EXPLORATORY DATA ANALYSIS OF LOAN APPROVAL DATASET

By Lucky Bisht

INTRODUCTION

A loan is a financial arrangement in which a lender provides money to a borrower, who agrees to repay the borrowed amount, called the principal, along with interest within a specified timeframe. The interest serves as compensation to the lender for taking on the risk and for the opportunity cost of lending the money. Loan agreements typically detail the repayment schedule, interest rate, and any applicable fees.
Loans may be secured, requiring the borrower to provide collateral as a guarantee, or unsecured, which depends on the borrower's credit history and reliability. Common examples include personal loans, mortgages, and car loans. Failure to meet the agreed repayment terms can result in penalties, harm to the borrower's credit score, or legal action.

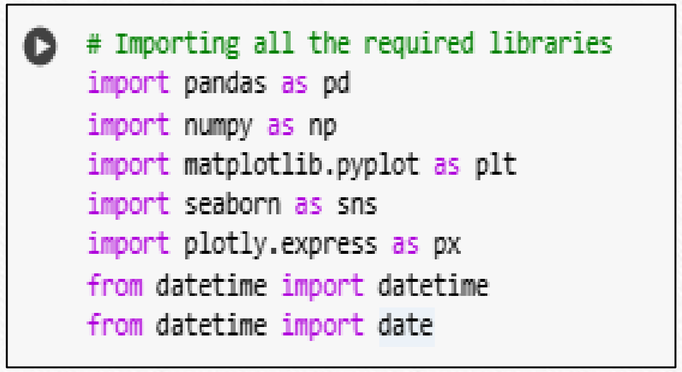
❖ About the DATASET

I worked with a dataset containing 367 unique loan applications, a widely used resource for practicing exploratory data analysis (EDA). This dataset offers comprehensive information about loan applications, making it an excellent tool for examining approval trends, understanding borrower profiles, and identifying the critical factors that impact loan approval decisions.

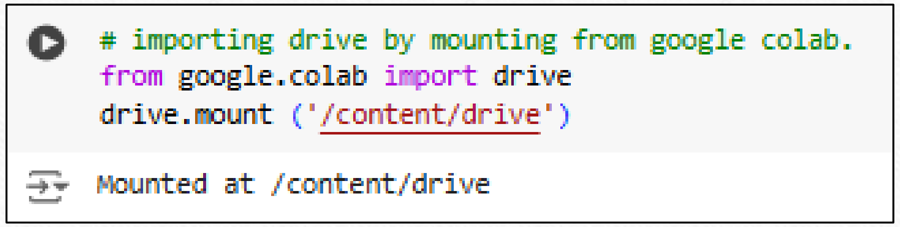
❖ Goal of the Project

The primary objective of this loan approval analysis project is to thoroughly analyze loan application data. This includes identifying and correcting any data inconsistencies, summarizing important statistics, and visualizing key insights such as approval rates and borrower demographics. The project focuses on understanding loan status, uncovering the factors influencing loan approvals, and evaluating associated risks. Ultimately, the aim is to provide actionable insights and recommendations to streamline the loan approval process and improve risk management strategies.

❖ Loading the Dataset



```
# Importing all the required libraries
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
```

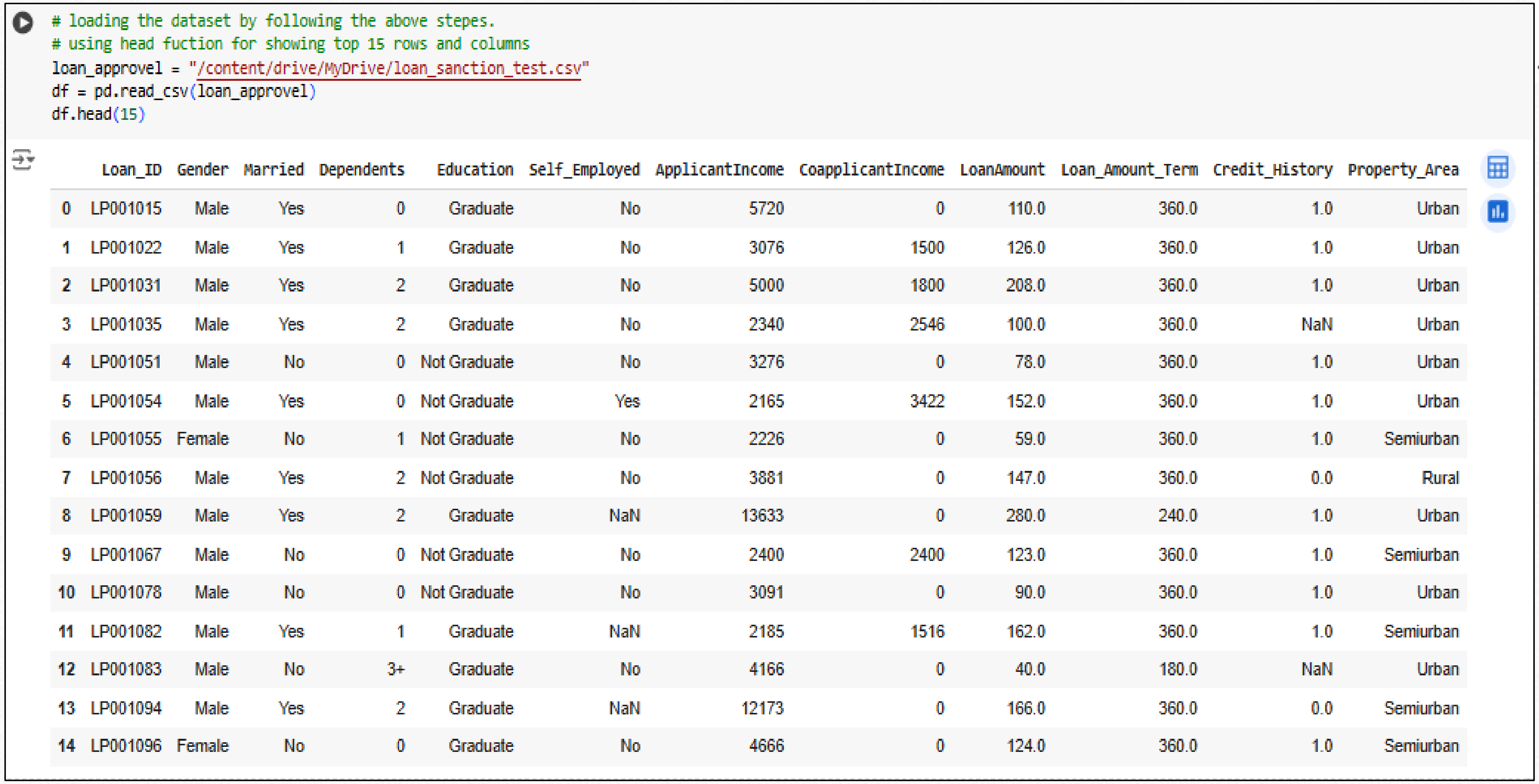


```
# importing drive by mounting from google colab.
from google.colab import drive
drive.mount ('/content/drive')
```

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

- ✓ I carried out my analysis using Google Colab Notebook.
- ✓ The dataset was imported directly from Google Drive.
- ✓ To begin the Exploratory Data Analysis (EDA), the dataset was assigned the name ‘df’.
- ✓ It consists of 367 rows and 12 columns.
- ✓ For data cleaning, I utilized popular Python libraries such as NumPy, Pandas, Matplotlib and Seaborn.
- ✓ Duplicate entries identified in the dataset were removed to ensure data accuracy.

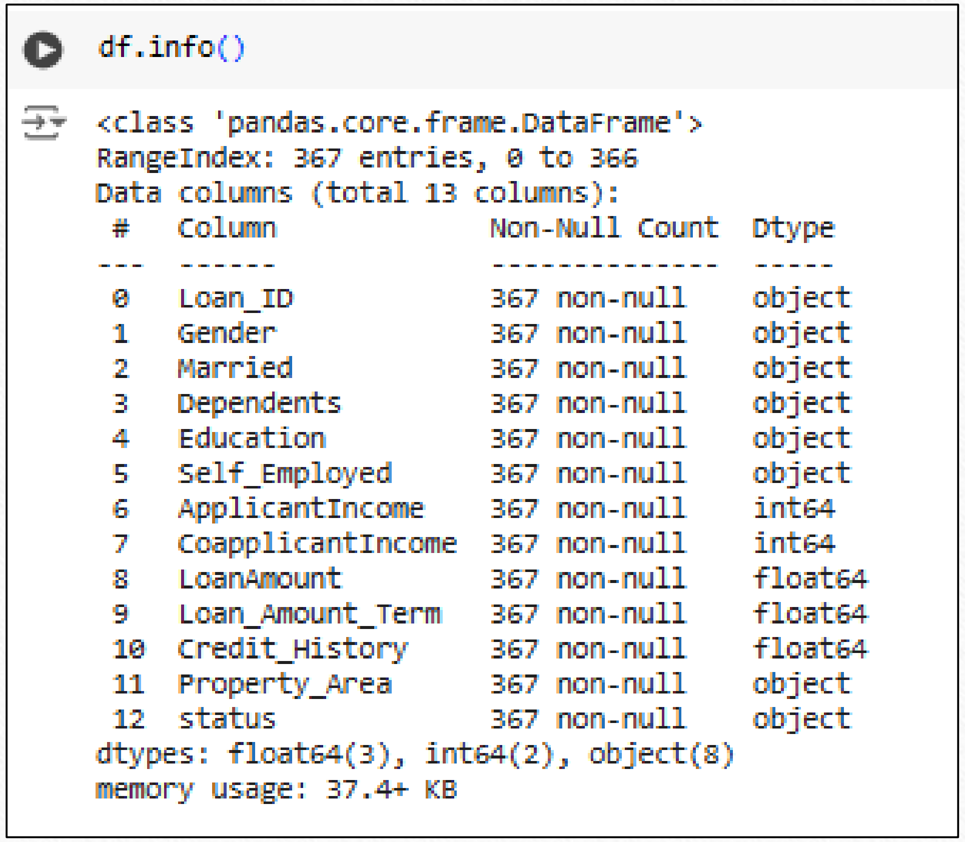
❖ Using Head function.



```python
# loading the dataset by following the above stepes.
# using head fuction for showing top 15 rows and columns
loan_approvel = "/content/drive/MyDrive/loan_sanction_test.csv"
df = pd.read_csv(loan_approvel)
df.head(15)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban
5	LP001054	Male	Yes	0	Not Graduate	Yes	2165	3422	152.0	360.0	1.0	Urban
6	LP001055	Female	No	1	Not Graduate	No	2226	0	59.0	360.0	1.0	Semiurban
7	LP001056	Male	Yes	2	Not Graduate	No	3881	0	147.0	360.0	0.0	Rural
8	LP001059	Male	Yes	2	Graduate	NaN	13633	0	280.0	240.0	1.0	Urban
9	LP001067	Male	No	0	Not Graduate	No	2400	2400	123.0	360.0	1.0	Semiurban
10	LP001078	Male	No	0	Not Graduate	No	3091	0	90.0	360.0	1.0	Urban
11	LP001082	Male	Yes	1	Graduate	NaN	2185	1516	162.0	360.0	1.0	Semiurban
12	LP001083	Male	No	3+	Graduate	No	4166	0	40.0	180.0	NaN	Urban
13	LP001094	Male	Yes	2	Graduate	NaN	12173	0	166.0	360.0	0.0	Semiurban
14	LP001096	Female	No	0	Graduate	No	4666	0	124.0	360.0	1.0	Semiurban

- The dataset contains details collected from a range of borrowers and seems to primarily focus on home loans. An initial review reveals the presence of outliers and missing values within the data.



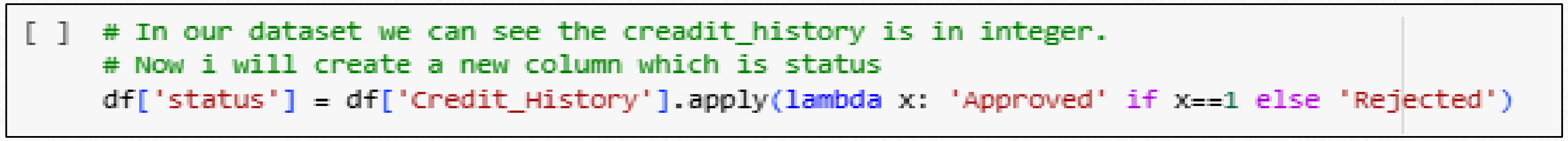
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            367 non-null    object 
 1   Gender             367 non-null    object 
 2   Married            367 non-null    object 
 3   Dependents         367 non-null    object 
 4   Education          367 non-null    object 
 5   Self_Employed      367 non-null    object 
 6   ApplicantIncome    367 non-null    int64  
 7   CoapplicantIncome  367 non-null    int64  
 8   LoanAmount         367 non-null    float64
 9   Loan_Amount_Term   367 non-null    float64
 10  Credit_History     367 non-null    float64
 11  Property_Area      367 non-null    object 
 12  status             367 non-null    object 
dtypes: float64(3), int64(2), object(8)
memory usage: 37.4+ KB
```

Key Features include:

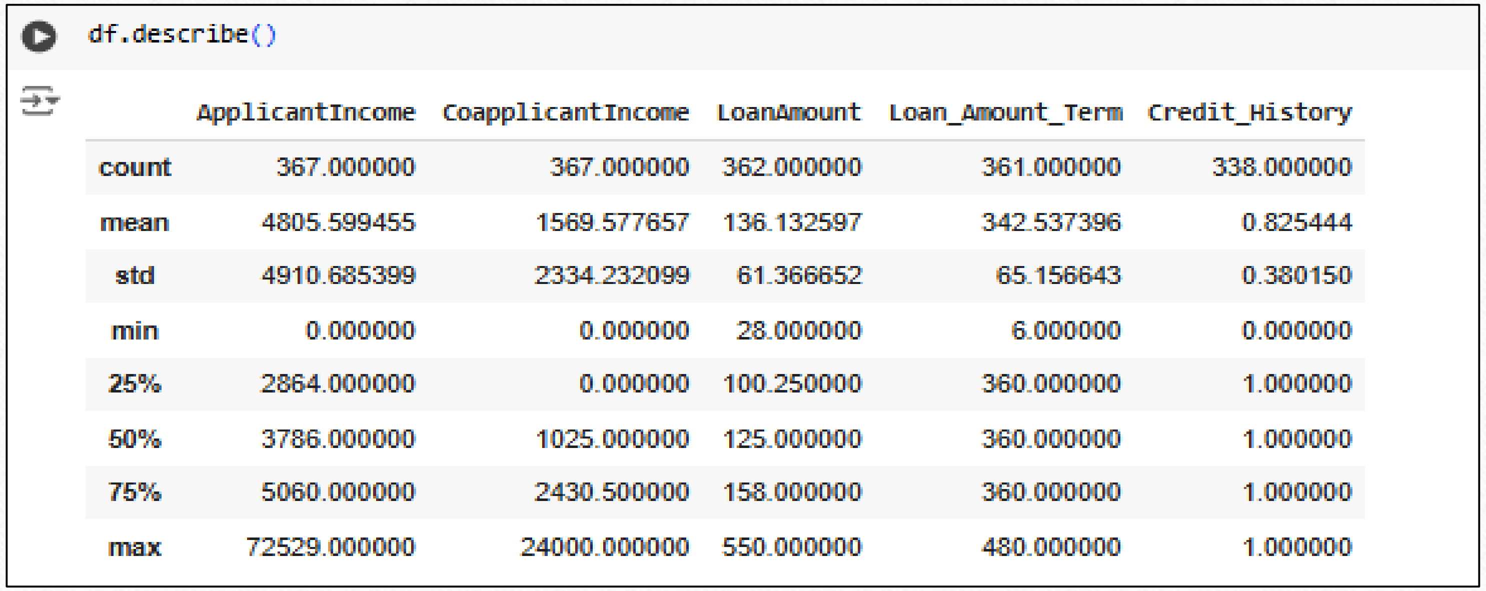
- Loan ID: A unique identifier assigned to each loan application for tracking and reference purposes.
- Gender: The gender of the applicant (e.g., Male, Female).
- Married: Marital status of the applicant (e.g., Yes, No).
- Dependents: The number of dependents or family members who rely on the applicant for financial support.
- Education: The educational qualification of the applicant (e.g., Graduate, Not Graduate).
- Self Employed: Indicates whether the applicant is self-employed or not (e.g., Yes, No).
- Applicant Income: The monthly income of the applicant.
- Co-applicant Income: The monthly income of the co-applicant, if any.
- Loan Amount: The total amount of the loan requested by the applicant.
- Loan Amount Term: The tenure or duration of the loan in months.
- Credit History: A numerical indicator of the applicant's credit history, reflecting their ability to repay the loan. (e.g., 1, 0).
- Property Area: The geographical area or type of area where the property is located (e.g., Urban, Semiurban, Rural).

✓ Here I created a new column “status” to check whether the loan is approved or not.



```
[ ]  # In our dataset we can see the creadit_history is in integer.
     # Now i will create a new column which is status
     df['status'] = df['Credit_History'].apply(lambda x: 'Approved' if x==1 else 'Rejected')
```

❖ Describing the dataset

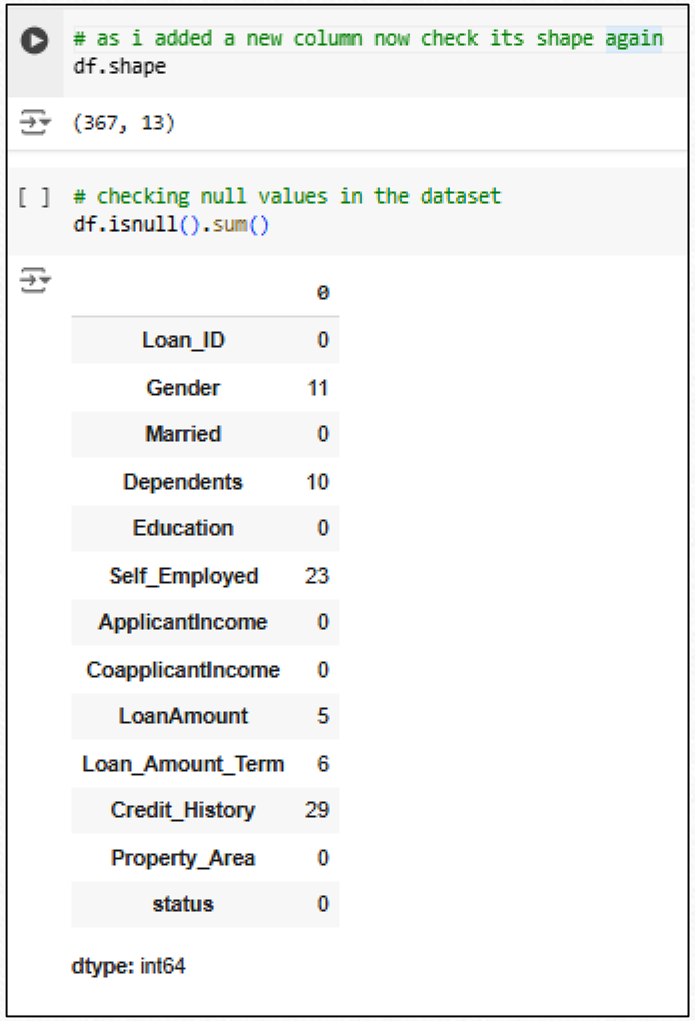


```
df.describe()
```

	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.

❖ Handling missing value

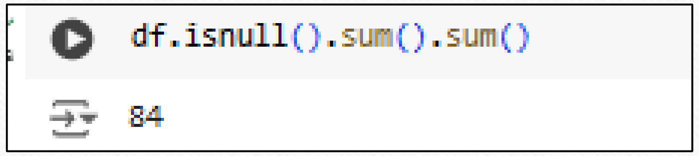


```
# as i added a new column now check its shape again
df.shape
```

```
(367, 13)
```

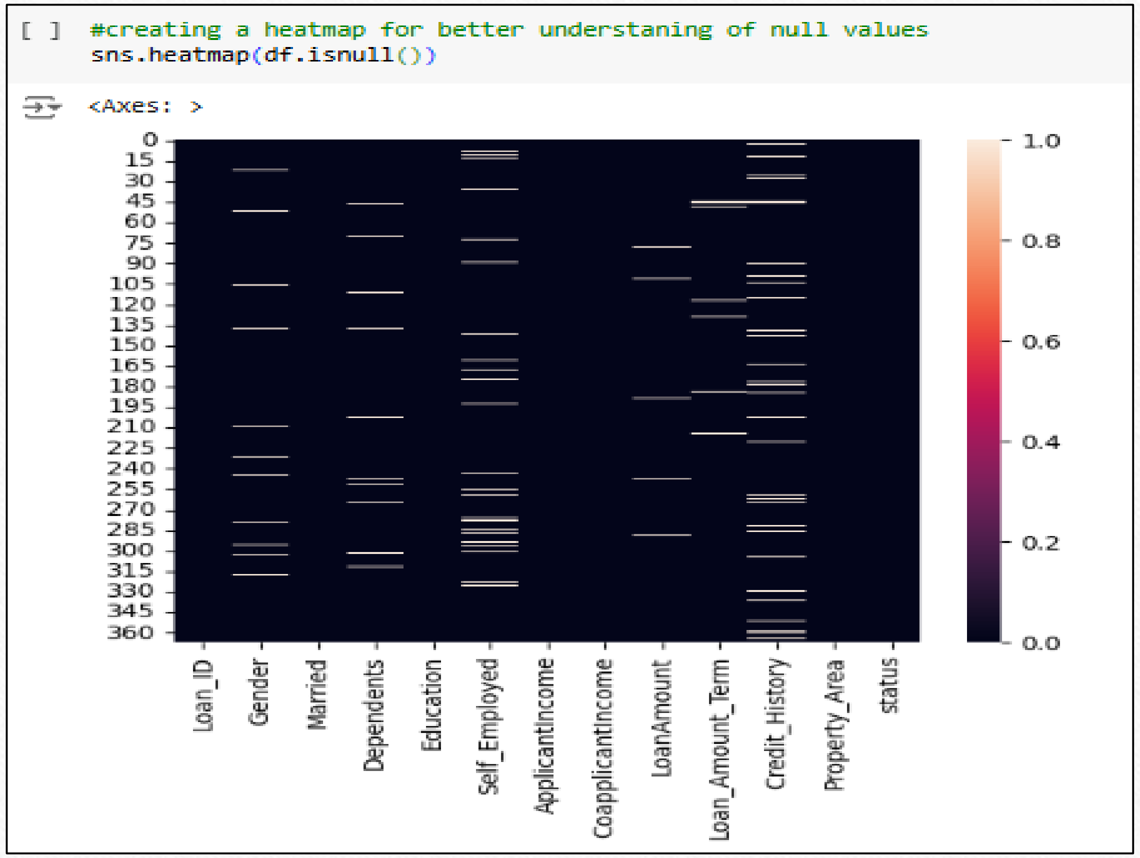
```
[ ] # checking null values in the dataset
    df.isnull().sum()
```

```
                    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
           status   0

dtype: int64
```



```
df.isnull().sum().sum()
```

```
84
```



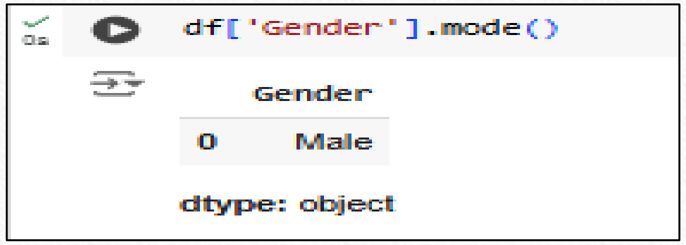
```
[ ] #creating a heatmap for better understaning of null values
    sns.heatmap(df.isnull())
```

```
<Axes: >
```

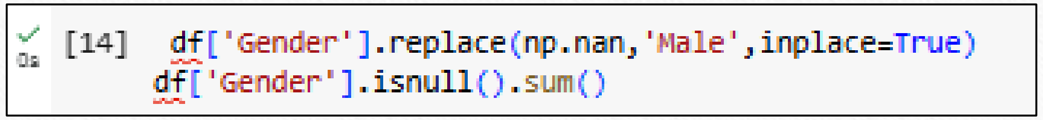
- The dataset contains 84 missing values, excluding the Loan_Status column. Specifically:
  - Self_Employed has 23 missing entries.
  - Credit_History has 29 missing entries.
  - Dependents has 10 missing entries.
  - Gender has 11 missing entries.
  - LoanAmount has 5 missing entries.
  - Loan_Amount_Term has 6 missing entries.

- A heatmap was used to visualize these null values, highlighting areas that require attention. Addressing these missing values is essential before proceeding with further analysis.

# replacing null values in "Gender" column with Mode

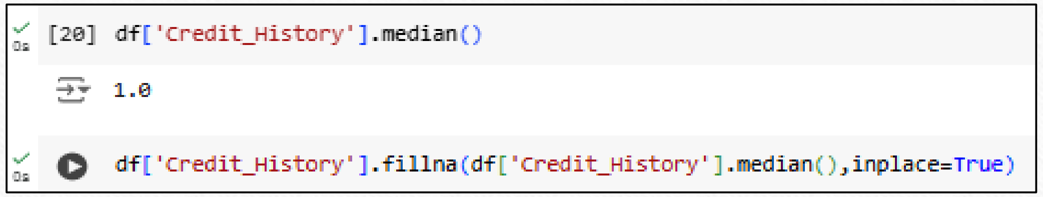


```
df['Gender'].mode()

        Gender
0       Male

dtype: object
```



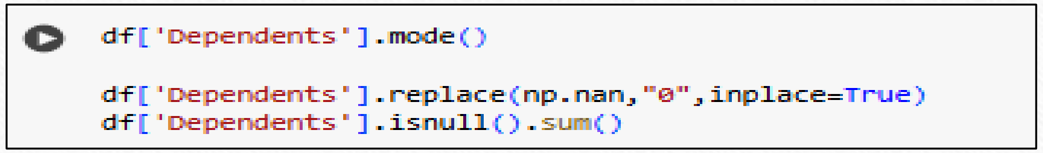
```
[14]  df['Gender'].replace(np.nan,'Male',inplace=True)
      df['Gender'].isnull().sum()
```

# replacing null values in “creadit_History” column with Median.



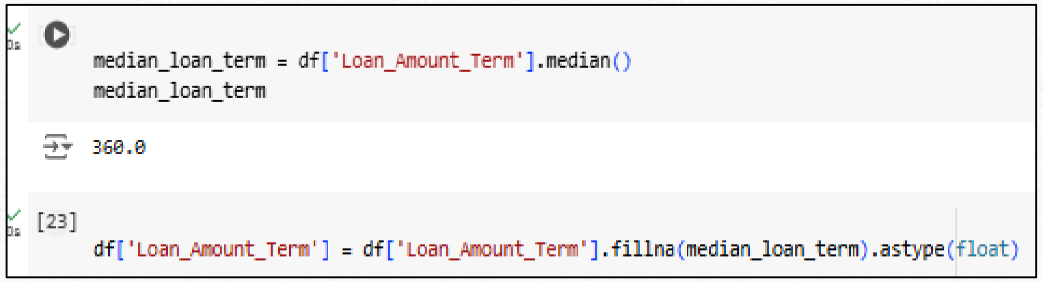
```
[20] df['Credit_History'].median()

     1.0

     df['Credit_History'].fillna(df['Credit_History'].median(),inplace=True)
```

# replacing null values in “Dependents” column with Mode.



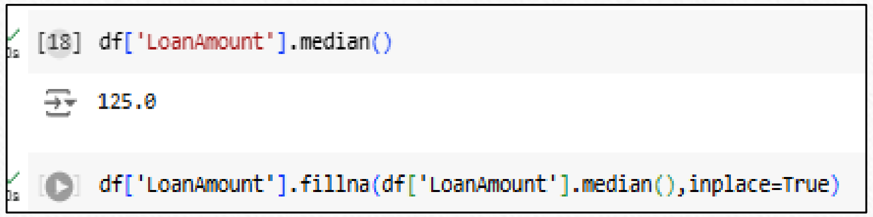
```
df['Dependents'].mode()

df['Dependents'].replace(np.nan,"0",inplace=True)
df['Dependents'].isnull().sum()
```

# replacing null values in “Loan_Amount_term” column with Median.



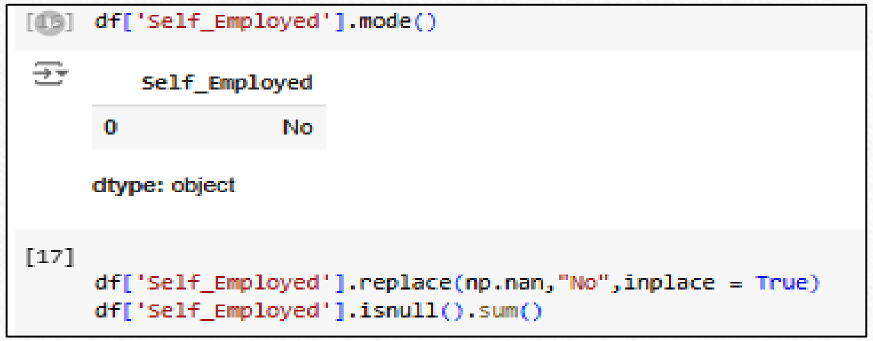
```
median_loan_term = df['Loan_Amount_Term'].median()
median_loan_term

360.0

[23]
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(median_loan_term).astype(float)
```

# replacing null values in "LoanAmount" column with Median.



```
[18] df['LoanAmount'].median()

     125.0

     df['LoanAmount'].fillna(df['LoanAmount'].median(),inplace=True)
```

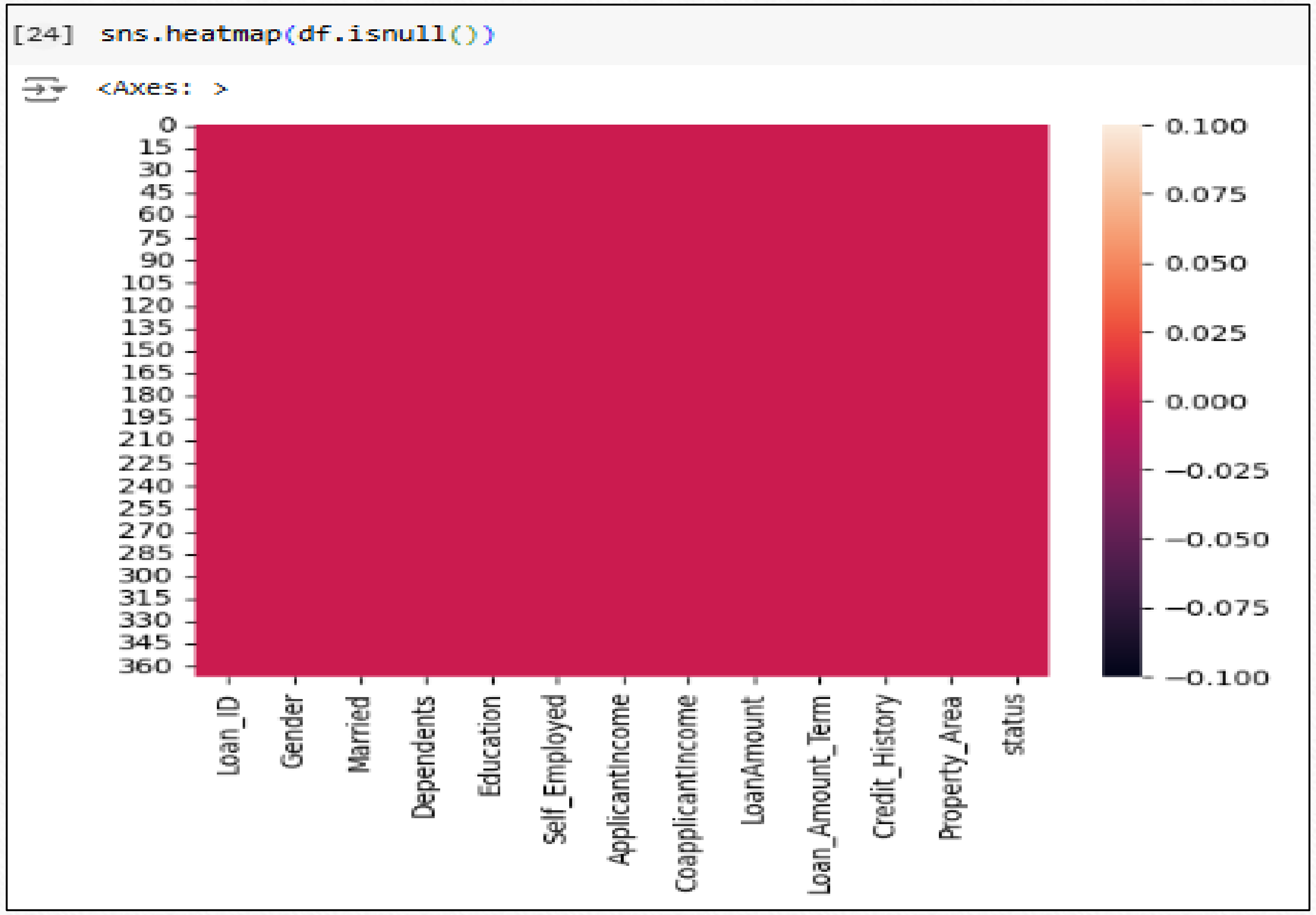
# replacing null values in "self_Employed" column with Median.



```
[15] df['Self_Employed'].mode()

         Self_Employed
     0              No

     dtype: object

[17]
     df['Self_Employed'].replace(np.nan,"No",inplace = True)
     df['Self_Employed'].isnull().sum()
```

- ✓ In the provided code snippets, missing values in categorical columns were filled using the mode of each column. For numerical columns like LoanAmount, Credit_History, and Loan_Amount_Term, the null values were filled with the median of the respective columns. This approach was chosen because the median is less affected by outliers, making it a more robust measure for handling extreme values.

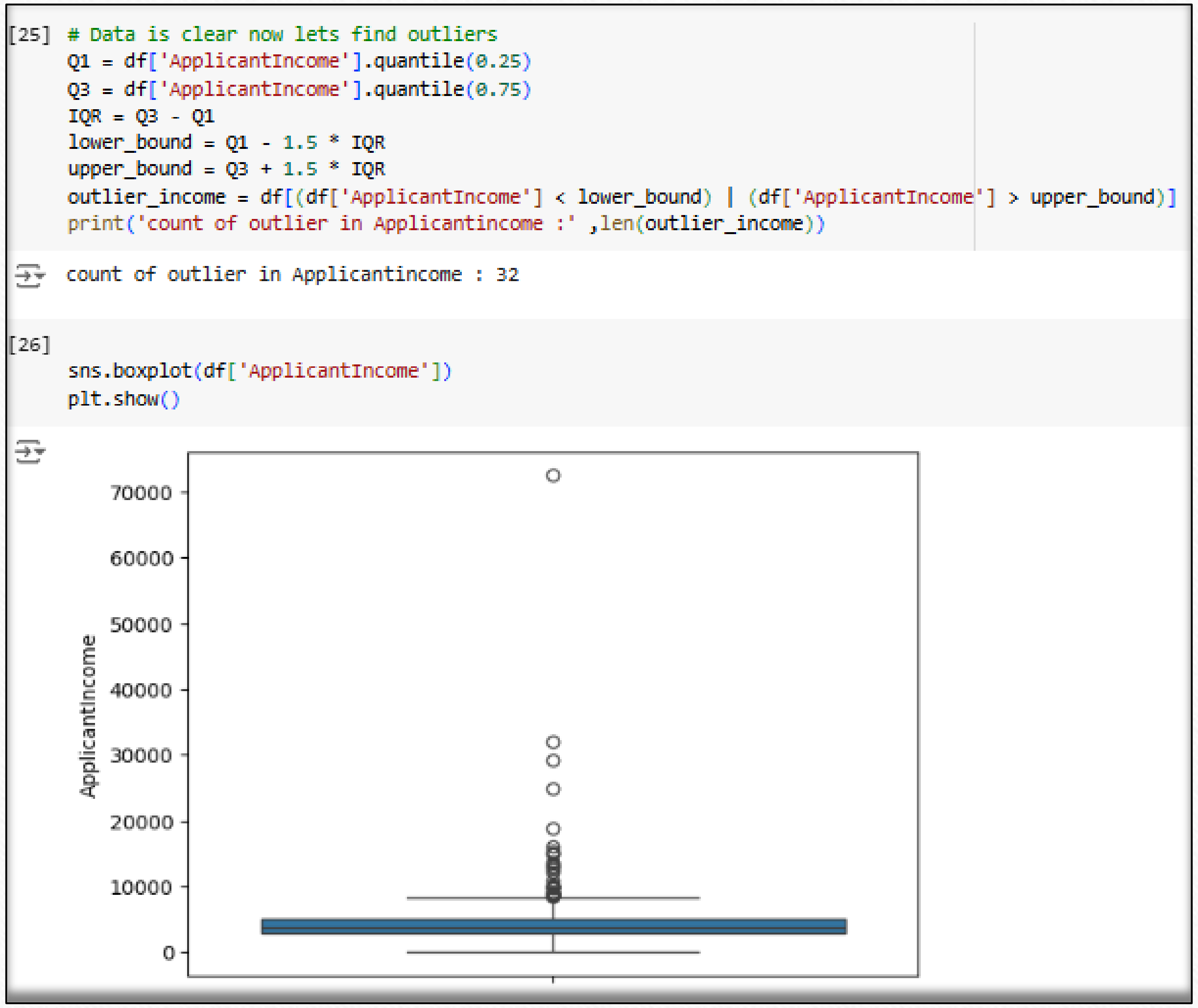
✓ **This heatmap shows that the data set is clear.**



```
[24] sns.heatmap(df.isnull())
```

```
<Axes: >
```

❖ Finding Outliers



```
[25] # Data is clear now lets find outliers
     Q1 = df['ApplicantIncome'].quantile(0.25)
     Q3 = df['ApplicantIncome'].quantile(0.75)
     IQR = Q3 - Q1
     lower_bound = Q1 - 1.5 * IQR
     upper_bound = Q3 + 1.5 * IQR
     outlier_income = df[(df['ApplicantIncome'] < lower_bound) | (df['ApplicantIncome'] > upper_bound)]
     print('count of outlier in Applicantincome :' ,len(outlier_income))
```

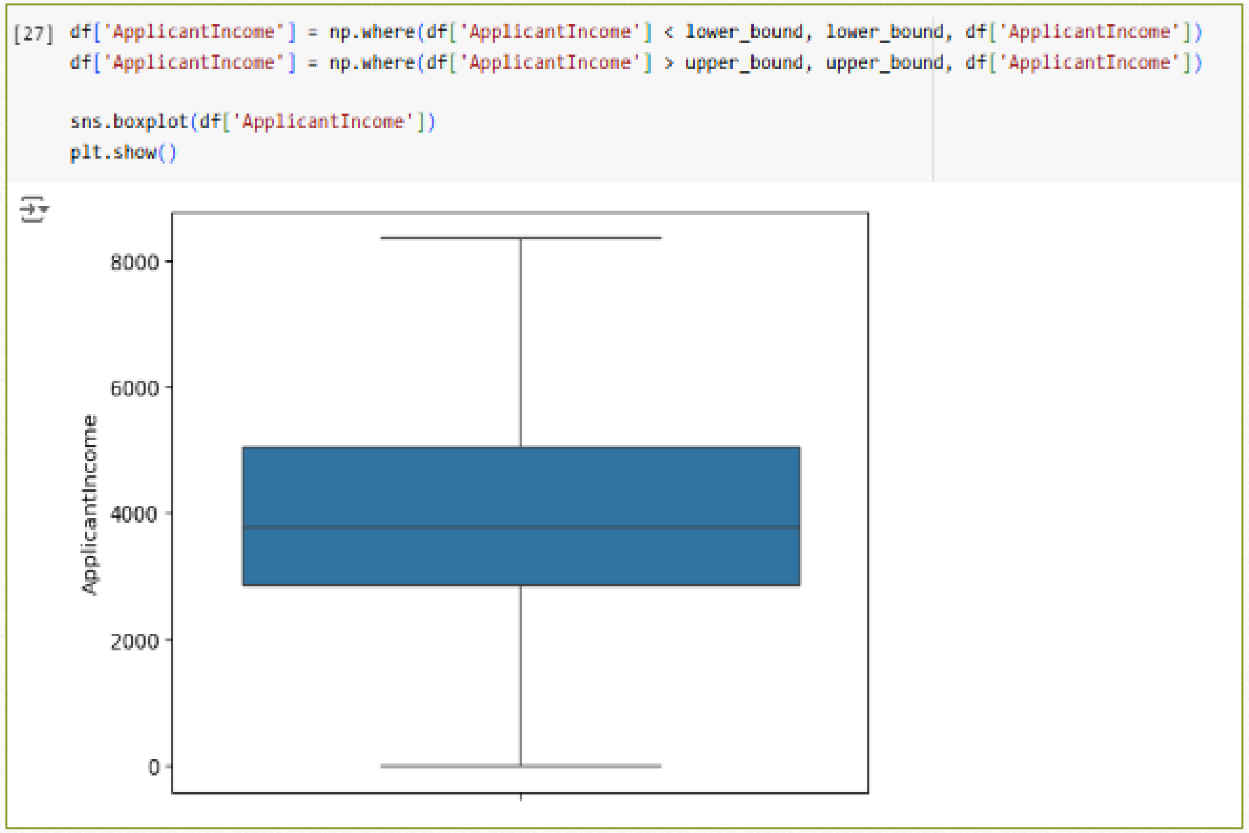
```
count of outlier in Applicantincome : 32
```

```
[26]
     sns.boxplot(df['ApplicantIncome'])
     plt.show()
```

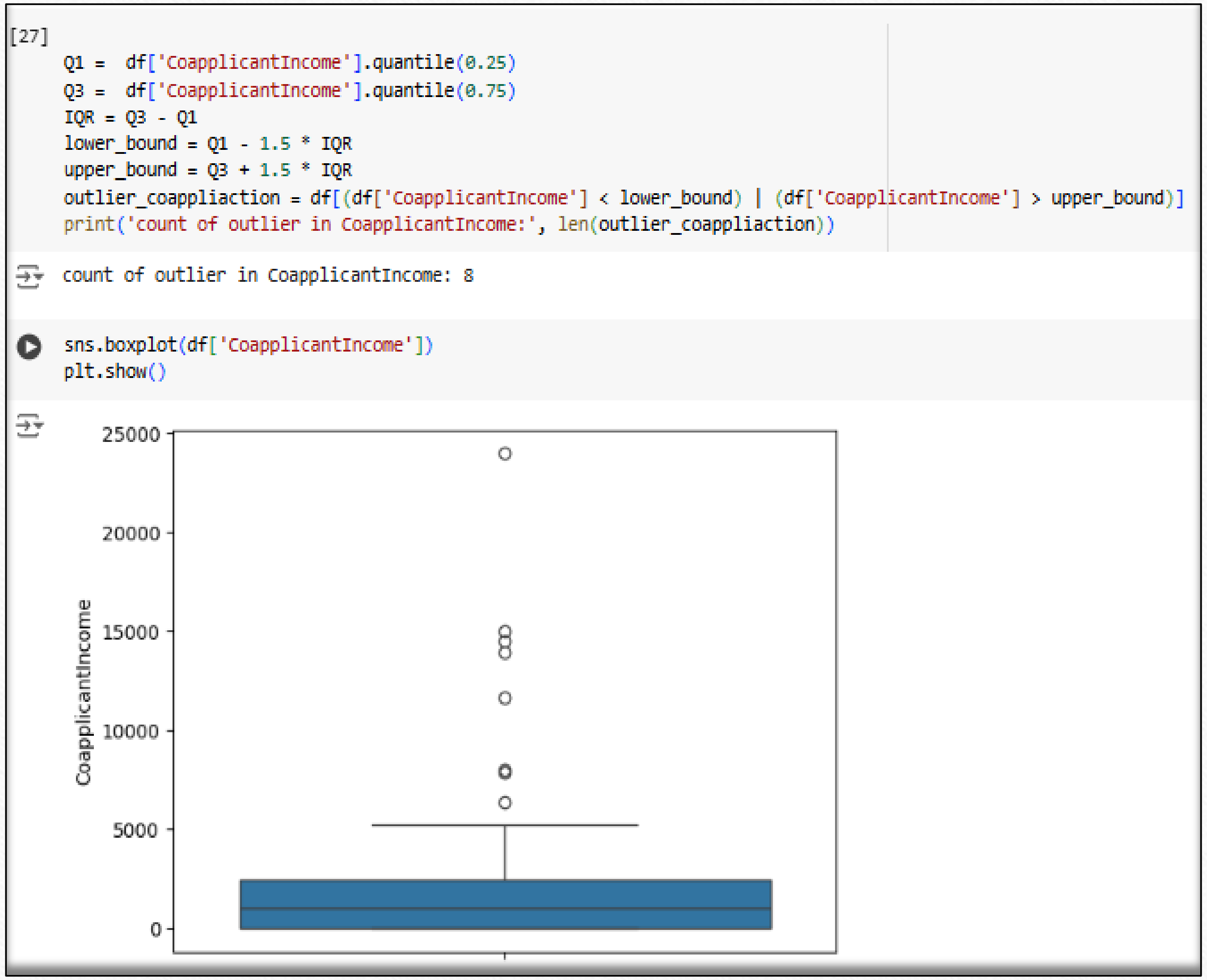
# outlier in "ApplicantIncome"

- ➢ Here i used IQR method for finding outliers and box plot for visualization of outliers.

- ✓ **The plot reveals several points beyond the upper whisker, indicating outliers in the "ApplicantIncome" variable. A total of 32 outliers were identified in this column. These outliers were addressed by replacing them with the column's median. A boxplot was created post-cleaning to confirm the changes.**



```
[27] df['ApplicantIncome'] = np.where(df['ApplicantIncome'] < lower_bound, lower_bound, df['ApplicantIncome'])
     df['ApplicantIncome'] = np.where(df['ApplicantIncome'] > upper_bound, upper_bound, df['ApplicantIncome'])

     sns.boxplot(df['ApplicantIncome'])
     plt.show()
```


[27]

```
Q1 =  df['CoapplicantIncome'].quantile(0.25)
Q3 =  df['CoapplicantIncome'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outlier_coappliaction = df[(df['CoapplicantIncome'] < lower_bound) | (df['CoapplicantIncome'] > upper_bound)]
print('count of outlier in CoapplicantIncome:', len(outlier_coappliaction))
```

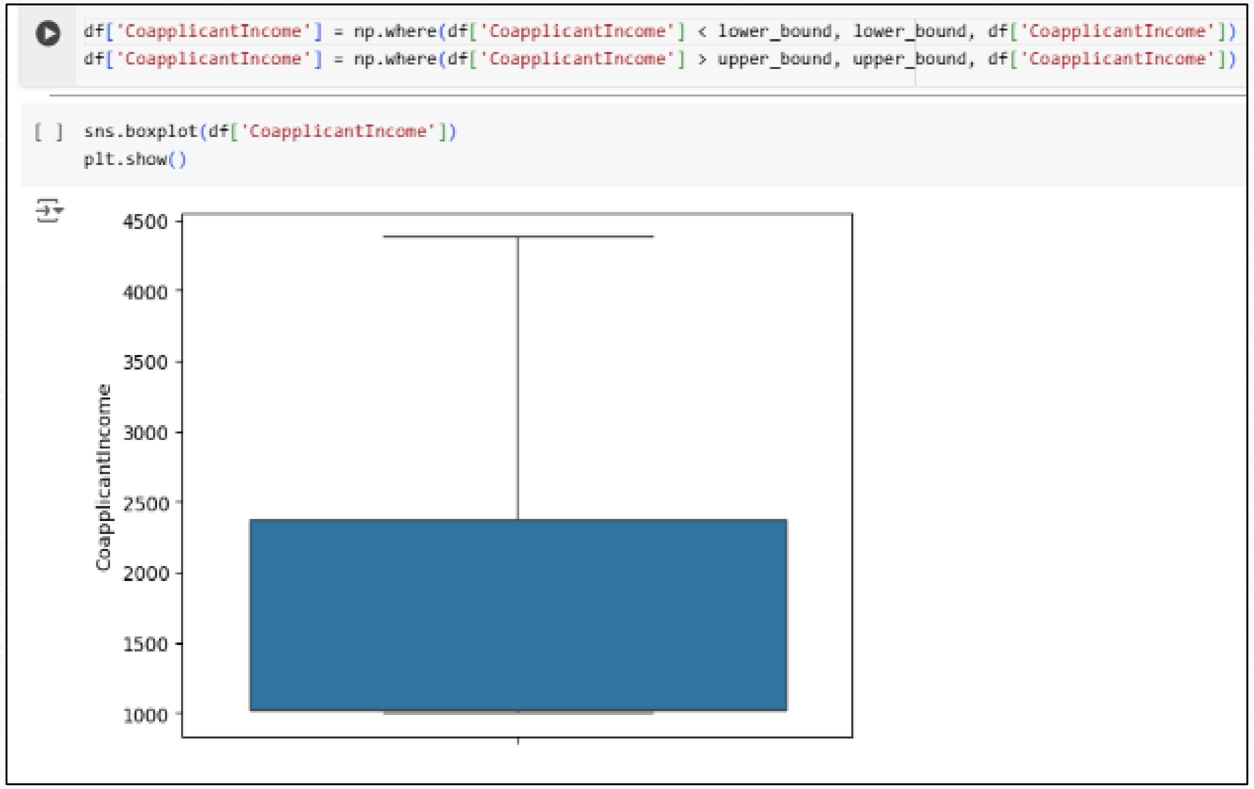
```
count of outlier in CoapplicantIncome: 8
```

```
sns.boxplot(df['CoapplicantIncome'])
plt.show()
```

# outlier in “CoapplicantIncome”

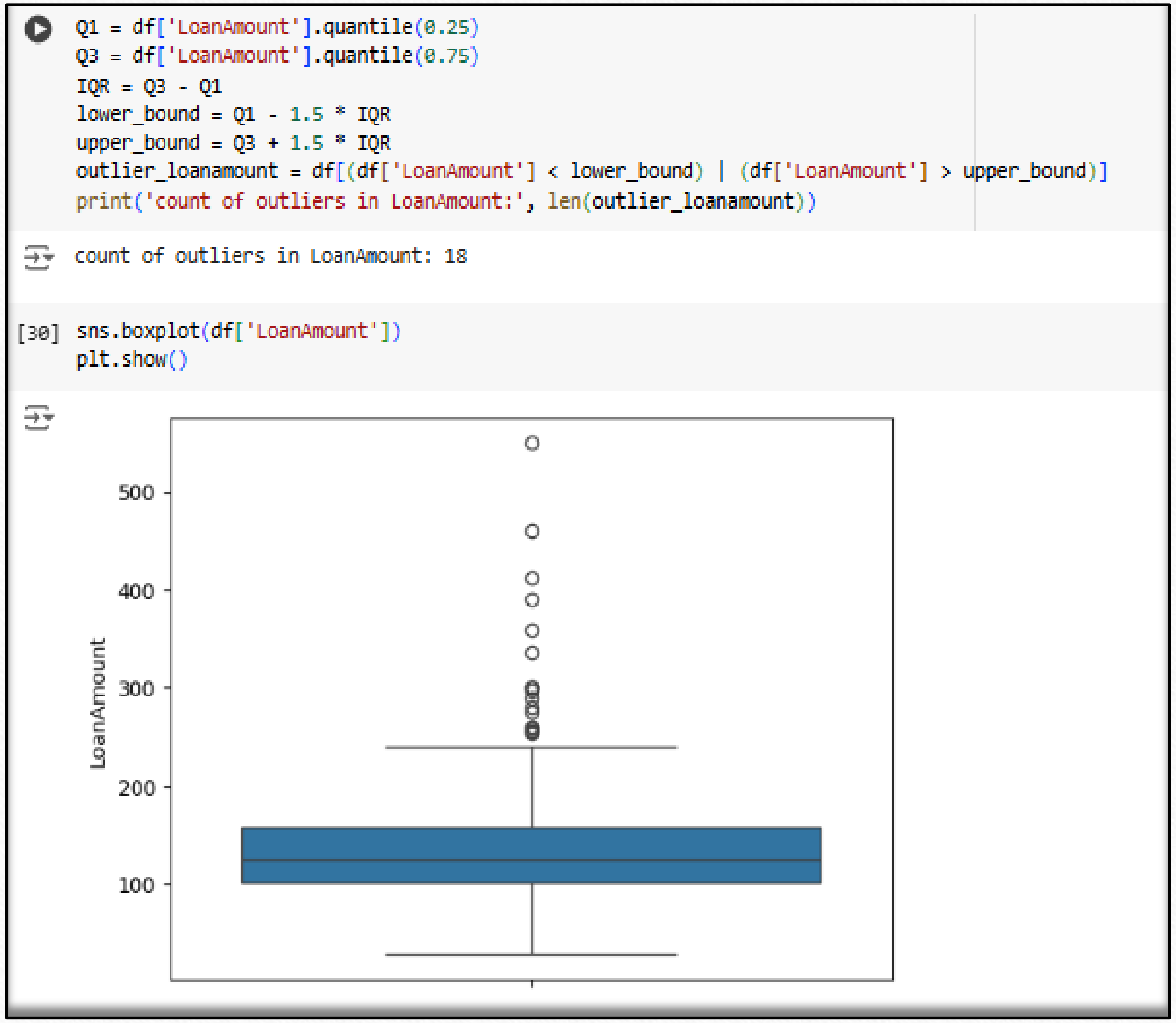
- Here i used IQR method for finding outliers and box plot for visualization of outliers.

- ✓ **This code identifies eight outliers in the "coapplicantincome" column. To visually represent these outliers, I used a plot. To handle the outliers, I opted for the median as the central tendency measure because it is less influenced by extreme values compared to the mean, making it a more robust choice in such scenarios.**



```
df['CoapplicantIncome'] = np.where(df['CoapplicantIncome'] < lower_bound, lower_bound, df['CoapplicantIncome'])
df['CoapplicantIncome'] = np.where(df['CoapplicantIncome'] > upper_bound, upper_bound, df['CoapplicantIncome'])
```

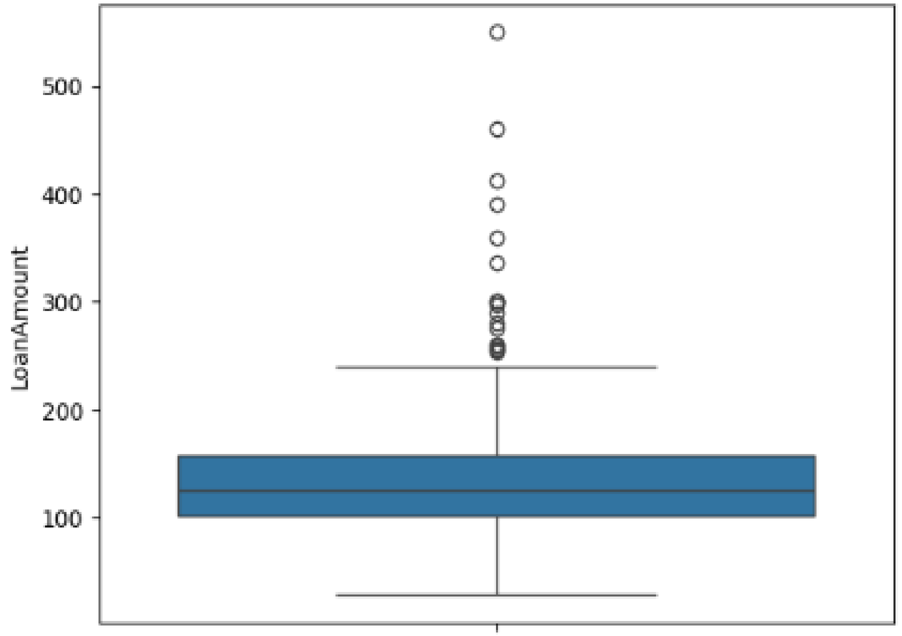
```
sns.boxplot(df['CoapplicantIncome'])
plt.show()
```



```
Q1 = df['LoanAmount'].quantile(0.25)
Q3 = df['LoanAmount'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outlier_loanamount = df[(df['LoanAmount'] < lower_bound) | (df['LoanAmount'] > upper_bound)]
print('count of outliers in LoanAmount:', len(outlier_loanamount))
```

```
count of outliers in LoanAmount: 18
```

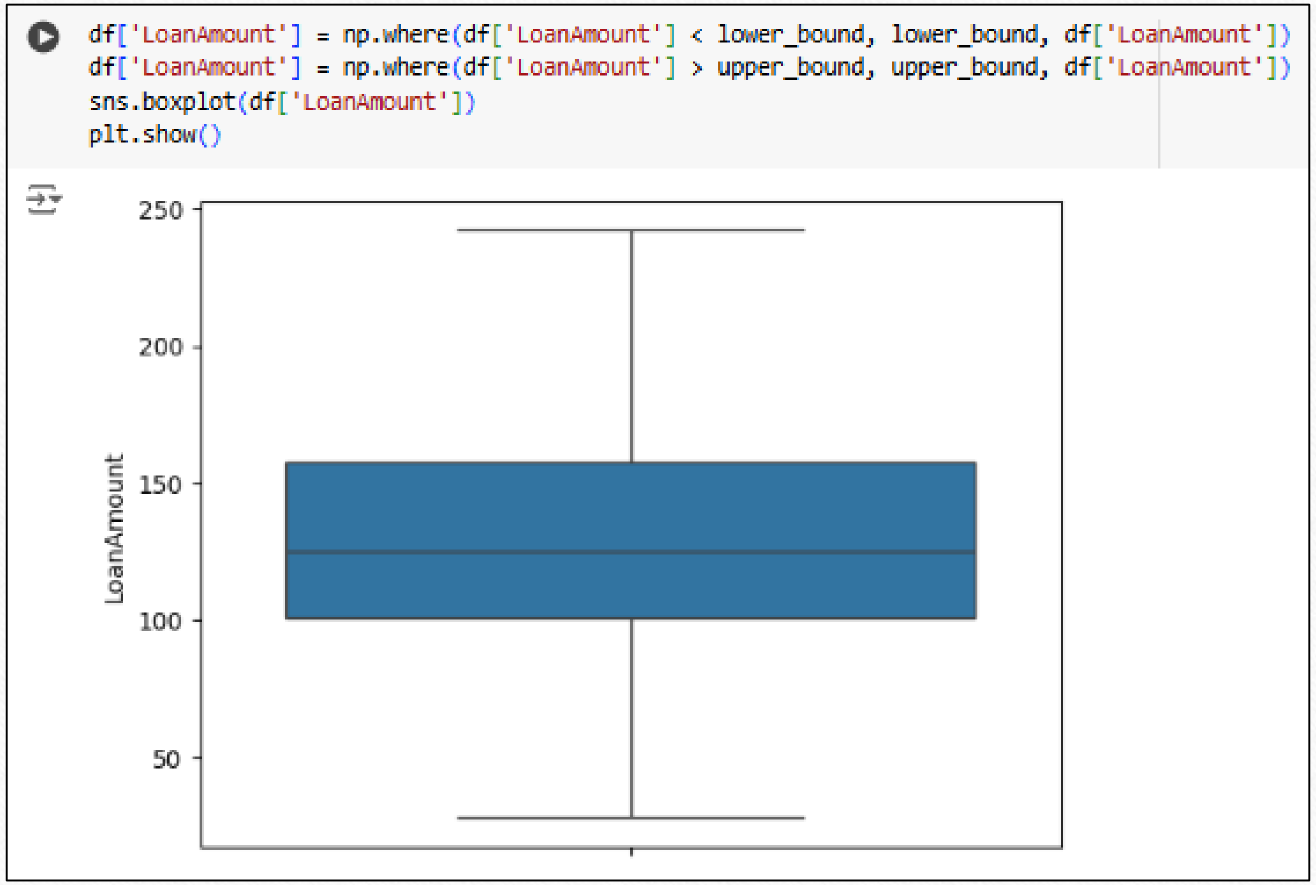
```
[30] sns.boxplot(df['LoanAmount'])
     plt.show()
```



# outlier in “LoanAmount”

- Here i used IQR method for finding outliers and box plot for visualization of outliers.

✓ **This attribute originally had 18 outliers, which were addressed by imputing with the median. The graph below demonstrates that the LoanAmount attribute now has no outliers.**

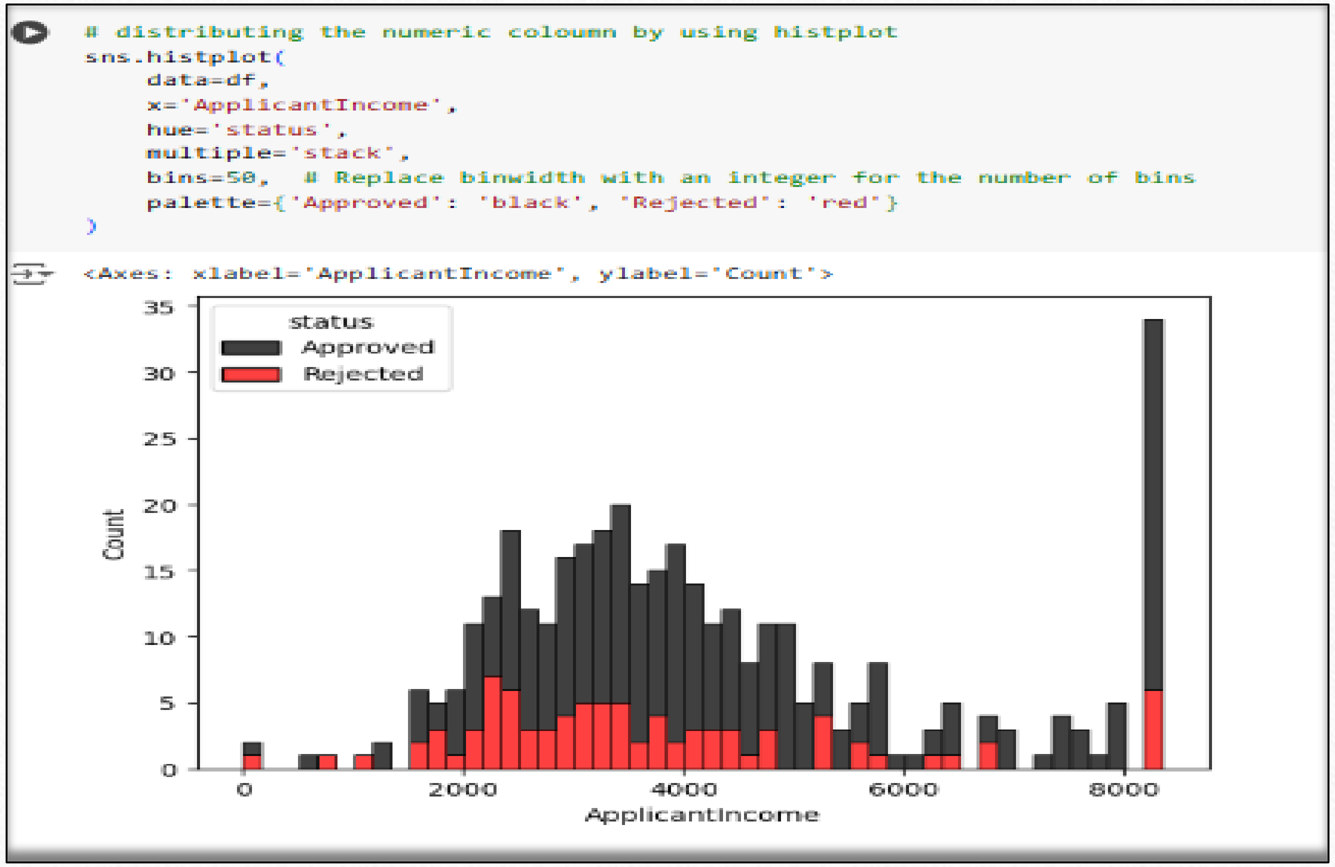


```
df['LoanAmount'] = np.where(df['LoanAmount'] < lower_bound, lower_bound, df['LoanAmount'])
df['LoanAmount'] = np.where(df['LoanAmount'] > upper_bound, upper_bound, df['LoanAmount'])
sns.boxplot(df['LoanAmount'])
plt.show()
```

- ✓ Outlier detection is not applied to "Loan_Amount_Term" and "Credit_History" because their nature makes such methods irrelevant. "Loan_Amount_Term" represents the loan duration in months and is a categorical variable with a limited range of discrete values. Similarly, "Credit_History" is a binary variable (0 or 1) that indicates whether the credit history is good or bad. Outlier detection techniques are unsuitable for these types of variables.

❖ Analyze the Distribution of Numeric Columns Using Histograms

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

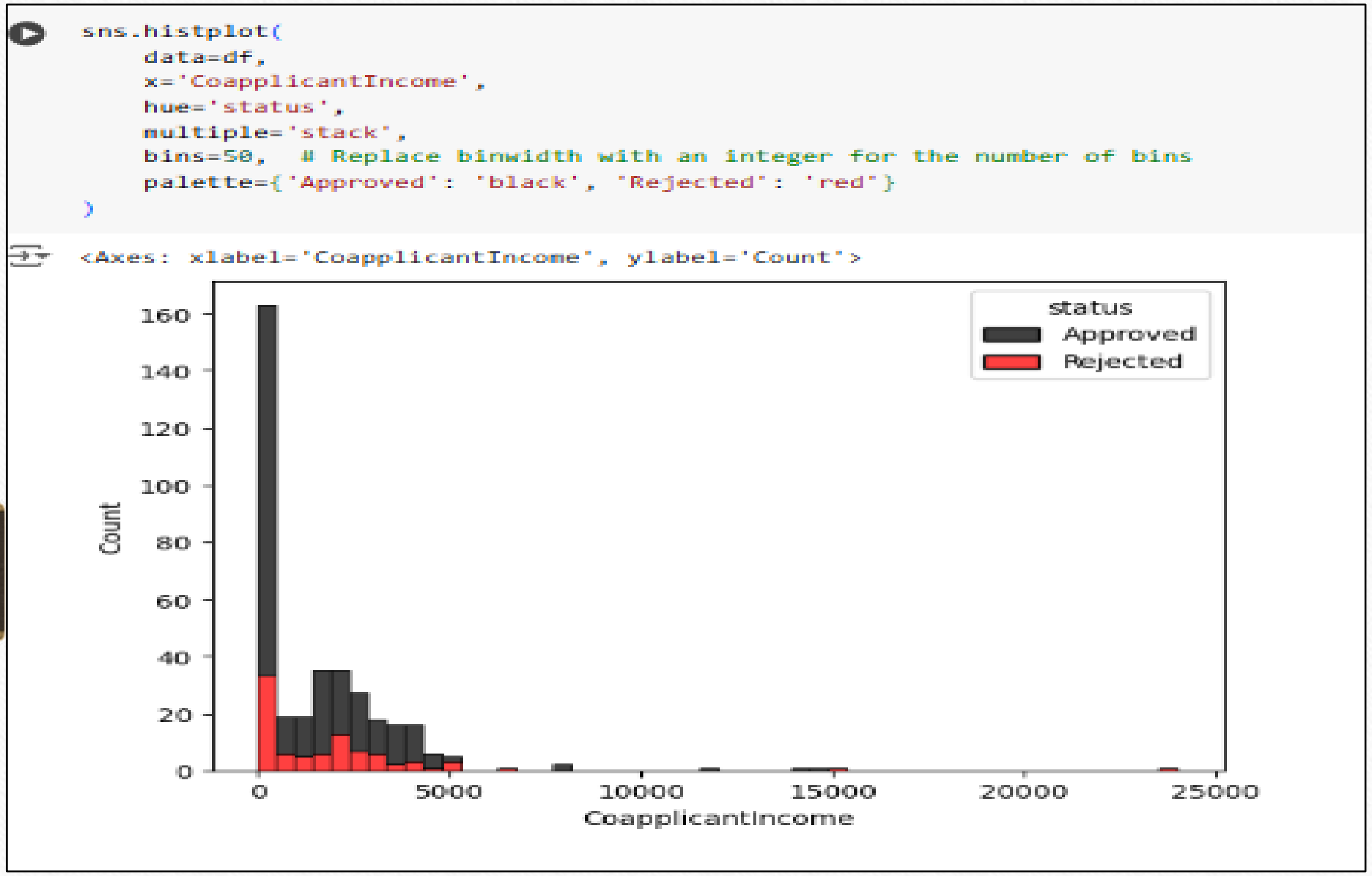


```
# distributing the numeric coloumn by using histplot
sns.histplot(
        data=df,
        x='ApplicantIncome',
        hue='status',
        multiple='stack',
        bins=50,  # Replace binwidth with an integer for the number of bins
        palette={'Approved': 'black', 'Rejected': 'red'}
)
```

```
<Axes: xlabel='ApplicantIncome', ylabel='Count'>
```

# histplot of "ApplicantIncome"

- ✓ The graph shows that lower-income applicants are more frequent and have higher approval rates, while high-income applicants are less frequent and face higher rejection rates.



```
sns.histplot(
    data=df,
    x='CoapplicantIncome',
    hue='status',
    multiple='stack',
    bins=50,  # Replace binwidth with an integer for the number of bins
    palette={'Approved': 'black', 'Rejected': 'red'}
)
```

```
<Axes: xlabel='CoapplicantIncome', ylabel='Count'>
```

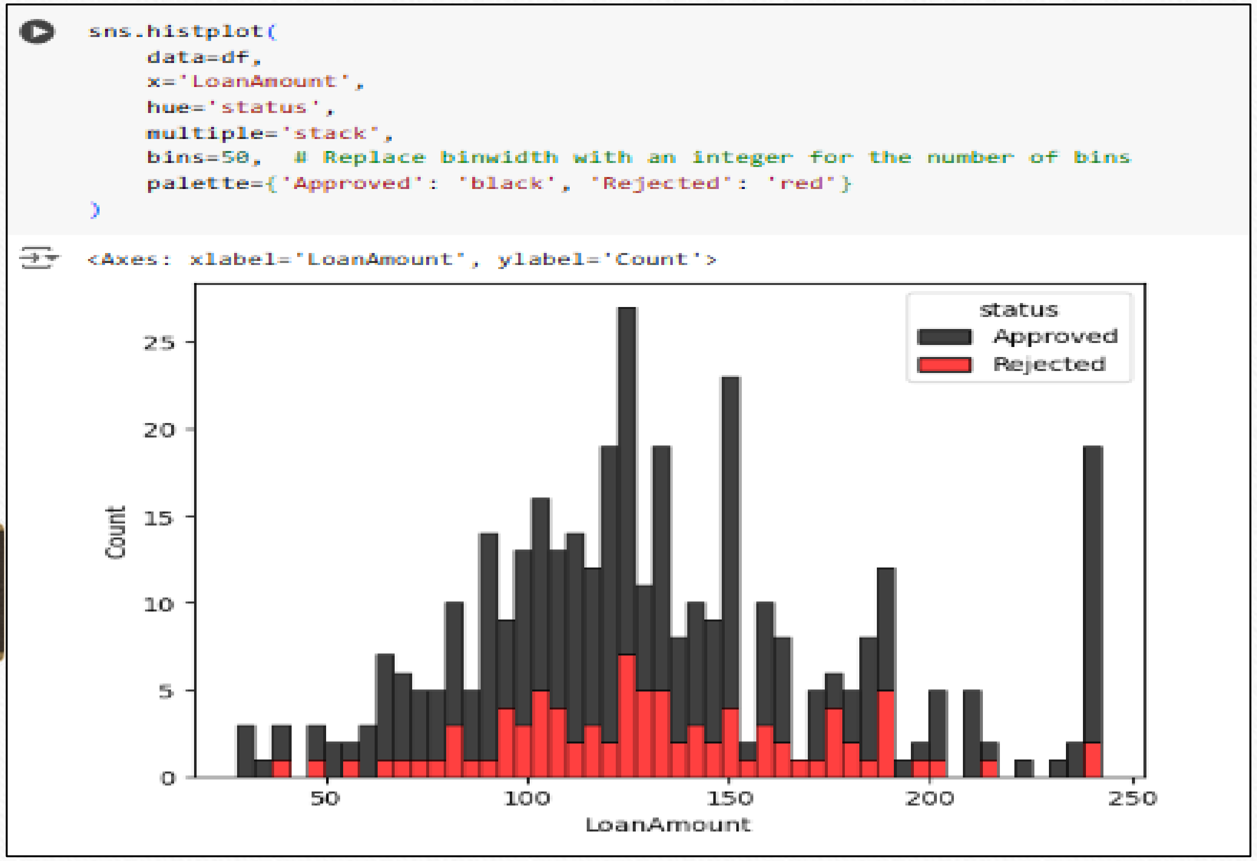
# Histplot of "CoapplicanIncome"

❑ The above graph reveals the following insights:

✓ Most coapplicants have incomes up to ₹5,000, with higher-income coapplicants being less common.

✓ Loans are more likely to be approved when the coapplicant's income is lower.

✓ As the coapplicant's income increases, both the number of approved and rejected loans decreases.



```
sns.histplot(
    data=df,
    x='LoanAmount',
    hue='status',
    multiple='stack',
    bins=50,  # Replace binwidth with an integer for the number of bins
    palette={'Approved': 'black', 'Rejected': 'red'}
)
```

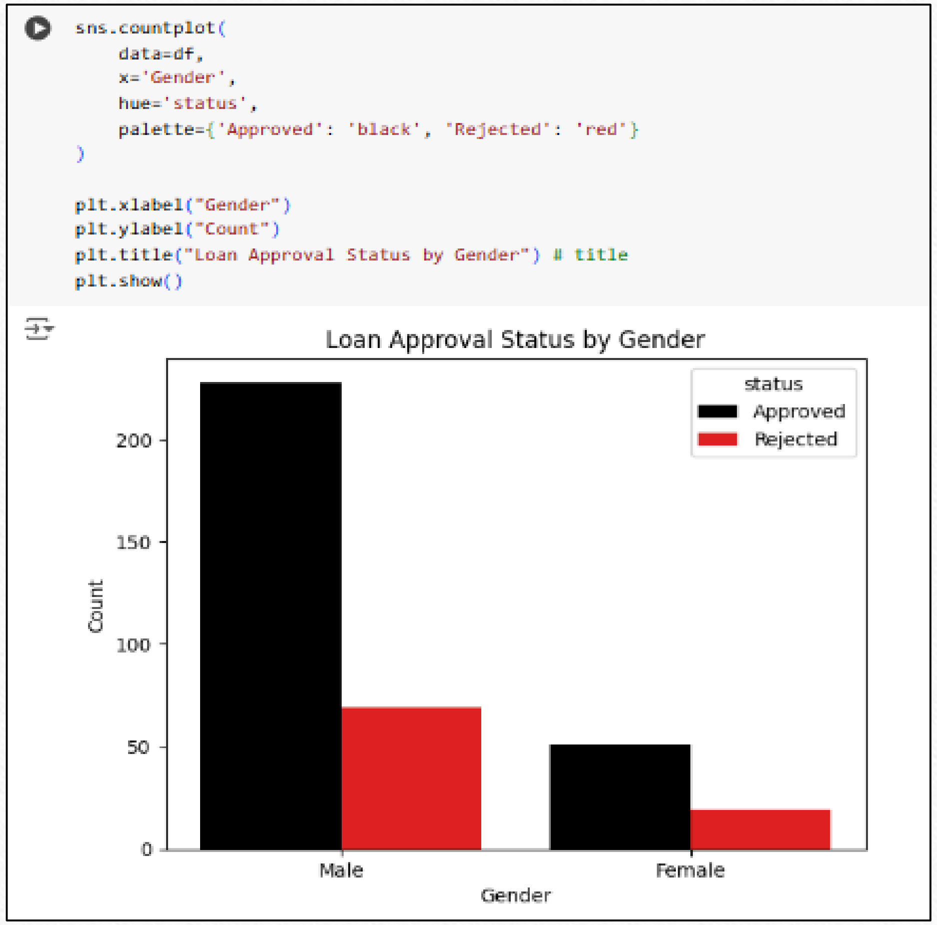
```
<Axes: xlabel='LoanAmount', ylabel='Count'>
```

# Histplot of "LoanAmount"

- ❑ The above graph reveals the following insights:
- ✓ The majority of loans are for amounts up to ₹100, with fewer loans granted for higher amounts.
- ✓ Approval rates: Loans with smaller amounts are more likely to receive approval.
- ✓ Trends: As the loan amount increases, both approved and rejected loans become less frequent.

❖Visualize the frequency distribution of categorical variables.

✓ This dataset contains six categorical columns: 'Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', and 'Property_Area'.



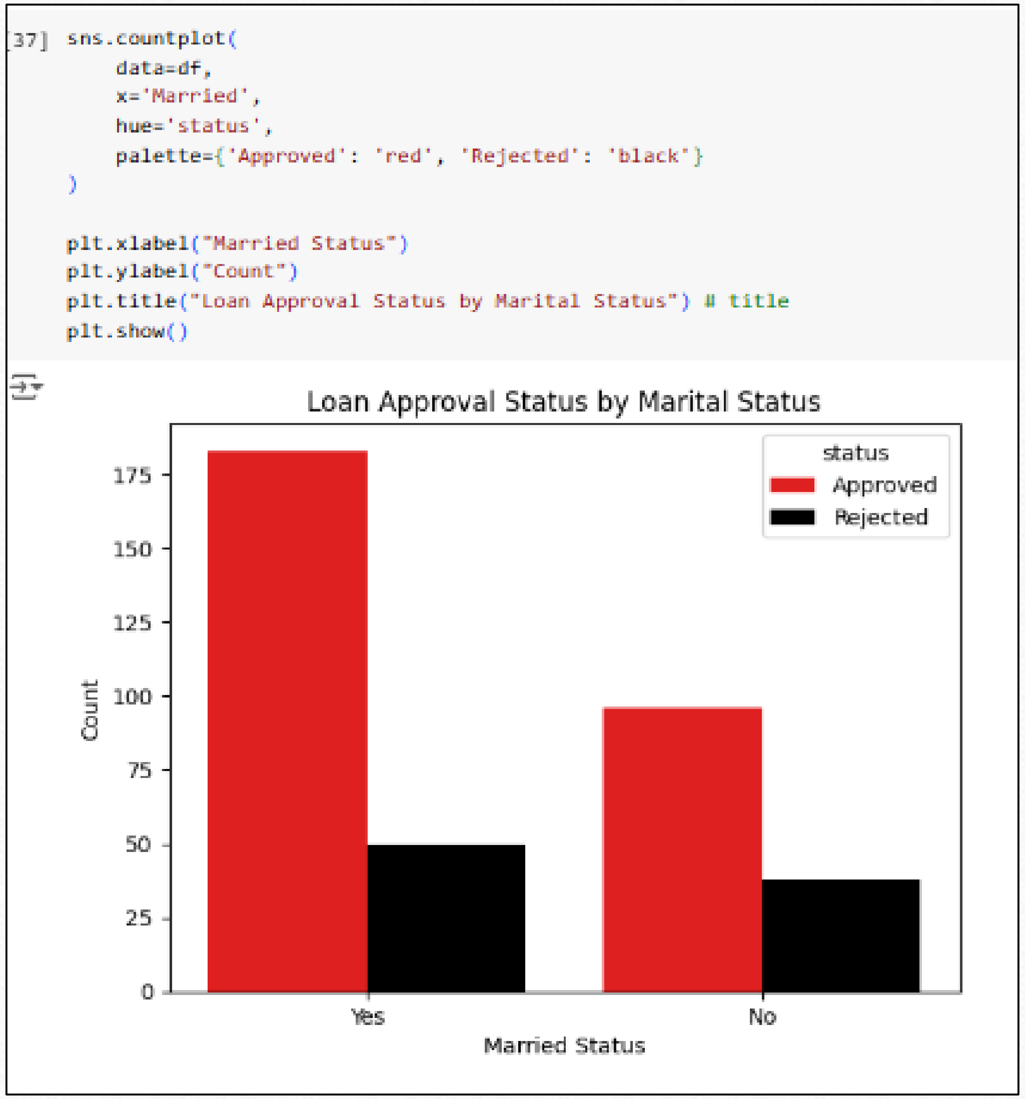
```
sns.countplot(
    data=df,
    x='Gender',
    hue='status',
    palette={'Approved': 'black', 'Rejected': 'red'}
)

plt.xlabel("Gender")
plt.ylabel("Count")
plt.title("Loan Approval Status by Gender") # title
plt.show()
```

# Barplot of "Gender"

❑ The above graph reveals the following insights:

✓ A higher number of males apply for loans compared to females.

✓ Loan approval rates are similar for both genders, with a slightly higher approval rate for males.



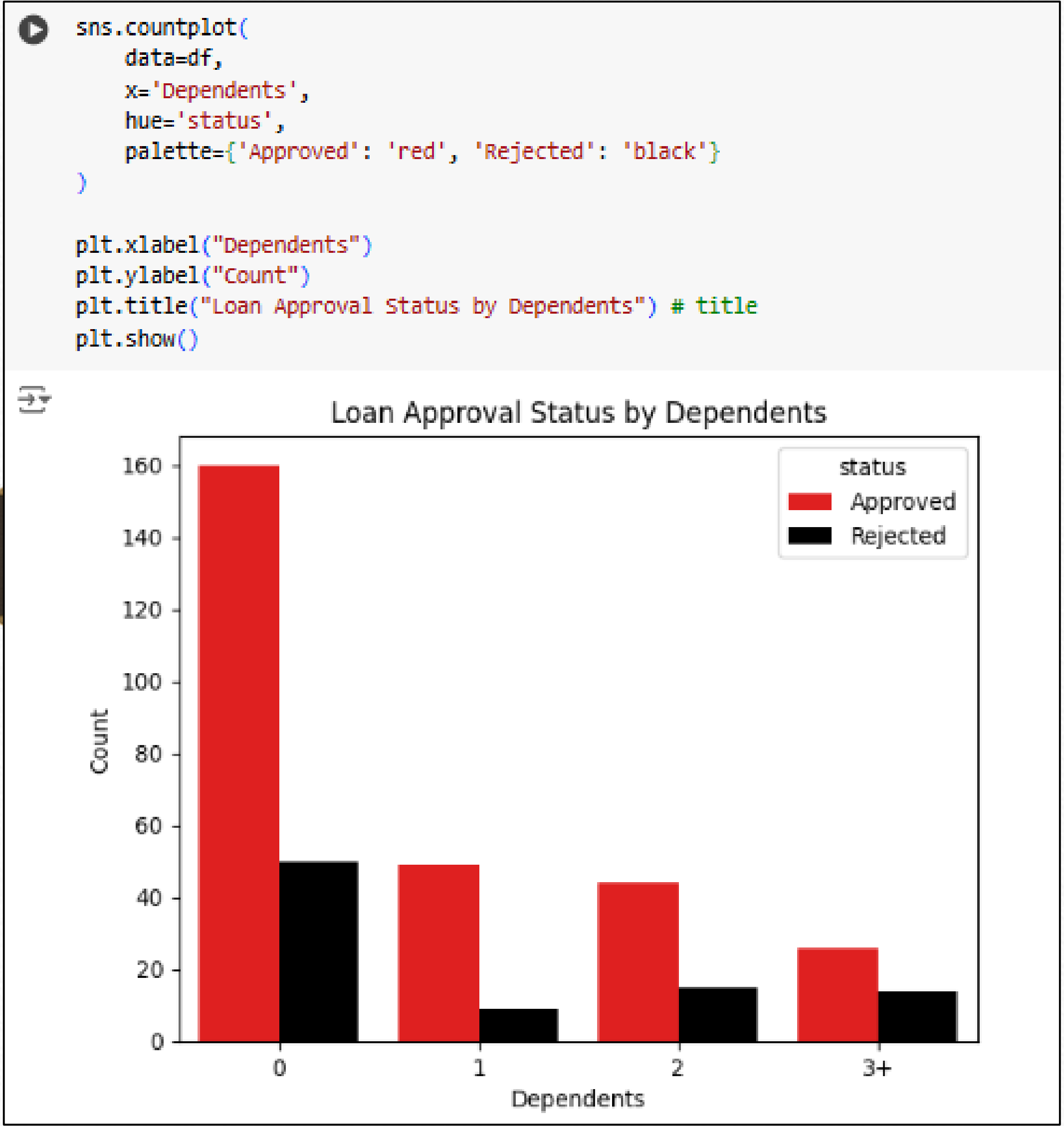
```
[37] sns.countplot(
         data=df,
         x='Married',
         hue='status',
         palette={'Approved': 'red', 'Rejected': 'black'}
     )

     plt.xlabel("Married Status")
     plt.ylabel("Count")
     plt.title("Loan Approval Status by Marital Status") # title
     plt.show()
```

# Barplot of "Married"

❑ The above graph reveals the following insights:

✓ Married individuals are more likely to apply for loans than their unmarried counterparts.

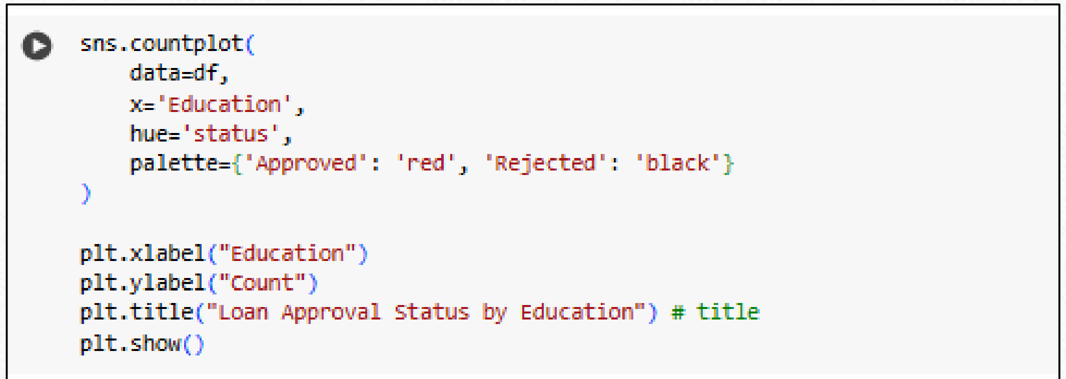
✓ Married individuals also have a slightly higher loan approval rate compared to unmarried individuals.



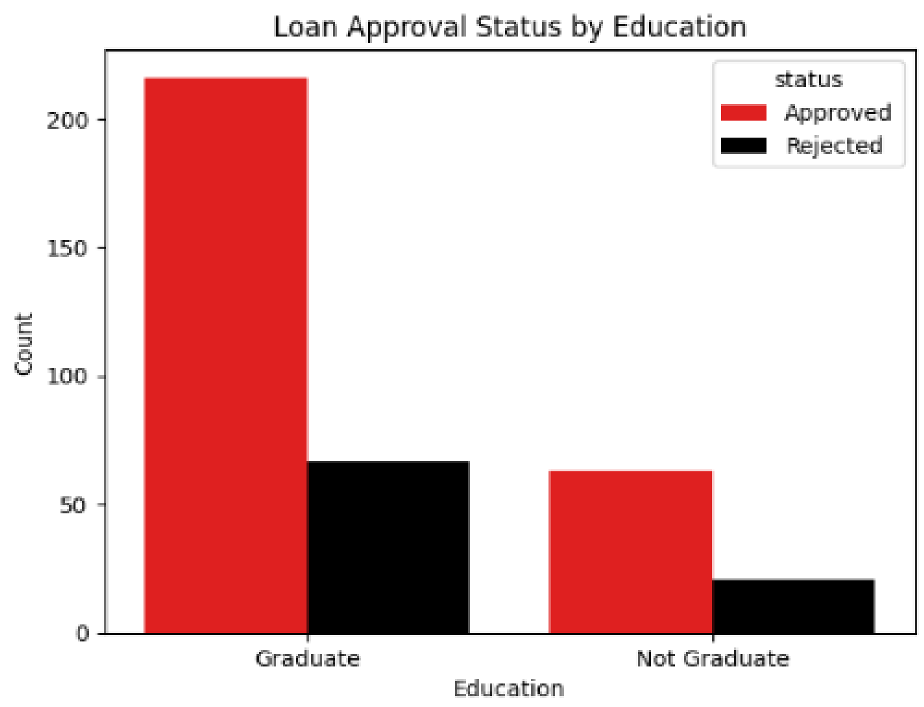
```
sns.countplot(
    data=df,
    x='Dependents',
    hue='status',
    palette={'Approved': 'red', 'Rejected': 'black'}
)

plt.xlabel("Dependents")
plt.ylabel("Count")
plt.title("Loan Approval Status by Dependents") # title
plt.show()
```

# Barplot of "Dependents"

- ❑ The above graph reveals the following insights:
- ✓ Applicants with no dependents are the most common, followed by those with 1 or 2 dependents.
- ✓ Loan approval rates seem to be relatively consistent, regardless of the number of dependents.

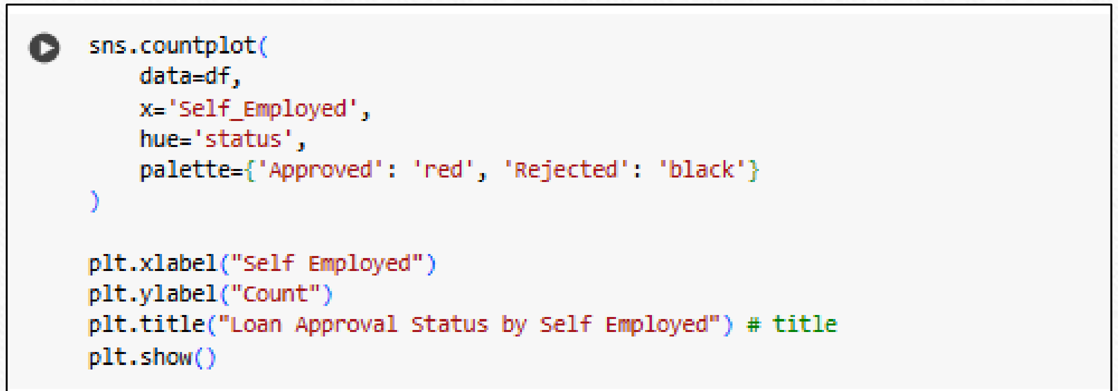


```
sns.countplot(
    data=df,
    x='Education',
    hue='status',
    palette={'Approved': 'red', 'Rejected': 'black'}
)

plt.xlabel("Education")
plt.ylabel("Count")
plt.title("Loan Approval Status by Education") # title
plt.show()
```

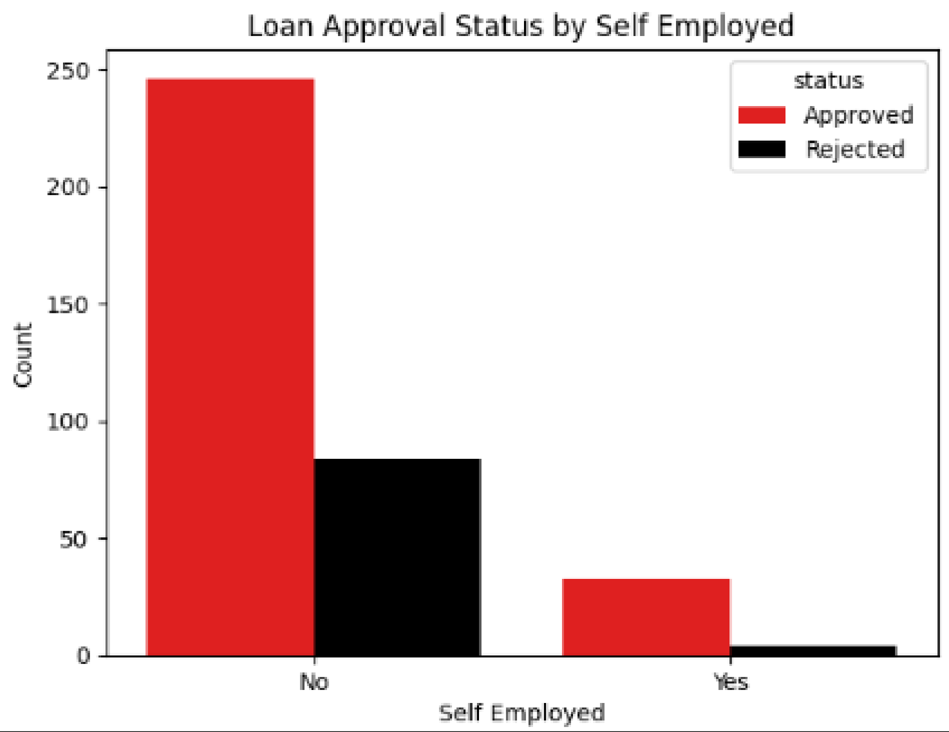


# Barplot of "Education"

- ❑ The above graph reveals the following insights:
- ✓ Graduates tend to apply for loans more frequently than non-graduates.
- ✓ They also have a higher loan approval rate compared to non-graduates.

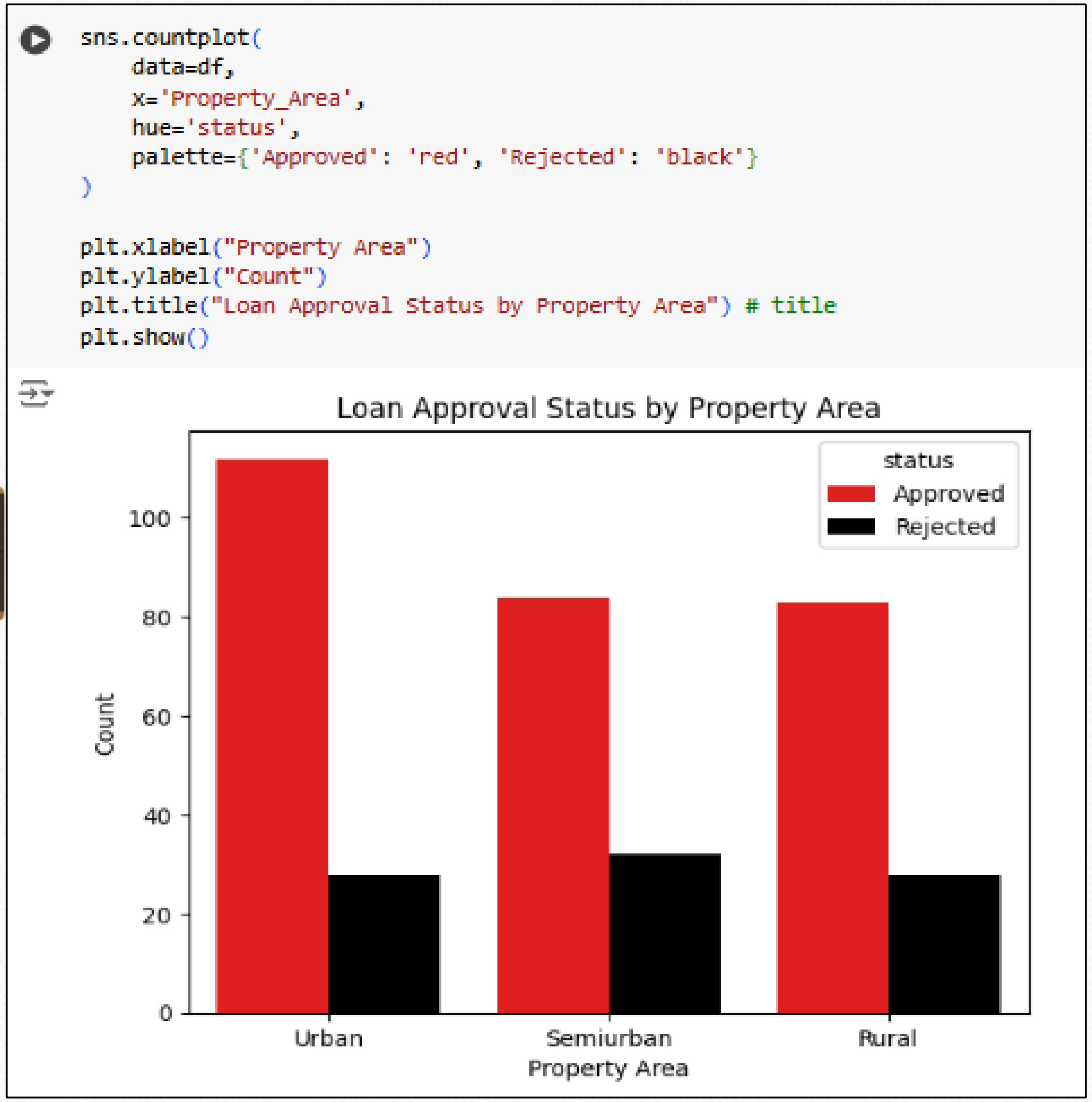


```
sns.countplot(
    data=df,
    x='Self_Employed',
    hue='status',
    palette={'Approved': 'red', 'Rejected': 'black'}
)

plt.xlabel("Self Employed")
plt.ylabel("Count")
plt.title("Loan Approval Status by Self Employed") # title
plt.show()
```



# Barplot of "self Employed"

- ❑ The above graph reveals the following insights:
- ✓ Most loan applicants are not self-employed.
- ✓ Loan approval rates are comparable for self-employed and non-self-employed individuals.



```
sns.countplot(
    data=df,
    x='Property_Area',
    hue='status',
    palette={'Approved': 'red', 'Rejected': 'black'}
)

plt.xlabel("Property Area")
plt.ylabel("Count")
plt.title("Loan Approval Status by Property Area") # title
plt.show()
```

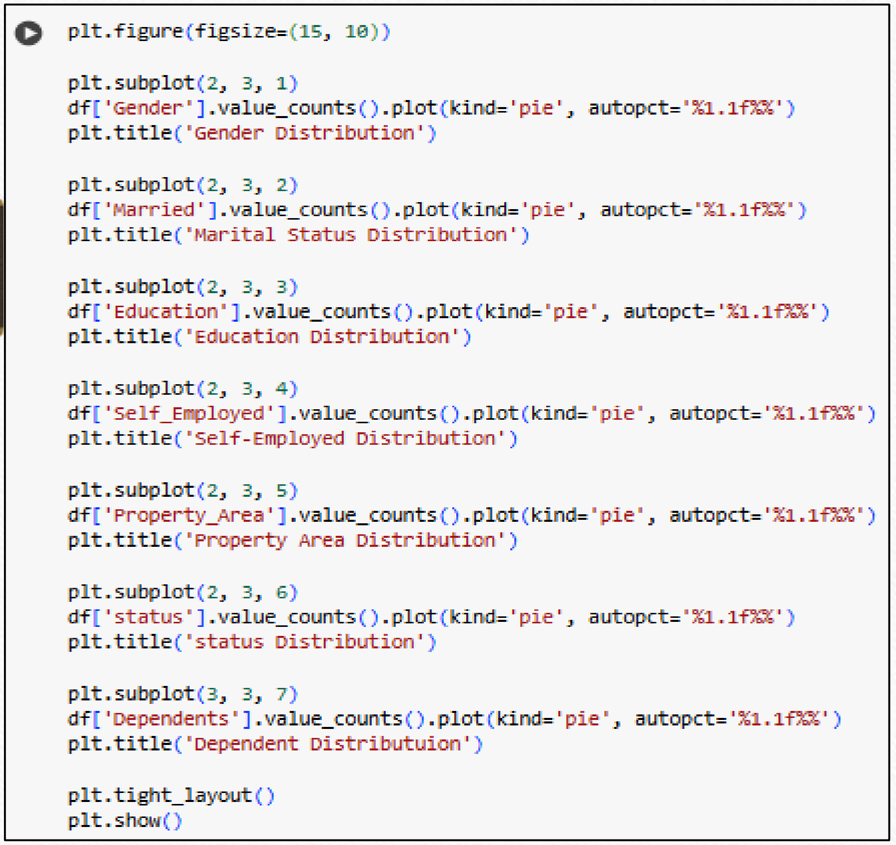
# Barplot of "Property Area"

- ❑ The above graph reveals the following insights:

- ✓ The majority of loan applications originate from semi-urban areas, followed by urban and rural regions.
- ✓ Semi-urban areas have the highest loan approval rates, with urban areas next and rural areas the lowest.

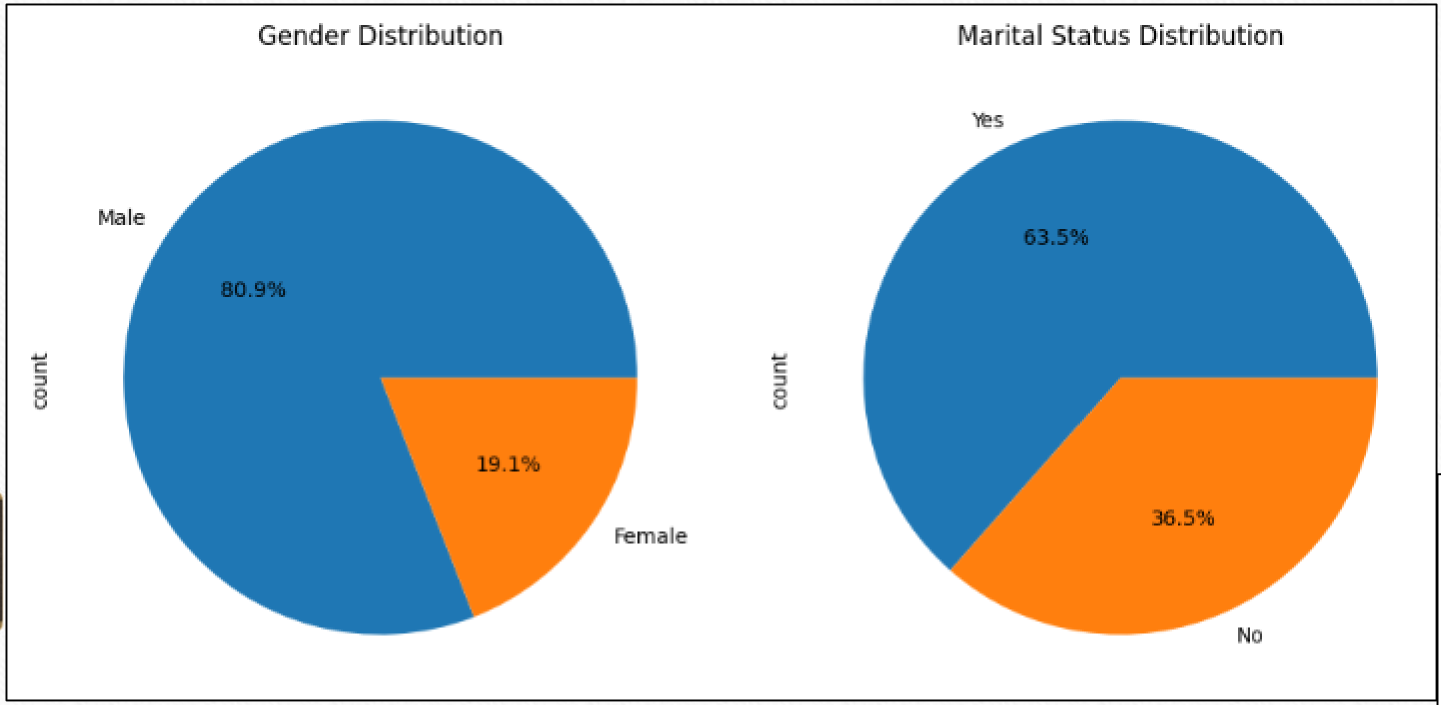
❖ Pie Charts: Visualizing the Distribution of Categorical Variables

- The pie charts visually depict the proportions of each category within a variable.



```
plt.figure(figsize=(15, 10))

plt.subplot(2, 3, 1)
df['Gender'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Gender Distribution')

plt.subplot(2, 3, 2)
df['Married'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Marital Status Distribution')

plt.subplot(2, 3, 3)
df['Education'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Education Distribution')

plt.subplot(2, 3, 4)
df['Self_Employed'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Self-Employed Distribution')

plt.subplot(2, 3, 5)
df['Property_Area'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Property Area Distribution')

plt.subplot(2, 3, 6)
df['status'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('status Distribution')

plt.subplot(3, 3, 7)
df['Dependents'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Dependent Distributuion')

plt.tight_layout()
plt.show()
```

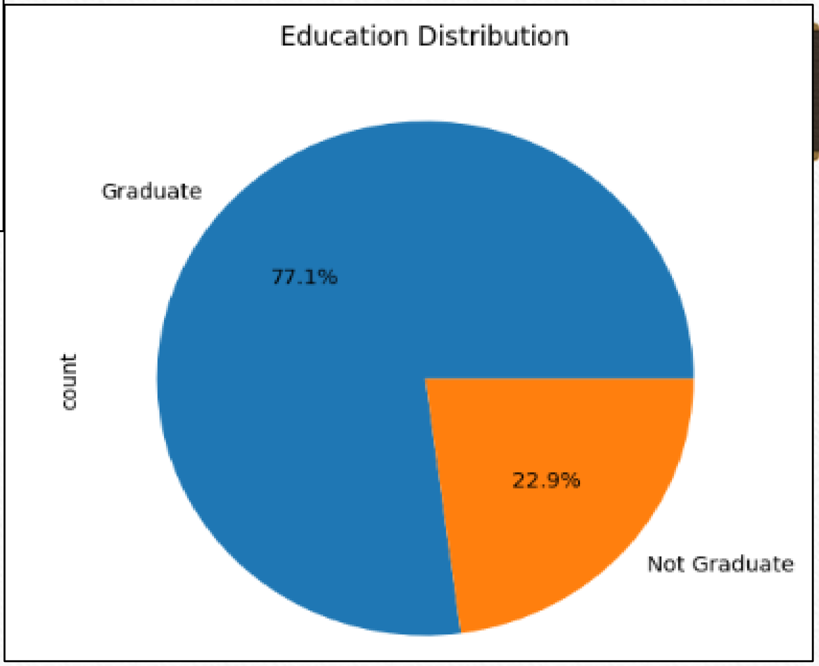
✓ This code is to draw pie chart for all the columns to show proportion of each catogary.

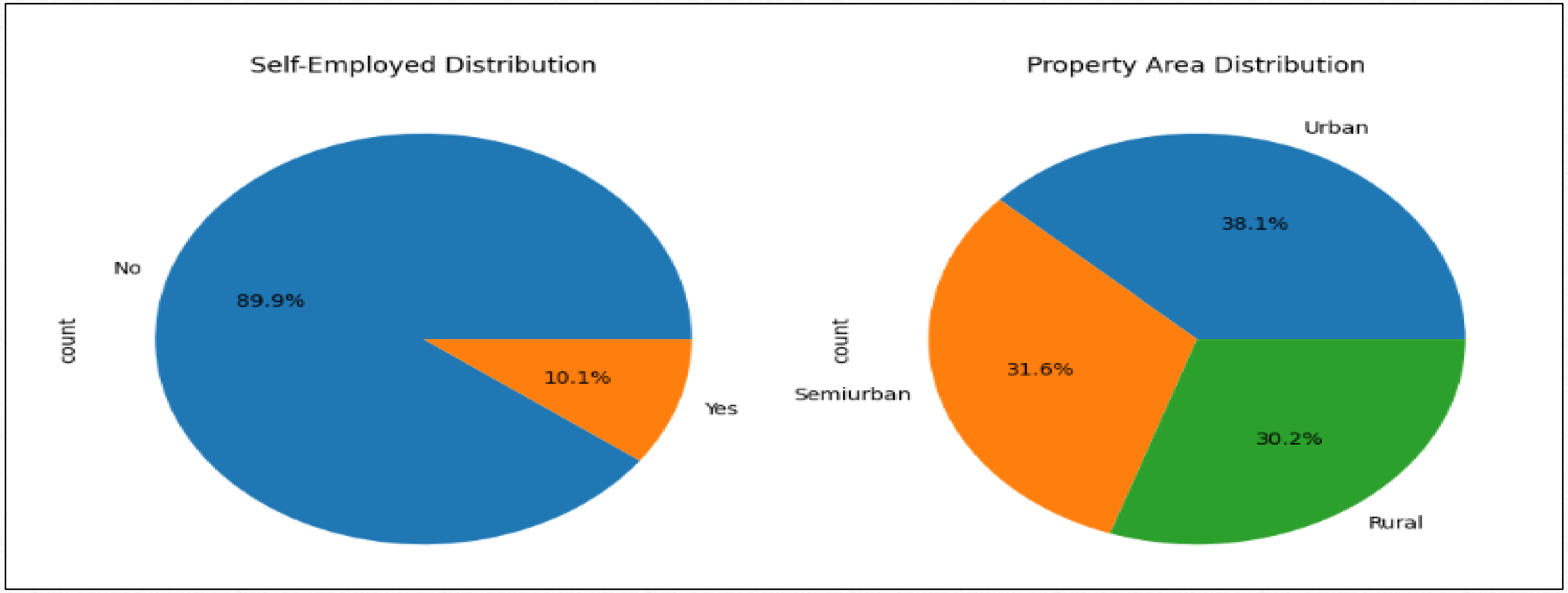


# piechart of “Gender”

piechart of “Maritial status”

# piechart of “Education”

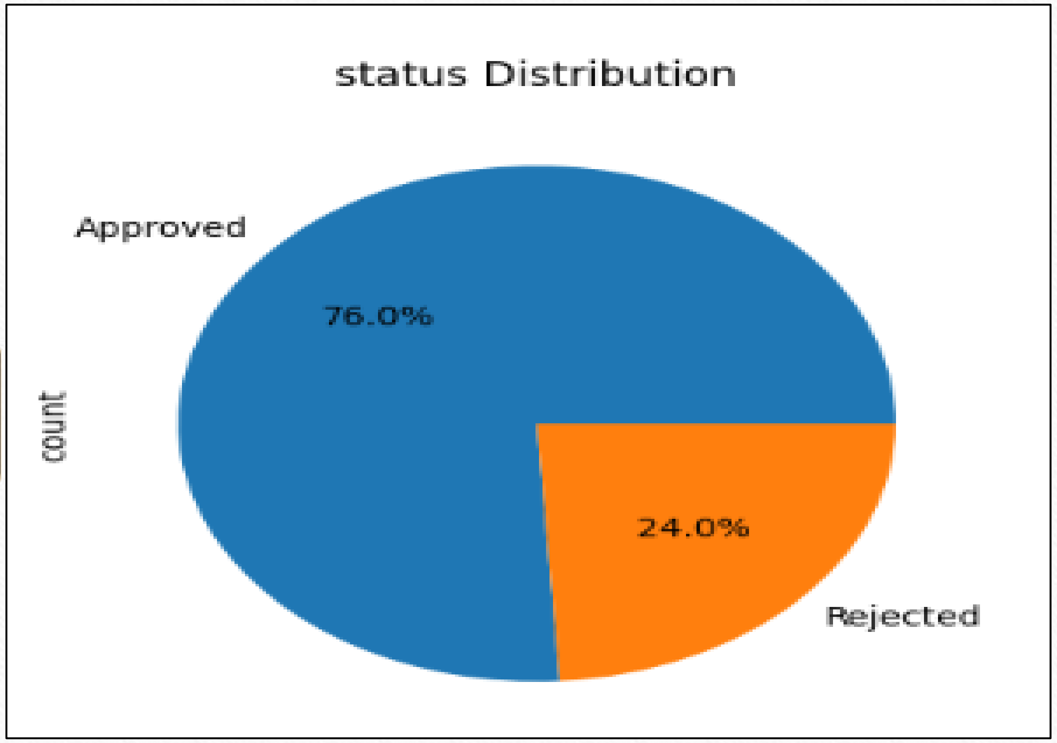




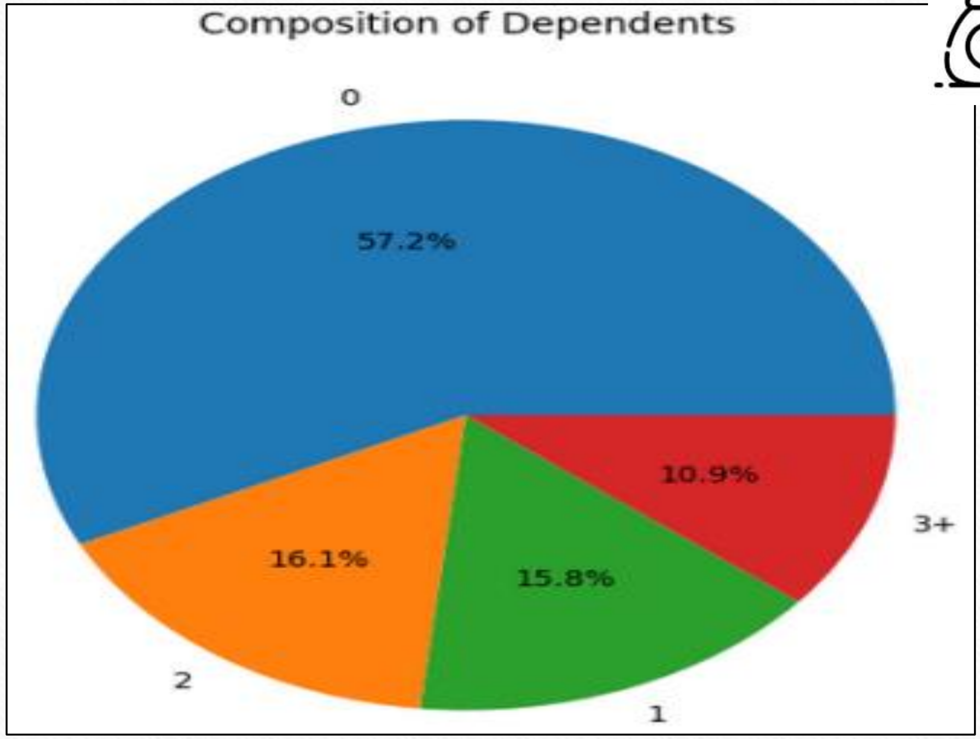
# piechart of "self Employes"

piechart of "Property Area"

# piechart of “status”



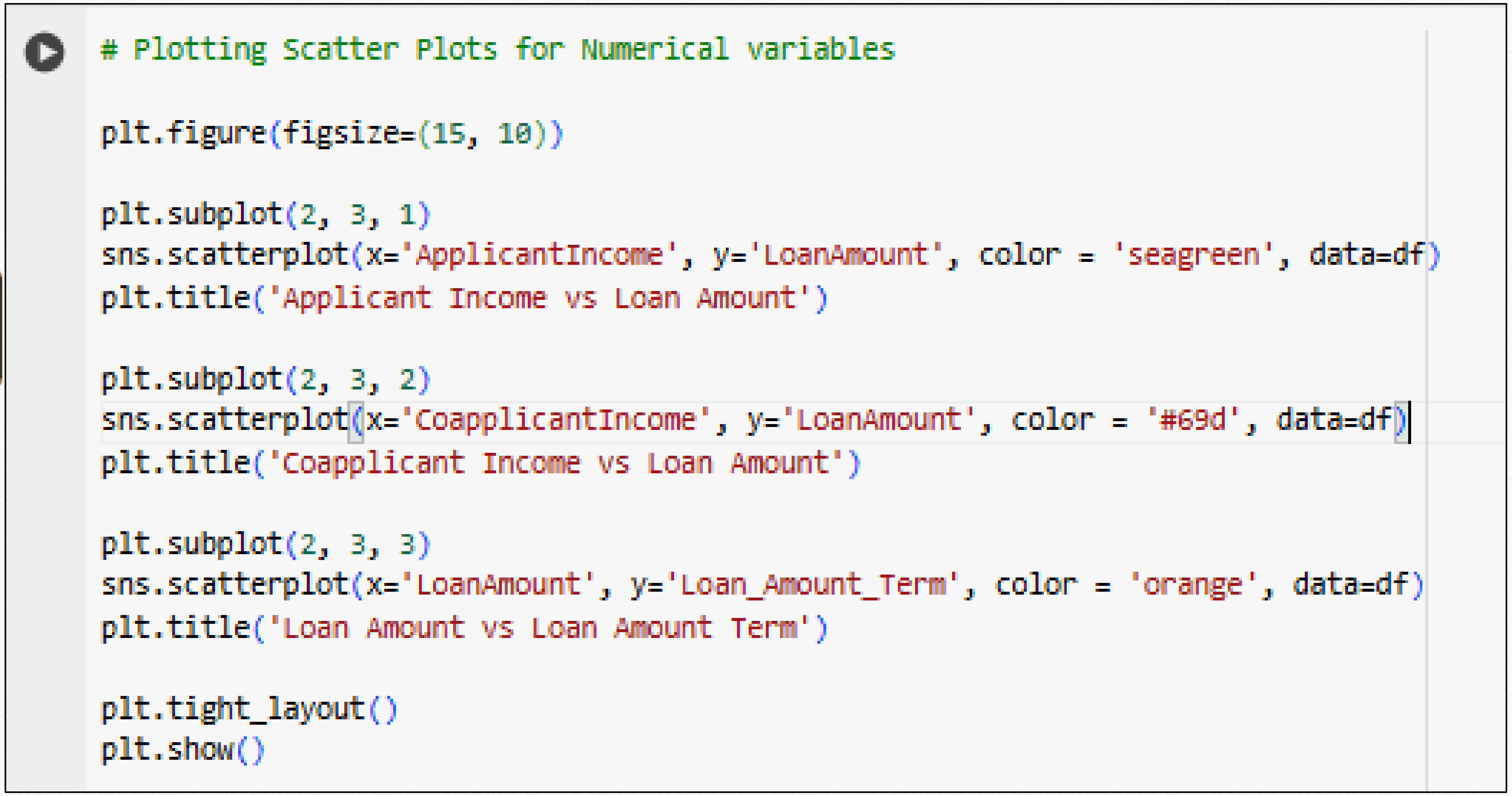
piechart of “Dependents”



- ✓ **Gender Distribution:** Approximately 81% of loan applicants are male, reflecting a notable gender disparity in loan applications.

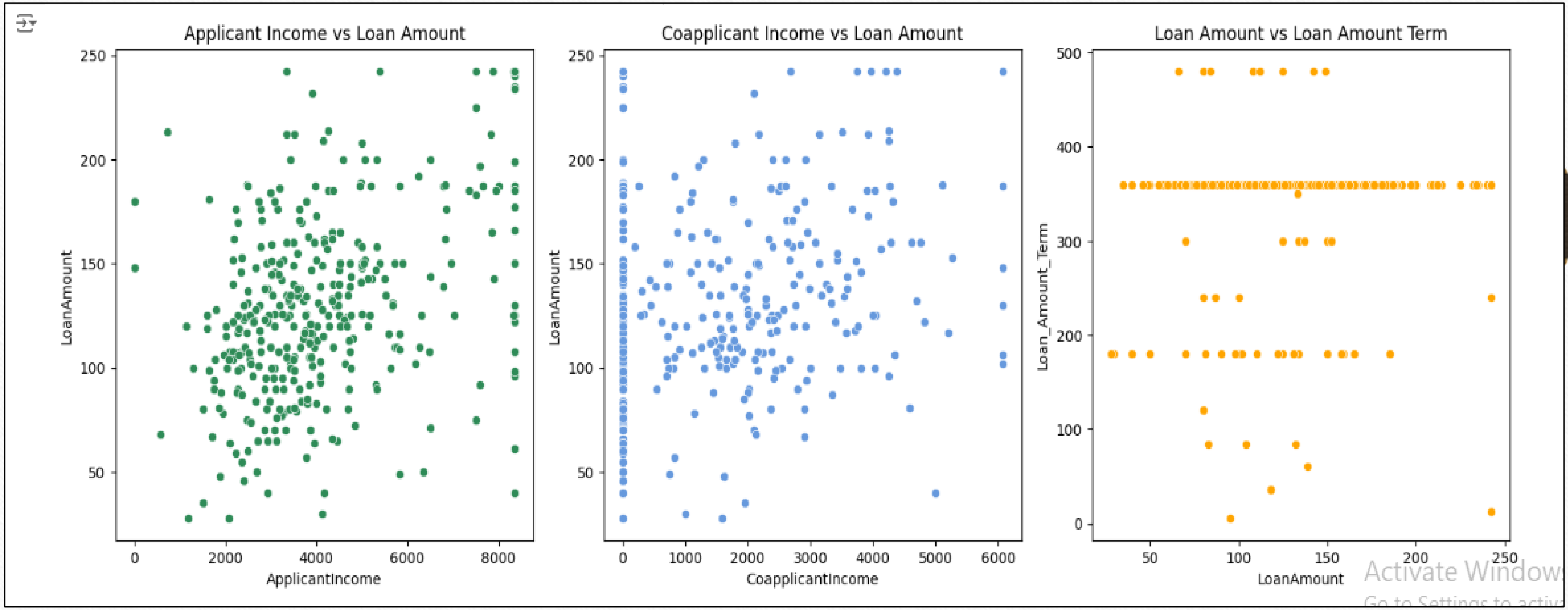
- ✓ **Marital Status Distribution:** Around 65% of applicants are married, indicating that married individuals are more likely to apply for loans.
- ✓ **Education Distribution:** About 78% of applicants are graduates, suggesting that individuals with higher education levels are more inclined to apply for loans.
- ✓ **Self-Employed Distribution:** Only 14% of applicants are self-employed, showing that most loan applications come from salaried individuals.
- ✓ **Property Area Distribution:** Applicants are fairly evenly distributed across property areas (Semiurban, Urban, Rural), with semiurban areas having a slightly higher share at approximately 38%.
- ✓ **Status:** 76% applicant got Approved meanwhile 25% applicant got rejected.
- ✓ **Dependents:** The pie chart shows the composition of dependents, with the majority (57.2%) having no dependents, followed by those with 2 dependents (16.1%), 1 dependent (15.8%), and 3 or more dependents (10.9%).

❖ Creating scatter plots to explore relationships between pairs of numeric variables.



```
# Plotting Scatter Plots for Numerical variables

plt.figure(figsize=(15, 10))

plt.subplot(2, 3, 1)
sns.scatterplot(x='ApplicantIncome', y='LoanAmount', color = 'seagreen', data=df)
plt.title('Applicant Income vs Loan Amount')

plt.subplot(2, 3, 2)
sns.scatterplot(x='CoapplicantIncome', y='LoanAmount', color = '#69d', data=df)
plt.title('Coapplicant Income vs Loan Amount')

plt.subplot(2, 3, 3)
sns.scatterplot(x='LoanAmount', y='Loan_Amount_Term', color = 'orange', data=df)
plt.title('Loan Amount vs Loan Amount Term')

plt.tight_layout()
plt.show()
```

# scatterplot of different columns vs each other



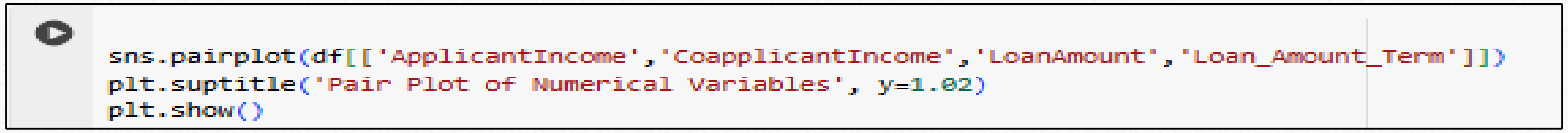
Key Features include:

•**Applicant Income vs. Loan Amount:** There is a slight positive correlation, indicating that higher-income applicants generally request larger loans. However, the correlation is weak, suggesting that other factors also influence the loan amount.

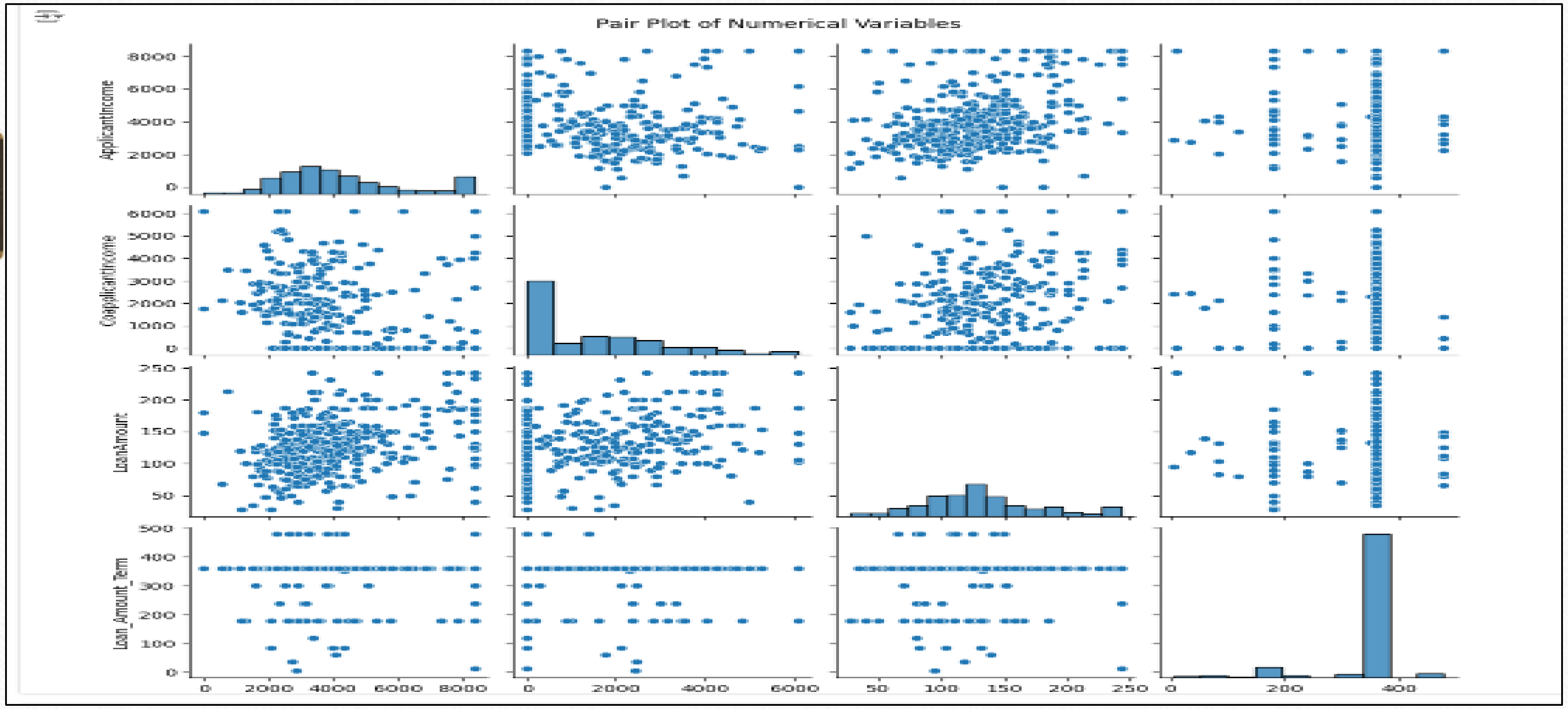
•**Co-applicant Income vs. Loan Amount:** The correlation is even weaker compared to applicant income, indicating that co-applicant income has a lesser impact on determining the loan amount.

•**Loan Amount vs. Loan Term:** No significant correlation is observed, suggesting that loan term and loan amount are largely independent and that loan terms are influenced by factors other than the loan amount.

❖ Utilize pair plots (scatter matrix) to examine the interactions and relationships between multiple numeric variables simultaneously.



```
sns.pairplot(df[['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term']])
plt.suptitle('Pair Plot of Numerical Variables', y=1.02)
plt.show()
```



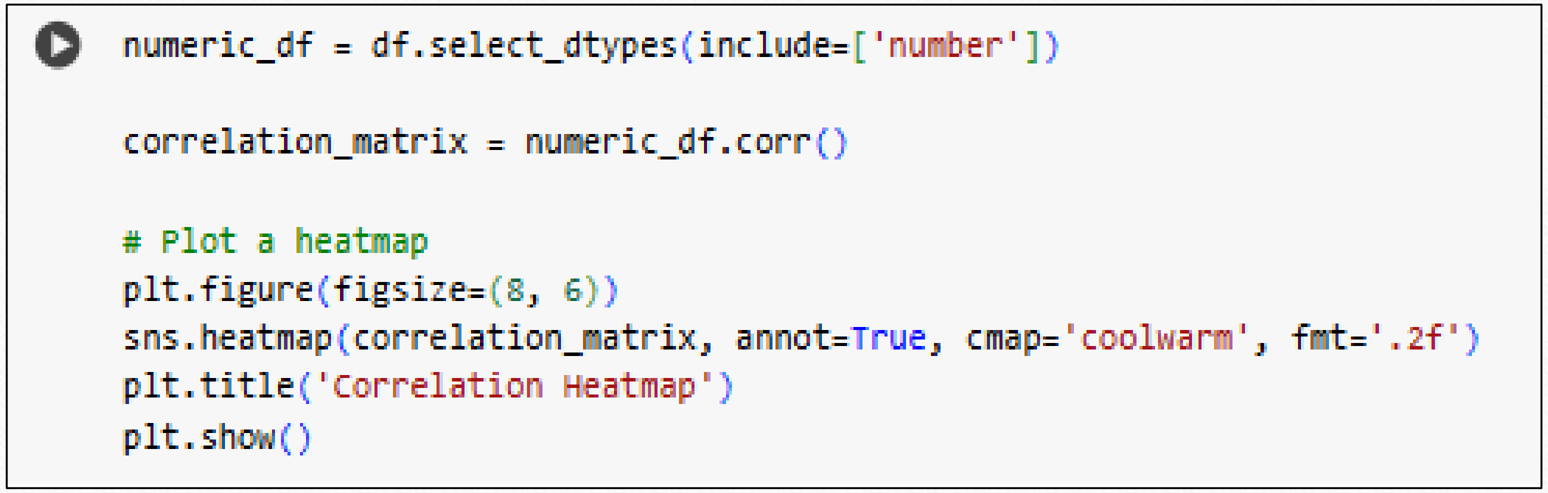
Column 1: ApplicantIncome
- Distribution: Right-skewed, indicating most applicants have lower incomes.
- Correlation: Weak positive with CoapplicantIncome.
- Correlation: Moderate positive with LoanAmount.
- No strong relationship with Loan_Amount_Term.

Column 2: CoapplicantIncome
- Distribution: Right-skewed, similar to ApplicantIncome.
- Correlation: Weak positive with ApplicantIncome.
- No strong relationship with LoanAmount.
- No strong relationship with Loan_Amount_Term.

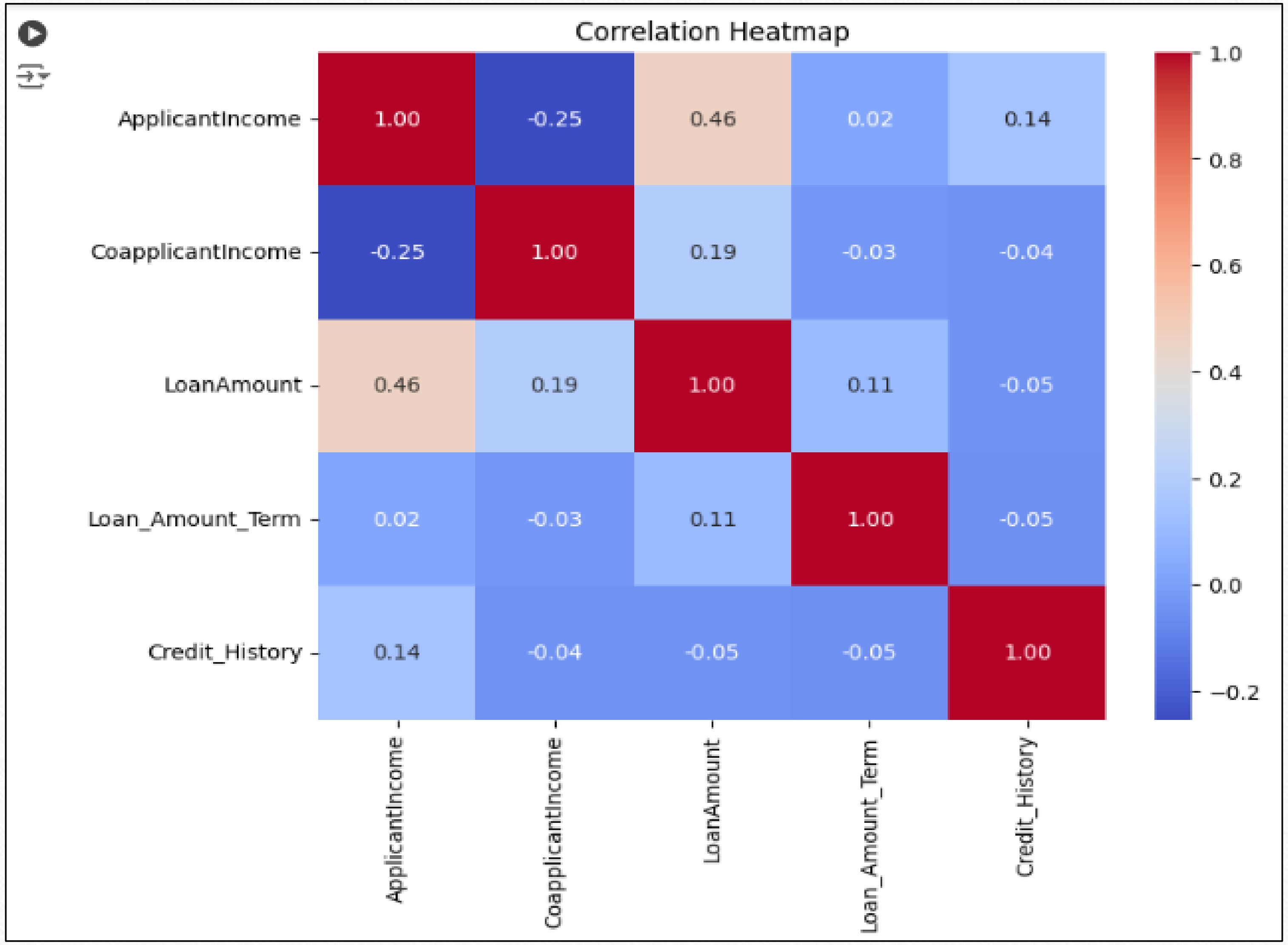
Column 3: LoanAmount
- Distribution: Right-skewed, most loans are smaller amounts.
- Correlation: Moderate positive with ApplicantIncome.
- No strong relationship with CoapplicantIncome.
- No strong relationship with Loan_Amount_Term.

Column 4: Loan_Amount_Term
- Distribution: Concentrated around a specific value (likely the most common term).
- No strong relationship with ApplicantIncome.
- No strong relationship with CoapplicantIncome.
- No strong relationship with LoanAmount.

❖ Conduct a correlation analysis to uncover relationships between numeric variables and display the correlations using a heatmap.



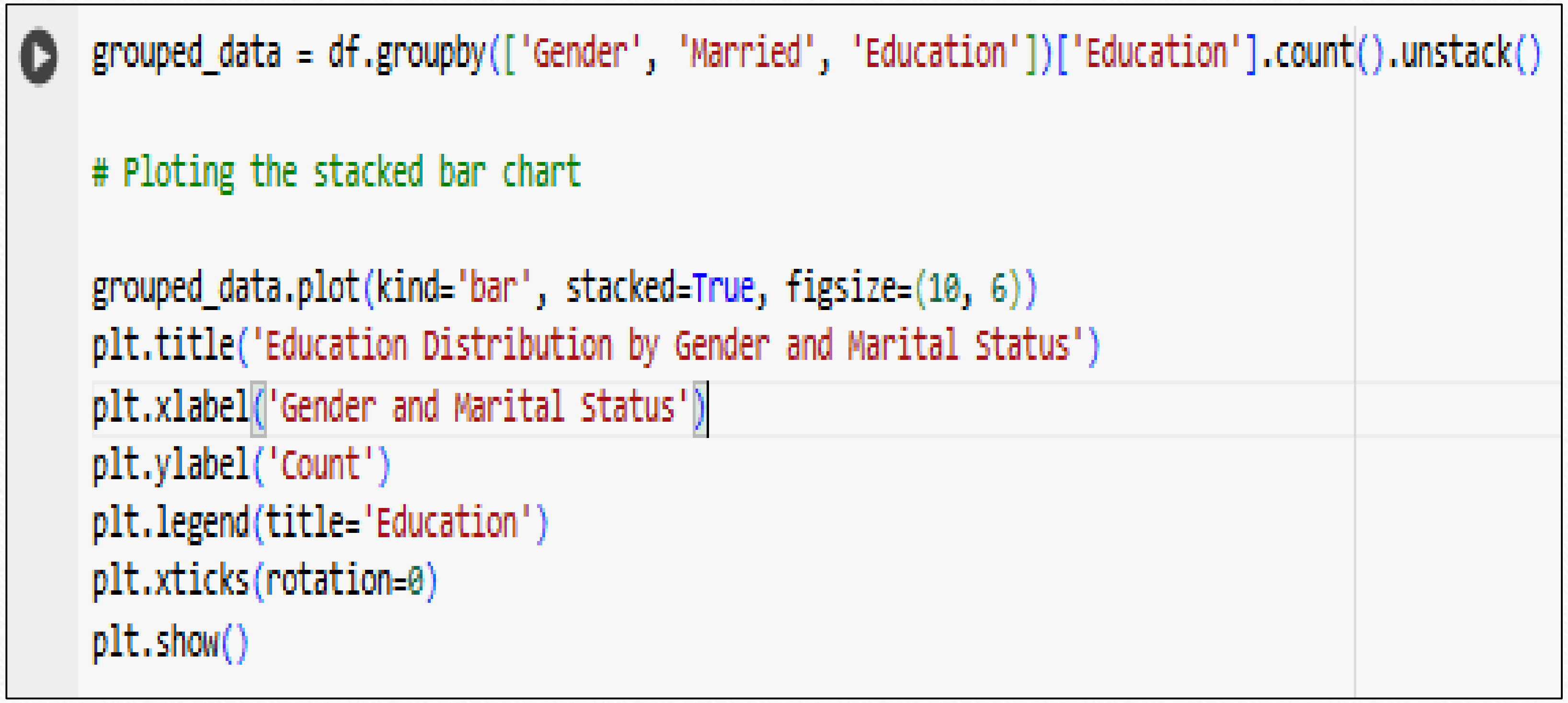
```python
numeric_df = df.select_dtypes(include=['number'])

correlation_matrix = numeric_df.corr()

# Plot a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```



	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
ApplicantIncome	1.00	-0.25	0.46	0.02	0.14
CoapplicantIncome	-0.25	1.00	0.19	-0.03	-0.04
LoanAmount	0.46	0.19	1.00	0.11	-0.05
Loan_Amount_Term	0.02	-0.03	0.11	1.00	-0.05
Credit_History	0.14	-0.04	-0.05	-0.05	1.00

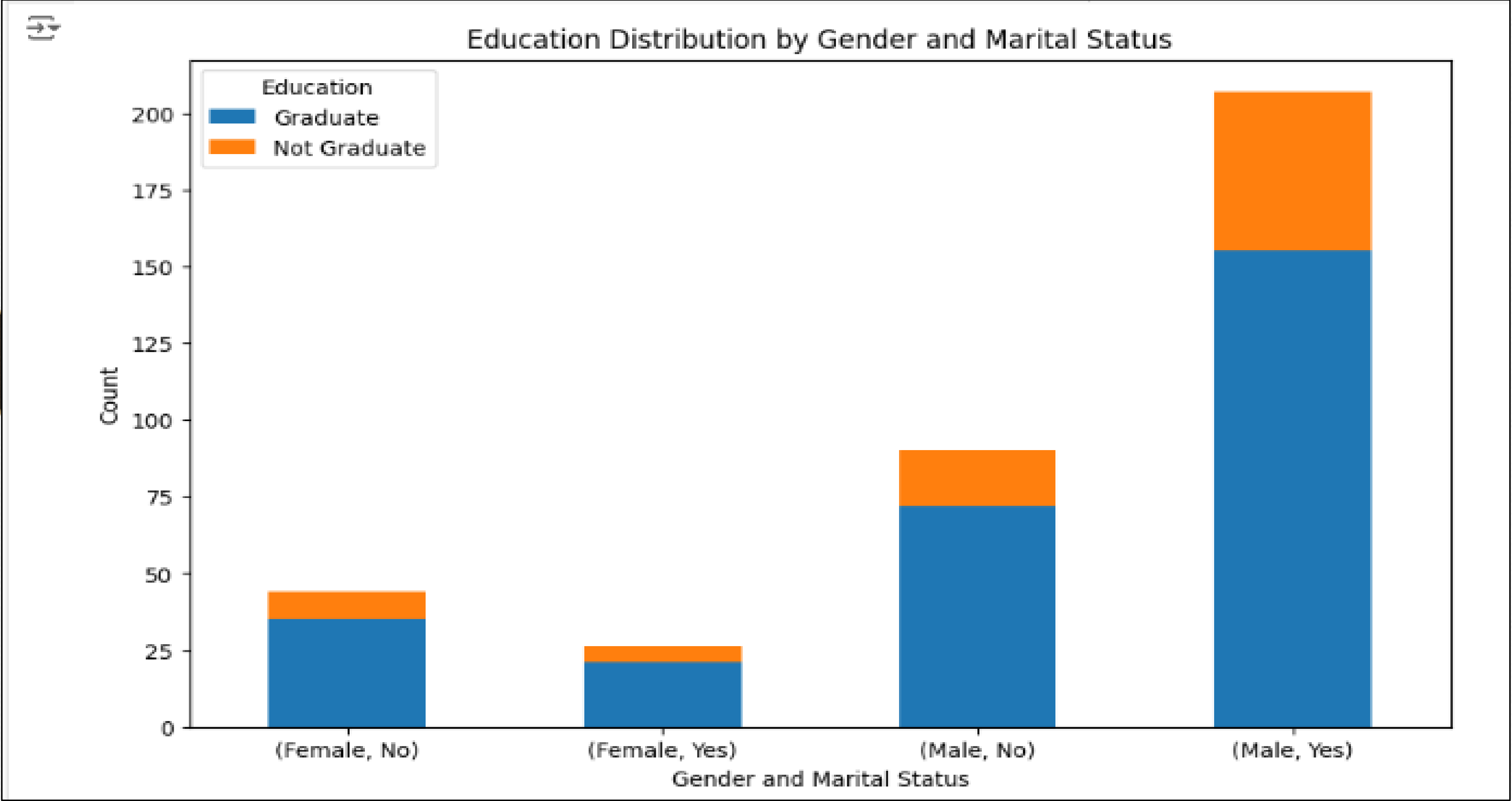
# Correlation Heatmap

❖ Create a stacked bar chart to show the distribution of categorical variables across multiple categories.



```python
grouped_data = df.groupby(['Gender', 'Married', 'Education'])['Education'].count().unstack()

# Ploting the stacked bar chart

grouped_data.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Education Distribution by Gender and Marital Status')
plt.xlabel('Gender and Marital Status')
plt.ylabel('Count')
plt.legend(title='Education')
plt.xticks(rotation=0)
plt.show()
```

# Barplot 3 multiple column after groupby each other



Key Features include:

•**Education and Gender:** Males, both married and unmarried, are more likely to have a graduate degree than females. This difference is more noticeable among married individuals.

•**Education and Marital Status:** Across both genders, married individuals are more likely to have a graduate degree compared to their unmarried counterparts. This suggests a potential link between higher education and the likelihood of marriage.

•**Overall Trend:** A majority of loan applicants, regardless of gender or marital status, hold a graduate degree. This indicates that higher education may be a common characteristic among those applying for loans.

•These insights can be explored further to assess how education influences loan approval rates and other related factors.

KEY FINDINGS:
Distribution of Numeric Variables:

•Both Applicant Income and Co-applicant Income are right-skewed, showing that most applicants have lower incomes, with a few high earners.

•Loan Amount follows a relatively normal distribution, although it includes some outliers,

•Loan Amount Term is mostly concentrated at 360 months.

•Credit History is negatively skewed, meaning the majority of applicants have a positive credit history.

•The Dependents distribution reveals a larger proportion of applicants with no dependents.

Categorical Variable Analysis:

•A majority of applicants are male and married.

•Most applicants are graduates and not self-employed.

•The distribution of applicants across property areas is fairly balanced between urban, semiurban, and rural locations

Relationships between Variables:

•A positive correlation exists between Applicant Income and Loan Amount, indicating that applicants with higher incomes typically request larger loans.

•A weak positive correlation is observed between Co-applicant Income and Loan Amount.

•No significant linear relationship is found between Loan Amount and Loan Term.

•Box plots show variations in Loan Amount distribution across different categories, including Gender, Marital Status, Education, Self Employment, and Property Area.

Conclusions :

Income and Loan Amount: Applicant income is a key factor in determining the loan amount.

Demographic Factors: Gender, marital status, education, and employment status impact the characteristics of loan applications.

Credit History: Most applicants have a positive credit history.

Recommendations:

•**Targeted Marketing:** Customize loan products and marketing efforts based on income levels and demographic traits.

•**Risk Assessment:** Factor in income, credit history, and other variables when evaluating loan approvals and assessing risk.

•**Product Diversification:** Develop loan offerings with different terms and amounts to meet the diverse needs of customers.

Thank You

Made by Lucky Bisht
Gmail : Bishtlucky543@gmail.com