Bank Churn Analysis

Business Problem-

you are given with data of 10000 customers of a bank, the task is to perform analysis and create a report to answer.

```
Why Customer Left the bank ?
```

Analytics Process -

1. Data Exploration, Domain Understanding
2. Data Cleaning
3. Data Analytics
    - Univariate Analytics - exploring every attribute independently
    - Bivariate Analytics - comparing every feature with label
    - Multivariate Analytics - comparing multiple features with label
4. Prepare Report

In [1]:

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

In [3]:

```python
df = pd.read_csv(r"C:\Users\gorav\Desktop\data\Bank_churn_modelling.csv")
df.shape
```

Out[3]:

```
(10000, 14)
```

Data exploration

In [4]:

```
df.head()
```

Out[4]:

	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

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
RowNumber          10000 non-null int64
CustomerId         10000 non-null int64
Surname            10000 non-null object
CreditScore        10000 non-null int64
Geography          10000 non-null object
Gender             10000 non-null object
Age                10000 non-null int64
Tenure             10000 non-null int64
Balance            10000 non-null float64
NumOfProducts      10000 non-null int64
HasCrCard          10000 non-null int64
IsActiveMember     10000 non-null int64
EstimatedSalary    10000 non-null float64
Exited             10000 non-null int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [6]:

```
df.describe()
```

Out[6]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000

In [7]:

```
df.Gender.unique()
```

Out[7]:

```
array(['Female', 'Male'], dtype=object)
```

In [8]:

```
df.Geography.unique()
```

Out[8]:

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

In [10]:

```
# df = df.sample(0.25) (for large number of data we take a sample of that data)
```

Data Cleaning

In [11]:

```
# check for duplicates
df.duplicated().sum()
```

Out[11]:

```
0
```

In [12]:

```
# check for missing values
df.isnull().sum()
```

Out[12]:

```
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 [14]:

```
df.drop(['RowNumber', 'CustomerId', 'Surname'], axis = 1, inplace = True)
```

Data Analytics

Univariate Analytics

In [15]:

```
# scatter plot for numeric
# histogram / countplot for categorical
df.columns
```

Out[15]:

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

In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
CreditScore        10000 non-null int64
Geography          10000 non-null object
Gender             10000 non-null object
Age                10000 non-null int64
Tenure             10000 non-null int64
Balance            10000 non-null float64
NumOfProducts      10000 non-null int64
HasCrCard          10000 non-null int64
IsActiveMember     10000 non-null int64
EstimatedSalary    10000 non-null float64
Exited             10000 non-null int64
dtypes: float64(2), int64(7), object(2)
memory usage: 859.5+ KB
```

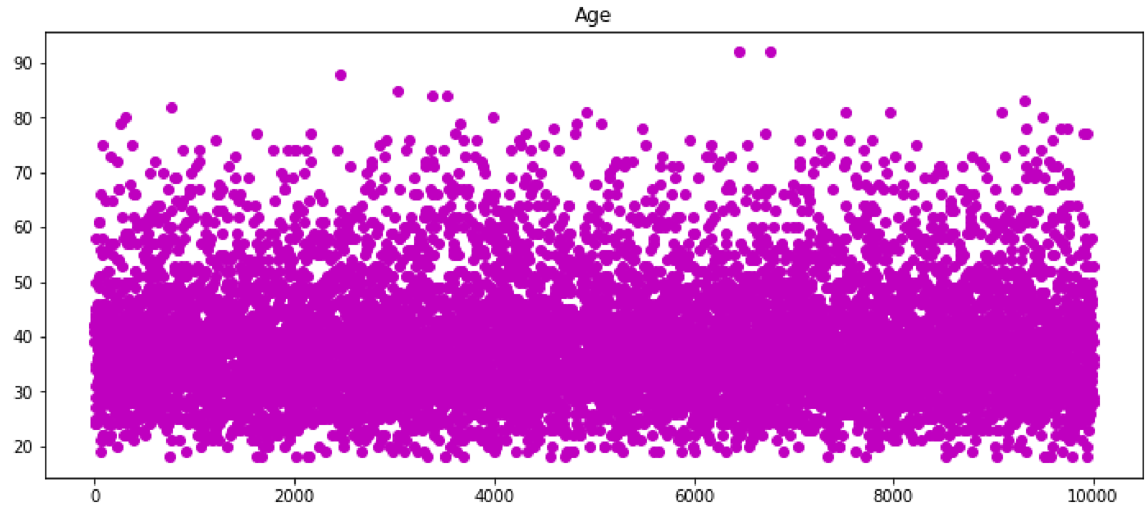
In [17]:

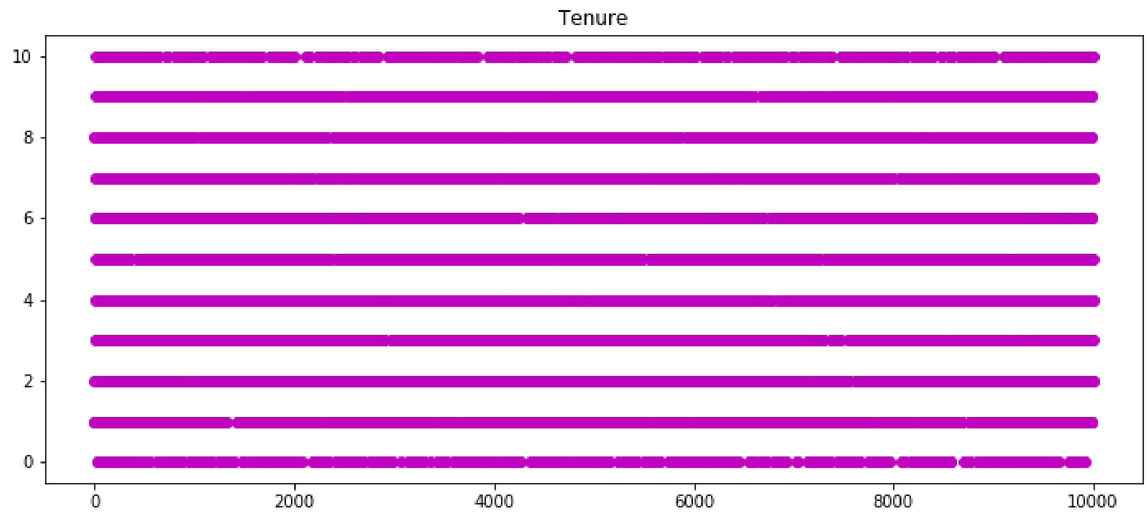
```
numerics = ['CreditScore', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'EstimatedSalary']
cat = ['Geography', 'Gender', 'HasCrCard', 'IsActiveMember', 'Exited']
```

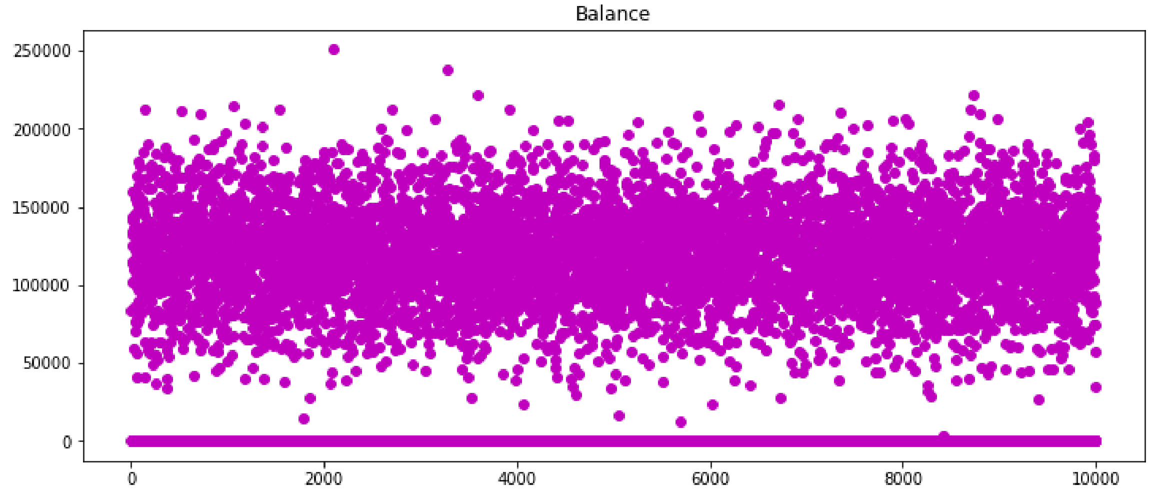
In [45]:

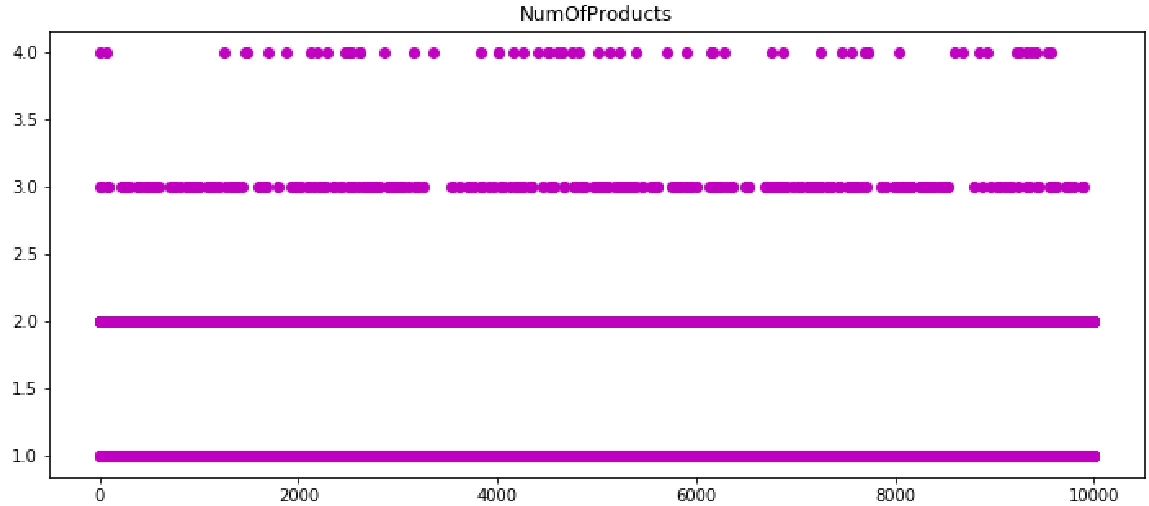
```
# numerics graph

for col in numerics:
    plt.figure(figsize=(12, 5))
    plt.scatter(np.arange(10000), df[col], c = 'm')
    plt.title(col)
    plt.show()
```

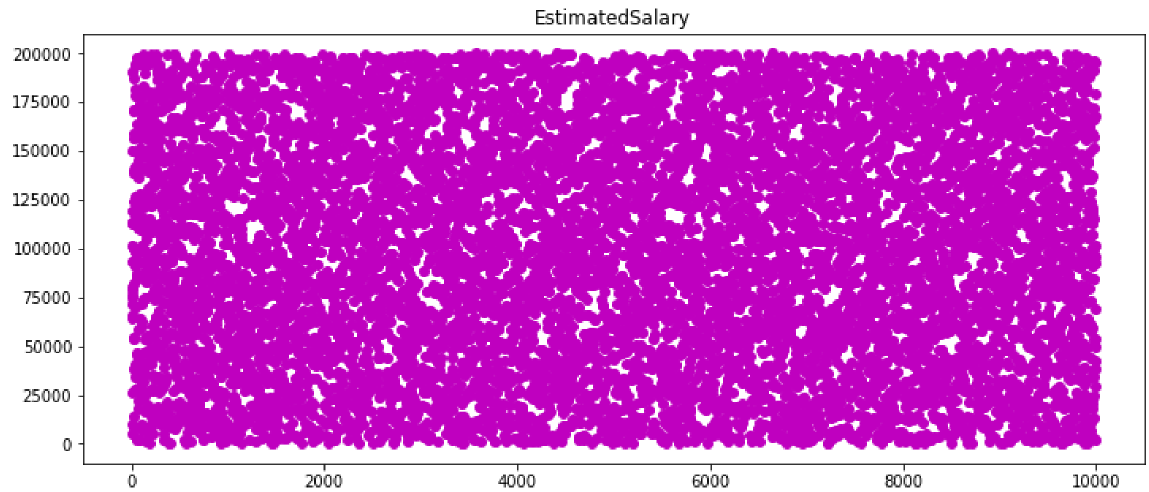

CreditScore
800
700
600
500
400
0
2000
4000
6000
8000
10000


Age
90
80
70
60
50
40
30
20
0
2000
4000
6000
8000
10000


Tenure
10
8
6
4
2
0
0
2000
4000
6000
8000
10000

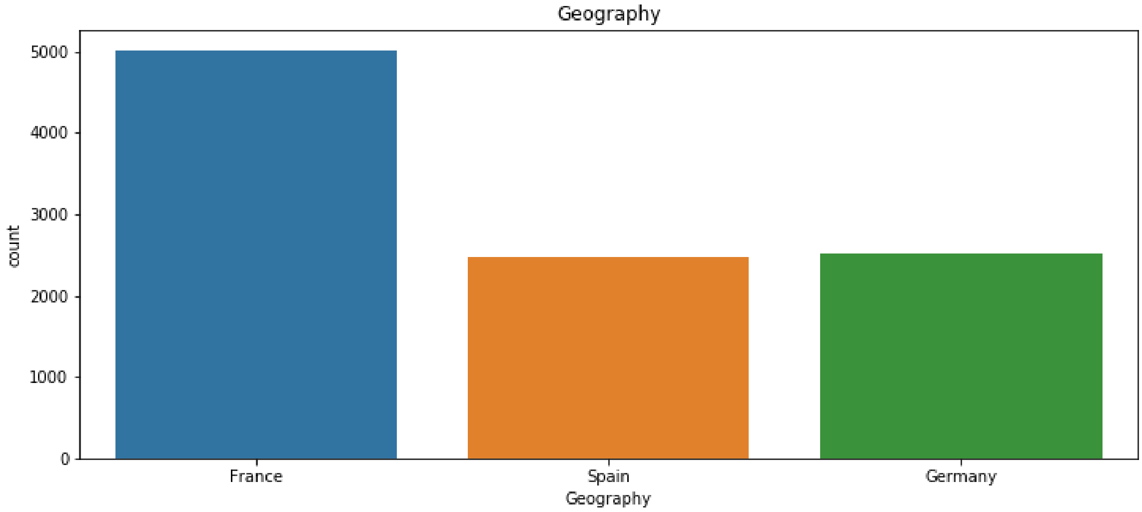

Balance
250000
200000
150000
100000
50000
0
0
2000
4000
6000
8000
10000

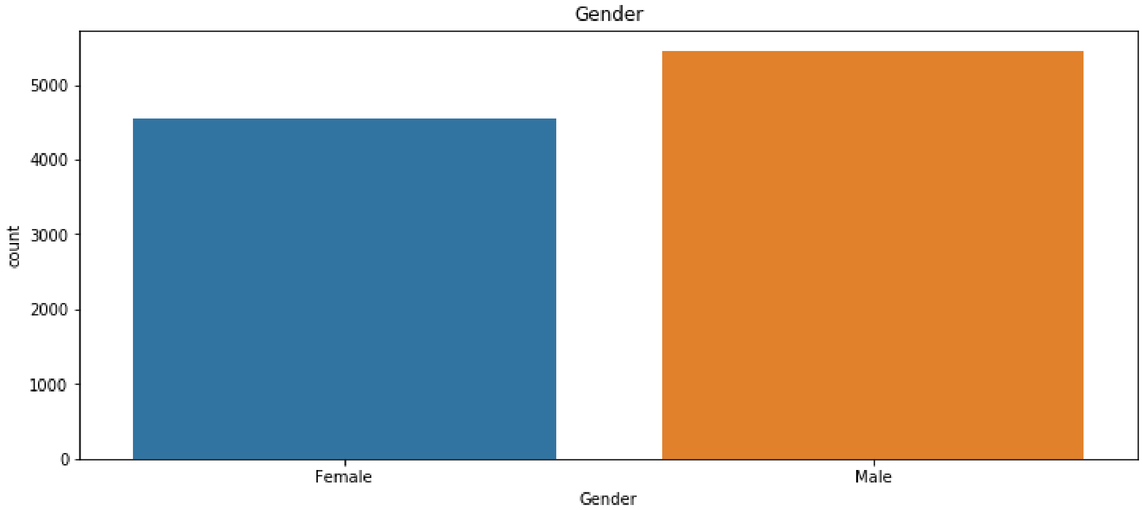

NumOfProducts
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
2000
4000
6000
8000
10000

