Loan Case Study

Introduction:

This case study aims to give you an idea of applying EDA in a real business scenario. In this case study, apart from applying the techniques that you have learnt in the EDA module, you will also develop a basic understanding of risk analytics in banking and financial services and understand how data is used to minimize the risk of losing money while lending to customers.

Project Description:

This project contains the following datasets:

1. **'application_data.csv'** contains all the information of the client at the time of application. The data is about whether a client has payment difficulties.

2. **'previous_application.csv'** contains information about the client's previous loan data. It contains the data whether the previous application had been Approved, Cancelled, Refused or Unused offer.

3. **'columns_description.csv'** is data dictionary which describes the meaning of the variables.

Approach:

I loaded the dataset, understand it, clean it and perform the tasks.

Tech-Stack Used:

I have used the google online notebook Collab for this project. The purpose behind using Collab is that we don't need to install any notebook software locally on my system.

Insights:

With this project, I learnt how to approach the problem, how to convert the logic into code and hidden insights I can get via libraries like matplotlib which helps to plot the graphs.

Result:

Now I am comfortable working with python libraries like Numpy and Pandas, because this project contains a whole lot of problems which I solved.

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 as 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:

• If the applicant is likely to repay the loan, then not approving the loan results in a loss of business to the company.

• 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 on different stages of the process.
In this case study, you will use EDA to understand how consumer attributes and loan attributes influence the tendency of default.

Business Objectives

This case study aims to **identify patterns** which indicate if a client has difficulty paying their installments 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 applicants 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 utilize this knowledge for its portfolio and risk assessment.

To develop your understanding of the domain, you are advised to independently research a little about **risk analytics** – understanding the types of variables and their significance should be enough).

Data Understanding

1. '**application_data.csv**' contains all the information of the client at the time of application. The data is about whether a client has payment difficulties.
2. '**previous_application.csv**' contains information about the client's previous loan data. It contains the data whether the previous application had been Approved, Cancelled, Refused or Unused offer.
3. '**columns_description.csv**' is data dictionary which describes the meaning of the variables.

DATASET LINK

DRIVE LINK

Complete Notebook is provided below:

Loan Case Study

AIM:

This case study aims to identify patterns which indicate if a client has difficulty paying their installments 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 applicants using EDA is the aim of this case study.

1. Importing the libraries and files

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
plt.style.use('dark_background')

pd.set_option('display.max_columns', 300) # to display all the columns
pd.set_option('display.max_rows', 30) # to display all the rows
pd.set_option('display.width', 1000)

warnings.filterwarnings('ignore') #To ignore the warnings
```

```
newapp = pd.read_csv('application_data.csv')
preapp = pd.read_csv('previous_application.csv')
```

2. newapp Data Routine Check

```
newapp.head()
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALT
0	100002	1	Cash loans	M	N	`
1	100003	0	Cash loans	F	N	N
2	100004	0	Revolving loans	M	Y	`

```
newapp.shape
```

```
(307511, 122)
```

```
newapp.info(verbose=True, null_counts=True)
```

```
 66   FLOORSMIN_MODE                98869 non-null   float64
 67   LANDAREA_MODE                 124921 non-null  float64
 68   LIVINGAPARTMENTS_MODE         97312 non-null   float64
 69   LIVINGAREA_MODE               153161 non-null  float64
 70   NONLIVINGAPARTMENTS_MODE      93997 non-null   float64
 71   NONLIVINGAREA_MODE            137829 non-null  float64
 72   APARTMENTS_MEDI               151450 non-null  float64
 73   BASEMENTAREA_MEDI             127568 non-null  float64
 74   YEARS_BEGINEXPLUATATION_MEDI  157504 non-null  float64
 75   YEARS_BUILD_MEDI              103023 non-null  float64
 76   COMMONAREA_MEDI               92646 non-null   float64
 77   ELEVATORS_MEDI                143620 non-null  float64
 78   ENTRANCES_MEDI                152683 non-null  float64
 79   FLOORSMAX_MEDI                154491 non-null  float64
 80   FLOORSMIN_MEDI                98869 non-null   float64
 81   LANDAREA_MEDI                 124921 non-null  float64
 82   LIVINGAPARTMENTS_MEDI         97312 non-null   float64
 83   LIVINGAREA_MEDI               153161 non-null  float64
 84   NONLIVINGAPARTMENTS_MEDI      93997 non-null   float64
 85   NONLIVINGAREA_MEDI            137829 non-null  float64
 86   FONDKAPREMONT_MODE            97216 non-null   object
 87   HOUSETYPE_MODE                153214 non-null  object
 88   TOTALAREA_MODE                159080 non-null  float64
 89   WALLSMATERIAL_MODE            151170 non-null  object
 90   EMERGENCYSTATE_MODE           161756 non-null  object
 91   OBS_30_CNT_SOCIAL_CIRCLE      306490 non-null  float64
 92   DEF_30_CNT_SOCIAL_CIRCLE      306490 non-null  float64
 93   OBS_60_CNT_SOCIAL_CIRCLE      306490 non-null  float64
 94   DEF_60_CNT_SOCIAL_CIRCLE      306490 non-null  float64
 95   DAYS_LAST_PHONE_CHANGE        307510 non-null  float64
 96   FLAG_DOCUMENT_2               307511 non-null  int64
 97   FLAG_DOCUMENT_3               307511 non-null  int64
 98   FLAG_DOCUMENT_4               307511 non-null  int64
 99   FLAG_DOCUMENT_5               307511 non-null  int64
 100  FLAG_DOCUMENT_6               307511 non-null  int64
 101  FLAG_DOCUMENT_7               307511 non-null  int64
 102  FLAG_DOCUMENT_8               307511 non-null  int64
 103  FLAG_DOCUMENT_9               307511 non-null  int64
 104  FLAG_DOCUMENT_10              307511 non-null  int64
 105  FLAG_DOCUMENT_11              307511 non-null  int64
 106  FLAG_DOCUMENT_12              307511 non-null  int64
 107  FLAG_DOCUMENT_13              307511 non-null  int64
 108  FLAG_DOCUMENT_14              307511 non-null  int64
 109  FLAG_DOCUMENT_15              307511 non-null  int64
 110  FLAG_DOCUMENT_16              307511 non-null  int64
```

```
 110   FLAG_DOCUMENT_16              307511 non-null  int64
 111   FLAG_DOCUMENT_17              307511 non-null  int64
 112   FLAG_DOCUMENT_18              307511 non-null  int64
 113   FLAG_DOCUMENT_19              307511 non-null  int64
 114   FLAG_DOCUMENT_20              307511 non-null  int64
 115   FLAG_DOCUMENT_21              307511 non-null  int64
 116   AMT_REQ_CREDIT_BUREAU_HOUR    265992 non-null  float64
 117   AMT_REQ_CREDIT_BUREAU_DAY     265992 non-null  float64
 118   AMT_REQ_CREDIT_BUREAU_WEEK    265992 non-null  float64
 119   AMT_REQ_CREDIT_BUREAU_MON     265992 non-null  float64
 120   AMT_REQ_CREDIT_BUREAU_QRT     265992 non-null  float64
 121   AMT_REQ_CREDIT_BUREAU_YEAR    265992 non-null  float64
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

```
newapp.describe()
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	30
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	2
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	1
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	1
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	2
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	3
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	25

3. preapp data check

```
preapp.head()
```

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREI
0	2030495	271877	Consumer loans	1730.430	17145.0	1714

```
preapp.shape
```

```
(1421211, 37)
```

```
preapp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1421211 entries, 0 to 1421210
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   SK_ID_PREV                   1421211 non-null  int64  
 1   SK_ID_CURR                   1421211 non-null  int64  
 2   NAME_CONTRACT_TYPE           1421211 non-null  object 
 3   AMT_ANNUITY                  1105616 non-null  float64
 4   AMT_APPLICATION              1421211 non-null  float64
 5   AMT_CREDIT                   1421210 non-null  float64
 6   AMT_DOWN_PAYMENT             663145 non-null   float64
 7   AMT_GOODS_PRICE              1094755 non-null  float64
 8   WEEKDAY_APPR_PROCESS_START   1421211 non-null  object 
 9   HOUR_APPR_PROCESS_START      1421211 non-null  int64  
 10  FLAG_LAST_APPL_PER_CONTRACT  1421211 non-null  object 
 11  NFLAG_LAST_APPL_IN_DAY       1421211 non-null  int64  
 12  RATE_DOWN_PAYMENT            663145 non-null   float64
 13  RATE_INTEREST_PRIMARY        5114 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5114 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1421211 non-null  object 
 16  NAME_CONTRACT_STATUS         1421211 non-null  object 
 17  DAYS_DECISION                1421211 non-null  int64  
 18  NAME_PAYMENT_TYPE            1421211 non-null  object 
 19  CODE_REJECT_REASON           1421211 non-null  object 
 20  NAME_TYPE_SUITE              723810 non-null   object 
 21  NAME_CLIENT_TYPE             1421211 non-null  object 
 22  NAME_GOODS_CATEGORY          1421210 non-null  object 
 23  NAME_PORTFOLIO               1421210 non-null  object 
 24  NAME_PRODUCT_TYPE            1421210 non-null  object 
 25  CHANNEL_TYPE                 1421210 non-null  object 
 26  SELLERPLACE_AREA             1421210 non-null  float64
 27  NAME_SELLER_INDUSTRY         1421210 non-null  object 
 28  CNT_PAYMENT                  1105619 non-null  float64
 29  NAME_YIELD_GROUP             1421210 non-null  object 
 30  PRODUCT_COMBINATION          1420917 non-null  object 
 31  DAYS_FIRST_DRAWING           850901 non-null   float64
 32  DAYS_FIRST_DUE               850901 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    850901 non-null   float64
 34  DAYS_LAST_DUE                850901 non-null   float64
 35  DAYS_TERMINATION             850901 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    850901 non-null   float64
dtypes: float64(16), int64(5), object(16)
memory usage: 401.2+ MB
```

```
preapp.describe()
```

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DO
count	1.421211e+06	1.421211e+06	1.105616e+06	1.421211e+06	1.421210e+06	6
mean	1.922488e+06	2.783605e+05	1.589526e+04	1.744340e+05	1.951721e+05	6
std	5.326845e+05	1.028281e+05	1.474704e+04	2.917007e+05	3.174550e+05	2
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00	0.000000e+00	-9
25%	1.460752e+06	1.893380e+05	6.300000e+03	1.890000e+04	2.425950e+04	0
50%	1.922831e+06	2.786900e+05	1.125000e+04	7.082550e+04	8.016300e+04	1
75%	2.383644e+06	3.675400e+05	2.053210e+04	1.800000e+05	2.156400e+05	7
max	2.845382e+06	4.562550e+05	4.180581e+05	6.905160e+06	6.905160e+06	3

4. Data Analysis For newapp Data

4.1 Checking the newapp dataset

```
# Finding the percentage of missing values in all columns
round(newapp.isnull().mean()*100,2).sort_values(ascending = False)
```

```
FLAG_DOCUMENT_8              0.00
NAME_CONTRACT_TYPE           0.00
CODE_GENDER                  0.00
FLAG_OWN_CAR                 0.00
DAYS_LAST_PHONE_CHANGE       0.00
FLAG_DOCUMENT_2              0.00
FLAG_DOCUMENT_3              0.00
FLAG_DOCUMENT_4              0.00
FLAG_DOCUMENT_5              0.00
FLAG_DOCUMENT_6              0.00
FLAG_DOCUMENT_7              0.00
FLAG_DOCUMENT_9              0.00
FLAG_DOCUMENT_21             0.00
FLAG_DOCUMENT_10             0.00
FLAG_DOCUMENT_11             0.00
FLAG_OWN_REALTY              0.00
FLAG_DOCUMENT_13             0.00
FLAG_DOCUMENT_14             0.00
FLAG_DOCUMENT_15             0.00
FLAG_DOCUMENT_16             0.00
FLAG_DOCUMENT_17             0.00
FLAG_DOCUMENT_18             0.00
FLAG_DOCUMENT_19             0.00
FLAG_DOCUMENT_20             0.00
FLAG_DOCUMENT_12             0.00
AMT_CREDIT                   0.00
AMT_INCOME_TOTAL             0.00
FLAG_PHONE                   0.00
```

```
FLAG_PHONE                      0.00
LIVE_CITY_NOT_WORK_CITY         0.00
REG_CITY_NOT_WORK_CITY          0.00
TARGET                          0.00
REG_CITY_NOT_LIVE_CITY          0.00
LIVE_REGION_NOT_WORK_REGION     0.00
REG_REGION_NOT_WORK_REGION      0.00
REG_REGION_NOT_LIVE_REGION      0.00
HOUR_APPR_PROCESS_START         0.00
WEEKDAY_APPR_PROCESS_START      0.00
REGION_RATING_CLIENT_W_CITY     0.00
REGION_RATING_CLIENT            0.00
CNT_FAM_MEMBERS                 0.00
FLAG_EMAIL                      0.00
FLAG_CONT_MOBILE                0.00
ORGANIZATION_TYPE               0.00
FLAG_WORK_PHONE                 0.00
FLAG_EMP_PHONE                  0.00
FLAG_MOBIL                      0.00
DAYS_ID_PUBLISH                 0.00
DAYS_REGISTRATION               0.00
DAYS_EMPLOYED                   0.00
DAYS_BIRTH                      0.00
REGION_POPULATION_RELATIVE      0.00
NAME_HOUSING_TYPE               0.00
NAME_FAMILY_STATUS              0.00
NAME_EDUCATION_TYPE             0.00
NAME_INCOME_TYPE                0.00
AMT_ANNUITY                     0.00
SK_ID_CURR                      0.00
dtype: float64
```

```
# Removing all the columns with more than 50% nulls values/Keeping all of them with <= 50%
newapp = newapp.loc[:,newapp.isnull().mean()<=0.5]
newapp.shape
```

```
(307511, 81)
```

```
#Selecting columns with less or equal to than 13% null vallues
list(newapp.columns[(newapp.isnull().mean()<=0.13) & (newapp.isnull().mean()>0)])

#We will check those columns for possible imputation
```

```
['AMT_ANNUITY',
 'AMT_GOODS_PRICE',
 'NAME_TYPE_SUITE',
 'CNT_FAM_MEMBERS',
 'EXT_SOURCE_2',
 'OBS_30_CNT_SOCIAL_CIRCLE',
 'DEF_30_CNT_SOCIAL_CIRCLE',
 'OBS_60_CNT_SOCIAL_CIRCLE',
 'DEF_60_CNT_SOCIAL_CIRCLE',
 'DAYS_LAST_PHONE_CHANGE']
```

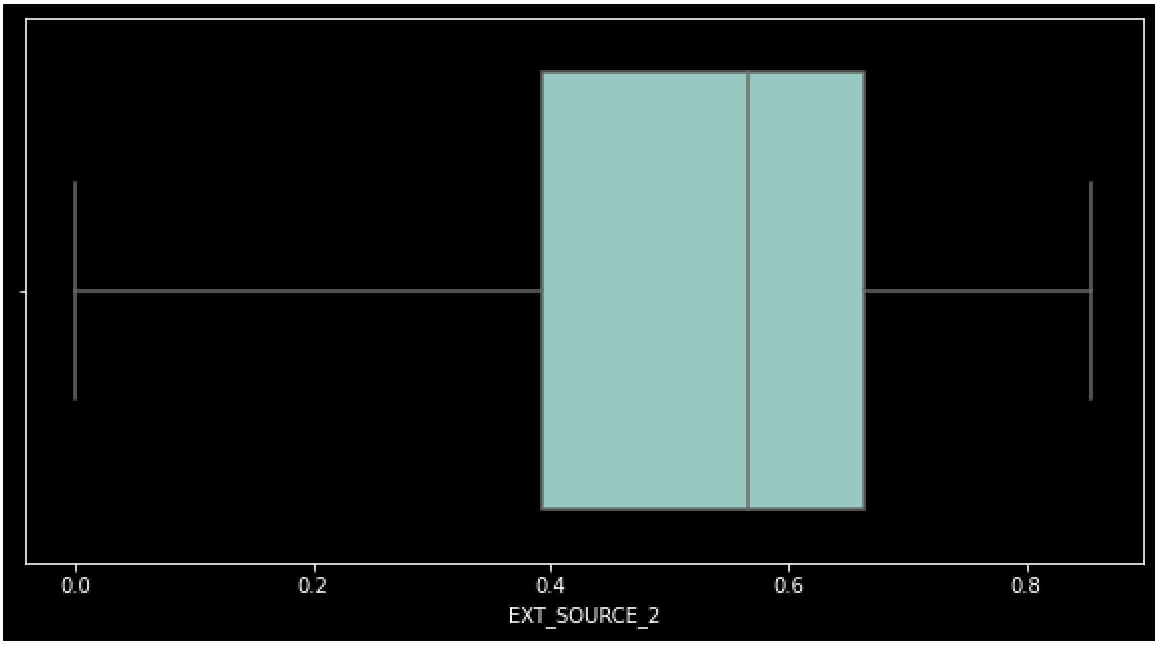
4.2 Checking for values to impute in columns

4.2.1. EXT_SOURCE_2 imputation

```
newapp['EXT_SOURCE_2'].value_counts()
```

```
0.285898    721
0.262258    417
0.265256    343
0.159679    322
0.265312    306
           ...
0.004725      1
0.257313      1
0.282030      1
0.181540      1
0.267834      1
Name: EXT_SOURCE_2, Length: 119831, dtype: int64
```

```
# EXT_SOURCE_2 is a continuous variable. So checking for outliers
plt.style.use('dark_background')
plt.figure(figsize=[10,5])
sns.boxplot(newapp['EXT_SOURCE_2'])
plt.show()
```




```
# Since EXT_SOURCE_2 has no outlier, we can choose mean to impute the column
imputVAL = round(newapp['EXT_SOURCE_2'].mean(),2)
print(f'Since EXT_SOURCE_2 has no outlier, the column can be imputed using the mean of the
```

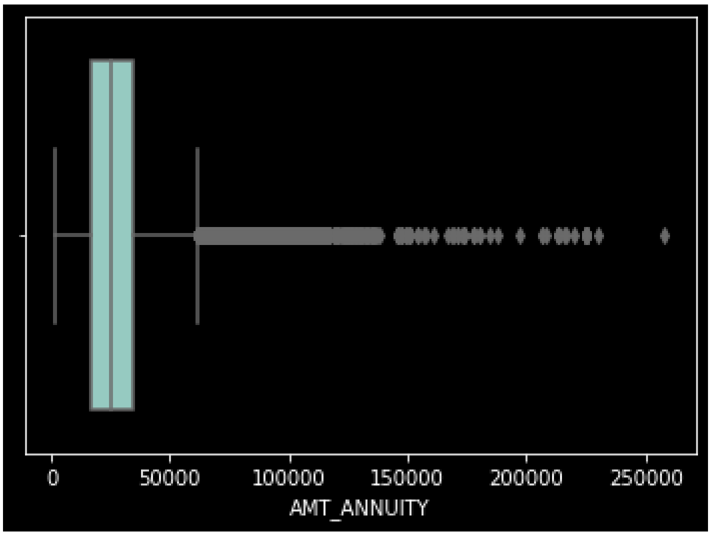
```
Since EXT_SOURCE_2 has no outlier, the column can be imputed using the mean of the co
```

4.2.2. OCCUPATION_TYPE imputation

```
newapp['AMT_ANNUITY'].value_counts()
```

```
9000.0      6385
13500.0     5514
6750.0      2279
10125.0     2035
37800.0     1602
            ...
79902.0        1
106969.5       1
60885.0        1
59661.0        1
77809.5        1
Name: AMT_ANNUITY, Length: 13672, dtype: int64
```

```
# Since AMT_ANNUITY is a continuous variable. So checking for outliers
sns.boxplot(newapp['AMT_ANNUITY'])
plt.show()
```




```
imputVAL = round(newapp['AMT_ANNUITY'].median(),2)
print(f'Since AMT_ANNUITY has outliers, the column can be imputed using the median of the
```

```
Since AMT_ANNUITY has outliers, the column can be imputed using the median of the cou
```

4.2.3. NAME_TYPE_SUITE imputation

```
newapp['NAME_TYPE_SUITE'].value_counts()
```

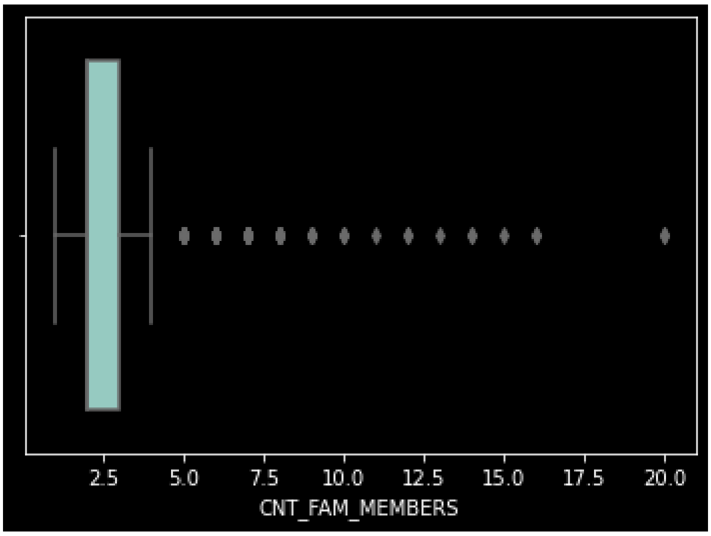
```
Unaccompanied      248526
Family              40149
Spouse, partner     11370
Children             3267
Other_B              1770
Other_A               866
Group of people       271
Name: NAME_TYPE_SUITE, dtype: int64
```

```
imputVAL = newapp['NAME_TYPE_SUITE'].mode()
print(f'Clearly the column NAME_TYPE_SUITE is a categorical column. So this column can be
```

```
Clearly the column NAME_TYPE_SUITE is a categorical column. So this column can be imp
```

4.2.4. CNT_FAM_MEMBERS imputation

```
# Since this is count of family members, this is a continuous variable and we can impute t
sns.boxplot(newapp['CNT_FAM_MEMBERS'])
plt.show()
```




```
imputVAL = round(newapp['CNT_FAM_MEMBERS'].median(),2)
print(f'Since CNT_FAM_MEMBERS has outliers, the column can be imputed using the median of
```

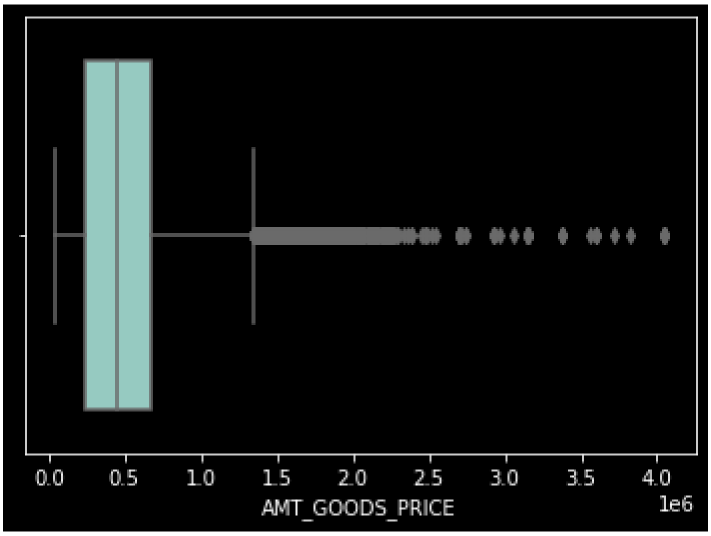
```
Since CNT_FAM_MEMBERS has outliers, the column can be imputed using the median of the
```

4.2.5. AMT_GOODS_PRICE imputation

```
newapp['AMT_GOODS_PRICE'].value_counts()
```

```
450000.0     26022
225000.0     25282
675000.0     24962
900000.0     15416
270000.0     11428
             ...
1265751.0        1
503266.5         1
810778.5         1
666090.0         1
743863.5         1
Name: AMT_GOODS_PRICE, Length: 1002, dtype: int64
```

```
# AMT_GOODS_PRICE is a continuous variable. So checking for outliers
sns.boxplot(newapp['AMT_GOODS_PRICE'])
plt.show()
```




```
# Since this is a continuous variable with outliers we can impute column using median valu
imputVAL = round(newapp['AMT_GOODS_PRICE'].median(),2)
print(f'Since AMT_GOODS_PRICE has outliers, the column can be imputed using the median of
```

```
Since AMT_GOODS_PRICE has outliers, the column can be imputed using the median of the
```

4.3 Check datatypes of columns and modify them appropriately

```
#Checking the float type columns
newapp.select_dtypes(include='float64').columns
```

```
Index(['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'REGION_P(
```

```
#Converting these count columns to int64
ColumnToConvert = ['OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOC
                   'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE', 'AMT_REQ_CREDIT_B
                   'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CRE
                   'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR']
newapp.loc[:,ColumnToConvert]=newapp.loc[:,ColumnToConvert].apply(lambda col: col.astype('
```

```
#Checking the object type columns
ColumnToConvert = list(newapp.select_dtypes(include='object').columns)
```

```
newapp.loc[:,ColumnToConvert]=newapp.loc[:,ColumnToConvert].apply(lambda col: col.astype('
```

```
newapp.head()
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALT
0	100002	1	Cash loans	M	N	`
1	100003	0	Cash loans	F	N	I
2	100004	0	Revolving loans	M	Y	`
3	100006	0	Cash loans	F	N	`
4	100007	0	Cash loans	M	N	`

```
#Making Gender more readable
newapp['CODE_GENDER'].value_counts()
```

```
F      202448
M      105059
XNA         4
Name: CODE_GENDER, dtype: int64
```

```
# Dropping the Gender = XNA from the data set as there is not enough data regarding that
newapp = newapp[newapp['CODE_GENDER']!='XNA']
newapp['CODE_GENDER'].replace(['M','F'],['Male','Female'],inplace=True)
```

4.4 Binning variables for analysis

```
newapp['AMT_INCOME_TOTAL'].quantile([0,0.1,0.3,0.6,0.8,1])
```

```
0.0        25650.0
0.1        81000.0
0.3       112500.0
0.6       162000.0
0.8       225000.0
1.0    117000000.0
Name: AMT_INCOME_TOTAL, dtype: float64
```

```
#Creating A new categorical variable based on income total
newapp['INCOME_GROUP']=pd.qcut(newapp['AMT_INCOME_TOTAL'],
                                       q=[0,0.1,0.3,0.6,0.8,1],
                                       labels=['VeryLow','Low','Medium','High','VeryHigh']
```

```
#Binning DAYS_BIRTH
abs(newapp['DAYS_BIRTH']).quantile([0,0.1,0.3,0.6,0.8,1])
```

```
0.0     7489.0
0.1    10284.6
```

```
0.3    13140.0
0.6    17220.0
0.8    20474.0
1.0    25229.0
Name: DAYS_BIRTH, dtype: float64
```

```
#Creating a column AGE using DAYS_BIRTH
newapp['AGE']=abs(newapp['DAYS_BIRTH'])//365.25
```

```
newapp['AGE'].describe()
```

```
count    307507.000000
mean         43.405223
std          11.945763
min          20.000000
25%          33.000000
50%          43.000000
75%          53.000000
max          69.000000
Name: AGE, dtype: float64
```

```
## Since the AGE varies from 20 to 69, we can create bins of 5 years starting from 20 to 7
newapp['AGE_GROUP'] = pd.cut(newapp['AGE'],bins=np.arange(20,71,5))
```

```
## Adding one more column that will be used for analysis later
newapp['CREDIT_INCOME_RATIO']=round((newapp['AMT_CREDIT']/newapp['AMT_INCOME_TOTAL']))
```

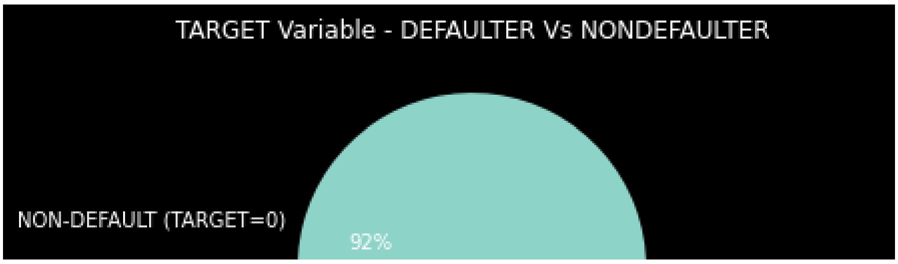
```
### Getting the percentage of social circle who defaulted
newapp['SOCIAL_CIRCLE_30_DAYS_DEF_PERC']=newapp['DEF_30_CNT_SOCIAL_CIRCLE']/newapp['OBS_30
newapp['SOCIAL_CIRCLE_60_DAYS_DEF_PERC']=newapp['DEF_60_CNT_SOCIAL_CIRCLE']/newapp['OBS_60
```

4.5 - Checking for imbalance in Target

```
newapp['TARGET'].value_counts(normalize=True)*100
```

```
0    91.927013
1     8.072987
Name: TARGET, dtype: float64
```

```
plt.pie(newapp['TARGET'].value_counts(normalize=True)*100,labels=['NON-DEFAULT (TARGET=0)'
plt.title('TARGET Variable - DEFAULTER Vs NONDEFAULTER')
plt.show()
```



> Its clear that there is an imbalance between people who defaulted and who didn't default. More than 92% of people didn't default as opposed to 8% who defaulted.

```
# From the remaining columns about 30 are selected based on their description and relevanc
#for further analysis
FinalColumns = ['SK_ID_CURR','TARGET','CODE_GENDER','FLAG_OWN_CAR','FLAG_OWN_REALTY','INCO
'CREDIT_INCOME_RATIO','NAME_INCOME_TYPE','NAME_EDUCATION_TYPE','NAME_FAMILY_STATUS','NAME_
'DAYS_REGISTRATION','FLAG_EMAIL','OCCUPATION_TYPE',
'CNT_FAM_MEMBERS','REGION_RATING_CLIENT_W_CITY','ORGANIZATION_TYPE','SOCIAL_CIRCLE_30_DAYS_
'SOCIAL_CIRCLE_60_DAYS_DEF_PERC','AMT_REQ_CREDIT_BUREAU_DAY',
'AMT_REQ_CREDIT_BUREAU_MON','AMT_REQ_CREDIT_BUREAU_QRT','NAME_CONTRACT_TYPE','AMT_ANNUITY'
```

```
newapp_Final=newapp[FinalColumns]
```

```
newapp_Final.shape
```

    (307507, 30)

4.6 - Splitting the dataframe into two separate dfs

```
NEWAPP0=newapp_Final[newapp_Final.TARGET==0]    # Dataframe with all the data related to n
NEWAPP1=newapp_Final[newapp_Final.TARGET==1]    # Dataframe with all the data related to d
```

4.7 Univariate Analysis

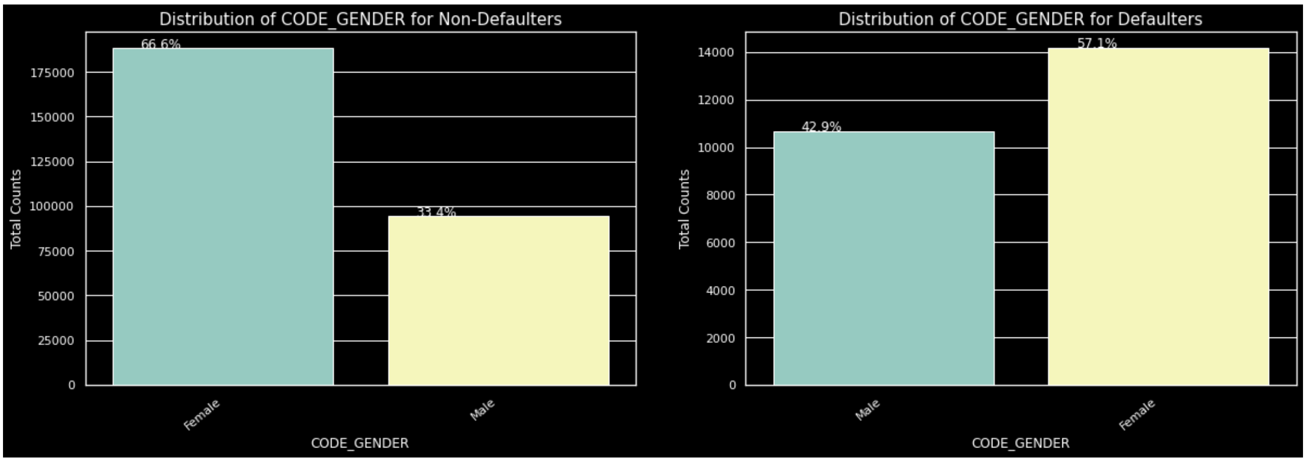
Function to plot the univariate categorical variables

```
# function to count plot for categorical variables
def plotuninewapp(var):
    plt.style.use('dark_background')
    sns.despine
    fig,(ax1,ax2) = plt.subplots(1,2,figsize=(20,6))

    sns.countplot(x=var, data=NEWAPP0,ax=ax1)
    ax1.set_ylabel('Total Counts')
    ax1.set_title(f'Distribution of {var} for Non-Defaulters',fontsize=15)
    ax1.set_xticklabels(ax1.get_xticklabels(), rotation=40, ha="right")
```

```
    # Adding the normalized percentage for easier comparision between defaulter and non-de
    for p in ax1.patches:
        ax1.annotate('{:.1f}%'.format((p.get_height()/len(NEWAPP0))*100), (p.get_x()+0.1,

    sns.countplot(x=var, data=NEWAPP1,ax=ax2)
    ax2.set_ylabel('Total Counts')
    ax2.set_title(f'Distribution of {var} for Defaulters',fontsize=15)
    ax2.set_xticklabels(ax2.get_xticklabels(), rotation=40, ha="right")

    # Adding the normalized percentage for easier comparision between defaulter and non-de
    for p in ax2.patches:
        ax2.annotate('{:.1f}%'.format((p.get_height()/len(NEWAPP1))*100), (p.get_x()+0.1,

    plt.show()
```

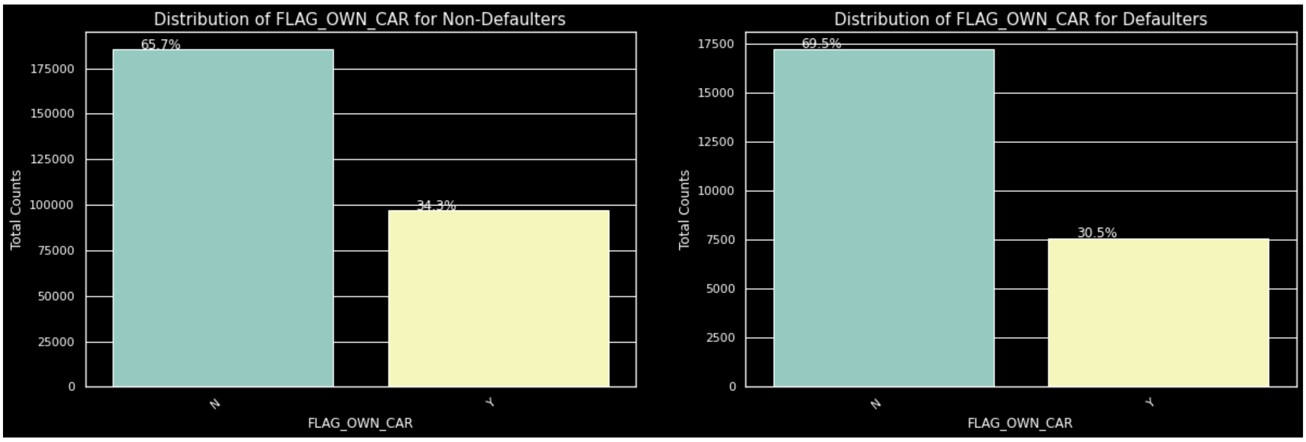
4.7.1 Univariate Categorical Ordered Analysis

```
plotuninewapp('CODE_GENDER')
```



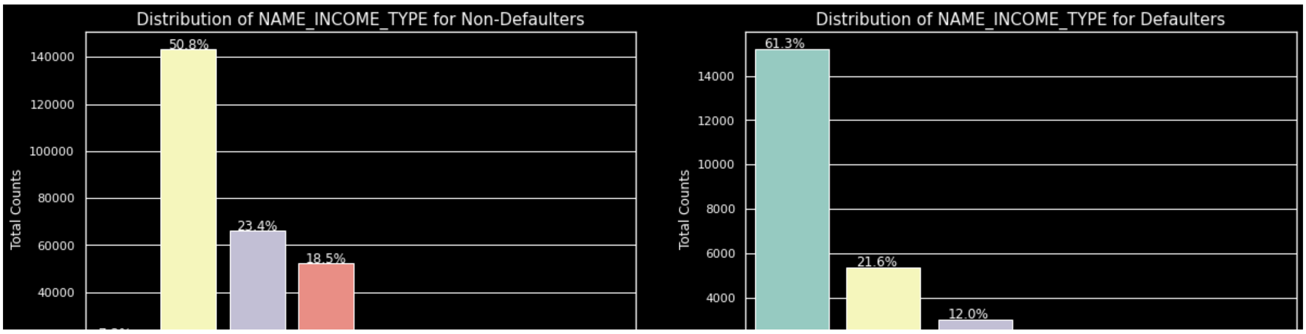
> We can see that Female contribute 67% to the non-defaulters while 57% to the defaulters. We can conclude that
> We see more female applying for loans than males and hence the more number of female defaulters as well.
> **But the rate of defaulting of FEMALE is much lower compared to their MALE counterparts.**

```
plotuninewapp('FLAG_OWN_CAR')
```



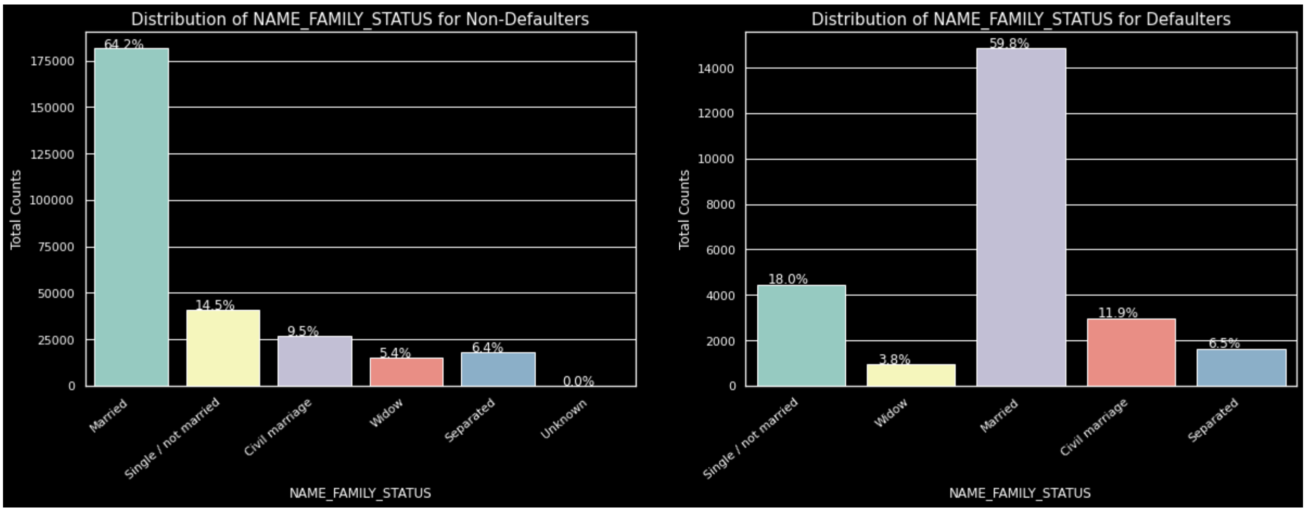
> We can see that people with cars contribute 65.7% to the non-defaulters while 69.5% to the defaulters. We can conclude that
> While people who have car default more often, the reason could be there are simply more people without cars
> Looking at the percentages in both the charts, we can conclude that the rate of default of people having car is low compared to people who don't.

```
plotuninewapp('NAME_INCOME_TYPE')
```



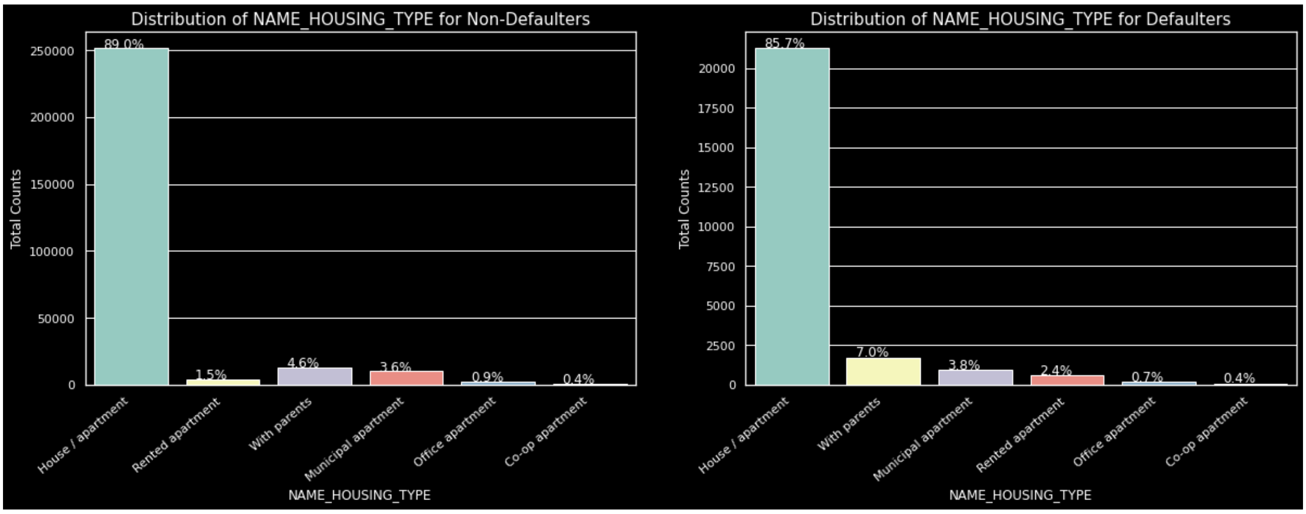
> We can notice that the students don't default. The reason could be they are not required to pay during the time they are students.
> We can also see that the BusinessMen never default.
> Most of the loans are distributed to working class people
> We also see that working class people contribute 51% to non defaulters while they contribute to 61% of the defaulters. Clearly, the chances of defaulting are more in their case.

```
plotuninewapp('NAME_FAMILY_STATUS')
```



> Married people tend to apply for more loans comparatively.
> But from the graph we see that Single/non Married people contribute 14.5% to Non Defaulters and 18% to the defaulters. So there is more risk associated with them.

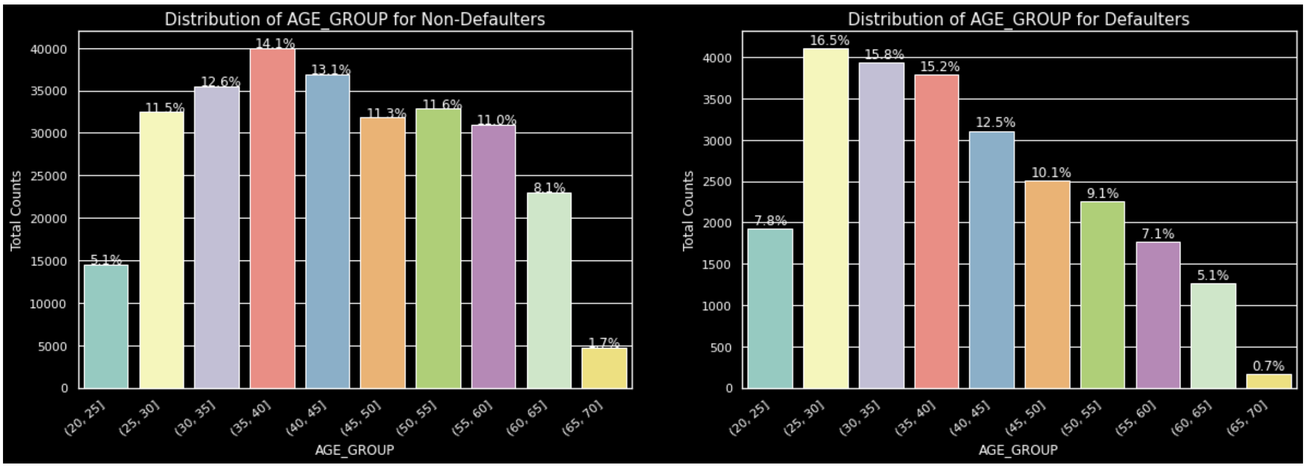
```
plotuninewapp('NAME_HOUSING_TYPE')
```



> It is clear from the graph that people who have House/Appartment, tend to apply for more loans.
> People living with parents tend to default more often when compared with others.The reason could be their living expenses are more due to their parents living with them.

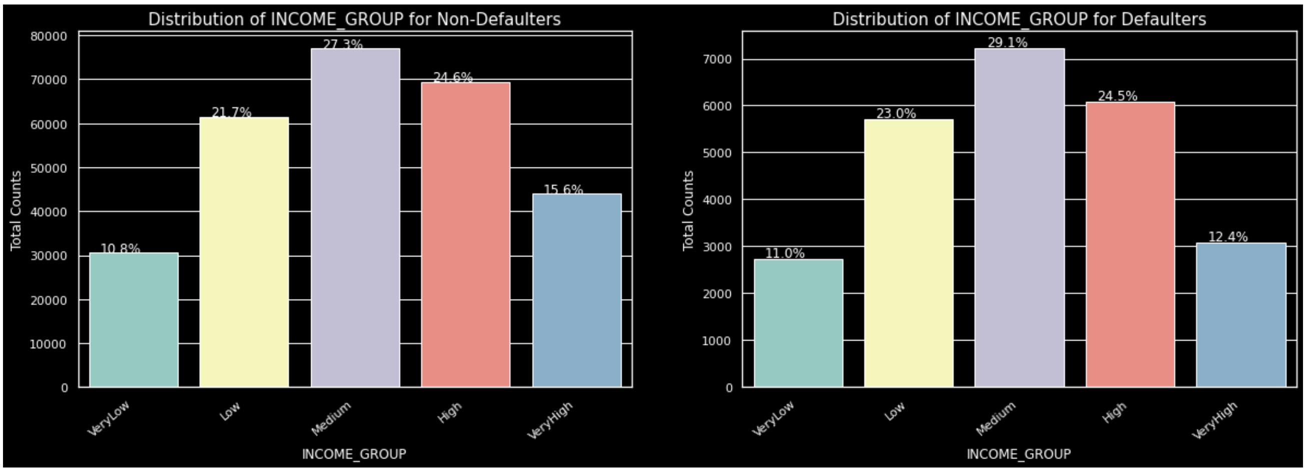
4.7.2 Univariate Categorical Ordered Analysis

```
plotuninewapp('AGE_GROUP')
```



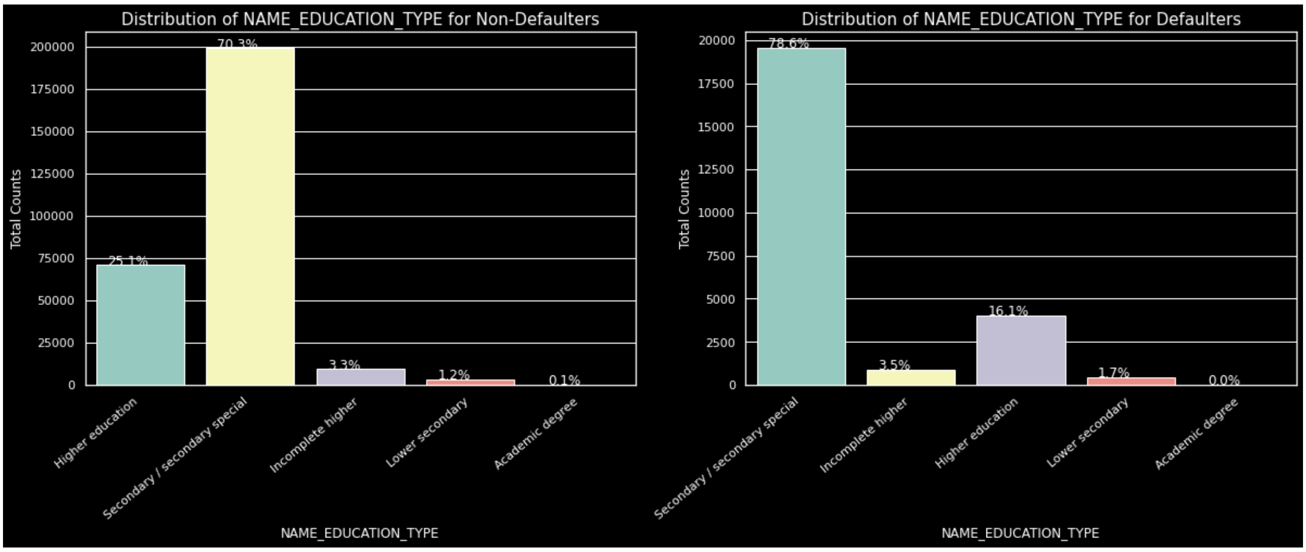
> We see that (25,30] age group tend to default more often. So they are the riskiest people to loan to.
> With increasing age group, people tend to default less starting from the age 25. One of the reasons could be they get employed around that age and with increasing age, their salary also increases.

```
plotuninewapp('INCOME_GROUP')
```



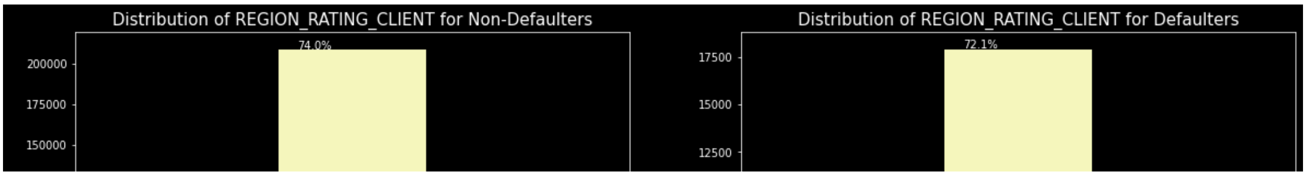
> The Very High income group tend to default less often. They contribute 12.4% to the total number of defaulters, while they contribute 15.6% to the Non-Defaulters.

```
plotuninewapp('NAME_EDUCATION_TYPE')
```



> Almost all of the Education categories are equally likely to default except for the higher educated ones who are less likely to default and secondary educated people are more likely to default

```
plotuninewapp('REGION_RATING_CLIENT')
```

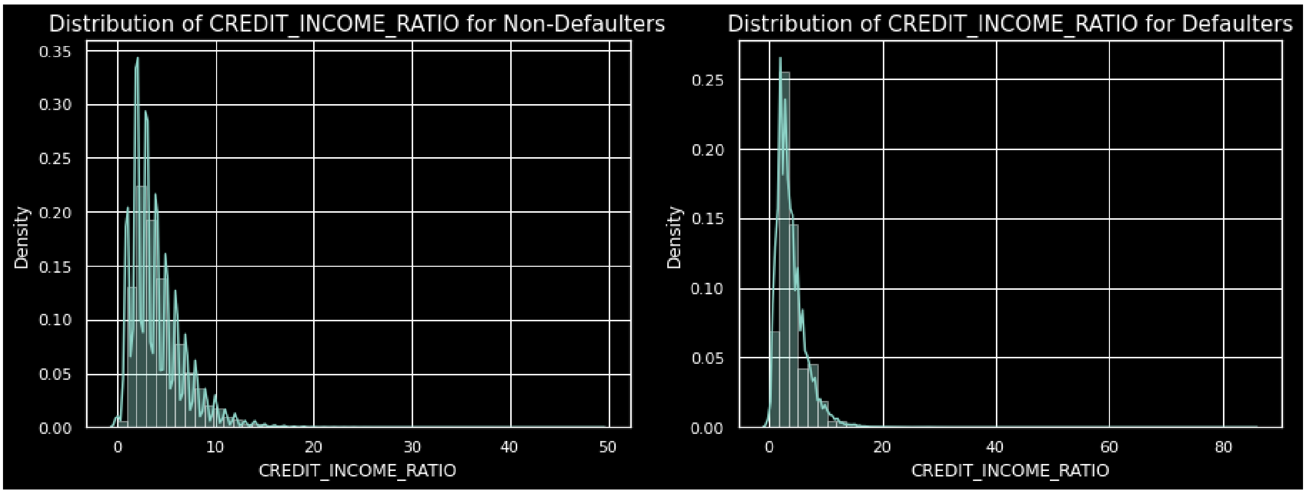


> More people from second tier regions tend to apply for loans.
> We can infer that people living in better areas(Rating 3) tend contribute more to the defaulters by their weightage.
> People living in 1 rated areas

4.7.3 Univariate continuous variable analysis

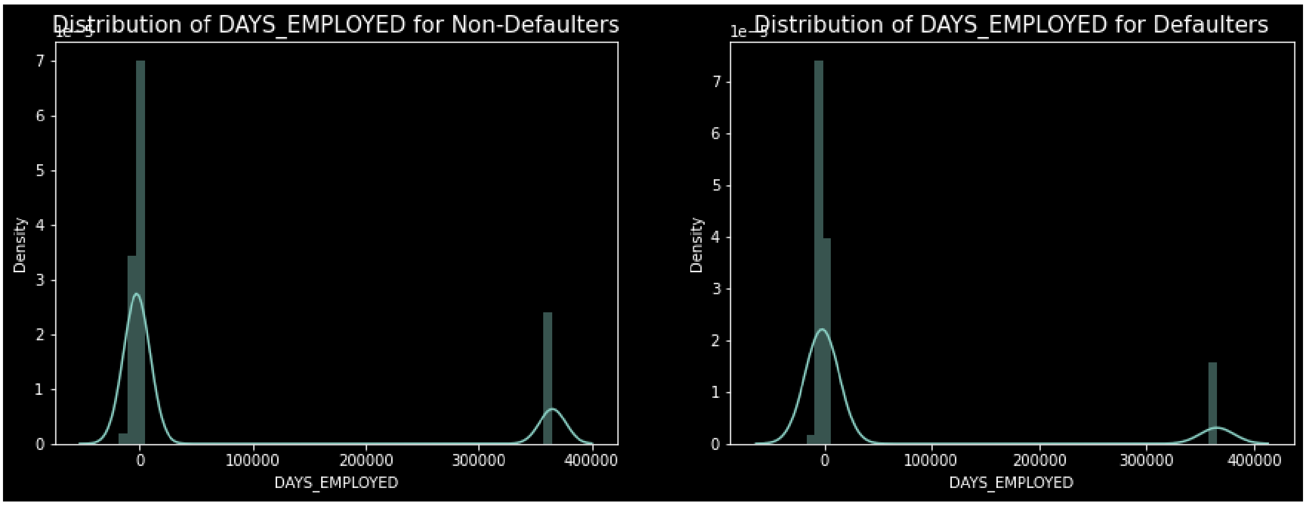
```
# function to dist plot for continuous variables
def plotunidist(var):

    plt.style.use('dark_background')
    sns.despine
    fig,(ax1,ax2) = plt.subplots(1,2,figsize=(15,5))

    sns.distplot(a=NEWAPP0[var],ax=ax1)

    ax1.set_title(f'Distribution of {var} for Non-Defaulters',fontsize=15)

    sns.distplot(a=NEWAPP1[var],ax=ax2)
    ax2.set_title(f'Distribution of {var} for Defaulters',fontsize=15)

    plt.show()
```

```
plotunidist('CREDIT_INCOME_RATIO')
```



> Credit income ratio the ratio of AMT_CREDIT/AMT_INCOME_TOTAL.
> Although there doesn't seem to be a clear distiguish between the group which defaulted vs the group which didn't when compared using the ratio, we can see that when the CREDIT_INCOME_RATIO is more than 50, people default.

```
plotunidist('DAYS_EMPLOYED')
```

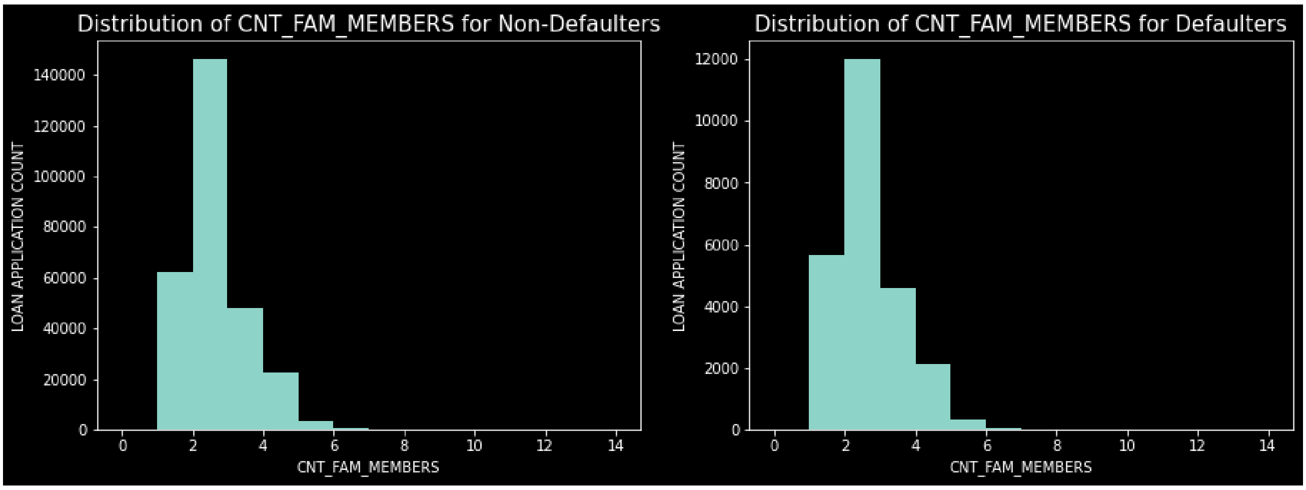



```
NEWAPP1['CNT_FAM_MEMBERS'].value_counts()
```

```
2.0     12009
1.0      5675
3.0      4608
4.0      2136
5.0       327
6.0        55
7.0         6
8.0         6
10.0        1
13.0        1
11.0        1
Name: CNT_FAM_MEMBERS, dtype: int64
```

```
plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)
NEWAPP0['CNT_FAM_MEMBERS'].plot.hist(bins=range(15))
plt.title('Distribution of CNT_FAM_MEMBERS for Non-Defaulters',fontsize=15)
plt.xlabel('CNT_FAM_MEMBERS')
plt.ylabel('LOAN APPLICATION COUNT')

plt.subplot(1, 2, 2)
NEWAPP1['CNT_FAM_MEMBERS'].plot.hist(bins=range(15))
plt.title(f'Distribution of CNT_FAM_MEMBERS for Defaulters',fontsize=15)
```

```
plt.xlabel('CNT_FAM_MEMBERS')
plt.ylabel('LOAN APPLICATION COUNT')

plt.show()
```



> We can see that a family of 3 applies loan more often than the other families

4.8 Getting the top 10 correlation of the selected columns

```
#Getting the top 10 correlation in NEWAPP0
corr=NEWAPP0.corr()
corr_df = corr.where(np.triu(np.ones(corr.shape),k=1).astype(np.bool)).unstack().reset_ind
corr_df.columns=['Column1','Column2','Correlation']
corr_df.dropna(subset=['Correlation'],inplace=True)
corr_df['Abs_Correlation']=corr_df['Correlation'].abs()
corr_df = corr_df.sort_values(by=['Abs_Correlation'], ascending=False)
corr_df.head(10)
```

	Column1	Column2	Correla
308	AMT_GOODS_PRICE	AMT_CREDIT	0.98

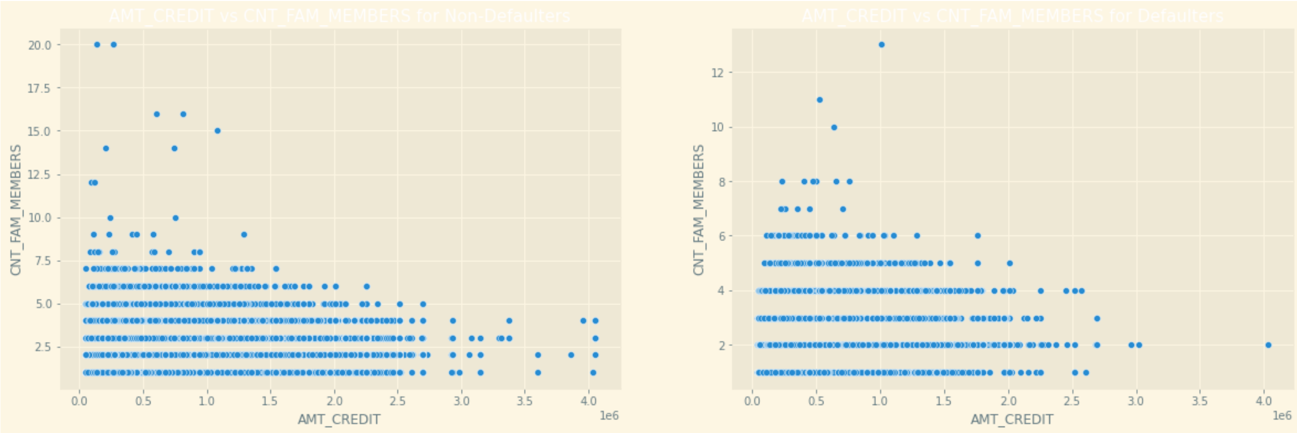
```
#Getting the top 10 correlation NEWAPP1
corr=NEWAPP1.corr()
corr_df = corr.where(np.triu(np.ones(corr.shape),k=1).astype(np.bool)).unstack().reset_ind
corr_df.columns=['Column1','Column2','Correlation']
corr_df.dropna(subset=['Correlation'],inplace=True)
corr_df['Abs_Correlation']=corr_df['Correlation'].abs()
corr_df = corr_df.sort_values(by=['Abs_Correlation'], ascending=False)
corr_df.head(10)
```

	Column1	Column2	Correla
308	AMT_GOODS_PRICE	AMT_CREDIT	0.98
297	REGION_RATING_CLIENT	REGION_RATING_CLIENT_W_CITY	0.95
208	SOCIAL_CIRCLE_60_DAYS_DEF_PERC	SOCIAL_CIRCLE_30_DAYS_DEF_PERC	0.87
321	AMT_GOODS_PRICE	AMT_ANNUITY	0.75
272	AMT_ANNUITY	AMT_CREDIT	0.75
74	CREDIT_INCOME_RATIO	AMT_CREDIT	0.63
310	AMT_GOODS_PRICE	CREDIT_INCOME_RATIO	0.62
274	AMT_ANNUITY	CREDIT_INCOME_RATIO	0.38
113	DAYS_REGISTRATION	DAYS_EMPLOYED	-0.18
149	CNT_FAM_MEMBERS	DAYS_EMPLOYED	-0.18

4.9 Bivariate Analysis of numerical variables

```
# function for scatter plot for continuous variables
def plotbivar(var1,var2):

    plt.style.use('Solarize_Light2')
    sns.despine
    fig,(ax1,ax2) = plt.subplots(1,2,figsize=(20,6))

    sns.scatterplot(x=var1, y=var2,data=NEWAPP0,ax=ax1)
    ax1.set_xlabel(var1)
    ax1.set_ylabel(var2)
    ax1.set_title(f'{var1} vs {var2} for Non-Defaulters',fontsize=15)

    sns.scatterplot(x=var1, y=var2,data=NEWAPP1,ax=ax2)
    ax2.set_xlabel(var1)
    ax2.set_ylabel(var2)
    ax2.set_title(f'{var1} vs {var2} for Defaulters',fontsize=15)

    plt.show()
```

```
plotbivar('AMT_CREDIT','CNT_FAM_MEMBERS')
```



> We can see that the density in the lower left corner is similar in both the case, so the people are equally likely to default if the family is small and the AMT_CREDIT is low. We can observe that larger families and people with larger AMT_CREDIT default less often

```
plotbivar('AMT_GOODS_PRICE','AMT_CREDIT')
```

5. Data Analysis For Previous Application Data

5.1 Doing some more routine check

```
preapp.head(3)
```

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREI
0	2030495	271877	Consumer loans	1730.430	17145.0	1714
1	2802425	108129	Cash loans	25188.615	607500.0	67967
2	2523466	122040	Cash loans	15060.735	112500.0	13644

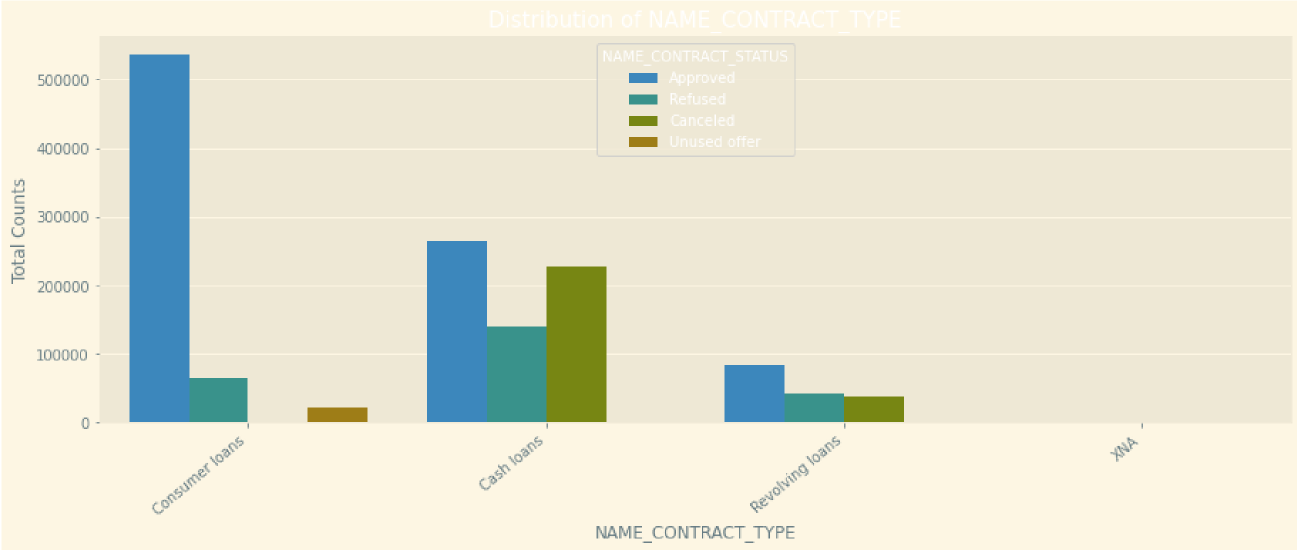
```
# Removing all the columns with more than 50% of null values
preapp = preapp.loc[:,preapp.isnull().mean()<=0.5]
preapp.shape
```

```
(1421211, 33)
```

5.2 Univariate analysis

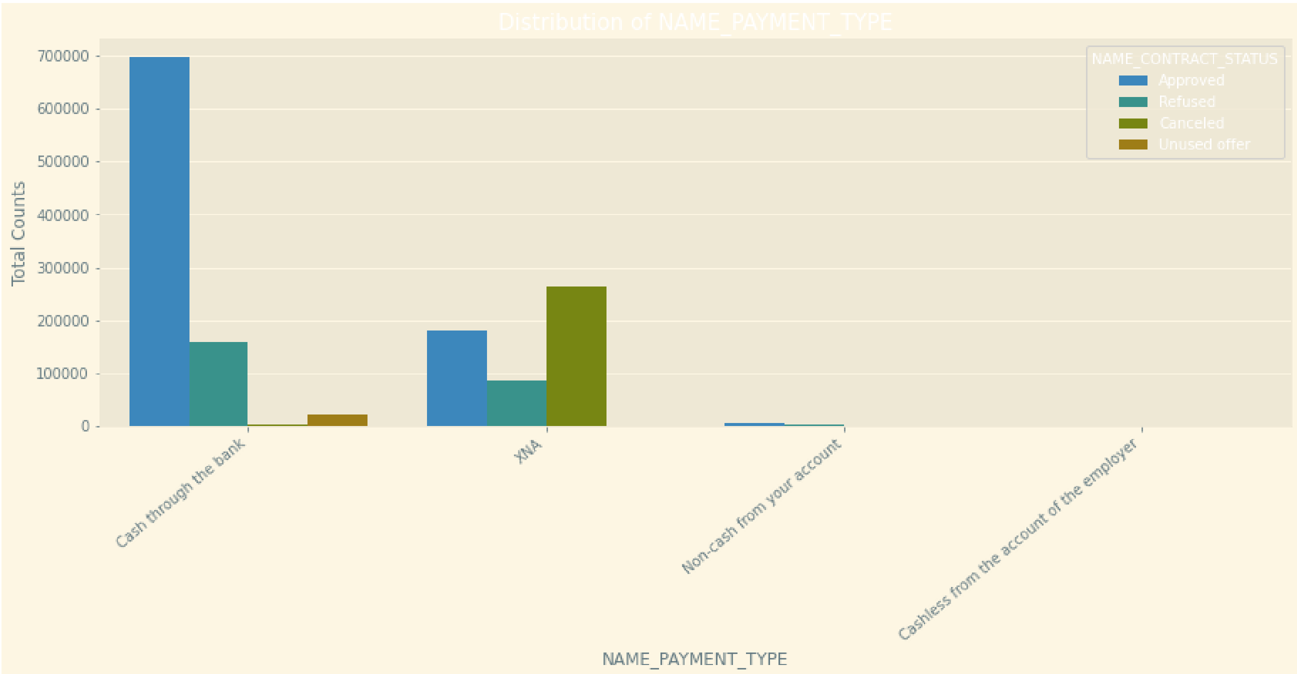
```
# function to count plot for categorical variables
def plot_uni(var):

    plt.style.use('Solarize_Light2')
    sns.despine
    fig,ax = plt.subplots(1,1,figsize=(15,5))

    sns.countplot(x=var, data=preapp,ax=ax,hue='NAME_CONTRACT_STATUS')
    ax.set_ylabel('Total Counts')
    ax.set_title(f'Distribution of {var}',fontsize=15)
    ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")

    plt.show()
```

```
plot_uni('NAME_CONTRACT_TYPE')
```



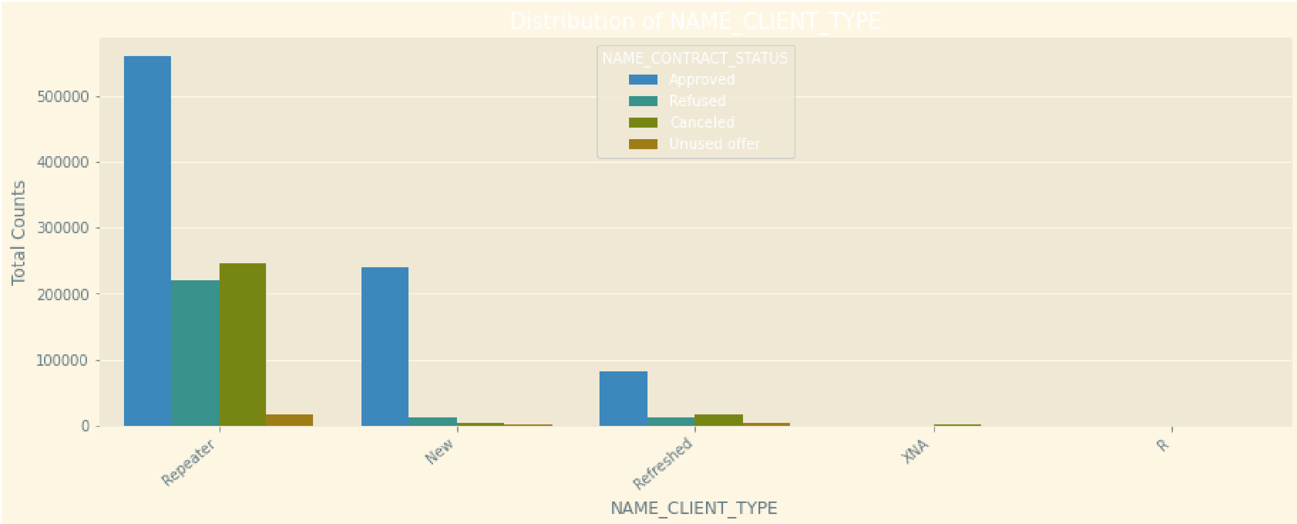
> From the above chart, we can infer that, most of the applications are for 'Cash loan' and 'Consumer loan'. Although the cash loans are refused more often than others.

```
plot_uni('NAME_PAYMENT_TYPE')
```



> From the above chart, we can infer that most of the clients chose to repay the loan using the 'Cash through the bank' option
> We can also see that 'Non-Cash from your account' & 'Cashless from the account of the employee' options are not at all popular in terms of loan repayment amongst the customers.

```
plot_uni('NAME_CLIENT_TYPE')
```



> Most of the loan applications are from repeat customers, out of the total applications 70% of customers are repeaters. They also get refused most often.

5.3 Checking the correlation in the preapp dataset

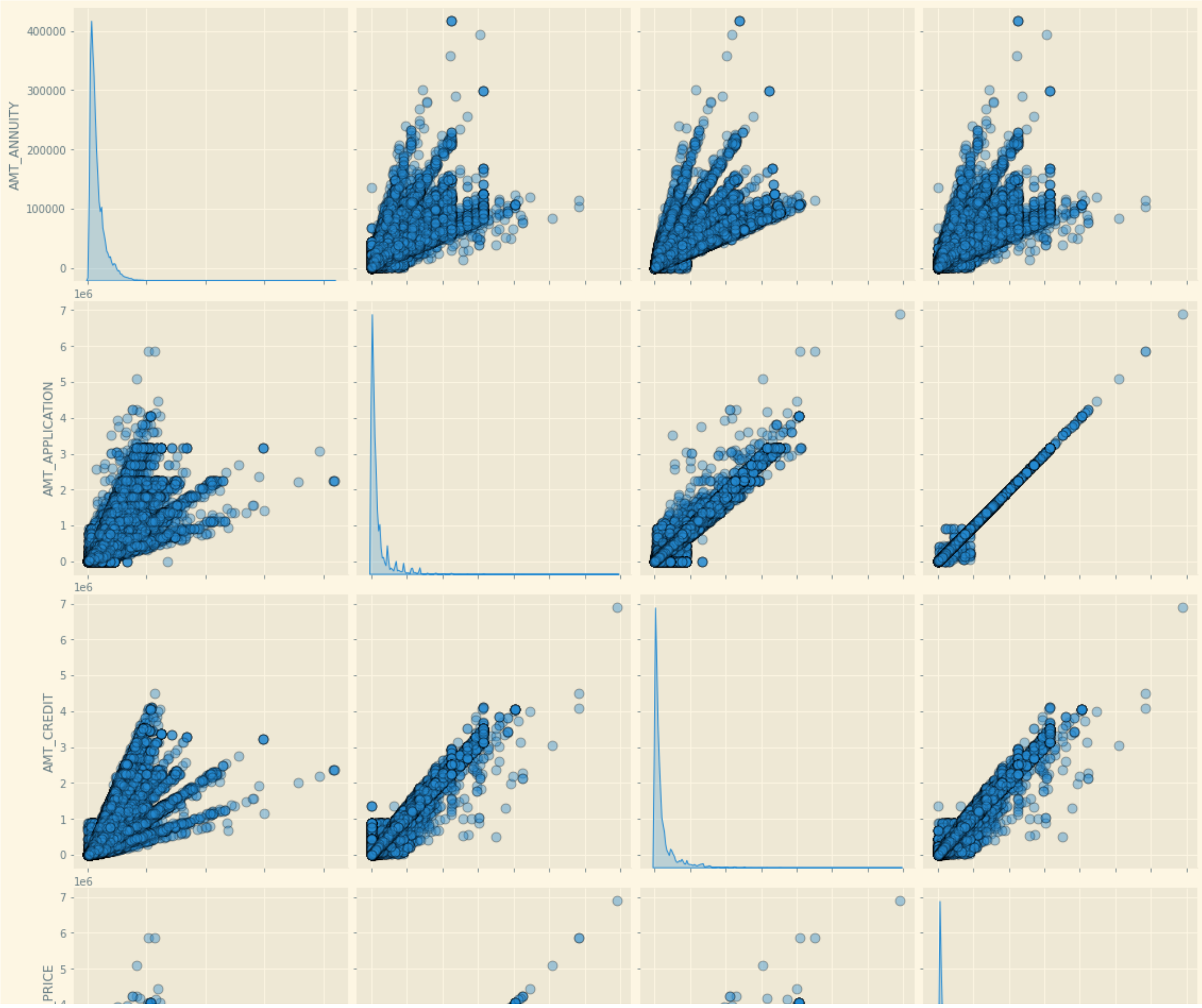
```
#Getting the top 10 correlation preapp
corr=preapp.corr()
corr_df = corr.where(np.triu(np.ones(corr.shape),k=1).astype(np.bool)).unstack().reset_ind
corr_df.columns=['Column1','Column2','Correlation']
corr_df.dropna(subset=['Correlation'],inplace=True)
corr_df['Abs_Correlation']=corr_df['Correlation'].abs()
corr_df = corr_df.sort_values(by=['Abs_Correlation'], ascending=False)
corr_df.head(10)
```

	Column1	Column2	Correlation	Abs_Correlat
88	AMT_GOODS_PRICE	AMT_APPLICATION	0.999889	0.999
89	AMT_GOODS_PRICE	AMT_CREDIT	0.993028	0.993
71	AMT_CREDIT	AMT_APPLICATION	0.975717	0.975
269	DAYS_TERMINATION	DAYS_LAST_DUE	0.928359	0.928
87	AMT_GOODS_PRICE	AMT_ANNUITY	0.820672	0.820
70	AMT_CREDIT	AMT_ANNUITY	0.816175	0.816
232	DAYS_LAST_DUE_1ST_VERSION	DAYS_FIRST_DRAWING	-0.803300	0.803

5.4 Using pairplot to perform bivariate analysis on numerical columns

```
#plotting the relation between correlated highly corelated numeric vriables
plt.figure(figsize=[20,8])
sns.pairplot(preapp[['AMT_ANNUITY','AMT_APPLICATION','AMT_CREDIT','AMT_GOODS_PRICE','NAME_
             diag_kind = 'kde',
             plot_kws = {'alpha': 0.4, 's': 80, 'edgecolor': 'k'},
             size = 4)
plt.show()
```

<Figure size 1440x576 with 0 Axes>



1. Annuity of previous application has a very high and positive influence over: (Increase of annuity increases below factors)
   (1) How much credit did client asked on the previous application
   (2)Final credit amount on the previous application that was approved by the bank
   (3) Goods price of good that client asked for on the previous application.

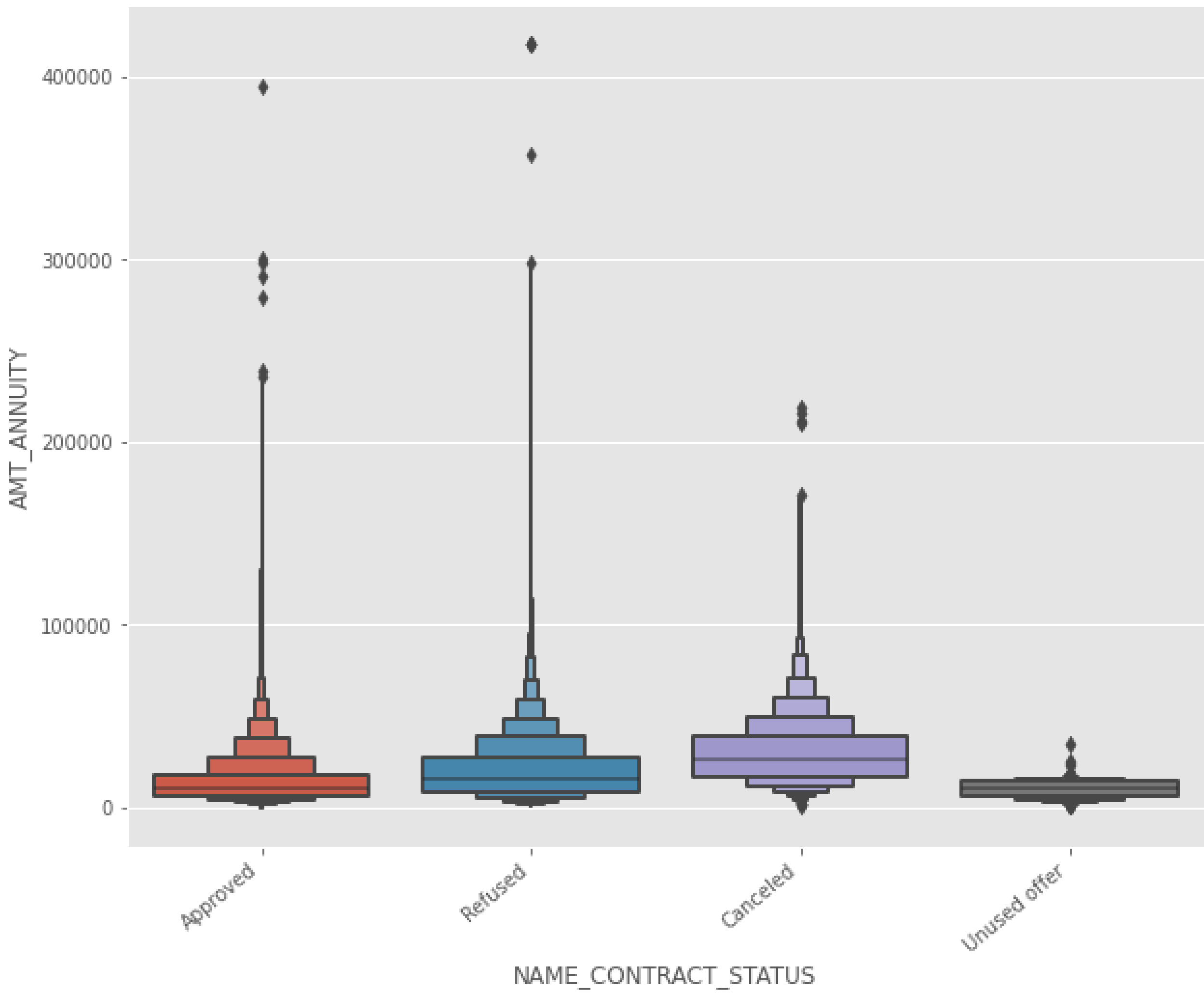
2. For how much credit did client ask on the previous application is highly influenced by the Goods price of good that client has asked for on the previous application

3. Final credit amount disbursed to the customer previously, after approval is highly influence by the application amount and also the goods price of good that client asked for on the previous application.

5.5 Using box plot to do some more bivariate analysis on categorical vs numeric columns

```
#by variant analysis function
def plot_by_cat_num(cat, num):

    plt.style.use('ggplot')
    sns.despine
    fig,ax = plt.subplots(1,1,figsize=(10,8))

    sns.boxenplot(x=cat,y = num, data=preapp)
    ax.set_ylabel(f'{num}')
    ax.set_xlabel(f'{cat}')

    ax.set_title(f'{cat} Vs {num}',fontsize=15)
    ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")

    plt.show()
```

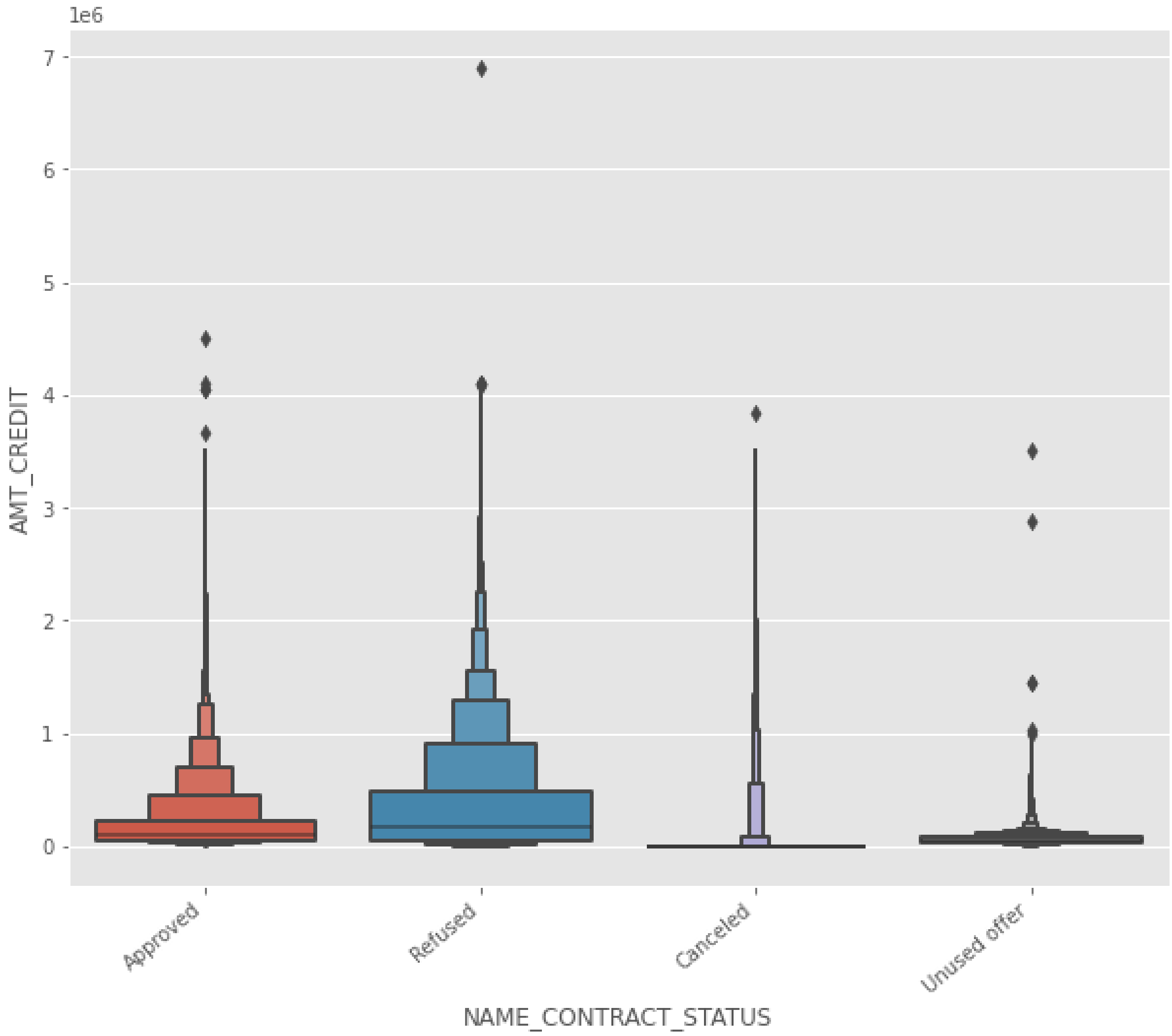
```
#by-varient analysis of Contract status and Annuity of previous appliction
plot_by_cat_num('NAME_CONTRACT_STATUS', 'AMT_ANNUITY')
```



> From the above plot we can see that loan application for people with lower AMT_ANNUITY gets canceled or Unused most of the time.

> We also see that applications with too high AMT ANNUITY also got refused more often than others.

```
#by-varient analysis of Contract status and Final credit amount disbursed to the customer
plot_by_cat_num('NAME_CONTRACT_STATUS', 'AMT_CREDIT')
```



> We can infer that when the AMT_CREDIT is too low, it get's cancelled/unused most of the time.

6. Merging the files and analyzing the data

```
## Merging the two files to do some analysis
NewLeftPrev = pd.merge(newapp_Final, preapp, how='left', on=['SK_ID_CURR'])
```

6.1 Basic checks on NewLeftPrev

```
NewLeftPrev.shape
```

```
(1228171, 62)
```

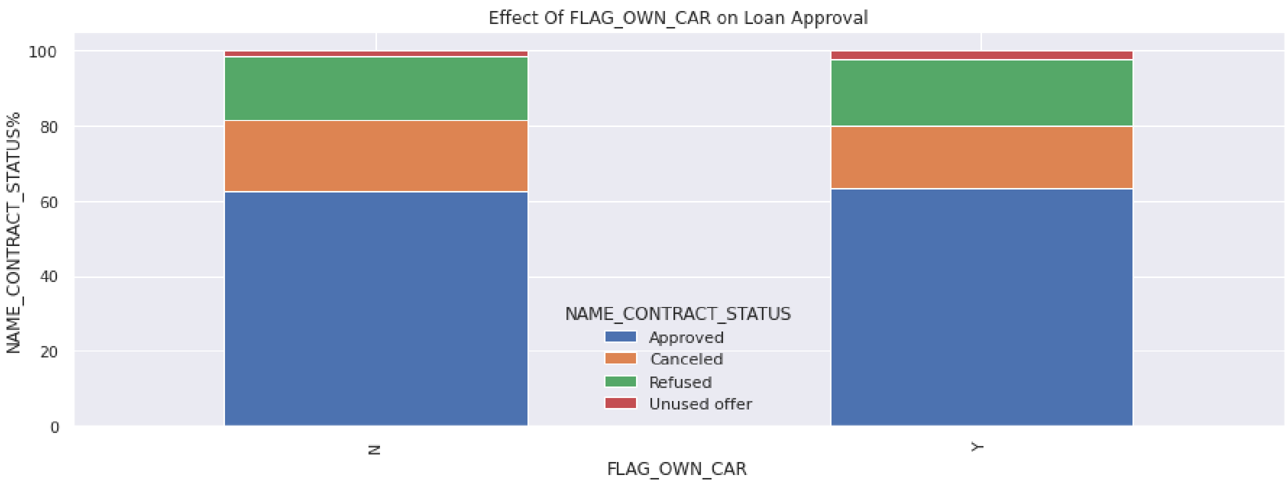
```
NewLeftPrev.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1228171 entries, 0 to 1228170
Data columns (total 62 columns):
 #   Column                           Non-Null Count    Dtype
---  ------                           --------------    -----
 0   SK_ID_CURR                       1228171 non-null  int64
 1   TARGET                           1228171 non-null  int64
 2   CODE_GENDER                      1228171 non-null  object
 3   FLAG_OWN_CAR                     1228171 non-null  object
 4   FLAG_OWN_REALTY                  1228171 non-null  object
 5   INCOME_GROUP                     1228171 non-null  category
 6   AGE_GROUP                        1228167 non-null  category
 7   AMT_CREDIT_x                     1228171 non-null  float64
 8   AMT_INCOME_TOTAL                 1228171 non-null  float64
 9   CREDIT_INCOME_RATIO              1228171 non-null  float64
 10  NAME_INCOME_TYPE                 1228171 non-null  object
 11  NAME_EDUCATION_TYPE              1228171 non-null  object
 12  NAME_FAMILY_STATUS               1228171 non-null  object
 13  NAME_HOUSING_TYPE                1228171 non-null  object
 14  DAYS_EMPLOYED                    1228171 non-null  int64
 15  DAYS_REGISTRATION                1228171 non-null  float64
 16  FLAG_EMAIL                       1228171 non-null  int64
 17  OCCUPATION_TYPE                  1228171 non-null  object
 18  CNT_FAM_MEMBERS                  1228169 non-null  float64
 19  REGION_RATING_CLIENT_W_CITY      1228171 non-null  int64
 20  ORGANIZATION_TYPE                1228171 non-null  object
 21  SOCIAL_CIRCLE_30_DAYS_DEF_PERC   587571 non-null   float64
 22  SOCIAL_CIRCLE_60_DAYS_DEF_PERC   584729 non-null   float64
 23  AMT_REQ_CREDIT_BUREAU_DAY        1085119 non-null  float64
 24  AMT_REQ_CREDIT_BUREAU_MON        1085119 non-null  float64
 25  AMT_REQ_CREDIT_BUREAU_QRT        1085119 non-null  float64
 26  NAME_CONTRACT_TYPE_x             1228171 non-null  object
 27  AMT_ANNUITY_x                    1228094 non-null  float64
 28  REGION_RATING_CLIENT             1228171 non-null  int64
 29  AMT_GOODS_PRICE_x                1227137 non-null  float64
 30  SK_ID_PREV                       1203051 non-null  float64
 31  NAME_CONTRACT_TYPE_y             1203051 non-null  object
 32  AMT_ANNUITY_y                    942537 non-null   float64
 33  AMT_APPLICATION                  1203051 non-null  float64
 34  AMT_CREDIT_y                     1203050 non-null  float64
 35  AMT_GOODS_PRICE_y                932463 non-null   float64
 36  WEEKDAY_APPR_PROCESS_START       1203051 non-null  object
 37  HOUR_APPR_PROCESS_START          1203051 non-null  float64
 38  FLAG_LAST_APPL_PER_CONTRACT      1203051 non-null  object
 39  NFLAG_LAST_APPL_IN_DAY           1203051 non-null  float64
 40  NAME_CASH_LOAN_PURPOSE           1203051 non-null  object
 41  NAME_CONTRACT_STATUS             1203051 non-null  object
 42  DAYS_DECISION                    1203051 non-null  float64
 43  NAME_PAYMENT_TYPE                1203051 non-null  object
 44  CODE_REJECT_REASON               1203051 non-null  object
 45  NAME_TYPE_SUITE                  612462 non-null   object
 46  NAME_CLIENT_TYPE                 1203051 non-null  object
 47  NAME_GOODS_CATEGORY              1203050 non-null  object
 48  NAME_PORTFOLIO                   1203050 non-null  object
```

```
 49  NAME_PRODUCT_TYPE              1203050 non-null  object
 50  CHANNEL_TYPE                   1203050 non-null  object
 51  SELLERPLACE_AREA               1203050 non-null  float64
 52  NAME_SELLER_INDUSTRY           1203050 non-null  object
```

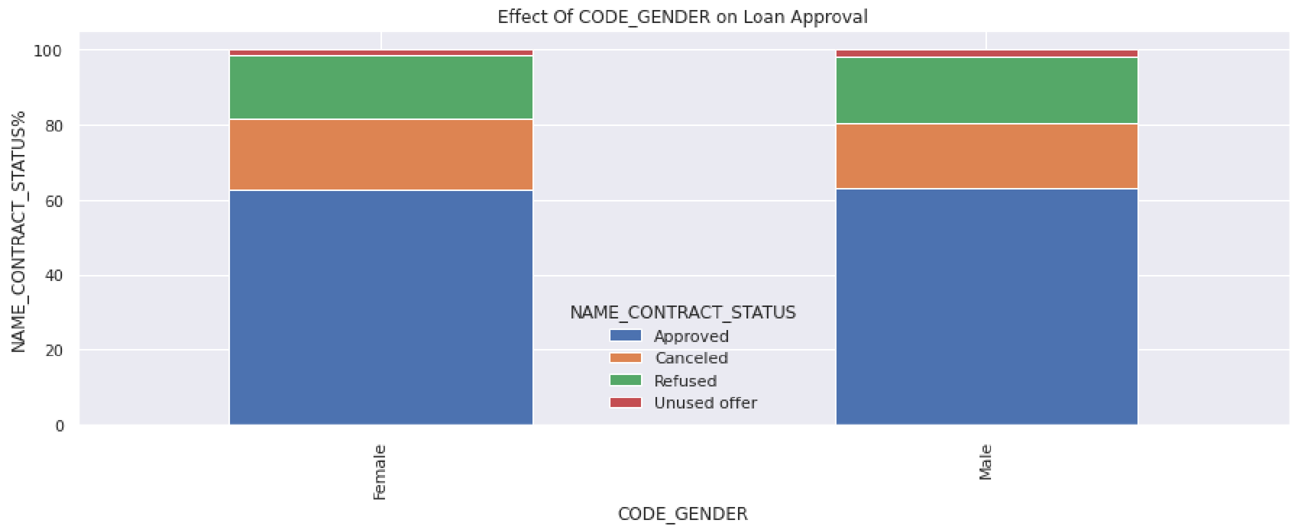
```
def plotuni_combined(Varx,Vary):
    # 100% bar chart
    plt.style.use('seaborn-darkgrid')
    sns.despine
    NewDat = NewLeftPrev.pivot_table(values='SK_ID_CURR',
                      index=Varx,
                      columns=Vary,
                      aggfunc='count')
    NewDat=NewDat.div(NewDat.sum(axis=1),axis='rows')*100
    sns.set()
    NewDat.plot(kind='bar',stacked=True,figsize=(15,5))
    plt.title(f'Effect Of {Varx} on Loan Approval')
    plt.xlabel(f'{Varx}')
    plt.ylabel(f'{Vary}%')
    plt.show()
```

```
plotuni_combined('FLAG_OWN_CAR','NAME_CONTRACT_STATUS')
```



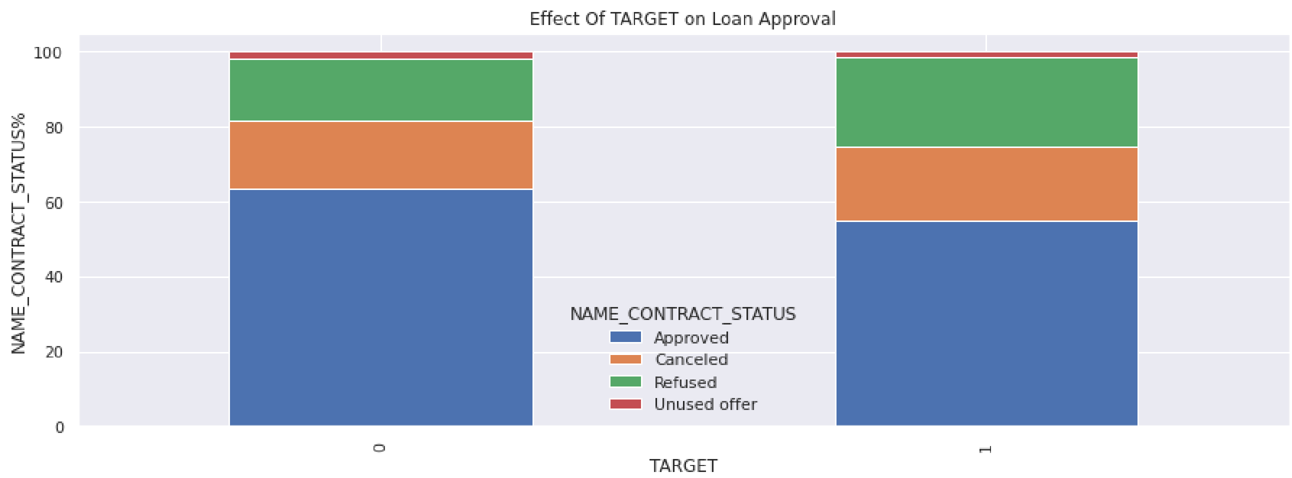
> We see that car ownership doesn't have any effect on application approval or rejection. But we saw earlier that the people who has a car has lesser chances of default. The bank can add more weightage to car ownership while approving a loan amount

```
plotuni_combined('CODE_GENDER','NAME_CONTRACT_STATUS')
```



> We see that code gender doesn't have any effect on application approval or rejection.
> But we saw earlier that female have lesser chances of default compared to males.
> The bank can add more weightage to female while approving a loan amount.

```
plotuni_combined('TARGET','NAME_CONTRACT_STATUS')
```



Target variable (0 - Non Defaulter 1 - Defaulter)

We can see that the people who were approved for a loan earlier, defaulted less often where as people who were refused a loan earlier have higher chances of defaulting.

3s completed at 3:04 PM