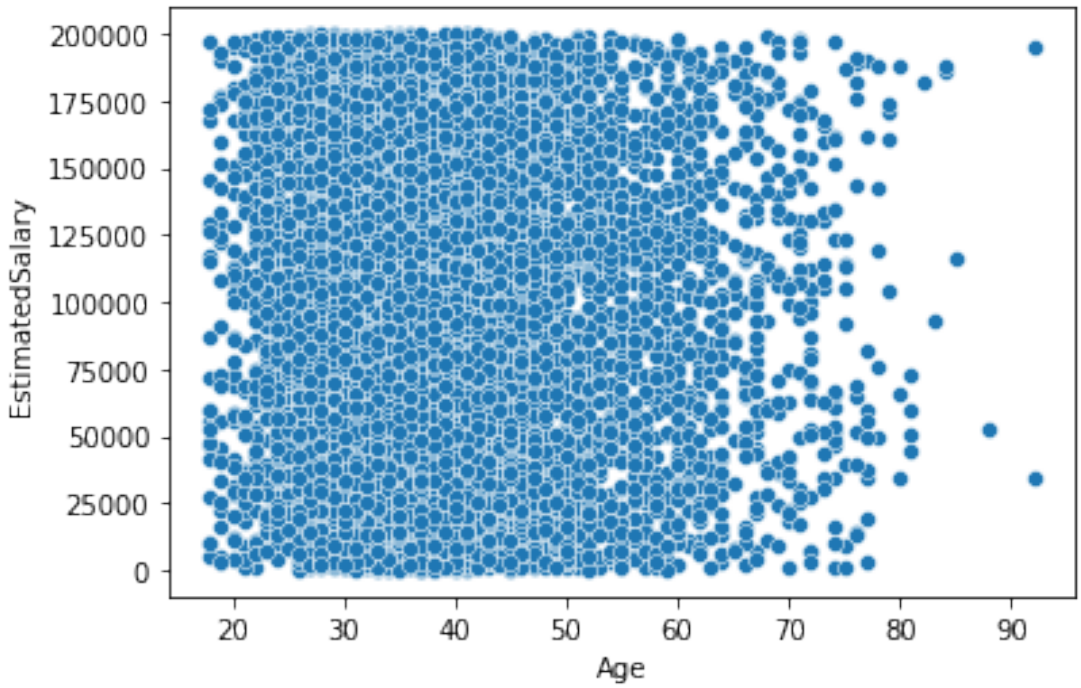
```
                 CreditScore       Age    Tenure   Balance
NumOfProducts  \
CreditScore         1.000000 -0.003965  0.000842  0.006268
0.012238
Age                -0.003965  1.000000 -0.009997  0.028308      -
0.030680
Tenure              0.000842 -0.009997  1.000000 -0.012254
0.013444
Balance             0.006268  0.028308 -0.012254  1.000000      -
0.304180
NumOfProducts       0.012238 -0.030680  0.013444 -0.304180
1.000000
EstimatedSalary    -0.001384 -0.007201  0.007784  0.012797
0.014204

                 EstimatedSalary
CreditScore            -0.001384
Age                    -0.007201
Tenure                  0.007784
Balance                 0.012797
NumOfProducts           0.014204
EstimatedSalary         1.000000
```

```
sns.scatterplot(df.Age, df.CreditScore)
plt.show()
sns.scatterplot(df.Age, df.EstimatedSalary)
plt.show()
sns.scatterplot(df.EstimatedSalary, df.Balance)
plt.show()
```

