In [21]:

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'

!pip install seaborn --upgrade
import seaborn as sns
sns.set_style('darkgrid')
import sklearn

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder


print('✔ Libraries Imported!')
```

```
Requirement already up-to-date: seaborn in c:\users\sarah\anaconda3\lib\site-packages (0.12.1)
Requirement already satisfied, skipping upgrade: typing_extensions; python_version < "3.8" in c:\users\sarah\anaconda3\lib
\site-packages (from seaborn) (4.2.0)
Requirement already satisfied, skipping upgrade: pandas>=0.25 in c:\users\sarah\anaconda3\lib\site-packages (from seaborn)
(1.3.5)
Requirement already satisfied, skipping upgrade: matplotlib!=3.6.1,>=3.1 in c:\users\sarah\anaconda3\lib\site-packages (fro
m seaborn) (3.5.1)
Requirement already satisfied, skipping upgrade: numpy>=1.17 in c:\users\sarah\anaconda3\lib\site-packages (from seaborn)
(1.21.6)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.7.3 in c:\users\sarah\anaconda3\lib\site-packages (from
pandas>=0.25->seaborn) (2.8.2)
Requirement already satisfied, skipping upgrade: pytz>=2017.3 in c:\users\sarah\anaconda3\lib\site-packages (from pandas>=
0.25->seaborn) (2019.1)
Requirement already satisfied, skipping upgrade: packaging>=20.0 in c:\users\sarah\anaconda3\lib\site-packages (from matplo
tlib!=3.6.1,>=3.1->seaborn) (21.3)
Requirement already satisfied, skipping upgrade: cycler>=0.10 in c:\users\sarah\anaconda3\lib\site-packages (from matplotli
b!=3.6.1,>=3.1->seaborn) (0.10.0)
Requirement already satisfied, skipping upgrade: kiwisolver>=1.0.1 in c:\users\sarah\anaconda3\lib\site-packages (from matp
lotlib!=3.6.1,>=3.1->seaborn) (1.1.0)
Requirement already satisfied, skipping upgrade: pillow>=6.2.0 in c:\users\sarah\anaconda3\lib\site-packages (from matplotl
ib!=3.6.1,>=3.1->seaborn) (9.1.0)
Requirement already satisfied, skipping upgrade: fonttools>=4.22.0 in c:\users\sarah\anaconda3\lib\site-packages (from matp
lotlib!=3.6.1,>=3.1->seaborn) (4.31.2)
Requirement already satisfied, skipping upgrade: pyparsing>=2.2.1 in c:\users\sarah\anaconda3\lib\site-packages (from matpl
otlib!=3.6.1,>=3.1->seaborn) (2.4.0)
Requirement already satisfied, skipping upgrade: six>=1.5 in c:\users\sarah\anaconda3\lib\site-packages (from python-dateut
il>=2.7.3->pandas>=0.25->seaborn) (1.12.0)
Requirement already satisfied, skipping upgrade: setuptools in c:\users\sarah\anaconda3\lib\site-packages (from kiwisolver>
=1.0.1->matplotlib!=3.6.1,>=3.1->seaborn) (41.0.1)
✔ Libraries Imported!
```

In [3]:

```
#conda install -c conda-forge scikit-plot
```

```
  - defaults/noarch::dask==2.1.0=py_0
  - defaults/win-64::h5py==2.9.0=py37h5e291fa_0
  - defaults/win-64::imageio==2.5.0=py37_0
  - defaults/win-64::mkl-service==2.0.2=py37he774522_0
  - defaults/win-64::mkl_fft==1.0.12=py37h14836fe_0
  - defaults/win-64::mkl_random==1.0.2=py37h343c172_0
  - defaults/win-64::numba==0.44.1=py37hf9181ef_0
  - defaults/win-64::numexpr==2.6.9=py37hdce8814_0
  - defaults/win-64::numpy==1.16.4=py37h19fb1c0_0
  - defaults/noarch::numpydoc==0.9.1=py_0
  - defaults/win-64::patsy==0.5.1=py37_0
  - defaults/win-64::pytables==3.5.2=py37h1da0976_1
  - defaults/win-64::pytest==5.0.1=py37_0
  - defaults/win-64::pytest-arraydiff==0.3=py37h39e3cac_0
  - defaults/win-64::pytest-astropy==0.5.0=py37_0
  - defaults/win-64::pytest-doctestplus==0.3.0=py37_0
  - defaults/win-64::pytest-openfiles==0.3.2=py37_0
  - defaults/win-64::pytest-remotedata==0.3.1=py37_0
  - defaults/win-64::pywavelets==1.0.3=py37h8c2d366_1
  - defaults/win-64::scikit-image==0.15.0=py37ha925a31_0
```

In [2]:

```
df = pd.read_csv("C:/BI/CIND 820/Files/Churn_Modelling.csv", encoding = 'utf-8')

print('✔ Dataset Imported Successfully!\n')
print('It contains {} rows and {} columns.'.format(df.shape[0], df.shape[1]))
```

```
✔ Dataset Imported Successfully!

It contains 10000 rows and 14 columns.
```

In [3]:

```
df.head()
```

Out[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedS
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	1013
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	1125
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	1139
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	938
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	790

In [4]:

```
#!pip install pandas-profiling
#import sys
#!{sys.executable} -m pip install pandas-profiling
```

In [5]:

```
from pandas_profiling import ProfileReport
ProfileReport(df) #to display the report
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

6665	1	< 0.1%
6666	1	< 0.1%
6667	1	< 0.1%
6668	1	< 0.1%
6669	1	< 0.1%
6670	1	< 0.1%
6672	1	< 0.1%
Other values (9990)	9990	99.9%

Value	Count	Frequency (%)
1	1	< 0.1%
2	1	< 0.1%
3	1	< 0.1%
4	1	< 0.1%
5	1	< 0.1%
6	1	< 0.1%
7	1	< 0.1%
8	1	< 0.1%
9	1	< 0.1%
10	1	< 0.1%

Out[5]:

In [ ]:

```

```

In [4]:

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

Out[4]:

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

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   CreditScore      10000 non-null  int64  
 1   Geography        10000 non-null  object 
 2   Gender           10000 non-null  object 
 3   Age              10000 non-null  int64  
 4   Tenure           10000 non-null  int64  
 5   Balance          10000 non-null  float64
 6   NumOfProducts    10000 non-null  int64  
 7   HasCrCard        10000 non-null  int64  
 8   IsActiveMember   10000 non-null  int64  
 9   EstimatedSalary  10000 non-null  float64
 10  Exited           10000 non-null  int64  
dtypes: float64(2), int64(7), object(2)
memory usage: 859.5+ KB
```

In [6]:

```
df.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
CreditScore	10000.0	650.528800	96.653299	350.00	584.00	652.000	718.0000	850.00
Age	10000.0	38.921800	10.487806	18.00	32.00	37.000	44.0000	92.00
Tenure	10000.0	5.012800	2.892174	0.00	3.00	5.000	7.0000	10.00
Balance	10000.0	76485.889288	62397.405202	0.00	0.00	97198.540	127644.2400	250898.09
NumOfProducts	10000.0	1.530200	0.581654	1.00	1.00	1.000	2.0000	4.00
HasCrCard	10000.0	0.705500	0.455840	0.00	0.00	1.000	1.0000	1.00
IsActiveMember	10000.0	0.515100	0.499797	0.00	0.00	1.000	1.0000	1.00
EstimatedSalary	10000.0	100090.239881	57510.492818	11.58	51002.11	100193.915	149388.2475	199992.48
Exited	10000.0	0.203700	0.402769	0.00	0.00	0.000	0.0000	1.00

In [9]:

```
#spliting our dataset into a train and test

random_state = 42
train_df, test_df = train_test_split(df, test_size=0.2, random_state=random_state)

train_df.reset_index(drop=True, inplace=True)
test_df.reset_index(drop=True, inplace=True)

print('Train set: {} rows x {} columns'.format(train_df.shape[0],
                                               train_df.shape[1]))
print(' Test set: {} rows x {} columns'.format(test_df.shape[0],
                                               test_df.shape[1]))
```

```
Train set: 8000 rows x 11 columns
 Test set: 2000 rows x 11 columns
```

In [11]:

```
def plot_continuous(feature):
    '''Plot a histogram and boxplot for the churned and retained distributions for the specified feature.'''
    df_func = train_df.copy()
    df_func['Exited'] = df_func['Exited'].astype('category')

    fig, (ax1, ax2) = plt.subplots(2,
                                   figsize=(9, 7),
                                   sharex=True,
                                   gridspec_kw={'height_ratios': (.7, .3)})

    for df, color, label in zip([df_retained, df_churned], colors, ['Retained', 'Churned']):
        sns.histplot(data=df,
                     x=feature,
                     bins=15,
                     color=color,
                     alpha=0.66,
                     edgecolor='firebrick',
                     label=label,
                     kde=False,
                     ax=ax1)
    ax1.legend()

    sns.boxplot(x=feature, y='Exited', data=df_func, palette=colors, ax=ax2)
    ax2.set_ylabel('')
    ax2.set_yticklabels(['Retained', 'Churned'])

    plt.tight_layout();
```

In [12]:

```
def plot_categorical(feature):
    '''For a categorical feature, plot a seaborn.countplot for the total counts of each category next to a barplot for the churn rate.'''
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))

    sns.countplot(x=feature,
                  hue='Exited',
                  data=train_df,
                  palette=colors,
                  ax=ax1)
    ax1.set_ylabel('Count')
    ax1.legend(labels=['Retained', 'Churned'])

    sns.barplot(x=feature,
                y='Exited',
                data=train_df,
                palette=colors_cat,
                ax=ax2)
    ax2.set_ylabel('Churn rate')

    if (feature == 'HasCrCard' or feature == 'IsActiveMember'):
        ax1.set_xticklabels(['No', 'Yes'])
        ax2.set_xticklabels(['No', 'Yes'])

    plt.tight_layout();
```

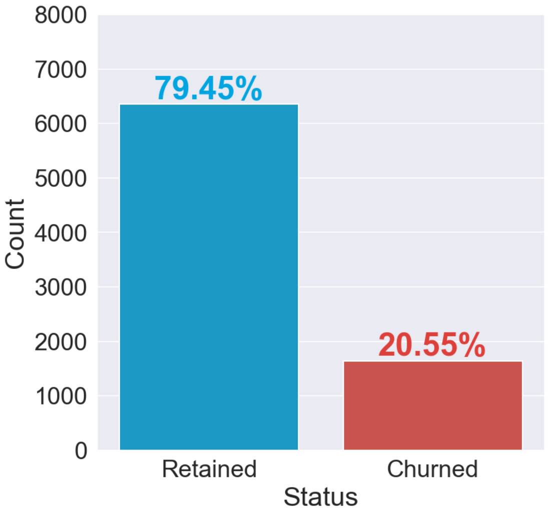
In [15]:

```
font_size = 20
plt.rcParams['axes.labelsize'] = font_size
plt.rcParams['axes.titlesize'] = font_size + 2
plt.rcParams['xtick.labelsize'] = font_size - 2
plt.rcParams['ytick.labelsize'] = font_size - 2
plt.rcParams['legend.fontsize'] = font_size - 2

colors = ['#00A5E0', '#DD403A']
colors_cat = ['#E8907E', '#D5CABD', '#7A6F86', '#C34A36', '#B0A8B9', '#845EC2', '#8f9aaa', '#FFB86F', '#63BAAA', '#9D88B3', '#38c4e3']
colors_comp = ['steelblue', 'seagreen', 'black', 'darkorange', 'purple', 'firebrick', 'slategrey']

random_state = 42
scoring_metric = 'recall'
comparison_dict, comparison_test_dict = {}, {}

print('✔ Default Parameters and Variables Set!')
```

```
✔ Default Parameters and Variables Set!
```

In [16]:

```
#Exploratory Data Analysis

fig, ax = plt.subplots(figsize=(6, 6))

sns.countplot(x='Exited', data=train_df, palette=colors, ax=ax)

for index, value in enumerate(train_df['Exited'].value_counts()):
    label = '{}%'.format(round((value / train_df['Exited'].shape[0]) * 100, 2))
    ax.annotate(label,
                xy=(index, value + 250),
                ha='center',
                va='center',
                color=colors[index],
                fontweight='bold',
                size=font_size + 4)

ax.set_xticklabels(['Retained', 'Churned'])
ax.set_xlabel('Status')
ax.set_ylabel('Count')
ax.set_ylim([0, 8000]);
```



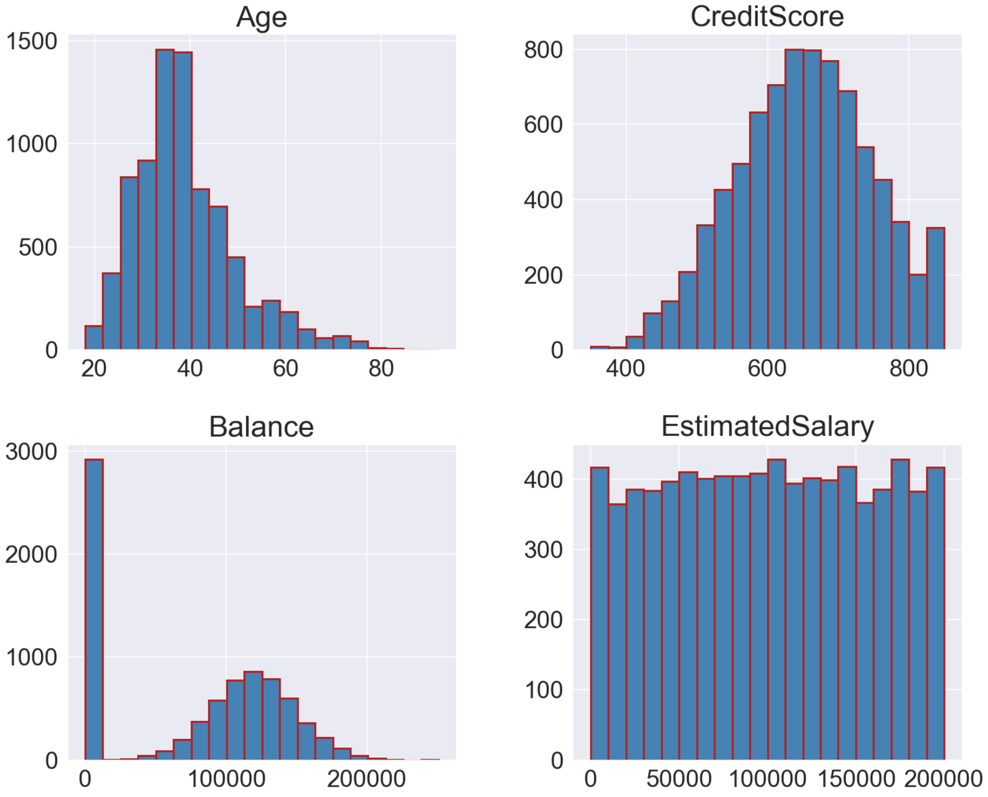
In [17]:

```
continuous = ['Age', 'CreditScore', 'Balance', 'EstimatedSalary']
categorical = ['Geography', 'Gender', 'Tenure', 'NumOfProducts', 'HasCrCard', 'IsActiveMember']

print('Continuous: ', ', '.join(continuous))
print('Categorical: ', ', '.join(categorical))
```

```
Continuous:  Age, CreditScore, Balance, EstimatedSalary
Categorical:  Geography, Gender, Tenure, NumOfProducts, HasCrCard, IsActiveMember
```

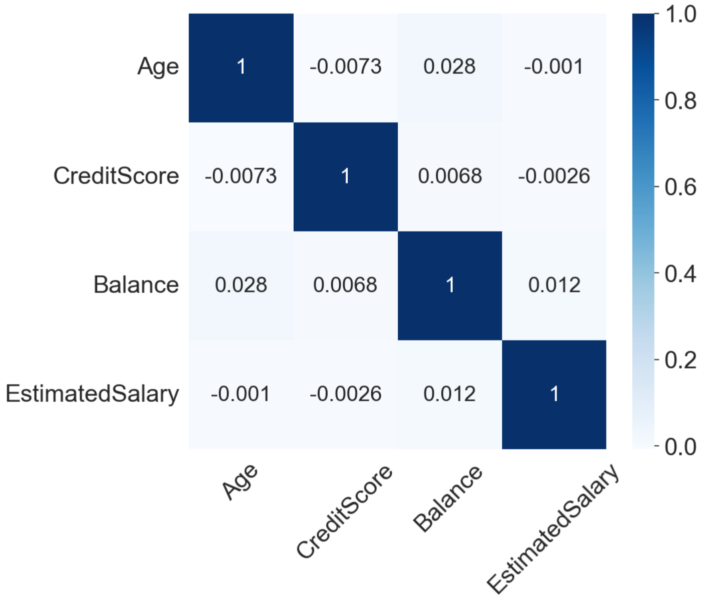
In [21]:

```
train_df[continuous].hist(figsize=(12, 10),
                          bins=20,
                          layout=(2, 2),
                          color='steelblue',
                          edgecolor='firebrick',
                          linewidth=1.5);
```



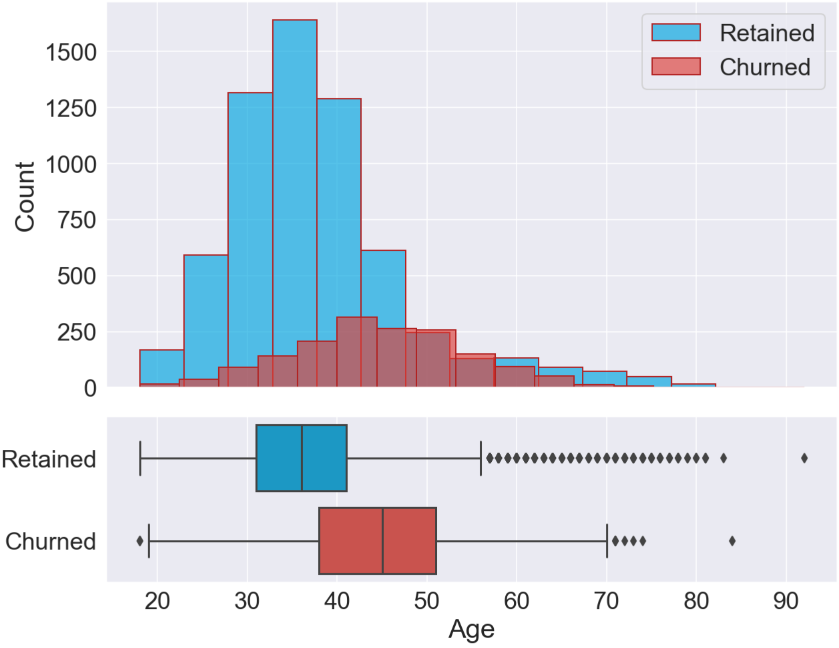
In [18]:

```
fig, ax = plt.subplots(figsize=(7, 6))

sns.heatmap(train_df[continuous].corr(),
            annot=True,
            annot_kws={'fontsize': 16},
            cmap='Blues',
            ax=ax)

ax.tick_params(axis='x', rotation=45)
ax.tick_params(axis='y', rotation=360);
```



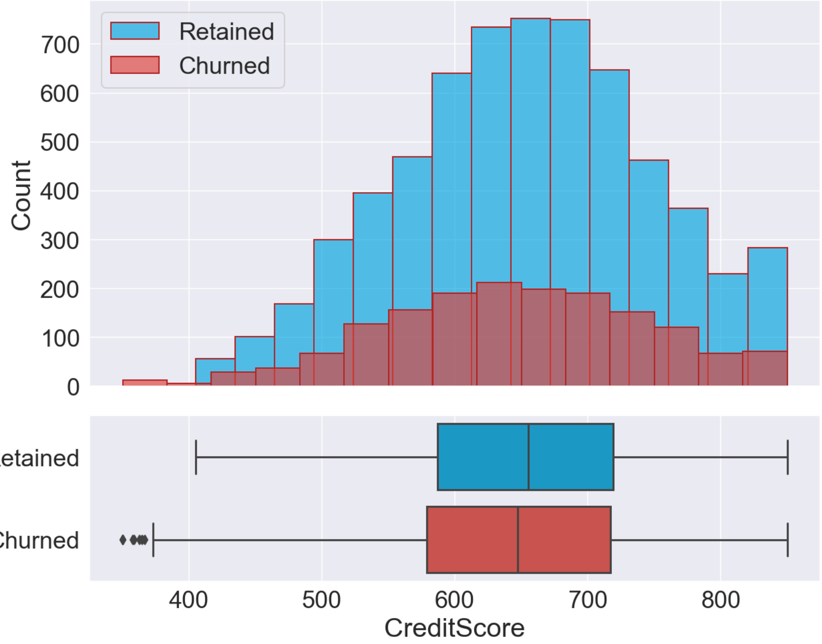
In [19]:

```
df_churned = train_df[train_df['Exited'] == 1]
df_retained = train_df[train_df['Exited'] == 0]

plot_continuous('Age')
```



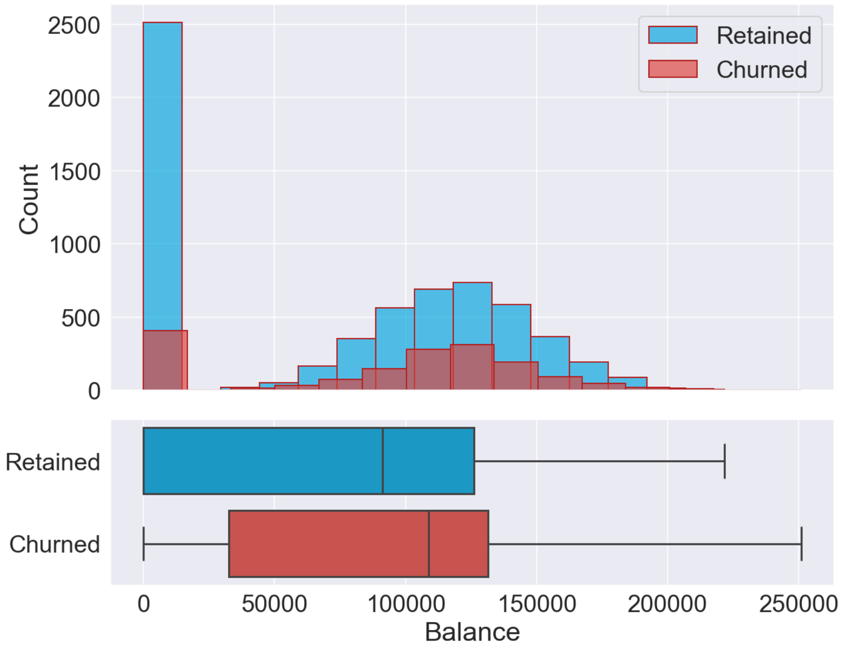
In [20]:

```
plot_continuous('CreditScore')
```



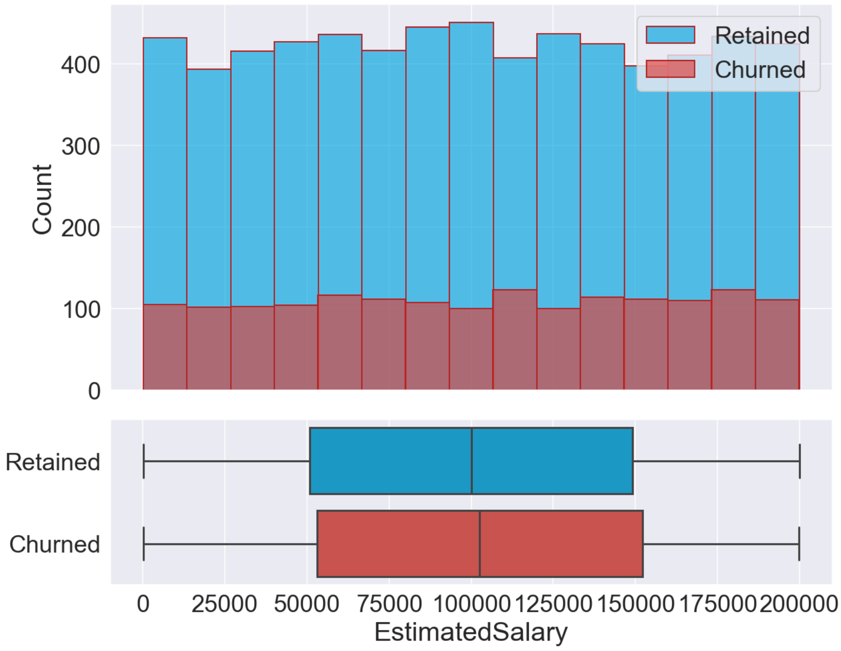
In [21]:

```
plot_continuous('Balance')
```



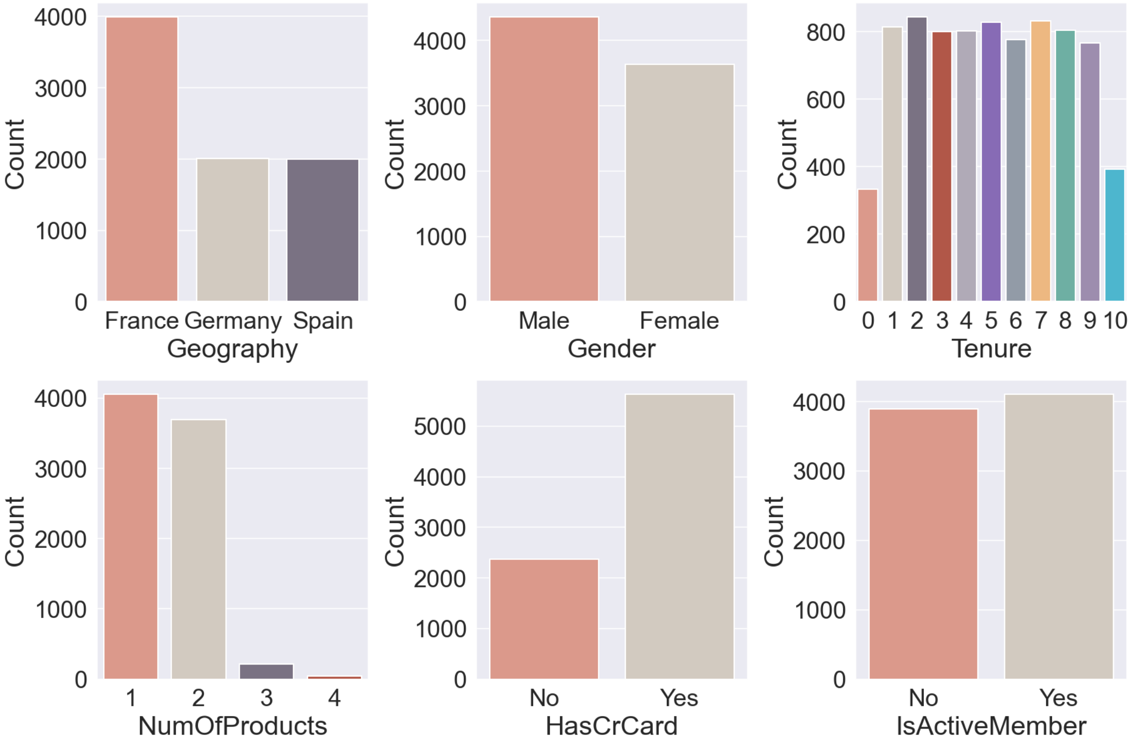
In [22]:

```
plot_continuous('EstimatedSalary')
```



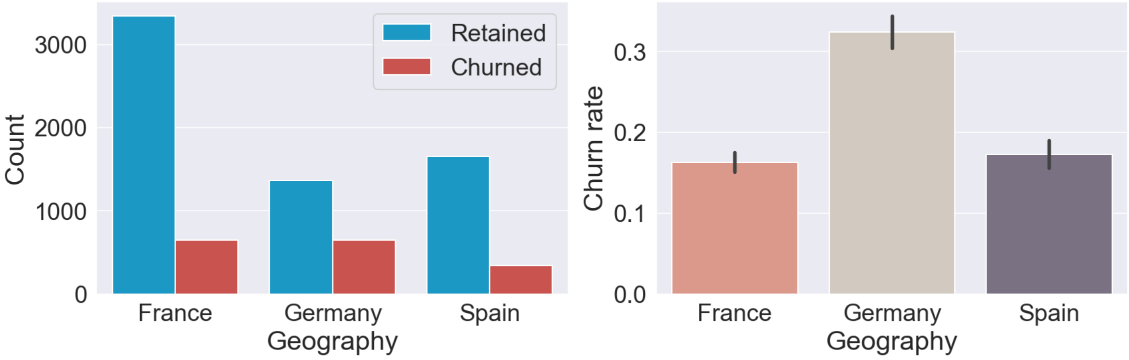
In [24]:

```
df_cat = train_df[categorical]

fig, ax = plt.subplots(2, 3, figsize=(12, 8))

for index, column in enumerate(df_cat.columns):

    plt.subplot(2, 3, index + 1)
    sns.countplot(x=column, data=train_df, palette=colors_cat)

    plt.ylabel('Count')
    if (column == 'HasCrCard' or column == 'IsActiveMember'):
        plt.xticks([0, 1], ['No', 'Yes'])

plt.tight_layout();
```



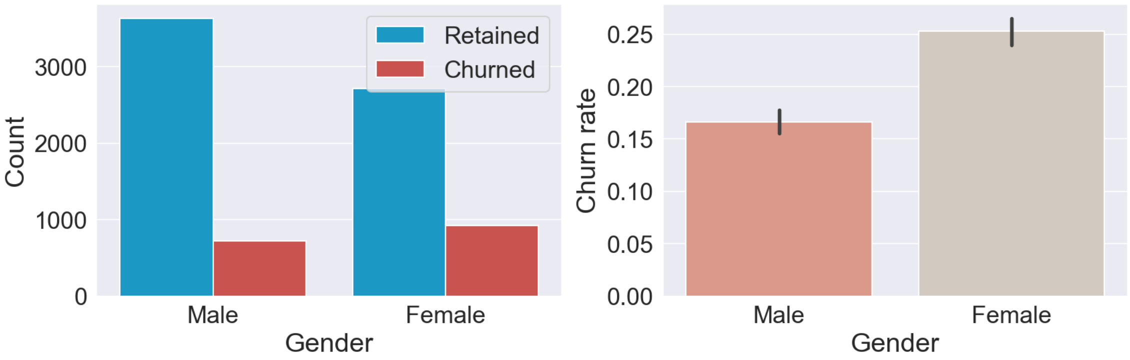
In [24]:

```
plot_categorical('Geography')
```



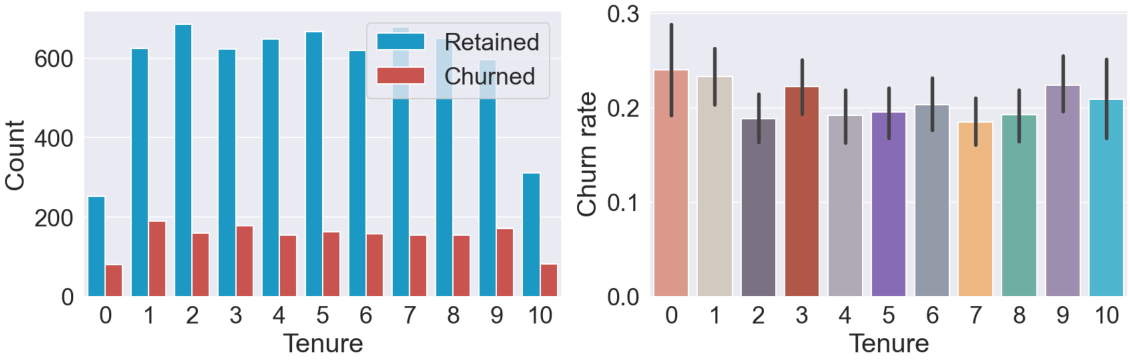
In [25]:

```
plot_categorical('Gender')
```



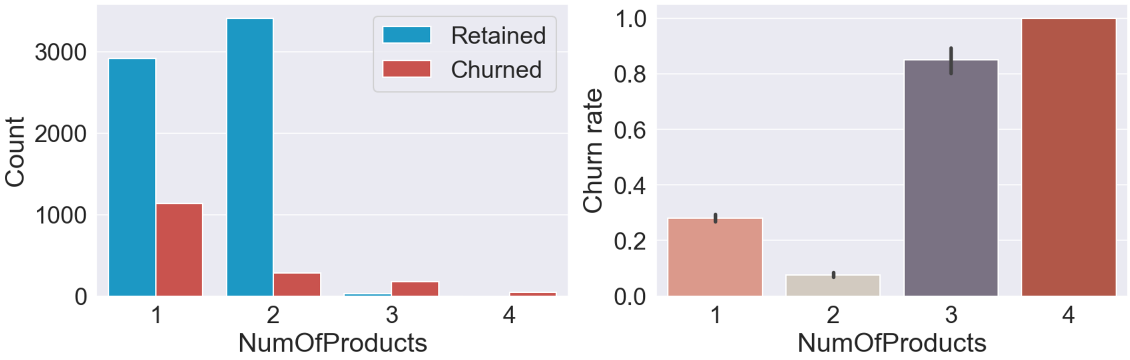
In [26]:

```
plot_categorical('Tenure')
```



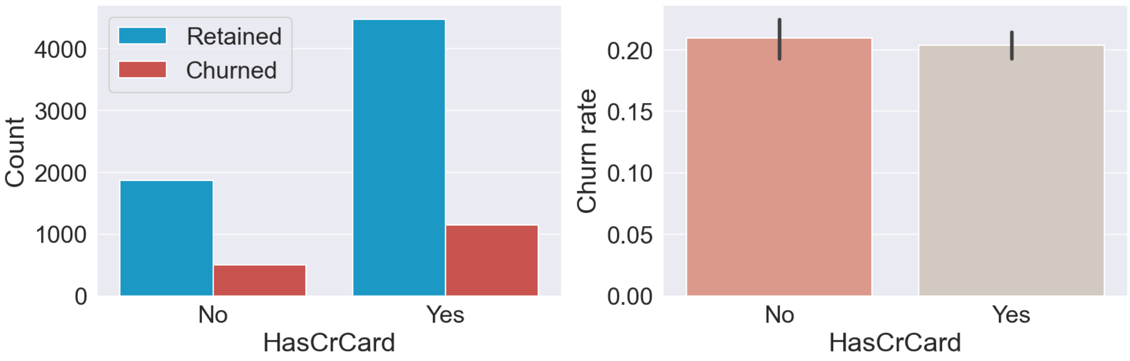
In [27]:

```
plot_categorical('NumOfProducts')
```



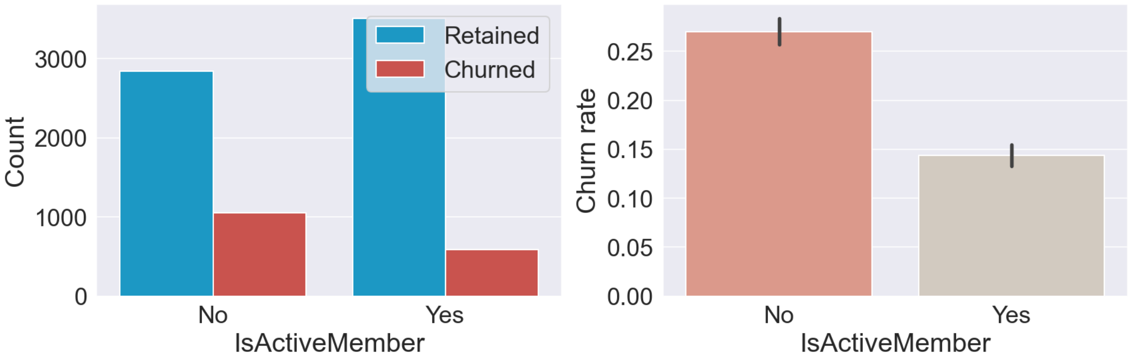
In [28]:

```
plot_categorical('HasCrCard')
```



In [29]:

```
plot_categorical('IsActiveMember')
```



In [22]:

```
train_df['Gender'] = LabelEncoder().fit_transform(train_df['Gender'])

train_df['Geography'] = train_df['Geography'].map({
    'Germany': 1,
    'France': 0,
    'Spain': 0
})

print('✔ Features Encoded!')
```

```
✔ Features Encoded!
```

train_df['Gender'] = LabelEncoder().fit_transform(train_df['Gender'])

train_df['Geography'] = train_df['Geography'].map({ 'Germany': 1, 'Spain': 0, 'France': 0 })

print('✔ Features Encoded!')

In [ ]: