REPORT ON FINAL PROJECT

NAME : BANUPRIYA N

BATCH NUMBER : TN-DA-FNB03

CONTACT NUMBER : +91 6374056072

EMAIL ID : banupriyamba98@gmail.com

DATASET DOMAIN : BANKING AND FINANCIAL SERVICES

PROJECT TITLE : BANK CUSTOMER CHURN ANALYSIS AND RETENTION STRATEGY USING PYTHON FOR DATA CLEANING AND VISUALIZATIONS IN PYTHON & POWER BI

SUBMISSION DATE : 09-12-2025

MENTOR NAME : KUMARAN M

RAW DATASET LINK:

https://drive.google.com/file/d/14Sralyzi-0dTrilhoSuGX7sHkgciypeA/view?usp=drive_link

CLEANED DATASET LINK:

https://drive.google.com/file/d/17o9-PBAjsvztIbrMo6wIoLj0E2thOw5G/view?usp=drive_link

BANK CUSTOMER CHURN ANALYSIS AND RETENTION STRATEGY USING PYTHON FOR DATA CLEANING AND VISUALIZATIONS IN PYTHON & POWER BI

DOMAIN:

Banking and Financial Services

OBJECTIVE:

1. To clean and preprocess the customer churn dataset using python to ensure data accuracy, consistency and readiness for analysis.

2. To perform Exploratory Data Analysis to understand the factors influencing customer churn in banking sector.

3. To visualize the Customer Behaviour using python and Power BI.

4. To provide actionable retention strategies based on analytical findings to help banks to reduce customer churn and improve customer satisfaction level.

OUTCOME:

This project helps to uncover key patterns, trends and insights from the customer churn dataset through detailed analysis and visualization.

The findings provide clear understanding of customer behaviour and factors influencing customer churn.

This analysis and results help the bank to make evidence-based decisions, improve strategies in existing customer retention to reduce churn, customer satisfaction and strengthen customer loyalty.

Some Examples,

1.Understanding of churn patterns in banking

2.Actionable retention strategies for bank

3.Identifying high risk customer groups.

DATASET INFORMATION:

Source: Kaggle

Year / Timeline: According to Kaggle, the dataset was last updated approximately one year ago

Dataset Description:

RowNumber - Corresponds to the record (Row) number.

CustomerId -Unique Identification number for each customer.

Surname - It tells the LastName of the customers.

CreditScore - Numerical Score representing the customers credit worthiness.

Geography - It represents the customers Country/Location.

Gender - It Represents the gender of the customers.

Age - It tells the age of the Customers.

Tenure - Number of years the customer holds relationship with bank.

Balance - Current Balance of customer account.

NumberOfProducts - Number of products the customer purchased from bank.

HasCrCard - It indicates whether the customers have a credit card (1=Yes, 0=No).

IsActiveMember - It tells whether the customer actively using the account or not (1=Active, 0=Inactive).

EstimatedSalary - Estimated Salary of customers.

Exited - It indicates whether customer exited from bank or still holding relationship (Target Variable).

Complain - It indicates whether customer ever raised complaint due to unsatisfied Services (1=Yes,0=No).

Satisfaction Score - It tells how the customer satisfied by bank service.

Card Type - It indicates the variant of the card customer holds.

Point Earned - Reward Points earned by the customer.

TYPE OF ANALYSIS:

Descriptive Analysis:

The first mandatory steps include,

What percentage of customers churned?

Distribution of churn across Geography, Products, Gender, Card type

Diagnostic Analysis:

In this step we will find the root causes of churn,

Are inactive Customers/Elder Customers more likely to churn?

Is satisfaction score linked to customer churn?

Is customer who complain often churn more?

Predictive Analysis :(No)

Prescriptive Analysis:

Improve doorstep Service for elder customers

Improve Customer engagement for Inactive Customers

Focus on customers who given low rating

Improve satisfaction score to reduce customer churn

Focus on customers with 1 Product or 0 product.

Introducing Preapproved offers for the long-term customers and who maintaining High Balance.

Rewarding the customers who hold long term relationship, holds High Balance, Holds High variant cards.

STAGES FOR DA PROJECT

Stage 1 – Problem Definition and Dataset Selection

Define the business problem and expected outcome:

Business Problem Definition:

In banking existing customer retention is major challenge, because customer will switch their banking to other banks due to poor customer service, dissatisfaction with products and services, better offers and ROI. Customer churn impacts the profitability, references, customer life time value and overall business growth.

Expected Outcome:

The expected outcome of this project is to understand the factors influencing customer churn, and to provide insights that helps banks to reduce customer churn effectively. By performing multiple analysis this project will give evidence-based finding to make the bank better decisions.

Some Examples,

1.Understanding of churn patterns in banking

2.Actionable retention strategies for bank

3.Identifying high risk customer groups.

Choose dataset and explain its source, size, and features:

The Customer churn Dataset is Publicly available in Kaggle, it contains banking customer records with their demographic, financial details.

Dataset Source:

Kaggle-Customer Churn Records (the dataset was last updated approximately one year ago)

Size:

Rows:1000

Columns: 18

Features:

RowNumber, CustomerId, Surname, CreditScore, Geography, Gender, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary, Exited, Complain, Satisfaction Score, Card Type, Point Earned

Import libraries, load dataset

Dataset description (rows, columns, features)

Initial EDA (head, info, describe, shape, null checks)

Import libraries, load dataset:

Python code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
dataset = pd.read_csv("/content/Customer-Churn-Records.csv")
```

Dataset Description:

Shape of the dataset:

Python code:

```
print("Number of Rows and Columns:", dataset.shape)
```

Output:

Number of Rows and Columns: (10000, 18)

Dataset Features:

Python code:

```
print(dataset.columns.tolist())
```

Output:

['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary', 'Exited', 'Complain', 'Satisfaction Score', 'Card Type', 'Point Earned']

Initial EDA:

Displaying first 10 records:

Python Code:

```
print(dataset.head(10))
```

Output:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	SatisfactionScore	CardType	PointEarned
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1	1	2	DIAMOND	464
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0	1	3	DIAMOND	456
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1	1	3	DIAMOND	377
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0	0	5	GOLD	350
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0	0	5	GOLD	425
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	1	1	5	DIAMOND	484
6	7	15592531	Bartlett	822	France	Male	50	7	0.00	2	1	1	10062.80	0	0	2	SILVER	206
7	8	15656148	Obinna	376	Germany	female	29	4	115046.74	4	1	0	119346.88	1	1	2	DIAMOND	282
8	9	15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.50	0	0	3	GOLD	251

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	SatisfactionScore	CardType	PointEarned
9	10	15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.73	0	0	3	GOLD	342

Displaying last 10 records:

Python code:

```
print(dataset.tail(10))
```

Output:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Card Type	PointEarned
9990	9990	15605622	McMillan	841	Spain	Male	28	4	0.00	2	1	1	179436.60	0	0	5	GOLD	393
9991	9991	15798964	Nkemakonam	714	NaN	Male	33	3	35016.60	1	1	0	53667.08	0	0	3	GOLD	791
9992	9992	15769959	Ajulukchukwu	597	France	Female	53	4	88381.21	1	1	0	69384.71	1	1	3	GOLD	369
9993	9993	15657105	Chukwualuka	726	Spain	Male	36	2	0.00	1	1	0	195192.40	0	0	5	SILVER	560
9994	9994	15569266	Rahman	644	France	Male	28	7	155060.41	1	1	0	29179.52	0	0	5	DIAMOND	715
9995	9995	15719294	Wood	800	France	Female	29	2	0.00	2	0	0	167773.55	0	0	4	PLATINUM	311
9996	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0	0	1	DIAMOND	300

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Card Type	Point Earned
9997	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0	0	5	PLATINUM	771
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1	1	2	GOLD	339
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0	0	3	DIAMOND	911

Total Number of Elements:

Python code:

```
dataset.size
```

Output:

180000

Displaying Datatypes of dataset:

Python code:

```
print(dataset.dtypes)
```

Output:

```
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
Complain             int64
Satisfaction Score   int64
Card Type            object
Point Earned         int64
dtype: object
```

Check for missing values:

Python code:

```
dataset.isnull().sum()
```

Output:

	0
RowNumber	0
CustomerId	0
Surname	0
CreditScore	0
Geography	3
Gender	2
Age	0
Tenure	0

	0
Balance	0
NumOfProducts	0
HasCrCard	0
IsActiveMember	0
EstimatedSalary	0
Exited	0
Complain	0
Satisfaction Score	0
Card Type	1
Point Earned	0

dtype: int64

Info about the dataset:

Python code:

```
dataset.info()
```

Output:

```
<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          9997 non-null   object
5   Gender             9998 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          9999 non-null   object
17  Point Earned       10000 non-null  int64
dtypes: float64(2), int64(12), object(4)
memory usage: 1.4+ MB
```

Statistical summary:

Python code:

```
dataset.describe()
```

	RowNumber	CustomerId	Credit Score	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Point Earned
count	10000.000000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000

	RowNumber	CustomerId	Credit Score	Age	Tenure	Balance	NumOf Products	HasCrCard	IsActive Member	EstimatedSalary	Exited	Complain	Satisfaction Score	Point Earned
mean	5000.347700	1.569095e+07	650.530000	38.921500	5.012500	76493.142833	1.530200	0.705600	0.515100	100096.424472	0.203700	0.204300	3.014000	606.532600
std	2886.672172	7.192836e+04	96.654099	10.487933	2.892124	62392.728875	0.581654	0.455795	0.499797	57507.579795	0.402769	0.403209	1.406059	225.928323
min	1.000000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.000000	0.000000	11.580000	0.000000	0.000000	1.000000	119.000000
25%	2500.750000	1.562855e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.000000	0.000000	51014.837500	0.000000	0.000000	2.000000	410.000000
50%	5000.500000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.000000	1.000000	100218.210000	0.000000	0.000000	3.000000	605.000000
75%	7500.250000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.000000	1.000000	149388.247500	0.000000	0.000000	4.000000	801.000000
max	10000.000000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.000000	1.000000	199992.480000	1.000000	1.000000	5.000000	1000.000000

Check for Duplicate Records:

Python code:

```
print("Number of Duplicate Records:", dataset.duplicated().sum())
```

Output:

Number of Duplicate Records: 1

Stage 2 – Data Cleaning and Pre-processing:

Handle missing values (impute or drop)

Handle duplicates

Treat outliers if required

Check skewness and apply transformations

Convert data types if needed

Feature transformations (date parts, derived fields if required for analysis)

Imputing Missing values for 'Geography' Column:

Python code:

```
print(dataset[['Geography','Gender','Card Type']].isnull().sum())
dataset['Geography'] = dataset['Geography'].fillna(dataset['Geography'].mode()[0])
print("\nAfter Cleaning:", dataset[['Geography','Gender','Card Type']].isnull().sum())
print("\nMissing Values Imputed for 'Geography' Column")
```

Output:

```
Geography    3
Gender       2
Card Type    1
dtype: int64

After Cleaning: Geography    0
Gender       2
Card Type    1
dtype: int64

Missing Values Imputed for 'Geography' Column
```

Explanation:

The Geography column contained three missing values, which were handled using mode imputation since it is a categorical feature. The most frequently occurring country was used to replace the missing entries, ensuring no data loss and maintaining the original data distribution. After imputation, the Geography column contained zero missing values.

Imputing Missing values for 'Gender' Column:

Python code:

```
print(dataset[['Gender','Card Type']].isnull().sum())


dataset['Gender'] = dataset['Gender'].fillna(dataset['Gender'].mode()[0])
print("\n After Cleaning:", dataset[['Gender','Card Type']].isnull().sum())
print("\nMissing Values Imputed for 'Gender' Column")
```

Output:

```
Gender      2
Card Type    1
dtype: int64


 After Cleaning: Gender      0
Card Type    1
dtype: int64


Missing Values Imputed for 'Gender' Column
```

Explanation:

The Gender column contained two missing values, which were imputed using the mode since it is a categorical variable. This approach ensures that the most frequently occurring gender category replaces the missing entries without affecting the overall data distribution. After imputation, the Gender column no longer contains any missing values.

Imputing Missing Value for 'Card Type' Column:

Python code:

```
print("Null value :", dataset['Card Type'].isnull().sum())

dataset['Card Type'] = dataset['Card Type'].fillna(dataset['Card Type'].mode()[0])

print("\nAfter Cleaning:", dataset['Card Type'].isnull().sum())

print("\nMissing Values Imputed for 'Card Type' Column")
```

Output:

```
Null value : 1


After Cleaning: 0


Missing Values Imputed for 'Card Type' Column
```

Explanation:

The Card Type column contained one missing value, which was handled using mode imputation as it is a categorical feature. The most frequently occurring card type was used to replace the missing entry, ensuring data consistency without introducing bias. After the imputation process, the Card Type column contains no missing values

Duplicate Records Removal:

Python Code:

```
print("Number of Duplicate Record:",dataset.duplicated().sum())

dataset = dataset.drop_duplicates(subset=['RowNumber','CustomerId','Card Type'], keep='first')

print("Number of duplicate records after removal:",dataset.duplicated().sum())

print("Duplicate Records Removed. Hence, there is no Duplicate Record available in dataset")
```

Output:

```
Number of Duplicate Record: 1

Number of duplicate records after removal: 0

Duplicate Records Removed. Hence, there is no Duplicate Record available in dataset
```

Explanation:

The dataset initially contained one duplicate record, which was identified using the duplicate check. This duplicate was removed based on the unique combination of RowNumber, CustomerId, and Card Type by keeping only the first occurrence. After removing the duplicate entry, the dataset no longer contains any duplicate records, ensuring data uniqueness and reliability for further analysis.

Converting Binary values to Categorical labels for 'HasCrCard' Column using replace():

Python code:

```
dataset['HasCrCard'] = dataset['HasCrCard'].replace({1:'Yes', 0:'No'})

print(dataset)
```

Output:

	HasCrCard
0	*Yes*
1	*No*
2	*Yes*
3	*No*
4	*Yes*
...	*...*
9995	*No*
9996	*Yes*
9997	*Yes*
9998	*Yes*
9999	*Yes*

9999 rows × 1 columns

dtype: *object*

Explanation:

The binary values in the HasCrCard column were converted into meaningful categorical labels using the replace () function, where 1 was mapped to “Yes” and 0 to “No”. This transformation improves data readability and makes the variable more suitable for interpretation and visualization. After conversion, the column now clearly represents whether a customer holds a credit card or not.

Converting Binary Values to Categorical Labels for "IsActiveMember" Column using replace():

Python code:

```
dataset['IsActiveMember'] = dataset['IsActiveMember'].replace({1:'Yes',0:'No'})

print(dataset)
```

Output:

	IsActiveMember
0	Yes
1	Yes
2	No
3	No
4	Yes
...	...
9995	No
9996	No
9997	Yes
9998	No
9999	No

9999 rows × 1 columns

dtype: object

Explanation:

The binary values in the IsActiveMember column were converted into meaningful categorical labels using the replace () function, where 1 was mapped to “Yes” and 0 to “No”. This transformation improves data readability and makes the variable more suitable for interpretation and visualization. After conversion, the column now clearly represents whether a customer holds a credit card or not.

Converting Binary Values to Categorical labels for 'Exited' Column:

Python code:

```
dataset['Exited'] = dataset['Exited'].replace({1:'Yes', 0:'No'})

print(dataset)
```

Output:

	Exited
0	Yes
1	No
2	Yes
3	No
4	No
...	...
9995	No
9996	No
9997	No
9998	Yes
9999	No

9999 rows × 1 columns

dtype: object

Explanation:

The binary values in the Exited column were converted into meaningful categorical labels using the replace () function, where 1 was mapped to “Yes” and 0 to “No”. This transformation improves data readability and makes the variable more suitable for interpretation and visualization. After conversion, the column now clearly represents whether a customer holds a credit card or not.

Converting Binary Values to Category Labels for 'Complain' Column:

Python code:

```
dataset['Complain'] = dataset['Complain'].replace({1:'Yes',0:'No'})

print(dataset)
```

Output:

	Complain
0	Yes
1	Yes
2	Yes
3	No
4	No
...	...
9995	No
9996	No
9997	No
9998	Yes
9999	No

9999 rows × 1 columns

dtype: object

Explanation:

The binary values in the Complain column were converted into meaningful categorical labels using the replace () function, where 1 was mapped to “Yes” and 0 to “No”. This transformation improves data readability and makes the variable more suitable for interpretation and visualization. After conversion, the column now clearly represents whether a customer holds a credit card or not.

Standardizing Text Case in 'Gender' Column:

Python code:

```
print("Before Cleaning:",dataset['Gender'].unique())

dataset['Gender'] = dataset['Gender'].str.title()

print("After Cleaning:",dataset['Gender'].unique())
```

Output:

Before Cleaning: ['Female' 'Male' 'female' 'male']

After Cleaning: ['Female' 'Male']

Explanation:

The Gender column initially contained inconsistent text formats such as “Male”, “male”, “Female”, and “female”. To maintain uniformity, all values were standardized using the str.title() function, which converts text to proper case. After cleaning, the column contains only two consistent categories: “Male” and “Female”, improving data quality and avoiding duplicate category issues during analysis.

Replacing Wrong Text in 'Geography' Column:

Python code:

```
print("Before Cleaning:", dataset['Geography'].unique())

dataset['Geography'] = dataset['Geography'].replace({'F4ance':'France'})

print("After Cleaning:", dataset['Geography'].unique())
```

Output:

Before Cleaning: ['France' 'Spain' 'Germany' 'F4ance']

After Cleaning: ['France' 'Spain' 'Germany']

Explanation:

The Geography column contained an incorrect text value “F4ance”, which was a data entry error. This incorrect value was replaced with the correct label “France” using the replace() function to maintain consistency in categorical data. After cleaning, the column now contains only valid and standardized country names, improving data accuracy for further analysis.

Special Character Removal in 'Surname' Column:

Python code:

```
dataset['Surname'] = dataset['Surname'].str.replace(r'[^A-Za-z]','', regex=True)

dataset
```

Output:

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Card Type	PointEarned	LoanEligibility	CustomerSegment	Age_Transformed
0	15634602	Hargrave	619	France	Female	42	2	0.00	1	Yes	Yes	101348.88	Yes	Yes	2	DIAMOND	464	Not Eligible	Basic	6.48
1	15647311	Hill	608	Spain	Female	41	1	83807.86	1	No	Yes	112542.58	No	Yes	3	DIAMOND	456	Not Eligible	Gold	6.40
2	15619304	Onio	502	France	Female	42	8	159660.80	3	Yes	No	113931.57	Yes	Yes	3	DIAMOND	377	Not Eligible	Premium	6.48
3	15701354	Boni	699	France	Female	39	1	0.00	2	No	No	93826.63	No	No	5	GOLD	350	Not Eligible	Basic	6.24
4	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	Yes	Yes	79084.10	No	No	5	GOLD	425	Eligible	Premium	6.56
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
99	15719294	Wood	800	France	Female	29	2	0.00	2	No	No	167773.55	No	No	4	PLATINUM	311	Not Eligible	Basic	5.39

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Card Type	PointEarned	LoanEligibility	CustomerSegment	Age_Transformed
95																				
9966	15606229	Obijiaku	771	France	Male	39	5	0.00	2	Yes	No	96270.64	No	No	1	DIAMOND	300	Not Eligible	Basic	6.24
9967	15569892	Johnstone	516	France	Male	35	10	57369.61	1	Yes	Yes	101699.77	No	No	5	PLATINUM	771	Not Eligible	Gold	5.92
9968	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	Yes	No	92888.52	Yes	Yes	2	GOLD	339	Not Eligible	Gold	6.48
9969	15628319	Walker	792	France	Female	28	4	130142.79	1	Yes	No	38190.78	No	No	3	DIAMOND	911	Not Eligible	Premium	5.29

9999 rows × 20 columns

Explanation:

Special characters were removed from the Surname column using a regular expression to retain only alphabetical characters. This cleaning step ensures that all customer names contain valid text values without symbols or numbers, improving data consistency and preventing issues during text analysis or reporting. After the transformation, the Surname column contains clean and standardized name values.

Dropping Unwanted Columns:

Python code:

```
dataset = dataset.drop(['RowNumber'], axis=1)

dataset
```

Output:

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	SatisfactionScore	CardType	PointEarned
0	15634602	Hargrave	619	France	Female	42	2	0.00	1	Yes	Yes	101348.88	Yes	Yes	2	DIAMOND	464
1	15647311	Hill	608	Spain	Female	41	1	83807.86	1	No	Yes	112542.58	No	Yes	3	DIAMOND	456
2	15619304	Onio	502	France	Female	42	8	159660.80	3	Yes	No	113931.57	Yes	Yes	3	DIAMOND	377
3	15701354	Boni	699	France	Female	39	1	0.00	2	No	No	93826.63	No	No	5	GOLD	350
4	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	Yes	Yes	79084.10	No	No	5	GOLD	425
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	15719294	Wood	800	France	Female	29	2	0.00	2	No	No	167773.55	No	No	4	PLATINUM	311
9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	Yes	No	96270.64	No	No	1	DIAMOND	300
9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	Yes	Yes	101699.77	No	No	5	PLATINUM	771
9998	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	Yes	No	92888.52	Yes	Yes	2	GOLD	339
9999	15628319	Walker	792	France	Female	28	4	130142.79	1	Yes	No	38190.78	No	No	3	DIAMOND	911

9999 rows × 17 columns

Explanation:

The RowNumber column was identified as an unwanted feature and removed from the dataset since it does not contribute any meaningful information for data analysis or model prediction. Dropping this column helps reduce data redundancy and improves the overall efficiency of data processing. After removal, the dataset now contains only relevant and useful features for further analysis.

Adding new column Based on CreditScore Column using apply():

Python code:

```
def eligibility(score):
  if score >= 850:
    return "Eligible"
  else:
    return "Not Eligible"


dataset['LoanEligibility'] = dataset['CreditScore'].apply(eligibility)
dataset
```

Output:

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Card Type	Point Earned	LoanEligibility
0	15634602	Hargrave	619	France	Female	42	2	0.00	1	Yes	Yes	101348.88	Yes	Yes	2	DIAMOND	464	Not Eligible
1	15647311	Hill	608	Spain	Female	41	1	83807.86	1	No	Yes	112542.58	No	Yes	3	DIAMOND	456	Not Eligible
2	15619304	Onio	502	France	Female	42	8	159660.80	3	Yes	No	113931.57	Yes	Yes	3	DIAMOND	377	Not Eligible
3	15701354	Boni	699	France	Female	39	1	0.00	2	No	No	93826.63	No	No	5	GOLD	350	Not Eligible

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Card Type	Point Earned	LoanEligibility
4	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	Yes	Yes	79084.10	No	No	5	GOLD	425	Eligible
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	15719294	Wood	800	France	Female	29	2	0.00	2	No	No	167773.55	No	No	4	PLATINUM	311	Not Eligible
9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	Yes	No	96270.64	No	No	1	DIAMOND	300	Not Eligible
9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	Yes	Yes	101699.77	No	No	5	PLATINUM	771	Not Eligible
9998	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	Yes	No	92888.52	Yes	Yes	2	GOLD	339	Not Eligible
9999	15628319	Walker	792	France	Female	28	4	130142.79	1	Yes	No	38190.78	No	No	3	DIAMOND	911	Not Eligible

9999 rows × 18 columns

Explanation:

A new column named LoanEligibility was created based on the CreditScore using the apply() function. Customers with a credit score of 850 or above were classified as “Eligible,” while all others were marked as “Not Eligible.” This feature engineering step helps in clearly identifying loan-eligible customers and supports better decision-

Adding new column Based on Balance Column using apply():

Python code:

```
def customer_segment (row):
  if row['Balance']>100000:
    return "Premium"
  elif row['Balance']>50000:
    return "Gold"
  else:
    return "Basic"
dataset['CustomerSegment'] = dataset.apply(customer_segment,axis=1)
dataset
```

Output:

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	SatisfactionScore	CardType	PointEarned	LoanEligibility	CustomerSegment
0	15634602	Hargrave	619	France	Female	42	2	0.00	1	Yes	Yes	101348.88	Yes	Yes	2	DIAMOND	464	Not Eligible	Basic
1	15647311	Hill	608	Spain	Female	41	1	83807.86	1	No	Yes	112542.58	No	Yes	3	DIAMOND	456	Not Eligible	Gold
2	15619304	Onio	502	France	Female	42	8	159660.80	3	Yes	No	113931.57	Yes	Yes	3	DIAMOND	377	Not Eligible	Premium

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Card Type	Point Earned	LoanEligibility	CustomerSegment
3	15701354	Boni	699	France	Female	39	1	0.00	2	No	No	93826.63	No	No	5	GOLD	350	Not Eligible	Basic
4	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	Yes	Yes	79084.10	No	No	5	GOLD	425	Eligible	Premium
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	15719294	Wood	800	France	Female	29	2	0.00	2	No	No	167773.55	No	No	4	PLATINUM	311	Not Eligible	Basic
9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	Yes	No	96270.64	No	No	1	DIAMOND	300	Not Eligible	Basic
9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	Yes	Yes	101699.77	No	No	5	PLATINUM	771	Not Eligible	Gold
9998	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	Yes	No	92888.52	Yes	Yes	2	GOLD	339	Not Eligible	Gold
9999	15628319	Walker	792	France	Female	28	4	130142.79	1	Yes	No	38190.78	No	No	3	DIAMOND	911	Not Eligible	Premium

9999 rows × 19 columns

Explanation:

A new column named CustomerSegment was created based on the customers' Balance using the apply() function. Customers with a balance above 1,00,000 were categorized as "Premium," those with a balance above 50,000 as "Gold," and the remaining customers as "Basic." This

segmentation helps in classifying customers based on their financial value for better business analysis and targeted strategies.

Converting Object Columns to Categorical Columns (Converting data types):

Python code:

```
categorical_col = ['Geography','Gender','Card Type','HasCrCard','IsActiveMember','Exited','Complain','LoanEligibility','CustomerSegment']

dataset[categorical_col] = dataset[categorical_col].astype('category')

print(dataset.dtypes)
```

Output:

```
CustomerId            int64
Surname              object
CreditScore           int64
Geography          category
Gender             category
Age                   int64
Tenure                int64
Balance             float64
NumOfProducts         int64
HasCrCard          category
IsActiveMember     category
EstimatedSalary     float64
Exited             category
Complain           category
Satisfaction Score    int64
Card Type          category
```

```
Point Earned          int64
LoanEligibility      category
CustomerSegment      category
dtype: object
```

Explanation:

Several object-type columns such as Geography, Gender, Card Type, HasCrCard, IsActiveMember, Exited, Complain, LoanEligibility, and CustomerSegment were converted into the **categorical data type** using astype('category'). This conversion helps reduce memory usage, improves computational efficiency, and ensures that categorical variables are treated appropriately during analysis. After conversion, the dataset clearly reflects the correct data types for each feature.

Checking Skewness:

Python code:

```
numeric_column = dataset.select_dtypes(include=['int64','float64']).columns
print("Numerical Columns:", numeric_column)


print("\n",dataset[numeric_column].skew())
```

Output:

```
Numerical Columns: Index(['CustomerId', 'CreditScore', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'EstimatedSalary', 'Satisfaction Score',
       'Point Earned'],
      dtype='object')


 CustomerId          0.001029
CreditScore         -0.071448
```

```
Age                 1.011200
Tenure              0.011165
Balance            -0.141317
NumOfProducts       0.745426
EstimatedSalary     0.001886
Satisfaction Score -0.008939
Point Earned        0.008288
dtype: float64
```

Explanation:

We check skewness because to understand data distribution, to decide if transformation is needed to improve machine learning model performance, to identify outliers and imbalance. Skewness tells us how much a data distribution is tilted.

The skewness analysis shows that most numerical features such as CreditScore, Tenure, Balance, EstimatedSalary, Satisfaction Score, and Point Earned are nearly symmetric and normally distributed. However, Age and Number of Products show positive skewness, this means More customers are younger, few customers are very old. The distribution has a long right tail. NumOfProducts is Most people have 1 or 2 products, very few customers have 3 or 4 products. This indicates a natural imbalance in customer behaviour patterns.

Applying Transformation for 'Age' Column:

Python code:

```
dataset['Age_Transformed'] = np.sqrt(dataset['Age']).round(2)

dataset
```

Output:

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Card Type	Point Earned	LoanEligibility	CustomerSegment	Age_Transformed
0	15634602	Hargrave	619	France	Female	42	2	0.00	1	Yes	Yes	101348.88	Yes	Yes	2	DIAMOND	464	Not Eligible	Basic	6.48

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Card Type	PointEarned	LoanEligibility	CustomerSegment	Age_Transformed
1	15647311	Hill	608	Spain	Female	41	1	83807.86	1	No	Yes	112542.58	No	Yes	3	DIAMOND	456	Not Eligible	Gold	6.40
2	15619304	Onio	502	France	Female	42	8	159660.80	3	Yes	No	113931.57	Yes	Yes	3	DIAMOND	377	Not Eligible	Premium	6.48
3	15701354	Boni	699	France	Female	39	1	0.00	2	No	No	93826.63	No	No	5	GOLD	350	Not Eligible	Basic	6.24
4	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	Yes	Yes	79084.10	No	No	5	GOLD	425	Eligible	Premium	6.56
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	15719294	Wood	800	France	Female	29	2	0.00	2	No	No	167773.55	No	No	4	PLATINUM	311	Not Eligible	Basic	5.39
9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	Yes	No	96270.64	No	No	1	DIAMOND	300	Not Eligible	Basic	6.24
9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	Yes	Yes	101699.77	No	No	5	PLATINUM	771	Not Eligible	Gold	5.92
9998	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	Yes	No	92888.52	Yes	Yes	2	GOLD	339	Not Eligible	Gold	6.48
9999	15628319	Walker	792	France	Female	28	4	130142.79	1	Yes	No	38190.78	No	No	3	DIAMOND	911	Not Eligible	Premium	5.29

9999 rows × 20 columns

Explanation:

Skewness values greater than ±1 indicate high skewness and require transformation, values between ±0.5 and ±1 show moderate skewness, and values below ±0.5 represent near-normal distributions. The Age feature showed significant positive skewness (1.01), indicating a right-skewed distribution. Therefore, a square root transformation was applied to normalize the data and reduce the impact of extreme values. Other numerical features were not transformed as they already followed near-normal distributions.

Highly skewed data can Reduce ML model accuracy, affect mean & variance, give biased predictions, so applied a transformation to reduce skewness, make data more normal, Reduce the effect of outliers.

EDA After Data cleaning:

Shape of the dataset

Python code:

```
print("Number of Rows and Columns:", dataset.shape)
```

Output:

Number of Rows and Columns: (9999, 20)

Dataset Features:

Python code:

```
print(dataset.columns.tolist())
```

Output:

['CustomerId', 'Surname', 'CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary', 'Exited', 'Complain', 'Satisfaction Score', 'Card Type', 'Point Earned', 'LoanEligibility', 'CustomerSegment', 'Age_Transformed']

Displaying first 10 records:

Python code:

```
print(dataset.head(10))
```

Output:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	SatisfactionScore	CardType	PointEarned
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1	1	2	DIAMOND	464
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0	1	3	DIAMOND	456
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1	1	3	DIAMOND	377
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0	0	5	GOLD	350
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0	0	5	GOLD	425
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	1	1	5	DIAMOND	484
6	7	15592531	Bartlett	822	France	Male	50	7	0.00	2	1	1	10062.80	0	0	2	SILVER	206
7	8	15656148	Obinna	376	Germany	female	29	4	115046.74	4	1	0	119346.88	1	1	2	DIAMOND	282
8	9	15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.50	0	0	3	GOLD	251
9	10	15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.73	0	0	3	GOLD	342

Displaying Last 10 records:

Python code:

```
print(dataset.tail(10))
```

Output:

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Card Type	PointEarned	LoanEligibility	CustomerSegment	Age_Transformed
9990	15605622	McMillan	841	Spain	Male	28	4	0.00	2	Yes	Yes	179436.60	No	No	5	GOLD	393	Not Eligible	Basic	5.29
9991	15798964	Nkemakonam	714	France	Male	33	3	35016.60	1	Yes	No	53667.08	No	No	3	GOLD	791	Not Eligible	Basic	5.74
9992	15769959	Ajuluchukwu	597	France	Female	53	4	88381.21	1	Yes	No	69384.71	Yes	Yes	3	GOLD	369	Not Eligible	Gold	7.28
9993	15657105	Chukwualuka	726	Spain	Male	36	2	0.00	1	Yes	No	195192.40	No	No	5	SILVER	560	Not Eligible	Basic	6.00
9994	15569266	Rahman	644	France	Male	28	7	155060.41	1	Yes	No	29179.52	No	No	5	DIAMOND	715	Not Eligible	Premium	5.29
9995	15719294	Wood	800	France	Female	29	2	0.00	2	No	No	167773.55	No	No	4	PLATINUM	311	Not Eligible	Basic	5.39
9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	Yes	No	96270.64	No	No	1	DIAMOND	300	Not Eligible	Basic	6.24
9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	Yes	Yes	101699.77	No	No	5	PLATINUM	771	Not Eligible	Gold	5.92
9998	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	Yes	No	92888.52	Yes	Yes	2	GOLD	339	Not Eligible	Gold	6.48

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Card Type	Point Earned	Loan Eligibility	CustomerSegment	Age_Transformed
9999	15628319	Walker	792	France	Female	28	4	130142.79	1	Yes	No	38190.78	No	No	3	DIAMOND	911	Not Eligible	Premium	5.29

Statistical summary:

Python code:

```
print(dataset.describe())
```

Output:

	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	Satisfaction Score	Point Earned	Age_Transformed
count	9.999000e+03	9999.000000	9999.000000	9999.000000	9999.000000	9999.000000	9999.000000	9999.000000	9999.000000	9999.000000
mean	1.569095e+07	650.522952	38.922092	5.012601	76493.538642	1.530253	100096.040927	3.013801	606.519352	6.185379
std	7.193191e+04	96.656363	10.488290	2.892251	62395.836502	0.581659	57510.442886	1.405989	225.935737	0.812155
min	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	11.580000	1.000000	119.000000	4.240000
25%	1.562854e+07	584.000000	32.000000	3.000000	0.000000	1.000000	51013.655000	2.000000	410.000000	5.660000
50%	1.569074e+07	652.000000	37.000000	5.000000	97208.460000	1.000000	100200.400000	3.000000	605.000000	6.080000
75%	1.575324e+07	718.000000	44.000000	7.000000	127646.040000	2.000000	149392.065000	4.000000	801.000000	6.630000
max	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	199992.480000	5.000000	1000.000000	9.590000

Info about the dataset:

Python code:

```
dataset.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
Index: 9999 entries, 0 to 9999
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   CustomerId        9999 non-null   int64
 1   Surname           9999 non-null   object
 2   CreditScore       9999 non-null   int64
 3   Geography         9999 non-null   category
 4   Gender            9999 non-null   category
 5   Age               9999 non-null   int64
 6   Tenure            9999 non-null   int64
 7   Balance           9999 non-null   float64
 8   NumOfProducts     9999 non-null   int64
 9   HasCrCard         9999 non-null   category
 10  IsActiveMember    9999 non-null   category
 11  EstimatedSalary   9999 non-null   float64
 12  Exited            9999 non-null   category
 13  Complain          9999 non-null   category
 14  Satisfaction Score  9999 non-null   int64
 15  Card Type         9999 non-null   category
 16  Point Earned      9999 non-null   int64
 17  LoanEligibility   9999 non-null   category
```

18 CustomerSegment 9999 non-null category

19 Age_Transformed 9999 non-null float64

dtypes: category(9), float64(3), int64(7), object(1)

memory usage: 1.0+ MB

Check for missing values:

Python code:

```
dataset.isnull().sum()
```

Output:

	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

	0
Point Earned	0
LoanEligibility	0
CustomerSegment	0
Age_Transformed	0

dtype: int64

Check for Duplicate Records:

Python code:

```
print("Number of Duplicate Records:", dataset.duplicated().sum())
```

Output:

Number of Duplicate Records: 0

Total Number of elements:

Python code:

```
dataset.size
```

Output:

199980

Displaying Datatypes of dataset:

Python code:

```
print(dataset.dtypes)
```

Output:

CustomerId int64

Surname object

```
CreditScore            int64
Geography              category
Gender                 category
Age                    int64
Tenure                 int64
Balance                float64
NumOfProducts          int64
HasCrCard              category
IsActiveMember         category
EstimatedSalary        float64
Exited                 category
Complain               category
Satisfaction Score     int64
Card Type              category
Point Earned           int64
LoanEligibility        category
CustomerSegment        category
Age_Transformed        float64
dtype: object
```

Stage 3 – EDA and Visualizations

Univariate Analysis → distribution of single variables (countplot, histogram, boxplot)

Bivariate Analysis → relation between two variables (scatterplot, barplot, correlation heatmap)

Multivariate Analysis → relation among 3+ variables (pairplot, grouped analysis, pivot tables, advanced plots)

Interpretation MUST with every visualization

Focus on business story not just charts

Each chart must be in separate cells.

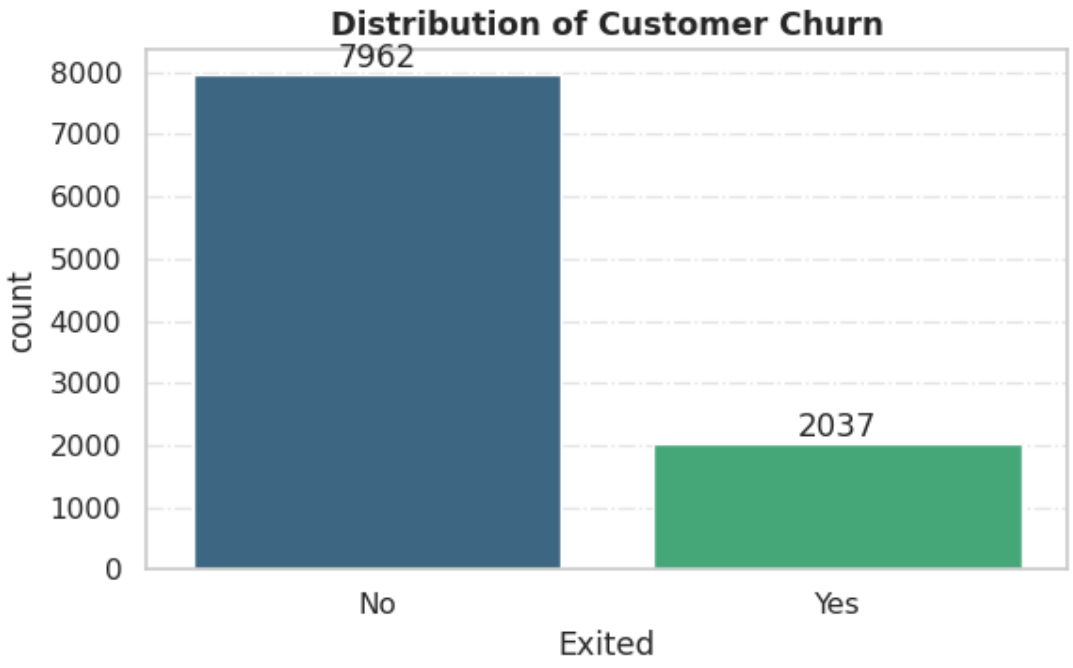
Univariate Analysis: Distribution of customer churn

Python code:

```
plt.figure(figsize=(6,4))
ax=sns.countplot(x='Exited',
            hue='Exited',
            palette ='viridis',
            data=dataset
            )
for container in ax.containers:
  ax.bar_label(container)

plt.title("Distribution of Customer Churn",fontweight='bold')
plt.grid(axis='y',linestyle='-.',alpha=0.5)
plt.tight_layout()
plt.show()
```

Output:



checking count:

Python code:

```
print(dataset['Exited'].value_counts().reset_index())
```

Output:

```
  Exited  count
0     No   7962
1    Yes   2037
```

Interpretation of Customer churn count plot chart:

This Chart is to display the distribution of bank customer churn,

No = Customer is continuing their accounts with the bank

Yes = Customer left the bank

This chart is to visually compare how many customers stayed and how many left

what it is saying:

This chart clearly shows, large number of customers did not churn (7962), only a smaller portion of customers churned (2037).

The churn rate is significantly lower than the retention rate.

Features used:

Only one Feature 'Exited' is used for this univariate analysis. palette is used for colour preferences, the same Feature only used for hue variable.

From this what you are showing us:

This chart helps to show churn distribution of bank customer base, and helps to answer the below questions,

How many Customers left?

How many Customers Retained?

Is churn a serious issue?

It helps Bank Seniors to understand the severity of the churn problem.

Overall, most customers are retained, but a noticeable portion has Exited. This shows the bank needs to focus on reducing customer churn to improve long-term profitability.

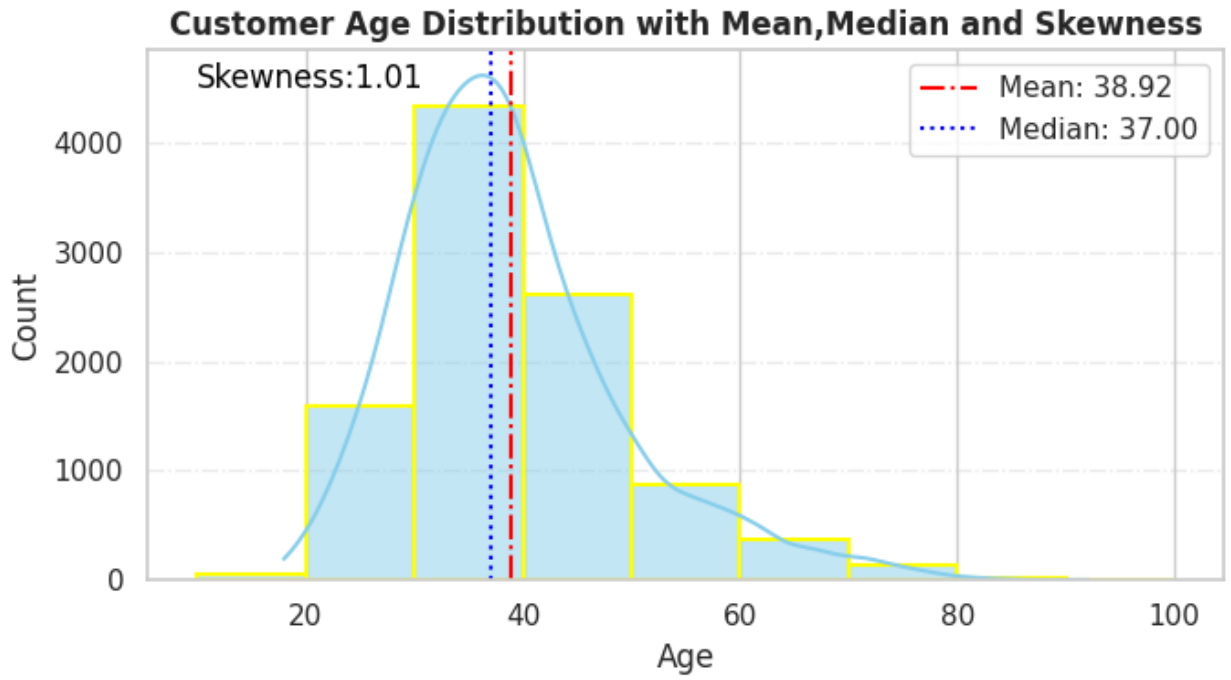
Univariate Analysis: Distribution of Age with Mean, Median and skewness.

Python code:

```
age_mean = dataset['Age'].mean()
age_median = dataset['Age'].median()
age_skew = dataset['Age'].skew()


plt.figure(figsize=(7,4))
sns.histplot(dataset['Age'],
             bins=range(10, 101, 10),
             kde=True,
             color='skyblue',
             edgecolor='yellow',
             linewidth=1.5)
plt.axvline(age_mean, color='red', linestyle='-.',label=f'Mean: {age_mean:.2f}')
plt.axvline(age_median, color='blue', linestyle=':', label=f'Median: {age_median:.2f}')
plt.text(10,4500,"Skewness:" + str(round(age_skew,2)), color='Black')
plt.title("Customer Age Distribution with Mean,Median and Skewness", fontweight='bold')
plt.xlabel("Age")
plt.ylabel("Count")
plt.grid(axis='y', linestyle='-.', alpha=0.4)
plt.legend()
plt.tight_layout()
plt.show()
```

Output:



Interpretation of Distribution of Age chart

The chart is a Histogram with KDE curve displaying how customer age distributed. Each Histogram bar displaying the number of customers distributed in each range like 10-20, 20-30,30-40 etc. KDE curve represents the density of the ages, red line (-.) represents the mean age, Blue dotted line (.) represents the median (central tendency) of age, skewness value shows is biased towards younger or older age groups.

What is it saying?

The chart displays the which age group has the most customers. Mean and Median is to find the skewness of age Distribution. In this case Mean>Median which is distribution is Right skewed, which means younger groups are higher than old age group.

What features you used?

This is univariate Analysis so only "Age" Feature is used. Mean,Median and skewness are derived from age column.

From this what you are showing us?

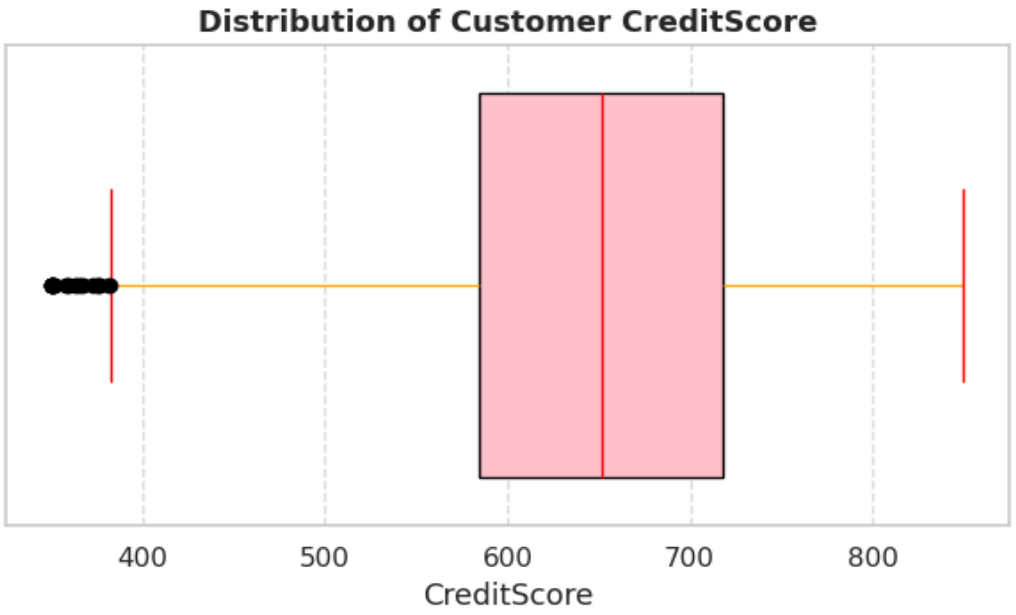
This chart is showing Distribution pattern of customer age,mean and median,and skewness. If most customers are in younger age groups banks can target them for marketing.

Univariate Analysis: Distribution of Customer CreditScore

Python code:

```
plt.figure(figsize=(6,4))
sns.boxplot(x=dataset['CreditScore'],
        boxprops=dict(facecolor='pink', edgecolor='black'),
        whiskerprops=dict(color='orange'),
        capprops=dict(color='red'),
        medianprops=dict(color='red'),
        flierprops=dict(markerfacecolor='black', markeredgecolor='black')
        )
plt.title("Distribution of Customer CreditScore", fontweight='bold')
plt.xlabel("CreditScore")
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

Output:



Interpretation of Distribution of CreditScore chart

This boxplot visualizes how customer credit scores are distributed. The pink box represents the central 50% of the credit scores, and the red vertical line inside the box marks the median score. The orange whiskers show how far the credit scores stretch toward the minimum and maximum values.

The chart indicates that the median credit score is around 650, meaning half of the customers have a score below 650 and half have a score above. The whiskers extend roughly from 580 to 850, showing that most customers have moderate to high credit scores. The presence of black dots on the lower end indicates a few customers with very low credit scores — these are outliers.

Features used:

This is a Univariate Analysis, using only one feature — CreditScore.

What this shows us:

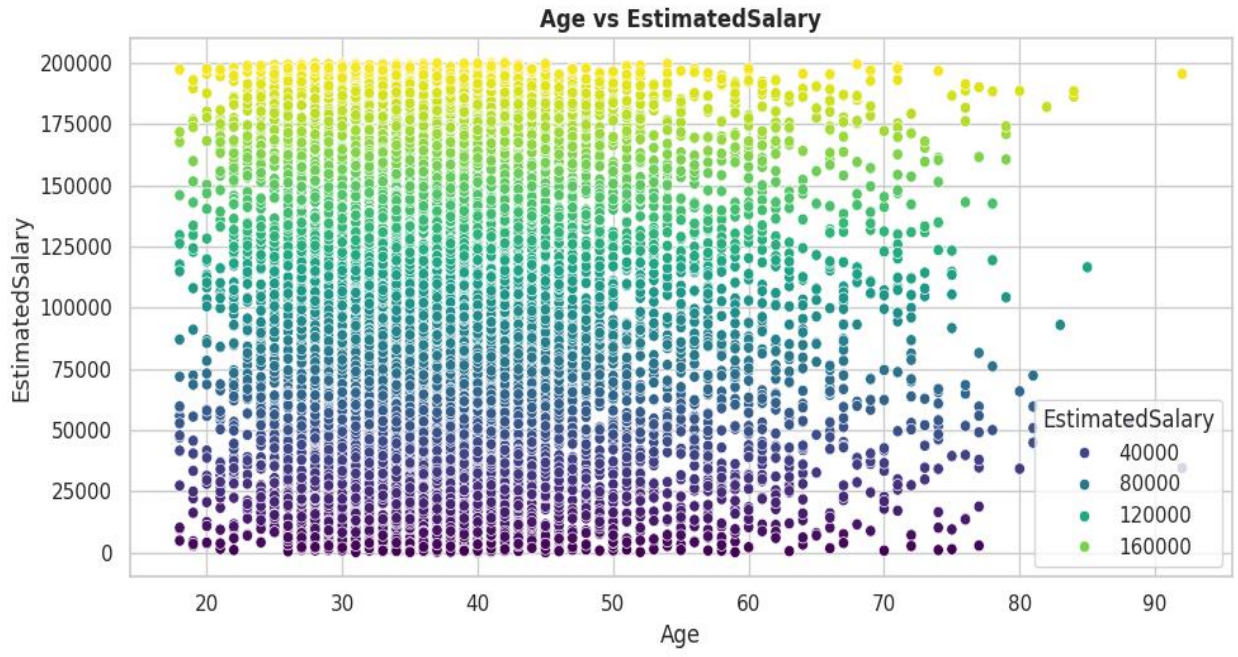
This chart helps us understand the overall creditworthiness of customers. The spread shows that most customers have moderate to high credit scores, but a small group has very low scores, which may indicate higher risk. Such insights help the bank identify high-risk customers and make better lending or retention decisions.

Bivariate Analysis: Age vs EstimatedSalary

Python code:

```
plt.figure(figsize=(10,5))
sns.scatterplot(data=dataset,
                x='Age',
                y='EstimatedSalary',
                hue='EstimatedSalary',
                palette='viridis',
                marker='o')
plt.title("Age vs EstimatedSalary",fontweight='bold')
plt.xlabel("Age")
plt.ylabel("EstimatedSalary")
plt.tight_layout()
plt.show()
```

Output:



Interpretation of Scatter Plot chart (Age vs EstimatedSalary)

This scatter plot showing the relationship between Age and Estimated Salary of bank customers.

Each point represents a customer, and the colour scale represents the level of customer's salary.

What is it saying?

This chart tells, there is no trend, such as increasing salary with age and decreasing salary with age, salary appears to be evenly distributed, all age customers have a wide range of salary. In this dataset age is not influencing the salary.

What features you used?

For this visualization, Age and EstimatedSalary features are used.

From this what you are showing us?

Age has no correlation with estimated salary in this dataset, because there is no meaningful relationships with these two variables, this pair is not a strong indicator for predicting churn.

Average Balance by Geography:

Python code:

```
geo_avg_balance = dataset.groupby('Geography')['Balance'].mean().round(2)

print(geo_avg_balance)
```

Output:

```
Geography

France      62120.45

Germany    119768.23

Spain       61818.15

Name: Balance, dtype: float64
```

Bivariate Analysis: Geography vs Balance

Python code:

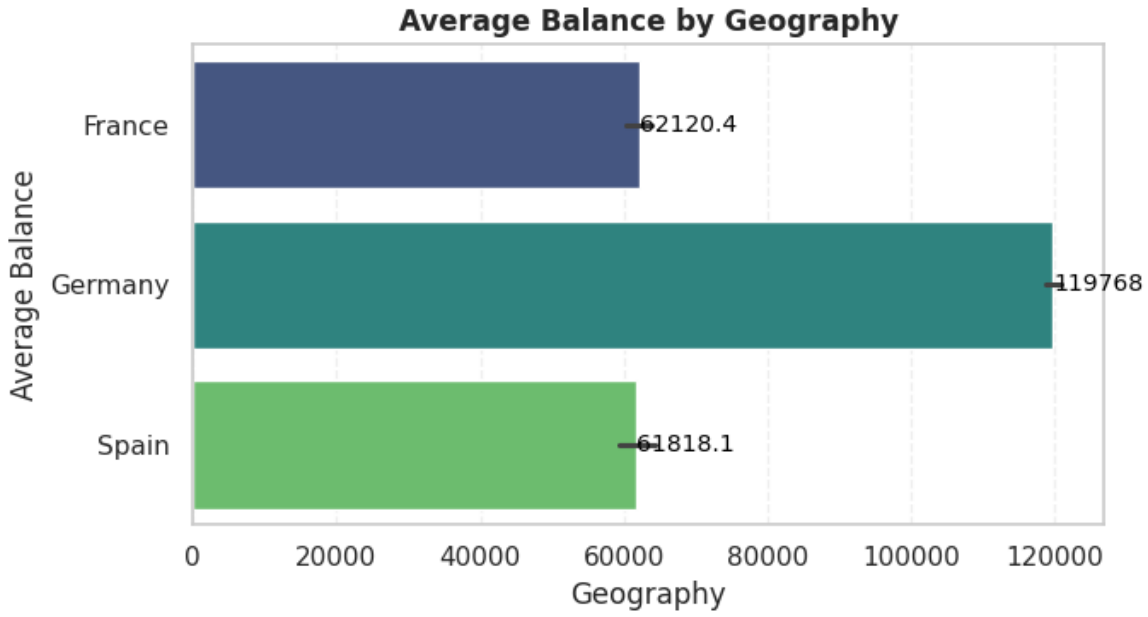
```
plt.figure(figsize=(7,4))

ax=sns.barplot(
        data=dataset,
        x='Balance',
        y='Geography',
        estimator='mean',
        hue='Geography',
        legend=False,
        palette='viridis'
)

for container in ax.containers:
  ax.bar_label(container, fontsize=10, color='black')

plt.title("Average Balance by Geography", fontweight='bold')
plt.xlabel("Geography")
plt.ylabel("Average Balance")
plt.grid(axis='x', linestyle='--', alpha=0.3)
plt.tight_layout()
plt.show()
```

Output:



Interpretation of bar plot chart (Geography vs Balance)

The bar plot visualizes the average account balance of customers grouped by their geographical location. Each bar represents the mean balance for a specific country, a quick visual comparison of balances across regions.

What is it saying?

Customers in Germany maintain the highest average account balance, Customers in France have a moderate average balance and Customers in Spain hold the lowest average balance. This indicates that financial engagement differs by region, which could potentially influence churn behaviour.

What features you used?

One numerical feature Balance and one categorical feature Geography used for this Bivariate analysis using bar plot.

From this what you are showing us?

The chart highlights regional differences in bank account balances. It suggests that Geography can be an important factor in analyzing customer behaviour and predicting churn. Regions with higher or lower average balances may show different financial engagement patterns with the bank.

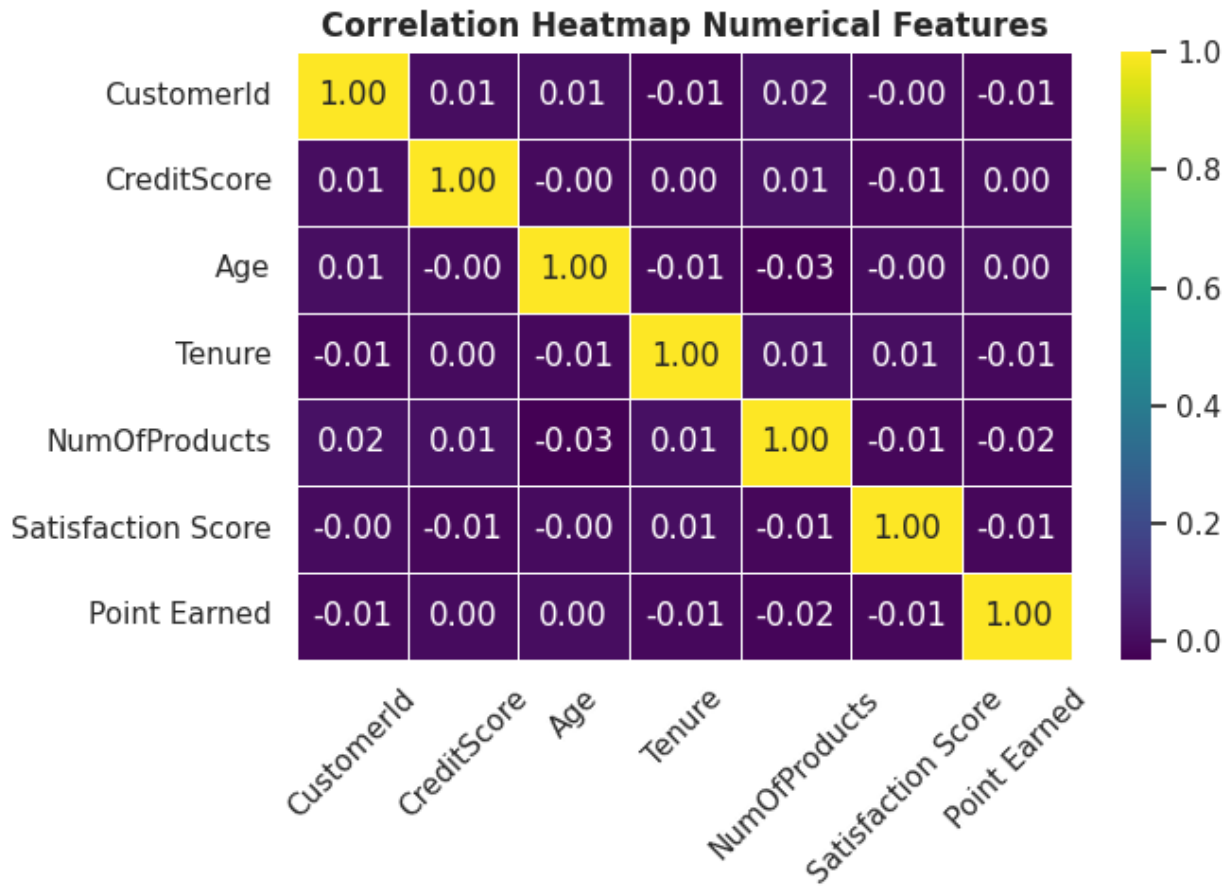
Bivariate Analysis:Correlation Heatmap Numerical Features

Python code:

```
ncol=dataset.select_dtypes('n').columns
corr_matrix = dataset[ncol].corr()
corr_matrix

plt.figure(figsize=(7,5))
sns.heatmap(
        corr_matrix,
        annot=True,
        fmt=".2f",
        cmap='viridis',
        linewidth=0.5,
        linecolor='white'
)
plt.title("Correlation Heatmap Numerical Features",fontweight='bold')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Output:



Interpretation of Correlation Heatmap Chart

The heatmap visualizes the pairwise correlation between all numerical features in the dataset. Each cell shows the Pearson correlation coefficient between two numerical variables.

What is it saying?

No strong linear relationships exist among these numerical features.

Each feature behaves independently, so multicollinearity is not a concern.

What features you used?

All numerical Features in the Dataset, CustomerId, CreditScore, Age, Tenure, NumOfProducts, Satisfaction Score, Point Earned.

From this what you are showing us?

This heatmap highlights the pairwise linear relationships between numerical features. It shows that these features are largely independent, meaning each can provide unique information for further analysis.

Customer Churn Count Across Different Features:

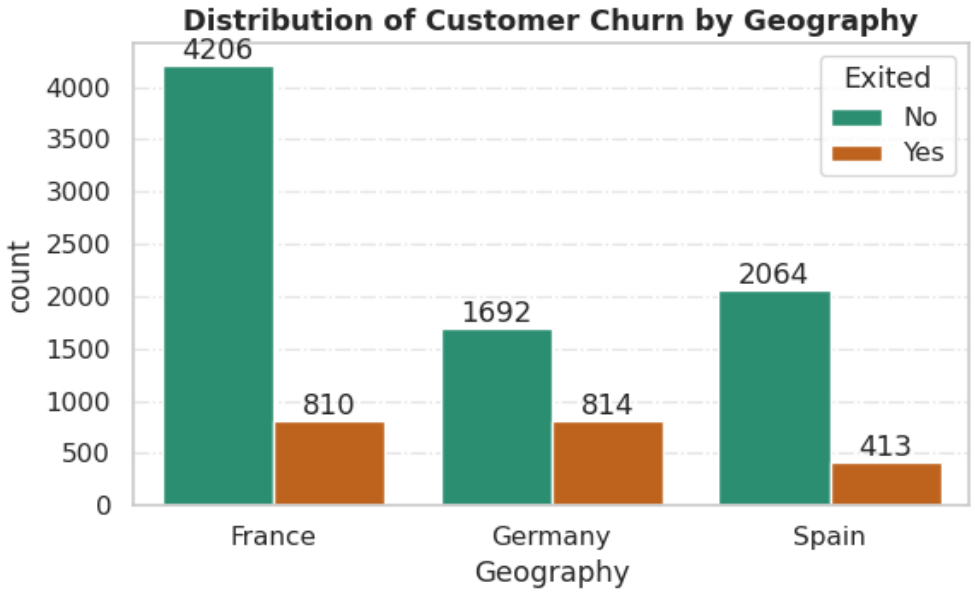
Python code:

```
def churn_countplot(column_name, data, palette):
 plt.figure(figsize=(6,4))
 ax=sns.countplot(x=column_name,
          hue='Exited',
          data=dataset,
          palette=palette)
 for container in ax.containers:
   ax.bar_label(container)
 plt.title(f"Distribution of Customer Churn by {column_name}",fontweight='bold')
 plt.grid(axis='y',linestyle='-.',alpha=0.5)
 plt.tight_layout()
 plt.show()
```

Distribution of Customer Churn by Geography:

Python code:

```
churn_countplot('Geography', dataset, 'Dark2')
```



Interpretation for Customer churn by Geography Chart

This bar chart shows the distribution of customer churn Exited vs Not Exited across different Geographies France, Germany, Spain.

France has the highest number of customers overall, with 4206 not exited and 810 exited.

Germany has fewer customers than France, but the number of exited customers (814) is slightly higher than France's, despite having fewer total customers.

Spain has a moderate total customer count, with 2064 not exited and 413 exited.

The proportion of customers who exited is highest in Germany relative to its total customers, followed by France, then Spain.

Geography and Exited Features are Used.

This chart shows How customer churn varies by geography. Identifies regions with higher churn risk, which can help target retention strategies.

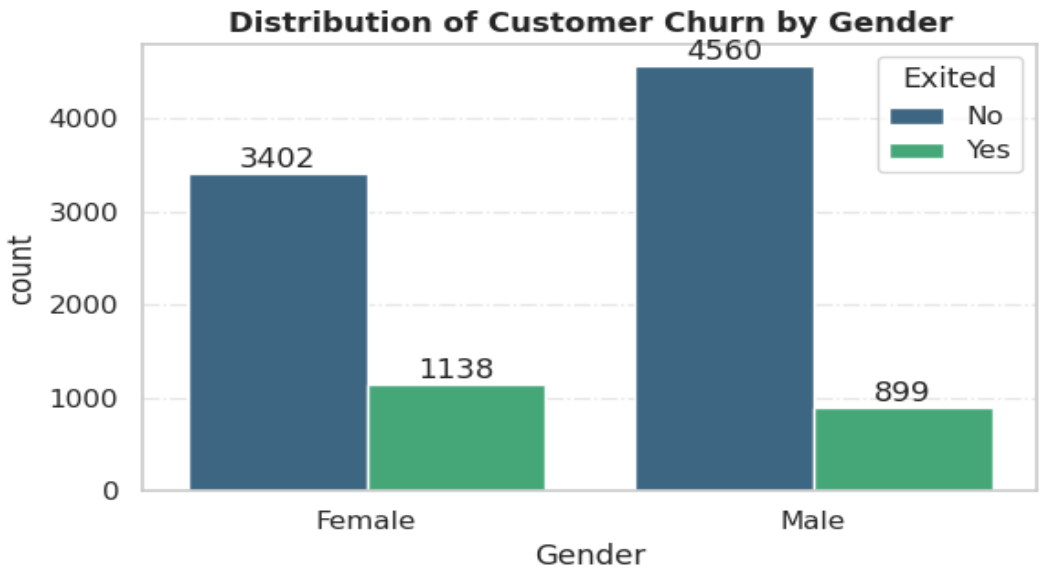
Overall, Some regions show higher churn compared to others. The bank should focus more on customer satisfaction and services in high-churn locations.

Distribution of Customer Churn by Gender:

Python code:

```
churn_countplot('Gender', dataset, 'viridis')
```

Output:



Interpretation for Customer churn by Gender Chart

This bar chart shows the distribution of customer churn Exited vs Not Exited across Gender.

Female customers 3402 not exited and 1138 exited. And Male customers 4560 not exited and 899 exited. Although male customers have a higher total count (5459), the number of churned customers is higher among females than males.

This indicates that female customers show a higher churn tendency compared to male customers in this dataset.

Gender and Exited Features are Used.

The charts show the relationships between gender and churn behaviour. It helps identify which gender group has a higher risk of churn, which is useful for targeted retention strategies.

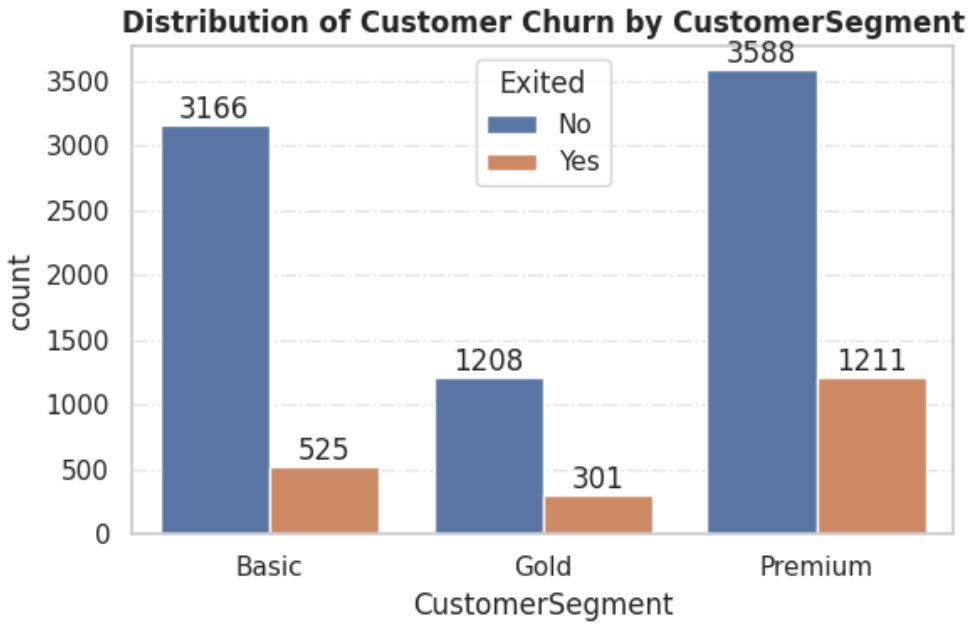
Overall, Churn is slightly higher in one gender group, showing that customer behaviour differs by gender. Targeted offers can help reduce churn in the high-risk group.

Distribution of Customer Churn by CustomerSegment:

Python code:

```
churn_countplot('CustomerSegment', dataset, 'deep')
```

Output:



Interpretation for Customer churn by CustomerSegment Chart

This bar chart shows the distribution of customer churn Exited vs Not Exited across CustomerSegment Basic, Gold, Premium.

From Basic Segment 3166 customers did not churn and 525 customers churned, from Gold Segment 1208 customers did not churn and 301 customers churned and from Premium Segment 3588 customers did not churn 1211 customers churned. The Premium segment has the highest number of churned customers, indicating that churn risk is highest in this segment. The Gold segment shows the lowest churn count, making it the most stable segment among the three.

CustomerSegment and Exited Features are Used.

The chart shows the relationship between customer segment and churn behaviour. It helps identify which customer segment is at higher risk of churn, which is useful for designing targeted retention strategies.

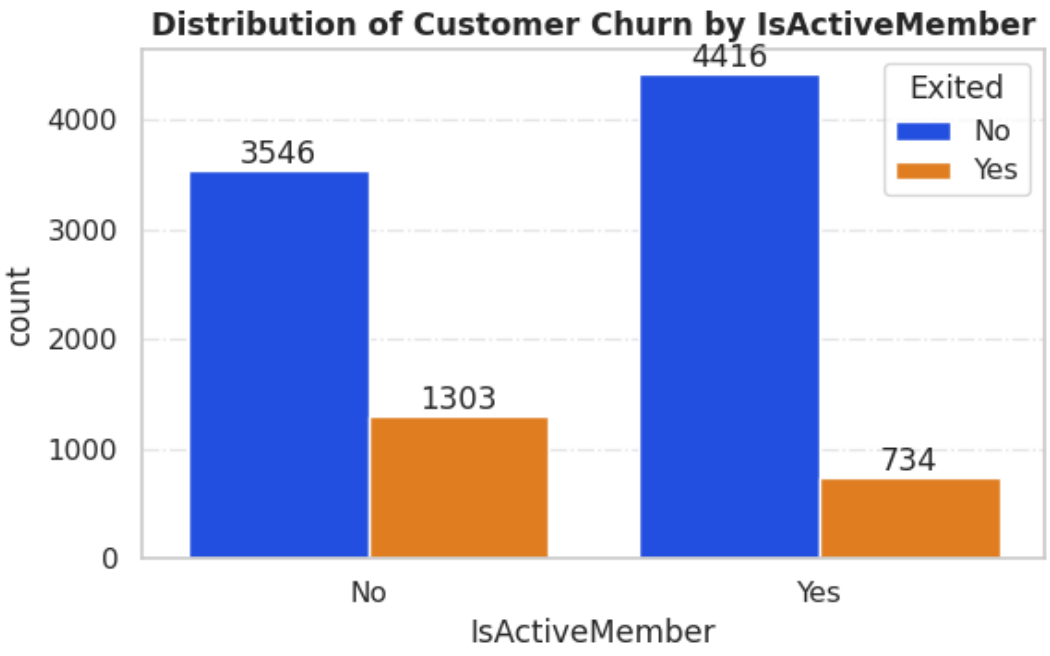
Overall, Basic customers show higher churn compared to Premium customers. The bank should focus on upgrading basic customers to higher-value segments.

Distribution of Customer Churn by IsActiveMember:

Python code:

```
churn_countplot('IsActiveMember', dataset, 'bright')
```

Output:



Interpretation for Customer churn by IsActiveMember Chart

This bar chart shows the distribution of customer churn Exited vs Not Exited based on whether the customer is Active Member or Not.

Non_Active customers 3546 not exited and 1303 exited. And Active customers 4416 not exited and 734 exited. Non-active members have a significantly higher churn count compared to active members. Active customers are more loyal, as shown by their lower churn count and higher retention.

IsActiveMember and Exited Features are Used.

The charts show the impact of Customer engagement on churn behaviour. This chart clearly shows that the inactive customers are at a much higher risk of churning, making this a critical factor for retention analysis.

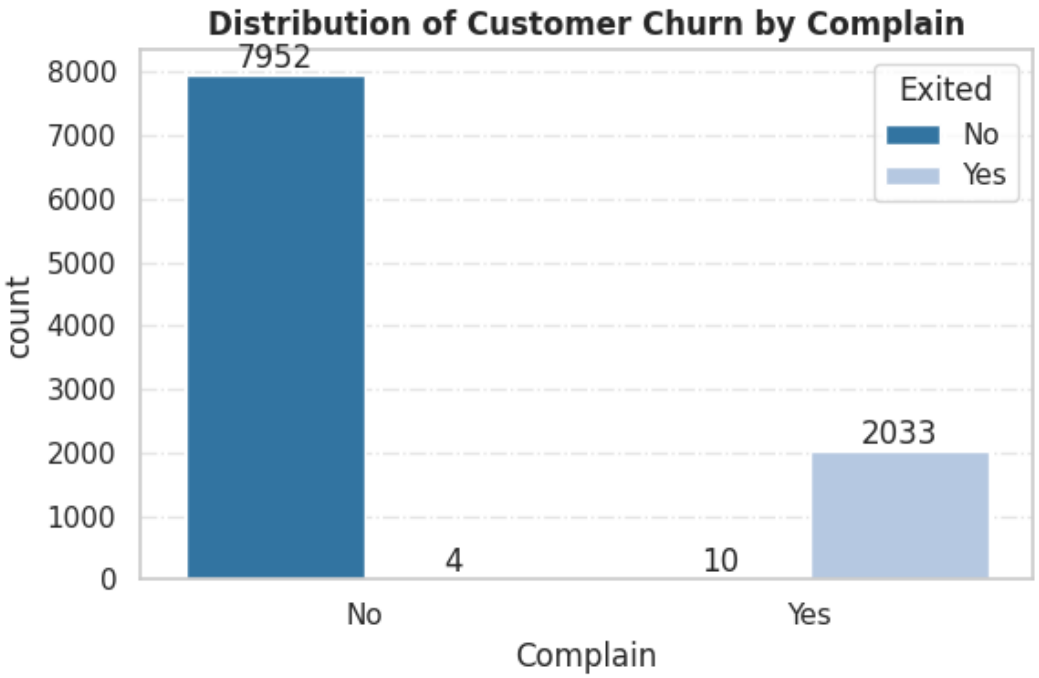
So, Inactive members have a much higher churn rate. This shows that improving customer engagement can significantly reduce churn.

Distribution of Customer Churn by Complain:

Python code:

```
churn_countplot('Complain', dataset, 'tab20')
```

Output:



Interpretation for Customer churn by Complain Chart

This bar chart shows the distribution of customer churn Exited vs Not Exited based on whether the customer has raised a complaint or not.

Customers with No complaints 7952 did not exited and only 4 exited. And customers with complaints only 10 not exited and 2033 were exited. Customers who raised a complaint have an extremely high churn count, while customers with no complaints are highly retained.

This clearly shows that complaints are a very strong indicator of customer churn.

Complain and Exited Features are Used.

The Chart shows strong relationship between customer complaints and churn behaviour. It highlights that customers who raise complaints are at very high risk of leaving the bank, making complaint handling and improving customer service a critical factor for customer retention.

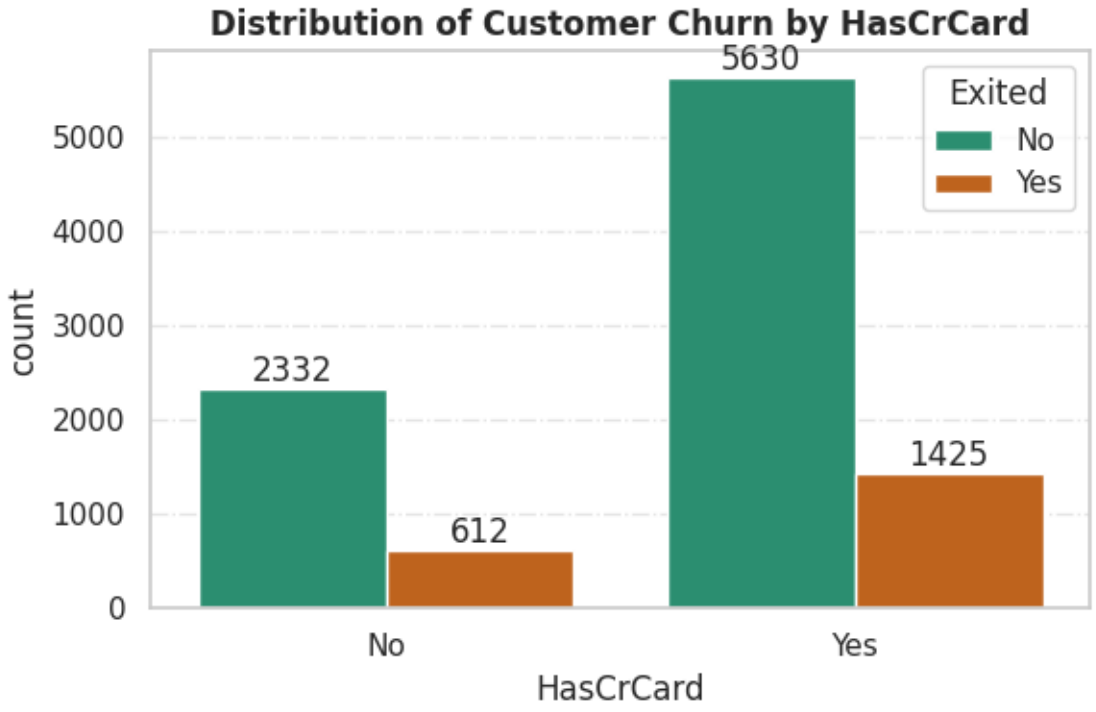
So, Customers who raised complaints have a very high churn rate. Faster complaint resolution is crucial to prevent customer loss.

Distribution of Customer Churn by HasCrCard:

Python code:

```
churn_countplot('HasCrCard', dataset, 'Dark2')
```

Output:



Interpretation for Customer churn by HasCrCard Chart

This bar chart shows the distribution of customer churn Exited vs Not Exited based on whether the customer has a Credit Card or not.

The number of customers with a credit card is much higher, they also show a higher absolute churn count. However, in both groups, the number of retained customers is significantly higher than churned customers, indicating that having a credit card alone does not strongly drive churn.

HasCrCard and Exited Features are Used.

The Chart shows strong relationship between credit card ownership and churn behaviour. It shows that credit card ownership alone is not a strong churn indicator, and other factors must be considered for churn prediction.

Overall, Customers without credit cards show slightly higher churn. Offering easy credit card options may help improve retention.

Customer Churn Across Different Features:

Python code:

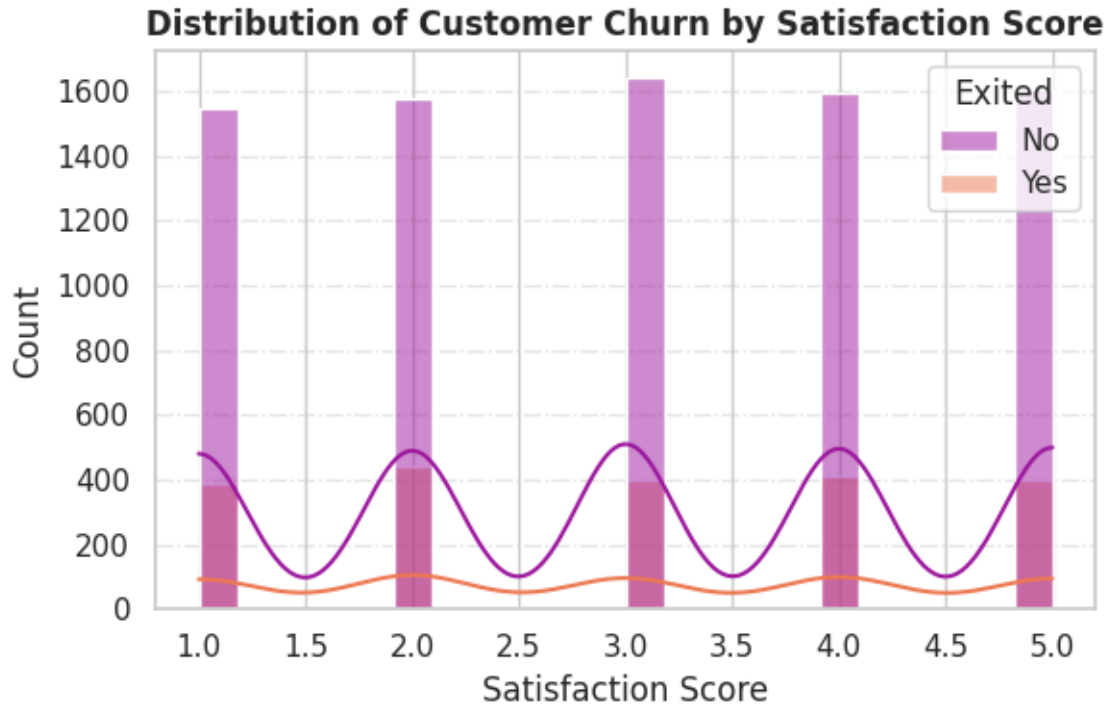
```
def continous_plot(column_name, data, palette):
  plt.figure(figsize=(6,4))
  sns.histplot(x=column_name,
               hue='Exited',
               palette =palette,
               data=dataset,
               kde=True
               )
  plt.title(f"Distribution of Customer Churn by {column_name}",fontweight='bold')
  plt.grid(axis='y',linestyle='-.',alpha=0.5)
  plt.tight_layout()
  plt.show()
```

Distribution of Customer Churn by Satisfaction Score

Python code:

```
continous_plot('Satisfaction Score', dataset, 'plasma')
```

Output:



Interpretation for Histplot Chart

This Chart shows how customer Satisfaction score is related to churn exited vs not exited.

Most of the customers have high satisfaction score and majority of these customers did not exit. In the lower satisfaction range, exited customers are slightly more compared to higher satisfaction range customers. This indicates customers with very low satisfaction are more likely to churn. Very few customers exited even with higher satisfaction.

Overall, Customers with high satisfaction stays loyal.

Satisfaction Score and Exited Features used.

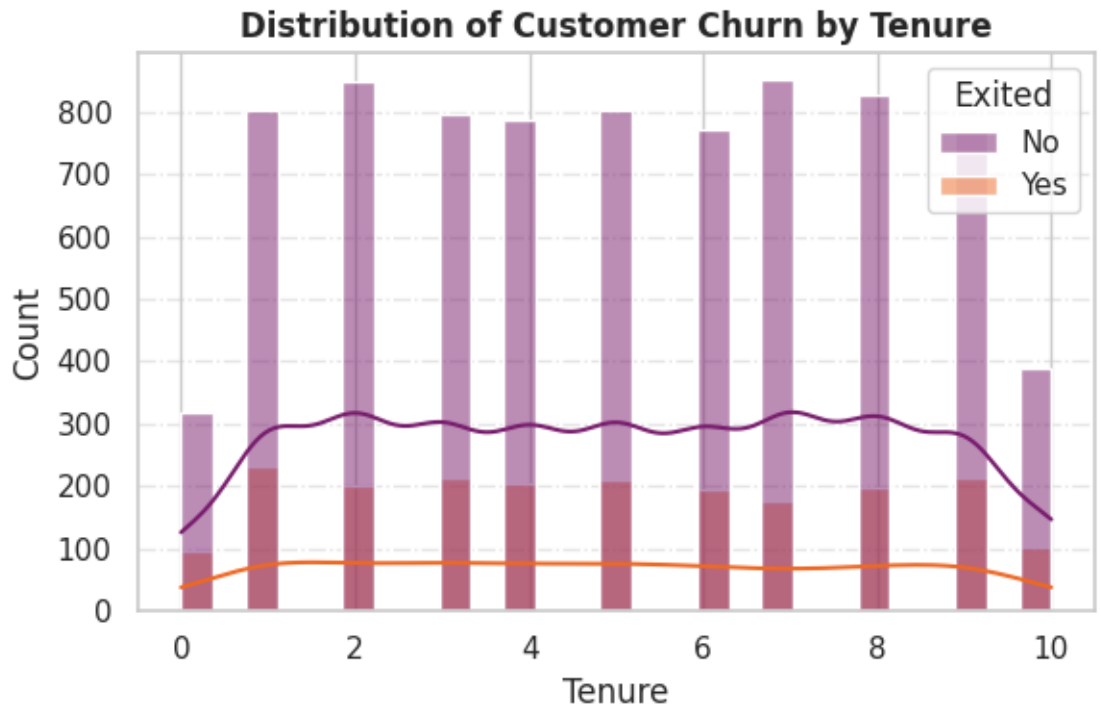
The chart shows the relationship between customer satisfaction and churn behaviour. It highlights that even highly satisfied customers can churn, meaning churn is influenced by multiple factors beyond satisfaction.

Distribution of Customer Churn by Tenure

Python code:

```
continous_plot('Tenure', dataset, 'inferno')
```

Output:



Interpretation of Distribution of churn by Tenure chart

This chart shows the distribution of customer churn Exited vs Not Exited based on customer Tenure.

This Chart shows, churn is present across all tenure levels, but longer-tenure customers show better retention compared to new customers.

Tenure and Exited Features used.

The chart shows the relationship between customer tenure and churn behaviour. It highlights that new and mid-tenure customers contribute more to churn compared to long-tenure customers, which is important for designing retention strategies.

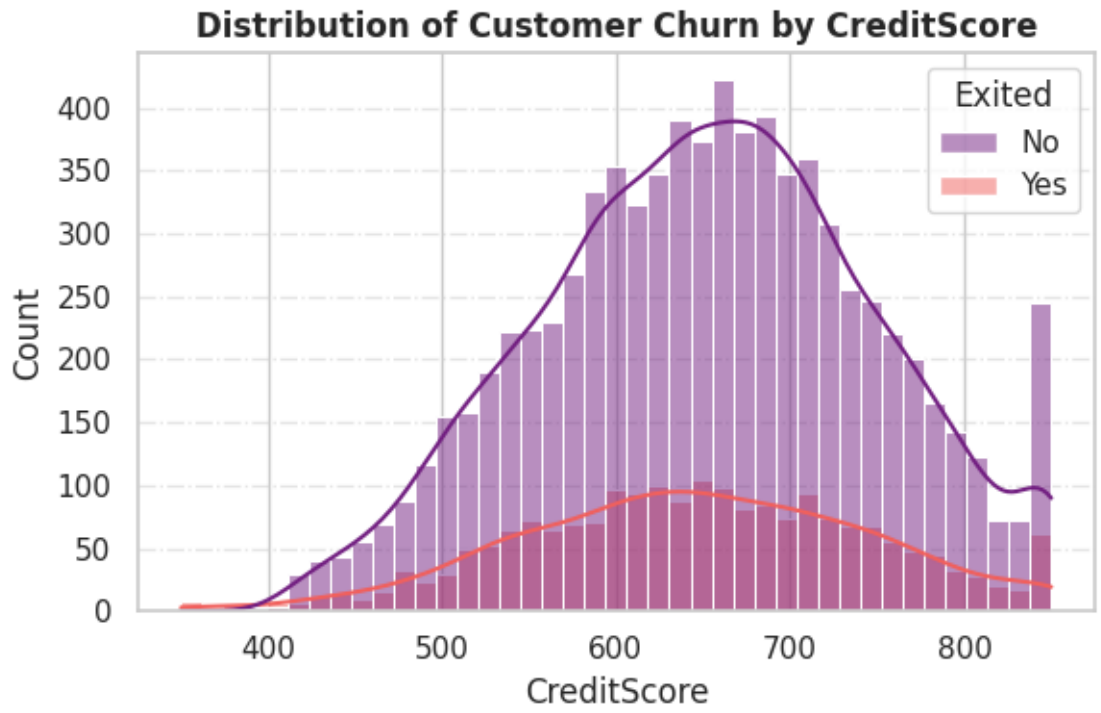
Overall, Customers with lower tenure are more likely to leave the bank. Strong onboarding and early engagement can help reduce early churn.

Distribution of Customer Churn by CreditScore

Python code:

```
continous_plot('CreditScore', dataset, 'magma')
```

Output:



Interpretation of Churn distribution by CreditScore Chart

This chart shows the distribution of customer churn Exited vs Not Exited based on customer Credit Score.

The Median CreditScore of both churned and not churned customers are approximately equal.

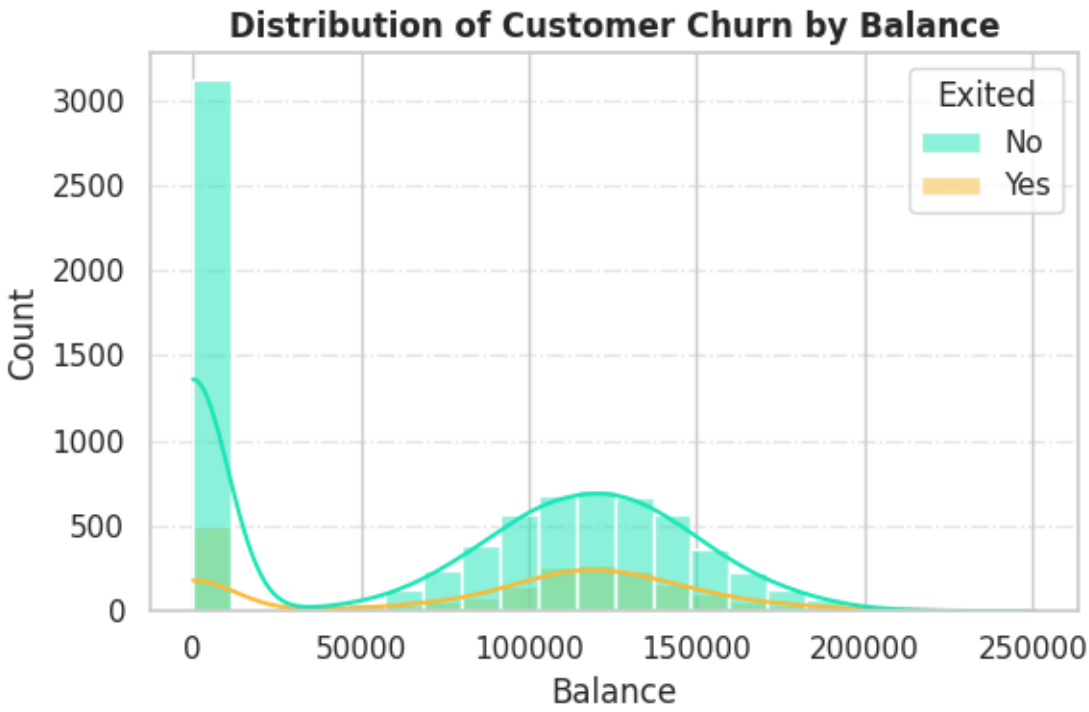
CreditScore and Exited Features used.

Since the values are approximately equal for both churn status we can't generate any relevant inference.

Distribution of Customer Churn by Balance

Python code:

```
continous_plot('Balance', dataset, 'turbo')
```



Interpretation of Churn distribution by Balance Chart

This chart shows the distribution of customer churn Exited vs Not Exited based on customer Account Balance.

More than 3000 customers are having their account balance equal to zero. Customers with zero balance are more likely to deactivate their account and more likely to churn. And balance between 1,00,000 and 1,50,000 customers also churned lower than zero balance customers and higher than other balances.

CreditScore and Exited Features used.

Overall, the chart shows the customers with zero balance and customers with balance around 1,20,000 also churned lower compared to zero balance and higher compared to other balance range, which is important for designing retention strategies.

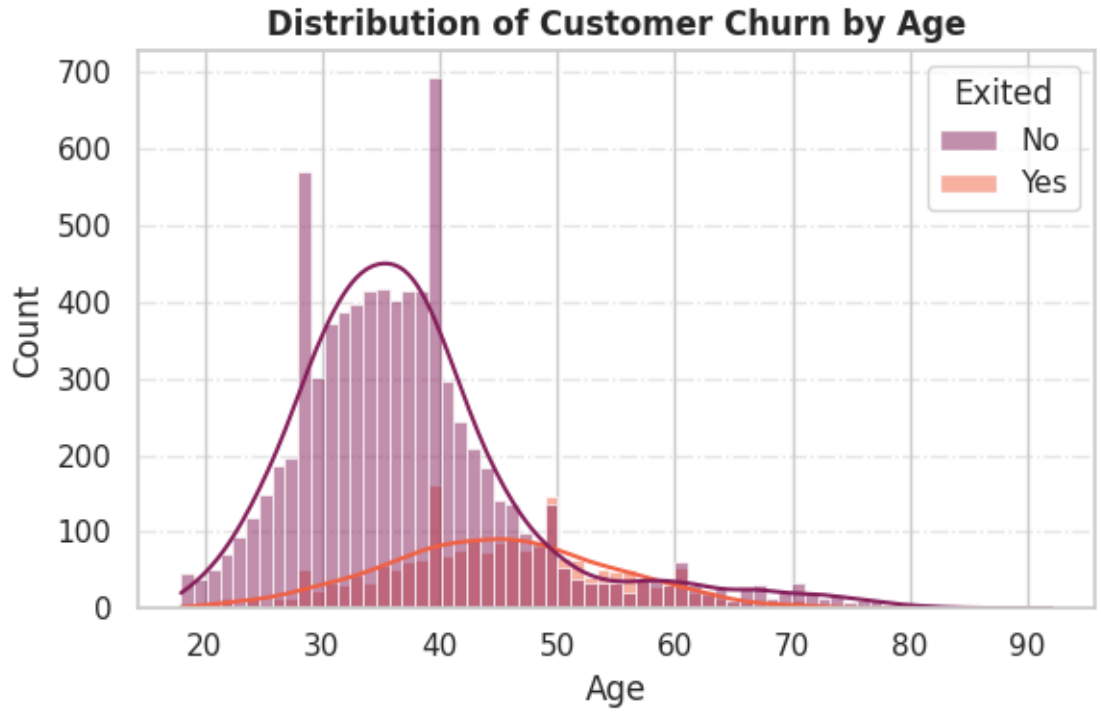
So, Customers with low balance have a higher chance of churning. The bank should encourage customers to maintain higher balances through reward programs.

Distribution of Customer Churn by Age

Python code:

```
continous_plot('Age', dataset, 'rocket')
```

Output:



Interpretation of Churn distribution by Age Chart

This chart shows the distribution of customer churn Exited vs Not Exited based on customer Age.

Middle-aged customers age between 40 and 50 show higher churn compared to younger customers. And Young-Aged Customers churn rate is lower than retention rate. Few senior aged customers also churned.

Age and Exited Features used.

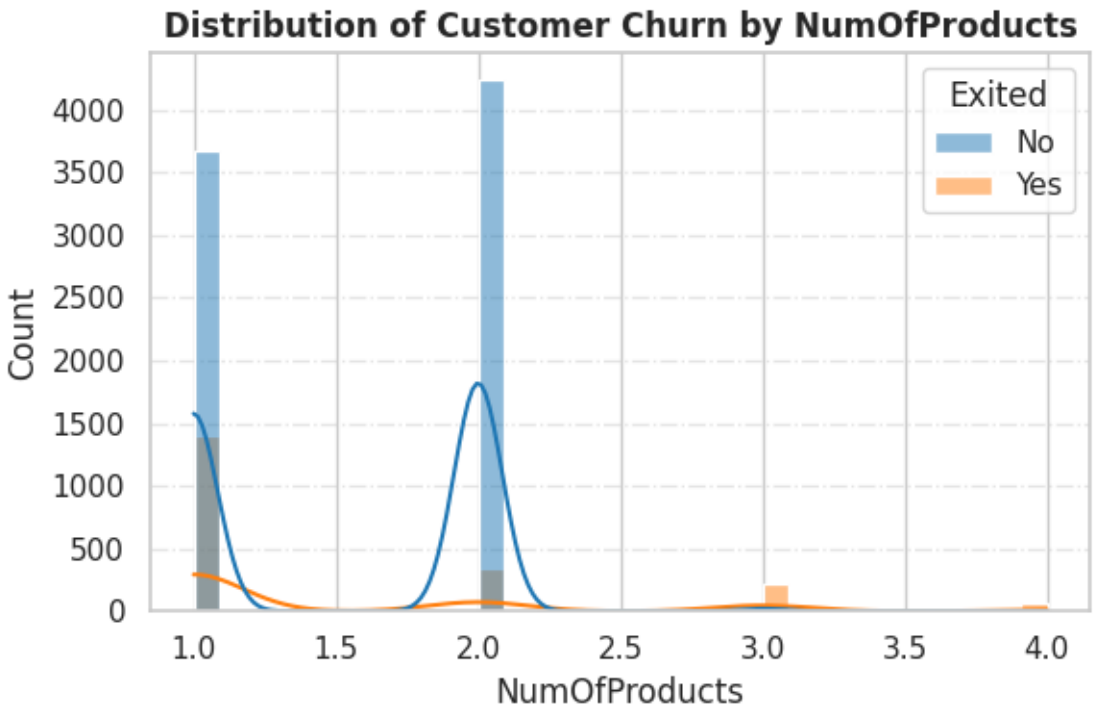
overall, Age-based financial products and offers can help improve retention. Bank can Reduce churn ratio by Offering Special Services to senior aged customers.

Distribution of Customer Churn by NumOfProducts

Python code:

```
continous_plot('NumOfProducts', dataset, 'tab10')
```

Output:



Interpretation for Distribution of churn by NumOfProducts chart:

The chart shows the distribution of customer churn based on the number of products held.

Customers with 1 or 2 products form the largest group, and most of them have not exited. However, churn is noticeably higher among customers with 3 and 4 products, even though their overall count is low.

Customers holding 3 or more products show a higher tendency to churn, indicating possible dissatisfaction with bundled services. The bank should analyze product complexity and improve service quality for multi-product customers to reduce churn. And also bank can offer multiple products to the customers those who has 1 and 2 products to reduce churn.

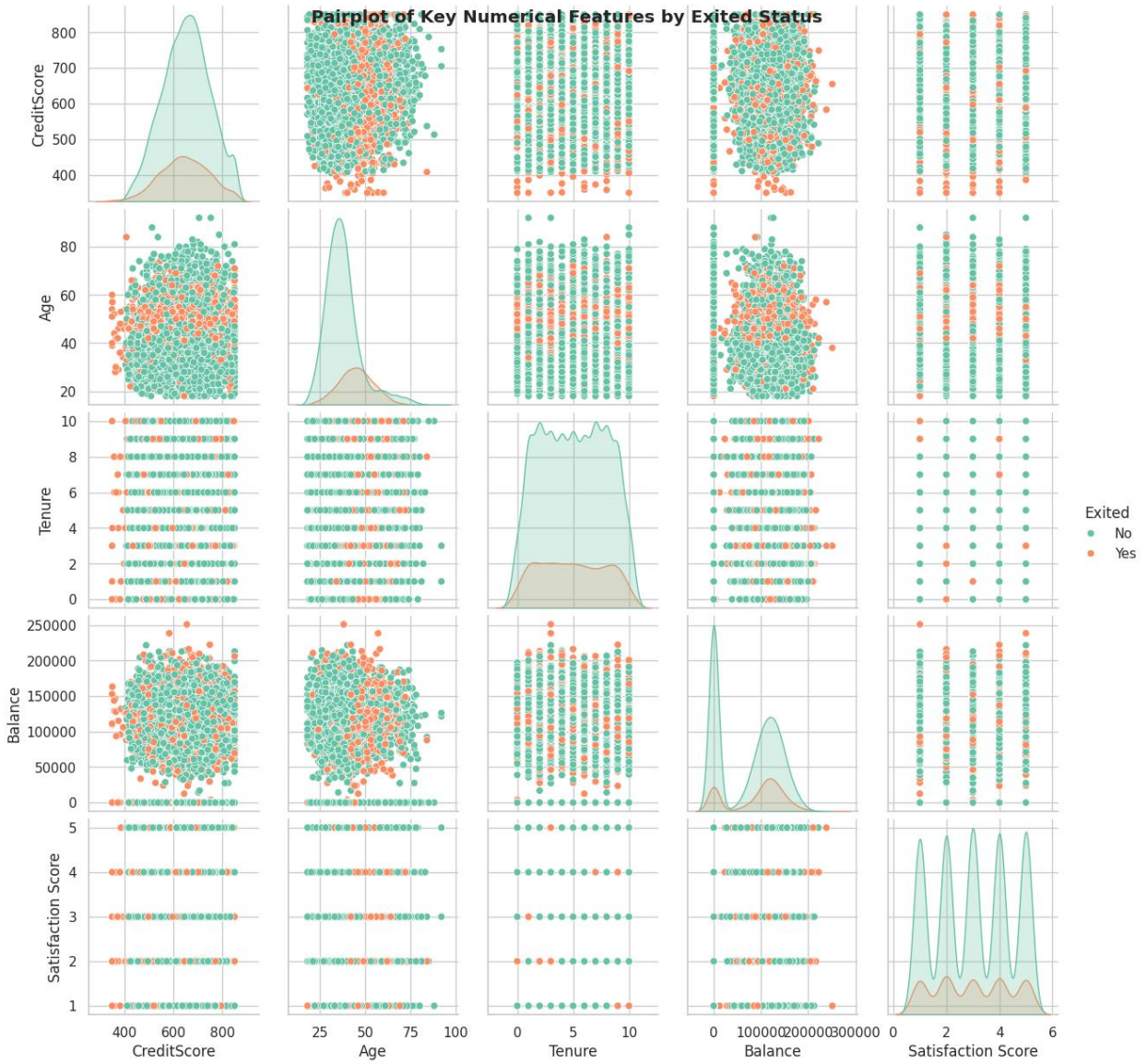
Multivariate Analysis: Relationships among Numerical Features with Respect to Customer Exit.

Python code:

```
ncol=['CreditScore','Age','Tenure','Balance','Satisfaction Score']


sns.set(style="whitegrid")
sns.pairplot(
        data=dataset,
        vars=dataset[ncol],
        hue='Exited',
        diag_kind='kde',
        palette='Set2',
        kind='scatter'
)
plt.suptitle("Pairplot of Key Numerical Features by Exited Status", fontweight='bold')
plt.show()
```

Output:



Interpretation of Pair plot Chart:

This pair plot displays the relationship among multiple numerical features such as CreditScore, Age, Tenure, Balance and Satisfaction Score and how they differ between customers who Exited and those who Stayed. This chart shows how two variables are related to each other.

What is it saying?

Every plot compares two numerical features and coloured them based on customer exited status. Each KDE curves shows the distribution of each numerical feature. The red and blue point indicates whether Exited and Stayed.

What features you used?

For this Analysis Numerical Features were used such as CreditScore, Age, Tenure, Balance and Satisfaction Score. And Feature Exited is used for variable hue.

From this what you are showing us?

This shows how each numerical relates to customer churn, Whether the customer who exited is older, having higher income or lower balance and different tenure. It helps to identify which features might be useful for churn prediction.

Pivot: Loan Eligibility by Customer Segment

Python code:

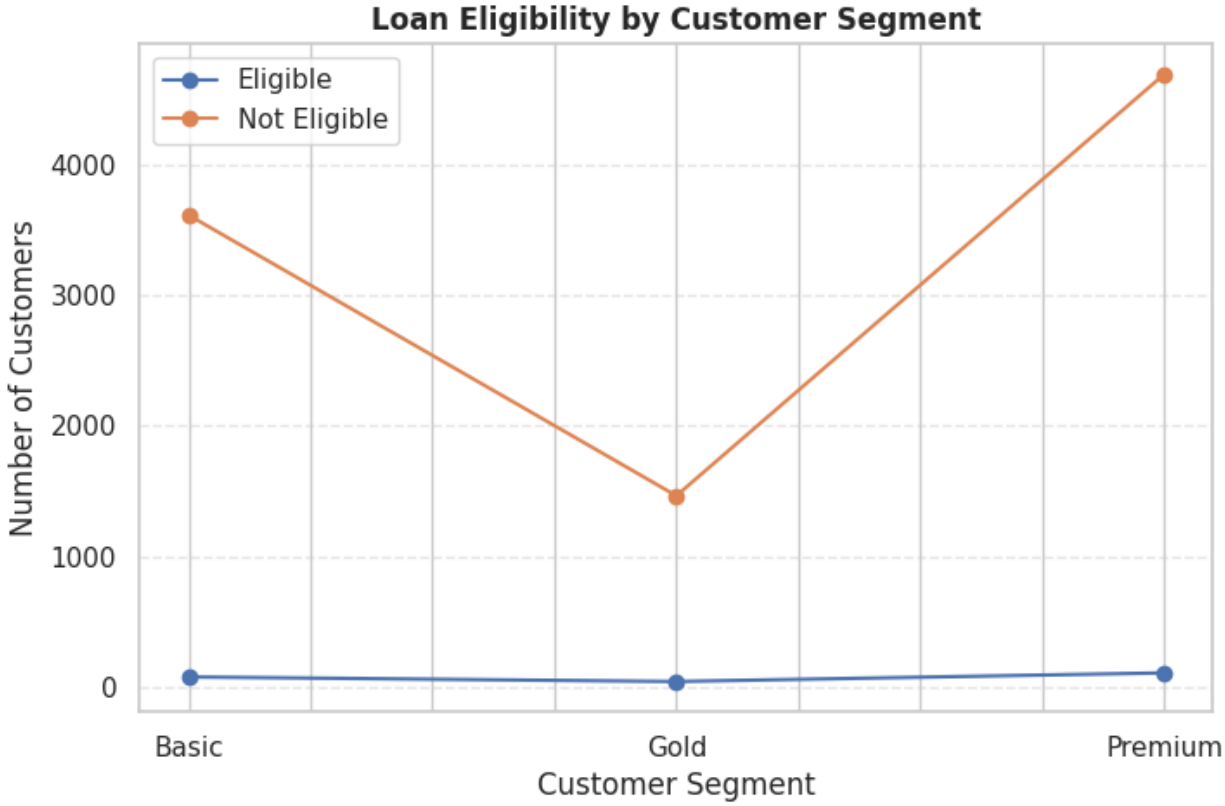
```
pivot_segment = dataset.pivot_table(
   index='CustomerSegment',
   columns='LoanEligibility',
   values='EstimatedSalary',
   aggfunc='count'
)
display(pivot_segment)


plt.figure(figsize=(4,1))
pivot_segment.plot(kind='line', marker='o', figsize=(8,5))
plt.title("Loan Eligibility by Customer Segment", fontweight='bold')
plt.ylabel("Number of Customers")
plt.xlabel("Customer Segment")
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.legend()
plt.show()
```

Output:

LoanEligibility	Eligible	Not Eligible
CustomerSegment		
Basic	79	3612
Gold	44	1465
Premium	110	4689

<Figure size 400x100 with 0 Axes>



Interpretation of your above chart

It shows the number of customers who are Eligible vs Not Eligible for a loan, broken down by Customer Segment.

Across all segments, the Not Eligible customers are far higher than Eligible customers, indicating stricter loan eligibility criteria. Premium customers have the highest number of both Eligible (110) and Not Eligible (4689), reflecting that this segment has more customers overall. Basic and Gold segments have fewer Eligible customers (79 and 44, respectively), and a large number of Not Eligible customers (3612 and 1465).

CustomerSegment, LoanEligibility and EstimatedSalary used for this analysis.

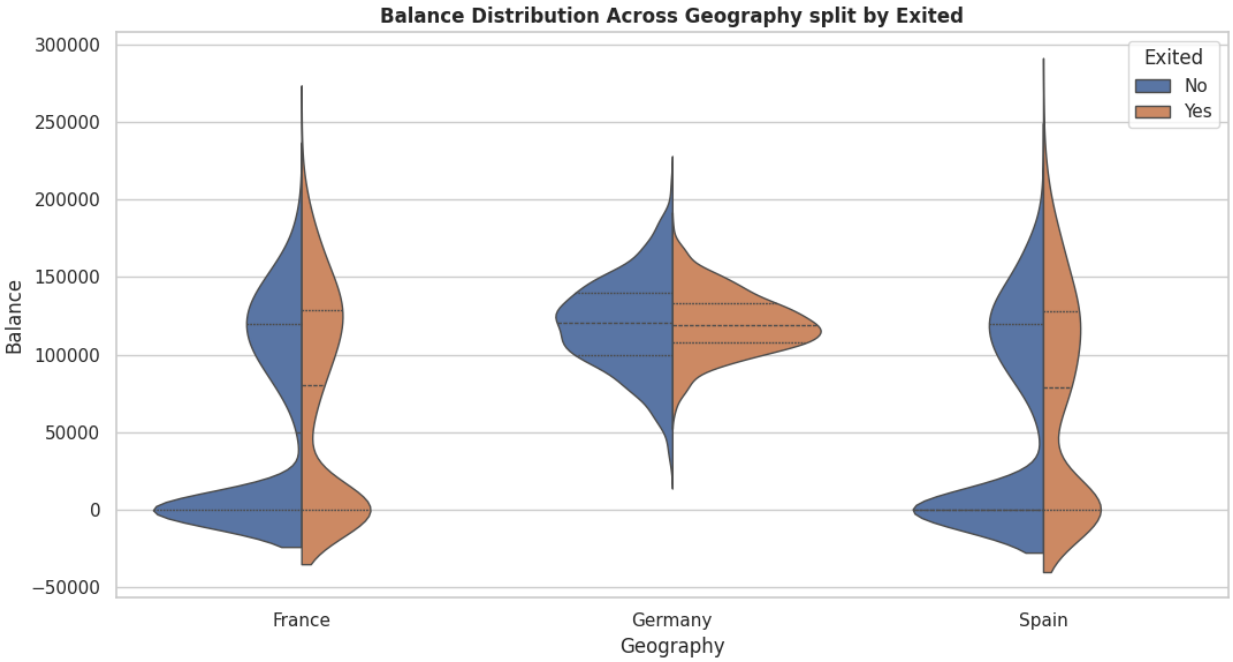
Overall, the chart shows the distribution of loan eligibility across different customer segments. It highlights that eligibility is not just dependent on segment, because even Premium customers have a high number of Not Eligible cases.

Multivariate Analysis: Distribution of Balance by Geography split by Exited

Python code:

```
plt.figure(figsize=(11,6))
sns.violinplot(data=dataset,
               x='Geography',
               y='Balance',
               hue='Exited',
               split=True,
               inner='quart',
               linewidth=1
               )
plt.title("Balance Distribution Across Geography split by Exited", fontweight='bold')
plt.xlabel("Geography")
plt.ylabel('Balance')
plt.legend(title='Exited')
plt.tight_layout()
plt.show()
```

Output:



Interpretation of violin plot chart

This is a violin plot that shows how the Balance amounts of customers are distributed across different Geography categories, and how this distribution differs between customers who Exited the bank and those who did not.

What is it saying?

Each Geography such as France, Spain, Germany has two distributions, Exited is Yes means customers who left the bank and Exited is No means customers who stayed. The shape and spread of the Balance values tell us, Where most customers balances lies and how balanced customers differ between exited and non-exited groups.

What Features You Used?

This plot used three Features from the dataset such as Geography, Balance and Exited used as hue to split the violins. So, this chart combines one numerical variable and two categorical variables.

From this what you are showing us?

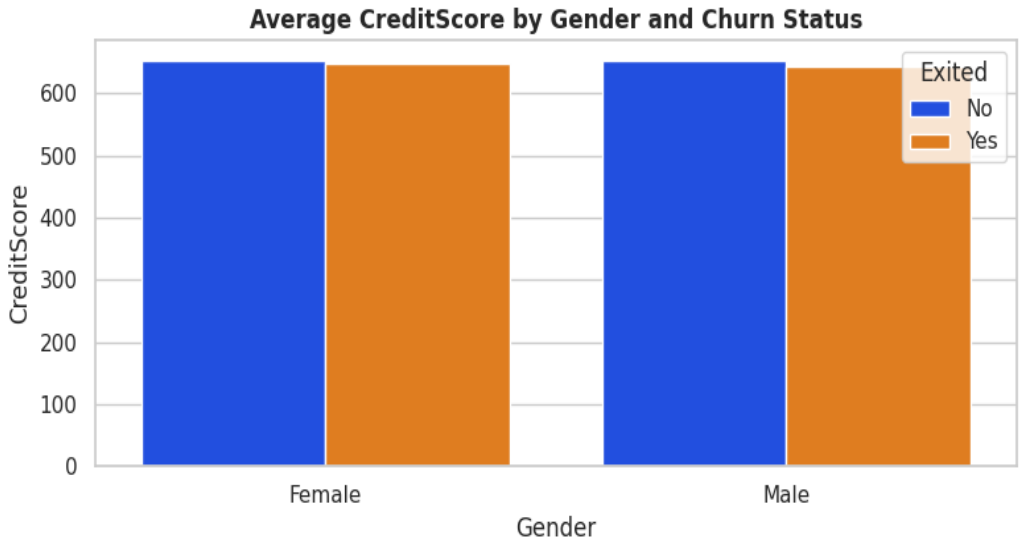
This Plot clearly shows How Account Balance varies from one Geography to another and Whether Exited customers have different Balance levels compared to non-exited customers.

Multivariate Analysis: Average CreditScore by Gender split by Exited:

Python code:

```
plt.figure(figsize=(8,4))
sns.barplot(data=dataset,
        x='Gender',
        y='CreditScore',
        hue='Exited',
        estimator='mean',
        palette='bright',
        errorbar=None
        )
plt.title("Average CreditScore by Gender and Churn Status",fontweight='bold')
plt.xlabel("Gender")
plt.ylabel('CreditScore')
plt.legend(title='Exited')
plt.tight_layout()
plt.show()
```

Output:



Interpretation of Bar plot Chart:

This is a bar plot that compares the average CreditScore of customers based on their Gender and whether they have Exited or Not Exited the bank. Each gender has two bars, one bar for customers who Exited and One bar for customers who did not exit. The height of each bar represents the average CreditScore of that group.

What is it saying?

The chart shows Whether males or females have higher or lower CreditScores on average, Whether customers who churned have different CreditScores from those who stayed and How the combination of Gender, Exited status influences CreditScore.The bar plot helps identify any relationship between creditworthiness and churn.

What Features You Used?

This bar plot used three Features from the dataset such as Gender,CreditScore and Exited. So this chart combines one numerical variable and two categorical variables.

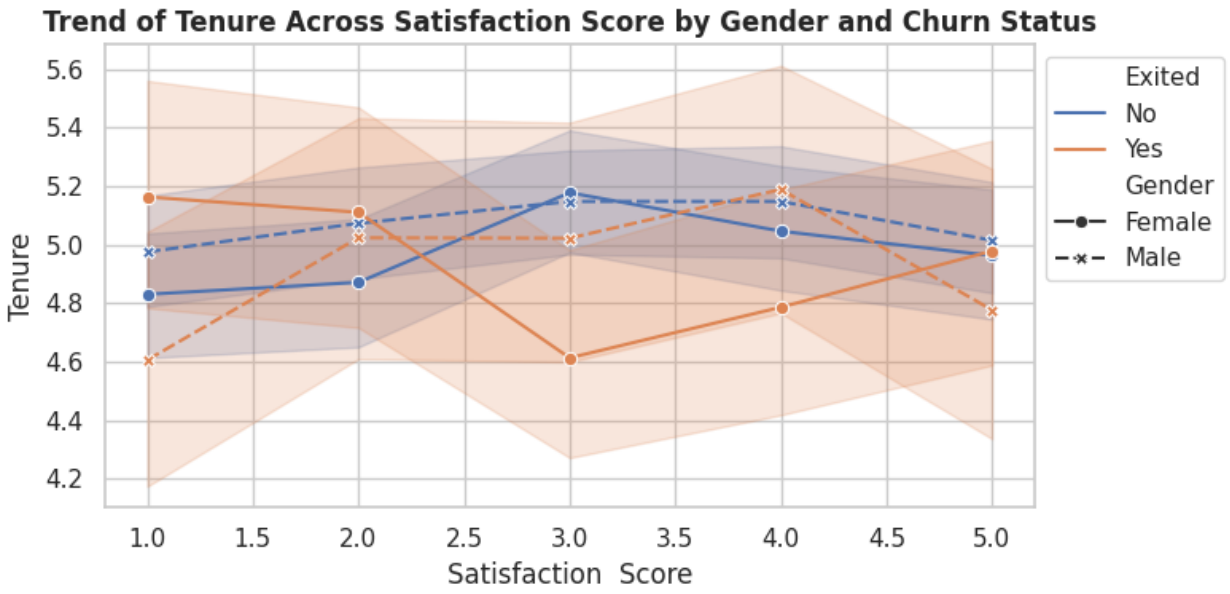
From this what you are showing us?

This Chart shows How average CreditScore differs between genders, How churned vs non-churned customers differ in terms of CreditScore. This helps in understanding whether CreditScore influences churn behavior and whether it varies across gender groups.

Multivariate Analysis: Trend of Tenure Across Satisfaction Score by Gender and Churn Status

Python code:

```
plt.figure(figsize=(8,4))
sns.lineplot(x='Satisfaction Score',
             y='Tenure',
             hue='Exited',
             style='Gender',
             markers=True,
             estimator='mean',
             data=dataset
             )
plt.title("Trend of Tenure Across Satisfaction Score by Gender and Churn Status", fontweight='bold')
plt.xlabel("Satisfaction  Score")
plt.ylabel("Tenure")
plt.legend(bbox_to_anchor=(1,1))
plt.tight_layout()
plt.show()
```



Interpretation for Line plot chart:

This is a multivariate line plot showing how the average Tenure of customers changes across different Satisfaction Score levels. The chart is further broken down by:

Exited (Yes/No) whether the customer left the bank

Gender (Male/Female) shown using different line styles and markers. So this chart helps us understand how tenure varies with satisfaction, and whether this pattern differs by gender and churn status.

What is it saying?

The Chart tells, for both genders and for both churn groups, Tenure is mostly between 4.6 and 5.2 years. The blue lines (Exited = No) tend to stay a bit higher than the orange lines (Exited = Yes). This suggests customers who remain with the bank usually have been associated longer. Male and Female lines follow similar patterns. The orange lines have more ups and downs, indicating greater variation in tenure among customers who left.

What Features You Used?

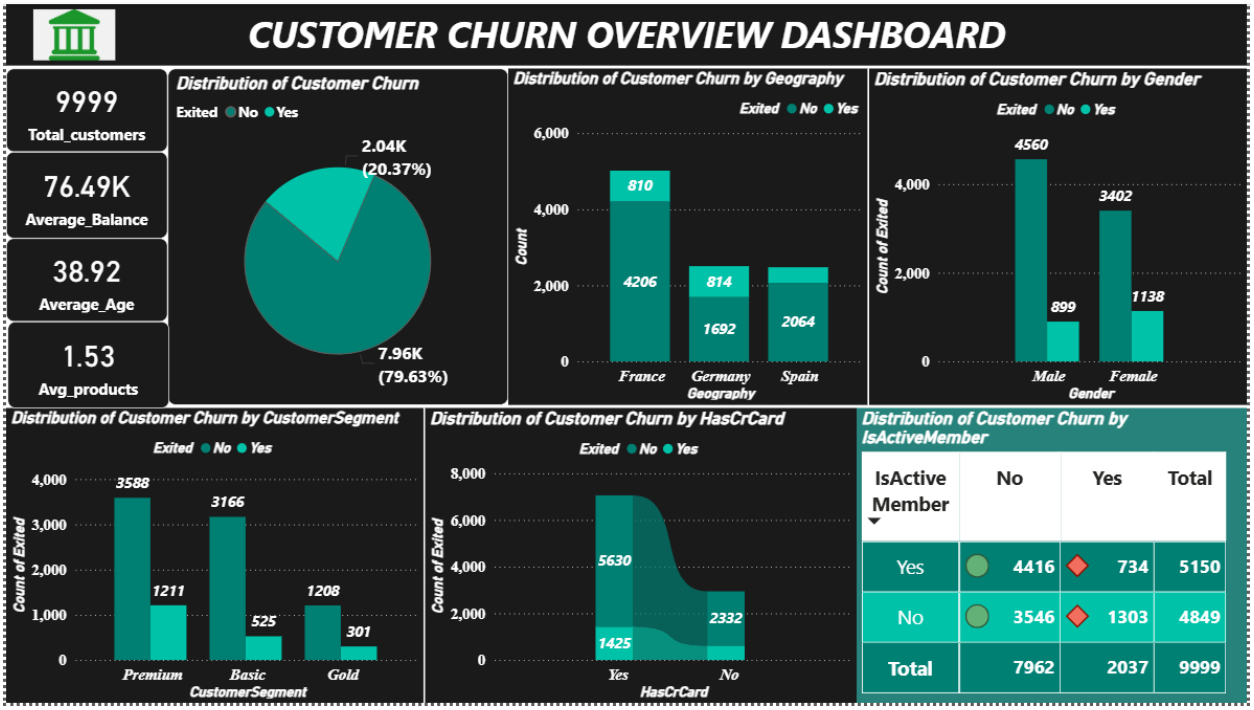
Satisfaction Score, Tenure, Exited and Gender Features are used for this analysis.

From this what you are showing us?

Overall, the plot shows tenure is not strongly influenced by satisfaction score, but churned customers show slightly lower and more unstable tenure patterns.

DASHBOARD

CUSTOMER CHURN OVERVIEW DASHBOARD:

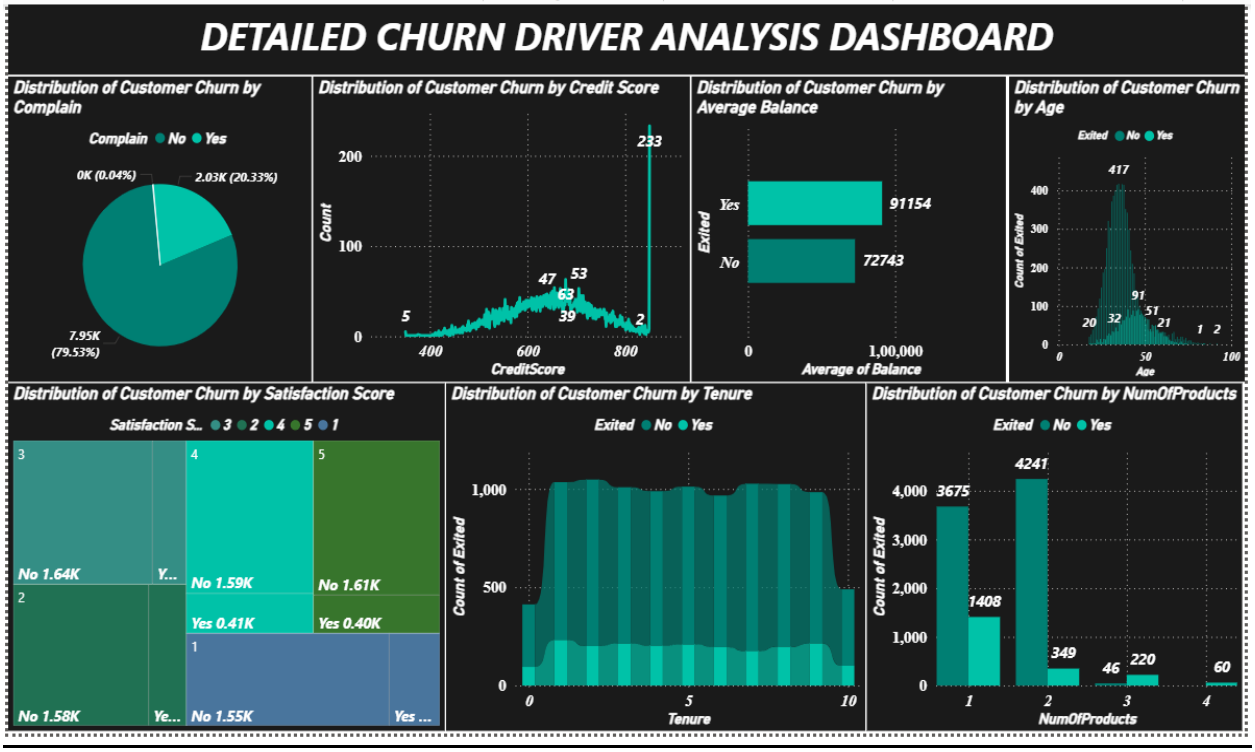


Insights Generated:

1. **Overall churn is around 20%**, meaning 1 out of every 5 customers is leaving, a moderate churn level that needs attention.
2. **Geography strongly influences churn.** Germany shows significantly higher churn compared to France and Spain, indicating region-specific dissatisfaction or competition.
3. **Gender differences are visible.** Female customers churn less than male customers, suggesting male customers may require targeted engagement or product alignment.
4. **Customer Segment impact is clear.**
   Premium and Basic segments show higher churn volume, whereas the Gold-segment retains customers better. This indicates retention strategies should focus on Basic & Premium customers.
5. **Inactive customers churn more.** Customers who are not active members show much higher churn, proving that increasing engagement activities (offers, personalized communication, own app usage) can reduce churn.

6. **Having Credit card and churn are highly connected.** Customers with credit card churn less than without cards. Credit card holders seem more committed, suggesting that cross-selling financial products boosts customer stickiness.

DETAILED CHURN DRIVER ANALYSIS DASHBOARD:



Insights Generated:

1. **Customers who raised complaints have a significantly higher churn rate.** The pie chart shows that customers with complaints churn far more than those without.
   This means unresolved issues and service dissatisfaction are major churn triggers.

2. **Customers with lower credit score churn more frequently.** As credit score increases, churn drops sharply. This implies high-risk customers financially unstable are more likely to leave the bank.

3. **Higher account balance is associated with higher churn.** Exited customers hold higher balances on average compared to non-exited customers. This shows that high-value customers are dissatisfied, and may be shifting to banks with better premium offerings.

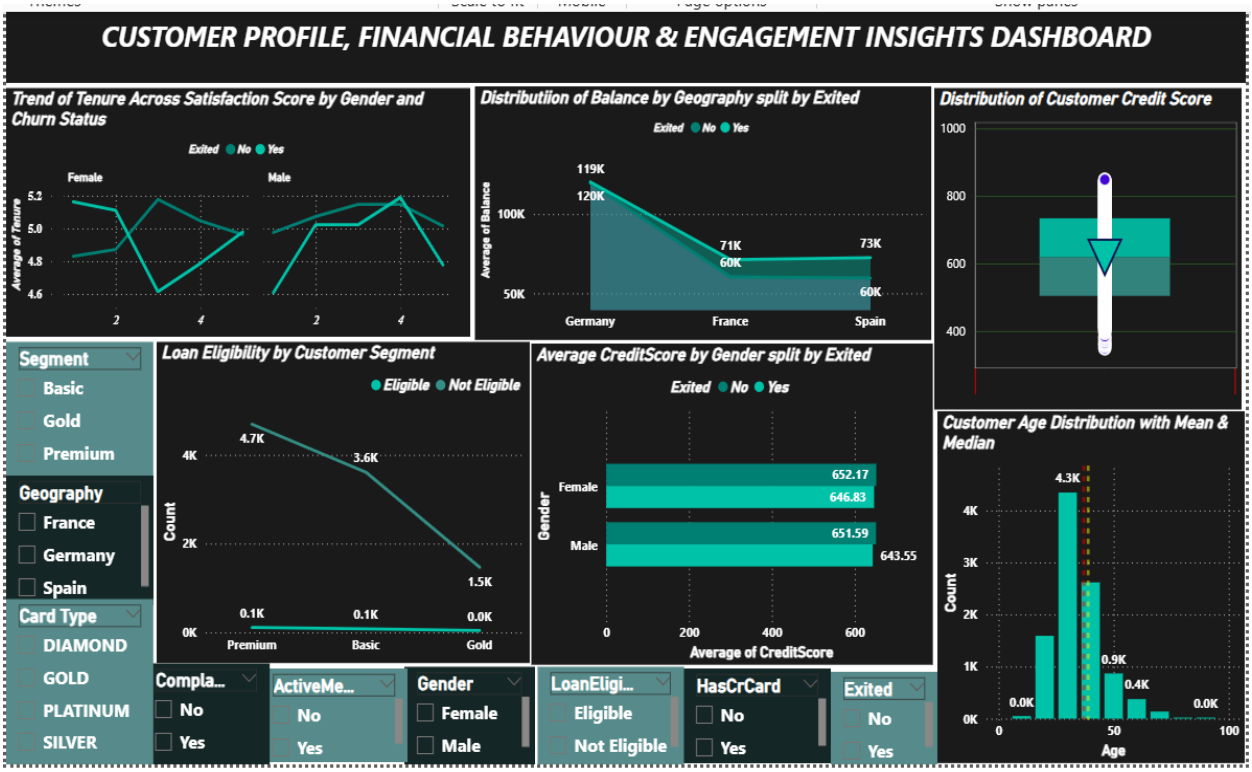
4. **Churn is most common in the mid-age group (35–45 years).** The age chart shows a very sharp churn spike around 40 years. This group likely expects better financial products, benefits, and stability, making them sensitive to service gaps.

5. **Low satisfaction scores directly drive churn.** As satisfaction score decreases, churn increases. Customers rating 1–3 contribute heavily to churn, showing that experience quality is a critical churn factor.

6. **Churn is higher among customers with very short or very long tenure**. Churn is high for new customers 0-2 years and very long tenure customers 8-10 years. This indicates early churn due to onboarding issues, and late-stage churn due to better competitor options or Service issues.

7. **Customers with only 1–2 products churn the most.** The churn rate drops significantly for customers having 3 or more products. This highlights the importance of cross-selling more products create stronger customer stickiness.

CUSTOMER PROFILE, FINANCIAL BEHAVIOUR & ENGAGEMENT INSIGHTS DASHBOARD:



Insights Generated:

1. **Tenure trends vary by satisfaction and gender.** Customers with low satisfaction score 1–2 show lower average tenure, meaning they leave early. Male customers show slightly more fluctuation across satisfaction levels. This highlights that experience quality directly impacts retention duration.

2. **Germany holds the highest balances, but churn is also higher in this group.** Even though customers in Germany maintain the highest average balances, a noticeable portion is exiting. This indicates that high-value customers in Germany may feel underserved or find better options elsewhere.

3. **Credit score distribution shows a wide spread, with churners slightly below non-churners.** The box plot shows that exited customers generally fall in the mid–low credit score range, implying that customers with weaker financial profiles are more prone to churn.

4. **Customer segments differ strongly in loan eligibility.** Premium segment customers have the highest loan eligibility, while Gold and Basic show a drop. This indicates that financial strength and engagement increase eligibility, which can be used for targeted cross-selling.

5. **Average credit score differs minimally between genders.** Both male and female customers show similar average credit scores, whether they exited or not.
   This suggests that gender is not a meaningful predictor of credit behaviour or churn.

6. **Customer age peaks heavily in the 30–45 range.** Most customers fall in this age bracket. This age group is more financially active and expects better product value, making them sensitive to dissatisfaction.

Future Enhancements:

1. Incorporate more recent customer records, additional branches, or multi-year data to capture evolving churn patterns and improve the reliability of insights.
2. Set up automated data pipelines and real-time Power BI dashboards so stakeholders can monitor churn trends, customer complaints, and satisfaction changes instantly.
3. Develop machine learning models (Logistic Regression, Random Forest, XGBoost) to identify high-risk customers and predict churn probability with high accuracy.
4. Combine additional factors like transaction patterns, digital engagement, complaint resolution time, and product usage frequency to get richer behavioural insights.

5. Since the Age column was skewed, I applied a transformation to reduce skewness. This transformed Age feature can be used in future machine-learning models to improve prediction accuracy and reduce the effect of outliers.

Conclusion:

The Customer Churn Analysis project provided a clear understanding of the key demographic, financial, and behavioural factors influencing customer attrition in the bank. Through comprehensive Python-based exploration and Power BI visualizations, the study revealed that churn is strongly associated with low satisfaction scores, higher complaint rates, lower tenure, mid-to-low credit scores, and moderate engagement with bank products.

The dashboards provide a clear and interactive view of churn patterns, making it easier to understand which customer groups are more likely to leave the bank and helping business teams identify high-risk customers and prioritize retention actions.

This project forms a good foundation for future improvements such as adding new features, enhancing design, and exploring simple predictive models. Overall, it provides solid insights and supports better decision-making for customer retention.