Problem Statement

Business Understanding

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 to their advantage by becoming a defaulter. Suppose you work for a consumer finance company which specialises in lending various types of loans to urban customers. You have to use EDA to analyse the patterns present in the data. This will ensure that the applicants capable of repaying the loan are not rejected.

When the company receives a loan application, the company has to decide for loan approval based on the applicant's profile. Two types of risks are associated with the bank's decision:

1. If the applicant is likely to repay the loan, then not approving the loan results in a loss of business to the company.
2. If the applicant is not likely to repay the loan, i.e. he/she is likely to default, then approving the loan may lead to a financial loss for the company.

The data given below contains the information about the loan application at the time of applying for the loan. It contains two types of scenarios:

The client with payment difficulties: he/she had late payment more than X days on at least one of the first Y instalments of the loan in our sample,

All other cases: All other cases when the payment is paid on time.

When a client applies for a loan, there are four types of decisions that could be taken by the client/company):

1. Approved: The Company has approved loan Application
2. Cancelled: The client cancelled the application sometime during approval. Either the client changed her/his mind about the loan or in some cases due to a higher risk of the client, he received worse pricing which he did not want.
3. Refused: The company had rejected the loan (because the client does not meet their requirements etc.).
4. Unused offer: Loan has been cancelled by the client but at different stages of the process.

Business Objectives

This case study aims to identify patterns which indicate if a client has difficulty paying their instalments which may be used for taking actions such as denying the loan, reducing the amount of loan, lending (to risky applicants) at a higher interest rate, etc. This will ensure that the consumers capable of repaying the loan are not rejected. Identification of such applicant's using EDA is the aim of this case study. In other words, the company wants to understand the driving factors (or driver variables) behind loan default, i.e., the variables which are strong indicators of default. The company can utilise this knowledge for its portfolio and risk assessment.

Analysis steps

We followed the below EDA steps to solve the business problem and provide the solution:

- Data sourcing
- Data cleaning
- Univariate analysis
- Bivariate and multivariate analysis

Data Sourcing

Before proceeding with the Analysis, we do import some Python libraries such as Pandas, NumPy, Matplotlib and Seaborn.

```
import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Data sourcing is the very first step of EDA. Below is the data and its code for reading the data.

```
myfile_application = pd.read_csv('/content/drive/MyDrive/Data Visualization/application_data.csv')
myfile_application.head()
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY
0	100002	1	Cash loans	M	N	Y	0	202500.0	406597.5	24700.5
1	100003	0	Cash loans	F	N	N	0	270000.0	1293502.5	35698.5
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	135000.0	6750.0
3	100006	0	Cash loans	F	N	Y	0	135000.0	312682.5	29686.5
4	100007	0	Cash loans	M	N	Y	0	121500.0	513000.0	21865.5

5 rows × 122 columns

Data Cleaning

In data cleaning, we've to deal with missing values, anomalies/outliers, incorrect format and inconsistent spelling, incorrect data types. Below are the steps,

1. Identifying the data types
2. Fixing the rows and columns
3. Imputing/removing missing values
4. Handling outliers
5. Standardising the values
6. Fixing invalid values
7. Filtering the data

Missing Values

In this case study, we did find lot of null values. Below is the code snippet.

```
myfile_application.isnull().sum()

SK_ID_CURR                        0
TARGET                            0
NAME_CONTRACT_TYPE                0
CODE_GENDER                       0
FLAG_OWN_CAR                      0
                                ...
AMT_REQ_CREDIT_BUREAU_DAY     41519
AMT_REQ_CREDIT_BUREAU_WEEK    41519
AMT_REQ_CREDIT_BUREAU_MON     41519
AMT_REQ_CREDIT_BUREAU_QRT     41519
AMT_REQ_CREDIT_BUREAU_YEAR    41519
Length: 122, dtype: int64
```

We calculated the percentage of missing values for each column. Below is the code snippet.

```
def check_missing_values(data):
  total = myfile_application.isnull().sum()
  percent = (data.isnull().sum()/data.isnull().count()*100)
  return pd.concat([total, percent], axis=1, keys=['Total', 'Percent']).sort_values(by="Percent", ascending=False)

application_data=check_missing_values(myfile_application)
application_data.head(50)
```

	Total	Percent
COMMONAREA_MEDI	214865	69.872297
COMMONAREA_AVG	214865	69.872297
COMMONAREA_MODE	214865	69.872297
NONLIVINGAPARTMENTS_MODE	213514	69.432963
NONLIVINGAPARTMENTS_AVG	213514	69.432963
NONLIVINGAPARTMENTS_MEDI	213514	69.432963
FONDKAPREMONT_MODE	210295	68.386172
LIVINGAPARTMENTS_MODE	210199	68.354953

Dropping columns with High Missing Values

```
final_cols=list(application_data[(application_data.Percent<60)].index)
myfile_application=myfile_application[final_cols]
myfile_application.describe()
```

```
final_cols=list(application_data[(application_data.Percent<60)].index)
myfile_application=myfile_application[final_cols]
myfile_application.describe()
```

	LANDAREA_MEDI	LANDAREA_MODE	LANDAREA_AVG	BASEMENTAREA_MEDI	BASEMENTAREA_AVG	BASEMENTAREA_MODE	EXT_SOURCE_1	NONLIVINGAREA_MODE
count	124921.000000	124921.000000	124921.000000	127568.000000	127568.000000	127568.000000	134133.000000	137829.000000
mean	0.067169	0.064958	0.066333	0.087955	0.088442	0.087543	0.502130	0.027022
std	0.082167	0.081750	0.081184	0.082179	0.082438	0.084307	0.211062	0.070254
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.014568	0.000000
25%	0.018700	0.016600	0.018700	0.043700	0.044200	0.040700	0.334007	0.000000
50%	0.048700	0.045800	0.048100	0.075800	0.076300	0.074600	0.505998	0.001100
75%	0.086800	0.084100	0.085600	0.111600	0.112200	0.112400	0.675053	0.023100
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.962693	1.000000

8 rows × 90 columns

Checking columns with very less missing values

```
low_missing=pd.DataFrame(application_data[(application_data.Percent>0)&
(application_data.Percent<15)]) low_missing
```

```
low_missing=pd.DataFrame(application_data[(application_data.Percent>0)&(application_data.Percent<15)])
low_missing
```

	Total	Percent
AMT_REQ_CREDIT_BUREAU_HOUR	41519	13.501631
AMT_REQ_CREDIT_BUREAU_DAY	41519	13.501631
AMT_REQ_CREDIT_BUREAU_WEEK	41519	13.501631
AMT_REQ_CREDIT_BUREAU_MON	41519	13.501631
AMT_REQ_CREDIT_BUREAU_QRT	41519	13.501631
AMT_REQ_CREDIT_BUREAU_YEAR	41519	13.501631
NAME_TYPE_SUITE	1292	0.420148
OBS_30_CNT_SOCIAL_CIRCLE	1021	0.332021
DEF_30_CNT_SOCIAL_CIRCLE	1021	0.332021
OBS_60_CNT_SOCIAL_CIRCLE	1021	0.332021

Date Types

We checked the data types of each column. Below is the code snippet.

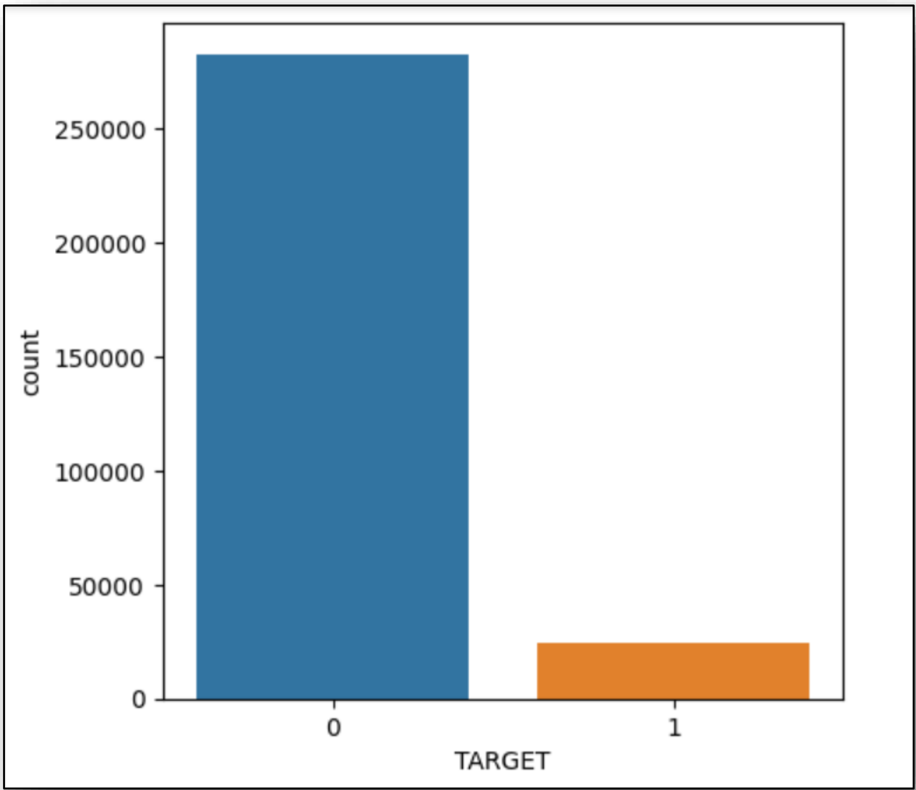
```
myfile_application.dtypes

LANDAREA_MEDI            float64
LANDAREA_MODE            float64
LANDAREA_AVG             float64
BASEMENTAREA_MEDI        float64
BASEMENTAREA_AVG         float64
                           ...
NAME_HOUSING_TYPE         object
NAME_FAMILY_STATUS        object
NAME_EDUCATION_TYPE       object
NAME_INCOME_TYPE          object
SK_ID_CURR                 int64
Length: 105, dtype: object
```

Univariate Analysis

Univariate analysis involves the analysis of a single variable at a time. The concept of univariate analysis is divided into ordered and unordered category of variables.

In this case, we focused on **TARGET** column which has the targeted audience who had late payment vs others.

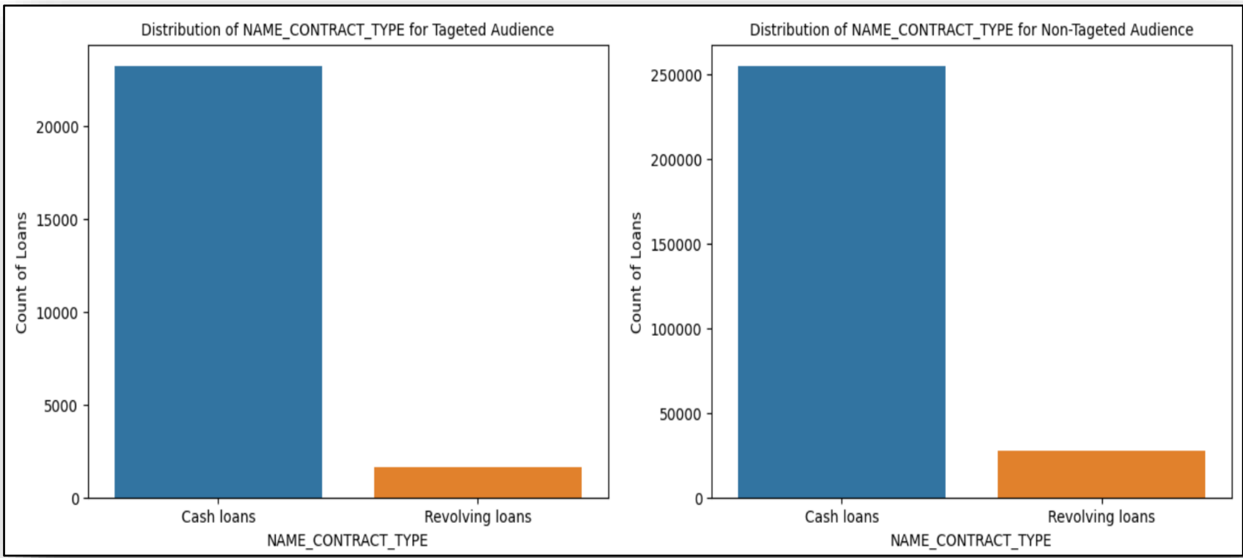
```
plt.figure(figsize=(5,5))
sns.countplot(x=myfile_application['TARGET'], data=myfile_application)
plt.show()
```



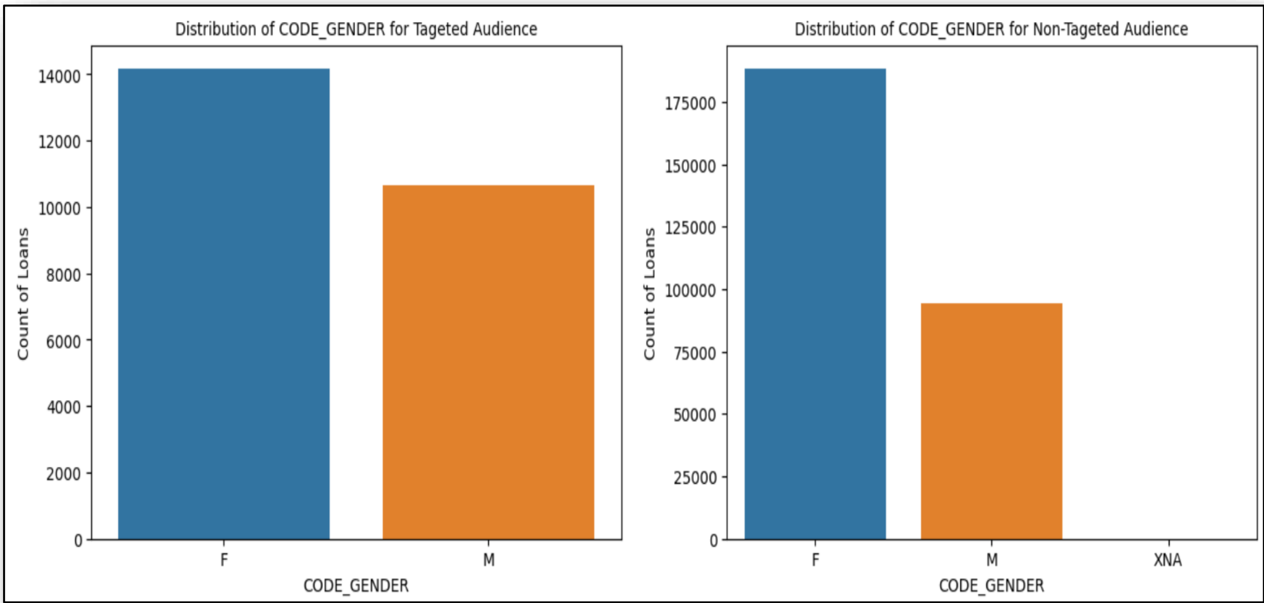
Exploratory Data Analysis

We focused on columns with dtype=object, to identify columns for categorical analysis. We can refer the code snippet of dtype=object above in Data Types heading. We'll analyse each column one-by-one with TARGET data.

Below, we can see that the number of Cash loans is much higher than the number of Revolving loans for both Target = 0 and Target = 1.

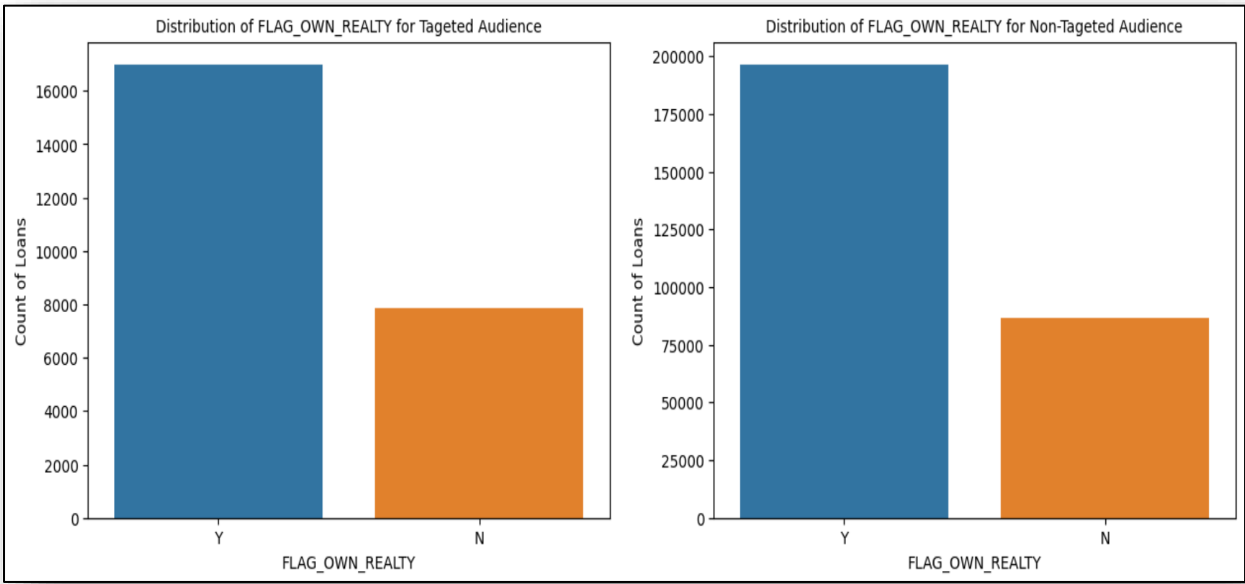


Below, we observe that the number of Females taking loans is much higher than the number of Males for both Target = 0 and Target = 1