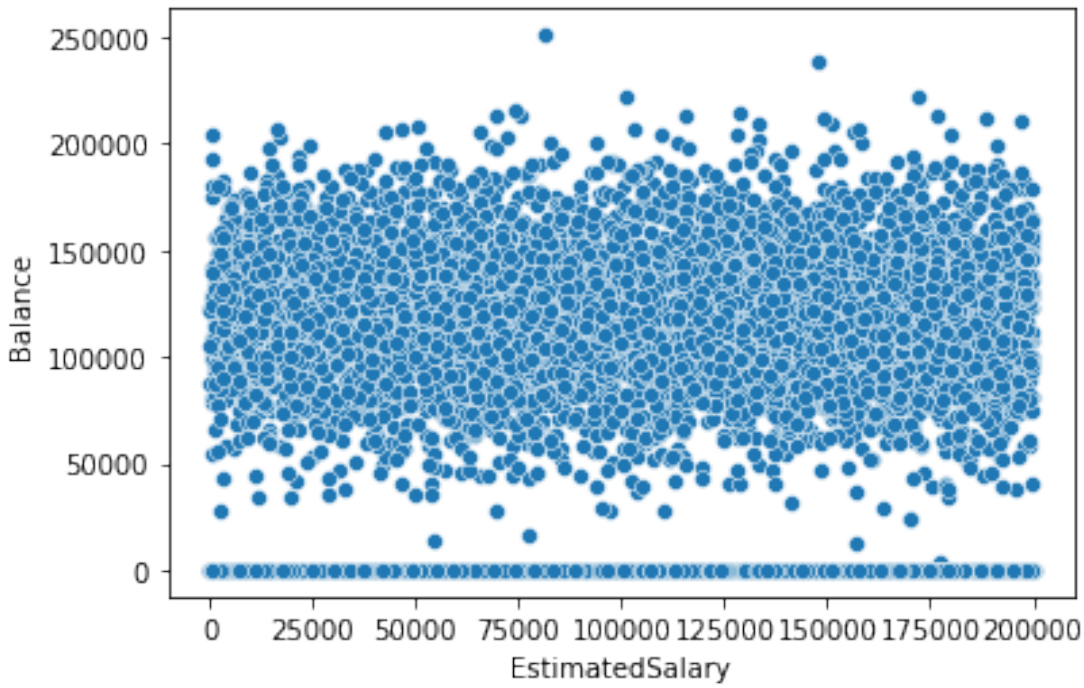
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.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




```
/usr/local/lib/python3.7/dist-packages/seaborn/_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/_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
```



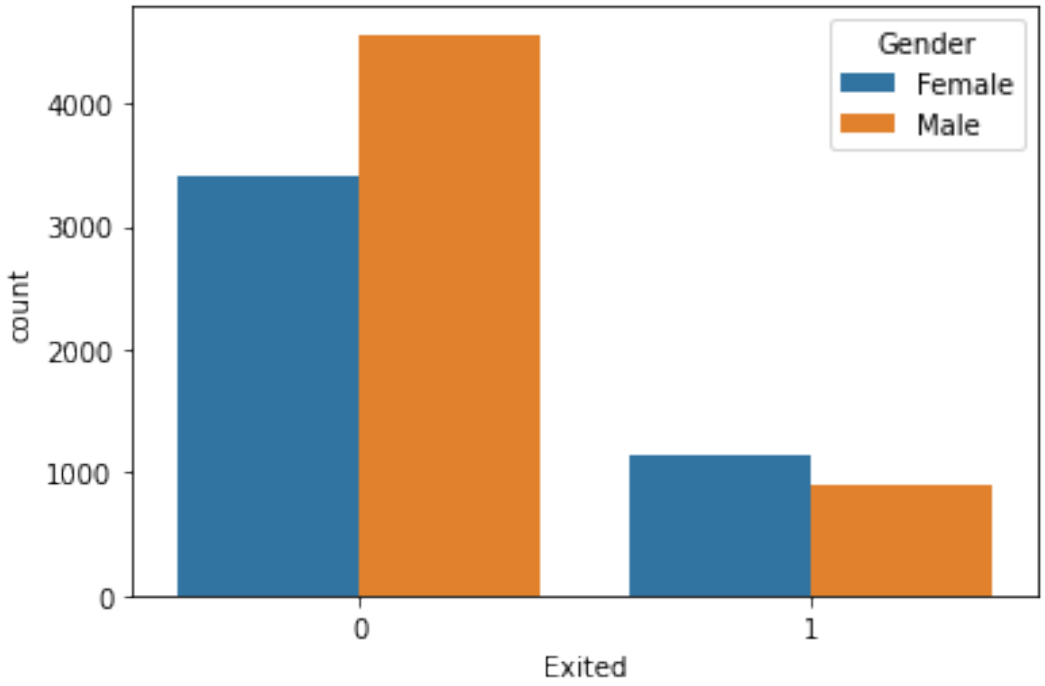

```
# Between Continuous and Categorical Variables

df.groupby(by='Exited').agg('mean')[['CreditScore', 'Age', 'Tenure',
'Balance', 'NumOfProducts', 'EstimatedSalary']]
```

Exited	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary
0	651.853196	37.408389	5.033279	72745.296779	1.544267	99738.391772
1	645.351497	44.837997	4.932744	91108.539337	1.475209	101465.677531

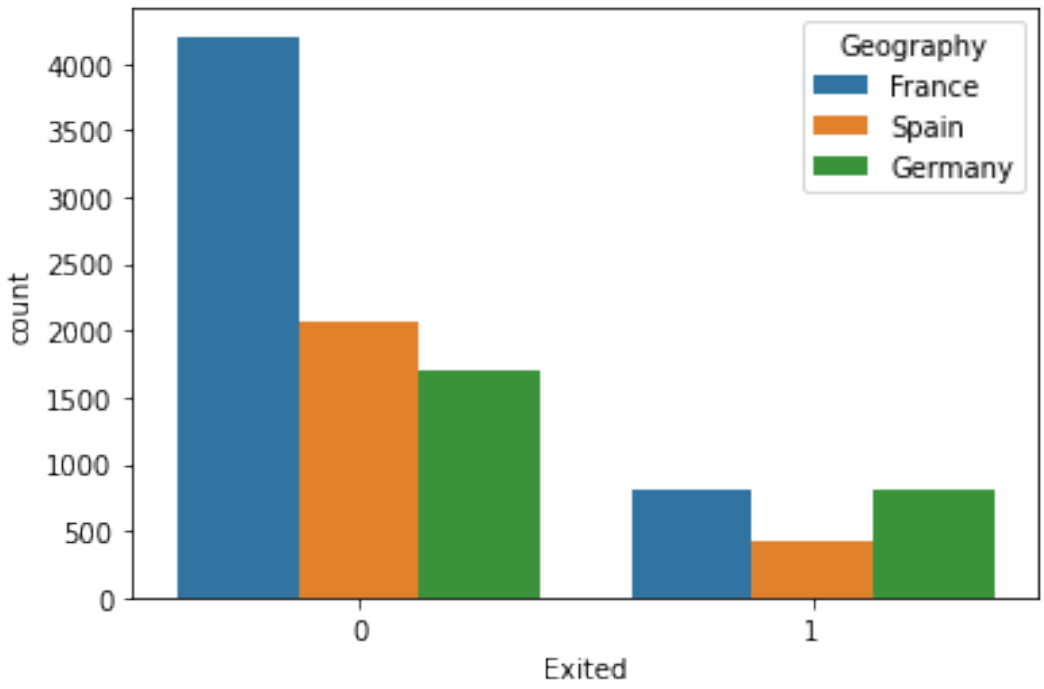
```
# Between Categorical Variables

sns.countplot(df.Exited, hue=df.Gender)
```

```
plt.show()
sns.countplot(df.Exited, hue=df.Geography)
plt.show()
sns.countplot(df.Exited, hue=df.HasCrCard)
plt.show()
sns.countplot(df.Exited, hue=df.IsActiveMember)
plt.show()
```

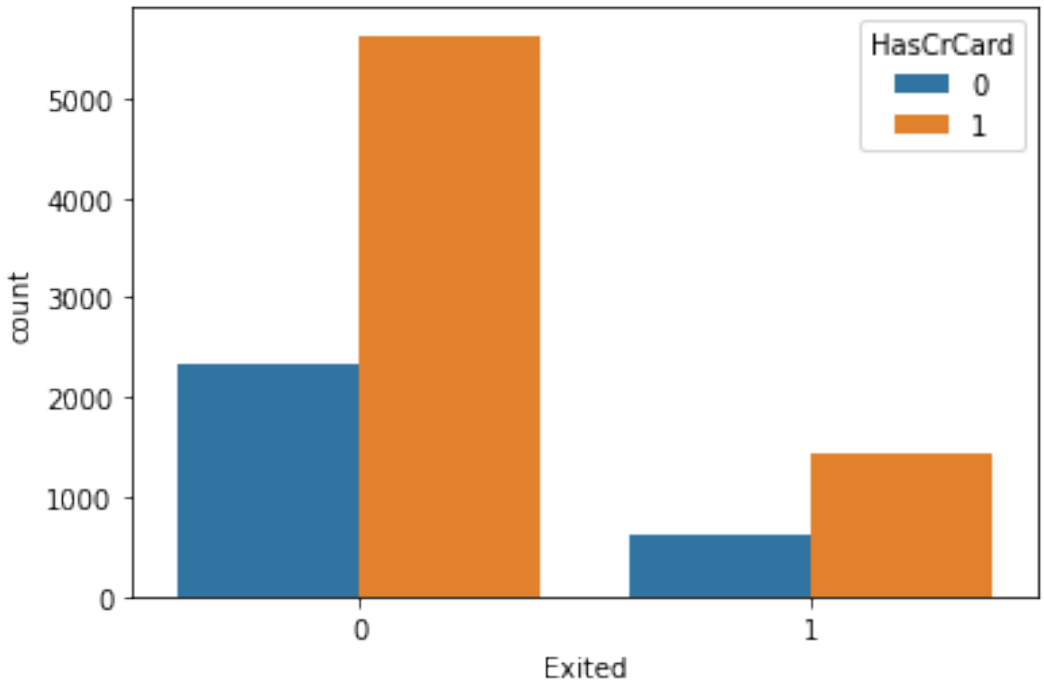
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.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



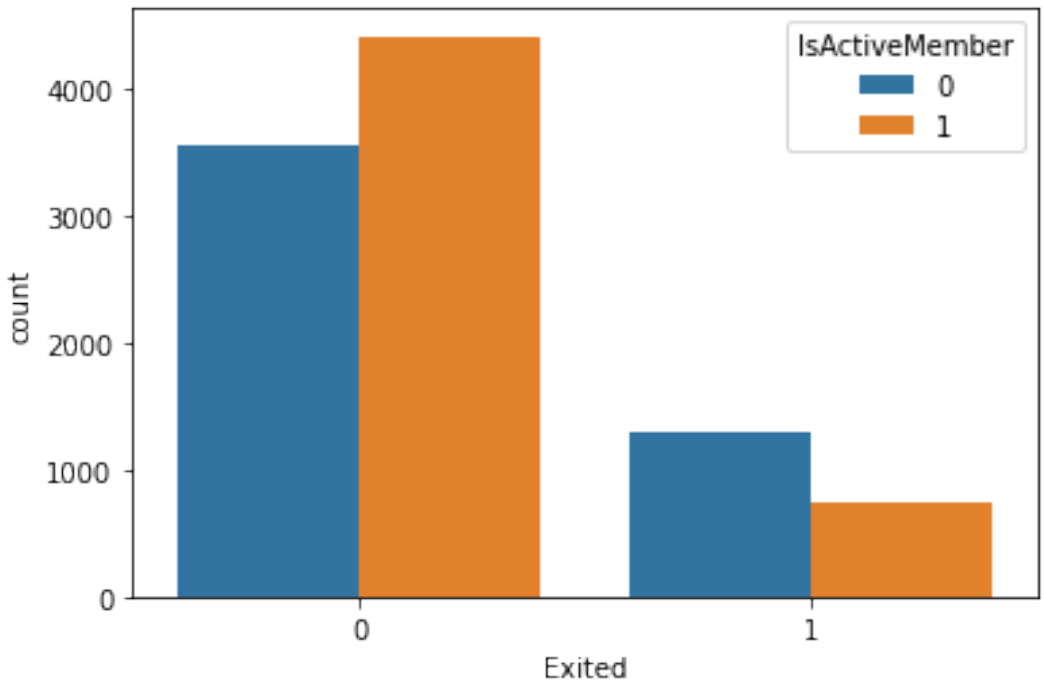

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.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




```
/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
```



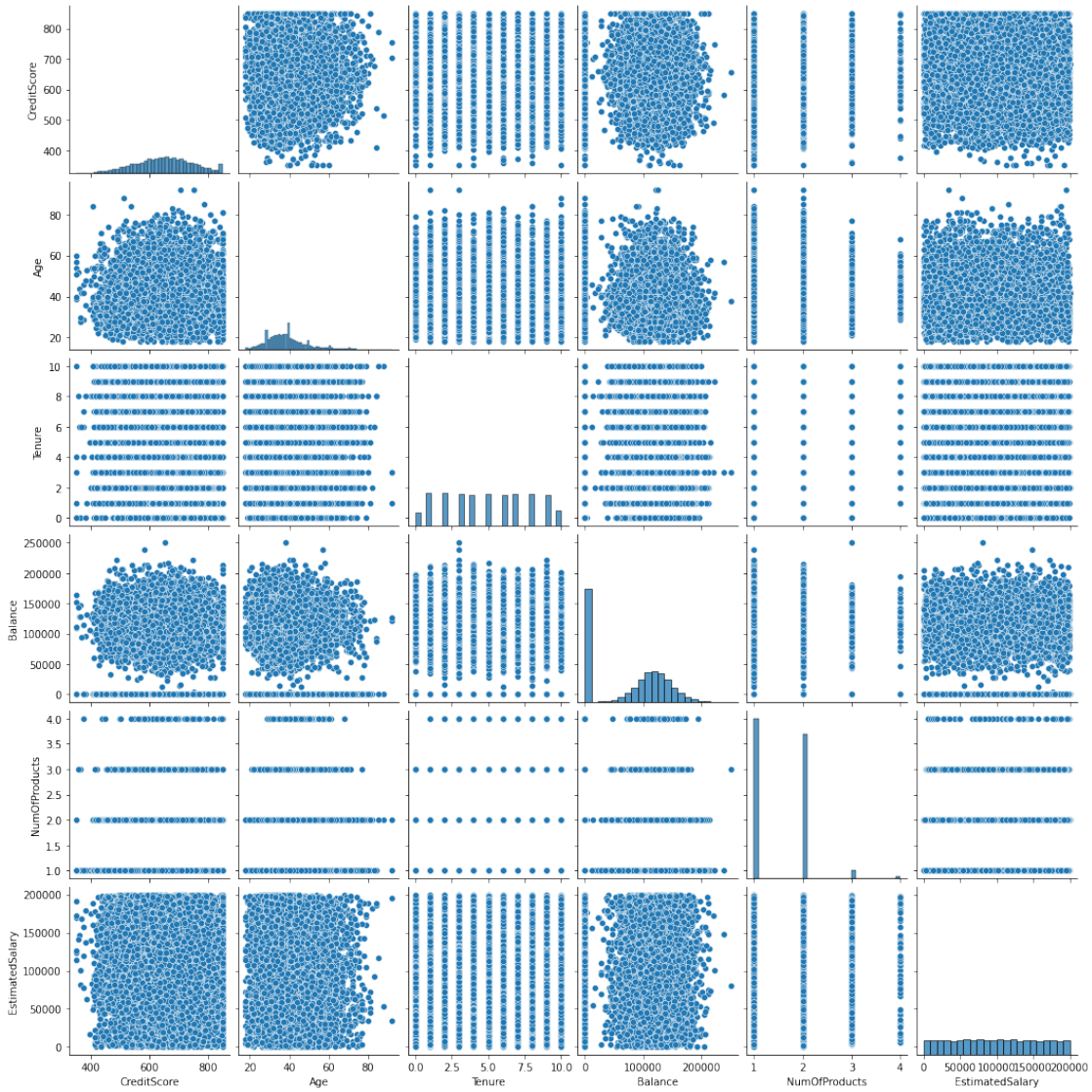
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.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



Multivariate Analysis

```
Exited = {0: 'X', 1: 'Y'}
sns.pairplot(data=df[['CreditScore', 'Age', 'Tenure', 'Balance',
'NumOfProducts', 'EstimatedSalary']])
```

```
<seaborn.axisgrid.PairGrid at 0x7f36f4f60f90>
```



3. Descriptive Analysis

```
df.head()
```

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

```
   HasCrCard  IsActiveMember  EstimatedSalary  Exited
0          1               1        101348.88       1
1          0               1        112542.58       0
2          1               0        113931.57       1
3          0               0         93826.63       0
4          1               1         79084.10       0
```

```
df.describe()
```

```
        CreditScore           Age        Tenure        Balance
NumOfProducts  \
count  10000.000000  10000.000000  10000.000000   10000.000000
10000.000000
mean     650.528800     38.921800      5.012800   76485.889288
1.530200
std       96.653299     10.487806      2.892174   62397.405202
0.581654
min      350.000000     18.000000      0.000000       0.000000
1.000000
25%      584.000000     32.000000      3.000000       0.000000
1.000000
50%      652.000000     37.000000      5.000000   97198.540000
1.000000
75%      718.000000     44.000000      7.000000  127644.240000
2.000000
max      850.000000     92.000000     10.000000  250898.090000
4.000000

         HasCrCard  IsActiveMember  EstimatedSalary        Exited
count  10000.00000    10000.000000     10000.000000  10000.000000
mean       0.70550        0.515100    100090.239881      0.203700
std        0.45584        0.499797     57510.492818      0.402769
min        0.00000        0.000000        11.580000      0.000000
25%        0.00000        0.000000     51002.110000      0.000000
50%        1.00000        1.000000    100193.915000      0.000000
75%        1.00000        1.000000    149388.247500      0.000000
max        1.00000        1.000000    199992.480000      1.000000
```

```
df.var()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this
will raise TypeError.  Select only valid columns before calling the
reduction.
  """Entry point for launching an IPython kernel.

CreditScore        9.341860e+03
Age                1.099941e+02
Tenure             8.364673e+00
Balance            3.893436e+09
```

```
NumOfProducts      3.383218e-01
HasCrCard          2.077905e-01
IsActiveMember     2.497970e-01
EstimatedSalary    3.307457e+09
Exited             1.622225e-01
dtype: float64
```

```
df.describe(include='object')
```

```
       Geography Gender
count      10000  10000
unique         3      2
top       France   Male
freq        5014   5457
```

```
df['Gender'].value_counts().to_frame()
```

```
        Gender
Male      5457
Female    4543
```

```
df['Geography'].value_counts().to_frame()
```

```
         Geography
France        5014
Germany       2509
Spain         2477
```

4. Handle the Missing Values

```
df.dropna(axis=0, inplace=True)
```

5. Find the outliers and replace the outliers

```
df
```

```
      CreditScore Geography  Gender  Age  Tenure    Balance
NumOfProducts  \
0             619    France  Female   42       2       0.00
1
1             608     Spain  Female   41       1   83807.86
1
2             502    France  Female   42       8  159660.80
3
3             699    France  Female   39       1       0.00
2
4             850     Spain  Female   43       2  125510.82
1
...           ...       ...     ...  ...     ...        ...
...
9995          771    France    Male   39       5       0.00
2
9996          516    France    Male   35      10   57369.61
```

```
1
9997          709    France  Female   36       7       0.00
1
9998          772   Germany    Male   42       3   75075.31
2
9999          792    France  Female   28       4  130142.79
1

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

[10000 rows x 11 columns]
```

```
from scipy import stats

z = np.abs(stats.zscore(df[['CreditScore', 'Age', 'Tenure', 'Balance',
'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary']]))
print(np.where(z > 3))
```

```
(array([   7,   70,   85,  158,  230,  252,  310,  371,  602,  766,
888,
       1009, 1039, 1055, 1205, 1254, 1342, 1405, 1410, 1469, 1488,
1614,
       1631, 1701, 1790, 1838, 1876, 1933, 1962, 2002, 2012, 2108,
2124,
       2159, 2164, 2196, 2285, 2433, 2458, 2462, 2473, 2499, 2509,
2541,
       2553, 2614, 2617, 2772, 2778, 2855, 2872, 2901, 2925, 3033,
3110,
       3142, 3152, 3311, 3317, 3365, 3366, 3378, 3382, 3387, 3396,
3499,
       3531, 3602, 3651, 3691, 3702, 3813, 3826, 3841, 3880, 3888,
3994,
       4013, 4014, 4162, 4166, 4256, 4260, 4273, 4318, 4366, 4378,
4403,
       4501, 4511, 4516, 4590, 4606, 4644, 4654, 4748, 4801, 4815,
4822,
       4832, 4931, 5010, 5068, 5137, 5197, 5223, 5235, 5299, 5313,
5377,
       5386, 5490, 5508, 5664, 5671, 5700, 5783, 5840, 5904, 5957,
```

```
6116,
       6150, 6167, 6172, 6173, 6230, 6278, 6279, 6366, 6443, 6530,
6581,
       6721, 6750, 6759, 6875, 7057, 7058, 7063, 7202, 7243, 7257,
7272,
       7302, 7362, 7375, 7457, 7499, 7523, 7526, 7552, 7567, 7692,
7698,
       7719, 7724, 7729, 7788, 7898, 7956, 8019, 8041, 8156, 8217,
8458,
       8469, 8590, 8683, 8686, 8723, 8762, 8787, 8850, 8865, 8900,
8923,
       9080, 9112, 9215, 9255, 9292, 9309, 9323, 9324, 9333, 9370,
9411,
       9425, 9472, 9490, 9540, 9555, 9565, 9587, 9589, 9624, 9646,
9671,
       9736, 9894, 9936]), array([4, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 4, 1, 0, 1, 4, 4, 1,
       0, 4, 1, 0, 4, 1, 0, 1, 1, 1, 4, 1, 1, 4, 4, 1, 1, 4, 0, 4, 4,
4,
       1, 4, 4, 1, 1, 1, 4, 1, 1, 1, 1, 1, 4, 1, 1, 4, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 4, 4, 1, 4, 1, 4, 1, 1, 1, 1,
4,
       1, 4, 4, 1, 4, 1, 4, 4, 1, 1, 4, 1, 1, 4, 1, 4, 1, 1, 4, 1, 1,
1,
       4, 1, 1, 1, 1, 4, 1, 1, 4, 1, 1, 4, 1, 4, 1, 1, 1, 4, 1, 1, 1,
1,
       1, 4, 1, 4, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 4, 1, 1, 1, 1, 4, 1,
4,
       1, 4, 4, 1, 1, 1, 1, 4, 1, 1, 1, 1, 4, 4, 1, 0, 0, 1, 4, 1, 1,
4,
       1, 1, 4, 4, 1, 1, 4, 1, 1, 4, 4, 1, 1, 1, 4, 1, 4, 1, 1, 0, 1,
1,
       1, 1, 1]))

df = df[(z < 3).all(axis=1)]
df

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

```
...
9995          771    France    Male   39       5       0.00
2
9996          516    France    Male   35      10   57369.61
1
9997          709    France  Female   36       7       0.00
1
9998          772   Germany    Male   42       3   75075.31
2
9999          792    France  Female   28       4  130142.79
1

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

[9799 rows x 11 columns]
```

5. Encoding Categorical Columns

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df['Geography'] = le.fit_transform(df['Geography'])
df['Gender'] = le.fit_transform(df['Gender'])
df

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy


      CreditScore  Geography  Gender  Age  Tenure    Balance
NumOfProducts  \
0             619          0       0   42       2       0.00
1
1             608          2       0   41       1   83807.86
1
2             502          0       0   42       8  159660.80
3
3             699          0       0   39       1       0.00
2
4             850          2       0   43       2  125510.82
1
...           ...        ...     ...  ...     ...        ...
...
9995          771          0       1   39       5       0.00
2
9996          516          0       1   35      10   57369.61
1
9997          709          0       0   36       7       0.00
1
9998          772          1       1   42       3   75075.31
2
9999          792          0       0   28       4  130142.79
1

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

[9799 rows x 11 columns]
```

6. Split the data into dependent and independent variables

```
indep_var = df.iloc[:,0:10].values
indep_var
```

```
array([[6.1900000e+02, 0.0000000e+00, 0.0000000e+00, ...,
1.0000000e+00,
        1.0000000e+00, 1.0134888e+05],
       [6.0800000e+02, 2.0000000e+00, 0.0000000e+00, ...,
0.0000000e+00,
        1.0000000e+00, 1.1254258e+05],
       [5.0200000e+02, 0.0000000e+00, 0.0000000e+00, ...,
1.0000000e+00,
        0.0000000e+00, 1.1393157e+05],
       ...,
       [7.0900000e+02, 0.0000000e+00, 0.0000000e+00, ...,
0.0000000e+00,
        1.0000000e+00, 4.2085580e+04],
       [7.7200000e+02, 1.0000000e+00, 1.0000000e+00, ...,
1.0000000e+00,
        0.0000000e+00, 9.2888520e+04],
       [7.9200000e+02, 0.0000000e+00, 0.0000000e+00, ...,
1.0000000e+00,
        0.0000000e+00, 3.8190780e+04]])
```

```
dep_var = df.iloc[:,10:].values
dep_var
```

```
array([[1],
       [0],
       [1],
       ...,
       [1],
       [1],
       [0]])
```

7. Splitting the data into train and test

```
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(indep_var, dep_var,
test_size=0.3, random_state=0)

xtrain
```

```
array([[5.5600000e+02, 0.0000000e+00, 0.0000000e+00, ...,
1.0000000e+00,
        0.0000000e+00, 1.7514920e+05],
       [6.6100000e+02, 2.0000000e+00, 0.0000000e+00, ...,
1.0000000e+00,
        0.0000000e+00, 6.9586270e+04],
       [7.6400000e+02, 0.0000000e+00, 1.0000000e+00, ...,
0.0000000e+00,
        0.0000000e+00, 1.3487834e+05],
       ...,
       [8.1100000e+02, 0.0000000e+00, 1.0000000e+00, ...,
1.0000000e+00,
```

```
        1.0000000e+00, 3.7977900e+03],
       [4.8700000e+02, 2.0000000e+00, 0.0000000e+00, ...,
1.0000000e+00,
        0.0000000e+00, 1.5875013e+05],
       [7.3200000e+02, 1.0000000e+00, 0.0000000e+00, ...,
1.0000000e+00,
        1.0000000e+00, 1.5752760e+05]])
```

```
xtest
```

```
array([[5.7900000e+02, 2.0000000e+00, 1.0000000e+00, ...,
0.0000000e+00,
        0.0000000e+00, 1.2021914e+05],
       [6.4600000e+02, 0.0000000e+00, 1.0000000e+00, ...,
0.0000000e+00,
        1.0000000e+00, 1.6425569e+05],
       [7.2100000e+02, 2.0000000e+00, 0.0000000e+00, ...,
1.0000000e+00,
        1.0000000e+00, 1.0393149e+05],
       ...,
       [7.6700000e+02, 2.0000000e+00, 1.0000000e+00, ...,
1.0000000e+00,
        1.0000000e+00, 1.9566800e+05],
       [7.5900000e+02, 1.0000000e+00, 1.0000000e+00, ...,
0.0000000e+00,
        0.0000000e+00, 8.6938000e+03],
       [5.0100000e+02, 0.0000000e+00, 1.0000000e+00, ...,
1.0000000e+00,
        1.0000000e+00, 4.7847190e+04]])
```

```
ytrain
```

```
array([[1],
       [1],
       [0],
       ...,
       [0],
       [0],
       [1]])
```

```
ytest
```

```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [0],
       [0]])
```