Bank Loan Status Analysis and Data Visualization

Submitted By:

Adabiya

Submitted To,

Shilpa

21-10-2025

CONTENTS

INTRODUCTION

This dataset appears to be a collection of loan application records, typically used for tasks like predictive modeling (e.g., predicting loan approval). The dataset is loaded using the pandas library in Python and contains 381 rows and 13 columns.

AIM:

The primary aim of this loan dataset is predictive modeling to determine if a loan application will be approved (Y) or denied (N), based on the applicant's and loan's characteristics.

OBJECTIVES:

Primary Objective

Prediction: To accurately predict the Loan Status (Approval 'Y' or Denial 'N') for new applicants using a machine learning classification model.

Supporting Objectives

1.Risk Assessment: To quantify the risk associated with a loan application by calculating the probability of default or non-approval based on the input features.

2.Feature Importance: To identify the most significant factors influencing loan approval, such as Credit_History, ApplicantIncome, LoanAmount, and Dependents. This helps in understanding the underlying criteria for lending.

3.Data Quality and Preparation: To clean the dataset by handling missing values (e.g., in Gender, LoanAmount) and converting categorical features (e.g., Education, Property_Area) into a format suitable for modeling.

DATASET DESCRIPTION

Loan_ID	Gender	Married	Dependen	Education	Self_Emplc	ApplicantIr	Coapplica	LoanAmou	Loan_Amo	Credit_His	Property_A	Loan_Status
LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
LP001006	Male	Yes	0	Not Gradu	No	2583	2358	120	360	1	Urban	Y
LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
LP001013	Male	Yes	0	Not Gradu	No	2333	1516	95	360	1	Urban	Y
LP001024	Male	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y
LP001027	Male	Yes	2	Graduate		2500	1840	109	360	1	Urban	Y
LP001029	Male	No	0	Graduate	No	1853	2840	114	360	1	Rural	N
LP001030	Male	Yes	2	Graduate	No	1299	1086	17	120	1	Urban	Y
LP001032	Male	No	0	Graduate	No	4950	0	125	360	1	Urban	Y
LP001034	Male	No	1	Not Gradu	No	3596	0	100	240		Urban	Y
LP001036	Female	No	0	Graduate	No	3510	0	76	360	0	Urban	N
LP001038	Male	Yes	0	Not Gradu	No	4887	0	133	360	1	Rural	N
LP001041	Male	Yes	0	Graduate		2600	3500	115		1	Urban	Y
LP001043	Male	Yes	0	Not Gradu	No	7660	0	104	360	0	Urban	N
LP001047	Male	Yes	0	Not Gradu	No	2600	1911	116	360	0	Semiurban	N
LP001050		Yes	2	Not Gradu	No	3365	1917	112	360	0	Rural	N
LP001068	Male	Yes	0	Graduate	No	2799	2253	122	360	1	Semiurban	Y
LP001073	Male	Yes	2	Not Gradu	No	4226	1040	110	360	1	Urban	Y
LP001086	Male	No	0	Not Gradu	No	1442	0	35	360	1	Urban	N
LP001087	Female	No	2	Graduate		3750	2083	120	360	1	Semiurban	Y
LP001095	Male	No	0	Graduate	No	3167	0	74	360	1	Urban	N
LP001097	Male	No	1	Graduate	Yes	4692	0	106	360	1	Rural	N

This dataset represents loan applicant information typically used for analyzing and predicting loan approval status. It contains 381 rows and 13 columns, with each row representing a unique applicant. The data includes both personal and financial details of individuals who have applied for loans.

Column Name	**Description**	**Data Type**
Loan_ID	Unique identifier for each loan application	object (string)
Gender	Gender of the applicant (Male / Female)	object (string)
Married	Marital status of the applicant (Yes / No)	object (string)
Dependents	Number of dependents (0, 1, 2, 3+)	object (string)
Education	Education level of the applicant (Graduate / Not Graduate)	object (string)

Column Name	Description	Data Type
Self_Employed	Whether the applicant is self-employed (Yes / No)	object (string)
ApplicantIncome	Income of the applicant (in monetary units, e.g., INR)	int64
CoapplicantIncome	Income of the co-applicant (if any)	float64
LoanAmount	Loan amount requested (in thousands, e.g., 128 = ₹128,000)	float64
Loan_Amount_Term	Term of loan repayment (in months)	float64
Credit_History	Credit history status (1.0 = good, 0.0 = bad / no history)	float64
Property_Area	Type of area where property is located (Urban, Rural, Semiurban)	object (string)
Loan_Status	Loan approval status (Y = Approved, N = Not Approved)	object (string)

No of rows:381

No of columns:13

Data set source:Kaggle

The given dataset provides comprehensive information about loan applicants, which can be utilized for analyzing patterns, trends, and factors influencing loan approval decisions. It includes demographic, educational, and financial attributes of applicants, along with their loan details and approval status.

Each record in the dataset represents a unique loan application identified by a Loan_ID. The dataset contains attributes such as Gender, Marital Status, Number of Dependents, Education Level, Employment Type, Applicant Income, Co-applicant Income, Loan Amount, Loan Amount Term, Credit History, Property Area, and Loan Status (approved or not approved).

The dataset can be used for exploratory data analysis (EDA), data visualization, and predictive modeling, particularly in understanding the factors that significantly affect loan approval. It serves as an excellent resource for data analytics and machine learning projects in the domain of financial risk assessment and credit scori

DATA CLEANING

CHECK IF THERE ANY NULL VALUES:

```
data.isnull().any()
```

	0
Loan_ID	False
Gender	True
Married	False
Dependents	True
Education	False
Self_Employed	True
ApplicantIncome	False
CoapplicantIncome	False
LoanAmount	False
Loan_Amount_Term	True
Credit_History	True
Property_Area	False
Loan_Status	False

dtype: bool

FIRST FIVE ROWS:

```
data.head(5)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
1	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
2	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
3	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
4	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y

LAST FIVE ROWS:

```
data.tail(5)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
376	LP002953	Male	Yes	3+	Graduate	No	5703	0.0	128.0	360.0	1.0	Urban	Y
377	LP002974	Male	Yes	0	Graduate	No	3232	1950.0	108.0	360.0	1.0	Rural	Y
378	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
379	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
380	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

DESCRIBE:

```
data.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	381.000000	381.000000	381.000000	370.000000	351.000000
mean	3579.845144	1277.275381	104.986877	340.864865	0.837607
std	1419.813818	2340.818114	28.358464	68.549257	0.369338
min	150.000000	0.000000	9.000000	12.000000	0.000000
25%	2600.000000	0.000000	90.000000	360.000000	1.000000
50%	3333.000000	983.000000	110.000000	360.000000	1.000000
75%	4288.000000	2016.000000	127.000000	360.000000	1.000000
max	9703.000000	33837.000000	150.000000	480.000000	1.000000

INFO:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 381 entries, 0 to 380
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            381 non-null    object
 1   Gender             376 non-null    object
 2   Married            381 non-null    object
 3   Dependents         373 non-null    object
 4   Education          381 non-null    object
 5   Self_Employed      360 non-null    object
 6   ApplicantIncome    381 non-null    int64
 7   CoapplicantIncome  381 non-null    float64
 8   LoanAmount         381 non-null    float64
 9   Loan_Amount_Term   370 non-null    float64
 10  Credit_History     351 non-null    float64
 11  Property_Area      381 non-null    object
 12  Loan_Status        381 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 38.8+ KB
```

COUNT NULL VALUES:

```
data.isnull().sum()
```

	0
Loan_ID	0
Gender	5
Married	0
Dependents	8
Education	0
Self_Employed	21
ApplicantIncome	0
CoapplicantIncome	0
LoanAmount	0
Loan_Amount_Term	11
Credit_History	30
Property_Area	0
Loan_Status	0

dtype: int64

CKECK DUPLICATES:

```
print(data.duplicated())
```

```
0      False
1      False
2      False
3      False
4      False
       ...
376    False
377    False
378    False
379    False
380    False
Length: 381, dtype: bool
```

RENAME:

```
data.rename(columns={
    "CoapplicantIncome":"Cop_Income",
    "ApplicantIncome":"Applicant_Income",
    "LoanAmount":"Loan_Amount",
},inplace=True)
data
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Cop_Income	Loan_Amount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
1	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
2	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
3	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
4	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...	...
376	LP002953	Male	Yes	3+	Graduate	No	5703	0.0	128.0	360.0	1.0	Urban	Y
377	LP002974	Male	Yes	0	Graduate	No	3232	1950.0	108.0	360.0	1.0	Rural	Y
378	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
379	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
380	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

381 rows × 13 columns

FILL WITH UNKNOWN:

```
data["Self_Employed"].fillna("unknown",inplace=True)
data
```

```
/tmp/ipython-input-3011384849.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation
```

data["Self_Employed"].fillna("unknown",inplace=True)

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Cop_Income	Loan_Amount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
1	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
2	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
3	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
4	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
376	Male	Yes	3+	Graduate	No	5703	0.0	128.0	360.0	1.0	Urban	Y
377	Male	Yes	0	Graduate	No	3232	1950.0	108.0	360.0	1.0	Rural	Y
378	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
379	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
380	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

381 rows × 12 columns

```
data["Gender"].fillna("unknown",inplace=True)
data
```

```
/tmp/ipython-input-3748653556.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation
```

data["Gender"].fillna("unknown",inplace=True)

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Cop_Income	Loan_Amount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
1	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
2	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
3	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
4	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
376	Male	Yes	3+	Graduate	No	5703	0.0	128.0	360.0	1.0	Urban	Y
377	Male	Yes	0	Graduate	No	3232	1950.0	108.0	360.0	1.0	Rural	Y
378	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
379	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
380	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

381 rows × 12 columns

```
data["Credit_History"].fillna(data["Credit_History"].mean(),inplace=True)
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Cop_Income	Loan_Amount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
1	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
2	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
3	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
4	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
376	Male	Yes	3+	Graduate	No	5703	0.0	128.0	360.0	1.0	Urban	Y
377	Male	Yes	0	Graduate	No	3232	1950.0	108.0	360.0	1.0	Rural	Y
378	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
379	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
380	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

381 rows × 12 columns

FILL WITH MEAN:

```
data["Dependents"].fillna("mean",inplace=True)#MEAN
data
```

```
/tmp/ipython-input-2650232722.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation
```

data["Dependents"].fillna("mean",inplace=True)#MEAN

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Cop_Income	Loan_Amount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
1	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
2	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
3	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
4	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
376	Male	Yes	3+	Graduate	No	5703	0.0	128.0	360.0	1.0	Urban	Y
377	Male	Yes	0	Graduate	No	3232	1950.0	108.0	360.0	1.0	Rural	Y
378	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
379	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
380	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

381 rows × 12 columns

```
data["Credit_History"].fillna(data["Credit_History"].mean(),inplace=True)
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Cop_Income	Loan_Amount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
1	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
2	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
3	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
4	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
376	Male	Yes	3+	Graduate	No	5703	0.0	128.0	360.0	1.0	Urban	Y
377	Male	Yes	0	Graduate	No	3232	1950.0	108.0	360.0	1.0	Rural	Y
378	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
379	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
380	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

381 rows × 12 columns

FILL WITH ZERO:

```
data["Loan_Amount_Term"].fillna(0,inplace=True)
data
```

```
/tmp/ipython-input-2100843975.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the c
```

data["Loan_Amount_Term"].fillna(0,inplace=True)

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Cop_Income	Loan_Amount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
1	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
2	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
3	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
4	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
376	Male	Yes	3+	Graduate	No	5703	0.0	128.0	360.0	1.0	Urban	Y
377	Male	Yes	0	Graduate	No	3232	1950.0	108.0	360.0	1.0	Rural	Y
378	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
379	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
380	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

381 rows × 12 columns

```
data["Credit_History"].fillna(0,inplace=True)
data
```

```
/tmp/ipython-input-3830827615.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the opera
```

```
data["Credit_History"].fillna(0,inplace=True)
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Cop_Income	Loan_Amount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
1	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
2	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
3	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
4	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
376	Male	Yes	3+	Graduate	No	5703	0.0	128.0	360.0	1.0	Urban	Y
377	Male	Yes	0	Graduate	No	3232	1950.0	108.0	360.0	1.0	Rural	Y
378	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
379	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
380	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

381 rows × 12 columns

CHECK IF THERE IS ANY NULL VALUES:

```
data.isnull().any()
```

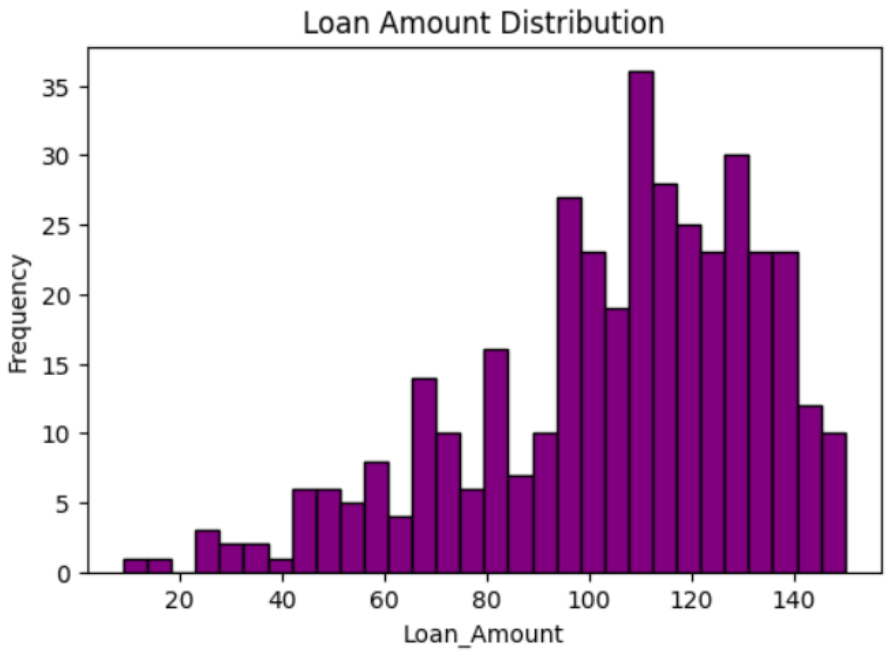
	0
Gender	False
Married	False
Dependents	False
Education	False
Self_Employed	False
Applicant_Income	False
Cop_Income	False
Loan_Amount	False
Loan_Amount_Term	False
Credit_History	False
Property_Area	False
Loan_Status	False

dtype: bool

EXPLORATORY DATA ANALYSIS(EDA):

What is the most frequent loan amount shown in this distribution?

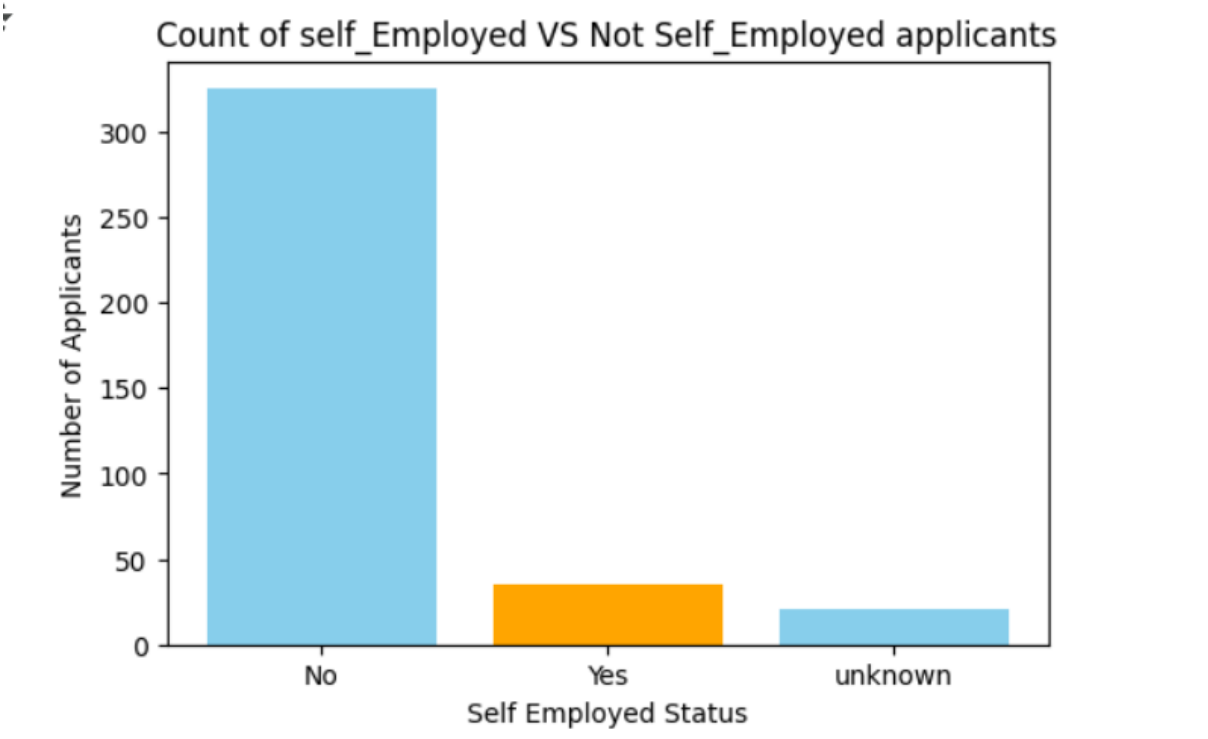
```
#1.loan amount distribution(histogram)
plt.figure(figsize=(6,4))
data["Loan_Amount"].plot(kind="hist",bins=30,color="Purple",edgecolor="black")
plt.title("Loan Amount Distribution")
plt.xlabel("Loan_Amount")
plt.ylabel("Frequency")
plt.show()
```



Most coomon loan anounts:between 100-140 *Less common loan amounts:below 40 or above 140 *The histogram shows that loan amounts are not evenly distributed--they are concentrated around a certain middle range.

Which self-employed status represents the largest group of applicants according to the chart?

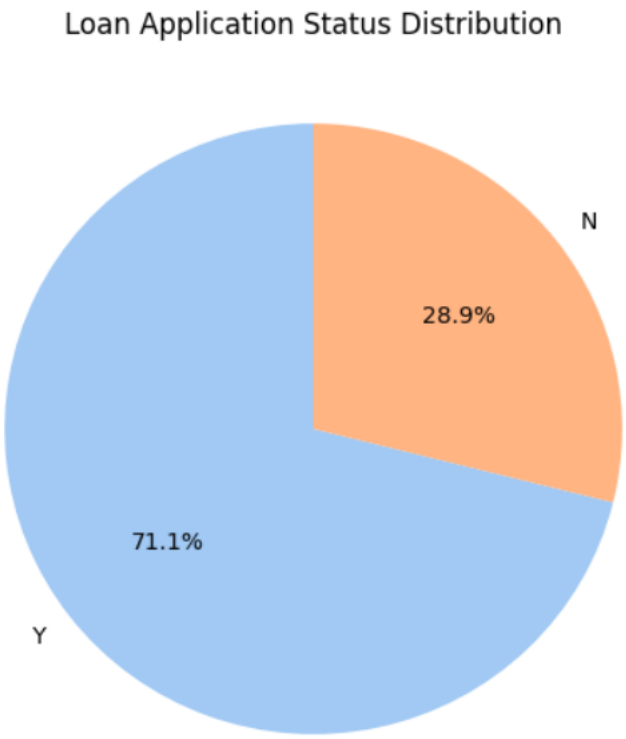
```
import pandas as pd
import matplotlib.pyplot as plt
counts=data["Self_Employed"].value_counts()
plt.figure(figsize=(6,4))
plt.bar(counts.index,counts.values,color=["skyblue","orange"])
plt.xlabel("Self Employed Status")
plt.ylabel("Number of Applicants")
plt.title("Count of self_Employed VS Not Self_Employed applicants")
plt.show()
```



- The majority of loan applicants in the dataset are salaried(not self-employed). *self-employed individuals represent a small proportion of total applicants.
- this insight can help in understanding applicant demographics.

What is the distribution of loan application statuses?

```
#3.Loan Application Status Distribution(pie chart)
import matplotlib.pyplot as plt
import seaborn as sns
#count loan status
Loan_Status_Counts=data["Loan_Status"].value_counts()
colors=sns.color_palette("pastel")[0:2]
#plot pie chart
plt.figure(figsize=(6,6,))
plt.pie(
    Loan_Status_Counts,
    labels=Loan_Status_Counts.index,
    autopct="%1.1f%%",
    colors=colors,
    startangle=90
    )
plt.title("Loan Application Status Distribution")
plt.show()
```

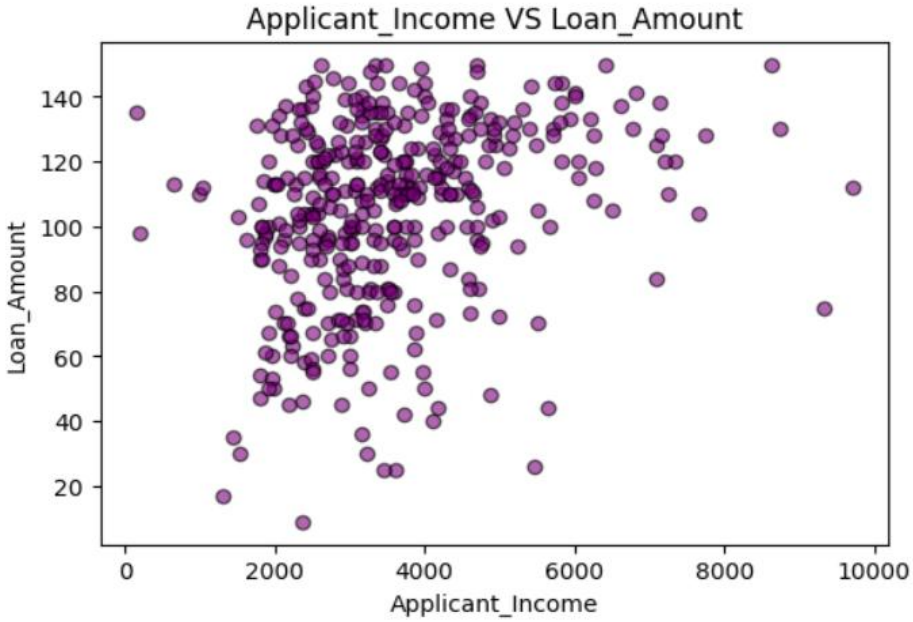


Pie chart shows the Loan Application Status Distribution- 1)Approved(Y) 2)Not Approved(N)

- Around 71% of loan applications were Approved(Y). *About 29% were Rejected(N).
- This means most applicants succesfully got loan Approval,while a smaller portion was denied.

What is the relationship between Applicant Income and Loan Amount?

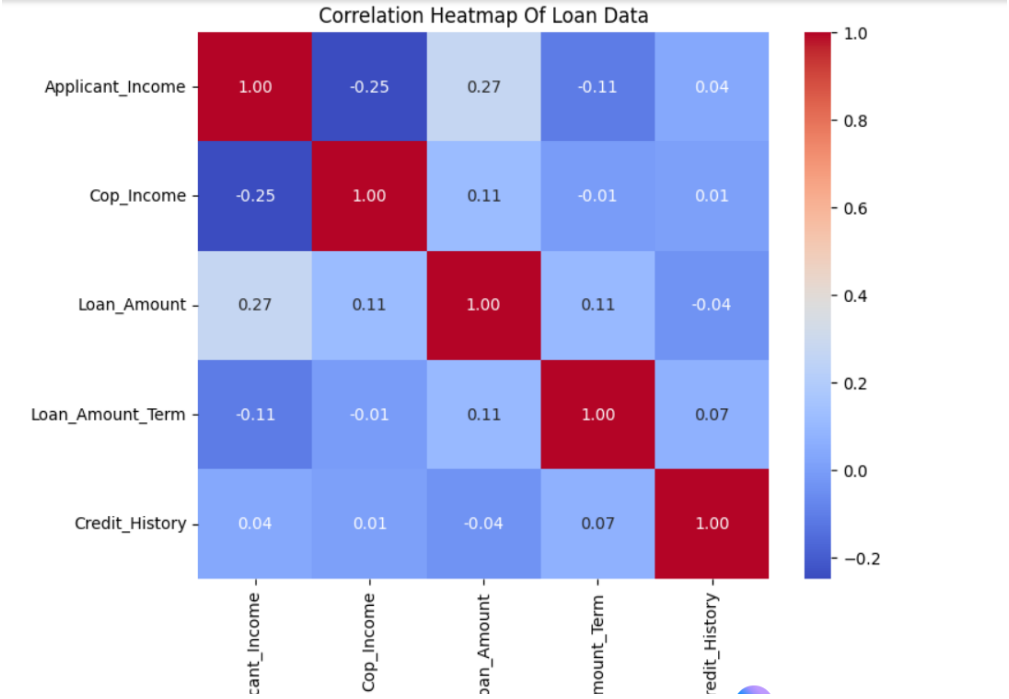
```
#4.Applicant_Income VS Loan Amount(scatter plot)
import pandas as pd
plt.figure(figsize=(6,4))
plt.scatter(data["Applicant_Income"],
            data["Loan_Amount"],color="Purple",
            alpha=0.6,edgecolor="black")
plt.title("Applicant_Income VS Loan_Amount")
plt.xlabel("Applicant_Income")
plt.ylabel("Loan_Amount")
plt.show()
```



- Each points represents an applicants income and their corresponding loan amount.
- The points are somewhat scattered without a strong pattern,but there is a slight upward trend-meaning that as applicant income increases,the loan amount also tends to increase.
- However,the spread is wide,suggesting that income alone does not strongly determine the loan amount.
- Most applicants have moderate incomes and moderate loan amounts,as seen from the dense cluster in the middle.

What is the strongest positive correlation between any two variables shown in the matrix?

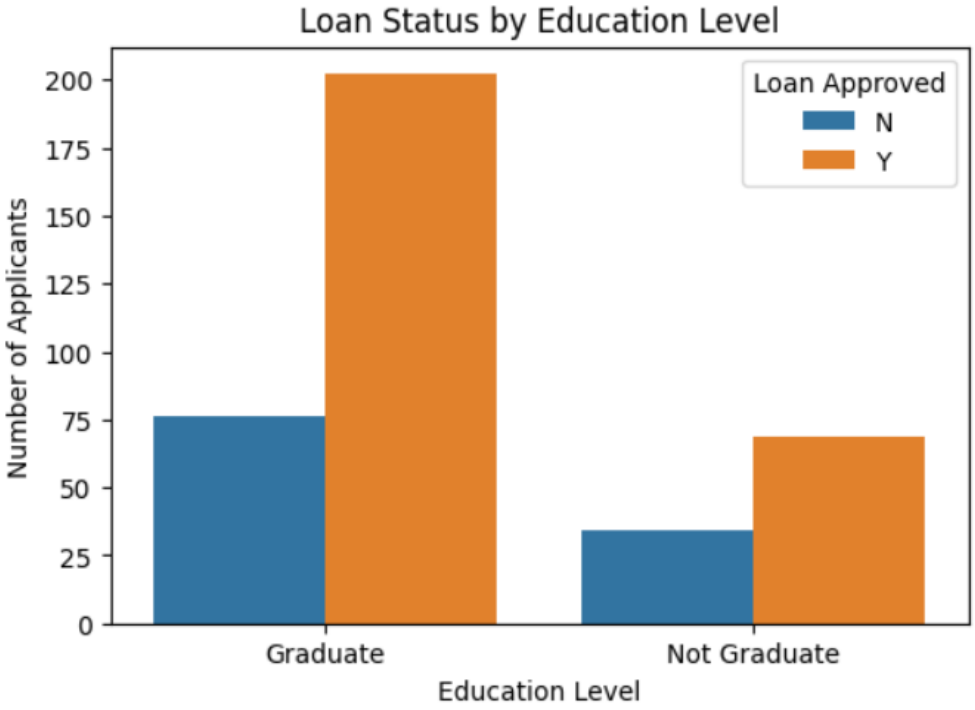
```
#5.Correlation Heatmap og Loan Data(heatmap)
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
#correlation matrix
corr=data.corr(numeric_only=True)
#only numeric columns
#plot heatmap
plt.figure(figsize=(8,6))
sns.heatmap(corr,annot=True,cmap="coolwarm",fmt=".2f")
plt.title("Correlation Heatmap Of Loan Data")
plt.show()
```



- Applicant_income and Loan_amount have a positive correlation (0.27)_meaning higher income tends to slightly increase loan amount.
- Applicant_income and coapplicant_income show a negative correlation(-0.25)_as ones income increases,the others may slightly decrease(maybe only one main earner per family).
- Loan_Amount_Term and Loan_Amount have a weak positive correlation (0.11)_longer loan terms are slightly linked to higher loan amounts.
- Credit_History has very low correlation with other variables_its almost independent.

Which educational level has the highest number if loan approvals(Y)?

```
#6.Loan Status by Education level(Clustered bar chart)
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(6,4))
sns.countplot(x="Education",hue="Loan_Status",data=data)
plt.title("Loan Status by Education Level")
plt.xlabel("Education Level")
plt.ylabel("Number of Applicants")
plt.legend(title="Loan Approved")
plt.show()
```



- Graduates:A greater number of graduates were approved for a loan(the orange bar for "Y")compared to those who were not approved(the blue for "N").
- Not Graduates:A greater number of non-graduates were nnot approved for a loan(the blue bar for "N")compare to those who were approved(the orange bar for "Y").
- overall:the graph suggests that graduates have a higher rate of loan approval compared to non-graduates.

Outlayer removal:

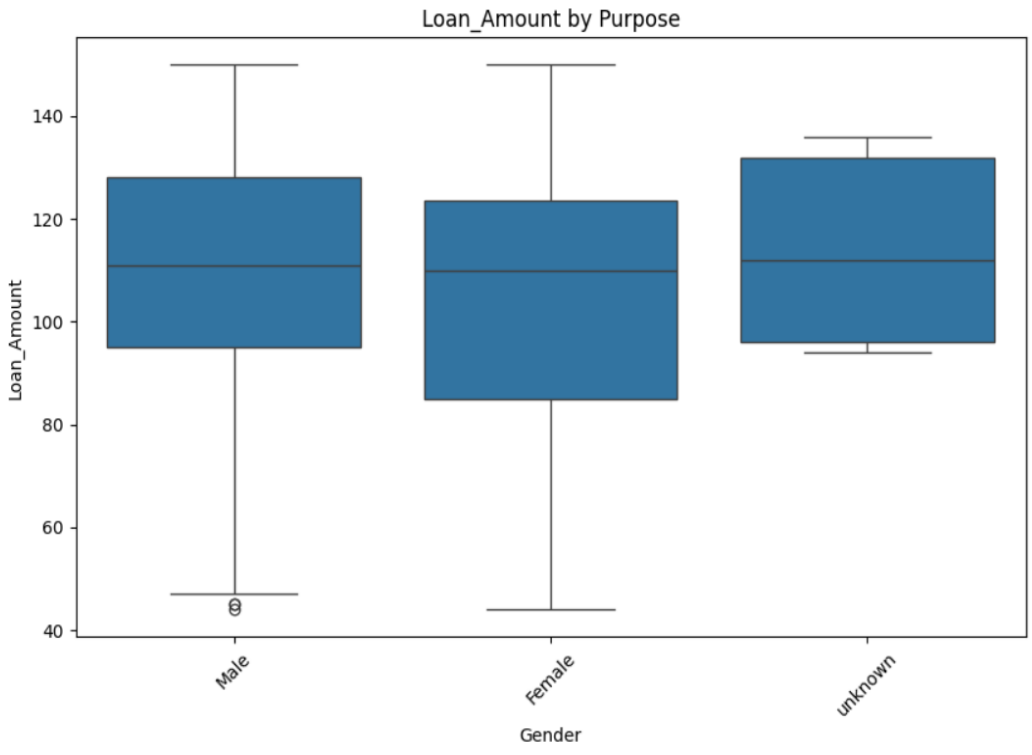
```
for col in ["Loan_Amount"]:
  Q1=data[col].quantile(0.25)
  Q3=data[col].quantile(0.75)
  IQR=Q3-Q1
  lower=Q1-1.5*IQR
  upper=Q3+1.5*IQR
  data=data[(data[col]>=lower)&(data[col]<=upper)]
data
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Cop_Income	Loan_Amount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
1	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
2	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
3	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
4	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
375	Male	No	mean	Graduate	No	2987	0.0	88.0	360.0	0.0	Semiurban	N
376	Male	Yes	3+	Graduate	No	5703	0.0	128.0	360.0	1.0	Urban	Y
377	Male	Yes	0	Graduate	No	3232	1950.0	108.0	360.0	1.0	Rural	Y
378	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
380	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

370 rows × 12 columns

Which group has the largest range in Loan Amount?

```
#7.Loan Amount by purpose(boxplot)
import seaborn as sns
plt.figure(figsize=(10,6))
sns.boxplot(data=data, x="Gender", y='Loan_Amount')
plt.xticks(rotation=45)
plt.title("Loan_Amount by Purpose")
plt.show()
```



Three gender categories:Male,Female,Unknown.A boxplot visualizes the distribution of a dataset by showing the median,quartiles,and potential outliers. Male:the median loan amount for males is
approximately 100.theinterquartilerange(IQR),representedbythebox,showsthat5080 and 120.thewhiskersindicatethefullrangeofthedata,whichappearstobefromabout40 to 140.Female:themedianloanamountforfemalesisslightlylowerthanformales,atapproximately90.the IQR for females is between approximately 70and110.the range of the data is similar to males,extending from about 40to140. Unknown:the distribution for the "unknown" gender category is very similar to the female category.the median loan amount is approximately 90,withanIQRfromabout70 to 110.therangeisalsosimilar,fromabout40 to $140.

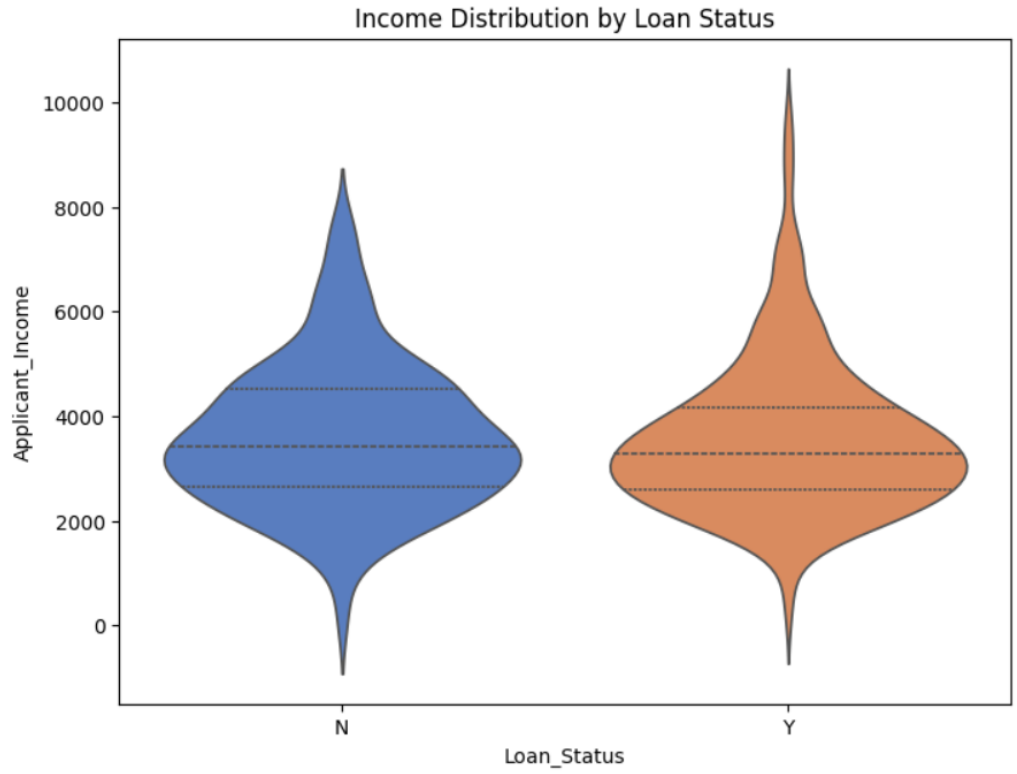
What is the median Applicant_Income for those with an Approved loan status (Y)compared to those with a Rejected status(N)?

```
#8.Income Distribution by Loan Status(violin plot)
import seaborn as sns
plt.figure(figsize=(8,6))
sns.violinplot(data=data, x="Loan_Status", y="Applicant_Income", inner='quartile', palette='muted')
plt.title("Income Distribution by Loan Status")
plt.show()
```

```
/tmp/ipython-input-1069699358.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.violinplot(data=data, x="Loan_Status", y="Applicant_Income", inner='quartile', palette='muted')
```

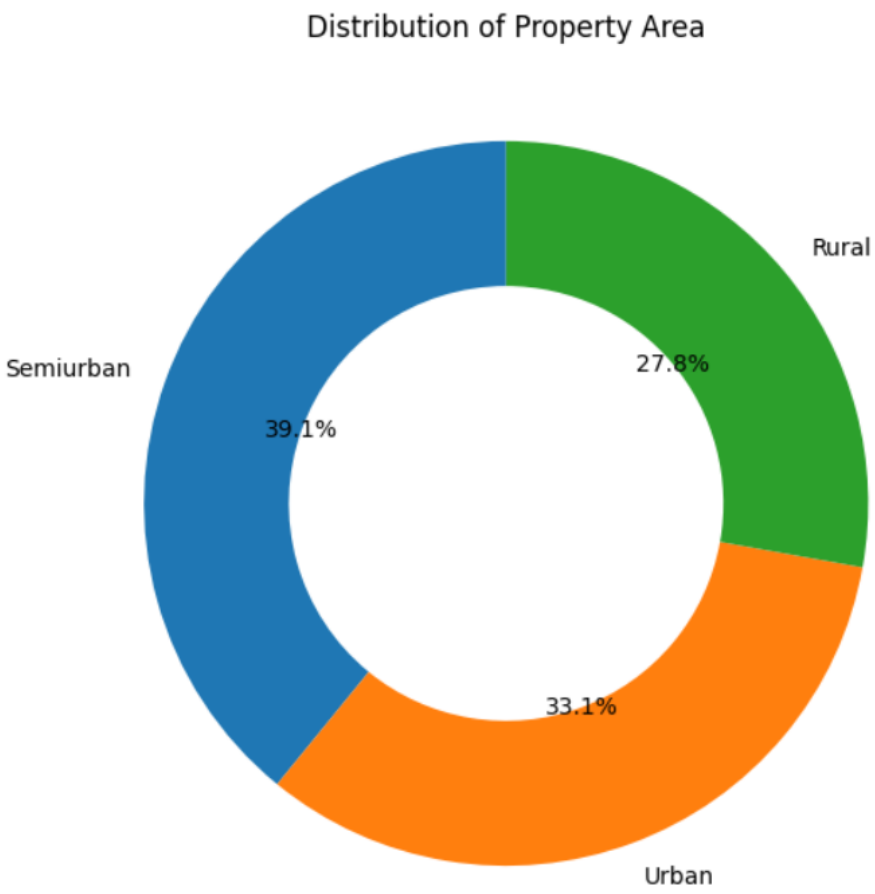


The violin plot shows the distribution applicant income based on loan status(N for no loan,Y for yes loan)

- Loan Status"N"(No Loan):the blue violin plot represents applicants who did not receive a loan.the shape of the plot indicates that the majority of these applicants have an income clustered between approximately 2000 and 6000.
- Loan Status"Y"(Yes Loan):the red violin plot represents applicants who received a loan.the wider section of this plot shows that the majority of these applicants have a higher income,centered around 6000,and the distribution extends to a much higher income level compared to the "N" group.

What is the percentage distribution of the three property area types?

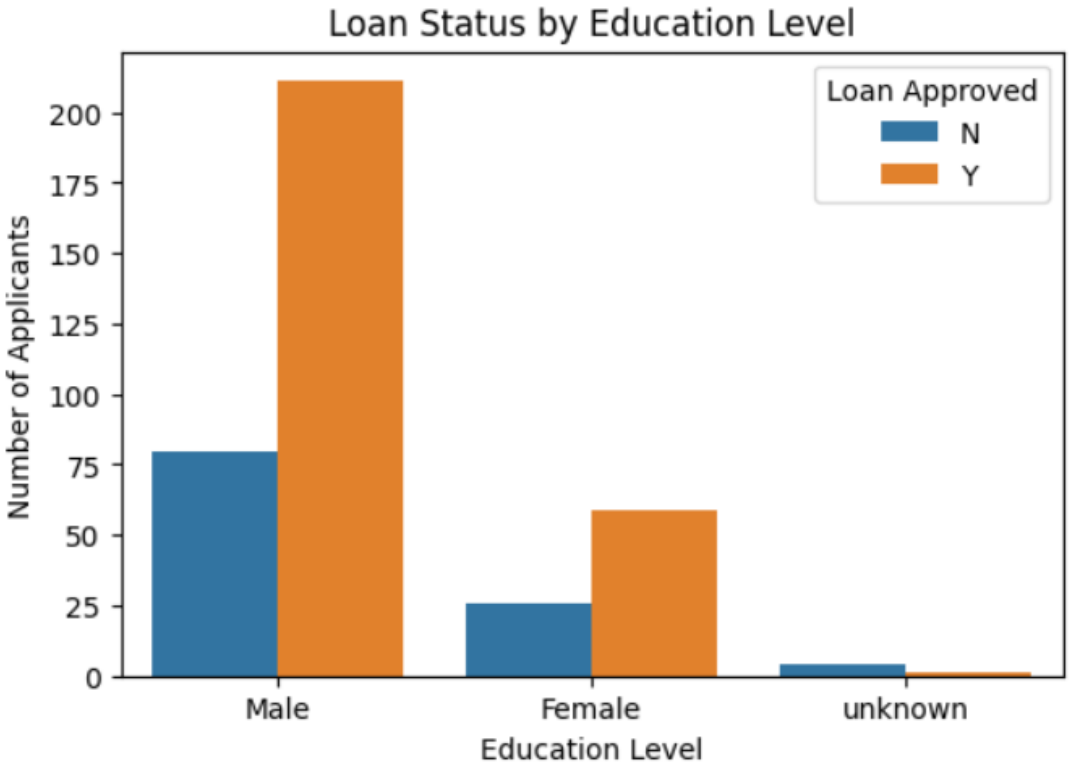
```
#9.Distribution of property area(donet chart)
column_name = "Property_Area"
counts = data[column_name].value_counts()
plt.figure(figsize=(7,7))
plt.pie(counts, labels=counts.index, autopct='%1.1f%%', startangle=90, wedgeprops={'width':0.4})
plt.title(f"Distribution of {column_name.replace('_', ' ').title()}")
plt.show()
```



"Distribution of Property Area",shows how property is divided among 3 types of areas. 1)Semiurban: areas accounts for the largest portion,representing 39.1% of the total propery area. 2)Urban:areas make up the second-largest portion at 33.1%. 3)Rural:areas have the smallest share,accounting for 27.8% of the total property area.

Which gender group has the highest number of approval loans(Y)?

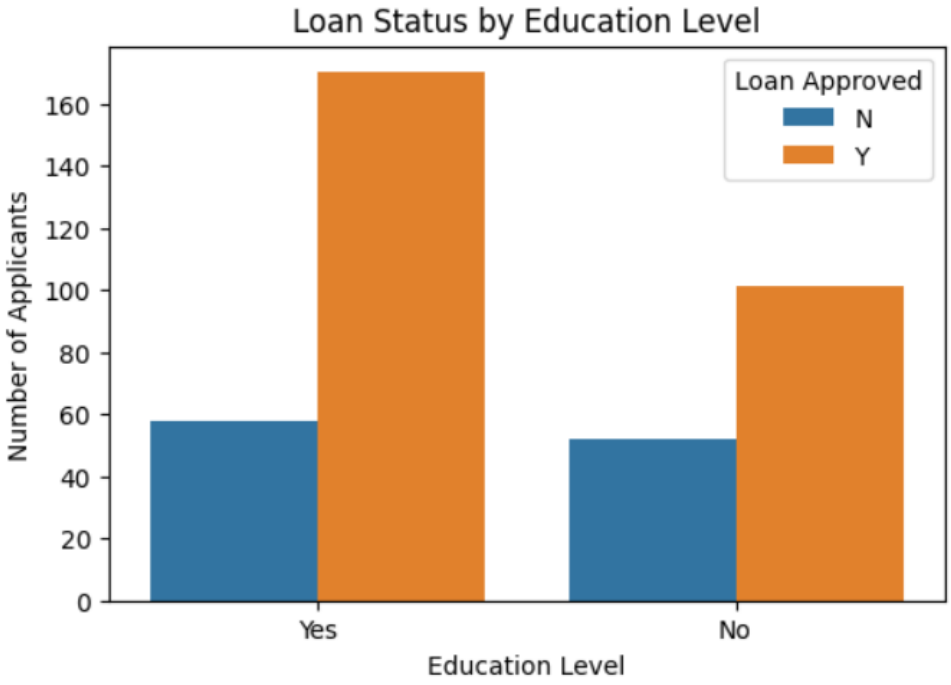
```
#10.Loan status by education level(clustered bar chart)
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(6,4))
sns.countplot(x="Gender",hue="Loan_Status",data=data)
plt.title("Loan Status by Education Level")
plt.xlabel("Education Level")
plt.ylabel("Number of Applicants")
plt.legend(title="Loan Approved")
plt.show()
```



- Among males,the number of applicants who get their loan approved(Y)is higher than those who were not approved(N).
- Among Females,the same trend appears -more applicants have their loans approved than rejected.
- overall,male applicants dominate in both approval and rejection counts compared to females.

Which marital status group(Yes or No) has a higher rejection rate?

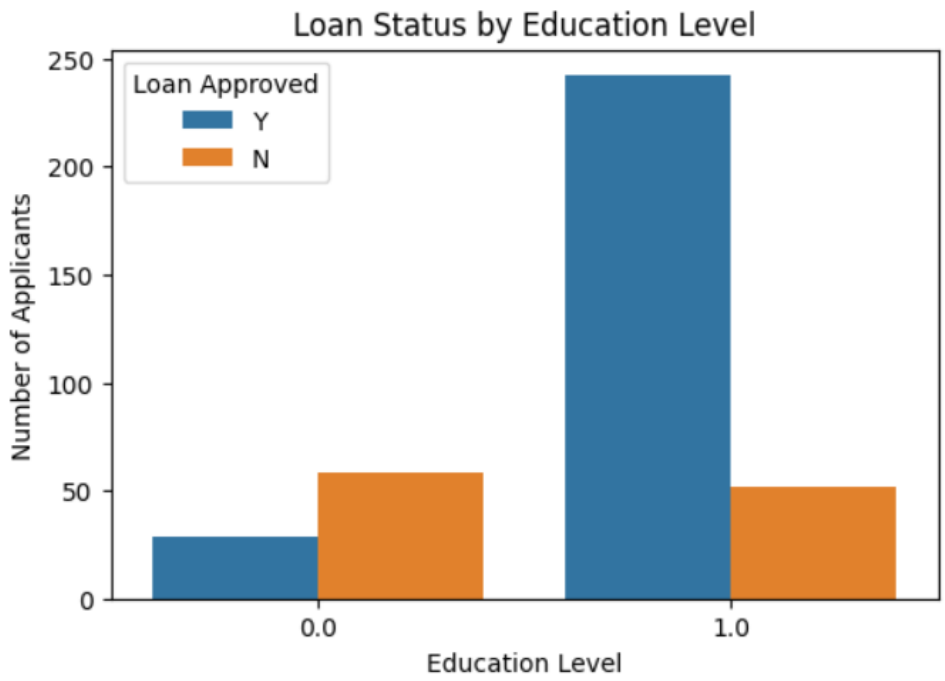
```
#11.Loan status by education level(clustered bar chart)
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(6,4))
sns.countplot(x="Married",hue="Loan_Status",data=data)
plt.title("Loan Status by Education Level")
plt.xlabel("Education Level")
plt.ylabel("Number of Applicants")
plt.legend(title="Loan Approved")
plt.show()
```



- Applicants with "Yes"(Graduated)education level have a higher number of approval loans(Y).
- Applicants with "No"(Not Graduated)education level have fewer approved loans.
- graduates are more likely to get their loans approved compared to non-graduates.

Which credit history status (0.0 or 1.0) has the highest number of loan approval(Y)?

```
#12.Loan status by education level(clustered bar chart)
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(6,4))
sns.countplot(x="Credit_History",hue="Loan_Status",data=data)
plt.title("Loan Status by Education Level")
plt.xlabel("Education Level")
plt.ylabel("Number of Applicants")
plt.legend(title="Loan Approved")
plt.show()
```



The x-axis represents Education Level (0 = Not Graduate, 1 = Graduate).

The y-axis shows the Number of Applicants.

The bars are divided by Loan Approval Status — Y (Yes) and N (No).

conclusion

The analysis reveals that the loan approval process is significantly driven by an applicant's Credit History and appears to be less influenced by income, gender, or marital status.

1. Dominant Factor: Credit History

Credit History is the most critical predictor of loan approval. Applicants with a satisfactory credit history (\text{1.0}) are overwhelmingly approved, demonstrating an approval rate of approximately \mathbf{83\%}.

Conversely, applicants with an unsatisfactory or missing credit history (\text{0.0}) are much more likely to be rejected, with a rejection rate of around \mathbf{66\%}.

2. Approval Distribution (Overall)

The overall approval rate for the entire dataset is quite high, with approximately \mathbf{71.1\%} of all loan applications being approved (Y) and \mathbf{28.9\%} being rejected (N).

3. Influence of Other Demographic and Financial Factors

While Credit History is dominant, other factors show the following patterns:

Education Level: Graduate applicants have a higher total number of both approved and rejected loans compared to "Not Graduate" applicants, but the proportion of approvals within each group is high, suggesting a slight advantage for Graduates.

Marital Status: Both Married (Yes) and Not Married (No) applicants show high approval rates. The Married group has a larger absolute number of approvals.

Gender: Male applicants account for the largest number of both approvals and rejections, simply due to their higher representation in the dataset. Female applicants also show a high approval rate relative to their total application count.

Income and Loan Amount: The scatter plot showed no clear linear correlation between Applicant Income and Loan Amount. The violin plot indicated that the median income is similar for both Approved (Y) and Rejected (N) applicants, suggesting that high income alone is not a guarantee of approval; the decision is highly dependent on Credit History.