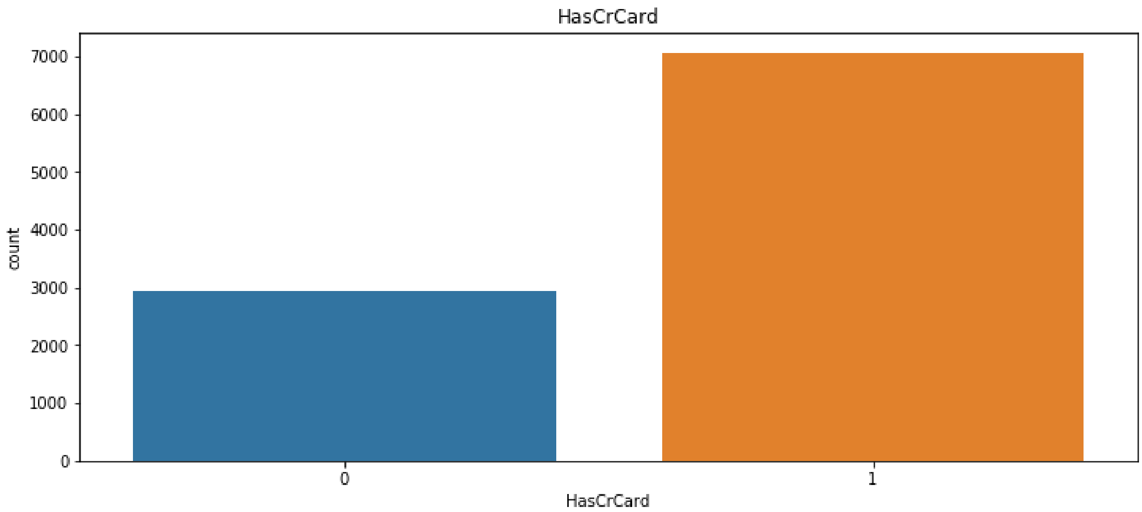

EstimatedSalary
200000
175000
150000
125000
100000
75000
50000
25000
0
0
2000
4000
6000
8000
10000

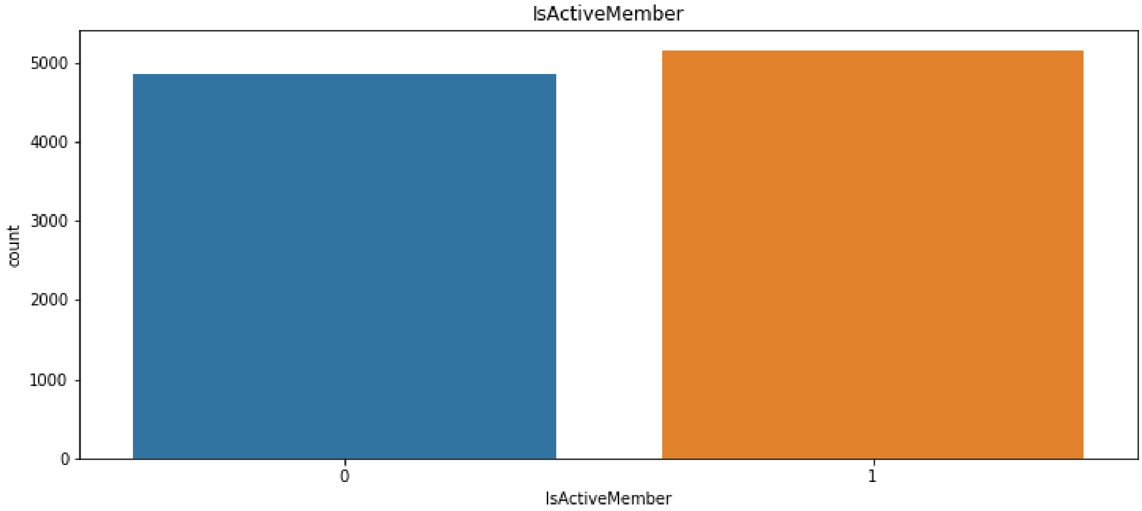
In [46]:

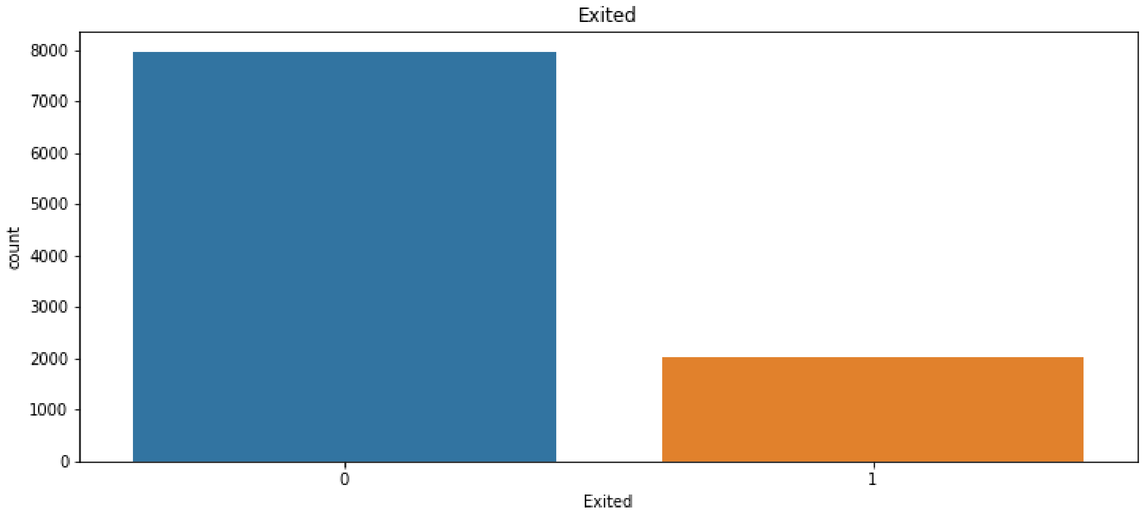
```
# categorical graph

for col in cat:
    plt.figure(figsize=(12, 5))
    sns.countplot(df[col])
    plt.title(col)
    plt.show()
```


Geography
5000
4000
3000
2000
1000
0
count
France
Spain
Germany
Geography


Gender
5000
4000
3000
2000
1000
0
count
Female
Male
Gender


HasCrCard
7000
6000
5000
4000
3000
2000
1000
0
count
0
1
HasCrCard

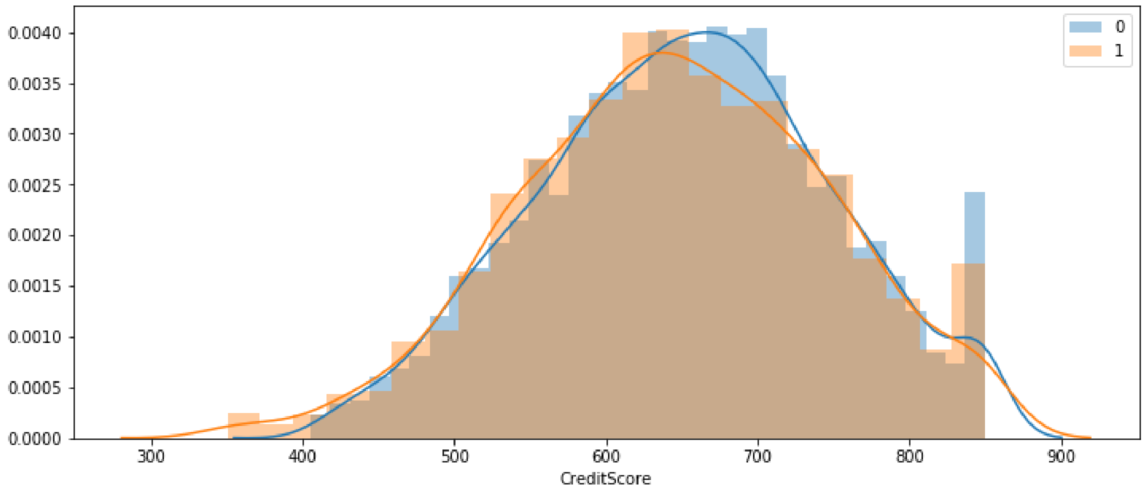




Bivariate Analytics

In [47]:

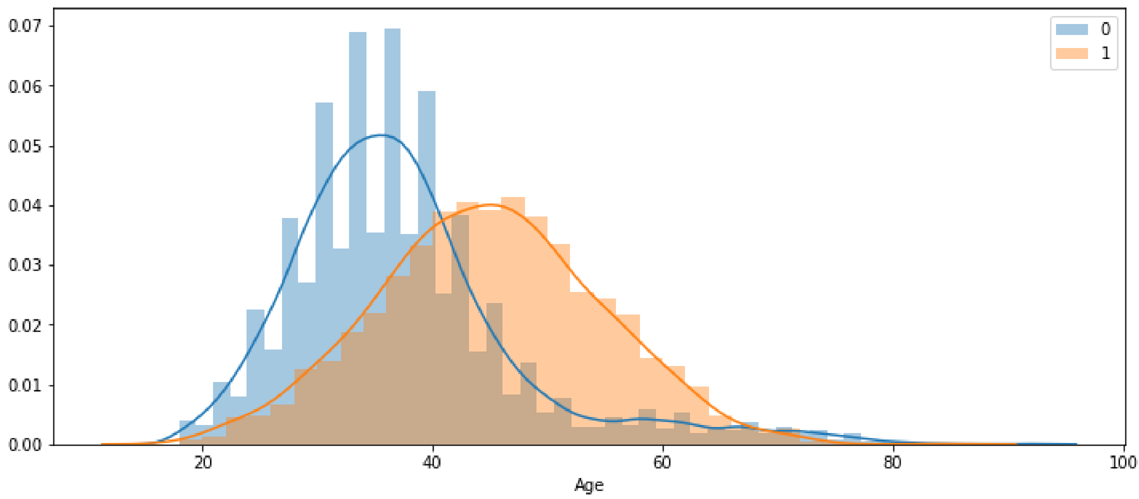
```
# numerical vs categorical
# Probablity density distribution
plt.figure(figsize=(12, 5))
sns.distplot(df.CreditScore[df.Exited==0])
sns.distplot(df.CreditScore[df.Exited==1])
plt.legend(['0', '1'])
plt.show()
```



NOTE: The probablity density plot for creditscore of customers leaving and staying in the bank is almost overlapping , which means chance of customers leaving and staying at every value of credit score is almost same. Hence CreditScore doesn't have sufficient information to say why customers left the bank.

In [48]:

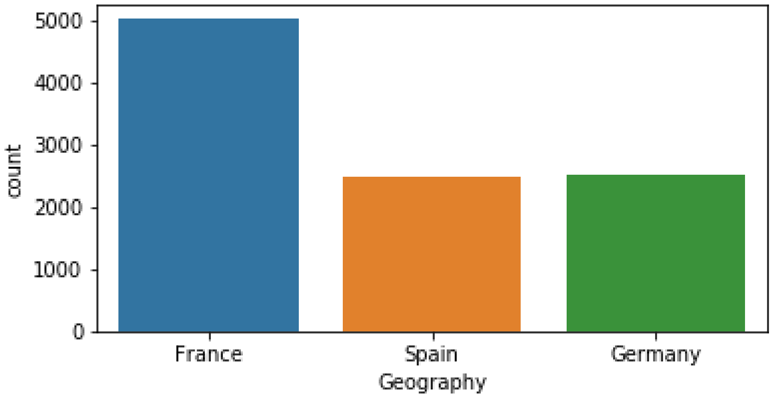
```
# numerical vs categorical
# Probablity density distribution
plt.figure(figsize=(12, 5))
sns.distplot(df.Age[df.Exited==0])
sns.distplot(df.Age[df.Exited==1])
plt.legend(['0', '1'])
plt.show()
```

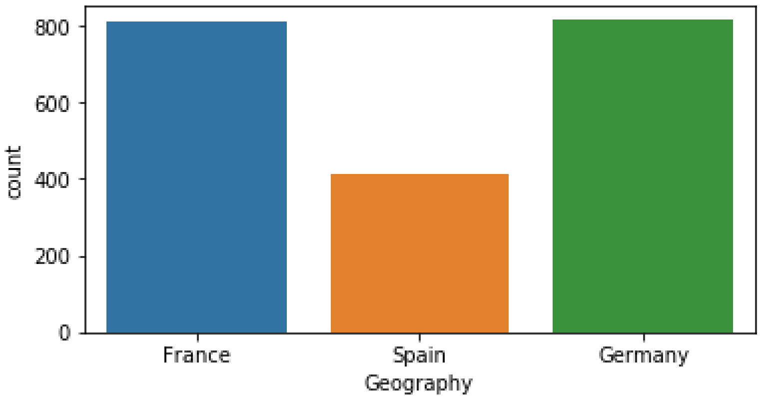


NOTE: For young age generally less than 40, there is high density of customers who did not leave, where as for old age customers generally higher than 40 there is high density of customers who left. Old age customers are having high dropout rate compare to young customers.

In [53]:

```
# categorical vs categorical
# Geography vs Exited
plt.figure(figsize=(6, 3))
sns.countplot(df['Geography'])
plt.show()
plt.figure(figsize=(6, 3))
sns.countplot(df['Geography'][df.Exited==1])
plt.show()
```

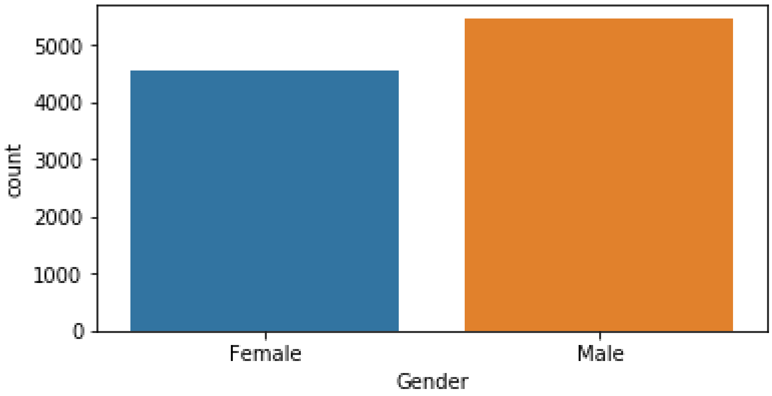


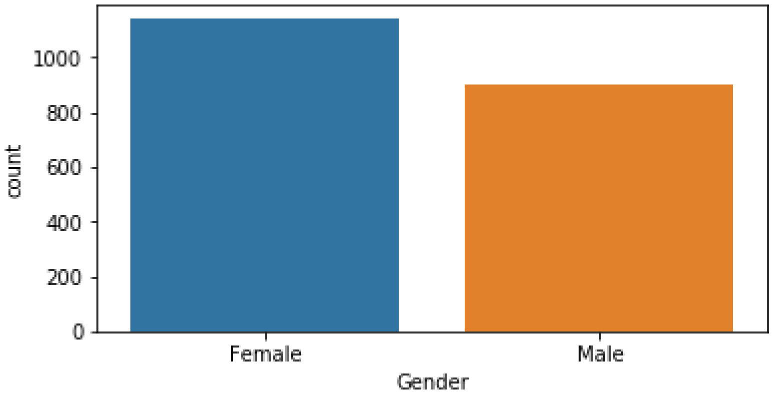


NOTE: Germany has higher dropout rate compared to other two countries

In [52]:

