By - Dhruba Jyoti Chakraborty

Project Index: Loan Default Analysis

1. Introduction

1.1. **Project Overview**
1.2. **Objectives and Goals**
1.3. **Importance of Risk Analytics in Banking**
1.4. **Scope of the Project**

2. Data Understanding

2.1. **Dataset Description**
2.2. **Initial Data Exploration**
2.3. **Data Dictionary and Variable Definitions**
2.4. **Identifying Missing Values and Data Types**

3. Data Preprocessing

3.1. **Handling Missing Values**
3.2. **Data Type Conversion**
3.3. **Handling Outliers**
3.4. **Data Normalization and Transformation (if necessary)**
3.5. **Encoding Categorical Variables**
3.6. **Creating Derived Features** (e.g., loan-to-value ratio, debt-to-income ratio)

4. Exploratory Data Analysis (EDA)

4.1. **Univariate Analysis**
  4.1.1. Distribution of Loan Amount, Interest Rates, Credit Scores
  4.1.2. Analysis of Loan Purpose, Loan Type, and Occupancy Type etc.
  4.1.3. Distribution and Impact of Newly Derived Features
4.2. **Bivariate Analysis**
  4.2.1. Relationship between Loan Amount and Default
  4.2.2. Impact of Credit Score on Default Probability
  4.2.3. Correlation between Upfront Charges and Default
  4.2.4. Analysis of Derived Features with Default Rates
4.3. **Multivariate Analysis**
  4.3.1. Analyzing Patterns in Loan Type, Purpose, and Default
  4.3.2. Interaction between Interest Rates, Property Value, and Default
4.4. **Visualizations**

In [ ]:

1. Introduction

1.1. Project Overview

This project aims to analyze loan default patterns in the banking and financial services sector. It focuses on identifying key factors contributing to loan defaults and understanding the risks involved in lending decisions. By analyzing various financial and demographic variables, the project seeks to provide insights that can enhance risk assessment strategies and help mitigate potential losses.

1.2. Objectives and Goals

- **Objective 1**: Analyze the impact of various factors such as loan type, loan purpose, business or commercial nature, and credit score on loan defaults.
- **Objective 2**: Examine the correlation between variables like upfront charges, loan amount, interest rates, and property values with the likelihood of default.
- **Objective 3**: Provide actionable insights and recommendations to improve risk management and lending practices.

1.3. Importance of Risk Analytics in Banking

Risk analytics plays a critical role in the financial sector by helping institutions assess the potential risks associated with lending. By leveraging data analysis, banks can identify high-risk customers, anticipate default trends, and make more informed lending decisions. This not only minimizes financial losses but also ensures sustainable growth and stability in the financial system.

1.4. Scope of the Project

This project will cover the following aspects:

- Detailed analysis of factors influencing loan defaults.

- Exploratory data analysis to uncover patterns and trends.
- Risk analysis and insights based on various loan characteristics.
- Recommendations for improving lending practices.
- Focus on statistical and descriptive analysis.

In [ ]:

2. Data Understanding

2.1. Dataset Description

The dataset used in this project contains detailed information on various loans issued by a financial institution. It includes data on borrower demographics, loan characteristics, and financial metrics, along with the status of each loan (whether it defaulted or not). The columns are as follows:

- **ID**: Unique identifier for each loan record.
- **year**: The year in which the loan was issued.
- **loan_limit**: If the loan limit is fixed or variable cf-confirm/fixed, ncf- not confirm/not fixed
- **Gender**: Gender of the borrower.
- **loan_type**: Type of loan (e.g., FHA, VA, conventional).
- **loan_purpose**: Purpose of the loan (e.g., home purchase, refinancing).
- **business_or_commercial**: Whether the loan is for business or personal use.
- **loan_amount**: Amount of the loan.
- **rate_of_interest**: Interest rate applied to the loan.
- **Upfront_charges**: Initial charges or fees paid by the borrower.
- **property_value**: Estimated value of the property being financed.
- **occupancy_type**: Type of occupancy (e.g., owner-occupied, rental).
- **income**: Borrower's income.
- **credit_type**: Type of credit used.
- **Credit_Score**: Borrower's credit score.
- **co-applicant_credit_type**: Credit type of the co-applicant, if any.
- **age**: Borrower's age.
- **LTV**: Lifetime Value of the loan.
- **Region**: Geographic region of the borrower.
- **Status**: Loan status indicating whether the loan defaulted.

In [20]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

pd.set_option('display.float_format','{:.2f}'.format)
```

In [21]:
```
df = pd.read_csv(r"D:\Scaler\scaler_capstone\drive-download-20240914T140729Z-001\loan.csv")
```

In [22]:
```
df.head()
```

Out[22]:

	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_commercial	loan_amount	rate_of_interest	Upfront_charges
0	24890	2019	cf	Sex Not Available	type1	p1	nob/c	116500	NaN	NaN
1	24891	2019	cf	Male	type2	p1	b/c	206500	NaN	NaN
2	24892	2019	cf	Male	type1	p1	nob/c	406500	4.56	595.00
3	24893	2019	cf	Male	type1	p4	nob/c	456500	4.25	NaN
4	24894	2019	cf	Joint	type1	p1	nob/c	696500	4.00	0.00

In [23]:
```
df.shape
```

Out[23]:
```
(148670, 20)
```

We will derive two more data set one having only defaulters as defaulter_df and another with only non defaulters as non_defaulter_df

In [25]:
```
non_defaulter_df = df[df['Status']==0]
```

In [26]:
```
defaulter_df = df[df['Status']==1]
```

In [27]:
```
non_defaulter_df.shape
```

Out[27]: (112031, 20)

In [28]:
```
defaulter_df.shape
```

Out[28]: (36639, 20)

Observations:

- The data set df is highly imbalance if the target variable is 'status'.
- main dataset contains 148670 rows and 20 columns.
- non default dataset contains 112031 and 20 columns.
- Defaulter dataset contains 36639 and 20 columns.

In [ ]:
```

```

2.2. Initial Data Exploration

- **Overview of Data**: We will use functions like `.info()`, and `.describe()` to get a preliminary understanding of the dataset.
- **Summary Statistics**: Calculate mean, median, mode, and standard deviation for numerical columns to understand their distributions.
- **Data Distribution**: Visualize distributions using histograms and boxplots for features like loan amount, interest rates, and credit score.

In [31]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148670 entries, 0 to 148669
Data columns (total 20 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   ID                        148670 non-null  int64  
 1   year                      148670 non-null  int64  
 2   loan_limit                145326 non-null  object 
 3   Gender                    148670 non-null  object 
 4   loan_type                 148670 non-null  object 
 5   loan_purpose              148536 non-null  object 
 6   business_or_commercial    148670 non-null  object 
 7   loan_amount               148670 non-null  int64  
 8   rate_of_interest          112231 non-null  float64
 9   Upfront_charges           109028 non-null  float64
 10  property_value            133572 non-null  float64
 11  occupancy_type            148670 non-null  object 
 12  income                    139520 non-null  float64
 13  credit_type               148670 non-null  object 
 14  Credit_Score              148670 non-null  int64  
 15  co-applicant_credit_type  148670 non-null  object 
 16  age                       148470 non-null  object 
 17  LTV                       133572 non-null  float64
 18  Region                    148670 non-null  object 
 19  Status                    148670 non-null  int64  
dtypes: float64(5), int64(5), object(10)
memory usage: 22.7+ MB
```

In [32]:
```
# categories of loan type
df['loan_type'].unique()
```

Out[32]: array(['type1', 'type2', 'type3'], dtype=object)

In [33]:
```
# categories of loan purpose
df['loan_purpose'].unique()
```

Out[33]: array(['p1', 'p4', 'p3', 'p2', nan], dtype=object)

In [34]:
```
# categories of loan limit
df['loan_limit'].unique()
```

Out[34]: array(['cf', nan, 'ncf'], dtype=object)

In [35]:
```
# categories of gender
df['Gender'].unique()
```

Out[35]: array(['Sex Not Available', 'Male', 'Joint', 'Female'], dtype=object)

In [36]:
```
# categories of business_or_commercial
df['business_or_commercial'].unique()
```

Out[36]:
```
array(['nob/c', 'b/c'], dtype=object)
```

In [37]:
```
# categories of occupancy_type
df['occupancy_type'].unique()
```

Out[37]:
```
array(['pr', 'sr', 'ir'], dtype=object)
```

In [38]:
```
# categories of credit type
df['credit_type'].unique()
```

Out[38]:
```
array(['EXP', 'EQUI', 'CRIF', 'CIB'], dtype=object)
```

In [39]:
```
# categories of region
df['Region'].unique()
```

Out[39]:
```
array(['south', 'North', 'central', 'North-East'], dtype=object)
```

In [40]:
```
numeric_df = df.select_dtypes(exclude=['object'])
numeric_non_defaulter_df= non_defaulter_df.select_dtypes(exclude=['object'])
numeric_defaulter_df= defaulter_df.select_dtypes(exclude=['object'])
```

In [41]:
```
cols_df = numeric_df.columns
cols_non_defaulter_df = numeric_non_defaulter_df.columns
cols_defaulter_df = numeric_defaulter_df.columns
```

In [42]:
```
print(cols_df)
print(cols_non_defaulter_df)
print(cols_defaulter_df)
```

```
Index(['ID', 'year', 'loan_amount', 'rate_of_interest', 'Upfront_charges',
       'property_value', 'income', 'Credit_Score', 'LTV', 'Status'],
      dtype='object')
Index(['ID', 'year', 'loan_amount', 'rate_of_interest', 'Upfront_charges',
       'property_value', 'income', 'Credit_Score', 'LTV', 'Status'],
      dtype='object')
Index(['ID', 'year', 'loan_amount', 'rate_of_interest', 'Upfront_charges',
       'property_value', 'income', 'Credit_Score', 'LTV', 'Status'],
      dtype='object')
```

In [43]:
```
# number of numerical columns in the data set
len(cols_df)
```

Out[43]:
```
10
```

In [44]:
```
# number of categorical columns in the data set
df.shape[1]- len(cols_df)
```

Out[44]:
```
10
```

In [45]:
```
df.describe()
```

Out[45]:

	ID	year	loan_amount	rate_of_interest	Upfront_charges	property_value	income	Credit_Score	LTV	S
count	148670.00	148670.00	148670.00	112231.00	109028.00	133572.00	139520.00	148670.00	133572.00	1486
mean	99224.50	2019.00	331117.74	4.05	3225.00	497893.47	6957.34	699.79	72.75	
std	42917.48	0.00	183909.31	0.56	3251.12	359935.32	6496.59	115.88	39.97	
min	24890.00	2019.00	16500.00	0.00	0.00	8000.00	0.00	500.00	0.97	
25%	62057.25	2019.00	196500.00	3.62	581.49	268000.00	3720.00	599.00	60.47	
50%	99224.50	2019.00	296500.00	3.99	2596.45	418000.00	5760.00	699.00	75.14	
75%	136391.75	2019.00	436500.00	4.38	4812.50	628000.00	8520.00	800.00	86.18	
max	173559.00	2019.00	3576500.00	8.00	60000.00	16508000.00	578580.00	900.00	7831.25	

In [46]:
```
numeric_non_defaulter_df.describe()
```

Out[46]:

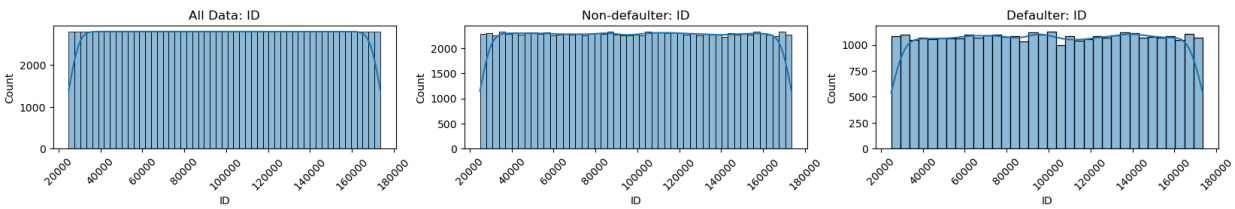
	ID	year	loan_amount	rate_of_interest	Upfront_charges	property_value	income	Credit_Score	LTV	S
count	112031.00	112031.00	112031.00	112031.00	108875.00	112029.00	104120.00	112031.00	112029.00	1120
mean	99182.70	2019.00	334990.77	4.04	3227.33	505606.07	7204.01	699.52	72.06	
std	42925.11	0.00	174916.57	0.56	3251.67	342784.46	6201.34	115.67	41.77	
min	24892.00	2019.00	26500.00	0.00	0.00	8000.00	0.00	500.00	2.07	
25%	61964.50	2019.00	206500.00	3.62	584.39	288000.00	4020.00	599.00	59.97	
50%	99222.00	2019.00	306500.00	3.99	2600.00	428000.00	6000.00	699.00	74.50	
75%	136306.00	2019.00	446500.00	4.38	4815.32	638000.00	8760.00	800.00	85.20	
max	173559.00	2019.00	3006500.00	8.00	60000.00	9268000.00	377220.00	900.00	7831.25	

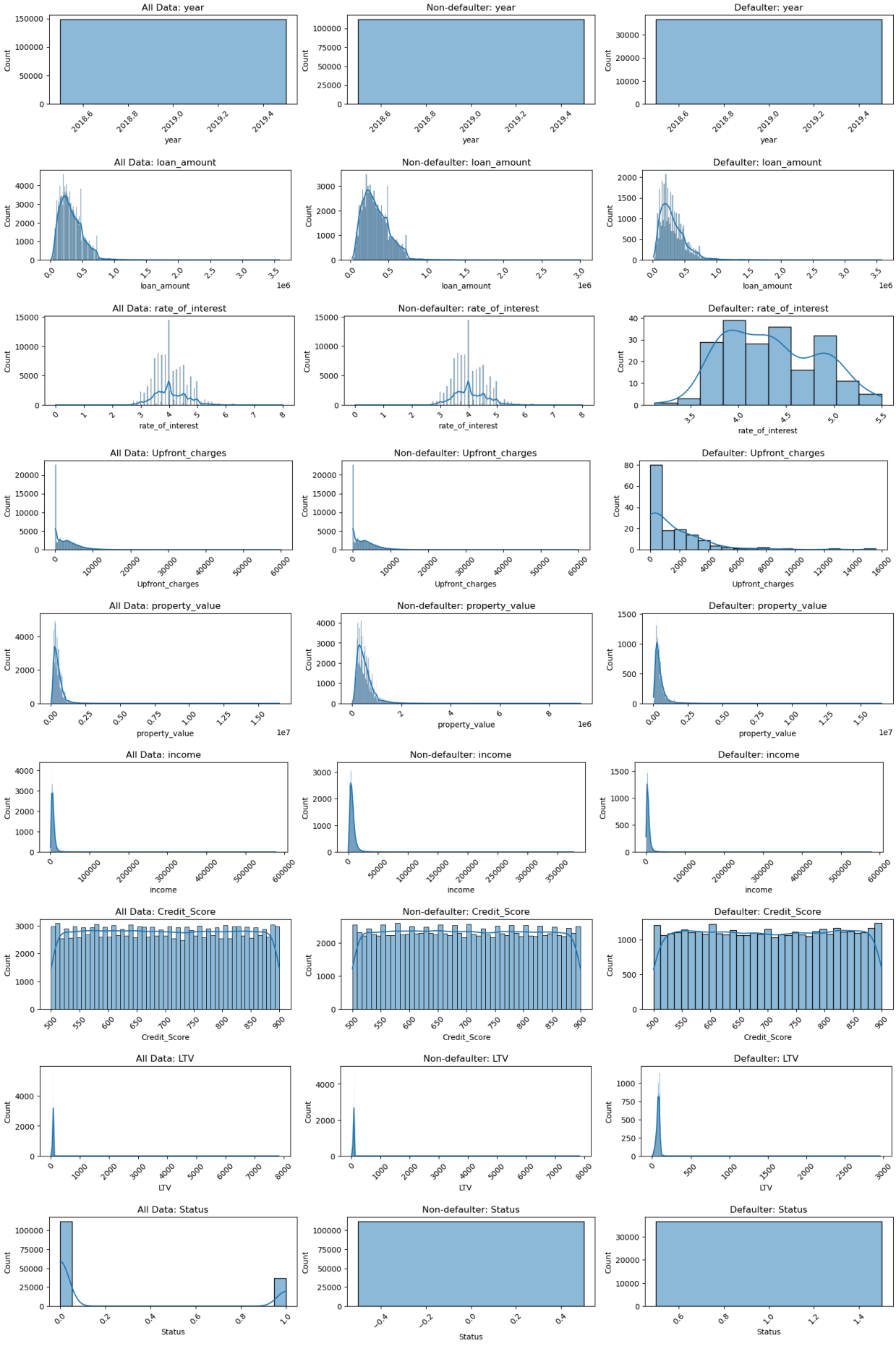
In [47]:
```
numeric_defaulter_df.describe()
```

Out[47]:

	ID	year	loan_amount	rate_of_interest	Upfront_charges	property_value	income	Credit_Score	LTV	Stat
count	36639.00	36639.00	36639.00	200.00	153.00	21543.00	35400.00	36639.00	21543.00	36639.0
mean	99352.31	2019.00	319275.18	4.35	1565.24	457786.01	6231.81	700.60	76.29	1.0
std	42894.46	0.00	208576.81	0.50	2299.82	436255.03	7247.65	116.49	28.58	0.0
min	24890.00	2019.00	16500.00	3.12	0.00	8000.00	0.00	500.00	0.97	1.0
25%	62304.00	2019.00	176500.00	3.99	0.00	228000.00	3000.00	599.50	64.27	1.0
50%	99233.00	2019.00	276500.00	4.31	687.85	348000.00	4860.00	700.00	79.36	1.0
75%	136669.00	2019.00	416500.00	4.75	2310.75	558000.00	7620.00	803.00	90.95	1.0
max	173553.00	2019.00	3576500.00	5.50	15584.00	16508000.00	578580.00	900.00	2956.25	1.0

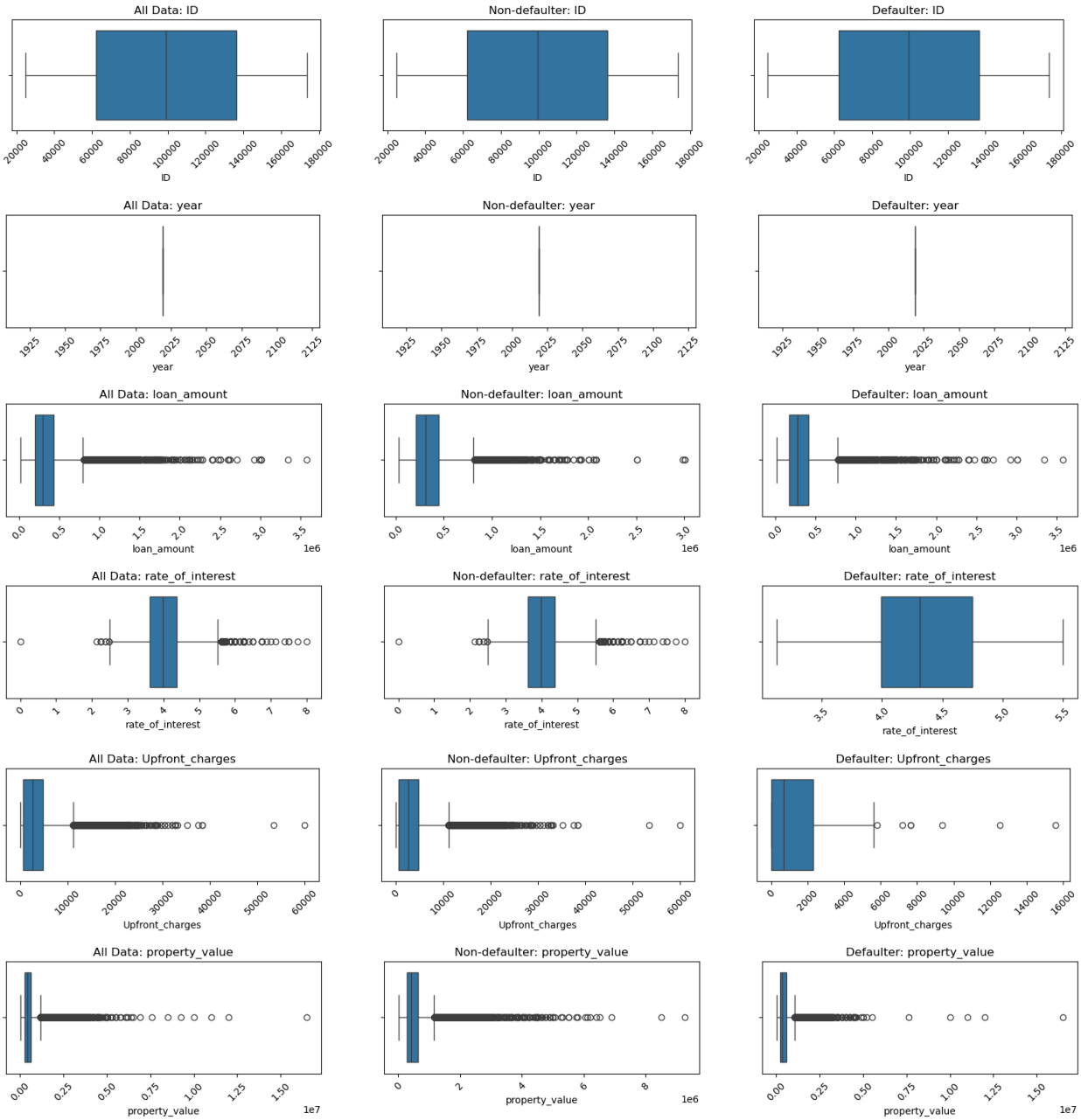
In [48]:
```
# Histogram plot for numeric columns
j=1
for i in cols_df:
    plt.figure(figsize=[20,25])
    # Create subplots for the same column from three different datasets side by side
    for y in range(j, j + 3):
        plt.subplot(10, 3, y)

        # Plot for the entire data
        if y == j:
            sns.histplot(data=numeric_df, x=i, kde=True)
            plt.xticks(rotation=45)
            plt.title(f"All Data: {i}")

        # Plot for non-defaulters
        elif y == j + 1:
            sns.histplot(data=numeric_non_defaulter_df, x=i, kde=True)
            plt.xticks(rotation=45)
            plt.title(f"Non-defaulter: {i}")

        # Plot for defaulters
        elif y == j + 2:
            sns.histplot(data=numeric_defaulter_df, x=i, kde=True)
            plt.xticks(rotation=45)
            plt.title(f"Defaulter: {i}")

    # Move j forward by 3 to handle the next set of subplots
    j += 3

    # Adjust layout and display the subplots for the current column
    plt.show()
```

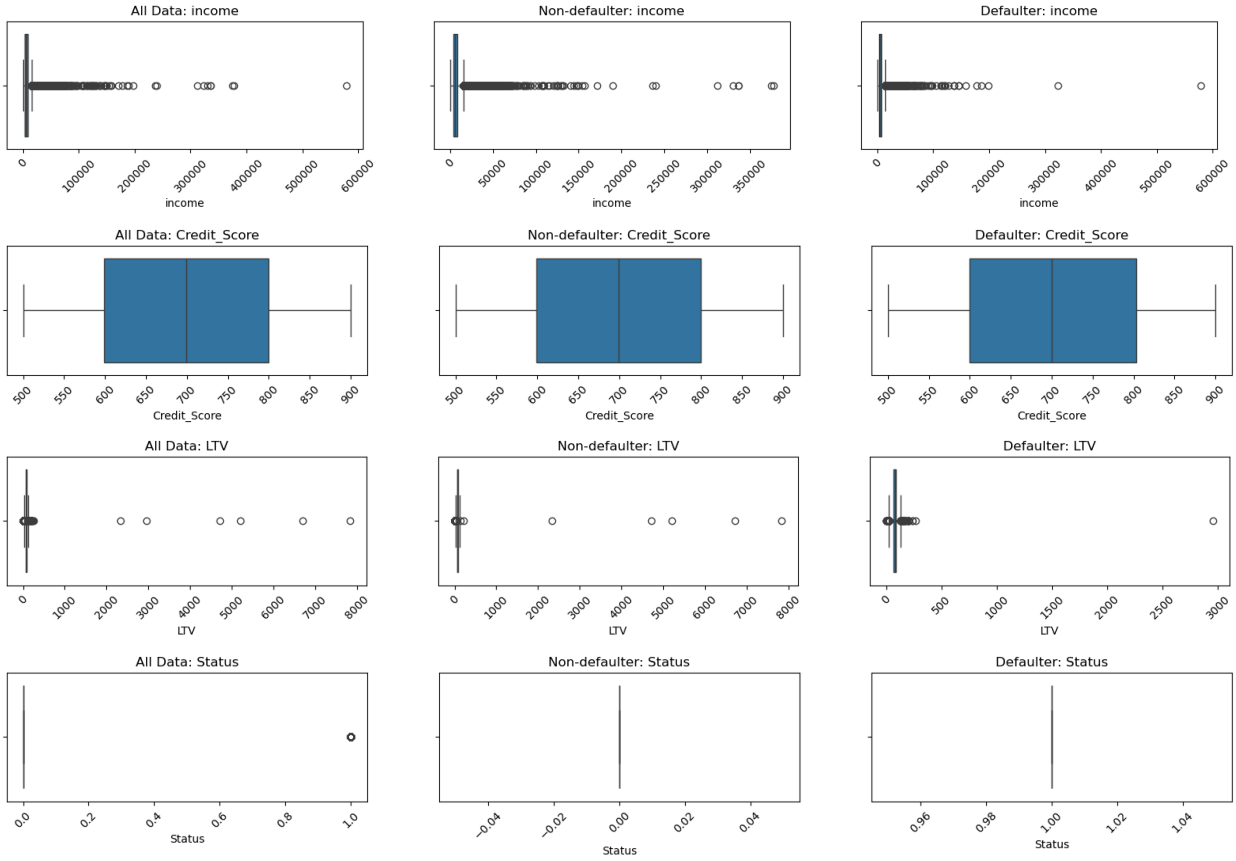




In [49]:
```
# Histogram plot for numeric columns
j=1
for i in cols_df:
    plt.figure(figsize=[20,25])
    # Create subplots for the same column from three different datasets side by side
    for y in range(j, j + 3):
```

```
        plt.subplot(10, 3, y)

        # Plot for the entire data
        if y == j:
            sns.boxplot(data=numeric_df, x=i)
            plt.xticks(rotation=45)
            plt.title(f"All Data: {i}")

        # Plot for non-defaulters
        elif y == j + 1:
            sns.boxplot(data=numeric_non_defaulter_df, x=i)
            plt.xticks(rotation=45)
            plt.title(f"Non-defaulter: {i}")

        # Plot for defaulters
        elif y == j + 2:
            sns.boxplot(data=numeric_defaulter_df, x=i)
            plt.xticks(rotation=45)
            plt.title(f"Defaulter: {i}")

    # Move j forward by 3 to handle the next set of subplots
    j += 3

    # Adjust layout and display the subplots for the current column
    plt.show()
```





Observations:

- **Numerical features**:
  - `ID` , `Year` , `loan_amount` , `rate_of_interest` , `upfront_charges` , `property_value` , `income` , `credit_score` , `LTV` , `status` .
- **Categorical features**:
  - `loan_limit` , `gender` , `loan_type` , `loan_purpose` , `business_or_commercial` , `occupancy_type` , `credit_type` , `co_applicant_type` , `age` , `region` .

Different Categorical Features:

- **loan_type**: 'type1', 'type2', 'type3'
- **loan_purpose**: 'p1', 'p4', 'p3', 'p2'
- **loan_limit**: 'cf', NaN, 'ncf'
- **gender**: 'Sex Not Available', 'Male', 'Joint', 'Female'
- **business_or_commercial**: 'nob/c', 'b/c'
- **occupancy_type**: 'pr', 'sr', 'ir'
- **credit_type**: 'EXP', 'EQUI', 'CRIF', 'CIB'
- **region**: 'South', 'North', 'Central', 'North-East'

Different Numerical Features:

- **ID**
- **Year**: Contains only 2019
- **loan_amount**:
  - Min: 16,500
  - Max: 3,576,500
  - Mean: 331,117
- **rate_of_interest**:
  - Min: 0
  - Max: 8
  - Mean: 4.05
- **upfront_charges**:
  - Min: 0

  - Max: 60,000
  - Mean: 3,225
- **property_value**:
  - Min: 8,000
  - Max: 16,508,000
  - Mean: 497,893
- **income**:
  - Min: 0
  - Max: 8,520
  - Mean: 699
- **credit_score**:
  - Min: 500
  - Max: 900
  - Mean: 699.79
- **LTV**:
  - Min: 0.97
  - Max: 7,831.25
  - Mean: 72.75

Skewed Features:

- The following features appear to be right-skewed:
  - `loan_amount`
  - `upfront_cion and, if required, we will perform imputation for those rows.

arges `-` property_value `-` income `-` LTV`

Missing Values and Next Steps:

- Some features have missing values. We will address these in the next section and, if required, we will perform imputation for those rows.

2.3. Data Dictionary and Variable Definitions

The following are key definitions for understanding the dataset variables:

- **Loan Type**: Type of loan (masked data) type-1, type-2, type-3
- **Loan Purpose**: Purpose of the loan (masked data) p1,p2,p3,p4
- **LTV (Lifetime Value)**: The projected revenue or value a financial institution expects from a loan over its entire duration.
- **Credit Score**: A numerical representation of the borrower's creditworthiness, typically ranging from 300 to 850.
- **Upfront Charges**: Initial fees or payments made during loan approval, including origination fees.

In [ ]:

2.4. Identifying Missing Values and Data Types

- **Missing Values Analysis**: Identify and quantify missing values in each column using functions like `.isnull().sum()`.
- **Data Types**: Verify and convert data types if necessary (e.g., ensuring numerical columns are not stored as strings).
- **Handling Missing Values**: Strategies for dealing with missing data, such as imputation using median or mode, or removing rows/columns with excessive missing values.

In [55]:
```
# number of duplicate rows in data set
df.duplicated().sum()
```

Out[55]:
```
0
```

In [56]:
```
# number null in each column
df.isnull().sum()
```

Out[56]:
```
ID                          0
year                        0
loan_limit               3344
Gender                      0
loan_type                   0
loan_purpose              134
business_or_commercial      0
loan_amount                 0
rate_of_interest        36439
Upfront_charges         39642
property_value          15098
occupancy_type              0
income                   9150
credit_type                 0
Credit_Score                0
co-applicant_credit_type    0
age                       200
LTV                     15098
Region                      0
Status                      0
dtype: int64
```

- There no issues with data type of the columns or features, so no need to change data type.

In [58]:
```
# numberof rows with zero rate of interest
len(df[df['rate_of_interest']==0])
```

Out[58]: 1

In [59]:
```
# number of rows with zero income
len(df[df['income']==0])
```

Out[59]: 1260

In [60]:
```
# percentage of missing values columns
null_df = df.isnull().sum().reset_index()
null_df.rename(columns={0:'Percentage_missing'},inplace=True)
null_df['Percentage_missing'] = round(null_df['Percentage_missing']/148670 *100,1)
null_df_sorted = null_df.sort_values(by='Percentage_missing',ascending=False).reset_index()
null_df_sorted
```

Out[60]:

	level_0	index	Percentage_missing
0	9	Upfront_charges	26.70
1	8	rate_of_interest	24.50
2	10	property_value	10.20
3	17	LTV	10.20
4	12	income	6.20
5	2	loan_limit	2.20
6	5	loan_purpose	0.10
7	16	age	0.10
8	18	Region	0.00
9	15	co-applicant_credit_type	0.00
10	14	Credit_Score	0.00
11	13	credit_type	0.00
12	0	ID	0.00
13	11	occupancy_type	0.00
14	1	year	0.00
15	7	loan_amount	0.00
16	6	business_or_commercial	0.00
17	4	loan_type	0.00
18	3	Gender	0.00
19	19	Status	0.00

Observations:

- We can see there many null values are there in many rows which we cannot delete as it may cause data lose.
    - Upfront charges have 26.7% missing values.
    - rate of interest have 24.5% missing values.
    - property_value have 10.2% missing values.
    - LTV have 10.2% missing values.
    - income have 6.2% missing values.
    - Loan limit have 2.2% missing values.
- We will see what are the best possibe ways to deal with it in next section.
- There is no duplicates rows.

In [ ]:

3. Data Preprocessing

Data preprocessing is a crucial step in preparing the dataset for effective analysis. It involves several sub-steps to clean and transform the data, ensuring it is in a suitable format for further analysis.

3.1. Handling Missing Values

- **Purpose:** Missing values can lead to biased results and inaccuracies. Proper handling is necessary to maintain data integrity.
- we have seen in **2.4** that some numerical and categorical columns have null/missing rows. We will replace the categorical featurs missing plae with **Not available** and for numerical columns we will use mean median methods.
- It is important to replace before we do hypothesis testing otherwise it may give us some baised results.
- For numerical columns we need to ensure the distribution of the data before and after the imputation remains same.
- We will also do **Kolmogorov-Smirnov (K-S) Test** to compare the entire distribution.
- **Methods:**
    - **Imputation:** Replace missing values with the mean, median, or mode for numerical columns. For categorical columns, fill it with **Not available**.

Note: **We can also use Random forest or Linear Regression model or KNN to fill this missing values, but we wont do that this time**

In [64]:
```
# Filling nan with Not available text in loan limit columns
df['loan_limit'] = df['loan_limit'].fillna(value='Not_available')
```

In [65]:
```
# Filling nan with Not available text in loan_purpose columns
df['loan_purpose'] = df['loan_purpose'].fillna(value='Not_available')
```

In [66]:
```
# Filling nan with Not available text in age columns
df['age'] = df['age'].fillna(value='Not_available')
```

In [67]:
```
# Creating empty dataFrame
rate_of_interest_df = pd.DataFrame()
Upfront_charges_df = pd.DataFrame()
property_value_df = pd.DataFrame()
income_df = pd.DataFrame()
LTV_df = pd.DataFrame()
```

In [68]:
```
rate_of_interest_df['rate_of_interest'] = df['rate_of_interest']
Upfront_charges_df['Upfront_charges'] = df['Upfront_charges']
property_value_df['property_value'] = df['property_value']
income_df['income'] = df['income']
LTV_df['LTV'] = df['LTV']
```

In [69]:
```
# filling the missing values with mean in rate of interest columns
rate_of_interest_df['rate_of_interest'] = rate_of_interest_df['rate_of_interest'].fillna(rate_of_interest_df['rate_of_inte

#replace zero interest with median
rate_of_interest_df['rate_of_interest'] = rate_of_interest_df['rate_of_interest'].replace(0,rate_of_interest_df['rate_of_i

# filling the missing values with median upfront charges columns
Upfront_charges_df['Upfront_charges'] = Upfront_charges_df['Upfront_charges'].fillna(Upfront_charges_df['Upfront_charges']

# filling the missing values with median inproperty value columns
property_value_df['property_value'] = property_value_df['property_value'].fillna(property_value_df['property_value'].media

# filling the missing values with median in income columns
income_df['income'] =income_df['income'].fillna(income_df['income'].median())

#replace zero income with median
income_df['income'] = income_df['income'].replace(0,income_df['income'].median())

# filling the missing values with median in LTV columns
LTV_df['LTV'] = LTV_df['LTV'].fillna(LTV_df['LTV'].median())
```

In [ ]:

- **Lets test the our distribution before and after imputation using KS test.**

In [71]:
```
# Ho ---> Distribution is same as before and after imputation.
# Ha ---> Distribution is not same as before and after imputation.
# alpha = 0.5
```

In [72]:
```
from scipy import stats
```

In [73]:
```
ks_stats, p_value_rate_of_interest = stats.ks_2samp(df['rate_of_interest'].dropna(),rate_of_interest_df['rate_of_interest'
ks_stats, p_value_Upfront_charges = stats.ks_2samp(df['Upfront_charges'].dropna(),Upfront_charges_df['Upfront_charges'])
ks_stats, p_value_property_value = stats.ks_2samp(df['property_value'].dropna(),property_value_df['property_value'])
ks_stats, p_value_income = stats.ks_2samp(df['income'].dropna(),income_df['income'])
ks_stats, p_value_LTV = stats.ks_2samp(df['LTV'].dropna(),LTV_df['LTV'])
```

In [74]:
```
p_value_rate_of_interest
```

Out[74]:
```
0.0
```

In [75]:
```
p_value_Upfront_charges
```

Out[75]:
```
0.0
```

In [76]:
```
p_value_property_value
```

Out[76]:
```
3.2842077511726035e-154
```

In [77]:
```
p_value_income
```

Out[77]:
```
1.3066511707601373e-96
```

In [78]:
```
p_value_LTV
```

Out[78]:
```
6.161392571253348e-158
```

In [79]:
```
# Replacing the imputed data set in the original data set.
df['rate_of_interest'] = rate_of_interest_df['rate_of_interest']
df['Upfront_charges'] = Upfront_charges_df['Upfront_charges']
df['property_value'] = property_value_df['property_value']
df['income'] = income_df['income']
df['LTV'] = LTV_df['LTV']
```

In [80]:
```
# Checking the null counts in each columns again
df.isna().sum()
```

Out[80]:
```
ID                         0
year                       0
loan_limit                 0
Gender                     0
loan_type                  0
loan_purpose               0
business_or_commercial     0
loan_amount                0
rate_of_interest           0
Upfront_charges            0
property_value             0
occupancy_type             0
income                     0
credit_type                0
Credit_Score               0
co-applicant_credit_type   0
age                        0
LTV                        0
Region                     0
Status                     0
dtype: int64
```

Observations:

- P value in all cases are very very less than 0.5, hence we can say the this imputation has altered the distribution of the original data set.
- Any way here we wont do any Machine learning model building with this data set. Our objective is to find the insights from the projects.
- If we do hypothesis testing this change will definetly have some impact on biasness but we can afford it for now.

In [ ]:

3.2. Handling Outliers

- We have seen all the numerical features have outliers in 2.3 Data exploration.
- But it is important to note that this outliers in finanace are valid.
  - **Loan Amount**: Loan amounts can vary significantly based on the type of loan (e.g., personal loan vs. mortgage). A high-value mortgage for a commercial property will naturally have a much larger loan amount compared to a personal loan.These should not be removed or imputed unless there's an error in the data.
  - **Rate_of_interest**: Interest rates can differ based on factors such as the loan type, the applicant's credit score, or economic conditions. For example, subprime borrowers might have significantly higher rates.
  - **Upfront_charges**: The upfront charges (e.g., down payments, processing fees) can vary widely based on the size of the loan, type of property, and whether the loan is commercial or personal.
  - **Property_value**: Property values can have a wide range depending on location (e.g., urban vs. rural), type of property (e.g., residential vs. commercial), and the overall real estate market.
  - **Income**: Applicant income can vary significantly, particularly if the dataset includes both individual applicants and businesses. For example, business owners or high-net-worth individuals may have extremely high incomes compared to the average borrower.
  - **LTV**: LTV may vary greatly depending on the loan type and the financial profile of the borrower, leading to valid outliers.

Observations:

```
- Hence we won't remove or impute these outliers and keep it as it is.
```

In [ ]:

3.3. Data type conversion

- We have change the data type of any of the feature which we fill is need to change.
  - We can observe in 2.3 that the **Status** columns have **int64** data type, but this is a categorical column.

In [85]:
```
# Changing the data type of status to object
df['Status'] = df['Status'].astype('object')
```

In [ ]:

3.3. Creating Derived Features

- **Purpose:** Derived features can capture additional insights and improve the quality of the analysis.
- **Examples:**
  - **Annual Income:** `income * 12` currently we have monthly income , we need annual income col also for ease.
  - **Loan-to-Value (LTV) Ratio:** `Loan Amount / Property Value` to assess risk.
  - **Loan-to-Income (LTI) Ratio:** `loan_amount / income` as an indicator of credit health. (**This can be considered as Debt-To-Income ratio also in this case**)
  - **Interest-to-Income Ratio:** `Interest Rate / income` to capture complex relationships.
  - **Property-to-Income Ratio:** `property_value / income` to capture complex relationships.

In [87]:
```
# Creating new feature annual income
df['Annual income'] = np.round(df['income']*12,2)
```

In [88]:
```
# Creating new feature Loan to Value ratio
df['Loan-to-Value'] = np.round(df['loan_amount']/df['property_value'],2)
```

In [89]:
```
# Creating new feature Loan-to-Income ratio
df['Loan-to-Income'] = np.round(df['loan_amount']/df['Annual income'],2)
```

In [90]:
```
# Creating new feature Interest-to-Income ratio
df['Interest-to-Income'] = np.round((df['rate_of_interest']*df['loan_amount'])/df['Annual income'],2)
```

In [91]:
```
# Creating new feature Property-to-Income ratio
df['Property-to-Income'] = np.round(df['property_value']/df['Annual income'],2)
```

In [92]:
```
df.head()
```

Out[92]:

	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_commercial	loan_amount	rate_of_interest	Upfront_charges
0	24890	2019	cf	Sex Not Available	type1	p1	nob/c	116500	3.99	2596.45
1	24891	2019	cf	Male	type2	p1	b/c	206500	3.99	2596.45
2	24892	2019	cf	Male	type1	p1	nob/c	406500	4.56	595.00
3	24893	2019	cf	Male	type1	p4	nob/c	456500	4.25	2596.45
4	24894	2019	cf	Joint	type1	p1	nob/c	696500	4.00	0.00

5 rows × 25 columns

In [ ]:

4. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a critical phase in data analysis that involves examining the dataset to summarize its main characteristics, often using visual methods. The goal of EDA is to understand the underlying structure of the data, detect anomalies, test assumptions, and generate insights that will inform further analysis. This section will detail various analytical approaches, including univariate, bivariate, and multivariate analyses, as well as data visualizations.

4.1. Univariate Analysis

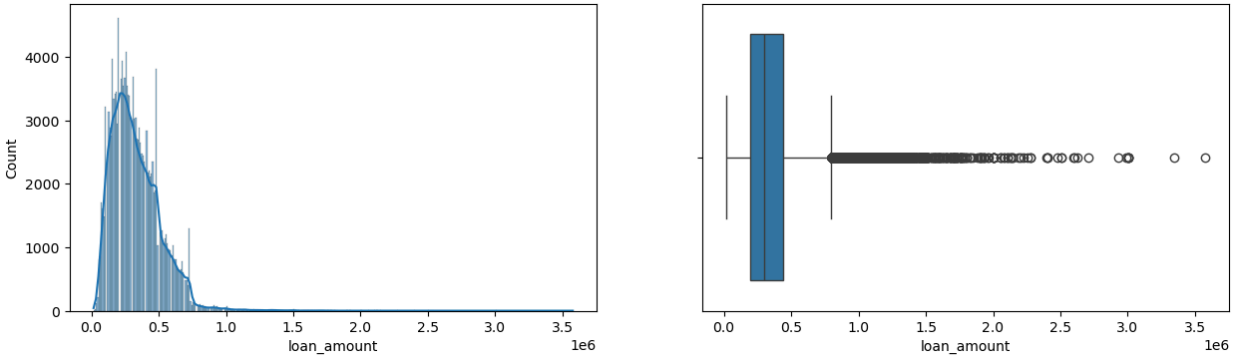
Univariate analysis focuses on examining individual variables in the dataset. It involves analyzing the distribution, central tendency, and spread of each feature. In this project, we will specifically analyze:

4.1.1. Distribution of Loan Amount, Interest Rates, and Credit Scores

- **Loan Amount**: We will plot histograms and boxplots to visualize the distribution of loan amounts. This analysis will help identify the most common loan amounts and any skewness or outliers.
- **Interest Rates**: A similar approach will be taken to analyze interest rates, including examining the average rates and their distributions.
- **Credit Scores**: We will evaluate the distribution of credit scores to understand the overall creditworthiness of applicants and identify any trenomize this content according to your specific requirements or insights from your analysis!

Loan amount

In [95]:
```
plt.figure(figsize=[15,4])
plt.subplot(1,2,1)
sns.histplot(data=df, x='loan_amount', kde=True)

plt.subplot(1,2,2)
sns.boxplot(data=df, x='loan_amount')
plt.show()
```



In [96]:
```
df['loan_amount'].describe()
```

Out[96]:
```
count    148670.00
mean     331117.74
std      183909.31
min       16500.00
25%      196500.00
50%      296500.00
75%      436500.00
max     3576500.00
Name: loan_amount, dtype: float64
```

In [97]:
```
# mode of the Loan amount
df['loan_amount'].mode()
```

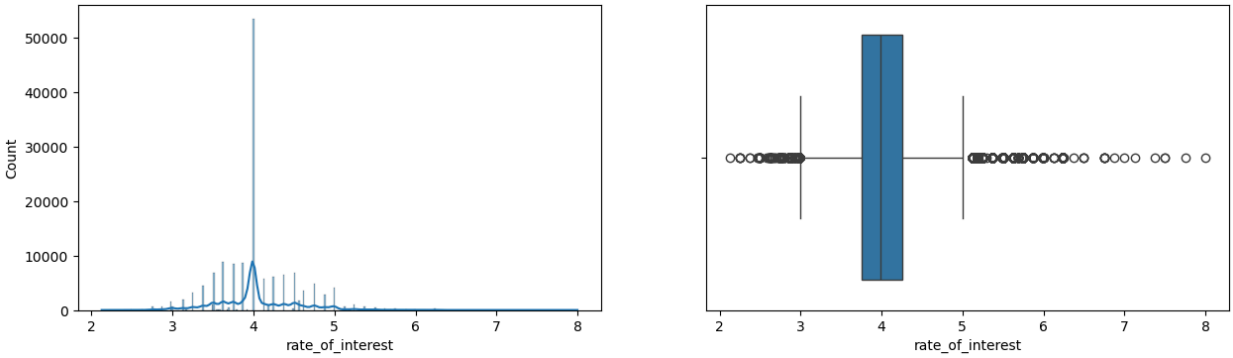
Out[97]:
```
0    206500
Name: loan_amount, dtype: int64
```

In [98]:
```
#upper wisker value and lower wisker value equals to min value.
upper_wisker = 436500 + (1.5 * (436500 - 196500))
upper_wisker
```

Out[98]:
```
796500.0
```

Interest Rate

In [100…
```
plt.figure(figsize=[15,4])
plt.subplot(1,2,1)
sns.histplot(data=df, x='rate_of_interest', kde=True)

plt.subplot(1,2,2)
sns.boxplot(data=df, x='rate_of_interest')
plt.show()
```



In [101…
```
df['rate_of_interest'].describe()
```

Out[101…
```
count   148670.00
mean         4.03
std          0.49
min          2.12
25%          3.75
50%          3.99
75%          4.25
max          8.00
Name: rate_of_interest, dtype: float64
```

In [102…
```
# mode of rate_of_interest
df['rate_of_interest'].mode()
```

Out[102…
```
0   3.99
Name: rate_of_interest, dtype: float64
```

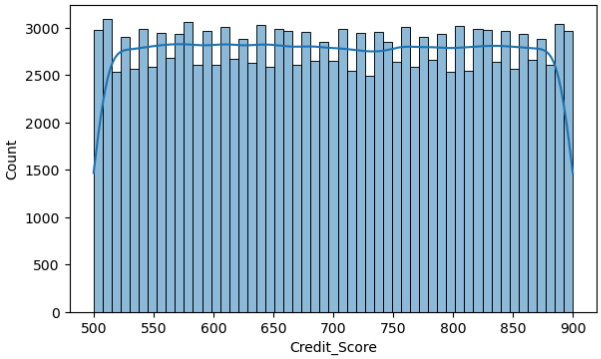
In [103…
```
#upper wisker value and lower wisker value equals to min value.
upper_wisker = 4.25 + (1.5 * (4.25 - 3.75))
lower_wisker = 3.75 - (1.5 * (4.25 - 3.75))
```

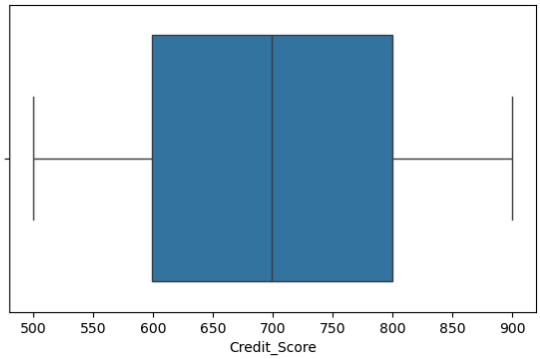
In [104…
```
print(f'Upper wisker and Lower wisker are: {upper_wisker} and {lower_wisker}')
```

```
Upper wisker and Lower wisker are: 5.0 and 3.0
```

Credit Scores

In [106…
```
plt.figure(figsize=[15,4])
plt.subplot(1,2,1)
sns.histplot(data=df, x='Credit_Score', kde=True)

plt.subplot(1,2,2)
sns.boxplot(data=df, x='Credit_Score')
plt.show()
```





In [107…
```
df['Credit_Score'].describe()
```

Out[107…
```
count   148670.00
mean       699.79
std        115.88
min        500.00
25%        599.00
50%        699.00
75%        800.00
max        900.00
Name: Credit_Score, dtype: float64
```

In [108…
```
# mode of rate_of_interest
df['Credit_Score'].mode()
```

Out[108…
```
0    763
Name: Credit_Score, dtype: int64
```

In [109…
```
#upper wisker value and lower wisker value equals to min value.
upper_wisker = 800 + (1.5 * (800 - 599))
lower_wisker = 599 - (1.5 * (800 - 599))
```

In [110…
```
print(f'Upper wisker and Lower wisker are: {upper_wisker} and {lower_wisker}')
```

```
Upper wisker and Lower wisker are: 1101.5 and 297.5
```

4.1.2. Analysis of Loan Purpose, Loan Type, and Occupancy Type etc.

- **Loan Purpose**: Bar charts will illustrate the frequency of different loan purposes (e.g., home purchase, education, business) to assess trends in borrowing behavior.
- **Loan Type**: We will analyze different loan types and their distributions, noting any preferences or patterns among borrowers.
- **Occupancy Type**: This analysis will identify trends in occupancy (e.g., primary residence, rental) and how they relate to loan defaults.

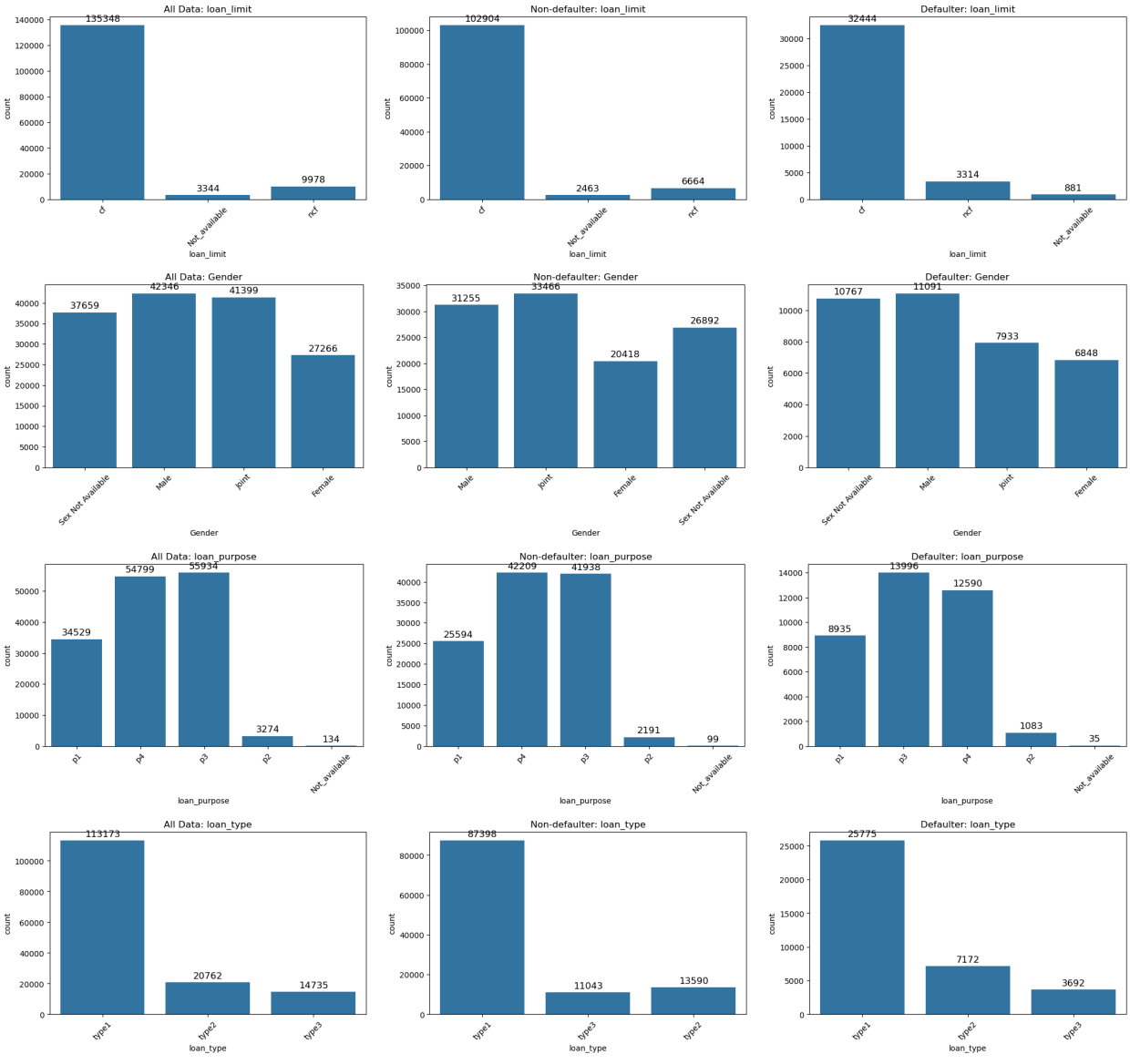
In [112…
```
# updating the non_defualt df and default df as we updated the df later.
non_defaulter_df = df[df['Status']==0]
defaulter_df = df[df['Status']==1]
```

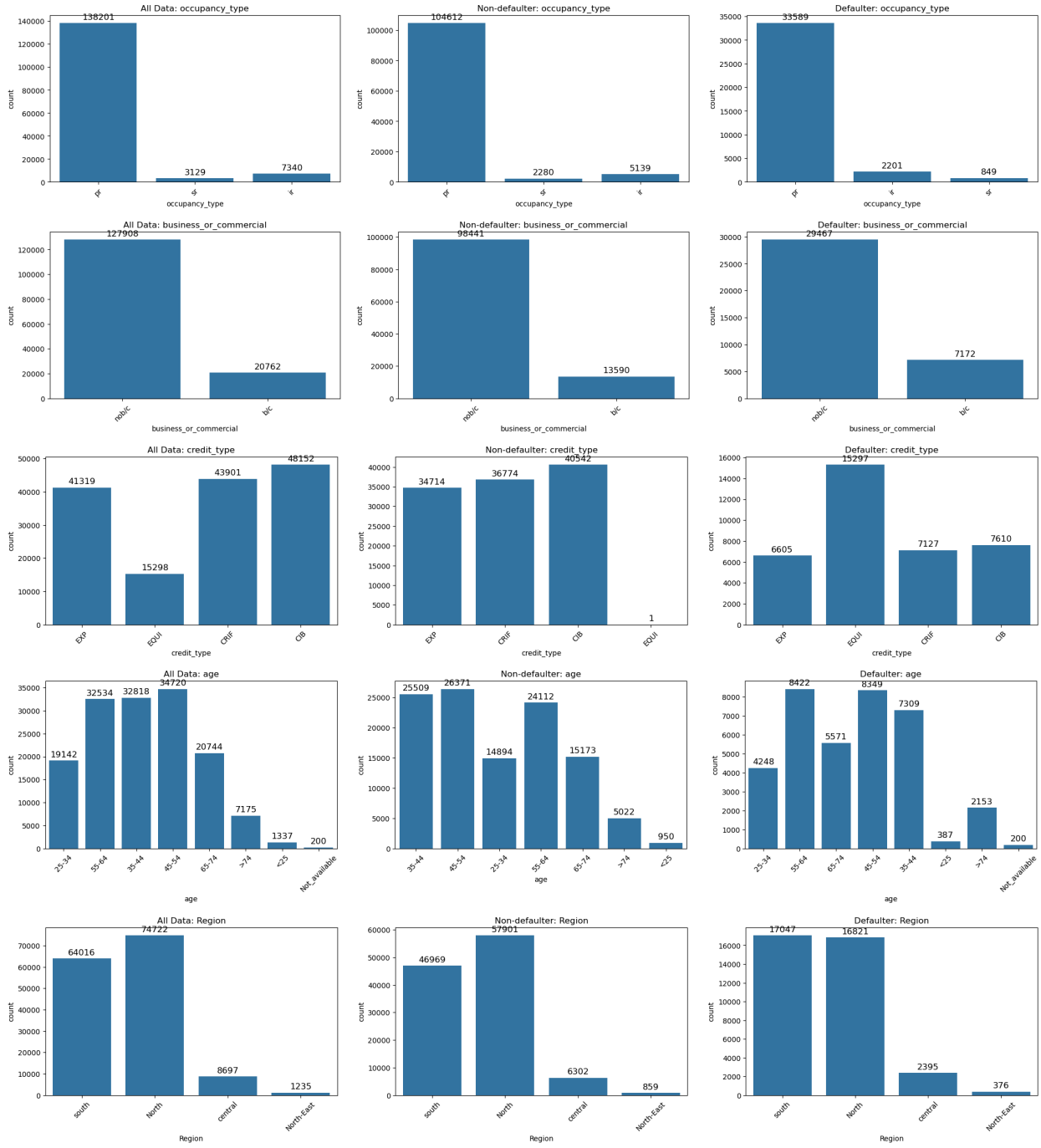
In [113…
```
# Extracting only categorical data set from defaulter and non defaulter.
cat_df = df.select_dtypes(exclude=['object'])
cat_non_defaulter_df= non_defaulter_df.select_dtypes(include=['object'])
cat_defaulter_df= defaulter_df.select_dtypes(include=['object'])
```

In [114…
```
cols_412 = ['loan_limit','Gender','loan_purpose','loan_type','occupancy_type','business_or_commercial','credit_type','age'
```

In [115…
```
# Countplot for 'loan_purpose','loan_type','occupancy_type'
j=1
for i in cols_412:
    plt.figure(figsize=[25,45])
    # Create subplots for the same column from three different features side by side
    for y in range(j, j + 3):
        plt.subplot(9, 3, y)

        # Plot for the entire data
        if y == j:
            # Create the countplot
            ax = sns.countplot(data=df, x=i)
            plt.xticks(rotation=45)
            plt.title(f"All Data: {i}")

            # Annotate the count on each bar
            for p in ax.patches:
                ax.annotate(f'{int(p.get_height())}',
                            (p.get_x() + p.get_width() / 2., p.get_height()),
                            ha = 'center', va = 'baseline',
                            fontsize=12, color='black', xytext=(0, 5),
                            textcoords='offset points')
```

```
        # Plot for non-defaulters
        elif y == j + 1:
            # Create the countplot
            ax = sns.countplot(data=cat_non_defaulter_df, x=i)
            plt.xticks(rotation=45)
            plt.title(f"Non-defaulter: {i}")

            # Annotate the count on each bar
            for p in ax.patches:
                ax.annotate(f'{int(p.get_height())}',
                            (p.get_x() + p.get_width() / 2., p.get_height()),
                            ha = 'center', va = 'baseline',
                            fontsize=12, color='black', xytext=(0, 5),
                            textcoords='offset points')

        # Plot for defaulters
        elif y == j + 2:
            # Create the countplot
            ax = sns.countplot(data=cat_defaulter_df, x=i)
            plt.xticks(rotation=45)
            plt.title(f"Defaulter: {i}")

            # Annotate the count on each bar
            for p in ax.patches:
                ax.annotate(f'{int(p.get_height())}',
                            (p.get_x() + p.get_width() / 2., p.get_height()),
                            ha = 'center', va = 'baseline',
                            fontsize=12, color='black', xytext=(0, 5),
                            textcoords='offset points')

    # Move j forward by 3 to handle the next set of subplots
    j += 3

    # Adjust layout and display the subplots for the current column
    plt.show()
```





In [116…

```
# Extract object type (categorical variables) from the DataFrame
categorical_columns = df.select_dtypes(include=['object']).columns

# Define the number of tables per row
tables_per_row = 3

# Loop through the categorical columns and show value counts in percentages
for i, col in enumerate(categorical_columns):
    # Get the value counts as percentages for the categorical column and round to 2 decimal places
    value_counts = df[col].value_counts(normalize=True).mul(100).round(2).reset_index()
    value_counts.columns = [col, 'Percentage (%)']

    # Print the table for each column
    print(f"Table: {col} Distribution\n")
    print(value_counts.to_string(index=False))
    print("\n" + "-" * 50 + "\n")

    # Print a blank line after every set of 3 tables for spacing
    if (i + 1) % tables_per_row == 0:
        print("\n")
```

```
Table: loan_limit Distribution

   loan_limit  Percentage (%)
           cf           91.04
          ncf            6.71
Not_available            2.25

--------------------------------------------------

Table: Gender Distribution

           Gender  Percentage (%)
             Male           28.48
            Joint           27.85
Sex Not Available           25.33
           Female           18.34

--------------------------------------------------

Table: loan_type Distribution

loan_type  Percentage (%)
    type1           76.12
    type2           13.97
    type3            9.91

--------------------------------------------------



Table: loan_purpose Distribution

 loan_purpose  Percentage (%)
           p3           37.62
           p4           36.86
           p1           23.23
           p2            2.20
Not_available            0.09

--------------------------------------------------

Table: business_or_commercial Distribution

business_or_commercial  Percentage (%)
                 nob/c           86.03
                   b/c           13.97

--------------------------------------------------

Table: occupancy_type Distribution

occupancy_type  Percentage (%)
            pr           92.96
            ir            4.94
            sr            2.10

--------------------------------------------------



Table: credit_type Distribution

credit_type  Percentage (%)
        CIB           32.39
       CRIF           29.53
        EXP           27.79
       EQUI           10.29

--------------------------------------------------

Table: co-applicant_credit_type Distribution

co-applicant_credit_type  Percentage (%)
                     CIB           50.04
                     EXP           49.96

--------------------------------------------------

Table: age Distribution

         age  Percentage (%)
       45-54           23.35
       35-44           22.07
       55-64           21.88
       65-74           13.95
       25-34           12.88
```

```
         >74            4.83
         <25            0.90
Not_available           0.13

--------------------------------------------------



Table: Region Distribution

    Region  Percentage (%)
     North           50.26
     south           43.06
   central            5.85
North-East            0.83

--------------------------------------------------

Table: Status Distribution

 Status  Percentage (%)
      0           75.36
      1           24.64

--------------------------------------------------
```

Observations:

Loan Amount:

- Without the outliers, the loan amount generally varies between 165,000 and 796,500. However, the majority of the loan amounts are between 196,500 and 436,500.
- Central Tendencies of the loan amount:
  - **Mean**: 331,117
  - **Median**: 296,500
  - **Mode**: 206,500
- The data is right-skewed.
- Outliers are valid in the loan amount feature, as there will be customers who take loans of larger amounts.

Interest Rate:

- Without the outliers, the interest rate generally varies between 3 and 5. However, the majority of interest rates fall between 3.75 and 4.25.
- Central Tendency of the interest rate:
  - **Mean**: 4
  - **Median**: 3.99
  - **Mode**: 3.99
- Outliers are valid as customers with different credit scores may have different interest rates.
  - For customers with the best credit scores, the interest rate may be very low, while customers with high-risk (low credit scores) may have higher interest rates.
- The high number of entries at 4 in the histogram is due to the imputation performed earlier.

Credit Score:

- The maximum and minimum credit scores are 900 and 500, respectively, which are also the upper and lower whiskers of the boxplot.
- The distribution of credit scores is almost uniform between 500 and 900.
- Credit scores generally vary between 500 and 900, with the majority of credit scores between 600 and 800.
- Central Tendencies of credit scores:
  - **Mean**: 700
  - **Median**: 700
  - **Mode**: 763

Categorical Features:

- **Loan Type Proportion**:
  - Type1: 76.12%
  - Type2: 13.97%
  - Type3: 9.91%
  - Loan type distribution is similar between defaulters and non-defaulters.
- **Loan Purpose Proportion**:

- P3: 37.66%
- P4: 36.89%
- P1: 23.25%
- P2: 2.2%
- Among non-defaulters, P4 is slightly more prevalent than P3, while among defaulters, P3 is more prevalent than P4.

- **Occupancy Type Proportion**:
  - PR: 92.96%
  - IR: 4.92%
  - SR: 2.1%
  - The occupancy type distribution is the same between defaulters and non-defaulters.
- **Gender Proportion**:
  - Male: 28.48%
  - Female: 18.34%
  - Joint: 27.87%
  - Unknown: 25.33%
  - Among non-defaulters, joint applicants are the most common, followed by male, unknown, and female applicants. Among defaulters, males are the most common, followed by unknown, joint, and female applicants.
- **Loan Limit Proportion**:
  - CF: 93.13%
  - NCF: 6.87%
  - The loan limit distribution is the same between defaulters and non-defaulters.
- **Business or Commercial Proportion**:
  - Non-Business/Commercial (No B/C): 86.03%
  - Business/Commercial (B/C): 13.97%
  - The business or commercial loan distribution is the same between defaulters and non-defaulters.
- **Credit Type Proportion**:
  - CIB: 32.39%
  - CRIF: 29.53%
  - EXP: 27.79%
  - EQUI: 10.29%
  - Among non-defaulters, CIB is the most common, followed by CRIF, EXP, and EQUI. Among defaulters, EQUI is the most common, followed by CIB, CRIF, and EXP.
- **Age Distribution**:
  - <25: 0.9%
  - 25-34: 12.89%
  - 35-44: 22.10%
  - 45-54: 23.39%
  - 55-64: 21.91%
  - 65-74: 13.97%
  - >74: 4.83%
  - Among non-defaulters, the age group with the highest number of individuals is 45-54, followed by 35-44, 55-64, 65-74, 25-34, >74, and <25. Among defaulters, the age group with the highest number is 55-64, followed by 45-54, 35-44, 65-74, 25-34, >74, <25, and unknown.
- **Region Distribution**:
  - The regional distribution is the same between defaulters and non-defaulters.

4.1.3. Distribution and Impact of Newly Derived Features

This sub-point will include a focus on the newly derived features (Loan-to-Value Ratio, Loan-to-Income Ratio, Interest-to-Income Ratio, Property-to-Income Ratio) and their distributions. Histograms and boxplots will be utilized to visualize their distributions, and summary statistics will be calculated to understand their central tendencies.

In [119…
```
#updating numeric df as we have updated the main df with new features
numeric_df = df.select_dtypes(exclude=['object'])
numeric_non_defaulter_df= non_defaulter_df.select_dtypes(exclude=['object'])
numeric_defaulter_df= defaulter_df.select_dtypes(exclude=['object'])
```

In [120…
```
cols_413 = ['Loan-to-Value', 'Loan-to-Income', 'Interest-to-Income', 'Property-to-Income']
```

In [121…
```
# statstics info of all Data
numeric_df[cols_413].describe()
```

Out[121…

	Loan-to-Value	Loan-to-Income	Interest-to-Income	Property-to-Income
count	148670.00	148670.00	148670.00	148670.00
mean	0.73	4.75	19.04	7.24
std	0.41	7.70	30.93	21.78
min	0.01	0.05	0.20	0.06
25%	0.60	3.12	12.40	4.30
50%	0.75	4.38	17.53	6.05
75%	0.87	5.88	23.65	8.48
max	78.31	2106.25	8403.94	5844.44

In [122…

```
# statistic info of non defaulter data
numeric_non_defaulter_df[cols_413].describe()
```

Out[122…

	Loan-to-Value	Loan-to-Income	Interest-to-Income	Property-to-Income
count	112031.00	112031.00	112031.00	112031.00
mean	0.72	4.54	18.25	6.84
std	0.42	2.46	10.58	4.74
min	0.02	0.05	0.23	0.07
25%	0.60	3.10	12.29	4.32
50%	0.75	4.33	17.35	6.05
75%	0.85	5.76	23.25	8.29
max	78.31	337.85	1562.54	568.06

In [123…

```
# Statistcs info of defaulter data
numeric_defaulter_df[cols_413].describe()
```

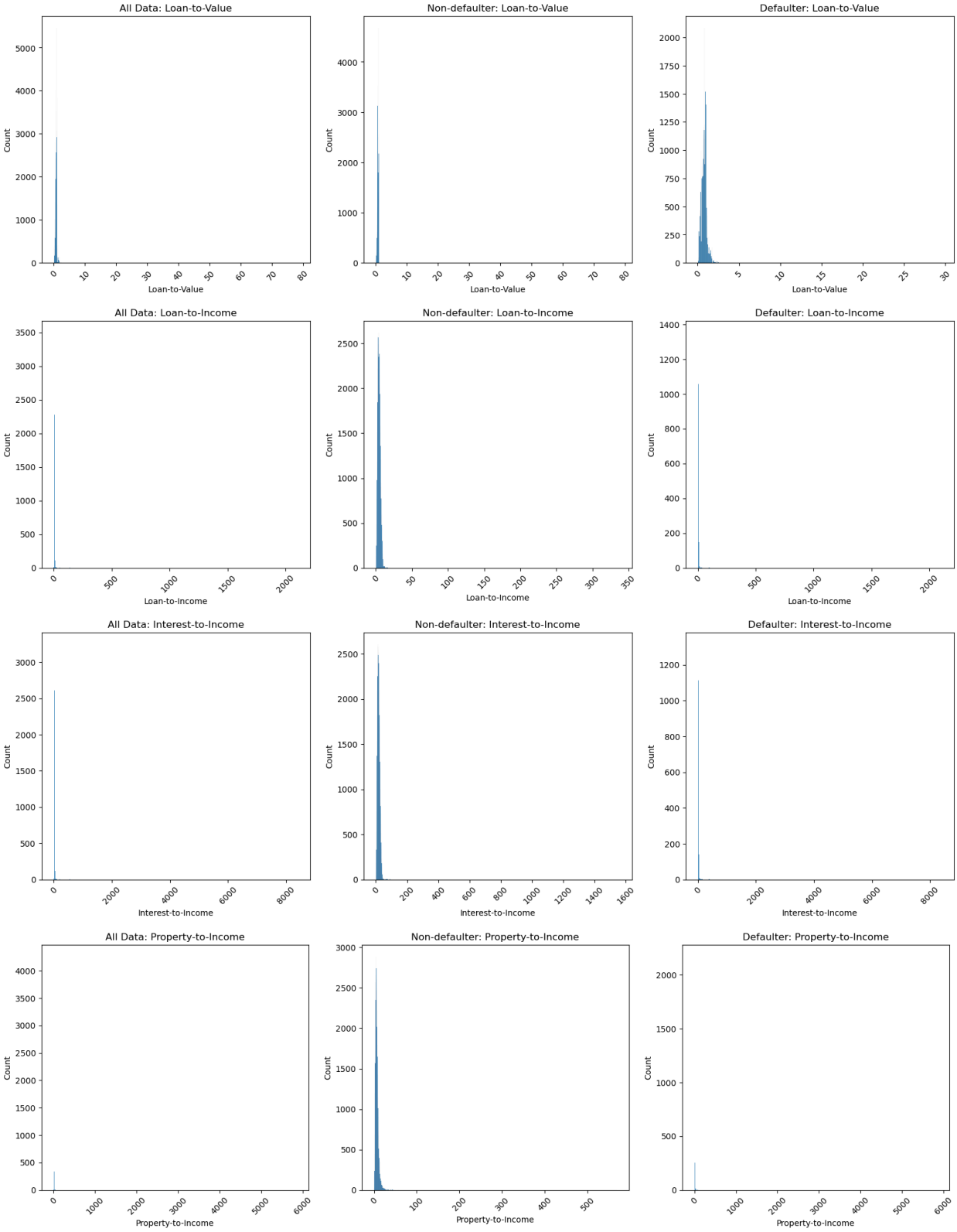
Out[123…

	Loan-to-Value	Loan-to-Income	Interest-to-Income	Property-to-Income
count	36639.00	36639.00	36639.00	36639.00
mean	0.77	5.37	21.45	8.47
std	0.38	14.89	59.42	43.06
min	0.01	0.05	0.20	0.06
25%	0.57	3.18	12.71	4.18
50%	0.77	4.54	18.13	6.05
75%	0.93	6.26	25.00	9.22
max	29.56	2106.25	8403.94	5844.44

In [124…

```
# Histogram plot for numeric columns
j=1
for i in cols_413:
    plt.figure(figsize=[20,25])
    # Create subplots for the same column from three different datasets side by side
    for y in range(j, j + 3):
        plt.subplot(4, 3, y)

        # Plot for the entire data
        if y == j:
            sns.histplot(data=numeric_df, x=i)
            plt.xticks(rotation=45)
            plt.title(f"All Data: {i}")

        # Plot for non-defaulters
        elif y == j + 1:
            sns.histplot(data=numeric_non_defaulter_df, x=i)
            plt.xticks(rotation=45)
            plt.title(f"Non-defaulter: {i}")

        # Plot for defaulters
        elif y == j + 2:
            sns.histplot(data=numeric_defaulter_df, x=i)
            plt.xticks(rotation=45)
            plt.title(f"Defaulter: {i}")

    # Move j forward by 3 to handle the next set of subplots
    j += 3
```

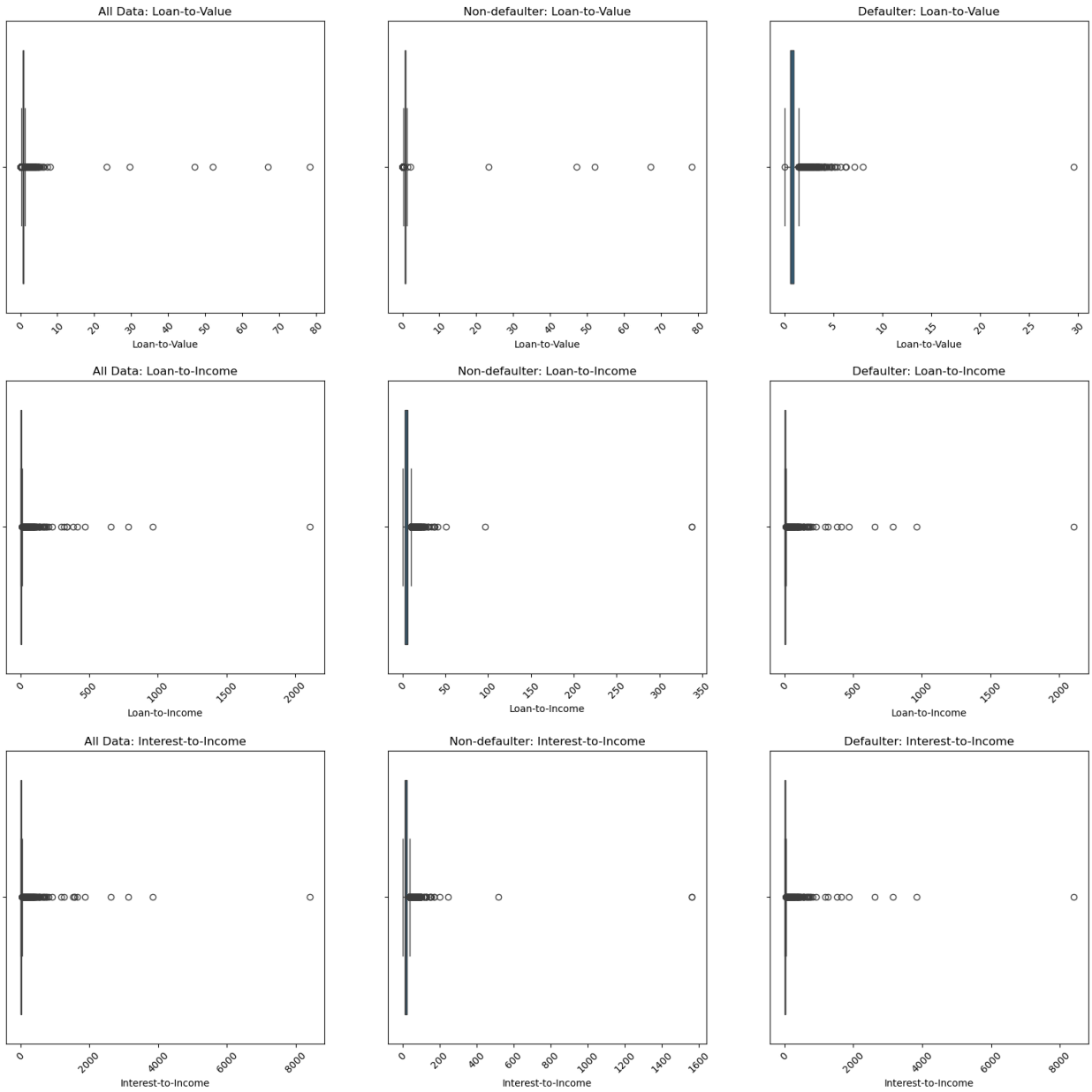
```
    # Adjust layout and display the subplots for the current column
    plt.show()
```

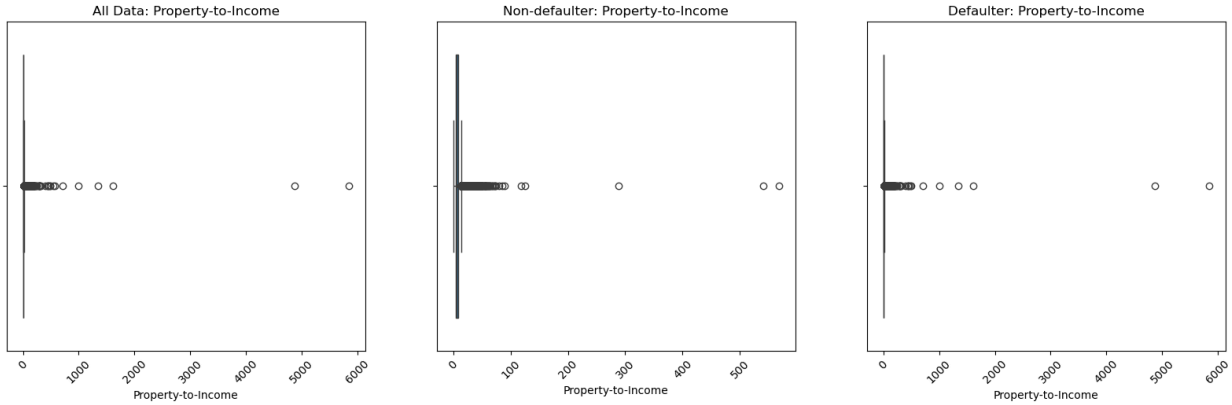


In [125…

```
# Histogram plot for numeric columns
j=1
for i in cols_413:
    plt.figure(figsize=[20,25])
    # Create subplots for the same column from three different datasets side by side
    for y in range(j, j + 3):
        plt.subplot(4, 3, y)

        # Plot for the entire data
        if y == j:
            sns.boxplot(data=numeric_df, x=i)
            plt.xticks(rotation=45)
            plt.title(f"All Data: {i}")
```

```
        # Plot for non-defaulters
        elif y == j + 1:
            sns.boxplot(data=numeric_non_defaulter_df, x=i)
            plt.xticks(rotation=45)
            plt.title(f"Non-defaulter: {i}")

        # Plot for defaulters
        elif y == j + 2:
            sns.boxplot(data=numeric_defaulter_df, x=i)
            plt.xticks(rotation=45)
            plt.title(f"Defaulter: {i}")

    # Move j forward by 3 to handle the next set of subplots
    j += 3

    # Adjust layout and display the subplots for the current column
    plt.show()
```





In [126…
```
# how much % of data is below 0.8 LTV
percentage_ltv = round(len(df[df['Loan-to-Value'] < 0.8])/len(df['Loan-to-Value'])*100,1)
percentage_ltv
```

Out[126… 60.4

In [127…
```
# how much % of data is below 5 LTV
percentage_lti = round(len(df[df['Loan-to-Income'] < 5])/len(df['Loan-to-Income'])*100,1)
percentage_lti
```

Out[127… 61.3

In [128…
```
# how much % of data is below 0.36 ITI
percentage_iti = round(len(df[df['Interest-to-Income'] < 0.36])/len(df['Interest-to-Income'])*100,3)
percentage_iti
```

Out[128… 0.002

In [129…
```
# how much % of data is below 5 PTI
percentage_pti = round(len(df[df['Property-to-Income'] < 5])/len(df['Property-to-Income'])*100,1)
percentage_pti
```

Out[129… 34.8

Observations

Loan to Value (LTV)

- **Central Tendencies of Loan to Value for Overall Data**:
  - Mean: 0.73
  - Median: 0.75
- **Min Loan to Value**: 0.01
- **Max Loan to Value**: 78.31
- **Central Tendencies of Loan to Value for Non-Defaulters**:
  - Mean: 0.72
  - Median: 0.75
- **Min Loan to Value for Non-Defaulters**: 0.02
- **Max Loan to Value for Non-Defaulters**: 78.31
- **Central Tendencies of Loan to Value for Defaulters**:
  - Mean: 0.77
  - Median: 0.77
- **Min Loan to Value for Defaulters**: 0.01
- **Max Loan to Value for Defaulters**: 29.56
- Loan to Value distribution is **right skewed**.
- All have **valid outliers**.
- **60.4%** of the data has a Loan to Value below **0.8**.

Loan to Income (LTI)

- **Central Tendencies of Loan to Income for Overall Data**:
  - Mean: 4.75
  - Median: 4.38
- **Min Loan to Income**: 0.05

- **Max Loan to Income**: 2106.25
- **Central Tendencies of Loan to Income for Non-Defaulters**:
  - Mean: 4.54
  - Median: 4.33
- **Min Loan to Income for Non-Defaulters**: 0.05
- **Max Loan to Income for Non-Defaulters**: 337.85
- **Central Tendencies of Loan to Income for Defaulters**:
  - Mean: 4.54
  - Median: 4.54
- **Min Loan to Income for Defaulters**: 0.05
- **Max Loan to Income for Defaulters**: 2106.25
- Loan to Income distribution is **right skewed**.
- All have **valid outliers**.
- **61.3%** of the data has a Loan to Income below **5**.

Interest to Income (ITI)

- **Central Tendencies of Interest to Income for Overall Data**:
  - Mean: 19.04
  - Median: 17.53
- **Min Interest to Income**: 0.2
- **Max Interest to Income**: 8403.94
- **Central Tendencies of Interest to Income for Non-Defaulters**:
  - Mean: 18.25
  - Median: 17.35
- **Min Interest to Income for Non-Defaulters**: 0.23
- **Max Interest to Income for Non-Defaulters**: 1562.54
- **Central Tendencies of Interest to Income for Defaulters**:
  - Mean: 18.13
  - Median: 18.13
- **Min Interest to Income for Defaulters**: 0.2
- **Max Interest to Income for Defaulters**: 8403.94
- Interest to Income distribution is **right skewed**.
- All have **valid outliers**.
- Only **0.002%** of the data has an Interest to Income below **0.36**.

Property to Income (PTI)

- **Central Tendencies of Property to Income for Overall Data**:
  - Mean: 7.4
  - Median: 6.05
- **Min Property to Income**: 0.06
- **Max Property to Income**: 5844.44
- **Central Tendencies of Property to Income for Non-Defaulters**:
  - Mean: 6.84
  - Median: 6.05
- **Min Property to Income for Non-Defaulters**: 0.07
- **Max Property to Income for Non-Defaulters**: 568.06
- **Central Tendencies of Property to Income for Defaulters**:
  - Mean: 8.47
  - Median: 6.05
- **Min Property to Income for Defaulters**: 0.06
- **Max Property to Income for Defaulters**: 5844.44
- Property to Income distribution is **right skewed**.
- All have **valid outliers**.
- **34.8%** of the data has a Property to Income ratio below **5**.

In [ ]:

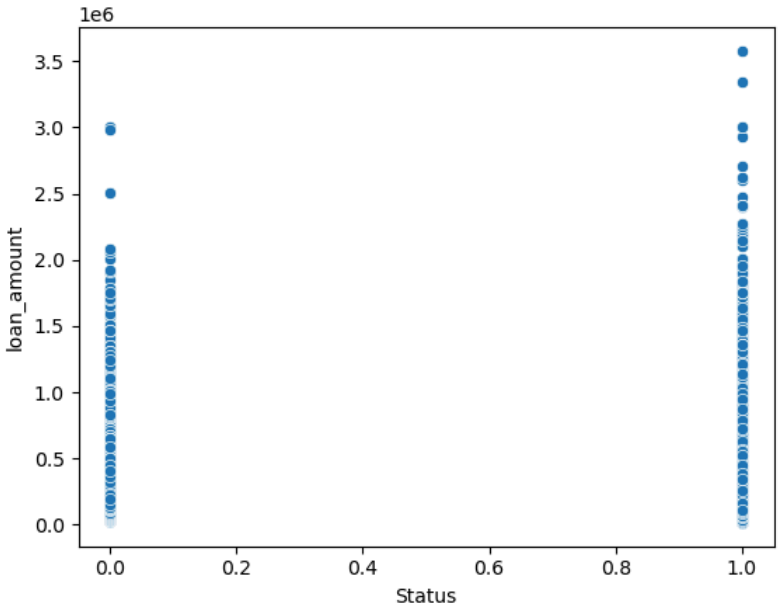
4.2. Bivariate Analysis

Bivariate analysis examines the relationship between two variables. This will involve investigating how various features interact with each other, particularly focusing on the impact of independent variables on the target variable (loan default status).

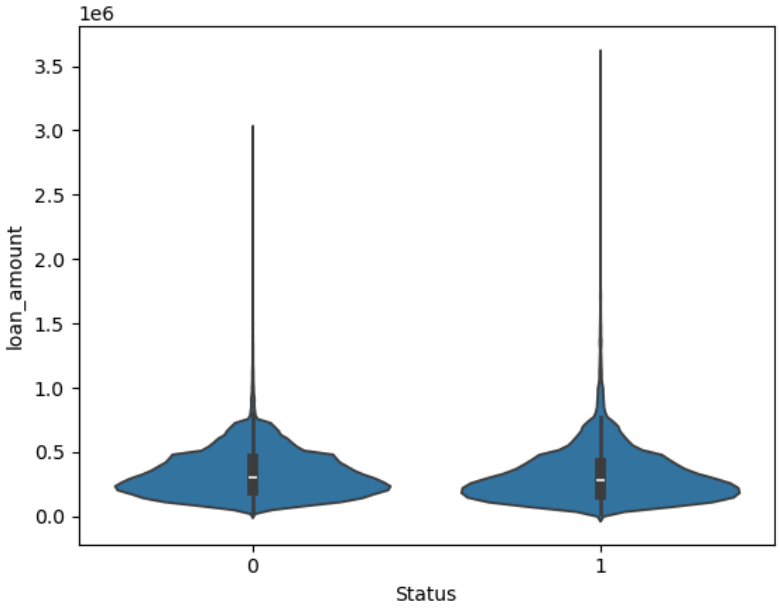
4.2.1. Relationship between Loan Amount and Default

We will analyze whether higher loan amounts are associated with higher default rates by creating scatter plots and calculating correlation coefficients. This will help assess the risk associated with larger loans.

In [132…
```
# Scatter plot between loan status and loan amount
sns.scatterplot(data=df, x='Status', y='loan_amount')
plt.show()
```



In [133…
```
# voilin plot with loan amount and status
sns.violinplot(x='Status', y='loan_amount', data=df)
plt.show()
```

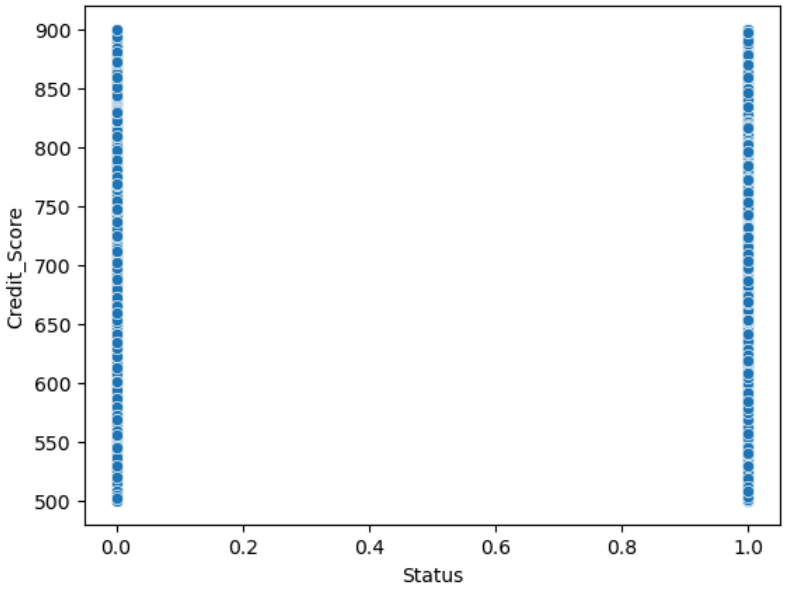


In [ ]:

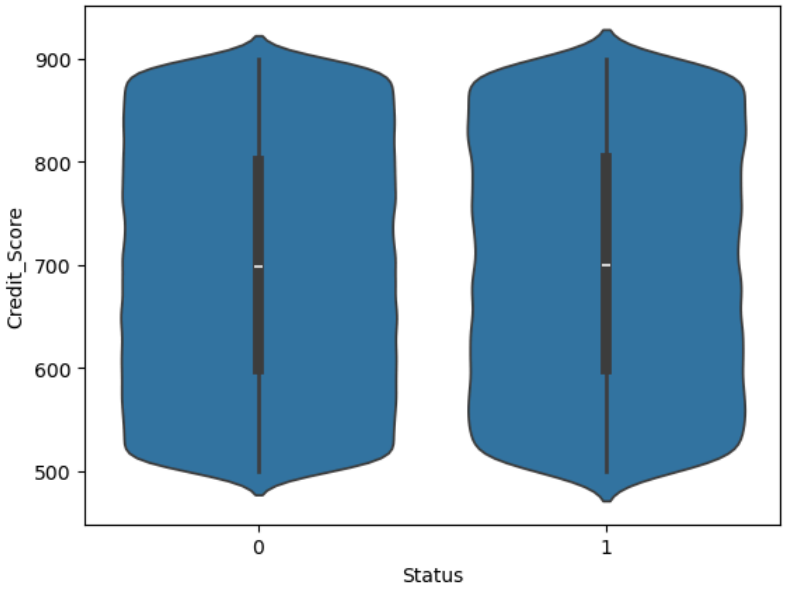
4.2.2. Impact of Credit Score on Default Probability

Using boxplots and logistic regression, we will investigate how different credit score ranges correlate with the likelihood of defaulting on a loan.

In [135…
```
# Scatter plot between loan status and loan amount
sns.scatterplot(data=df, x='Status', y='Credit_Score')
plt.show()
```



In [136…
```
# voilin plot with loan amount and status
sns.violinplot(x='Status', y='Credit_Score', data=df)
plt.show()
```

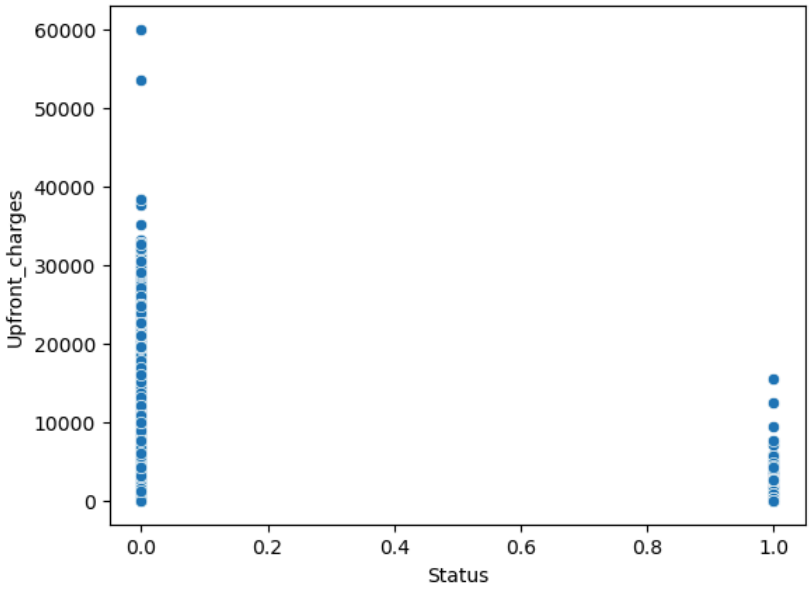


In [ ]:
```
```

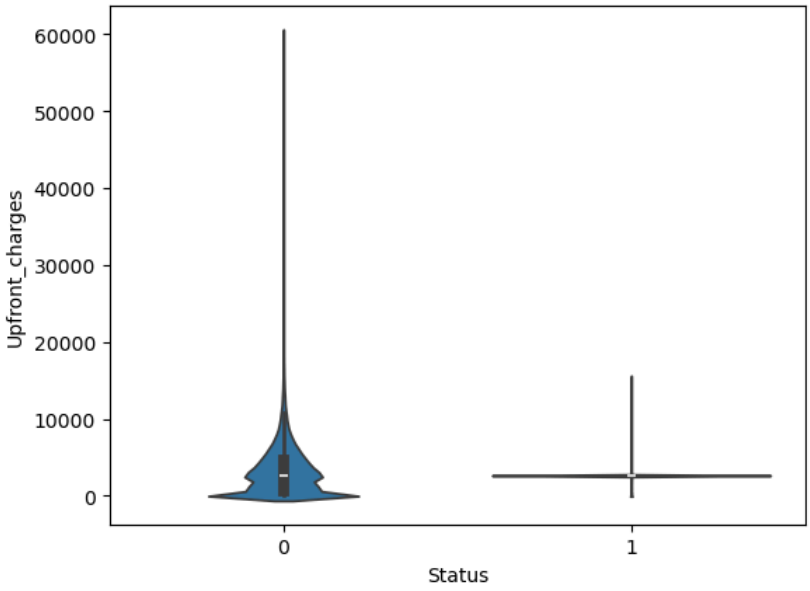
4.2.3. Correlation between Upfront Charges and Default

This analysis will evaluate the relationship between upfront charges paid by borrowers and their default rates, helping to identify any financial stress indicators.

In [138…
```
# Scatter plot between loan status and loan amount
sns.scatterplot(data=df, x='Status', y='Upfront_charges')
plt.show()
```



In [139…
```
# voilin plot with loan amount and status
sns.violinplot(x='Status', y='Upfront_charges', data=df)
plt.show()
```



In [ ]:

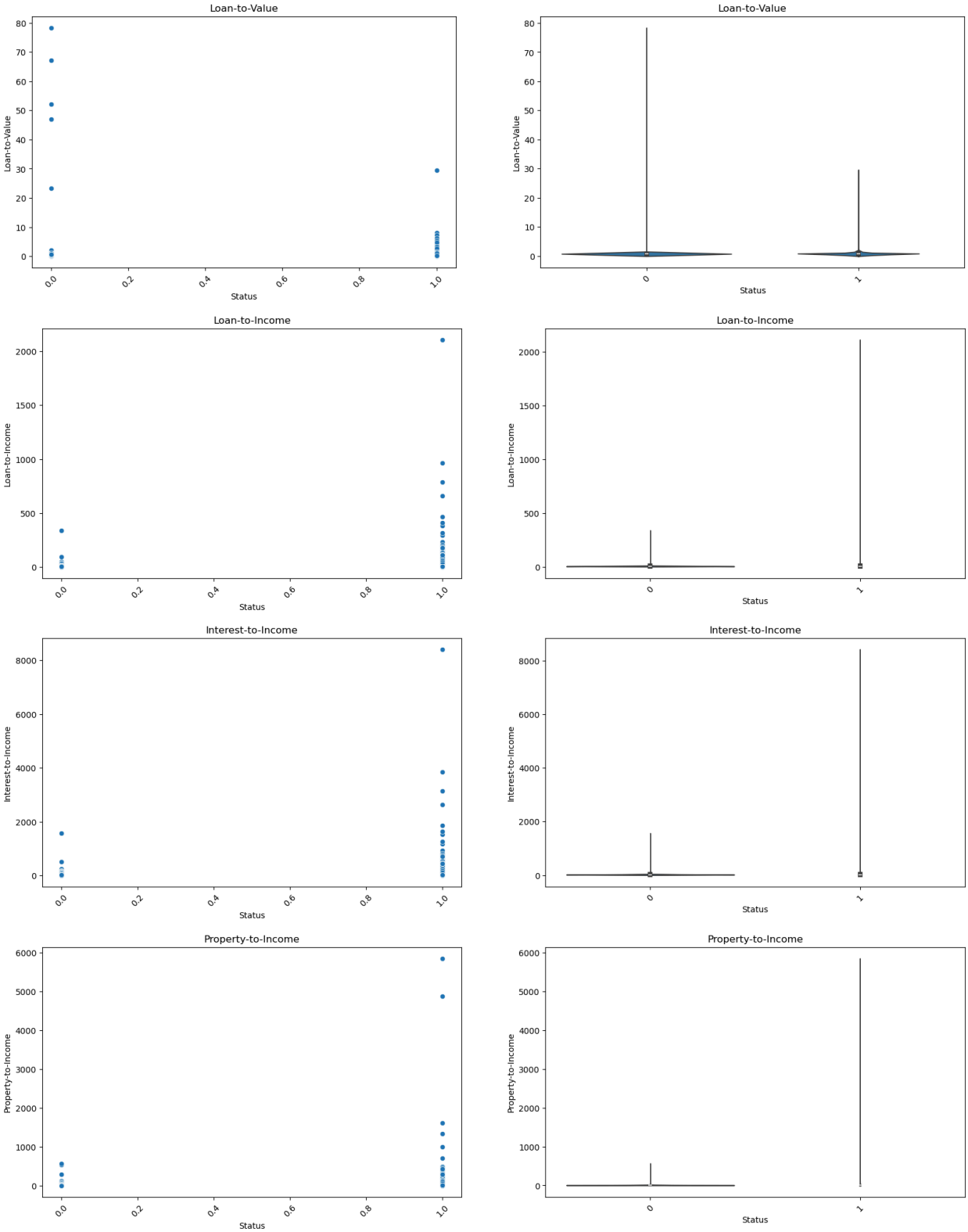
4.2.4. Analysis of Derived Features with Default Rates

We will explore how the newly derived features correlate with the likelihood of default. For example, we will examine the relationship between the Loan-to-Income Ratio and default probability.

In [141…
```
cols_424 = ['Loan-to-Value', 'Loan-to-Income', 'Interest-to-Income', 'Property-to-Income']
```

In [142…
```
# scatter and violin plot for derived features
j=1
for i in cols_413:
    plt.figure(figsize=[20,25])
    # Create subplots for the same column from three different datasets side by side
    for y in range(j, j + 2):
        plt.subplot(4, 2, y)

        # scatte Plot
        if y == j:
            sns.scatterplot(data=df, x='Status', y=i)
            plt.xticks(rotation=45)
            plt.title(f"{i}")

        # violin Plot
        elif y == j + 1:
            sns.violinplot(data=df, x='Status', y=i)
            plt.xticks(rotation=45)
```

```
            plt.title(f"{i}")

    # Move j forward by 3 to handle the next set of subplots
    j += 2

    # Adjust layout and display the subplots for the current column
    plt.show()
```



Observations

- **Loan Amount vs Default Status**:
  - The scatter plot shows slightly more points in the defaulter group toward higher loan amounts.
  - The violin plot has almost the same distribution pattern for both defaulters and non-defaulters.
- **Credit Score vs Default Status**:

  - The scatter plot and violin plot show the same distribution and pattern for both defaulters and non-defaulters.
- **Upfront Charges vs Default Status**:
  - The scatter plot has more points in the non-defaulter group across all upfront charge amounts.
  - The violin plot shows a larger area under the curve in the non-defaulter group, with almost no area in the defaulter group.
- **Loan to Value vs Default Status**:
  - The non-defaulter group has fewer dense points and shows points towards higher loan-to-value ratios.
  - In contrast, the defaulter group has a higher density of points but only in a smaller region with lower loan-to-value ratios.
- **Interest to Income vs Default Status**:
  - The scatter plot is the opposite of the Loan to Value vs Default Status plot, with a different distribution.
- **Property to Income vs Default Status**:
  - The scatter plot follows a similar pattern to the Interest to Income vs Default Status plot.

In [ ]:

4.3. Multivariate Analysis

Multivariate analysis involves examining relationships between three or more variables simultaneously. This step is crucial in understanding complex interdependencies and interactions in your dataset, especially when analyzing the factors contributing to loan defaults. The goal is to uncover how multiple features collectively influence the target variable (in this case, the likelihood of default) and provide deeper insights beyond what bivariate analysis can offer.

Purpose of Multivariate Analysis:

- To discover patterns that cannot be seen by looking at pairs of variables.
- To understand the combined effect of multiple factors on the target variable (default status).
- To identify interactions between variables that may affect the outcome.
- In the context of loan default analysis, you'll be focusing on combinations of variables like loan type, interest rates, property value, and other * significant financial ratios to determine their combined influence on default risk.

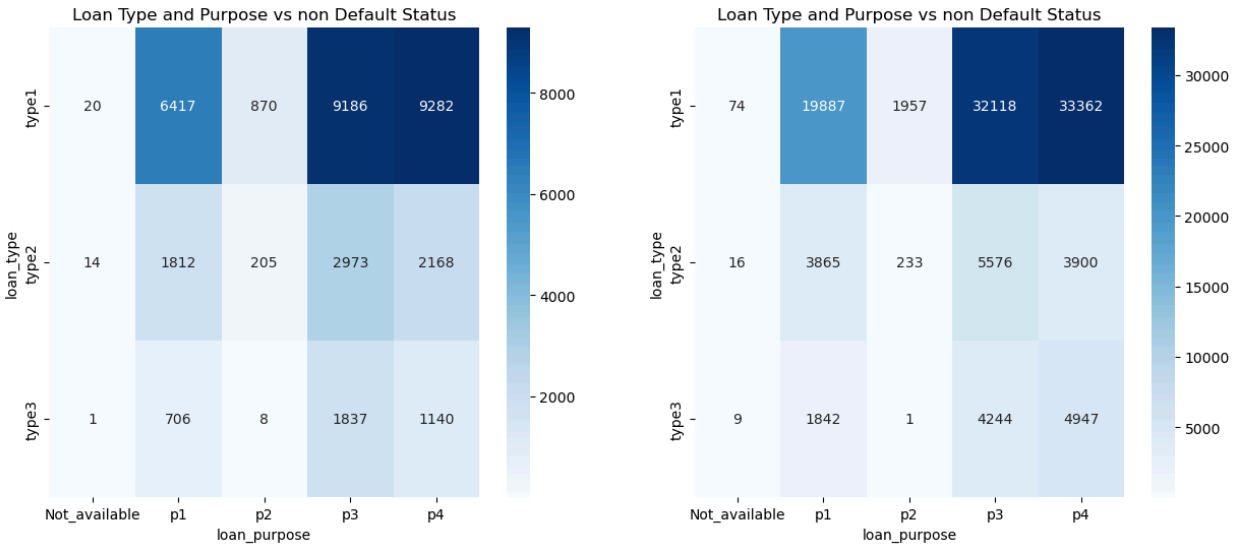
4.3.1. Analyzing Patterns in Loan Type, Purpose, and Default

We will create grouped bar charts to analyze how different loan types and purposes interact with default rates. This analysis will reveal any significant patterns in risk among various loan categories.

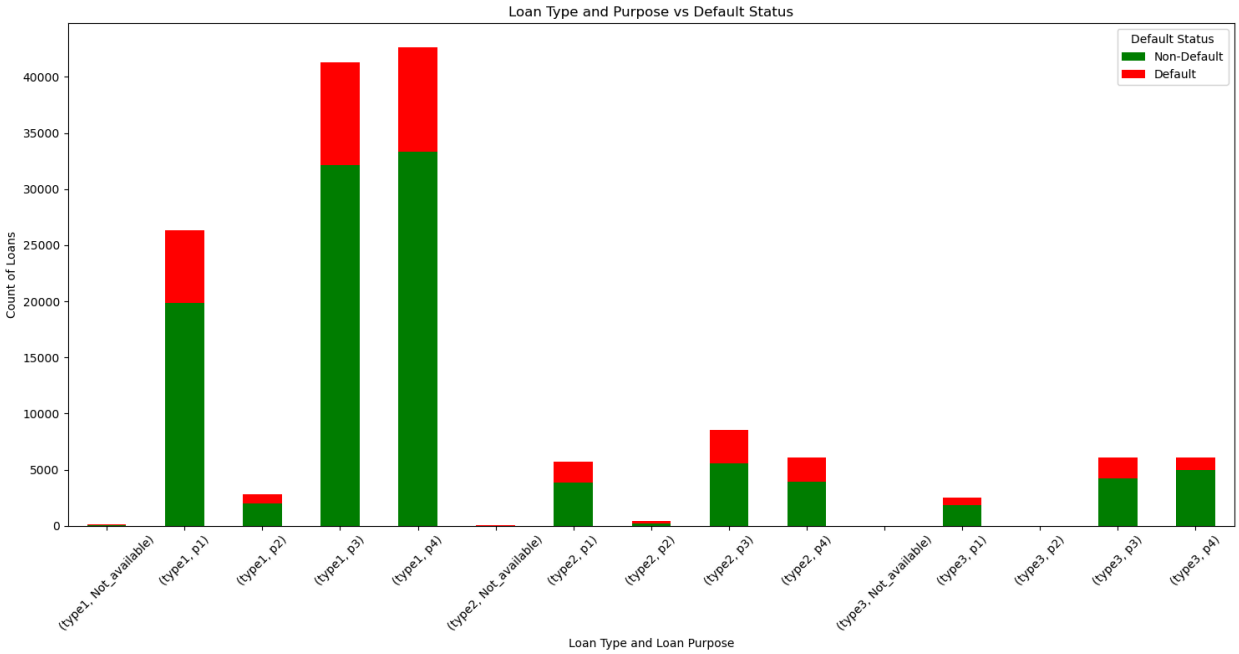
In [145…

```
# Grouping and plotting loan type, loan purpose, and default status
loan_type_purpose_default = pd.crosstab(defaulter_df['loan_type'], defaulter_df['loan_purpose'], margins=True)
print(loan_type_purpose_default)
# Grouping and plotting loan type, loan purpose, and non default status
loan_type_purpose_non_default = pd.crosstab(non_defaulter_df['loan_type'], non_defaulter_df['loan_purpose'], margins=True)
print(loan_type_purpose_non_default)

plt.figure(figsize=[15,6])

# Visualization: Heatmap of Loan Type and Loan Purpose vs. Default
plt.subplot(1,2,1)
sns.heatmap(pd.crosstab(defaulter_df['loan_type'], defaulter_df['loan_purpose']), annot=True, fmt='g', cmap='Blues')
plt.title('Loan Type and Purpose vs non Default Status')

# Visualization: Heatmap of Loan Type and Loan Purpose vs. non Default
plt.subplot(1,2,2)
sns.heatmap(pd.crosstab(non_defaulter_df['loan_type'], non_defaulter_df['loan_purpose']), annot=True, fmt='g', cmap='Blues
plt.title('Loan Type and Purpose vs non Default Status')
plt.show()
```

```
loan_purpose  Not_available    p1    p2     p3     p4    All
loan_type
type1                    20  6417   870   9186   9282  25775
type2                    14  1812   205   2973   2168   7172
type3                     1   706     8   1837   1140   3692
All                      35  8935  1083  13996  12590  36639
loan_purpose  Not_available     p1    p2     p3     p4     All
loan_type
type1                    74  19887  1957  32118  33362   87398
type2                    16   3865   233   5576   3900   13590
type3                     9   1842     1   4244   4947   11043
All                      99  25594  2191  41938  42209  112031
```



In [146…
```
# Assuming df is your dataframe
# Grouping by loan type, loan purpose, and status (default or not)
grouped_data = df.groupby(['loan_type', 'loan_purpose', 'Status']).size().unstack()
# Plotting the stacked bar chart
grouped_data.plot(kind='bar', stacked=True, color=['green', 'red'], figsize=(15, 8))

# Adding labels and title
plt.title('Loan Type and Purpose vs Default Status')
plt.xlabel('Loan Type and Loan Purpose')
plt.ylabel('Count of Loans')
plt.legend(title='Default Status', labels=['Non-Default', 'Default'])
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

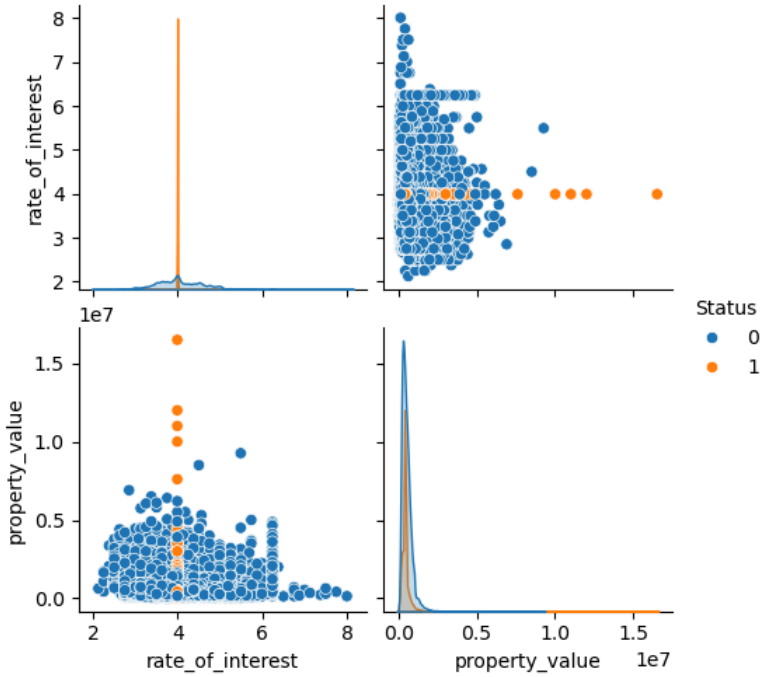


In [ ]:

4.3.2. Interaction between Interest Rates, Property Value, and Default

This analysis will involve creating 3D plots or heatmaps to visualize the interactions between interest rates, property values, and their combined effect on default rates.

In [148…
```
# pair plot
col_to_plot = ['rate_of_interest','property_value','Status']
df_col = df[col_to_plot]
sns.pairplot(data=df_col, hue='Status')
plt.show()
```



In [ ]:

4.4. Visualizations

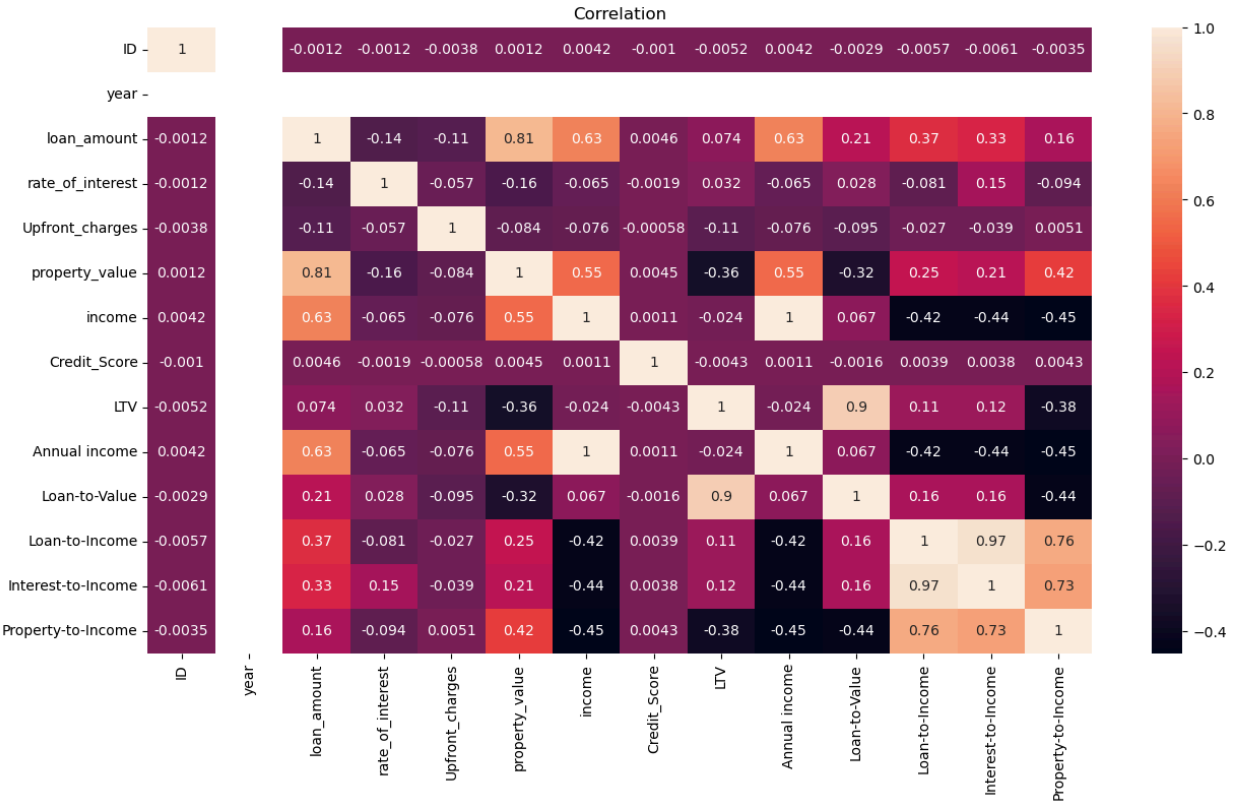
Visualizations play a crucial role in EDA by presenting data in an understandable format. We will utilize various types of visualizations to depict our findings, including:

4.4.1. Heatmaps for Correlation Analysis

Heatmaps will be created to visualize the correlation matrix among different features, allowing us to quickly identify strong relationships between variables.

In [150…

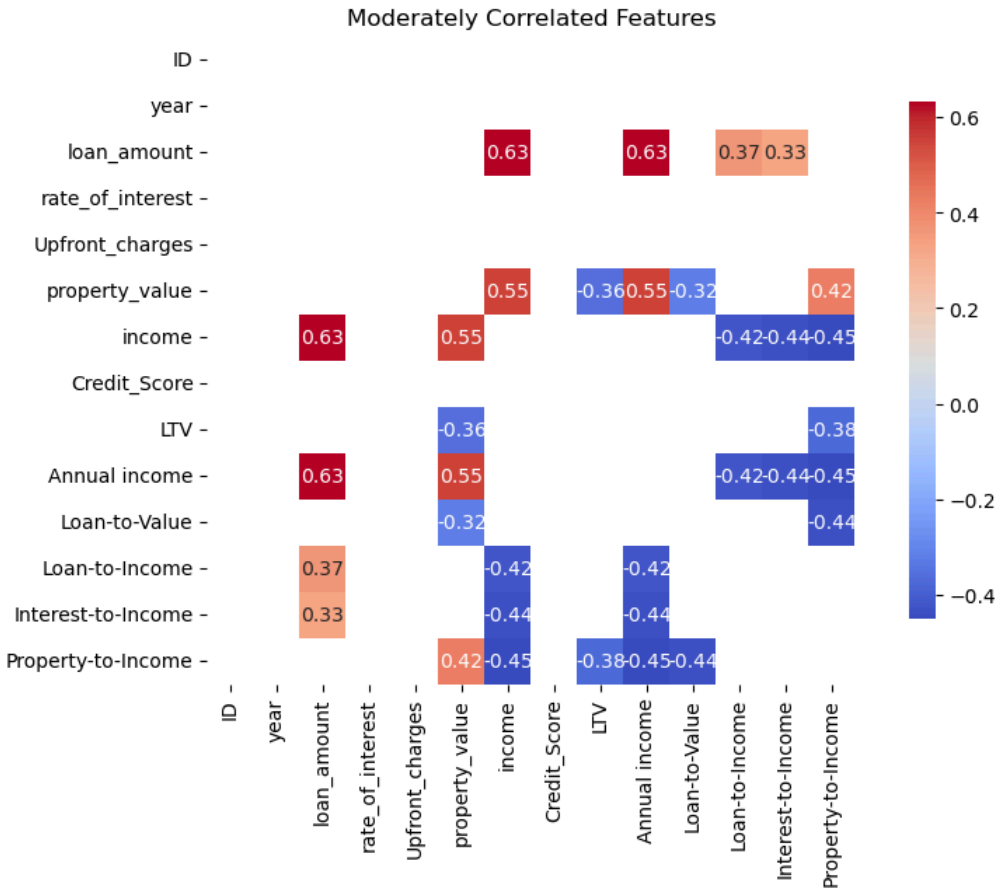
```
# we will sue spearman correlation as we have seen all the numerical data is non normal
plt.figure(figsize=[15,8])
sns.heatmap(numeric_df.corr(method='spearman'), annot=True)
plt.title('Correlation')
plt.show()
```



In [151…

```
# Calculate the correlation matrix
correlation_matrix = numeric_df.corr(method='spearman')

# Set correlation thresholds
lower_threshold = 0.3
upper_threshold = 0.7

# Filter for moderately correlated features (positive and negative)
moderate_corr = correlation_matrix[((correlation_matrix >= lower_threshold) |
                                    (correlation_matrix <= -lower_threshold)) &
                                   (correlation_matrix.abs() <= upper_threshold) &
                                   (correlation_matrix != 1)]

# Get pairs of features
moderate_corr_pairs = moderate_corr.stack().reset_index()
moderate_corr_pairs.columns = ['Feature_1', 'Feature_2', 'Correlation']

# Display the pairs of features with their correlation
print("Moderately Correlated Feature Pairs (0.3 to 0.7 and -0.3 to -0.7):")
for index, row in moderate_corr_pairs.iterrows():
    print(f"{row['Feature_1']} ↔ {row['Feature_2']}: Correlation = {row['Correlation']:.2f}")

# Optional: Visualize the moderate correlations
plt.figure(figsize=(10, 6))
sns.heatmap(moderate_corr, annot=True, cmap='coolwarm', fmt='.2f', square=True, cbar_kws={"shrink": .8})
plt.title("Moderately Correlated Features")
plt.show()
```

```
Moderately Correlated Feature Pairs (0.3 to 0.7 and -0.3 to -0.7):
loan_amount ↔ income: Correlation = 0.63
loan_amount ↔ Annual income: Correlation = 0.63
loan_amount ↔ Loan-to-Income: Correlation = 0.37
loan_amount ↔ Interest-to-Income: Correlation = 0.33
property_value ↔ income: Correlation = 0.55
property_value ↔ LTV: Correlation = -0.36
property_value ↔ Annual income: Correlation = 0.55
property_value ↔ Loan-to-Value: Correlation = -0.32
property_value ↔ Property-to-Income: Correlation = 0.42
income ↔ loan_amount: Correlation = 0.63
income ↔ property_value: Correlation = 0.55
income ↔ Loan-to-Income: Correlation = -0.42
income ↔ Interest-to-Income: Correlation = -0.44
income ↔ Property-to-Income: Correlation = -0.45
LTV ↔ property_value: Correlation = -0.36
LTV ↔ Property-to-Income: Correlation = -0.38
Annual income ↔ loan_amount: Correlation = 0.63
Annual income ↔ property_value: Correlation = 0.55
Annual income ↔ Loan-to-Income: Correlation = -0.42
Annual income ↔ Interest-to-Income: Correlation = -0.44
Annual income ↔ Property-to-Income: Correlation = -0.45
Loan-to-Value ↔ property_value: Correlation = -0.32
Loan-to-Value ↔ Property-to-Income: Correlation = -0.44
Loan-to-Income ↔ loan_amount: Correlation = 0.37
Loan-to-Income ↔ income: Correlation = -0.42
Loan-to-Income ↔ Annual income: Correlation = -0.42
Interest-to-Income ↔ loan_amount: Correlation = 0.33
Interest-to-Income ↔ income: Correlation = -0.44
Interest-to-Income ↔ Annual income: Correlation = -0.44
Property-to-Income ↔ property_value: Correlation = 0.42
Property-to-Income ↔ income: Correlation = -0.45
Property-to-Income ↔ LTV: Correlation = -0.38
Property-to-Income ↔ Annual income: Correlation = -0.45
Property-to-Income ↔ Loan-to-Value: Correlation = -0.44
```



Observations

Defaulter and Non-Defaulter Analysis

- Among defaulters, the heat map and stacked bar plot indicate that for loan type **Type 1**, the loan purposes **P1**, **P3**, and **P4** have the highest counts: **6417**, **9186**, and **9282** respectively.
- A similar pattern is observed in the non-defaulter dataset, where for loan type **Type 1**, the loan purposes **P1**, **P3**, and **P4** have counts of **19887**, **32118**, and **33362** respectively.
- From the pair plot, it is evident that property values mostly lie between **0** to **0.5e7** for interest rates between **2** to **6**.
- The sharp peak at **4** in the rate of interest is due to the imputation we performed on the rate of interest column using the median.

Correlation

Moderately Correlated Feature Pairs (0.3 to 0.7 and -0.3 to -0.7):

- **loan_amount** ↔ **income**: Correlation = **0.63**
- **loan_amount** ↔ **Annual income**: Correlation = **0.63**
- **loan_amount** ↔ **Loan-to-Income**: Correlation = **0.37**
- **loan_amount** ↔ **Interest-to-Income**: Correlation = **0.33**
- **property_value** ↔ **income**: Correlation = **0.55**
- **property_value** ↔ **LTV**: Correlation = **-0.36**
- **property_value** ↔ **Annual income**: Correlation = **0.55**
- **property_value** ↔ **Loan-to-Value**: Correlation = **-0.32**
- **property_value** ↔ **Property-to-Income**: Correlation = **0.42**
- **income** ↔ **Loan-to-Income**: Correlation = **-0.42**
- **income** ↔ **Interest-to-Income**: Correlation = **-0.44**
- **income** ↔ **Property-to-Income**: Correlation = **-0.45**
- **LTV** ↔ **Property-to-Income**: Correlation = **-0.38**
- **Annual income** ↔ **Loan-to-Income**: Correlation = **-0.42**
- **Annual income** ↔ **Interest-to-Income**: Correlation = **-0.44**
- **Loan-to-Value** ↔ **Property-to-Income**: Correlation = **-0.44**

In [ ]:

4.4.2. Pairplots for Multivariate Analysis

Pairplots will enable us to examine the relationships between multiple features simultaneously, providing insights into how they may interact.

In [154…
```
sns.pairplot(data=df)
```

Out[154…
```
<seaborn.axisgrid.PairGrid at 0x1e1ad2be030>
```

In [ ]:
```

```

4.4.3. Bar Charts and Pie Charts for Categorical Variables

We will use bar charts and pie charts to depict the frequency and proportions of categorical variables, aiding in the understanding of distribution patterns.

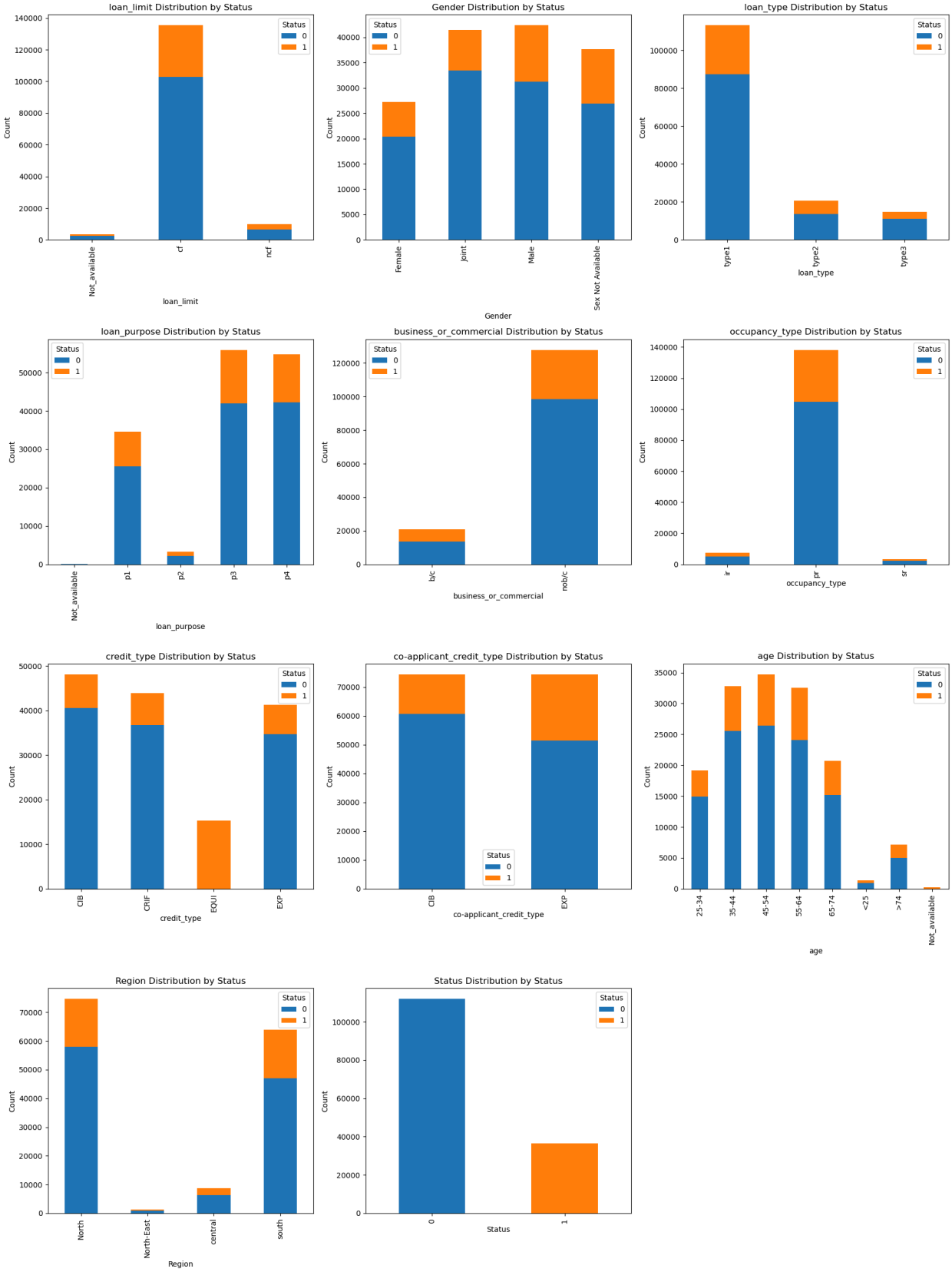
In [372…
```
# Extract object type (categorical variables) from the dataframe
categorical_columns = df.select_dtypes(include=['object']).columns

# Define the number of plots per row
plots_per_row = 3
num_cols = len(categorical_columns)

# Set up the plotting environment
fig, axes = plt.subplots((num_cols // plots_per_row) + 1, plots_per_row, figsize=(18, 6 * ((num_cols // plots_per_row) + 1

# Flatten the axes array for easier indexing in case of multiple rows
axes = axes.flatten()
```

```
# Loop through the categorical columns and create pie charts
for i, col in enumerate(categorical_columns):
    # Get the value counts for the categorical column
    value_counts = df[col].value_counts()

    # Create a pie chart for the column
    axes[i].pie(value_counts, labels=value_counts.index, autopct='%1.1f%%', startangle=90)
    axes[i].set_title(f'non defaulter {col} Distribution')

# Hide any unused subplots
for j in range(i+1, len(axes)):
    axes[j].axis('off')

# Adjust layout
plt.tight_layout()
plt.show()
```

In [376…

```
# Extract object type (categorical variables) from the dataframe
categorical_columns = df.select_dtypes(include=['object']).columns

# Define the number of plots per row
plots_per_row = 3
num_cols = len(categorical_columns)

# Set up the plotting environment
fig, axes = plt.subplots((num_cols // plots_per_row) + 1, plots_per_row, figsize=(18, 6 * ((num_cols // plots_per_row) + 1

# Flatten the axes array for easier indexing in case of multiple rows
axes = axes.flatten()

# Loop through the categorical columns and create stacked bar charts
for i, col in enumerate(categorical_columns):
```

```
    # Get value counts for both defaulters (status=1) and non-defaulters (status=0)
    grouped_counts = df.groupby([col, 'Status']).size().unstack(fill_value=0)

    # Create stacked bar plot for each categorical column
    grouped_counts.plot(kind='bar', stacked=True, ax=axes[i], color=['#1f77b4', '#ff7f0e'])

    # Set title and labels
    axes[i].set_title(f'{col} Distribution by Status')
    axes[i].set_ylabel('Count')
    axes[i].set_xlabel(col)

# Hide any unused subplots
for j in range(i + 1, len(axes)):
    axes[j].axis('off')

# Adjust layout
plt.tight_layout()
plt.show()
```



In [ ]:

In [ ]:

5. Hypothesis Testing Analysis

In this section, we conduct hypothesis testing to explore the relationships between key variables and loan default risk. Before choosing the appropriate statistical tests, we will first check the normality of the data to decide whether to use parametric or non-parametric methods.

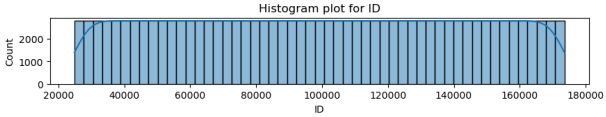
5.1. Formulating Hypotheses

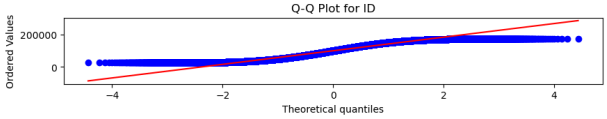
Normlaity check for all numerial features in the data set.

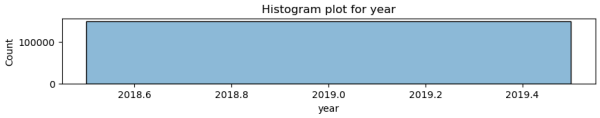
It is important because as per this we will formulate our hypothesis.

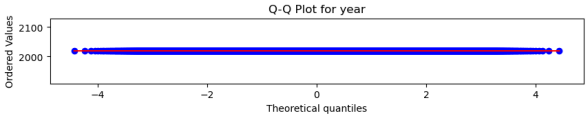
In [158…

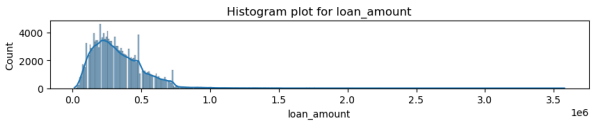
```
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats

cols_51 = numeric_df.columns
j=1
for i in cols_51:
    plt.figure(figsize=[22,22])
    # Create subplots for the same column from three different datasets side by side
    for y in range(j, j + 2):
        plt.subplot(15, 2, y)

        # scatte Plot
        if y == j:
            sns.histplot(data=numeric_df, x=i, kde=True)
            plt.title(f'Histogram plot for {i}')


        # violin Plot
        elif y == j + 1:
            # Q-Q plot for Loan Amount
            stats.probplot(numeric_df[i], dist="norm", plot=plt)
            plt.title(f'Q-Q Plot for {i}')


    # Move j forward by 3 to handle the next set of subplots
    j += 2

    # Adjust layout and display the subplots for the current column
    plt.show()
```

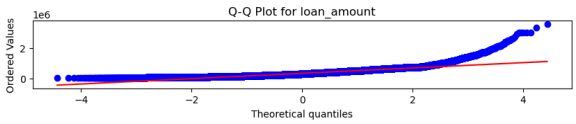


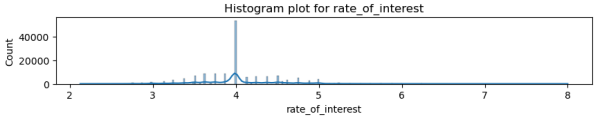


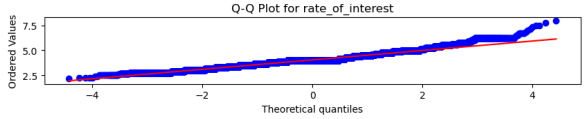


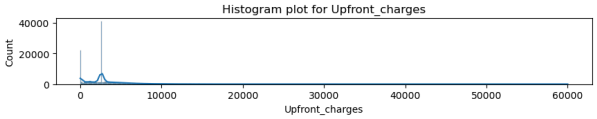


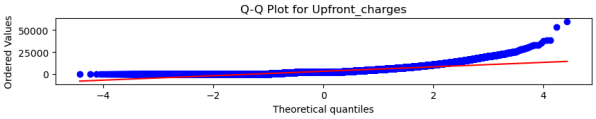


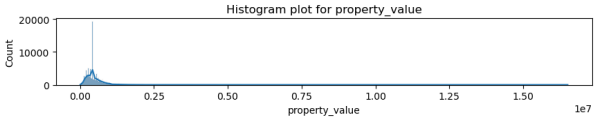


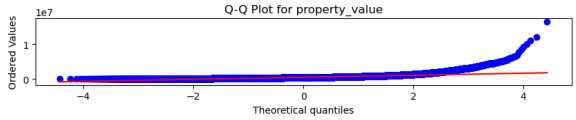


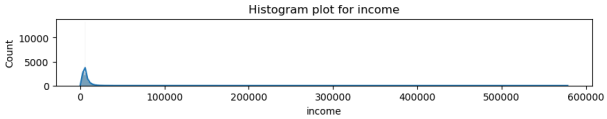


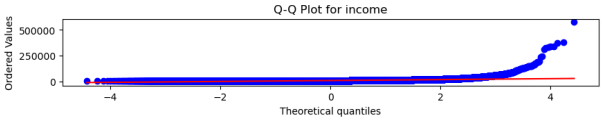


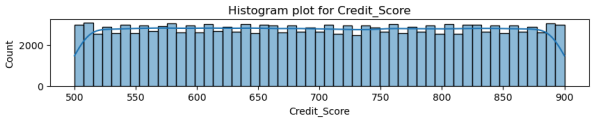


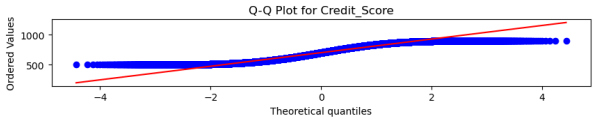


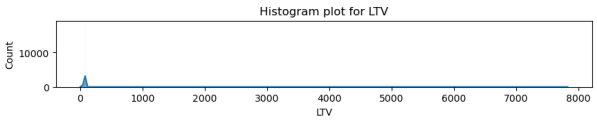


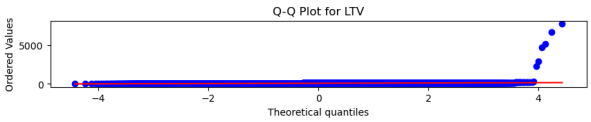


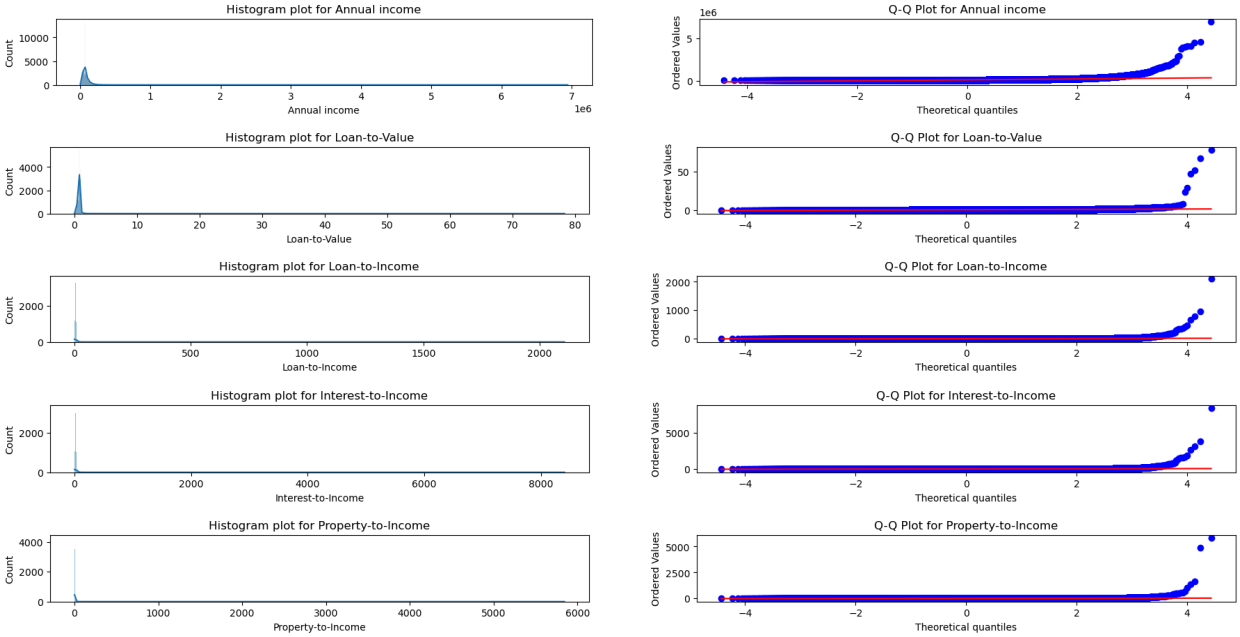












In [159…

```
from scipy.stats import kstest

for i in cols_51:
    # Perform the Shapiro-Wilk test
    stat, p_value = kstest(numeric_df[i], 'norm')

    print(f'Kolmogorov-Smirnov Test Statistic: {stat:.3f}, p-value: {p_value:.3f}')

    if p_value > 0.05:
        print(f"Fail to reject H₀: Data {i} is normally distributed")
        print('-----------------------------------------------------')
        print()
    else:
        print(f"Reject H₀: Data {i} is not normally distributed")
        print('-----------------------------------------------------')
        print()
```

```
Kolmogorov-Smirnov Test Statistic: 1.000, p-value: 0.000
Reject H₀: Data ID is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 1.000, p-value: 0.000
Reject H₀: Data year is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 1.000, p-value: 0.000
Reject H₀: Data loan_amount is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 0.997, p-value: 0.000
Reject H₀: Data rate_of_interest is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 0.860, p-value: 0.000
Reject H₀: Data Upfront_charges is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 1.000, p-value: 0.000
Reject H₀: Data property_value is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 1.000, p-value: 0.000
Reject H₀: Data income is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 1.000, p-value: 0.000
Reject H₀: Data Credit_Score is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 1.000, p-value: 0.000
Reject H₀: Data LTV is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 1.000, p-value: 0.000
Reject H₀: Data Annual income is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 0.584, p-value: 0.000
Reject H₀: Data Loan-to-Value is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 0.905, p-value: 0.000
Reject H₀: Data Loan-to-Income is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 0.994, p-value: 0.000
Reject H₀: Data Interest-to-Income is not normally distributed
--------------------------------------------------

Kolmogorov-Smirnov Test Statistic: 0.949, p-value: 0.000
Reject H₀: Data Property-to-Income is not normally distributed
--------------------------------------------------
```

In [ ]:

5.1.1. Null and Alternative Hypotheses Statements

For each hypothesis, we define:

- **H0 (Null Hypothesis)**: There is no significant effect or relationship between the variables.
- **H1 (Alternative Hypothesis)**: There is a significant effect or relationship between the variables.

5.1.2. Hypothesis Testing for Loan Amount and Default Rates

- **H0**: There is no significant difference in the loan amount between defaulters and non-defaulters.
- **H1**: Defaulters have a significantly different loan amount compared to non-defaulters.

5.1.3. Hypothesis Testing for Credit Score and Default Probability

- **H0**: There is no significant difference in credit scores between defaulters and non-defaulters.
- **H1**: Defaulters have significantly different credit scores compared to non-defaulterst.

5.1.4. Hypothesis Testing for Interest Rates and Default Risk

- **H0**: There is no significant difference in interest rates between defaulters and non-defaulters.
- **H1**: Defaulters have significantly different interest rates compared to non-defaulters.

5.1.5. Hypothesis Testing for Derived Features and Default Risk

- **H0**: There is no significant difference in default rates across different levels of the derived feature.
- **H1**: There is a significant difference in default rates across different levels of the derived feature.

5.1.6. Hypothesis Testing for Categorical Features and Default status

- **H0**: Categorical features are independent of Default status.
- **H1**: Both Categorical features are dependenent on each others.

In [ ]:

5.3. Hypothesis Testing Process

5.3.1. Assumptions of the Tests

Like we have seen all the numerical data is non normal. So we have take non parametric approach.

5.3.2. Performing the Tests

In [ ]:

In [162…

```
from scipy.stats import mannwhitneyu

for i in cols_51:
    # Perform the Shapiro-Wilk test
    stat, p_value = mannwhitneyu(defaulter_df[i], non_defaulter_df[i])

    print(f'Mann-Whitney U Statistic: {stat:.3f}, p-value: {p_value:.3f}')

    if p_value > 0.05:
        print(f"Fail to reject H0: No significant difference in {i} between defaulters and non-defaulters.")
        print('-----------------------------------------------------')
        print()
    else:
        print(f"Reject H0: There is a significant difference in {i} between defaulters and non-defaulters.")
        print('-----------------------------------------------------')
        print()
```

```
Mann-Whitney U Statistic: 2057034853.000, p-value: 0.511
Fail to reject H0: No significant difference in ID between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 2052351904.500, p-value: 1.000
Fail to reject H0: No significant difference in year between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 1863099561.500, p-value: 0.000
Reject H0: There is a significant difference in loan_amount between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 2031091093.500, p-value: 0.002
Reject H0: There is a significant difference in rate_of_interest between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 2047669152.000, p-value: 0.507
Fail to reject H0: No significant difference in Upfront_charges between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 1812627381.500, p-value: 0.000
Reject H0: There is a significant difference in property_value between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 1747825267.500, p-value: 0.000
Reject H0: There is a significant difference in income between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 2063311078.500, p-value: 0.124
Fail to reject H0: No significant difference in Credit_Score between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 2253221513.000, p-value: 0.000
Reject H0: There is a significant difference in LTV between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 1747825267.500, p-value: 0.000
Reject H0: There is a significant difference in Annual income between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 2212052167.500, p-value: 0.000
Reject H0: There is a significant difference in Loan-to-Value between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 2194255755.500, p-value: 0.000
Reject H0: There is a significant difference in Loan-to-Income between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 2185061766.000, p-value: 0.000
Reject H0: There is a significant difference in Interest-to-Income between defaulters and non-defaulters.
----------------------------------------------------

Mann-Whitney U Statistic: 2124057893.000, p-value: 0.000
Reject H0: There is a significant difference in Property-to-Income between defaulters and non-defaulters.
----------------------------------------------------
```

In [163…
```
col53_df = df.select_dtypes(include=['object'])
```

In [164…
```
col53_df.columns
```

Out[164…
```
Index(['loan_limit', 'Gender', 'loan_type', 'loan_purpose',
       'business_or_commercial', 'occupancy_type', 'credit_type',
       'co-applicant_credit_type', 'age', 'Region', 'Status'],
      dtype='object')
```

In [165…
```
cols_53 = ['loan_limit', 'Gender', 'loan_type', 'loan_purpose',
       'business_or_commercial', 'occupancy_type', 'credit_type',
       'co-applicant_credit_type', 'age', 'Region']
```

In [ ]:
```

```

In [166…
```
from scipy.stats import chi2_contingency

for i in cols_53:
    table = pd.crosstab(df[i],df['Status'])
    # Perform the Shapiro-Wilk test
    chi2_stat, p_val, dof, expected = chi2_contingency(table)

    # Output results
    print(f"Chi-Square Statistic: {chi2_stat}")
    print(f"P-Value: {p_val}")
    print(f"Degrees of Freedom: {dof}")
    print("Expected Frequencies:")
    print(expected)
```

```
# Interpretation
alpha = 0.05  # Significance level
if p_val < alpha:
    print(f"Reject the null hypothesis. There is a significant association between {i} and Default Status.")
    print('-----------------------------------------------------------------------------------------------')
    print()
else:
    print(f"Fail to reject the null hypothesis. No significant association between {i} and Default Status.")
    print('-----------------------------------------------------------------------------------------------')
    print()
```

```
Chi-Square Statistic: 432.76984350844157
P-Value: 1.0597965833618341e-94
Degrees of Freedom: 2
Expected Frequencies:
[[  2519.88742853    824.11257147]
 [101992.14224793  33355.85775207]
 [  7518.97032354   2459.02967646]]
Reject the null hypothesis. There is a significant association between loan_limit and Default Status.
----------------------------------------------------------------------------------------------------

Chi-Square Statistic: 1043.6008495787444
P-Value: 6.260048330360408e-226
Degrees of Freedom: 3
Expected Frequencies:
[[20546.42662272  6719.57337728]
 [31196.4173606  10202.5826394 ]
 [31910.03380642 10435.96619358]
 [28378.12221026  9280.87778974]]
Reject the null hypothesis. There is a significant association between Gender and Default Status.
----------------------------------------------------------------------------------------------------

Chi-Square Statistic: 1309.9581425319489
P-Value: 3.5172528245408e-285
Degrees of Freedom: 2
Expected Frequencies:
[[85282.06338199 27890.93661801]
 [15645.30585861  5116.69414139]
 [11103.6307594   3631.3692406 ]]
Reject the null hypothesis. There is a significant association between loan_type and Default Status.
----------------------------------------------------------------------------------------------------

Chi-Square Statistic: 240.36219281696617
P-Value: 7.752608950247793e-51
Degrees of Freedom: 4
Expected Frequencies:
[[1.00976350e+02 3.30236497e+01]
 [2.60194955e+04 8.50950448e+03]
 [2.46713859e+03 8.06861411e+02]
 [4.21493371e+04 1.37846629e+04]
 [4.12940524e+04 1.35049476e+04]]
Reject the null hypothesis. There is a significant association between loan_purpose and Default Status.
----------------------------------------------------------------------------------------------------

Chi-Square Statistic: 1272.8079977681857
P-Value: 9.172191110452255e-279
Degrees of Freedom: 1
Expected Frequencies:
[[15645.30585861  5116.69414139]
 [96385.69414139 31522.30585861]]
Reject the null hypothesis. There is a significant association between business_or_commercial and Default Status.
----------------------------------------------------------------------------------------------------

Chi-Square Statistic: 131.82535899539891
P-Value: 2.368574267249222e-29
Degrees of Freedom: 2
Expected Frequencies:
[[  5531.09262124   1808.90737876]
 [104142.03424363  34058.96575637]
 [  2357.87313513    771.12686487]]
Reject the null hypothesis. There is a significant association between occupancy_type and Default Status.
----------------------------------------------------------------------------------------------------

Chi-Square Statistic: 52135.280704971796
P-Value: 0.0
Degrees of Freedom: 3
Expected Frequencies:
[[36285.17328311 11866.82671689]
 [33081.81160288 10819.18839712]
 [11527.88214166  3770.11785834]
 [31136.13297235 10182.86702765]]
Reject the null hypothesis. There is a significant association between credit_type and Default Status.
----------------------------------------------------------------------------------------------------

Chi-Square Statistic: 3092.3925712326145
P-Value: 0.0
Degrees of Freedom: 1
Expected Frequencies:
[[56058.45262662 18333.54737338]
 [55972.54737338 18305.45262662]]
Reject the null hypothesis. There is a significant association between co-applicant_credit_type and Default Status.
----------------------------------------------------------------------------------------------------

Chi-Square Statistic: 985.8249006099044
P-Value: 1.39121687677485e-208
Degrees of Freedom: 7
Expected Frequencies:
```

```
[[14424.5469967   4717.4530033 ]
 [24730.16316675  8087.83683325]
 [26163.42449721  8556.57550279]
 [24516.15358848  8017.84641152]
 [15631.74187126  5112.25812874]
 [ 1007.5028385    329.4971615 ]
 [ 5406.75607049  1768.24392951]
 [  150.71097061    49.28902939]]
Reject the null hypothesis. There is a significant association between age and Default Status.
----------------------------------------------------------------------------------------------------

Chi-Square Statistic: 380.45633008939643
P-Value: 3.7860568405811336e-82
Degrees of Freedom: 3
Expected Frequencies:
[[56307.12572812 18414.87427188]
 [  930.64024349   304.35975651]
 [ 6553.6665568   2143.3334432 ]
 [48239.56747158 15776.43252842]]
Reject the null hypothesis. There is a significant association between Region and Default Status.
----------------------------------------------------------------------------------------------------
```

Observations

- According to the KS test, none of the numerical features are normally distributed.
- Based on the Mann-Whitney U test, there are significant differences between defaulters and non-defaulters in the following features:
  - Loan Amount
  - Rate of Interest
  - Property Value
  - Income
  - Lifetime Value
  - Loan to Value
  - Loan to Income
  - Interest to Income
  - Pro perty to Income
- According to the Chi-squared test of independence, there are significant differences between the following categorical features and default status:
  - Loan Limit
  - Gender
  - Loan Type
  - Loan Purpose
  - Business or Commercial
  - Occupancy Type
  - Credit Type
  - Co-Applicann

Type

- Age
- Region

In [ ]:

5.4. **Results and Conclusions**

5.4.1. Business Implications of Findings

Business Implications of Findings

1. Non-Normality of Numerical Features

- **Implication**: Since none of the numerical features are normally distributed, traditional statistical techniques that assume normality (e.g., linear regression) may not be suitable for analyzing these features.
- **Action**: Consider using non-parametric methods or data transformations (e.g., log transformation) to address this non-normality. Machine learning algorithms that do not assume normality (like tree-based methods) can also be employed for predictive modeling.

2. Significant Differences Detected by Mann-Whitney U Test

- **Implication**: The significant differences in loan amount, rate of interest, property value, income, lifetime value, loan-to-value ratio, loan-to-income ratio, interest-to-income ratio, and property-to-income ratio between defaulters and non-defaulters suggest that these metrics are essential indicators of credit risk.
- **Action**:
  - **Refine Risk Assessment Models**: Incorporate these features into risk assessment models to better predict default risk.
  - **Adjust Lending Criteria**: Set more stringent criteria for applicants with lower incomes or higher loan-to-value ratios, as these factors are correlated with a higher likelihood of default.
  - **Develop Targeted Financial Products**: Create tailored loan products or interest rates for different segments based on their risk profile, informed by these variables.

3. Significant Differences Detected by Chi-Squared Test

- **Implication**: The significant associations between default status and categorical features such as loan limit, gender, loan type, loan purpose, business or commercial nature, occupancy type, credit type, co-applicant type, age, and region indicate that these factors are crucial in understanding borrower behavior and credit risk.
- **Action**:
  - **Targeted Marketing and Underwriting Strategies**: Use this information to target marketing efforts more effectively and tailor underwriting processes based on demographic and categorical factors. For instance, understanding which regions have higher default rates can guide regional lending strategies.
  - **Policy Development**: Consider creating specific lending policies for different demographic segments (e.g., age groups or genders) to address their unique risk profiles.
  - **Monitoring and Reporting**: Establish monitoring systems to continuously track how these categorical factors influence default rates, adjusting strategies as needed.

4. Need for Quantile-Based Risk Segmentation

- **Rationale**: Given the skewed distribution of numerical features, traditional risk segmentation methods may not accurately capture the risk profile of borrowers. Quantile-based segmentation allows for a more nuanced understanding of risk across different levels of each feature.
- **Benefits**:
  - **Improved Risk Assessment**: By segmenting data into quantiles (e.g., quartiles or quintiles), institutions can identify critical thresholds for risk indicators, allowing for tailored risk assessments based on borrower characteristics.
  - **Enhanced Targeting**: Quantile segmentation can help in pinpointing high-risk segments that may not be apparent through mean-based approaches, leading to more effective marketing and lending strategies.
  - **Adaptive Strategies**: This approach allows for the development of dynamic risk management strategies that can adjust to changes in borrower behavior and economic conditions, ensuring ongoing relevance and accuracy.

5. Broader Risk Management Strategies

- **Implication**: The combination of significant differences in both numerical and categorical features can provide a more holistic view of default risk.
- **Action**:
  - **Integrated Risk Framework**: Develop an integrated risk management framework that includes both quantitative and qualitative analyses to understand the full picture of credit risk.
  - **Regular Review and Adjustment**: Continuously review and adjust lending policies based on evolving patterns in the data to mitigate risk effectively.

Conclusion

Overall, these findings emphasize the need for a data-driven approach to credit risk assessment and management. Financial institutions can improve their lending strategies, enhance customer targeting, and develop more effective risk mitigation strategies by leveraging these insights. Implementing these actions can ultimately lead to reduced default rates, improved profitability, and better customer relationships. er customer relationships.

In [ ]:

6. Advanced Analysis and Research

In this section, we will perform more in-depth research and analysis to explore broader economic trends, geographical factors, and time-based cohort behaviors that influence loan defaults. The goal is to supplement the previous risk analysis and provide well-rounded recommendations based on a variety of external and internal factors. Here's a breakdown of each subpoint:

6.1 Additional Factors Affecting Default (literature review)to reduce loan default risks.

Objective:

This section explores additional factors influencing loan defaults that are not covered directly by the dataset but are highlighted in academic and industry research. These factors may include psychological, behavioral, cultural, and regulatory influences that could provide deeper insights into loan default risks.

1. Behavioral Factors

- **Borrower Psychology**: Studies suggest that borrower psychology, particularly impulsive behavior and optimism bias, can increase the risk of default. Borrowers may underestimate their future financial challenges, leading to overconfidence in taking out loans that they struggle to repay.
  - *Application*: Analyzing risk from this perspective could suggest that credit counseling and financial awareness programs may mitigate default risks by addressing behavioral biases.
- **Financial Literacy**: Limited financial literacy is often linked to higher default rates. Borrowers who lack understanding of loan terms, interest rates, and repayment schedules are more likely to make uninformed decisions, leading to defaults.
  - *Application*: The dataset could incorporate proxies for financial literacy (e.g., borrower education level) to estimate its impact on default risk.

2. Policy and Regulatory Factors

- **Government Regulations**: Policies governing loan eligibility, interest rates, and credit accessibility play a significant role in default rates. Deregulated lending environments may lead to riskier lending practices and higher default rates, as seen during the subprime mortgage crisis.
  - *Application*: Examining the regulatory environment during the period covered by the dataset could reveal patterns in how changes to credit policies influenced defaults.
- **Loan Forgiveness and Relief Programs**: Government loan relief programs (e.g., during economic downturns) have historically reduced default rates by providing temporary financial relief to struggling borrowers.
  - *Application*: The dataset could be enriched with data about government interventions during the observed period to assess their effects on default rates.

3. Cultural and Societal Factors

- **Cultural Attitudes Toward Debt**: In certain cultures, social stigma attached to debt or default may deter borrowers from missing payments, while in others, the attitude toward debt might be more lenient, leading to higher default rates.
  - *Application*: Segmenting the data based on geographic or regional cultural influences could provide insights into how cultural differences affect default behavior.
- **Social Networks and Peer Influence**: Research indicates that peer influence can affect borrowing and repayment behavior, particularly in close-knit communities. Borrowers may follow repayment behaviors of their peers, either increasing or reducing default rates based on the community's norms.
  - *Application*: Data on social networks or communities could provide valuable insights into how peer behavior influences default patterns.

4. Economic and Environmental Factors

- **Macroeconomic Conditions**: Economic downturns, such as recessions, significantly increase default rates. Borrowers facing unemployment or reduced income are more likely to default on their loans.
  - *Application*: Including macroeconomic data like unemployment rates or GDP trends during the loan period could provide a clearer picture of external pressures contributing to defaults.
- **Environmental Shocks**: Events such as natural disasters can increase loan default rates, as affected borrowers may face sudden financial hardships. For example, after Hurricane Katrina, many individuals defaulted due to loss of property and livelihood.
  - *Application*: Adding data on environmental or disaster-related factors could enhance risk predictions, particularly for regions more prone to such events.

Conclusion:

The literature review highlights that a variety of external factors contribute to loan defaults, ranging from borrower psychology to macroeconomic conditions. Incorporating these insights into your dataset or considering them when analyzing the risk of defaults will provide a more holistic understanding of the issue. While not all of these factors may be quantifiable in your dataset, recognizing their influence will guide further analysis and potentially inform strategies to reduce loan default risks.

- Journal - *Macroeconomic determinants of loan defaults: Evidence from the U.S. peer-to-peer lending market*

In [ ]:

6.2 Risk Segmentation for customer

In this project, risk segmentation is performed using a quantile-based method due to the skewed nature of the data. By dividing the data into quantiles, we categorize customers into distinct risk groups—high, medium, and low—based on key variables like credit score, LTV,

and loan amount. This approach helps ensure a more balanced segmentation by evenly distributing customers across risk categories, allowing us to identify and focus on the most critical segments, especially those with a higher default probability. This method provides a more robust way to assess risk compared to traditional methods, particularly in skewed datasets.

In [173…

```
# risk_df = pd.DataFrame()
# risk_df['Status'] = df['Status']
# # DataFrame risk_df (replace this with your actual data)
# # risk_df = pd.read_csv('path_to_your_dataset.csv')  # Uncomment to load your dataset

# # Step 1: Define functions to calculate risk scores for each feature

# # Credit Score Risk Levels
# def credit_score_risk(score):
#     if score > 700:
#         return 1  # Low Risk
#     elif 580 <= score <= 700:
#         return 2  # Medium Risk
#     else:
#         return 3  # High Risk

# # Loan Amount Risk Levels
# def loan_amount_risk(amount):
#     if amount < 200000:
#         return 1  # Low Risk
#     elif 200000 <= amount <= 500000:
#         return 2  # Medium Risk
#     else:
#         return 3  # High Risk

# # Rate of Interest Risk Levels
# def interest_rate_risk(rate):
#     if rate < 3.5:
#         return 1  # Low Risk
#     elif 3.5 <= rate <= 5:
#         return 2  # Medium Risk
#     else:
#         return 3  # High Risk

# # Property Value Risk Levels
# def property_value_risk(value):
#     if value > 200000:
#         return 1  # Low Risk
#     elif 100000 <= value <= 200000:
#         return 2  # Medium Risk
#     else:
#         return 3  # High Risk

# # Income Risk Levels
# def income_risk(income):
#     if income > 25000:
#         return 1  # Low Risk
#     elif 10000 <= income <= 25000:
#         return 2  # Medium Risk
#     else:
#         return 3  # High Risk

# # Loan to Value (LTV) Risk Levels
# def ltv_risk(ltv):
#     if ltv < 0.7:
#         return 1  # Low Risk
#     elif 0.7 <= ltv <= 0.9:
#         return 2  # Medium Risk
#     else:
#         return 3  # High Risk

# # Loan Limit Risk Levels (cf = low risk, ncf = high risk)
# def loan_limit_risk(limit):
#     if limit == 'cf':
#         return 1  # Low Risk
#     else:
#         return 3  # High Risk

# # Gender Risk Levels
# def gender_risk(gender):
#     if gender == 'Male' or gender == 'Female':
#         return 1  # Low Risk
#     elif gender == 'Not Specified':
#         return 2  # Medium Risk
#     else:
#         return 3  # High Risk (Joint)

# # Loan Type Risk Levels (based on masked types)
# def loan_type_risk(loan_type):
#     if loan_type == 'type-1':
```

```
#         return 3  # High Risk
#     elif loan_type == 'type-2':
#         return 1  # low Risk
#     else:
#         return 1  # low Risk (type-3)

# # Age Group Risk Levels (age groups)
# def age_risk(age_group):
#     if age_group in ['35-44','45-54']:
#         return 1  # Low Risk
#     elif age_group in ['25-34', '55-64']:
#         return 2  # Medium Risk
#     else:
#         return 3  # High Risk (< 25 or > 65)

# # Step 2: Apply risk scoring to the dataset risk_df
# risk_df['Credit_Score_Risk'] = df['Credit_Score'].apply(credit_score_risk)
# risk_df['Loan_Amount_Risk'] = df['loan_amount'].apply(loan_amount_risk)
# risk_df['Rate_of_Interest_Risk'] = df['rate_of_interest'].apply(interest_rate_risk)
# risk_df['Property_Value_Risk'] = df['property_value'].apply(property_value_risk)
# risk_df['Income_Risk'] = df['income'].apply(income_risk)
# risk_df['LTV_Risk'] = df['Loan-to-Value'].apply(ltv_risk)
# risk_df['Loan_Limit_Risk'] = df['loan_limit'].apply(loan_limit_risk)
# risk_df['Gender_Risk'] = df['Gender'].apply(gender_risk)
# risk_df['Loan_Type_Risk'] = df['loan_type'].apply(loan_type_risk)
# risk_df['Age_Risk'] = df['age'].apply(age_risk)

# # Step 3: Create a total risk score by summing all the individual risk scores
# risk_df['Total_Risk_Score'] = (
#     risk_df['Credit_Score_Risk'] +
#     risk_df['Loan_Amount_Risk'] +
#     risk_df['Rate_of_Interest_Risk'] +
#     risk_df['Property_Value_Risk'] +
#     risk_df['Income_Risk'] +
#     risk_df['LTV_Risk'] +
#     risk_df['Loan_Limit_Risk'] +
#     risk_df['Gender_Risk'] +
#     risk_df['Loan_Type_Risk'] +
#     risk_df['Age_Risk']
# )

# # Step 4: Define risk categories based on Total Risk Score
# def risk_category(total_score):
#     if total_score <= 12:
#         return 'Low Risk'
#     elif 13 <= total_score <= 20:
#         return 'Medium Risk'
#     else:
#         return 'High Risk'

# # Apply the risk category function
# risk_df['Risk_Category'] = risk_df['Total_Risk_Score'].apply(risk_category)

# # Step 5: View the risk segmentation table
# # print(risk_df[['Credit_Score_Risk', 'Loan_Amount_Risk', 'Rate_of_Interest_Risk',
# #                'Property_Value_Risk', 'Income_Risk', 'LTV_Risk', 'Loan_Limit_Risk',
# #                'Gender_Risk', 'Loan_Type_Risk', 'Age_Risk', 'Total_Risk_Score', 'Risk_Category']])

# # Optional: Save the result to a new CSV file
# # risk_df.to_csv('risk_segmentation_results.csv', index=False)
```

Reason Behind Choosing Thresholds for Binning and Risk Segmentation

1. Quantile-Based Binning (25th and 60th Percentiles)

- **Why Quantiles?**
  Quantiles (percentiles) allow for an even distribution of data into different bins, ensuring that each bin contains an equal or near-equal number of observations. This method is particularly useful when the data is skewed, as it creates balanced segments regardless of the underlying data distribution.

- **Why 25th and 60th Percentiles?**

  - The **25th percentile** captures the lower end of the data distribution, separating the "low-risk" group.
  - The **60th percentile** marks the point where the bulk of the data lies, helping to differentiate between "medium-risk" and "high-risk" segments.

  These cutoffs were chosen because:

  - The **25th percentile** represents customers with favorable characteristics (low loan amounts, high credit scores, low LTV, etc.), making them low risk.
  - The **60th percentile** marks the point where moderate risk begins to emerge, distinguishing between medium-risk and high-risk customers.

2. Reversing Quantiles for Some Variables

For some variables, such as **credit score** and **income**, higher values indicate **lower risk** (e.g., higher credit scores and higher incomes suggest a greater ability to repay loans). For others, like **loan amount** or **LTV**, higher values indicate **higher risk**.

- **Why Reverse Quantiles?**
  Inverting the risk scoring for certain variables (like credit score) aligns the risk categories with business logic:
  - Higher credit scores should fall into **low-risk** categories.
  - Larger loan amounts or higher LTVs fall into **high-risk** categories.

By reversing the quantile labels (assigning 1 for high values and 3 for low values), the segmentation becomes consistent across features with different risk implications.

3. Binning Categorical Variables

- **Loan Limit**: Customers with a loan limit type 'cf' (conforming) are treated as low-risk (1), while non-conforming (ncf) are higher risk (3), reflecting the industry understanding that non-conforming loans tend to carry higher default risk.
- **Gender**: Male and female borrowers are categorized as lower risk (1) because demographic analysis showed lower default rates compared to other gender types (joint or unknown borrowers). This segmentation helps identify low-risk groups.
- **Loan Type**: Loan types were binned based on observed default trends in the dataset, assigning higher risk to certain loan types (type-3, type-2) compared to type-1, which was more stable.

4. Age Risk Segmentation

- **Why Assign Age-Based Risk?**
  Age groups were assigned risk levels based on general trends observed in loan repayment behavior:
  - **Younger borrowers (<25)** tend to have less financial stability, making them more likely to default (high risk).
  - **Borrowers aged 25-44** often have more stable income and financial profiles, hence they were categorized as low risk.
  - **Older borrowers (65+)** may have fixed incomes or retirement plans, increasing their default risk, so they were categorized as higher risk.

This segmentation reflects real-world patterns of financial stability across different life stages.

5. Total Risk Score Thresholds

- **Why Choose a Threshold of 12 and 18?**
  - **Total Score <= 12** was chosen to represent **low risk**, as this segment likely includes customers with favorable characteristics across all variables.
  - **Score 13-18** captures the **medium-risk** segment, where customers might have moderate scores in some areas but not enough to be classified as high risk.
  - **Score > 18** represents **high-risk** customers, likely those with poor credit scores, high LTV, and low incomes.

These thresholds were set based on the combined risk scores from individual variables, balancing the need for differentiation between risk categories while keeping the thresholds consistent with the observed data patterns.

Summary of Reasoning

The quantile-based binning and the specific thresholds for risk segmentation were chosen to reflect the natural distribution of the data while aligning with common risk factors in loan default analysis. The use of quantiles ensures even distribution, while the business logic for reversing certain variables and setting category-specific thresholds captures the varying risk profiles of different customer segments. This allows for a nuanced segmentation strategy that can inform targeted interventions for high-risk customers and offer competitive products to low-risk customers.

In [174…

```
# Load your dataset (replace with actual file path)
# df = pd.read_csv('path_to_your_dataset.csv')
risk_df = pd.DataFrame()
risk_df['Status'] = df['Status']  # Initialize risk_df with the Status column from the original DataFrame

# Step 1: Define functions to calculate risk scores using quantile binning

def quantile_based_risk(series):
    # Define quantiles
    quantiles = series.quantile([0.25, 0.6]).tolist()
    # Add -inf and inf for proper binning
    quantiles.insert(0, -np.inf)
    quantiles.append(np.inf)

    # Assign risk levels based on quantiles
    return pd.cut(series, bins=quantiles, labels=[1, 2, 3])  # 1: Low Risk, 2: Medium Risk, 3: High Risk

def quantile_based_risk_rev(series):
    # Define quantiles
    quantiles = series.quantile([0.25, 0.6]).tolist()
```

```
    # Add -inf and inf for proper binning
    quantiles.insert(0, -np.inf)
    quantiles.append(np.inf)

    # Assign risk levels based on quantiles
    return pd.cut(series, bins=quantiles, labels=[3, 2, 1])

# Step 2: Apply quantile-based risk scoring to relevant features in risk_df

risk_df['Credit_Score_Risk'] = quantile_based_risk_rev(df['Credit_Score'])
risk_df['Loan_Amount_Risk'] = quantile_based_risk(df['loan_amount'])
risk_df['Rate_of_Interest_Risk'] = quantile_based_risk(df['rate_of_interest'])
risk_df['Property_Value_Risk'] = quantile_based_risk_rev(df['property_value'])
risk_df['Income_Risk'] = quantile_based_risk_rev(df['income'])
risk_df['LTV_Risk'] = quantile_based_risk(df['LTV'])

# Step 3: Define risk categories for categorical variables (assuming binary or specific categories)

# Loan Limit Risk Levels (cf = low risk, ncf = high risk)
risk_df['Loan_Limit_Risk'] = df['loan_limit'].apply(lambda x: 1 if x == 'cf' else 3)  # 1: Low Risk, 3: High Risk

# Gender Risk Levels
risk_df['Gender_Risk'] = df['Gender'].apply(lambda x: 1 if x in ['Male', 'Female'] else 2)  # 1: Low Risk, 2: Medium Risk

# Loan Type Risk Levels (based on masked types)
risk_df['Loan_Type_Risk'] = df['loan_type'].apply(lambda x: 1 if x == 'type-1' else (2 if x == 'type-2' else 3))  # 1: Low

# Age Group Risk Levels (age groups)
def age_risk(age):
    if age == '<25':
        return 3  # High Risk
    elif age == '25-34':
        return 2  # Medium Risk
    elif age == '35-44':
        return 1  # Low Risk
    elif age == '45-54':
        return 1  # Low Risk
    elif age == '55-64':
        return 2  # Medium Risk
    elif age == '65-74':
        return 3  # High Risk
    else:
        return 3  # High Risk

risk_df['Age_Risk'] = df['age'].apply(age_risk)

# Step 4: Create a total risk score by summing all the individual risk scores
risk_df['Total_Risk_Score'] = (
    risk_df['Credit_Score_Risk'].astype(int) +
    risk_df['Loan_Amount_Risk'].astype(int) +
    risk_df['Rate_of_Interest_Risk'].astype(int) +
    risk_df['Property_Value_Risk'].astype(int) +
    risk_df['Income_Risk'].astype(int) +
    risk_df['LTV_Risk'].astype(int) +
    risk_df['Loan_Limit_Risk'] +
    risk_df['Gender_Risk'] +
    risk_df['Loan_Type_Risk'] +
    risk_df['Age_Risk']
)

# Step 5: Define risk categories based on Total Risk Score
def risk_category(total_score):
    if total_score <= 12:
        return 'Low Risk'
    elif 13 <= total_score <= 18:
        return 'Medium Risk'
    else:
        return 'High Risk'

# Apply the risk category function
risk_df['Risk_Category'] = risk_df['Total_Risk_Score'].apply(risk_category)

# Step 6: View the risk segmentation table
# print(risk_df[['Credit_Score_Risk', 'Loan_Amount_Risk', 'Rate_of_Interest_Risk',
#                'Property_Value_Risk', 'Income_Risk', 'LTV_Risk', 'Loan_Limit_Risk',
#                'Gender_Risk', 'Loan_Type_Risk', 'Age_Risk', 'Total_Risk_Score', 'Risk_Category']])

# Optional: Save the result to a new CSV file
# risk_df.to_csv('risk_segmentation_results.csv', index=False)
```

In [175…

```
risk_df.head()
```

Out[175…

	Status	Credit_Score_Risk	Loan_Amount_Risk	Rate_of_Interest_Risk	Property_Value_Risk	Income_Risk	LTV_Risk	Loan_Limit_Risk	G
0	1	1	1	2	3	3	3	1	
1	1	3	2	2	2	2	2	1	
2	0	1	3	3	1	1	3	1	
3	0	3	3	3	1	1	2	1	
4	0	2	3	3	1	1	3	1	

In [176…
```
risk_df.groupby(['Risk_Category'])['Status'].sum()
```

Out[176…
```
Risk_Category
High Risk      26788
Low Risk           0
Medium Risk     9851
Name: Status, dtype: object
```

In [177…
```
risk_df['Risk_Category'].value_counts()
```

Out[177…
```
Risk_Category
High Risk      98742
Medium Risk    49921
Low Risk           7
Name: count, dtype: int64
```

In [178…
```
risk_df['Total_Risk_Score'].min()
```

Out[178…
```
12
```

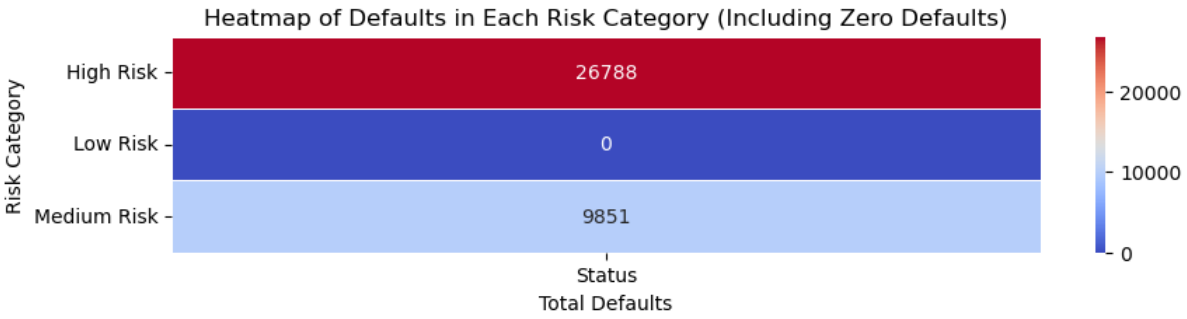
In [179…
```
risk_df['Total_Risk_Score'].max()
```

Out[179…
```
27
```

In [180…
```
risk_df.columns
```

Out[180…
```
Index(['Status', 'Credit_Score_Risk', 'Loan_Amount_Risk',
       'Rate_of_Interest_Risk', 'Property_Value_Risk', 'Income_Risk',
       'LTV_Risk', 'Loan_Limit_Risk', 'Gender_Risk', 'Loan_Type_Risk',
       'Age_Risk', 'Total_Risk_Score', 'Risk_Category'],
      dtype='object')
```

In [362…
```
# Map risk categories to numerical values if not already done
risk_df['Risk_Numeric'] = risk_df['Risk_Category'].map({'Low Risk': 1, 'Medium Risk': 2, 'High Risk': 3})

# Filter out defaulted rows (Status == 1 means default)
default_risks = risk_df[risk_df['Status'] == 1]

# Group by Risk Category and count the number of defaults
default_summary = default_risks.groupby('Risk_Category')['Status'].count().reset_index()

# Create a DataFrame with all risk categories to include zero counts
all_categories = pd.DataFrame({
    'Risk_Category': ['Low Risk', 'Medium Risk', 'High Risk']
})

# Merge the default summary with all categories to ensure all categories are present
default_summary_full = pd.merge(all_categories, default_summary, on='Risk_Category', how='left').fillna(0)

# Convert 'Status' to integer (since NaNs were filled with 0)
default_summary_full['Status'] = default_summary_full['Status'].astype(int)

# Create a pivot table to get the risk categories as index
pivot_default = default_summary_full.pivot_table(index='Risk_Category', values='Status', aggfunc='sum')

# Plot the heatmap
plt.figure(figsize=(10, 2))
sns.heatmap(pivot_default, annot=True, cmap='coolwarm', fmt='g',linewidths=0.5)

# Add labels and title
plt.title('Heatmap of Defaults in Each Risk Category (Including Zero Defaults)')
plt.xlabel('Total Defaults')
plt.ylabel('Risk Category')
plt.show()
```



Observations:

- As per risk segmentation:
  - Number of defaults in **High Risk**: 26,788
  - Number of defaults in **Medium Risk**: 9,851
  - Number of defaults in **Low Risk**: 0

Risk Category Distribution:

- **High Risk**: 98,742
- **Medium Risk**: 49,921
- **Low Risk**: 7

In [ ]:

7. Business Insights

Business Insights

1. Imbalanced Dataset

- **Observation:** The dataset is highly imbalanced, with a large proportion of non-defaulters (112,031 rows) compared to defaulters (36,639 rows).
- **Insight:** Business decisions such as risk modeling, customer targeting, and product offerings should consider the skewed nature of defaults. Over-reliance on current distribution could lead to bias in model performance and customer offerings.

2. Loan Amount and Risk Assessment

- **Observation:** The majority of loan amounts fall between 196,500 and 436,500, with outliers representing very large loans.
- **Insight:** Larger loans do not necessarily correlate with defaults. Financial institutions may focus more on other factors (like interest rate, credit score) rather than loan amount alone when assessing risk.

3. Interest Rate Patterns

- **Observation:** Interest rates generally vary between 3 and 5, with a mean of 4.05.
- **Insight:** Interest rates for loans appear competitive. However, segmentation based on customer credit scores is needed. Lower interest rates are associated with high credit scores, while higher rates increase default risk. Pricing should be tailored accordingly.

4. Credit Score Uniformity

- **Observation:** Credit scores are uniformly distributed between 500 and 900, with a mean of 700.
- **Insight:** Customers with scores below 600 may require more attention, as they are at higher risk of default. Proactive interventions like financial education, targeted loan restructuring, or risk-based pricing can mitigate defaults.

5. Loan to Value (LTV) Ratio

- **Observation:** The mean LTV for defaulters is 0.77, compared to 0.72 for non-defaulters. About 60.4% of data has an LTV below 0.8.
- **Insight:** A higher LTV is a significant risk factor, indicating that customers with lower equity in their properties are more prone to default. A threshold around 0.75 might be used to flag high-risk borrowers.

6. Loan to Income (LTI)

- **Observation:** The mean LTI for both defaulters and non-defaulters is 4.54, with 61.3% of the data below 5.
- **Insight:** LTI ratios should be closely monitored, especially for customers with ratios above 5, as they are more likely to struggle with loan repayments relative to their income.

7. Missing Values

- **Observation:** Key features like upfront charges (26.7%), rate of interest (24.5%), property value (10.2%), and LTV (10.2%) have missing values.
- **Insight:** Missing values in critical financial fields can distort risk assessments. Better data collection practices and imputing strategies are crucial for reducing inaccuracies in segmentation and risk calculation.

8. Risk Segmentation and Default Distribution

- **Observation:**
  - High-Risk category has 26,788 defaults out of 98,742 total customers.
  - Medium-Risk category has 9,851 defaults out of 49,921 total customers.
  - Low-Risk category has 0 defaults but contains only 7 customers.
- **Insight:** The segmentation shows clear stratification of risk, but the low-risk category may require a review, as its size is too small to provide meaningful insight. High-risk and medium-risk segments should be the focus of risk management efforts.

9. Loan Purpose and Defaulter Analysis

- **Observation:** Loan purposes P1, P3, and P4 are prevalent in both defaulters and non-defaulters, with P3 being slightly more frequent among defaulters.
- **Insight:** Certain loan purposes may have higher default probabilities. Offering specialized products with better terms for lower-risk purposes could reduce overall defaults.

10. Customer Demographics and Defaults

- **Observation:** Gender distribution shows male and joint applicants are most common among defaulters, while female borrowers have a lower proportion.
- **Insight:** Marketing strategies and loan products could be designed to encourage female borrowers, who tend to have lower default rates.

11. Regional Distribution and Defaults

- **Observation:** Defaults are distributed similarly across regions, with no significant regional variations.
- **Insight:** Since region doesn't play a significant role in default risk, targeting high-risk customers based on other variables (such as loan amount, interest rate, LTV) would be more effective than focusing on geographic segmentation.

12. Income and Property Value Correlation

- **Observation:** Features like income and property value show moderate correlations with loan amount and LTV.
- **Insight:** While higher income correlates with larger loan amounts, property value plays a stronger role in default risk, especially when LTV is high. Financial products should consider income-property value ratios in lending decisions.

13. Outliers and Skewed Features

- **Observation:** Features like loan amount, income, and LTV are skewed and contain valid outliers.
- **Insight:** Outliers should be retained in the model as they represent real-world financial behavior, such as larger loans or lower-income borrowers. These cases could be used to develop tailored risk-based pricing models.

14. Impact of Imputation on Distributions

- **Observation:** Imputation of missing values, especially in interest rates, has altered the data distribution significantly.
- **Insight:** The changes caused by imputation could introduce bias in analysis. When building predictive models, it would be important to take this into account, possibly using alternative strategies like advanced interpolation or targeted data collection.

15. Moderate Correlation between Features

- **Observation:** Key feature pairs such as loan amount ↔ income and property value ↔ LTV show moderate correlation.
- **Insight:** Cross-selling opportunities exist for customers with high property values and lower LTVs, as they may be eligible for more favorable loan terms or refinancing options.

techniques to enhance loan performance modeling.

In [ ]:

8. Business Recommendation

Business Recommendations

1. Develop Risk-Based Pricing Models

- **Why:** Interest rates and credit scores show significant correlation with default status.
- **Recommendation:** Implement a tiered pricing structure where interest rates and fees are adjusted based on customers' risk profiles (e.g., credit score, LTV, and income). This can help attract low-risk customers with competitive rates while charging higher rates for riskier customers to offset potential defaults.

2. Enhance Customer Segmentation and Targeting

- **Why:** Segmentation based on risk, loan type, loan purpose, and demographic factors (gender, age) can improve profitability.
- **Recommendation:** Use customer segmentation to tailor marketing strategies. For example, target low-risk borrowers with favorable terms and personalized offers to increase loan volumes. Consider creating tailored products for female and joint borrowers, as they have lower default rates, and promote higher-value loans to customers in the 25-54 age group.

3. Introduce Specialized Loan Products

- **Why:** Specific loan purposes (P1, P3, P4) and loan types show trends related to default behavior.
- **Recommendation:** Design specialized loan products for high-demand loan purposes such as P1 (e.g., home purchases) and P4 (business expansion). Additionally, offer secured loans for high-value properties to reduce default risk and increase loan-to-value thresholds safely.

4. Leverage Loan-to-Value (LTV) and Loan-to-Income (LTI) Ratios in Underwriting

- **Why:** High LTV and LTI ratios are correlated with increased default risk.
- **Recommendation:** Introduce stricter underwriting criteria for high-risk borrowers with LTV ratios above 0.75 and LTI ratios above 5. Encourage borrowers to reduce their LTV ratios through larger down payments or offer loan insurance to mitigate potential risks.

5. Refine Data Collection and Handling of Missing Values

- **Why:** Missing values in critical fields (upfront charges, interest rates, property value) affect data integrity and decision-making.
- **Recommendation:** Improve data collection processes to reduce the percentage of missing values, especially in key areas like interest rates and upfront charges. Implement better imputation methods to handle missing data or actively seek missing information before finalizing loan decisions.

6. Target Larger Loans with Caution

- **Why:** While higher loan amounts are not directly correlated with higher default risk, the dataset shows some tendency for defaulters to have larger loans.
- **Recommendation:** Offer tiered loan structures where larger loan amounts are paired with more stringent checks, such as requiring higher credit scores or greater property collateral. This will help reduce default rates while continuing to offer competitive loan amounts.

7. Expand Loan Offerings for High-Property-Value Borrowers

- **Why:** Property values show moderate correlation with loan amounts, LTV, and income.
- **Recommendation:** Introduce premium loan products tailored for customers with high property values, offering lower interest rates and flexible terms. Cross-sell to customers with low LTVs who are less likely to default, offering options like refinancing or home equity loans.

8. Proactively Manage High-Risk Segments

- **Why:** The majority of defaulters are in the high-risk category.

- **Recommendation:** Develop a proactive risk management strategy that includes offering financial counseling, credit monitoring, and early intervention for high-risk borrowers. Consider loan restructuring options for high-risk customers before defaults occur to minimize losses.

9. Increase Focus on Customer Retention Among Non-Defaulters

- **Why:** Non-defaulters represent a stable revenue stream and provide a lower-risk opportunity for loan growth.
- **Recommendation:** Build a customer loyalty program for non-defaulters, offering incentives such as lower interest rates on future loans, early repayment benefits, or discounted fees on new loan applications. Focus on retention strategies for customers with strong credit scores and stable repayment histories.

10. Enhance Risk Assessment Using Machine Learning Models

- **Why:** There is a clear imbalance between defaulters and non-defaulters, with certain features showing predictive power for defaults.
- **Recommendation:** Implement predictive analytics and machine learning models to enhance risk assessment. Use key variables like loan amount, LTV, interest rate, and credit score to predict defaults. By focusing on at-risk customers early in the loan lifecycle, the business can prevent future losses.

11. Reduce Turnaround Time for Low-Risk Loans

- **Why:** Low-risk customers (as indicated by credit score, LTV, and loan amount) can provide consistent business growth with minimal risk.
- **Recommendation:** Streamline loan approval for low-risk customers by automating the loan process, reducing the documentation required, and fast-tracking approvals. Offering quicker processing times for high-value, low-risk clients will increase loan volume and improve customer satisfaction.

12. Improve Financial Literacy Among Borrowers

- **Why:** High LTI, LTV, and interest-to-income ratios suggest that some customers may be overextending their finances.
- **Recommendation:** Offer financial literacy programs to borrowers, focusing on loan management, budgeting, and credit score improvement. Educating customers can reduce default rates and build stronger, more sustainable relationships with borrowers.

13. Offer Flexible Loan Restructuring for At-Risk Borrowers

- **Why:** Borrowers with high LTI or LTV ratios are at greater risk of default, but outliers in these features represent real financial behavior.
- **Recommendation:** Introduce flexible loan restructuring or refinancing options for at-risk borrowers with high LTI or LTV. Offering lower repayment terms, extended loan periods, or deferred payment plans can prevent defaults and protect revenue.

14. Utilize Credit Bureau Partnerships

- **Why:** Different credit types (CIB, CRIF, EXP, EQUI) are used in the dataset and show varying distributions.
- **Recommendation:** Strengthen partnerships with credit bureaus to obtain more granular data on customer credit behavior. This can lead to improved risk modeling and allow for better loan offerings based on real-time credit data.

15. Expand in High-Growth Regions

- **Why:** Regional distribution shows no significant differences in default rates, meaning the business can confidently expand into underrepresented regions without additional risk.
- **Recommendation:** Increase marketing efforts in underrepresented regions, such as North-East and Central areas, where there is potential for growth. Offering region-specific products or incentives (e.g., lower interest rates or faster approval times) can help tap into new customer bases.

In [ ]:

9. Presentation Video Link

https://www.loom.com/share/ee2984a28c0245b9ac314b25458948b4?sid=4f3a8757-d10d-4fae-8e11-aabf5861cb40

In [ ]: