In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [19]:
```
df = pd.read_csv(r"C:\Users\Meharaj\Churn_Modelling.csv")
```

In [21]:
```
df.head()
```

Out[21]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	Nu
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	

In [29]:
```
df.tail()
```

Out[29]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61
9997	9998	15584532	Liu	709	France	Female	36	7	0.00
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79

In [31]:
```
df.shape
```

Out[31]:
```
(10000, 14)
```

In [33]:
```
df.min()
```

Out[33]:
```
RowNumber              1
CustomerId          15565701
Surname               Abazu
CreditScore             350
Geography            France
Gender               Female
Age                      18
Tenure                    0
Balance                 0.0
NumOfProducts             1
HasCrCard                 0
IsActiveMember            0
EstimatedSalary       11.58
Exited                    0
dtype: object
```

In [35]:
```
df.max()
```

Out[35]:
```
RowNumber              10000
CustomerId          15815690
Surname               Zuyeva
CreditScore              850
Geography              Spain
Gender                  Male
Age                       92
Tenure                    10
Balance           250898.09
NumOfProducts             4
HasCrCard                 1
IsActiveMember            1
EstimatedSalary    199992.48
Exited                    1
dtype: object
```

In [37]:
```
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
```

In [39]:
```
df.describe()
```

Out[39]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfP
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4

In [41]:
```
df.columns
```

Out[41]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [43]:
```
df.nunique()
```

Out[43]:
```
RowNumber          10000
CustomerId         10000
Surname             2932
CreditScore          460
Geography              3
Gender                 2
Age                   70
Tenure                11
Balance             6382
NumOfProducts          4
HasCrCard              2
IsActiveMember         2
EstimatedSalary     9999
Exited                 2
dtype: int64
```

In [45]:
```
df.isnull().sum()
```

Out[45]:
```
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 [47]:
```
df.duplicated()
```

Out[47]:
```
0       False
1       False
2       False
3       False
4       False
        ...
9995    False
9996    False
9997    False
9998    False
9999    False
Length: 10000, dtype: bool
```

In [49]:
```
df.duplicated().sum()
```

Out[49]:
```
0
```

In [51]:
```
df.drop_duplicates()
```

Out[51]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	France	Female	42	2	0.00
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86
2	3	15619304	Onio	502	France	Female	42	8	159660.80
3	4	15701354	Boni	699	France	Female	39	1	0.00
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82
...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61
9997	9998	15584532	Liu	709	France	Female	36	7	0.00
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79

10000 rows × 14 columns

In [53]:
```
df["IsActiveMember"].value_counts()
```

Out[53]:
```
IsActiveMember
1    5151
0    4849
Name: count, dtype: int64
```

In [55]:
```
df["IsActiveMember"] = df["IsActiveMember"].astype(str)
df["IsActiveMember"]
```

Out[55]:
```
0       1
1       1
2       0
3       0
4       1
       ..
9995    0
9996    1
9997    1
9998    0
9999    0
Name: IsActiveMember, Length: 10000, dtype: object
```

In [57]:
```
df["HasCrCard"].value_counts()
```

Out[57]:
```
HasCrCard
1    7055
0    2945
Name: count, dtype: int64
```

In [59]:
```
df["HasCrCard"] = df["HasCrCard"].astype(str)
df["HasCrCard"]
```

Out[59]:
```
0       1
1       0
2       1
3       0
4       1
       ..
9995    1
9996    1
9997    0
9998    1
9999    1
Name: HasCrCard, Length: 10000, dtype: object
```

In [61]:
```
df["Exited"].value_counts()
```

Out[61]:
```
Exited
0    7963
1    2037
Name: count, dtype: int64
```

In [63]:
```
df["Exited"] = df["Exited"].astype(str)
df["Exited"]
```

Out[63]:
```
0       1
1       0
2       1
3       0
4       0
       ..
9995    0
9996    0
9997    1
9998    1
9999    0
Name: Exited, Length: 10000, dtype: object
```

In [65]:
```
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  object
 11  IsActiveMember   10000 non-null  object
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  object
dtypes: float64(2), int64(6), object(6)
memory usage: 1.1+ MB
```

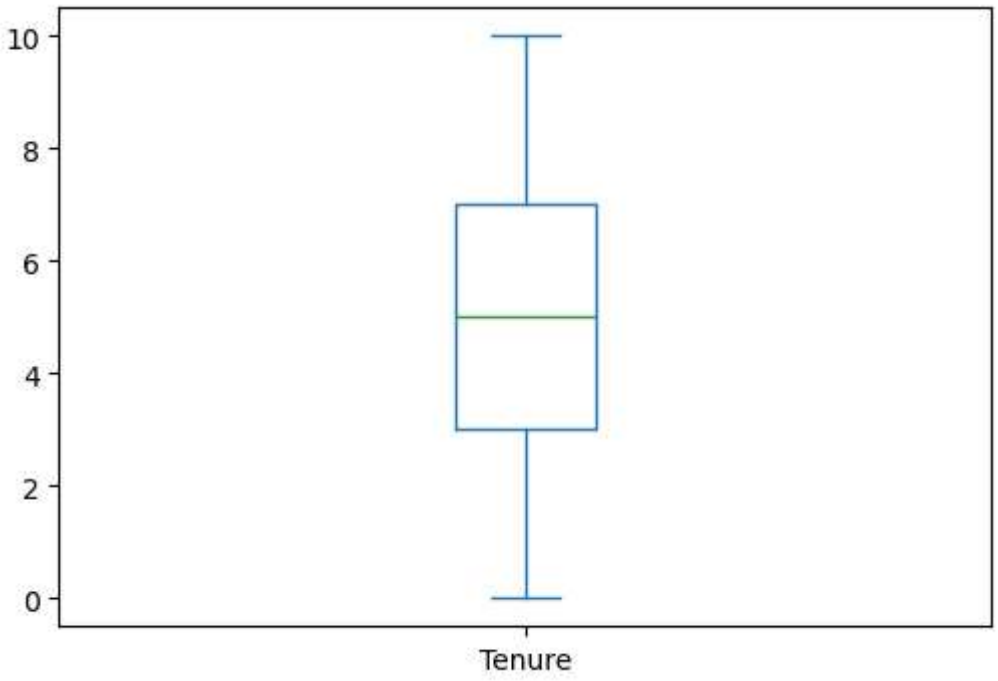
In [67]:
```
df.head()
```

Out[67]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	Nu
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	

In [69]:
```
df["Tenure"].plot(kind = "box",figsize = (6,4))
```

Out[69]: <Axes: >

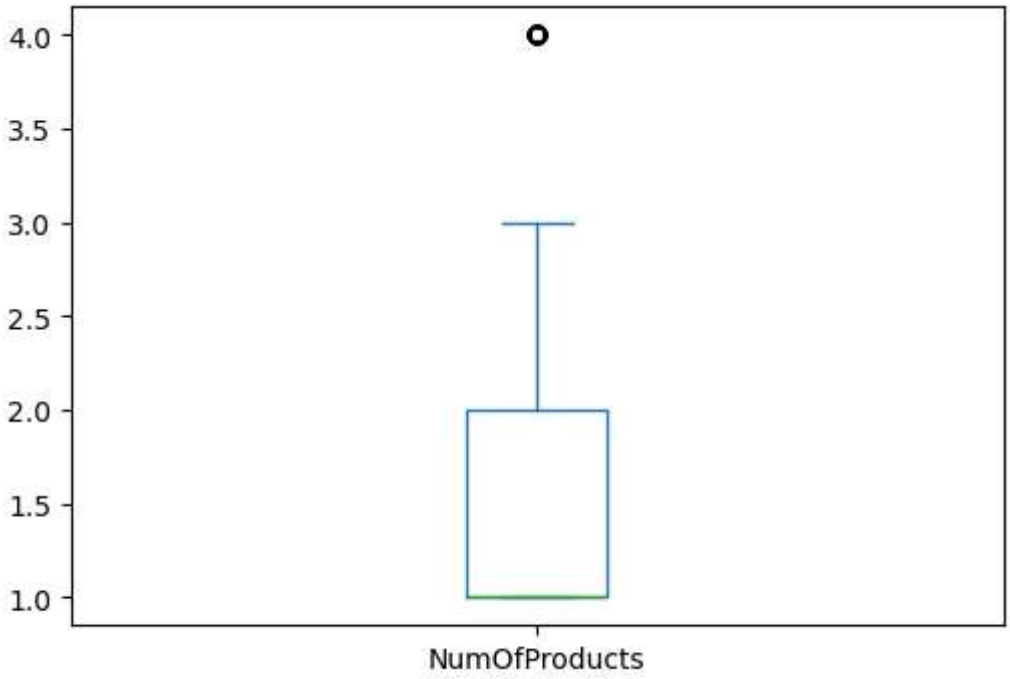


In [27]:
```
df["EstimatedSalary"].plot(kind = "box",figsize = (6,4))
```

Out[27]: <Axes: >

In [28]:
```
df["NumOfProducts"].plot(kind = "box",figsize = (6,4))
```

Out[28]: <Axes: >



univariate analysics - Statistical Non Visual Analysis

In [30]:
```
df.head(5)
```

Out[30]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	Nu
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	

In [71]:
```
df.columns
```

Out[71]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [73]:
```
discrete_df = df.select_dtypes(include=['object'])
numerical_df = df.select_dtypes(include=['int64','float64'])
```

In [75]:
```
numerical_df.columns
```

Out[75]:
```
Index(['RowNumber', 'CustomerId', 'CreditScore', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'EstimatedSalary'],
      dtype='object')
```

In [77]:
```
discrete_df.columns
```

Out[77]:
```
Index(['Surname', 'Geography', 'Gender', 'HasCrCard', 'IsActiveMember',
       'Exited'],
      dtype='object')
```

In [35]:
```
def discrete_univariate_analysis(discrete_data):
    for col_name in discrete_data:
        print('*'*0, col_name, '*'*10)
        print(discrete_data[col_name].agg(['count','nunique','unique']))
        print('Value Counts:\n',discrete_data[col_name].value_counts())
        print()
```

In [36]:
```
def numerical_univariate_analysis(numerical_data):
    for col_name in numerical_data:
        print("*"*10, col_name, "*"*10)
        print(numerical_data[col_name].agg(['min','max','median','std']))
        print()
```

In [37]:
```
numerical_univariate_analysis(numerical_df)
```

```
********** RowNumber **********
min           1.00000
max       10000.00000
median     5000.50000
std        2886.89568
Name: RowNumber, dtype: float64

********** CustomerId **********
min       1.556570e+07
max       1.581569e+07
median    1.569074e+07
std       7.193619e+04
Name: CustomerId, dtype: float64

********** CreditScore **********
min       350.000000
max       850.000000
median    652.000000
std        96.653299
Name: CreditScore, dtype: float64

********** Age **********
min       18.000000
max       92.000000
median    37.000000
std       10.487806
Name: Age, dtype: float64

********** Tenure **********
min        0.000000
max       10.000000
median     5.000000
std        2.892174
Name: Tenure, dtype: float64

********** Balance **********
min            0.000000
max       250898.090000
median     97198.540000
std        62397.405202
Name: Balance, dtype: float64

********** NumOfProducts **********
min       1.000000
max       4.000000
median    1.000000
std       0.581654
Name: NumOfProducts, dtype: float64

********** EstimatedSalary **********
min           11.580000
max       199992.480000
median    100193.915000
std        57510.492818
Name: EstimatedSalary, dtype: float64
```

In [38]:
```
discrete_univariate_analysis(discrete_df)
```

```
 Surname **********
count                                                 10000
nunique                                                2932
unique     [Hargrave, Hill, Onio, Boni, Mitchell, Chu, Ba...
Name: Surname, dtype: object
Value Counts:
 Surname
Smith       32
Scott       29
Martin      29
Walker      28
Brown       26
            ..
Izmailov     1
Bold         1
Bonham       1
Poninski     1
Burbidge     1
Name: count, Length: 2932, dtype: int64

 Geography **********
count                         10000
nunique                           3
unique     [France, Spain, Germany]
Name: Geography, dtype: object
Value Counts:
 Geography
France     5014
Germany    2509
Spain      2477
Name: count, dtype: int64

 Gender **********
count               10000
nunique                 2
unique     [Female, Male]
Name: Gender, dtype: object
Value Counts:
 Gender
Male      5457
Female    4543
Name: count, dtype: int64

 HasCrCard **********
count       10000
nunique         2
unique     [1, 0]
Name: HasCrCard, dtype: object
Value Counts:
 HasCrCard
1    7055
0    2945
Name: count, dtype: int64

 IsActiveMember **********
count       10000
nunique         2
unique     [1, 0]
Name: IsActiveMember, dtype: object
Value Counts:
 IsActiveMember
1    5151
0    4849
Name: count, dtype: int64

 Exited **********
```

```
count       10000
nunique         2
unique     [1, 0]
Name: Exited, dtype: object
Value Counts:
 Exited
0    7963
1    2037
Name: count, dtype: int64
```

univariate analysics - Visual Analysis

In [40]:
```
df.head()
```

Out[40]:

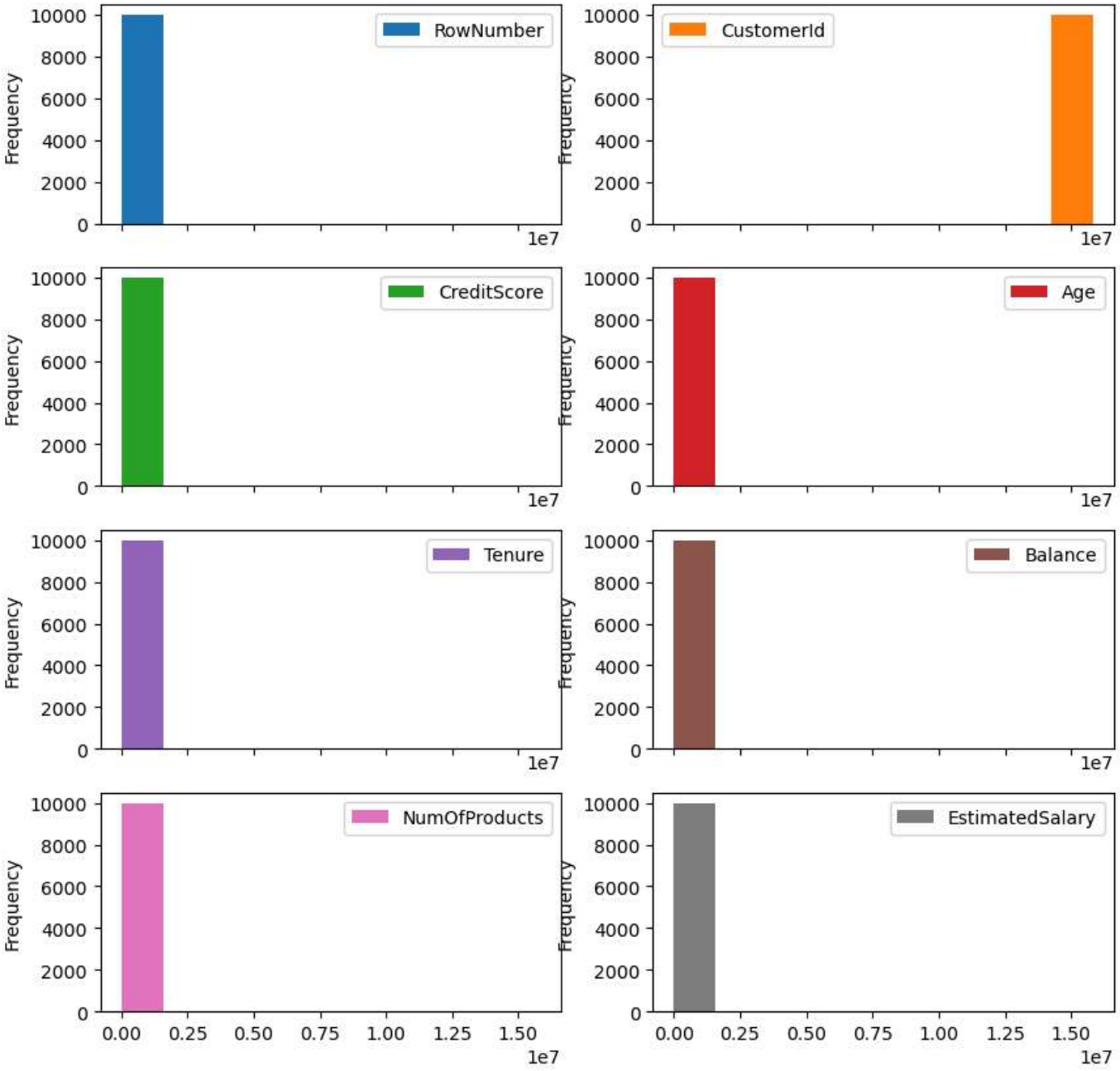
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	Nu
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	

In [41]:
```
df.shape
```

Out[41]:
```
(10000, 14)
```

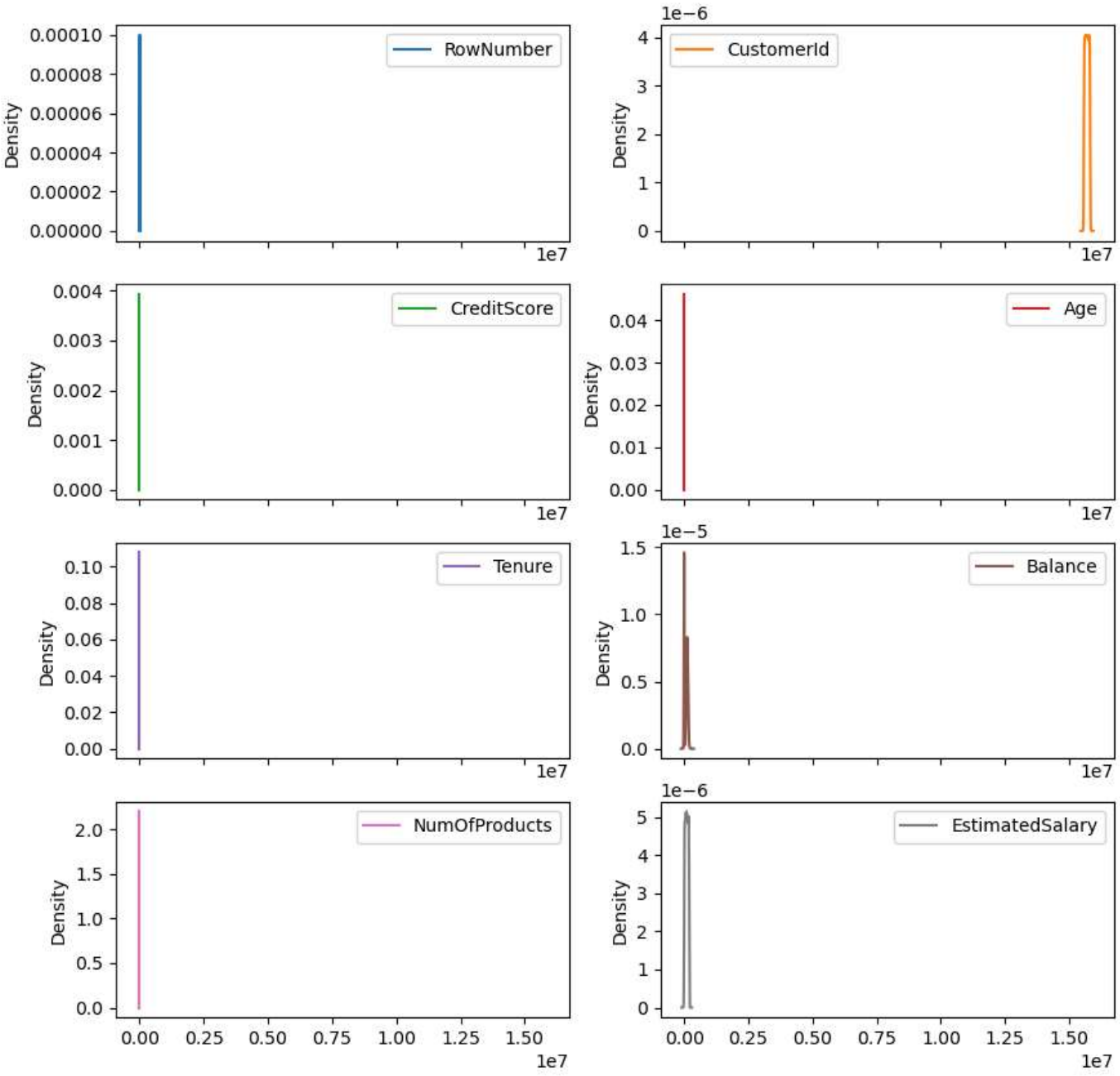
HISTOGRAM PLOT = A histogram plot is a graphical representation of the distribution of numerical data. It groups the data into bins (intervals) and displays the frequency of data points in each bin as rectangular bars.

In [42]:
```
df.plot(kind='hist', subplots= True,layout=(4,2),figsize=(10,10))
plt.show()
```



Kernel Density Estimation plot is a statistical tool used to estimate the probability density function of a continuous random variable. It provides a smoothed version of a histogram and is useful for visualizing the distribution of data.

In [43]:
```
df.plot(kind='kde', subplots= True,layout=(4,2),figsize=(10,10))
plt.show()
```



A box plot (also known as a whisker plot) is a standardized way of displaying the distribution of data based on a five-number summary:

Minimum: The smallest data point excluding outliers.

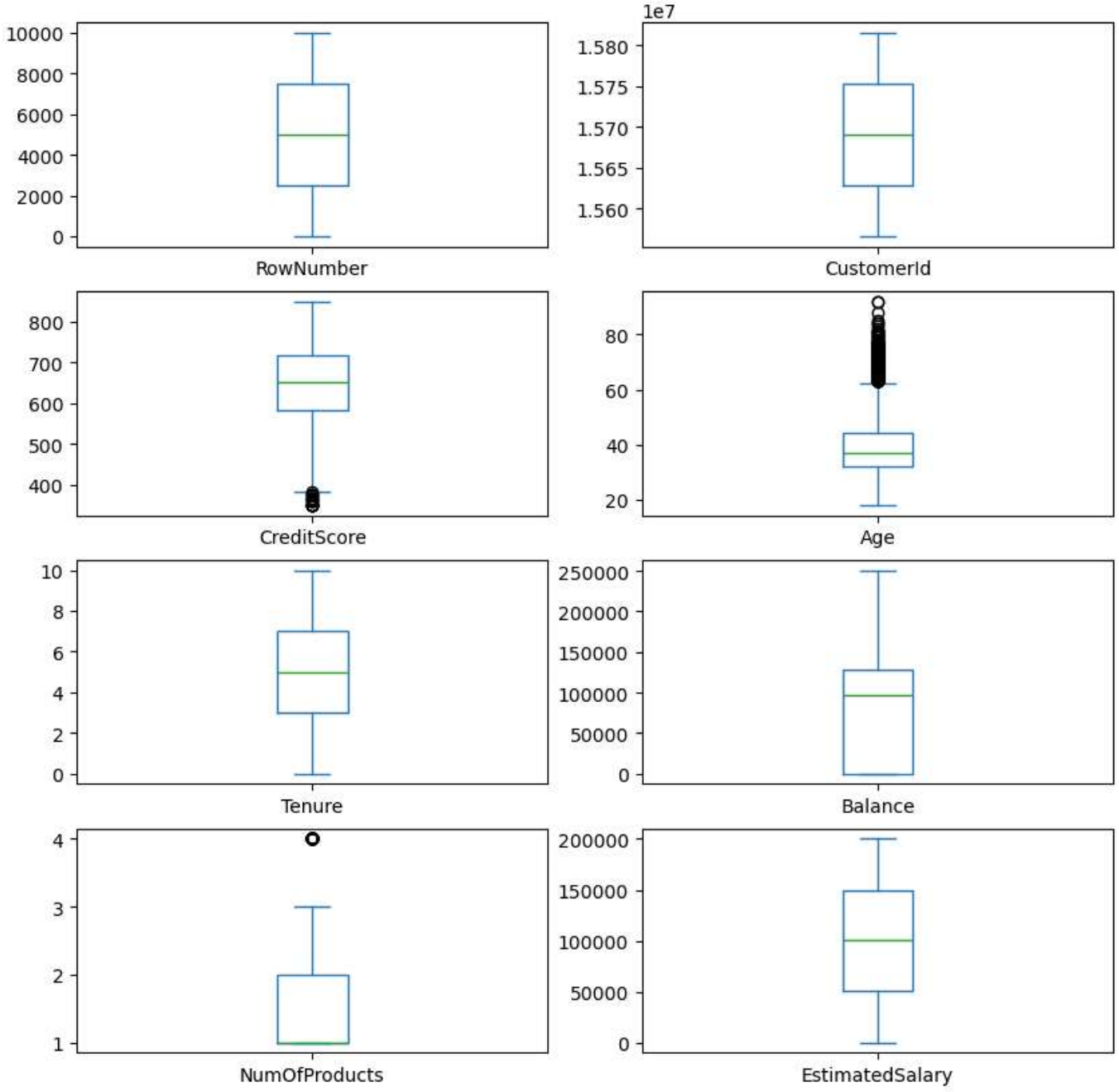
First Quartile (Q1): The median of the lower half of the dataset (25th percentile).

Median (Q2): The middle value of the dataset (50th percentile).

Third Quartile (Q3): The median of the upper half of the dataset (75th percentile).

Maximum: The largest data point excluding outliers.

In [44]:
```
df.plot(kind='box', subplots= True,layout=(4,2),figsize=(10,10))
plt.show()
```



In [45]:
```
df.head()
```

Out[45]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	Nu
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	

In [46]:
```
df.info()
```

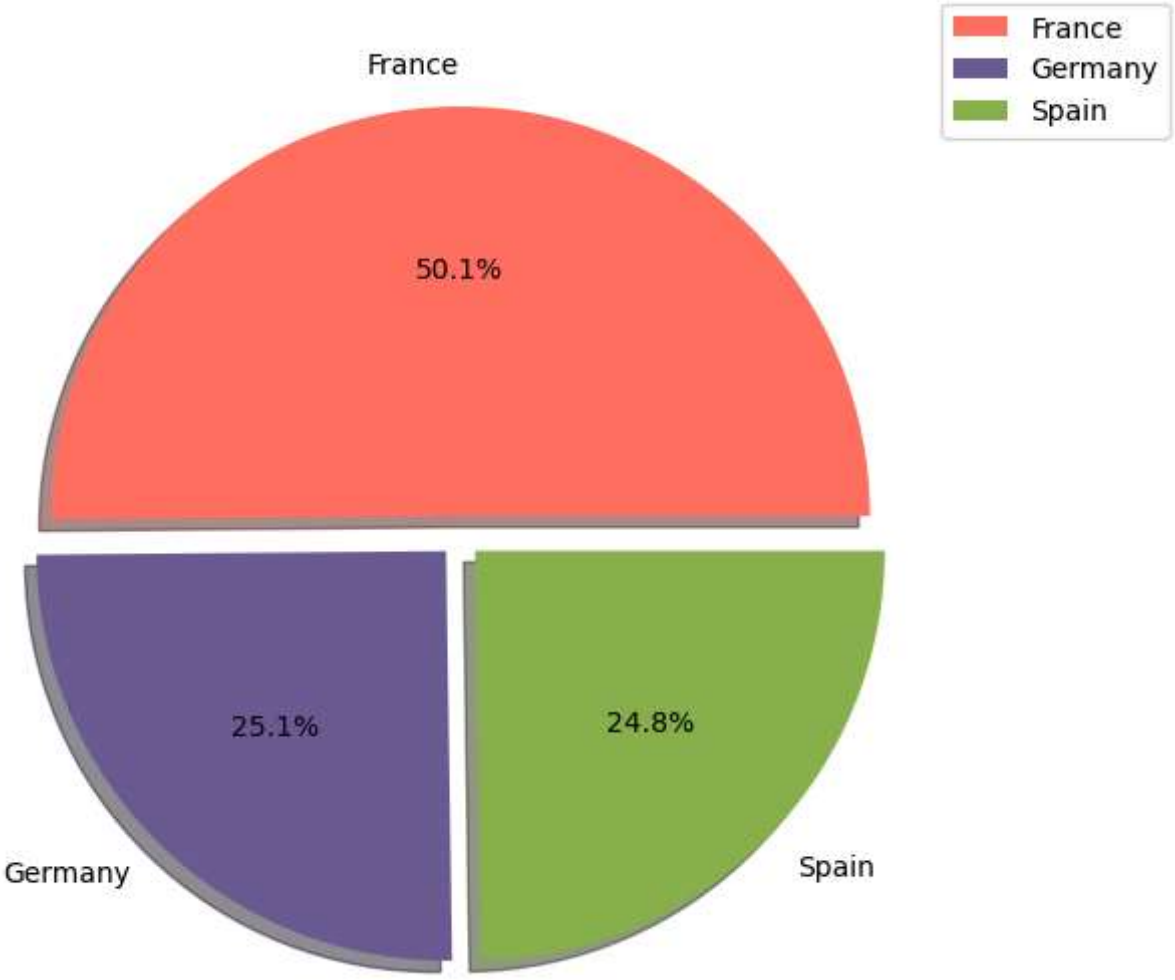
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   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  object 
 11  IsActiveMember   10000 non-null  object 
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  object 
dtypes: float64(2), int64(6), object(6)
memory usage: 1.1+ MB
```