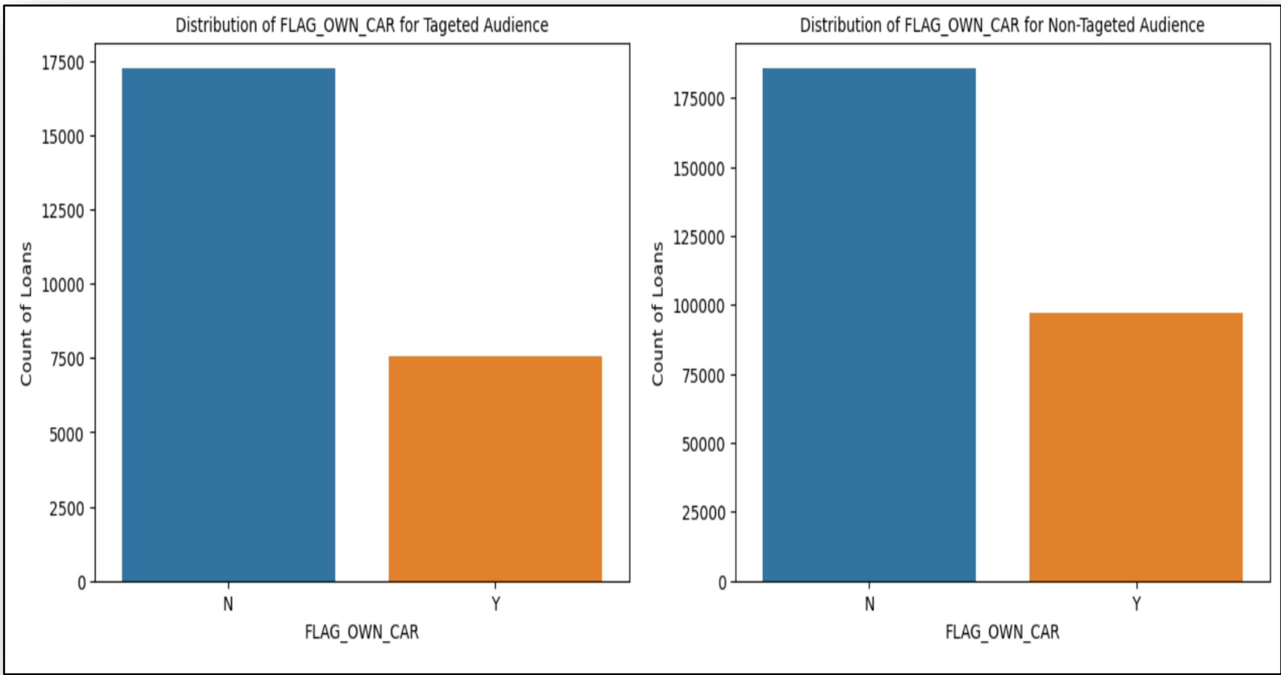


Exploratory Data Analysis

Below, we observed that the number of most people applying for loan do not own a car.

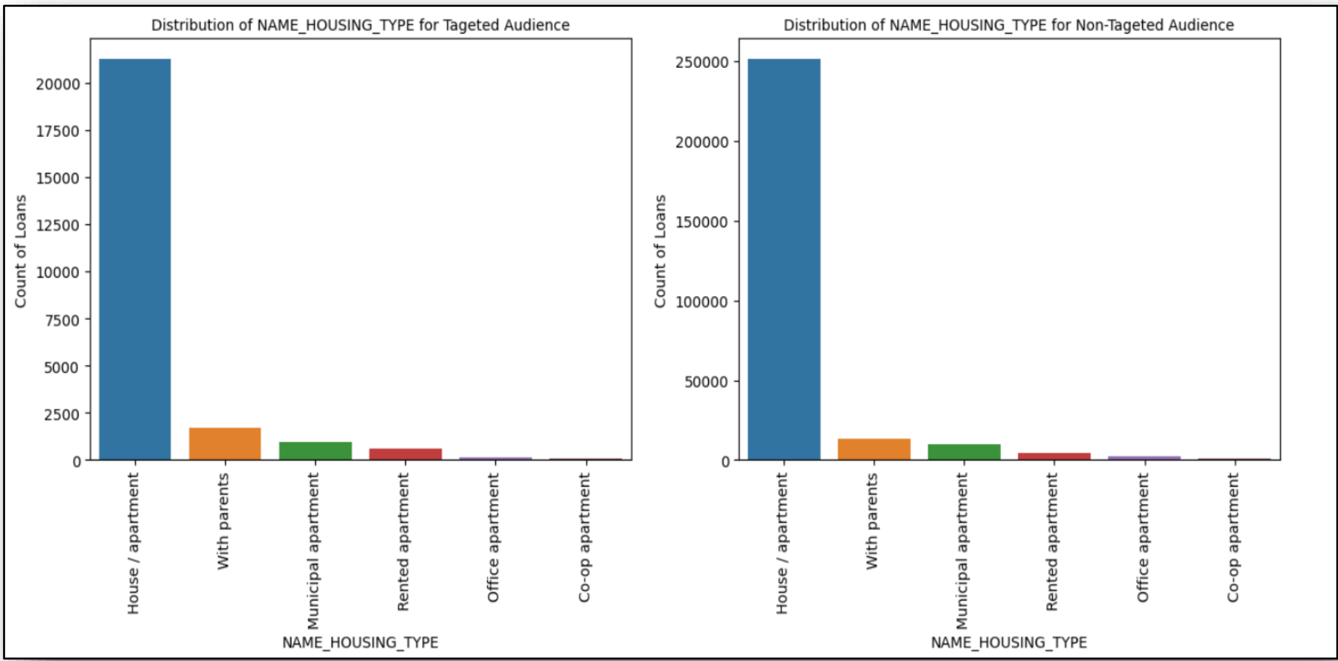


Below, we observed that the ratio of people who own a car is higher for non-targeted audience.

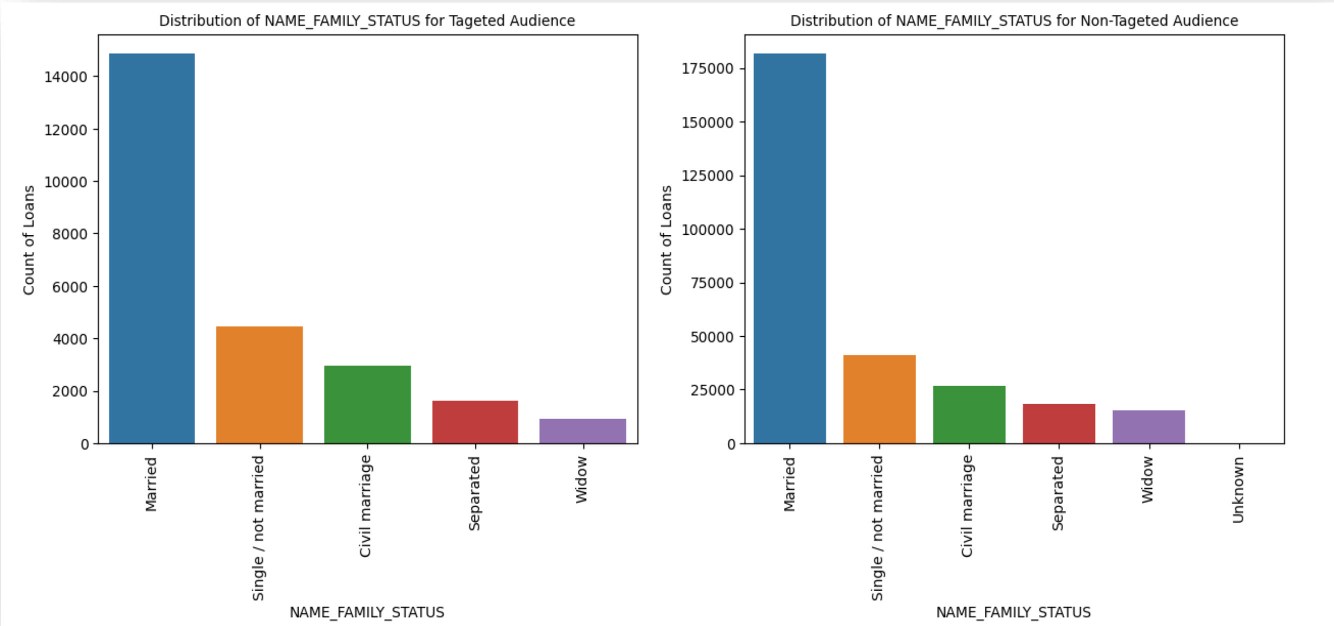


Exploratory Data Analysis

Below, we observed that people who live With Parents is more for targeted than non-targeted audience. It tells us that applicant who live with parents have a higher chance of having payment difficulties.



Below, we observed Single/Unmarried people is more for targeted than non-targeted audience. It tells us that Single/Unmarried people are more likely to have payment difficulties.



Bivariate and Multivariate Analysis

In this method, we’ve analysed two numerical variables and checked correlation coefficient. Correlation coefficient depicts only a linear relationship between numerical variables and does not depict any other relationship between variables.

Getting a list of columns with dtype=float64 and dtype=int64, to identify columns for analysis.

Input:
myfile_application.select_dtypes('float64').columns

Output:
```
Index(['LANDAREA_MEDI', 'LANDAREA_MODE', 'LANDAREA_AVG', 'BASEMENTAREA_MEDI',
       'BASEMENTAREA_AVG', 'BASEMENTAREA_MODE', 'EXT_SOURCE_1',
       'NONLIVINGAREA_MODE', 'NONLIVINGAREA_AVG', 'NONLIVINGAREA_MEDI',
       'ELEVATORS_MEDI', 'ELEVATORS_AVG', 'ELEVATORS_MODE', 'APARTMENTS_MEDI',
       'APARTMENTS_AVG', 'APARTMENTS_MODE', 'ENTRANCES_MEDI', 'ENTRANCES_AVG',
       'ENTRANCES_MODE', 'LIVINGAREA_AVG', 'LIVINGAREA_MODE',
       'LIVINGAREA_MEDI', 'FLOORSMAX_MODE', 'FLOORSMAX_MEDI', 'FLOORSMAX_AVG',
       'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BEGINEXPLUATATION_MEDI',
       'YEARS_BEGINEXPLUATATION_AVG', 'TOTALAREA_MODE', 'EXT_SOURCE_3',
       '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',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'EXT_SOURCE_2',
       'AMT_GOODS_PRICE', 'AMT_ANNUITY', 'CNT_FAM_MEMBERS',
       'DAYS_LAST_PHONE_CHANGE', 'AMT_CREDIT', 'AMT_INCOME_TOTAL',
       'DAYS_REGISTRATION', 'REGION_POPULATION_RELATIVE'],
      dtype='object')
```

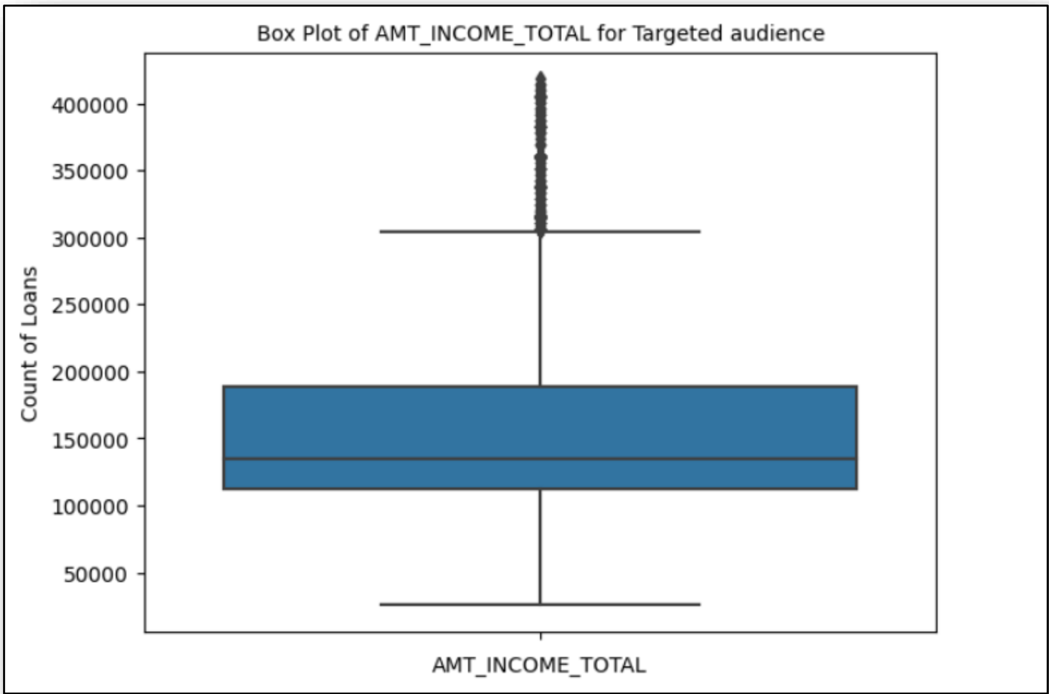
Input:
myfile_application.select_dtypes('int64').columns

Output:
```
Index(['CNT_CHILDREN', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3',
       'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6',
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_9', 'FLAG_DOCUMENT_21',
       'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', '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_12', 'FLAG_PHONE',
       'LIVE_CITY_NOT_WORK_CITY', 'REG_CITY_NOT_WORK_CITY', 'TARGET',
```

```
 'REG_CITY_NOT_LIVE_CITY', 'LIVE_REGION_NOT_WORK_REGION',
 'REG_REGION_NOT_WORK_REGION', 'REG_REGION_NOT_LIVE_REGION',
 'HOUR_APPR_PROCESS_START', 'REGION_RATING_CLIENT_W_CITY',
 'REGION_RATING_CLIENT', 'FLAG_EMAIL', 'FLAG_CONT_MOBILE',
 'FLAG_WORK_PHONE', 'FLAG_EMP_PHONE', 'FLAG_MOBIL', 'DAYS_ID_PUBLISH',
 'DAYS_EMPLOYED', 'DAYS_BIRTH', 'SK_ID_CURR'],
dtype='object')
```

In this Analysis, we've used '**AMT_INCOME_TOTAL**' which depict the income of clients. We see some outlier values in this column. After removing outlier values, we could see the below chart.

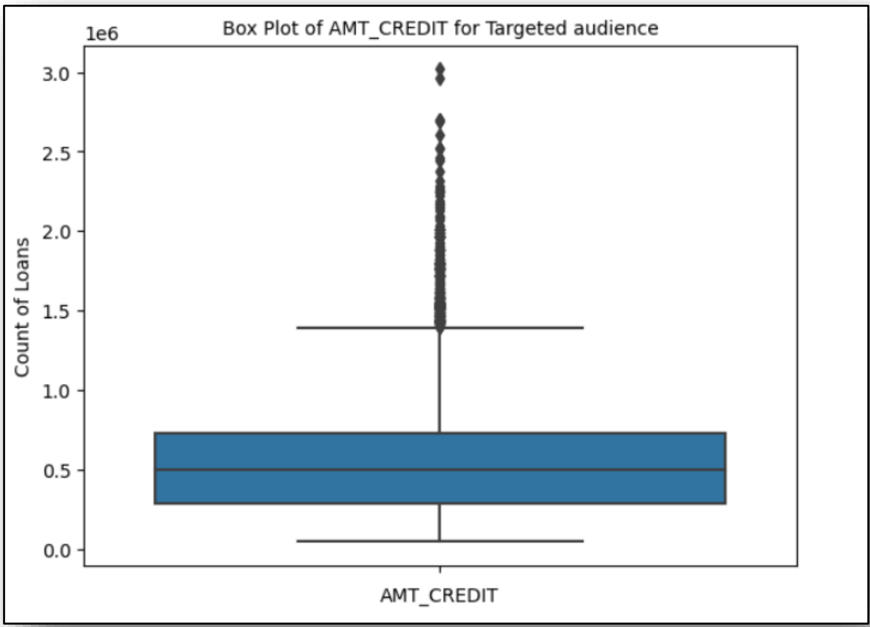
It tells us that most people with payment have incomes in the lower range between 100000 to 200000 which some on the higher end some on the lower



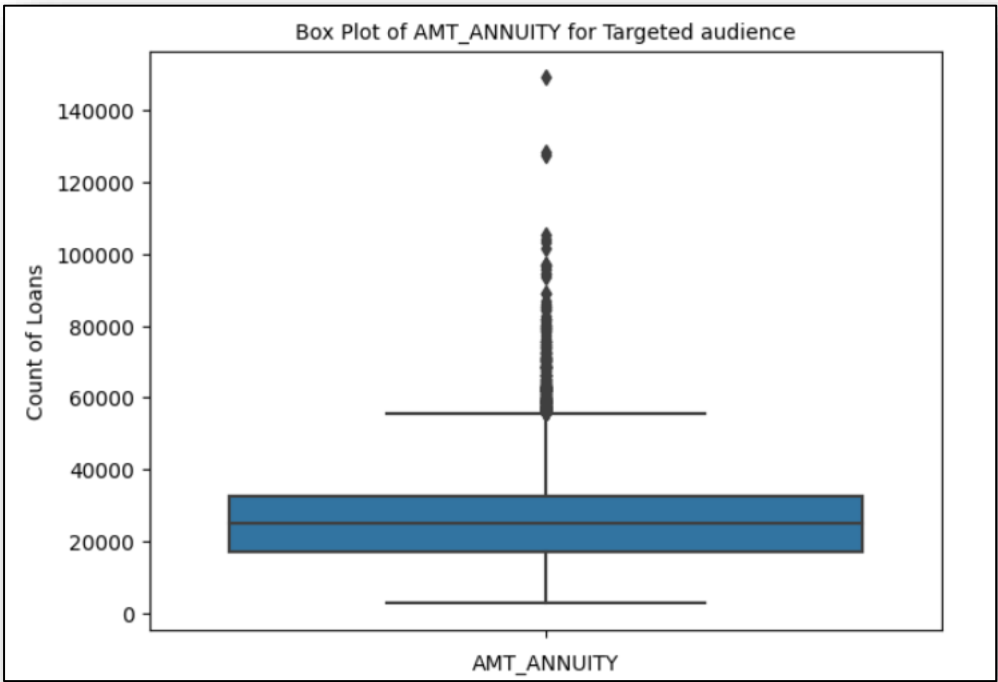
Exploratory Data Analysis

Next, ‘**AMT_CREDIT**’ which depict the Credit amount of the loan. We see some outlier values in this column too. After removing outlier values, we could see the below chart.

We can see that the credit amount lies between 250000 to around 500000 for Targeted audience.

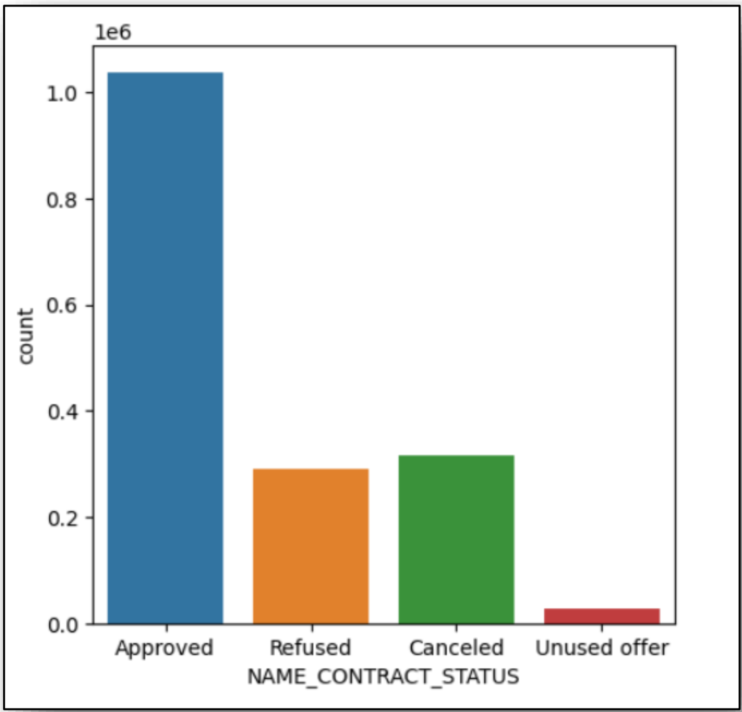


Next, ‘**AMT_ANNUITY**’ which depicts Loan annuity of the clients. We can see that the loan annuity lies between 18000 to around 30000 for Targeted audience.



Analysis of Previous Application Dataset

In this case, we focused on '**NAME_CONTRACT_STATUS'** which depicts the contract status (approved, cancelled, refused, unused) of previous application.



We identified some missing values into the **Previous Application Dataset.** We calculated the percentage of missing values and drop those columns using below code.

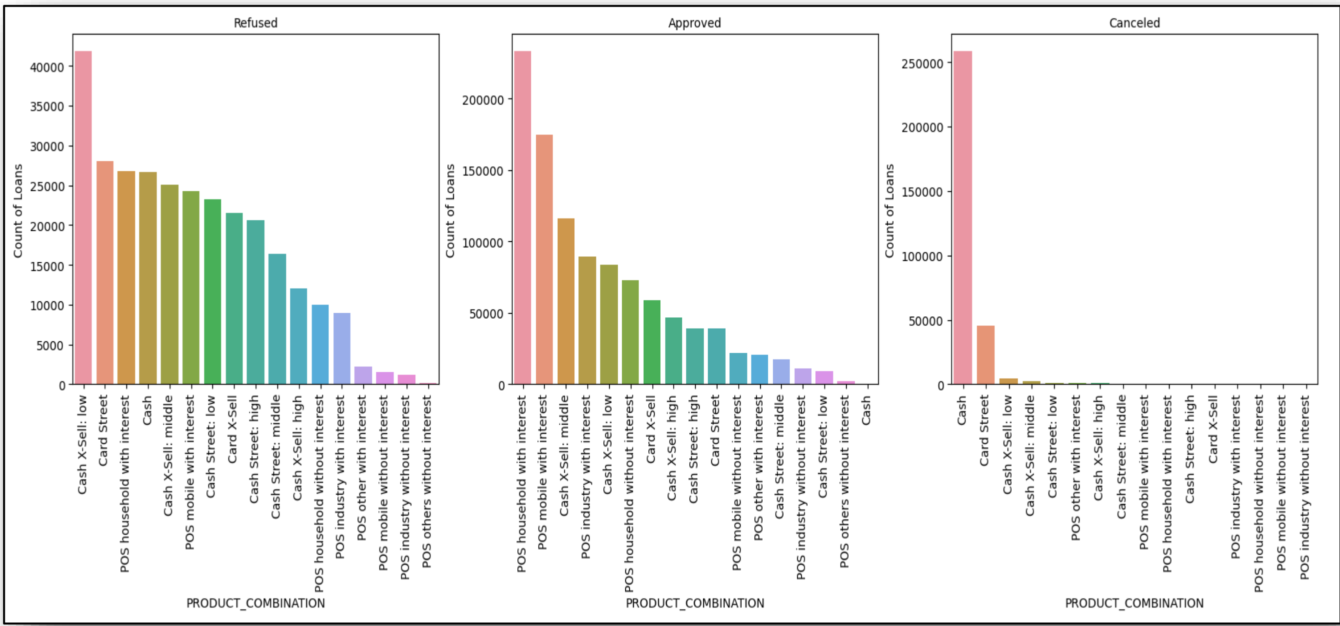
```
def check_missing_values2(data):
  total = myfile_prev_application.isnull().sum()
  percent = (data.isnull().sum()/data.isnull().count()*100)
  return pd.concat([total, percent], axis=1, keys=['Total',
'Percent']).sort_values(by="Percent", ascending=False)

application_prev_data=check_missing_values2(myfile_prev_application)
application_prev_data.head(50)
```

```
cols_to_keep=list(application_prev_data[(application_prev_data.Percent<
50)].index)
previous_data=myfile_prev_application[cols_to_keep]
previous_data.describe()
```

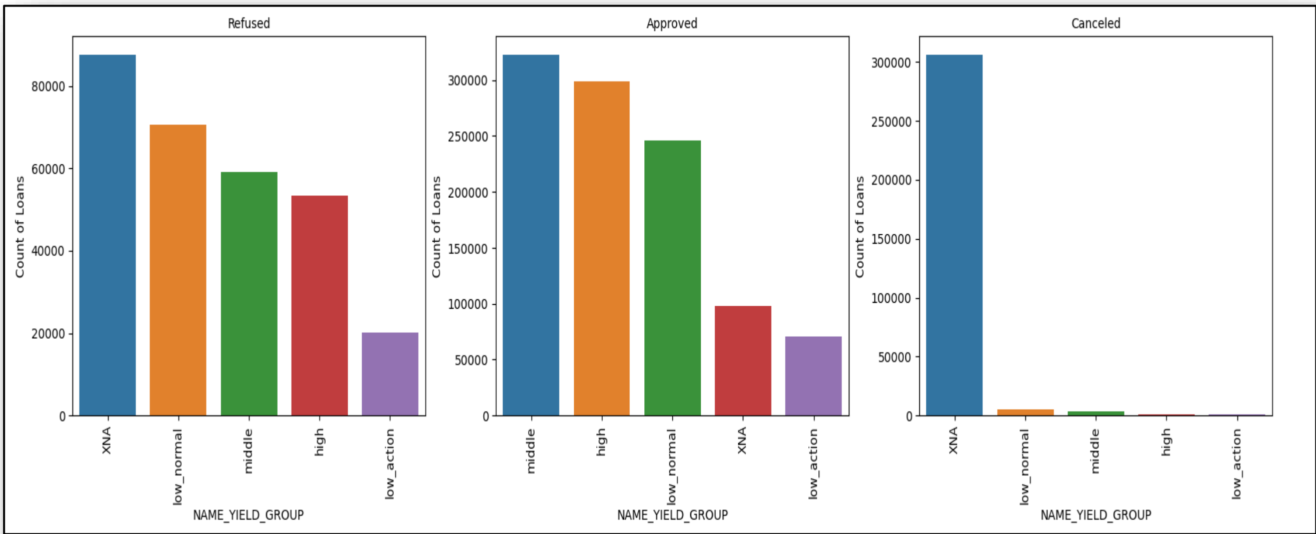
Exploratory Data Analysis

We started analysing each column one-by-one to see the relation.

Below, we observed most number of loans were approved for POS household with interest where most number of refused loans were of Cash X-Sell: Low Product combination and most Canceled loans were Cash loans

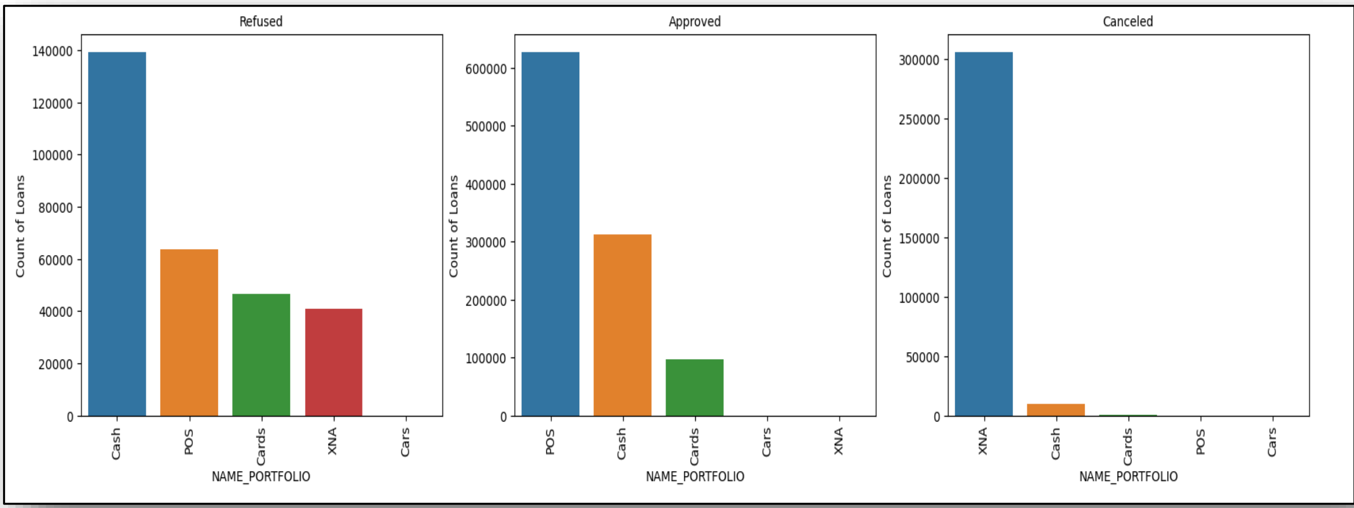


Below charts tells us that most approved loans were from Middle Yield Goup and most refused loans were from Yield Goups Not specified.

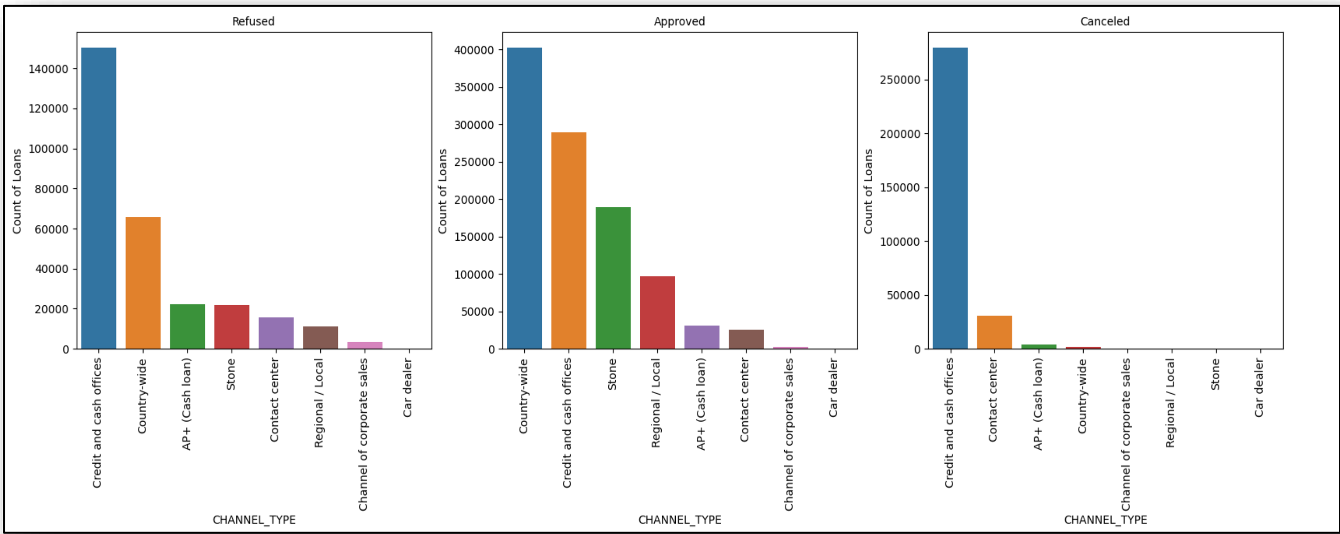


Exploratory Data Analysis

Below charts tells us that most approved loans were POS and most refused loans were Cash.



Below charts tells us that most approved loans were from Country-wide Channel and most refused loans were from Credit and Cash Offices Channel.



Summary

In the current application, we observe that:

- the number of Cash loans is much higher than the number of Revolving loans for both Target = 0 and Target = 1.
- the number of Females taking loans is much higher than the number of Males for both Target = 0 and Target = 1.
- the number of most people applying for loan do not own a car and the ratio of people who own a car is higher for non-targeted audience.
- the people who live With Parents is more for targeted than non-targeted audience which means that the applicant who live with parents have a higher chance of having payment difficulties.
- the Single/Unmarried people is more for targeted than non-targeted audience which means that the Single/Unmarried people are more likely to have payment difficulties.

In the previous application, we observe that:

- the most number of loans were approved for POS household with interest.
- the most number of refused loans were of Cash X-Sell: Low Product combination.
- the most Cancelled loans were Cash loans.
- the most approved loans were from Middle Yield Goup.
- the most refused loans were from Yield Goups Not specified.
- the number of Females taking loans is much higher than the number of Males for both Target = 0 and Target = 1.
- the most approved loans were POS.
- the most refused loans were Cash.
- the most approved loans were from Country-wide Channel.
- the most refused loans were from Credit and Cash Offices Channel.