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

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

```
df.head()
```

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

```
print(df.isnull().sum())
print(df.describe())
```

```
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
         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.shape
```

```
(10000, 14)
```

```
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
```

```
for col in df.columns:
  df[col]=df[col].fillna(df[col].mode()[0])
```

```
from scipy.stats import zscore
```

```
from sklearn.preprocessing import LabelEncoder
for col in df.columns:
  le=LabelEncoder()
  df[col]=le.fit_transform(df[col])
```

```
zscore_df = df.apply(zscore)
```

```
o_c=((zscore_df>3)|(zscore_df<-3)).sum()
o_c
```

```
RowNumber            0
CustomerId           0
Surname              0
CreditScore          0
Geography            0
Gender               0
Age                133
Tenure               0
Balance              0
```

```
NumOfProducts       60
HasCrCard            0
IsActiveMember       0
EstimatedSalary      0
Exited               0
dtype: int64
```

```
import pandas as pd
import numpy as np

# Assuming zscore_df is your DataFrame containing z-scores
# and you want to drop rows where z-score is greater than 3 or less than -3
outlier_mask = (zscore_df > 3) | (zscore_df < -3)

# Find rows where at least one z-score is an outlier
outlier_rows = np.any(outlier_mask, axis=1)

# Filter out rows with outliers
cleaned_df = zscore_df[~outlier_rows]
```

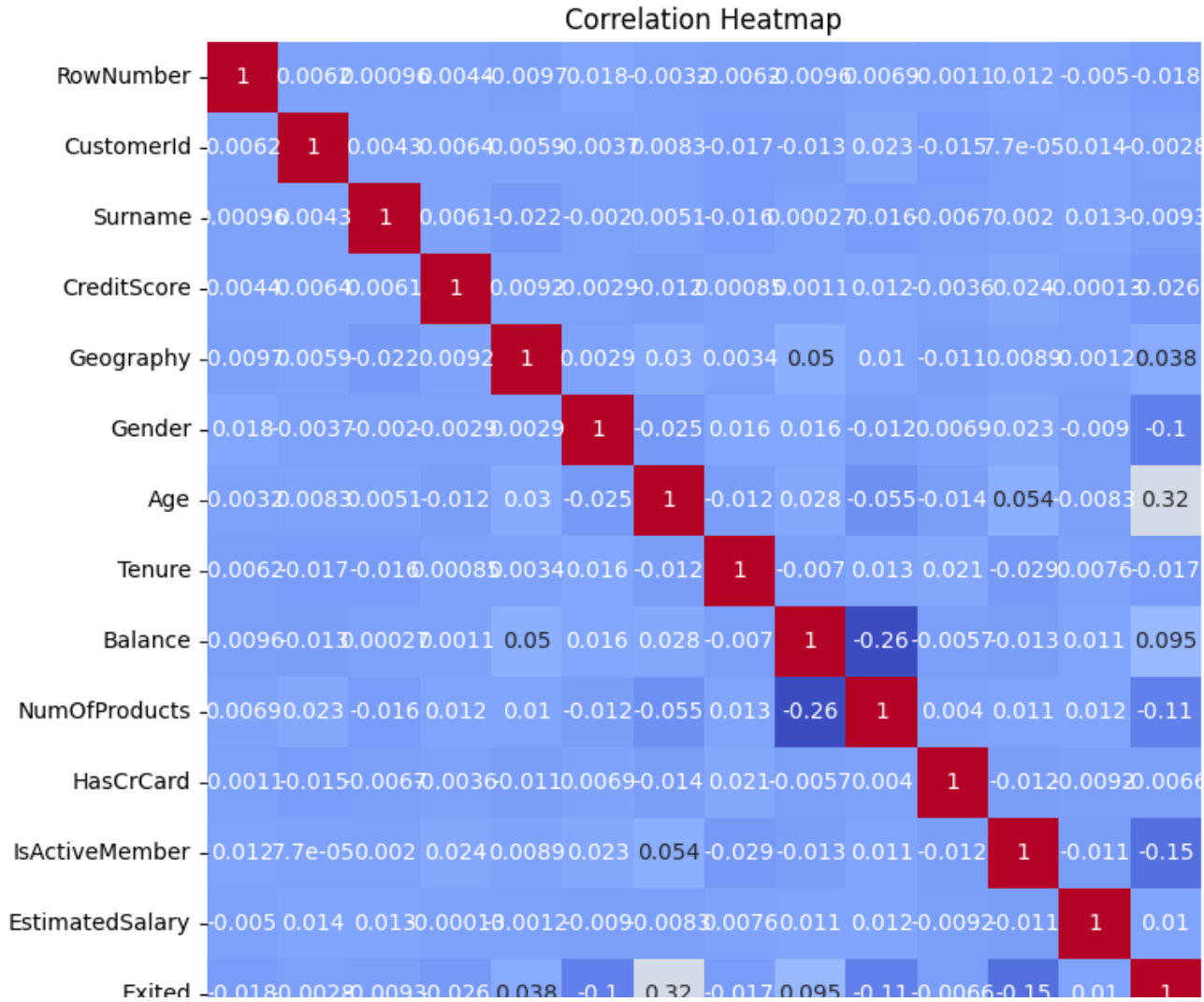
```
o_c=((new_df> 3) | (new_df < -3)).sum()
o_c
```

```
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
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Calculate correlations
correlation_matrix = cleaned_df.corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```




```
new_df=cleaned_df
```

```
df.head()
```

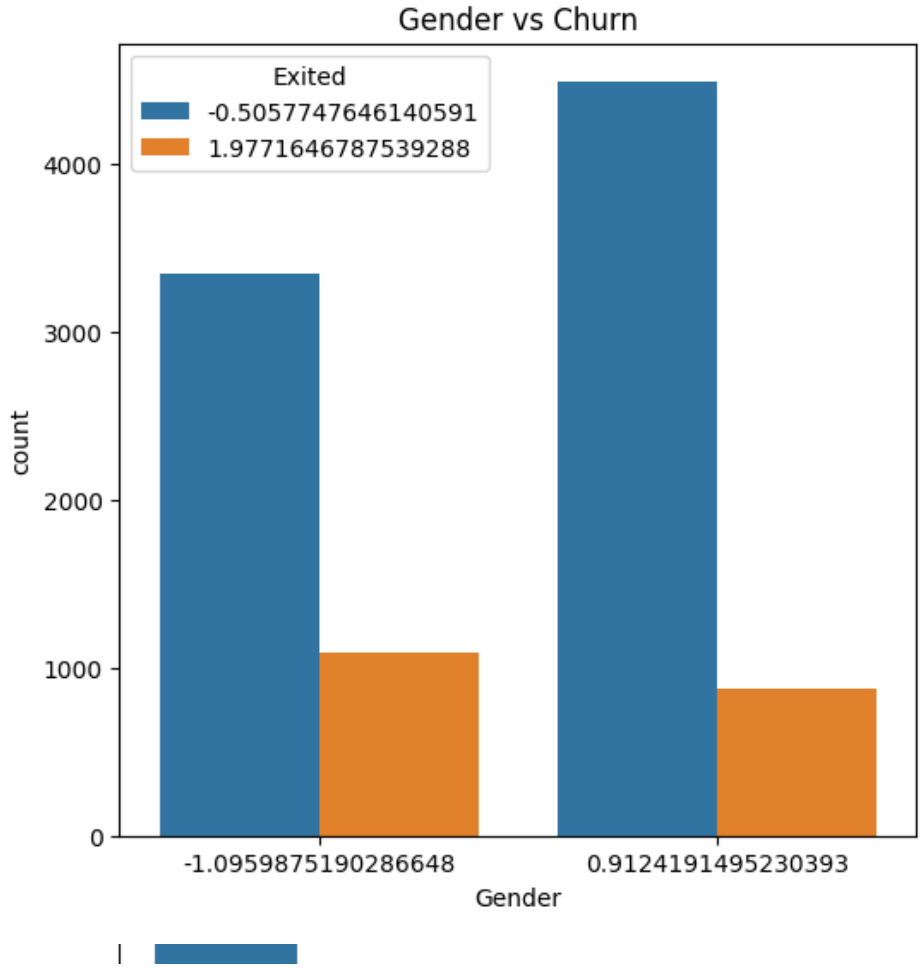
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts
0	0	2736	1115	228	0	0	24	2	0	0
1	1	3258	1177	217	2	0	23	1	743	0
2	2	2104	2040	111	0	0	24	8	5793	2
3	3	5435	289	308	0	0	21	1	0	1
4	4	6899	1822	459	2	0	25	2	3696	0

```
# Create count plot for categorical variables
categorical_vars = ["Gender", "Geography"]
for var in categorical_vars:
  plt.figure(figsize=(8, 6))
  sns.countplot(data=new_df, x=var, hue="Exited")
```

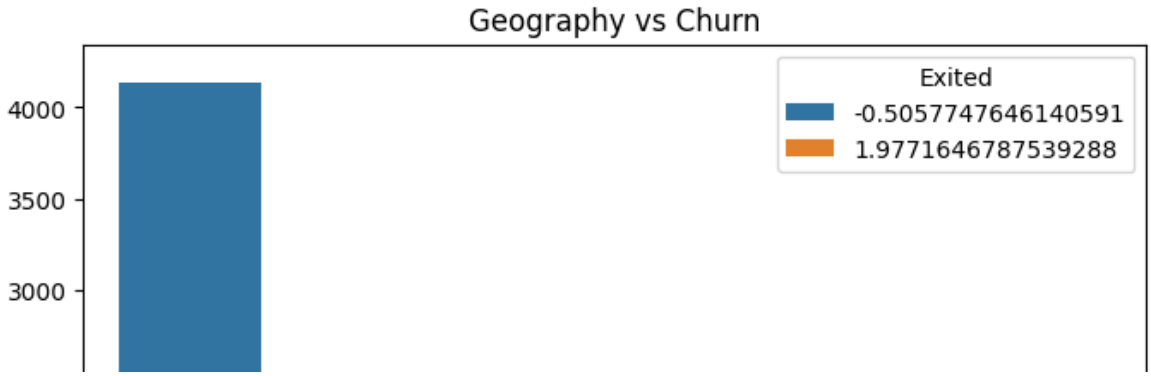
```
    plt.title(f"{var} vs Churn")
    plt.show()
```

Gender vs Churn

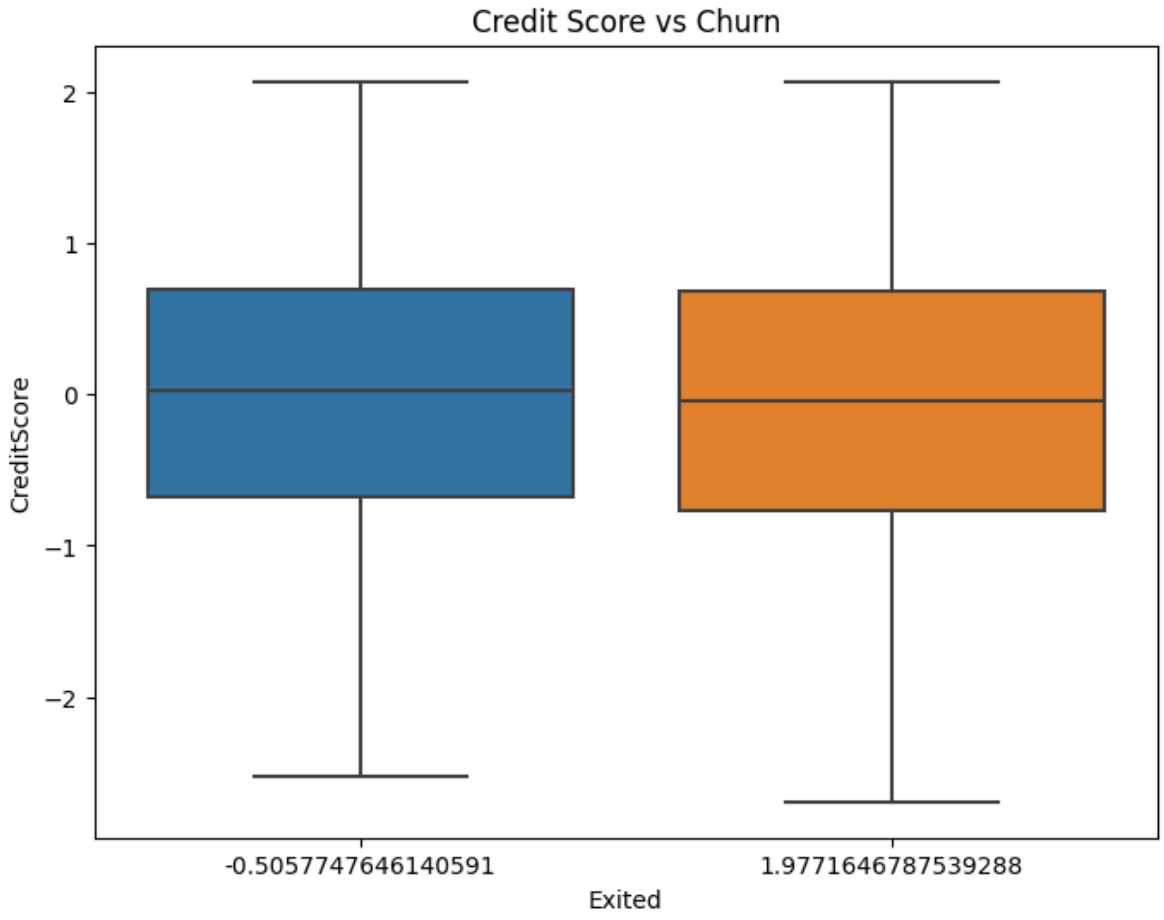
```
plt.figure(figsize=(6, 6))
sns.countplot(data=new_df, x="Gender", hue="Exited")
plt.title("Gender vs Churn")
plt.show()
```



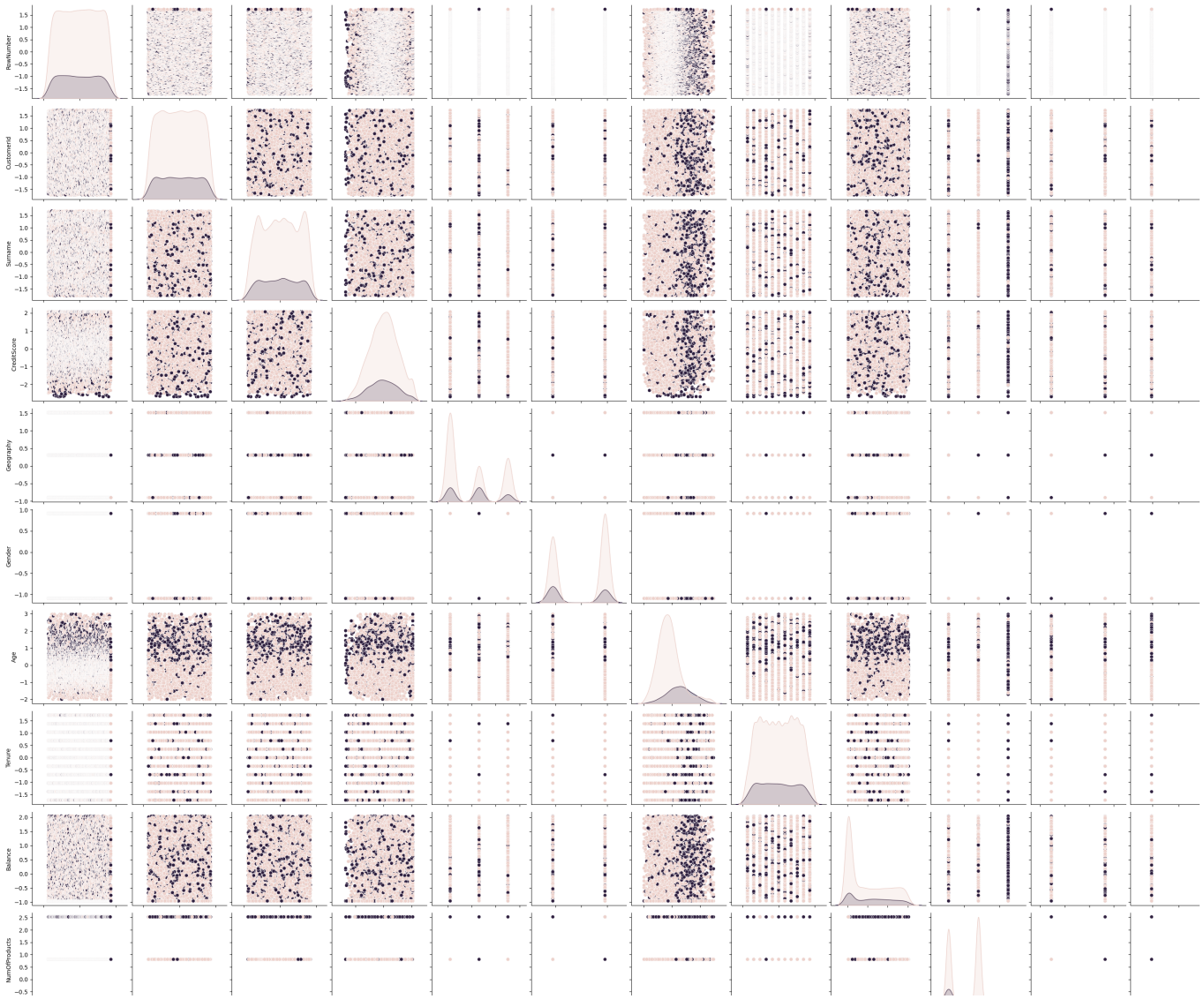

```
plt.figure(figsize=(8, 6))
sns.countplot(data=new_df, x="Geography", hue="Exited")
plt.title("Geography vs Churn")
plt.show()
```



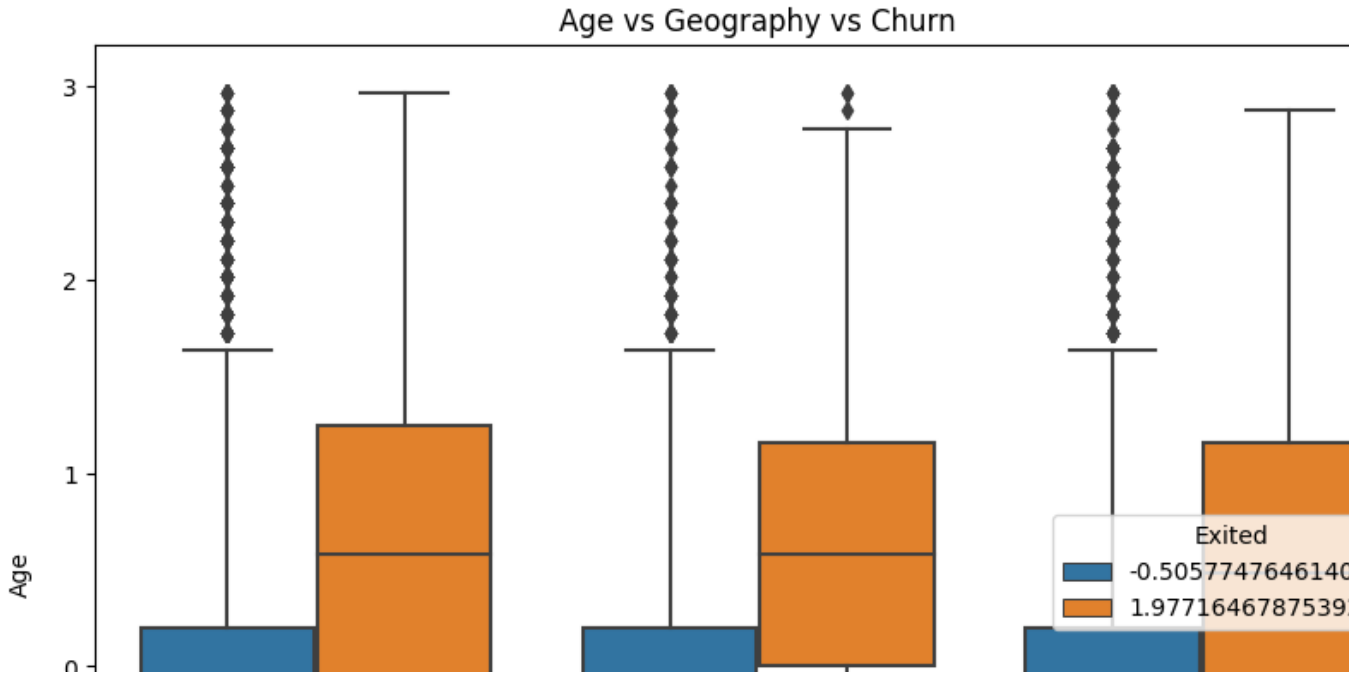

```
plt.figure(figsize=(8, 6))
sns.boxplot(data=new_df, x="Exited", y="CreditScore")
plt.title("Credit Score vs Churn")
plt.show()
```



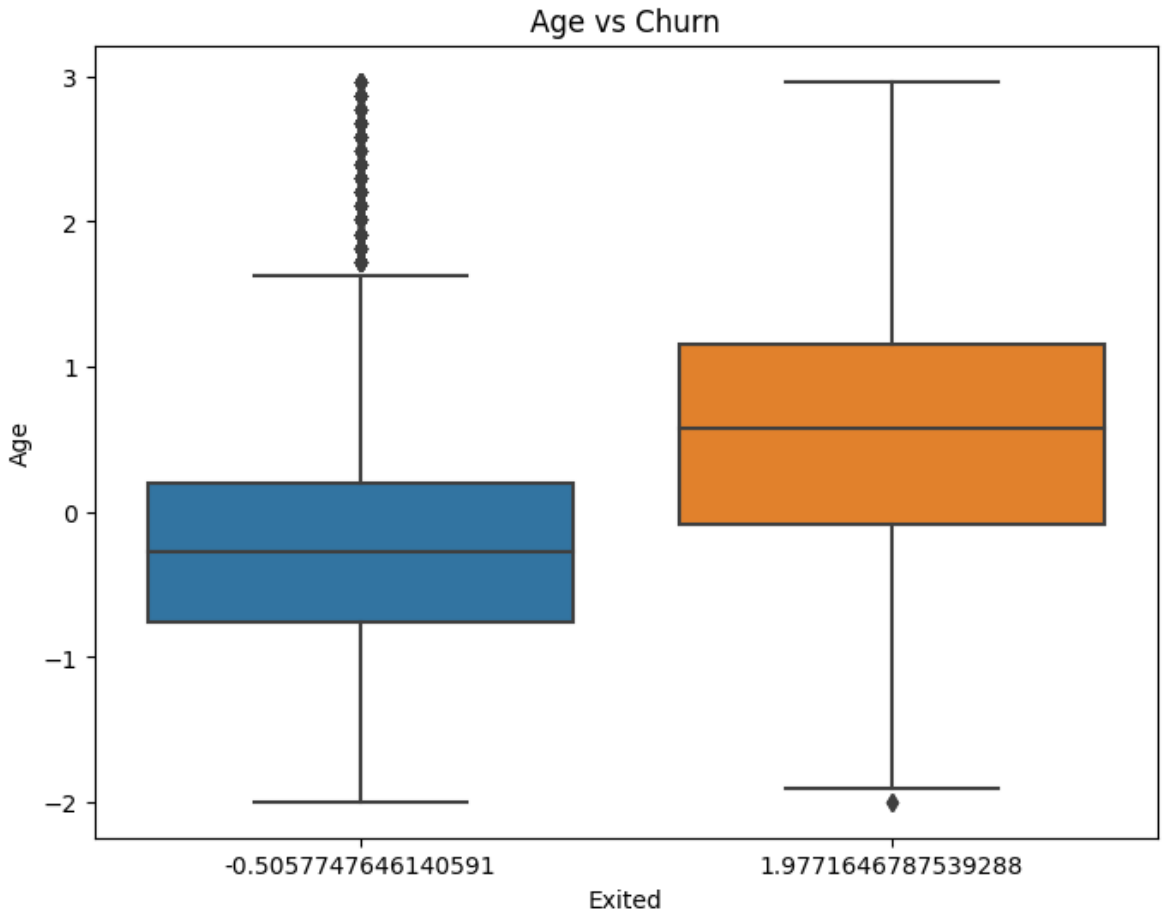

```
# Create pair plot
sns.pairplot(new_df, hue="Exited", diag_kind="kde")
plt.show()
```




```
# Create box plot or violin plot for numeric vs. categorical variables
numeric_var = "Age"
categorical_var = "Geography"
plt.figure(figsize=(10, 8))
sns.boxplot(data=new_df, x=categorical_var, y=numeric_var, hue="Exited")
plt.title(f"{numeric_var} vs {categorical_var} vs Churn")
plt.show()
```

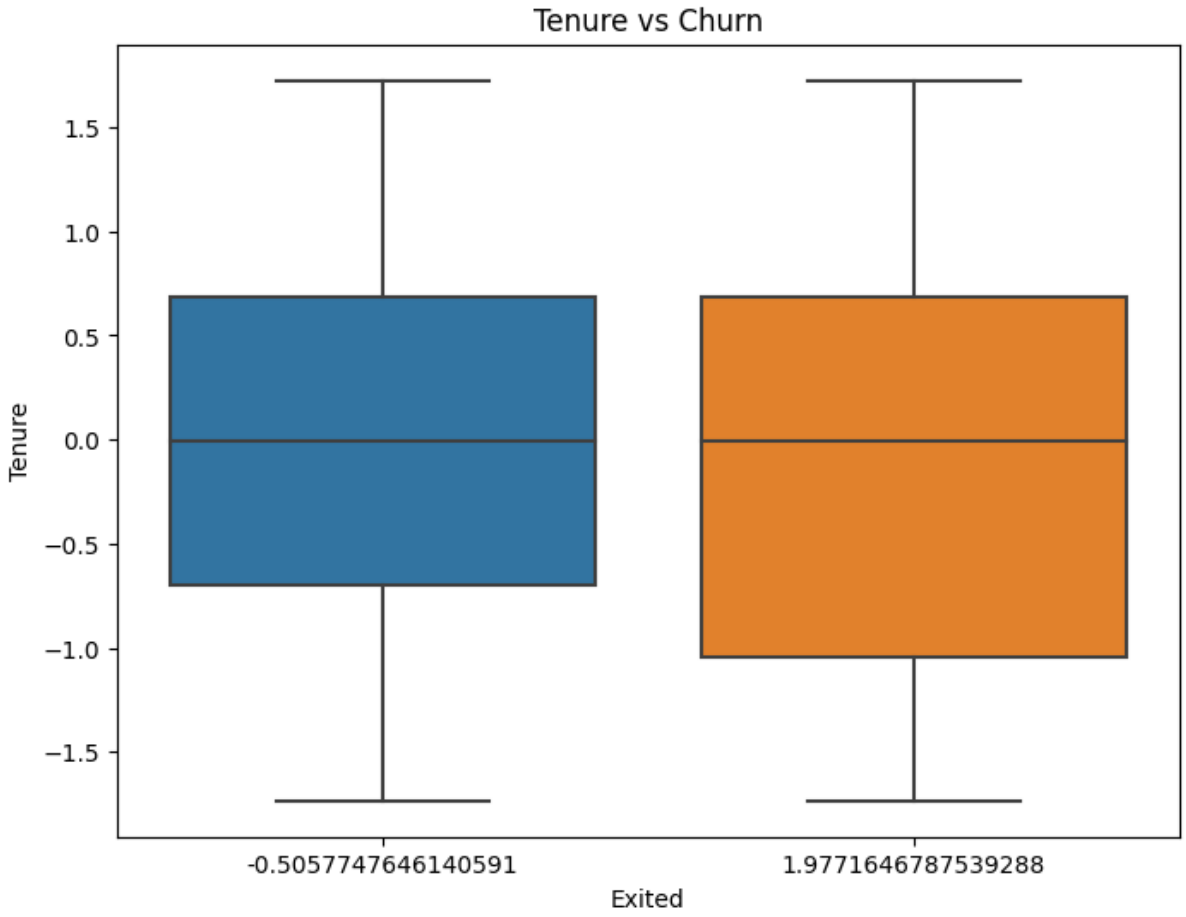



```
plt.figure(figsize=(8, 6))
sns.boxplot(data=new_df, x="Exited", y="Age")
plt.title("Age vs Churn")
plt.show()
```

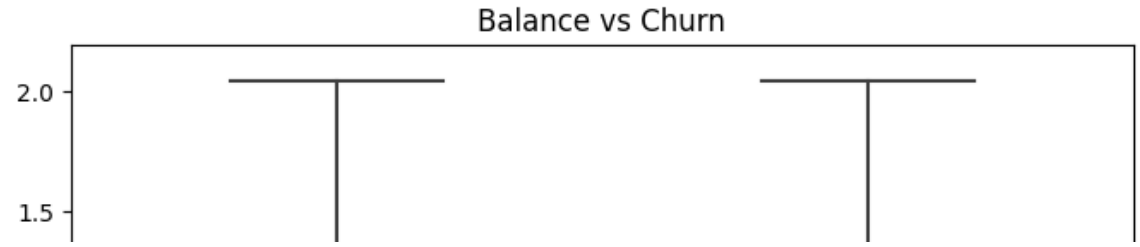



```
plt.figure(figsize=(8, 6))
sns.boxplot(data=new_df, x="Exited", y="Tenure")
```

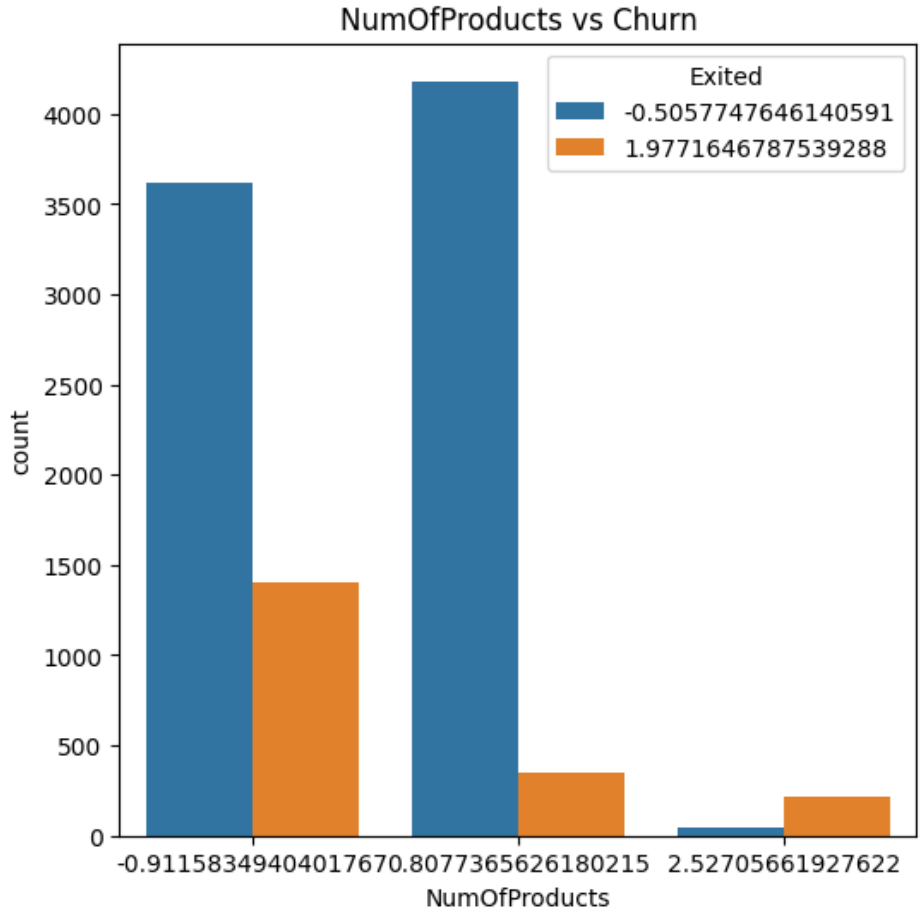
```
plt.title("Tenure vs Churn")
plt.show()
```



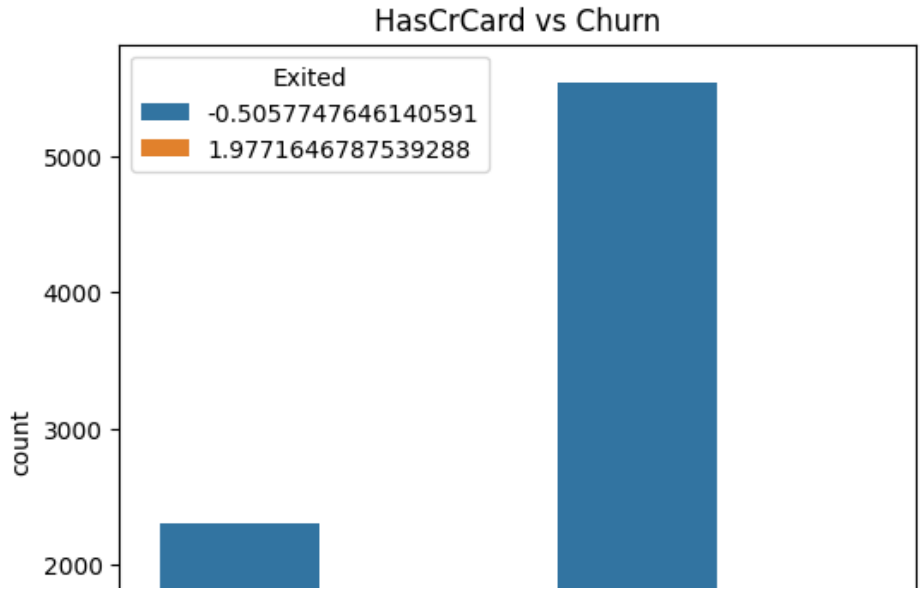

```
plt.figure(figsize=(8, 6))
sns.boxplot(data=new_df, x="Exited", y="Balance")
plt.title("Balance vs Churn")
plt.show()
```



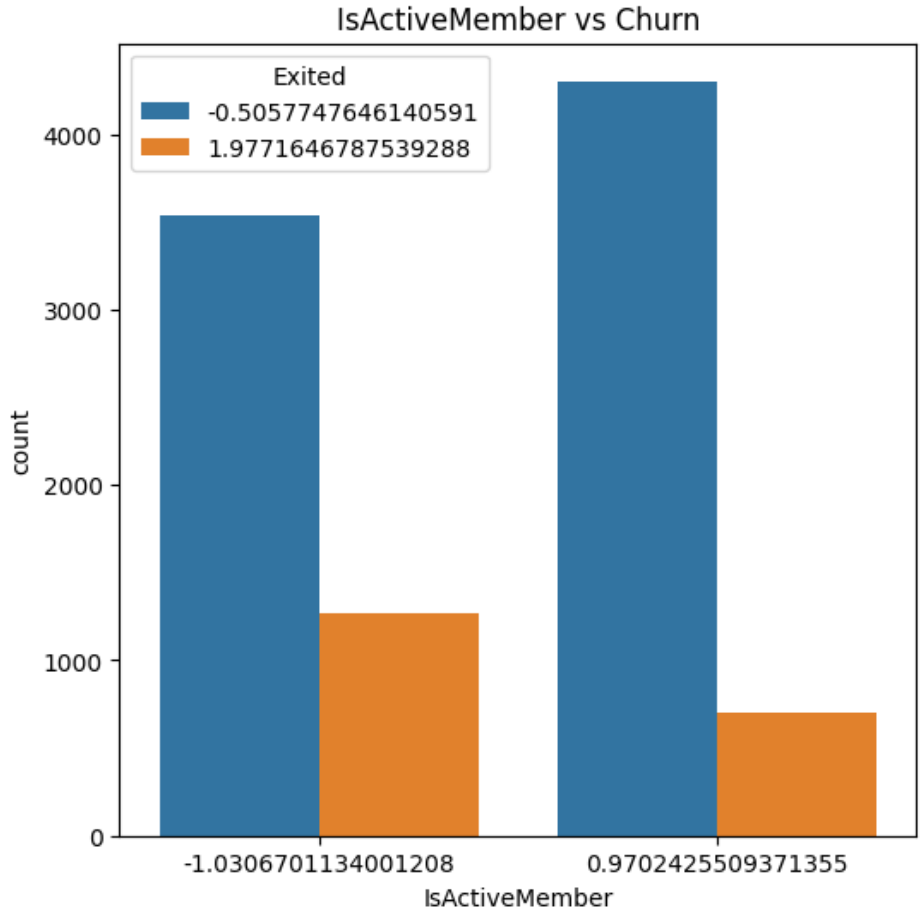

```
plt.figure(figsize=(6, 6))
sns.countplot(data=new_df, x="NumOfProducts", hue="Exited")
plt.title("NumOfProducts vs Churn")
plt.show()
```




```
plt.figure(figsize=(6, 6))
sns.countplot(data=new_df, x="HasCrCard", hue="Exited")
plt.title("HasCrCard vs Churn")
plt.show()
```




```
plt.figure(figsize=(6, 6))
sns.countplot(data=new_df, x="IsActiveMember", hue="Exited")
plt.title("IsActiveMember vs Churn")
plt.show()
```




```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA


s_col = ['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'EstimatedSalary']
```

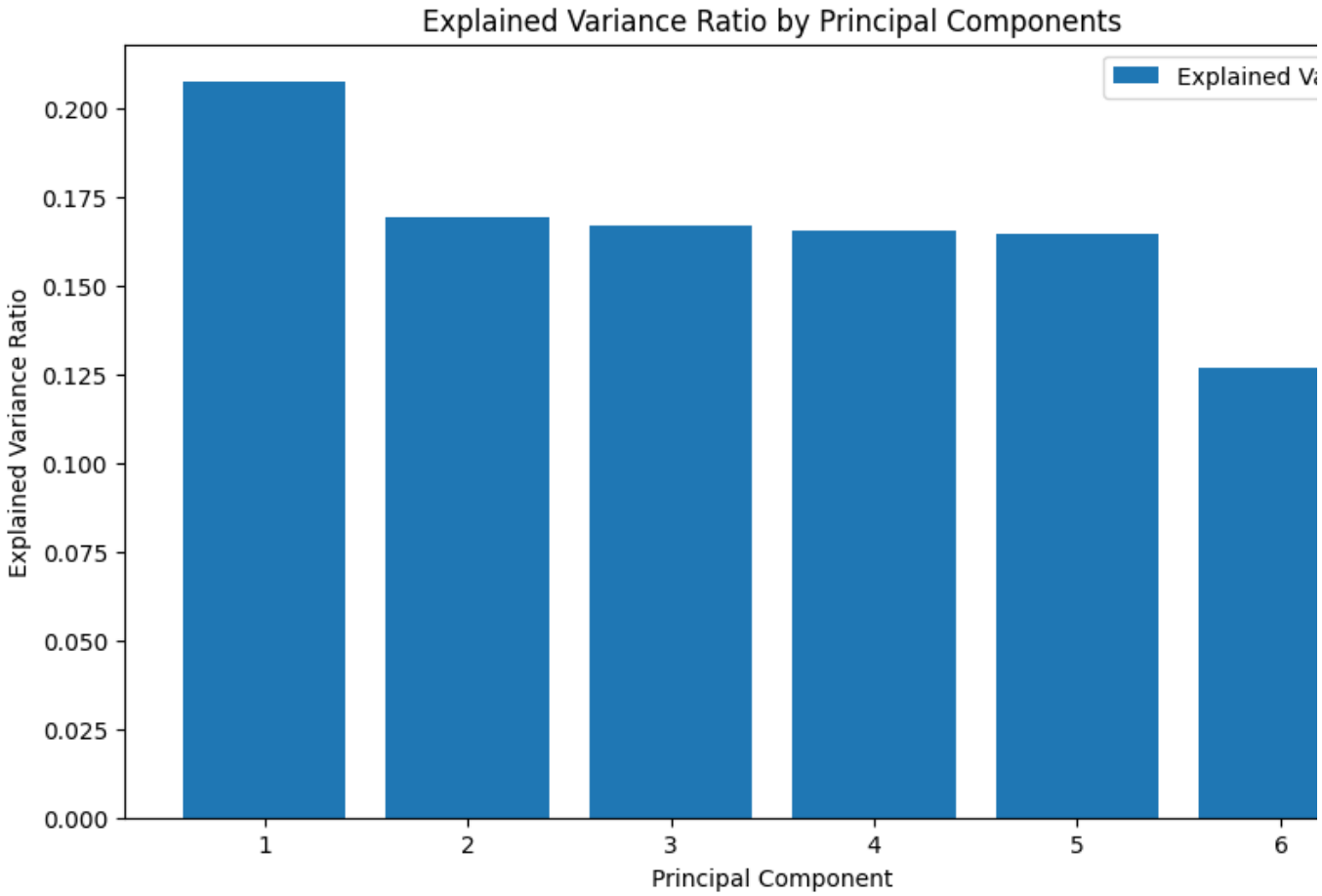
```
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df[s_col])


pca = PCA()
pca_result = pca.fit_transform(df_scaled)


explained_variance = pca.explained_variance_ratio_



plt.figure(figsize=(10, 6))
plt.bar(range(1, len(explained_variance) + 1), explained_variance, align='center', label='Explained Variance'
plt.xlabel('Principal Component')
plt.ylabel('Explained Variance Ratio')
plt.title('Explained Variance Ratio by Principal Components')
plt.legend()
plt.show()
```




```
for i, ratio in enumerate(explained_variance, start=1):
  print(f"Explained Variance for PC{i}: {ratio:.4f}")
```

```
Explained Variance for PC1: 0.2075
Explained Variance for PC2: 0.1690
Explained Variance for PC3: 0.1668
Explained Variance for PC4: 0.1654
Explained Variance for PC5: 0.1647
Explained Variance for PC6: 0.1266
```

```
n_components = 3


pca = PCA(n_components=n_components)
pca_result = pca.fit_transform(df_scaled)


pca_df = pd.DataFrame(data=pca_result, columns=[f'PC{i}' for i in range(1, n_components + 1)])


df_with_pca = pd.concat([df, pca_df], axis=1)


print(df_with_pca.head())
```

```
   RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age  \
0          0        2736     1115          228          0       0   24
1          1        3258     1177          217          2       0   23
2          2        2104     2040          111          0       0   24
3          3        5435      289          308          0       0   21
4          4        6899     1822          459          2       0   25

   Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary  \
0       2        0              0          1               1             5068
1       1      743              0          0               1             5639
2       8     5793              2          1               0             5707
3       1        0              1          0               0             4704
4       2     3696              0          1               1             3925

   Exited       PC1       PC2       PC3
0       1 -0.092565  0.868826  0.338661
1       0 -0.343014  0.863988  0.437003
2       1  0.524277 -0.620905  1.573100
3       0  1.154213  0.999448 -0.534307
4       0 -1.250275  0.690958 -2.127768
```

```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.preprocessing import KBinsDiscretizer

# Load your churn dataset (replace 'your_dataset.csv' with the actual filename)
churn = pd.read_csv('Churn_Modelling.csv')

# Extract the 'Age' column
X_age = churn[['Age']]

# Define the number of bins and strategy for binning
n_bins = 6
strategy = 'uniform'

# Create a KBinsDiscretizer instance
bin_encoder = KBinsDiscretizer(n_bins=n_bins, encode='ordinal', strategy=strategy)

# Fit and transform the 'Age' column using the bin encoder
X_age_binned = bin_encoder.fit_transform(X_age)

# Create a DataFrame to store the binned values with a meaningful column name
binned_df = pd.DataFrame(X_age_binned, columns=['Age_Binned'])

# Concatenate the original churn dataset with the binned values DataFrame
churn_binned = pd.concat([churn, binned_df], axis=1)
```

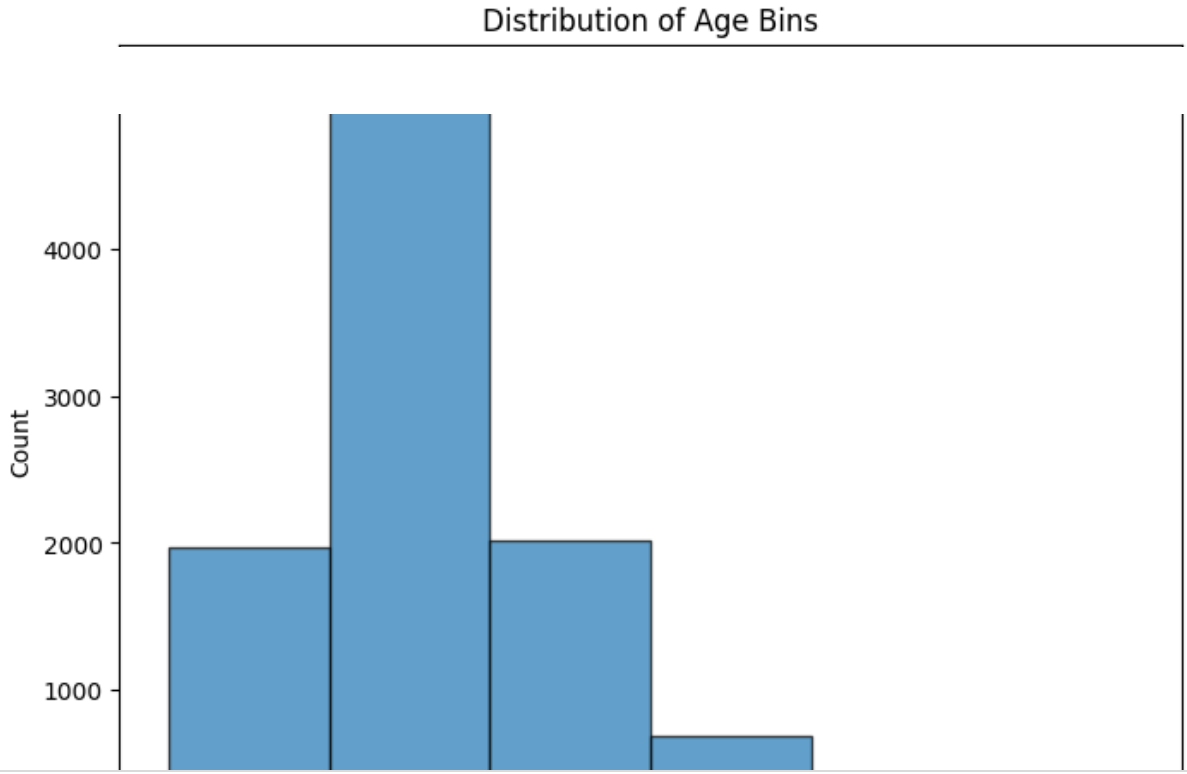
```
# Print the first few rows of the resulting DataFrame
print(churn_binned.head())
```

```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
0          1    15634602  Hargrave          619    France  Female   42
1          2    15647311      Hill          608     Spain  Female   41
2          3    15619304      Onio          502    France  Female   42
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited  Age_Binned
0        101348.88       1         1.0
1        112542.58       0         1.0
2        113931.57       1         1.0
3         93826.63       0         1.0
4         79084.10       0         2.0
```

```
plt.figure(figsize=(8, 6))
plt.hist(churn_binned['Age_Binned'], bins=n_bins, edgecolor='black', alpha=0.7)
plt.xlabel('Age Bins')
plt.ylabel('Count')
plt.title('Distribution of Age Bins')
plt.xticks(range(n_bins), [f'Bin {i}' for i in range(n_bins)])
plt.show()
```



✓ 0s completed at 11:06 PM