LOAN APPROVAL ANALYSIS

By :-
K.Mounika

INTRODUCTION

A loan is a financial contract where a lender provides funds to a borrower, who agrees to repay the amount, known as the principal, with added interest over a predetermined period. The interest compensates the lender for the risk and opportunity cost of lending. Loan agreements outline terms such as the repayment schedule, interest rate, and any associated fees. Loans can be secured, requiring collateral as protection for the lender, or unsecured, relying solely on the borrower's creditworthiness. Common types include personal loans, mortgages, and auto loans. Failure to repay as agreed can lead to penalties, damage to credit, and legal consequences.

Dataset

I used a dataset of loan applications, which is a common resource for learning exploratory data analysis (EDA). This dataset contains detailed records of **367 unique loan applications**. Analyzing this dataset provides insights into approval patterns, borrower characteristics, and key factors that influence loan approval outcomes.

Purpose Of The Project

The goal of the loan approval analysis project is to thoroughly explore and examine loan application data. This involves checking the data for errors, summarizing key statistics, and visualizing information like approval rates and borrower details. The project also aims to find loan status, understand what affects loan approvals, and assess risks. Ultimately, it seeks to offer useful insights and recommendations to enhance the loan approval process and manage risks better.

Dataset Loading

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from datetime import datetime
from datetime import date
```

The above lines of code import essential libraries for performing data analysis and creating visualizations in Python. **pandas** is used for **data manipulation**, **numpy** for **numerical operations**, and **matplotlib and seaborn** for **creating static plots**. **plotly.express** is used for **interactive visualizations**, while **datetime and date handle date and time operations.**

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
copy='/content/drive/MyDrive/Copy of loan_sanction_test.csv'
df=pd.read_csv(copy)
df
```

```
# creating the the copy of original data.
df_copy= df.copy()
```

```
df_copy
```

This code connects Google Drive to the Colab environment so you can access your files. It then sets the path to a CSV file with loan information and reads this file into a DataFrame using pandas. Finally, it display the data which is saved in **'df'** variable, then i **creates a new DataFrame (df_copy) that is a copy of the existing DataFrame (df), ensuring that changes to 'df_copy' do not affect df .**

Description of Dataset

There are total 367 rows and 12 columns present in our dataset.

```
df.shape
```

```
(367, 12)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            367 non-null    object 
 1   Gender             356 non-null    object 
 2   Married            367 non-null    object 
 3   Dependents         357 non-null    object 
 4   Education          367 non-null    object 
 5   Self_Employed      344 non-null    object 
 6   ApplicantIncome    367 non-null    int64  
 7   CoapplicantIncome  367 non-null    int64  
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object 
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

Loan_ID:- A unique identifier for each loan application.
Gender:- The gender of the applicant (Male/Female).
Married:- The marital status of the applicant (Yes/No).
Dependents:- The number of dependents the applicant has.
Education:- The applicant's educational background (Graduate/Not Graduate).
Self_Employed:-Whether the applicant is self-employed (Yes/No).
ApplicantIncome:- The applicant's annual income.

CoapplicantIncome:- The co-applicant's annual income (if applicable).
LoanAmount:- The amount of loan requested.
Loan_Amount_Term:- The term of the loan in months.
Credit_History:-The applicant's credit history (1 indicates good credit, 0 indicates bad credit).
Property_Area:- The location of the property for which the loan is being sought (Urban/Semiurban/Rural).
Loan_Status:- Whether the loan was approved or not.

```
df_copy['Loan_Status'] = df_copy['Credit_History'].apply(lambda x: 'Approved' if x == 1 else 'Rejected')
df_copy.head()
```

Here i have created the loan status column using credit history column.

```
[ ] df.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	367.000000	367.000000	362.000000	361.000000	338.000000
mean	4805.599455	1569.577657	136.132597	342.537396	0.825444
std	4910.685399	2334.232099	61.366652	65.156643	0.380150
min	0.000000	0.000000	28.000000	6.000000	0.000000
25%	2864.000000	0.000000	100.250000	360.000000	1.000000
50%	3786.000000	1025.000000	125.000000	360.000000	1.000000
75%	5060.000000	2430.500000	158.000000	360.000000	1.000000
max	72529.000000	24000.000000	550.000000	480.000000	1.000000

There are 5 numeric columns i.e ‘ApplicantIncome’, ‘CoapplicantIncome’, ‘LoanAmount’, ‘Credit_History’ ‘Loan_Amount_Term’ and rest of the columns are catagorical columns i.e ‘Loan_ID’, ‘Gender’, ‘Married’ ‘Self_Employed’, ‘Education’, ‘Dependents’, ‘Loan_Status’, ‘Property_Area’

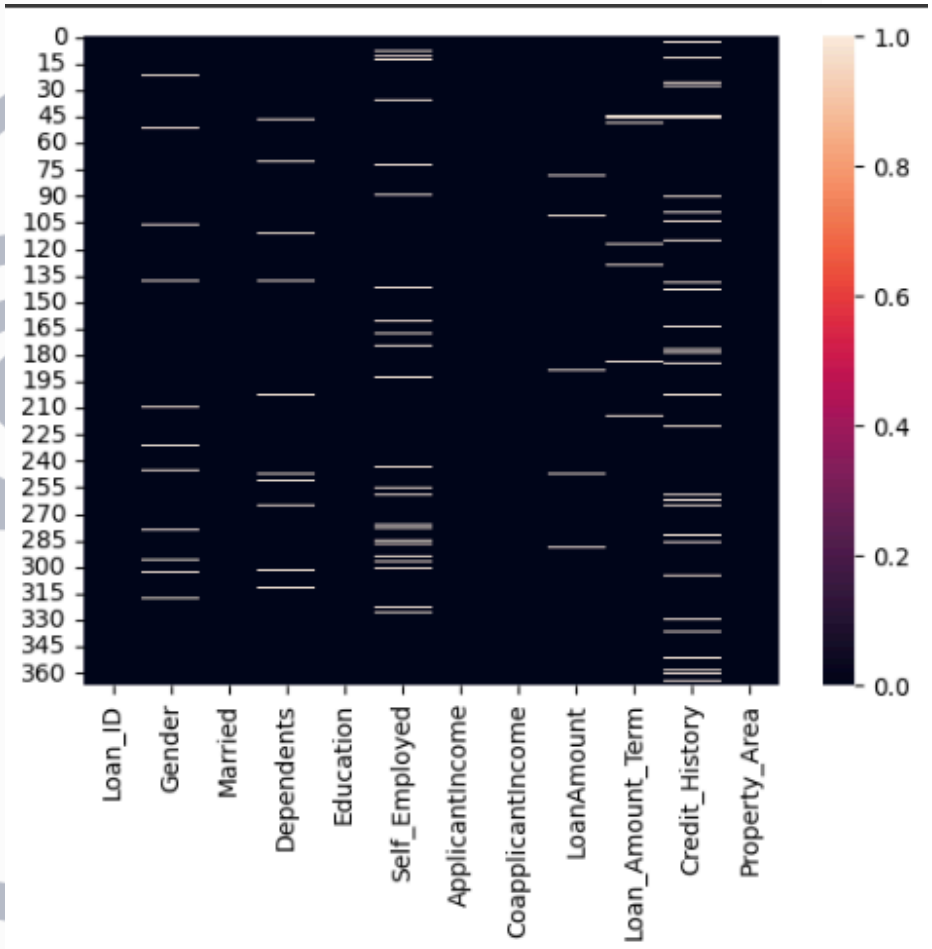
Handeling Null values

```
df_copy.isnull().sum()
```

	0
Loan_ID	0
Gender	11
Married	0
Dependents	10
Education	0
Self_Employed	23
ApplicantIncome	0
CoapplicantIncome	0
LoanAmount	5
Loan_Amount_Term	6
Credit_History	29
Property_Area	0

```
df_copy.isnull().sum().sum()
```

```
84
```



There are 84 null values **excluding the Loan_status** . For instance, **Self_Employed has 23 missing entries**, while **Credit_History has 29** and **Dependents has 10 missing entries, gender has 11 missing entries ,LoanAmount has 5 and Loan_Amount_Term has 6** . **I also use heatmap that depicts the null values** These missing values may need to be addressed before further analysis.

```
df_copy['LoanAmount'].median()
```

```
125.0
```

```
df_copy['LoanAmount'].fillna(df_copy['LoanAmount'].median(), inplace=True)
df_copy['LoanAmount'].isnull().sum()
```

```
0
```

```
df_copy['Credit_History'].median()
```

```
1.0
```

```
df_copy['Credit_History'].replace(np.NaN,1.0,inplace=True)
df_copy['Credit_History'].isnull().sum()
```

```
0
```

```
df_copy['Loan_Amount_Term'].median()
```

```
360.0
```

```
df_copy['Loan_Amount_Term'].replace(np.NaN,360.0,inplace=True)
df_copy['Loan_Amount_Term'].isnull().sum()
```

```
0
```

Filling the null values of LoanAmount, Credit_History, Loan_Amount_Term with **median of the column** because **the median is less sensitive to extreme values.**

```
df_copy['Gender'].mode()
```

	Gender
0	Male

```
df_copy['Gender'].replace(np.NaN,'Male',inplace=True)
df_copy['Gender'].isnull().sum()
```

```
0
```

```
df_copy['Dependents'].mode()
```

```
0    0
Name: Dependents, dtype: object
```

```
df_copy['Dependents'].replace(np.NaN,'0',inplace=True)
df_copy['Dependents'].isnull().sum()
```

```
0
```

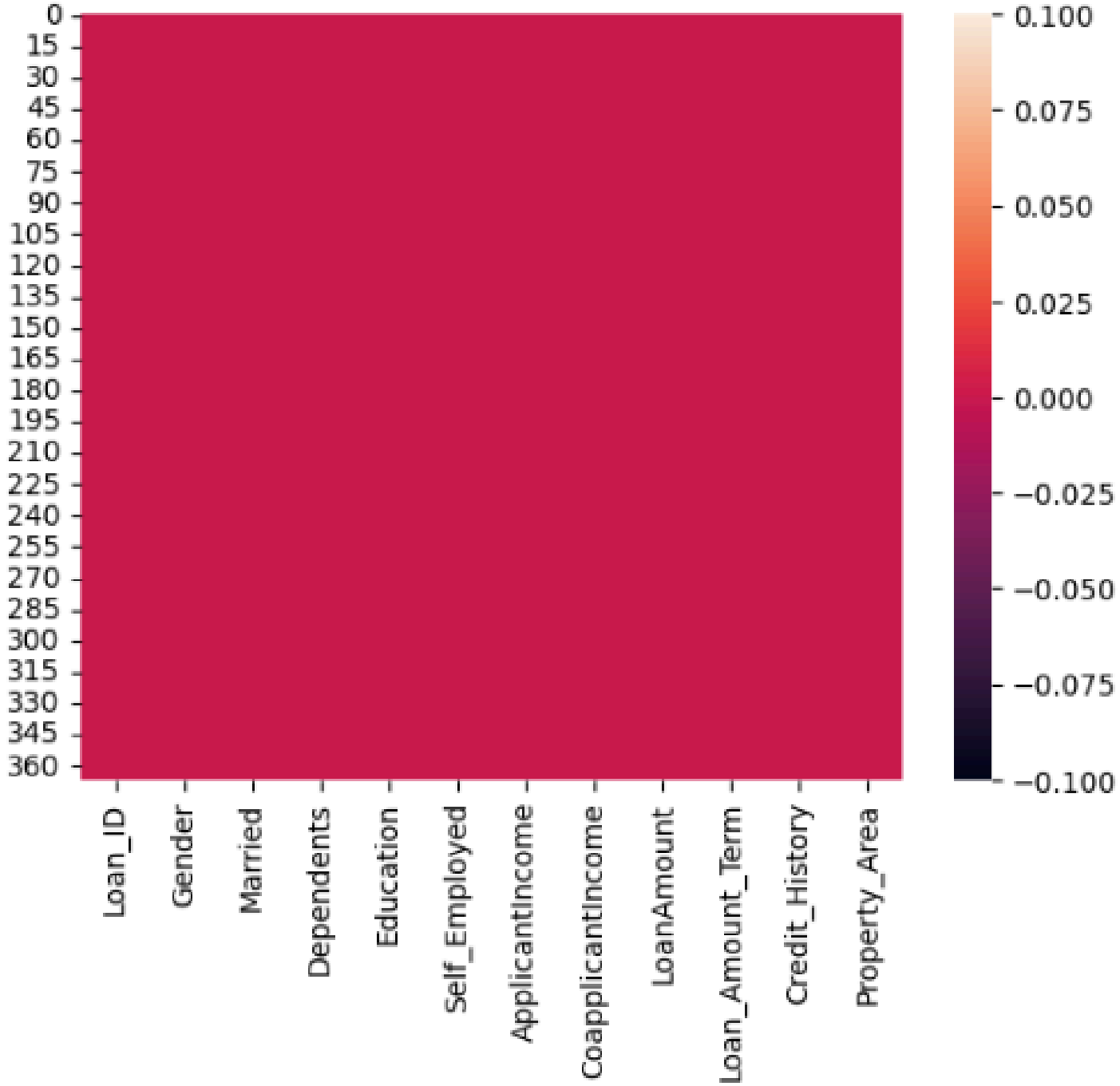
```
df_copy['Self_Employed'].mode()
```

```
0    No
Name: Self_Employed, dtype: object
```

```
df_copy['Self_Employed'].replace(np.NaN,'No',inplace=True)
df_copy['Self_Employed'].isnull().sum()
```

```
0
```

In all the above code snippets the null values were **filled with mode value of the column** because **it's a catagorical columns.**



This heatmap depicts the data with null values.

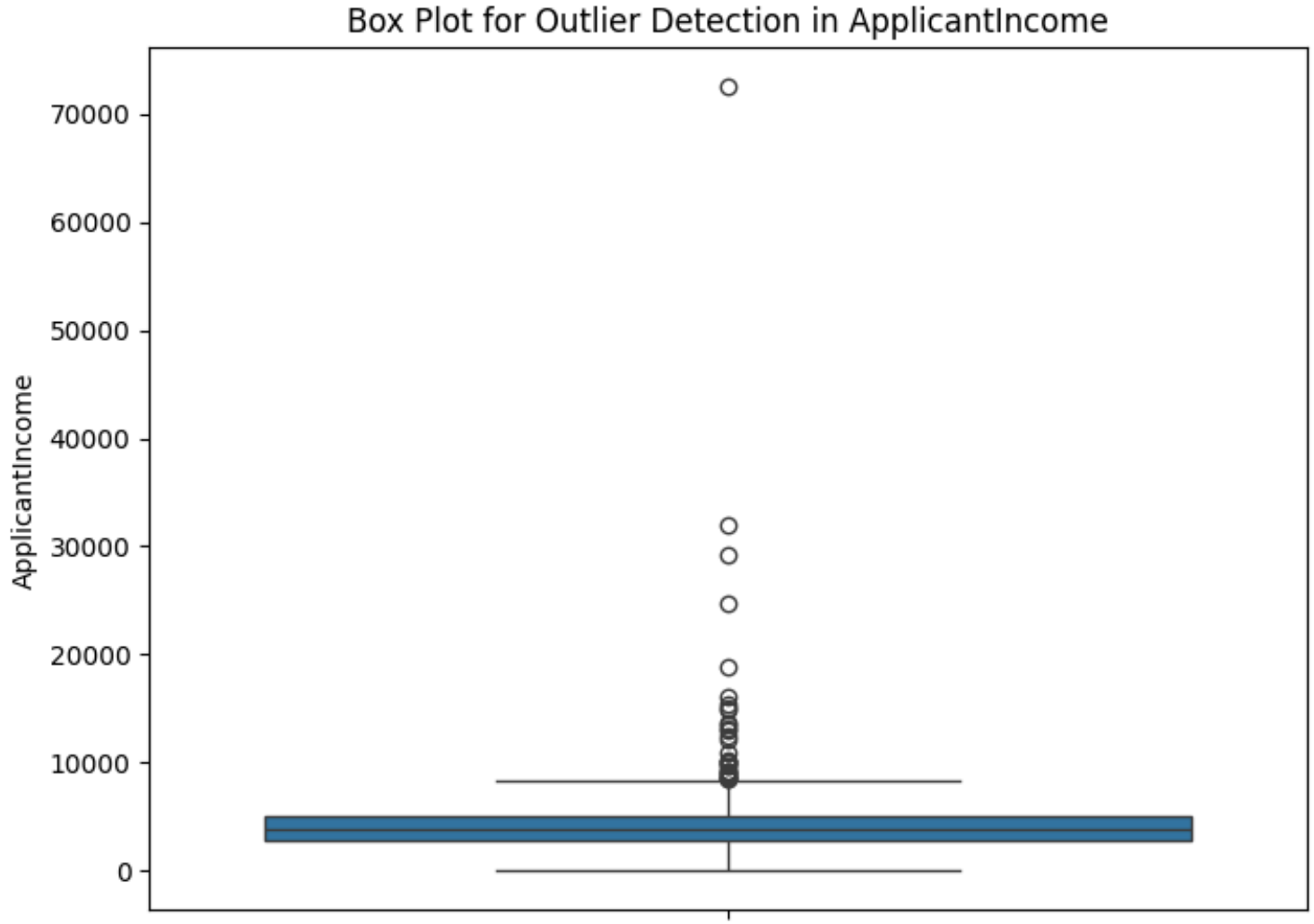
Handeling Outliers

Here i used **IQR method** for outliers detection and **box plot for visualization of outliers.**

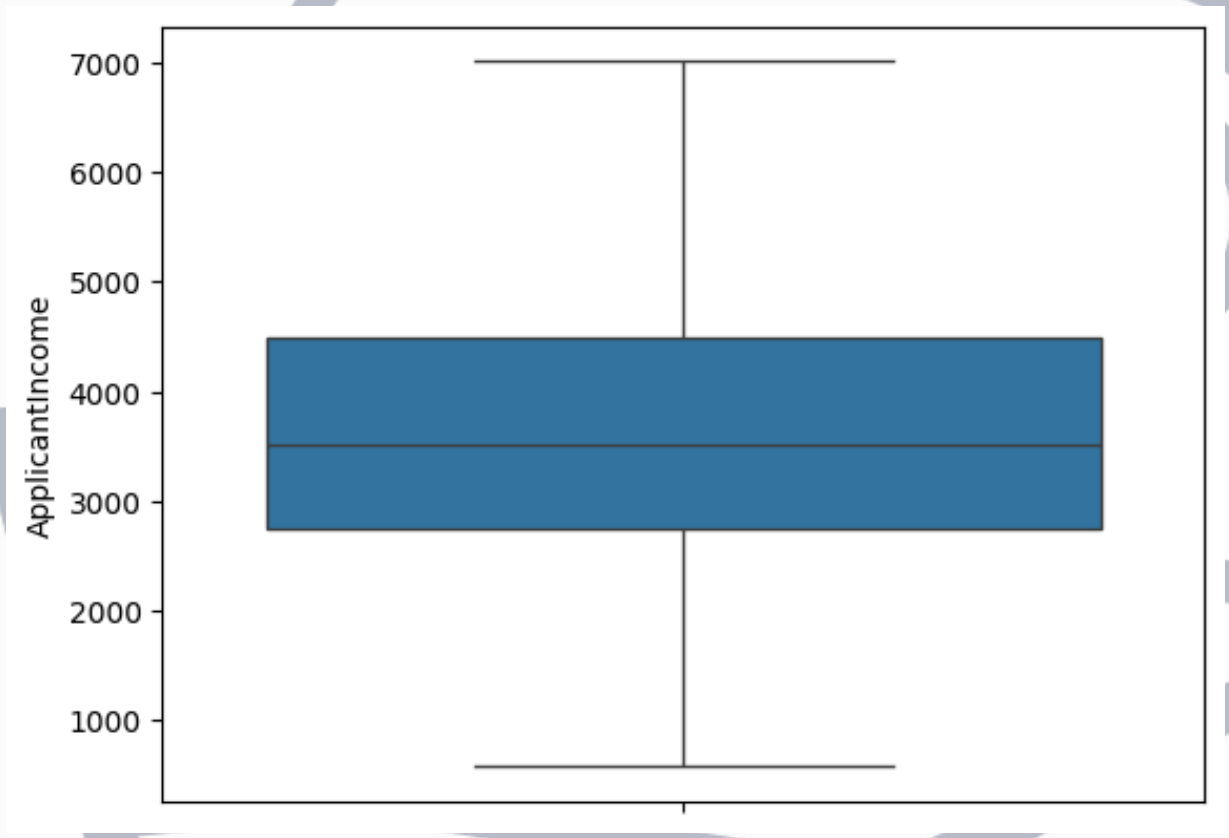
outliers in Applicantincome

```
Q1 = df_copy['ApplicantIncome'].quantile(0.25)
Q3 = df_copy['ApplicantIncome'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers_income = df_copy[(df_copy['ApplicantIncome'] < lower_bound) | (df_copy['ApplicantIncome'] > upper_bound)]
print('count of outliers in coapplicantincome :' ,len(outliers_income))
#print("Outliers in ApplicantIncome:\n", outliers_income)
```

```
count of outliers in coapplicantincome : 32
```



In the above plot, we can see several **points lying beyond the upper whisker**, indicating the presence of outliers in the **'ApplicantIncome' variable**. There are **32 outliers present** in this column and i have **filled it with median of the column. Boxplot after cleaning the outliers.**

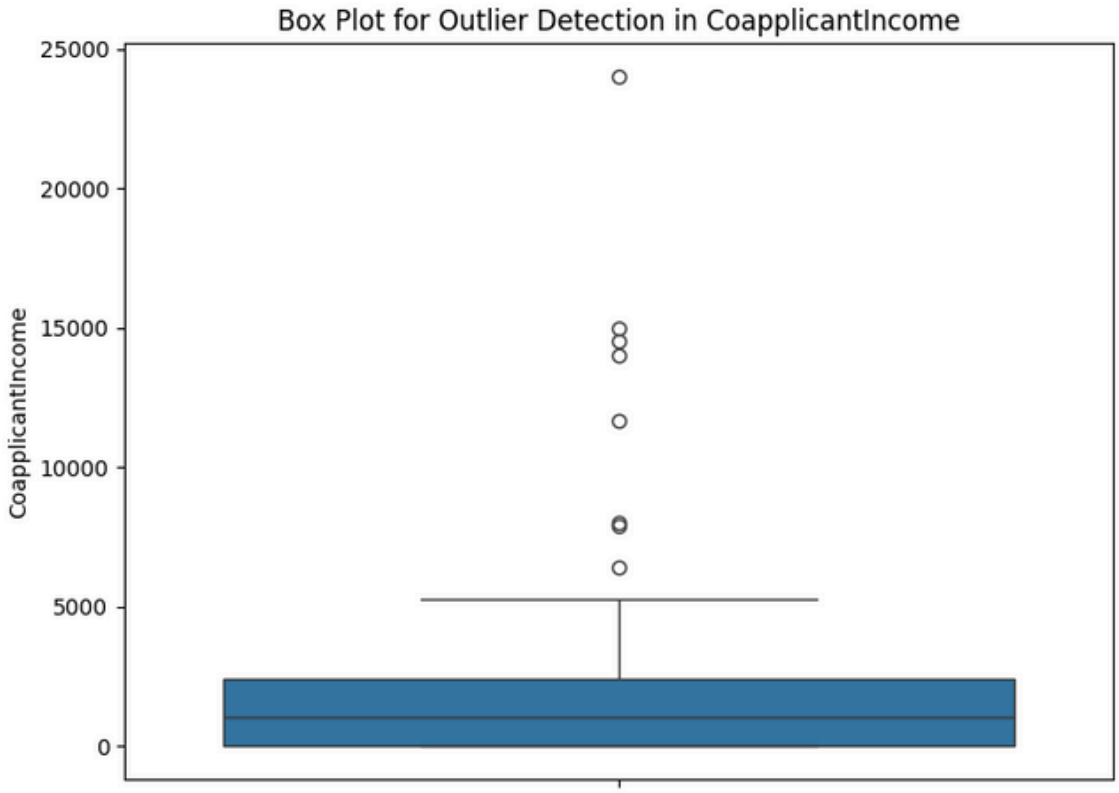


outliers in coapplicant

```
Q1 = df_copy['CoapplicantIncome'].quantile(0.25)
Q3 = df_copy['CoapplicantIncome'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers_coapplication = df_copy[(df_copy['CoapplicantIncome'] < lower_bound) | (df_copy['CoapplicantIncome'] > upper_bound)]
print('count of outliers in coapplicantincome :' ,len(outliers_coapplication))
#print("Outliers in CoapplicantIncome:\n", outliers_coapplication)
```

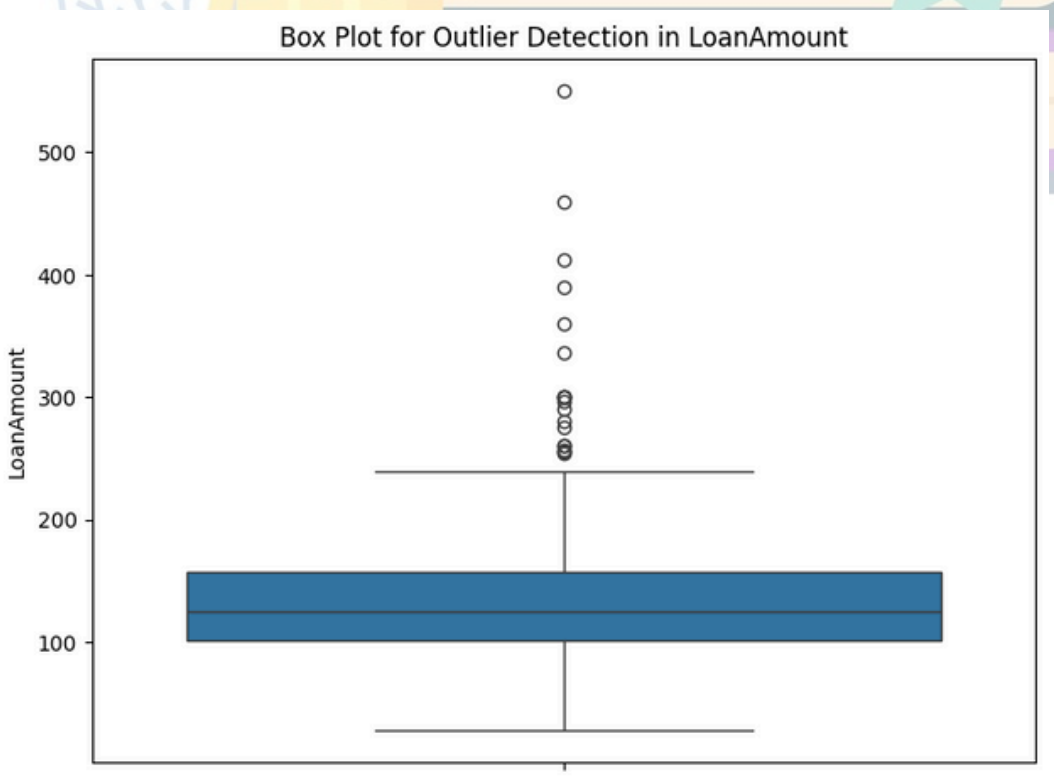
```
count of outliers in coapplicantincome : 8
```

This code indicates that There are 8 outlier in present in this column.



Visually representing the outliers of coapplicantincome attribute. Here also i used median to handle the ouliers beacuse **median is less sensitive to extreme values compared to mean**, providing a more robust central tendency measure in such cases.

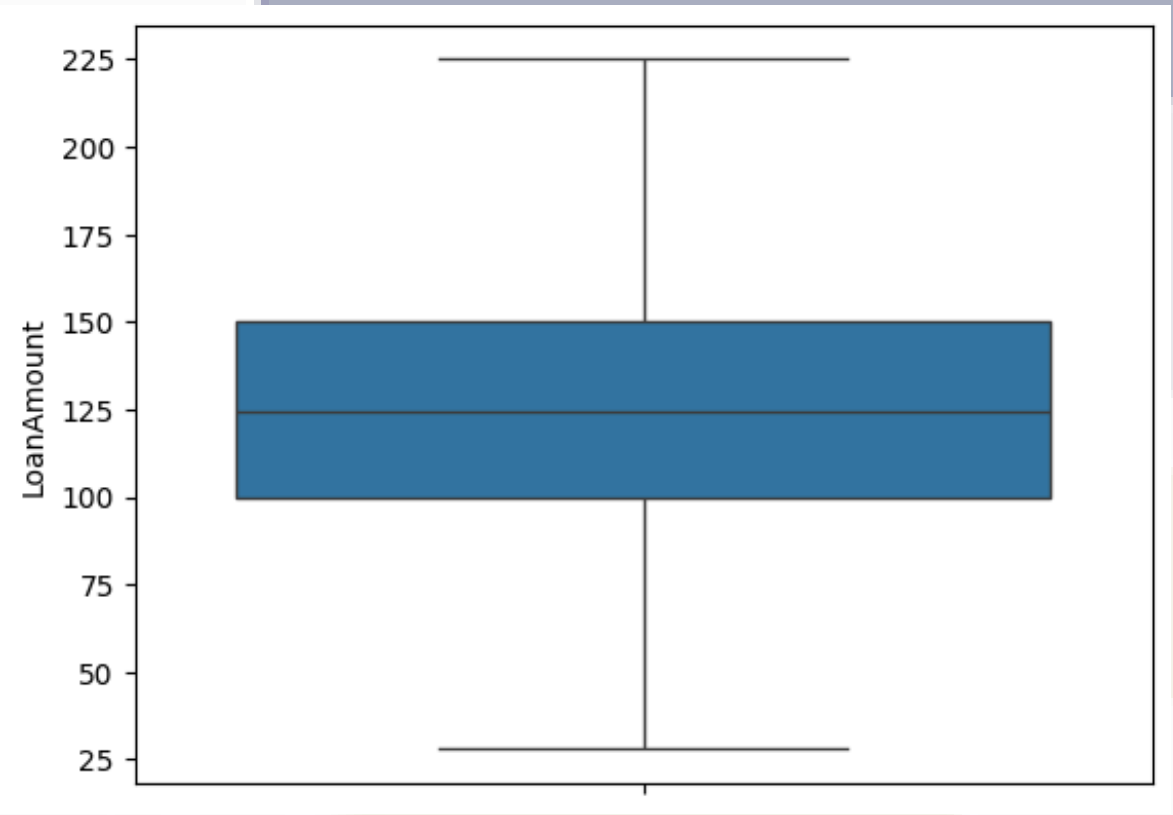
outliers in Loanamount



```
[ ] Q1 = df_copy['LoanAmount'].quantile(0.25)
    Q3 = df_copy['LoanAmount'].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers_loan_amount = df_copy[(df_copy['LoanAmount'] < lower_bound) | (df_copy['LoanAmount'] > upper_bound)]
    print('count of outliers in loan_amount :' ,len(outliers_loan_amount))
    #print("Outliers in LoanAmount:\n", outliers_loan_amount )

    count of outliers in loan_amount : 18
```

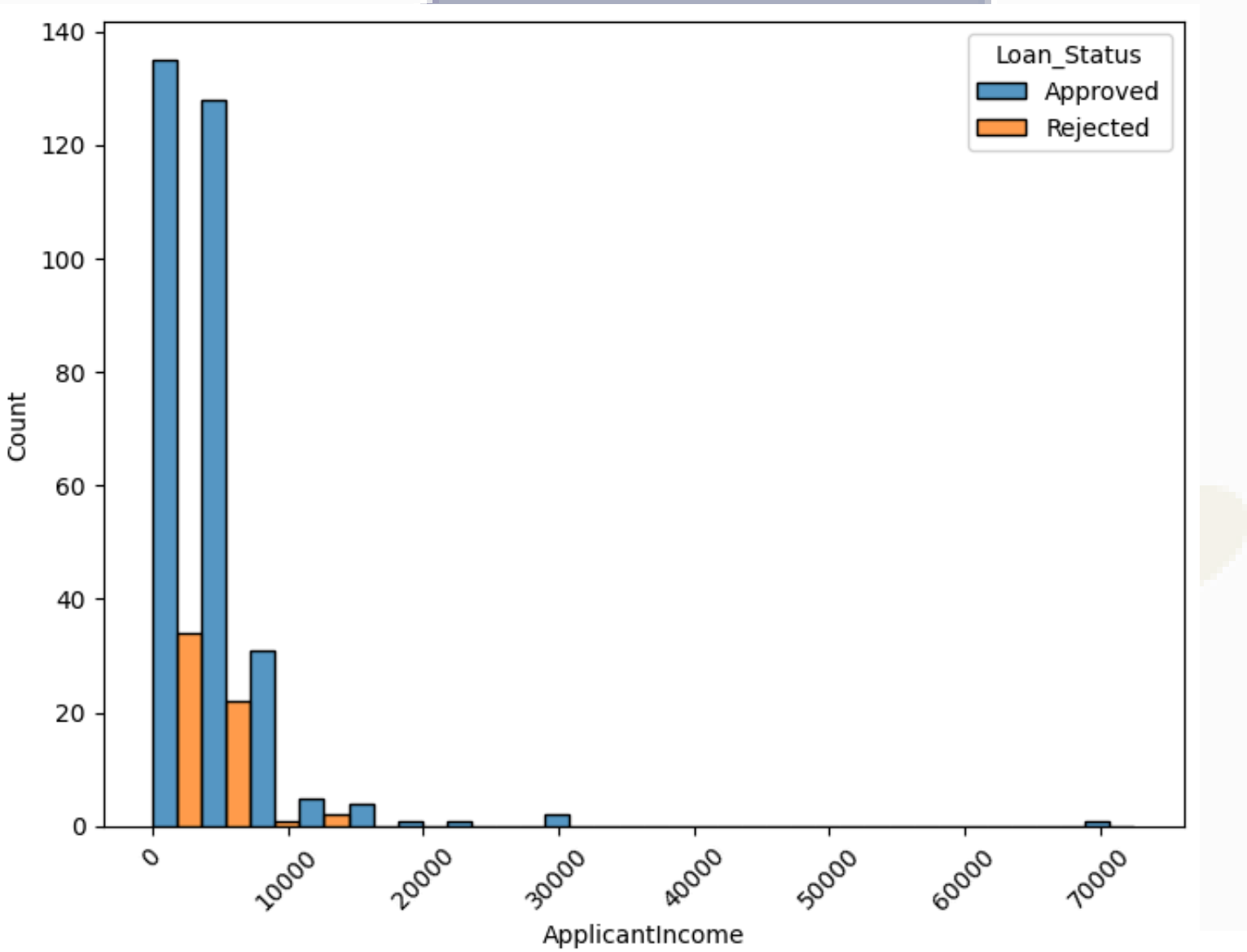
There are 18 outlier present in this attribute and it was imputed by median. The below graph represent the zero outliers in LoanAmount attribute.



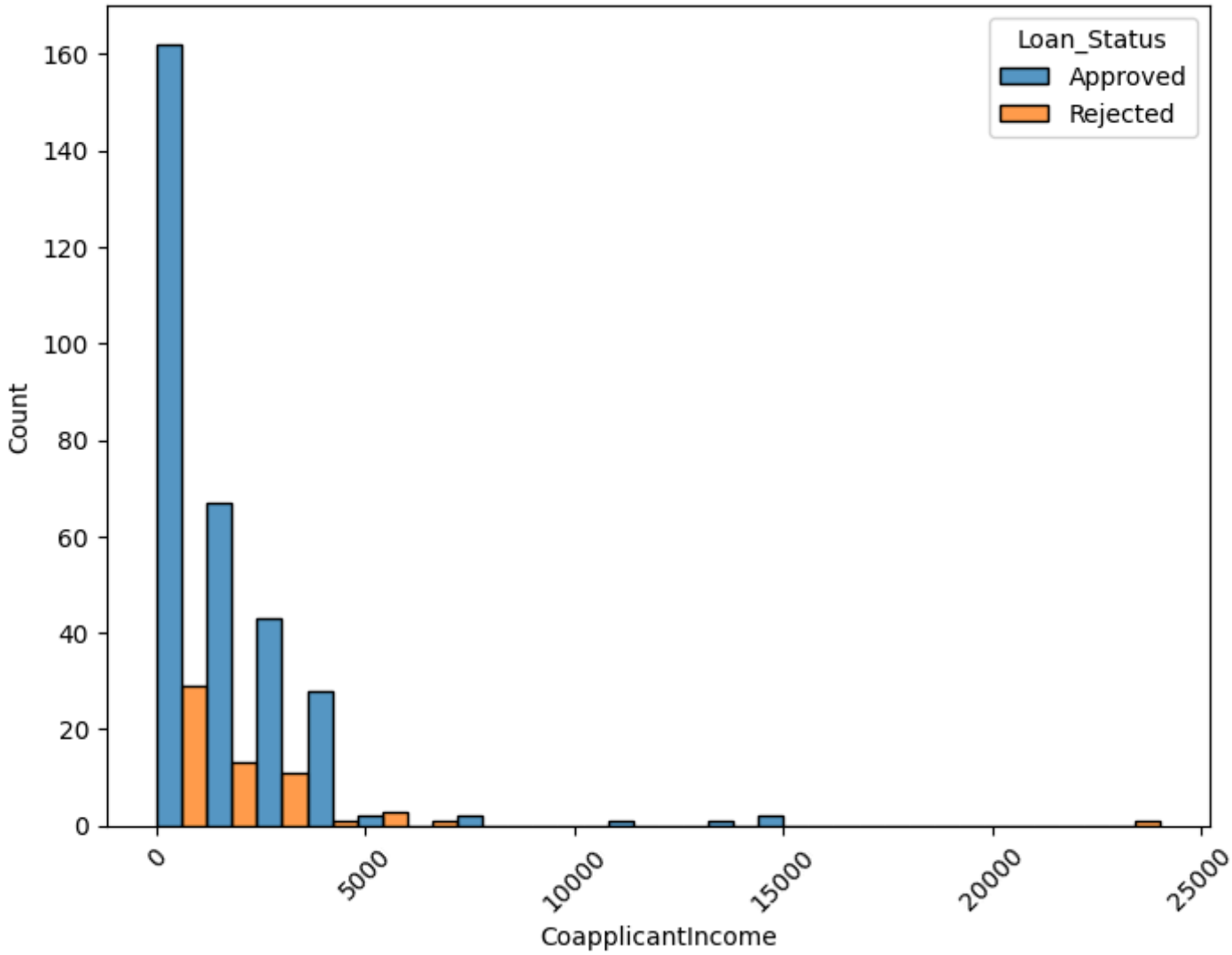
We don't use 'Loan_Amount_Term' and 'Credit_History' to find outliers because Loan_Amount_Term represents the duration of the loan in months. It's a categorical variable with a limited set of discrete values. Credit_History is binary (0 or 1), indicating good or bad credit history. Outlier detection methods are not relevant for binary variables.

Explore the distribution of numeric columns using the Histograms visualization.

In this data set there are 3 numeric column i.e 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount' .

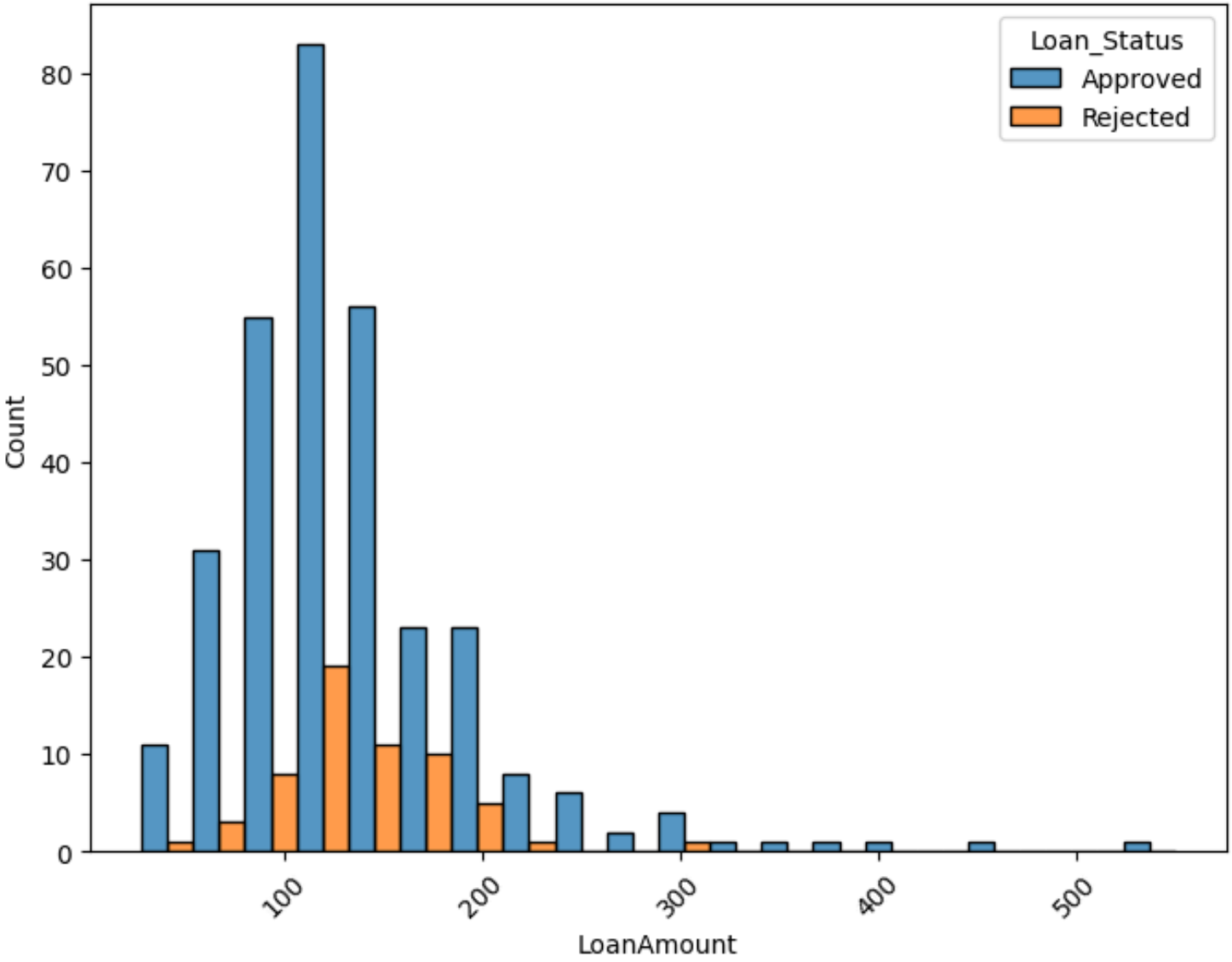


The graph indicates that **lower-income applicants apply more frequently and are more likely to be approved, whereas high-income applicants are less common and experience higher rejection rates.**



In the above graph the following:-

- Most coapplicants have incomes up to rs 5,000. Loans involving coapplicants with higher incomes are less common.
- Loans are more likely to be approved when the coapplicant's income is lower.
- As the coapplicant's income goes up, the number of approved loans decreases, and so does the number of rejected loans.

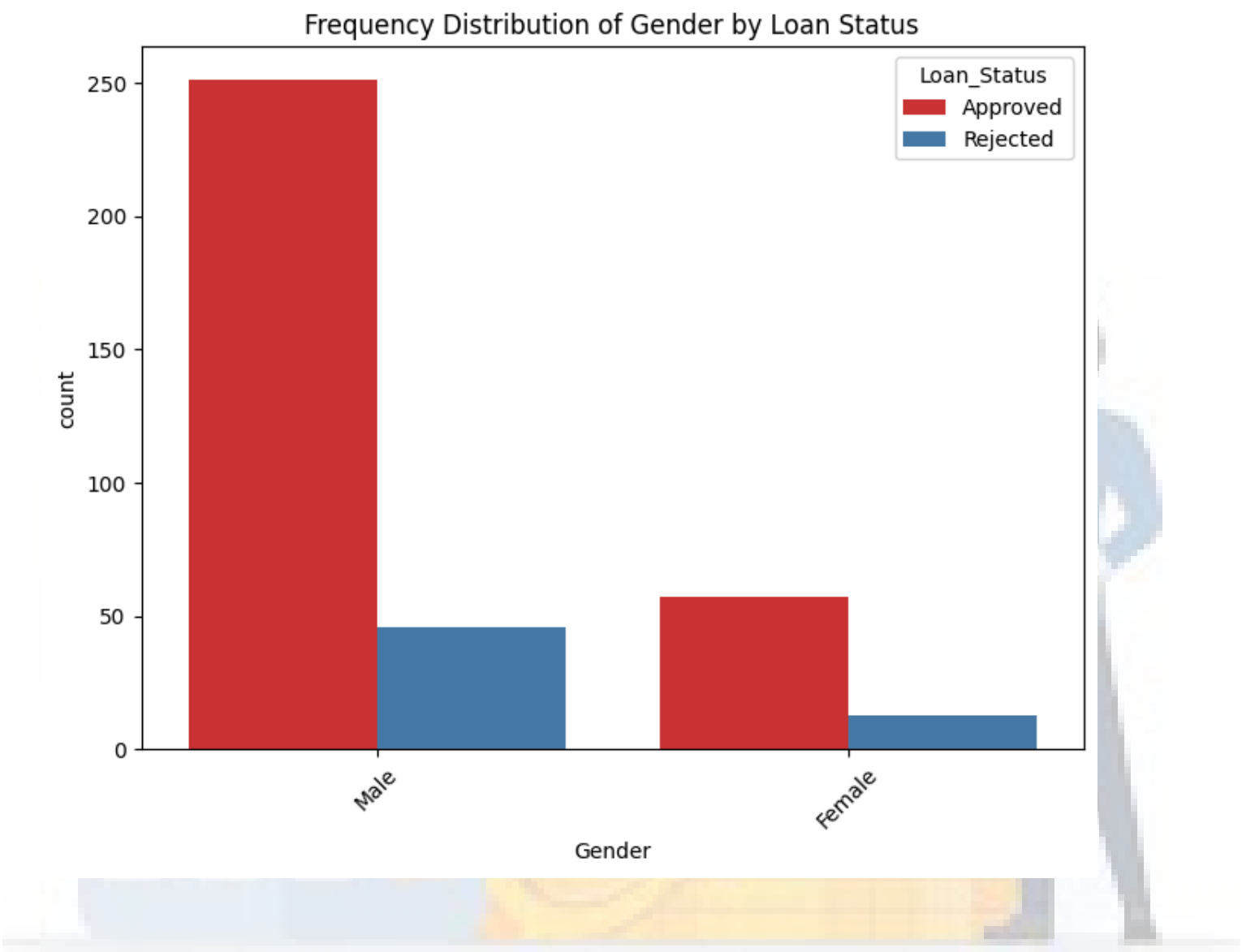


In the above graph the following:-

- Most loans are for amounts up to rs100. There are fewer loans for higher amounts.
- Approval Rates: Loans for smaller amounts are more likely to be approved.
- Trends: Fewer loans are approved or rejected as the loan amount goes up.

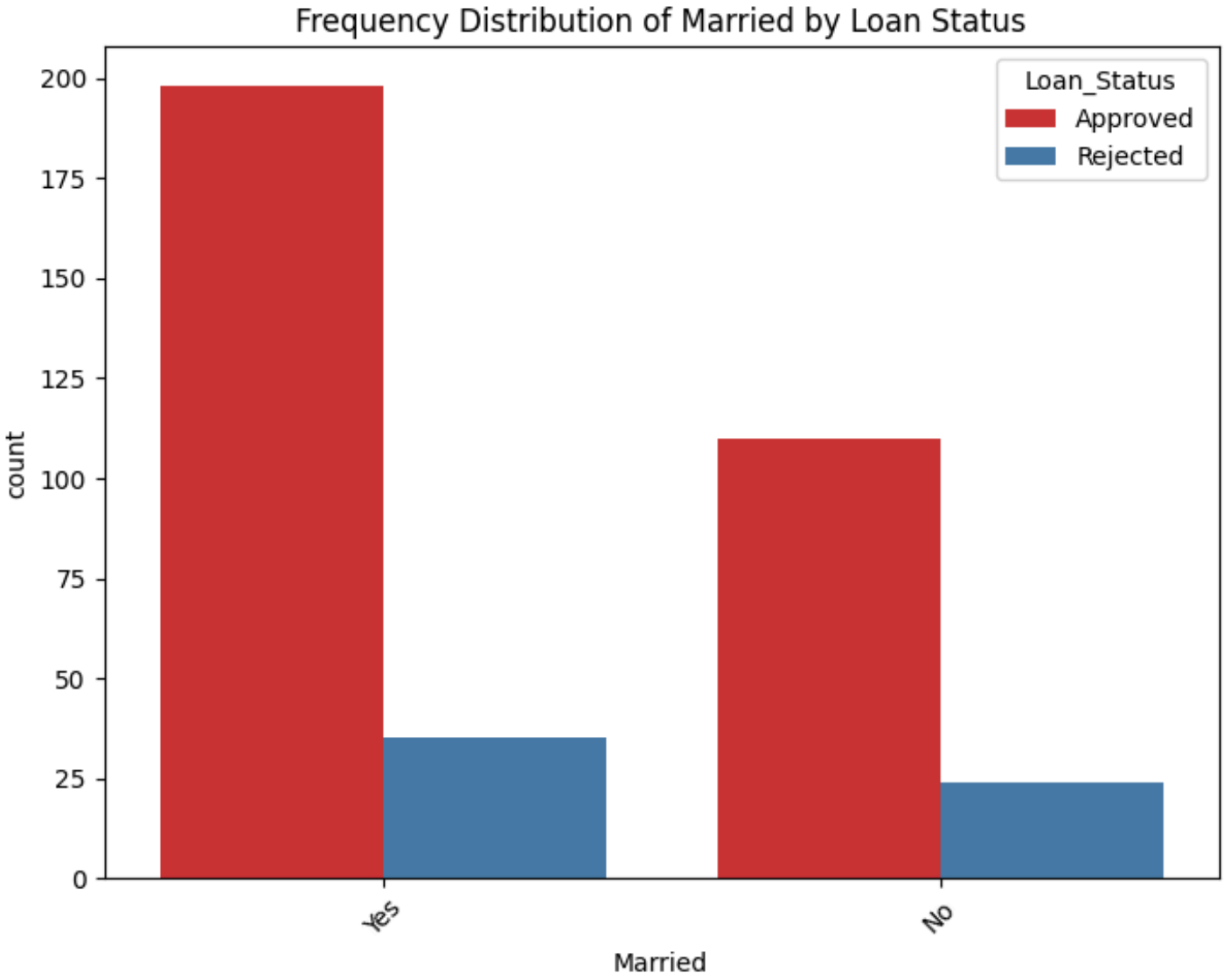
Visualize the frequency distribution of categorical variables.

In this data set there are 6 catagorical column i.e 'Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'Property_Area'.



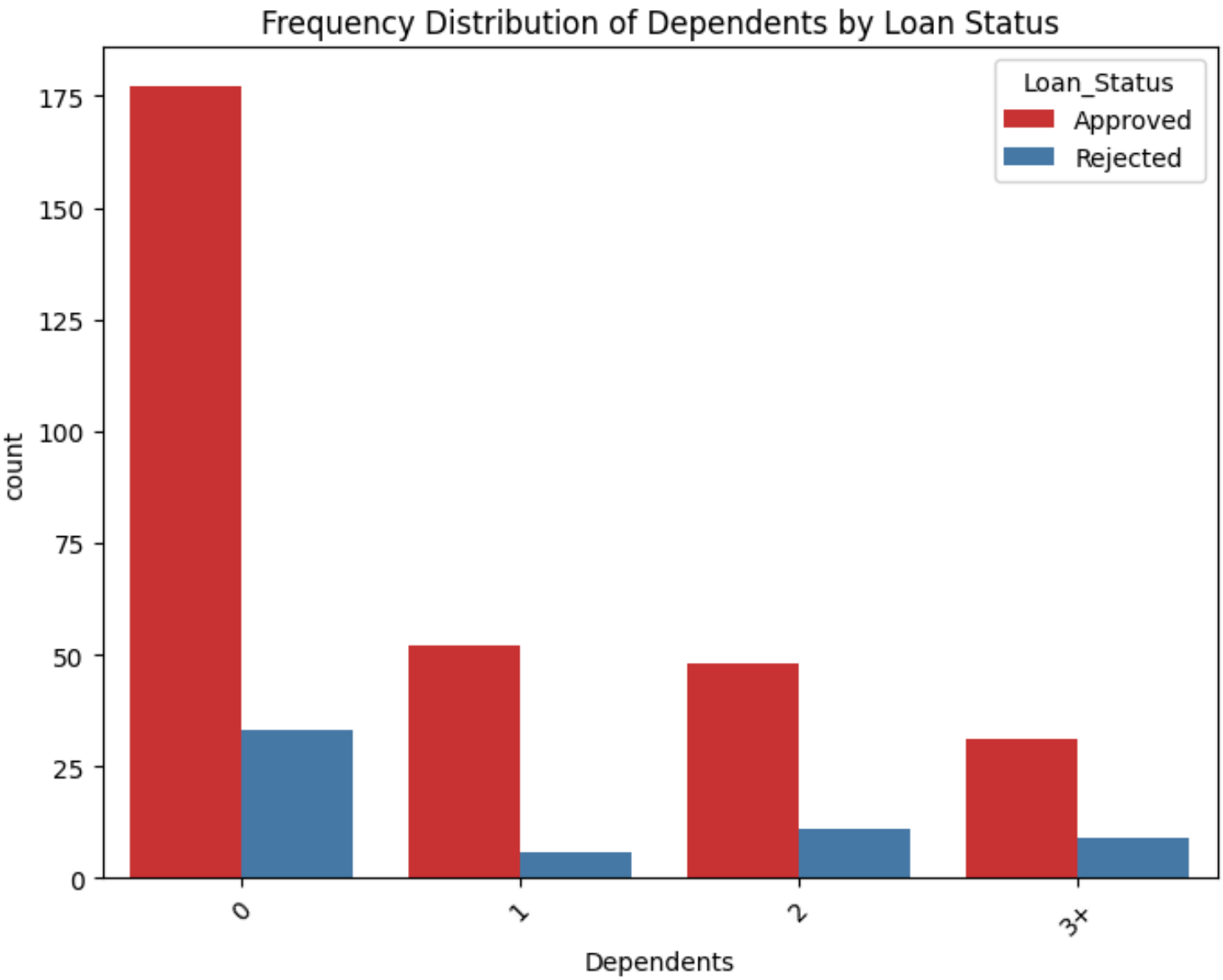
In the above graph the following:-

- More males apply for loans compared to females.
- Loan approval rates seem similar for both genders with a slightly higher approval rate for males.



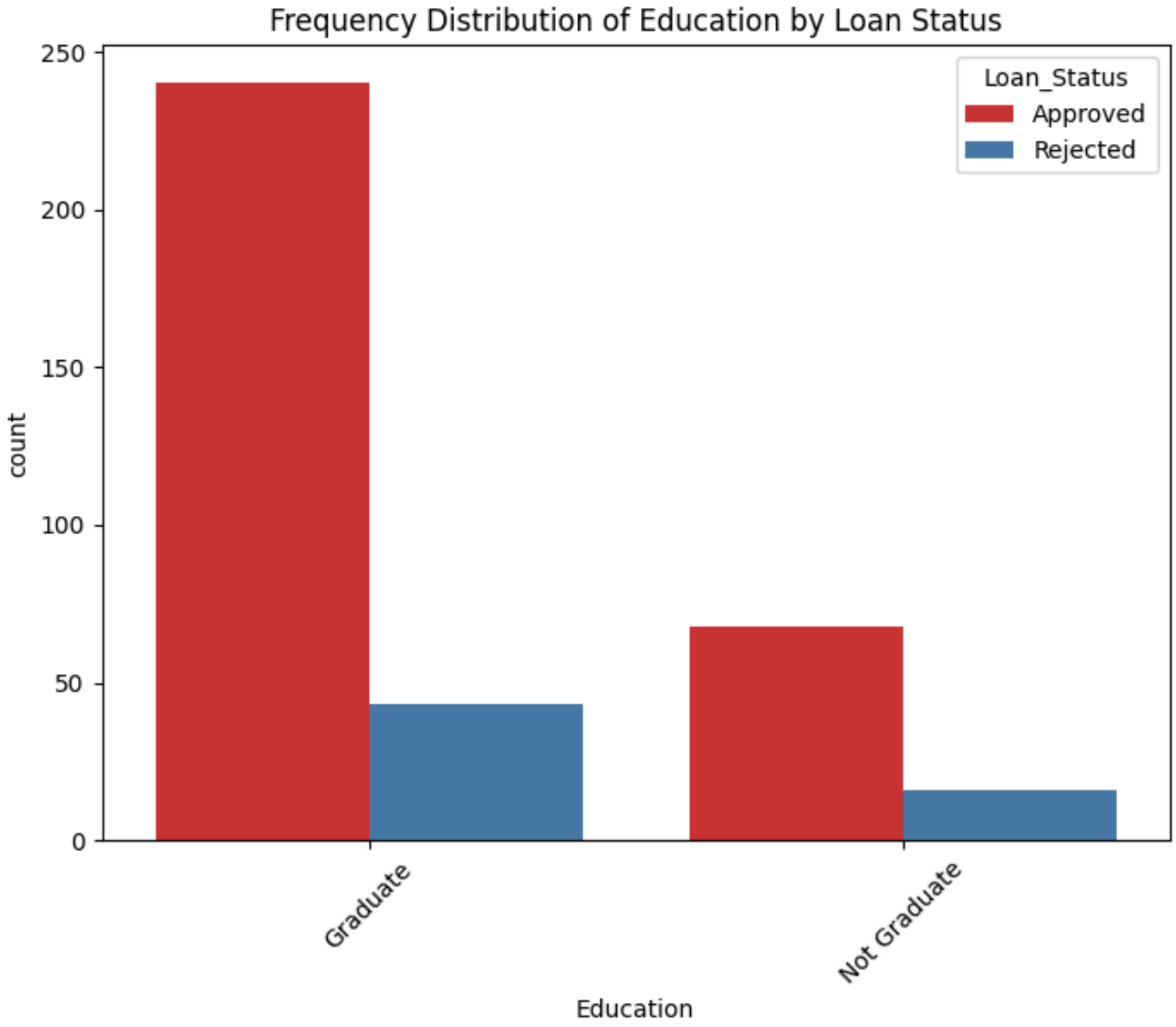
In the above graph the following:-

- Married individuals are more likely to apply for loans than unmarried individuals.
- Married individuals have a slightly higher loan approval rate compared to unmarried individuals.



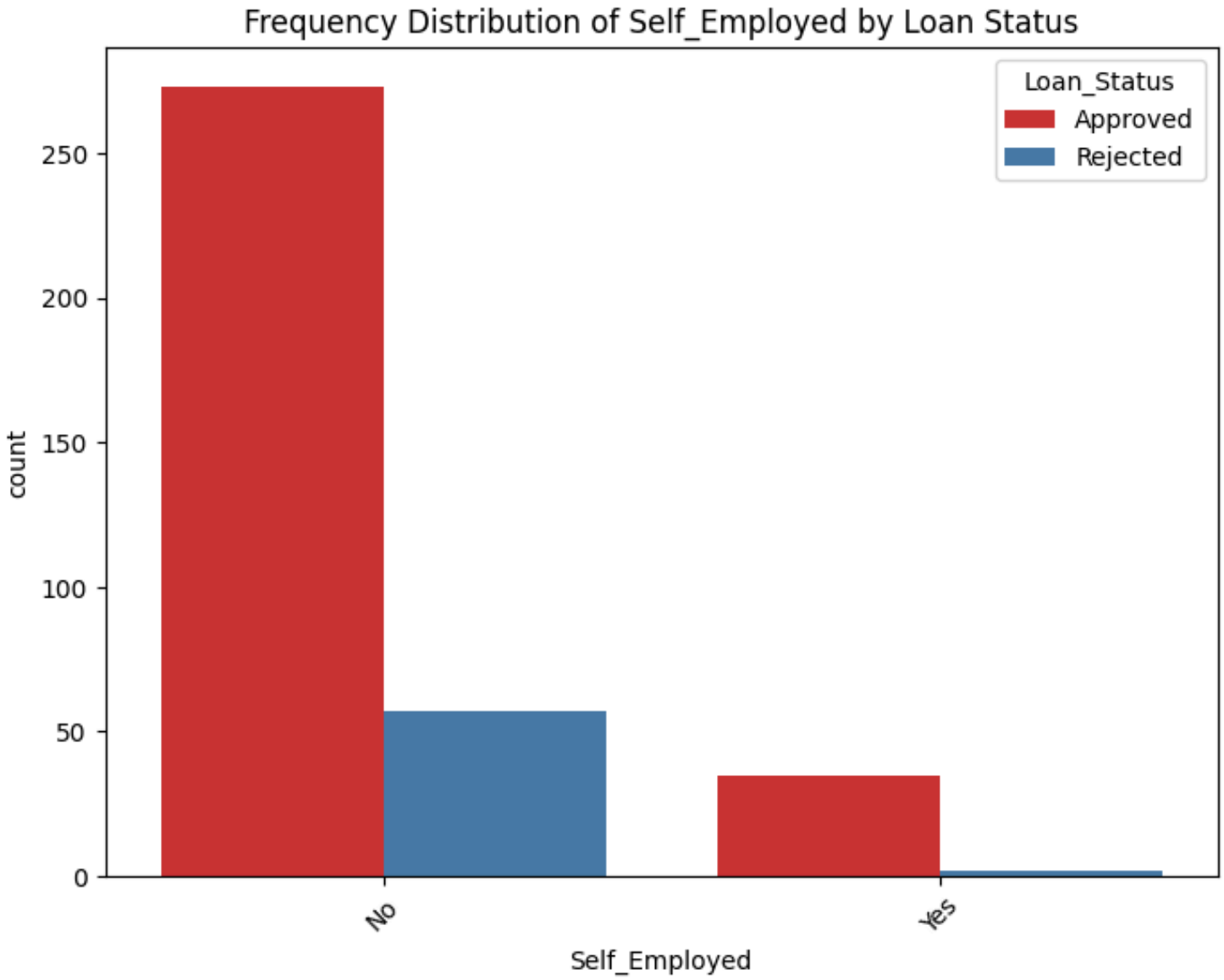
In the above graph the following:-

- Applicants with 0 dependents are the most common, followed by those with 1 or 2 dependents.
- Loan approval rates appear relatively similar across different numbers of dependents.



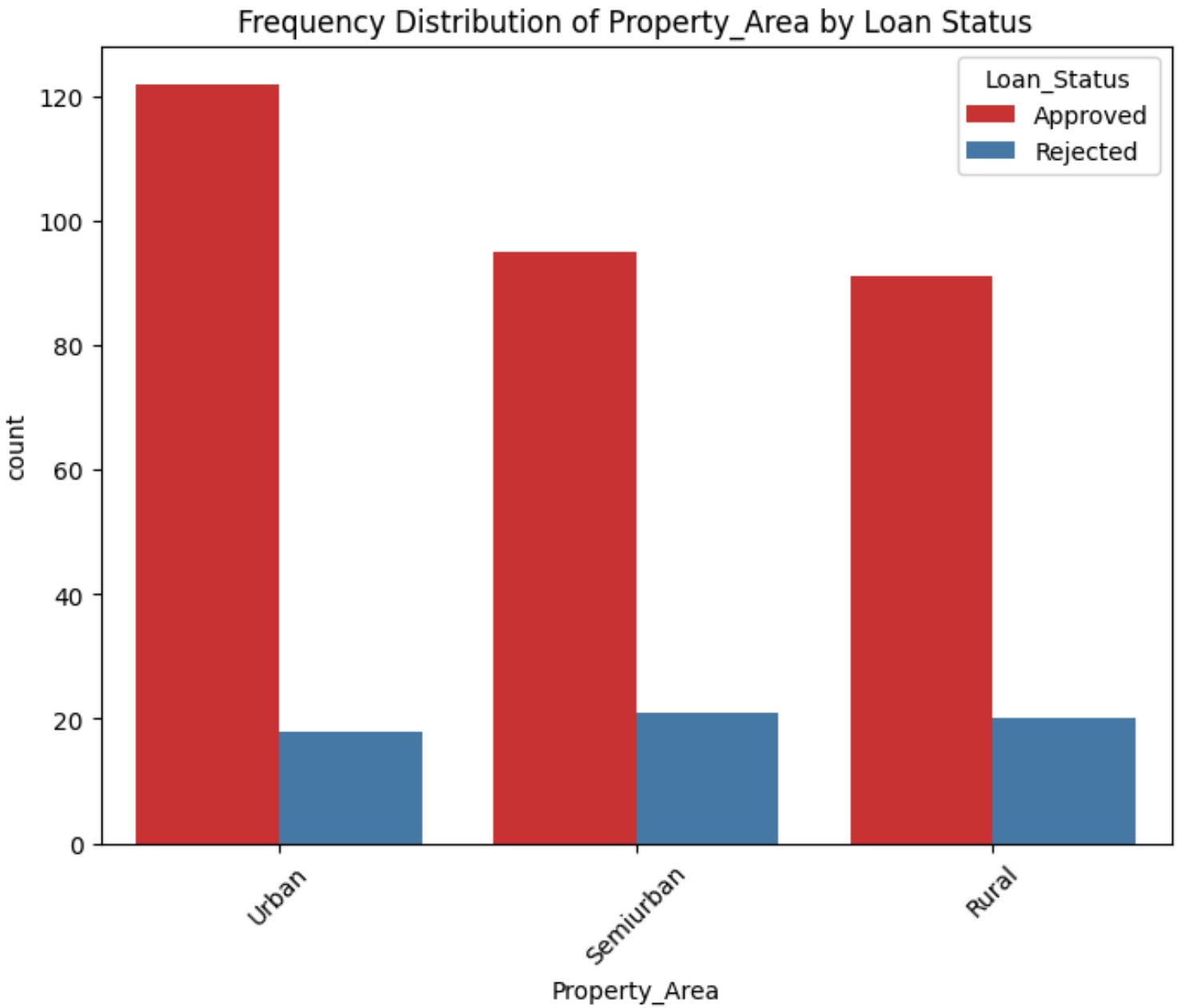
In the above graph the following:-

- Graduates are more likely to apply for loans compared to non-graduates.
- Graduates have a higher loan approval rate compared to non-graduates.



In the above graph the following:-

- The majority of loan applicants are not self-employed.
- Loan approval rates appear similar for both self-employed and non-self-employed individuals.

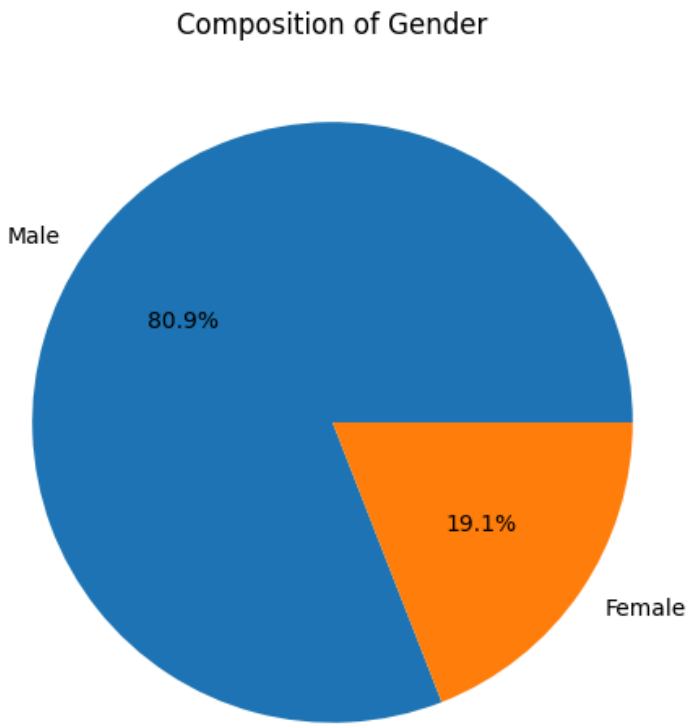


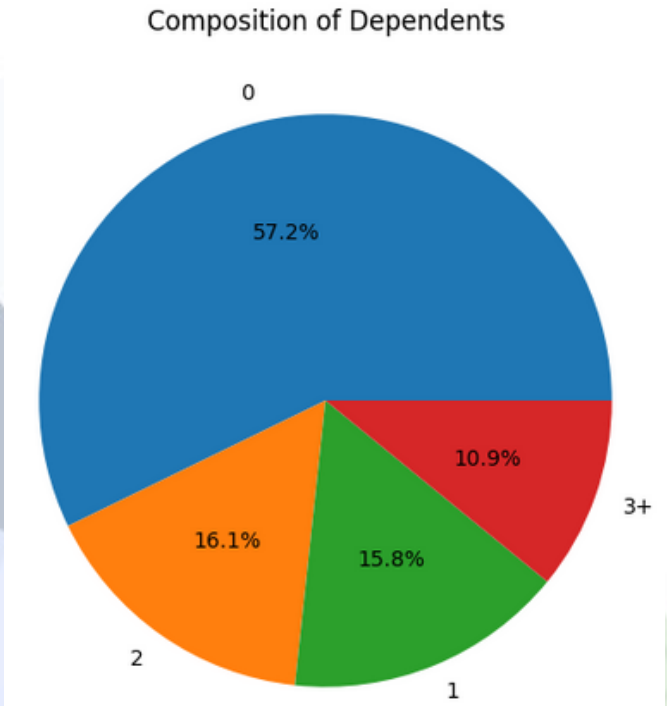
In the above graph the following:-

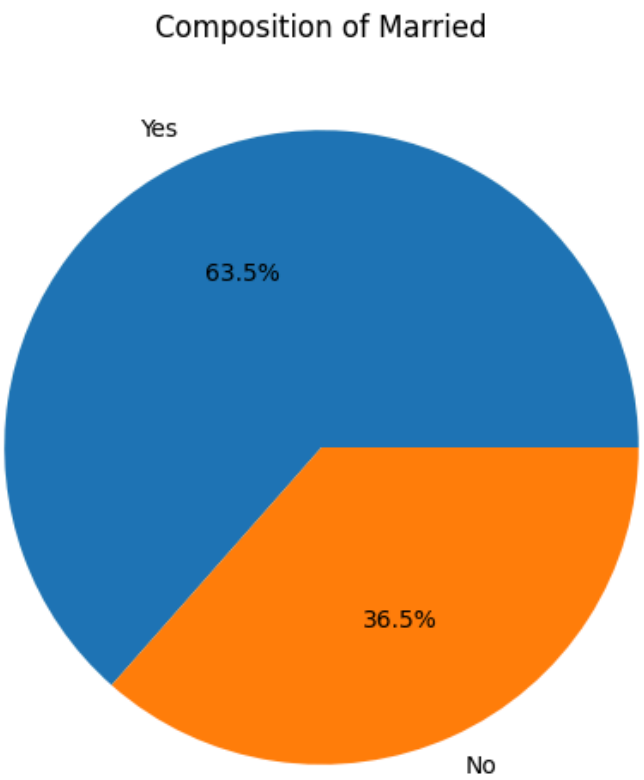
- Most loan applications come from semi-urban areas, followed by urban and rural areas.
- Loan approval rates are highest in semi-urban areas, followed by urban and then rural areas.

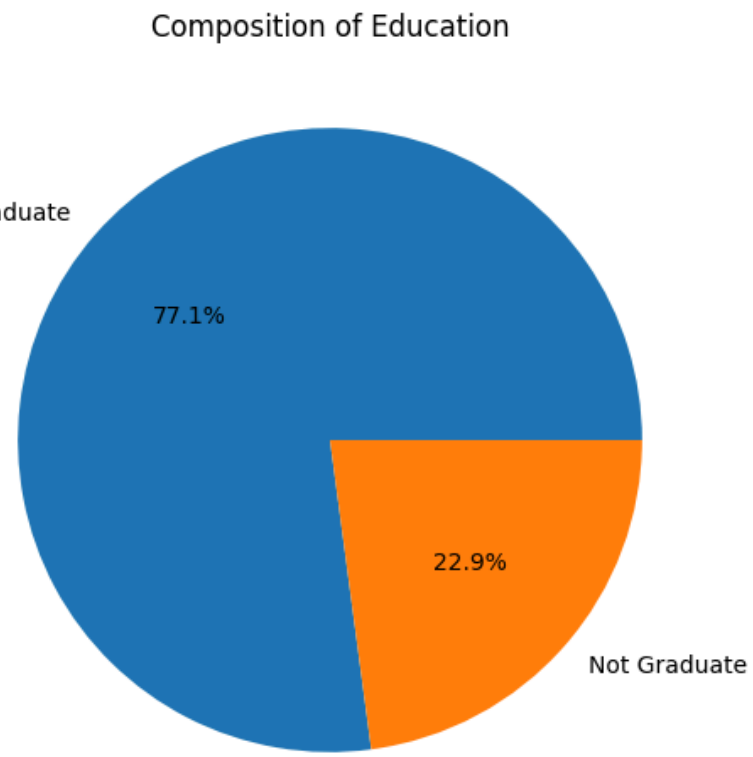
Pie Charts: Represent the composition of categorical variables

The pie charts provide a visual representation of the proportion of each category within a variable.

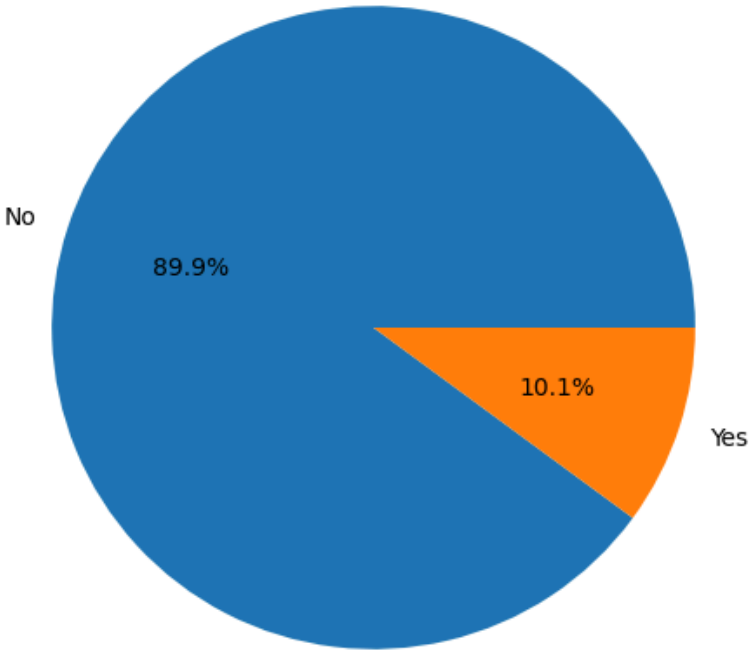




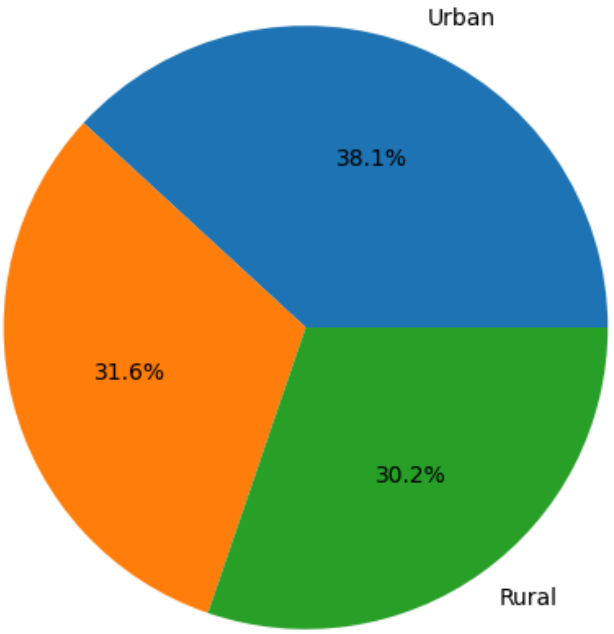




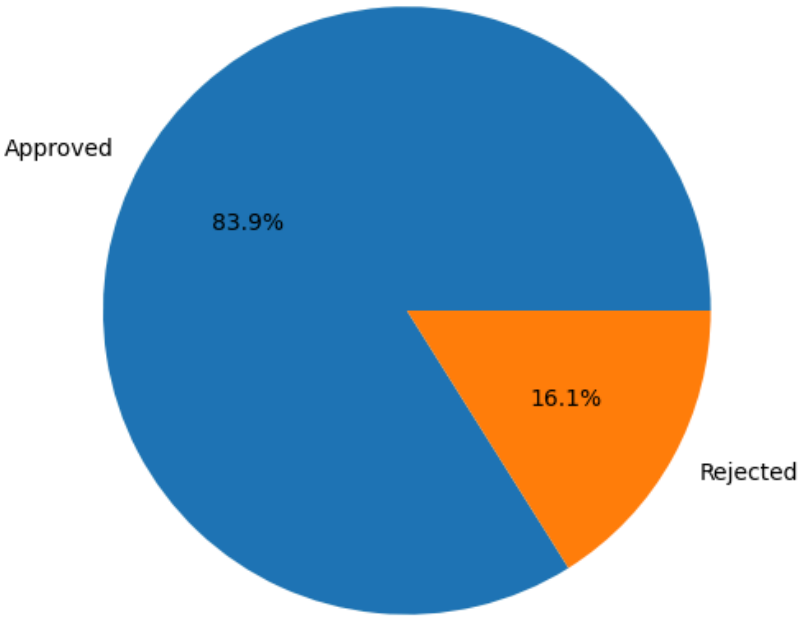
Composition of Self_Employed



Composition of Property_Area



Composition of Loan_Status



Perform a correlation analysis to identify relationships between numeric variables. Visualize correlations using a heatmap

Correlation: It ranges from -1 to +1

- +1 --> Positive Correlation
- -1 --> Negative Correlation
- ~0 --> No Correlation

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
ApplicantIncome	1.000000	-0.110335	0.488737	0.023319	0.094944
CoapplicantIncome	-0.110335	1.000000	0.150034	-0.008633	-0.058004
LoanAmount	0.488737	0.150034	1.000000	0.091867	-0.012932
Loan_Amount_Term	0.023319	-0.008633	0.091867	1.000000	-0.048189
Credit_History	0.094944	-0.058004	-0.012932	-0.048189	1.000000

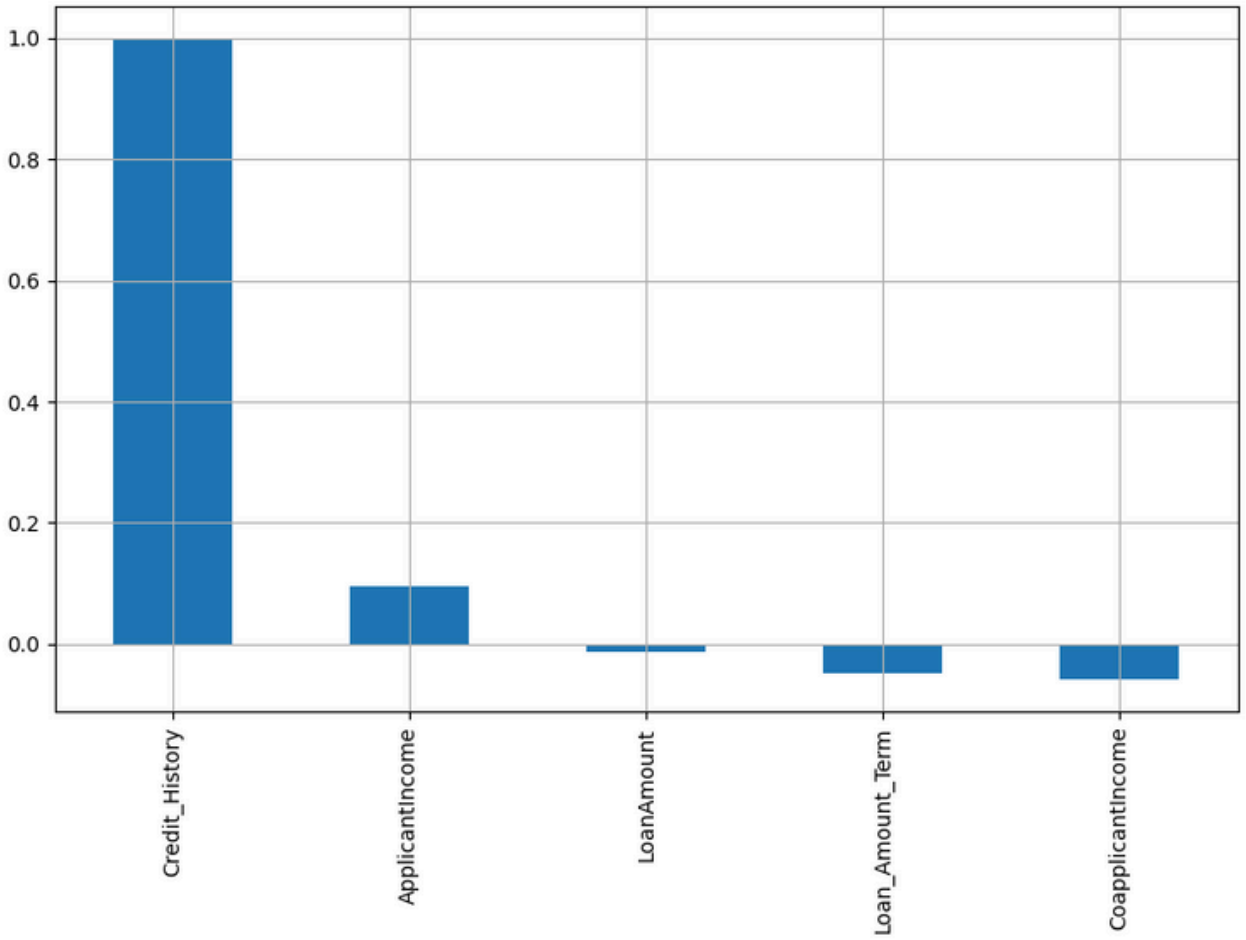
Positive Correlations:

- **ApplicantIncome & LoanAmount:** A positive correlation of 0.57 suggests that as the applicant's income increases, the loan amount they apply for also tends to increase. This is intuitive as higher income might imply a greater ability to repay a larger loan.
- **CoapplicantIncome & LoanAmount:** A positive correlation of 0.19 indicates a weaker but still positive relationship between coapplicant income and loan amount. This suggests that higher coapplicant income might also contribute to a slightly larger loan application.

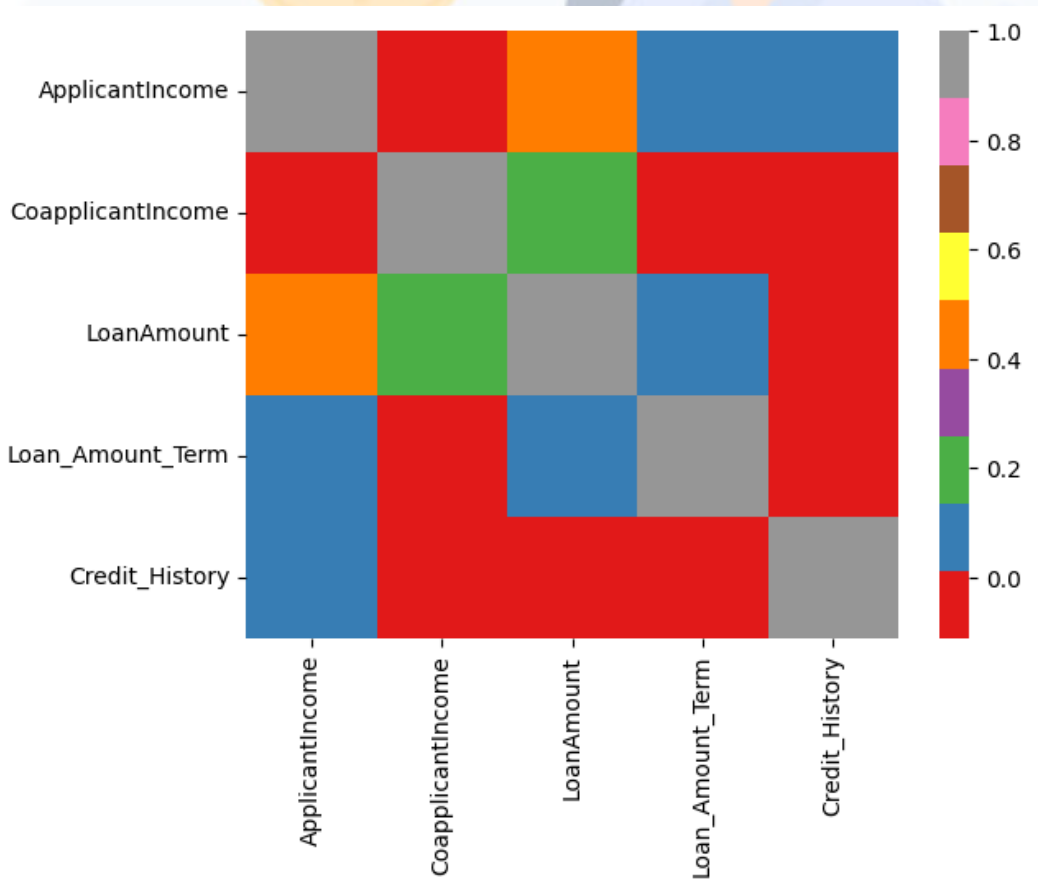
Negative Correlations:

- **Loan_Amount_Term & Credit_History:** A negative correlation of -0.002 suggests a very weak negative relationship between loan term and credit history. This is close to zero, indicating practically no linear association between these two variables. It might mean that the length of the loan term doesn't have a significant impact on whether a person has good or bad credit history.

Representing of correlation using histogram and heatmap



Positive correlations are shown in red, while negative correlations are shown in blue.



Thank you!