In [47]:
```
df["Geography"].value_counts()
```

Out[47]:
```
Geography
France     5014
Germany    2509
Spain      2477
Name: count, dtype: int64
```

PIE PLOT = A pie plot is a circular statistical graphic divided into slices to illustrate numerical proportions. Each slice's size represents its proportion of the total.

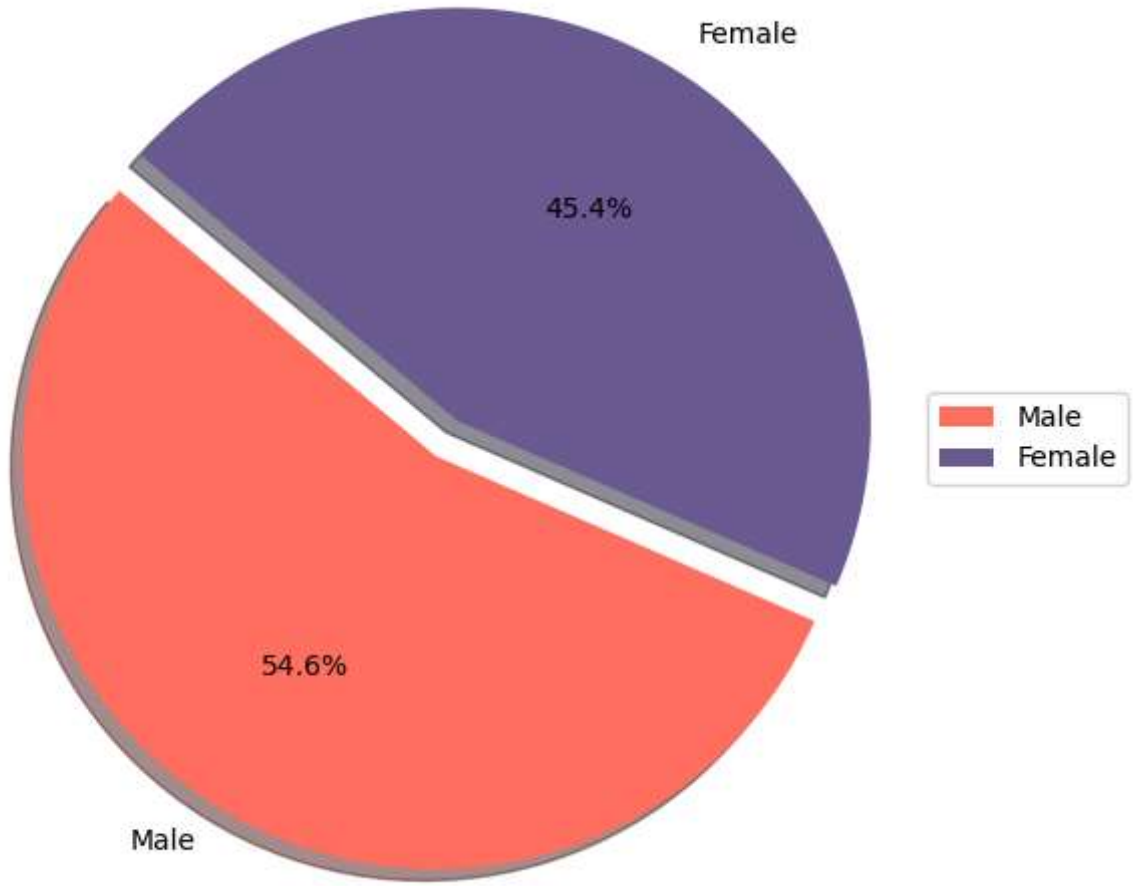
In [50]:
```
Geography = df['Geography'].value_counts()
custom_colors = ['#FF6F61', '#6B5B95', '#88B04B', '#F7CAC9', '#92A8D1']
explode = [0.05] * len(Geography)
plt.figure(figsize=(6, 6))
plt.pie(Geography,labels=Geography.index, autopct='%1.1f%%',
        colors=custom_colors,
        explode=explode,
        shadow=True)
plt.axis('equal')
plt.legend(Geography.index,  loc="center left", bbox_to_anchor=(1, 0.5, 0.5, 1))
plt.show()
```



The orange slice represents the percentage of france (50.1%), the blue slice represents the percentage of germany (25.1%), the green represent the perecentage of spain (24.8%)

In [52]:

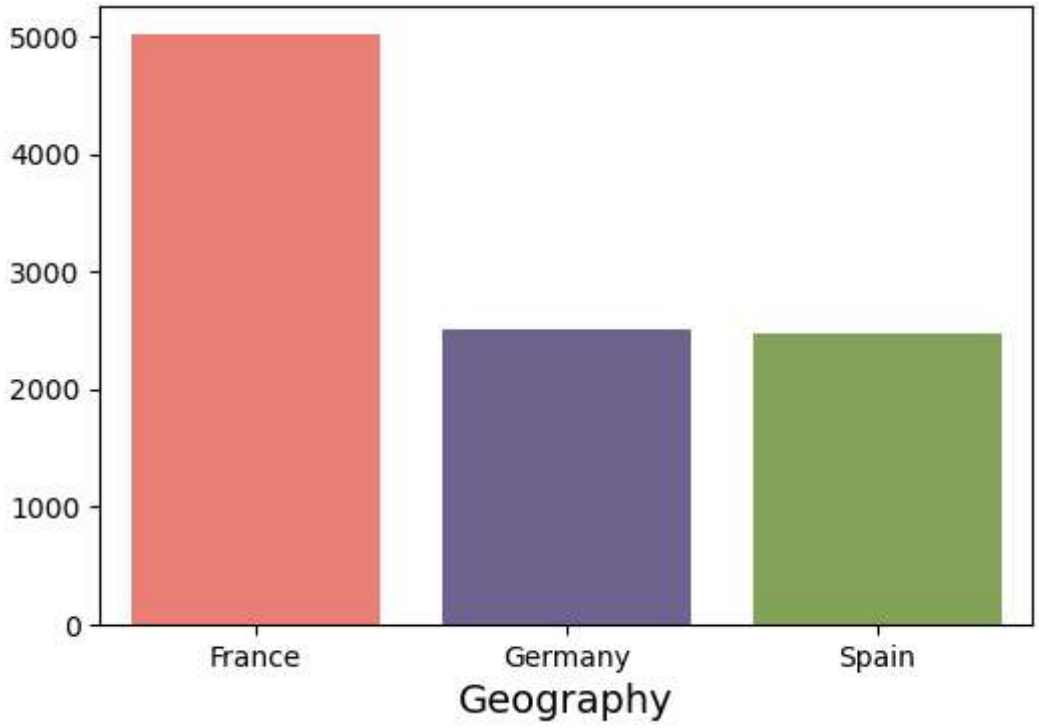
```
Gender = df['Gender'].value_counts()
custom_colors = ['#FF6F61', '#6B5B95', '#88B04B', '#F7CAC9', '#92A8D1']
explode = [0.05] * len(Gender)
plt.figure(figsize=(6, 6))
plt.pie(Gender,
        labels=Gender.index,
        autopct='%1.1f%%',
        startangle=140,
        colors=custom_colors,
        explode=explode,
        shadow=True)
plt.axis('equal')
plt.legend(Gender.index,  loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))
plt.show()
```



The orange slice represents the percentage of male (54.6%), and the blue slice represents the percentage of female (45.4%)

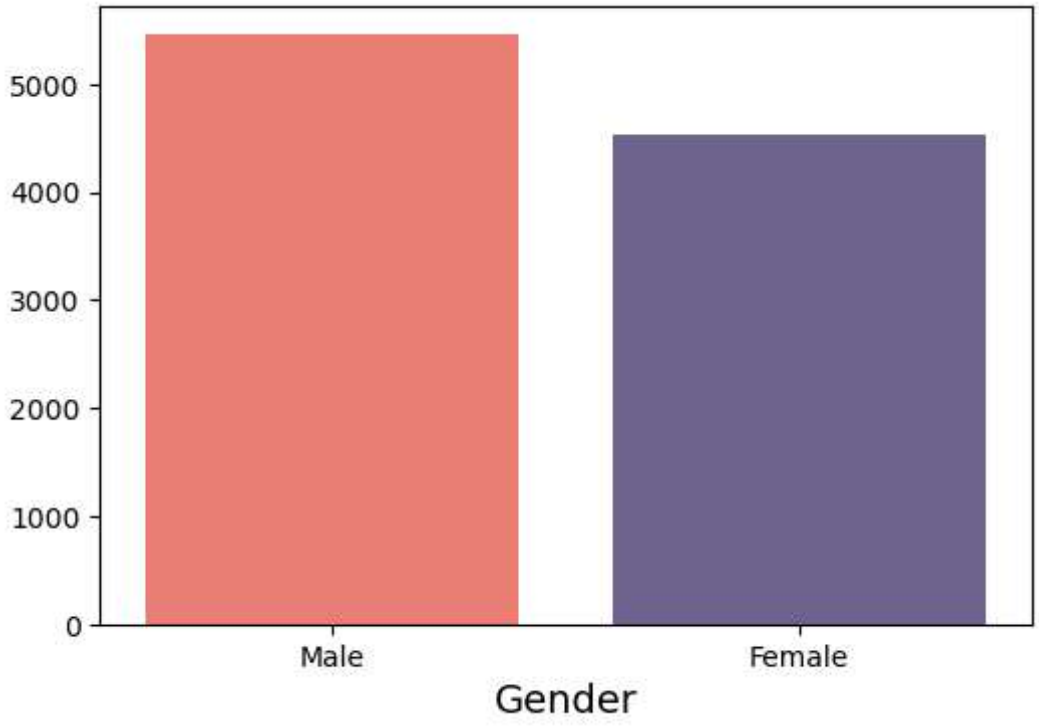
In [54]:
```
Geography = df['Geography'].value_counts()
custom_colors = ['#FF6F61', '#6B5B95', '#88B04B', '#F7CAC9']
plt.figure(figsize=(6, 4))
sns.barplot(x=Geography.index, y=Geography.values, palette=custom_colors)

plt.xlabel('Geography', fontsize=14)
plt.show()
```

```
C:\Users\Meharaj\AppData\Local\Temp\ipykernel_18640\1436398096.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=Geography.index, y=Geography.values, palette=custom_colors)
C:\Users\Meharaj\AppData\Local\Temp\ipykernel_18640\1436398096.py:4: UserWarning: The palette list has more values (4) than needed (3), which may not be intended.
  sns.barplot(x=Geography.index, y=Geography.values, palette=custom_colors)
```



In [55]:
```
Gender = df['Gender'].value_counts()
custom_colors = ['#FF6F61', '#6B5B95', '#88B04B', '#F7CAC9']
plt.figure(figsize=(6, 4))
sns.barplot(x=Gender.index, y=Gender.values, palette=custom_colors)

plt.xlabel('Gender', fontsize=14)
plt.show()
```

```
C:\Users\Meharaj\AppData\Local\Temp\ipykernel_18640\3056353670.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=Gender.index, y=Gender.values, palette=custom_colors)
C:\Users\Meharaj\AppData\Local\Temp\ipykernel_18640\3056353670.py:4: UserWarning: The palette list has more values (4) than needed (2), which may not be intended.
  sns.barplot(x=Gender.index, y=Gender.values, palette=custom_colors)
```

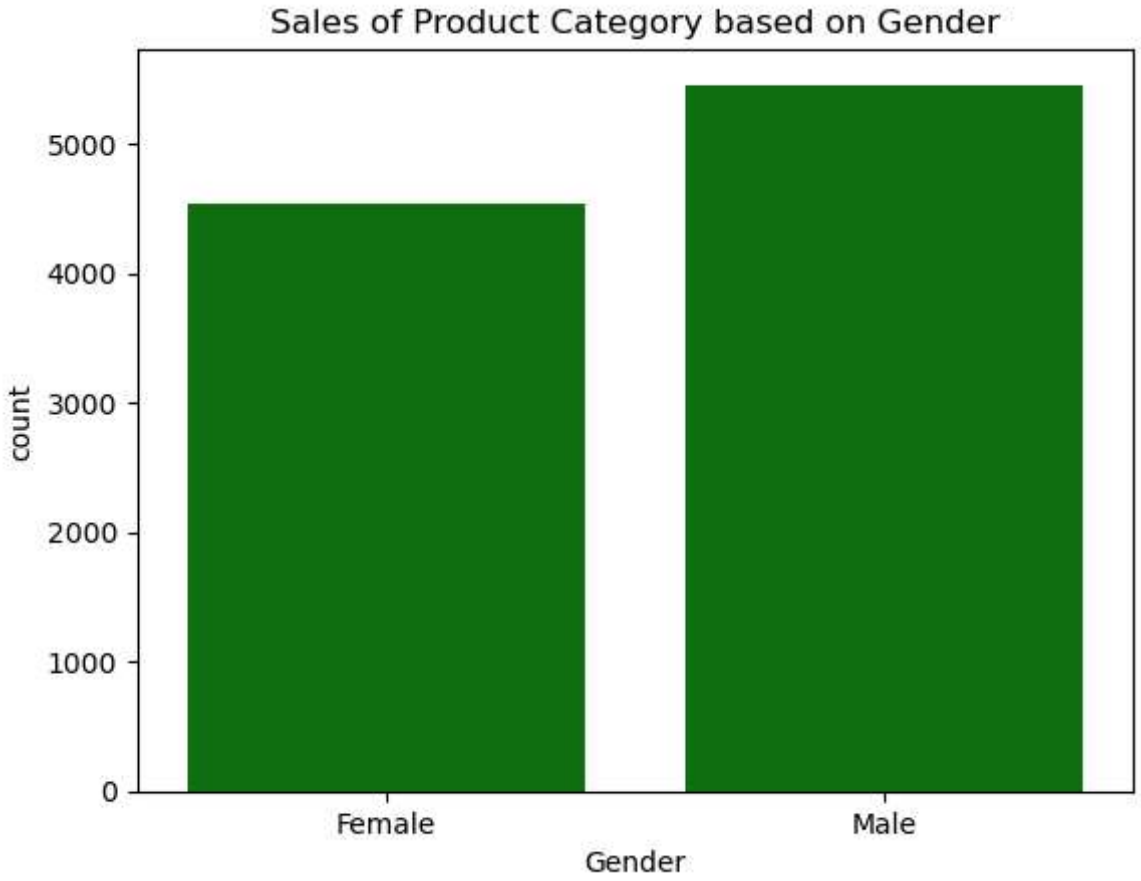


- male popluation has greater than 5000 above as we compare to female there are less

- Female population has in betweeen 4000 and 5000

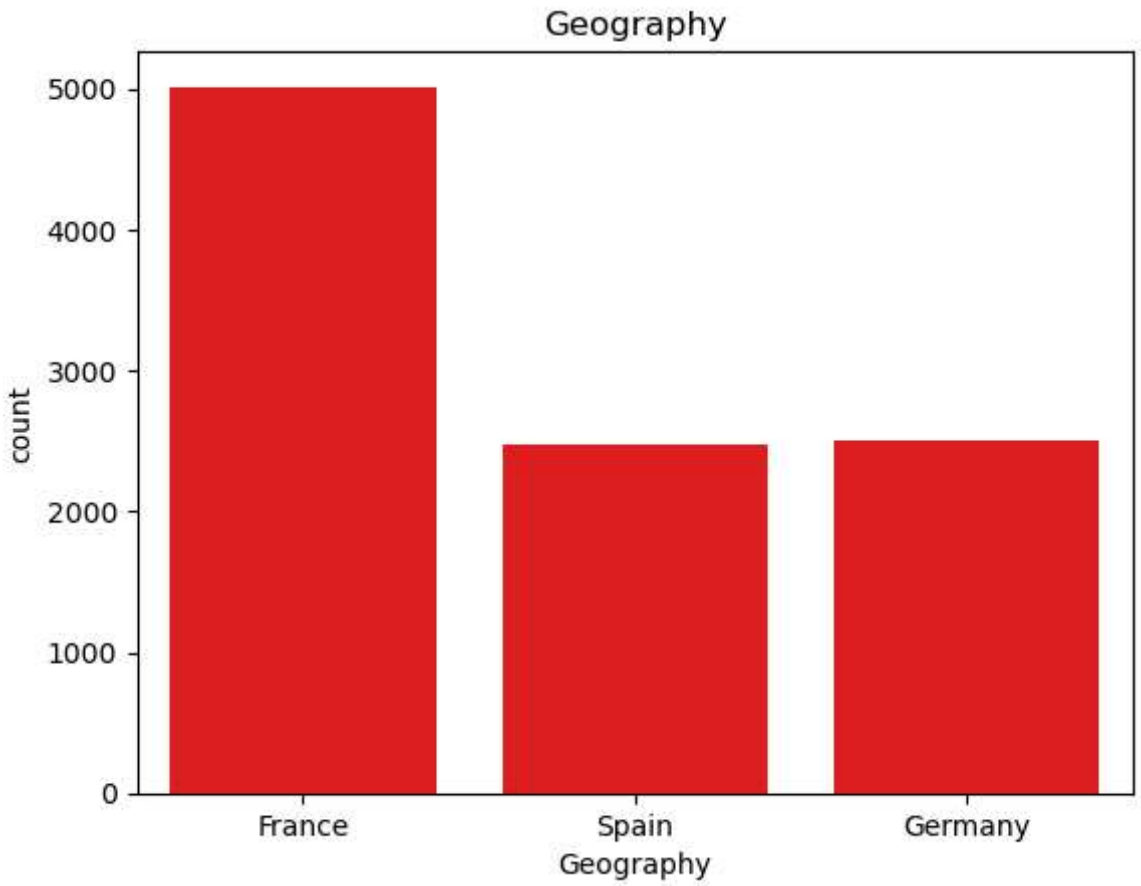
COUNT PLOT = It is a type of bar plot used to visualize the frequency (count) of observations for each category in a categorical variable. It provides a simple and effective way to see the distribution of categorical data.

In [44]:
```
sns.countplot(data = df,x = 'Gender',color = "green")
plt.title('Gender')
plt.show()
```



- There are more male useres are using credit cards
- There are less female users are using credit cards

In [46]:
```
sns.countplot(data = df,x = 'Geography',color = "red")
plt.title('Geography')
plt.show()
```



- There are more users of credit card in france and less users in spain and germany

Bivariate Analaysics

a. Continuous vs Continuous Numerical Data

In [58]:
```
df.head()
```

Out[58]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	Nu
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	

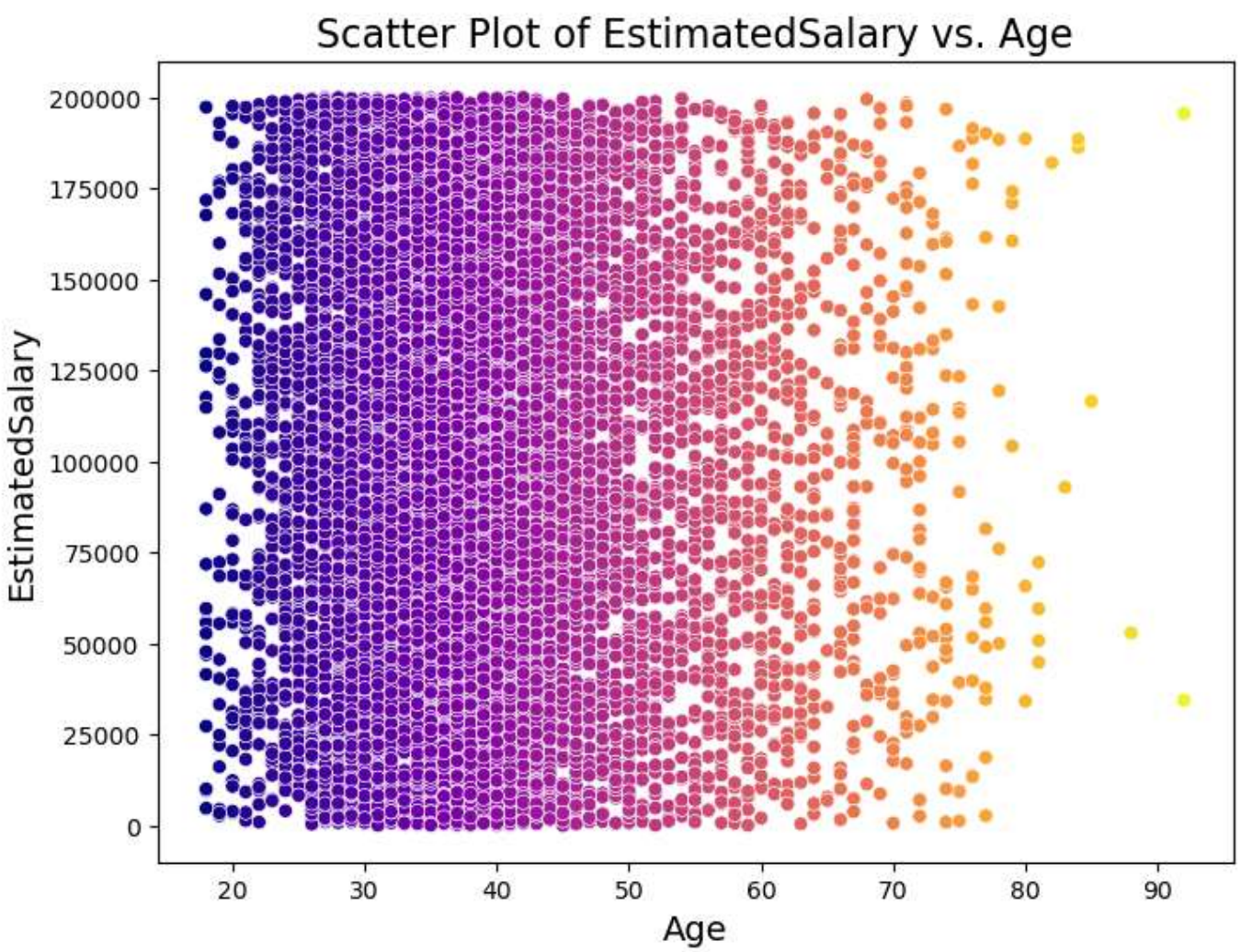
In [59]:
```
df.info()
```

```
<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  object
 11  IsActiveMember   10000 non-null  object
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  object
dtypes: float64(2), int64(6), object(6)
memory usage: 1.1+ MB
```

scatter plot

SCATTER PLOT = A scatter plot is a type of data visualization that displays individual data points on a two-dimensional axis, showing the relationship between two continuous variables. It's useful for identifying trends, correlations, and outliers in the data.

In [61]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Age', y='EstimatedSalary', data=df, hue='Age', palette="plasma", legend=No
plt.title('Scatter Plot of EstimatedSalary vs. Age', fontsize=16)
plt.xlabel('Age', fontsize=14)
plt.ylabel('EstimatedSalary', fontsize=14)
norm = plt.Normalize(df['Age'].min(), df["Age"].max())
sm = plt.cm.ScalarMappable(cmap='plasma', norm=norm)
plt.show()
```

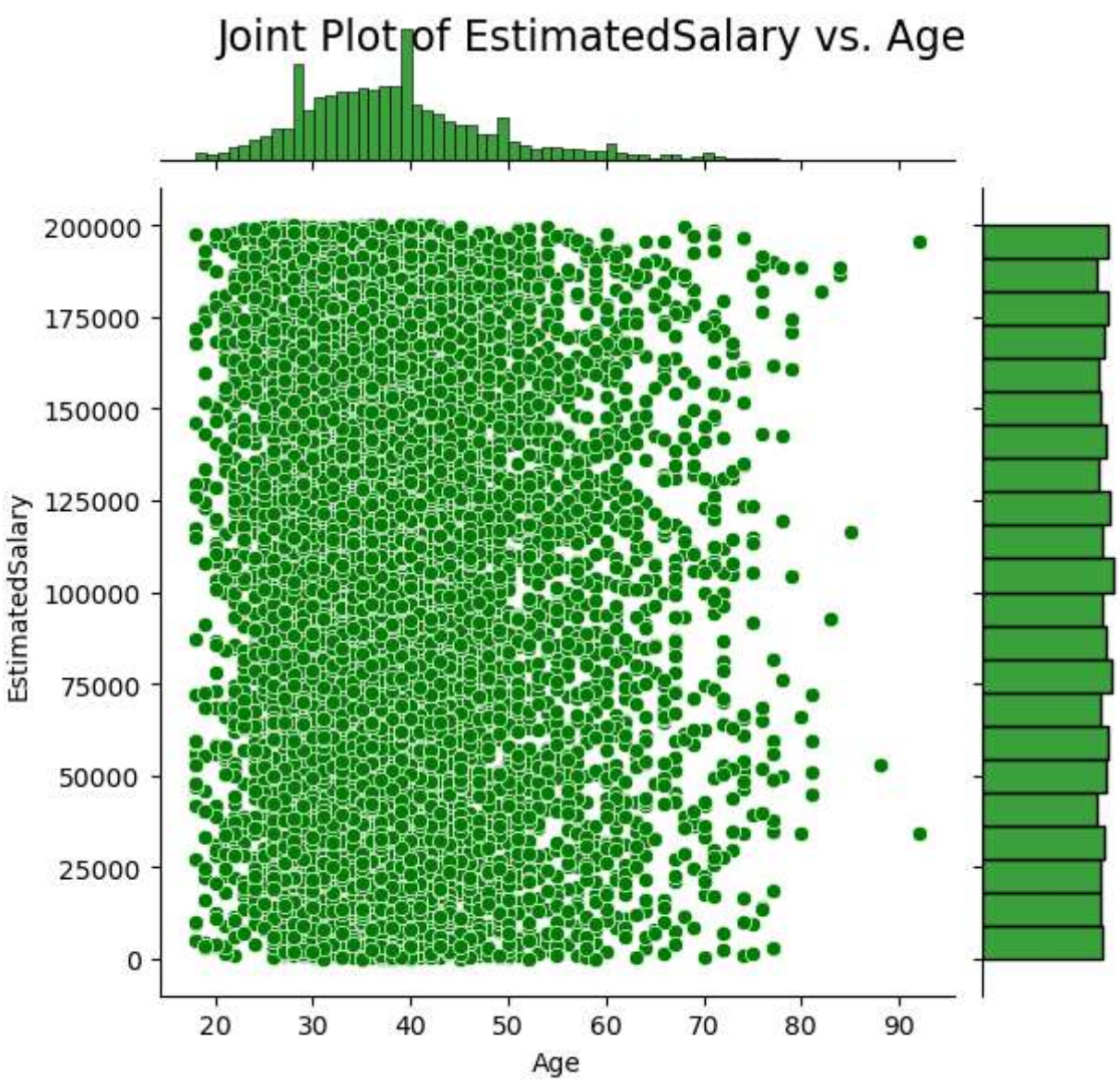


JOINT PLOT : A joint plot is a visualization that combines a scatterplot with marginal histograms or density plots to analyze the relationship between two variables and understand their distributions. It is widely used in data analysis to explore bivariate distributions

Joint plot

In [63]:
```
plt.figure(figsize=(8, 8))
sns.jointplot(x='Age', y='EstimatedSalary', data=df,kind = "scatter", color='green')
plt.suptitle('Joint Plot of EstimatedSalary vs. Age',  fontsize=16)
plt.show()
```

