Bank Loan Case Study

Project Description

This case study aims to give you an idea of applying EDA in a real business scenario. The loan providing companies find it hard to give loans to the people due to their insufficient or non-existent credit history. Because of that, some consumers use it as their advantage by becoming a defaulter. Suppose you work for a consumer finance company which specializes in lending various types of loans to urban customers.

The task is to use Exploratory Data Analysis (EDA) to analyze patterns in the data and ensure that capable applicants are not rejected. The main aim of this project is to identify patterns that indicate if a customer will have difficulty paying their installments. This information can be used to make decisions such as denying the loan, reducing the amount of loan, or lending at a higher interest rate to risky applicants. The company wants to understand the key factors behind loan default so it can make better decisions about loan approval.

When a customer applies for a loan, your company faces two risks:

1. If the applicant can repay the loan but is not approved, the company loses business.

2. If the applicant cannot repay the loan and is approved, the company faces a financial loss.

When a customer applies for a loan, there are four possible outcomes:

1. Approved: The company has approved the loan application.

2. Cancelled: The customer cancelled the application during the approval process.

3. Refused: The company rejected the loan.

4. Unused Offer: The loan was approved but the customer did not use it.

The goal in this project is to use EDA to understand how customer attributes and loan attributes influence the likelihood of default.

Approach and Tech Used

VS Code studio: For this project I have used VS code studio to run my queries and charts. The Vs code studio extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results.

Python Programming (Version 3.8): For the data analysis, python is the best and the easiest to use programming language.

Microsoft Word 2021: It is used to make a report (PDF) to be presented to the leadership team.

Approach:

Data Importing

Importing the libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from IPython.display import display
import warnings
warnings.filterwarnings('ignore')
import os
```

Reading the dataset

```
# Reading the input csv file
data = pd.read_csv(pwd + '/Data - application.csv')
pd.options.display.max_columns = None
data.shape
# Copying the data to another dataframe
df = data.copy()
df
```

Output:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	NAME_T
0	100002	1	Cash loans	M	N	Y	0	202500.0	406597.5	24700.5	351000.0	Unac
1	100003	0	Cash loans	F	N	N	0	270000.0	1293502.5	35698.5	1129500.0	
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	135000.0	6750.0	135000.0	Unac
3	100006	0	Cash loans	F	N	Y	0	135000.0	312682.5	29686.5	297000.0	Unac
4	100007	0	Cash loans	M	N	Y	0	121500.0	513000.0	21865.5	513000.0	Unac
...	...	...	...	...	...	...	...	...	...	...	...	
49994	157871	0	Cash loans	F	N	N	0	180000.0	1206000.0	45936.0	1206000.0	
49995	157872	0	Cash loans	M	N	N	0	126000.0	1125000.0	47794.5	1125000.0	Unac
49996	157873	0	Cash loans	M	N	N	1	112500.0	900000.0	26316.0	900000.0	Unac
49997	157874	0	Cash loans	F	N	Y	0	270000.0	820638.0	34897.5	733500.0	
49998	157875	0	Cash loans	F	N	Y	0	117000.0	254700.0	14751.0	225000.0	Unac

49999 rows × 122 columns

Data analytics tasks

A. Identify Missing Data and Deal with it Appropriately

- There are total of 122 columns in previous_data.csv file.
- Converting the negative days column into positive days.
- Created new columns AGE_GROUP, YEAR_EMPLOYED, CREDIT_GROUP
- There are 49 columns which with more than 40% missing data. So, dropping those columns.

```
# Missing Percentage
null_data = df.isna().mean().sort_values(ascending=False)*100
# Removing columns with 40% or more missing percentage
df = df.loc[:,null_data <= 40]
# Dropping unnecessary columns
df
df["AMT_CREDIT"] = df['AMT_CREDIT']/100000
df["AMT_CREDIT_GROUP"]= pd.cut(df['AMT_CREDIT'],bins=[0,1,2,3,4,5,6,7,8,9,10,100],labels=["0-
100k","100-200K","200-300K","300-400k","400-500k","500-600k","600-700k","700-800k",
                              "800-900k","900-1M","1M+"])
df["AGE_GROUP"] = pd.cut(df['AGE'],[0,30,40,50,60,99], labels=["<30","30-40","40-50","50-60","60+"])
age = df['AGE_GROUP'].value_counts()
df["YEAR_EMPLOYED_GROUP"] =
pd.cut(df['YEAR_EMPLOYED'],bins=[1,5,10,20,30,40,50,60,1000],labels=["0-5","5-10","10-20","20-
30","30-40","40-50","50-60","60+"])

null_data
```

Output

```
COMMONAREA_MEDI             69.921398
COMMONAREA_AVG              69.921398
COMMONAREA_MODE             69.921398
NONLIVINGAPARTMENTS_MODE    69.429389
NONLIVINGAPARTMENTS_AVG     69.429389
                               ...
NAME_HOUSING_TYPE            0.000000
NAME_FAMILY_STATUS           0.000000
NAME_EDUCATION_TYPE          0.000000
NAME_INCOME_TYPE             0.000000
SK_ID_CURR                   0.000000
Length: 122, dtype: float64
```

- On further analysis, we found that some of the columns has no correlation with the "TARGET" column and therefore dropping those unnecessary columns.

```
drop_columns =
['FLAG_MOBIL','FLAG_EMP_PHONE','FLAG_WORK_PHONE','FLAG_CONT_MOBILE','FLAG_PHON
E','FLAG_EMAIL','CNT_FAM_MEMBERS','REGION_RATING_CLIENT','REGION_RATING_CLIENT_
W_CITY','WEEKDAY_APPR_PROCESS_START','HOUR_APPR_PROCESS_START','REG_REGION_N
OT_LIVE_REGION','REG_REGION_NOT_WORK_REGION','LIVE_REGION_NOT_WORK_REGION','R
```

```
EG_CITY_NOT_LIVE_CITY','REG_CITY_NOT_WORK_CITY','LIVE_CITY_NOT_WORK_CITY','EXT_SOURCE_2','EXT_SOURCE_3','OBS_30_CNT_SOCIAL_CIRCLE','DEF_30_CNT_SOCIAL_CIRCLE','OBS_60_CNT_SOCIAL_CIRCLE','DEF_60_CNT_SOCIAL_CIRCLE','DAYS_LAST_PHONE_CHANGE','FLAG_DOCUMENT_2','FLAG_DOCUMENT_3','FLAG_DOCUMENT_4','FLAG_DOCUMENT_5','FLAG_DOCUMENT_6','FLAG_DOCUMENT_7','FLAG_DOCUMENT_8','FLAG_DOCUMENT_9','FLAG_DOCUMENT_10','FLAG_DOCUMENT_11','FLAG_DOCUMENT_12','FLAG_DOCUMENT_13','FLAG_DOCUMENT_14','FLAG_DOCUMENT_15','FLAG_DOCUMENT_16','FLAG_DOCUMENT_17','FLAG_DOCUMENT_18','FLAG_DOCUMENT_19','FLAG_DOCUMENT_20','FLAG_DOCUMENT_21','AMT_REQ_CREDIT_BUREAU_HOUR','AMT_REQ_CREDIT_BUREAU_DAY','AMT_REQ_CREDIT_BUREAU_WEEK','AMT_REQ_CREDIT_BUREAU_MON','AMT_REQ_CREDIT_BUREAU_QRT','AMT_REQ_CREDIT_BUREAU_YEAR']
df = df.drop(drop_columns, axis=1)
df
```

Output

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	NAME_T
0	100002	1	Cash loans	M	N	Y	0	202500.0	406597.5	24700.5	351000.0	Unac
1	100003	0	Cash loans	F	N	N	0	270000.0	1293502.5	35698.5	1129500.0	
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	135000.0	6750.0	135000.0	Unac
3	100006	0	Cash loans	F	N	Y	0	135000.0	312682.5	29686.5	297000.0	Unac
4	100007	0	Cash loans	M	N	Y	0	121500.0	513000.0	21865.5	513000.0	Unac
...	...	...	...	...	...	...	...	...	...	...	...	
49994	157871	0	Cash loans	F	N	N	0	180000.0	1206000.0	45936.0	1206000.0	
49995	157872	0	Cash loans	M	N	N	0	126000.0	1125000.0	47794.5	1125000.0	Unac
49996	157873	0	Cash loans	M	N	N	1	112500.0	900000.0	26316.0	900000.0	Unac
49997	157874	0	Cash loans	F	N	Y	0	270000.0	820638.0	34897.5	733500.0	
49998	157875	0	Cash loans	F	N	Y	0	117000.0	254700.0	14751.0	225000.0	Unac

49999 rows × 23 columns

Now the dataset contains 23 columns and 49999 rows

- Checking of null values

```
df.isnull().sum().sort_values(ascending=False)
```

Output

```
OCCUPATION_TYPE               15654
NAME_TYPE_SUITE                 192
AMT_GOODS_PRICE                  38
AMT_ANNUITY                       1
NAME_EDUCATION_TYPE               0
DAYS_ID_PUBLISH                   0
DAYS_REGISTRATION                 0
DAYS_EMPLOYED                     0
DAYS_BIRTH                        0
REGION_POPULATION_RELATIVE        0
NAME_HOUSING_TYPE                 0
NAME_FAMILY_STATUS                0
SK_ID_CURR                        0
NAME_INCOME_TYPE                  0
TARGET                            0
AMT_CREDIT                        0
AMT_INCOME_TOTAL                  0
CNT_CHILDREN                      0
FLAG_OWN_REALTY                   0
FLAG_OWN_CAR                      0
CODE_GENDER                       0
NAME_CONTRACT_TYPE                0
ORGANIZATION_TYPE                 0
dtype: int64
```

- Here occupation type has 15,654 missing values, and as it is an Categorical columns hence we cannot replace it with some other value. Therefore, dropping null values in OCCUPATION_TYPE column.

```
df.dropna(inplace=True)
df
```

Output

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	NAME_T
0	100002	1	Cash loans	M	N	Y	0	202500.0	406597.5	24700.5	351000.0	Unac
1	100003	0	Cash loans	F	N	N	0	270000.0	1293502.5	35698.5	1129500.0	
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	135000.0	6750.0	135000.0	Unac
3	100006	0	Cash loans	F	N	Y	0	135000.0	312682.5	29686.5	297000.0	Unac
4	100007	0	Cash loans	M	N	Y	0	121500.0	513000.0	21865.5	513000.0	Unac
...	...	...	...	...	...	...	...	...	...	...	...	
49987	157862	0	Cash loans	M	Y	Y	1	90000.0	900000.0	26446.5	900000.0	Unac
49989	157865	0	Revolving loans	F	N	Y	0	135000.0	270000.0	13500.0	270000.0	Unac
49990	157867	0	Revolving loans	F	N	Y	0	112500.0	180000.0	9000.0	180000.0	Unac
49995	157872	0	Cash loans	M	N	N	0	126000.0	1125000.0	47794.5	1125000.0	Unac
49997	157874	0	Cash loans	F	N	Y	0	270000.0	820638.0	34897.5	733500.0	

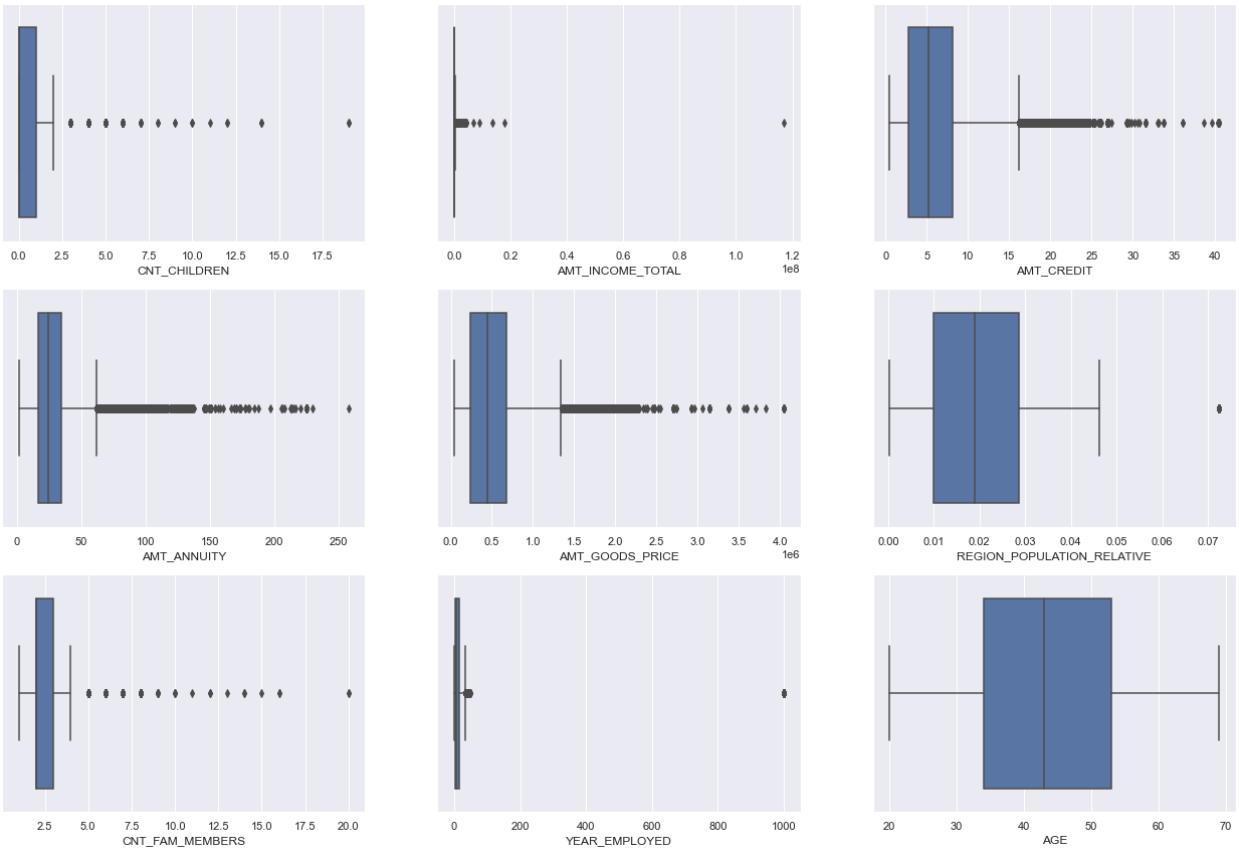
34210 rows × 23 columns

- The null values in GOODS_PRICE and AMT_ANNUITY has been filled with mean of the columns respectively.

```
df['AMT_ANNUITY'].fillna(df['AMT_ANNUITY'].mean,inplace=True)
df['AMT_GOODS_PRICE'].fillna(df['AMT_GOODS_PRICE'].mean,inplace=True)
```

B. Identify Outliers in the Dataset:

- AMT_ANNUITY, AMT_CREDIT, AMT_GOODS_PRICE, CNT_CHILDREN have some number of outliers.



From the above diagrams we can conclude that:

- Amt_Income_Total has a very high outlier, but it does not mean it could be an error. Some people have very high income
- Amt_Credit, Amt_Annuity, Amt_Goodprice, CNT_Fam_members, Cnt_Children have outliers which are normal considering some people have many kids or many family members,or high credit/annuity value
- Year employed has an outlier error where it shows some people have been employed for 1000 years
- Age Column has no outliers which means the data is proper

C. Analyze Data Imbalance:

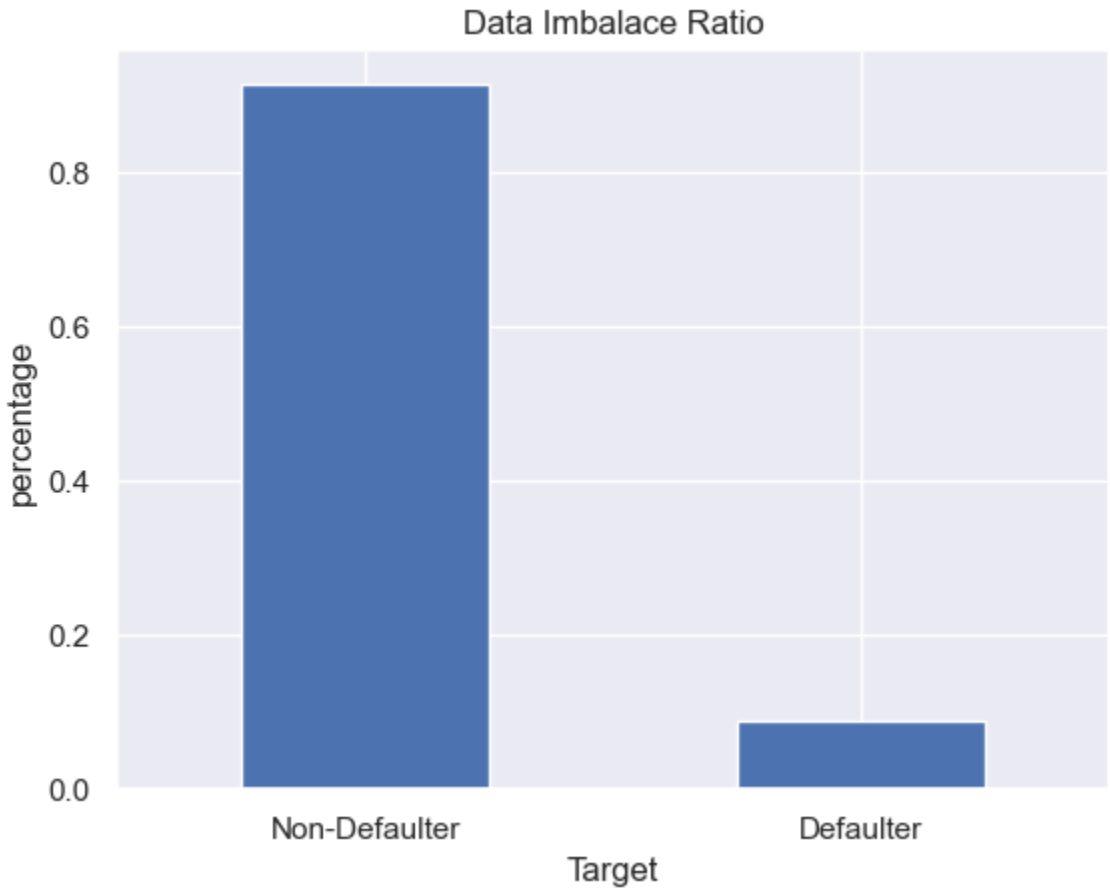
- This data is highly imbalanced as number of defaulters is very less in total population. Data Imbalance Ratio with respect to Repayment and Default: 10.42.

```
# Checking Data Imbalance/Ratio on Target Variable
df['TARGET'].value_counts(normalize=True)
df['TARGET'].value_counts(normalize=True).plot.bar()
plt.xticks((0,1),["Non-Defaulter","Defaulter"],rotation=360)
plt.ylabel("percentage")
plt.xlabel("Target")
plt.title("Data Imbalace Ratio")
plt.show()
```

Output:

```
TARGET
0    0.912482
1    0.087518
Name: proportion, dtype: float64
```

```
TARGET
0    31216
1     2994
Name: count, dtype: int64
```

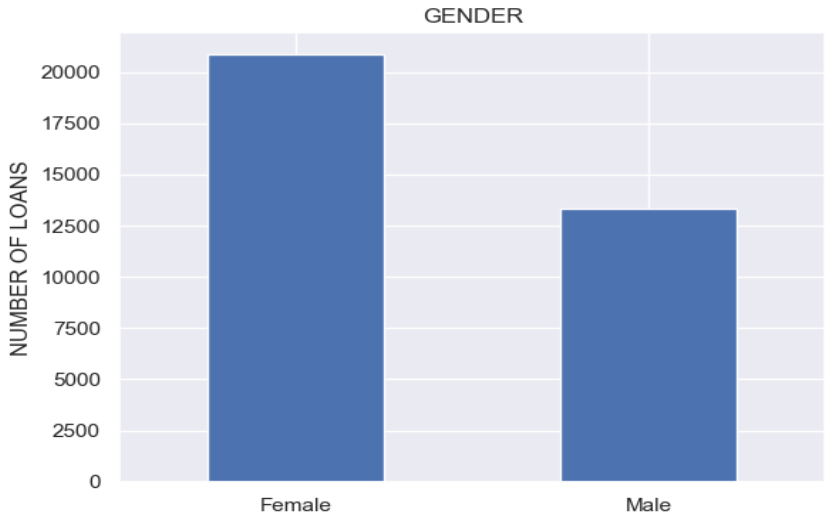


The data imbalance ratio is: 31216/2994 = 10.42

D. Perform Univariate, Segmented Univariate and Bivariate Analysis:

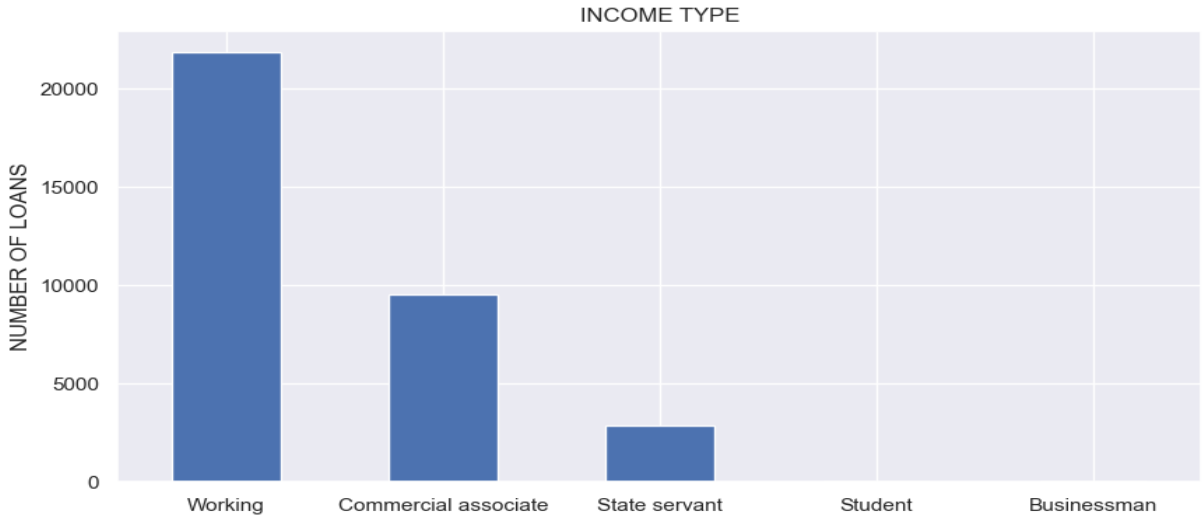
1.Gender

- 61% of loan appliers were in fact females and rest 38% being males



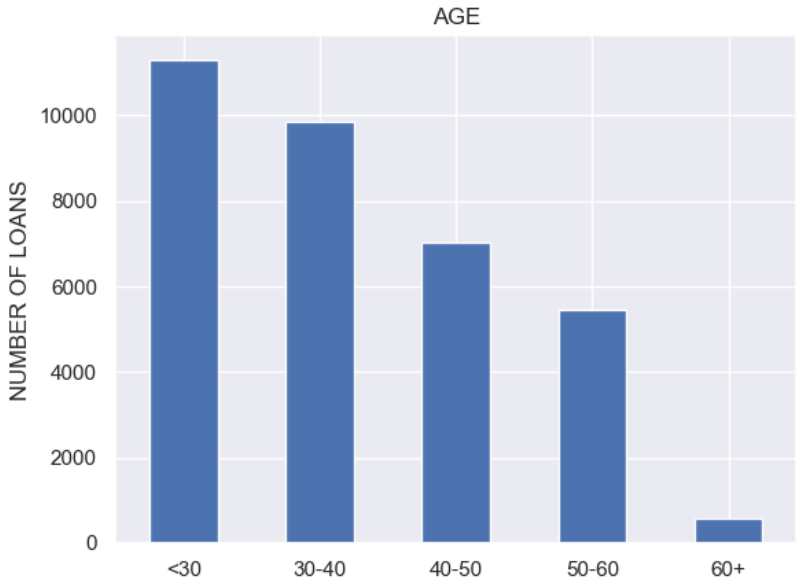
2.Income type

- Most of applicants for loans have income type as Working, followed by Commercial associate and State servant



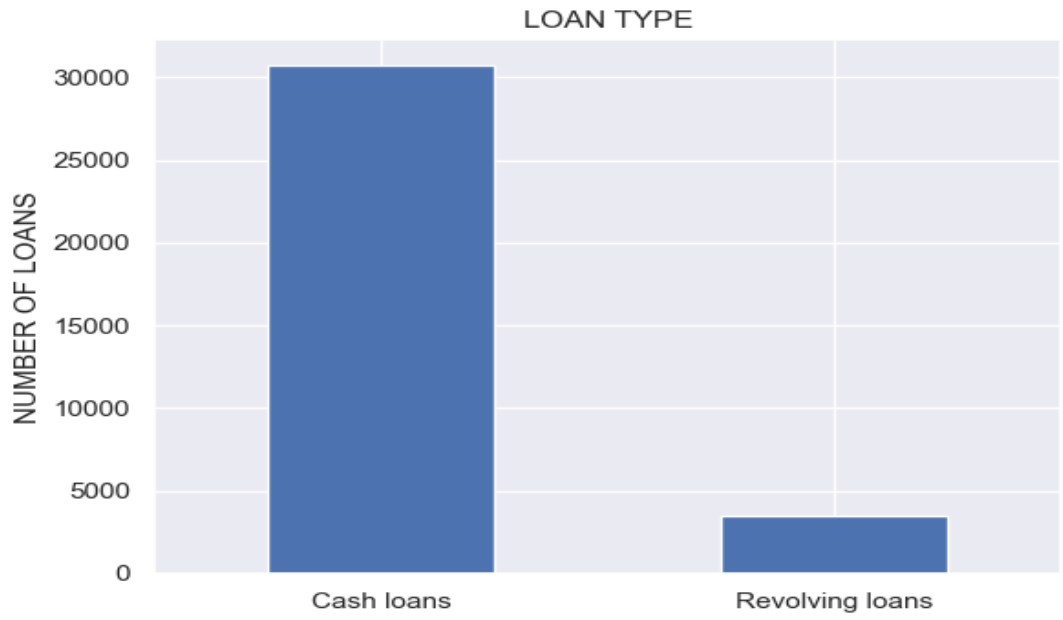
3.Age

- The middle-aged people ranging from 30-40 and 40-50 had the highest amount of loan applications
- Less 60+ aged group applicants were lower



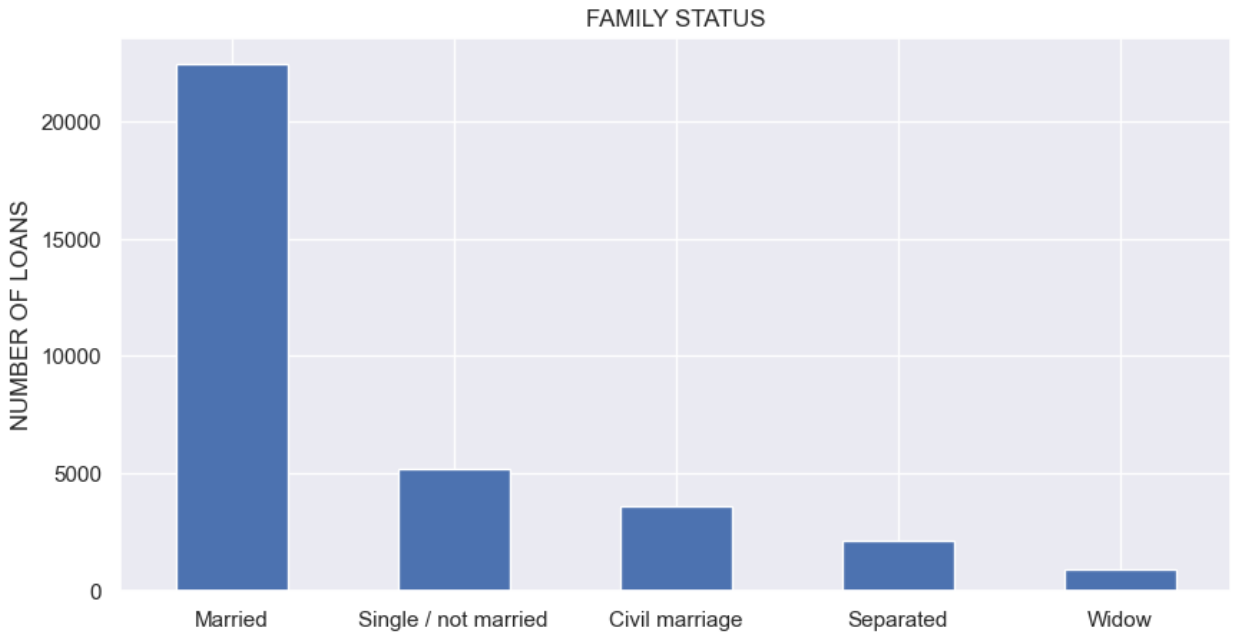
4.Loan type

- The Contract_type graph shows that CASH loans were taken by 90.47% of the people! and only 9.52% people took Revolving Loans



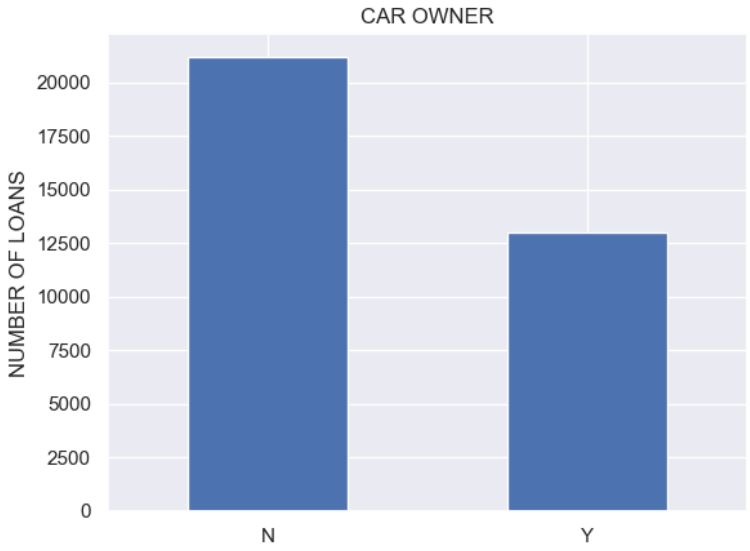
6.Family status

- Married people were the higher percentage of loan applicants with 63.87%. Followed by Single people having 14.77%
- Civil marriage, Separated, Widow were among the lower percentage category who applied for loans



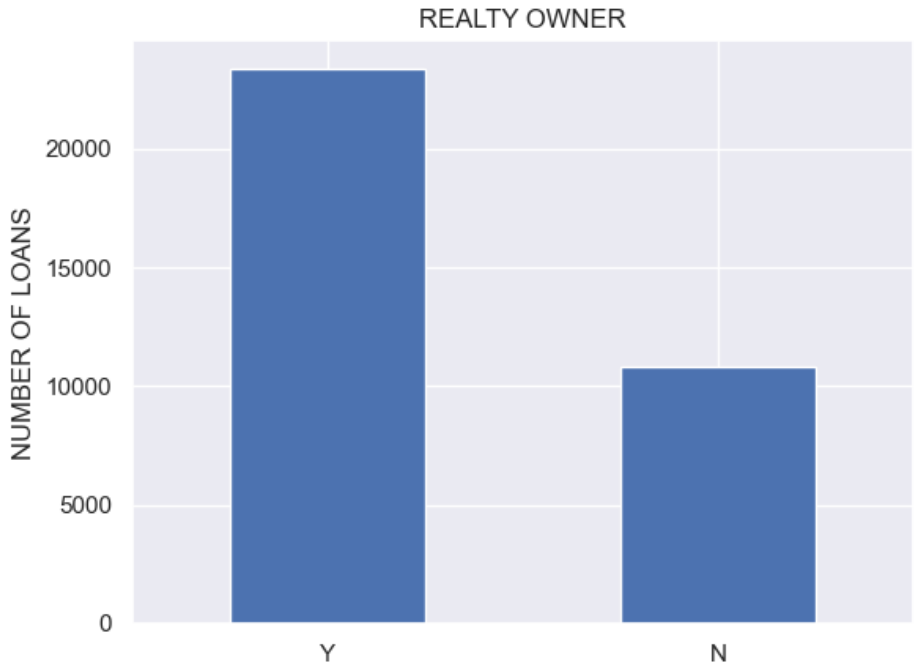
7.Car owners

- 65.98% of the people own a car, they were the majority who applied for loans
- 34.01% of the people who applied do not own a car



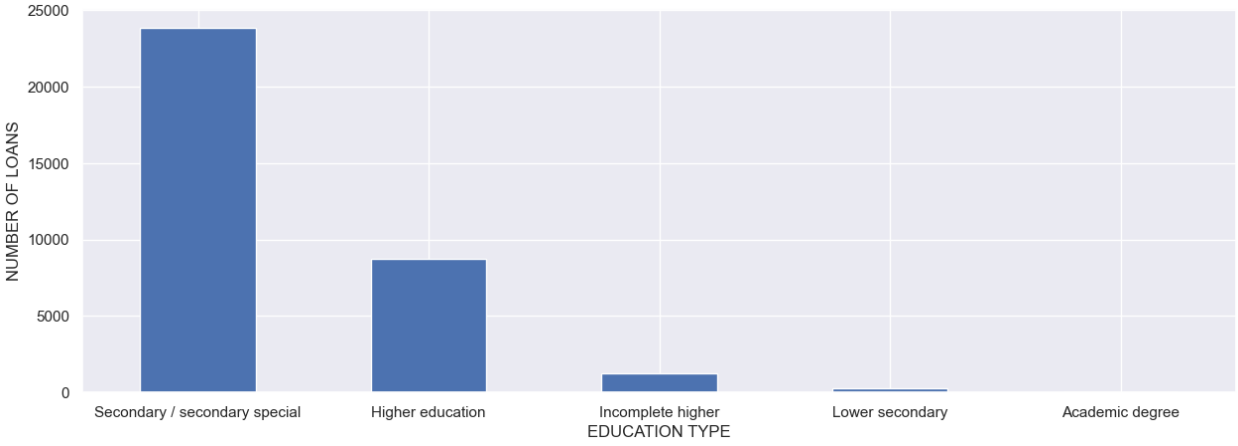
8.Realty owners

- 69.36% people don't own a Realty (House or flat)
- From the housing type column, we found that 88% people lived in House/ apartments but they did not own it.



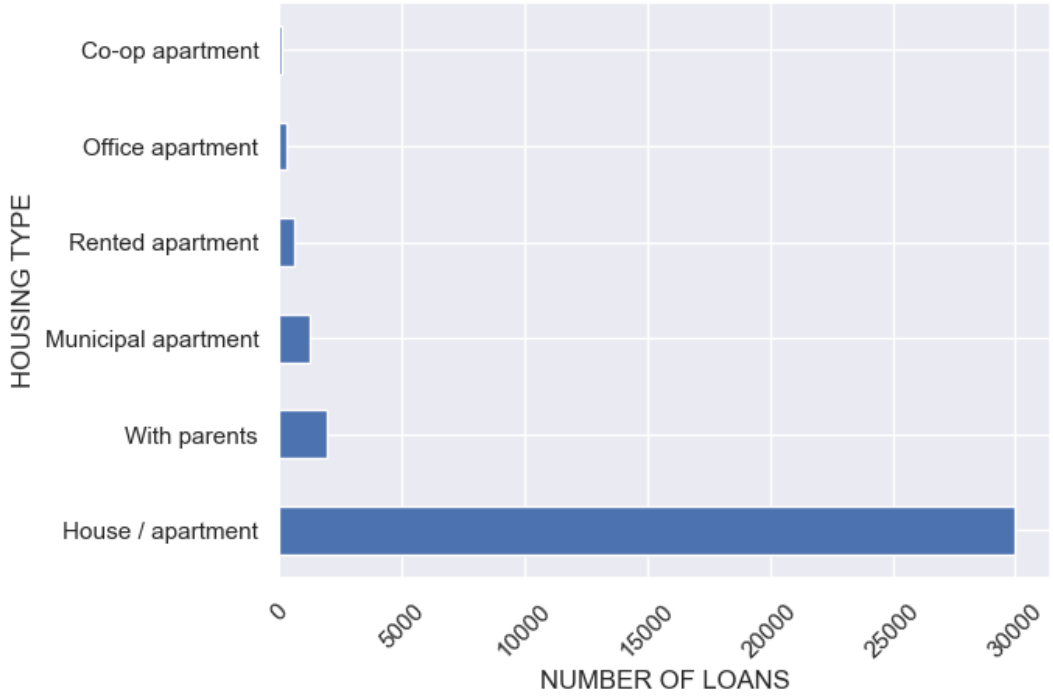
9.Education type

- Majority of the applicants had a Secondary education (71.01%) followed by Higher education (24.34%). Academic degree had the lowest percentage



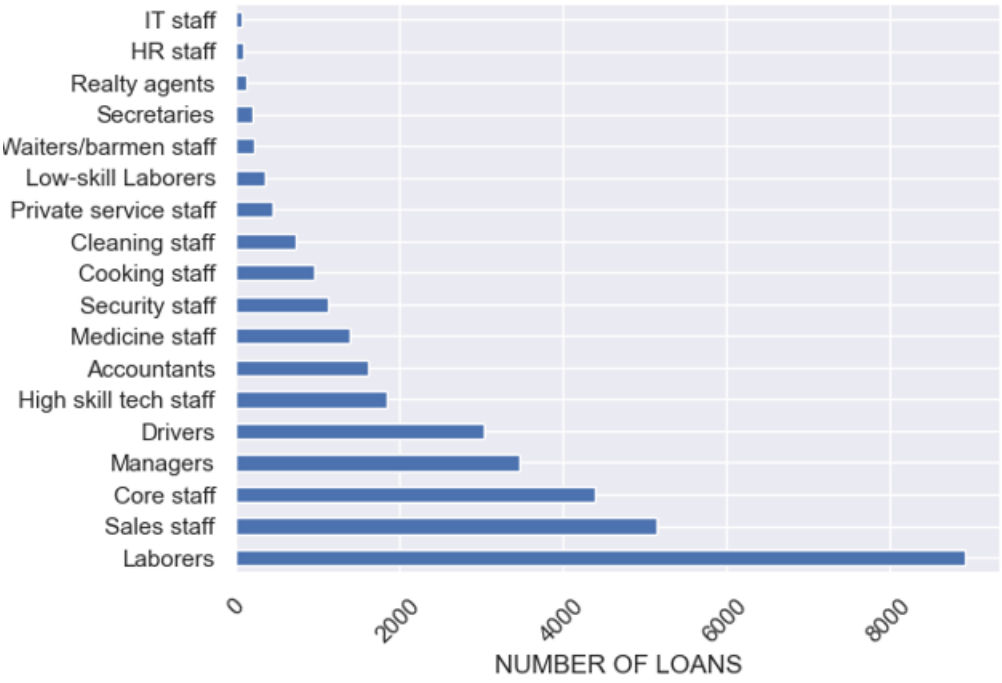
10.Housing type

- 88.73% of the applicants were living in a House / Apartment
- 4.82% were living with their parents followed by 3.63% living in Municipal Apartment
- Rented apartment accounted for 1.587265%, Office apartment accounted for 0.851029% and Co-op apartment accounted for 0.364866



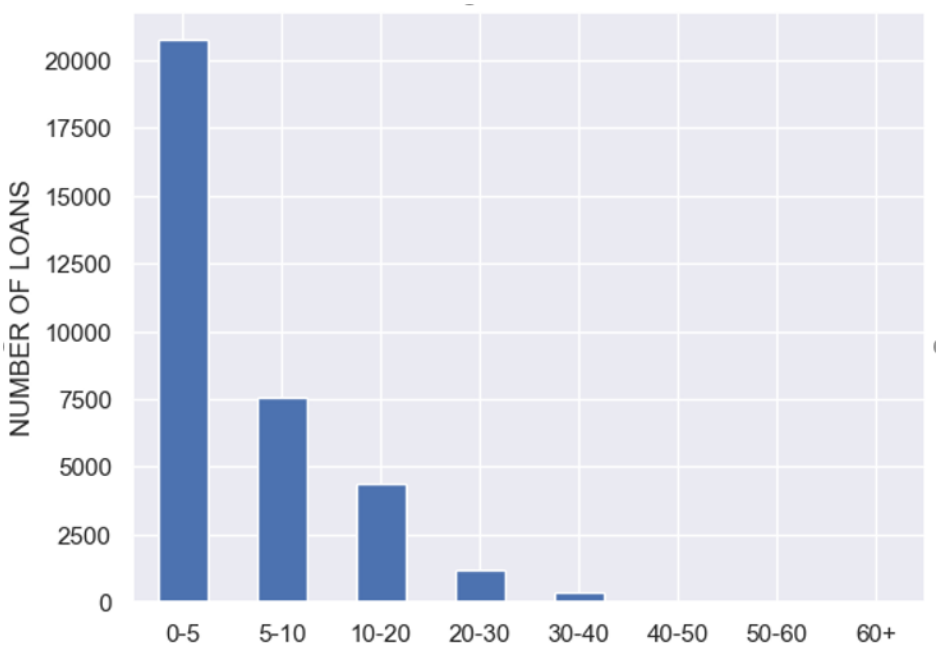
11.Occupation type

- As you can see from the above percentage value that Laborers were the highest percentage of people who applied for a loan having the value of 26.13%, Followed by Sales staff (15.20%), Core staff (13.05%), Managers (10.12%), Drivers (8.81%).



12.Years employed

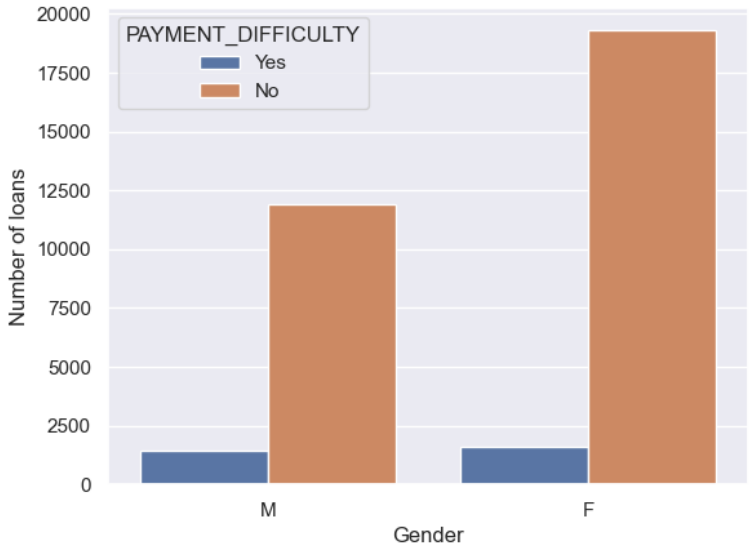
- 0-5 years of experience people topped the chart with 49.60%
- with none being from 50-60 years, and some being from 40-50 years



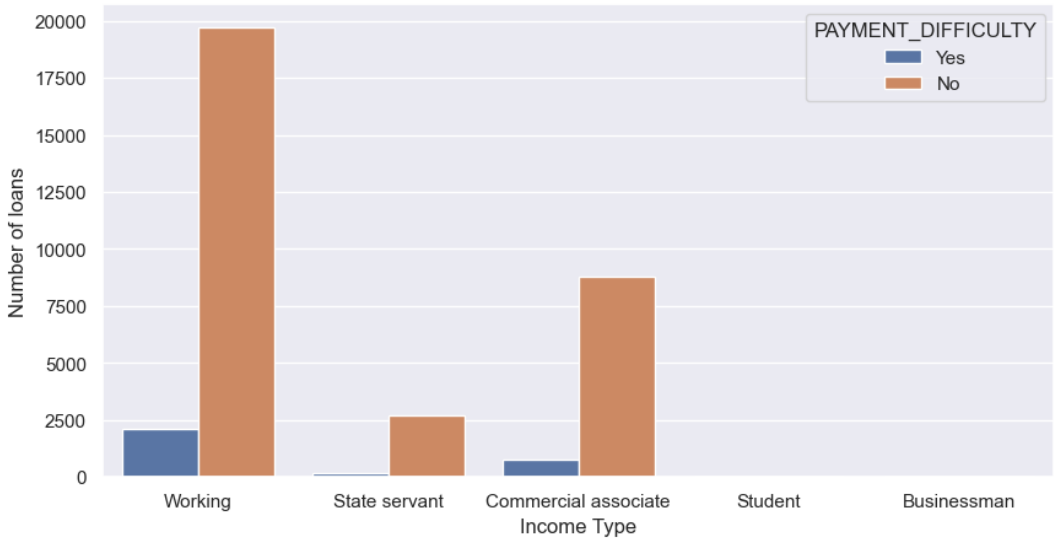
BIVARIATE ANALYSIS

- Creating a new column “PAYMENT DIFFICULTY” and inputting value “Yes” and “No”

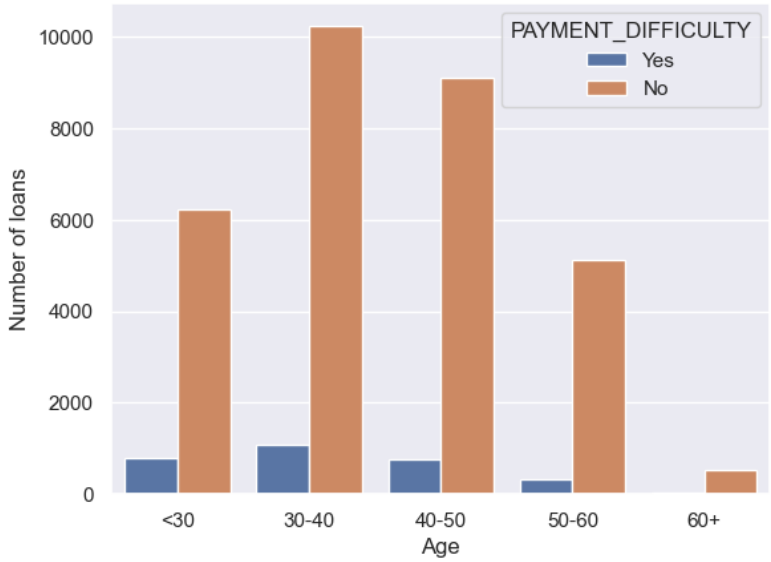
1.Gender and Payment difficulty



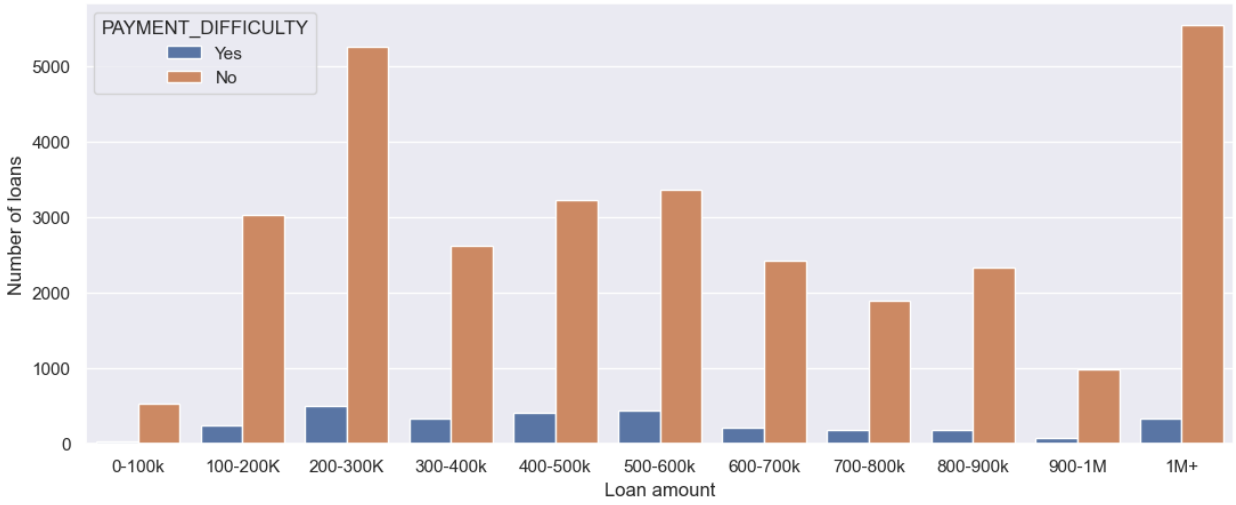
2.Income_type and payment difficulty



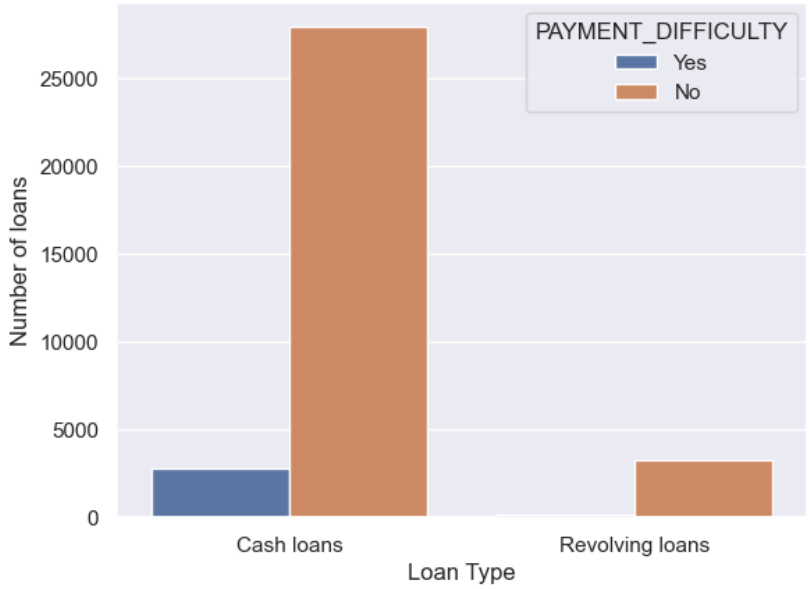
3.Age and Payment difficulty



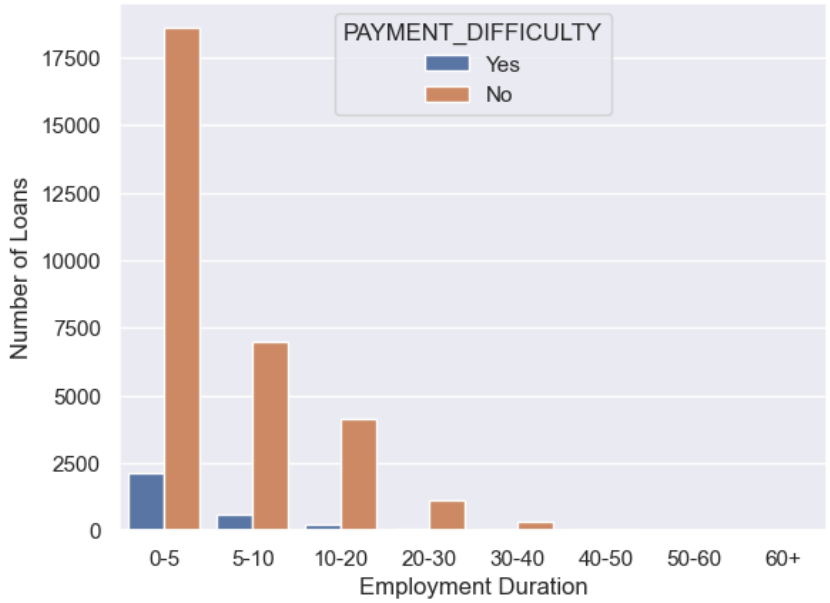
4.Loan amount and payment difficulty



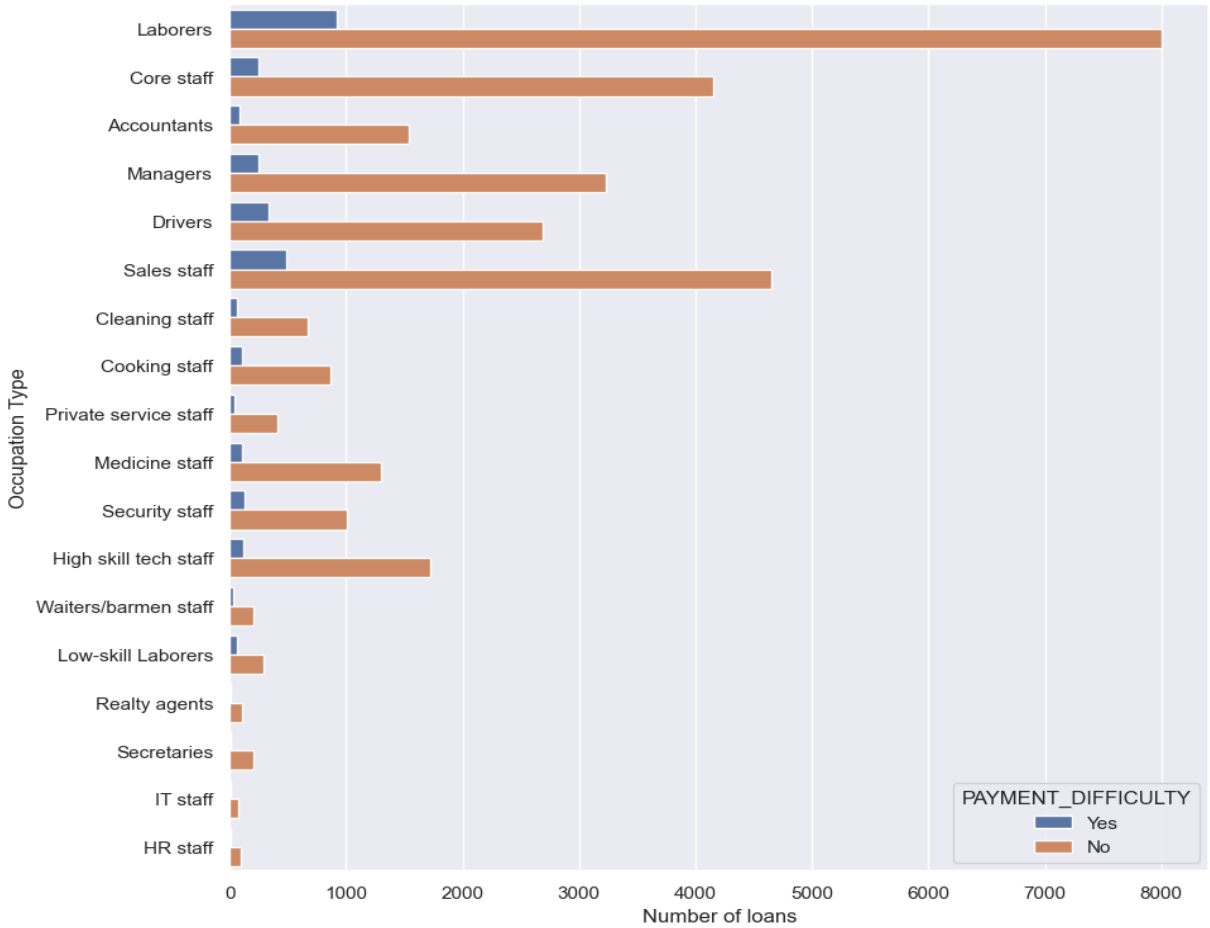
5.Loan type and payment difficulty



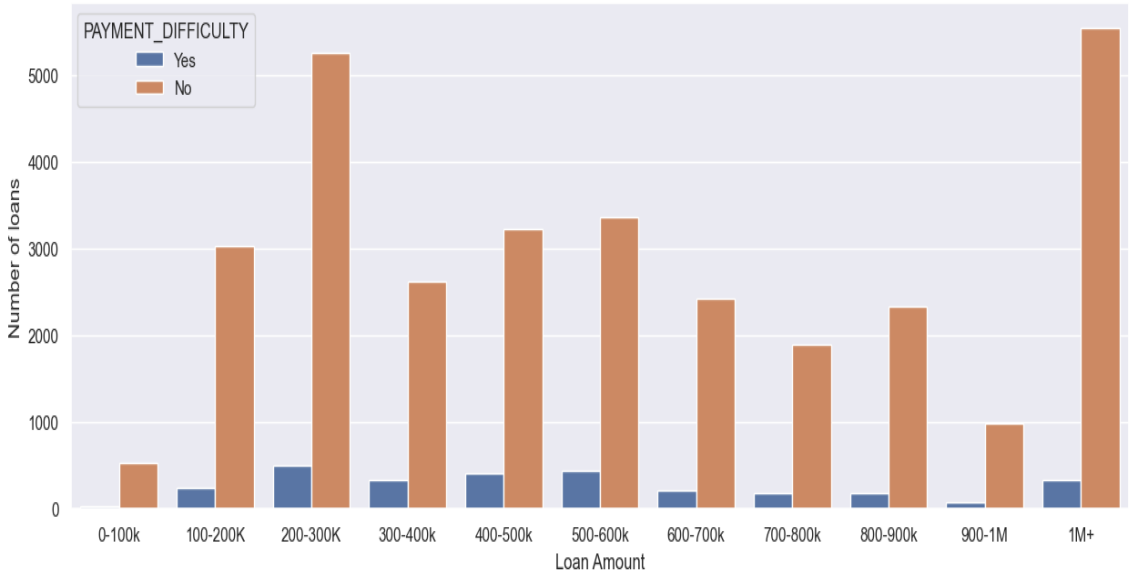
6.Year employed and Payment difficulty



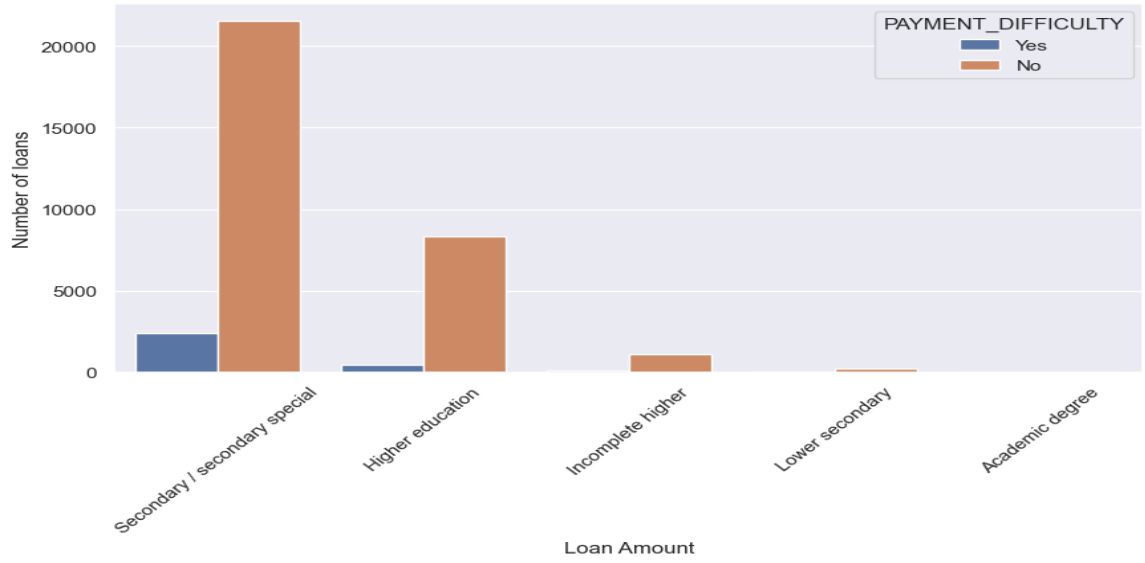
7.Occupation type and payment difficulty



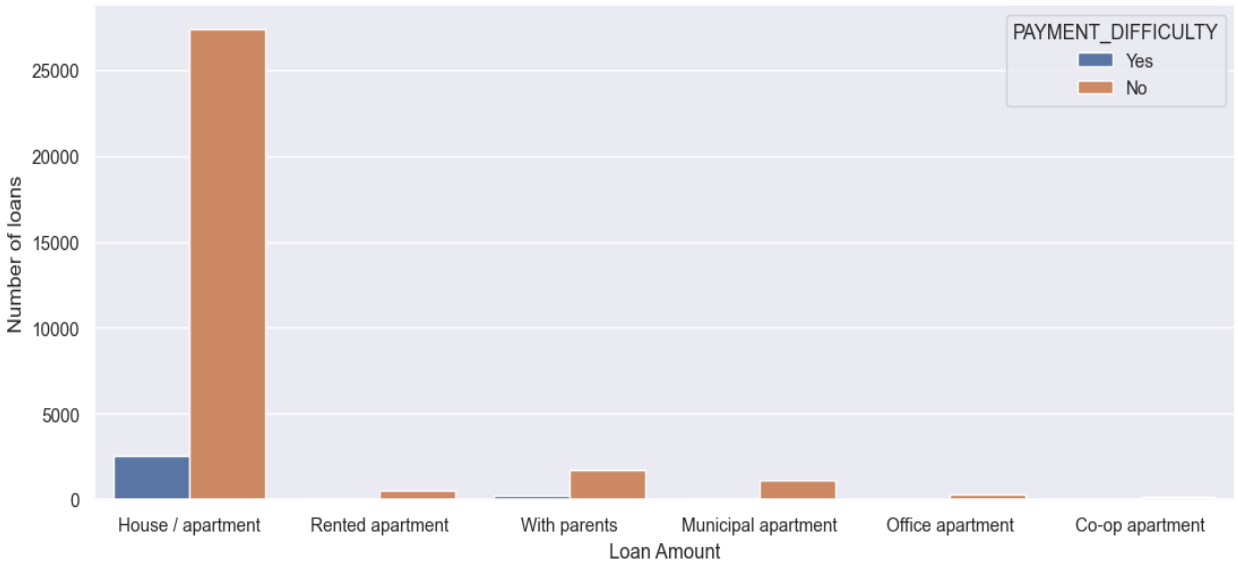
8.Credit amount and payment difficulty



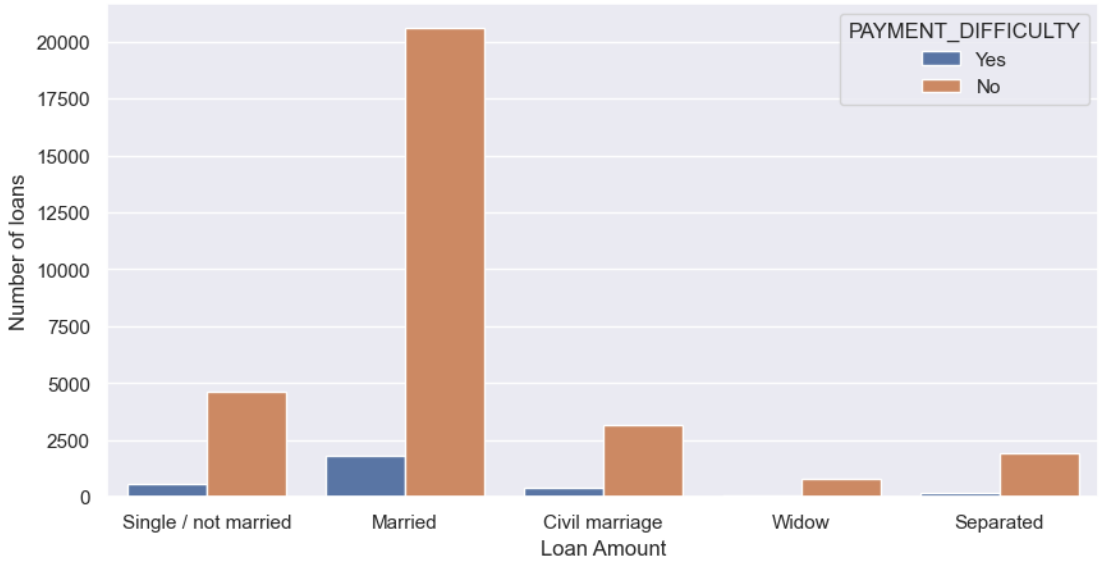
9.Education type and Payment difficulty



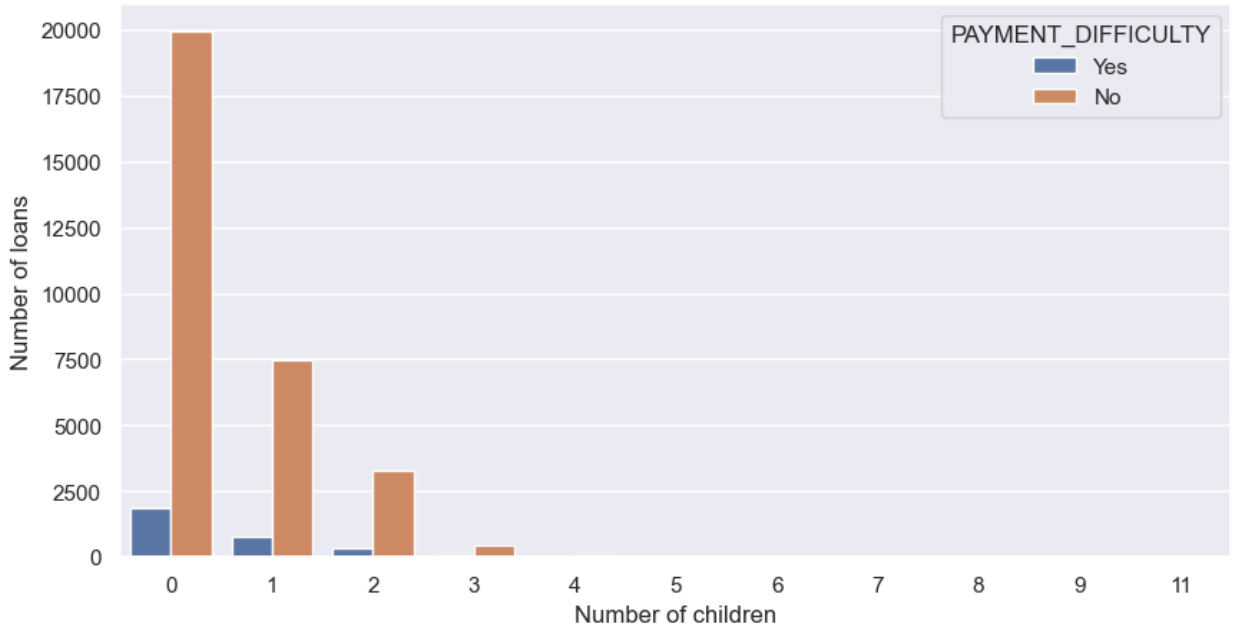
9.Housing type and payment difficulty



10.Family status and payment difficulty



11.Number of children and payment difficulty

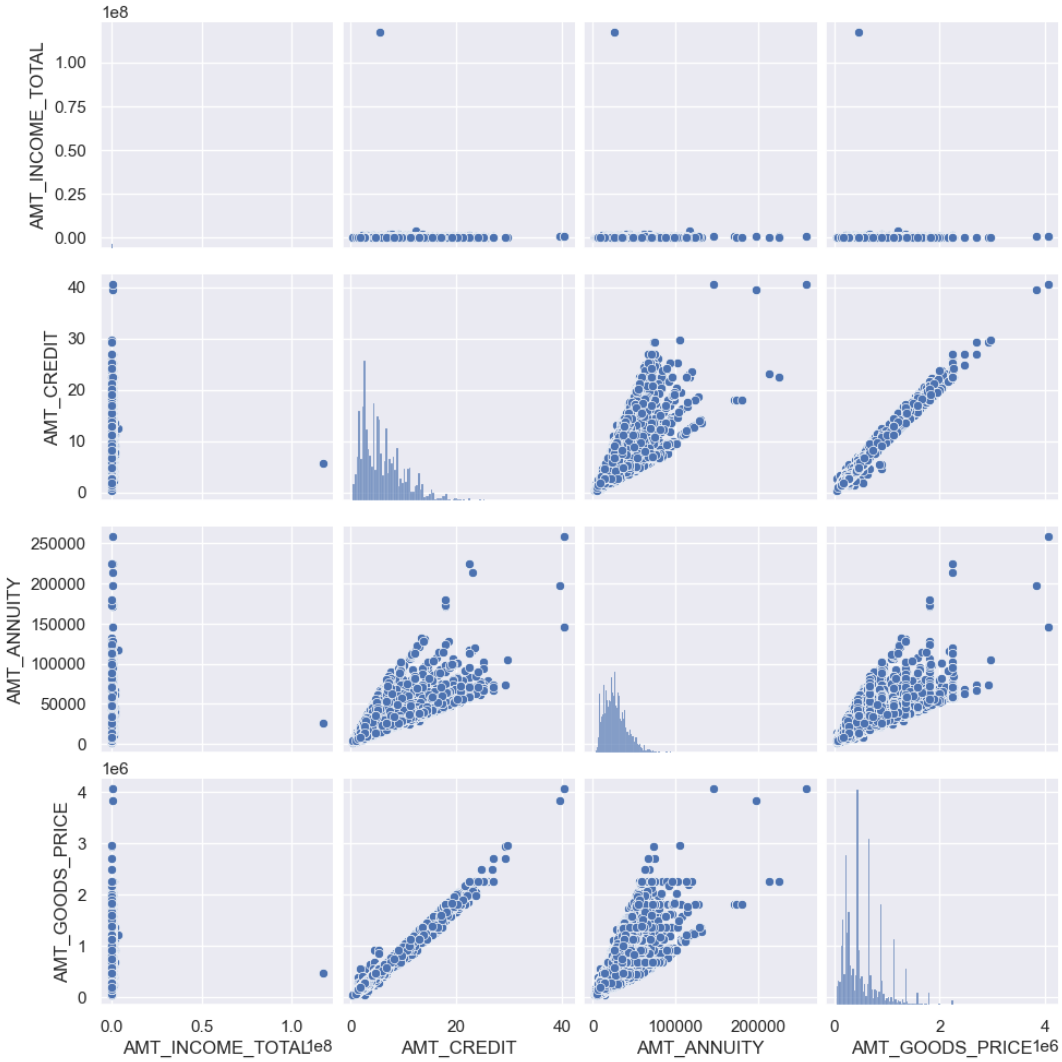

PAYMENT_DIFFICULTY
Yes
No
Number of loans
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
6
7
8
9
11
Number of children

Identify Top Correlations for Different Scenarios

Creating new variables “Defaulters” and “Non-Defaulters”

```
Defaulters = df[df["TARGET"] == 1]
NonDefaulters = df[df["TARGET"]== 0]
```

Creating pair plot to identify the relation between the features



There is a linear relationship between:

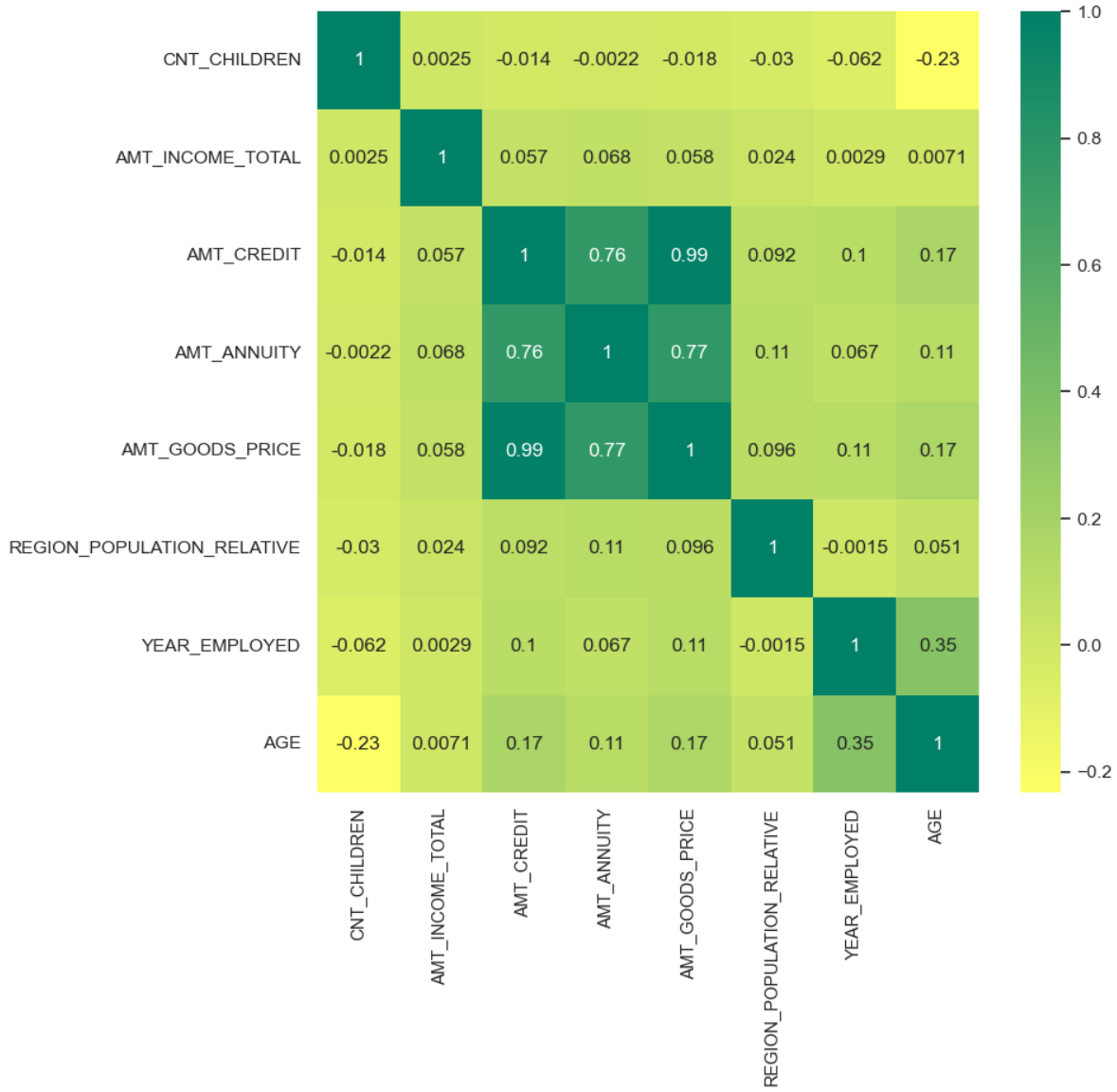
1. AMT_CREDIT and AMT_GOODS_PRICE
2. AMT_ANNUITY and GOODS_PRICE

Heatmap

```
plt.figure(figsize=[9,9])
sns.heatmap(df[["CNT_CHILDREN","AMT_INCOME_TOTAL","AMT_CREDIT","AMT_ANNUITY",
```

```
        "AMT_GOODS_PRICE","REGION_POPULATION_RELATIVE","YEAR_EMPLOYED","AGE"]].c
orr(), annot=True, cmap="summer_r")
plt.show()
```

Output



Previous_application.csv

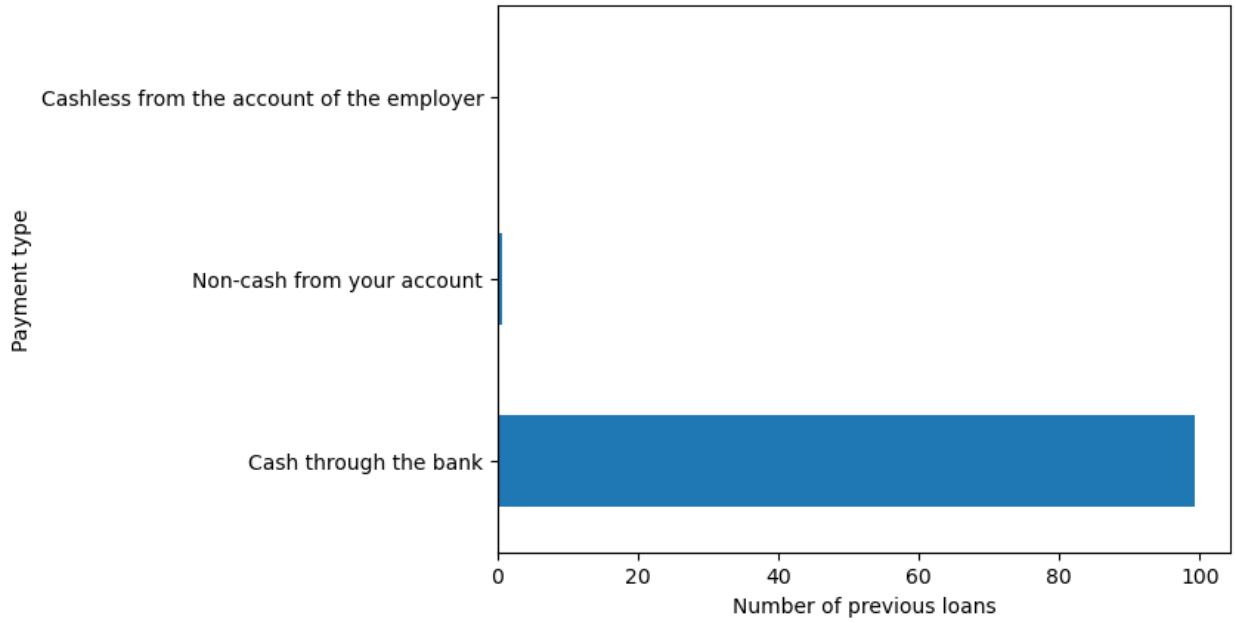
UNIVARIATE ANALYSIS

1.Payment type and previous applications

```
payment_type = df1['NAME_PAYMENT_TYPE'].value_counts(normalize=True)*100
payment_type.plot.barh()
```

```
#lt.xticks(rotation=45)
plt.ylabel("Payment type")
plt.xlabel("Number of previous loans")
```

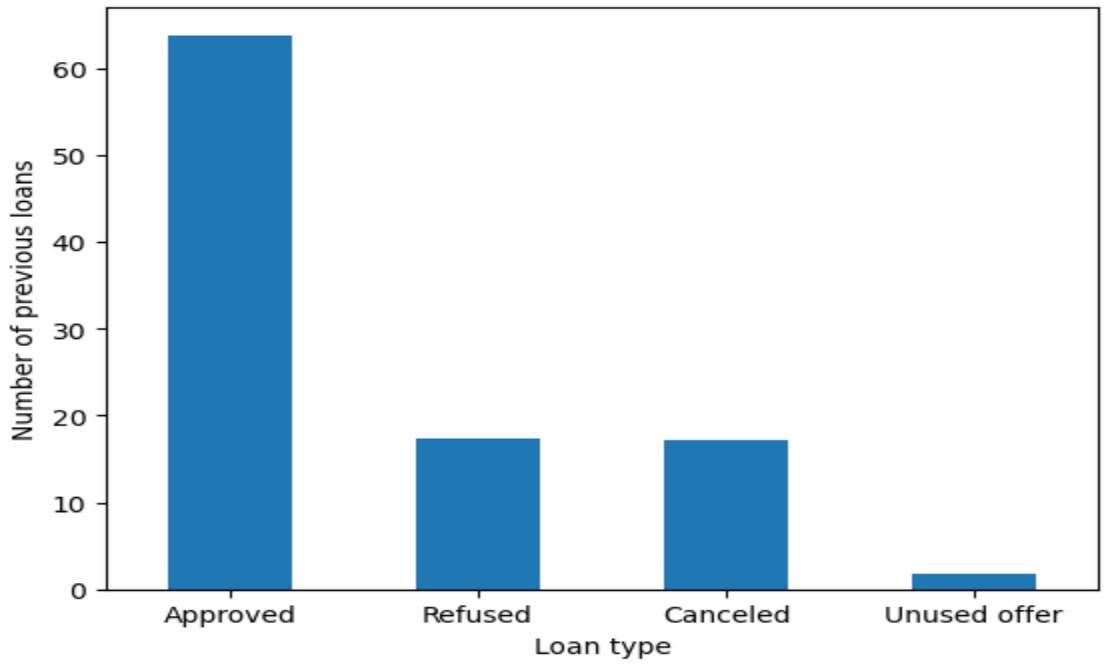
Output



2.Loan status and previous applications

```
loan_status = df1['NAME_CONTRACT_STATUS'].value_counts(normalize=True)*100
loan_status.plot.bar()
plt.xticks(rotation=360)
plt.xlabel("Loan type")
plt.ylabel("Number of previous loans")
```

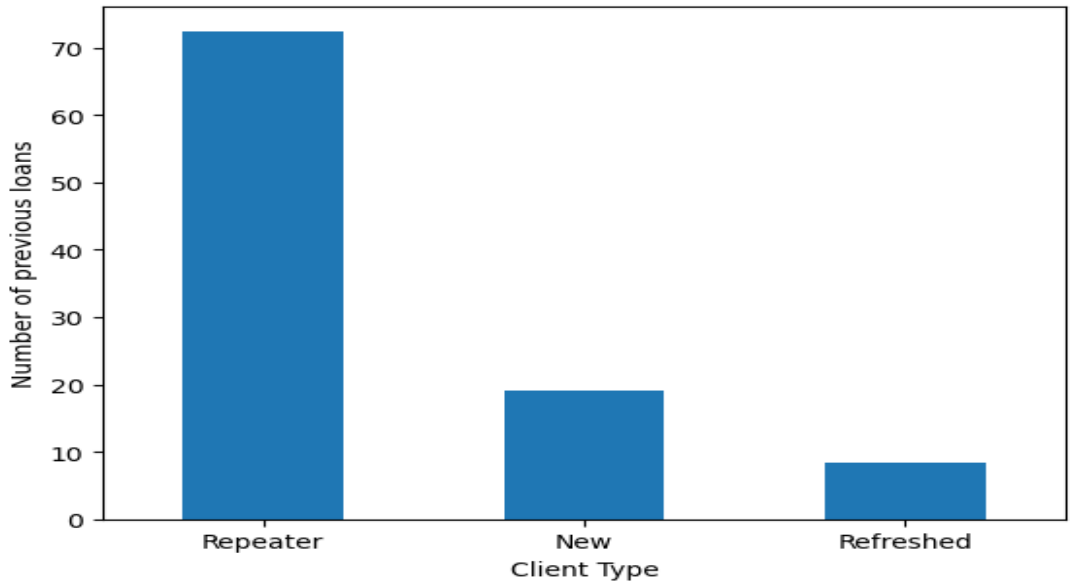
Output



3.Client type and previous applications

```
client_type = df1['NAME_CLIENT_TYPE'].value_counts(normalize=True)*100
client_type.plot.bar()
plt.xticks(rotation=360)
plt.xlabel("Client Type")
plt.ylabel("Number of previous loans")
```

Output

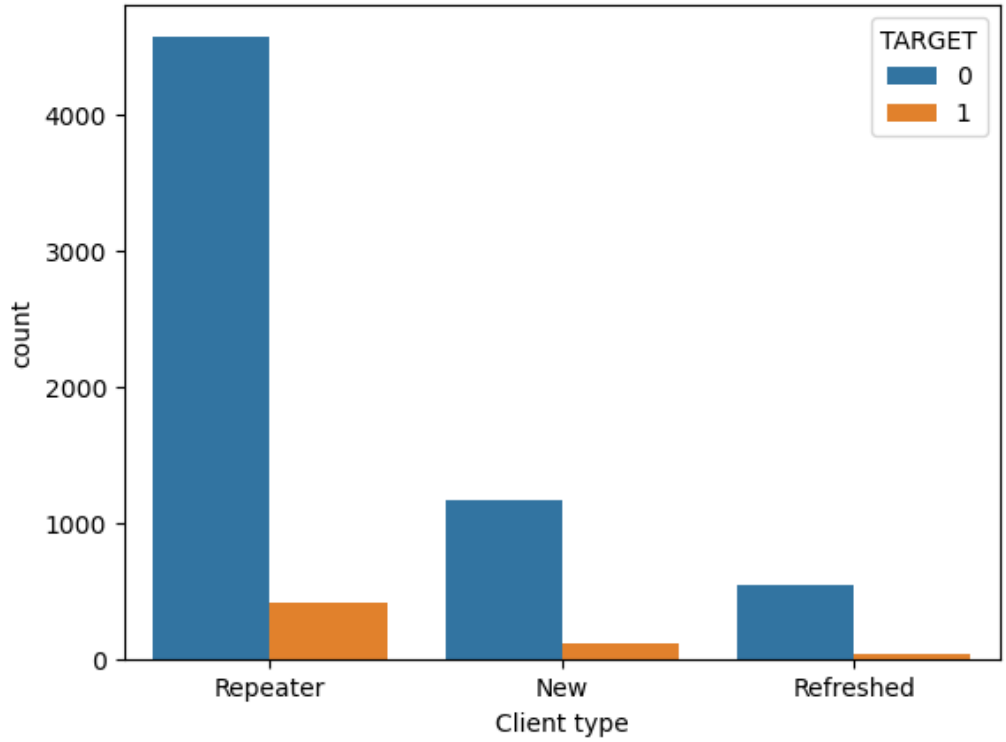


BIVARIATE ANALYSIS

1.Client type and target

```
sns.countplot(data=merge1,x="NAME_CLIENT_TYPE", hue="TARGET")
plt.xlabel("Client type")
```

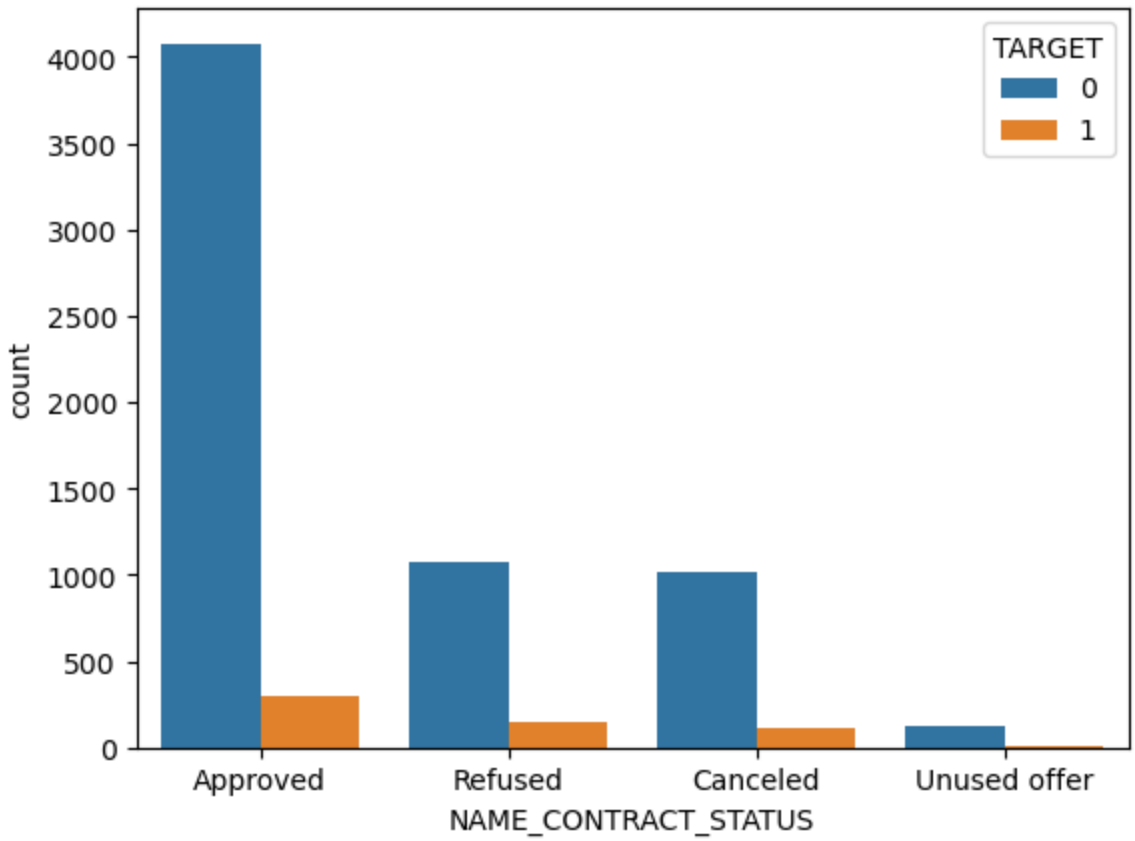
Output



2.Loan type and Target

```
sns.countplot(data=merge1,x="NAME_CONTRACT_STATUS", hue="TARGET")
```

Output



Correlation Analysis

Correlation between Loan amount, Goods price and Payment count

```
plt.figure(figsize=[5,5])
sns.heatmap(df1[["AMT_ANNUITY","AMT_CREDIT","AMT_GOODS_PRICE","CNT_PAYMENT"]].corr(
), annot=True, cmap="summer_r")
plt.show()
```

Output


AMT_ANNUITY
AMT_CREDIT
AMT_GOODS_PRICE
CNT_PAYMENT
1
0.78
0.81
0.39
0.78
1
0.94
0.63
0.81
0.94
1
0.65
0.39
0.63
0.65
1
AMT_ANNUITY
AMT_CREDIT
AMT_GOODS_PRICE
CNT_PAYMENT
1.0
0.9
0.8
0.7
0.6
0.5
0.4