ALLIANCE UNIVERSITY

Private University established in Karnataka State by Act No.34 of year 2010
Recognized by the University Grants Commission (UGC), New Delhi

PROJRCT REPORT

BACHELOR OF COMPUTER APPLICATION

SEMESTER – II

INTRODUCATION TO DATASCIENCE

Customer Churn Predicition in Banking Sector Using Data Science Techniques.

BY

PRAGATHI.BR

2411021240037

DEPARTMENT OF COMPUTER APPLICATION

ALLIANCE UNIVERSITY

CHANDPURA ANEKAL MAIN, ANEKAL

BANGALORE-562106

APRIL – 2025

📊 Customer Churn Predicition in Banking Sector Using Data Science Techniques.

NAME: PRAGATHI.BR

Registration no.: 2411021240037

GITHUB REPOSITORY LINK: https://github.com/Pragathi2006/Customer-Churn-Predicition-in-Banking-Sector-Using-Data-Science-Techniques.

📚 Table of Contents

Welcome to the project notebook! Below is the structured flow of analysis covered in this project.

📘 Project Title and Info

📊 **Customer Churn Predicition in Banking Sector Using Data Science Techniques**.

📜 **Course:** Introduction to Data Science

Semester: 2nd Semester – BCA
Project Type: Beginner-level Data Science & Machine Learning
Tools Used: Python, Pandas, Seaborn, Scikit-learn
Dataset: Bank Churners Dataset

🔰 Project Overview

This project utilizes the bank churners dataset to explore and model the behavior of bank customers to predict potential churn using statistical techniques. Customer Churn poses a significant challenge in the banking sector. this project aims to analyze customer behavior, identify churn patterns and build predictive models that helps in early identification of customers likely to leave the bank.

We focus on two major tasks:

- clearly define what churn means in the banking context.
- Understand why predicting churn matters- revenue impact, and retention cost vs acquisition cost.

We follow the **CRISP_DM process** to complete this project step-by-step and use various methods like exploratory data analysis (EDA), preprocessing, machine learning, and evaluation techniques aligned with our course syllabus.

🎯 Project Goal

To develop a predictive model using statistical methods and machine learning to classify whether a customer will churn or not, thereby enable proactive customer retention strategies.

- the goal of this project is to analyze customer behavior and build a predictive model to identify bank customers who are likely to churn.
- By analyzing customer demographics, account activity, and transactional behavior, the project aims to uncover the key factors driving customer attrition.
- The insights gained will help the bank implement targeted retention strategies, reduce churn rates, and improve customer satisfaction and probability.

❗ Challenges Faced

- Imbalanced dataset (more non-churners than churners)
- Feature selection and engineering
- Ensuring interpretability of models for business understanding
- Dealing with missing or irrevelent data

1.Bussiness Understanding

Goal: Predict Customer Churn from the Bank Churners

📦 Import Libraries & Load Dataset

In this section, we load the required libraries and import our dataset to begin analysis.

```
# 📌 1. Import Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error, r2_score,
accuracy_score, precision_score, recall_score, f1_score, confusion_matrix,
classification_report
from scipy import stats

# 📌 2. Load the Dataset
df = pd.read_csv(r"C:\Users\Pragathi BR\Downloads\BankChurners.csv")
df
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book	...
0	768805383	Existing Customer	45	M	3	High School	Married	$60K - 80K	Blue	39	...
1	818770008	Existing Customer	49	F	5	Graduate	Single	Less than $40K	Blue	44	...
2	713982108	Existing Customer	51	M	3	Graduate	Married	$80K - 120K	Blue	36	...
3	769911858	Existing Customer	40	F	4	High School	Unknown	Less than $40K	Blue	34	...
4	709106358	Existing Customer	40	M	3	Uneducated	Married	$60K - 80K	Blue	21	...
...	...	...	...	...	...	...	...	...	...	...	...
10122	772366833	Existing Customer	50	M	2	Graduate	Single	$40K - 60K	Blue	40	...
10123	710638233	Attrited Customer	41	M	2	Unknown	Divorced	$40K - 60K	Blue	25	...
10124	716506083	Attrited Customer	44	F	1	High School	Married	Less than $40K	Blue	36	...
10125	717406983	Attrited Customer	30	M	2	Graduate	Unknown	$40K - 60K	Blue	36	...
10126	714337233	Attrited Customer	43	F	2	Graduate	Married	Less than $40K	Silver	25	...

10127 rows × 23 columns

🧹 Data Preprocessing

Tasked Performed:

- **Clean the data:** Remove irrelevant columns (e.g., client IDs)
- **Encode Categorical Variables** for modeling.
- **Handle Class Imbalance** (e.g., more non churners than churners)-possibly use SMOTE or class weights.

```
# first columns of the dataset
df.head()
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book	...
0	768805383	Existing Customer	45	M	3	High School	Married	$60K - 80K	Blue	39	...
1	818770008	Existing Customer	49	F	5	Graduate	Single	Less than $40K	Blue	44	...
2	713982108	Existing Customer	51	M	3	Graduate	Married	$80K - 120K	Blue	36	...
3	769911858	Existing Customer	40	F	4	High School	Unknown	Less than $40K	Blue	34	...
4	709106358	Existing Customer	40	M	3	Uneducated	Married	$60K - 80K	Blue	21	...

5 rows × 23 columns

```
# Statistical Summary
df.describe()
```

	CLIENTNUM	Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit
count	1.012700e+04	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000
mean	7.391776e+08	46.325960	2.346203	35.928409	3.812580	2.341167	2.455317	8631.953698
std	3.690378e+07	8.016814	1.298908	7.986416	1.554408	1.010622	1.106225	9088.776650
min	7.080821e+08	26.000000	0.000000	13.000000	1.000000	0.000000	0.000000	1438.300000
25%	7.130368e+08	41.000000	1.000000	31.000000	3.000000	2.000000	2.000000	2555.000000
50%	7.179264e+08	46.000000	2.000000	36.000000	4.000000	2.000000	2.000000	4549.000000
75%	7.731435e+08	52.000000	3.000000	40.000000	5.000000	3.000000	3.000000	11067.500000
max	8.283431e+08	73.000000	5.000000	56.000000	6.000000	6.000000	6.000000	34516.000000

```
# 📌 3. Basic Info & Data Overview
print("Shape of the dataset:", df.shape)
```

```
Shape of the dataset: (10127, 23)
```

```python
# Check Basic Info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10127 entries, 0 to 10126
Data columns (total 23 columns):
 #   Column                                                                                                                               Non-Null Count
Dtype
---  ------                                                                                                                               --------------
-----
 0   CLIENTNUM                                                                                                                            10127 non-null
int64
 1   Attrition_Flag                                                                                                                       10127 non-null
object
 2   Customer_Age                                                                                                                         10127 non-null
int64
 3   Gender                                                                                                                               10127 non-null
object
 4   Dependent_count                                                                                                                      10127 non-null
int64
 5   Education_Level                                                                                                                      10127 non-null
object
 6   Marital_Status                                                                                                                       10127 non-null
object
 7   Income_Category                                                                                                                      10127 non-null
object
 8   Card_Category                                                                                                                        10127 non-null
object
 9   Months_on_book                                                                                                                       10127 non-null
int64
 10  Total_Relationship_Count                                                                                                             10127 non-null
int64
 11  Months_Inactive_12_mon                                                                                                               10127 non-null
int64
 12  Contacts_Count_12_mon                                                                                                                10127 non-null
int64
```

```python
# 📌 4. Data Cleaning
# Check for Missing Values
df.isnull().sum()
```

```
CLIENTNUM                                                                                                                             0
Attrition_Flag                                                                                                                        0
Customer_Age                                                                                                                          0
Gender                                                                                                                                0
Dependent_count                                                                                                                       0
Education_Level                                                                                                                       0
Marital_Status                                                                                                                        0
Income_Category                                                                                                                       0
Card_Category                                                                                                                         0
Months_on_book                                                                                                                        0
Total_Relationship_Count                                                                                                              0
Months_Inactive_12_mon                                                                                                                0
Contacts_Count_12_mon                                                                                                                 0
Credit_Limit                                                                                                                          0
Total_Revolving_Bal                                                                                                                   0
Avg_Open_To_Buy                                                                                                                       0
Total_Amt_Chng_Q4_Q1                                                                                                                  0
Total_Trans_Amt                                                                                                                       0
Total_Trans_Ct                                                                                                                        0
Total_Ct_Chng_Q4_Q1                                                                                                                   0
Avg_Utilization_Ratio                                                                                                                 0
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1    0
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2    0
dtype: int64
```

```python
#Check the null value
df.isna().sum()
```

```
CLIENTNUM                                                                                                                0
Attrition_Flag                                                                                                           0
Customer_Age                                                                                                             0
Gender                                                                                                                   0
Dependent_count                                                                                                          0
Education_Level                                                                                                          0
Marital_Status                                                                                                           0
Income_Category                                                                                                          0
Card_Category                                                                                                            0
Months_on_book                                                                                                           0
Total_Relationship_Count                                                                                                 0
Months_Inactive_12_mon                                                                                                   0
Contacts_Count_12_mon                                                                                                    0
Credit_Limit                                                                                                             0
Total_Revolving_Bal                                                                                                      0
Avg_Open_To_Buy                                                                                                          0
Total_Amt_Chng_Q4_Q1                                                                                                     0
Total_Trans_Amt                                                                                                          0
Total_Trans_Ct                                                                                                           0
Total_Ct_Chng_Q4_Q1                                                                                                      0
Avg_Utilization_Ratio                                                                                                    0
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1    0
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2    0
dtype: int64
```

```
# Handle Missing Values (drop or fill)
df.fillna(df.mean(numeric_only=True), inplace=True)
```

📊 Exploratory Data Analysis (EDA)

We explored the data using:

- **Visualize chart patterns** across Demographics, product, usage, tenure, etc.
- **Identify correlations** and feature importance (through plots and heatmaps).
- Create **Linear graphs/charts** as per report format (like churn vs. credit limit, churn vs. tenure).

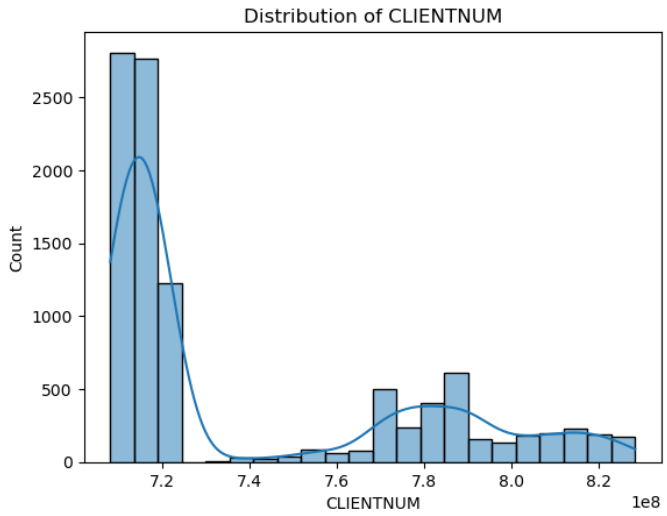
```
# 📌 5. Descriptive Statistics
df.describe()
```

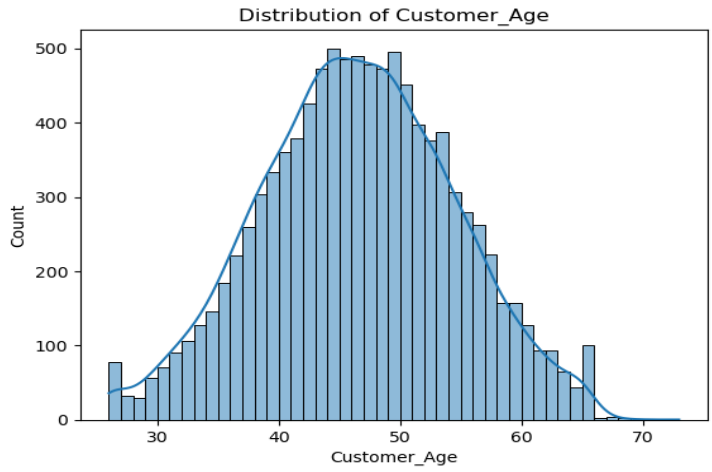
	CLIENTNUM	Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit
count	1.012700e+04	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000
mean	7.391776e+08	46.325960	2.346203	35.928409	3.812580	2.341167	2.455317	8631.953698
std	3.690378e+07	8.016814	1.298908	7.986416	1.554408	1.010622	1.106225	9088.776650
min	7.080821e+08	26.000000	0.000000	13.000000	1.000000	0.000000	0.000000	1438.300000
25%	7.130368e+08	41.000000	1.000000	31.000000	3.000000	2.000000	2.000000	2555.000000
50%	7.179264e+08	46.000000	2.000000	36.000000	4.000000	2.000000	2.000000	4549.000000
75%	7.731435e+08	52.000000	3.000000	40.000000	5.000000	3.000000	3.000000	11067.500000
max	8.283431e+08	73.000000	5.000000	56.000000	6.000000	6.000000	6.000000	34516.000000

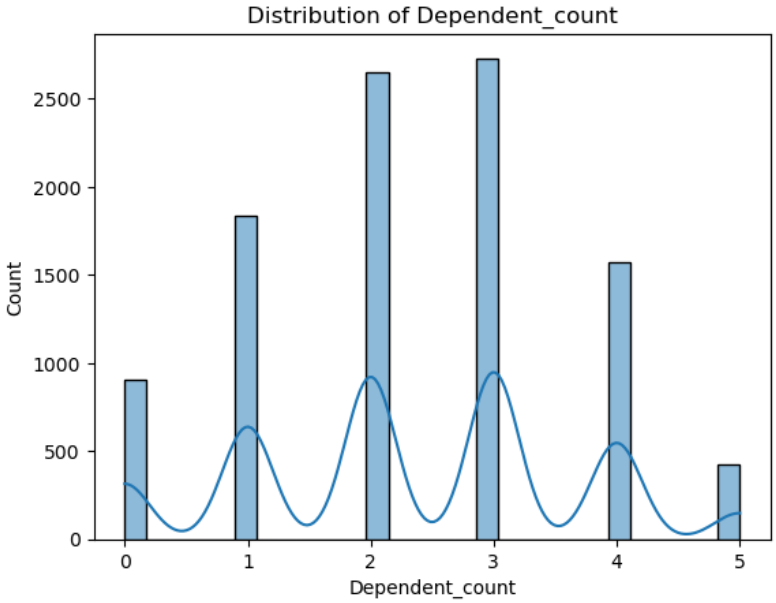
```
# columns of the dataset
df.columns
```

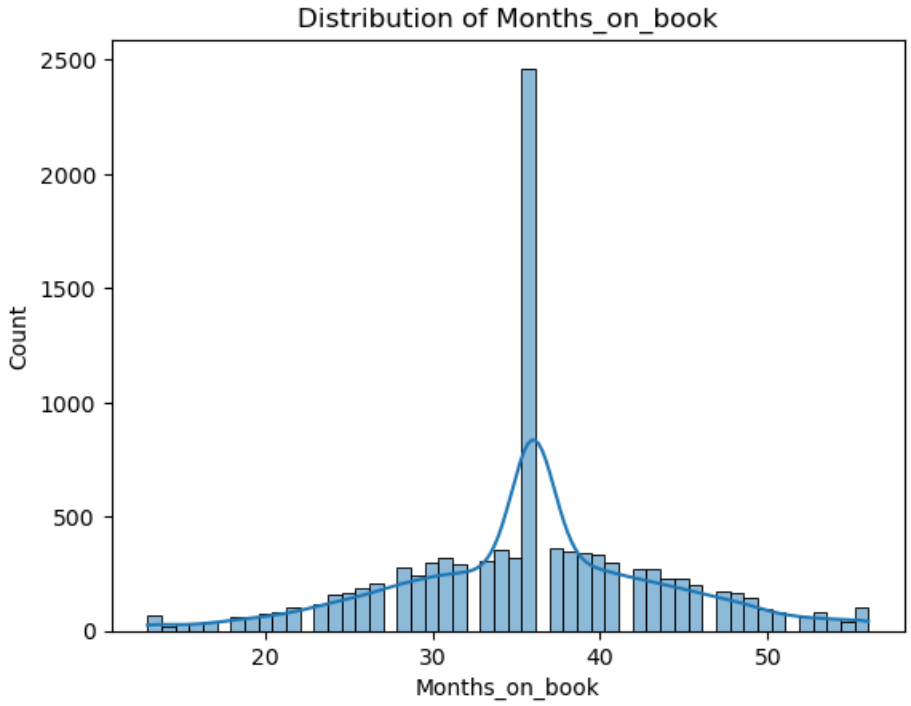
```
Index(['CLIENTNUM', 'Attrition_Flag', 'Customer_Age', 'Gender',
       'Dependent_count', 'Education_Level', 'Marital_Status',
       'Income_Category', 'Card_Category', 'Months_on_book',
       'Total_Relationship_Count', 'Months_Inactive_12_mon',
       'Contacts_Count_12_mon', 'Credit_Limit', 'Total_Revolving_Bal',
       'Avg_Open_To_Buy', 'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt',
       'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1', 'Avg_Utilization_Ratio',
       'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1',
       'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2'],
      dtype='object')
```

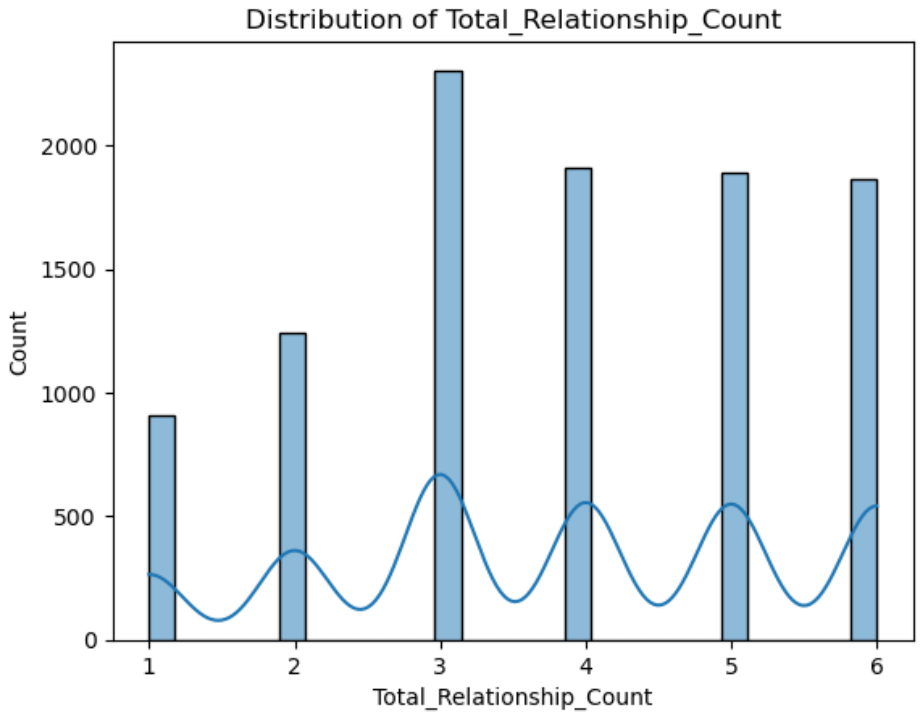
```
# 📌 6. Exploratory Data Analysis (EDA)
# Plotting distributions of numerical columns
numerical_cols = df.select_dtypes(include=np.number).columns
for col in numerical_cols:
    sns.histplot(df[col], kde=True)
    plt.title(f"Distribution of {col}")
    plt.show()
```

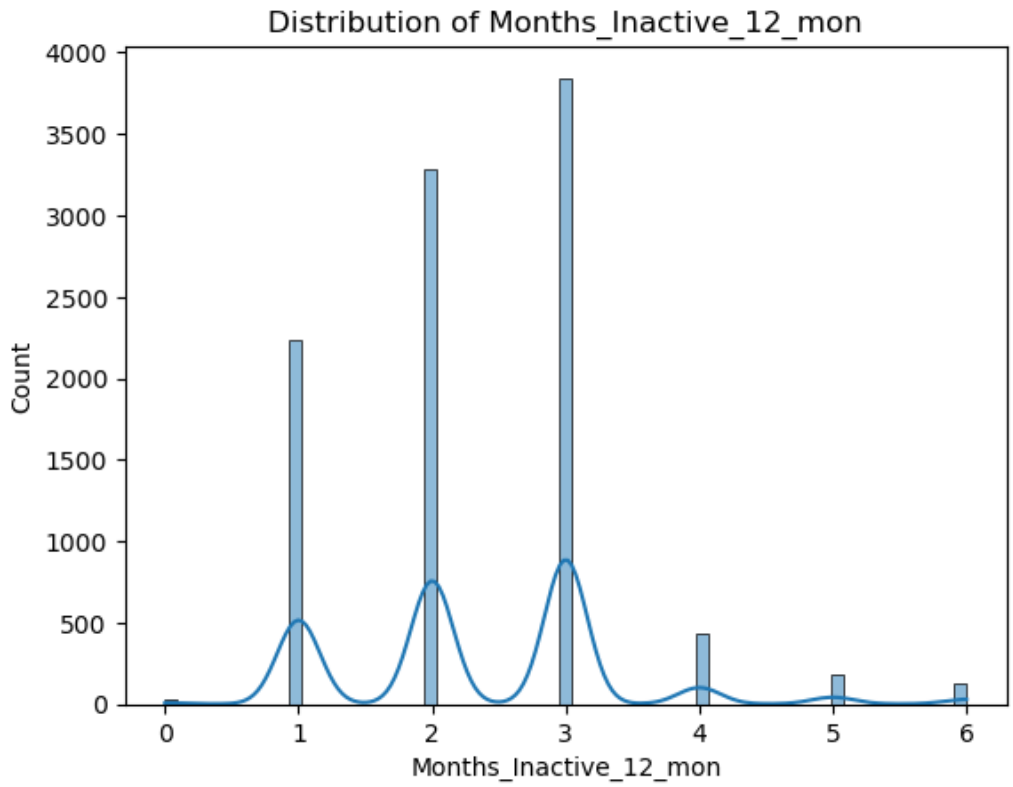


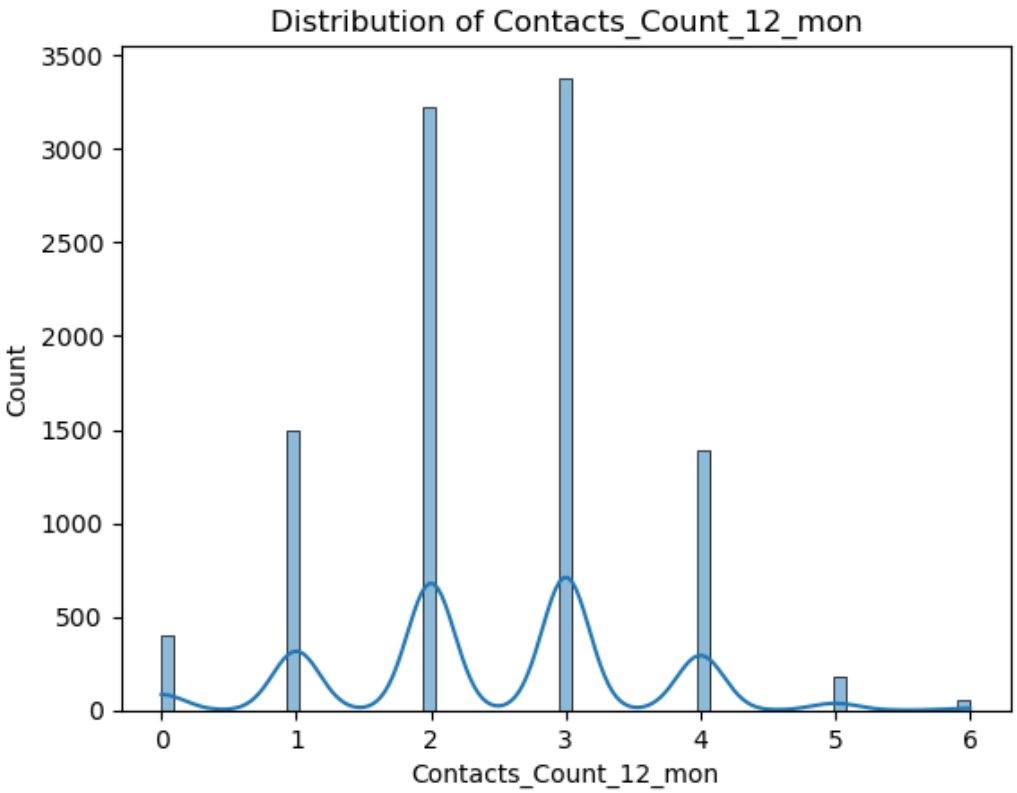


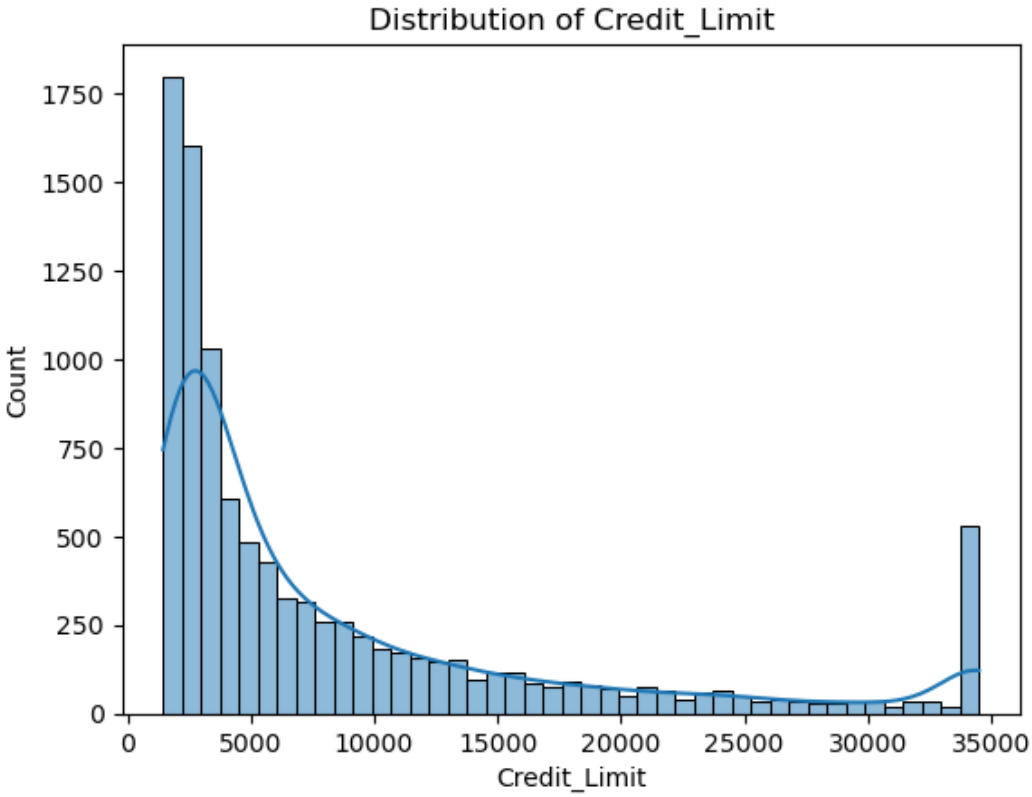

Distribution of Dependent_count
Count
Dependent_count
0
500
1000
1500
2000
2500
0
1
2
3
4
5

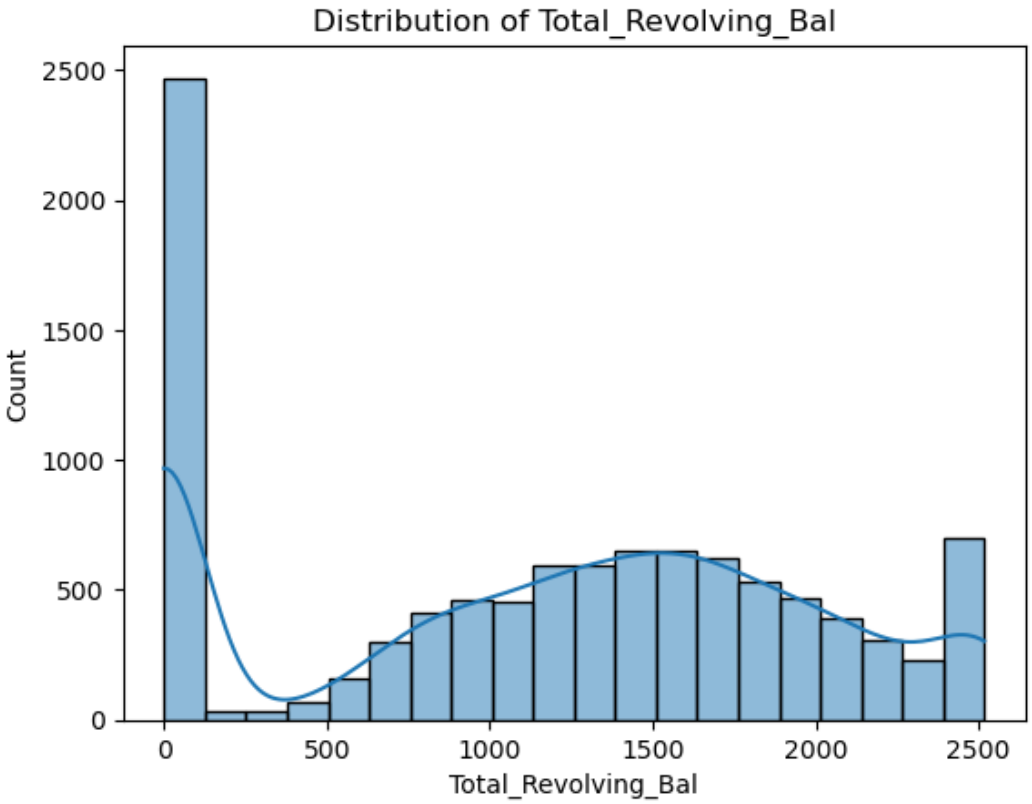

Distribution of Months_on_book
Count
Months_on_book
0
500
1000
1500
2000
2500
20
30
40
50

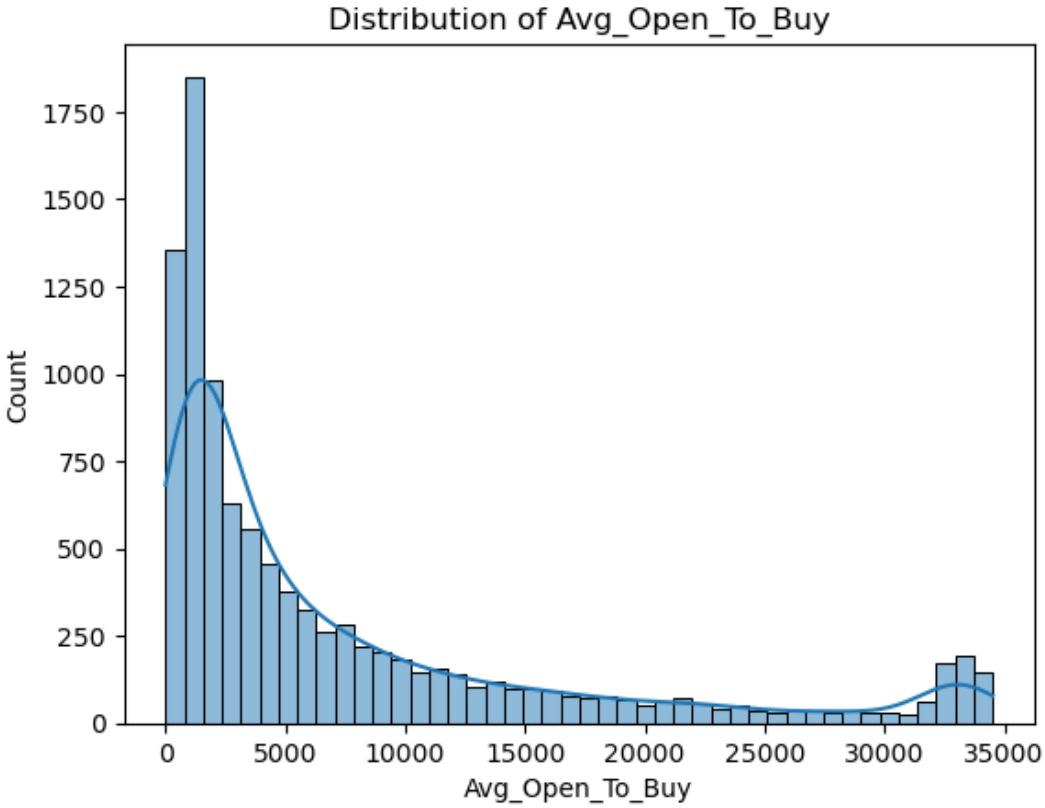

Distribution of Total_Relationship_Count
Count
0
500
1000
1500
2000
1
2
3
4
5
6
Total_Relationship_Count

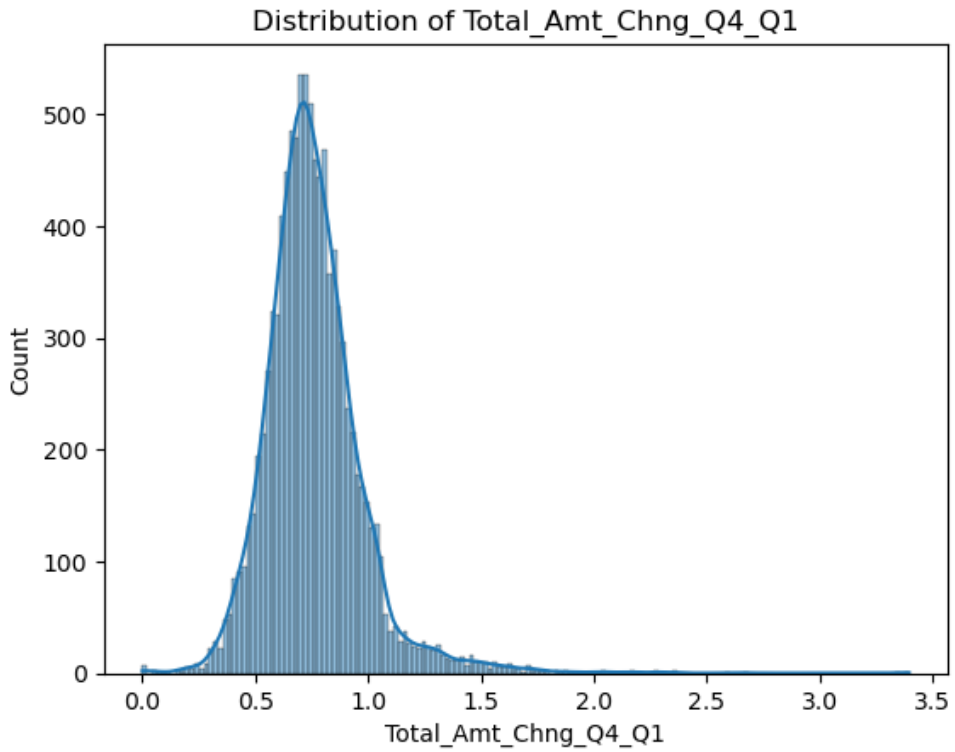

Distribution of Months_Inactive_12_mon
Count
0
500
1000
1500
2000
2500
3000
3500
4000
0
1
2
3
4
5
6
Months_Inactive_12_mon

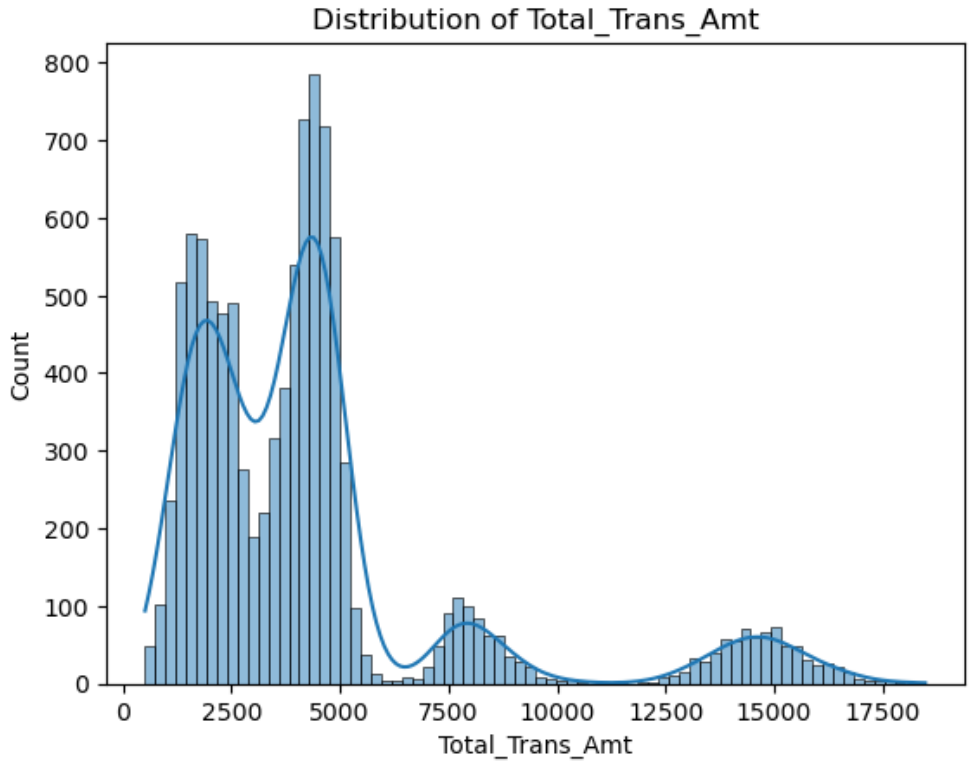

Distribution of Contacts_Count_12_mon
Count
Contacts_Count_12_mon
0
500
1000
1500
2000
2500
3000
3500
0
1
2
3
4
5
6

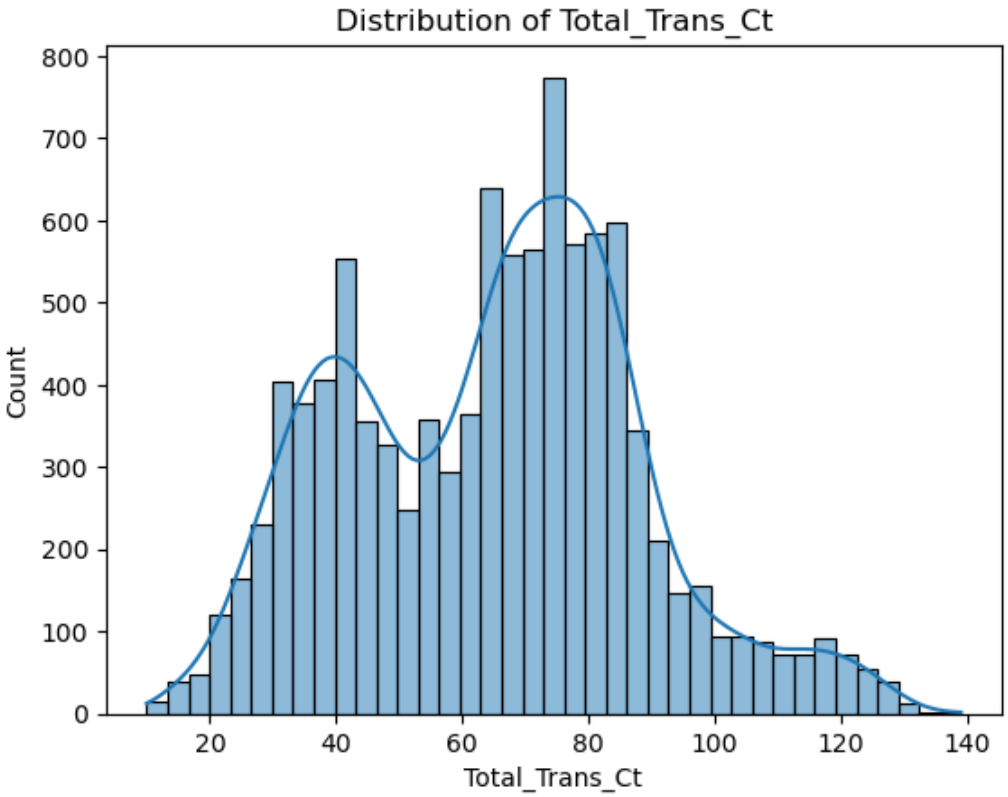

Distribution of Credit_Limit
Count
Credit_Limit
0
250
500
750
1000
1250
1500
1750
0
5000
10000
15000
20000
25000
30000
35000

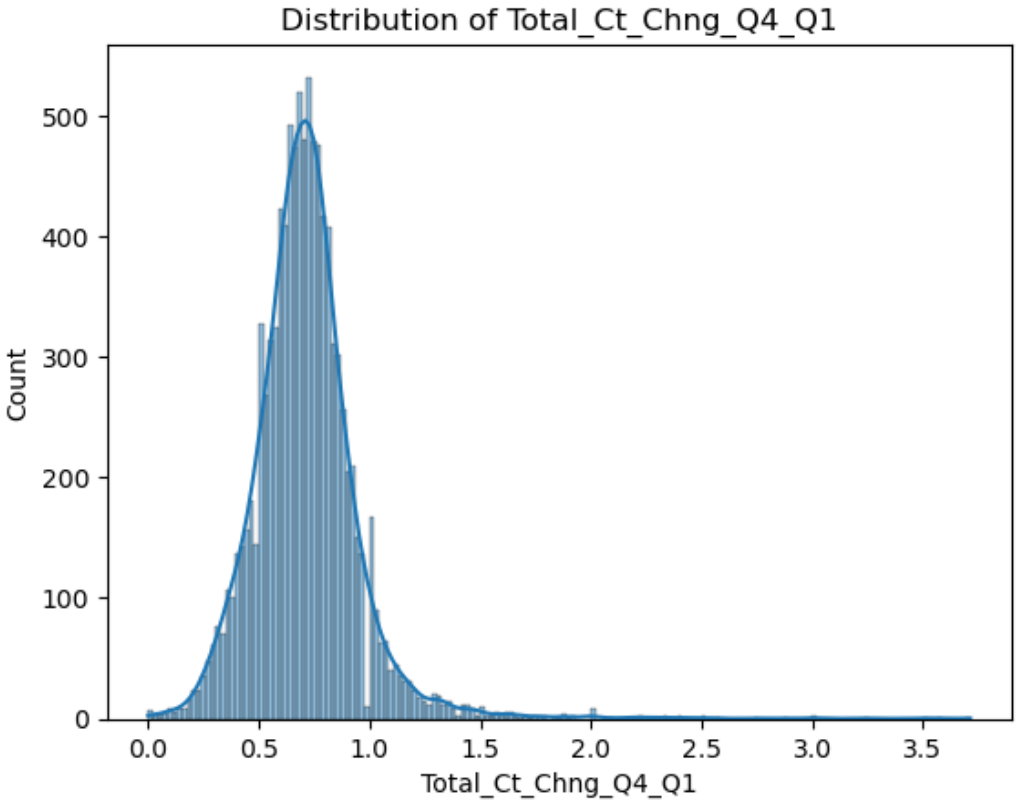

Distribution of Total_Revolving_Bal
Count
Total_Revolving_Bal
0
500
1000
1500
2000
2500

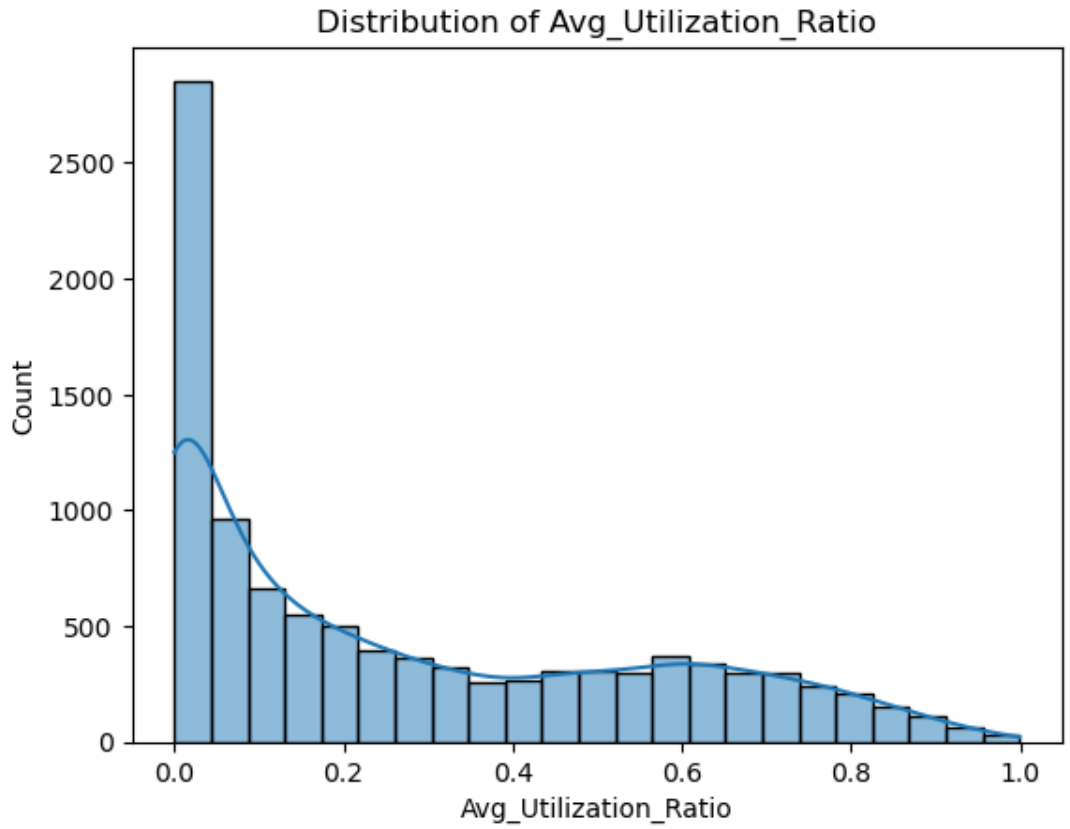

Distribution of Avg_Open_To_Buy
Count
Avg_Open_To_Buy
0
250
500
750
1000
1250
1500
1750
0
5000
10000
15000
20000
25000
30000
35000


Distribution of Total_Amt_Chng_Q4_Q1
Count
Total_Amt_Chng_Q4_Q1
0
100
200
300
400
500
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

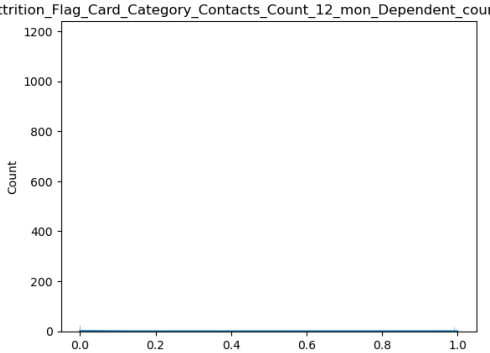

Distribution of Total_Trans_Amt
Count
Total_Trans_Amt
0
100
200
300
400
500
600
700
800
0
2500
5000
7500
10000
12500
15000
17500


Distribution of Total_Trans_Ct
Count
Total_Trans_Ct
0
100
200
300
400
500
600
700
800
20
40
60
80
100
120
140


Distribution of Total_Ct_Chng_Q4_Q1
Count
Total_Ct_Chng_Q4_Q1
0
100
200
300
400
500
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

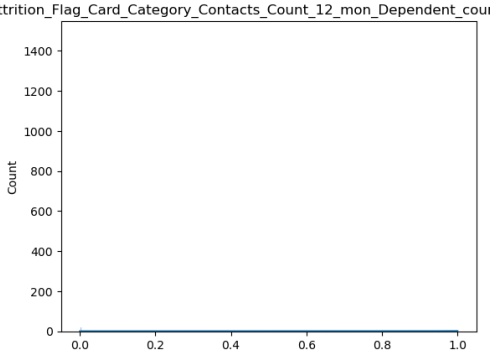


Distribution of Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1



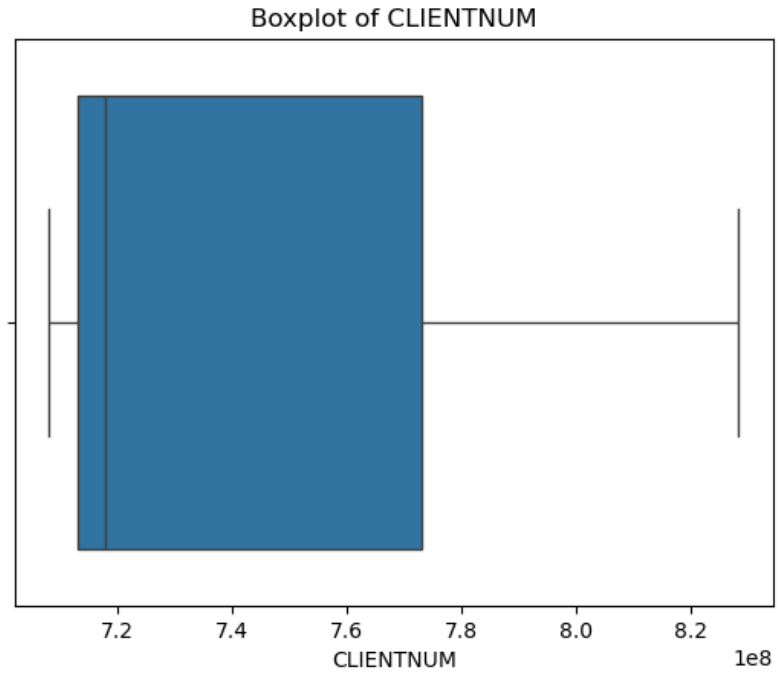
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1

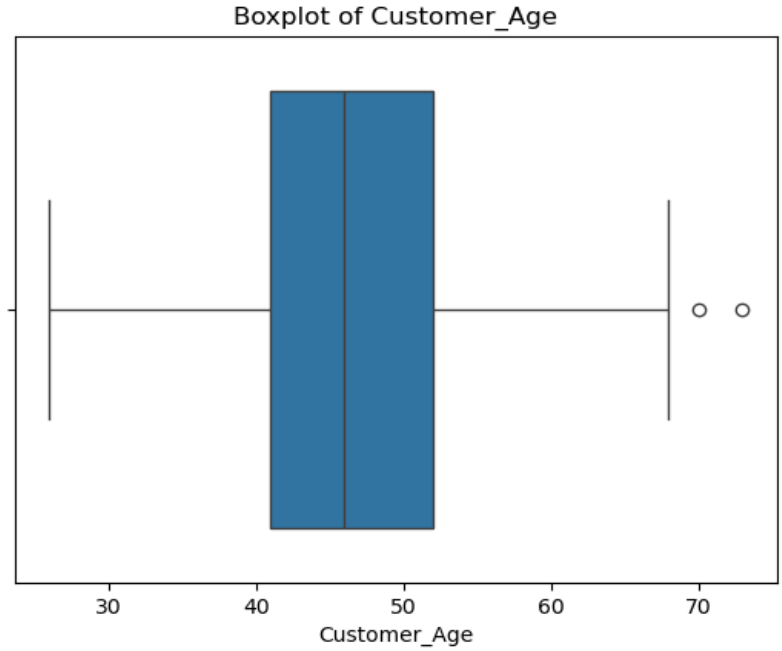
Distribution of Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2

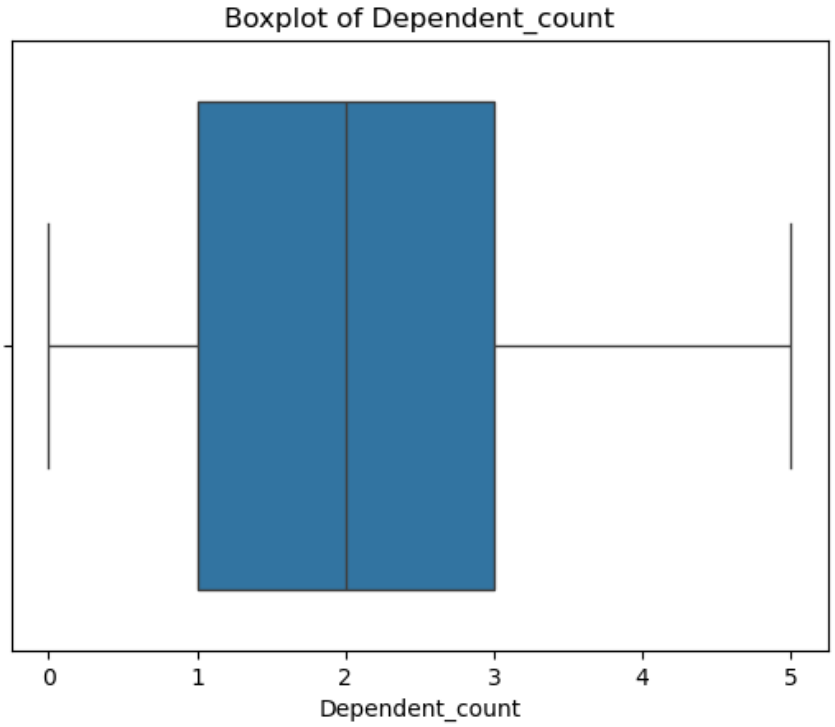


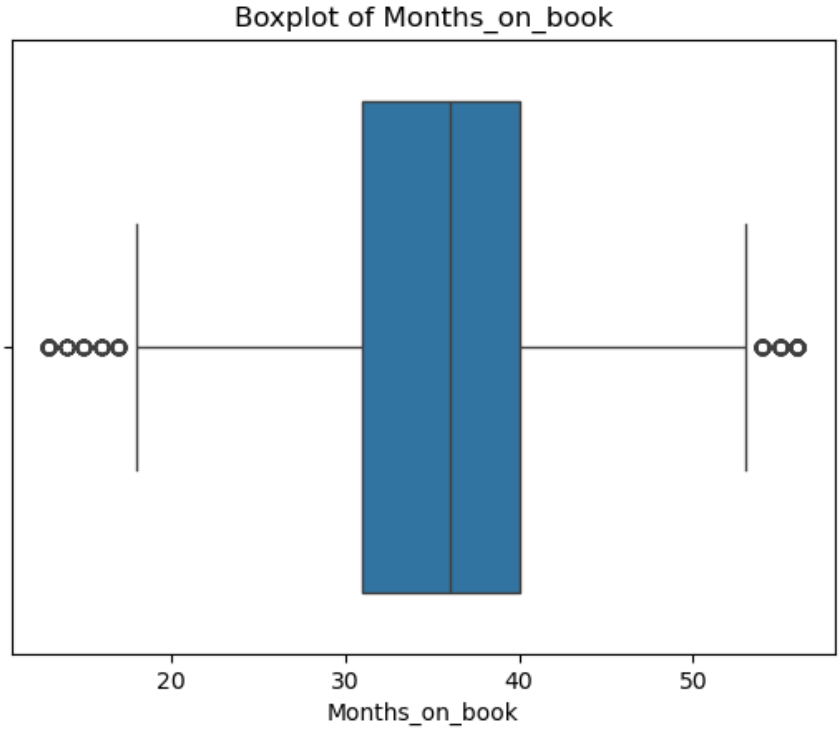
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2

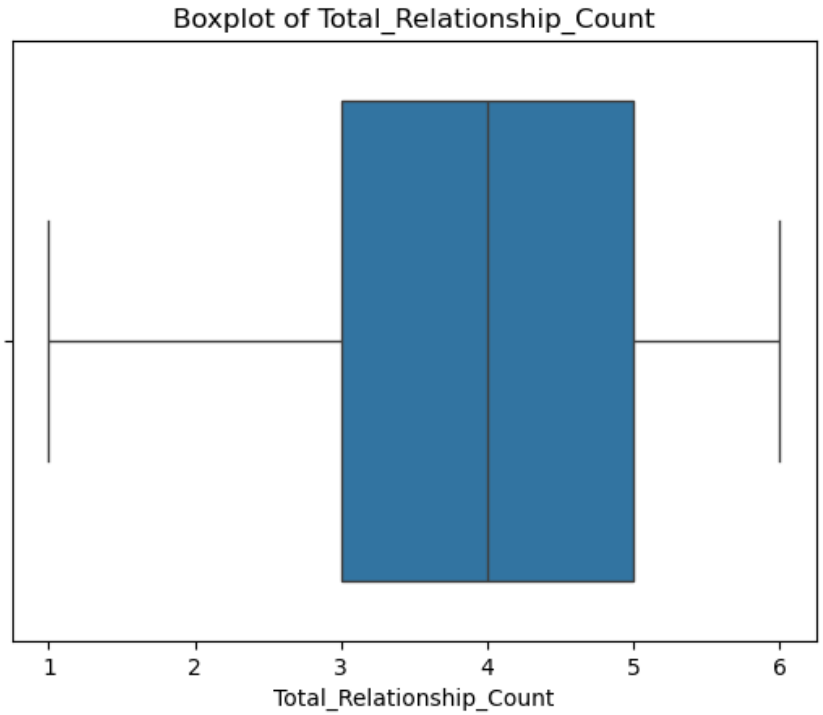
```python
# Box plots for outliers
for col in numerical_cols:
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot of {col}")
    plt.show()
```

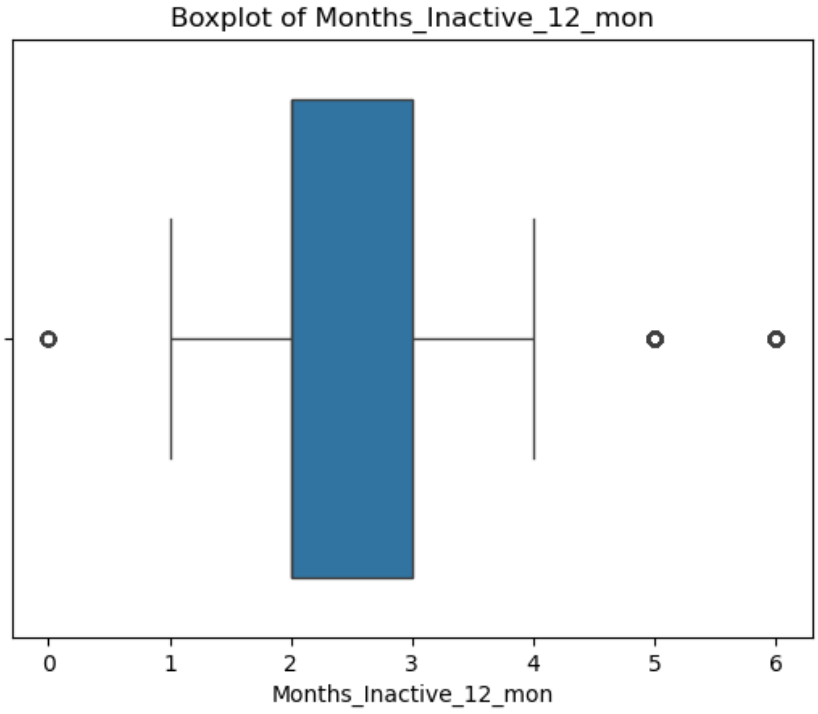


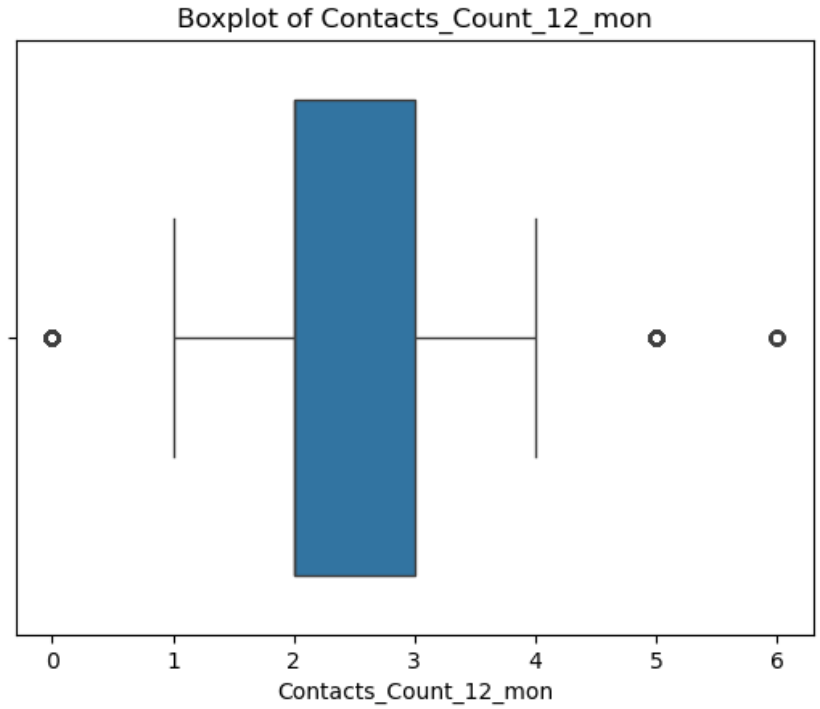


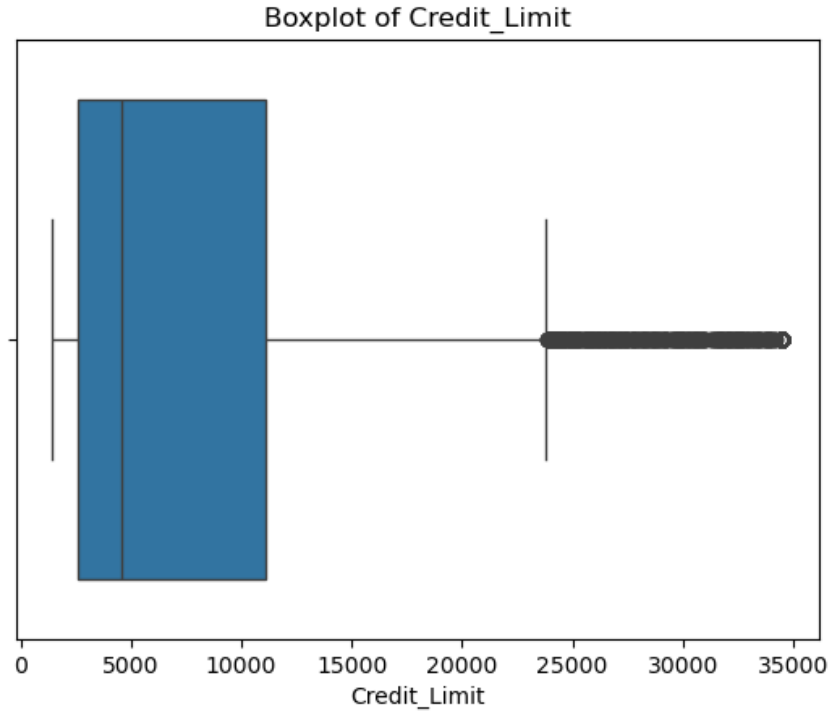

Boxplot of Dependent_count
0
1
2
3
4
5
Dependent_count

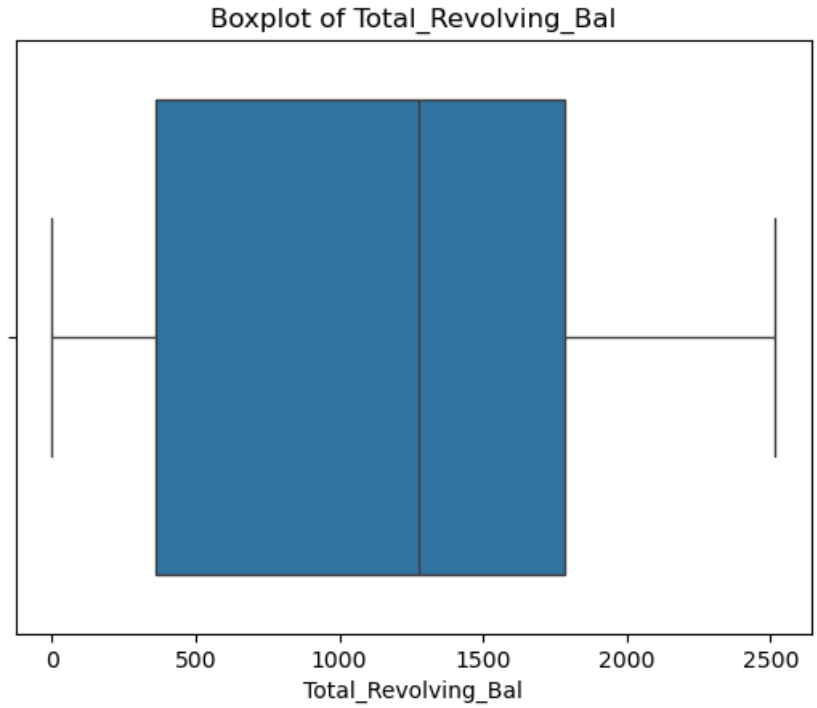

Boxplot of Months_on_book
20
30
40
50
Months_on_book

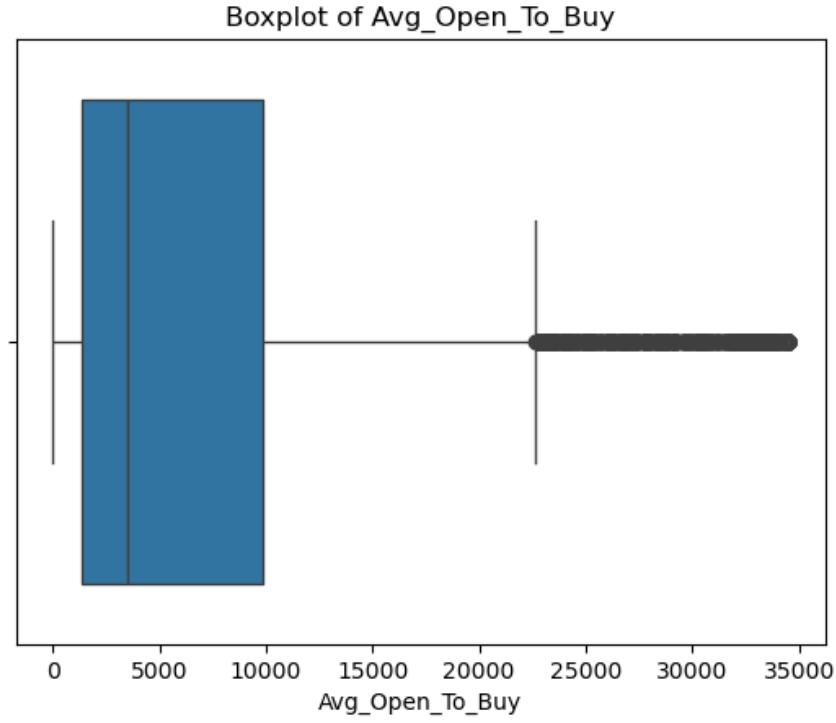

Boxplot of Total_Relationship_Count
1
2
3
4
5
6
Total_Relationship_Count

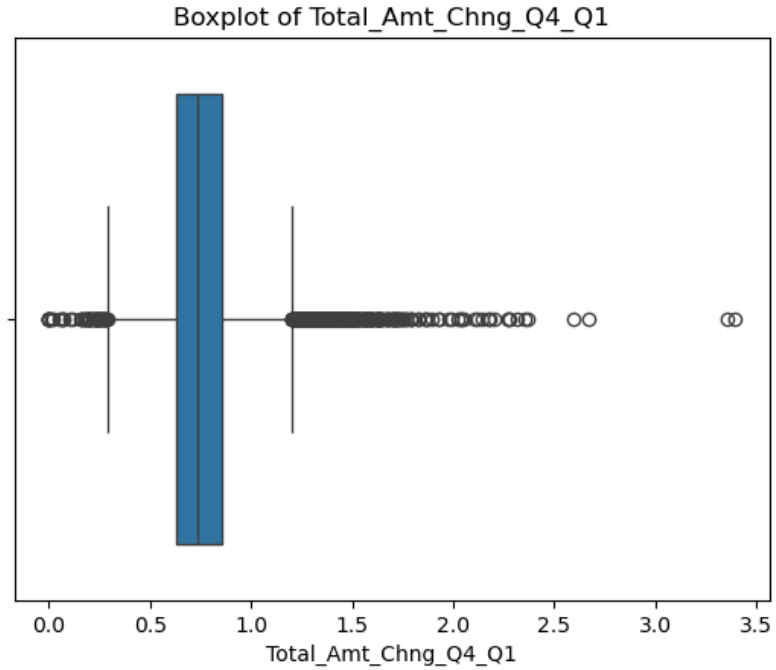

Boxplot of Months_Inactive_12_mon
0
1
2
3
4
5
6
Months_Inactive_12_mon

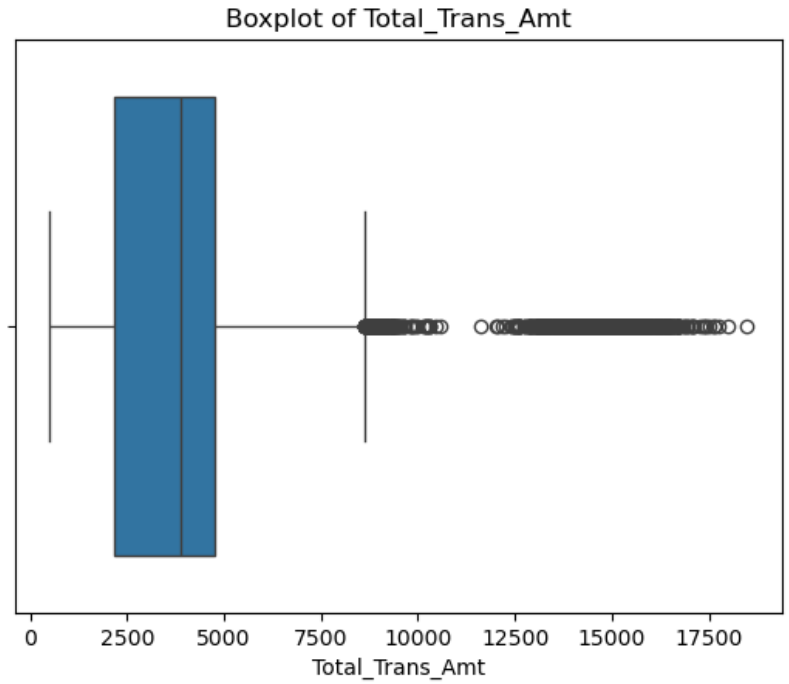

Boxplot of Contacts_Count_12_mon
0
1
2
3
4
5
6
Contacts_Count_12_mon

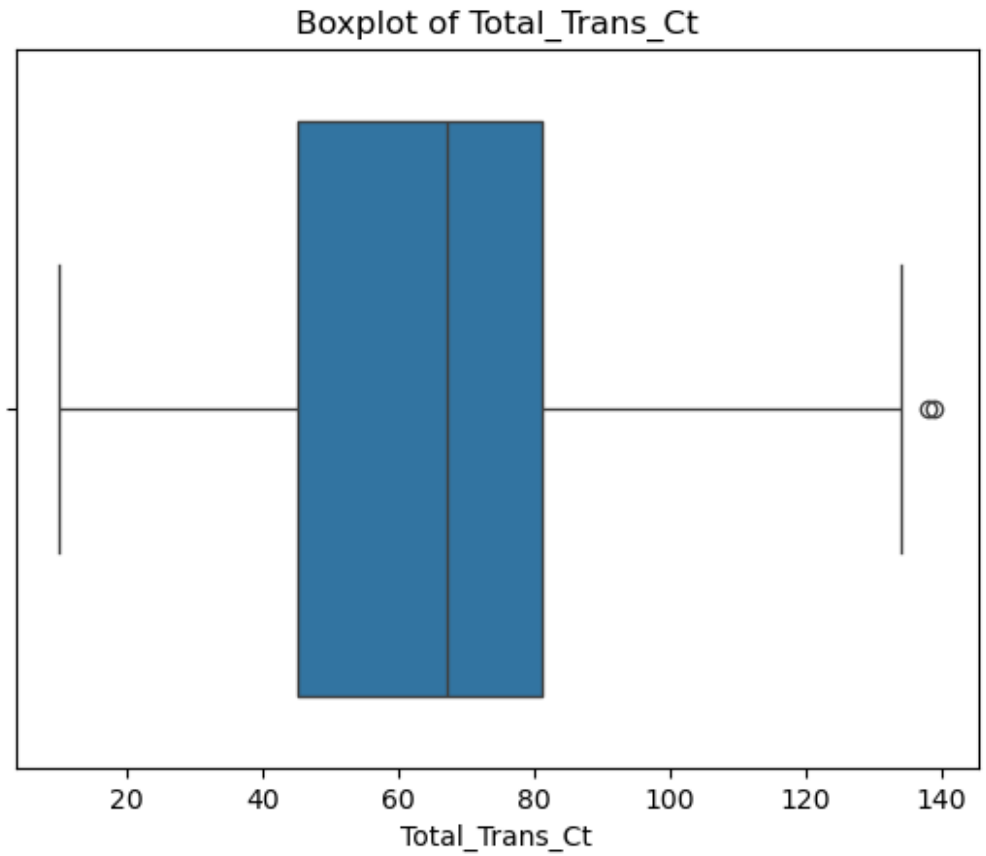

Boxplot of Credit_Limit
0
5000
10000
15000
20000
25000
30000
35000
Credit_Limit

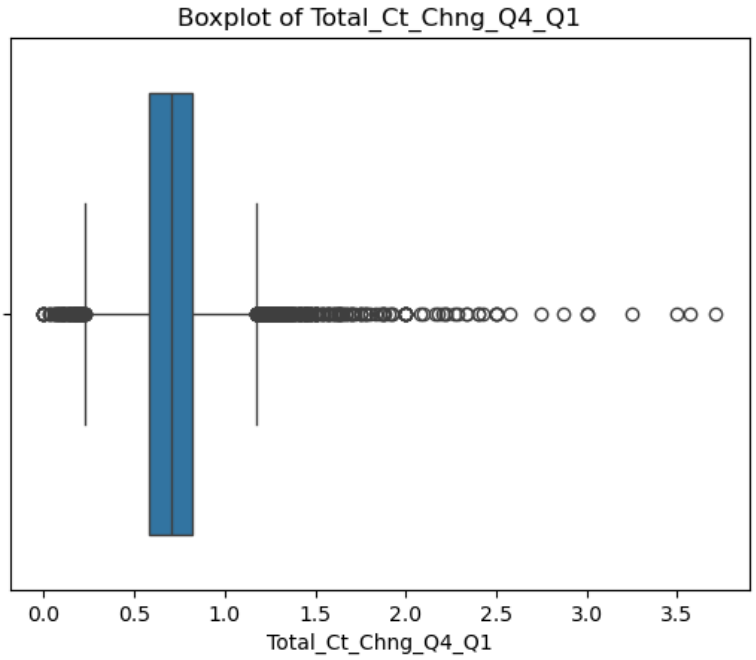

Boxplot of Total_Revolving_Bal
0
500
1000
1500
2000
2500
Total_Revolving_Bal

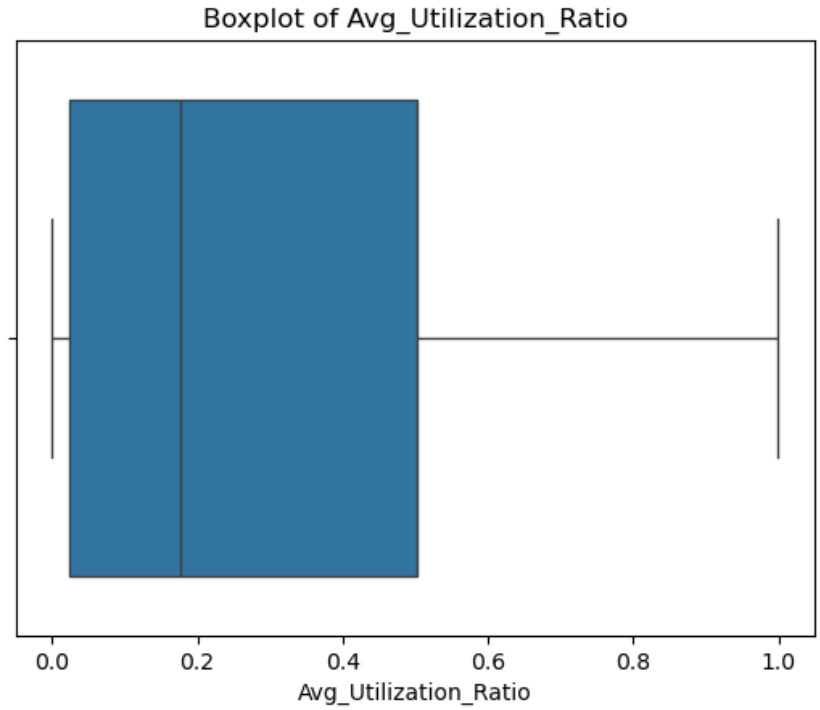

Boxplot of Avg_Open_To_Buy
0
5000
10000
15000
20000
25000
30000
35000
Avg_Open_To_Buy


Boxplot of Total_Amt_Chng_Q4_Q1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Total_Amt_Chng_Q4_Q1

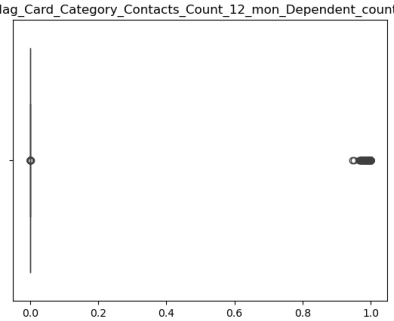

Boxplot of Total_Trans_Amt
0
2500
5000
7500
10000
12500
15000
17500
Total_Trans_Amt


Boxplot of Total_Trans_Ct
20
40
60
80
100
120
140
Total_Trans_Ct


Boxplot of Total_Ct_Chng_Q4_Q1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Total_Ct_Chng_Q4_Q1

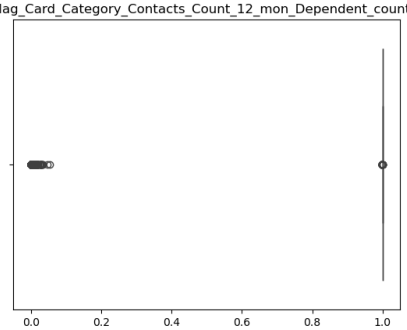


Boxplot of Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1



Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1

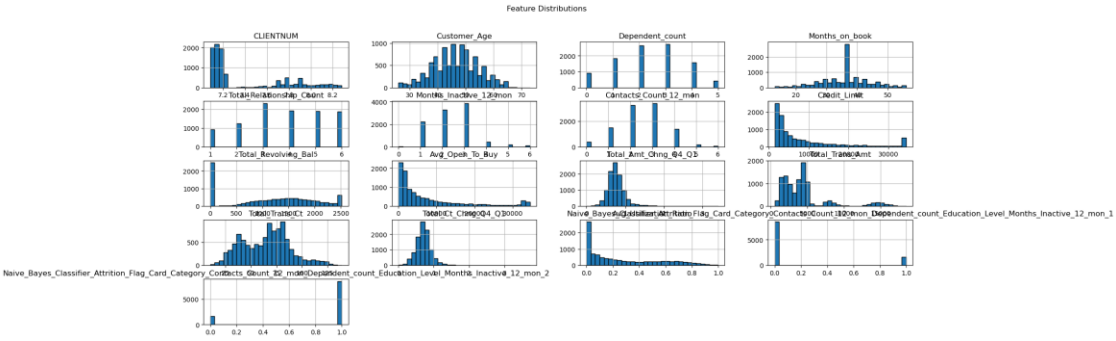
Boxplot of Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2



Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2

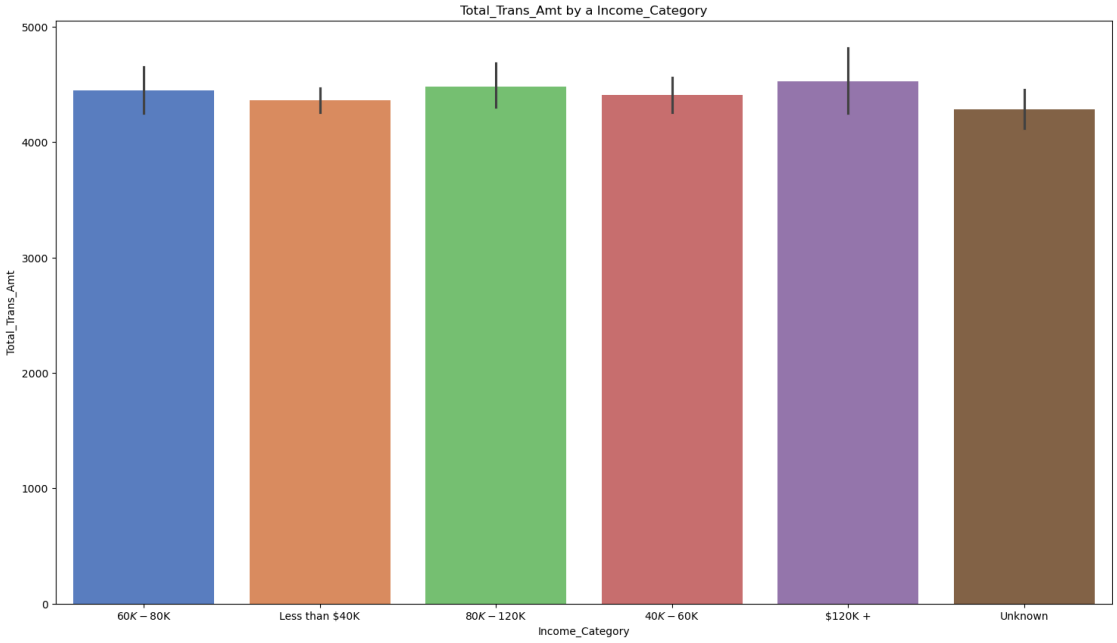
```
#EDA:Visualizing Data Distributions
plt.figure(figsize=(20,6))
df.hist(bins=30,figsize=(20,8),edgecolor="black")
plt.suptitle("Feature Distributions")
plt.show()
```

```
<Figure size 2000x600 with 0 Axes>
```

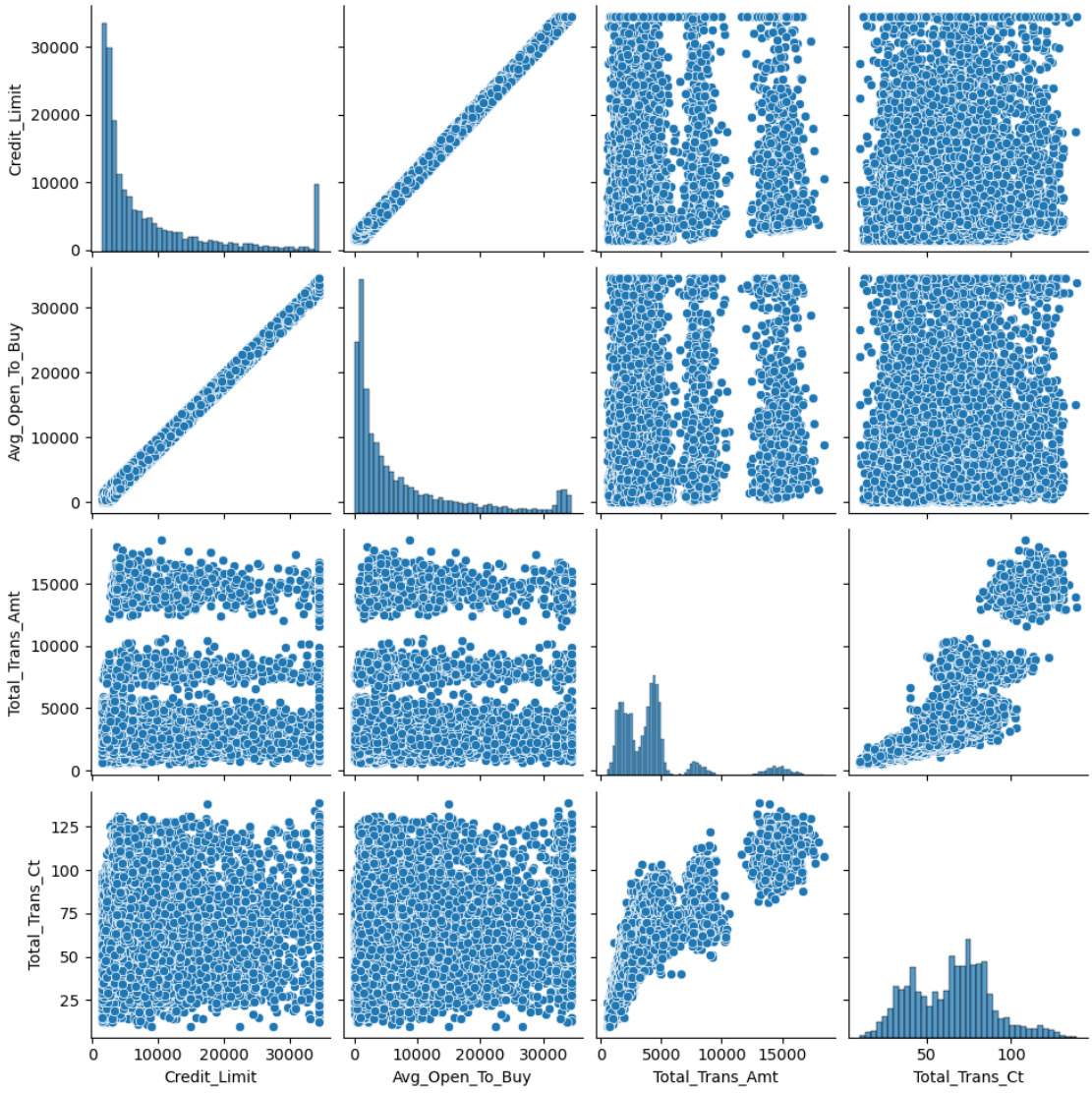


👆 This shows us each variable's that spreads a data across different value range

```
# Bar plot of Total_Trans_Amt by a Income_Category
plt.figure(figsize=(18, 10))
sns.barplot(x='Income_Category', y='Total_Trans_Amt', data=df ,
palette='muted')
plt.title('Total_Trans_Amt by a Income_Category')
plt.xlabel('Income_Category')
plt.ylabel('Total_Trans_Amt')
plt.show()
```



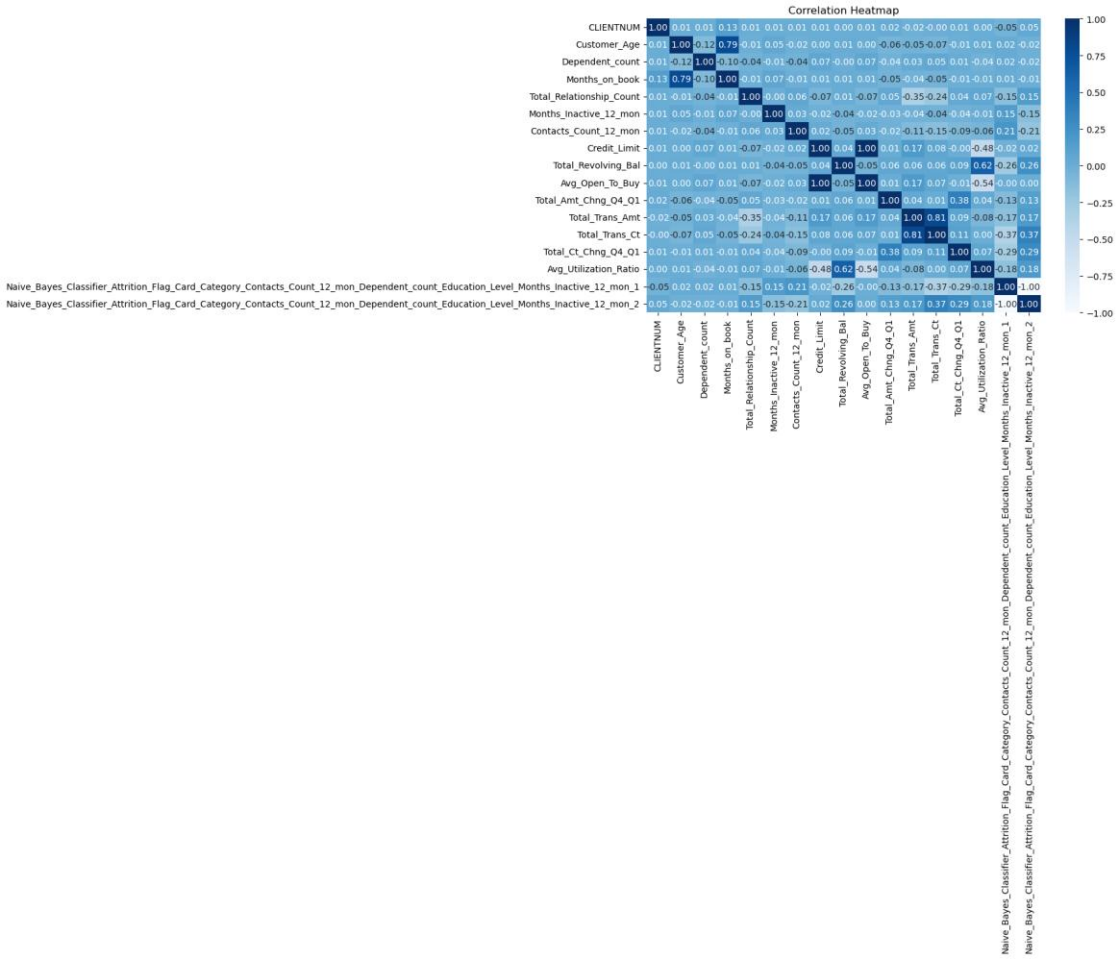
```
# Visualisations
# Pairplot to visualize relationships between Credit_Limit,
Avg_Open_To_Buy,Total_Trans_Amt,and Total_Trans_Ct
sns.pairplot(df[["Credit_Limit", "Avg_Open_To_Buy", "Total_Trans_Amt",
"Total_Trans_Ct"]])
plt.show()
```



🔥 Heatmap - Correlation Matrix

```
# 📌 7. Correlation Analysis
# Droping non-numeric (object) columns before heatmap to avoid errors
numeric_df = df.select_dtypes(exclude="object")

# Plot heatmap to visualize correlation between numerical features
plt.figure(figsize=(10, 6))
```

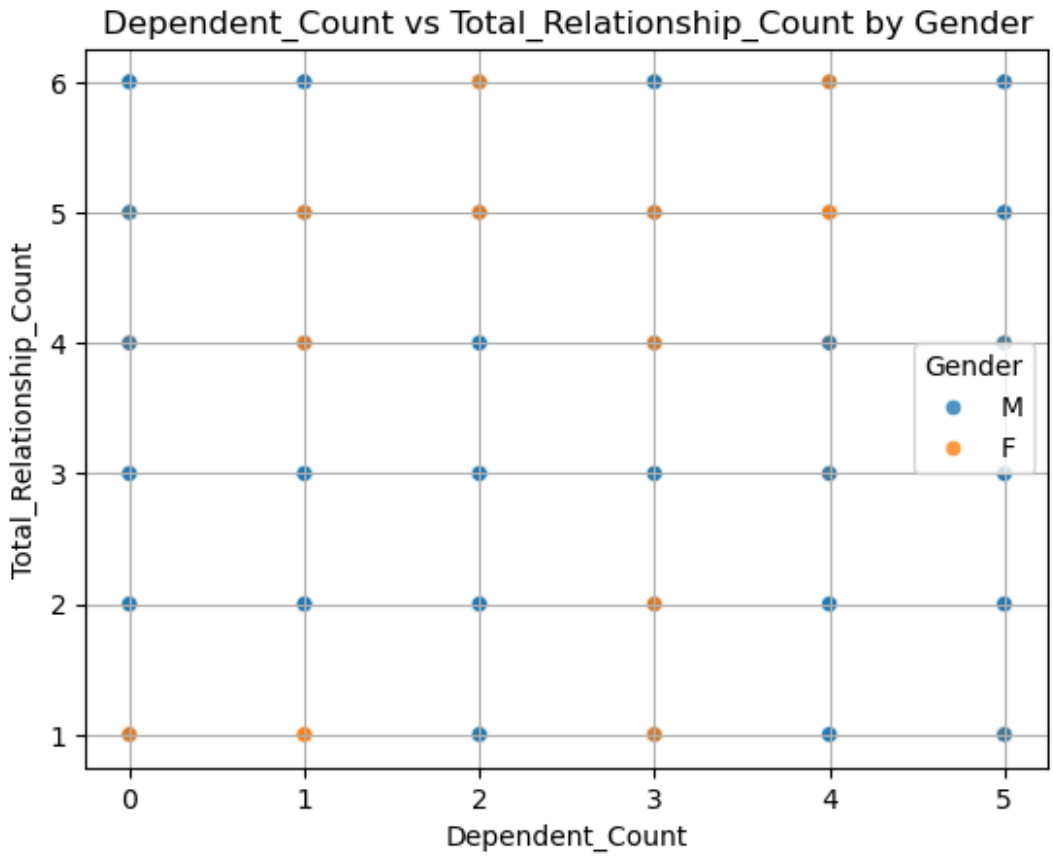
```
sns.heatmap(numeric_df.corr(), annot=True, cmap="Blues", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



☝ Correlation matrix helps us to find the most target variable and it shows the high relationship between variables in positive or negative ranges which is useful for predictation, and it detect multicollinearity features.

- +1 = Perfect positive correlation
- -1 = Perfect negative correlation
- 0 = No linear relationship

```
# scatterplot (Dependent_Count vs Total_Relationship_Count by Gender)
sns.scatterplot(x='Dependent_count',
y='Total_Relationship_Count',hue='Gender',  data=df, alpha=0.8)
plt.title('Dependent_Count vs Total_Relationship_Count by Gender')
plt.xlabel('Dependent_Count')
plt.ylabel('Total_Relationship_Count')
plt.grid(True)
plt.show()
```



☝ Scatterplot shows us the relationship between variables and also to find outliers.

- Show a negative or positive trend between Dependent_Count and Gender.
- Help us to make bank churners decision, while we notice a Total_Relation_Count increases and Gender increases.

```
# Boxplot - To detect Outliers
# Select numeric columns only
numeric_df = df.select_dtypes(exclude="object")

# Set up the figure size
plt.figure(figsize=(12, 6))

# Create a boxplot for each numeric column
sns.boxplot(data=numeric_df, orient="h")
plt.title("Boxplot of Numeric Columns - Outlier Detection")
plt.tight_layout()
plt.show()
```


```
## Outlier Detection (IQR method)
# Select only numeric columns for IQR outlier detection
numeric_df = df.select_dtypes(exclude="object")

# IQR-based outlier detection
Q1 = numeric_df.quantile(0.25)
Q3 = numeric_df.quantile(0.75)
IQR = Q3 - Q1

# Filter out outliers
df_num = df[~((numeric_df < (Q1 - 1.5 * IQR)) | (numeric_df > (Q3 + 1.5 *
IQR))).any(axis=1)]
```

📊 Univariate and Multivariate Analysis

◇ Univariate Analysis

Univariate analysis is a statistical examination of a single variable at the time.it helps to describe the basic characteristics of the data, such as its distribution, central tendency(mean,median,mode),and description(range,variance,standard deviation).

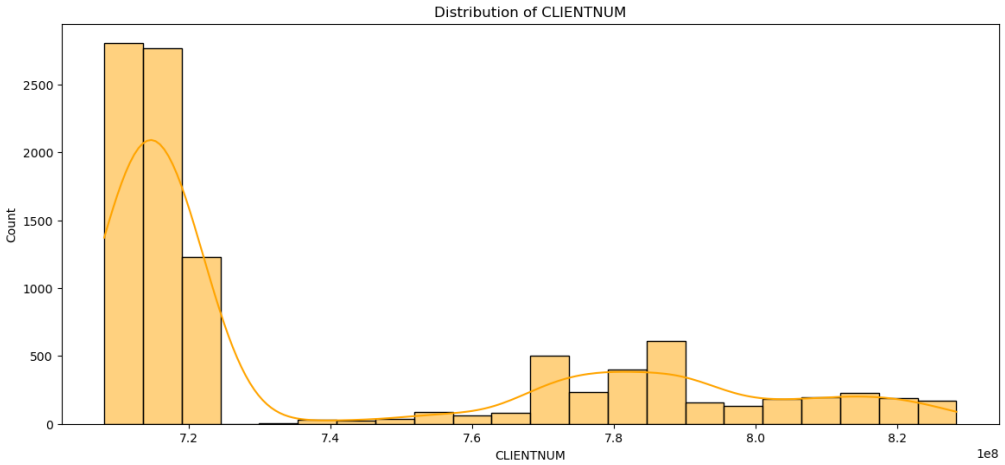
◇ Multivariate Analysis

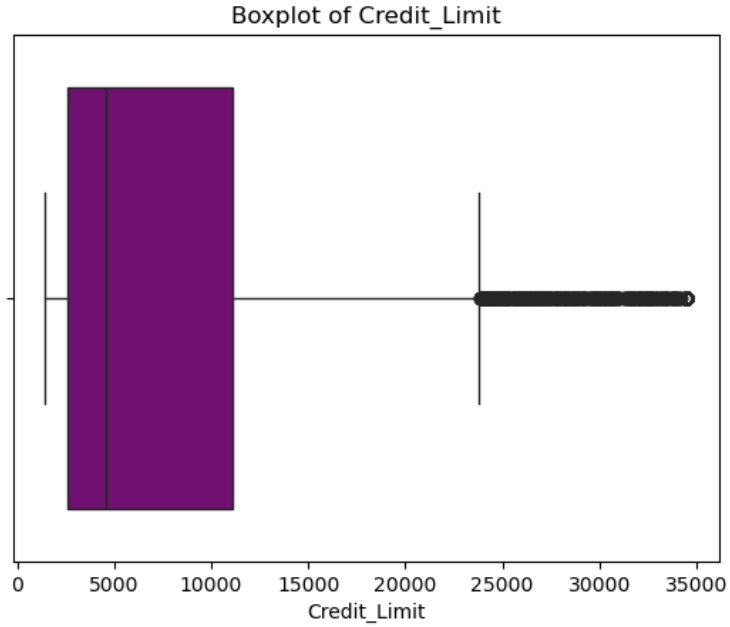
Multivariate analysis is a statistical technique used to examine the relationship between three or more variables at the same time. it helps to identify patterns, correlations and interactions among variables to understand how they influence each other.

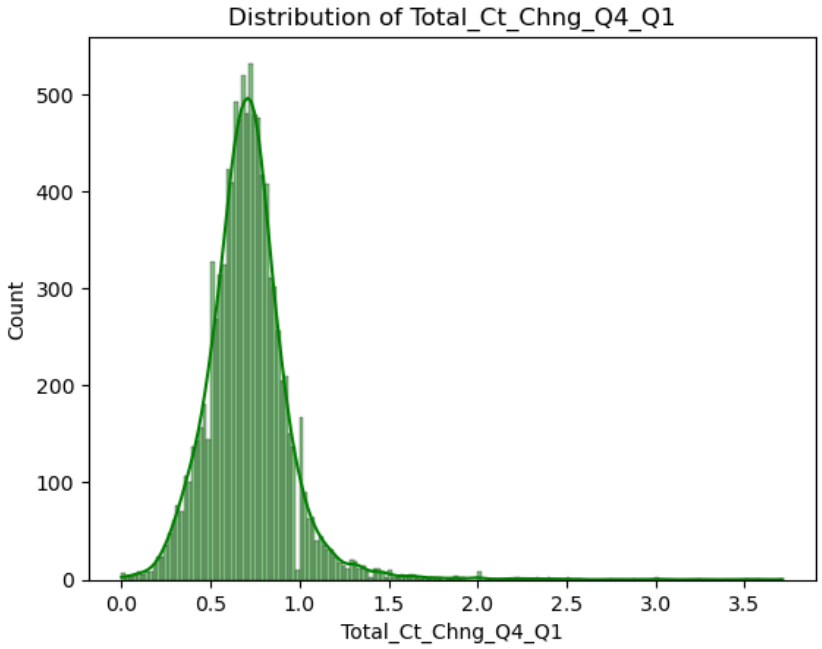
We'll explore both types of analysis using visualizations and summary Summary statistics for key columns like `Credit_Limit`, `Avg_Open_To_Buy`, `Avg_Utilization_Ratio`, `Total_Ct_Chng_Q4_Q1`,and `CLIENTNUM`

```
# Univariate Analysis - Distribution plots for numeric variables
plt.figure(figsize=(14, 6))
sns.histplot(df['CLIENTNUM'], kde=True, color='orange')
```

```
plt.title('Distribution of CLIENTNUM')
plt.show()

sns.boxplot(x=df['Credit_Limit'], color='purple')
plt.title('Boxplot of Credit_Limit')
plt.show()

sns.histplot(df['Total_Ct_Chng_Q4_Q1'], kde=True, color='green')
plt.title('Distribution of Total_Ct_Chng_Q4_Q1')
plt.show()
```

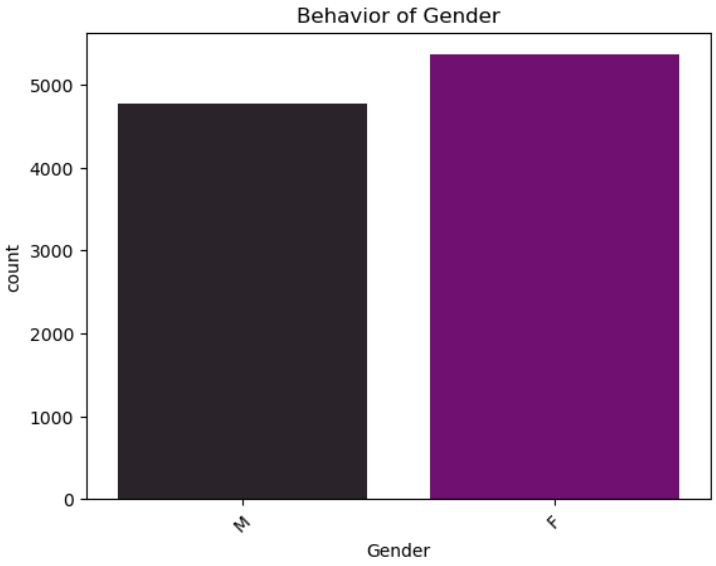


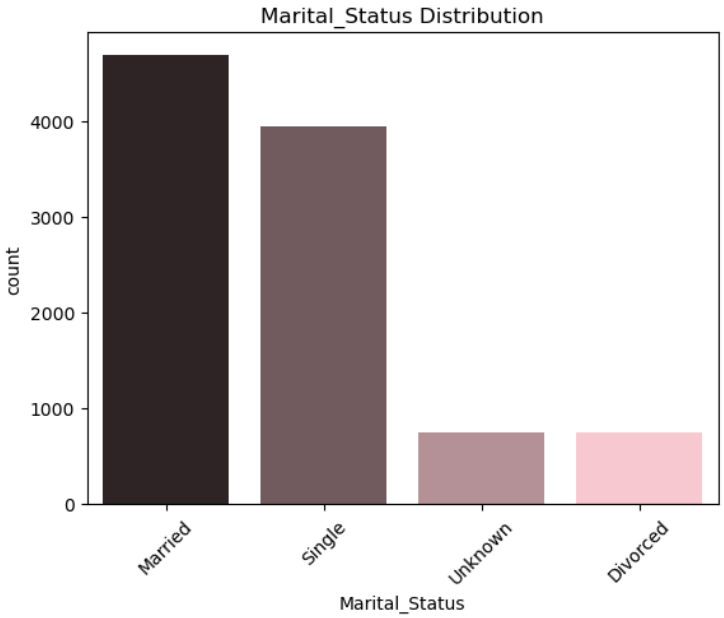




Univariate Analysis - 📐 Categorical variable count plot

```
# Univariate Analysis - Categorical variable count plot
sns.countplot(x='Gender', hue="Gender", data=df, color='purple')
plt.title('Behavior of Gender')
plt.xticks(rotation=45)
plt.show()

sns.countplot(x='Marital_Status', hue='Marital_Status',data=df, color='pink')
plt.title('Marital_Status Distribution')
plt.xticks(rotation=45)
plt.show()
```





👇 Piechat shows has that how much each product category contributes to the whole dataset like:

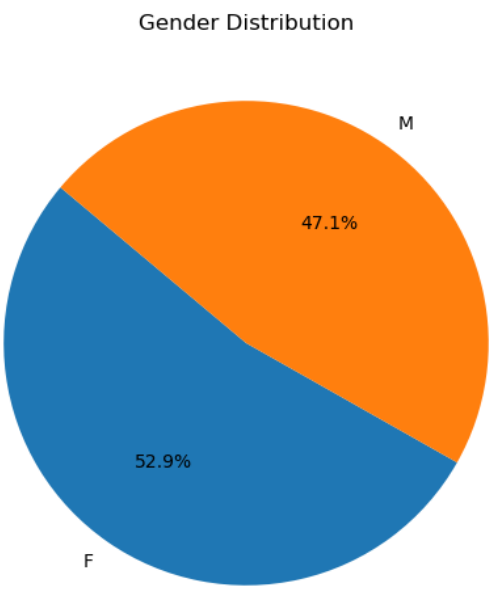
- Which category sells more.
- Which one to promote.
- Where to focus Marital status , Gender , Attrition_Flag.

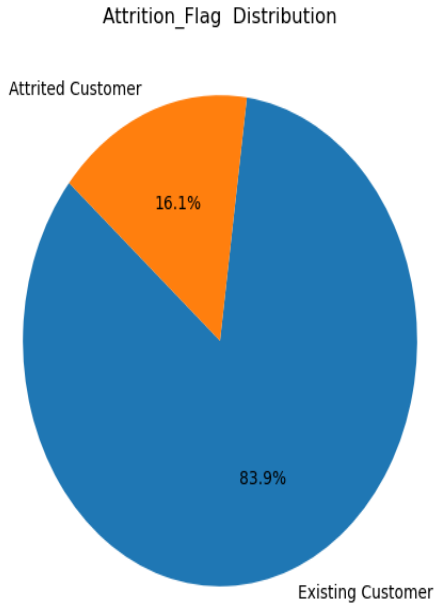
```
# Univariate Pie Chart: Gender distribution
Gender_counts = df['Gender'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(Gender_counts, labels=Gender_counts.index, autopct='%1.1f%%',
startangle=140)
plt.title('Gender Distribution')
plt.show()
```

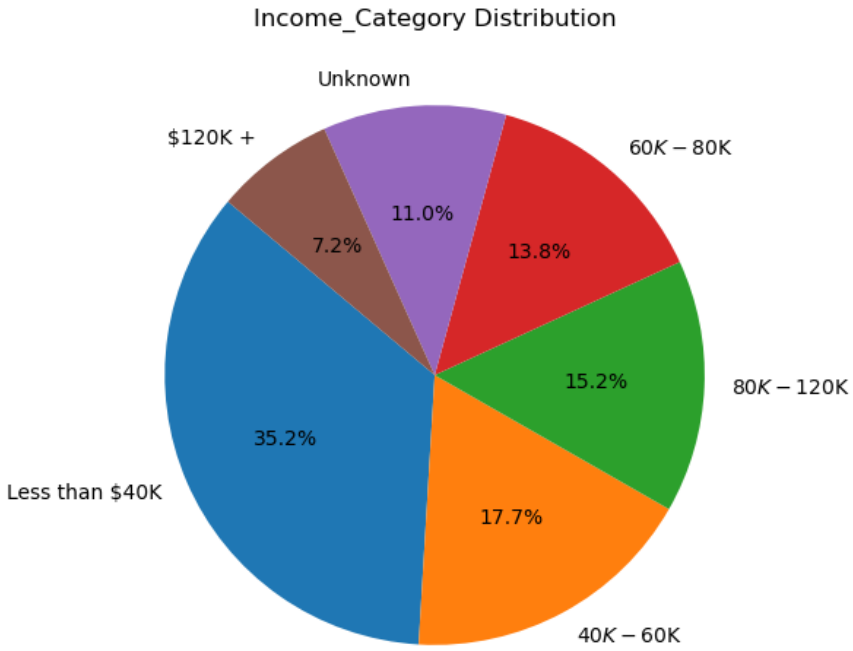
```
# Univariate Pie Chart: Attrition_Flag  distribution
Attrition_Flag_counts = df['Attrition_Flag'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(Attrition_Flag_counts, labels=Attrition_Flag_counts.index,
autopct='%1.1f%%', startangle=140)
plt.title('Attrition_Flag  Distribution')
plt.show()
```

```
# Univariate Pie Chart: Income_Category  distribution
Income_Category_counts = df['Income_Category'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(Income_Category_counts, labels=Income_Category_counts.index,
```

```
autopct='%1.1f%%', startangle=140)
plt.title('Income_Category Distribution')
plt.show()
```



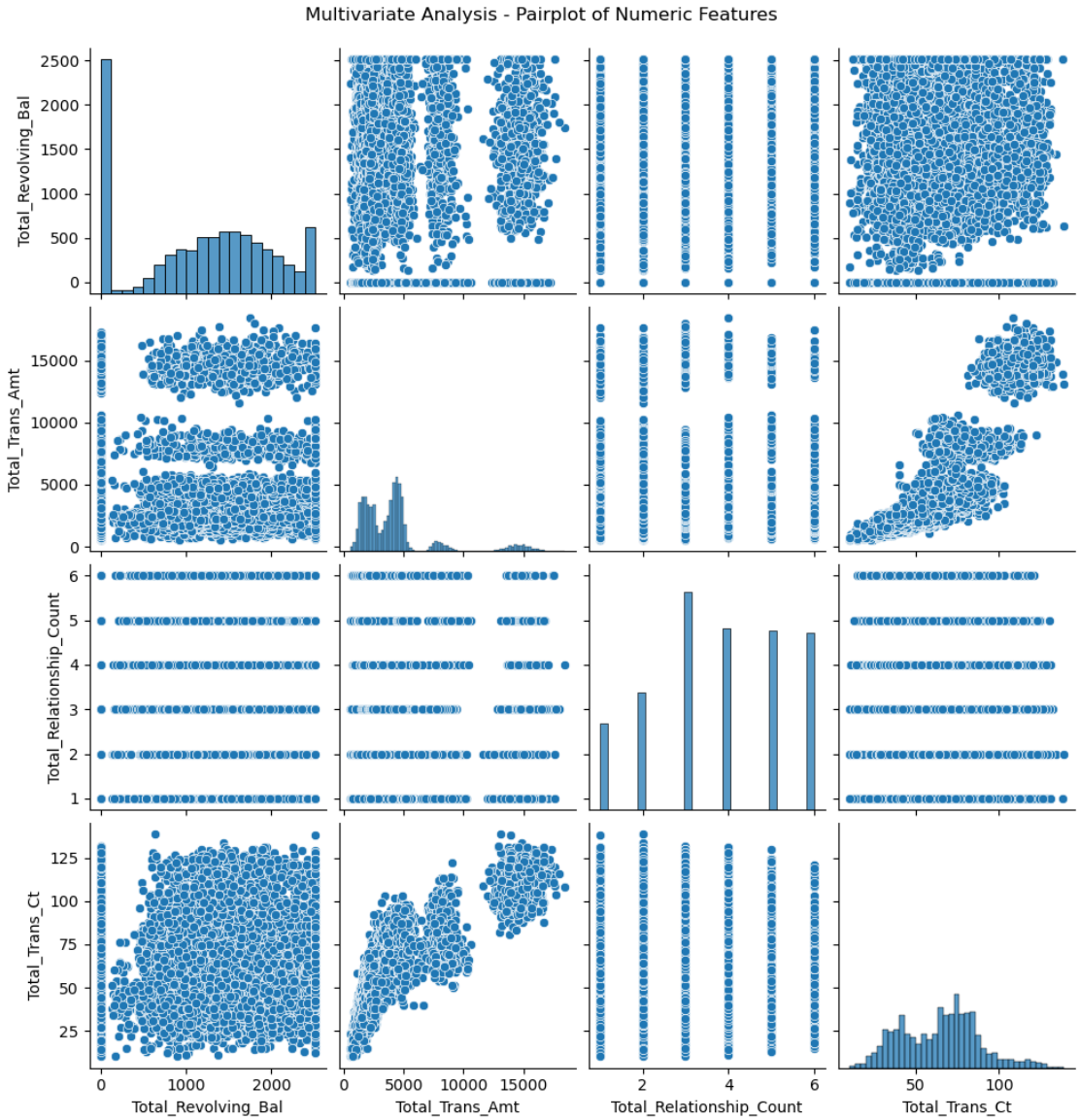




Multivariate Analysis - 🌀 Pairplot of key numeric features

```
# Multivariate Analysis - Pairplot of key numeric features
sns.pairplot(df[["Total_Revolving_Bal", "Total_Trans_Amt",
```

```
"Total_Relationship_Count", "Total_Trans_Ct"]])
plt.suptitle('Multivariate Analysis - Pairplot of Numeric Features', y=1.02)
plt.show()
```



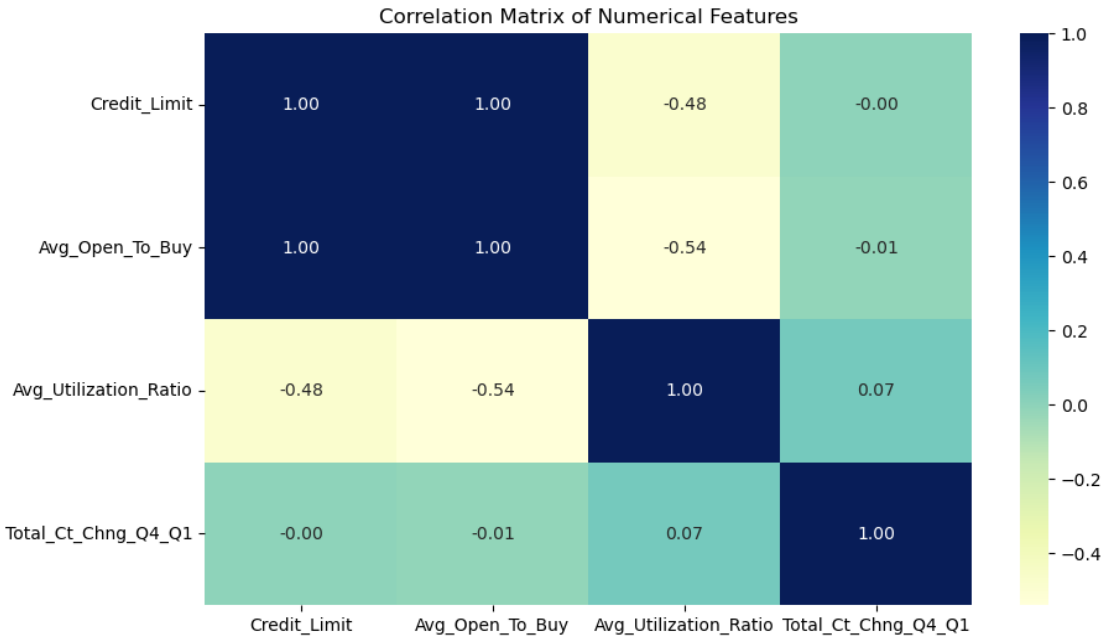
👆 Pair plot is used to Create scatter plots between every pair of numerical features.

- **pairplot**: [Total_Revolving_Bal,Total_Trans_Amt,Total_Relationship_Count,Total_Trans_Ct]

Multivariate Analysis - 🔥 Heatmap for correlation

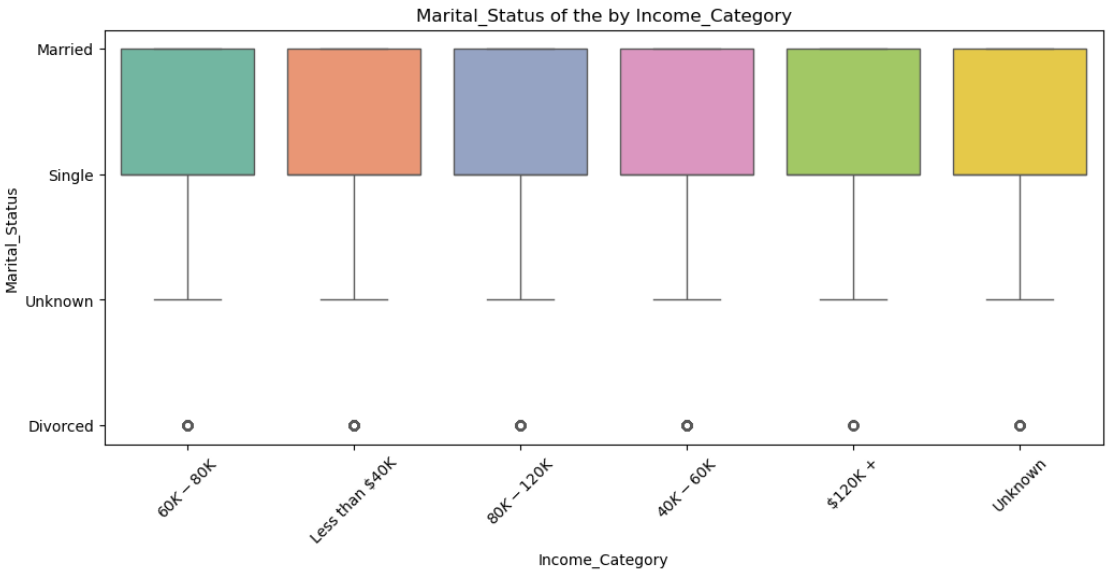
```
# Multivariate Analysis - Heatmap for correlation
plt.figure(figsize=(10, 6))
corr = df[['Credit_Limit', 'Avg_Open_To_Buy', 'Avg_Utilization_Ratio',
'Total_Ct_Chng_Q4_Q1']].corr()
```

```
sns.heatmap(corr, annot=True, cmap='YlGnBu', fmt='.2f')
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```



👆This heatmap is meaningful only if categorical variables are converted into numerical variables.

```
# Multivariate Analysis - Boxplot of Lifetime Value by Region
plt.figure(figsize=(12, 5))
sns.boxplot(x='Income_Category', y='Marital_Status', data=df, palette='Set2')
plt.title('Marital_Status of the by Income_Category')
plt.xticks(rotation=45)
plt.show()
```



👆🏻 Boxplot is used to spread data and detect outliers in the dataset and it helps us to find extreme values.

- Helps to compare how Martial_status vary with different Income_Category conditions.
- it is useful for identifying inconsistencies in stock levels based on Income demand .

📈 Probability & Hypothesis Testing

- Chi-Square test for Independence between category features and churn
- Performed T-test, ANOVA For main differences based on transaction volume vs churn
- Use statistical insects to validate assumptions

4. Statistical Methods & Feature Insights

📙 Probability Distribution Analysis

It is a process of understanding how values in a column are distributed meaning how frequently they occur and how they behave across the data set. To understand how certain numerical variables behave across the user behaviors,product prizes, categories and user sessions. so analyze the probability distribution helps you to understand customer behavior, product pricing patterns, business strategy, trends, and detect patterns.

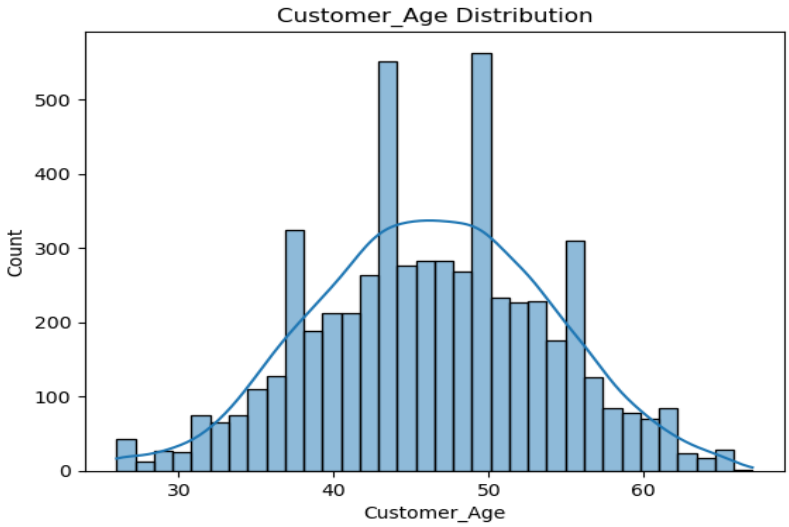
🧨Columns Chosen:

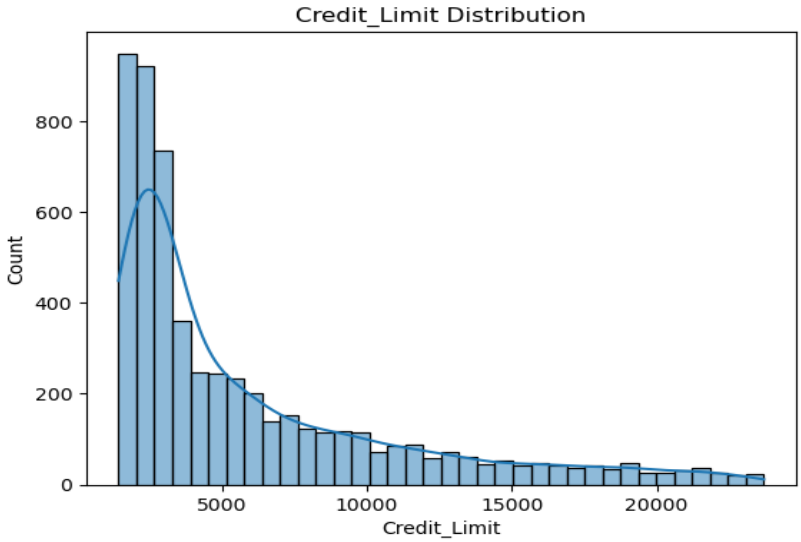
- **Customer_Age**- Check if age is normally distributed.understand age patterns in churners vs non-churners.
- **Credit_Limit**-Right-skewed distributions are common; Can indicate if high-limit customers are more stable.
- **Total_Ct_Chng_Q4_Q1**- Indicates behavior volalitity. dips may be linked with churn.
- **Total_Amt_Chng_Q4_Q1**- Measures how much spending has increased or decreased - often predictive of engagement drop-off
- **Avg_Open_To_Buy**- Helps accesses utilization and potential risk profile

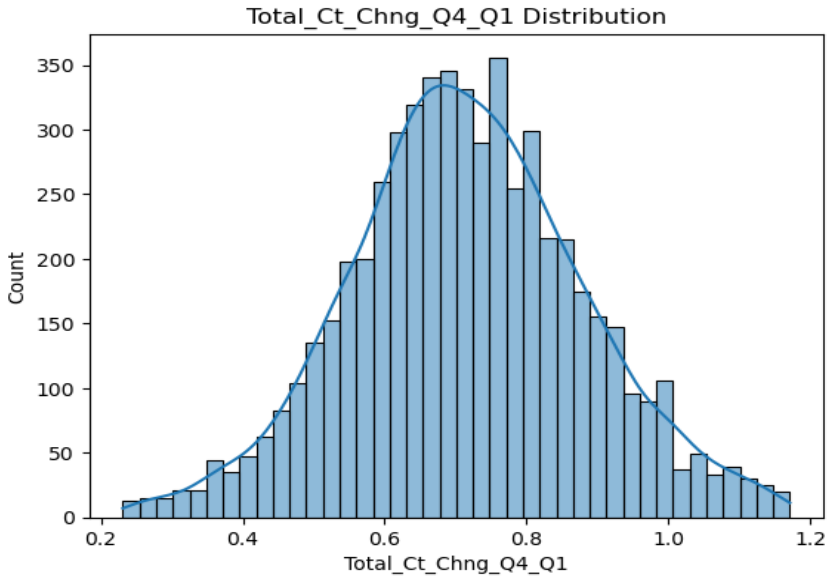
Each variable is visualized below with both histogram and normal distribution fit

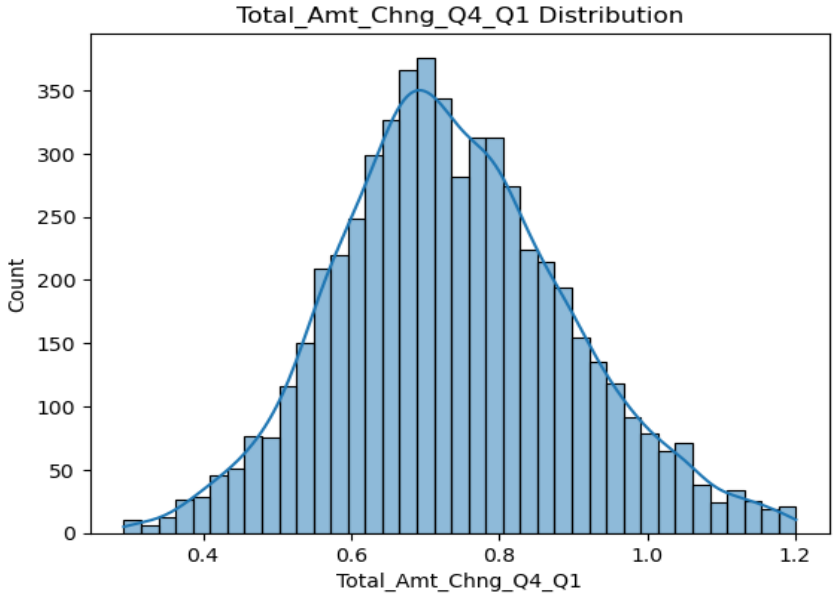
```
# Probability Distribution Check (e.g., Customer_Age)
sns.histplot(df_num['Customer_Age'], kde=True)
plt.title("Customer_Age Distribution")
plt.show()

# Probability Distribution Check (e.g., Credit_Limit)
sns.histplot(df_num["Credit_Limit"], kde=True)
plt.title("Credit_Limit Distribution")
plt.show()
```

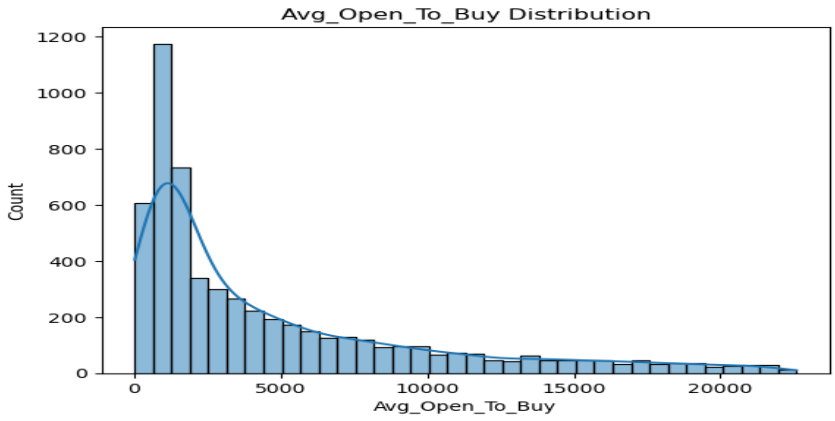
```
# Probability Distribution Check (e.g., Total_Ct_Chng_Q4_Q1)
sns.histplot(df_num["Total_Ct_Chng_Q4_Q1"], kde=True)
plt.title("Total_Ct_Chng_Q4_Q1 Distribution")
plt.show()

# Probability Distribution Check (e.g., Total_Amt_Chng_Q4_Q1)
sns.histplot(df_num["Total_Amt_Chng_Q4_Q1"], kde=True)
plt.title("Total_Amt_Chng_Q4_Q1 Distribution")
plt.show()

# Probability Distribution Check (e.g., Avg_Open_To_Buy)
sns.histplot(df_num["Avg_Open_To_Buy"], kde=True)
plt.title("Avg_Open_To_Buy Distribution")
plt.show()
```






Total_Ct_Chng_Q4_Q1 Distribution
Count
Total_Ct_Chng_Q4_Q1
0
50
100
150
200
250
300
350
0.2
0.4
0.6
0.8
1.0
1.2


Total_Amt_Chng_Q4_Q1 Distribution
Count
Total_Amt_Chng_Q4_Q1
0
50
100
150
200
250
300
350
0.4
0.6
0.8
1.0
1.2


Avg_Open_To_Buy Distribution
Count
Avg_Open_To_Buy
0
200
400
600
800
1000
1200
0
5000
10000
15000
20000

📝 Hypothesis Testing

by doing t-Test, chi-square test that helps us to make data-driven decisions by Statistically evaluating patterns assumptions and differences in data.

```
# Hypothesis Testing ( Credit Limit vs Customer age)
t_stat, p_val = stats.ttest_ind(df_num['Credit_Limit'],
df_num['Customer_Age'])
print("T-test: t=", t_stat, " p=", p_val)
```

```
T-test: t= 86.79636175302821  p= 0.0
```

📈 Customer Churn Prediction (Regression)

Customer Churn Predicition is Technically a classification problem (since the target Attrition_Flag is Categorial: churned or existing).

We used Linear Regression predict the target variable `Attrition_Flag`

- Split data into train and test
- Trained Linear Regression model
- Evaluated with MAE, MSE, and R^2 Score

🪵 Customer Churn Predicition (Regression)

We used Linear Regression to predict the Revenue Generated by a campaign based on key performance indicators.

📈 Features Used:

- **Customer_Age** Total age of the customer
- **Credit_Limit** Credit limit allocated to the customer
- **Total_Ct_Chng_Q4_Q1** The percentage change in transaction count from Q4 to Q1
- **Total_Amt_Chng_Q4_Q1** The percentage change in transaction amount from Q4 to Q1
- **Avg_Open_To_Buy** Average Available credit to spend for example credit limit - current balance

📐 Model Setup

- Split the dataset into training and testing sets (80% train, 20% test)
- Trained a Linear Regression model using scikit-learn
- Made predictions on the test data

📊 **Evaluation Metrics Mean Squared Error (MSE)** – Measures average squared difference between actual and predicted values

Mean Absolute Error (MAE) – Measures average absolute difference

R^2 Score – Indicates how well the model explains the variance in revenue (1.0 is perfect)

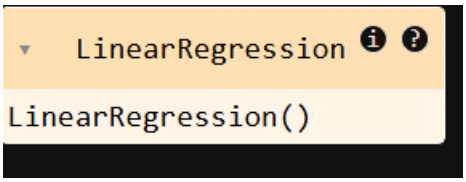
📉 Visualization

Plotted Actual vs Predicted Customer Churn using a scatter plot

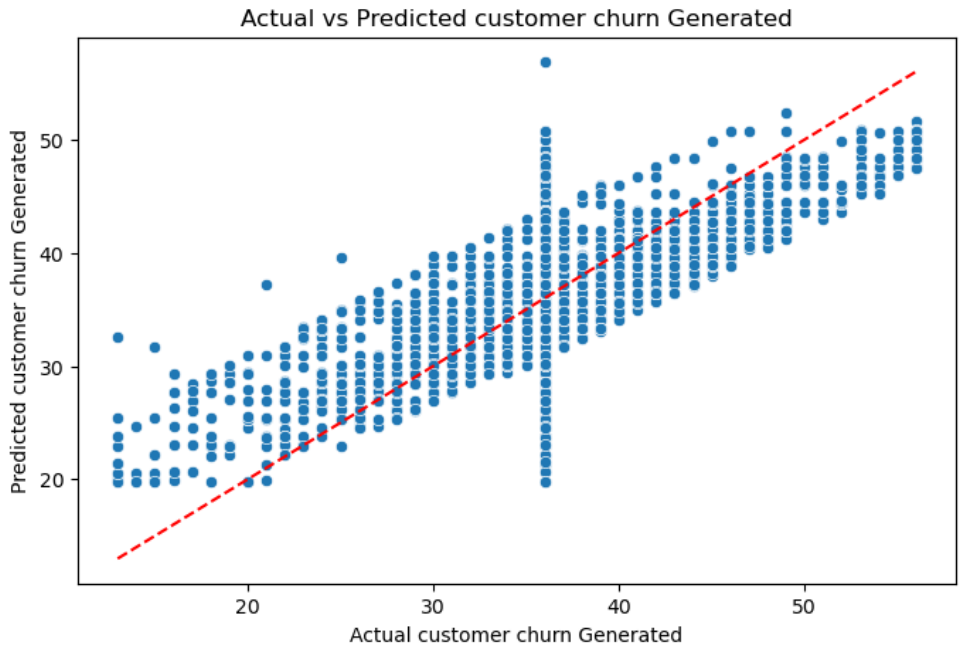
A red dashed line shows the ideal prediction line (perfect fit)

This helps evaluate both numerical performance and visual accuracy of our regression model.

```
# 📌 8. Linear Regression for  Prediction
features_reg = ["Customer_Age", "Avg_Open_To_Buy", "Total_Ct_Chng_Q4_Q1",
"Total_Amt_Chng_Q4_Q1"]
X_reg = df[features_reg]
y_reg = df["Months_on_book"]

X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg,
y_reg, test_size=0.2, random_state=42)

model_reg = LinearRegression()
model_reg
```




```
model_reg.fit(X_train_reg, y_train_reg)
y_pred_reg = model_reg.predict(X_test_reg)

# Plot predicted vs actual
plt.figure(figsize=(8, 5))
sns.scatterplot(x=y_test_reg, y=y_pred_reg)
plt.plot([y_test_reg.min(), y_test_reg.max()], [y_test_reg.min(),
y_test_reg.max()], 'r--')
plt.xlabel("Actual customer churn Generated")
plt.ylabel("Predicted customer churn Generated")
plt.title("Actual vs Predicted customer churn Generated")
plt.show()
```




```
# Evaluation Metrics
mse = mean_squared_error(y_test_reg, y_pred_reg)
mae = mean_absolute_error(y_test_reg, y_pred_reg)
r2 = r2_score(y_test_reg, y_pred_reg)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"R² Score: {r2:.2f}")
```

```
Mean Squared Error (MSE): 24.62
Mean Absolute Error (MAE): 3.91
R² Score: 0.60
```

🧠Random Forest Classification(Classification)

We created a new label Churn To classify Complex patterns in customer behavior Which handles non-linear relationship between features and provide feature importance to understand, what influence churn. and Which works well even if the data is bit imbalanced.

🛠️**Using feature importance random forest** tells you which feature matter most in predicting churn. for example:

- A high change in transaction amount or low credit utilization may reduce churn risk
- Older customers might have different churn patterns than younger ones.

High Risk(0) Low Risk(1) Medium Risk(2)

Dropped unnecessary columns like IDs and text fields.

Encoded categorical features using LabelEncoder.

🤖 Model Training

Trained the logistic regression model on the cleaned dataset

Used multi class classification with the multi nominal option

Split data into training and test sets.

📊 Evaluation Metrics

- **Accuracy**- measures how often model is correct overall. Overall correctness of predictions.
- **Precision**- Measures how many customers labeled as a class are actually that class. Predicted "High" were actually high
- **Recall**- measures how well the model identifies all actual customers of a class. Actual "High" work correctly predicted.

```
# Step 1: Create a binary target column
df['Churn'] = df['Attrition_Flag'].apply(lambda x: 1 if x == 'Attrited
Customer' else 0)

# Step 2: Drop irrelevant columns (IDs, text fields)
df_cleaned = df.drop(['Customer_Age', 'Credit_Limit', 'Total_Ct_Chng_Q4_Q1',
'Total_Amt_Chng_Q4_Q1',
                      'Avg_Open_To_Buy', ], axis=1)

# Step 3: Encode categorical features
label_encoders = {}
for col in df_cleaned.select_dtypes(include='object'):
    le = LabelEncoder()
    df_cleaned[col] = le.fit_transform(df_cleaned[col])
    label_encoders[col] = le

# Step 4: Define features (X) and target (y)
X_clf = df_cleaned.drop('Churn', axis=1)
y_clf = df_cleaned['Churn']

# Step 5: Split the dataset
X_train_clf, X_test_clf, y_train_clf, y_test_clf = train_test_split(X_clf,
y_clf, test_size=0.2, random_state=42)

# Step 6: Train the Logistic Regression model
model = LogisticRegression(max_iter=1000, class_weight='balanced')
model.fit(X_train_clf, y_train_clf)
```

```
LogisticRegression
LogisticRegression(class_weight='balanced', max_iter=1000)
```

⏩ Accuracy and AUC Score:

- **High accuracy**
- **High Recall** for churners(which is critical)
- **AUC Score** above 0.85 is considered very good

```
# Step 7: Predictions and Evaluation
y_pred_clf = model.predict(X_test_clf)
print("Accuracy:", accuracy_score(y_test_clf, y_pred_clf))
print("Confusion Matrix:\n", confusion_matrix(y_test_clf, y_pred_clf))
print("Classification Report:\n", classification_report(y_test_clf,
y_pred_clf))
```

```
Accuracy: 0.7828232971372162
Confusion Matrix:
 [[1326  373]
 [  67  260]]
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.78      0.86      1699
           1       0.41      0.80      0.54       327

    accuracy                           0.78      2026
   macro avg       0.68      0.79      0.70      2026
weighted avg       0.86      0.78      0.81      2026
```

📨 Confusion Matrix

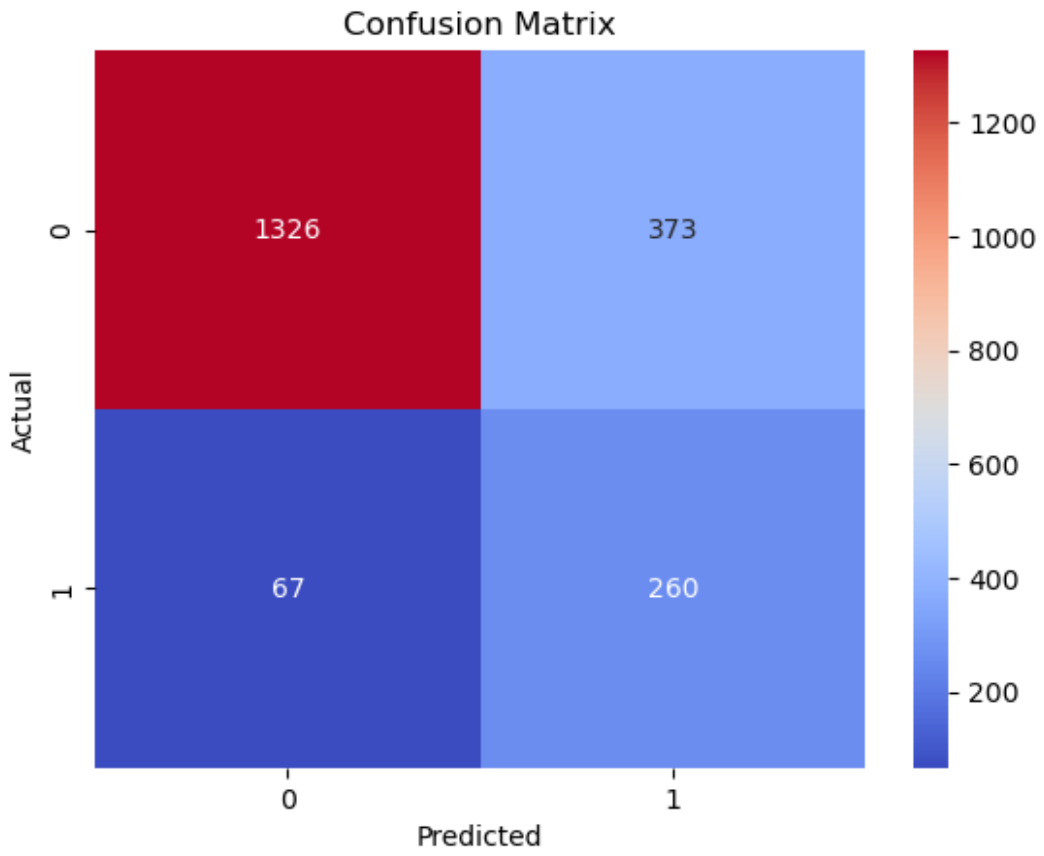
A confusion matrix is a table used to evaluate the performance of a classification model. It shows how well the model is predicting actual classes.

For a multi-class classification the Matrix compares:

- **Actual classes**(True labels) vs.
- **Predicted classes**(From your model)

```
cm = confusion_matrix(y_test_clf, y_pred_clf)
sns.heatmap(cm, annot=True, fmt="d", cmap="coolwarm")
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
```

```
plt.ylabel("Actual")
plt.show()
```




```
print("\nClassification Report:\n")
print(classification_report(y_test_clf, y_pred_clf))
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.95      0.78      0.86      1699
           1       0.41      0.80      0.54       327

    accuracy                           0.78      2026
   macro avg       0.68      0.79      0.70      2026
weighted avg       0.86      0.78      0.81      2026
```

📝 Final Conclusion

In this project aimed to predict **customer churn risk levels**(High,Medium,Low) Using the **BankChurners dataset** through data analysis and machice learning models, Particularly **Random Forest** and **Logistic Regression Classifiers**.

A structured data science process we extracted actionable insights and made a predictive models to support business decisions.

🎯 Project Goals

The Main objective of this project was to leverage customer banking data to **predict Churn risk levels** and understand customer behavior using two major machine learning tasks:

🪜 Customer Churn Predicition (Regression):

To model continuous customer related trends and relationships generated using linear regression. And predicted how much customer churn might happend in Bnakers churn.

📉 Random Forest Classification (Classification):

To categorizes customers into **High,Medium or Low** risk of churn Using logistic regression.

🔄 Workflow Followed

We applied the key steps of a typical data science project:

✅ **Collected and cleaned the data**

📊 **Explored and visualized patterns**

📐 **Applied statistical techniques**

🤖 **Built and evaluated machine learning models**

📌 Key Learnings

- Transaction activity and utilization rate are strong indicators of churn behavior.
- Regression helped understand trends in credit and usage limits tied to customer age.
- Classification enabled early identification of potential churners with high accuracy.
- Feature importance from models provided clear business insights, For example decrease in transaction frequency often precedes churn.
- Real world datasets often need significant preprocessing and feature creation to unlock predictive power.

References:

- **Imran Ali Shah**. (2023). *Sales and Customer Insights Dataset*. Kaggle. Available at: https://www.kaggle.com/datasets/whenamancodes/credit-card-customers-prediction

- Pandas Documentation – Data preprocessing, handling missing values, and exploratory data analysis. https://pandas.pydata.org/docs/

- **Virtanen, P., Gommers, R., Oliphant, T. E., et al.** (2020). *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*. Nature Methods, 17, 261–272. (Used for hypothesis testing, probability distributions)

- **McKinney, W.** (2010). *Data Structures for Statistical Computing in Python*. In Proceedings of the 9th Python in Science Conference, 51–56. (Used for structured data handling with Pandas)

- **Waskom, M. L.** (2021). *Seaborn: Statistical Data Visualization*. Journal of Open-Source Software, 6(60), 3021. (Used for histogram and KDE plots in probability analysis)

- **Pedregosa, F., Varoquaux, G., Gramfort, A., et al.** (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. (Used for linear regression and classification algorithms)