```
<Figure size 800x800 with 0 Axes>
```

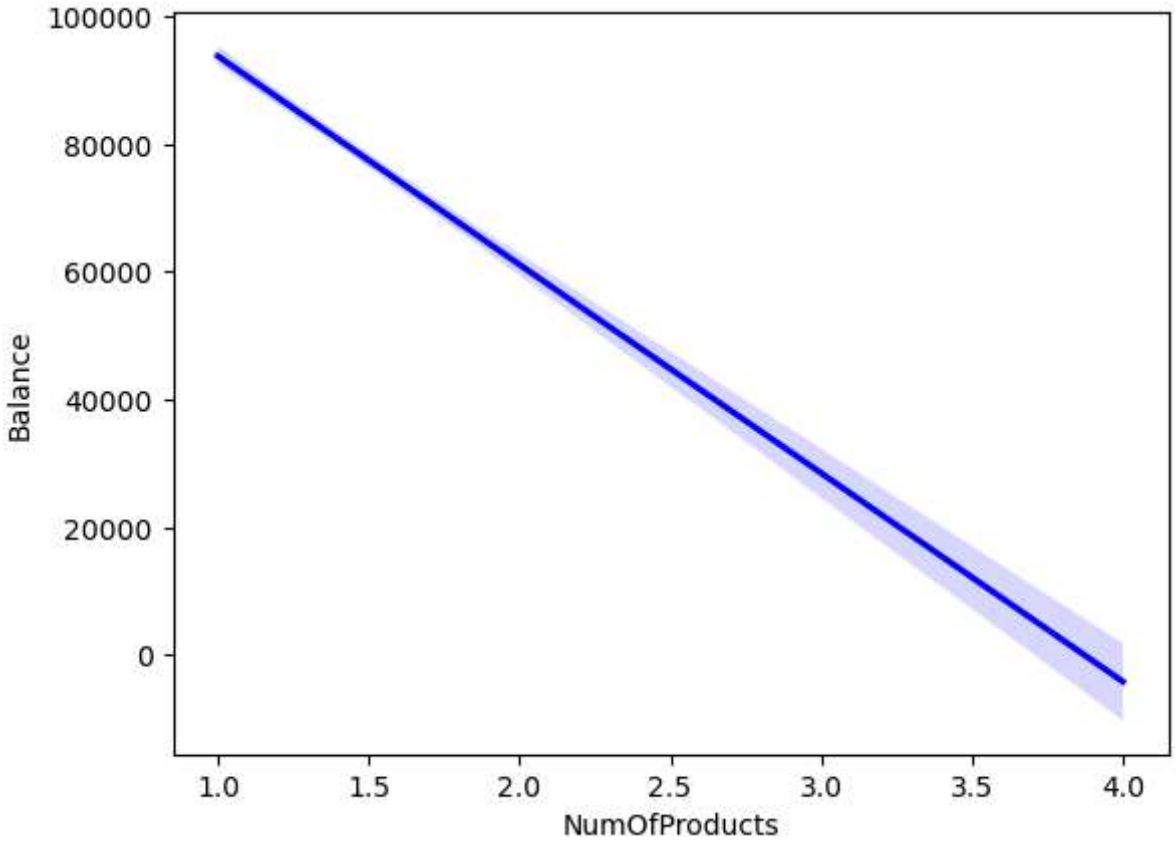


- joint plot has two variates like univariate and bivariate to the estimated salary and age
- The age in between 20-60 most of pepole having credit limit of 2lacs
- the age in between 70-90 less pepole having credit limit of 2lacs and less than 2lacs

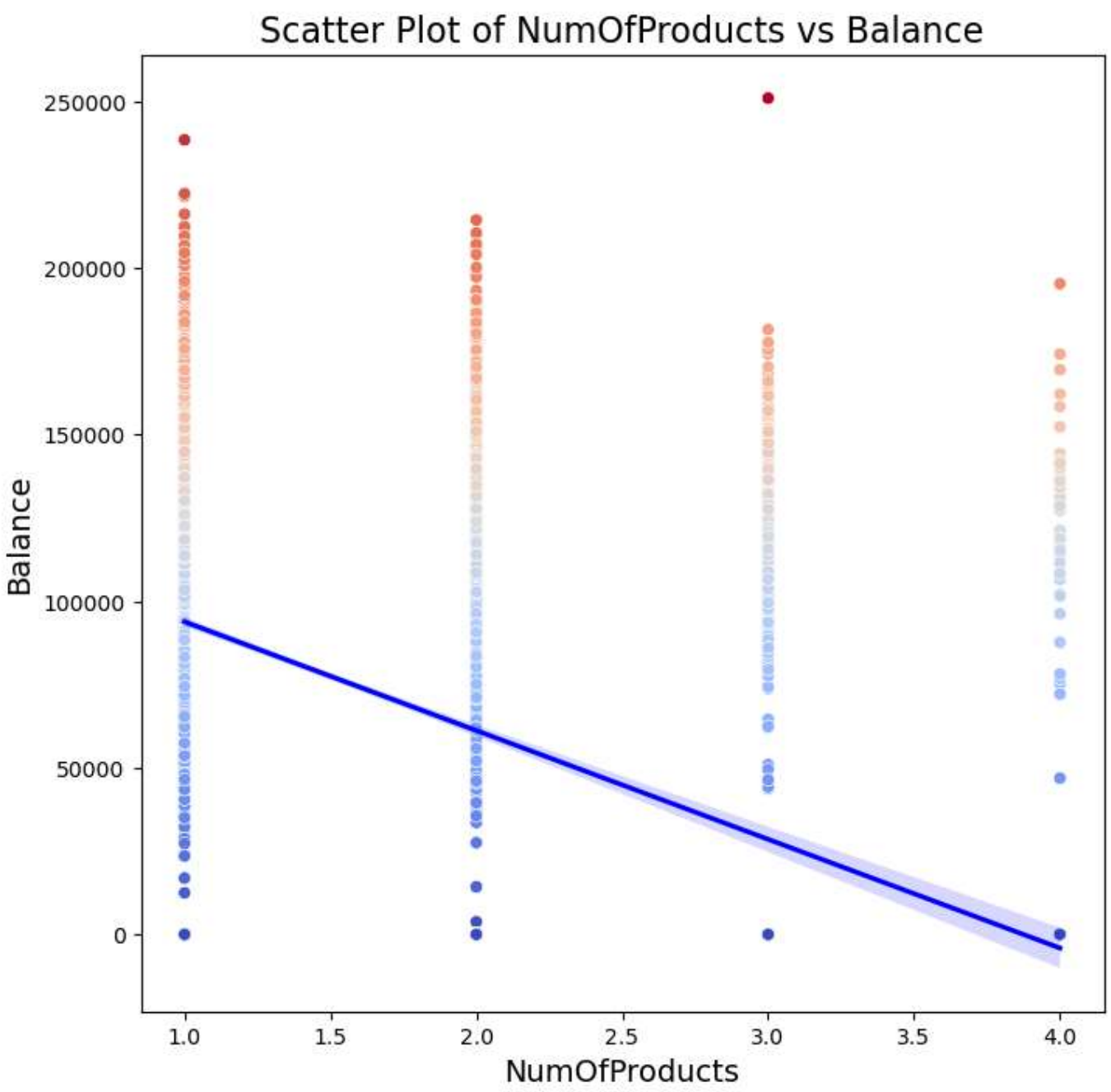
Scatter Plot

In [65]:
```
sns.regplot(x='NumOfProducts', y='Balance', data=df, scatter=False, color='blue', line_kws={'
```

Out[65]:
```
<Axes: xlabel='NumOfProducts', ylabel='Balance'>
```

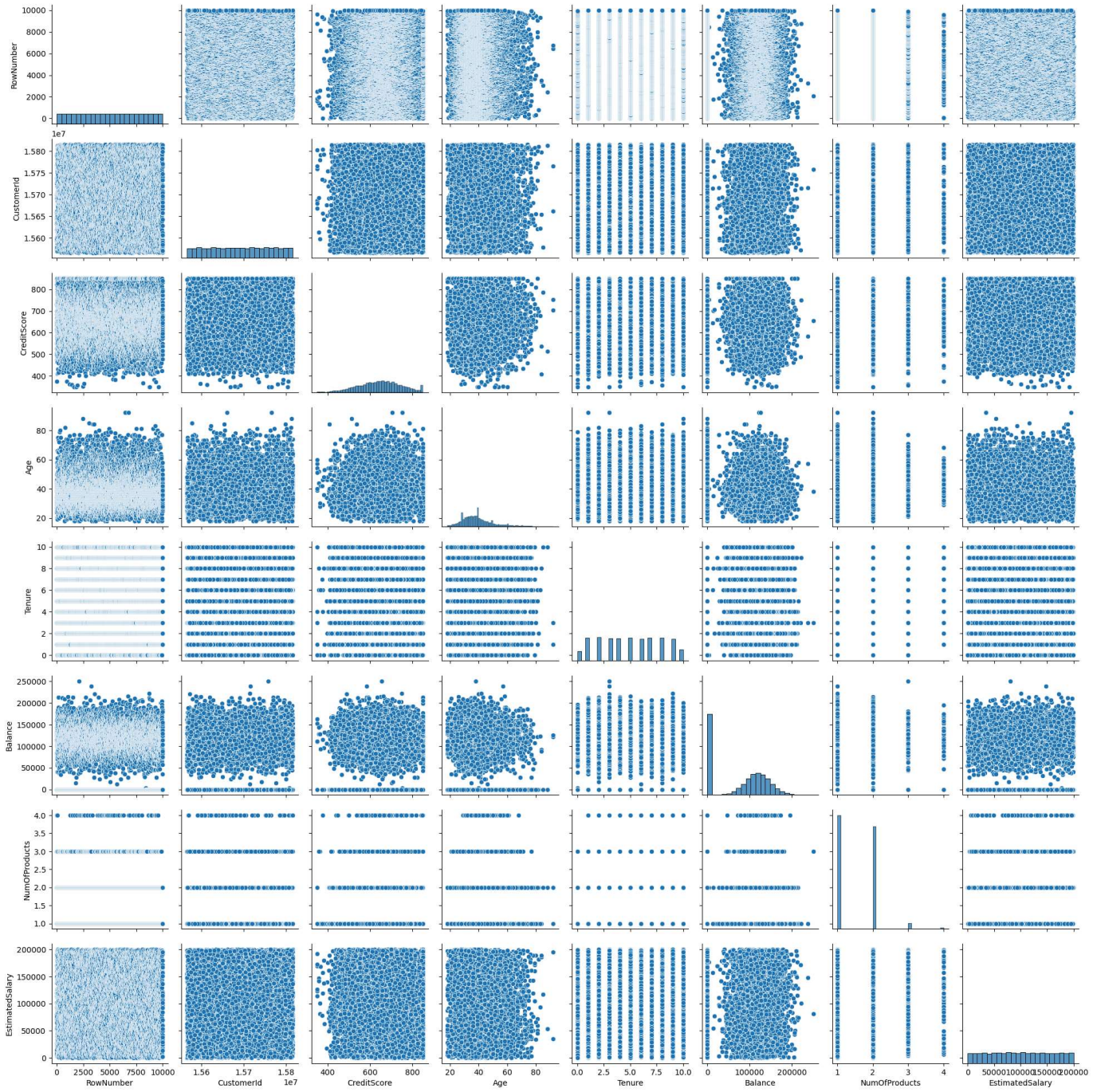


In [66]:
```
plt.figure(figsize=(8,8))
sns.scatterplot(x='NumOfProducts', y='Balance', data=df, hue='Balance', palette='coolwarm', s
sns.regplot(x='NumOfProducts', y='Balance', data=df, scatter=False, color='blue', line_kws={'
plt.title('Scatter Plot of NumOfProducts vs Balance', fontsize=16)
plt.xlabel('NumOfProducts', fontsize=14)
plt.ylabel('Balance', fontsize=14)
norm = plt.Normalize(df['Balance'].min(), df['Balance'].max())
sm = plt.cm.ScalarMappable(cmap='coolwarm', norm=norm)
plt.show()
```



PAIR PLOT = A pair plot (or scatterplot matrix) is a grid of scatterplots that displays pairwise relationships between numerical variables in a dataset. It is commonly used for exploratory data analysis to identify trends, correlations, and distributions.

In [83]:
```
sns.pairplot(numerical_df)
plt.show()
```



b. Discrete vs Discrete Data

In [69]:
```
df.head()
```

Out[69]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	Nu
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	

In [70]:
```
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  object
 11  IsActiveMember   10000 non-null  object
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  object
dtypes: float64(2), int64(6), object(6)
memory usage: 1.1+ MB
```

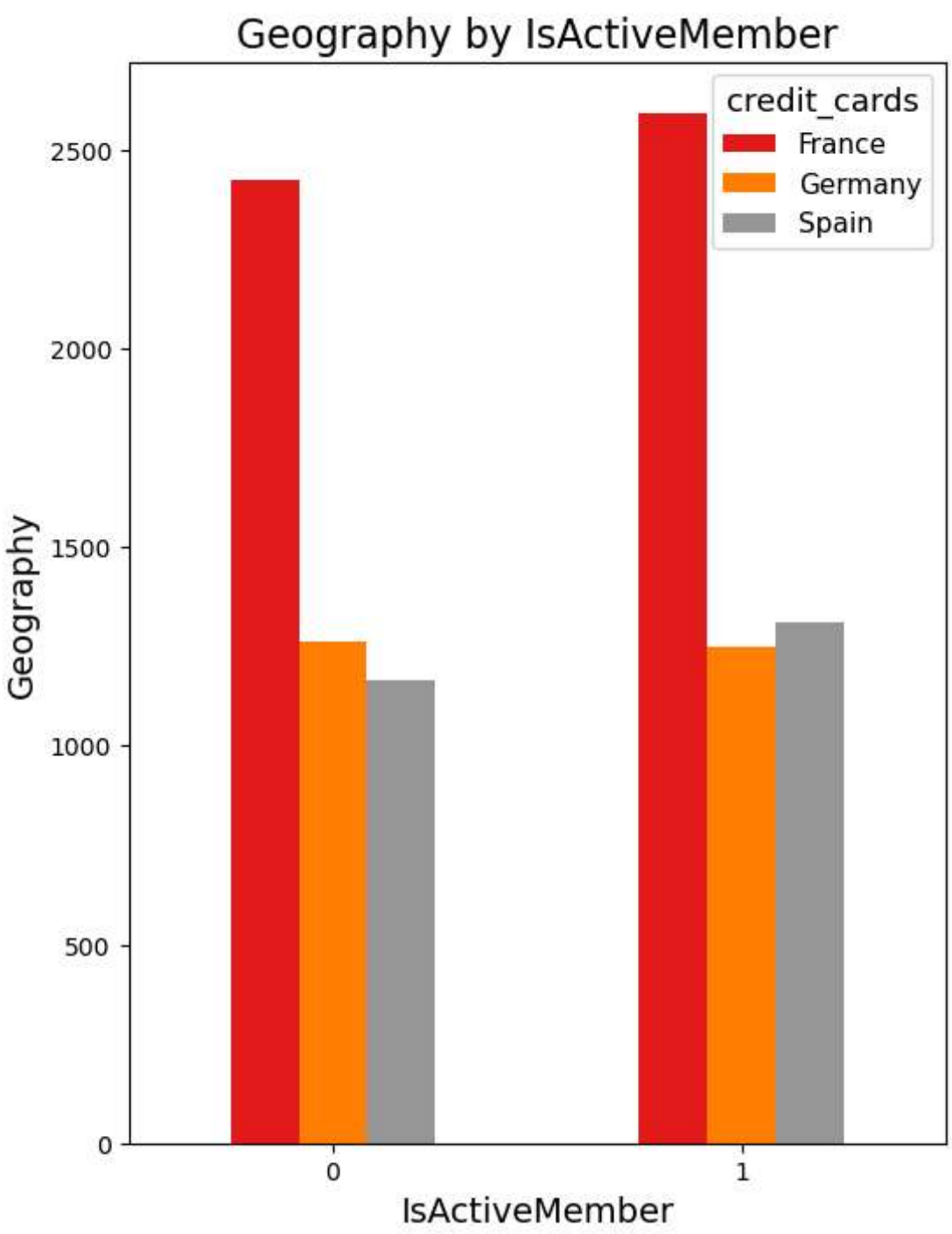
CROSS TAB : A cross tabulation is a statistical tool used to summarize the relationship between two or more categorical variables in a tabular format.

In [71]:
```
pd.crosstab(df['IsActiveMember'],df['Geography'])
```

Out[71]:

Geography	France	Germany	Spain
IsActiveMember			
0	2423	1261	1165
1	2591	1248	1312

In [85]:
```
tab = pd.crosstab(df['IsActiveMember'], df['Geography'])
ax = tab.plot(kind='bar', figsize=(6, 8), colormap='Set1')
plt.title('Geography by IsActiveMember', fontsize=16)
plt.xlabel('IsActiveMember', fontsize=14)
plt.ylabel('Geography', fontsize=14)
plt.xticks(rotation=0)
plt.legend(title='credit_cards', title_fontsize='13', fontsize='11')
plt.show()
```



- the population count of credit card active members are more in france as we compare to other countries

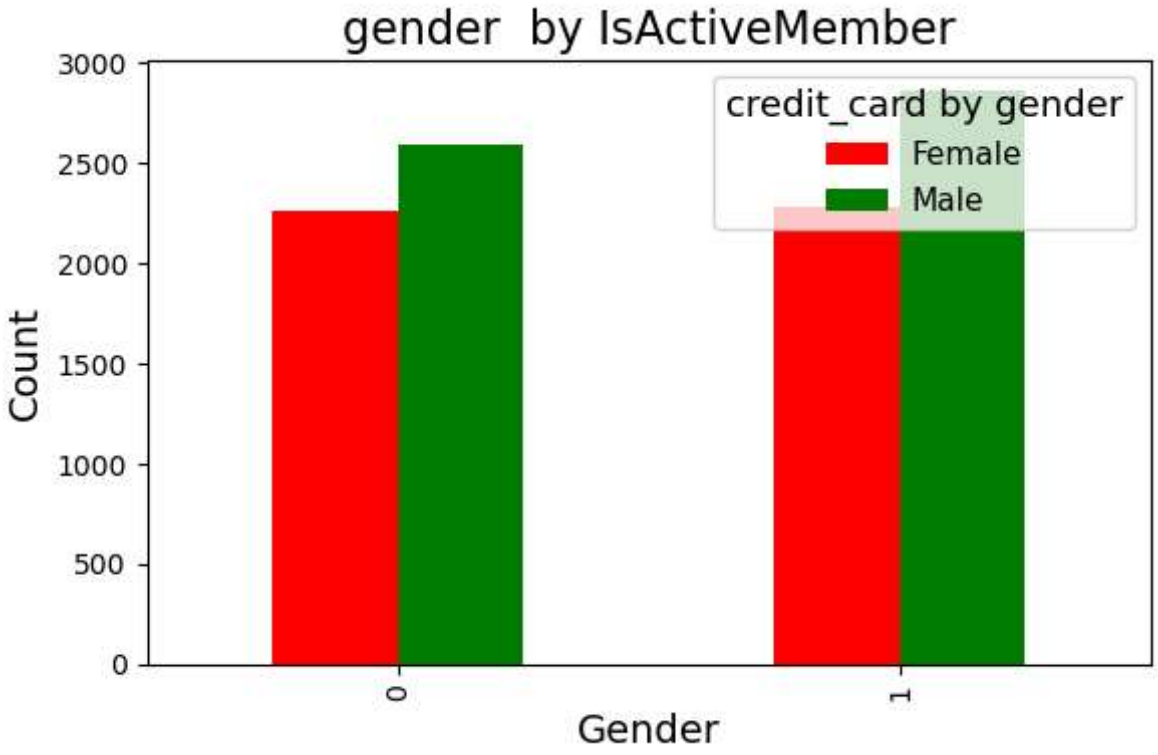
- the population count of credit card active members are less in germany as we compare to other france

- the population count of credit card active members are more less in spain as we compare to other countries

In [ ]:
```
pd.crosstab(df['IsActiveMember'],df['Gender'])
```

In [74]:
```
tab = pd.crosstab(df['IsActiveMember'], df['Gender'])
ax = tab.plot(kind='bar', figsize=(6, 4), color = ["red","green"])
plt.title('gender  by IsActiveMember', fontsize=16)
plt.xlabel('Gender', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.legend(title='credit_card by gender', title_fontsize='13', fontsize='11')
```

```
plt.tight_layout()
plt.show()
```

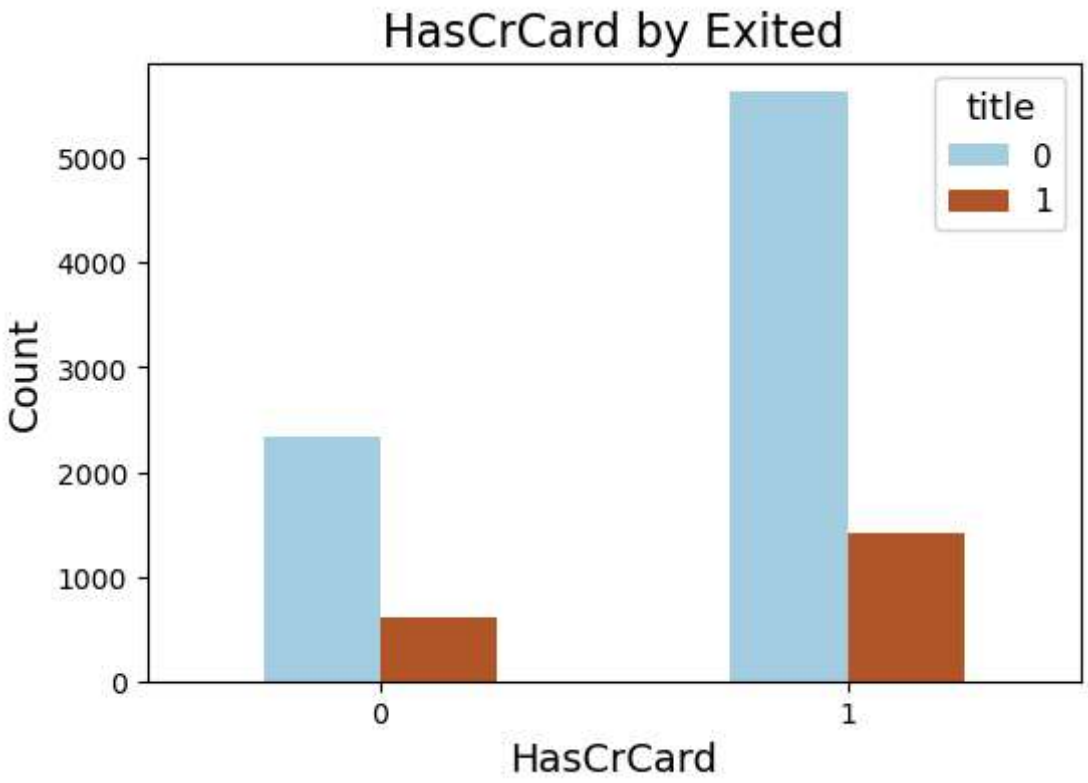


In [75]:
```
pd.crosstab(df['HasCrCard'], df['Exited'])
```

Out[75]:

Exited	0	1
HasCrCard		
0	2332	613
1	5631	1424

In [76]:
```
tab = pd.crosstab(df['HasCrCard'], df['Exited'])
ax = tab.plot(kind='bar', figsize=(6, 4), colormap='Paired')
plt.title('HasCrCard by Exited', fontsize=16)
plt.xlabel('HasCrCard', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(rotation=0)
plt.legend(title='title', title_fontsize='13', fontsize='11')
plt.show()
```

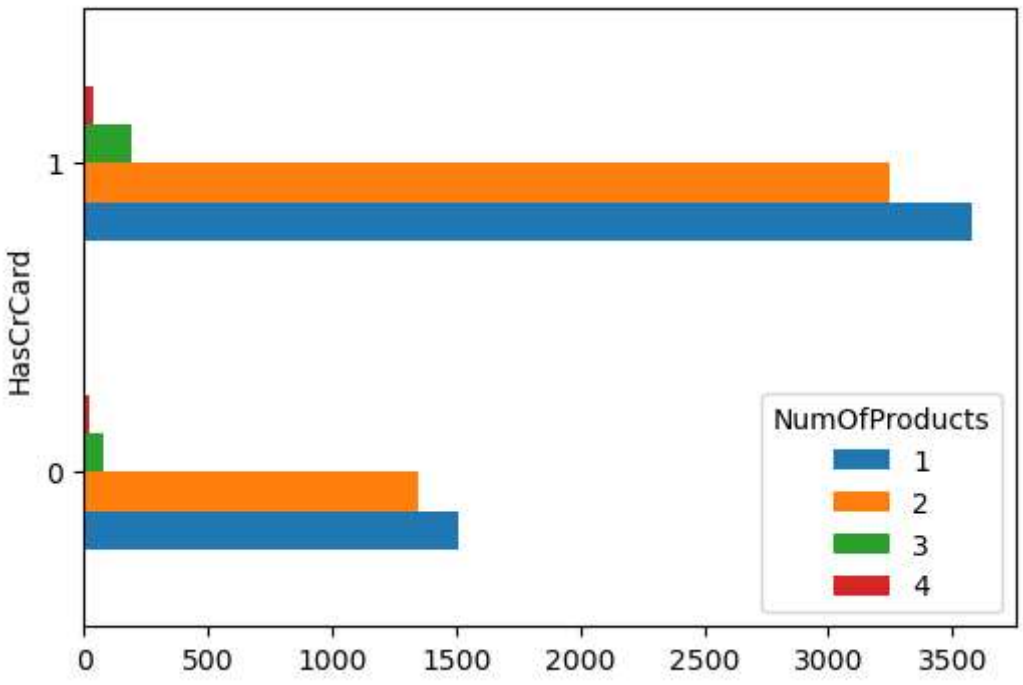


In [77]:
```
pd.crosstab(df['HasCrCard'], df['NumOfProducts'])
```

Out[77]:

HasCrCard	NumOfProducts 1	2	3	4
0	1506	1344	76	19
1	3578	3246	190	41

In [78]:
```
tab=pd.crosstab(df['HasCrCard'], df['NumOfProducts'])
tab.plot(kind='bar',figsize=(6,4))
plt.show()
```



c. Continuous Numerical vs Discrete Data

In [80]:
```
group= df.groupby('Exited')
```

```
group['Age'].agg(['min', 'max', 'mean', 'median'])
```

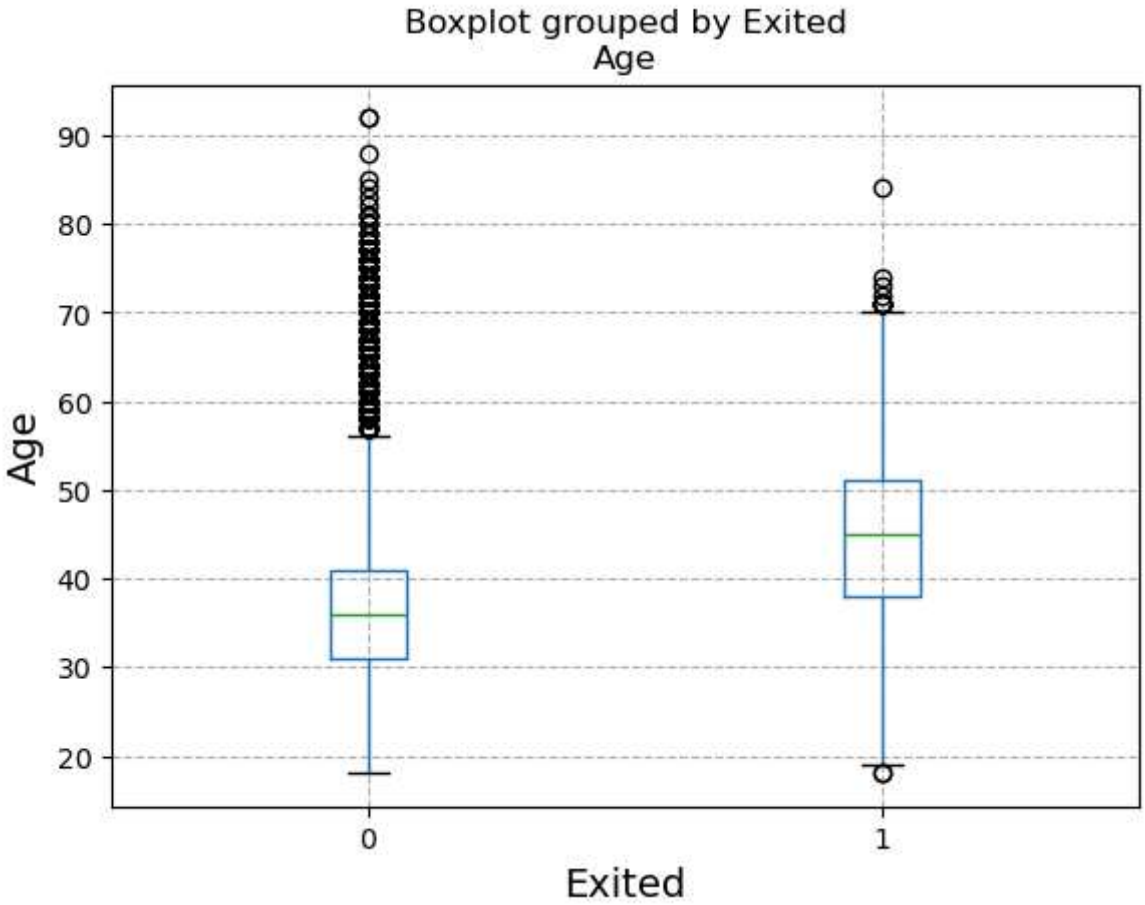
Out[80]:

Exited	min	max	mean	median
0	18	92	37.408389	36.0
1	18	84	44.837997	45.0

In [96]:

```
plt.figure(figsize=(6, 4))
ax = df.boxplot(by='Exited', column='Age', grid=False)
#plt.title('Age by Exited', fontsize=8)
plt.xlabel('Exited', fontsize=14)
plt.ylabel('Age', fontsize=14)
plt.xticks(rotation=0)
plt.grid(True, which='major', linestyle='--')
plt.show()
```

```
<Figure size 600x400 with 0 Axes>
```



- There are not most exited population in the age of 60-70 age group

In [82]:

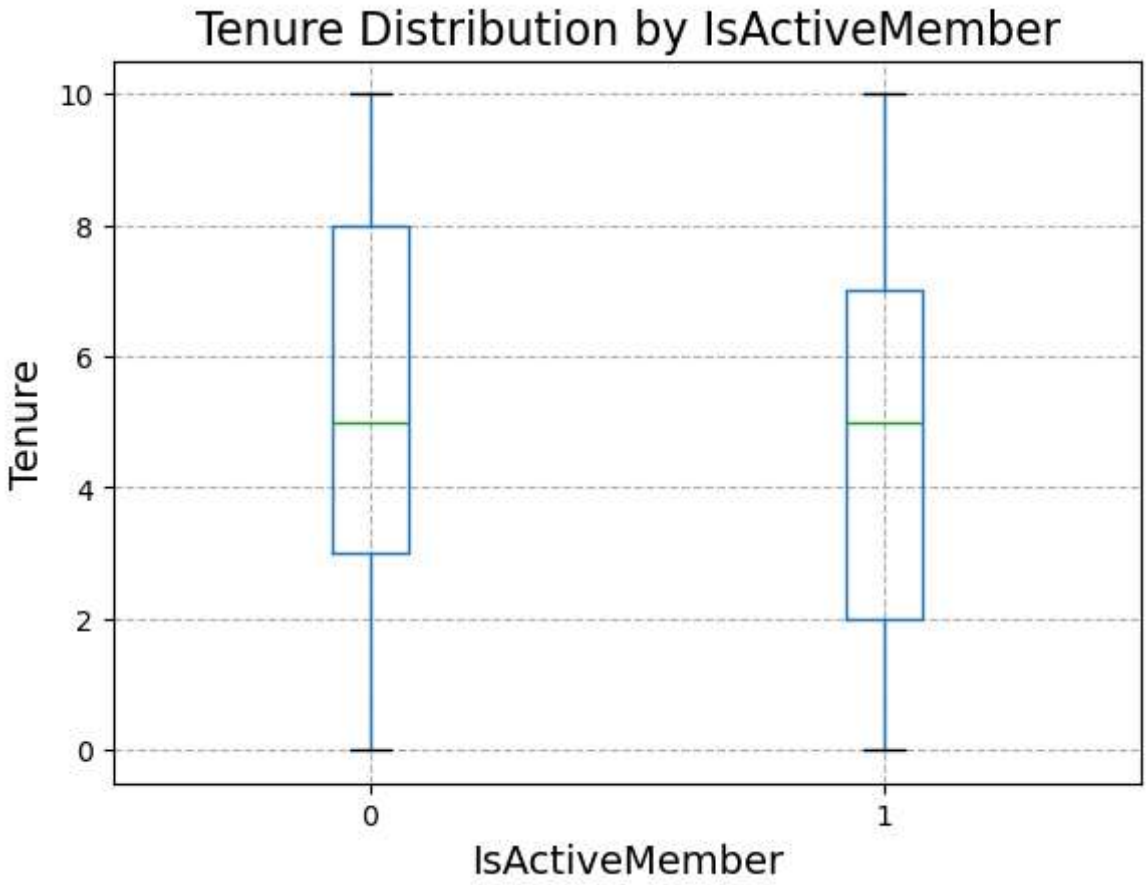
```
group= df.groupby('IsActiveMember')

group['Tenure'].agg(['min', 'max', 'mean', 'median'])
```

Out[82]:

IsActiveMember	min	max	mean	median
0	0	10	5.097340	5.0
1	0	10	4.933217	5.0

In [83]:
```
plt.figure(figsize=(6, 4))
ax = df.boxplot(by='IsActiveMember', column='Tenure', grid=False)
plt.title('Tenure Distribution by IsActiveMember', fontsize=16)
plt.suptitle('')
plt.xlabel('IsActiveMember', fontsize=14)
plt.ylabel('Tenure', fontsize=14)
plt.xticks(rotation=0)
plt.grid(True,  linestyle='--')
plt.show()
```

```
<Figure size 600x400 with 0 Axes>
```



Multivariant Analaysics

HEATMAP PLOT = A heatmap is a graphical representation of data where individual values are represented as colors. It's a great way to visualize relationships in a dataset, especially for correlation matrices or pivot tables

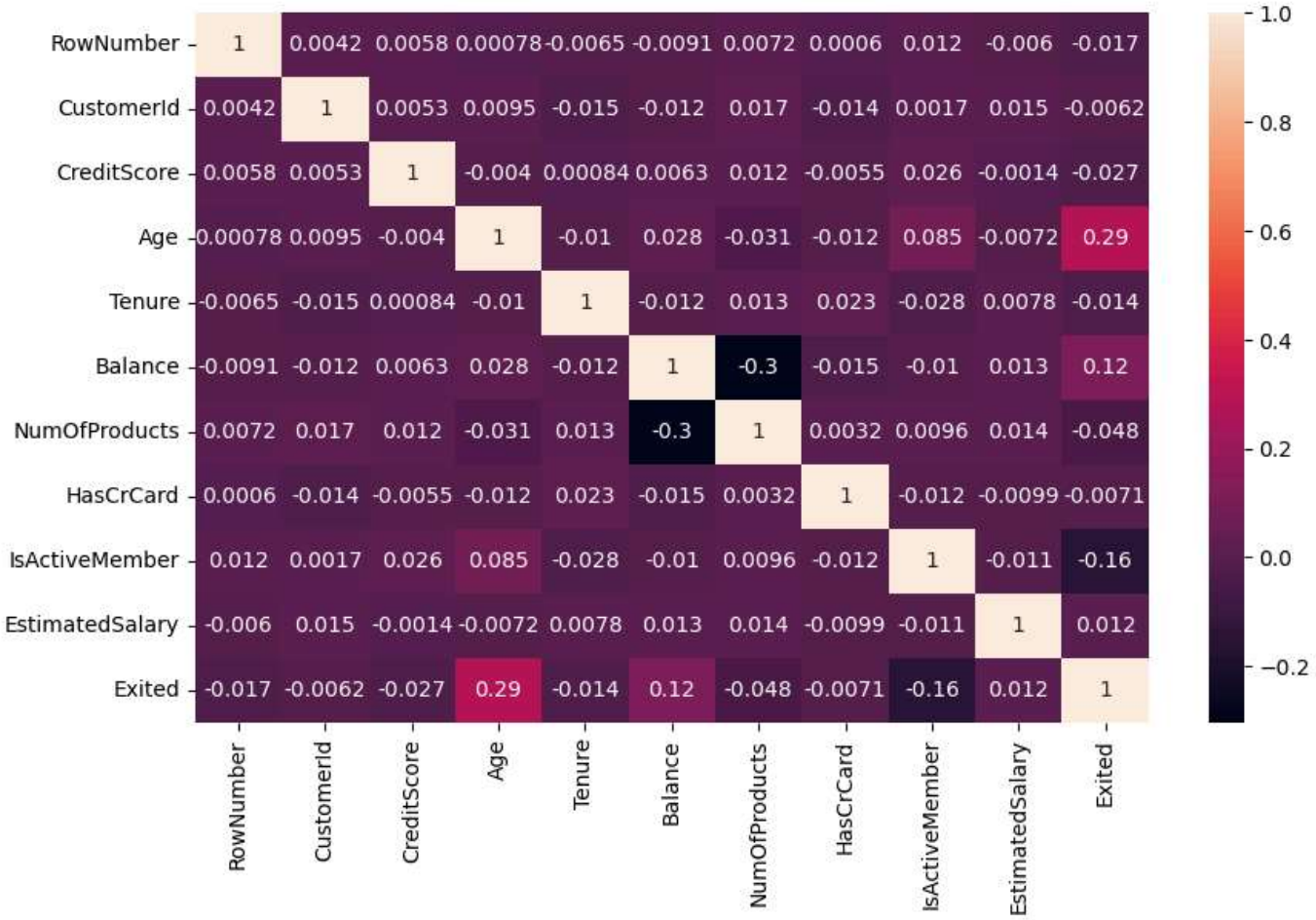
In [85]:
```
correlation_matrix= numerical_df.corr()
correlation_matrix
```

Out[85]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProduc
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495	-0.009067	0.0072
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883	-0.012419	0.0169
CreditScore	0.005840	0.005308	1.000000	-0.003965	0.000842	0.006268	0.0122
Age	0.000783	0.009497	-0.003965	1.000000	-0.009997	0.028308	-0.0306
Tenure	-0.006495	-0.014883	0.000842	-0.009997	1.000000	-0.012254	0.0134
Balance	-0.009067	-0.012419	0.006268	0.028308	-0.012254	1.000000	-0.3041
NumOfProducts	0.007246	0.016972	0.012238	-0.030680	0.013444	-0.304180	1.0000
EstimatedSalary	-0.005988	0.015271	-0.001384	-0.007201	0.007784	0.012797	0.0142

In [27]:

```
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(numeric_only=True),annot=True)
plt.show()
```



- Strong Positive :- Age and Exited¶
- Weak Positive:- Balance and Num of products
- Weak Negative:- Isactivemember and exited

INSIGHTS:

Univariate Analysics:

The orange slice represents the percentage of France (50.1%), the blue slice represents the percentage of Germany (25.1%), the green represent the percentage of Spain

The orange slice represents the percentage of male (54.6%), and the blue slice represents the percentage of female (45.4%) (24.8%)

There are more male users are using credit cards

There are less female users are using credit cards

There are more users of credit card in France and less users in Spain and Germany

male population has greater than 5000 above as we compare to female there are less

Female population has in between 4000 and 5000

Bivariate Analysis:

a. Continuous vs Continuous Numerical Data:

joint plot has two variates like univariate and bivariate to the estimated salary and age

The age in between 20-60 most of people having credit limit of 2lacs

The age in between 70-90 less people having credit limit of 2lacs and less than 2lacs

b. Discrete vs Discrete Data

The population count of credit card active members are more in France as we compare to other countries

The population count of credit card active members are less in Germany as we compare to other france

The population count of credit card active members are more less in Spain as we compare to other countries

c. Continuous Numerical vs Discrete Data

There are not most exited population in the age of 60-70 age group

Multivariant Analysis:

Strong Positive :- Age and Exited¶

Weak Positive:- Balance and Num of products

Weak Negative:- Isactivemembers and exited

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: