```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read_csv("Customer-Churn-Records.csv")

df
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Te
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	
...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	
9996	9997	15569892	Johnstone	516	France	Male	35	
9997	9998	15584532	Liu	709	France	Female	36	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	
9999	10000	15628319	Walker	792	France	Female	28	

10000 rows × 18 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

Q. How many rows and columns are there i

Q. How many rows and columns are there in the dataset?

```
##number of rows and columns
number_of_rows = len(df)
number_of_columns = len(df.columns)
print("number of rows :",number_of_rows)
print("number of columns :",number_of_columns)
```

```
number of rows : 10000
number of columns : 18
```

Q. Check the percentage of missing data and handle accordingly.

```
missing_percentage = df.isnull().mean() * 100
print(missing_percentage)
```

```
RowNumber            0.0
CustomerId           0.0
Surname              0.0
CreditScore          0.0
Geography            0.0
Gender               0.0
Age                  0.0
Tenure               0.0
Balance              0.0
NumOfProducts        0.0
HasCrCard            0.0
IsActiveMember       0.0
EstimatedSalary      0.0
Exited               0.0
Complain             0.0
Satisfaction Score   0.0
Card Type            0.0
Point Earned         0.0
dtype: float64
```

```
# Columns with missing data
print(missing_percentage[missing_percentage > 0])
```

```
Series([], dtype: float64)
```

```
#Handling missing data
# Drop columns with more than 50% missing
df = df.dropna(thresh=len(df)*0.5, axis=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 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
 14  Complain            10000 non-null  int64
 15  Satisfaction Score  10000 non-null  int64
 16  Card Type           10000 non-null  object
 17  Point Earned        10000 non-null  int64
dtypes: float64(2), int64(12), object(4)
memory usage: 1.4+ MB
```

```
# no missing values
print(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
Complain             0
Satisfaction Score   0
Card Type            0
Point Earned         0
dtype: int64
```

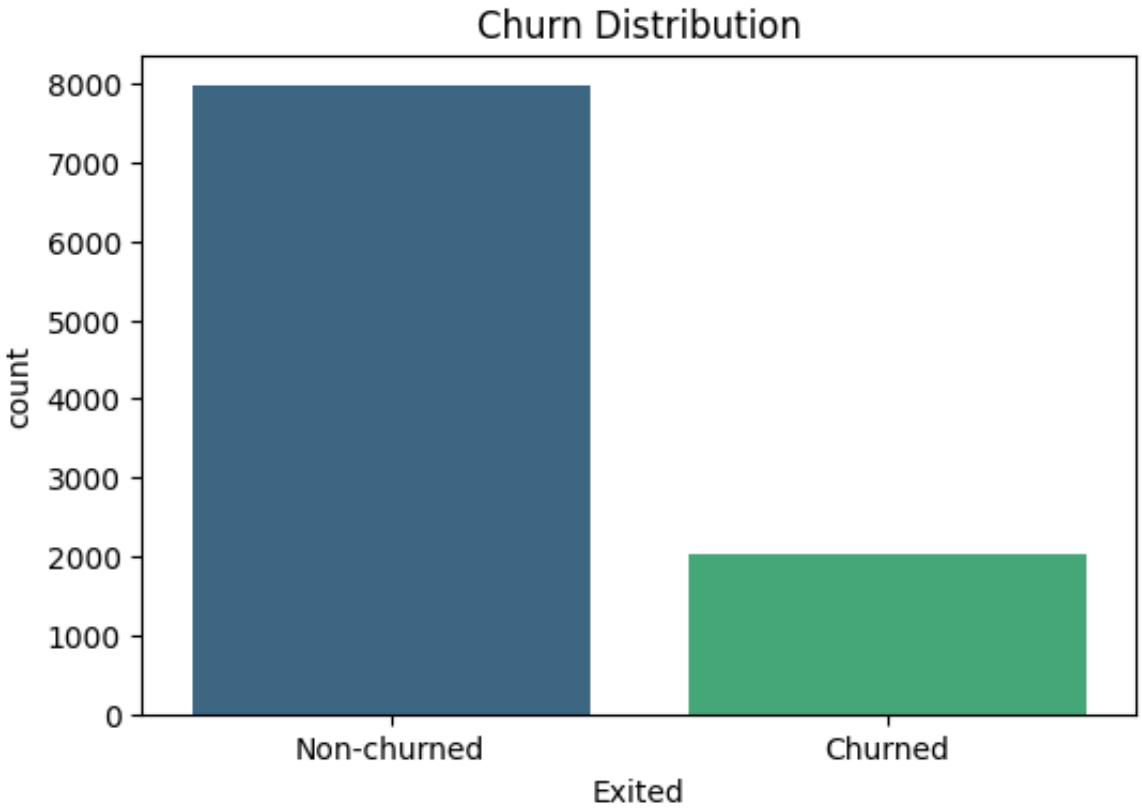
```
# percentage of missing values
percentage_missing_per_column = (df.isnull().sum() / len(df)) * 100
print("Percentage of missing values per column:")
print(percentage_missing_per_column)
```

```
Percentage of missing values per column:
RowNumber            0.0
CustomerId           0.0
Surname              0.0
CreditScore          0.0
Geography            0.0
Gender               0.0
Age                  0.0
Tenure               0.0
Balance              0.0
NumOfProducts        0.0
HasCrCard            0.0
IsActiveMember       0.0
EstimatedSalary      0.0
Exited               0.0
Complain             0.0
Satisfaction Score   0.0
Card Type            0.0
Point Earned         0.0
dtype: float64
```

Q. What is the distribution of churned vs. non-churned customers?

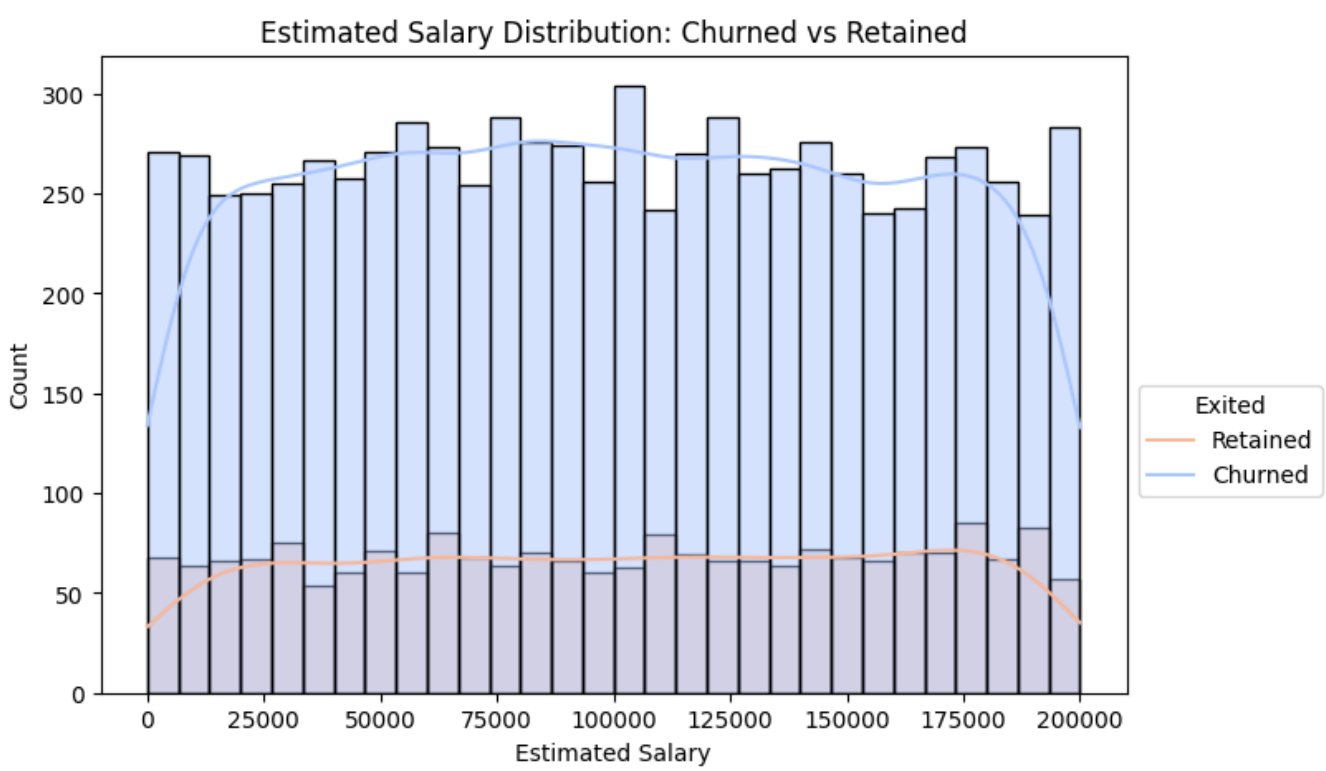
```
# Plot churn distribution
plt.figure(figsize=(6,4))
sns.countplot(data=df, x='Exited', palette='viridis')
plt.xticks([0,1], ['Non-churned', 'Churned'])
plt.title('Churn Distribution')
plt.show()
```

```
/tmp/ipython-input-2584591398.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed

  sns.countplot(data=df, x='Exited', palette='viridis')
```



Q. What is the distribution of EstimatedSalary of churned and retained Customers?

```
# Estimated Salary Distribution: Churned vs Retained
plt.figure(figsize=(8,5))
sns.histplot(data=df, x='EstimatedSalary', hue='Exited', bins=30, kde=True, pal
plt.title('Estimated Salary Distribution: Churned vs Retained')
plt.xlabel('Estimated Salary')
plt.ylabel('Count')
plt.legend(title='Exited', labels=['Retained', 'Churned'],bbox_to_anchor=(1, 0.
plt.show()
```

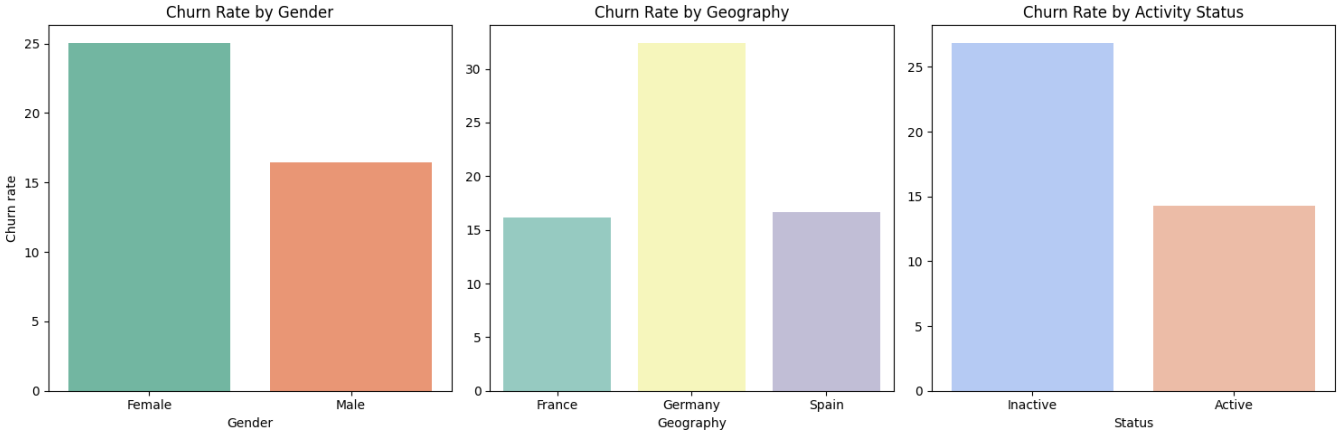


Q. How do churn rates vary by Gender, Geography, and IsActiveMember?

```
# churn rates by gender geography and active member
churn_by_gender = df.groupby('Gender')['Exited'].mean() * 100
churn_by_geo = df.groupby('Geography')['Exited'].mean() * 100
churn_by_active = df.groupby('IsActiveMember')['Exited'].mean() * 100

fig, axes = plt.subplots(1, 3, figsize=(15,5))
sns.barplot(x=churn_by_gender.index, y=churn_by_gender.values, palette='Set2',
axes[0].set_title('Churn Rate by Gender')
axes[0].set_ylabel('Churn rate')
```

```
axes[0].set_xlabel('Gender')

sns.barplot(x=churn_by_geo.index, y=churn_by_geo.values, palette='Set3', ax=axe
axes[1].set_title('Churn Rate by Geography')
axes[1].set_xlabel('Geography')

labels = ['Inactive', 'Active']
sns.barplot(x=labels, y=churn_by_active.values, palette='coolwarm', ax=axes[2])
axes[2].set_title('Churn Rate by Activity Status')
axes[2].set_xlabel('Status')

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

```
/tmp/ipython-input-1238498722.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed

  sns.barplot(x=churn_by_gender.index, y=churn_by_gender.values, palette='S
/tmp/ipython-input-1238498722.py:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed

  sns.barplot(x=churn_by_geo.index, y=churn_by_geo.values, palette='Set3',
/tmp/ipython-input-1238498722.py:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed

  sns.barplot(x=labels, y=churn_by_active.values, palette='coolwarm', ax=ax
```



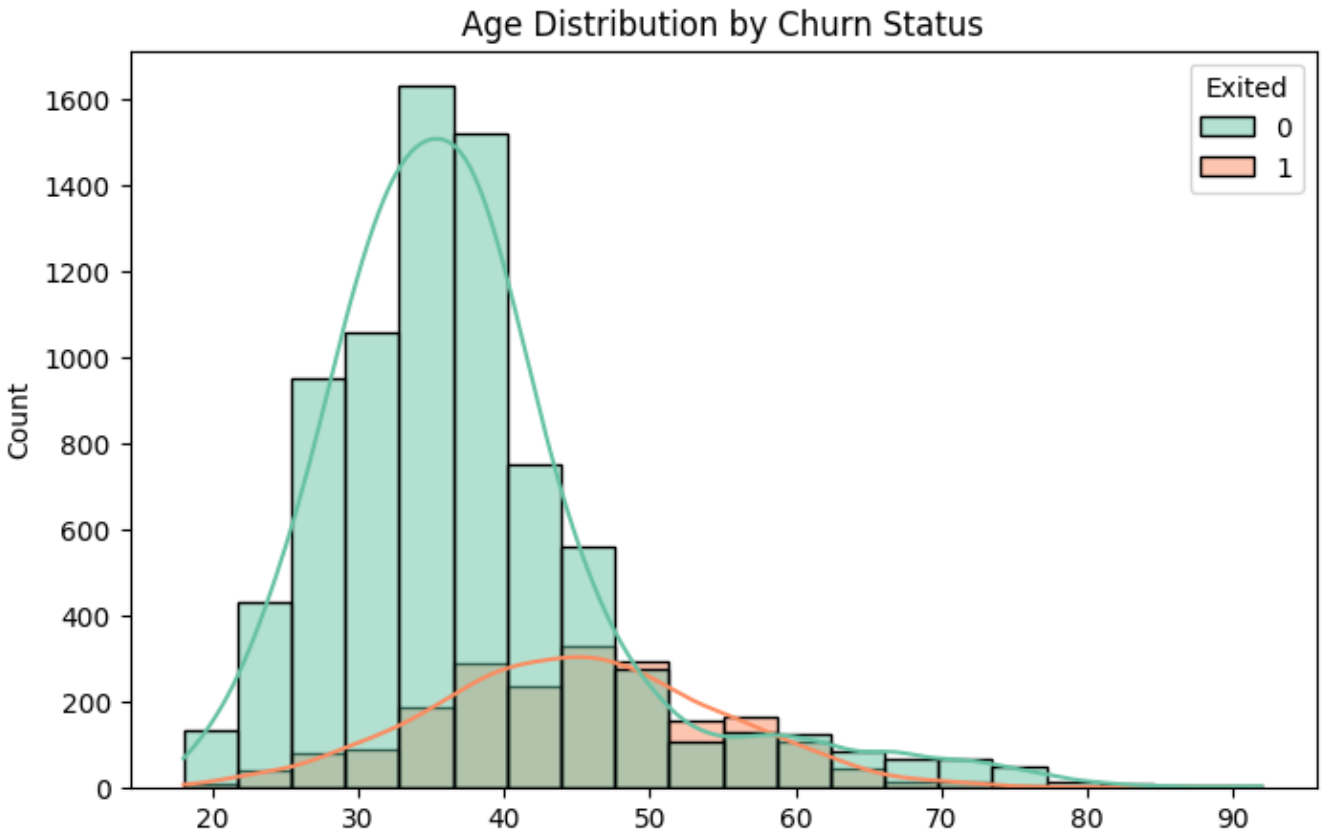
Q. What is the average CreditScore, Balance, and EstimatedSalary of churned vs. retained customers?

```
# average CreditScore, Balance, and EstimatedSalary for churned vs retained cus
avg_metrics = df.groupby('Exited')[['CreditScore', 'Balance', 'EstimatedSalary'
print("Average CreditScore, Balance & EstimatedSalary (Churned vs Retained):\n"
print(avg_metrics)
```

```
Average CreditScore, Balance & EstimatedSalary (Churned vs Retained):

        CreditScore       Balance  EstimatedSalary
Exited
0        651.837855  72742.750663     99726.853141
1        645.414622  91109.476006    101509.908783
```

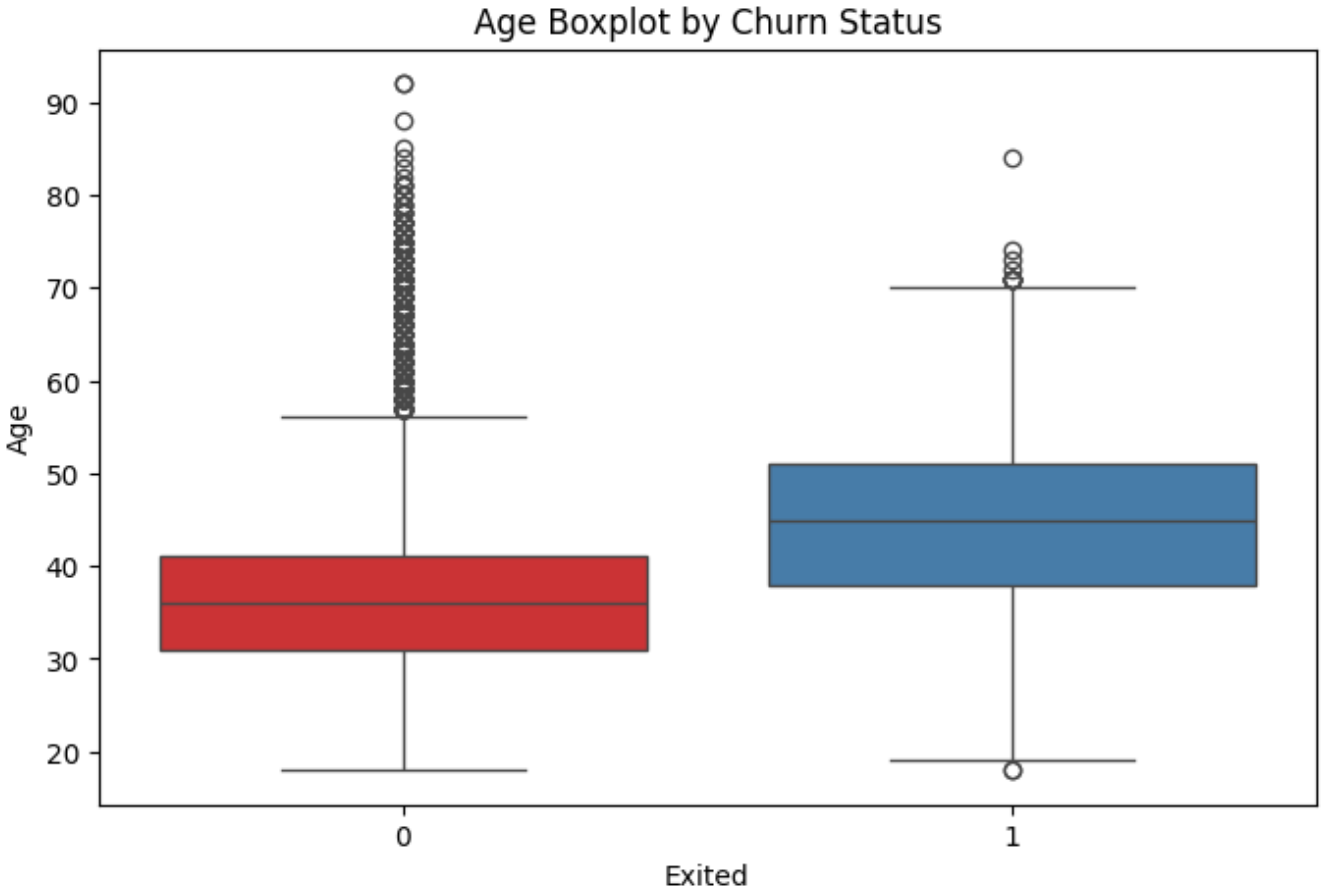
Q. How does Age impact churn? Plot histograms and boxplots for churned and non-churned groups

```
# age impact churn
plt.figure(figsize=(8,5))
sns.histplot(data=df, x='Age', hue='Exited', kde=True, bins=20, palette='Set2')
plt.title('Age Distribution by Churn Status')
plt.show()

plt.figure(figsize=(8,5))
sns.boxplot(data=df, x='Exited', y='Age', palette='Set1')
plt.title('Age Boxplot by Churn Status')
plt.show()
```



Age

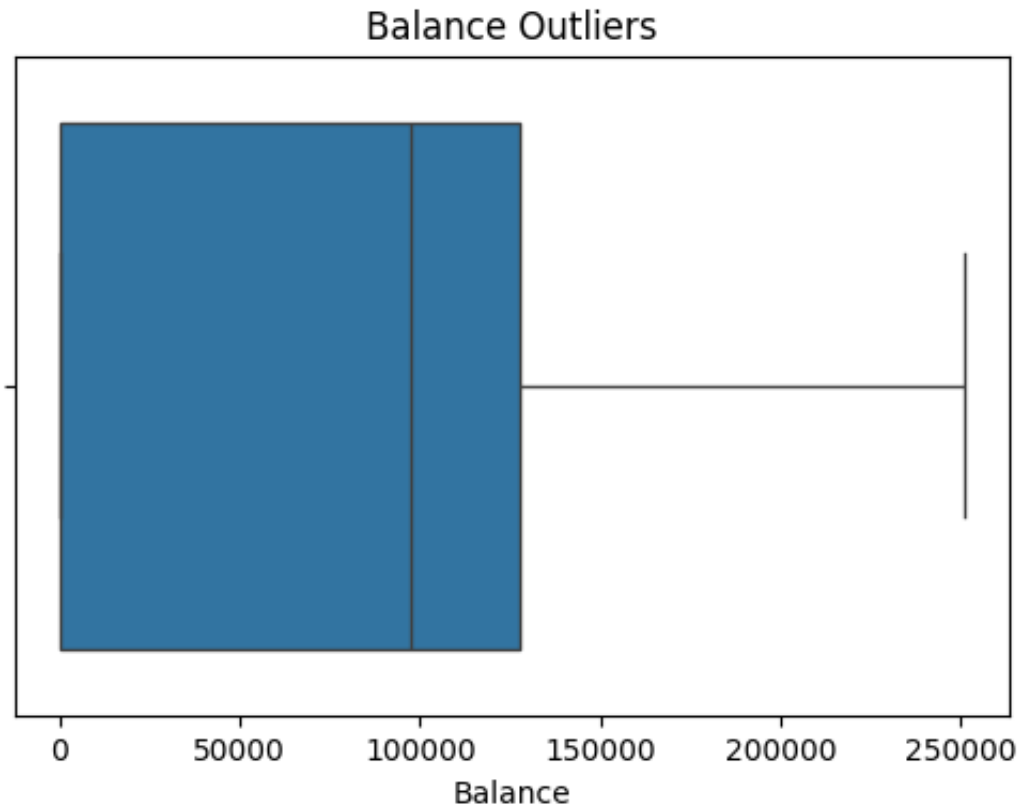
```
/tmp/ipython-input-2355977733.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed

  sns.boxplot(data=df, x='Exited', y='Age', palette='Set1')
```

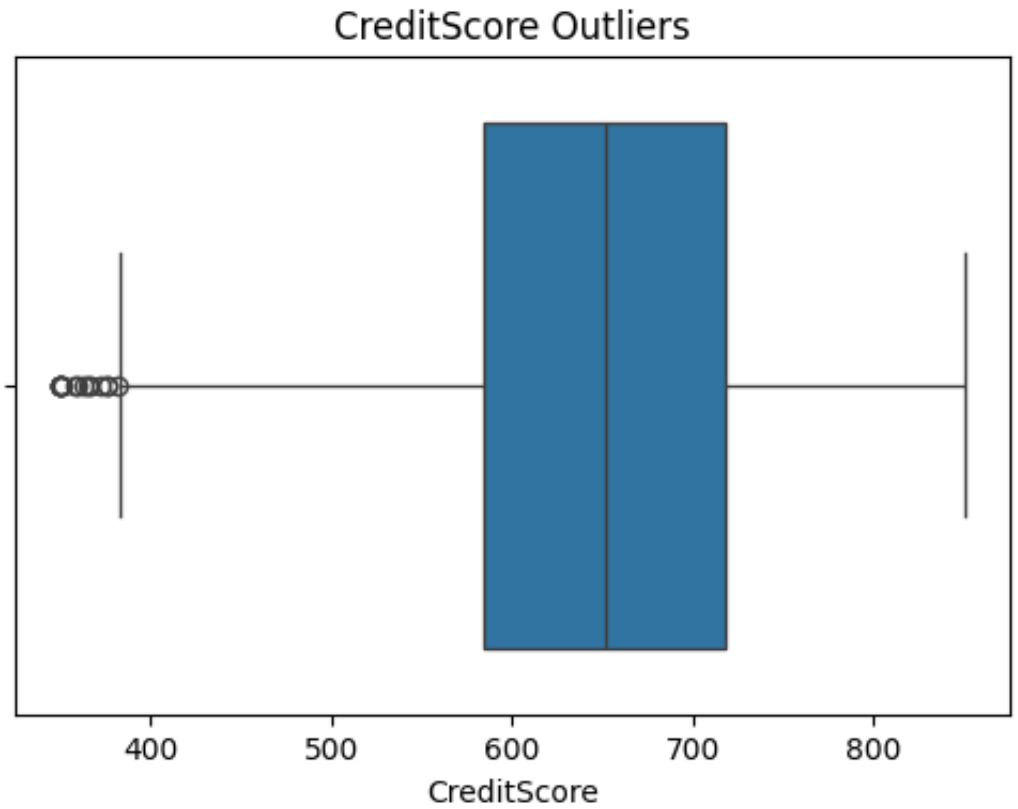


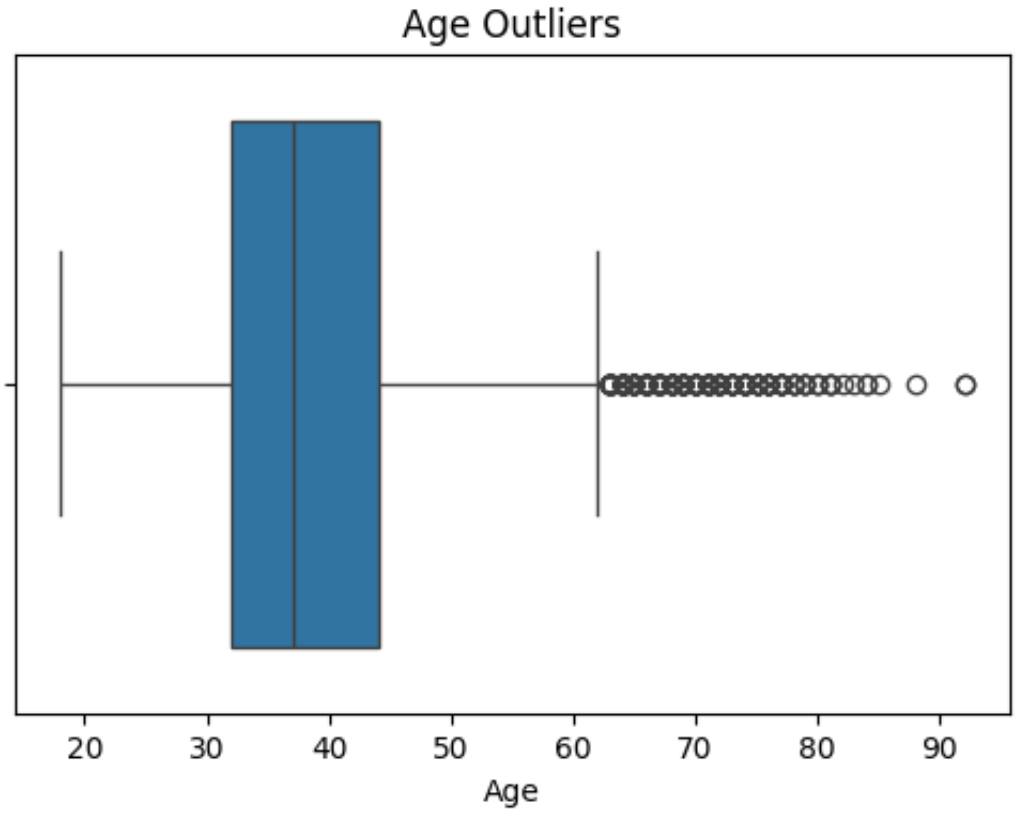
Q. Are there outliers in Balance, CreditScore, or Age that are mostly associated with churn?

```
#Outliers
plt.figure(figsize=(6,4))
sns.boxplot(data=df, x='Balance')
plt.title('Balance Outliers')
plt.show()

plt.figure(figsize=(6,4))
sns.boxplot(data=df, x='CreditScore')
plt.title('CreditScore Outliers')
```

```
plt.show()

plt.figure(figsize=(6,4))
sns.boxplot(data=df, x='Age')
plt.title('Age Outliers')
plt.show()
```

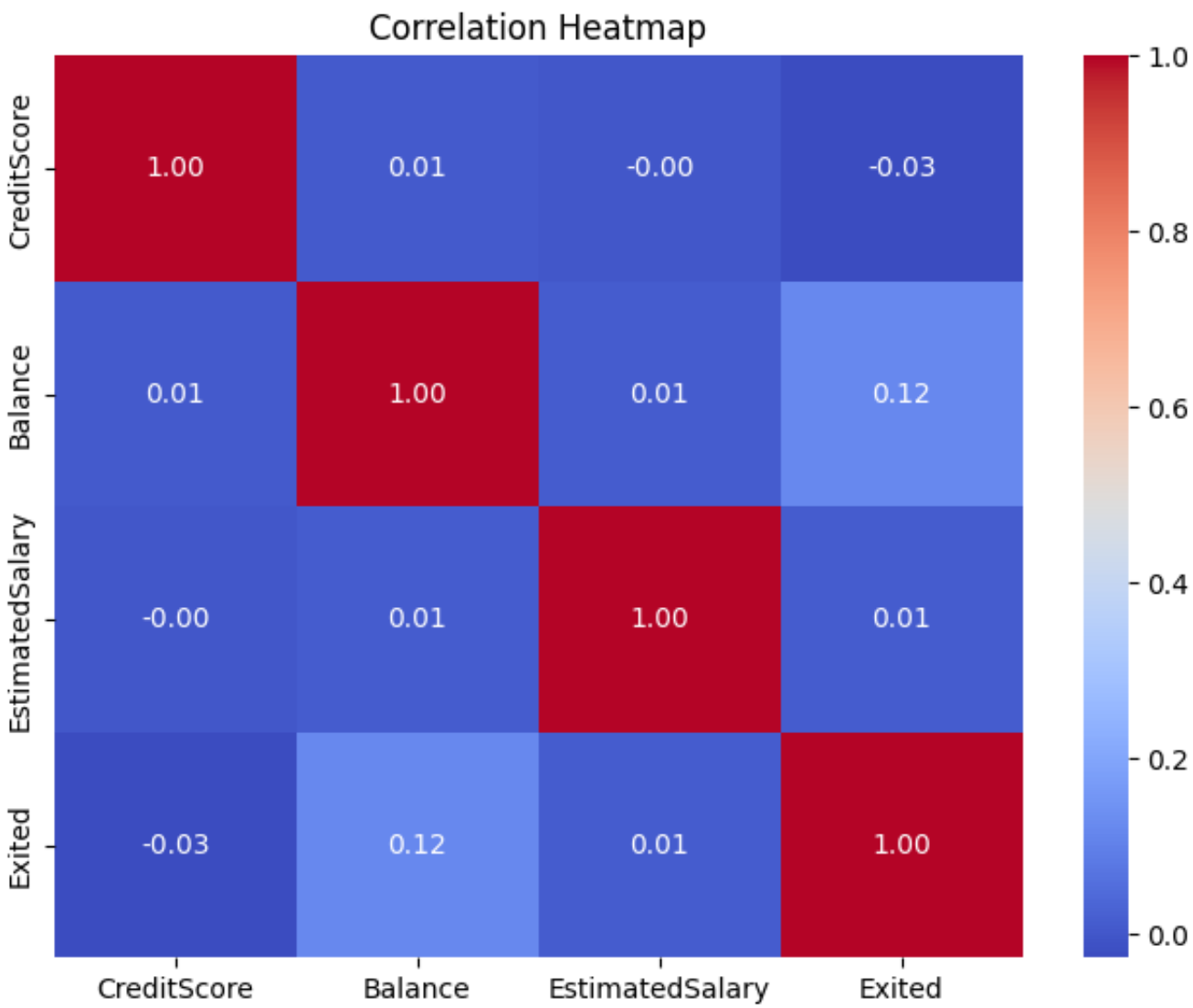






Q. Is there any correlation among numeric features like CreditScore, Balance, and EstimatedSalary?

```
#correlation between numeric feature
numeric_cols = ['CreditScore', 'Balance', 'EstimatedSalary','Exited']
corr = df[numeric_cols].corr()

plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



Q. What does a heatmap reveal about feature interactions with churn?

Answer: As per heatMap , numerical values don't have any correlations

Q. Group customers into age brackets (e.g.,18-30 as Adults, 30-50 as middle age and 50-100 as seniors.). How does churn rate change across them?

```
# churn rate by age group
bins = [18, 30, 50, 100]
labels = ['Adults', 'Middle Age', 'Seniors']
df['AgeGroup'] = pd.cut(df['Age'], bins=bins, labels=labels)
churn_by_agegroup = df.groupby('AgeGroup',observed=True)['Exited'].mean() * 100
print("\nChurn Rate by Age Group (%):\n", churn_by_agegroup)
```

```
Churn Rate by Age Group (%):
 AgeGroup
Adults         7.502569
Middle Age    19.598287
Seniors       44.647105
Name: Exited, dtype: float64
```

Q. Are customers with only one product (NumOfProducts = 1) more likely to churn than those with multiple?

```
# Churn Rate by Number of Products
single_product_churn = df[df['NumOfProducts'] == 1]['Exited'].mean() * 100
multi_product_churn = df[df['NumOfProducts'] > 1]['Exited'].mean() * 100
print(f"\nChurn Rate (1 Product): {single_product_churn:.2f}%")
print(f"Churn Rate (>1 Product): {multi_product_churn:.2f}%")
```

```
Churn Rate (1 Product): 27.71%
Churn Rate (>1 Product): 12.79%
```

Start coding or generate with AI.