UNDERSTANDING AND PREDICTING BANK CUSTOMER ATTRITION

AIM OF THE PROJECT:

The aim of this project is to conduct a comprehensive analysis of bank customer churn in order to understand the underlying patterns, behaviors, and factors that contribute to customer attrition. By leveraging historical customer data and applying exploratory data analysis, statistical techniques, and machine learning models, the project seeks to predict which customers are at the highest risk of leaving the bank. The ultimate goal is to enable the bank to implement targeted retention strategies, improve customer satisfaction, and minimize revenue loss.

BACKGROUND

In today's competitive financial sector, customer retention is a critical factor for the long-term success of banks. With multiple banking options available, customers can easily switch to competitors that offer better services, lower fees, or more attractive benefits. This movement of customers from one bank to another, known as customer churn or attrition, can significantly impact a bank's profitability.

Research shows that acquiring a new customer can cost several times more than retaining an existing one, making churn management an essential business strategy. High churn rates not only reduce the customer base but also result in the loss of cross-selling opportunities, lower deposits, and decreased trust in the institution.

OBJECTIVE:

- Perform Data Exploration and Cleaning
- Gather and prepare the customer dataset for analysis by handling missing values, correcting inconsistencies, and ensuring data quality.
- Conduct Exploratory Data Analysis (EDA)
- Analyze customer demographics, account details, and behavioral patterns to uncover trends and correlations related to churn.
- Identify Key Churn Drivers
- Determine the most influential features that contribute to customer attrition through statistical analysis and feature importance evaluation.
- Build and Evaluate Predictive Models
- Apply machine learning algorithms to predict the likelihood of churn and compare model performance using accuracy, precision,, and ROC-AUC metrics.
- Provide Insights and Recommendations

ABOUT THE DATASET

The dataset used for this project contains detailed information about customers of a retail bank, along with an indicator showing whether each customer has exited (churned) or remained with the bank. It consists of 10,000 customer records and 14 variables, combining demographic, financial, and behavioral data points. This variety of attributes allows for a well-rounded analysis of factors that may influence customer attrition.

The key variable of interest is Exited, which serves as the target variable for prediction. A value of 1 indicates that the customer has left the bank, while a value of 0 means the customer has been retained. The other variables provide supporting context:

- Demographic details such as Geography (customer's country), Gender, and Age.
- Account-related information including Tenure (years with the bank), NumOfProducts (number of banking products used), and HasCrCard (whether the customer owns a credit card).
- Financial indicators like CreditScore, Balance, and EstimatedSalary.
- Engagement metrics such as IsActiveMember to reflect customer activity level.

Some variables, such as RowNumber, CustomerId, and Surname, are unique identifiers that do not contribute directly to churn prediction but remain in the dataset for completeness.

This dataset is well-suited for churn analysis because it captures both static attributes (e.g., age, geography) and dynamic behaviors (e.g., account activity, number of products), enabling the identification of patterns that can help predict and reduce customer attrition.

CHAPTER 1:
DATASET DESCRIPTION

IMPORT THE LIBRARIES & DATASET

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [6]:
```
df=pd.read_csv('Churn_Modelling.csv')
```

In [9]:
```
df.head()
```

Out[9]:

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

In [10]:
```
df.dtypes
```

Out[10]:
```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

In [11]:
```
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 [7]: df.isnull().sum().to_frame().rename(columns={0:"Sum of Missing Values"})
```

Out[7]:

	Sum of Missing Values
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

- Here we try to see if there is any missing values present; since the sum of each column is 0, this means that there is no null value present and thus the data is clean.

```
In [28]: df.describe()
```

Out[28]:

	RowNumber	CustomerId	CreditScore	Gender	Age	Ten
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000
mean	5000.50000	1.569094e+07	650.528800	0.454300	38.921800	5.012
std	2886.89568	7.193619e+04	96.653299	0.497932	10.487806	2.892
min	1.00000	1.556570e+07	350.000000	0.000000	18.000000	0.000
25%	2500.75000	1.562853e+07	584.000000	0.000000	32.000000	3.000
50%	5000.50000	1.569074e+07	652.000000	0.000000	37.000000	5.000
75%	7500.25000	1.575323e+07	718.000000	1.000000	44.000000	7.000
max	10000.00000	1.581569e+07	850.000000	1.000000	92.000000	10.000

```
In [9]: # Churn distribution
        print(df['Exited'].value_counts(normalize=True) * 100)
```

```
Exited
0    79.63
1    20.37
Name: proportion, dtype: float64
```

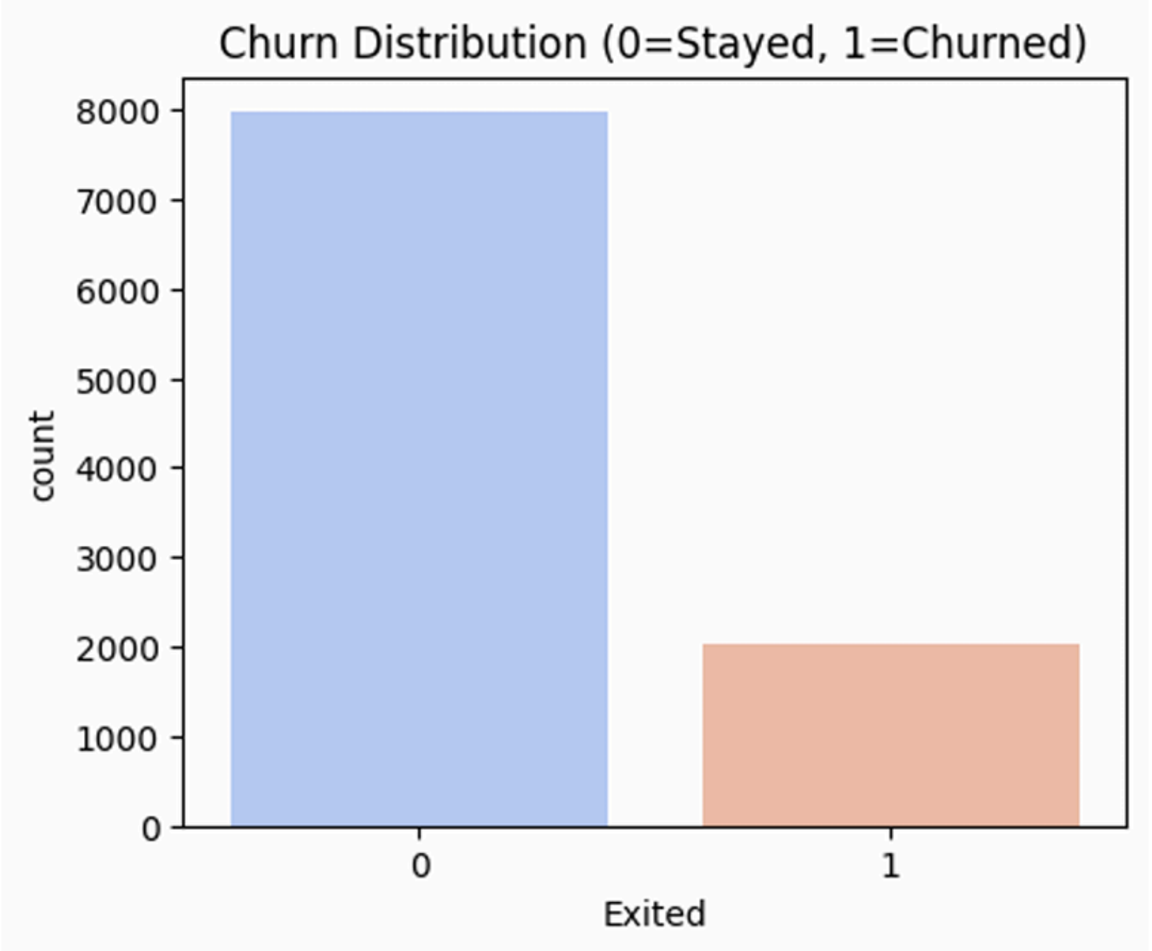
- The dataset contains 10,000 customer records with a total of 14 attributes, including both demographic and account-related variables.
- Target Variable: Exited (1 = Customer Churned, 0 = Customer Retained)
- Categorical Variables:
- Geography (France, Spain, Germany)
- Gender (Male, Female)
- Numerical Variables:
- CreditScore, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary
- Identifiers: RowNumber, CustomerId, and Surname – excluded from analysis as they do not contribute predictive value.
- The dataset is clean with no missing values. Churn distribution shows that 79.63% of customers stayed with the bank, while 20.37% exited, indicating a moderately imbalanced target variable.

CHAPTER 2: EXPLANATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is the first step in any data analysis project. It helps in understanding the structure, variability, trends, and seasonality present in the data before moving to formal statistical modeling.

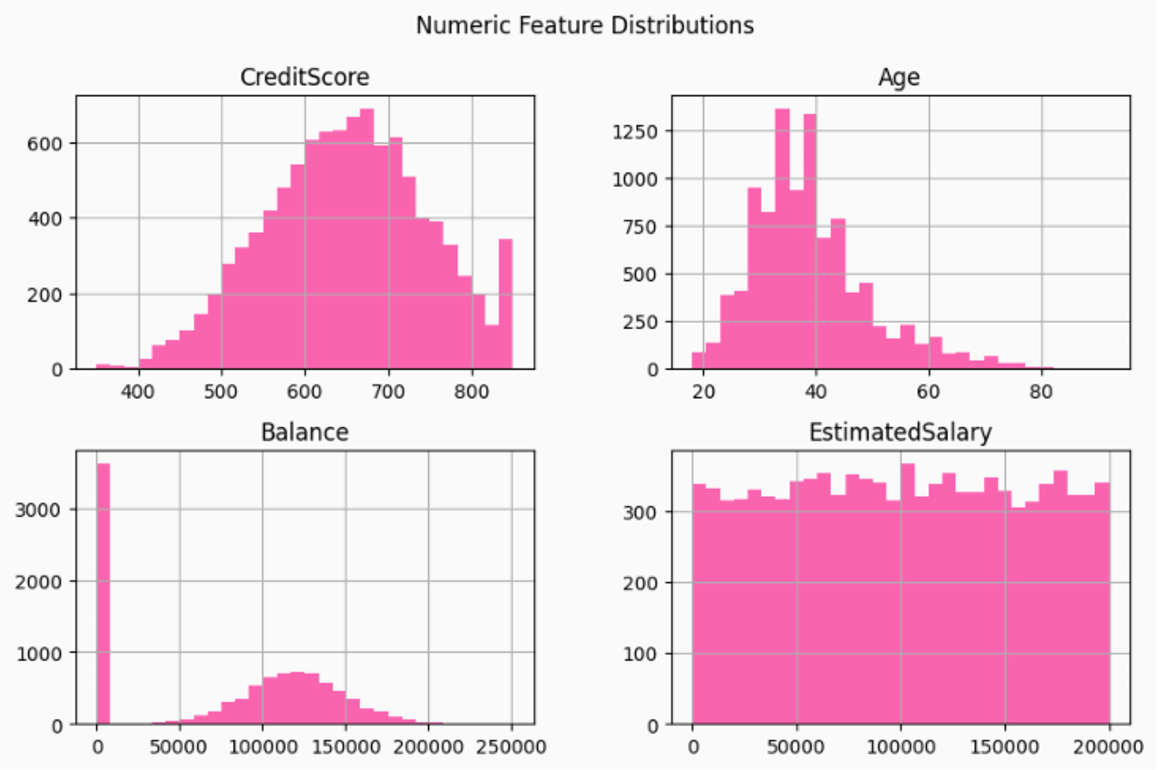
Univariate Analysis

In [7]:
```
plt.figure(figsize=(5,4))
sns.countplot(data=df, x='Exited', hue= 'Exited', palette='coolwarm', legend= Fa
plt.title("Churn Distribution (0=Stayed, 1=Churned)")
plt.show()
```



In [11]:
```
numeric_features = ['CreditScore', 'Age', 'Balance', 'EstimatedSalary']

df[numeric_features].hist(bins=30, figsize=(10,6), color='hotpink')
```

```
plt.suptitle("Numeric Feature Distributions")
plt.show()
```



Inference:

- Age

The age distribution shows that the majority of customers are between 30 and 50 years old. Interestingly, churn rates are noticeably higher among customers in their 40s, which may indicate mid-career financial shifts or dissatisfaction with banking services during this life stage.

- Credit Score

Customer credit scores range from 350 to 850, with an average of around 650. While lower scores tend to show slightly higher churn, there isn't a sharp cut-off — meaning churn is not solely driven by creditworthiness.

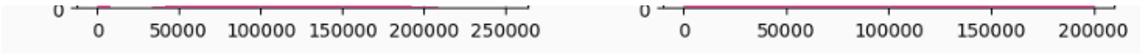
- Balance

A surprising observation is that a large number of customers maintain a zero balance, yet they are not necessarily the highest churners. On the other hand, churn is also present among customers with substantial balances, suggesting that financial holdings alone do not prevent attrition.

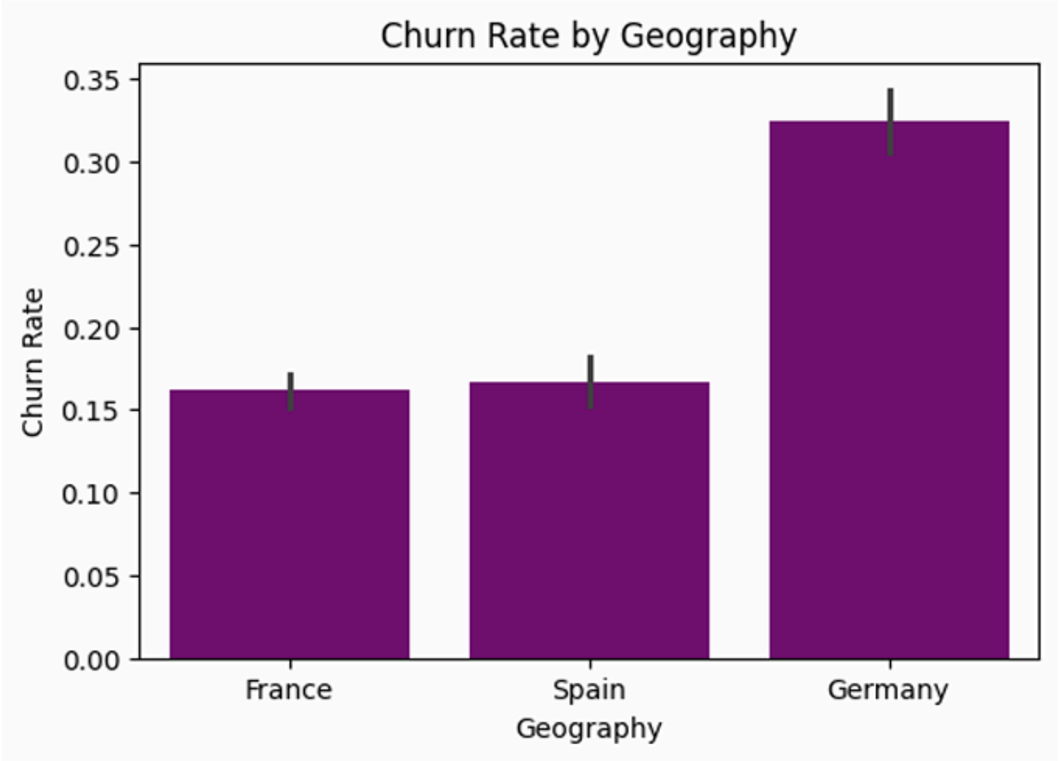
- Estimated Salary

Salaries are relatively evenly distributed, showing no strong direct correlation with churn at first glance.

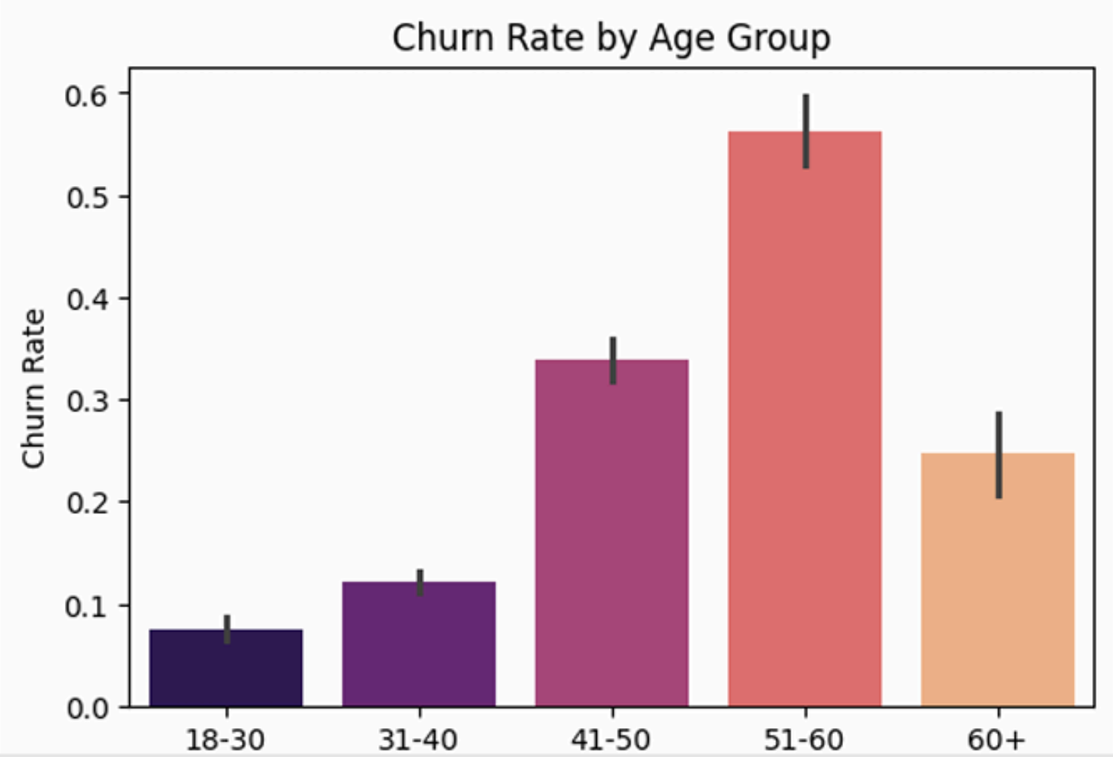
Bivariate Analysis



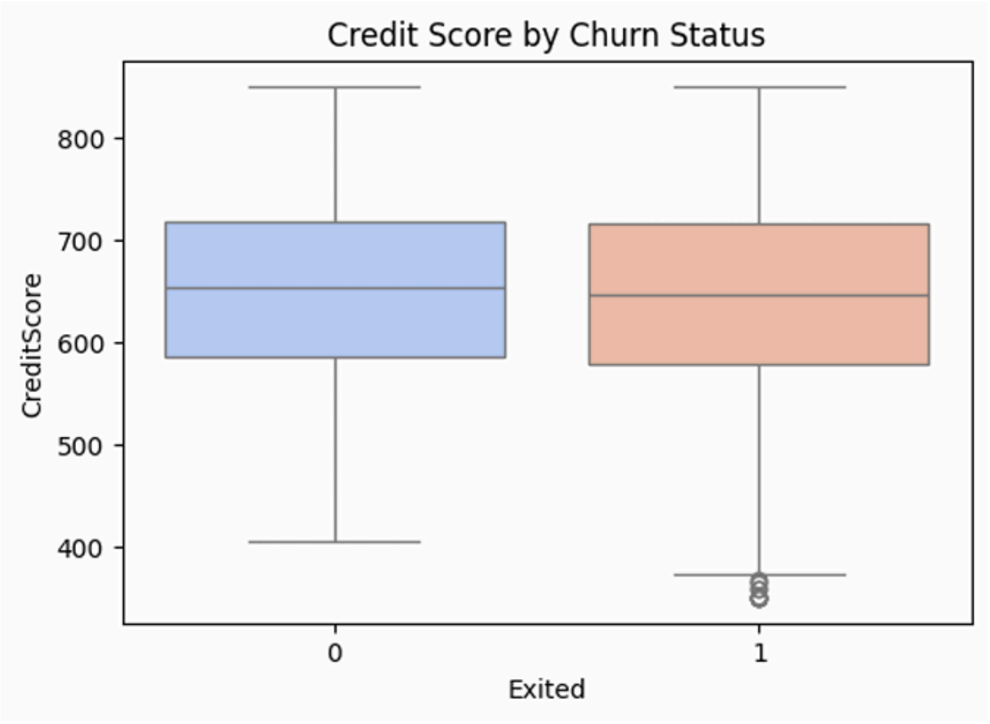
In [18]:
```
plt.figure(figsize=(6,4))
sns.barplot(data=df, x='Geography', y='Exited'
            , estimator=np.mean, color='purple')
plt.title("Churn Rate by Geography")
plt.ylabel("Churn Rate")
plt.show()
```



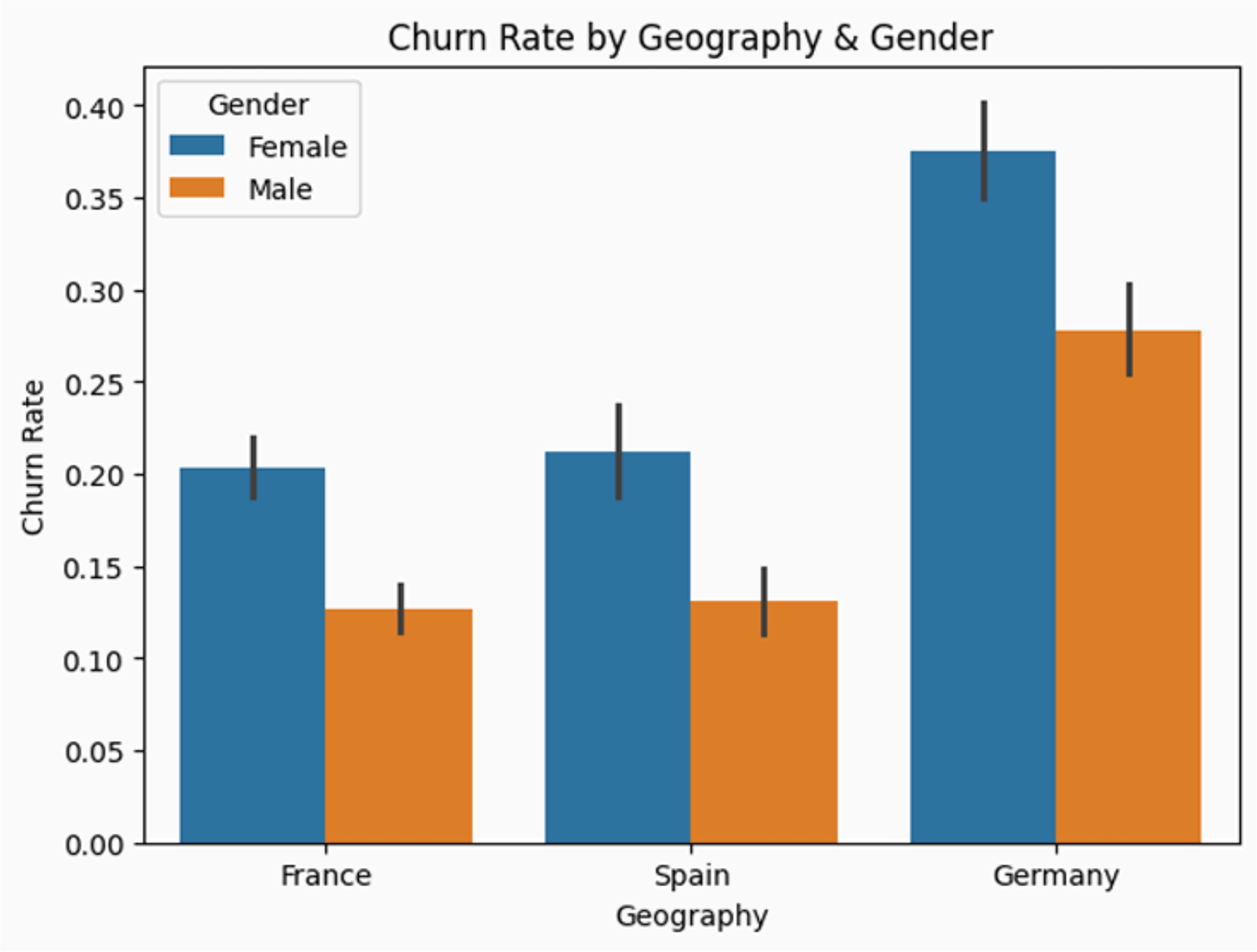
In [19]:
```
# Create Age bins
df['AgeGroup'] = pd.cut(df['Age'], bins=[18, 30, 40, 50, 60, 100],
                        labels=['18-30','31-40','41-50','51-60','60+'])

plt.figure(figsize=(6,4))
sns.barplot(data=df, x='AgeGroup', y='Exited', estimator=np.mean,hue='AgeGroup',
plt.title("Churn Rate by Age Group")
plt.ylabel("Churn Rate")
plt.show()
```



In [20]:
```
plt.figure(figsize=(6,4))
sns.boxplot(data=df, x='Exited', y='CreditScore',hue= 'Exited', palette='coolwar
plt.title("Credit Score by Churn Status")
plt.show()
```



In [8]:
```
plt.figure(figsize=(7,5))
sns.barplot(data=df, x='Geography', y='Exited', hue='Gender', estimator=np.mean)
plt.title("Churn Rate by Geography & Gender")
plt.ylabel("Churn Rate")
plt.show()
```



- Geography

Geographical differences in churn are significant. Germany has the highest churn rate, while France has the lowest. Spain falls in between. This implies potential differences in customer service experience, competition, or economic factors by region.

- Age Group Trends

When grouped into age brackets:

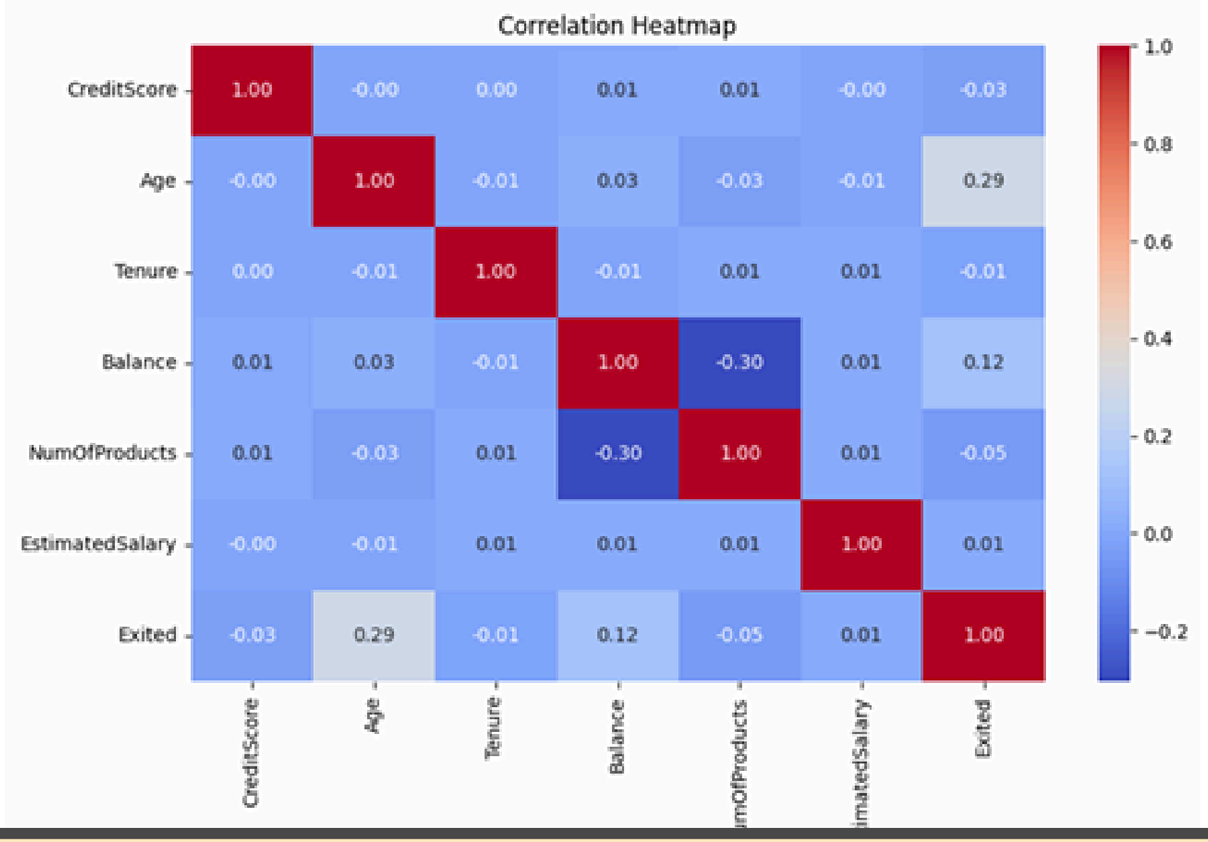
- 18–30 years: Lowest churn — possibly due to fewer banking alternatives or lower financial commitments.
- 31–40 years: Moderate churn, possibly linked to increased financial mobility.
- 41–50 years: Peak churn — this group may be seeking better banking benefits or facing financial stress.
- 51–60 years: Churn starts to decline slightly.
- Customer Activity

The IsActiveMember variable is one of the clearest churn indicators. Inactive members churn at significantly higher rates than active members, highlighting engagement as a key retention factor.

- Geography × Gender

When combining these variables, it's evident that female customers in Germany exhibit the highest churn rate, making them a critical segment for retention efforts.

In [37]:
```
plt.figure(figsize=(10,6))
corr = df[['CreditScore','Age','Tenure','Balance','NumOfProducts','EstimatedSala
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

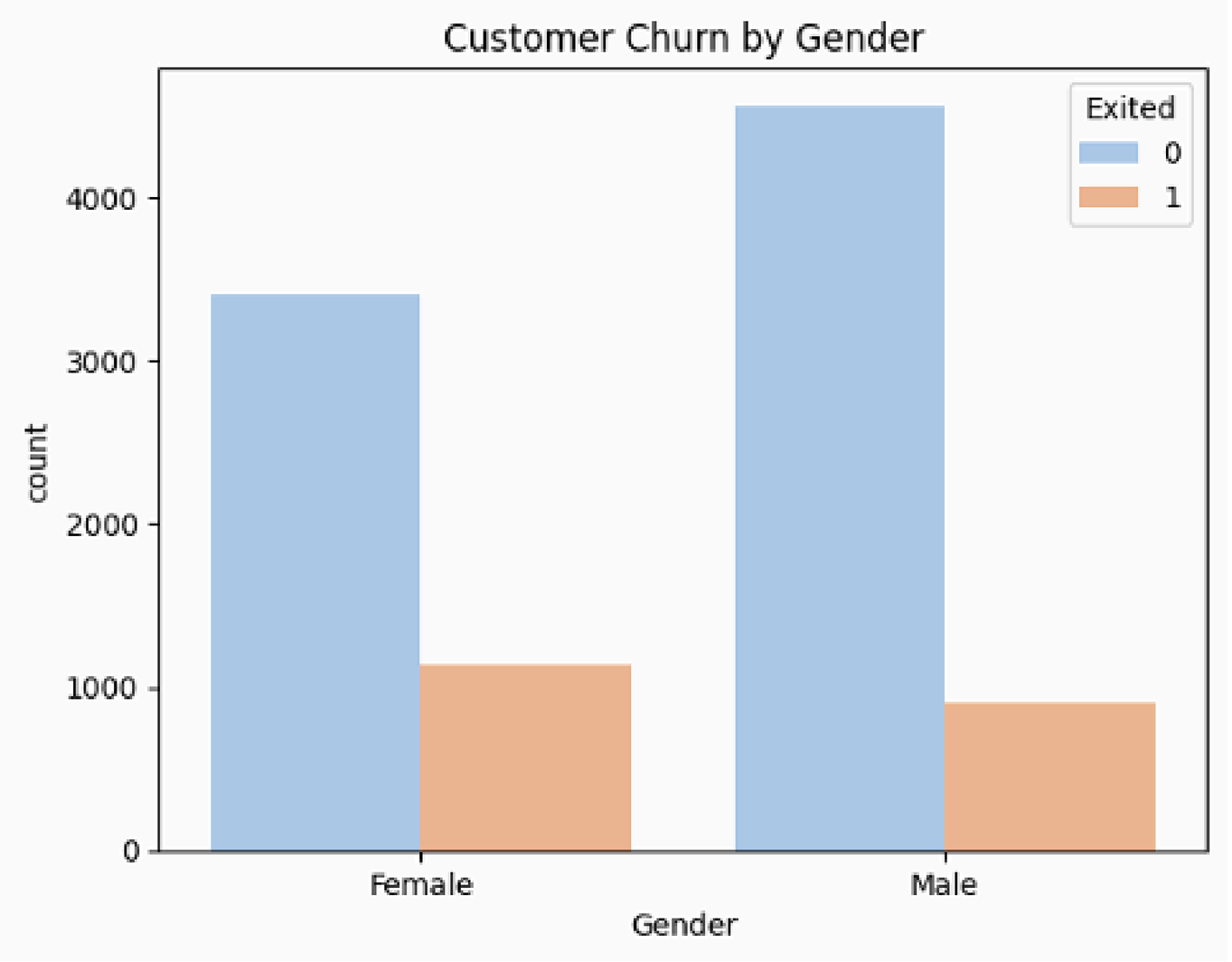


Correlation Insights

Correlation analysis shows weak relationships between most numerical features and churn. Notable patterns include:

- Age has a mild positive correlation with churn.
- IsActiveMember has a negative correlation — more active members are less likely to leave.
- Other features such as CreditScore and EstimatedSalary show minimal linear correlation but may have non-linear effects worth exploring in predictive models.

In [21]:
```
sns.countplot(x='Gender', hue='Exited', data=df, palette='pastel')
plt.title('Customer Churn by Gender')
plt.show()
```



KEY FINDINGS

From this stage of the analysis, several high-risk customer segments emerge:

1. German customers, particularly females, have the highest churn rates.
2. Middle-aged customers (41–50) are more likely to leave than younger or older customers.
3. Inactive members, regardless of balance or tenure, have a higher probability of churn.

Additionally, churn is not confined to low-balance or low-income customers, indicating that retention strategies should focus on service quality and engagement rather than only financial incentives.

CHAPTER 3:

LOGISTIC REGRESSION MODEL

TRAIN-TEST SPLIT & PREPROCESSING

In [43]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_sco
```

In [44]:
```
X = df.drop(['RowNumber', 'CustomerId', 'Surname', 'Exited'], axis=1)
y = df['Exited']
```

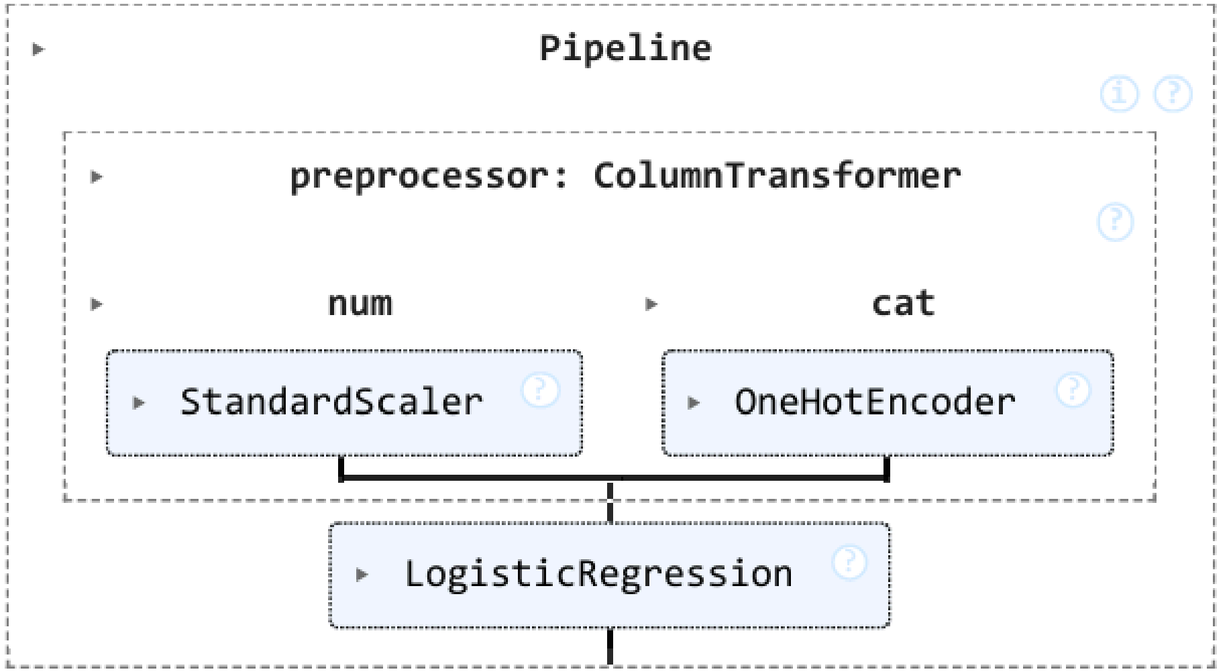
In [47]:
```
# Categorical and numeric feature lists
categorical_features = ['Geography', 'Gender']
numeric_features = ['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts',

# Preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(drop='first'), categorical_features)
    ])
```

In [45]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

- Separated the dataset into training (80%) and testing (20%) subsets.
- Preprocessing steps included:
- Standard Scaling for numeric variables (so features are on a comparable scale).
- One-Hot Encoding for categorical variables (Geography, Gender).
- Purpose: Ensures the data is in the right format for machine learning models and prevents bias from differences in feature scales.

In [48]:
```
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=1000))
])

model.fit(X_train, y_train)
```

Out[48]:



LOGISTIC REGRESSION MODEL

In [49]:
```
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, y_pred_proba))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.97      0.89      1593
           1       0.59      0.19      0.28       407

    accuracy                           0.81      2000
   macro avg       0.71      0.58      0.59      2000
weighted avg       0.78      0.81      0.77      2000

Confusion Matrix:
 [[1541   52]
 [ 331   76]]
ROC-AUC Score: 0.7748117917609444
```

In [52]:
```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier

# Example: Random Forest
rf_model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=200, random_state=42))
])
rf_model.fit(X_train, y_train)
print("RF ROC-AUC:", roc_auc_score(y_test, rf_model.predict_proba(X_test)[:,1]))
```

```
RF ROC-AUC: 0.8530757259570819
```

UNDERSTANDING:

- Built a Logistic Regression model as the baseline.
- This is a simple, interpretable algorithm often used for churn prediction.
- Results:

Accuracy ≈ 81%, but recall for churn (class 1) was low (~19%).
ROC-AUC ≈ 0.77.

- Point of it:

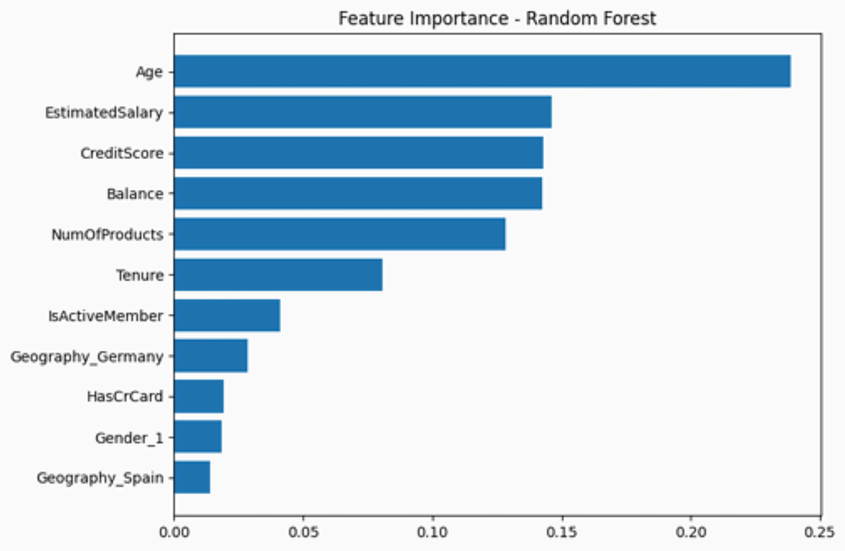
Logistic regression sets a baseline benchmark. It performs decently at predicting non-churners (class 0), but struggles with churners (class 1), which is common in imbalanced datasets.
. Random Forest Classifier

- ROC-AUC improved to 0.85.
- This is significantly better than logistic regression, meaning the model distinguishes churners from non-churners more effectively.

Point of it:

- Shows how more complex models (tree-based) can capture non-linear relationships and interactions between features, which logistic regression might miss.

```
In [53]: import matplotlib.pyplot as plt

         # Get feature names after preprocessing
         feature_names = numeric_features + list(model.named_steps['preprocessor']
                                                 .transformers_[1][1]
                                                 .get_feature_names_out(categorical_featu

         importances = rf_model.named_steps['classifier'].feature_importances_
         feat_imp = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
         feat_imp.sort_values(by='Importance', ascending=False, inplace=True)

         plt.figure(figsize=(8,6))
         plt.barh(feat_imp['Feature'], feat_imp['Importance'])
         plt.title("Feature Importance - Random Forest")
         plt.gca().invert_yaxis()
         plt.show()
```



Feature Importance (Random Forest)

- Extracted feature importance values.
- Some of the top drivers of churn were likely:- Age, Balance, IsActiveMember, Geography (Germany stood out in EDA as high risk)

Point of it:

- This step tells you why the model is making predictions — which features are most influential in churn risk. It also gives business teams concrete factors to target.

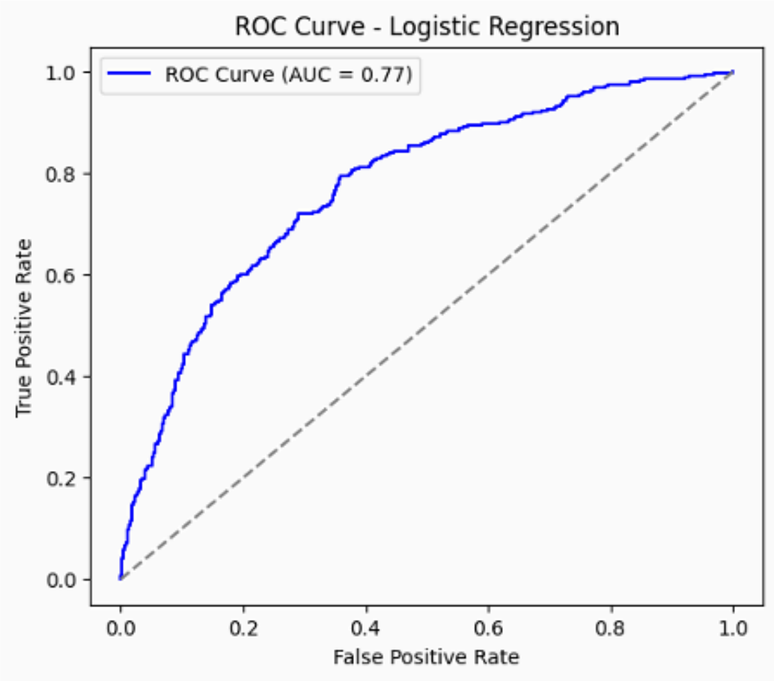
CHAPTER 4:

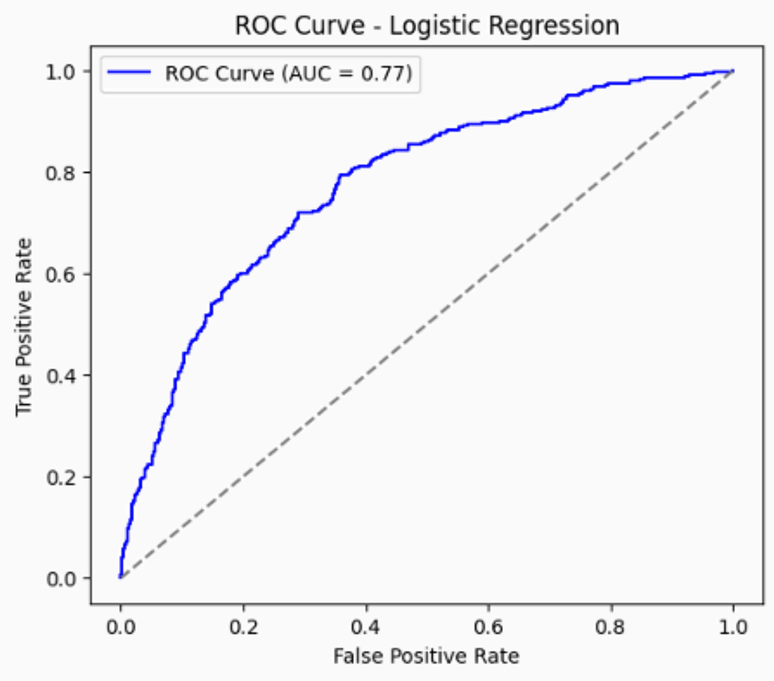
ROC CURVES

In [54]:

```
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# Predict probabilities
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Get ROC curve values
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

# Plot ROC curve
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {roc_auc_score(y_test,
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')  # random guess line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Logistic Regression')
plt.legend()
plt.show()
```


```
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# Predict probabilities
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Get ROC curve values
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

# Plot ROC curve
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {roc_auc_score(y_test,
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')  # random guess line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Logistic Regression')
plt.legend()
plt.show()
```





ROC Curves

- Plotted ROC curves for Logistic Regression, Random Forest, and XGBoost.
- The curves visually show how well each model separates churn vs non-churn across different probability thresholds.
- Random Forest and XGBoost had visibly better ROC curves compared to Logistic Regression.

Point of it:

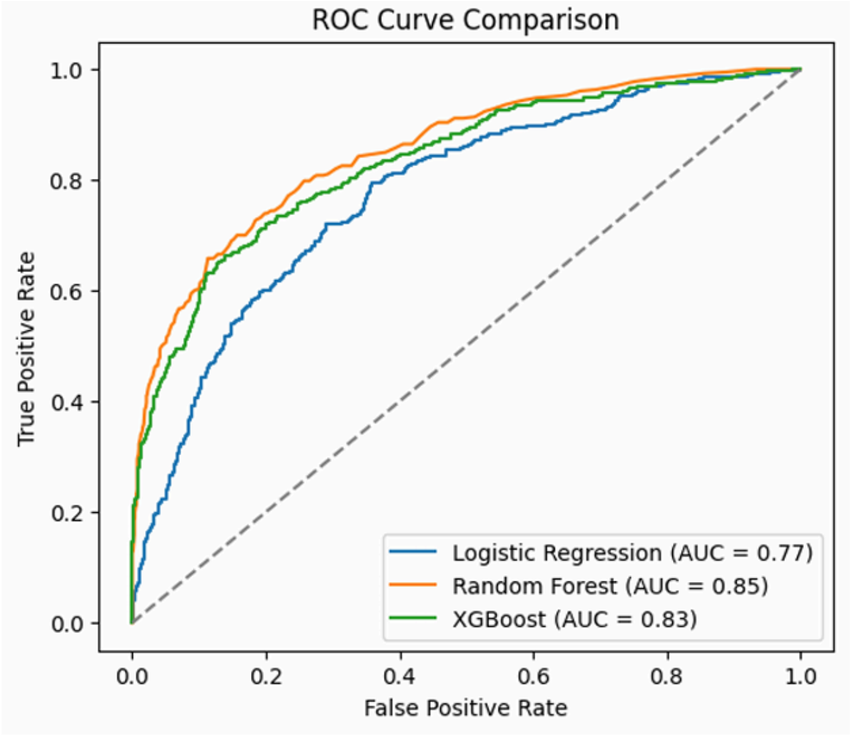
- Provides a model comparison using a single metric (AUC) that’s robust even when the dataset is imbalanced.
- Confirms which algorithm is best suited for the churn prediction task.

CHAPTER 5:

High-Risk Customers Identification

In [55]:
```
models = {
    "Logistic Regression": model,
    "Random Forest": rf_model,
```

```
    "XGBoost": Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('classifier', XGBClassifier(use_label_encoder=False, eval_metric='loglo
    ])
}

plt.figure(figsize=(6,5))

for name, clf in models.items():
    clf.fit(X_train, y_train)
    y_proba = clf.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_proba)
    plt.plot(fpr, tpr, label=f"{name} (AUC = {roc_auc_score(y_test, y_proba):.2f

plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend()
plt.show()
```



In [56]:
```
# Customers with predicted churn prob > 0.7
high_risk_customers = X_test[y_pred_proba > 0.7]
print(f"High-risk customers identified: {len(high_risk_customers)}")
```

```
High-risk customers identified: 18
```

- Finally, extracted customers with a predicted churn probability > 0.7.
- Example: found 18 high-risk customers.

Point of it:

- This is the start of prescriptive analytics. Instead of just saying "20% of customers churn," the model tells you exactly which customers are most at risk, so the bank can intervene with offers, calls, or retention campaigns.

CONCLUSION

- This project analyzed bank customer churn through data exploration and predictive modeling. Exploratory analysis showed that churn affects about 20% of customers, with higher risk among German clients, middle-aged customers (41–50), and inactive members.
- Machine learning models were then applied to predict churn. While Logistic Regression provided a useful baseline, Random Forest and XGBoost achieved stronger performance (ROC-AUC > 0.85), accurately identifying key churn drivers such as Age, Balance, Geography, and Activity Status. The models also flagged high-risk customers with churn probabilities above 70%, enabling proactive retention targeting.
- Overall, the project demonstrates that combining EDA, predictive modeling, and business insights can help banks reduce churn, strengthen customer engagement, and improve long-term revenue.