1. **Total Balance for Each Geography**:

total\_balance\_geography = df.groupby('Geography')['Balance'].sum()

print(total\_balance\_geography)

1. **Age Group with the Highest Average Credit Score**:

# Create age groupsreplace

age\_bins = [18, 30, 40, 50, 60, 70]

age\_labels = ['18-30', '31-40', '41-50', '51-60', '61-70']

df['AgeGroup'] = pd.cut(df['Age'], bins=age\_bins, labels=age\_labels, right=False)

highest\_credit\_score\_age\_group = df.groupby('AgeGroup')['CreditScore'].mean().idxmax()

print(highest\_credit\_score\_age\_group)

1. **Average Estimated Salary for Each Gender**:

average\_salary\_gender = df.groupby('Gender')['EstimatedSalary'].mean()

print(average\_salary\_gender)

1. **Average Balance for Active vs. Inactive Members**:

average\_balance\_members = df.groupby('IsActiveMember')['Balance'].mean()

print(average\_balance\_members)

1. **Geography with the Highest Number of Customers with a Credit Card**:

credit\_card\_customers = df[df['HasCrCard'] == 'Yes'].groupby('Geography')['CustomerId'].count()

highest\_credit\_card\_geography = credit\_card\_customers.idxmax()

print(highest\_credit\_card\_geography)

1. **Total Value of Balances for Each Gender**:

total\_balance\_gender = df.groupby('Gender')['Balance'].sum()

print(total\_balance\_gender)

1. **Age Group with the Highest Churn Rate**:

churn\_rate\_age\_group = df.groupby('AgeGroup')['Exited'].mean()

highest\_churn\_rate\_age\_group = churn\_rate\_age\_group.idxmax()

print(highest\_churn\_rate\_age\_group)

1. **Average Tenure of Customers Who Exited vs. Those Who Did Not**:

average\_tenure\_exited = df.groupby('Exited')['Tenure'].mean()

print(average\_tenure\_exited)

1. **Total Number of Products Owned by Customers in Each Geography**:

total\_products\_geography = df.groupby('Geography')['NumOfProducts'].sum()

print(total\_products\_geography)

1. **Age Group with the Highest Average Estimated Salary**:

average\_salary\_age\_group = df.groupby('AgeGroup')['EstimatedSalary'].mean()

highest\_salary\_age\_group = average\_salary\_age\_group.idxmax()

print(highest\_salary\_age\_group)

1. **Distribution of Number of Products Held by Customers Who Exited vs. Those Who Did Not**:

sns.countplot(x='NumOfProducts', hue='Exited', data=df)

plt.title('Distribution of Number of Products by Exit Status')

plt.show()

1. **Average Balance for Customers with Different Numbers of Products**:

average\_balance\_products = df.groupby('NumOfProducts')['Balance'].mean()

print(average\_balance\_products)

1. **Gender with Higher Average Balance for Customers Who Exited**:

average\_balance\_exited\_gender = df[df['Exited'] == 1].groupby('Gender')['Balance'].mean()

higher\_balance\_gender = average\_balance\_exited\_gender.idxmax()

print(higher\_balance\_gender)

1. **Most Common Age Group for Customers in Each Geography**:

most\_common\_age\_group = df.groupby('Geography')['AgeGroup'].agg(lambda x: x.mode()[0])

print(most\_common\_age\_group)

1. **Average Credit Score for Customers with and without Credit Cards**:

average\_credit\_score\_card = df.groupby('HasCrCard')['CreditScore'].mean()

print(average\_credit\_score\_card)

1. **Combination of Geography and Gender with Highest Average Estimated Salary**:

highest\_salary\_combination = df.groupby(['Geography', 'Gender'])['EstimatedSalary'].mean().idxmax()

print(highest\_salary\_combination)

1. **Most Common Tenure Period Among Customers Who Exited**:

most\_common\_tenure\_exited = df[df['Exited'] == 1]['Tenure'].mode()[0]

print(most\_common\_tenure\_exited)

1. **Average Balance Varying by Credit Score Groups**:

credit\_score\_bins = [0, 600, 700, 800, 900]

credit\_score\_labels = ['Low', 'Medium', 'High']

df['CreditScoreGroup'] = pd.cut(df['CreditScore'], bins=credit\_score\_bins, labels=credit\_score\_labels, right=False)

average\_balance\_credit\_score = df.groupby('CreditScoreGroup')['Balance'].mean()

print(average\_balance\_credit\_score)

1. **Total Number of Active Members in Each Geography**:

total\_active\_members = df[df['IsActiveMember'] == 'Yes'].groupby('Geography')['CustomerId'].count()

print(total\_active\_members)

1. **Average Estimated Salary for Customers with a Zero Balance**:

average\_salary\_zero\_balance = df[df['Balance'] == 0]['EstimatedSalary'].mean()

print(average\_salary\_zero\_balance)

1. **Correlation Analysis**:

correlation\_matrix = df.corr()

print(correlation\_matrix)

1. **Top Three Factors Correlated with Customer Churn**:

churn\_correlation = correlation\_matrix['Exited'].abs().sort\_values(ascending=False)

top\_factors = churn\_correlation[1:4] # Exclude 'Exited' itself

print(top\_factors)

**Conclusion**

These code snippets will help you analyze the dataset effectively. You can run these in a environment with Pandas installed to get the results. If you have any specific questions about the analyses or need further assistance, feel free to ask!