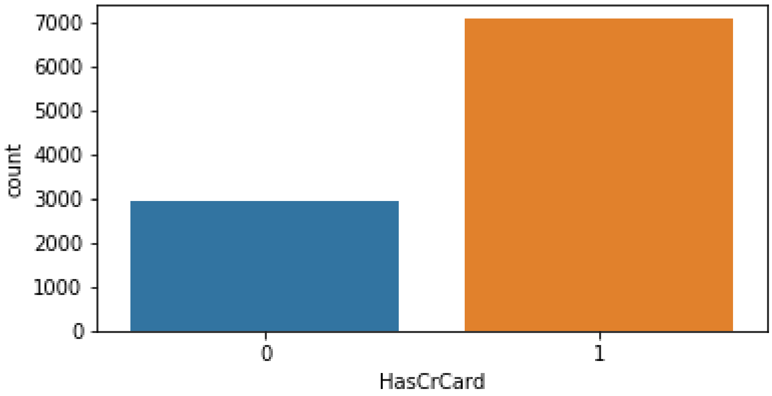
```
# Gender vs Exited
plt.figure(figsize=(6, 3))
sns.countplot(df['Gender'])
plt.show()
plt.figure(figsize=(6, 3))
sns.countplot(df['Gender'][df.Exited==1])
plt.show()
```

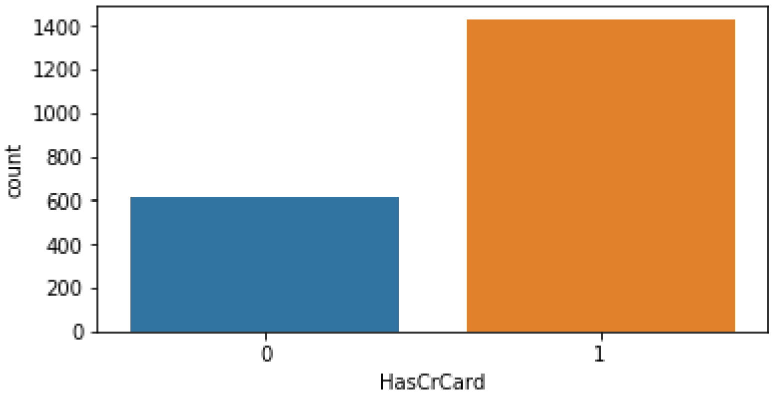




In [54]:

```
# HasCreditCard vs Exited
plt.figure(figsize=(6, 3))
sns.countplot(df['HasCrCard'])
plt.show()
plt.figure(figsize=(6, 3))
sns.countplot(df['HasCrCard'][df.Exited==1])
plt.show()
```



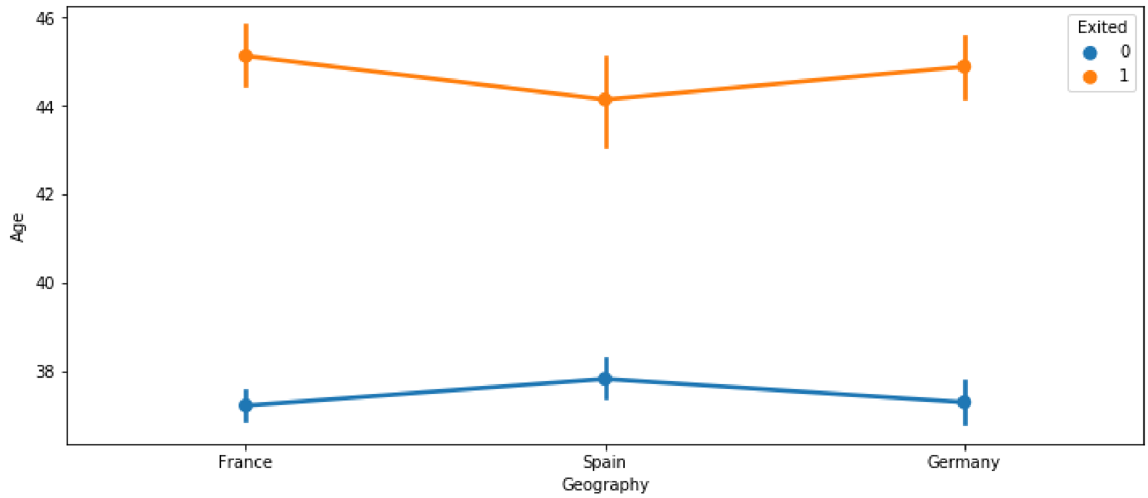


NOTE: Having Cedit card is not impacting the exiting of customer.

Multivariate Analysis

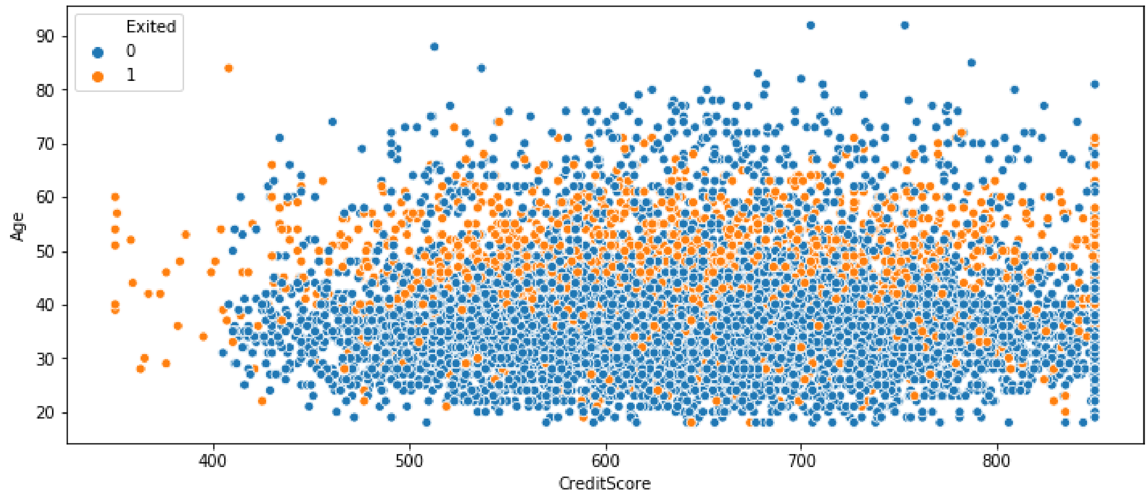
In [57]:

```
# numeric vs categorical  vs categorical - pointplot
# Age vs Geography vs Exited
plt.figure(figsize=(12, 5))
sns.pointplot(x = 'Geography', y = 'Age', hue = 'Exited', data = df)
# dot  = mean
# line = standard deviation
plt.show()
```



In [62]:

```
# numeric vs numeric vs categorical - scatter plot
# Age vs CreditScore vs Exited
plt.figure(figsize=(12, 5))
sns.scatterplot(x = 'CreditScore', y = 'Age', hue = 'Exited', data = df)
plt.show()
```

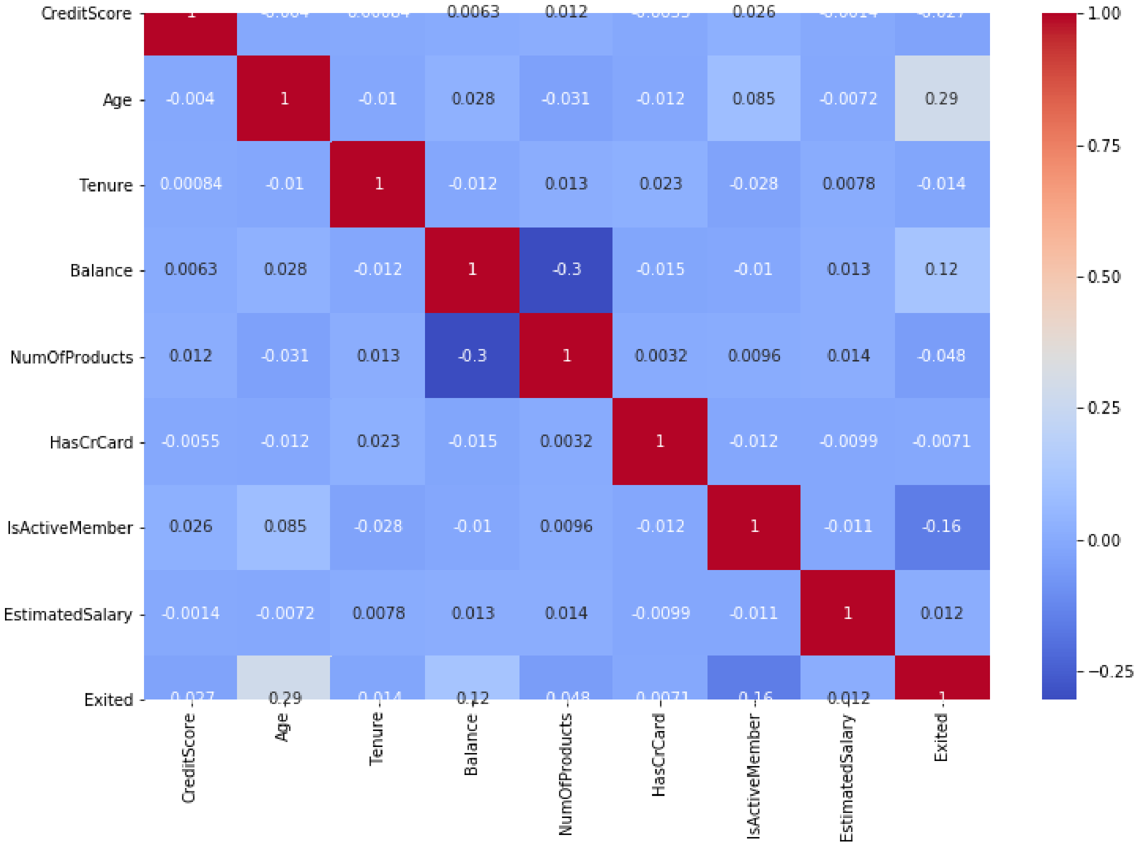


In [64]:

```
# plt.sctter(x = df.CreditScore, y = df.Age, c = df.Exited)
```

In [69]:

```
cor = df.corr() # clculating correlation matrix
# plotting correlation using heatmap
# cor>0.5 v.good
# cor<0.5 and >0.1 good
# cor<-0.5 v.good
# cor>-0.5 and <-0.1 good
# -0.1 to +0.1 ~0 bad
plt.figure(figsize=(12, 8))
sns.heatmap(cor, annot = True, cmap = 'coolwarm')
plt.show()
```



In [ ]: