Question 4 Perform EDA, Correlation Analysis, Customer Churn Analysis

In [2]:

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

Load the dataset

In [4]:

```
data = pd.read_csv('Churn_Modelling.csv')
```

Explore the dataset

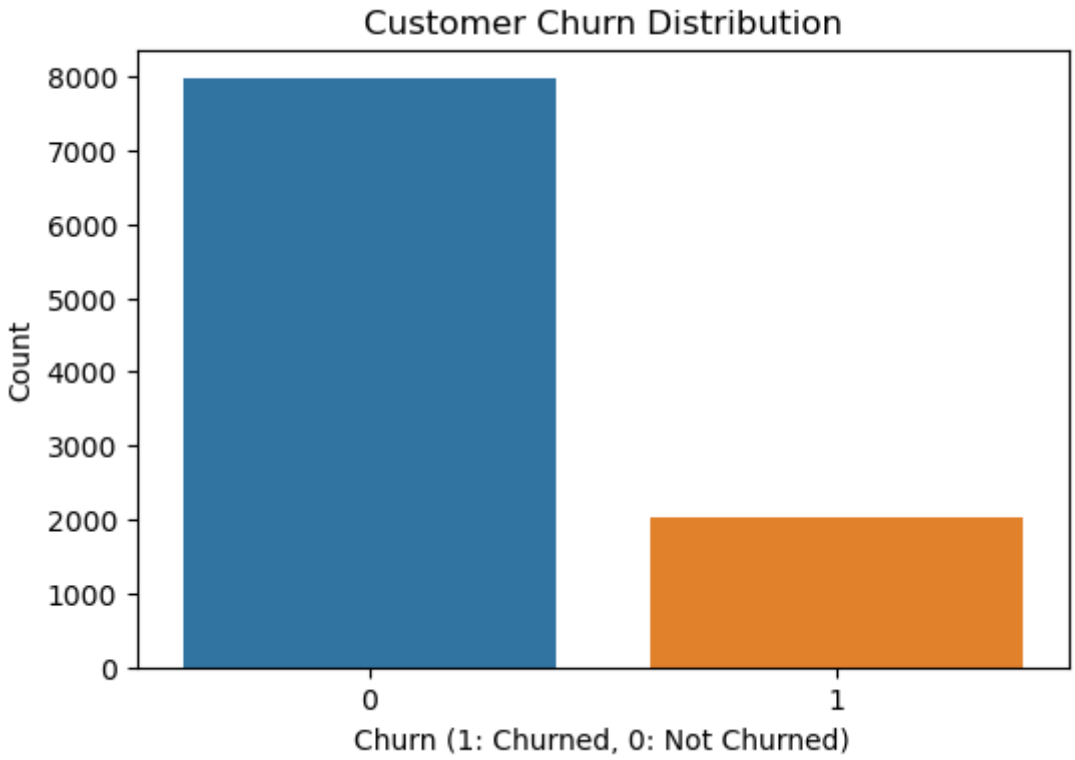
In [ ]:

```
print(data.head())  # Display the first few rows of the dataset
print(data.info())  # Get information about the dataset
print(data.describe())  # Get statistical summary of the numerical columns
```

Visualize the distribution of the target variable (Churn)

In [5]:

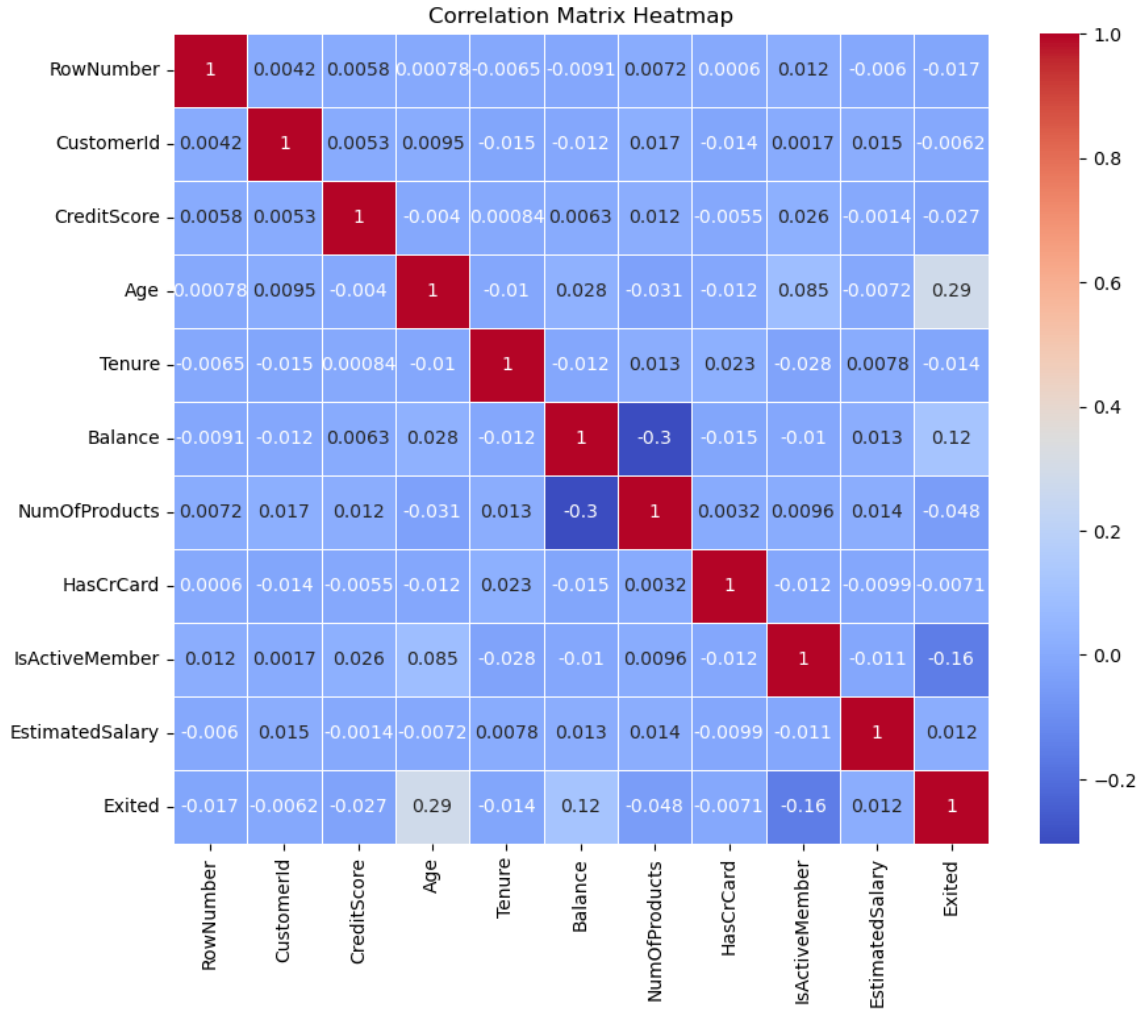
```
plt.figure(figsize=(6, 4))
sns.countplot(x='Exited', data=data)
plt.title('Customer Churn Distribution')
plt.xlabel('Churn (1: Churned, 0: Not Churned)')
plt.ylabel('Count')
plt.show()
```



Correlation Analysis

In [7]:

```
corr_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



Customer Churn Analysis

Select relevant features (e.g., CreditScore, Age, Tenure, NumOfProducts, IsActiveMember, EstimatedSalary)

In [8]:

```
features = ['CreditScore', 'Age', 'Tenure', 'NumOfProducts', 'IsActiveMember', 'Estimated
```

Split data into features (X) and target (y)

In [9]:

```
X = data[features]
y = data['Exited']
```

Split data into training and testing sets

In [10]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Build and Train a Machine Learning Model (Logistic Regression example)

In [11]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [12]:

```
model = LogisticRegression(random_state=42)
model.fit(X_train, y_train)
```

Out[12]:

```
LogisticRegression(random_state=42)
```

Make predictions

In [14]:

```
y_pred = model.predict(X_test)
```

Evaluate the model

In [15]:

```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```

```
Accuracy: 0.7975
Confusion Matrix:
[[1575   32]
 [ 373   20]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.98      0.89      1607
           1       0.38      0.05      0.09       393

    accuracy                           0.80      2000
   macro avg       0.60      0.52      0.49      2000
weighted avg       0.73      0.80      0.73      2000
```

In [ ]: