Credit EDA Assignment

In this assignment aims to give us an idea of applying EDA in a real business scenario.

In this assignment, apart from applying the techniques that i have learnt in the EDA module, i will also develop a basic understanding of risk analytics in banking and financial services and understand how data is used to minimise the risk of losing money while lending to customers.

Introduction:-

For the purpose of this case study, three data sets were provided are as :
1. Application Data
2. Previous Application Data
3. Columns description Data

Firstly we start with importing important libraries of numpy / Pandas / matplolib / seaborn / warnings into juypter notebook following withimporting of Application data .

```
# Importing Analytical libraries
import numpy as np
import pandas as pd

# Importing Visualisation library
import matplotlib.pyplot as plt
import seaborn as sns

# Importing warnings
import warnings
warnings.filterwarnings('ignore')
```

```
# Setting option to display all columns and rows

pd.set_option("Display.max_columns",None)
pd.set_option("Display.max_rows",None)
```

```
# Importing dataset __application_data.csv__
df= pd.read_csv('application_data.csv')
```

Next Step is to do the Routine Structure Check of Application data by the use of following python function :-

1. info
2. describe
3. shape
4. dtype
5.columns
6. head
7. tail
8. unique
9. nunique
10. value_counts

Routine Structure Check of DataFrame

```
# Checking shape i.e. number of rows and columns of dataset
df.shape
```

```
(307511, 122)
```

```
# Checking information of dataset
df.info('all')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 122 columns):
 #     Column                                Dtype
---    ------                                -----
```

```
# Checking description of dataset
df.describe()
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_POPULATION_RELATI
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	307499.000000	3.072330e+05	307511.0000
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	27108.573909	5.383962e+05	0.0208
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	14493.737315	3.694465e+05	0.0138
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1615.500000	4.050000e+04	0.0002
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16524.000000	2.385000e+05	0.0100
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	24903.000000	4.500000e+05	0.0188
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.000000	6.795000e+05	0.0286
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	258025.500000	4.050000e+06	0.0725

```
# Checking datatypes of dataset columns
df.dtypes
```

```
SK_ID_CURR                    int64
TARGET                        int64
NAME_CONTRACT_TYPE           object
CODE_GENDER                  object
FLAG_OWN_CAR                 object
FLAG_OWN_REALTY              object
CNT_CHILDREN                  int64
```


After Routine Structure check we do Data Quality Check in following steps :-

Step 1:- **Find the percentage of Missing Values in DataFrame**

```
# creating null function which gives percentage of missing values in each column of dataframe
def null_cal(dataframe):
    null_calculation = 100*dataframe.isnull().mean()
    return null_calculation
```

```
null_cal(df)
```

```
SK_ID_CURR                    0.000000
TARGET                        0.000000
NAME_CONTRACT_TYPE            0.000000
CODE_GENDER                   0.000000
FLAG_OWN_CAR                  0.000000
FLAG_OWN_REALTY               0.000000
```

Step 2- Remove Columns with High Missing Percentage

```
# Creating a function to return list of columns having null values more than cutoff
def col_having_high_null_val(dataframe,cut_off_val):
    colms_to_drop = list(dataframe.columns[null_cal(dataframe)>=cut_off_val])
    return colms_to_drop
```

```
# Setting column cut_off_val percentage to 45 and finding names of columns with such percentage
df_null_val_cols = col_having_high_null_val(df,45)
df_null_val_cols
```

```
['OWN_CAR_AGE',
 'EXT_SOURCE_1',
 'APARTMENTS_AVG',
 'BASEMENTAREA_AVG',
```

```
# Dropping columns with null percentage >= 45
df = df.drop(df_null_val_cols,axis=1)
```

```
df.columns
```

```
Index(['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_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
```

Step 3- Checking the shape after deleting high null percentage columns and finding 5 Columns from Data Frame whose null % < 13 and we have to impute these null values

```
# Checking shape of df
df.shape
```

```
(307511, 53)
```

Imputing Missing Value Strategy:-
1. For numerical/continuous we can use mean or median, median is preferred as it is unaffected by extreme values.
2. For categorical/discrete columns we can use mode for imputing the missing values.

For checking missing value lets design two function
1. missing_val_func_num() - to describe column values for continuous column of data frame

```
## missing_val_func function - to describe column values
def missing_val_func_num(dataframe,col):
    desc = print(dataframe[col].describe())
    median = print("Median ",dataframe[col].median())
    qua90 = print("Quantile",dataframe[col].quantile([0.90]))
    qua95 = print("Quantile",dataframe[col].quantile([0.95]))
    qua99 = print("Quantile",dataframe[col].quantile([0.99]))
    return

```

2. missing_val_func_cat() - to describe column values for categorical column of data frame.

```
## missing_val_func_cat function - to describe column values
def missing_val_func_cat(dataframe,col):
    desc = print(100*dataframe[col].value_counts(normalize=True))
    median = print("Mode ",dataframe[col].mode())
    return
```

Following are the 5 columns i have shortlisted to impute values
AMT_ANNUITY = 0.003902
AMT_GOODS_PRICE = 0.090403
NAME_TYPE_SUITE = 0.420148
DAYS_LAST_PHONE_CHANGE = 0.000325
OCCUPATION_TYPE = 31.345545

By using missing_val_func_num() and missing_val_func_cat() we analyse the columns having null values and on the basis of describe and type of Column we will impute for missing values.

1. **In AMT_ANNUITY we use median to impute missing values because there is huge difference between 99 percentile and max value which can cause mean to deviate**
2. **In AMT_GOODS_PRICE we use median to impute missing values because there is huge difference between 99 percentile and max value which can cause mean to deviate**
3. **In NAME_TYPE_SUITE column we can impute Unaccompanied in missing values**
4. **In DAYS_LAST_PHONE_CHANGE we can use any of mean/median to impute missing values**
5. **In OCCUPATION_TYPE we can impute others inplace of missing values**

Step 4 - Check the datatypes of all the columns and change the Datatype also check anomalies like negative age in data

```
# Converting values of columns having negative values to positive
col_to_pos=['DAYS_BIRTH','DAYS_REGISTRATION','DAYS_ID_PUBLISH','DAYS_LAST_PHONE_CHANGE','DAYS_EMPLOYED']
for i in col_to_pos:
    df[i] = abs(df[i])
```

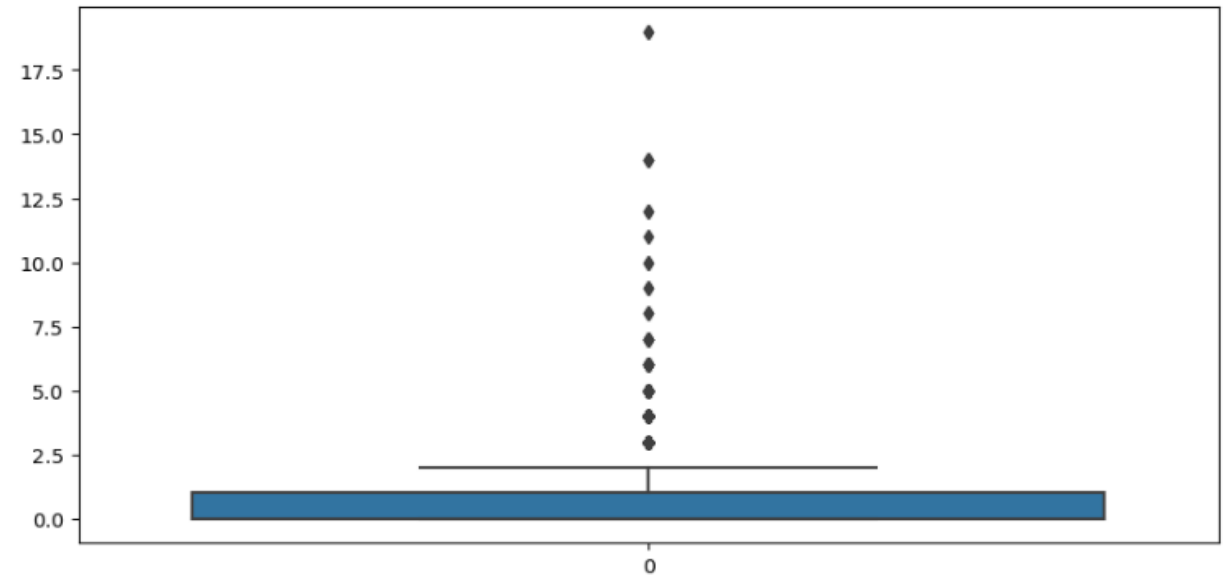
Step 5- Finding Columns where we identified Presence of Outliers.

After anlaysing we will create a variable having presence of ouliers and then with the help of describe and box plot we will analyse all columns and choose how to impute for outliers .

```
# Creating a variable for which we have to find outliers
df_outlier_col = ['CNT_CHILDREN','AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY',
                  'AMT_GOODS_PRICE','CNT_FAM_MEMBERS','DAYS_EMPLOYED','DAYS_REGISTRATION']

```

```
# Plotting boxplot for df_outlier_col
for i in df_outlier_col:
    print(i,'\n',df[i].describe(),'\n')
    plt.figure(figsize=[10,5])
    sns.boxplot(df[i])
    plt.show()
```

```
CNT_CHILDREN
 count     307511
unique        15
top            0
freq      215371
Name: CNT_CHILDREN, dtype: int64
```



Insights on anlaysis data and plotting boxplot after anlysing columns for outliers :-

1. In CNT_CHILDREN column we have 99 percent have children <=3 so few have only > 3, we can use binning to handle outliers.
2. In AMT_INCOME_TOTAL column we have 99 percent have income <=472500 so few have high income, we can use binning to handle outliers
3. In AMT_CREDIT column we have 99 percent have Credit <= 1854000 so few have high Credit, we can use binning to handle outliers
4. In AMT_ANNUITY column we have 99 percent have Annuity <= 70006.5 so few have high Annuity, we can use binning to handle outliers
5. In AMT_GOODS_PRICE column we have 99 percent have Goods price <= 1800000.0 so few have high Goods price, we can use binning to handle outliers
6. In CNT_FAM_MEMBERS column we have 99 percent have count of family members <= 5 so few have high count of family members, we can use binning to handle outliers
7. In DAYS_EMPLOYED column we have 80 percent have days employed <= 9188.0 so one have high days employed which clear is a mistake, we can delete this outlier or use IQR to rectify.
8. In DAYS_REGISTRATION column we have 99 percent have days registration <= 13879.0 so few have high days registration, we can use mean, median or IQR to rectify.

Step 6- Next you need to finding Columns for Binning and Do the Binning

Columns for binning in application data are :-

1. DAYS_BIRTH
2. DAYS_EMPLOYED
3. AMT_INCOME_TOTAL
4. AMT_CREDIT
5. AMT_ANNUITY
6. AMT_GOODS_PRICE

But Before binning for DAYS_BIRTH and DAYS_EMPLOYED we need to convert it into years.

```
# before binning DAYS_BIRTH,DAYS_EMPLOYED we like to convert into years
df['Age'] = df['DAYS_BIRTH'].apply(lambda x: round(x/365,2))
df['YEARS_EMPLOYED'] = df['DAYS_EMPLOYED'].apply(lambda x:round(x/365,2))
```

Binning for Age_group

```
# binning for age
df['Age_Group'] = pd.cut(df['Age'],[0,30,40,50,60,999],
                         labels=['0-30','30-40','40-50','50-60','60+'])
```

Binning for YEARS_EMPLOYED

```
# binning for YEARS_EMPLOYED
df['YEARS_EMPLOYED_BUCKET'] = pd.cut(df['YEARS_EMPLOYED'],[0,2,4,6,8,10,15,1001],
                                     labels=['0-2','2-4','4-6','6-8','8-10','10-15','15+'])
```

Binning for AMT_INCOME_BUCKET

```
# binning the AMT_INCOME_TOTAL column
df['AMT_INCOME_BUCKET'] = pd.cut(df['AMT_INCOME_TOTAL'],
                                 bins=[0,100000,200000,300000,400000,500000,600000,700000,800000,10000000000],
                                 labels=['0-100000','100000-200000','200000-300000','300000-400000',
                                         '400000-500000','500000-600000','600000-700000','700000-800000','800000+'])
```

Binning for AMT_CREDIT_BUCKET

```
# binning the AMT_CREDIT column
df['AMT_CREDIT_BUCKET'] = pd.cut(df['AMT_CREDIT'],
                                 bins=[0,100000,200000,300000,400000,500000,600000,700000,800000,10000000000],
                                 labels=['0-100000','100000-200000','200000-300000','300000-400000',
                                         '400000-500000','500000-600000','600000-700000','700000-800000','800000+'])
```

Binning for AMT_ANNUITY_BUCKET

```
# Creating AMT_ANNUITY_BUCKET for AMT_ANNUITY
df['AMT_ANNUITY_BUCKET'] = pd.cut(df['AMT_ANNUITY'],
                                  bins=[0,100000,200000,300000,400000,500000,600000,700000,800000,10000000000],
                                  labels=['0-100000','100000-200000','200000-300000','300000-400000',
                                          '400000-500000','500000-600000','600000-700000','700000-800000','800000+'])
```

Binning for AMT_GOODS_PRICE

```
# binning for AMT_GOODS_PRICE
df['AMT_GOODS_PRICE_BUCKET'] = pd.cut(df['AMT_GOODS_PRICE'],
                                  bins=[0,100000,200000,300000,400000,500000,600000,700000,800000,10000000000],
                                  labels=['0-100000','100000-200000','200000-300000','300000-400000',
                                          '400000-500000','500000-600000','600000-700000','700000-800000','800000+'])
```

Analysis for application data .

That is we check the Imbalance percentage. No balancing technique required. On TARGET_Column we need to apply value_counts() to find the % imbalance.
We will create two data frames df_0 for non defaulters having TARGET = 0 and df_1 for defaulters TARGET = 1

```
df.TARGET.value_counts()
```

```
TARGET
0     282686
1      24825
Name: count, dtype: int64
```

df_0 dataframe contains data of only those applicants whose Target Column value Matches with 0 (i.e non defaulters)

```
# Creating df_0 which consists applicants having TARGET =0
df_0 = df[df['TARGET']==0]
```

```
# Creating df_1 which consists applicants having TARGET =1
df_1=df[df['TARGET']==1]
```

After creating df_0,df_1 we create variable for categorical column so that we can run a for loop to plot graph for categorical columns.

```
# Using info for df to have a insight of columns of dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 61 columns):
 #   Column                       Non-Null Count   Dtype
```

```
# Creating a varible cat_columns for all columns which are categorial
cat_columns= ['NAME_CONTRACT_TYPE','CODE_GENDER','NAME_TYPE_SUITE','NAME_INCOME_TYPE','NAME_EDUCATION_TYPE',
              'FLAG_OWN_CAR','NAME_FAMILY_STATUS','NAME_HOUSING_TYPE','OCCUPATION_TYPE', 'WEEKDAY_APPR_PROCESS_START',
              'ORGANIZATION_TYPE','FLAG_OWN_REALTY','LIVE_CITY_NOT_WORK_CITY','REG_CITY_NOT_LIVE_CITY',
              'REG_CITY_NOT_WORK_CITY','REG_REGION_NOT_WORK_REGION','LIVE_REGION_NOT_WORK_REGION','REGION_RATING_CLIENT',
              'WEEKDAY_APPR_PROCESS_START','REGION_RATING_CLIENT_W_CITY','CNT_CHILDREN','CNT_FAM_MEMBERS']
```

```
# Checking all columns which are categorial
cat_columns
```

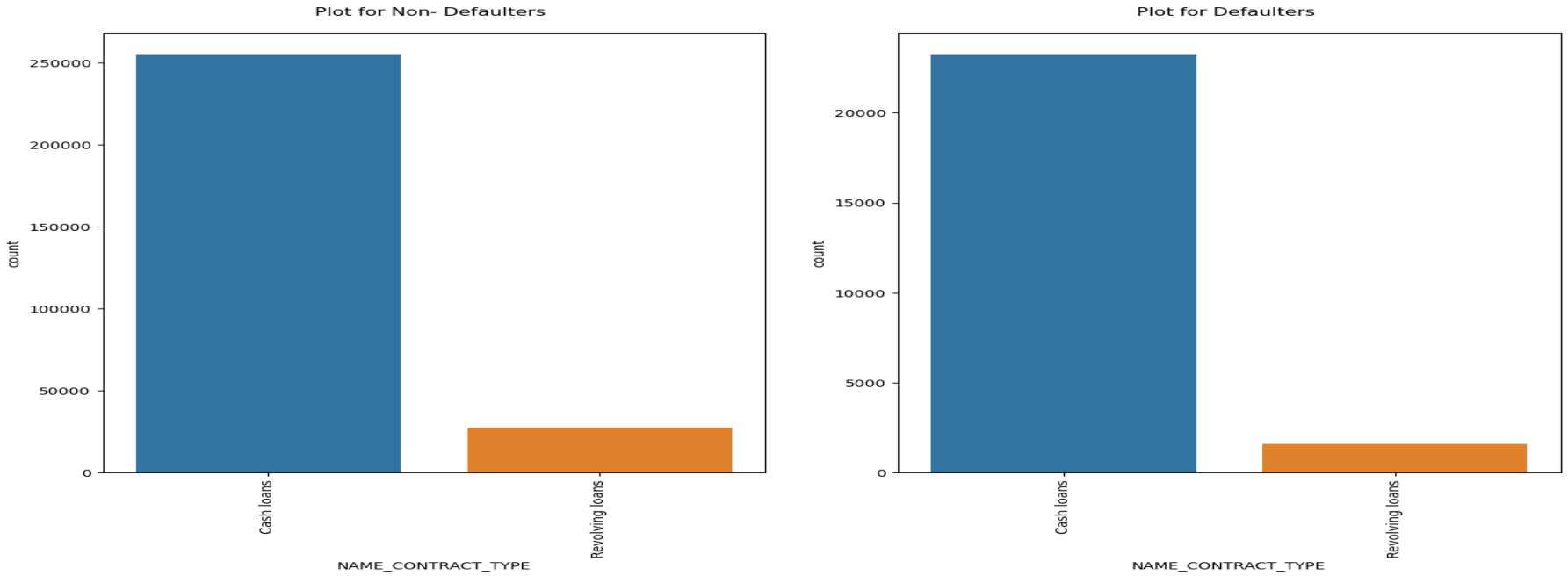
```
['NAME_CONTRACT_TYPE',
 'CODE_GENDER',
 'NAME_TYPE_SUITE',
 'NAME_INCOME_TYPE',
 'NAME_EDUCATION_TYPE',
```

First we do univariate analysis for categorical columns for both df_0 i.e. non-defaulters and df_1 i.e defaulters.
Using for loop to plot countplot for non-defaulters and defaulters.

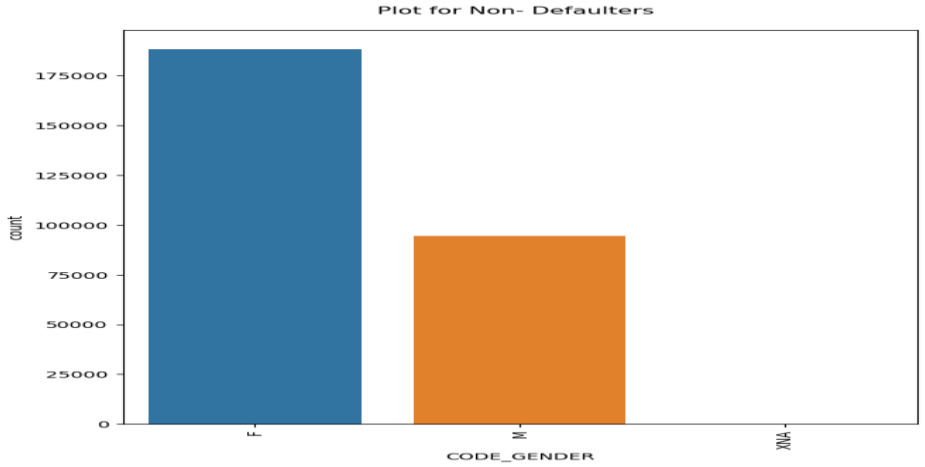
```
# plotting countplot chart for all categorial columns of data set for both defaulters and non defaulters

for i in cat_columns:
    plt.figure(figsize=[16,8])
# plotting subplot for Non- Defaulters
    plt.subplot(1,2,1)
    plt.title("Plot for Non- Defaulters \n")
    sns.countplot(data=df_0, x=i)
    plt.xticks(rotation=90)
# plotting subplot for Defaulters
    plt.subplot(1,2,2)
    plt.title("Plot for Defaulters \n")
    sns.countplot(data=df_1, x=i)
    plt.xticks(rotation=90)
    plt.show()
```

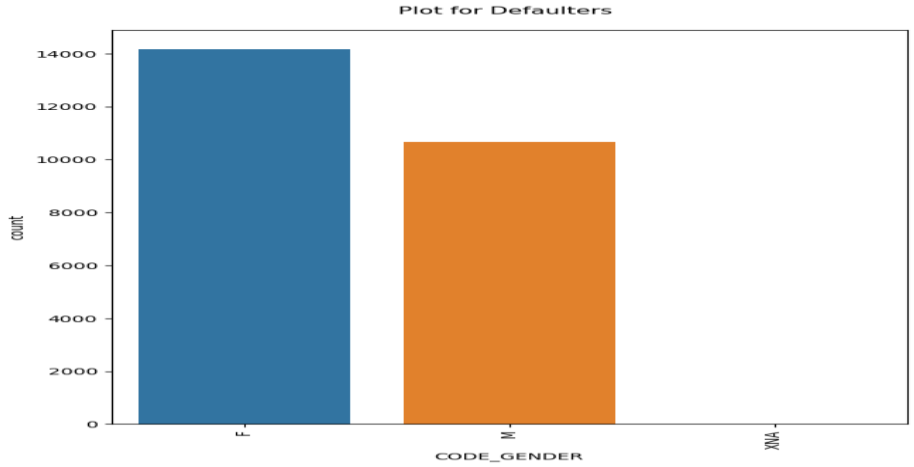
After plotting countplot few points which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:

1. NAME_CONTRACT TYPE- Cash Loans are large part of the company's portfolio. For Target 0 - 90.2 % and almost 93.5% for Target-1

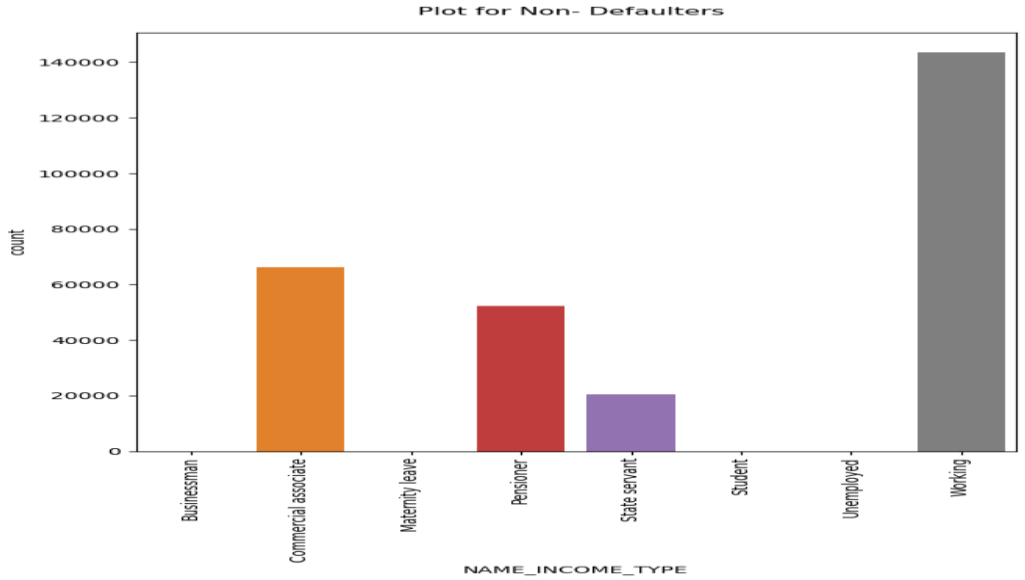


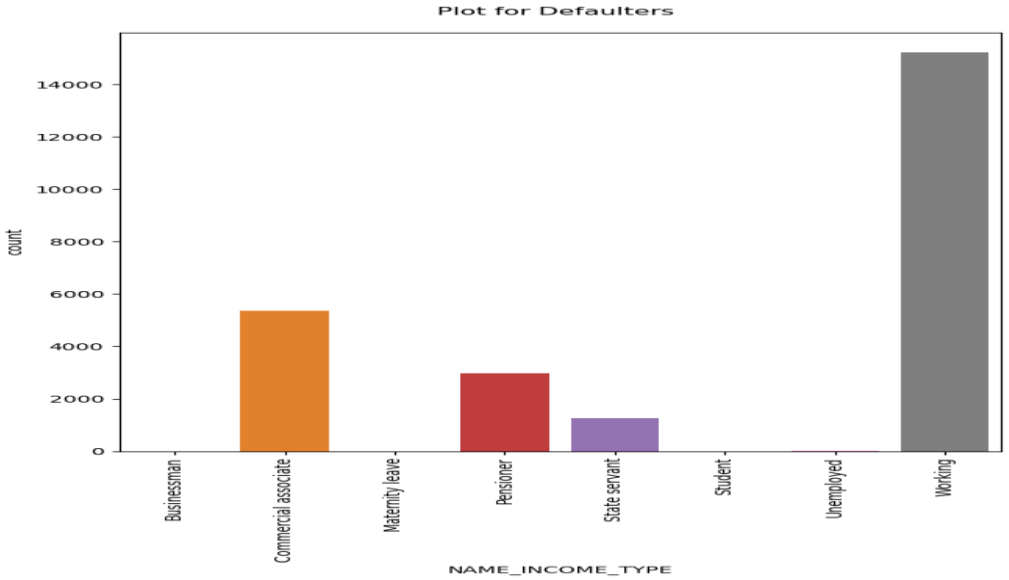
2. 'CODE_GENDER' - Ratio of F to M in Target 0 is 1.99 and F to M in Target 0 - 1.33 indicating that MEN are defaulting more than Women



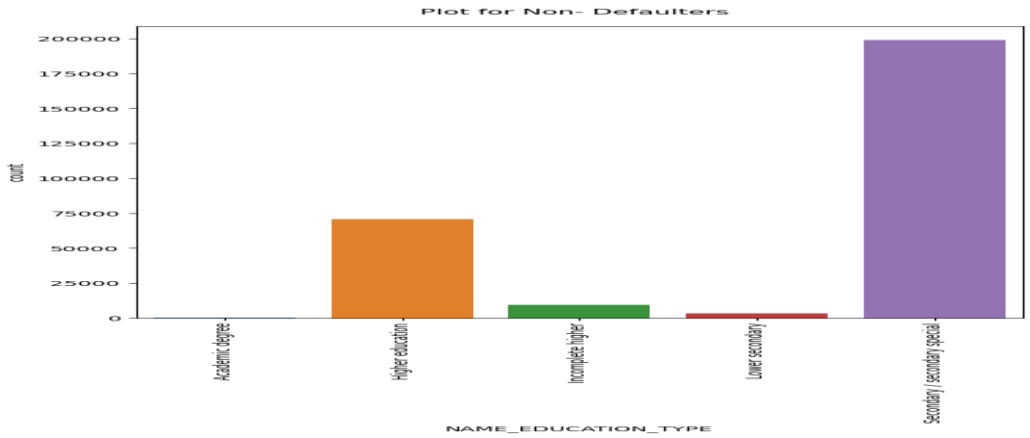


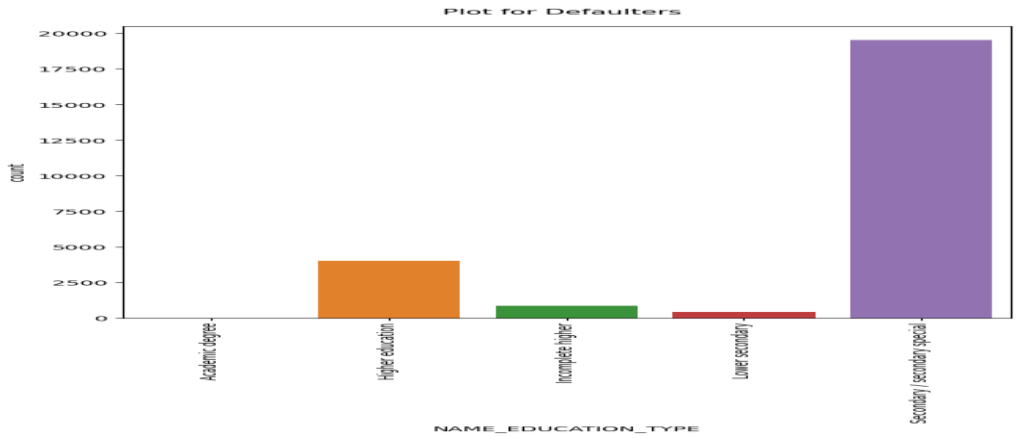
3. NAME_INCOME_TYPE - 50.78% working in case of Target 0 and 61.33% in case of Target 1 are working income types.



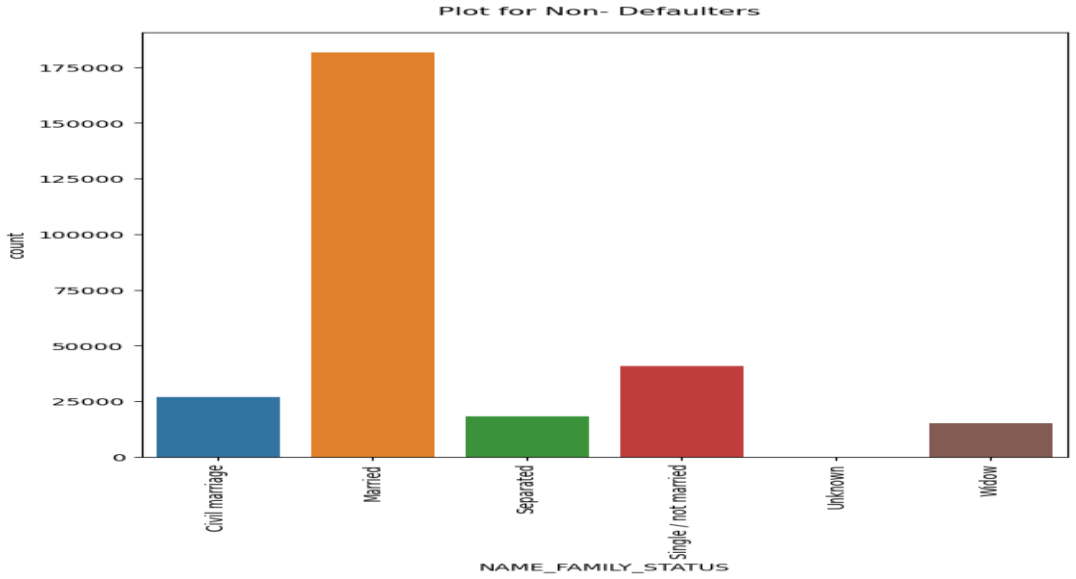


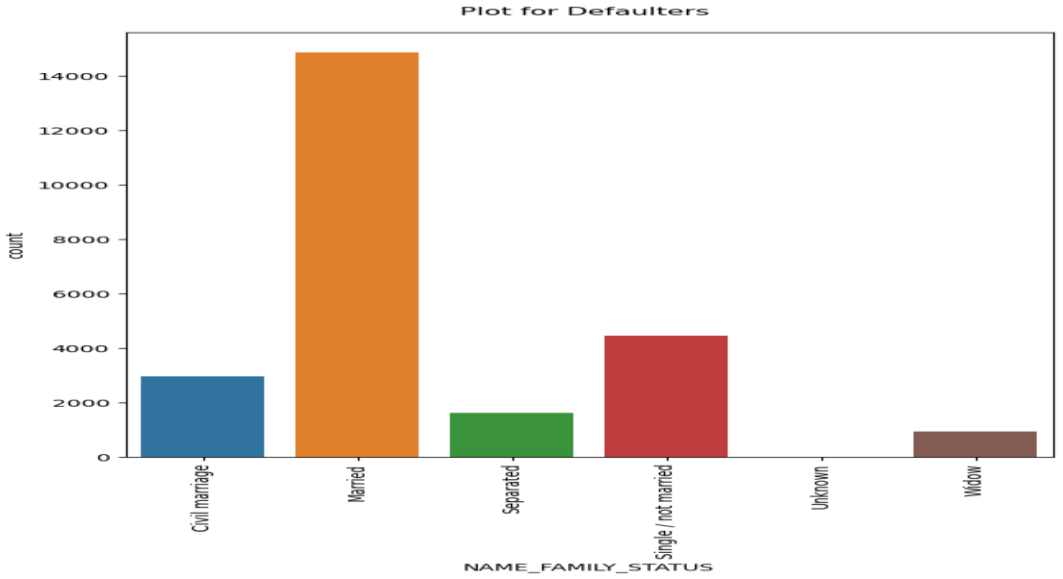
4. NAME_EDUCATION_TYPE - In both Target 0 and 1, applicants with Secondary Education has applied for loans more than others. 90% of defaulting payments are from applicants with secondary income.Needs further analysis



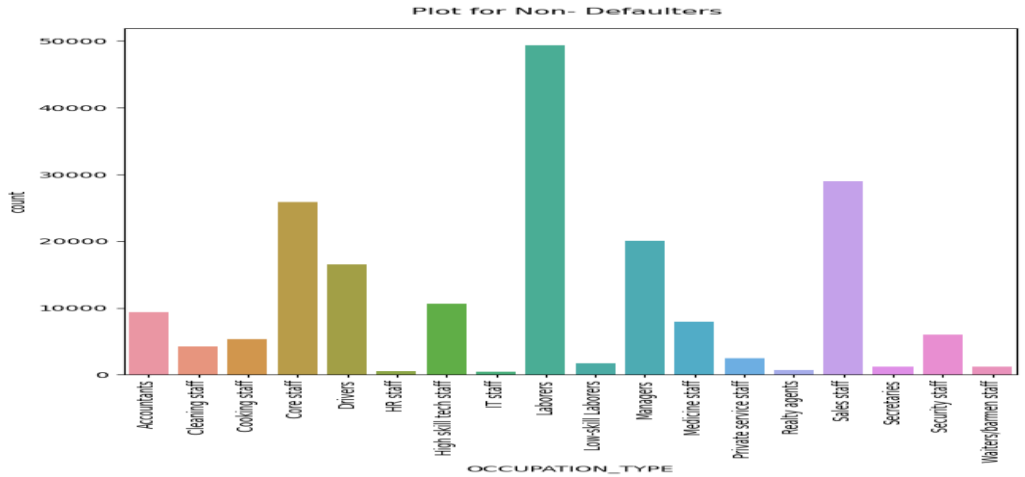


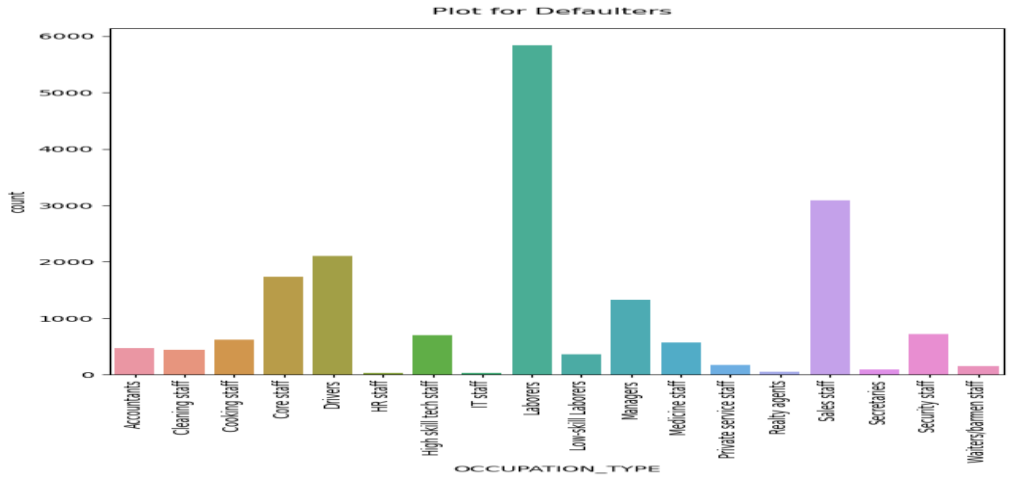
5. NAME_FAMILY_STATUS - Married applicants - almost 60% have defaulted on payments



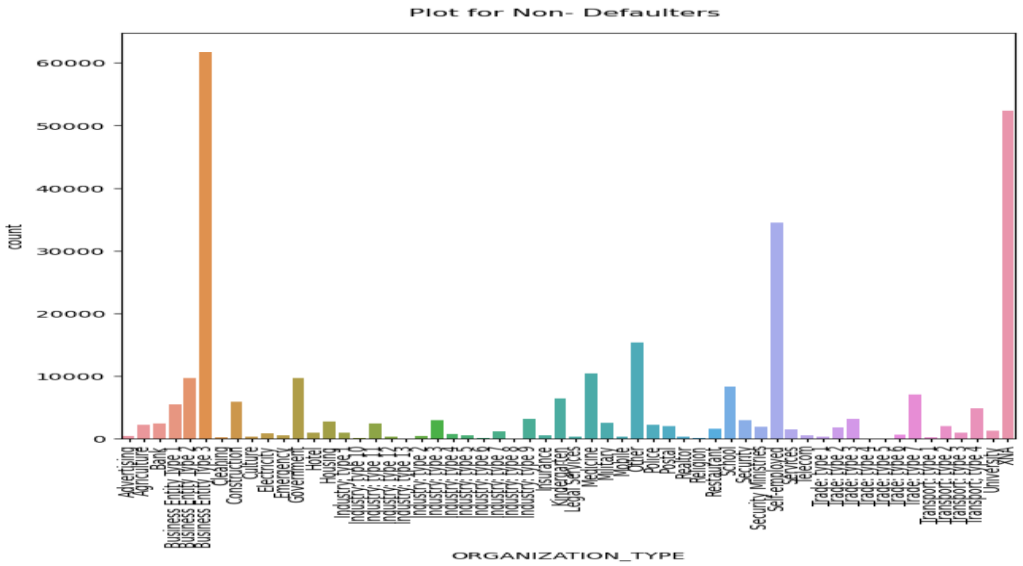


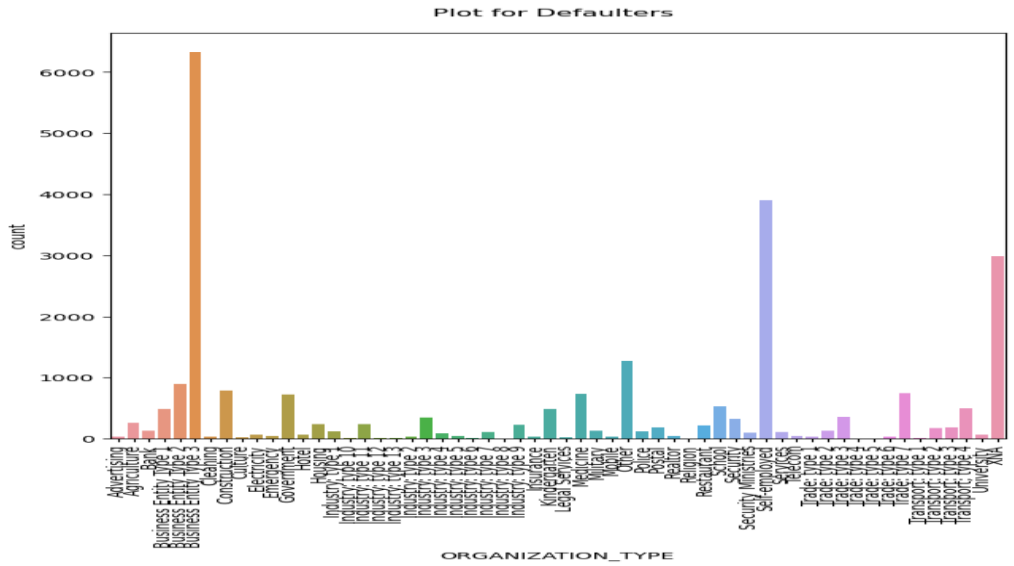
6. OCCUPATION_TYPE - Labourers, sales staff, drivers,core staffconstitute of 68.87 % of defaulters.Labourers is the highest percentage of applicants too





7.. ORGANIZATION_TYPE - Business ENTITY TYPE 3 AND SELF EMPLOYED add upto 41.5 % defaulters. The highest % of loan takers are also this category





After univariate analysis of categorical columns we do univariate analysis for continuous columns, but before we need to make a variable containing continuous columns of data.

```
# Univariate Analysis on continous variables
# making cont_col list which contains columns having datatypes as float
cont_col1 = df.columns[df.dtypes=='float'].tolist()
cont_col2 = df.columns[df.dtypes=='int'].tolist()
cont_col= list(set(cont_col1 + cont_col2))
print(cont_col)
```

```
['DAYS_REGISTRATION', 'AMT_INCOME_TOTAL', 'OBS_30_CNT_SOCIAL_CIRCLE', 'AMT_ANNUITY', 'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CRE
DIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_DAY', 'DEF_60_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCL
E', 'OBS_60_CNT_SOCIAL_CIRCLE', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'AMT_CREDIT', 'Age', 'YEARS_EMPLOYED', 'REGION_POPULATION_RELAT
IVE', 'AMT_REQ_CREDIT_BUREAU_HOUR', 'DAYS_LAST_PHONE_CHANGE', 'AMT_REQ_CREDIT_BUREAU_YEAR', 'AMT_GOODS_PRICE']
```

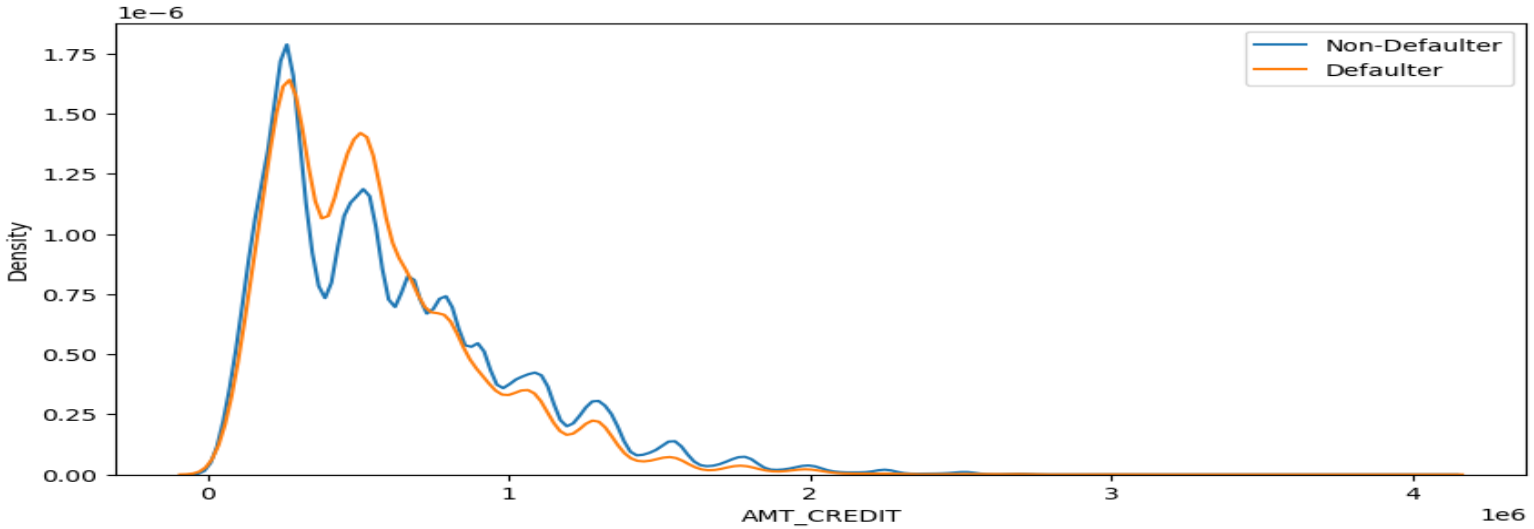
```
# selecting few important columns from cont_col list
cont_col_imp = ['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE',
                'REGION_POPULATION_RELATIVE', 'DAYS_REGISTRATION','EXT_SOURCE_3',
                'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE',
                'DEF_60_CNT_SOCIAL_CIRCLE','CNT_FAM_MEMBERS',
                'DAYS_LAST_PHONE_CHANGE', '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', 'Age','YEARS_EMPLOYED']
```

First we do univariate analysis for continous columns for both df_0 i.e. non-defaulters and df_1 i.e defaulters.
Using for loop to plot distplot for non-defaulters and defaulters.

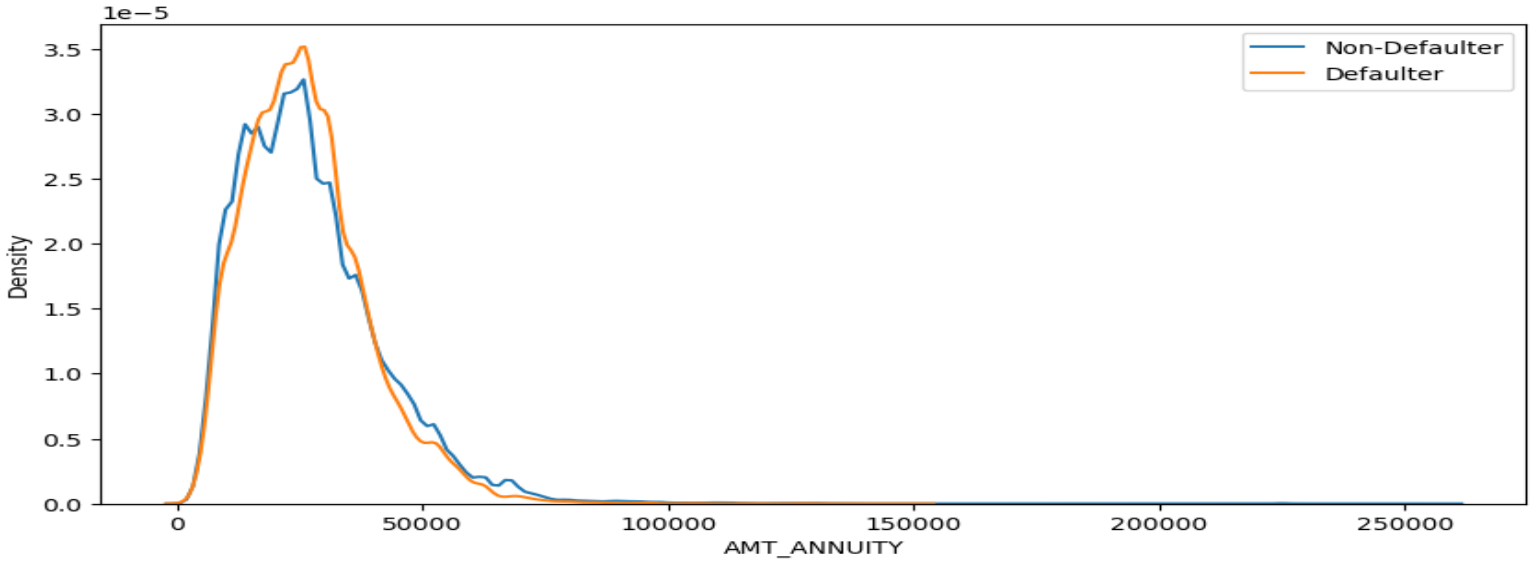
```
# plotting distplot for all the continous columns from count_col_imp for non-defaulters and defaulters
for i in cont_col_imp:
    plt.figure(figsize=[10,5])
    sns.distplot(df_0[i],hist=False,label='Non-Defaulter')
    sns.distplot(df_1[i],hist=False,label='Defaulter')
    plt.legend()
    plt.show()
```


After plotting disttplot few points which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:

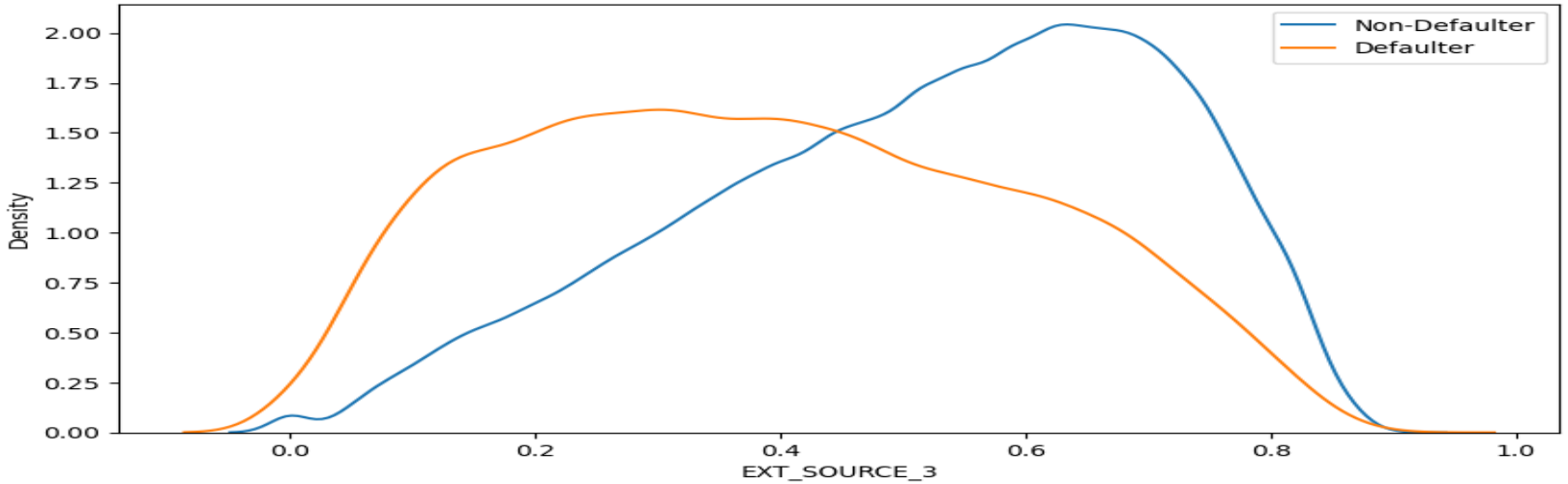
1. AMT_CREDIT - Appears lower for TARGET 1, which is a good sign as lesser default loss to the company.



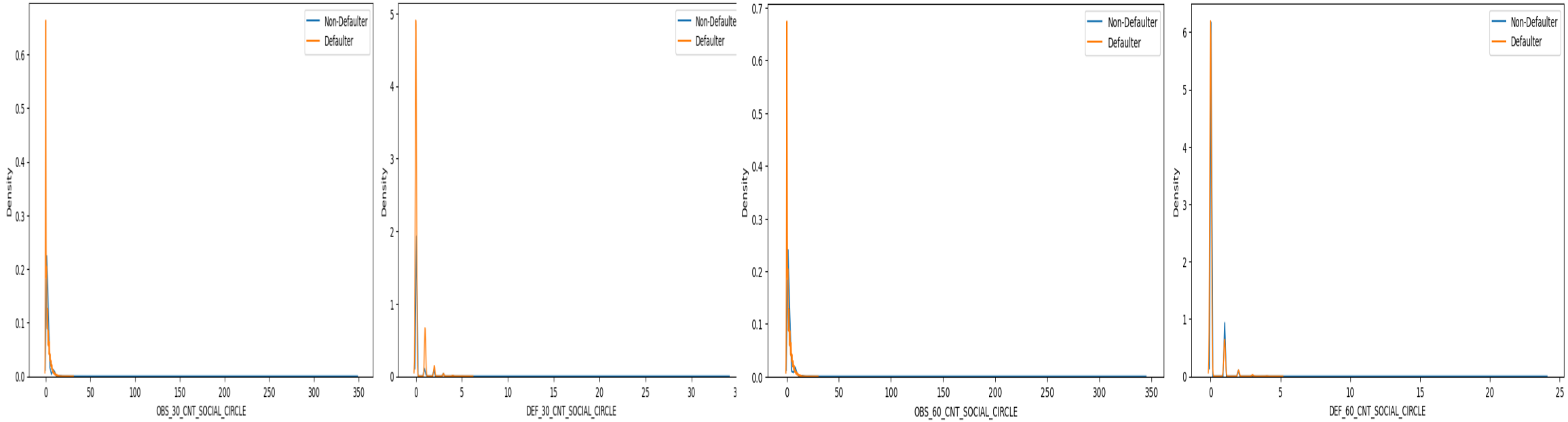
2 . AMT_ANNUITY - Appears higher for TARGET 1, which shows defaulter gets loans on high emi/intrest than TARGET 0..



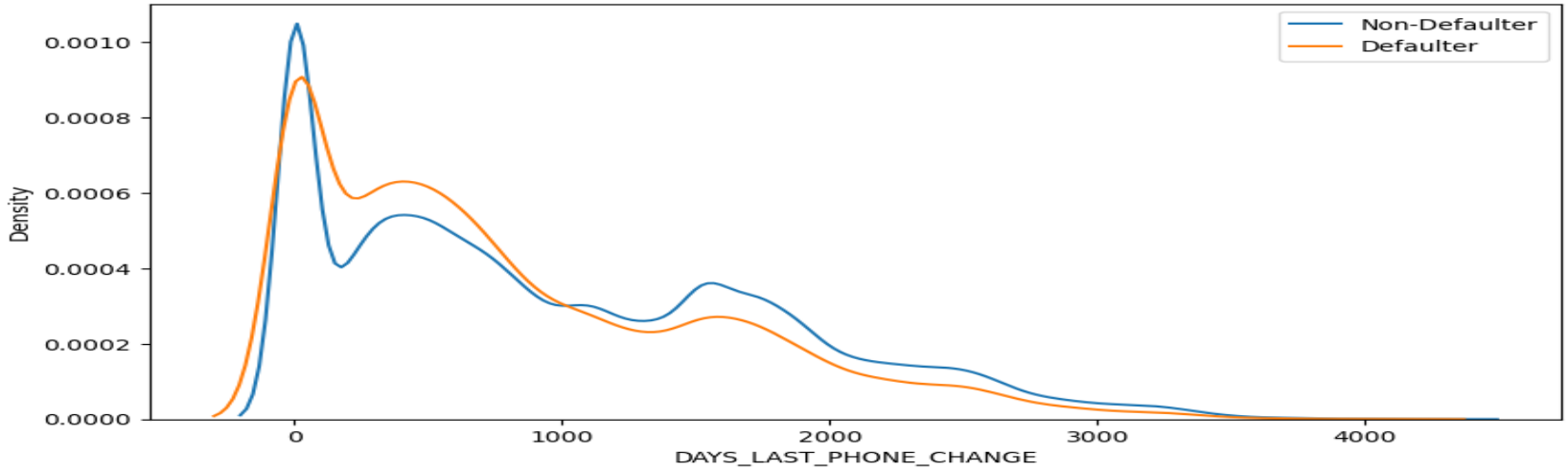
3. EXT_SOURCE_3-is clearing showing that TARGET 0 has higher densiy of higher scores.



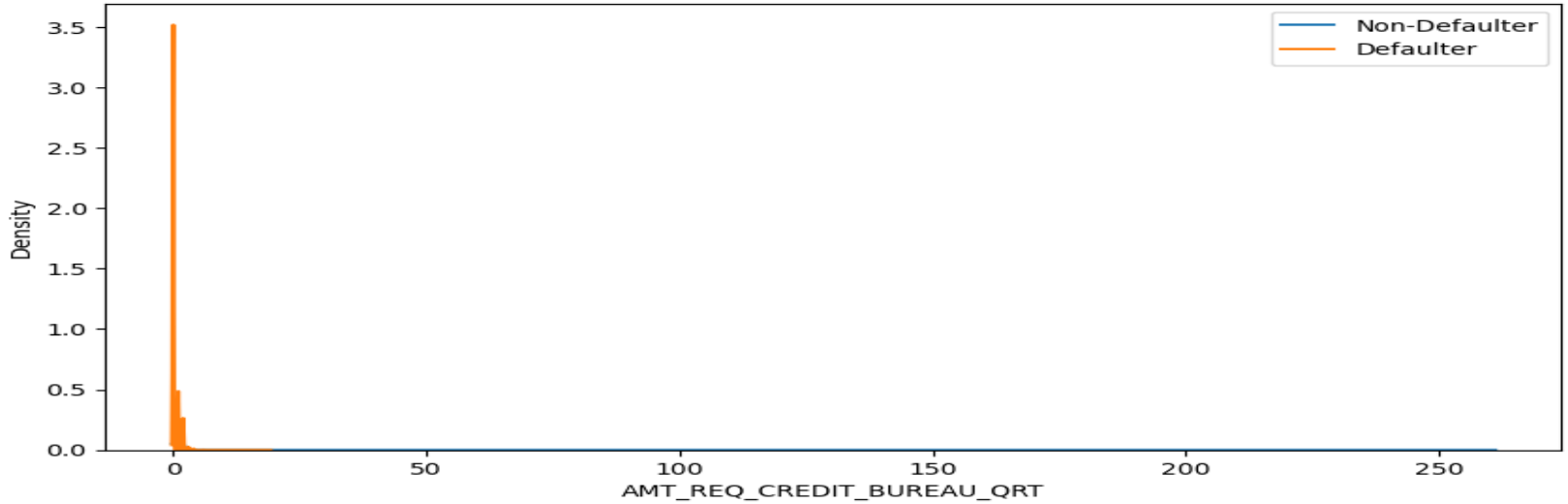
4. OBS_30_CNT_SOCIAL_CIRCLE, DEF_30_CNT_SOCIAL_CIRCLE, OBS_60_CNT_SOCIAL_CIRCLE, DEF_60_CNT_SOCIAL_CIRCLE -Very clearly visible that for Target 1, 30 DPD and 60 DPD observed in social surrounding is higher.



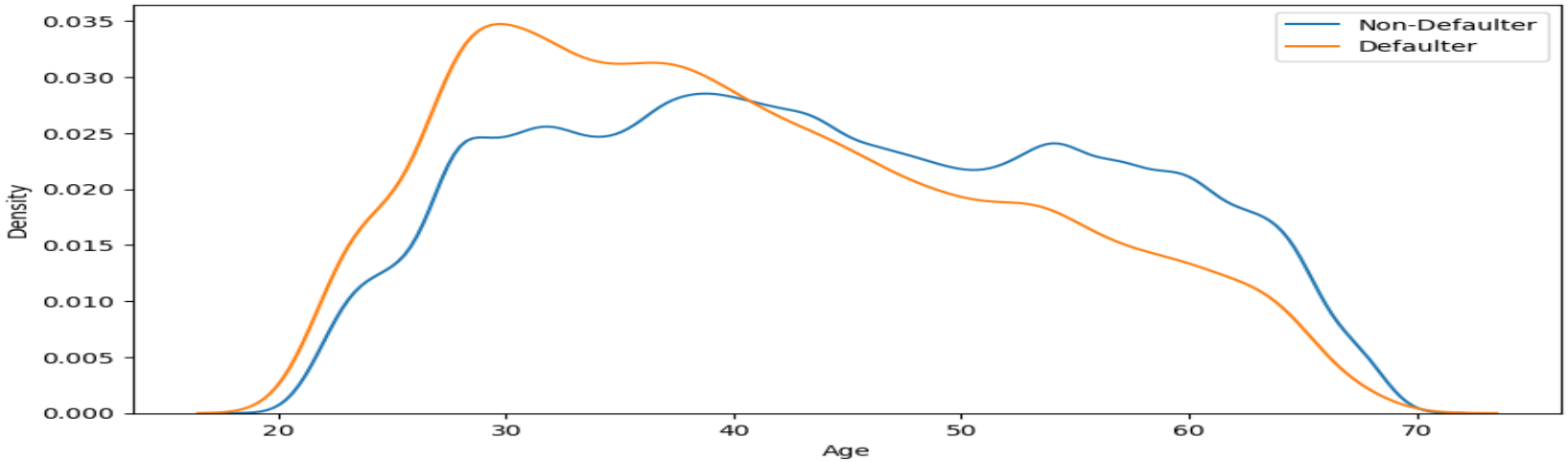
5. DAYS_LAST_PHONE_CHANGE- More people from the Target 1 have changed their phone earlier than Target 1. Indicating intention issues in repaying loan..



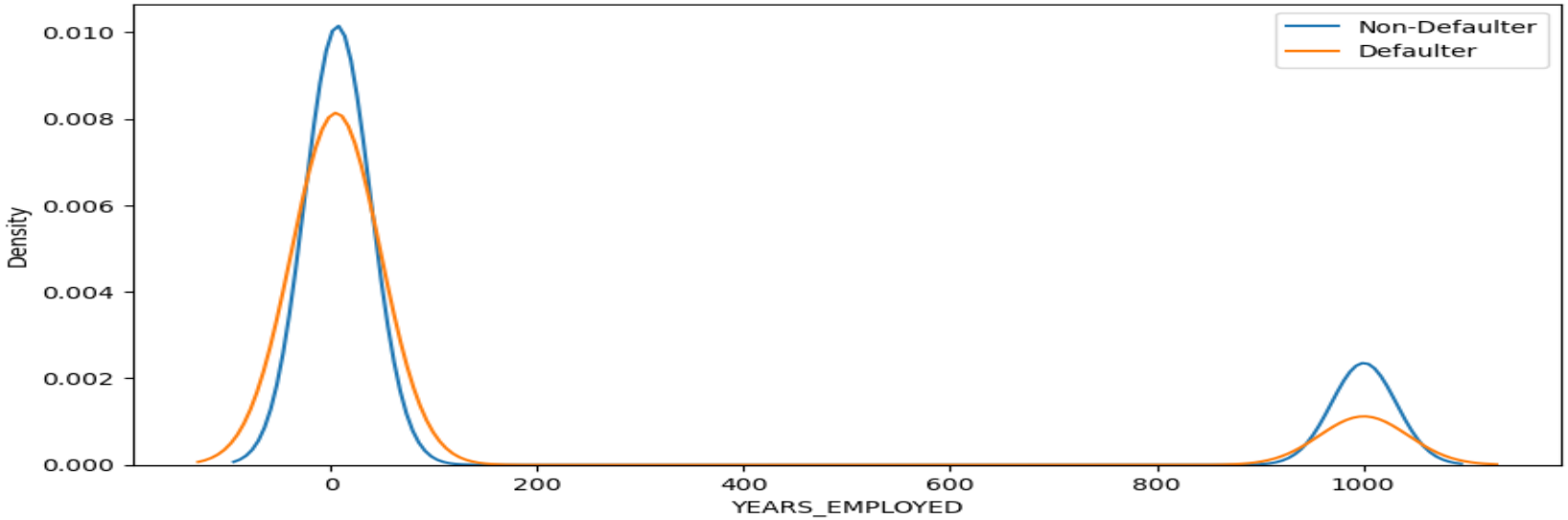
6. AMT_REQ_CREDIT_BUREAU_QRT - Shows clear spike for TARGET 1 is higher than TARGET 0.



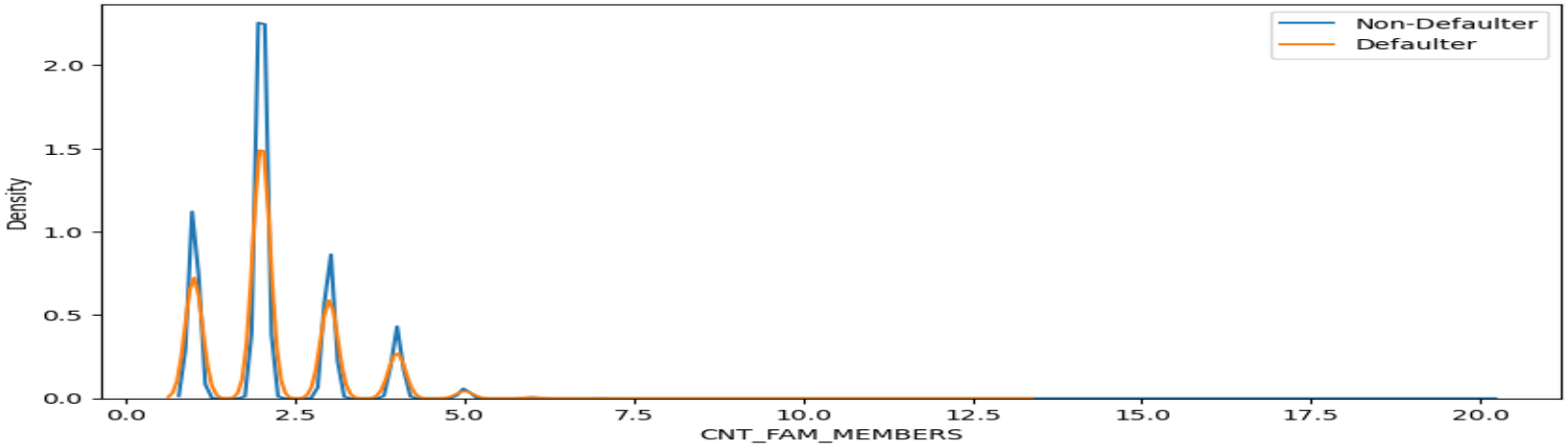
7. AGE_IN_YEARS Density of 30 years in Target 1 larger, indicating younger are defaulting more.



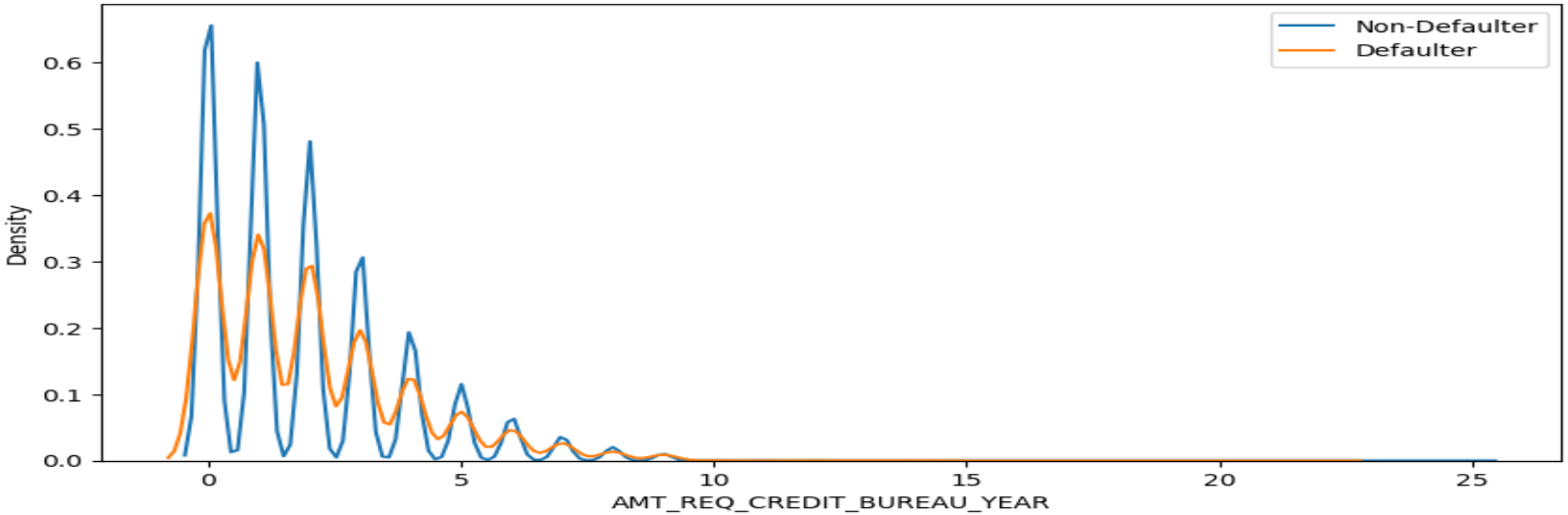
8. YEARS_EMPLOYED has large no of rows of incorrect data and hence the data representation is incorrect.



9. CNT_FAM_MEMBERS - TARGET 1 has on average less count of family members.



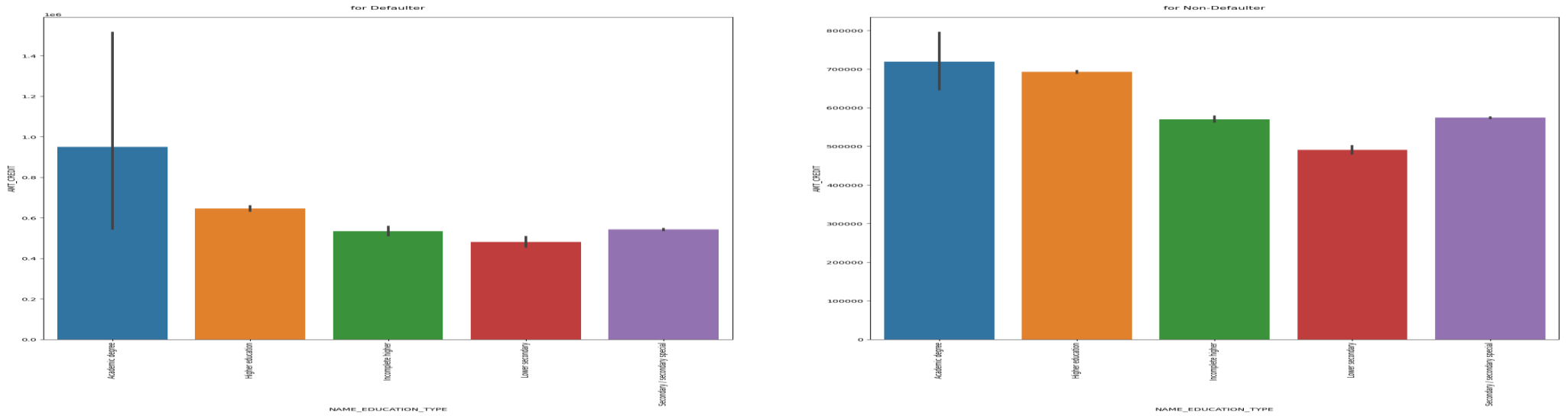
10. AMT_REQ_CREDIT_BUREAU_YEAR, month, week, hour - has lesser Target 1 at 0 hits.This could indicate they are looking getting loans from various financial companies.



After univariate analysis we do bivariate analysis,Bivariate analysis of Categorial and continous variables.
By using a function bivariate_cat_cont(df1,df2,x_col,y_col) it will be easy for us to do analysis.

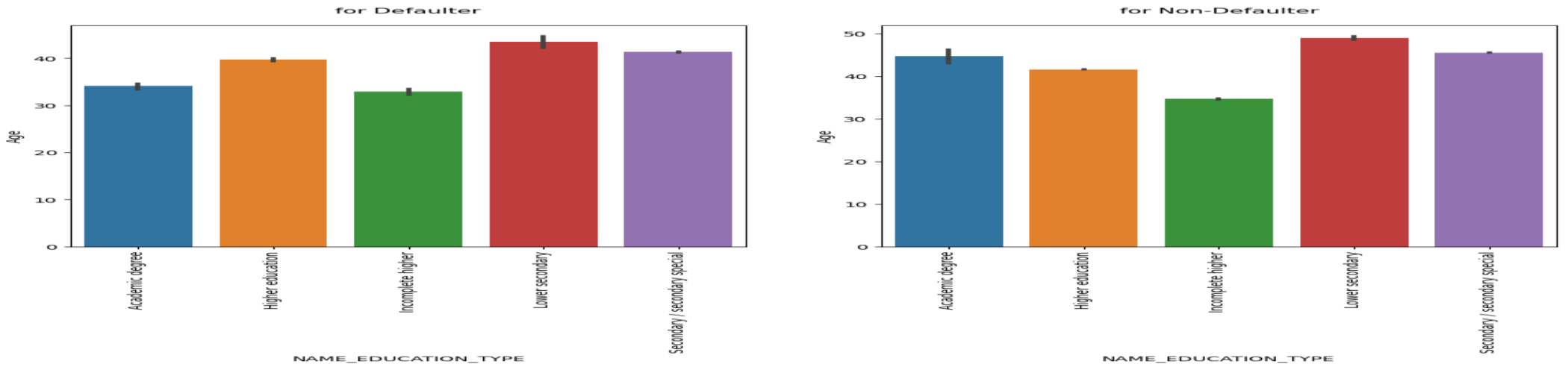
```
# Creating a function for bivariate categorial and continous column
def bivariate_cat_cont(df1,df2,x_col,y_col):
    plt.figure(figsize=[20,12])
    # plotting subplot for defaulters
    plt.subplot(1,2,1)
    plt.title('for Defaulter\n')
    sns.barplot(x=df1[x_col],y=df1[y_col],data=df1)
    plt. xticks(rotation=90)
    # plotting subplot for Non-defaulters
    plt.subplot(1,2,2)
    plt.title('for Non-Defaulter\n')
    sns.barplot(x=df2[x_col],y=df2[y_col],data=df2)
    plt. xticks(rotation=90)
    return  plt.show()
```

```
# Plotting for NAME_EDUCATION_TYPE & AMT_CREDIT
bivariate_cat_cont(df_1,df_0,'NAME_EDUCATION_TYPE','AMT_CREDIT')
```



1. Point which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-
- Academic degree and Higher education category applicants got high Amount Credit.
-. Academic degree and Higher education category applicants are top two in defaulters category.

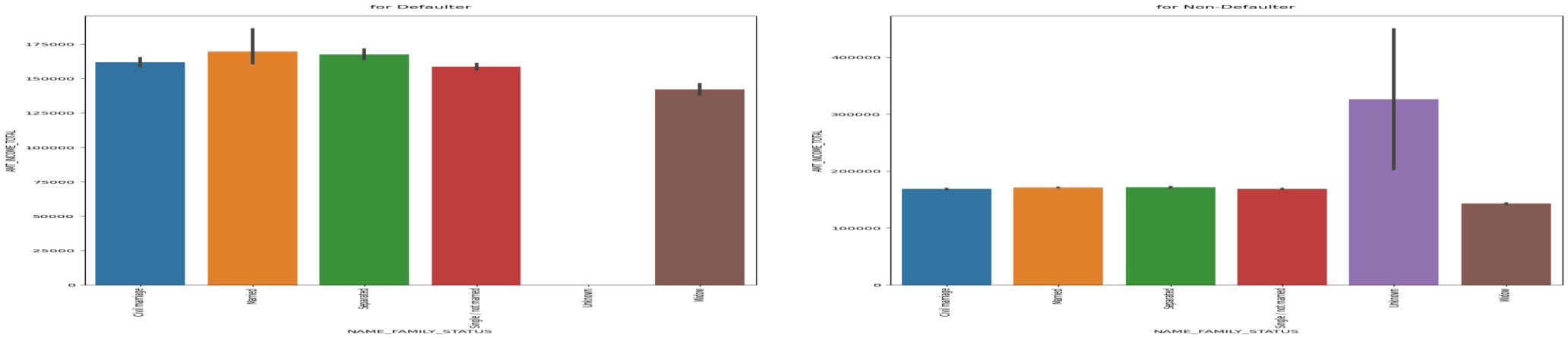
```
# Plotting for NAME_EDUCATION_TYPE & Age
bivariate_cat_cont(df_1,df_0,'NAME_EDUCATION_TYPE','Age')
```



2. Point which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-

- Lower secondary and Secondary/secondary special education category applicants are top defaulters categories

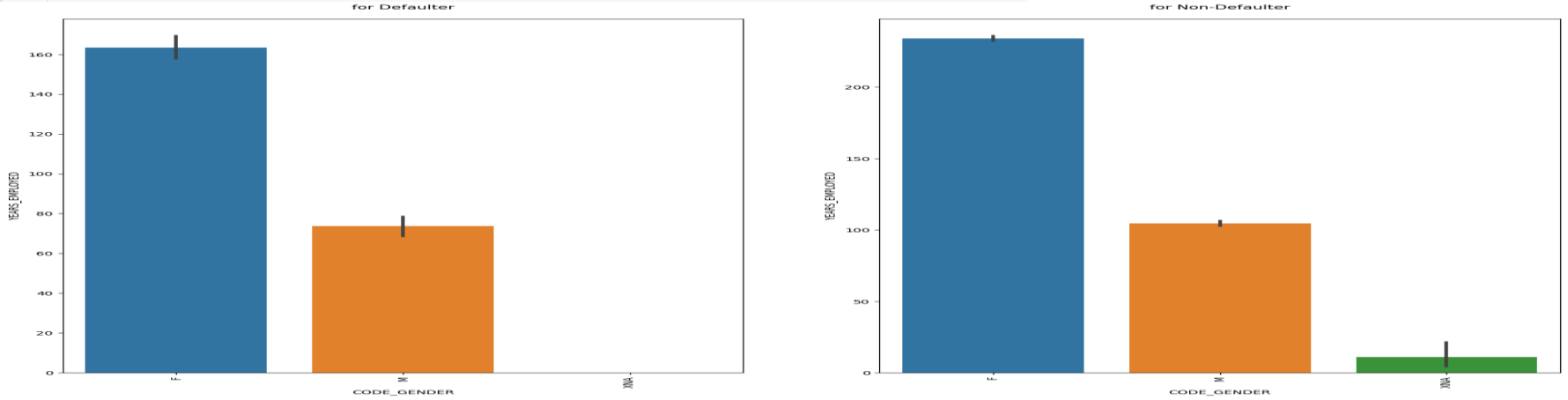
```
# Plotting for NAME_FAMILY_STATUS & AMT_INCOME_TOTAL
bivariate_cat_cont(df_1,df_0,'NAME_FAMILY_STATUS','AMT_INCOME_TOTAL')
```



3. Point which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-

- For Non Defaulters Unknown NAME_FAMILY_STATUS has high AMT_INCOME_TOTAL.
- For Defaulters Civil marriage, Married, Seperated NAME_FAMILY_STATUS are top in 3 category.

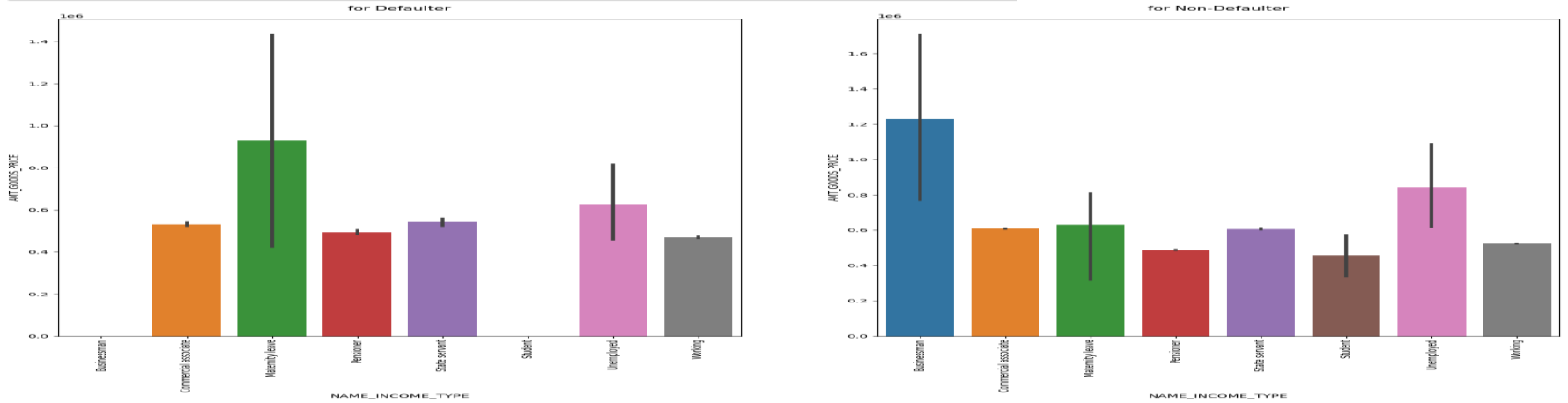
```
# Plotting for CODE_GENDER & YEARS_EMPLOYED
bivariate_cat_cont(df_1,df_0,'CODE_GENDER','YEARS_EMPLOYED')
```



4. Point which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-

- There is strong relation ship for CODE_GENDER and YEARS_EMPLOYED for Defaluters and Non-Defaulters , Defaulter has lesser YEARS_EMPLOYED for both male
- and female than Non-Defaulters

```
# Plotting for NAME_INCOME_TYPE & AMT_GOODS_PRICE
bivariate_cat_cont(df_1,df_0,'NAME_INCOME_TYPE','AMT_GOODS_PRICE')
```



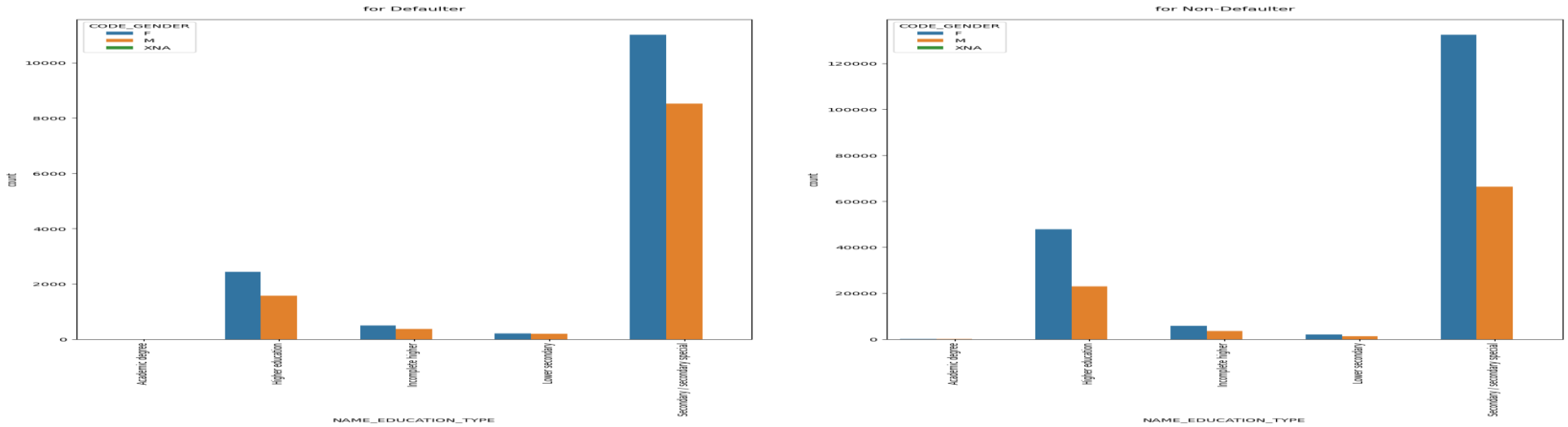
5. Point which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-

- For Defaluters Maternity leave NAME_INCOME_TYPE has high AMT_GOODS_PRICE.
- For Non-Defaulters Businessman NAME_INCOME_TYPE has high AMT_GOODS_PRICE

Bivariate Categorical and Categorical Variables, for this we create a function bivariate_cat_cat(df1,df2,x_col,y_col)

```
# Creating a function for bivariate categorial and categorial column
def bivariate_cat_cat(df1,df2,x_col,y_col):
    plt.figure(figsize=[20,12])
    # plotting subplot for defaulters
    plt.subplot(1,2,1)
    plt.title('for Defaulter\n')
    sns.countplot(x=df1[x_col], hue = df1[y_col], data =df1)
    plt. xticks(rotation=90)
    # plotting subplot for Non-defaulters
    plt.subplot(1,2,2)
    plt.title('for Non-Defaulter\n')
    sns.countplot(x=df2[x_col], hue = df2[y_col], data =df2)
    plt. xticks(rotation=90)
    return  plt.show()
```

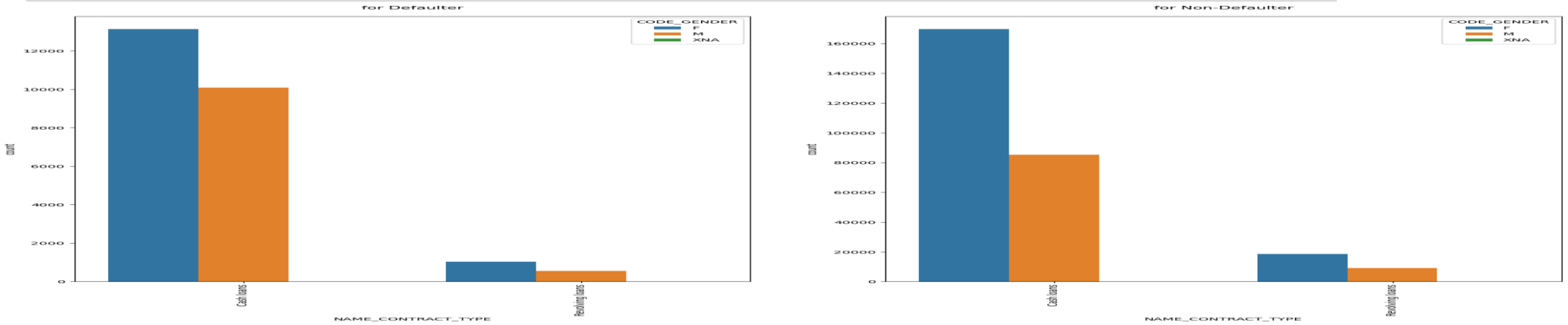
```
# Plotting for NAME_EDUCATION_TYPE & CODE_GENDER
bivariate_cat_cat(df_1,df_0,'NAME_EDUCATION_TYPE','CODE_GENDER')
```



1. **Point which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-**

- Males having Secondary / secondary special as NAME_EDUCATION_TYPE are high in Defaulters category

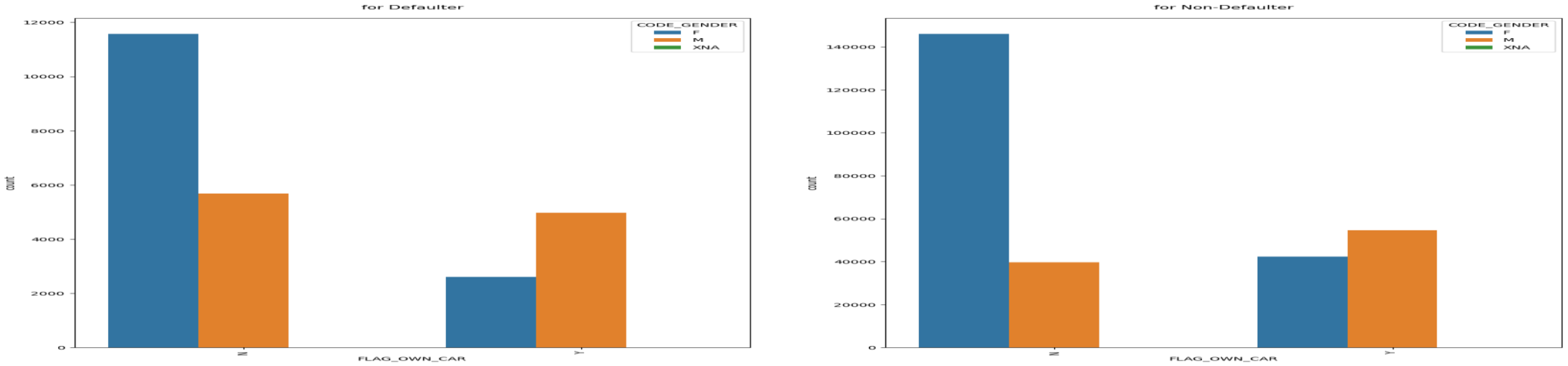
```
# Plotting for NAME_CONTRACT_TYPE & CODE_GENDER
bivariate_cat_cat(df_1,df_0,'NAME_CONTRACT_TYPE','CODE_GENDER')
```



2. Point which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-

- The Ratio of Cash loans and Revolving loans for Defaulter is 14.47 which is quite higher than Non- Defaulters 9.21.

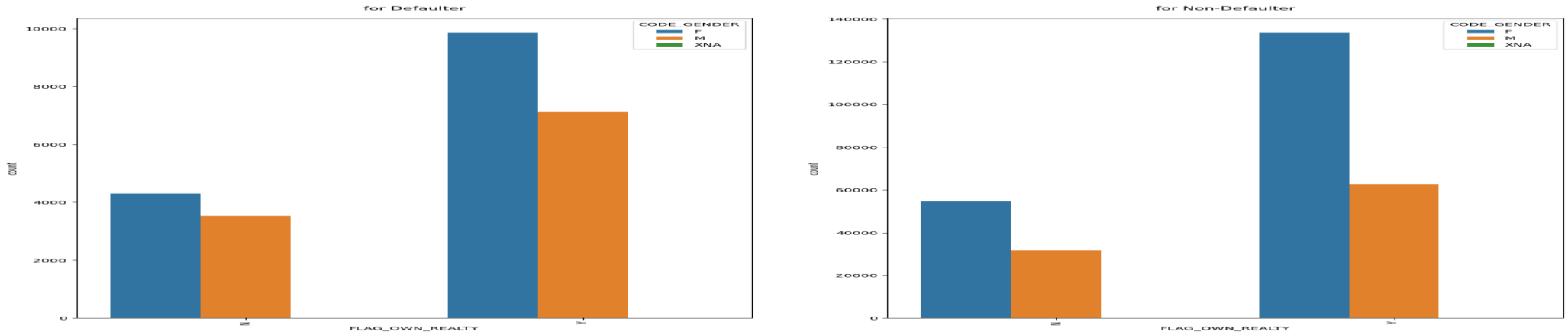
```
# Plotting for FLAG_OWN_CAR & CODE_GENDER
bivariate_cat_cat(df_1,df_0,'FLAG_OWN_CAR','CODE_GENDER')
```



3.Point which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-

- Males and Females for Defaulters is having less car owners percent as compare to than Non- Defaulters.

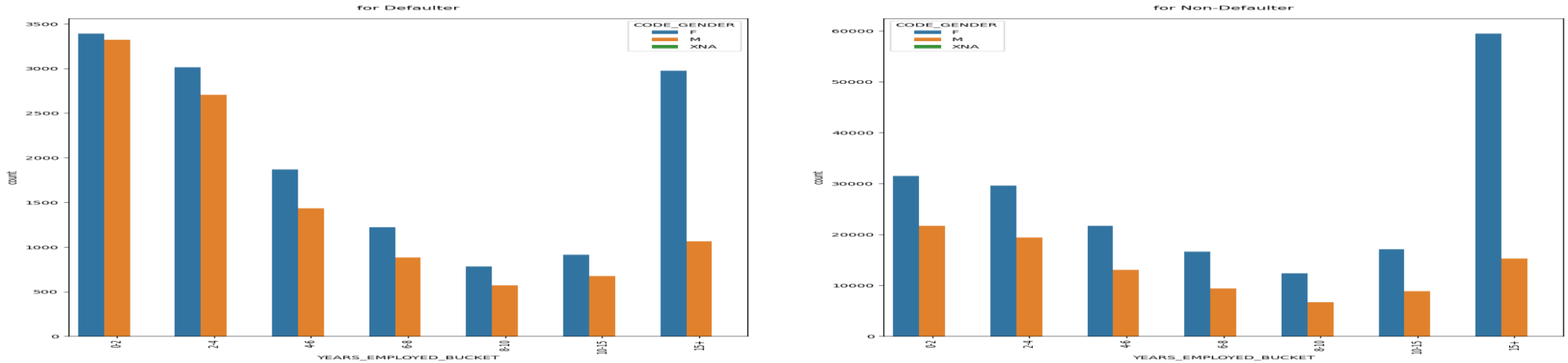
```
# Plotting for FLAG_OWN_REALTY & CODE_GENDER
bivariate_cat_cat(df_1,df_0,'FLAG_OWN_REALTY','CODE_GENDER')
```



4. Point which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-

- Males and Females for Defaulters is having less difference in percent of owning FLAG_OWN_REALTY as compare to than Non- Defaulters

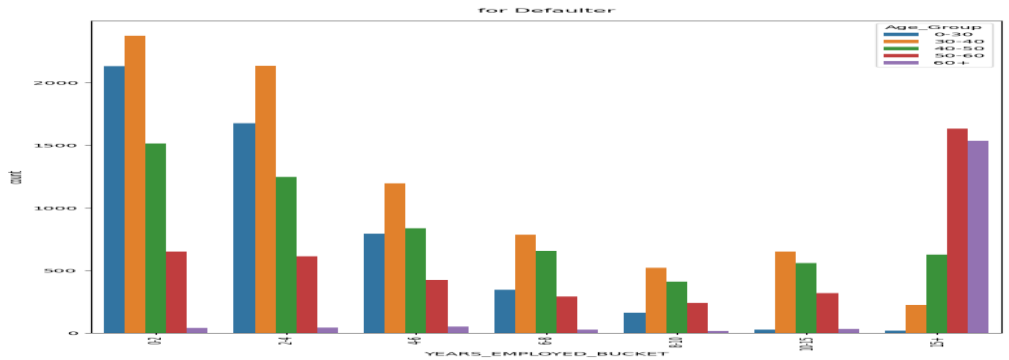
```
# Plotting for YEARS_EMPLOYED_BUCKET & CODE_GENDER
bivariate_cat_cat(df_1,df_0,'YEARS_EMPLOYED_BUCKET','CODE_GENDER')
```

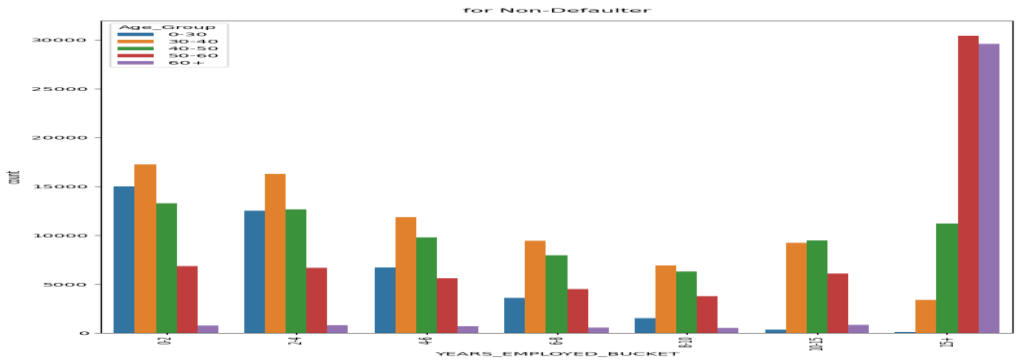


5.Point which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-

- Males and Females for Defaulters is having less YEAR_EMPLOYED_BUCKET are clear more defaulters except some cases of 15+ as compare to than Non- Defaulters.

```
# Plotting for YEARS_EMPLOYED_BUCKET & Age_Group
bivariate_cat_cat(df_1,df_0,'YEARS_EMPLOYED_BUCKET','Age_Group')
```

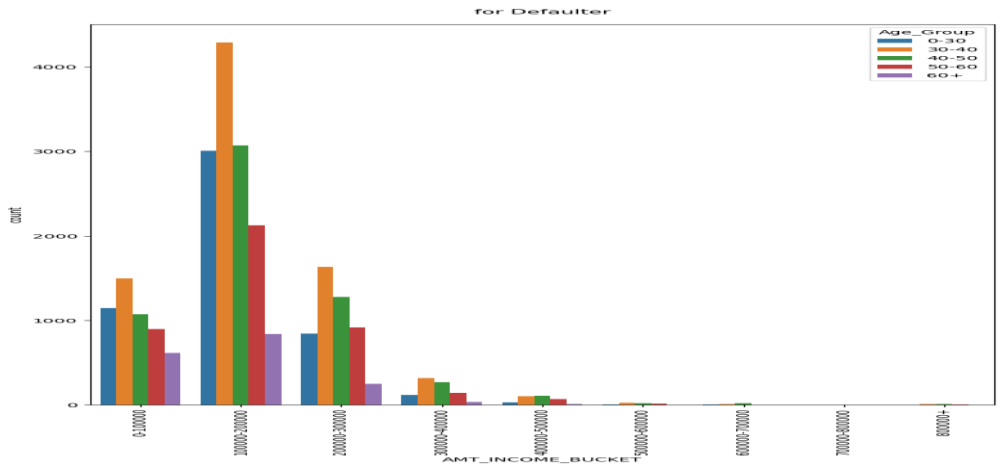


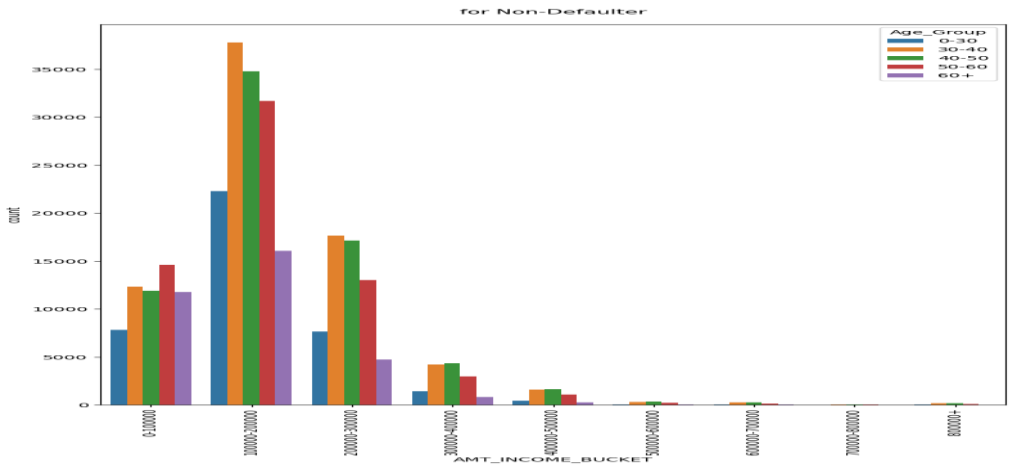


6. Point which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-

- For Defaulters YEARS_EMPLOYED_BUCKET 0-2,2-4,4-6 and Age_Group 30-40,0-30,40-50 is are clear more defaulters except some cases of YEARS_EMPLOYED_BUCKET 15+ and Age_Group 50-60,60+ as compare to than Non- Defaulters.

```
# Plotting for AMT_INCOME_BUCKET & Age_Group
bivariate_cat_cat(df_1,df_0,'AMT_INCOME_BUCKET','Age_Group')
```



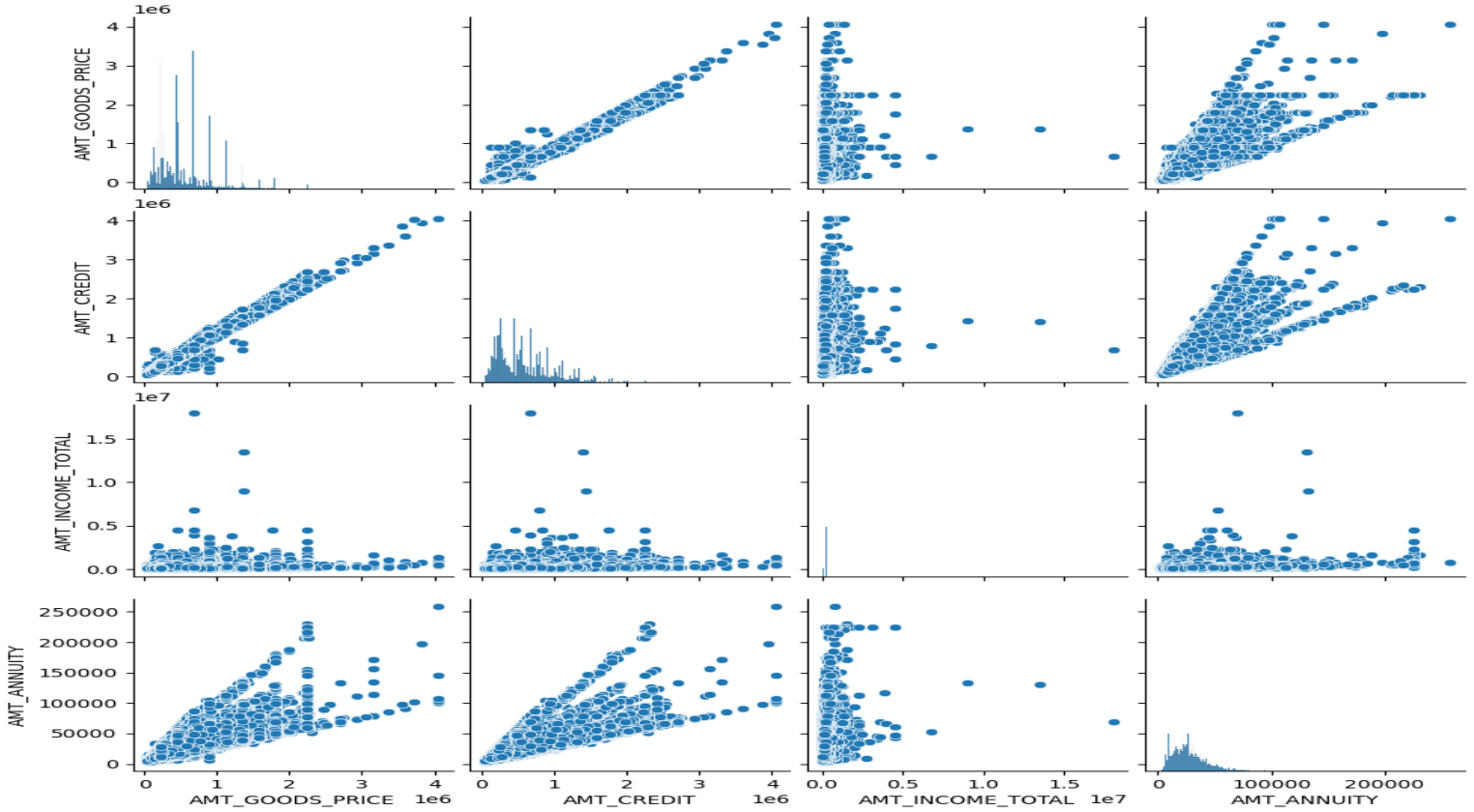


7. Point which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-

- For Defaulters AMT_INCOME_BUCKET range 100000-200000 with all Age_Group are clear more defaulters as compare to than Non- Defaulters beacuse most applicants
- are from these range.

Bivariate Continous and Continous Variables, Firslty plotting a pair plot for non defaulters

```
# plotting a pair plot for non defaulters df_0 for columns AMT_GOODS_PRICE,AMT_CREDIT,AMT_INCOME_TOTAL,AMT_ANNUITY
plt.figure(figsize=[18,7])
sns.pairplot(df_0[['AMT_GOODS_PRICE','AMT_CREDIT','AMT_INCOME_TOTAL','AMT_ANNUITY']])
plt.show()
```

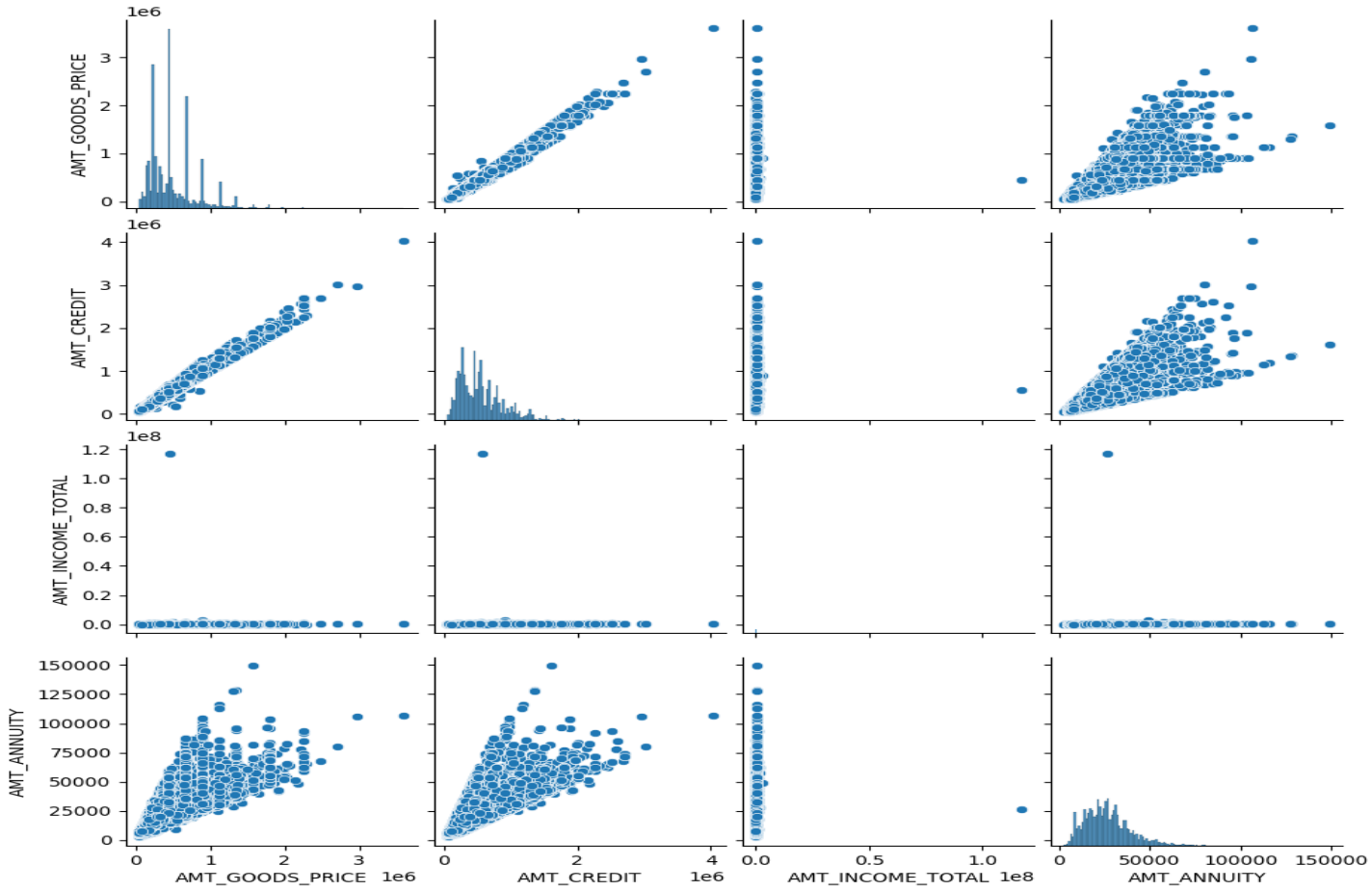


Few points which are noted for non-defaulters are as follows:-

1. AMT_GOODS_PRICE - has high correlation with AMT_CREDIT,AMT_ANNUITY
2. AMT_CREDIT- has high correlation with AMT_ANNUITY.

Bivariate Continous and Continous Variables, plotting a pair plot for defaulters

```
# plotting a pair plot for defaulter df_0 for columns AMT_GOODS_PRICE,AMT_CREDIT,AMT_INCOME_TOTAL,AMT_ANNUITY
plt.figure(figsize=[18,7])
sns.pairplot(df_1[['AMT_GOODS_PRICE','AMT_CREDIT','AMT_INCOME_TOTAL','AMT_ANNUITY']])
plt.show()
```



Few points which are noted for defaulters are as follows:-

1. AMT_GOODS_PRICE - has high correlation with AMT_CREDIT,AMT_ANNUITY
2. AMT_CREDIT- has high correlation with AMT_ANNUITY.

Now we make correlation matrix for df_0 i.e non-defaulters and df_1 i.e. defaulters and on basis of correlation matrix we plot heat map Which is providing correlation with graph along with annotation between different variables for dataframe.

First we will make a variable which have various columns of dataframe which we will utilize to have correlation.

```
# bisecting the df dataframe based on Target value 0 and 1 for correlation and other analysis
cols_for_corr = ['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_REALTY',
                 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY',
                 'AMT_GOODS_PRICE','NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
                 'NAME_FAMILY_STATUS','NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
                 'DAYS_EMPLOYED','DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'OCCUPATION_TYPE',
                 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT','REGION_RATING_CLIENT_W_CITY',
                 'WEEKDAY_APPR_PROCESS_START','HOUR_APPR_PROCESS_START','REG_REGION_NOT_LIVE_REGION',
                 'REG_REGION_NOT_WORK_REGION','LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
                 'REG_CITY_NOT_WORK_CITY','LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_TYPE',
                 'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE','DAYS_LAST_PHONE_CHANGE',
                 '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']


# Defaulters dataframe
df_1 = df.loc[df['TARGET']==1, cols_for_corr]

# Non-Defaulters dataframe
df_0 = df.loc[df['TARGET']==0, cols_for_corr]

```

```
len(cols_for_corr)
```

```
40
```


Correlation between continous variable

```
# Getting  top 10 correlation for the Non-Defaulters dataframe
corr_non_defaulters = df_0.corr(numeric_only=True)
corr_non_defaulters = corr_non_defaulters.where(np.triu(np.ones(corr_non_defaulters.shape),k=1).astype(bool)).unstack().rese
corr_non_defaulters.columns =['VARIABLE1','VARIABLE2','Correlation']
corr_non_defaulters.dropna(subset = ["Correlation"], inplace = True)
corr_non_defaulters["Correlation"]=corr_non_defaulters["Correlation"]
corr_non_defaulters.sort_values(by='Correlation', ascending=False, inplace=True)
corr_non_defaulters.head(10)

```

	VARIABLE1	VARIABLE2	Correlation
61	AMT_GOODS_PRICE	AMT_CREDIT	0.987250
62	AMT_GOODS_PRICE	AMT_ANNUITY	0.776686
41	AMT_ANNUITY	AMT_CREDIT	0.771309
125	DAYS_EMPLOYED	DAYS_BIRTH	0.626114
40	AMT_ANNUITY	AMT_INCOME_TOTAL	0.418953
60	AMT_GOODS_PRICE	AMT_INCOME_TOTAL	0.349462
20	AMT_CREDIT	AMT_INCOME_TOTAL	0.342799
145	DAYS_REGISTRATION	DAYS_BIRTH	0.333151
166	DAYS_ID_PUBLISH	DAYS_EMPLOYED	0.276663
165	DAYS_ID_PUBLISH	DAYS_BIRTH	0.271314

From above correlation data we got to know that for non-deafulters

1. Top three high correaltion are:

- Amount goods price and amount credit = 0.987,Amount goods price and amount annunity = 0.776,Amount Annuity and amount credit = 0.771

2. Bottom two low correaltion are:

- Days ID publish and Days Employed = 0.276, Days ID publish and Days Birth = 0.229

```
# Getting  top 10 correlation for the Defaulters dataframe
corr_defaulters = df_1.corr(numeric_only=True)
corr_defaulters = corr_defaulters.where(np.triu(np.ones(corr_defaulters.shape),k=1).astype(bool)).unstack().reset_index()
corr_defaulters.columns =['VARIABLE1','VARIABLE2','Correlation']
corr_defaulters.dropna(subset = ["Correlation"], inplace = True)
corr_defaulters["Correlation"]=corr_defaulters["Correlation"]
corr_defaulters.sort_values(by='Correlation', ascending=False, inplace=True)
corr_defaulters.head(10)
```

	VARIABLE1	VARIABLE2	Correlation
61	AMT_GOODS_PRICE	AMT_CREDIT	0.983103
62	AMT_GOODS_PRICE	AMT_ANNUITY	0.752699
41	AMT_ANNUITY	AMT_CREDIT	0.752195
125	DAYS_EMPLOYED	DAYS_BIRTH	0.582185
145	DAYS_REGISTRATION	DAYS_BIRTH	0.289114
251	DEF_60_CNT_SOCIAL_CIRCLE	OBS_60_CNT_SOCIAL_CIRCLE	0.264159
165	DAYS_ID_PUBLISH	DAYS_BIRTH	0.252863
314	AMT_REQ_CREDIT_BUREAU_DAY	AMT_REQ_CREDIT_BUREAU_HOUR	0.246741
166	DAYS_ID_PUBLISH	DAYS_EMPLOYED	0.229090
146	DAYS_REGISTRATION	DAYS_EMPLOYED	0.192455

From above correlation data we got to know that for defaulters

1. Top three high correaltion are:

- Amount goods price and amount credit = 0.983,Amount goods price and amount annunity = 0.753, Amount Annuity and amount credit = 0.752

2. Bottom two low correaltion are:

- Days Registration and Days Employed = 0.192,Days ID publish and Days Employed = 0.229

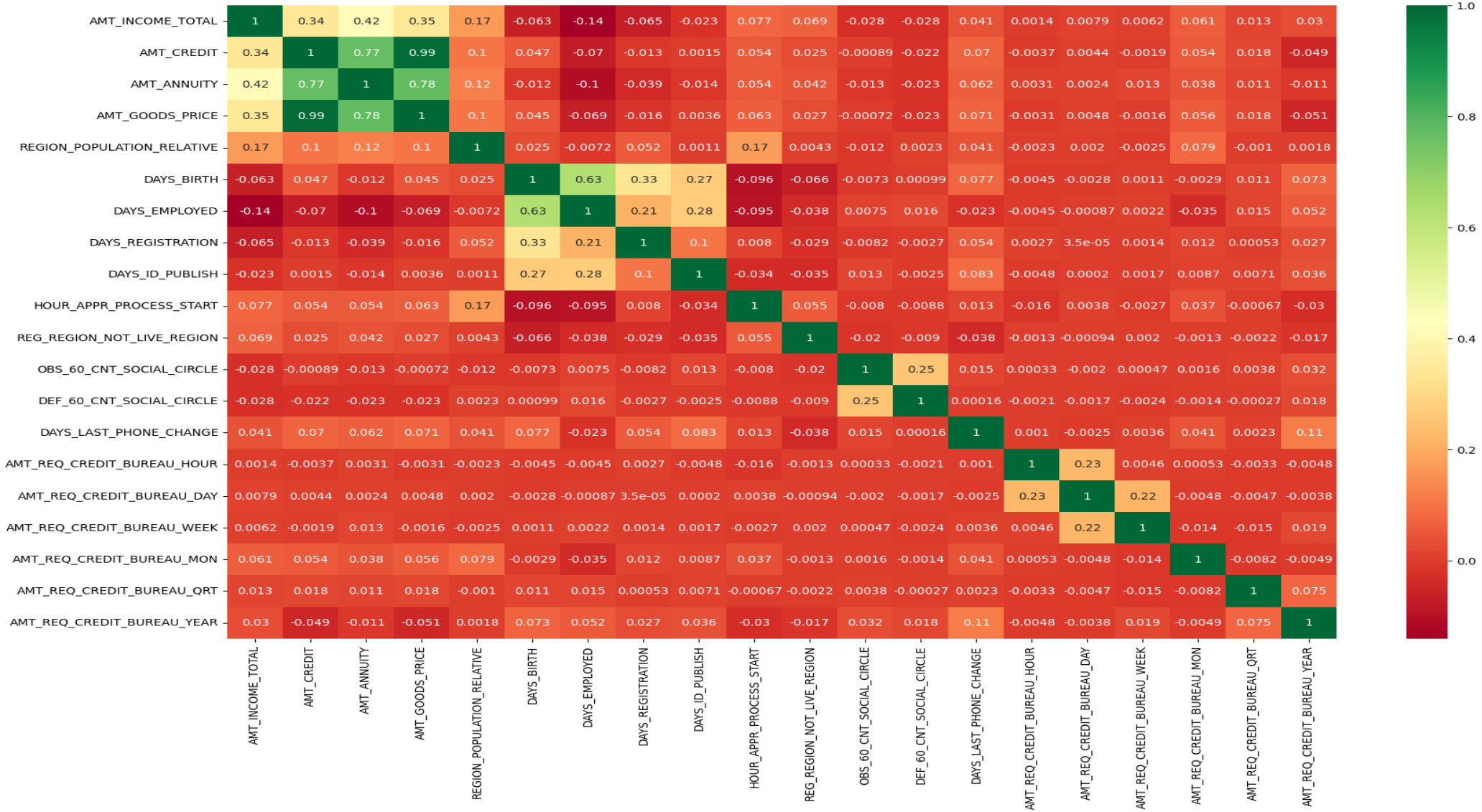
Creating correlation table for non defaulters with certain imp columns.

```
# # Creating correlation table for non defaulters with certain imp columns
df_0.corr(numeric_only=True)
```

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_POPULATION_RELATIVE	DAYS_BIRTH	D
AMT_INCOME_TOTAL	1.000000	0.342799	0.418953	0.349462	0.167851	-0.062609	
AMT_CREDIT	0.342799	1.000000	0.771309	0.987250	0.100604	0.047378	
AMT_ANNUITY	0.418953	0.771309	1.000000	0.776686	0.120988	-0.012263	
AMT_GOODS_PRICE	0.349462	0.987250	0.776686	1.000000	0.103827	0.044565	
REGION_POPULATION_RELATIVE	0.167851	0.100604	0.120988	0.103827	1.000000	0.025244	
DAYS_BIRTH	-0.062609	0.047378	-0.012263	0.044565	0.025244	1.000000	
DAYS_EMPLOYED	-0.140392	-0.070104	-0.104978	-0.068609	-0.007198	0.626114	
DAYS_REGISTRATION	-0.064937	-0.013477	-0.039436	-0.015916	0.052083	0.333151	
DAYS_ID_PUBLISH	-0.022896	0.001464	-0.014113	0.003649	0.001071	0.271314	
HOUR_APPR_PROCESS_START	0.076743	0.053619	0.053589	0.062766	0.172814	-0.095916	
REG_REGION_NOT_LIVE_REGION	0.068510	0.024617	0.041992	0.026731	0.004306	-0.066252	
OBS_60_CNT_SOCIAL_CIRCLE	-0.027690	-0.000892	-0.012893	-0.000723	-0.011591	-0.007316	
DEF_60_CNT_SOCIAL_CIRCLE	-0.027593	-0.022225	-0.023385	-0.023171	0.002255	0.000990	
DAYS_LAST_PHONE_CHANGE	0.041338	0.069540	0.062013	0.071373	0.041164	0.076510	
AMT_REQ_CREDIT_BUREAU_HOUR	0.001417	-0.003734	0.003148	-0.003116	-0.002265	-0.004461	
AMT_REQ_CREDIT_BUREAU_DAY	0.007862	0.004409	0.002392	0.004820	0.001969	-0.002772	
AMT_REQ_CREDIT_BUREAU_WEEK	0.006234	-0.001883	0.012681	-0.001597	-0.002480	0.001069	
AMT_REQ_CREDIT_BUREAU_MON	0.061470	0.054071	0.038184	0.055850	0.078630	0.002855	

Generating a heat map for non defaulters on basis of correlation

```
# Creating a heat map for non defaulters on basis of correaltion variables
plt.figure(figsize=[20,12])
sns.heatmap(data=df_0.corr(numeric_only=True),annot= True,cmap='RdYlGn')
plt.show()
```



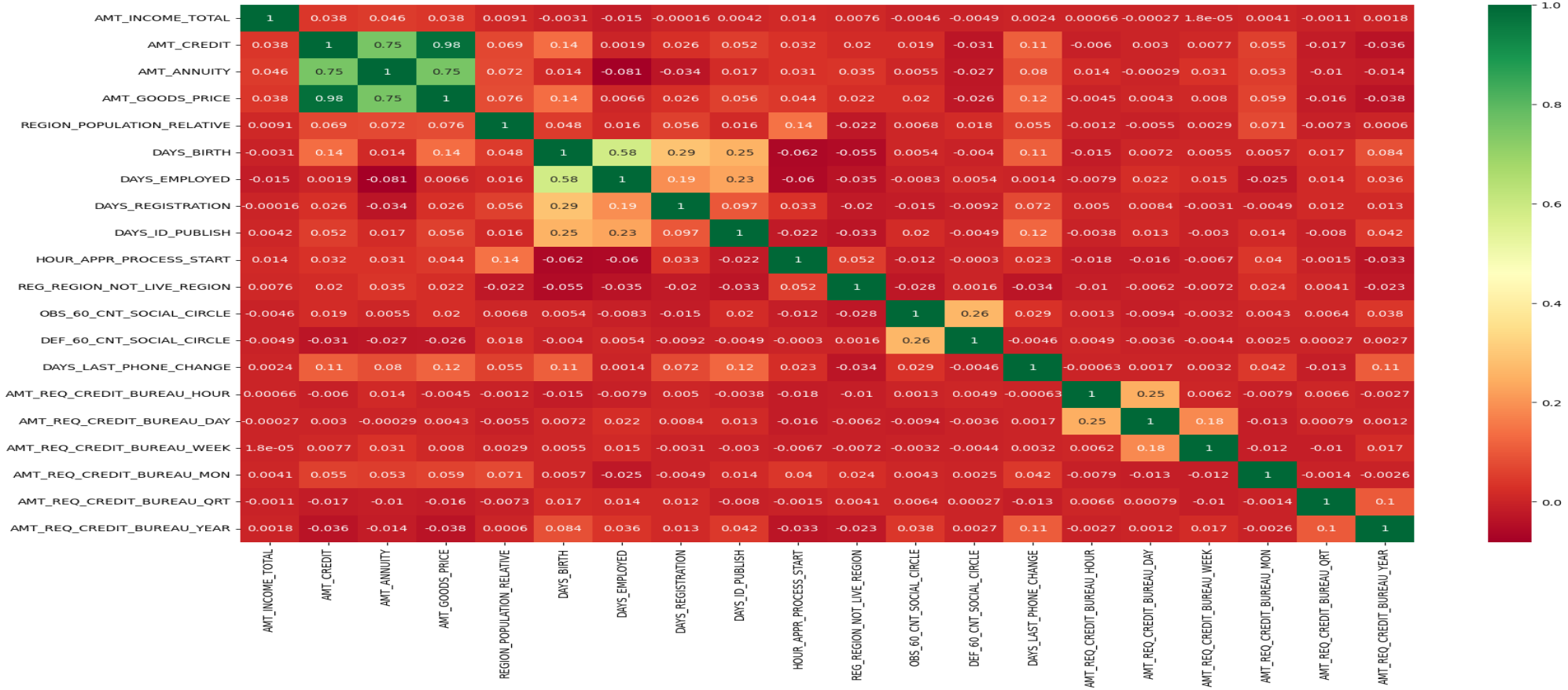
Creating correlation table for non defaulters with certain imp columns.

```
# # Creating correlation table for defaulters with certain imp columns
df_1.corr(numeric_only=True)
```

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_POPULATION_RELATIVE	DAYS_BIRTH	D
AMT_INCOME_TOTAL	1.000000	0.038131	0.046421	0.037583	0.009135	-0.003096	
AMT_CREDIT	0.038131	1.000000	0.752195	0.983103	0.069161	0.135316	
AMT_ANNUITY	0.046421	0.752195	1.000000	0.752699	0.071690	0.014303	
AMT_GOODS_PRICE	0.037583	0.983103	0.752699	1.000000	0.076049	0.135810	
REGION_POPULATION_RELATIVE	0.009135	0.069161	0.071690	0.076049	1.000000	0.048190	
DAYS_BIRTH	-0.003096	0.135316	0.014303	0.135810	0.048190	1.000000	
DAYS_EMPLOYED	-0.014977	0.001930	-0.081207	0.006642	0.015532	0.582185	
DAYS_REGISTRATION	-0.000158	0.025854	-0.034279	0.025679	0.056222	0.289114	
DAYS_ID_PUBLISH	0.004215	0.052329	0.016767	0.056086	0.015537	0.252863	
HOUR_APPR_PROCESS_START	0.013775	0.031782	0.031236	0.044315	0.142744	-0.062172	
REG_REGION_NOT_LIVE_REGION	0.007577	0.019540	0.034807	0.022392	-0.022129	-0.054654	
OBS_60_CNT_SOCIAL_CIRCLE	-0.004616	0.019487	0.005500	0.020385	0.006793	0.005391	
DEF_60_CNT_SOCIAL_CIRCLE	-0.004866	-0.030880	-0.027495	-0.026179	0.018231	-0.004001	
DAYS_LAST_PHONE_CHANGE	0.002429	0.110851	0.079870	0.118303	0.055139	0.111199	
AMT_REQ_CREDIT_BUREAU_HOUR	0.000656	-0.005981	0.014039	-0.004496	-0.001238	-0.014654	
AMT_REQ_CREDIT_BUREAU_DAY	-0.000272	0.003008	-0.000294	0.004280	-0.005483	0.007248	
AMT_REQ_CREDIT_BUREAU_WEEK	0.000018	0.007650	0.031242	0.007980	0.002904	0.005537	

Generating a heat map for defaulters on basis of correlation

```
# Creating a heat map for defaulters on basis of correaltion variables
plt.figure(figsize=[20,12])
ax=sns.heatmap(data=df_1.corr(numeric_only=True),annot= True,cmap='RdYlGn')
plt.show()
```



Insights after comparing correlation matrix and heat maps for non defaulters and defaulters :-
Credit amount is highly correlated with good price amount which is same as non defaulters

1. Loan annuity correlation with credit amount has slightly reduced in defaulters(0.75) when compared to non defaulters(0.77).
2. We can also see that non defaulters have high correlation in number of days employed(0.63) when compared defaulters(0.58).
3. There is a severe drop in the correlation between total income of the client and the credit amount(0.038) amongst defaulters whereas it is 0.342 among non defaulters.
4. There is a slight increase in defaulted to observed count in social circle among defaulters(0.26) when compared to non defaulters(0.25).

Now we upload previous application data .

```
# Importing dataset previous_application.csv
df_prev = pd.read_csv('previous_application.csv',encoding='utf-8')
```

Next Step is to do the Routine Structure Check of Application data by the use of following python function :-

1. info
2. describe
3. shape
4. dtype
5.columns
6. head
7. tail
8. unique
9. nunique
10. value_counts

```
# Checking shape i.e. number of rows and columns of dataset
df_prev.shape
```

```
(1670214, 37)
```

```
# Checking information of dataset
df_prev.info('all')
```

```
# Displaying head of dataset
df_prev.head()
```

```
# Displaying tail of dataset
df_prev.tail()
```

```
# Checking description of dataset
df_prev.describe()
```

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	HOUR_APPR_PROCESS_STA
count	1.670214e+06	1.670214e+06	1.297979e+06	1.670214e+06	1.670213e+06	7.743700e+05	1.284699e+06	1.670214e+
mean	1.923089e+06	2.783572e+05	1.595512e+04	1.752339e+05	1.961140e+05	6.697402e+03	2.278473e+05	1.248418e+
std	5.325980e+05	1.028148e+05	1.478214e+04	2.927798e+05	3.185746e+05	2.092150e+04	3.153966e+05	3.334028e+
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00	0.000000e+00	-9.000000e-01	0.000000e+00	0.000000e+
25%	1.461857e+06	1.893290e+05	6.321780e+03	1.872000e+04	2.416050e+04	0.000000e+00	5.084100e+04	1.000000e+
50%	1.923110e+06	2.787145e+05	1.125000e+04	7.104600e+04	8.054100e+04	1.638000e+03	1.123200e+05	1.200000e+
75%	2.384280e+06	3.675140e+05	2.065842e+04	1.803600e+05	2.164185e+05	7.740000e+03	2.340000e+05	1.500000e+
max	2.845382e+06	4.562550e+05	4.180581e+05	6.905160e+06	6.905160e+06	3.060045e+06	6.905160e+06	2.300000e+

After doing routine structure check we do data quality check in following steps :-

Step 1:-Find the percentage of Missing Values in DataFrame by using null_cal function we have already created

```
# Checking null values percentage of columns of df_prev
null_cal(df_prev)
```

Step 2- Remove Columns with High Missing Percentage

```
# Setting column cut off percentage to 50 and finding names of columns with such percentage
cols_to_drop_per=col_having_high_null_val(df_prev,50)
cols_to_drop_per
```

```
['AMT_DOWN_PAYMENT',
 'RATE_DOWN_PAYMENT',
 'RATE_INTEREST_PRIMARY',
 'RATE_INTEREST_PRIVILEGED']
```

```
# Dropping columns with null percentage >= 50
df_prev = df_prev.drop(cols_to_drop_per,axis=1)
```

Step 3- Checking the shape after deleting high null percentage columns and finding Columns from DataFrame whose null % and we have to impute these null values

```
# Checking shape of df_prev after removal of null values columns
df_prev.shape
```

```
(1670214, 33)
```

```
# percent null values of columns after removal of high percent null value columns
null_cal(df_prev)
```

```
SK_ID_PREV                    0.000000
SK_ID_CURR                    0.000000
```

Following are the 5 columns i have shortlisted to impute values :-

1. AMT_ANNUITY = 22.286665
2. AMT_GOODS_PRICE = 23.081773
3. NAME_TYPE_SUITE = 49.119754
4. CNT_PAYMENT = 22.2863665
5. PRODUCT_COMBINATION = 0.020716

Missing Value Strategy:-

1. For numerical/continuous we can use mean or median, median is preferred as it is unaffected by extreme values.
2. For categorical/discrete columns we can use mode for imputing the missing values.

```
# Checking missing values for column AMT_ANNUITY
missing_val_func_num(df_prev,'AMT_ANNUITY')
```

```
count    1.297979e+06
mean     1.595512e+04
std      1.478214e+04
min      0.000000e+00
25%      6.321780e+03
50%      1.125000e+04
75%      2.065842e+04
max      4.180581e+05
Name: AMT_ANNUITY, dtype: float64
Median  11250.0
Quantile 0.9    34703.82
Name: AMT_ANNUITY, dtype: float64
Quantile 0.95    45336.78
Name: AMT_ANNUITY, dtype: float64
Quantile 0.99    69685.7886
Name: AMT_ANNUITY, dtype: float64
```

In AMT_ANNUITY we use median to impute missing values because there is huge difference between 99 percentile and max value which can cause mean to deviate.

1. In AMT_ANNUITY we use median to impute missing values because there is huge difference between 99 percentile and max value which can cause mean to deviate**.**
2. In NAME_TYPE_SUITE highest value count category is Unaccompanied so we can impute Unaccompanied in place of null values.
3. In AMT_GOODS_PRICE we use median to impute missing values because there is huge difference between 99 percentile and max value which can cause mean to deviate**.**
4. In CNT_PAYMENT we use IQR method to impute missing values because there is large difference between 99 percentile and max value which can cause mean to deviate**.**
5. In PRODUCT_COMBINATION we can impute Cash in place of missing values**.**

Step 4 - Check the datatypes of all the columns and change the Datatype also check anomalies like negative age in data.

```
# Check the datatypes of all the columns of df_prev
df_prev.dtypes
```

```
# Checking columns having negative values
df_prev.head(10)
```

Columns having negative values are as follows:-
1. DAYS_DECISION,
2. DAYS_FIRST_DRAWING,
3. DAYS_FIRST_DUE,
4. DAYS_LAST_DUE_1ST_VERSION,
5. DAYS_LAST_DUE
6. DAYS_TERMINATION

```
# To convert columns have negative values to postive values creating a variable "cols_to_pos"
cols_to_pos = ['DAYS_DECISION','DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION',
               'DAYS_LAST_DUE', 'DAYS_TERMINATION','SELLERPLACE_AREA']
```

```
# Converting columns having negative values to positive values by using abs() and imputing values in df_prev
for i in cols_to_pos:
    df_prev[i] = abs(df_prev[i])
```

```
# Checking the negative values are converted to positive values
df_prev.head()
```

Columns to convert from Object dtypes to Categorial dtypes

1. NAME_CONTRACT_TYPE
2. NAME_CASH_LOAN_PURPOSE
3. NAME_CONTRACT_STATUS
4. NAME_PAYMENT_TYPE
5. CODE_REJECT_REASON
6. NAME_CLIENT_TYPE
7. NAME_GOODS_CATEGORY
8. NAME_PORTFOLIO
9. NAME_PRODUCT_TYPE
10. CHANNEL_TYPE
11. NAME_SELLER_INDUSTRY
12. NAME_YIELD_GROUP
13. PRODUCT_COMBINATION

```
#Converting required categoical columns from object dtype to categorical dtype
prev_cat_col = ['NAME_CONTRACT_TYPE','NAME_CASH_LOAN_PURPOSE','NAME_CONTRACT_STATUS','NAME_PAYMENT_TYPE',
                    'CODE_REJECT_REASON','NAME_CLIENT_TYPE','NAME_GOODS_CATEGORY','NAME_PORTFOLIO',
                   'NAME_PRODUCT_TYPE','CHANNEL_TYPE','NAME_SELLER_INDUSTRY','NAME_YIELD_GROUP','PRODUCT_COMBINATION']

for i in prev_cat_col:
    df_prev[i] =pd.Categorical(df_prev[i])
```

```
# Checking conversion of categorical dtypes from object
df_prev.dtypes
```


Step 5- Finding Columns where we identified Presence of Outliers

```
# Using describe function to find out which columns has outliers (i.e. difference between 75% and max is large )
df_prev.describe()
```

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_GOODS_PRICE	HOUR_APPR_PROCESS_START	NFLAG_LAST_APPL
count	1.670214e+06	1.670214e+06	1.297979e+06	1.670214e+06	1.670213e+06	1.284699e+06	1.670214e+06	1.67
mean	1.923089e+06	2.783572e+05	1.595512e+04	1.752339e+05	1.961140e+05	2.278473e+05	1.248418e+01	9.96
std	5.325980e+05	1.028148e+05	1.478214e+04	2.927798e+05	3.185746e+05	3.153966e+05	3.334028e+00	5.93
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.00
25%	1.461857e+06	1.893290e+05	6.321780e+03	1.872000e+04	2.416050e+04	5.084100e+04	1.000000e+01	1.00
50%	1.923110e+06	2.787145e+05	1.125000e+04	7.104600e+04	8.054100e+04	1.123200e+05	1.200000e+01	1.00
75%	2.384280e+06	3.675140e+05	2.065842e+04	1.803600e+05	2.164185e+05	2.340000e+05	1.500000e+01	1.00
max	2.845382e+06	4.562550e+05	4.180581e+05	6.905160e+06	6.905160e+06	6.905160e+06	2.300000e+01	1.00

Columns which columns has outliers are as :-

1. AMT_ANNUITY
2. AMT_APPLICATION
3. AMT_CREDIT
4. SELLERPLACE_AREA
5. CNT_PAYMENT
6. DAYS_FIRST_DUE
7. DAYS_LAST_DUE_1ST_VERSION
8. DAYS_LAST_DUE
9. DAYS_TERMINATION

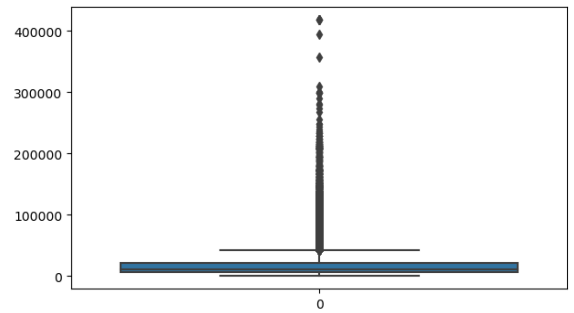
For selected columns we create a variable and run a for loop to box plot chart and describe for every column to have clear idea about ouliers.

```
# Creating a variable for columns having outliers
prev_outlier_col = ['AMT_ANNUITY','AMT_APPLICATION','AMT_CREDIT','SELLERPLACE_AREA',
                    'CNT_PAYMENT','DAYS_FIRST_DUE','DAYS_LAST_DUE_1ST_VERSION','DAYS_LAST_DUE','DAYS_TERMINATION']
```

```
for i in prev_outlier_col:
    print(i,'\n',df_prev[i].describe())
    plt.figure(figsize=[7,4])
    sns.boxplot(df_prev[i])
    plt.show()
```

For AMT_ANNUITY image below how we will analyse each and every column.

```
AMT_ANNUITY
 count    1.297979e+06
mean     1.595512e+04
std      1.478214e+04
min      0.000000e+00
25%      6.321780e+03
50%      1.125000e+04
75%      2.065842e+04
max      4.180581e+05
Name: AMT_ANNUITY, dtype: float64
```



Insights on anlaysis data and plotting boxplot alon with how we can impute for ouliers :-

1. In AMT_ANNUITY column we have 75 percent have Annuity <= 20658.42so few have high Annuity, we can use binning to handle outliers.
2. In AMT_APPLICATION column we have 99 percent have Amount <= 1350000.0 so few have high Amount, we can use binning to handle outliers.
3. In AMT_CREDIT column we have 95 percent have Amount Credit <= 886500.0 so few have high Amount Credit, we can use binning to handle outliers.
4. In SELLERPLACE_AREA column we have 90 percent have Sellerplace Area <= 919.0 so few have high Sellerplace Area we can use binning to handle outliers.
5. In CNT_PAYMENT column we have 95 percent have values <= 48so few have high values, we can use binning to handle outliers.
6. In DAYS_FIRST_DUE column we have 95 percent have values <= 2826.0 so few very have high values,which clear shows its by mistake which we can modify by using median or IQR.
7. In DAYS_LAST_DUE_1ST_VERSION column we have 90 percent have values <= 2651.0 so few very have high values,which clear shows its by mistake which we can modify by using median,IQR.
8. In DAYS_LAST_DUE column we have 78 percent have values <= 2640.0 so few very have high values,which clear shows its by mistake which we can modify by using median,IQR.
9. In DAYS_TERMINATION column we have 77 percent have values <= 2656.0 so few very have high values,which clear shows its by mistake which we can modify by using median,IQR.

Step 6:- Finding columns for binning and do the binning

Columns for binning are:-

1. AMT_CREDIT
2. AMT_ANNUITY
3. AMT_GOODS_PRICE

We first use describe to have insight to do binning and then we do binning for selected columns as below

```
# Using describe to analyse value and allot bins
df_prev.AMT_CREDIT.describe()
```

```
count    1.670213e+06
mean     1.961140e+05
std      3.185746e+05
min      0.000000e+00
25%      2.416050e+04
50%      8.054100e+04
75%      2.164185e+05
max      6.905160e+06
Name: AMT_CREDIT, dtype: float64
```

```
# Creating bins for AMT_CREDIT
df_prev['AMT_CREDIT_BUCKET'] = pd.cut(df_prev['AMT_CREDIT'],
                                      bins=[0,100000,200000,300000,400000,500000,600000,700000,800000,10000000000],
                                      labels=['0-100000','100000-200000','200000-300000','300000-400000',
                                              '400000-500000','500000-600000','600000-700000','700000-800000','800000+'])
```

Now we do Analysis for previous application (df_prev).

Before starting analysis we will check all the columns of data frame and create two variable cat_col_prev for categorical columns and con_col_prev for continuous columns.

```
# Checking all the continous/numerical and assigning into variable cat_col_prev variable
cat_col_prev = ['NAME_CONTRACT_TYPE','NAME_CASH_LOAN_PURPOSE','NAME_CONTRACT_STATUS','NAME_PAYMENT_TYPE',
                'CODE_REJECT_REASON','NAME_CLIENT_TYPE','NAME_GOODS_CATEGORY','NAME_PORTFOLIO',
                'NAME_PRODUCT_TYPE','CHANNEL_TYPE','NAME_SELLER_INDUSTRY','NAME_YIELD_GROUP',
                'PRODUCT_COMBINATION','AMT_CREDIT_BUCKET','AMT_ANNUITY_BUCKET','AMT_GOODS_PRICE_BUCKET']
```

```
# Checking all the continous/numerical and assigning into variable con_col_prev variable
con_col_prev = ['AMT_ANNUITY','AMT_APPLICATION', 'AMT_CREDIT', 'AMT_GOODS_PRICE', 'SELLERPLACE_AREA', 'CNT_PAYMENT',
               'DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE','DAYS_LAST_DUE_1ST_VERSION', 'DAYS_LAST_DUE', 'DAYS_TERMINATION',
                'NFLAG_INSURED_ON_APPROVAL']
```

After that we check data balance by using value count on NAME_CLIENT_TYPE and will create 3 dataframe based on New, Repeater and Refreshed type of client.

```
100*df_prev.NAME_CLIENT_TYPE.value_counts(normalize=True)
```

```
NAME_CLIENT_TYPE
Repeater     73.718757
New          18.043376
Refreshed     8.121654
XNA           0.116213
Name: proportion, dtype: float64
```

```
# Creating dataframe df_prev_new for new clients
df_prev_new = df_prev[df_prev.NAME_CLIENT_TYPE=='New']
```

```
# Creating dataframe df_prev_rep for Repeater clients
df_prev_rep = df_prev[df_prev.NAME_CLIENT_TYPE=='Repeater']
```

```
# Creating dataframe df_prev_ref for Refreshed clients
df_prev_ref = df_prev[df_prev.NAME_CLIENT_TYPE=='Refreshed']
```

Now we do Univariate Analysis for Catergorical Variables.

We will use for loop to plot count plot for all categorical columns for New, Repeater, Refreshed dataframe we have created to find link between different variables .

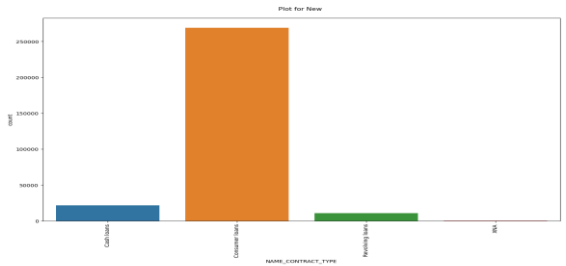
```
# plotting countplot chart for all categorial columns of data set for NAME_CLIENT_TYPE New,Repeater,Refreshed

for i in cat_col_prev :
    plt.figure(figsize=[15,8])

    plt.subplot(3,1,1)
    plt.title("Plot for New \n")
    sns.countplot(data=df_prev_new,x=i)
    plt.xticks(rotation=90)

    plt.subplot(3,1,2)
    plt.title("Plot for Repeater \n")
    sns.countplot(data=df_prev_rep,x=i)
    plt.xticks(rotation=90)

    plt.subplot(3,1,3)
    plt.title("Plot for Refreshed \n")
    sns.countplot(data=df_prev_ref,x=i)
    plt.xticks(rotation=90)
    plt.subplots_adjust(left=0.1,bottom=-1.4,right=0.9,top=1.8,wspace=0.4,hspace=0.4)
    plt.show()
```

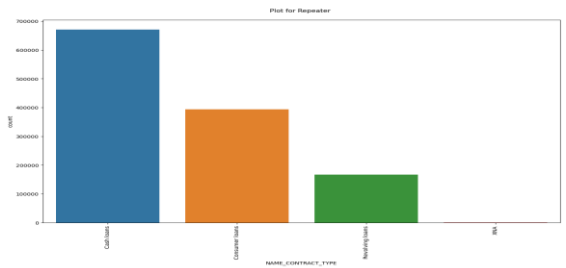
Insights and plots from categorial columns of data set for NAME_CLIENT_TYPE New,Repeater,Refreshed are as:-

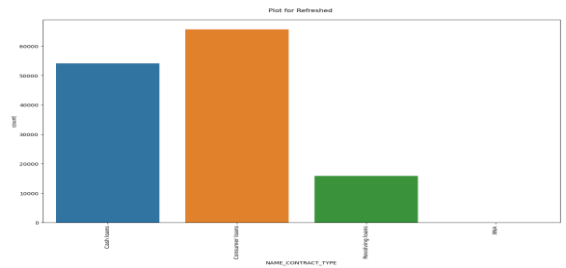
1. For NAME_CONTRACT_TYPE values for
- New has highest for Consumer loans = 89.27 %
- Repeater has highest for Cash loans = 54.46 %
- Refreshed has highest for Consumer loans = 48.41 %

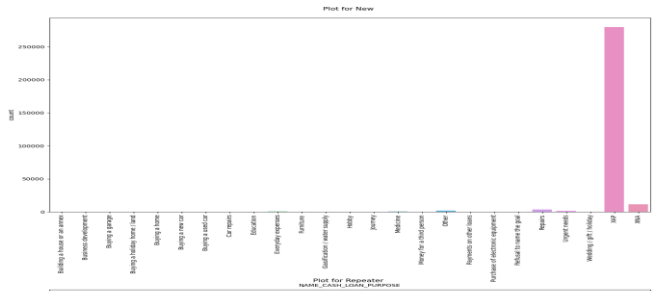
2. For NAME_CASH_LOAN_PURPOSE values for
- New has highest for XAP = 92.82,XNA = 3.87
- Repeater has highest for XNA = 49.89,XAP = 45.53
- Refreshed has highest for XAP = 60.12,XNA = 37.39

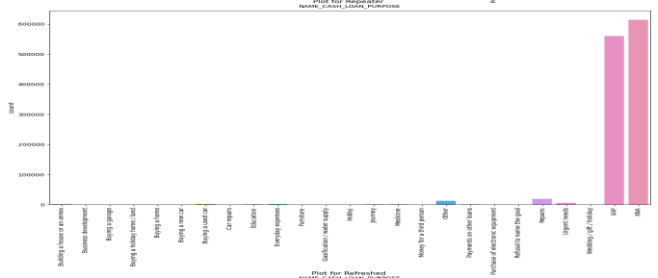
3. For NAME_CONTRACT_STATUS values for
- Approval is in order of New>Refreshed>Repeater
- Canceled is in order of Repeater>Refreshed>New
- Refused is in order of Repeater>Refreshed>New

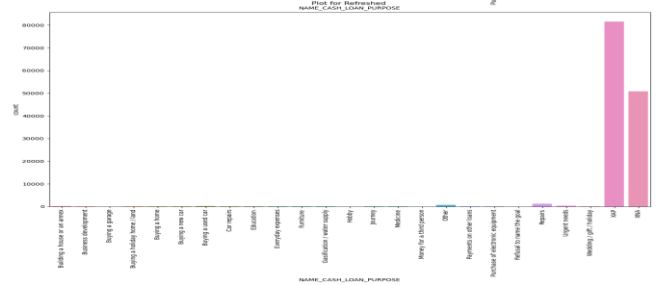


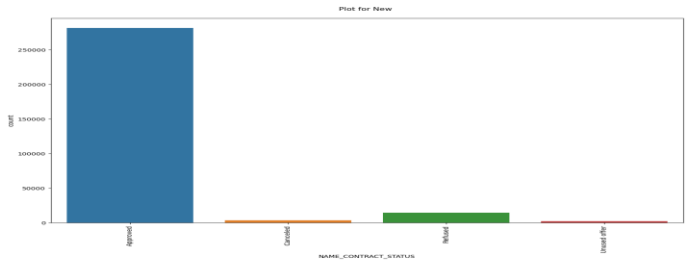


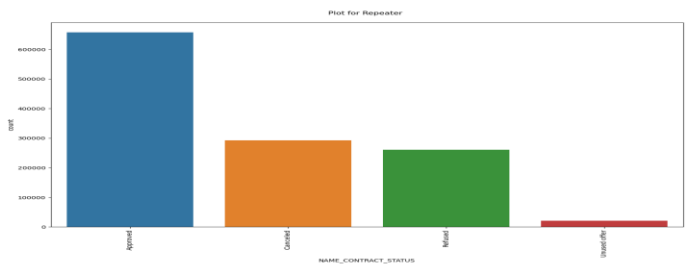


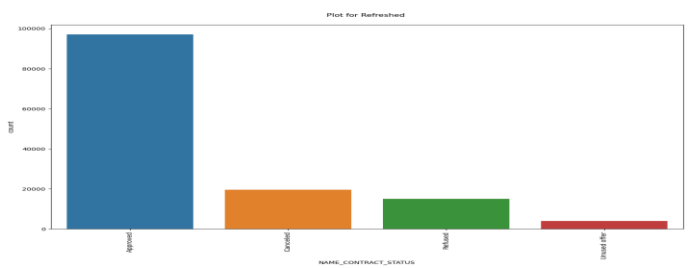








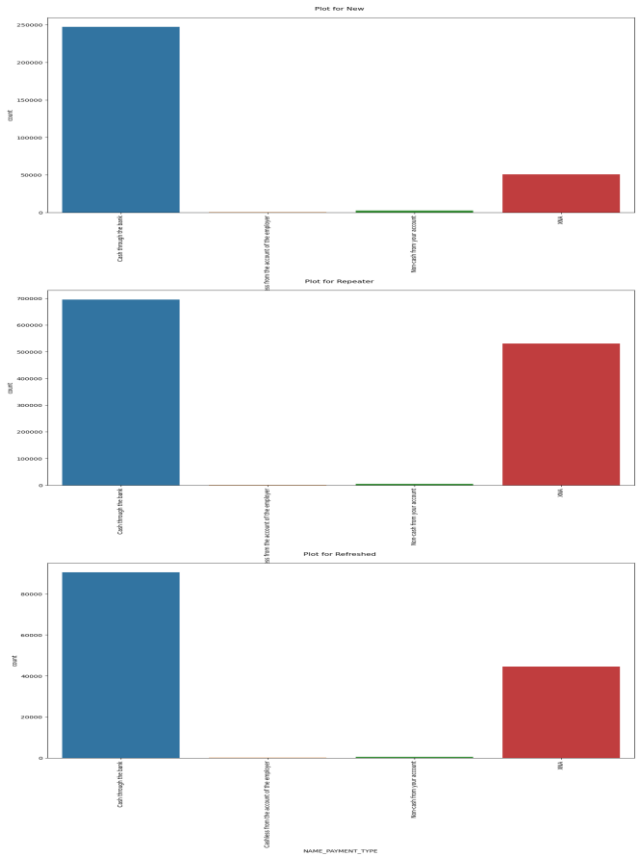


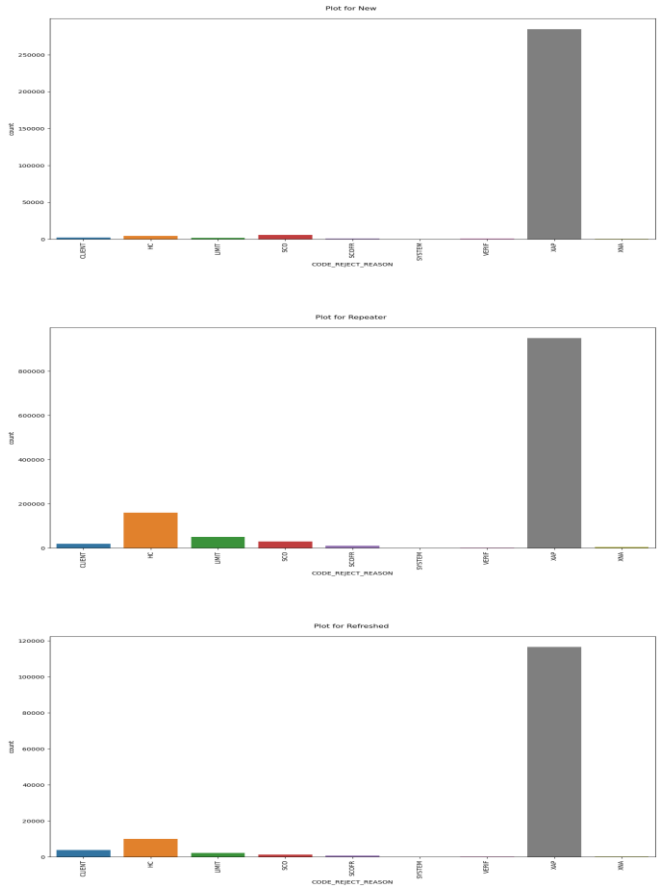


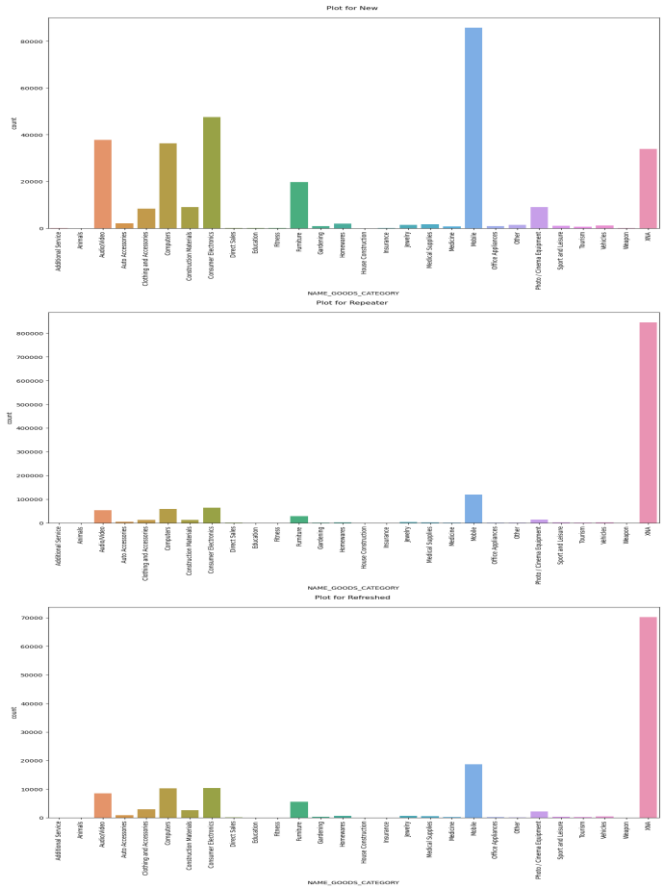
4. For NAME_PAYMENT_TYPE values for
- Cash through the bank is highest used for all 3 categories followed by XNA .

5. For CODE_REJECT_REASON values for
- XAP is most highest CODE_REJECT_REASON for New,Repeater,Refreshed
- followed by SCO for New, and HC for Repeater & Refreshed

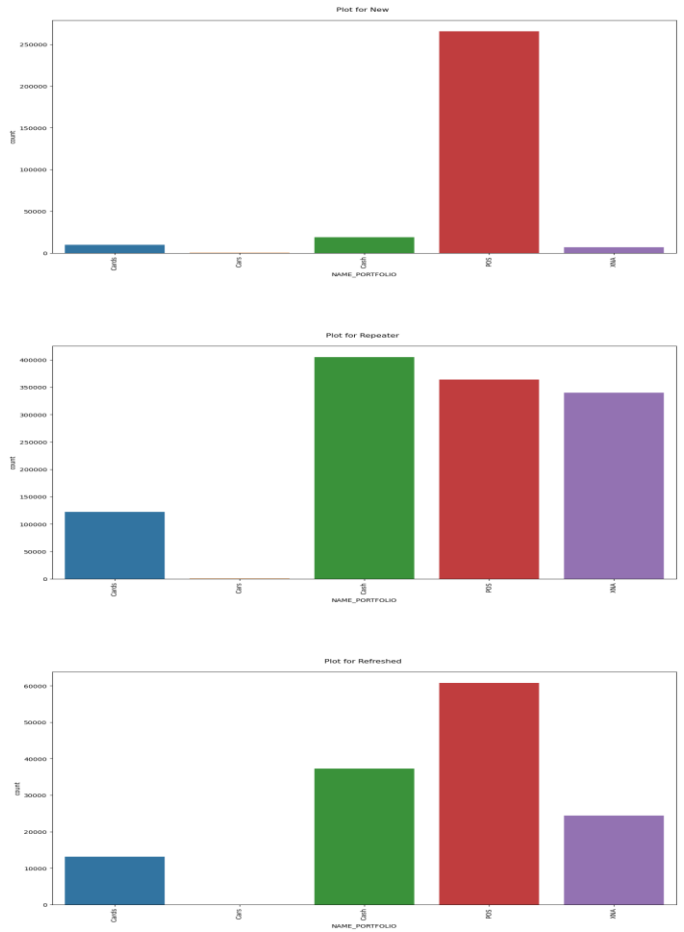
6. For NAME_GOODS_CATEGORY values for
- New used loan for purchased in NAME_GOODS_CATEGORY mobile,consumer electornics,audio/video,computers,XNA
- Repeater & Refreshed used loan for purchased in NAME_GOODS_CATEGORY XNA,mobile,consumer electornics,computers,audio/video



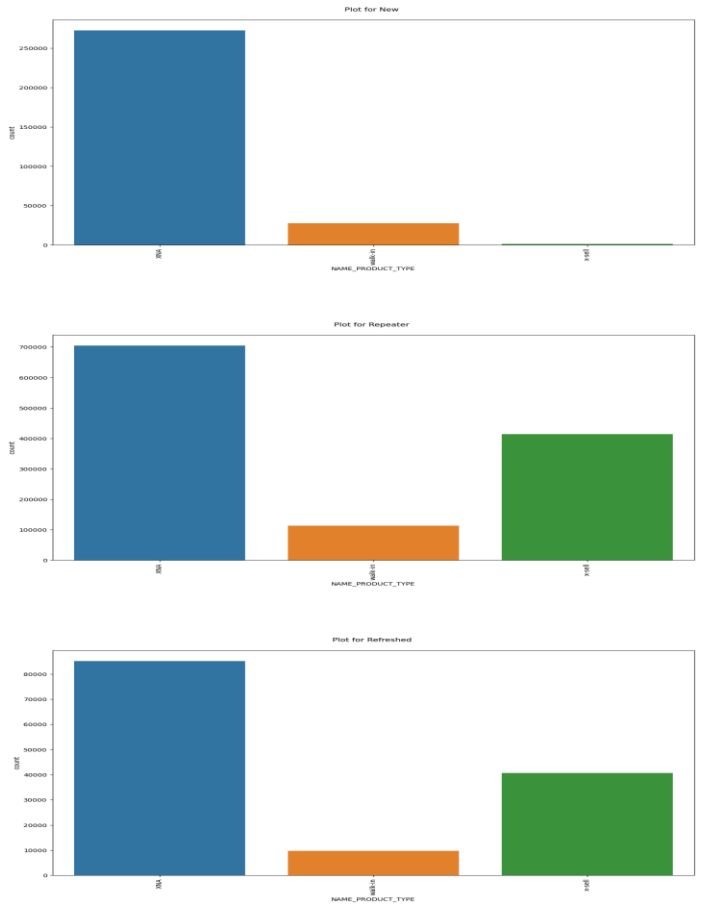




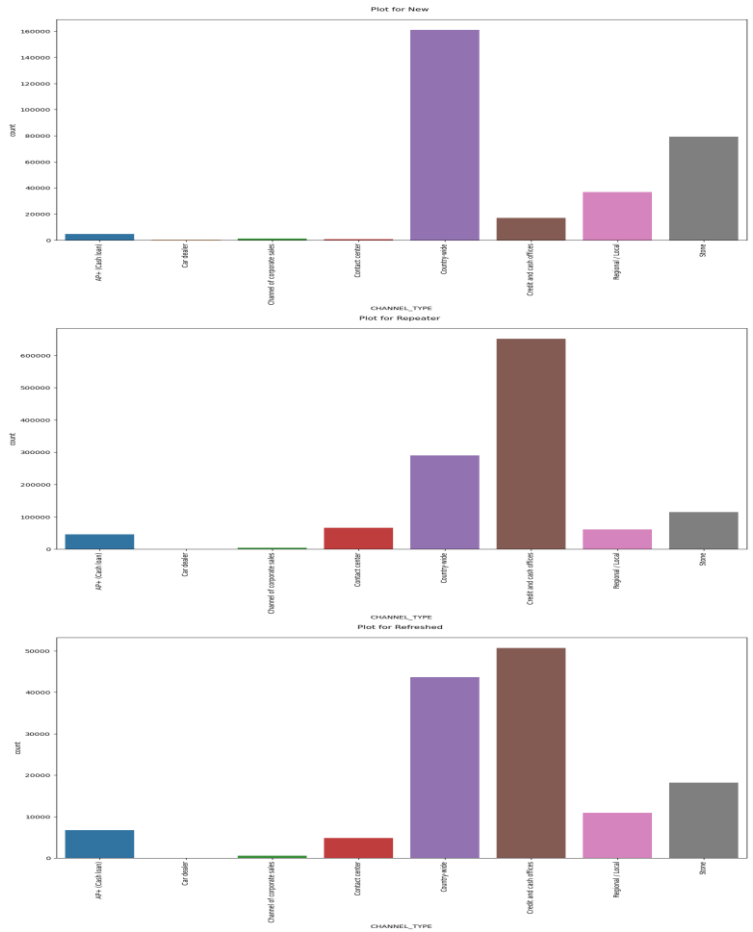
7. NAME_PORTFOLIO
- New & Refreshed has used POS,Cash as most modes.
- Repeater has used Cash and POS as most modes.



8. NAME_PRODUCT_TYPE
- New has PRODUCT_TYPE as XNA = 90.43 %,walk-in = 9.09 %
- Repeater PRODUCT_TYPE as XNA = 57.20 %, x-sell = 33.61 %
- Refreshed PRODUCT_TYPE as XNA = 62.81 %, x-sell = 29.98 %



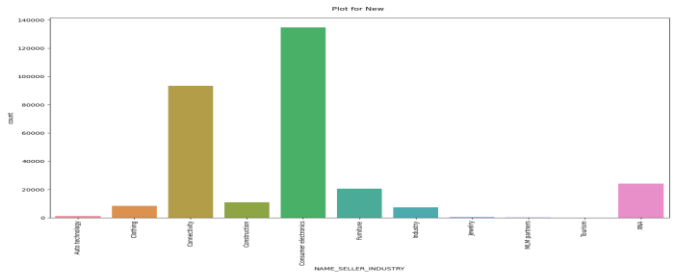
9. CHANNEL_TYPE
- New has CHANNEL_TYPE as Country-wide = 53.41 %,Stone=26.33 %,Regional / Local=12.26 %
- Repeater has CHANNEL_TYPE as Credit and cash offices = 52.86 %,Country-wide = 23.53 %,Stone= 9.29 %
- Refreshed has CHANNEL_TYPE as Credit and cash offices = 37.39 %,Country-wide = 32.16 %,Stone = 13.41 %

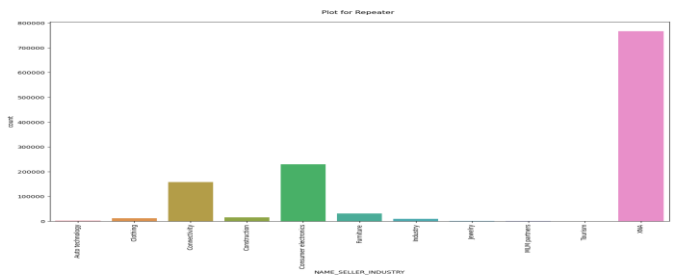


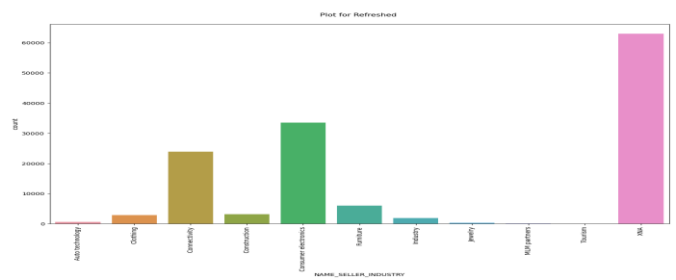
10. NAME_SELLER_INDUSTRY
- New has NAME_SELLER_INDUSTRY as Consumer electronics=44.68 %,Connectivity=31.00 %,XNA=7.98%
- Repeater has NAME_SELLER_INDUSTRY as XNA=62.31%,Consumer electronics=18.66 %,Connectivity=12.87%
- Refreshed has NAME_SELLER_INDUSTRY as XNA=46.46%,Consumer electronics=24.69 %,Connectivity=17.64 %

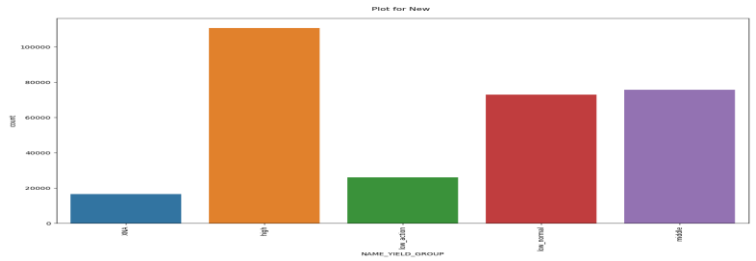
11. NAME_YIELD_GROUP
- for New NAME_YIELD_GROUP high=36.68 %, middle=25.07 %, low_normal=24.15 %
- for Repeater NAME_YIELD_GROUP XNA=37.52 %,middle=22.0 7%,low_normal=17.93 %
- for Refreshed NAME_YIELD_GROUP middle=7.88 %,XNA=27.69 %,low_normal=20.77 %

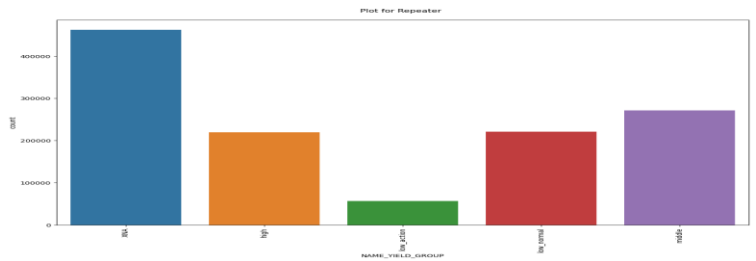
12. PRODUCT_COMBINATION
- for New PRODUCT_COMBINATION as POS household with interest=32.09 %,POS mobile with interest=27.46 %,POS industry with interest=11.82 %
- for Repeater PRODUCT_COMBINATION as Cash=21.58 %,POS household with interest=11.70 %,Cash X-Sell: middle=10.48 %
- for Refreshed PRODUCT_COMBINATION as POS household with interest=16.72 %, POS mobile with interest=13.50 %,Cash=12.37 %

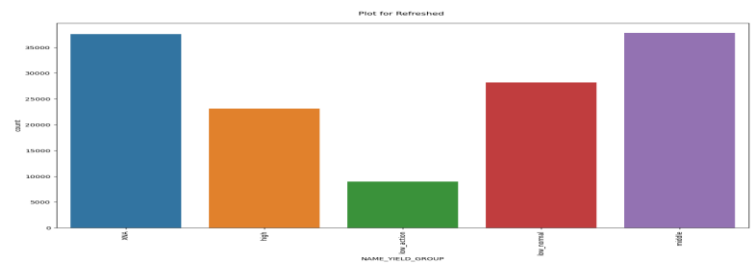


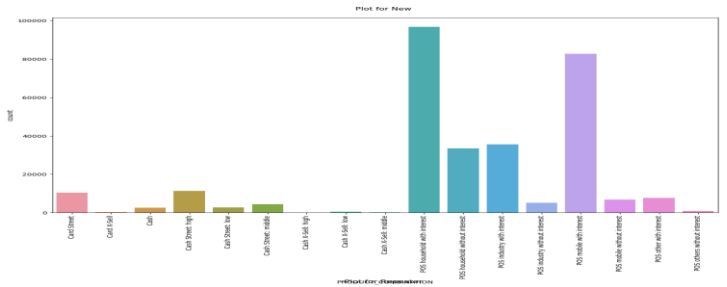


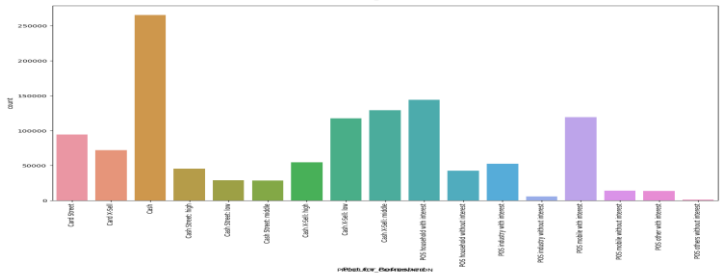


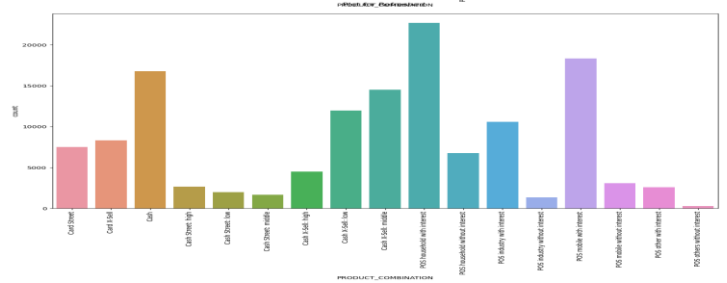




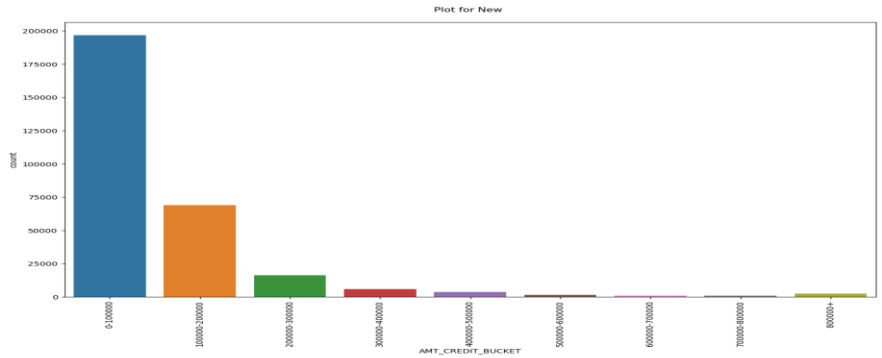


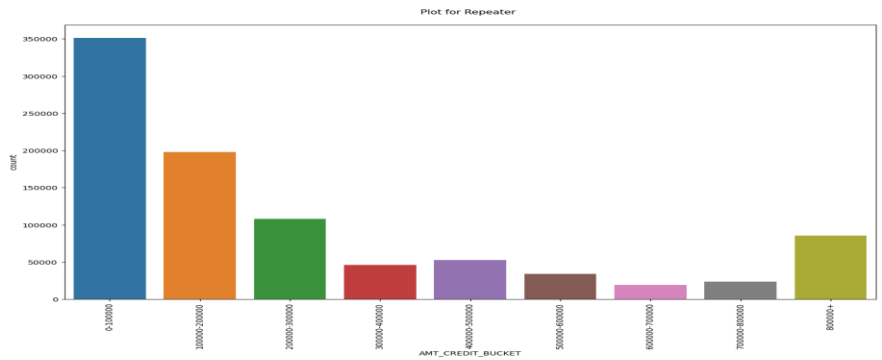


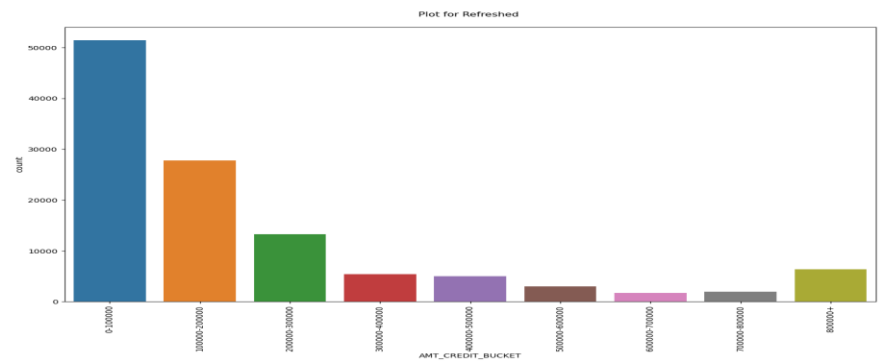




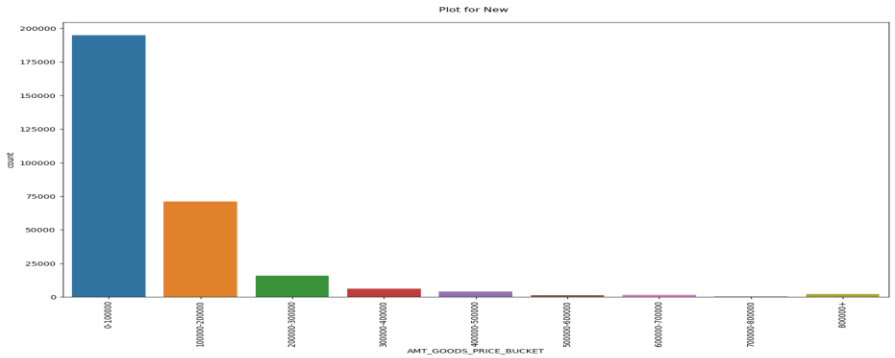
13. AMT_CREDIT_BUCKET
- Top two AMT_CREDIT_BUCKET for New,Repeater,Refreshed are 0-100000,100000-200000.

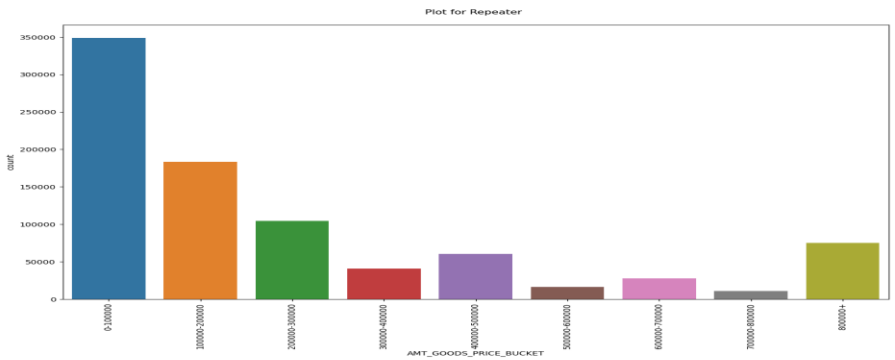


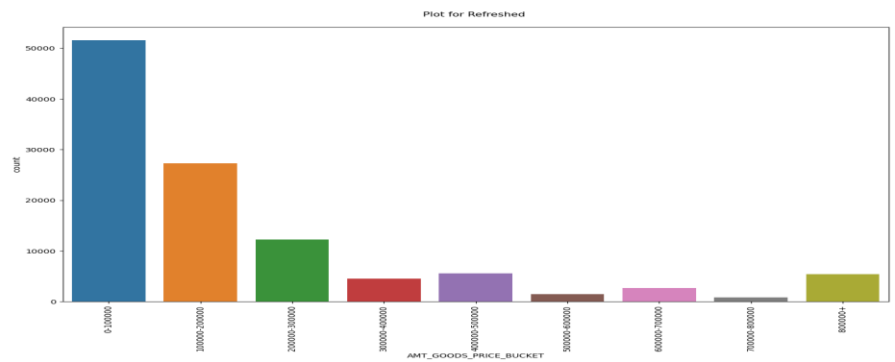




14. AMT_GOODS_PRICE_BUCKET
- Top two AMT_GOODS_PRICE_BUCKET for New,Repeater,Refreshed are 0-100000,100000-200000.







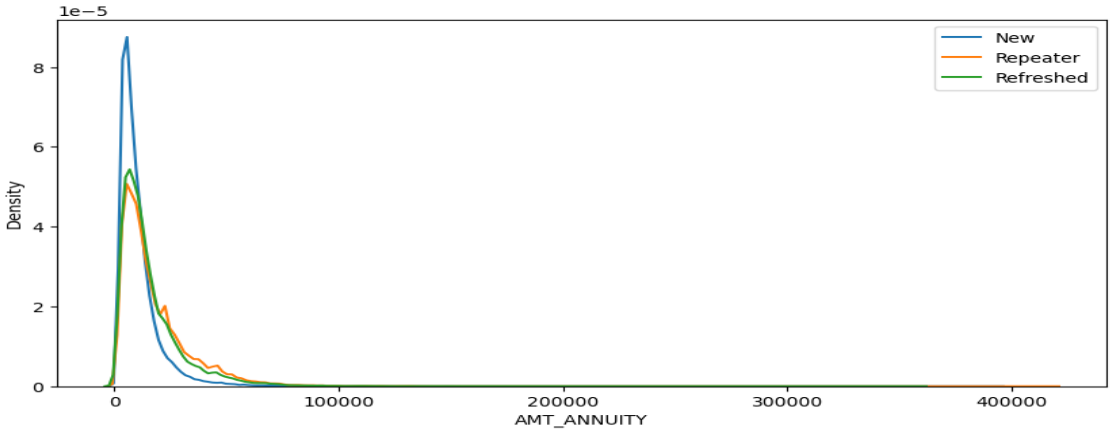
Now we do Univariate Analysis for Continous Variables.

We will use for loop to plot count plot for all continous columns for New, Repeater, Refreshed dataframe we have created to find correlation between different variables .

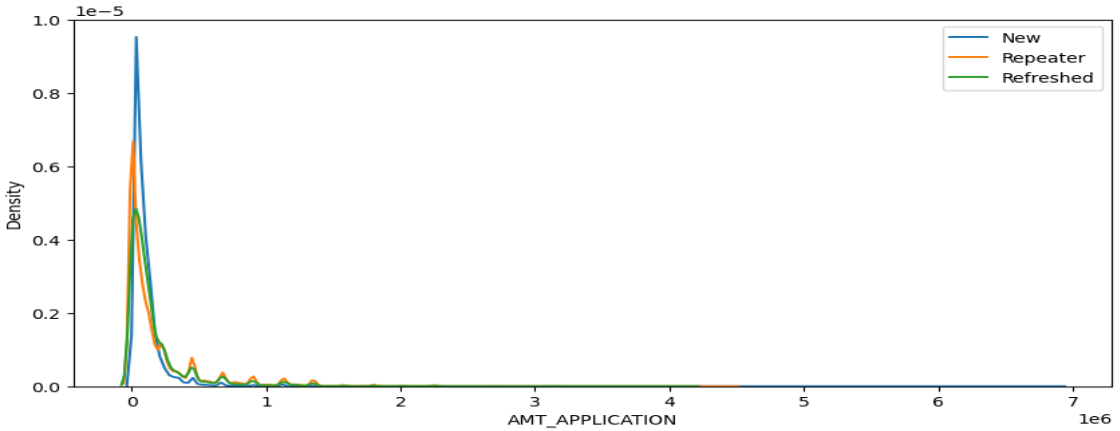
```
# plotting distplot chart for all continous columns of data set for NAME_CLIENT_TYPE New,Repeater,Refreshed
for i in con_col_prev:
    plt.figure(figsize=[10,5])
    sns.distplot(df_prev_new[i],hist=False,label='New')
    sns.distplot(df_prev_rep[i],hist=False,label='Repeater')
    sns.distplot(df_prev_ref[i],hist=False,label='Refreshed')
    plt.legend()
    plt.show()
```

Few insights and plots for continous columns of data set for NAME_CLIENT_TYPE New,Repeater,Refreshed are as:-

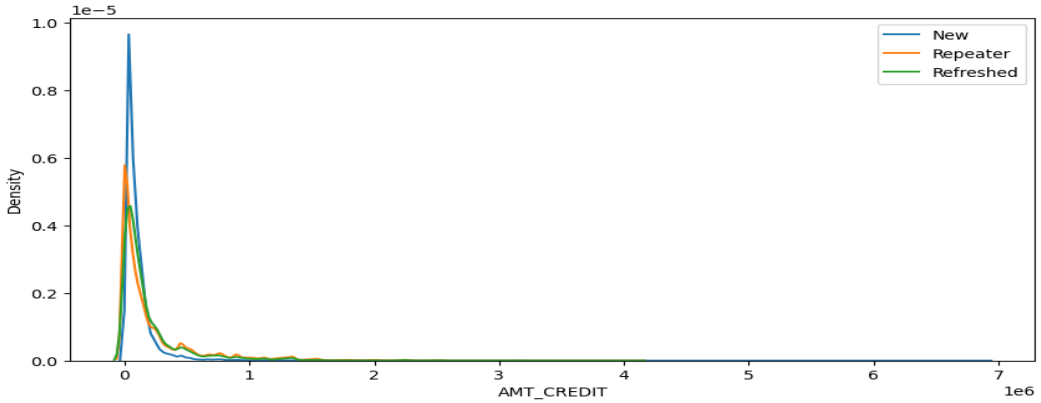
1. for AMT_ANNUITY
- for New Initially AMT_ANNUITY is low then density is high which is decreasing
- with high AMT_ANNUITY while for Repeater and Refreshed with AMT_ANNUITY is increasing then density is increasing



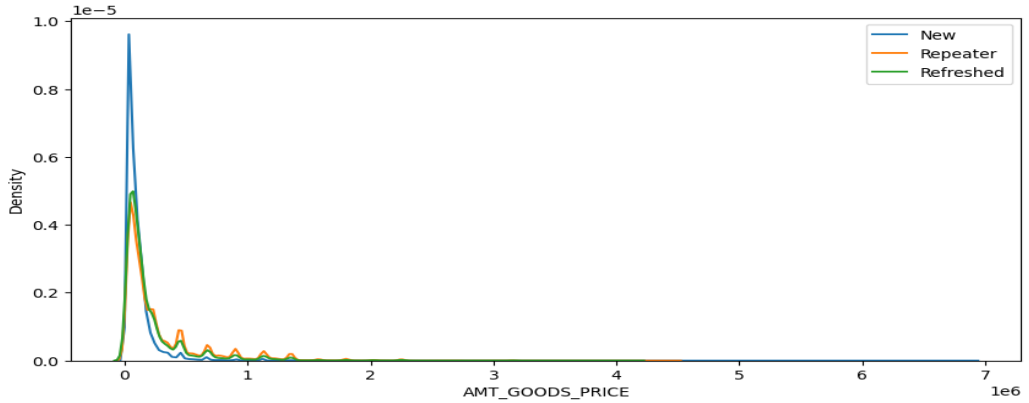
2. for AMT_APPLICATION
- for New Initially AMT_APPLICATION is low then density is high which is decreasing
- with high AMT_APPLICATION while for Refreshed with AMT_APPLICATION is increasing then density is increasing but less than Repeater



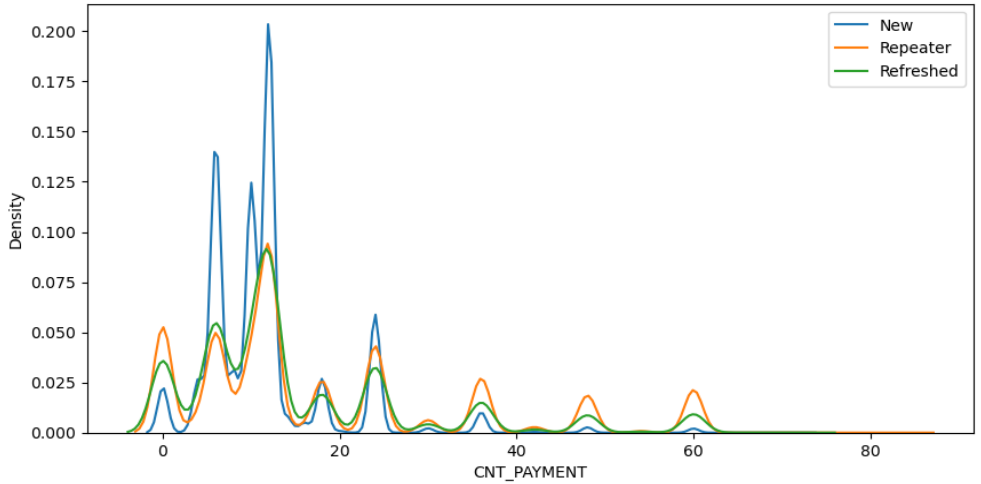
3. for AMT_CREDIT
- at low AMT_CREDIT count of New >Repeaters > Refreshed, but as the AMT_CREDIT values increases count of Repeaters > Refreshed >New



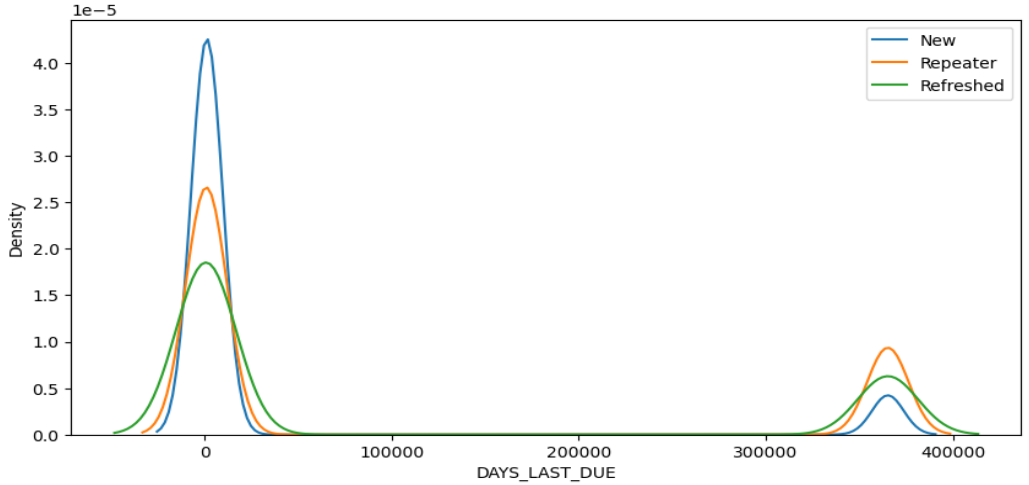
4. for AMT_GOODS_PRICE
- at low value for AMT_GOODS_PRICE count of New>Refreshed>Repeaters
- but as value for AMT_GOODS_PRICE count of Repeaters > Refreshed >New



5. for CNT_PAYMENT
- on observing the chart and data it is clear that CNT_PAYMENT is high for New then for Repeaters and then for Refreshed.



6. for DAYS_FIRST_DUE,DAYS_LAST_DUE_1ST_VERSION,DAYS_LAST_DUE,DAYS_TERMINATION
- on observing the chart and data it is clear that for low value all are in order New >Repeaters>Refreshed.
- for high value all are in order for Repeaters>Refreshed>New.



Now we merge application and previous data frames and analyse the merged dataframe and its display head.

```
# Merging dataframes df anfd df_prev by using inner joint on SK_ID_CURR
df_loan = pd.merge(df,df_prev, how='inner',on='SK_ID_CURR')
```

```
# Checking head of new dataframe
df_loan.head()
```

ON	DAYS_LAST_DUE	DAYS_TERMINATION	NFLAG_INSURED_ON_APPROVAL	AMT_CREDIT_BUCKET_y	AMT_ANNUITY_BUCKET_y	AMT_GOODS_PRICE_BUCKET_
25.0	25.0	17.0	0.0	100000-200000	0-100000	100000-2000
36.0	536.0	527.0	1.0	800000+	0-100000	80000
17.0	647.0	639.0	0.0	300000-400000	0-100000	300000-4000
30.0	1980.0	1976.0	1.0	0-100000	0-100000	0-1000
04.0	724.0	714.0	0.0	0-100000	0-100000	0-1000

After displaying head we check shape and info of merged i.e. df_loan .

```
# Checking the shape of df_loan
df_loan.shape
```

```
(1413701, 96)
```

```
# Checking the info of df_loan
df_loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1413701 entries, 0 to 1413700
Data columns (total 96 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   SK_ID_CURR                   1413701 non-null  int64
 1   TARGET                       1413701 non-null  int64
 2   NAME_CONTRACT_TYPE_x         1413701 non-null  category
```

We use describe to have an insight of df_loan .

```
# using describe function for df_loan to have an insight
df_loan.describe()
```

	SK_ID_CURR	TARGET	AMT_INCOME_TOTAL	AMT_CREDIT_x	AMT_ANNUITY_x	AMT_GOODS_PRICE_x	REGION_POPULATION_RELATIVE	DAYS_BI
count	1.413701e+06	1.413701e+06	1.413701e+06	1.413701e+06	1.413608e+06	1.412493e+06	1.413701e+06	1.413701e
mean	2.784813e+05	8.655296e-02	1.733160e+05	5.875537e+05	2.701702e+04	5.277186e+05	2.074985e-02	1.632105e
std	1.028118e+05	2.811789e-01	1.985734e+05	3.849173e+05	1.395116e+04	3.532465e+05	1.334702e-02	4.344557e
min	1.000020e+05	0.000000e+00	2.565000e+04	4.500000e+04	1.615500e+03	4.050000e+04	2.900000e-04	7.489000e
25%	1.893640e+05	0.000000e+00	1.125000e+05	2.700000e+05	1.682100e+04	2.385000e+05	1.003200e-02	1.273900e
50%	2.789920e+05	0.000000e+00	1.575000e+05	5.084955e+05	2.492550e+04	4.500000e+05	1.885000e-02	1.604400e
75%	3.675560e+05	0.000000e+00	2.070000e+05	8.079840e+05	3.454200e+04	6.795000e+05	2.866300e-02	1.998000e
max	4.562550e+05	1.000000e+00	1.170000e+08	4.050000e+06	2.250000e+05	4.050000e+06	7.250800e-02	2.520100e

Now we check null values of all columns of df_loan.

```
# Checking  null percentage of columns of dataframe
null_cal(df_loan)
```



```
SK_ID_CURR                    0.000000
TARGET                        0.000000
NAME_CONTRACT_TYPE_x          0.000000
```

Now we drop few columns having high null values percentage

```
col_to_drop_loan = ['DAYS_EMPLOYED','DAYS_BIRTH','WEEKDAY_APPR_PROCESS_START_y','HOUR_APPR_PROCESS_START_y']
df_loan = df_loan.drop(col_to_drop_loan,axis=1)
```

Now we convert columns from object datatype to category datatype

```
# Checking whether changes made to object type datatypes in columns is reflected or not
df_loan[['FLAG_LAST_APPL_PER_CONTRACT','NAME_TYPE_SUITE_y']].dtypes
```

```
FLAG_LAST_APPL_PER_CONTRACT    category
NAME_TYPE_SUITE_y              category
dtype: object
```

```
# Checking null values in different columns of dataframe df_loan
null_cal(df_loan)
```

Now we create variable for categorical and continuous columns

```
# Creating a variable cat_col_loan containing categorical columns of df_loan
cat_col_loan = ['NAME_CONTRACT_TYPE_x','CODE_GENDER','FLAG_OWN_CAR','FLAG_OWN_REALTY','CNT_CHILDREN',
                    'NAME_TYPE_SUITE_x','NAME_INCOME_TYPE','NAME_EDUCATION_TYPE',
                       'NAME_FAMILY_STATUS','NAME_HOUSING_TYPE','OCCUPATION_TYPE','CNT_FAM_MEMBERS',
                       'ORGANIZATION_TYPE','Age_Group','YEARS_EMPLOYED_BUCKET','AMT_INCOME_BUCKET','AMT_CREDIT_BUCKET_x',
                    'AMT_ANNUITY_BUCKET_x','AMT_GOODS_PRICE_BUCKET_x','NAME_CONTRACT_TYPE_y','NAME_CASH_LOAN_PURPOSE',
                    'NAME_CONTRACT_STATUS','NAME_PAYMENT_TYPE', 'CODE_REJECT_REASON','NAME_CLIENT_TYPE','NAME_GOODS_CATEGORY
                    'NAME_PORTFOLIO','NAME_PRODUCT_TYPE','CHANNEL_TYPE','NAME_SELLER_INDUSTRY','NAME_YIELD_GROUP',
                    'PRODUCT_COMBINATION']


```

```
#Creating a variable cont_col_loan containing continous columns of df_loan
cont_col_loan = ['AMT_INCOME_TOTAL','AMT_CREDIT_x','AMT_ANNUITY_x','AMT_GOODS_PRICE_x','REGION_POPULATION_RELATIVE',
                 'DAYS_REGISTRATION','DAYS_ID_PUBLISH','OBS_30_CNT_SOCIAL_CIRCLE','DEF_30_CNT_SOCIAL_CIRCLE' ,
                 'OBS_60_CNT_SOCIAL_CIRCLE','DEF_60_CNT_SOCIAL_CIRCLE','DAYS_LAST_PHONE_CHANGE' ,'Age',
                 'YEARS_EMPLOYED','CNT_PAYMENT']
```

Now we are going to check data imbalance on TARGET column

```
# doing data imbalance on TARGET column of df_loan
df_loan.TARGET.value_counts(normalize=True)
```

```
TARGET
0    0.913447
1    0.086553
Name: proportion, dtype: float64
```

On basis of data imbalance we create tow dataframes df_loan_0 for TARGET = 0 and df_loan_1 for TARGET = 1

```
# Creating a dataframe df_loan_0 for TARGET =0 value
df_loan_0=df_loan[df_loan.TARGET==0]
```

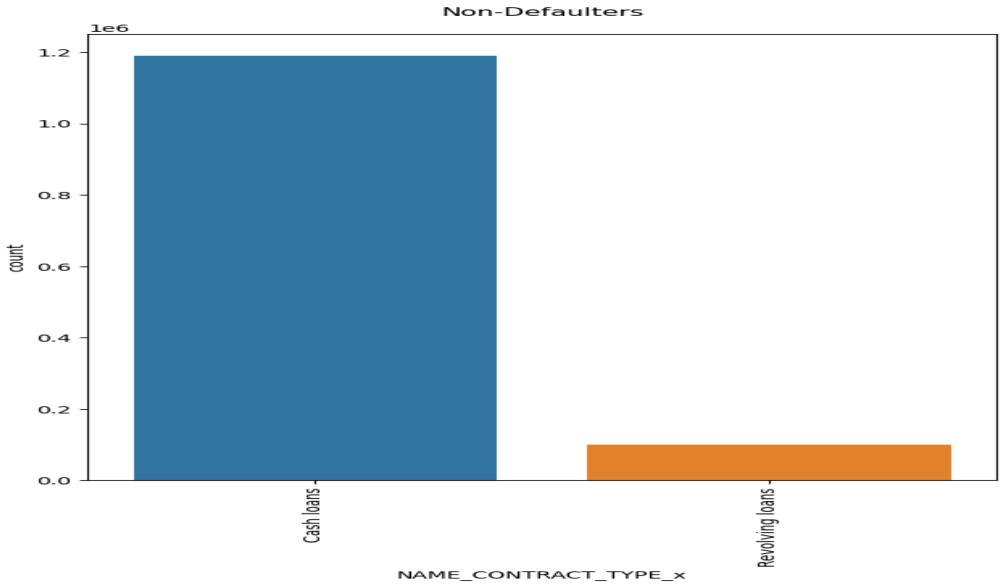
```
# Creating a dataframe df_loan_0 for TARGET =1 value
df_loan_1=df_loan[df_loan.TARGET==1]
```

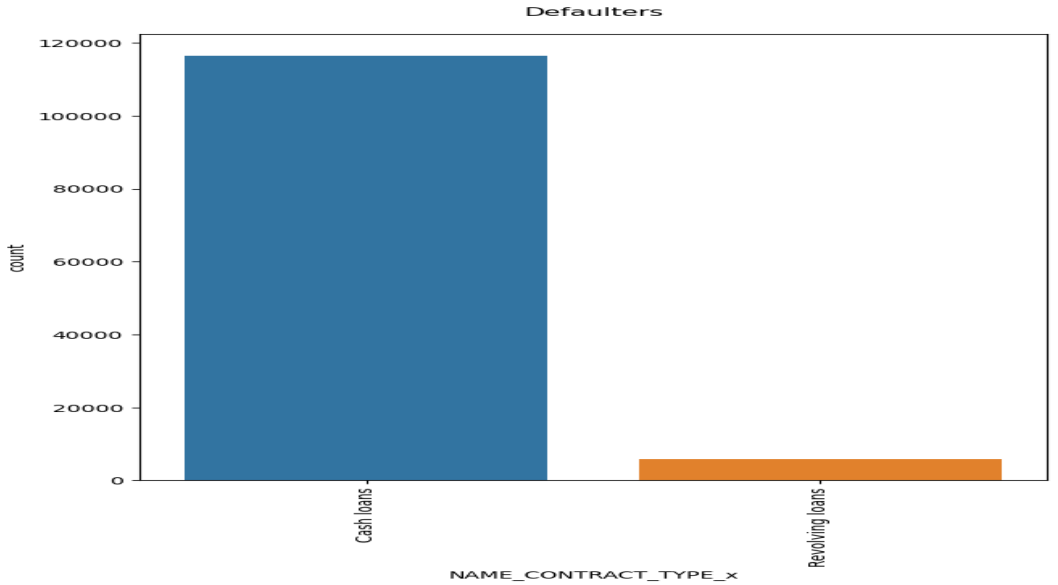
We now do univariate analysis of categorical columns for both dataframes df_loan_0 (non-defaulters) df_loan_1(defaulters)

```
# plotting count chart for all categorial columns of data set
for i in cat_col_loan:
    plt.figure(figsize=[16,8])
# plotting subplot for non-defaulters
    plt.subplot(1,2,1)
    plt.title('Non-Defaulters\n')
    sns.countplot(data=df_loan_0, x=i)
    plt.xticks(rotation=90)
# plotting subplot for non-defaulters
    plt.subplot(1,2,2)
    plt.title('Defaulters\n')
    sns.countplot(data=df_loan_1, x=i)
    plt.xticks(rotation=90)
    plt.show()
```

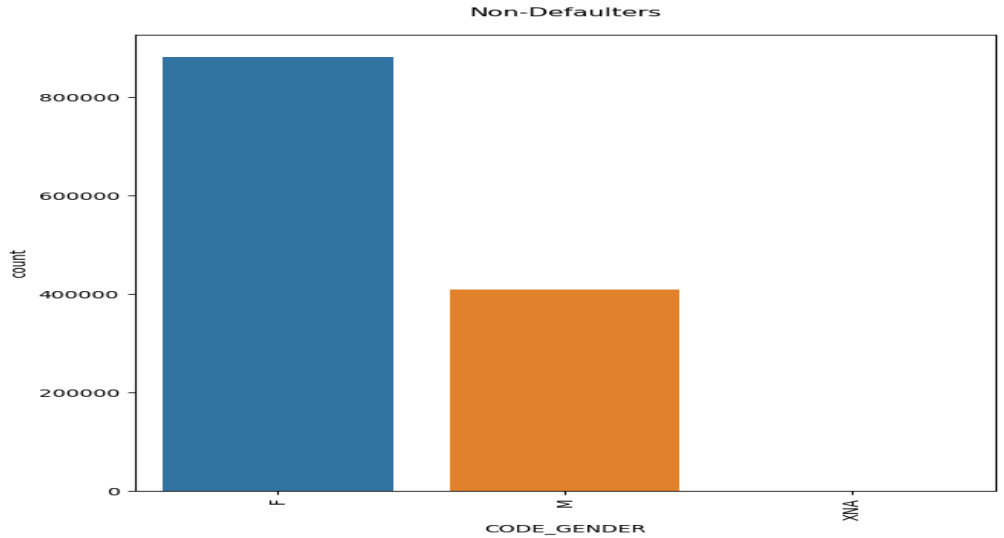
Few points along with countplots which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-

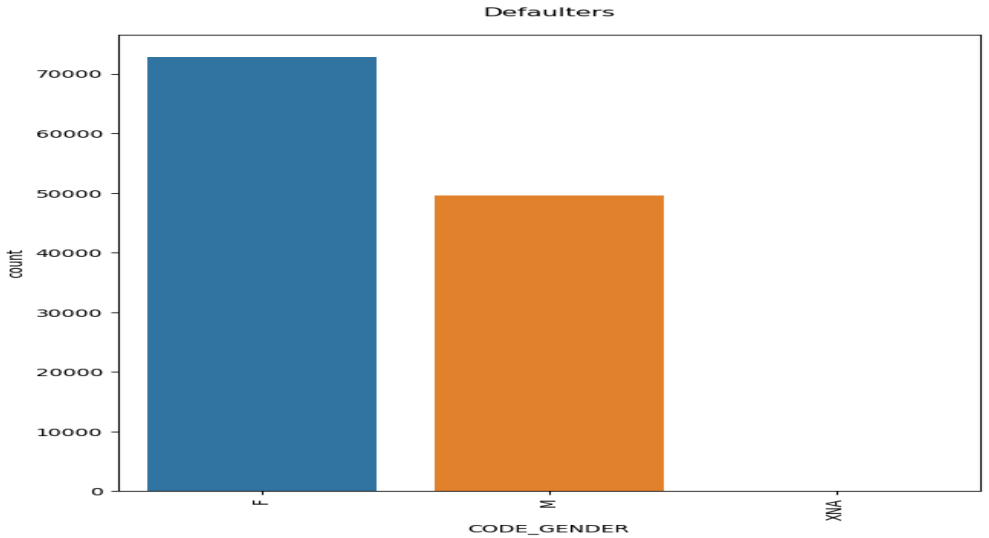
1. NAME_CONTRACT TYPE- Cash Loans are large part of the company's portfolio. For Target 0 is 92.19 % and almost 95.23 % for Target 1.



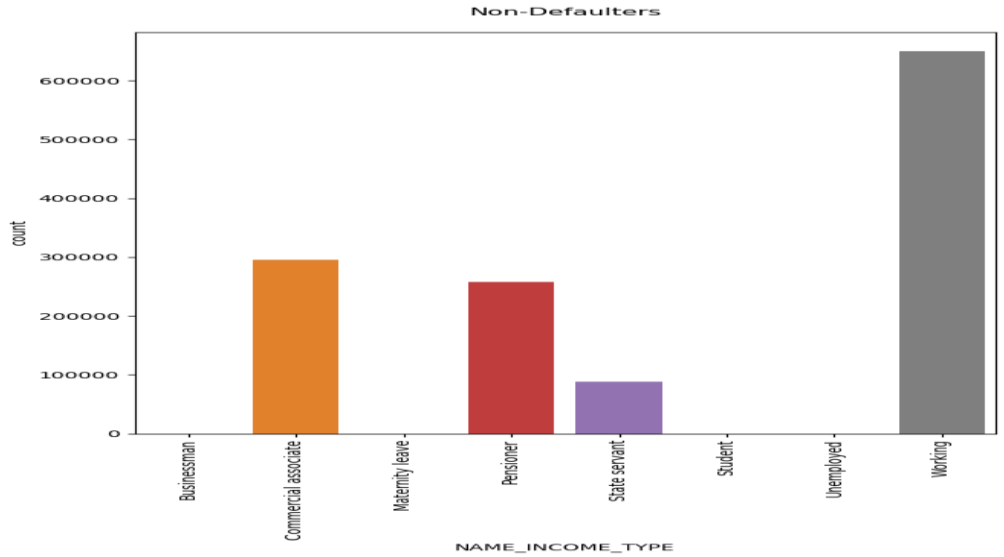


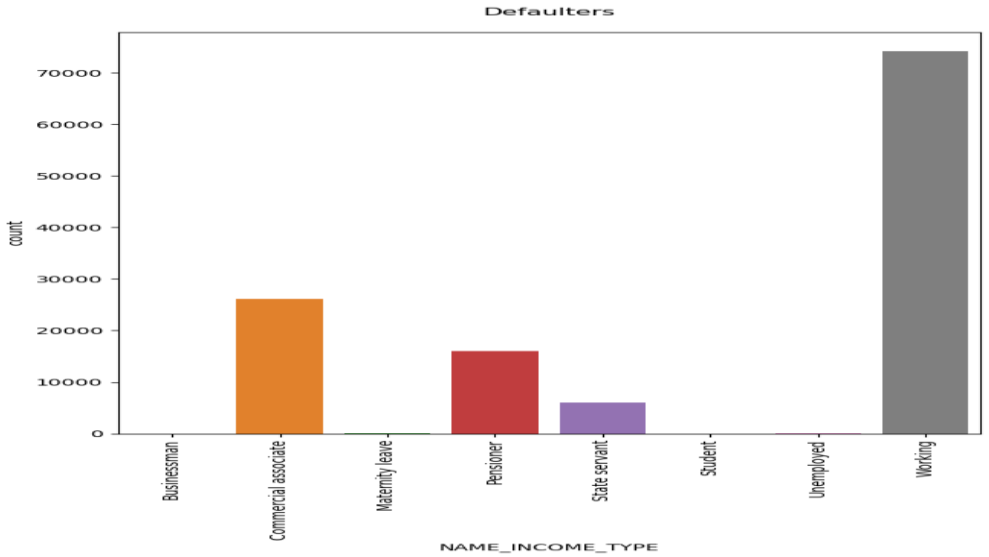
2. CODE_GENDER - Ratio of F to M in Target 0 is 2.157 and F to M in Target 0 is 1.47 indicating that MEN are defaulting more than Women.



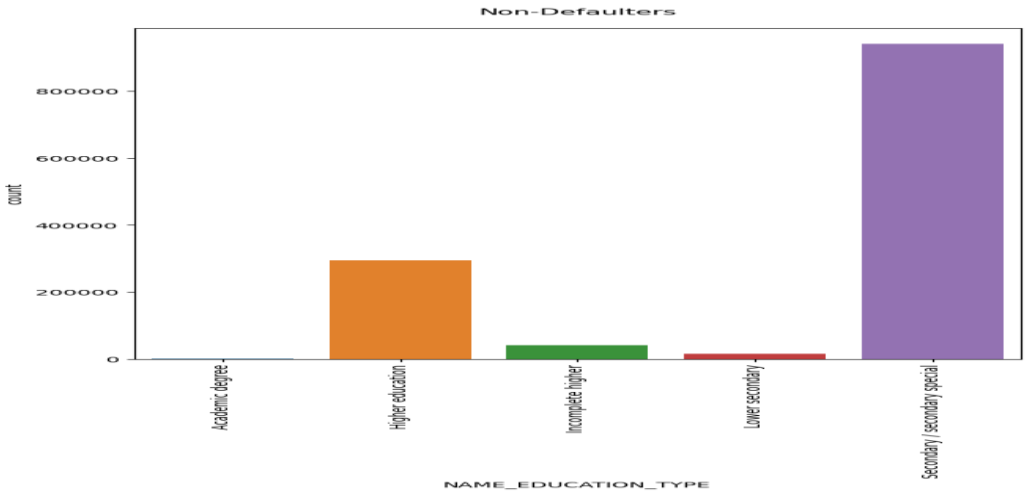


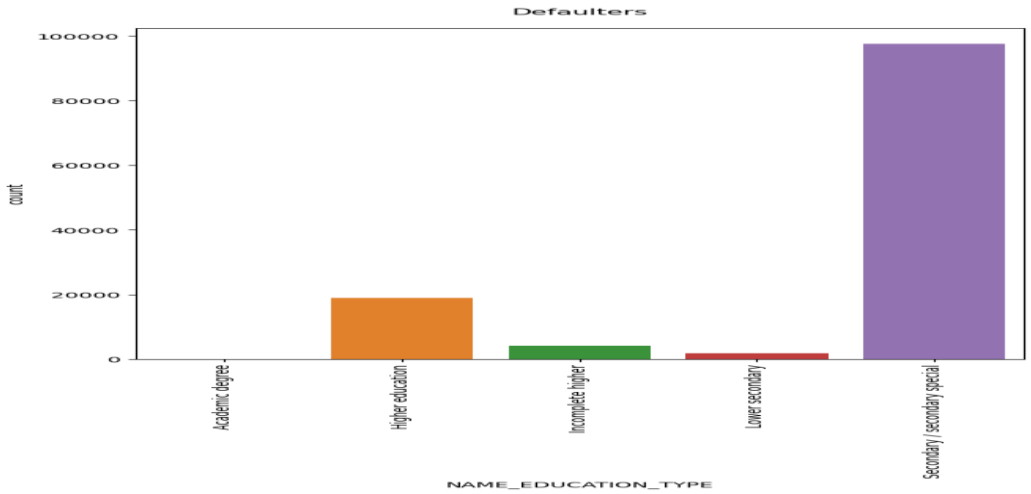
3. NAME_INCOME_TYPE - 50.32 % working in case of Target 0 and 60.62 % in case of Target 1 are working income types.



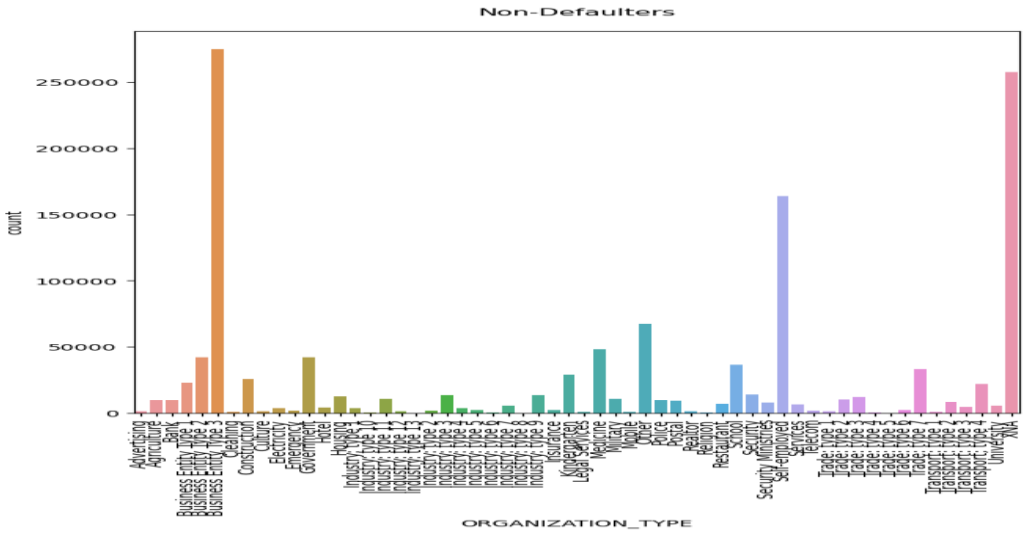


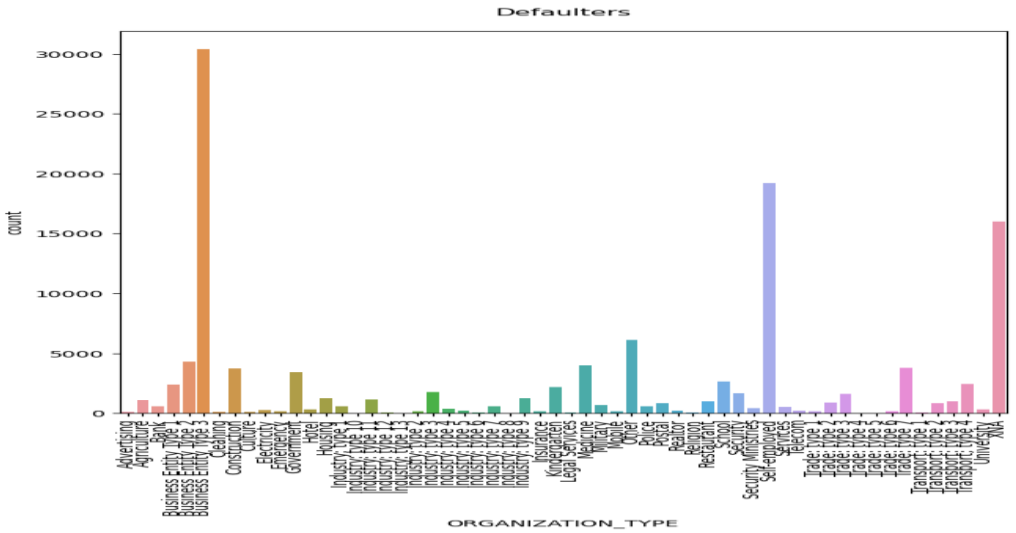
4. NAME_EDUCATION_TYPE - In both Target 0 and 1, applicants with Secondary Education has applied for loans more than others. 79 % of defaulting payments are from applicants with Secondary / secondary special,Needs further analysis.



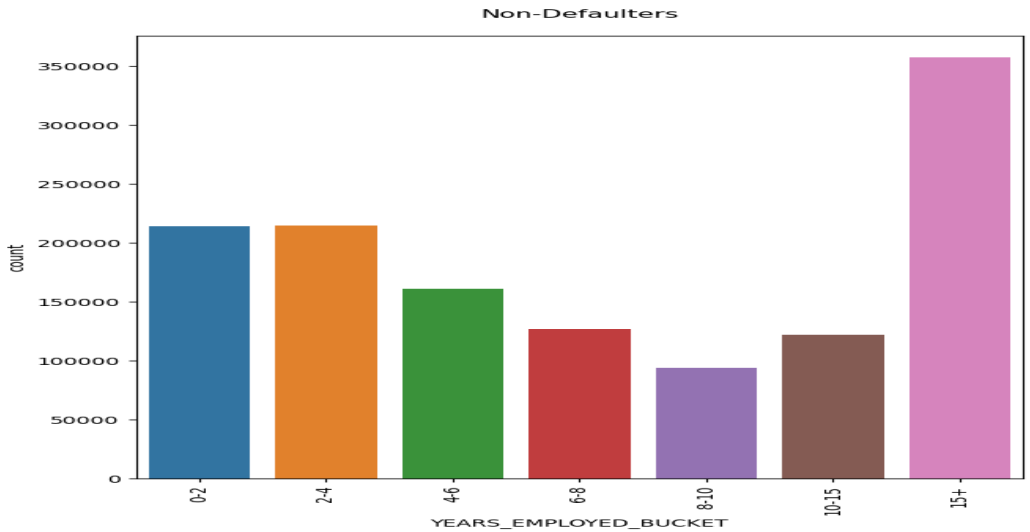


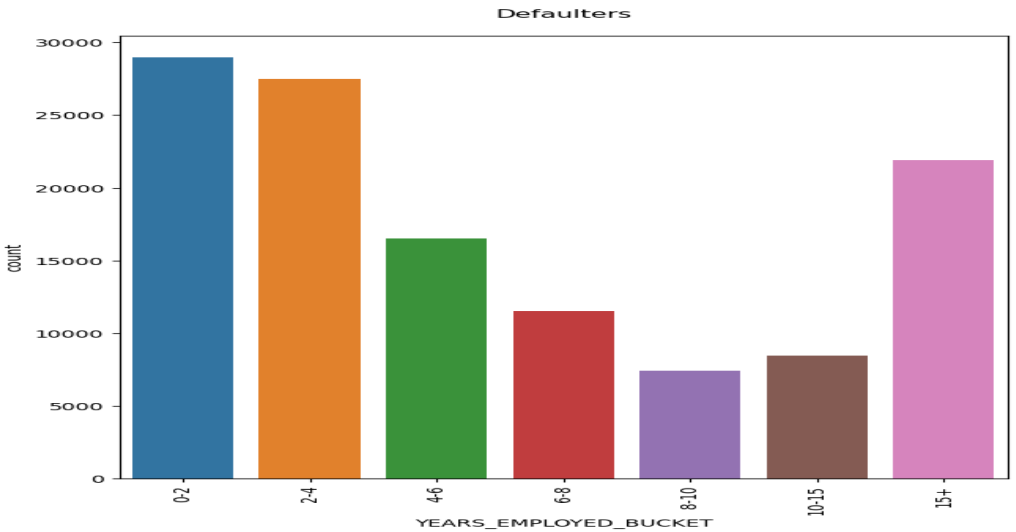
5. ORGANIZATION_TYPE - Business ENTITY TYPE 3 AND SELF EMPLOYED add upto 40.5 % defaulters. The highest % of loan takers are also this category



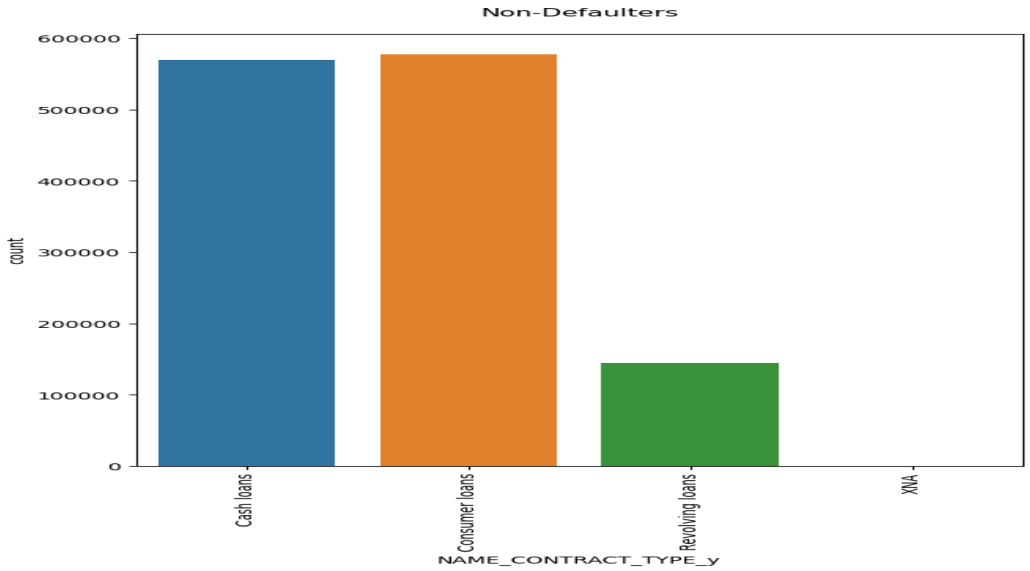


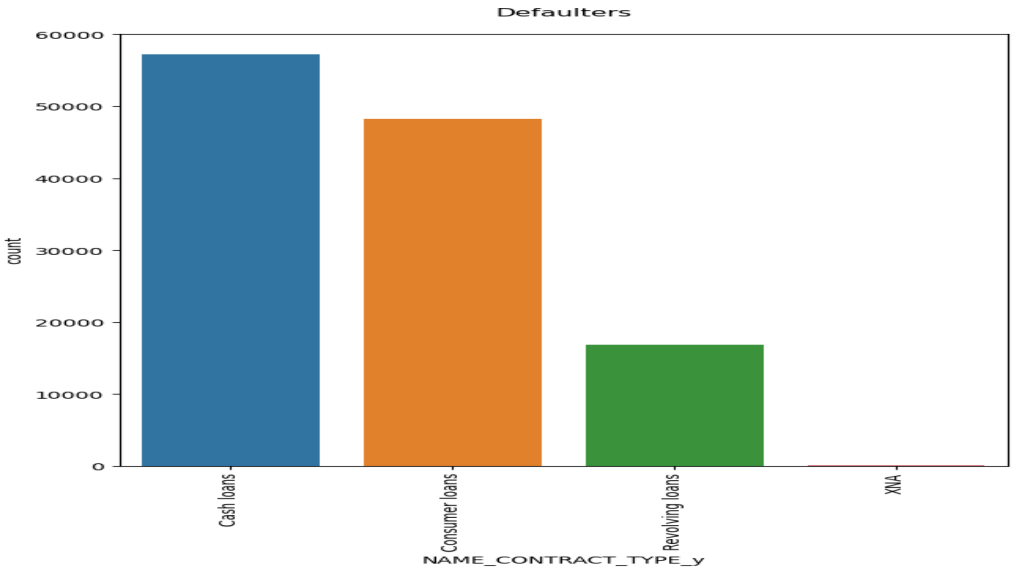
6. YEARS_EMPLOYED_BUCKET - for Defaulters 46 % comes from 0-2,2-4 category bank need to check while providing loans to less years clients.



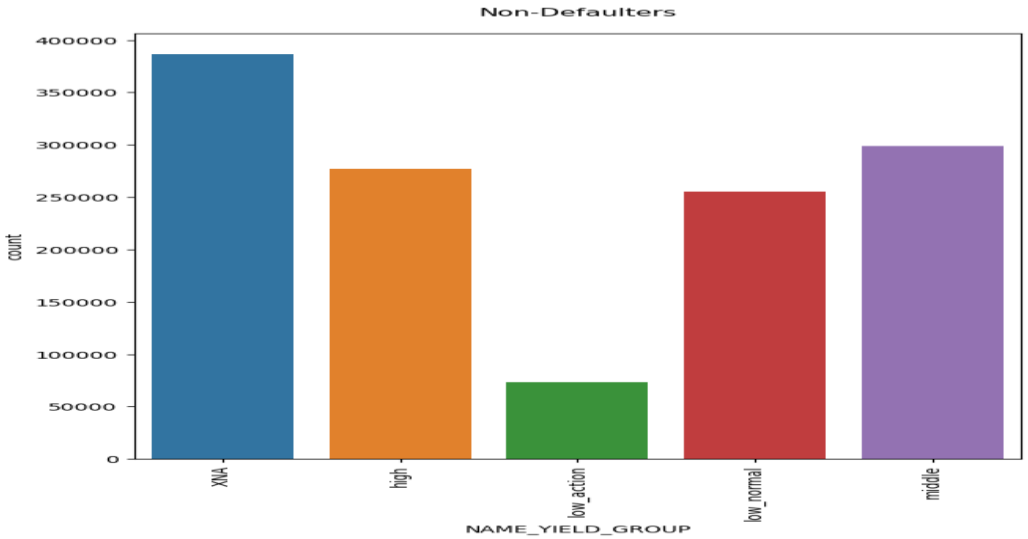


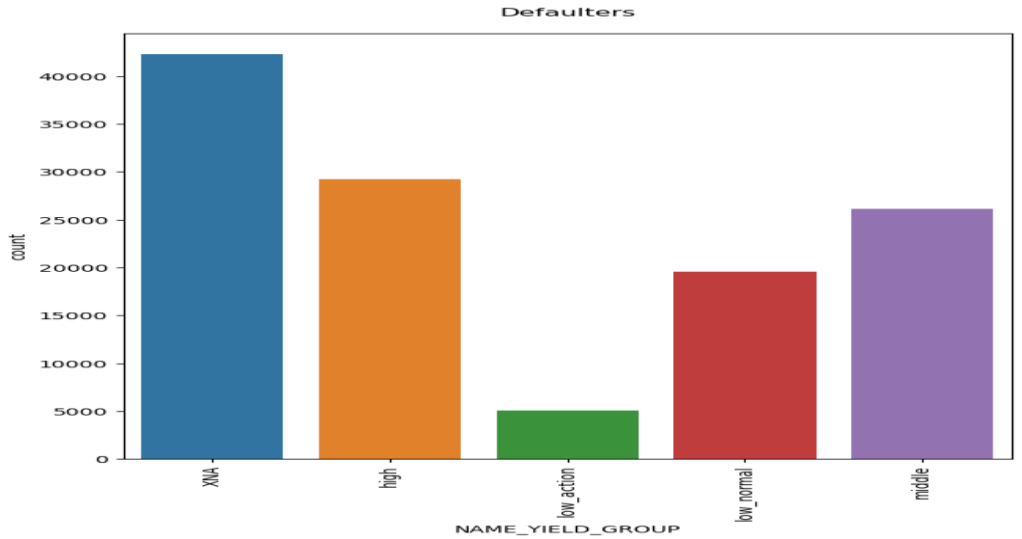
7. NAME_CONTRACT_TYPE_y - for Defaulters Cash loans = 46.74 % so clearly cash loans need to be taken care of.





8. NAME_YIELD_GROUP - for defaulters top two cat of Grouped interest rate are XNA=34.59 %,high=23.87 %



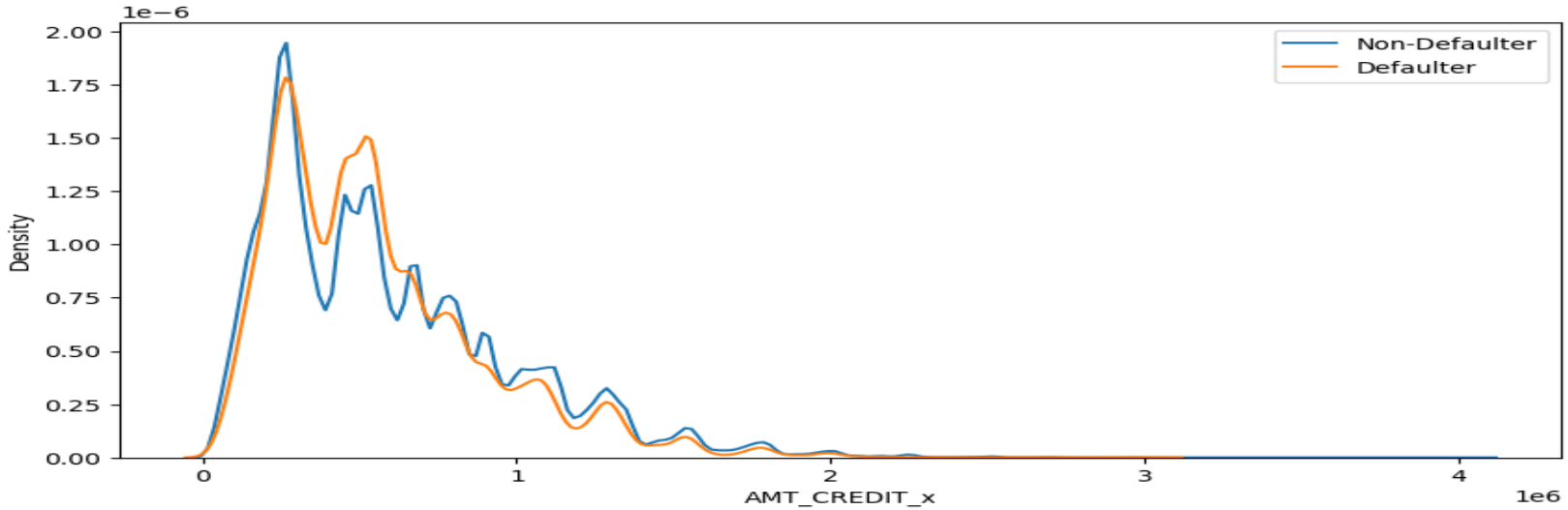


Now we do Univariate Analysis of continous Variables

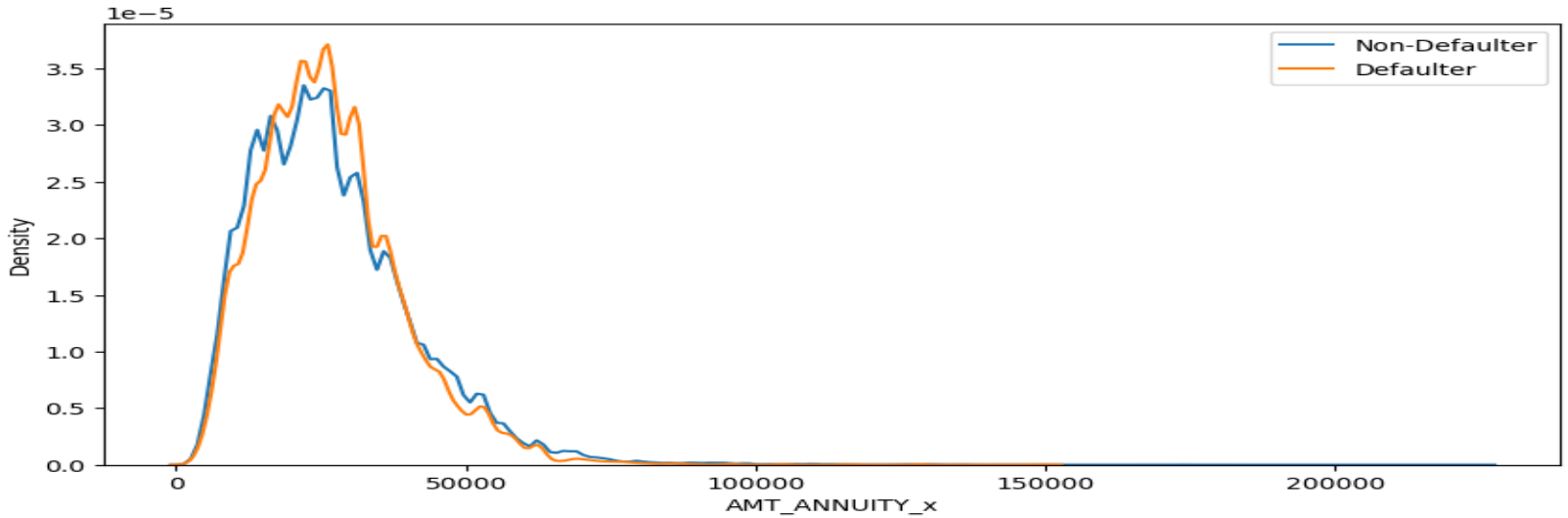
```
# plotting distplot for all the continous columns from count_col_loan for non-defaulters and defaulters
for i in cont_col_loan:
    plt.figure(figsize=[10,5])
    sns.distplot(df_loan_0[i],hist=False,label='Non-Defaulter')
    sns.distplot(df_loan_1[i],hist=False,label='Defaulter')
    plt.legend()
    plt.show()
```

Few points along with distplots which are noted for non- defaulters(TARGET=0) and defaulters(TARGET=1) are as follows:-

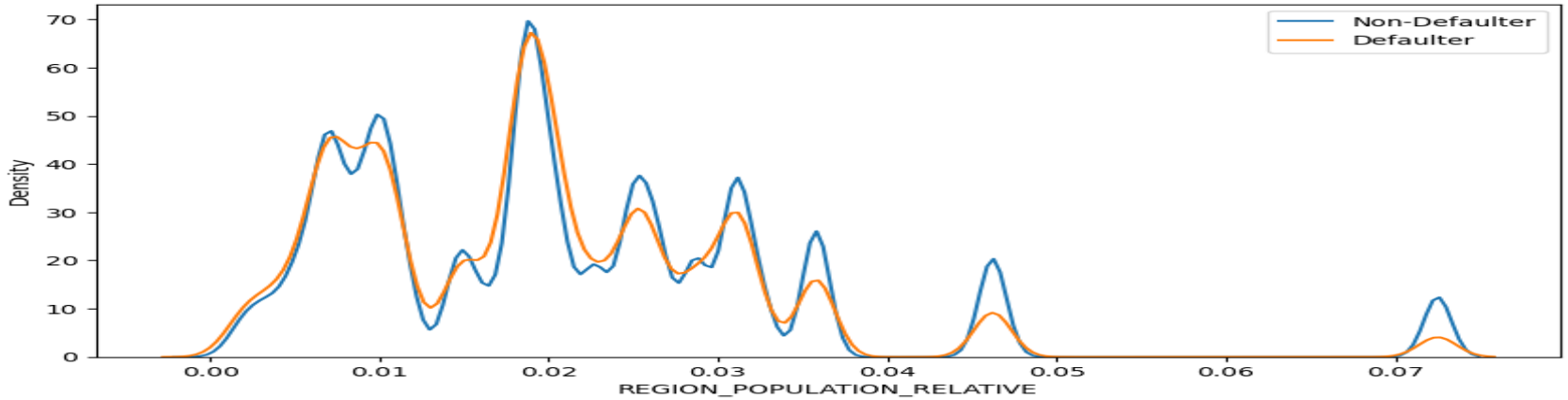
1. AMT_CREDIT_x - Appears lower for TARGET 1, which is a good sign as lesser default loss to the company.



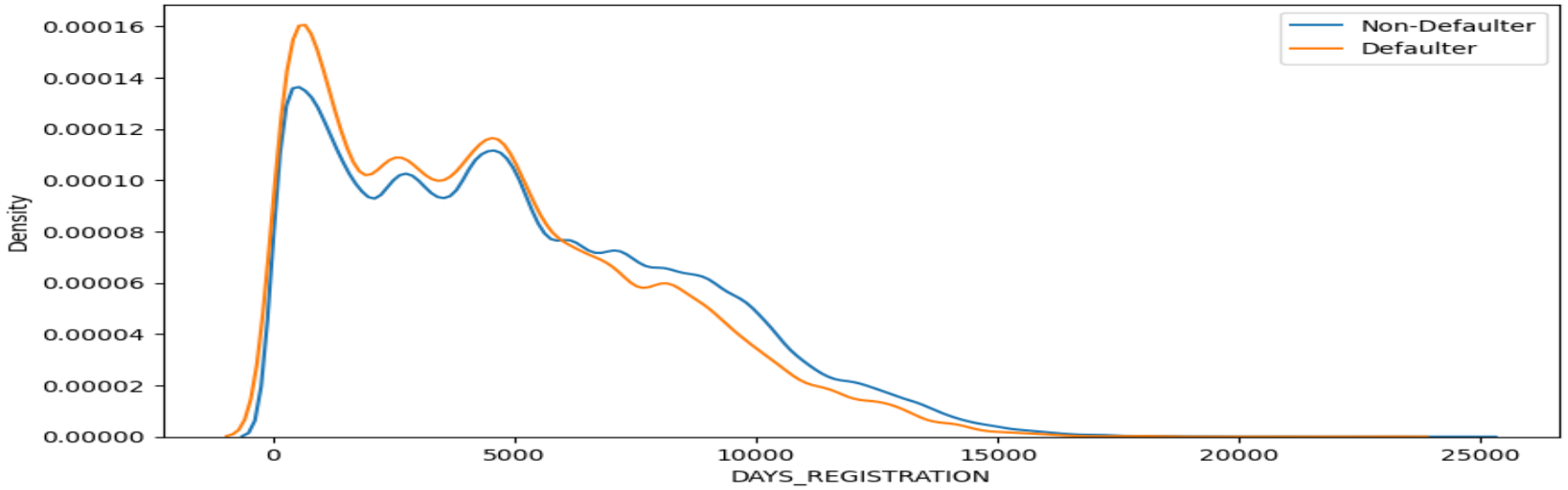
2. AMT_ANNUITY_x - Appears higher for TARGET 1, which shows defaulter gets loans on high emi/intrest than TARGET 0.



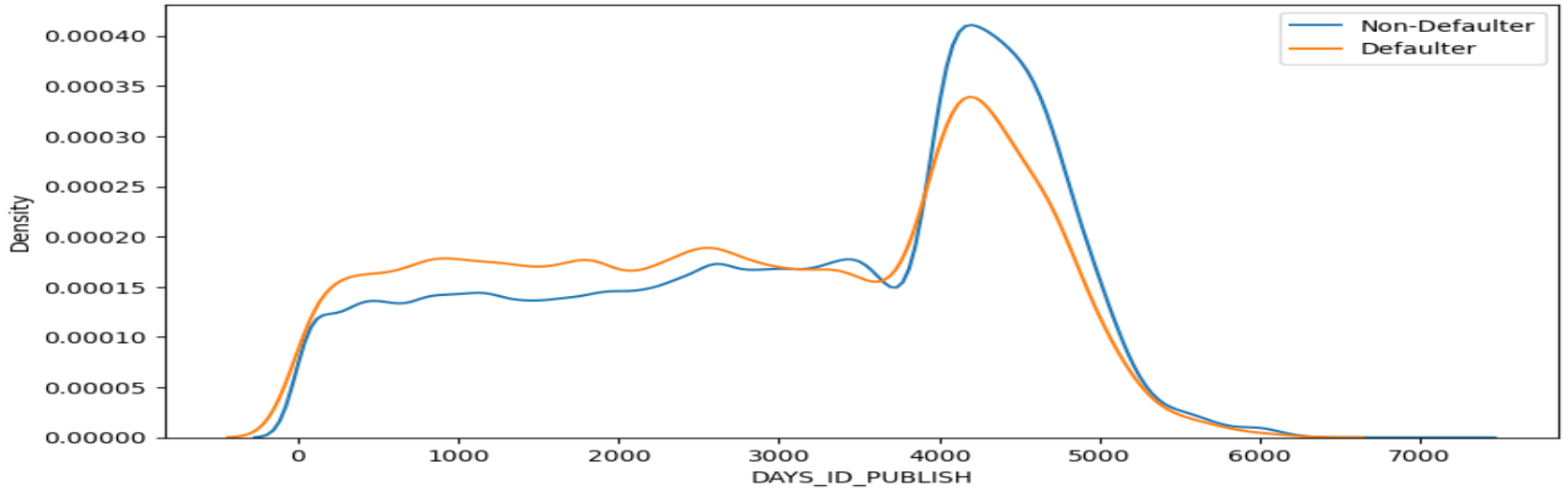
3. REGION_POPULATION_RELATIVE - For low value density for both defaulters and non defaulters is nearly same but as value is increasing density is also decreasing for defaulters.



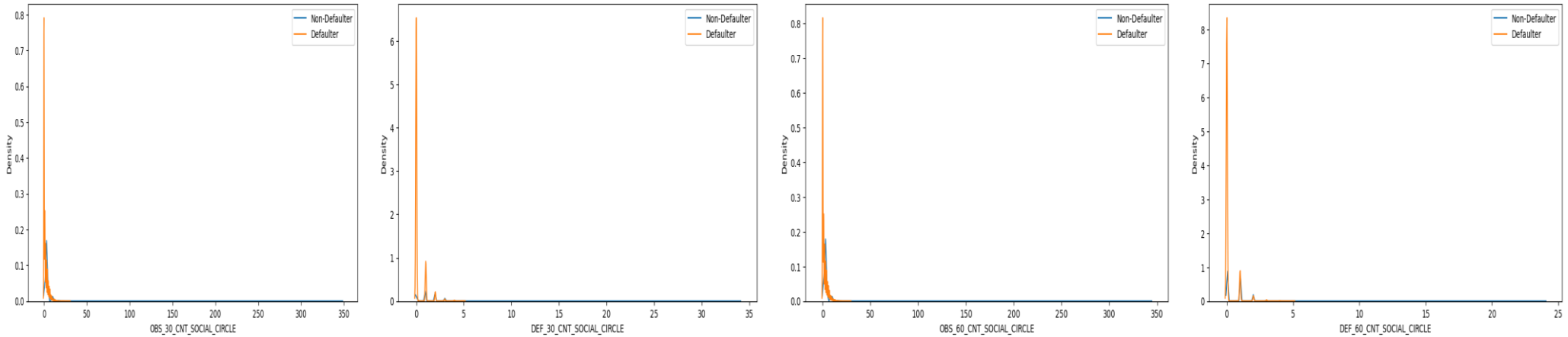
4. DAYS_REGISTRATION - for defaulters density is high for low value of days registration value which clearly shows most defaulters change registation recently.



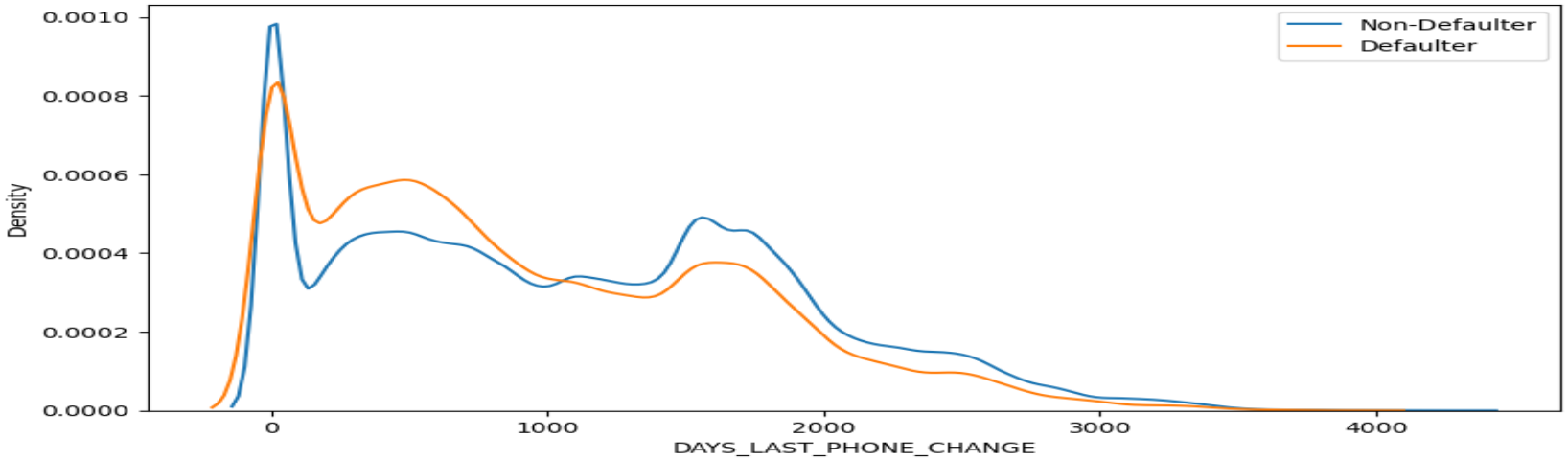
5. DAYS_ID_PUBLISH - for defaulters density is high for low value of days id publish value which clearly shows most defaulters change ID recently.



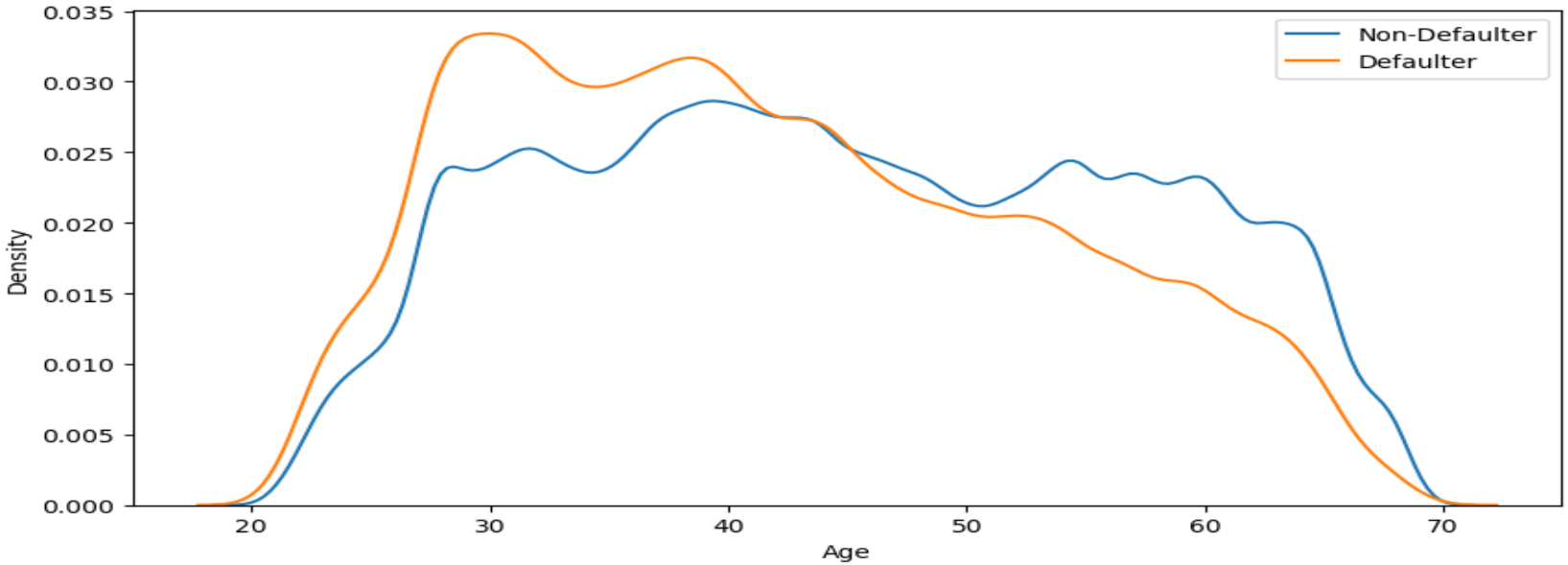
6. OBS_30_CNT_SOCIAL_CIRCLE,DEF_30_CNT_SOCIAL_CIRCLE,OBS_60_CNT_SOCIAL_CIRCLE,DEF_60_CNT_SOCIAL_CIRCLE- all values shows high density spike for defaulters for smaller values .



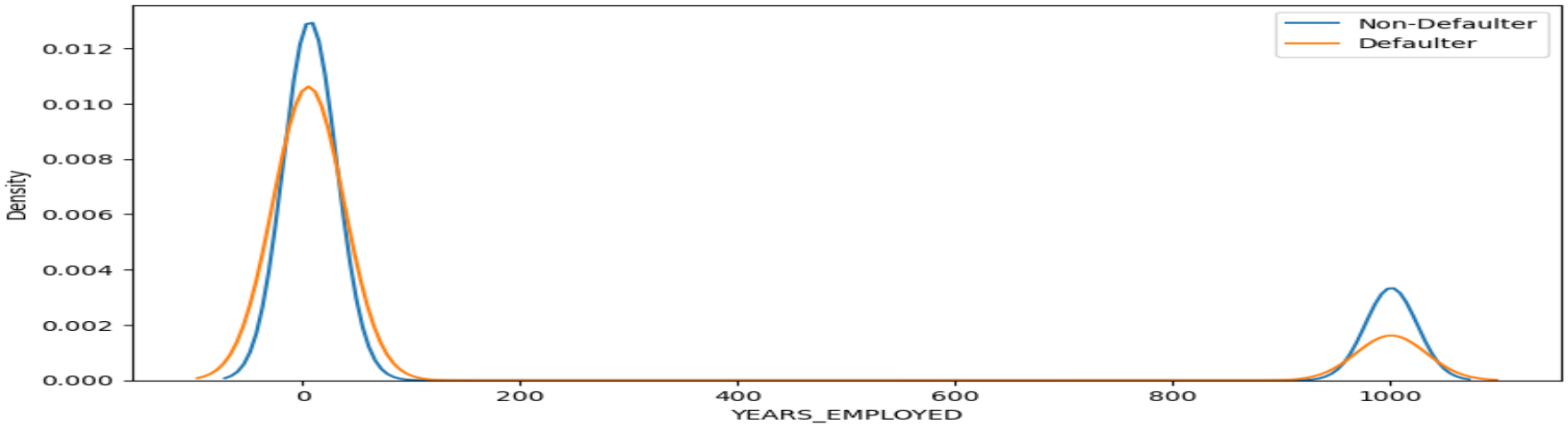
7. DAYS_LAST_PHONE_CHANGE- More people from the Target 1 have changed their phone earlier than Target 1. Indicating intention issues in repaying loan.



8. AGE_IN_YEARS Density of 40 years in Target 1 larger, indicating younger are defaulting more.



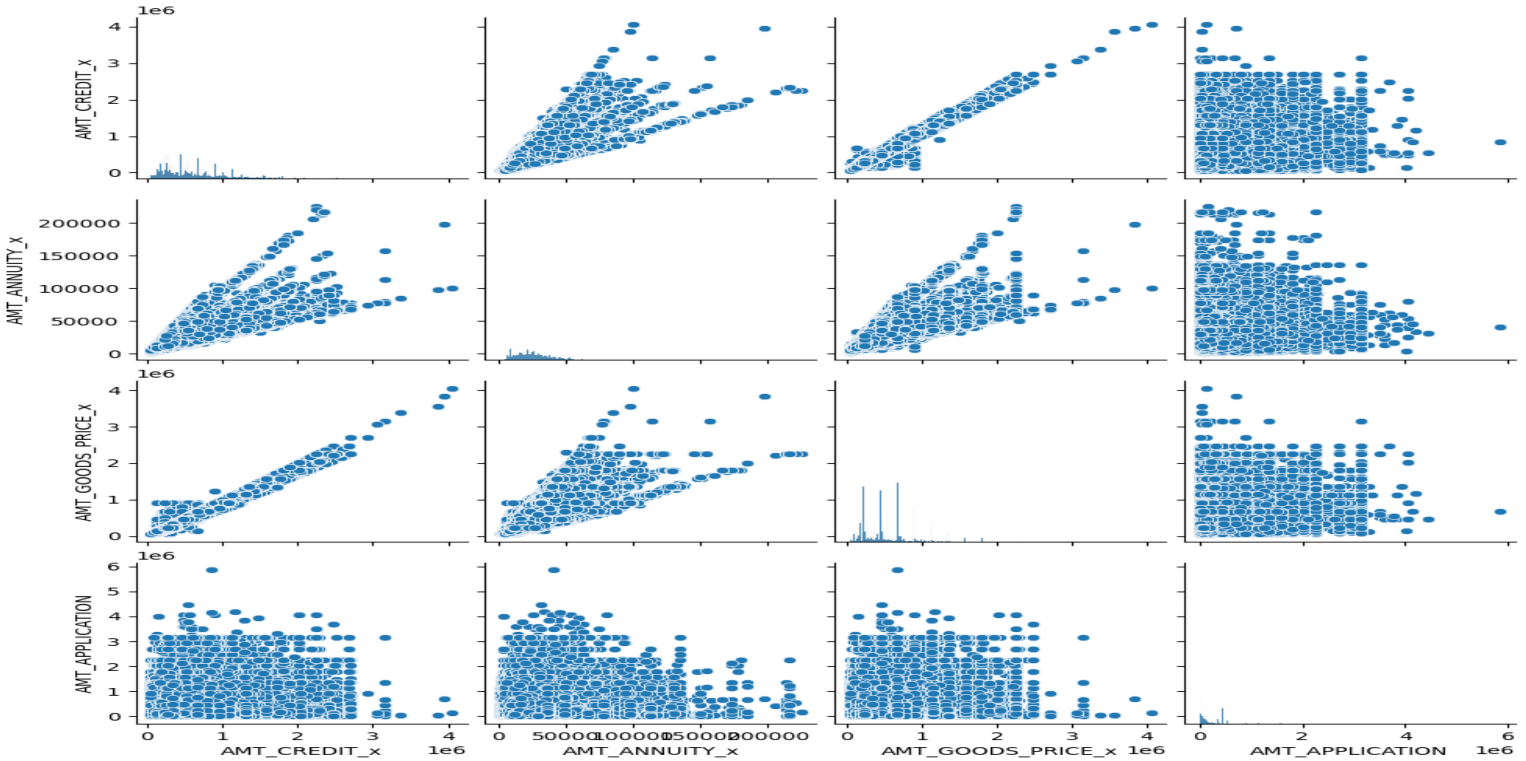
9. YEARS_EMPLOYED for most defaulters years emloyed is less but YEARS_EMPLOYED large no of rows of incorrect data and hence the data representation need to further analysed after imputing outliers



Bivariate Analysis Continous and Continous Variables

Plotting for non defaulters

```
# plotting a pair plot for defaulters df_loan_0 for columns
# 'AMT_CREDIT_x', 'AMT_ANNUITY_x','AMT_GOODS_PRICE_x','AMT_APPLICATION'
plt.figure(figsize=[22,10])
sns.pairplot(df_loan_0[['AMT_CREDIT_x', 'AMT_ANNUITY_x','AMT_GOODS_PRICE_x','AMT_APPLICATION']])
plt.show()
```

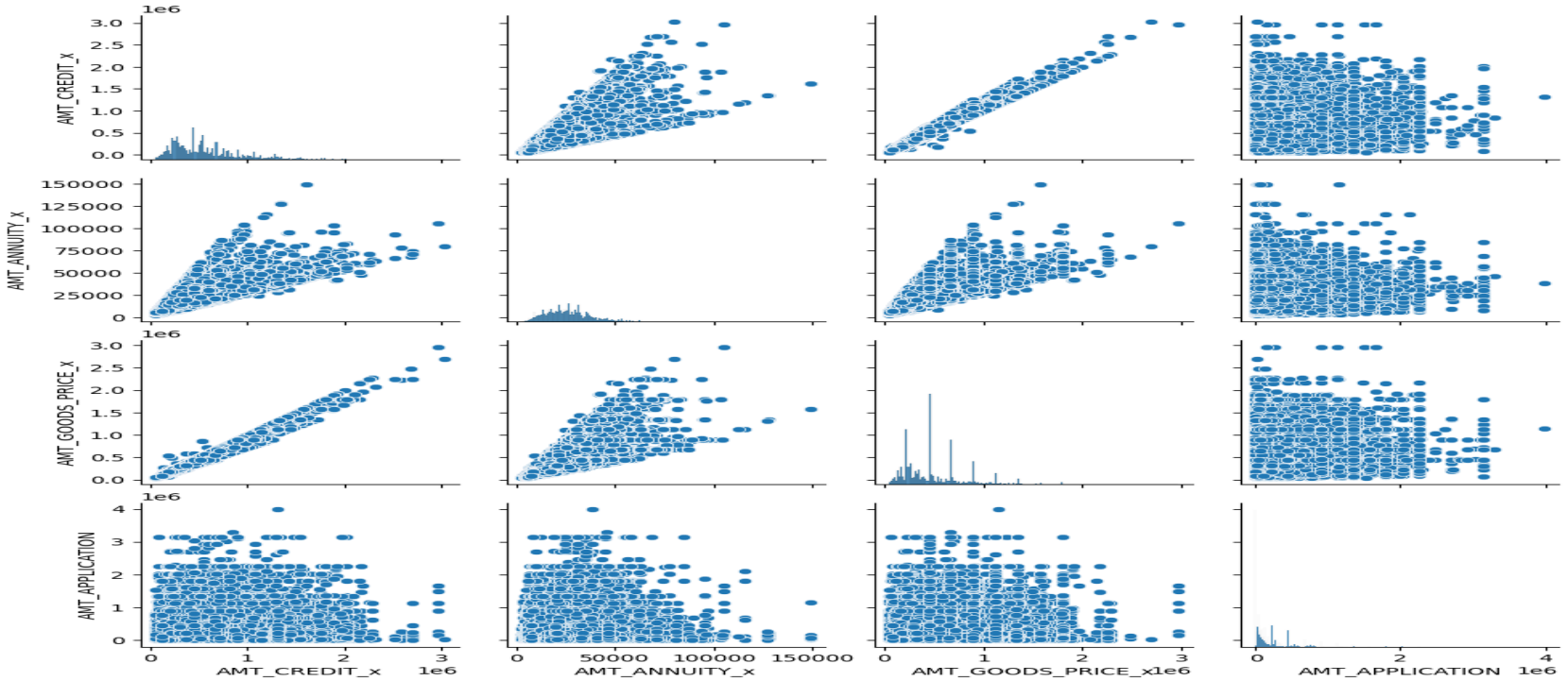


Few points which are noted for non-defaulters are as follows:-

1. AMT_CREDIT_x- has high correlation with AMT_ANNUITY_x,AMT_GOODS_PRICE_x.
2. AMT_ANNUITY_x- has high correlation with AMT_CREDIT_x,AMT_GOODS_PRICE_x.
3. AMT_GOODS_PRICE_x- has high correlation with AMT_CREDIT_x,AMT_ANNUITY_x.

Plotting for defaulters

```
# plotting a pair plot for defaulters df_loan_0 for columns
# 'AMT_CREDIT_x', 'AMT_ANNUITY_x','AMT_GOODS_PRICE_x','AMT_APPLICATION'
plt.figure(figsize=[22,10])
sns.pairplot(df_loan_1[['AMT_CREDIT_x', 'AMT_ANNUITY_x','AMT_GOODS_PRICE_x','AMT_APPLICATION']])
plt.show()
```



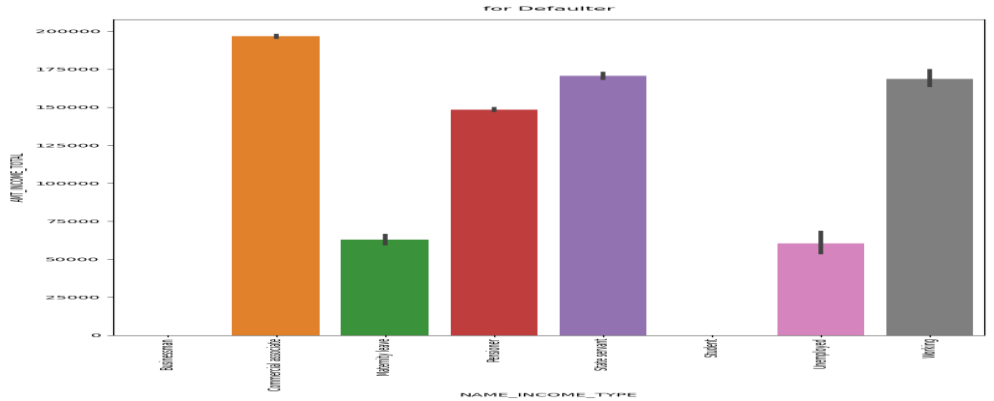
Few points which are noted for defaulters are as follows:-

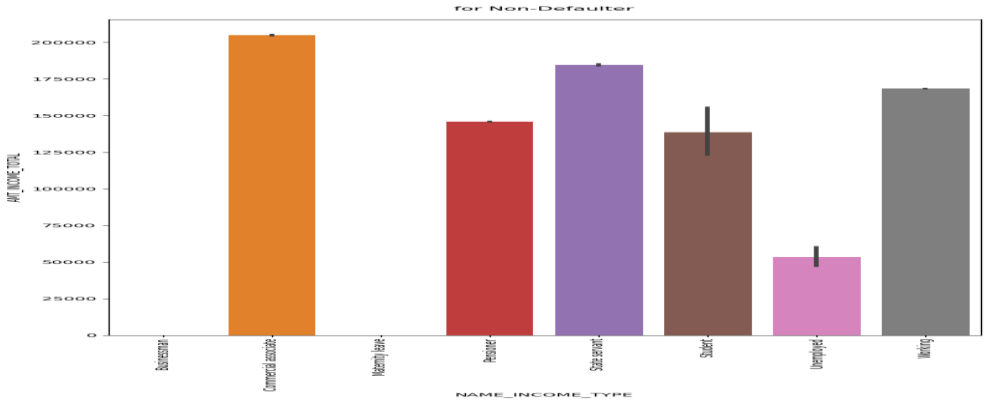
1. AMT_CREDIT_x- has high correlation with AMT_ANNUITY_x,AMT_GOODS_PRICE_x.
2. AMT_ANNUITY_x- has high correlation with AMT_CREDIT_x,AMT_GOODS_PRICE_x.
3. AMT_GOODS_PRICE_x- has high correlation with AMT_CREDIT_x,AMT_ANNUITY_x.

Bivariate Analysis of Continous and Categorial Variables

```
## Plotting for NAME_INCOME_TYPE,AMT_INCOME_TOTAL
bivariate_cat_cont(df_loan_1,df_loan_0,'NAME_INCOME_TYPE','AMT_INCOME_TOTAL')
```

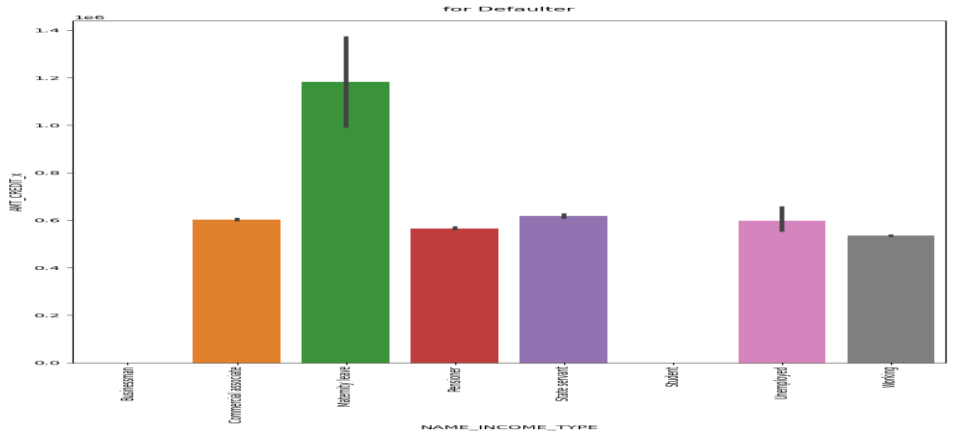
1. NAME_INCOME_TYPE & AMT_INCOME_TOTAL
- for defaulters NAME_INCOME_TYPE Maternity leave is having high compare than non-defaulters.

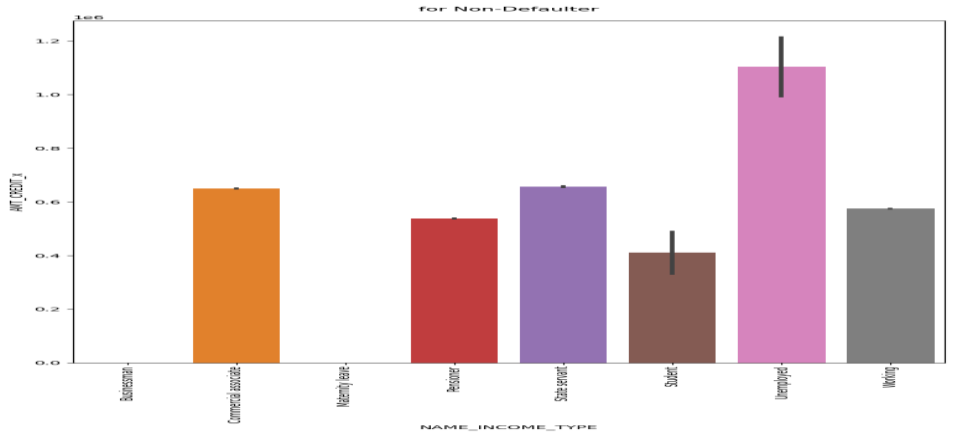





```
# Plotting for NAME_INCOME_TYPE,AMT_CREDIT_x
bivariate_cat_cont(df_loan_1,df_loan_0,'NAME_INCOME_TYPE','AMT_CREDIT_x')
```

2. NAME_INCOME_TYPE & AMT_CREDIT_x
- for defaulters NAME_INCOME_TYPE Maternity leave is having highest AMT_CREDIT_x as compare to other NAME_INCOME_TYPE.
- for non-defaulters NAME_INCOME_TYPE Unemployed is having highest AMT_CREDIT_x as compare to other NAME_INCOME_TYPE.

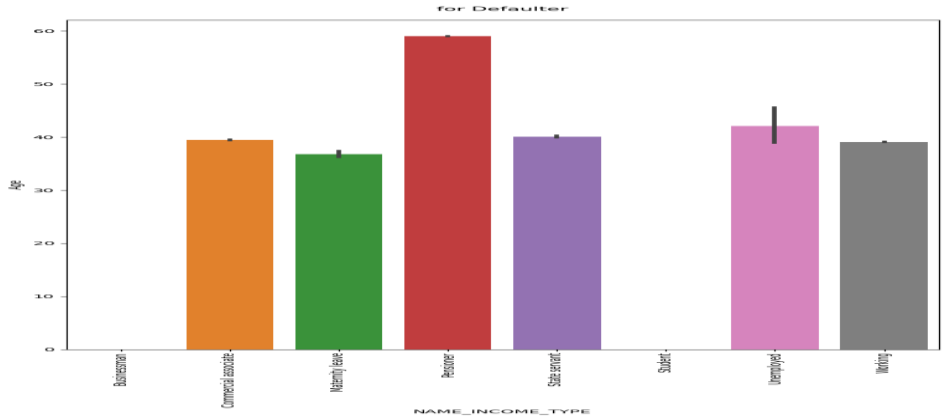


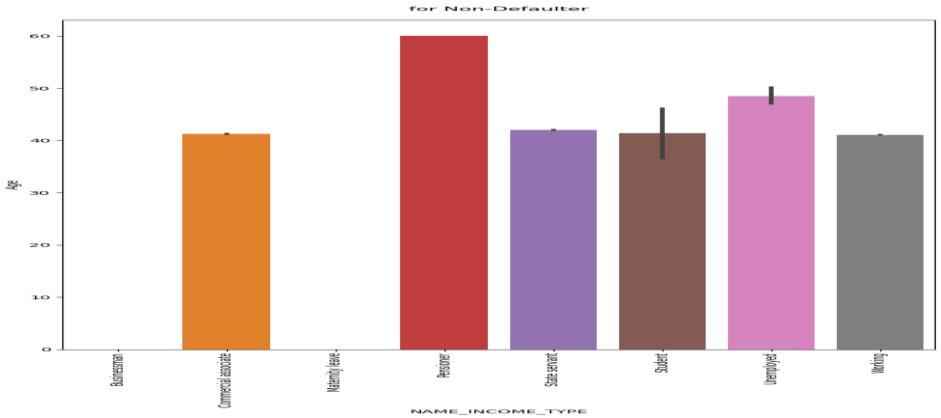


```
# Plotting for NAME_INCOME_TYPE,Age
bivariate_cat_cont(df_loan_1,df_loan_0,'NAME_INCOME_TYPE','Age')
```

3. NAME_INCOME_TYPE & AGE

- for defaulters NAME_INCOME_TYPE Maternity leave is having high % as compare to non-defaulters NAME_INCOME_TYPE Maternity leave

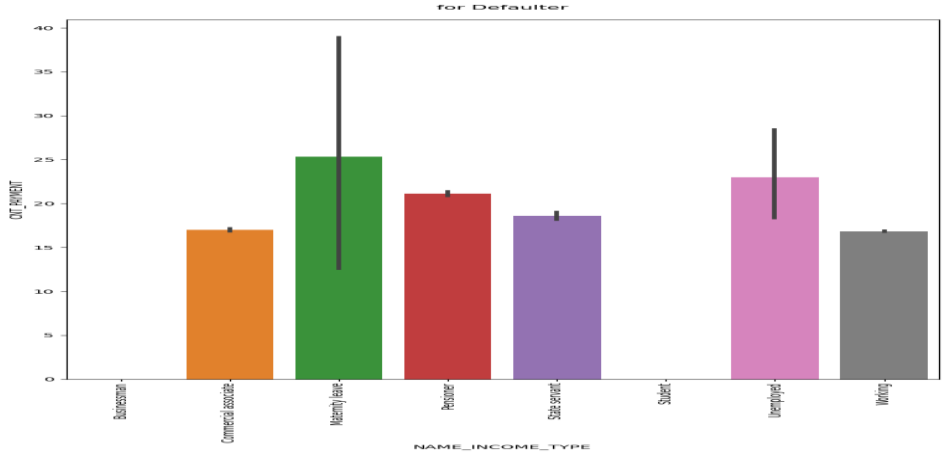


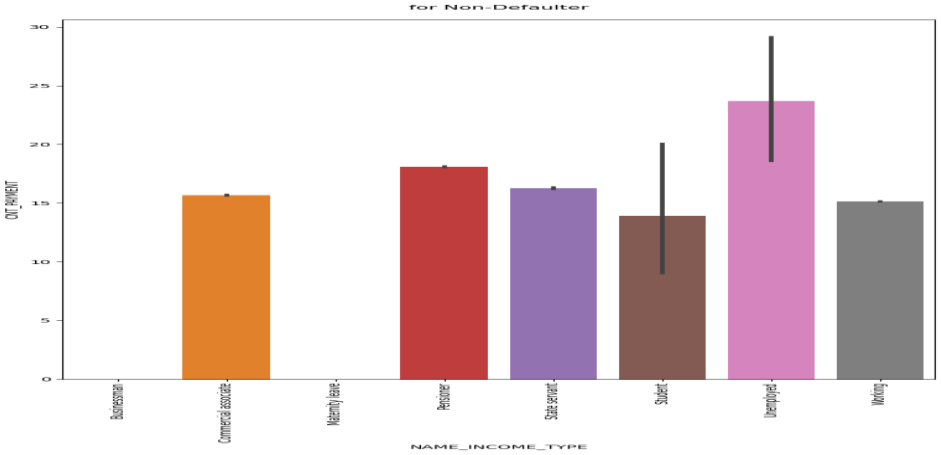



```
# Plotting for NAME_INCOME_TYPE,CNT_PAYMENT
bivariate_cat_cont(df_loan_1,df_loan_0,'NAME_INCOME_TYPE','CNT_PAYMENT')
```

4. NAME_INCOME_TYPE & CNT_PAYMENT

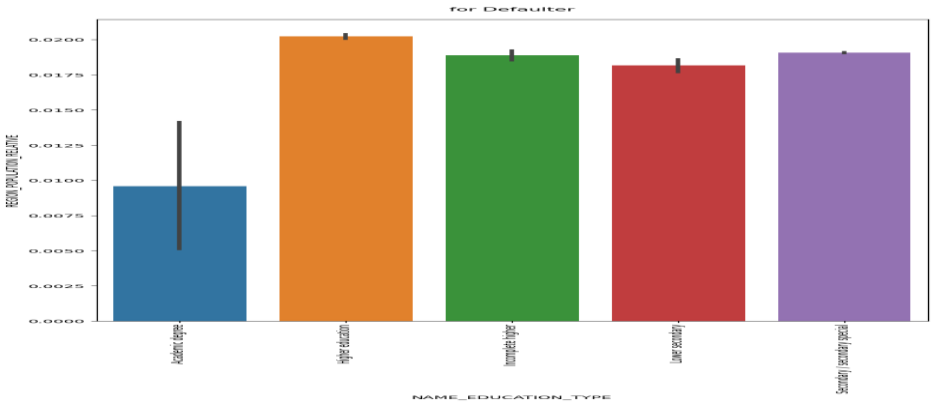
- for defaulters NAME_INCOME_TYPE Maternity leave is having high value for CNT_PAYMENT as compare to other NAME_INCOME_TYPE.
- for non-defaulters NAME_INCOME_TYPE Unemployed is having high CNT_PAYMENT as compare to other NAME_INCOME_TYPE

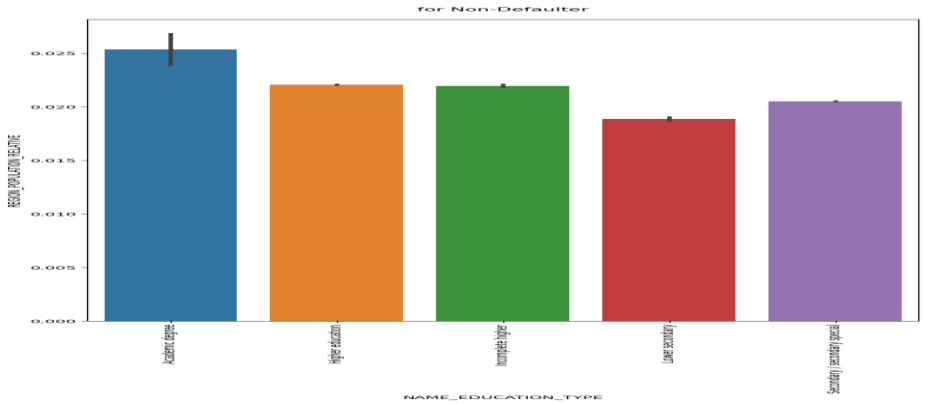




```
# Plotting for NAME_EDUCATION_TYPE,REGION_POPULATION_RELATIVE
bivariate_cat_cont(df_loan_1,df_loan_0,'NAME_EDUCATION_TYPE' ,'REGION_POPULATION_RELATIVE')
```

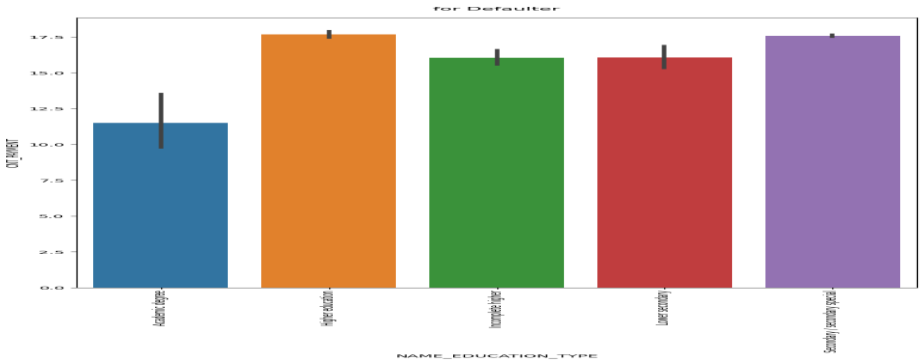
5. NAME_EDUCATION_TYPE & REGION_POPULATION_RELATIVE
- for defaulters NAME_EDUCATION_TYPE Higher education is having high REGION_POPULATION_RELATIVE is highest % as compare to other NAME_EDUCATION_TYPE.
- for non-defaulters NAME_EDUCATION_TYPE Unemployed is having high REGION_POPULATION_RELATIVE is highest % as compare to other NAME_EDUCATION_TYPE.

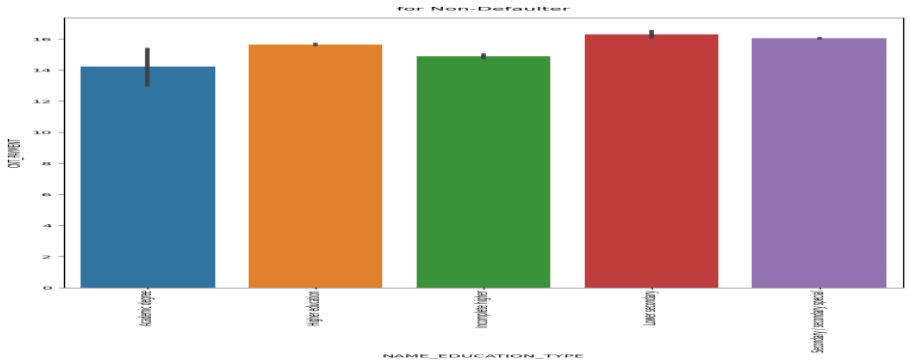





```
# Plotting for NAME_EDUCATION_TYPE,CNT_PAYMENT
bivariate_cat_cont(df_loan_1,df_loan_0,'NAME_EDUCATION_TYPE','CNT_PAYMENT')
```

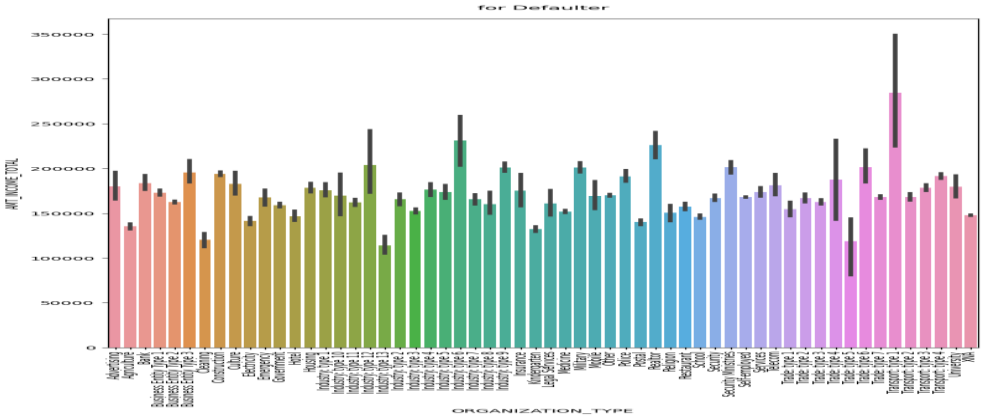
6. NAME_EDUCATION_TYPE & CNT_PAYMENT
- for defaulters NAME_EDUCATION_TYPE Higher education is having highest CNT_PAYMENT as compare to other NAME_EDUCATION_TYPE.
- for non-defaulters NAME_EDUCATION_TYPE Lower secondary is having highest CNT_PAYMENT as compare to other NAME_EDUCATION_TYPE

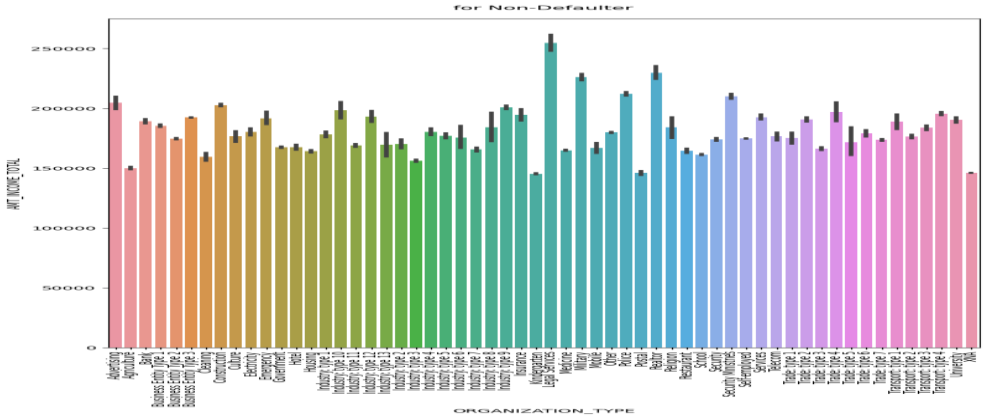




```
# Plotting for ORGANIZATION_TYPE,AMT_INCOME_TOTAL
bivariate_cat_cont(df_loan_1,df_loan_0,'ORGANIZATION_TYPE','AMT_INCOME_TOTAL')
```

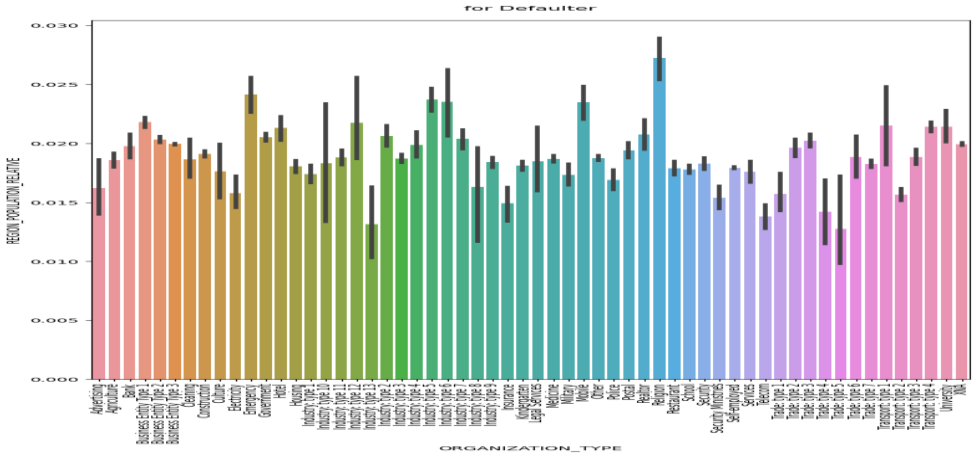
7. ORGANIZATION_TYPE & AMT_INCOME_TOTAL
- for defaulters ORGANIZATION_TYPE Transport-type 1 is having high AMT_INCOME_TOTAL as compare to other ORGANIZATION_TYPE.
- for non-defaulters ORGANIZATION_TYPE Legal services is having high AMT_INCOME_TOTAL as compare to other ORGANIZATION_TYPE.

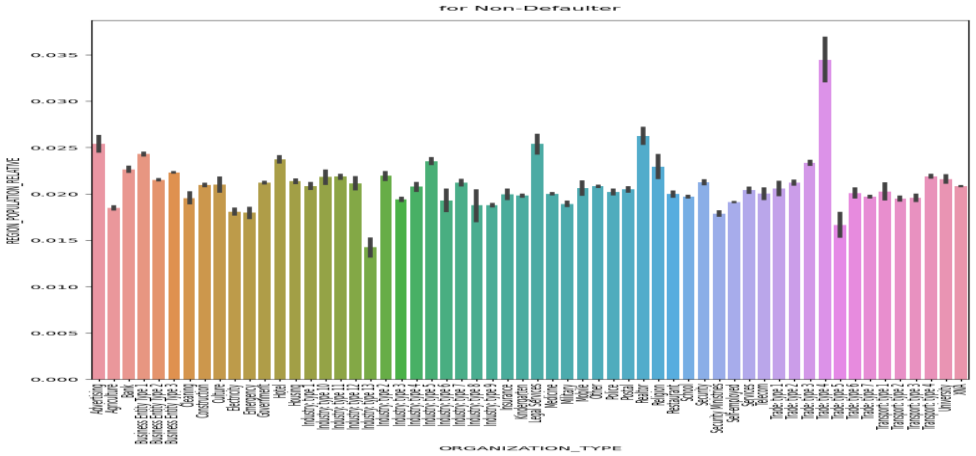





```
# Plotting for ORGANIZATION_TYPE,REGION_POPULATION_RELATIVE
bivariate_cat_cont(df_loan_1,df_loan_0,'ORGANIZATION_TYPE','REGION_POPULATION_RELATIVE')
```

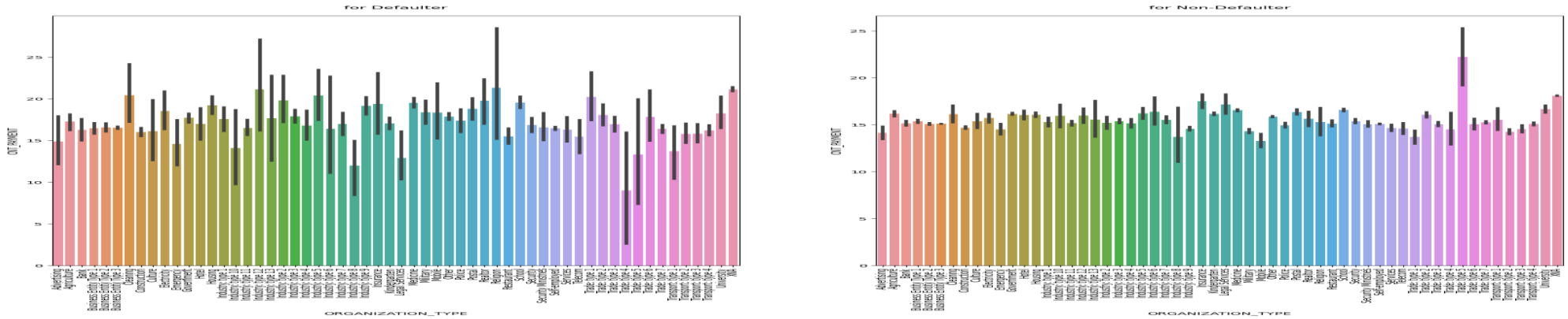
8. ORGANIZATION_TYPE & REGION_POPULATION_RELATIVE
- for defaulters ORGANIZATION_TYPE Religion is having high REGION_POPULATION_RELATIVE as compare to other ORGANIZATION_TYPE.
- for non-defaulters ORGANIZATION_TYPE Trade-type 4 is having high REGION_POPULATION_RELATIVE as compare to other ORGANIZATION_TYPE





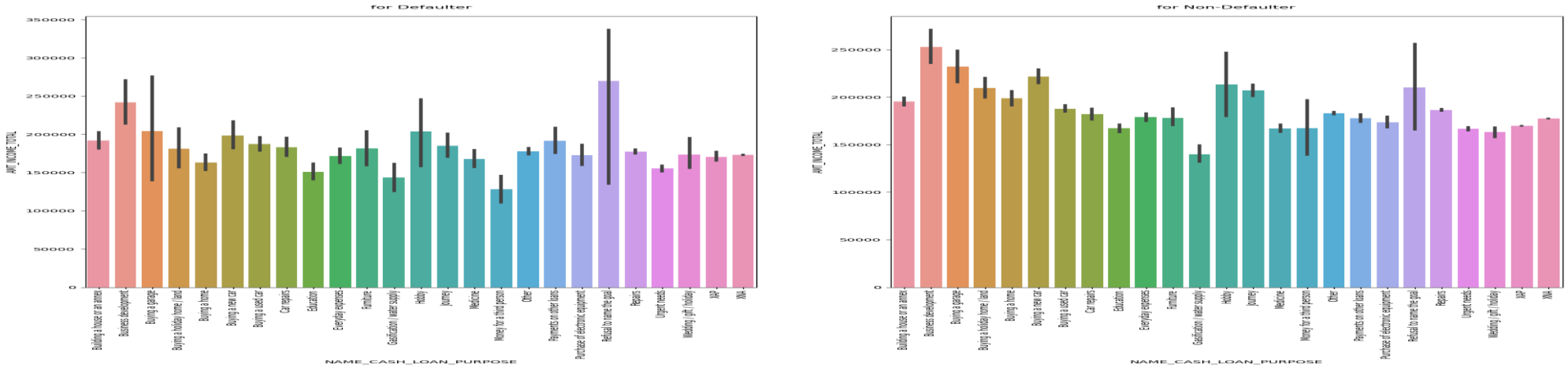
```
# Plotting for ORGANIZATION_TYPE,CNT_PAYMENT
bivariate_cat_cont(df_loan_1,df_loan_0,'ORGANIZATION_TYPE','CNT_PAYMENT')
```

9. ORGANIZATION_TYPE & CNT_PAYMENT
- for defaulters ORGANIZATION_TYPE Religion, Industry -type 12 is having highe CNT_PAYMENT as compare to other ORGANIZATION_TYPE.
- for non-defaulters ORGANIZATION_TYPE Trade-type 5 is having highe CNT_PAYMENT as compare to other ORGANIZATION_TYPE




```
# Plotting for NAME_CASH_LOAN_PURPOSE,AMT_INCOME_TOTAL
bivariate_cat_cont(df_loan_1,df_loan_0,'NAME_CASH_LOAN_PURPOSE','AMT_INCOME_TOTAL')
```

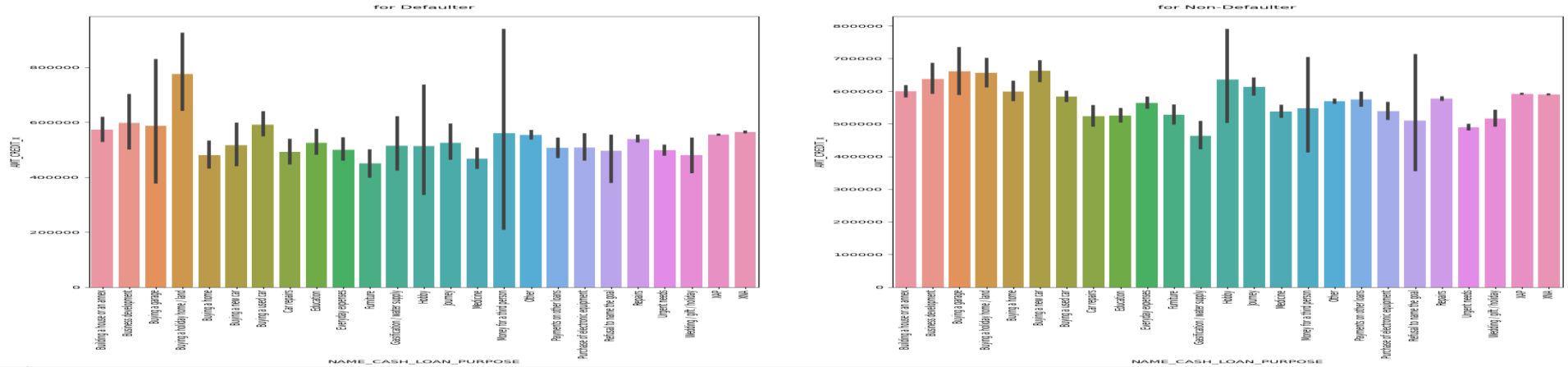
10. NAME_CASH_LOAN_PURPOSE & AMT_INCOME_TOTAL
- for defaulters NAME_CASH_LOAN_PURPOSE Refuse to name the goal is having high AMT_INCOME_TOTAL as compare to other NAME_CASH_LOAN_PURPOSE.
- for non-defaulters NAME_CASH_LOAN_PURPOSE Buying a garage is having highe AMT_INCOME_TOTAL as compare to other NAME_CASH_LOAN_PURPOSE.



```
# Plotting for NAME_CASH_LOAN_PURPOSE,AMT_CREDIT_x
bivariate_cat_cont(df_loan_1,df_loan_0,'NAME_CASH_LOAN_PURPOSE','AMT_CREDIT_x')
```

11. NAME_CASH_LOAN_PURPOSE & AMT_CREDIT_x

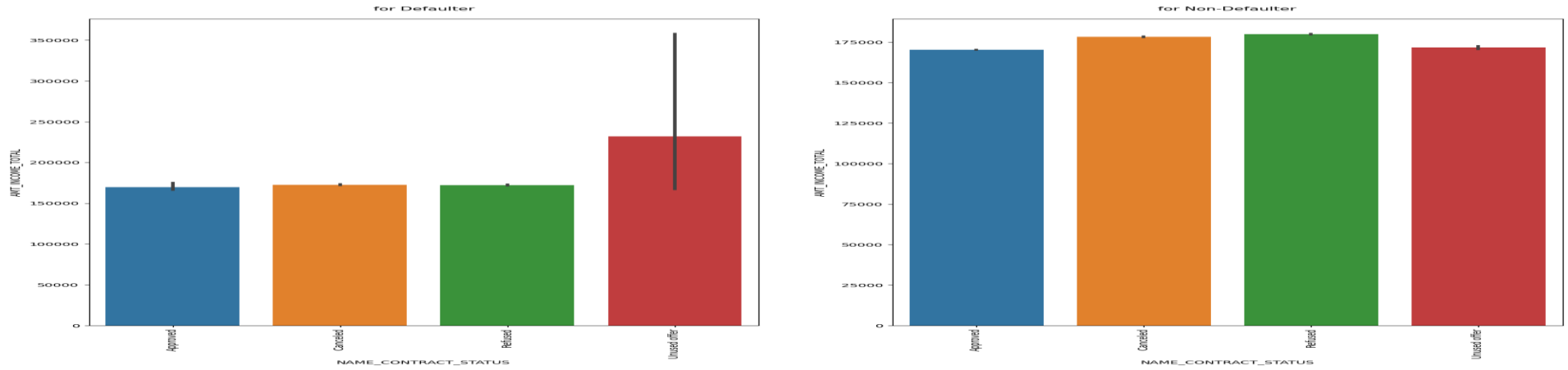
- for defaulters NAME_CASH_LOAN_PURPOSE Buying a holiday home is having high AMT_CREDIT_x as compare to other NAME_CASH_LOAN_PURPOSE.
- for non-defaulters NAME_CASH_LOAN_PURPOSE buying a new car,buying a garage is having highe AMT_CREDIT_x as compare to other NAME_CASH_LOAN_PURPOSE




```
# Plotting for NAME_CONTRACT_STATUS,AMT_INCOME_TOTAL
bivariate_cat_cont(df_loan_1,df_loan_0,'NAME_CONTRACT_STATUS','AMT_INCOME_TOTAL')
```

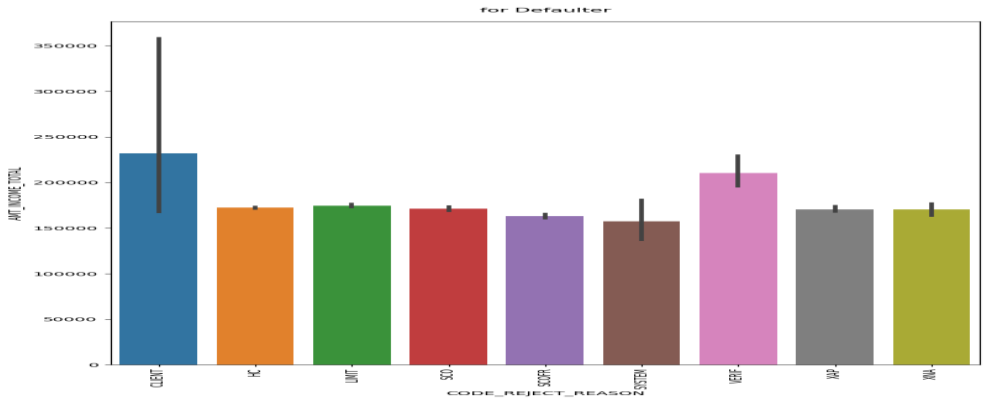
12. NAME_CONTRACT_STATUS & AMT_INCOME_TOTAL

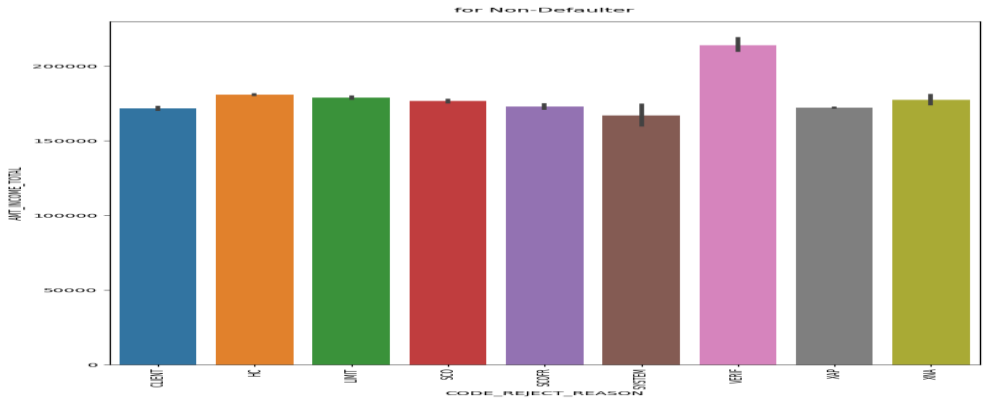
- for defaulters NAME_CONTRACT_STATUS Unused offer is having high AMT_INCOME_TOTAL as compare to other NAME_CONTRACT_STATUS.
- for non-defaulters NAME_CONTRACT_STATUS Refushed is slightly high AMT_INCOME_TOTAL as compare to other NAME_CONTRACT_STATUS.



```
# Plotting for CODE_REJECT_REASON,AMT_INCOME_TOTAL
bivariate_cat_cont(df_loan_1,df_loan_0,'CODE_REJECT_REASON','AMT_INCOME_TOTAL')
```

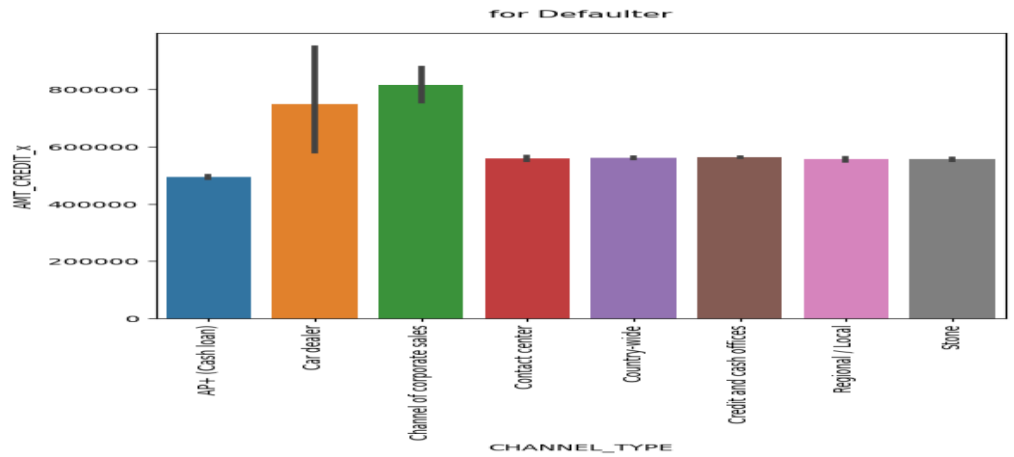
13. CODE_REJECT_REASON & AMT_INCOME_TOTAL
- for defaulters CODE_REJECT_REASON Client is having highe AMT_INCOME_TOTAL as compare to other CODE_REJECT_REASON.
- for non-defaulters CODE_REJECT_REASON Verif is having high AMT_INCOME_TOTAL as compare to other CODE_REJECT_REASON.

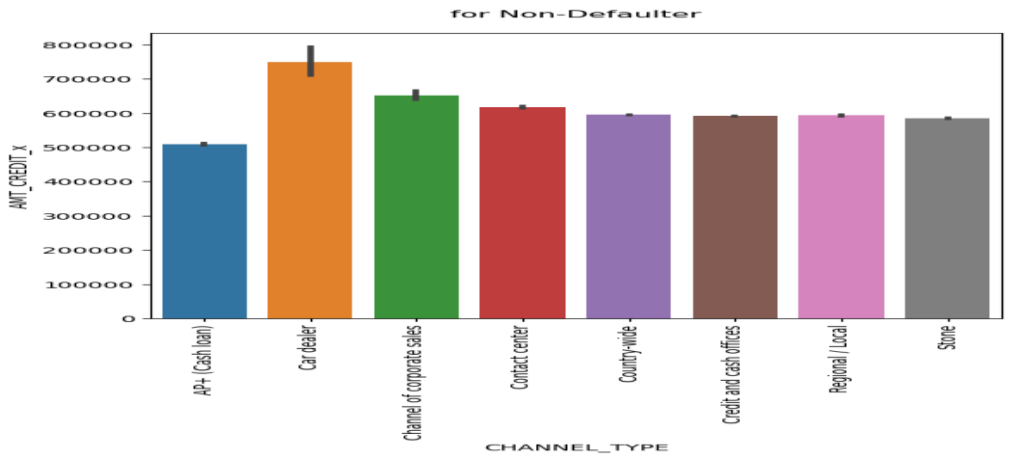





```
# Plotting for CHANNEL_TYPE,AMT_CREDIT_x
bivariate_cat_cont(df_loan_1,df_loan_0,'CHANNEL_TYPE','AMT_CREDIT_x')
```

14. CHANNEL_TYPE & AMT_CREDIT_x
- for defaulters CHANNEL_TYPE Channel of corporate sales is having high AMT_CREDIT_x as compare to other CHANNEL_TYPE.
- for non-defaulters CHANNEL_TYPE Car dealer is having high AMT_CREDIT_x as compare to other CHANNEL_TYPE.

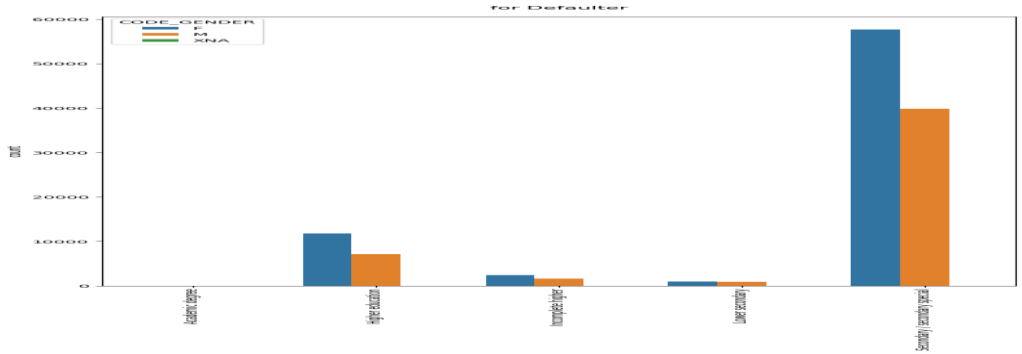


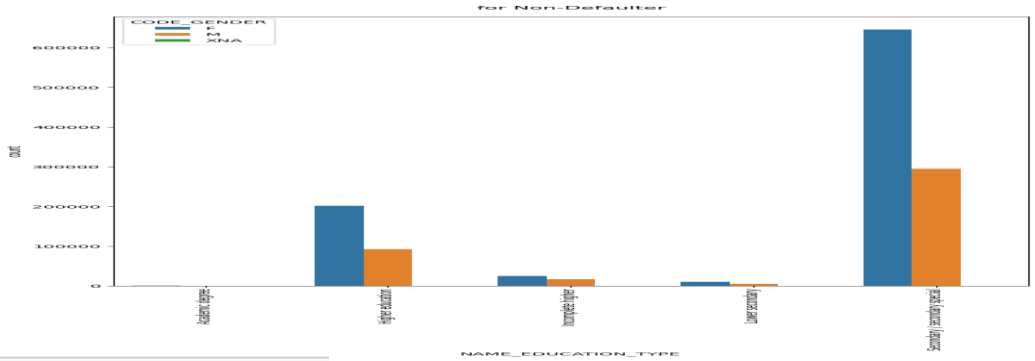


Bivariate Analysis for Categorical and Categorical Variables

```
# Plotting for NAME_EDUCATION_TYPE,CODE_GENDER
bivariate_cat_cat(df_loan_1,df_loan_0,'NAME_EDUCATION_TYPE','CODE_GENDER')
```

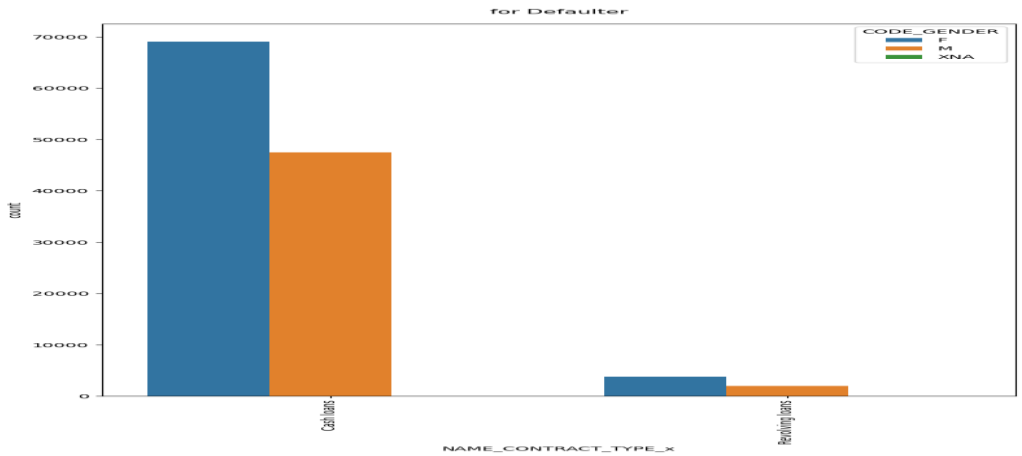
1. NAME_EDUCATION_TYPE & CODE_GENDER
for defaulters NAME_EDUCATION_TYPE Secondary/secondary special is having low ratio of F/M as compare to non-defaulters NAME_EDUCATION_TYPE Secondary/secondary special,hence it s evident that males are defaulting more

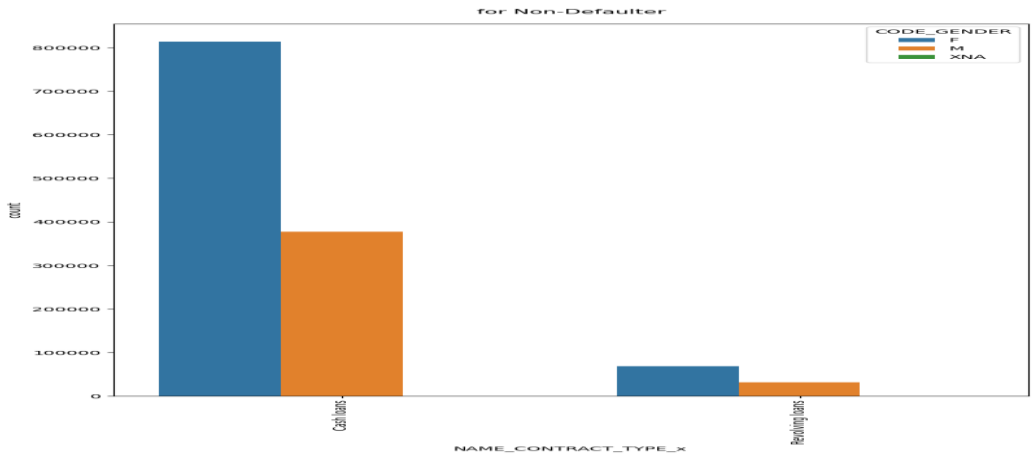





```
# Plotting for NAME_CONTRACT_TYPE_x ,CODE_GENDER
bivariate_cat_cat(df_loan_1,df_loan_0,'NAME_CONTRACT_TYPE_x' ,'CODE_GENDER')
```

2. NAME_CONTRACT_TYPE_x & CODE_GENDER
for defaulters NAME_CONTRACT_TYPE_x is having low ratio for cash loans as compare to non-defaulters, hence it s evident that males are defaulting more.

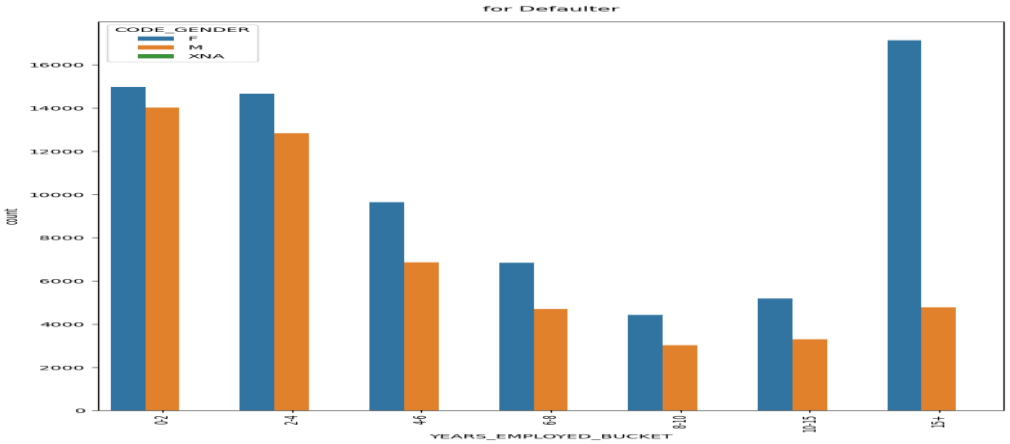


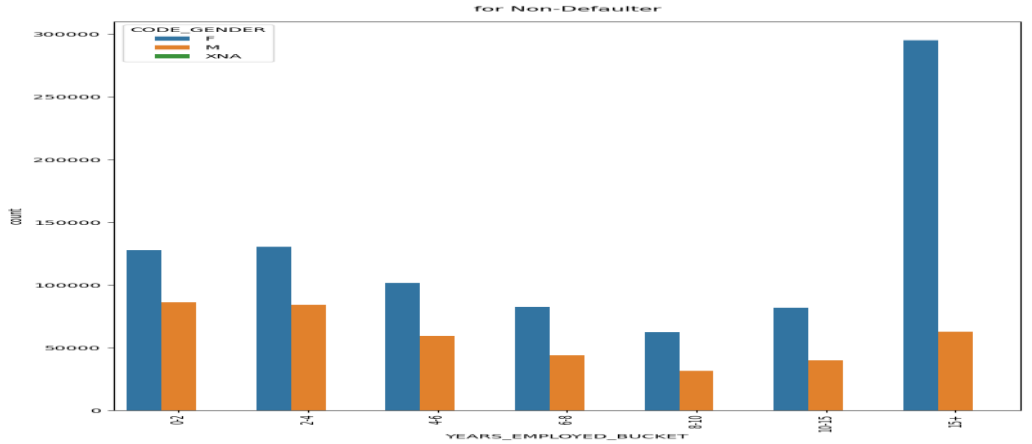


```
# plotting for YEARS_EMPLOYED_BUCKET,CODE_GENDER
bivariate_cat_cat(df_loan_1,df_loan_0,'YEARS_EMPLOYED_BUCKET','CODE_GENDER')
```

3. YEARS_EMPLOYED_BUCKET & CODE_GENDER

for defaulters YEARS_EMPLOYED_BUCKET for 0-2,2-4 are deafulting most as compare to other YEARS_EMPLOYED_BUCKET values.

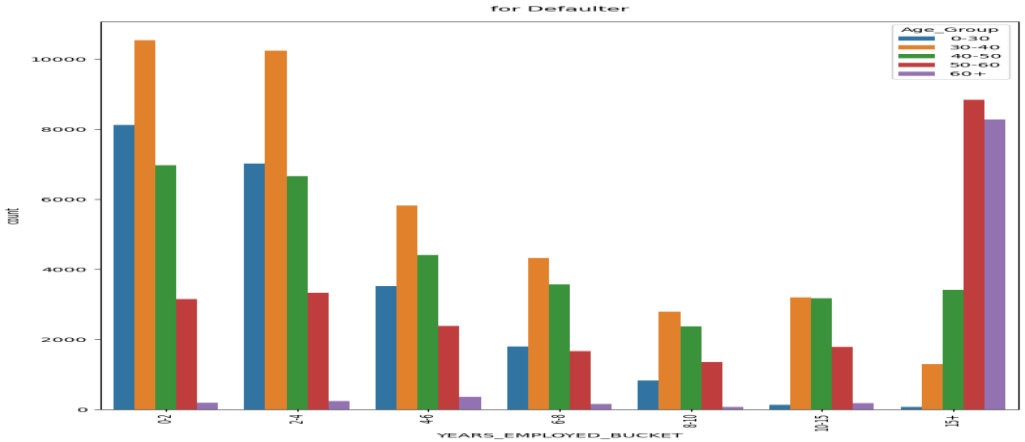


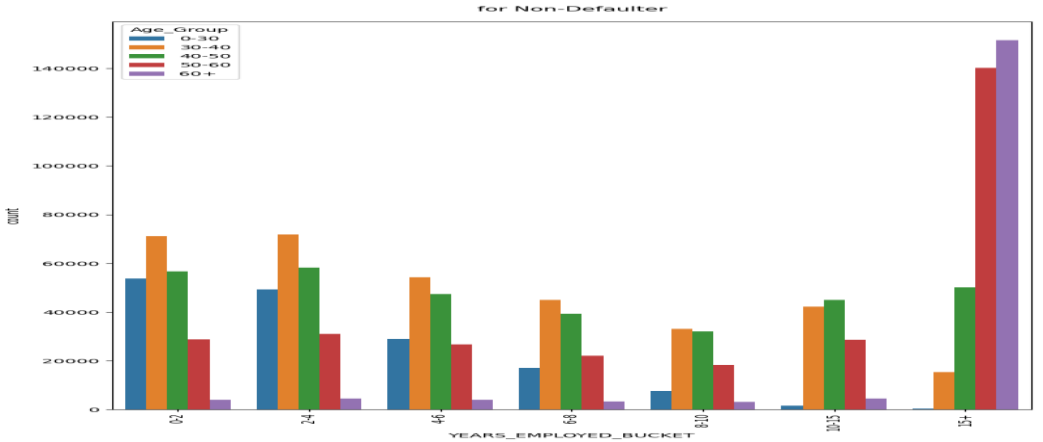



```
# Plotting for YEARS_EMPLOYED_BUCKET,Age_Group
bivariate_cat_cat(df_loan_1,df_loan_0,'YEARS_EMPLOYED_BUCKET','Age_Group')
```

4. YEARS_EMPLOYED_BUCKET & Age_Group

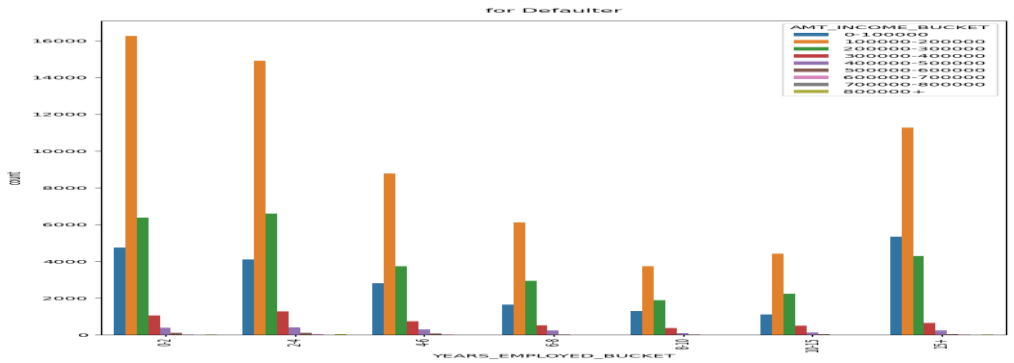
for defaulters YEARS_EMPLOYED_BUCKET 0-2,2-4 and Age_Group of 0-30,30-40 are more defaulters as compare to other YEARS_EMPLOYED_BUCKET and age group.

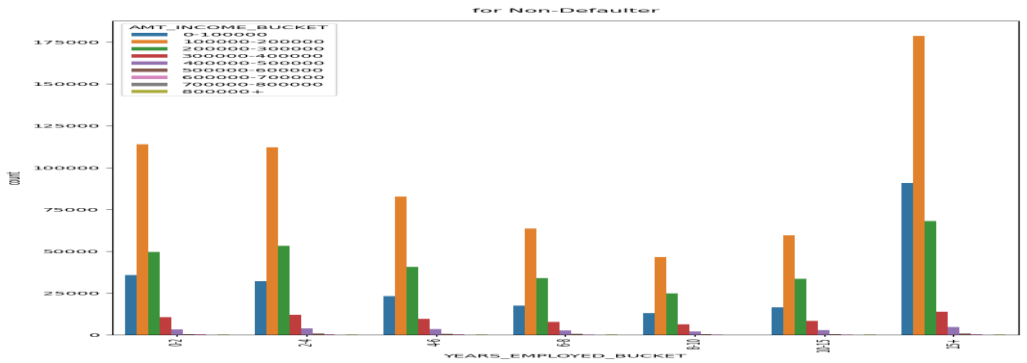




```
# Plotting for YEARS_EMPLOYED_BUCKET ,AMT_INCOME_BUCKET
bivariate_cat_cat(df_loan_1,df_loan_0,'YEARS_EMPLOYED_BUCKET' ,'AMT_INCOME_BUCKET')
```

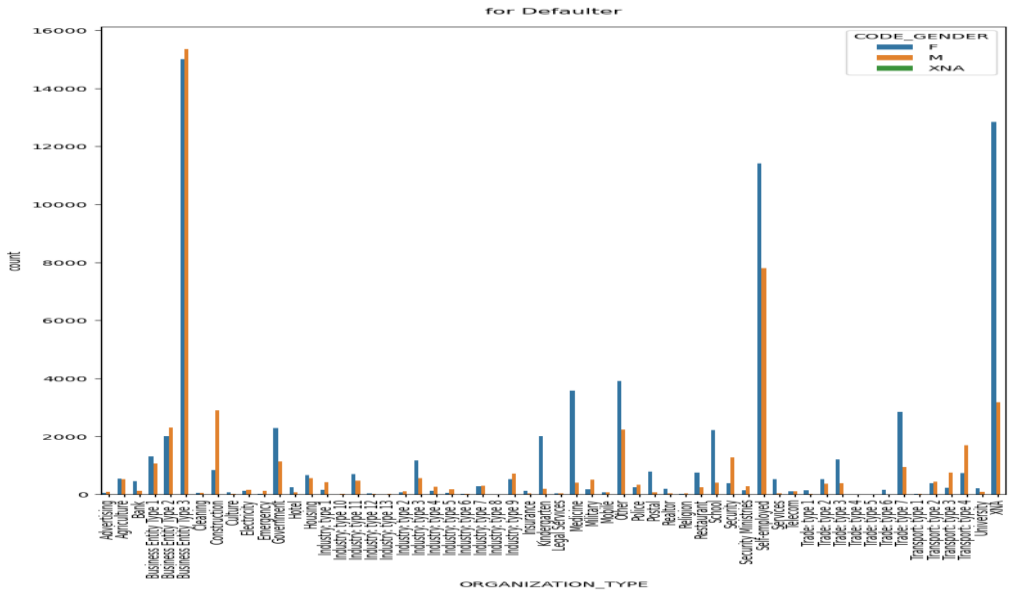
5. YEARS_EMPLOYED_BUCKET & AMT_INCOME_BUCKET
for defaulters YEARS_EMPLOYED_BUCKET 0-2,2-4 with AMT_INCOME_BUCKET 100000-200000 are more as compare to other YEARS_EMPLOYED_BUCKET And AMT_INCOME_BUCKET.

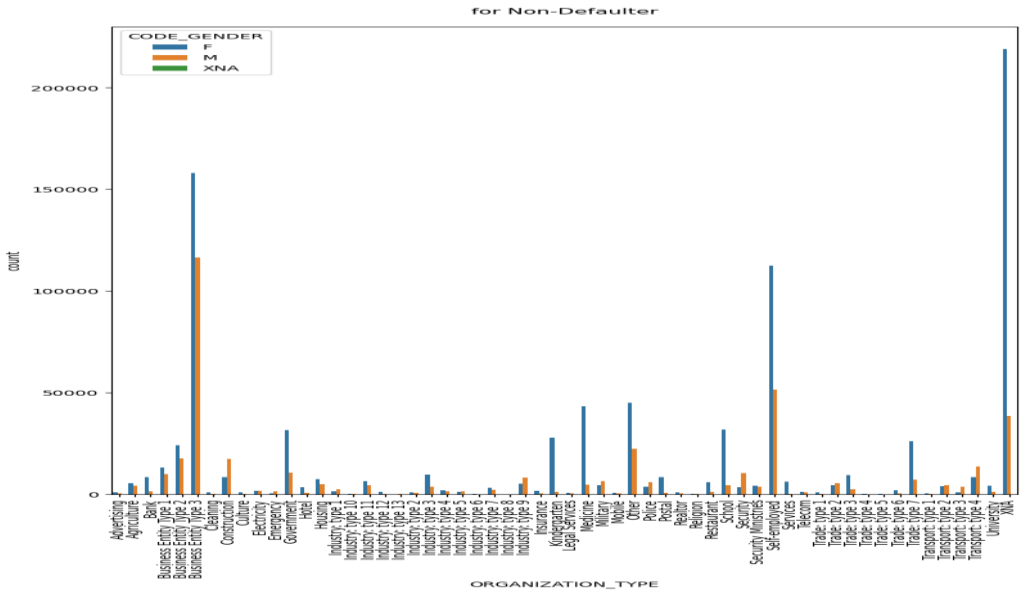





```
# Plotting for ORGANIZATION_TYPE,CODE_GENDER
bivariate_cat_cat(df_loan_1,df_loan_0,'ORGANIZATION_TYPE','CODE_GENDER')
```

6. ORGANIZATION_TYPE & CODE_GENDER
for defaulters ORGANIZATION_TYPE Business Entity type 3 and CODE_GENDER Males are more as compare to other ORGANIZATION_TYPE and CODE_GENDER.

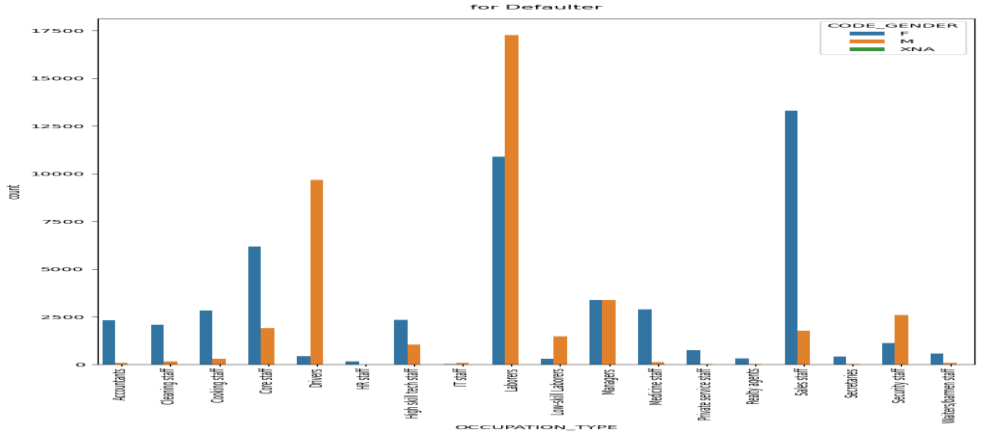


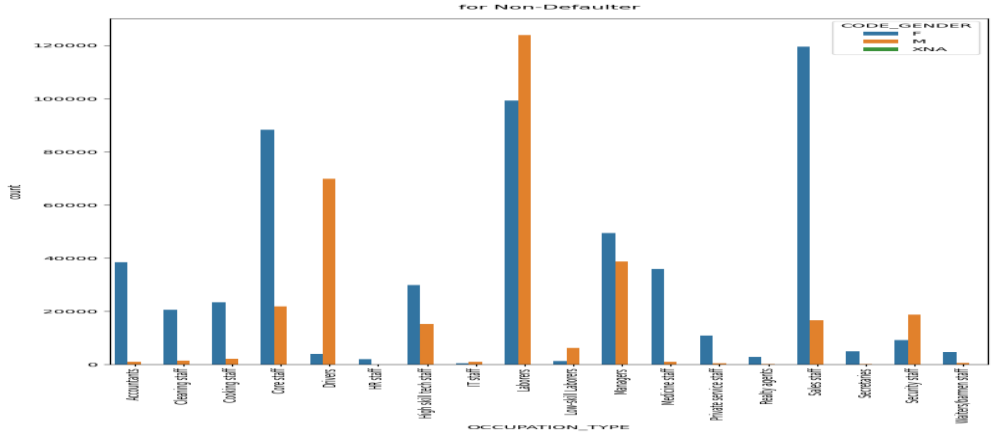


```
# Plotting for OCCUPATION_TYPE,CODE_GENDER
bivariate_cat_cat(df_loan_1,df_loan_0,'OCCUPATION_TYPE','CODE_GENDER')
```

7. OCCUPATION_TYPE & CODE_GENDER

for defaulters OCCUPATION_TYPE laborers and Drivers with CODE_GENDER Males are more as compare to other OCCUPATION_TYPE and CODE_GENDER, but for non-defaulters Males in OCCUPATION_TYPE laborers and Drivers are also high so needs further analysis .

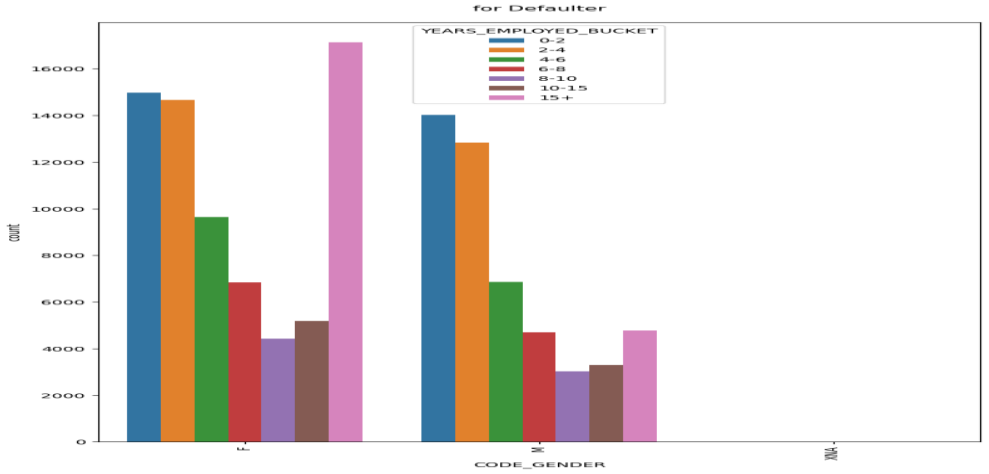


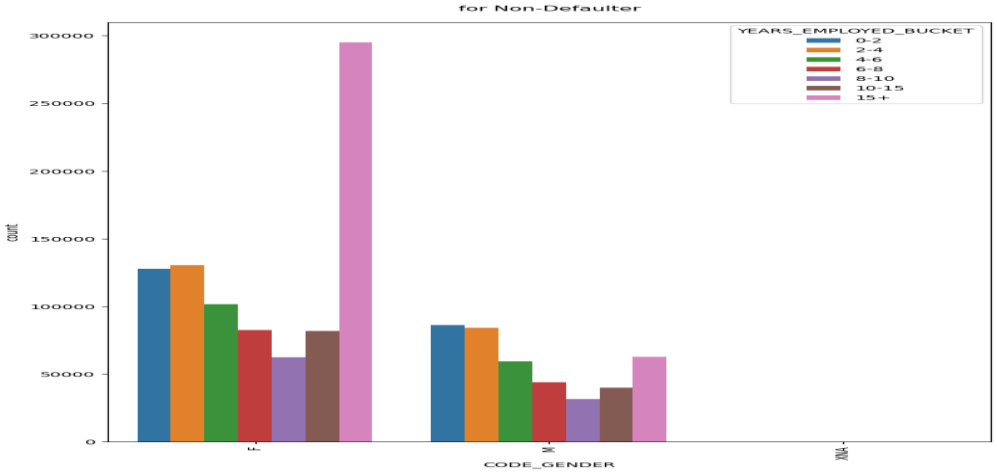



```
# Plotting for CODE_GENDER,YEARS_EMPLOYED_BUCKET
bivariate_cat_cat(df_loan_1,df_loan_0,'CODE_GENDER','YEARS_EMPLOYED_BUCKET')
```

8. CODE_GENDER & YEARS_EMPLOYED_BUCKET

for defaulters YEARS_EMPLOYED_BUCKET 0-2,2-4 both males and females is having high values as compare to other YEARS_EMPLOYED_BUCKET Values.

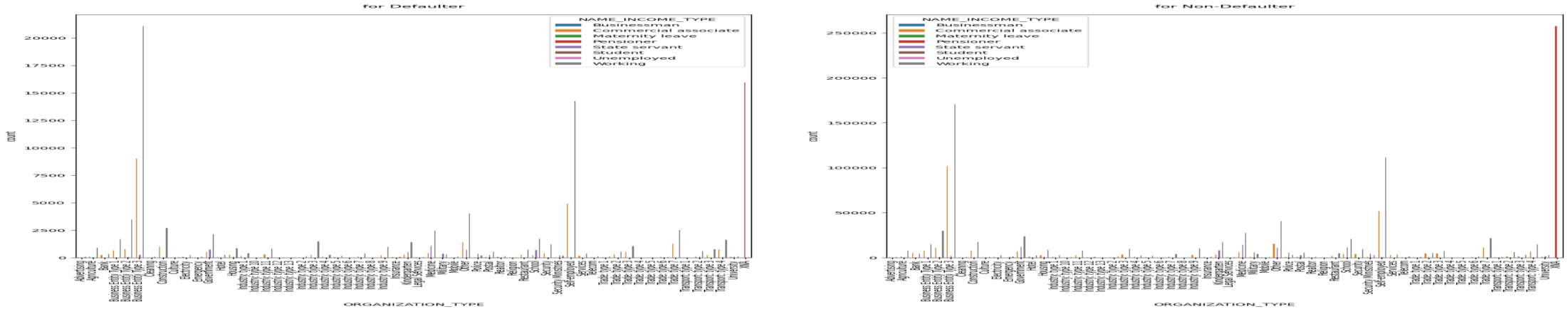




```
# Plotting for ORGANIZATION_TYPE,NAME_INCOME_TYPE
bivariate_cat_cat(df_loan_1,df_loan_0,'ORGANIZATION_TYPE','NAME_INCOME_TYPE')
```

9. ORGANIZATION_TYPE & NAME_INCOME_TYPE

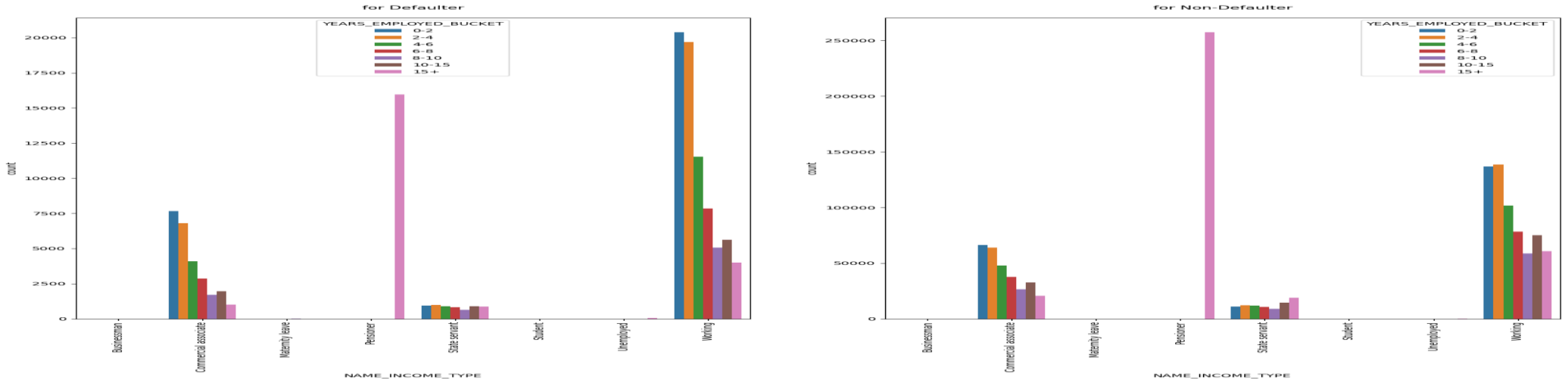
for defaulters ORGANIZATION_TYPE Business Entity type 3 and NAME_INCOME_TYPE Working is having high values as compare to other ORGANIZATION_TYPE .




```
# Plotting for NAME_INCOME_TYPE,YEARS_EMPLOYED_BUCKET
bivariate_cat_cat(df_loan_1,df_loan_0,'NAME_INCOME_TYPE','YEARS_EMPLOYED_BUCKET')
```

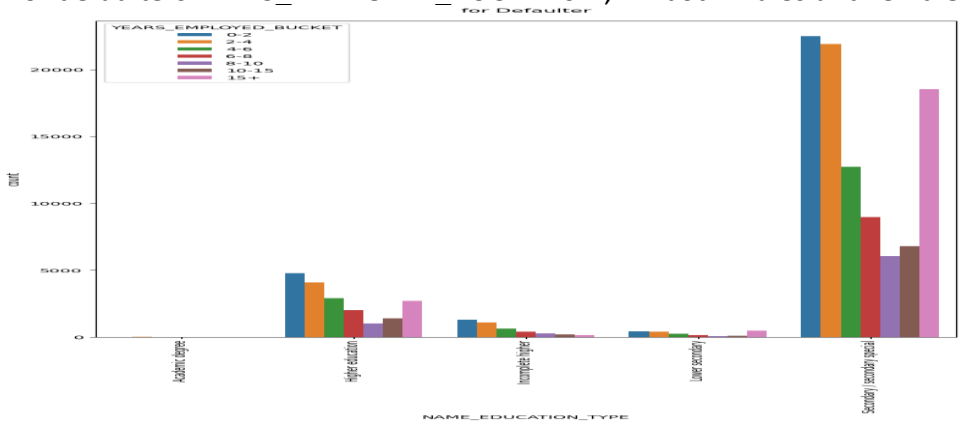
10. NAME_INCOME_TYPE & YEARS_EMPLOYED_BUCKET

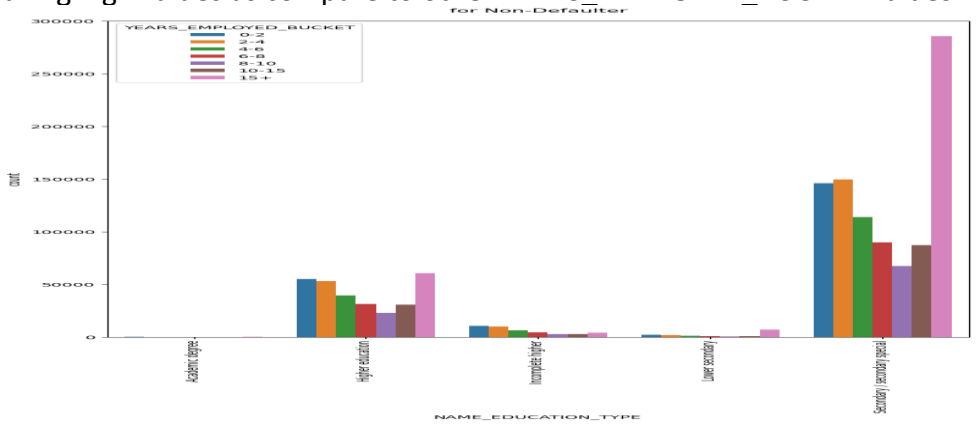
for defaulters NAME_INCOME_TYPE Working and 0-2,2-4 YEARS_EMPLOYED_BUCKET is having high values as compare to other NAME_INCOME_TYPE .



```
# Plotting for NAME_EDUCATION_TYPE,YEARS_EMPLOYED_BUCKET
bivariate_cat_cat(df_loan_1,df_loan_0,'NAME_EDUCATION_TYPE','YEARS_EMPLOYED_BUCKET')
```

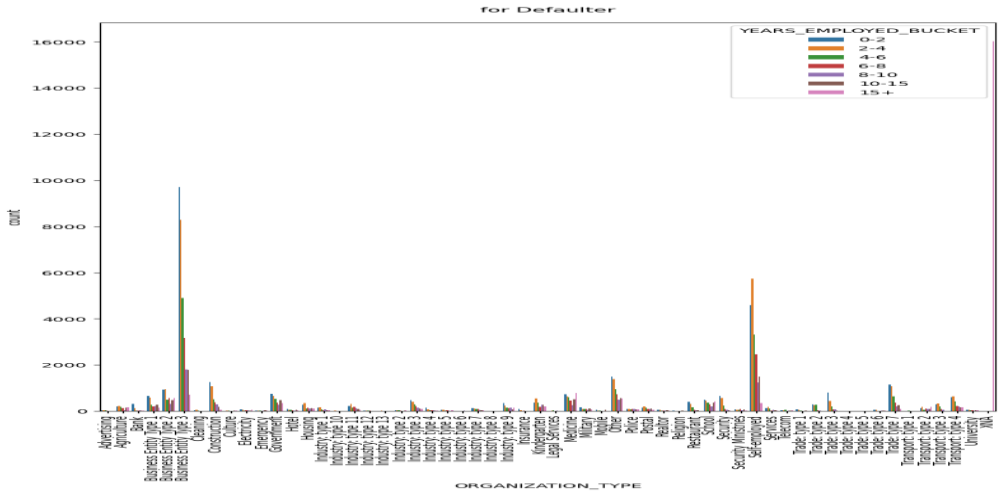
11. NAME_EDUCATION_TYPE & YEARS_EMPLOYED_BUCKET
for defaulters YEARS_EMPLOYED_BUCKET 0-2,2-4 both males and females is having high values as compare to other YEARS_EMPLOYED_BUCKET Values.

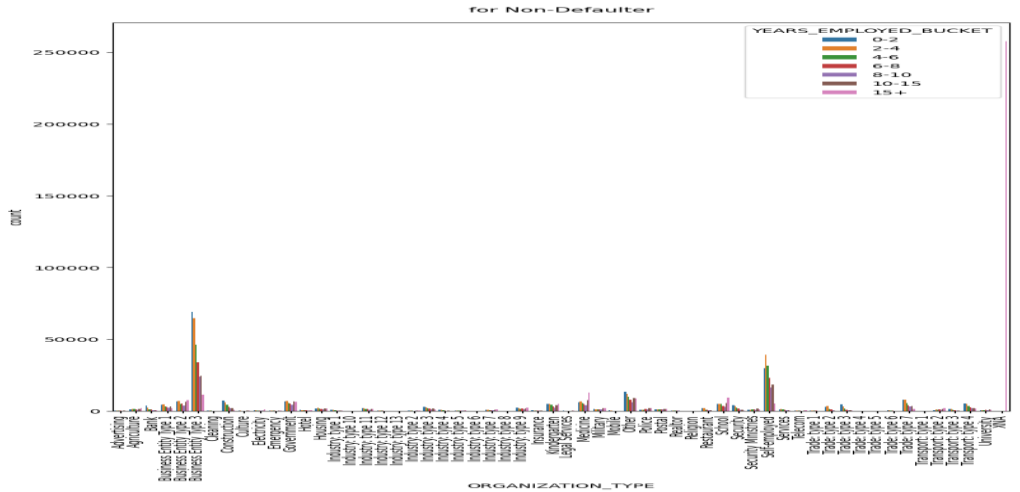





```
# Plotting for ORGANIZATION_TYPE,YEARS_EMPLOYED_BUCKET
bivariate_cat_cat(df_loan_1,df_loan_0,'ORGANIZATION_TYPE','YEARS_EMPLOYED_BUCKET')
```

12. ORGANIZATION_TYPE & YEARS_EMPLOYED_BUCKET
for defaulters ORGANIZATION_TYPE 0-2,2-4 YEARS_EMPLOYED_BUCKET is having high values as compare to other ORGANIZATION_TYPE & YEARS_EMPLOYED_BUCKET.

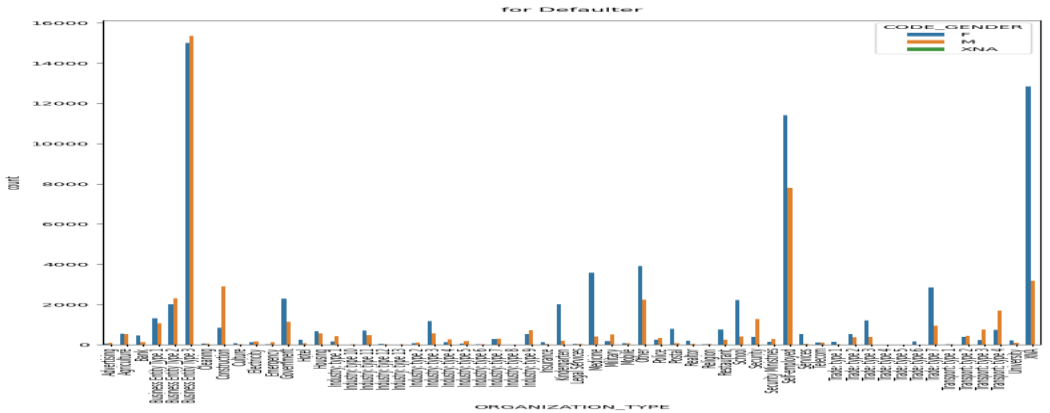


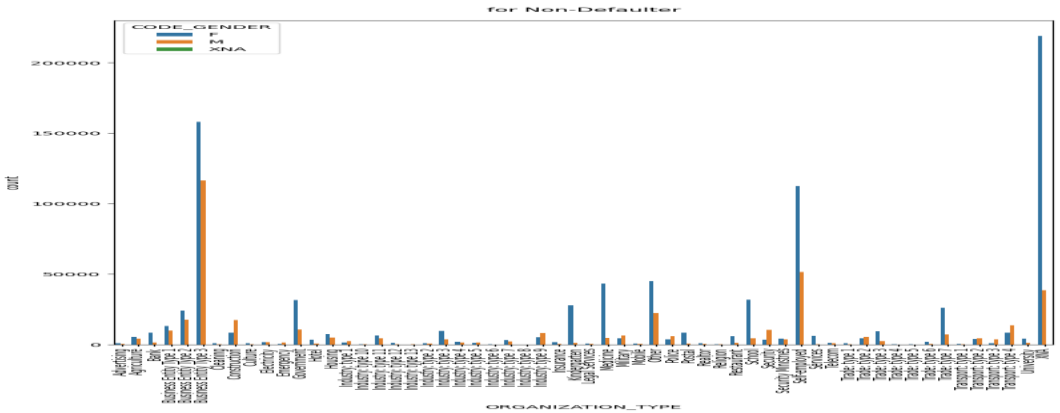


```
# Plotting for ORGANIZATION_TYPE,CODE_GENDER
bivariate_cat_cat(df_loan_1,df_loan_0,'ORGANIZATION_TYPE','CODE_GENDER')
```

13. ORGANIZATION_TYPE & CODE_GENDER

for defaulters ORGANIZATION_TYPE Business Entity 3 Type both males and females is having high values as compare to other ORGANIZATION_TYPE,CODE_GENDER.

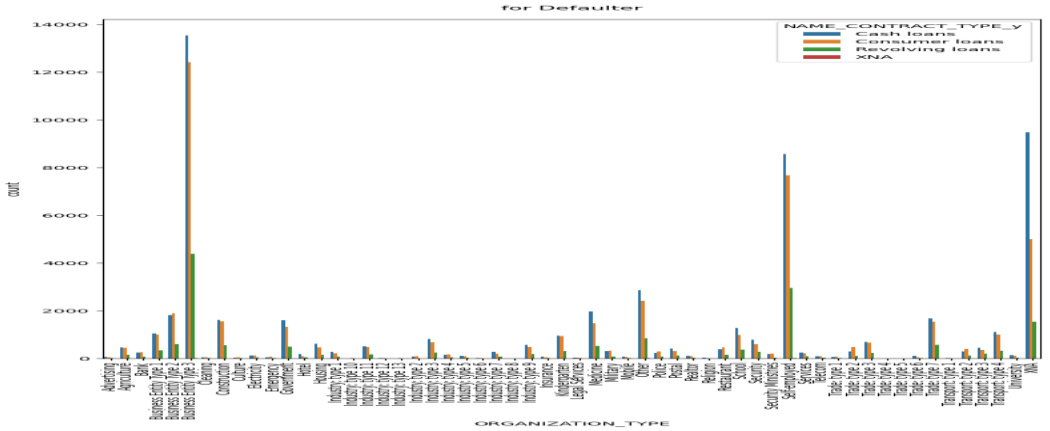


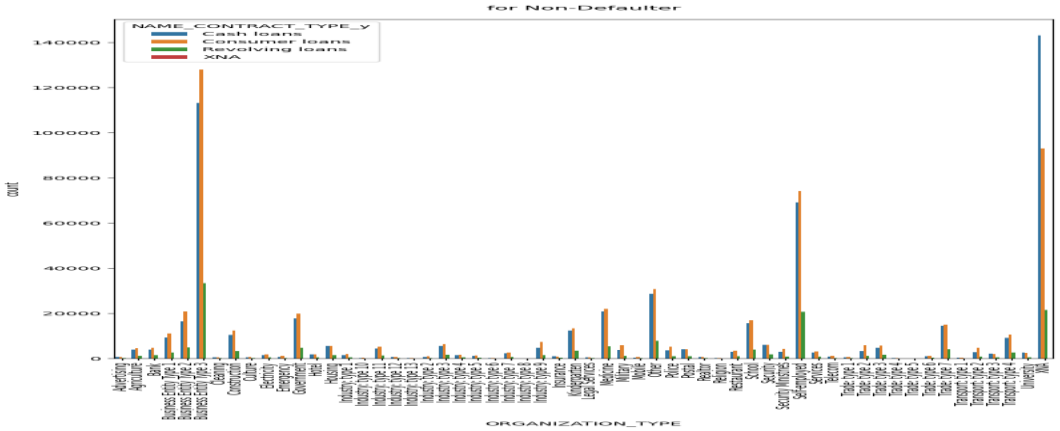



```
# Plotting for ORGANIZATION_TYPE,NAME_CONTRACT_TYPE_y
bivariate_cat_cat(df_loan_1,df_loan_0,'ORGANIZATION_TYPE','NAME_CONTRACT_TYPE_y')
```

14. ORGANIZATION_TYPE & NAME_CONTRACT_TYPE_y

for defaulters ORGANIZATION_TYPE Business Entity Type 3 and cash loans NAME_CONTRACT_TYPE_y is having high values as compare to other ORGANIZATION_TYPE,NAME_CONTRACT_TYPE_y.

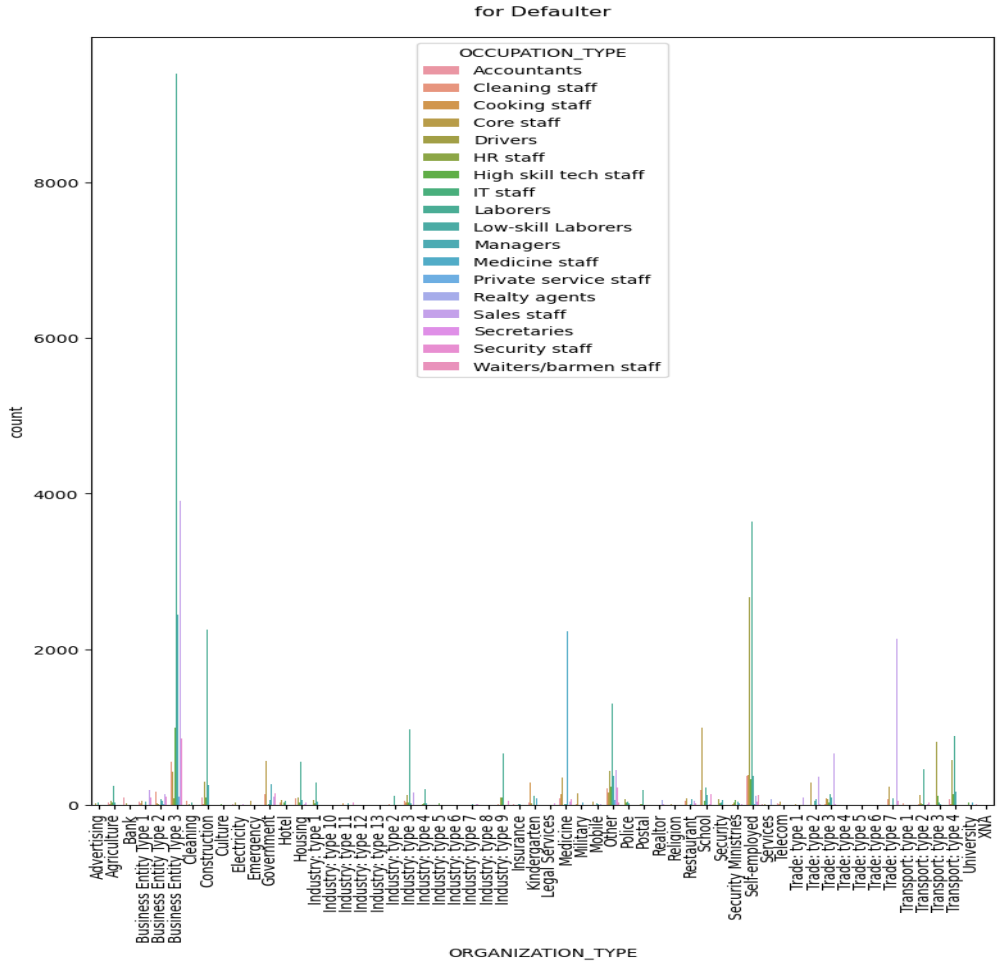


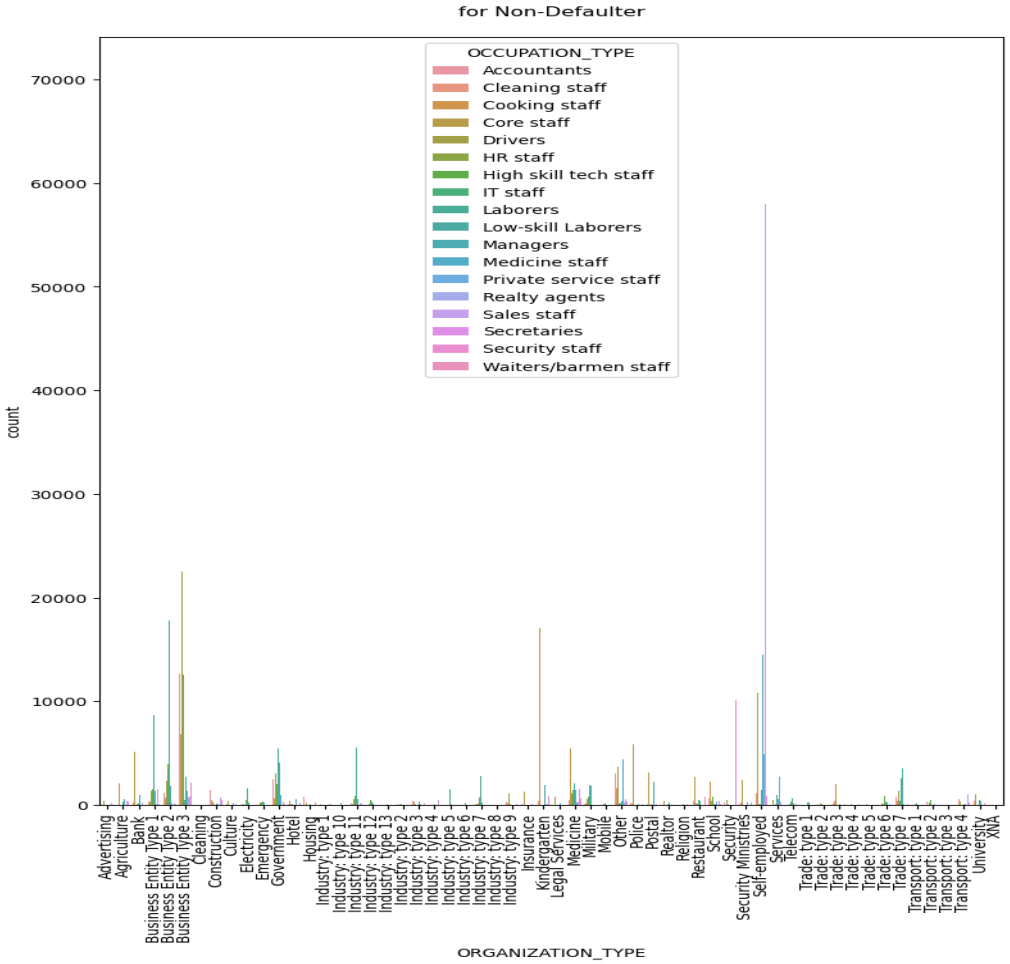


```
# Plotting for ORGANIZATION_TYPE,OCCUPATION_TYPE
bivariate_cat_cat(df_loan_1,df_loan_0,'ORGANIZATION_TYPE','OCCUPATION_TYPE')
```

15. ORGANIZATION_TYPE & OCCUPATION_TYPE

for defaulters ORGANIZATION_TYPE Business Entity Type 3 and labourers OCCUPATION_TYPE is having high values as compare to other ORGANIZATION_TYPE, OCCUPATION_TYPE.





Now we find correlation matrix and plotting heat maps

```
# bisecting the df dataframe based on Target value 0 and 1 for correlation and other analysis
cols_for_corr_loan = ['NAME_CONTRACT_TYPE_x', 'CODE_GENDER','FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_T
                      'AMT_CREDIT_x','AMT_ANNUITY_x', 'AMT_GOODS_PRICE_x','NAME_TYPE_SUITE_x','NAME_INCOME_TYPE',
                      'NAME_EDUCATION_TYPE','NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE','REGION_POPULATION_RELATIVE',
                      'DAYS_REGISTRATION','DAYS_ID_PUBLISH','OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
                      'REGION_RATING_CLIENT_W_CITY', 'REG_REGION_NOT_LIVE_REGION','REG_REGION_NOT_WORK_REGION',
                      'LIVE_REGION_NOT_WORK_REGION','REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY','LIVE_CITY_NOT_WORK_C
                      'ORGANIZATION_TYPE', 'EXT_SOURCE_2','EXT_SOURCE_3', 'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIR
                      'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE','DAYS_LAST_PHONE_CHANGE', 'Age', 'YEARS_EMPLOYE
                      'Age_Group','YEARS_EMPLOYED_BUCKET', 'AMT_INCOME_BUCKET', 'AMT_CREDIT_BUCKET_x','AMT_ANNUITY_BUCKET_x'
                      'AMT_GOODS_PRICE_BUCKET_x','NAME_CONTRACT_TYPE_y', 'AMT_ANNUITY_y', 'AMT_APPLICATION','AMT_CREDIT_y',
                      'AMT_GOODS_PRICE_y', 'NAME_CASH_LOAN_PURPOSE','NAME_CONTRACT_STATUS', 'DAYS_DECISION', 'NAME_PAYMENT_T
                      'CODE_REJECT_REASON', 'NAME_TYPE_SUITE_y', 'NAME_CLIENT_TYPE','NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO',
                      'NAME_PRODUCT_TYPE','CHANNEL_TYPE', 'NAME_SELLER_INDUSTRY','CNT_PAYMENT', 'NAME_YIELD_GROUP',
                      'PRODUCT_COMBINATION','DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION', 'DAYS_LAST_
                      'DAYS_TERMINATION', 'NFLAG_INSURED_ON_APPROVAL','AMT_CREDIT_BUCKET_y', 'AMT_ANNUITY_BUCKET_y',
                      'AMT_GOODS_PRICE_BUCKET_y']


# Defaulters dataframe
df_loan_1 = df_loan.loc[df_loan['TARGET']==1, cols_for_corr_loan]

# Non-Defaulters dataframe
df_loan_0 = df_loan.loc[df_loan['TARGET']==0, cols_for_corr_loan]
```

Correlation for the Non-Defaulters dataframe

```
# Getting  top 10 correlation for the Non-Defaulters dataframe
corr_non_defaulters = df_loan_0.corr(numeric_only=True)
corr_non_defaulters = corr_non_defaulters.where(np.triu(np.ones(corr_non_defaulters.shape),k=1).astype(bool)).unstack().rese
corr_non_defaulters.columns =['VARIABLE1','VARIABLE2','Correlation']
corr_non_defaulters.dropna(subset = ["Correlation"], inplace = True)
corr_non_defaulters["Correlation"]=corr_non_defaulters["Correlation"]
corr_non_defaulters.sort_values(by='Correlation', ascending=False, inplace=True)
corr_non_defaulters.head(10)
```

	VARIABLE1	VARIABLE2	Correlation
598	AMT_GOODS_PRICE_y	AMT_APPLICATION	0.999888
358	OBS_60_CNT_SOCIAL_CIRCLE	OBS_30_CNT_SOCIAL_CIRCLE	0.998578
599	AMT_GOODS_PRICE_y	AMT_CREDIT_y	0.993299
88	AMT_GOODS_PRICE_x	AMT_CREDIT_x	0.986593
569	AMT_CREDIT_y	AMT_APPLICATION	0.975725
809	DAYS_TERMINATION	DAYS_LAST_DUE	0.926621
388	DEF_60_CNT_SOCIAL_CIRCLE	DEF_30_CNT_SOCIAL_CIRCLE	0.863099
597	AMT_GOODS_PRICE_y	AMT_ANNUITY_y	0.821027
568	AMT_CREDIT_y	AMT_ANNUITY_y	0.816541
539	AMT_APPLICATION	AMT_ANNUITY_y	0.809023

Correlation for the Defaulters dataframe

```
# Getting  top 10 correlation for the Defaulters dataframe
corr_defaulters = df_loan_1.corr(numeric_only=True)
corr_defaulters = corr_defaulters.where(np.triu(np.ones(corr_defaulters.shape),k=1).astype(bool)).unstack().reset_index()
corr_defaulters.columns =['VARIABLE1','VARIABLE2','Correlation']
corr_defaulters.dropna(subset = ["Correlation"], inplace = True)
corr_defaulters["Correlation"]=corr_defaulters["Correlation"]
corr_defaulters.sort_values(by='Correlation', ascending=False, inplace=True)
corr_defaulters.head(10)
```

	VARIABLE1	VARIABLE2	Correlation
598	AMT_GOODS_PRICE_y	AMT_APPLICATION	0.999676
358	OBS_60_CNT_SOCIAL_CIRCLE	OBS_30_CNT_SOCIAL_CIRCLE	0.998378
599	AMT_GOODS_PRICE_y	AMT_CREDIT_y	0.992302
88	AMT_GOODS_PRICE_x	AMT_CREDIT_x	0.982912
569	AMT_CREDIT_y	AMT_APPLICATION	0.975377
809	DAYS_TERMINATION	DAYS_LAST_DUE	0.937547
388	DEF_60_CNT_SOCIAL_CIRCLE	DEF_30_CNT_SOCIAL_CIRCLE	0.858281
568	AMT_CREDIT_y	AMT_ANNUITY_y	0.840461
597	AMT_GOODS_PRICE_y	AMT_ANNUITY_y	0.840196
539	AMT_APPLICATION	AMT_ANNUITY_y	0.824962

Creating correlation table for non defaulters with continuous variables

In [195]:
```
# # Creating correlation table for non defaulters with certain imp columns
df_loan_0.corr(numeric_only=True)
```

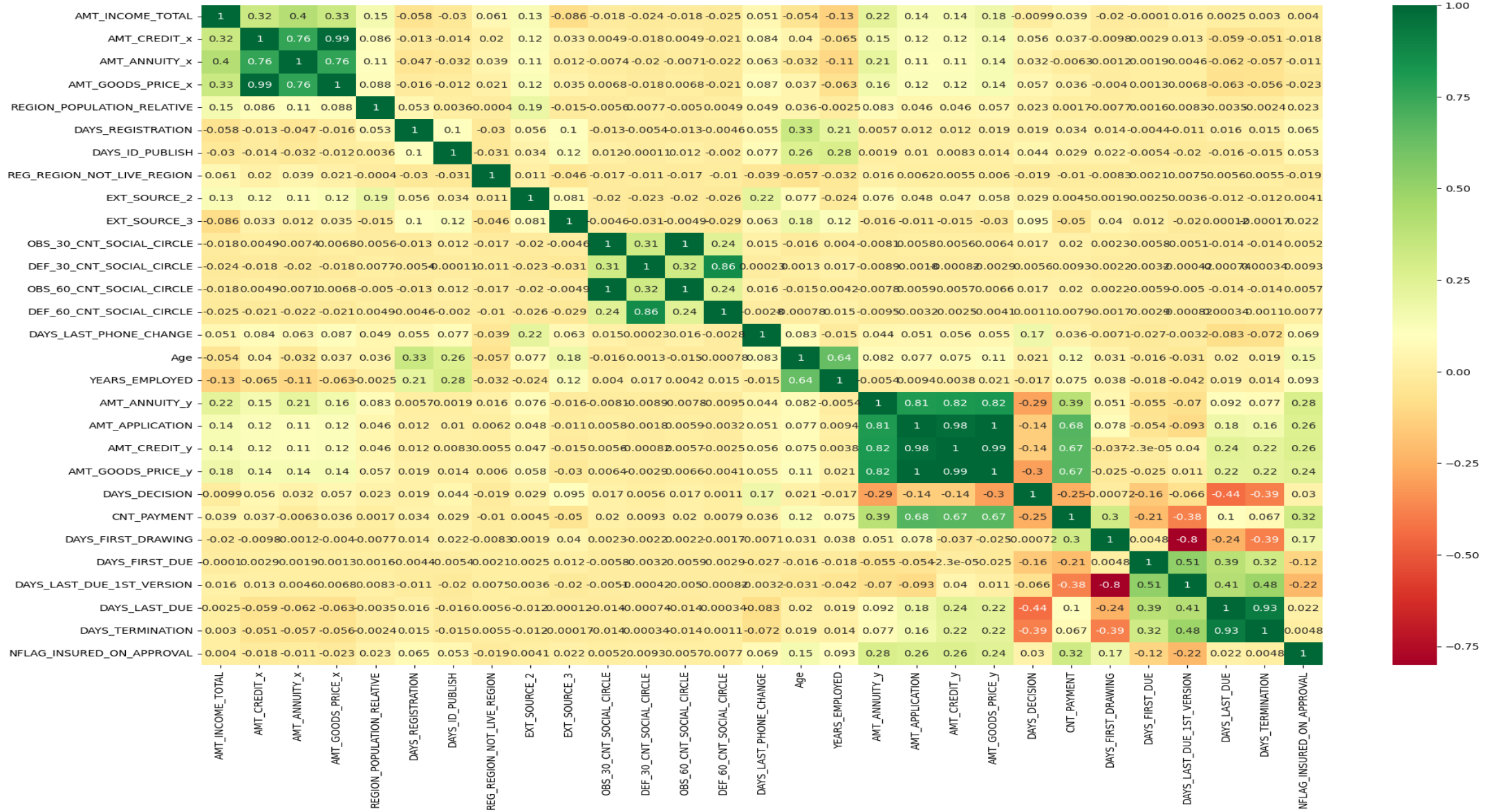
Out[195]:

	AMT_INCOME_TOTAL	AMT_CREDIT_x	AMT_ANNUITY_x	AMT_GOODS_PRICE_x	REGION_POPULATION_RELATIVE	DAYS_REG
AMT_INCOME_TOTAL	1.000000	0.324845	0.402081	0.328136	0.152624	
AMT_CREDIT_x	0.324845	1.000000	0.759440	0.986593	0.086088	
AMT_ANNUITY_x	0.402081	0.759440	1.000000	0.763653	0.105076	
AMT_GOODS_PRICE_x	0.328136	0.986593	0.763653	1.000000	0.088035	
REGION_POPULATION_RELATIVE	0.152624	0.086088	0.105076	0.088035	1.000000	
DAYS_REGISTRATION	-0.057918	-0.013149	-0.046565	-0.015895	0.053157	
DAYS_ID_PUBLISH	-0.030465	-0.013904	-0.032421	-0.011584	0.003646	
REG_REGION_NOT_LIVE_REGION	0.061409	0.019611	0.038829	0.020952	-0.000404	
EXT_SOURCE_2	0.130360	0.115283	0.109517	0.121084	0.187428	
EXT_SOURCE_3	-0.085904	0.032902	0.011963	0.035418	-0.014691	
OBS_30_CNT_SOCIAL_CIRCLE	-0.018478	0.004919	-0.007383	0.006849	-0.005606	
DEF_30_CNT_SOCIAL_CIRCLE	-0.023870	-0.017561	-0.020410	-0.017707	0.007718	
OBS_60_CNT_SOCIAL_CIRCLE	-0.018359	0.004890	-0.007102	0.006751	-0.005033	
DEF_60_CNT_SOCIAL_CIRCLE	-0.024984	-0.020881	-0.022032	-0.020854	0.004942	
DAYS_LAST_PHONE_CHANGE	0.051216	0.084176	0.063124	0.087483	0.049102	
Age	-0.054275	0.039691	-0.031793	0.036983	0.036430	
YEARS_EMPLOYED	-0.132058	-0.065100	-0.110477	-0.063366	-0.002462	
AMT_ANNUITY_y	0.222339	0.153864	0.205261	0.155762	0.083478	
AMT_APPLICATION	0.139347	0.118680	0.113231	0.120322	0.045859	
AMT_CREDIT_y	0.137830	0.116667	0.108790	0.116896	0.046429	
AMT_GOODS_PRICE_y	0.175692	0.138156	0.135783	0.139107	0.057108	
DAYS_DECISION	-0.009916	0.055825	0.032325	0.056699	0.022661	
CNT_PAYMENT						

Heat map for non defaulter

```
# Creating a heat map for non defaulters on basis of correaltion variables
plt.figure(figsize=[22,14])
sns.heatmap(data=df_loan_0.corr(numeric_only=True),annot= True,cmap='RdYlGn')
plt.show()
```

Heat map for non defaulters showing correlation among different variables



Creating correlation table for defaulters with continuous variables

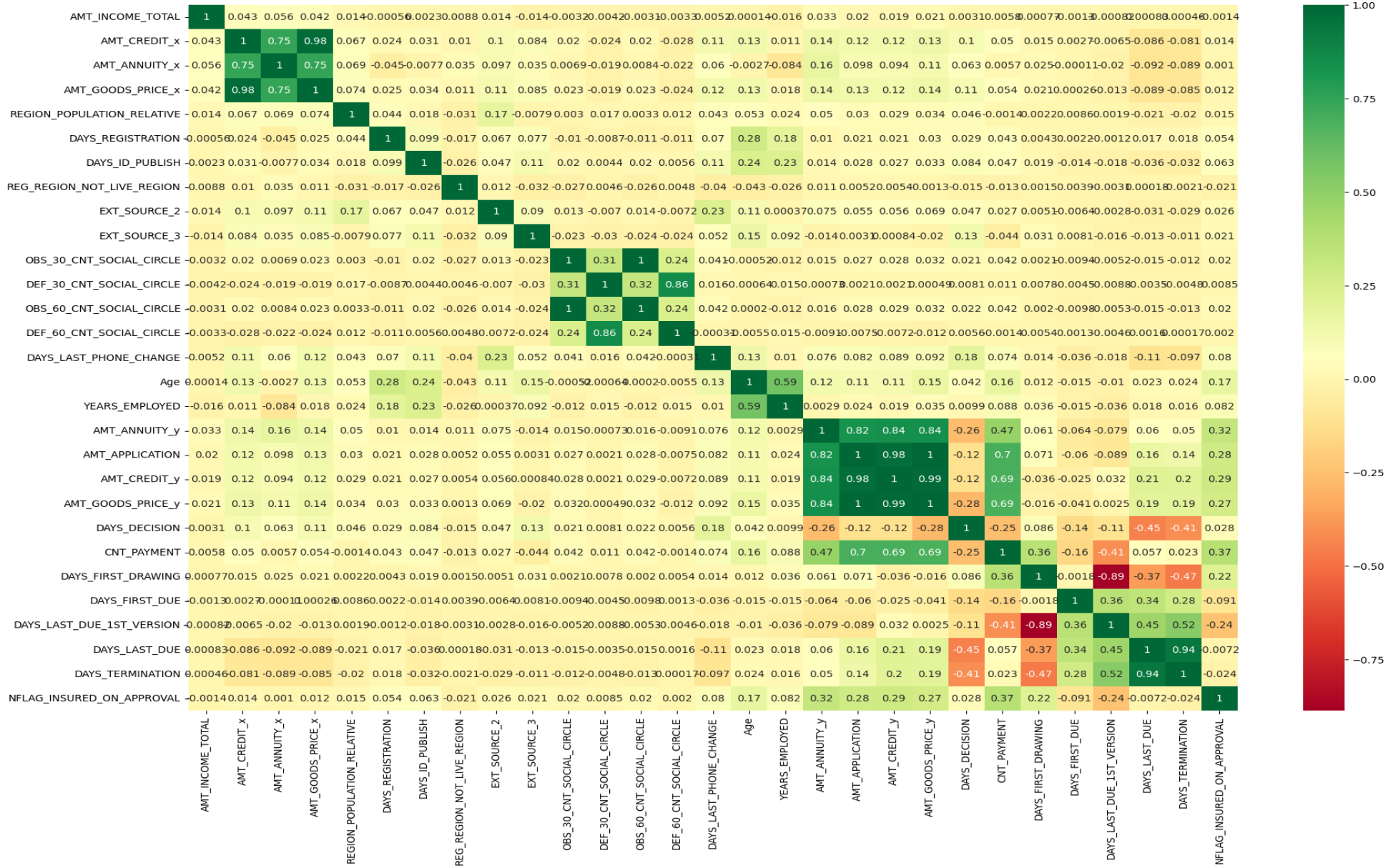
```
# # Creating correlation table for defaulters with certain imp columns
df_loan_1.corr(numeric_only=True)
```

	AMT_INCOME_TOTAL	AMT_CREDIT_x	AMT_ANNUITY_x	AMT_GOODS_PRICE_x	REGION_POPULATION_RELATIVE	DAYS_REG
AMT_INCOME_TOTAL	1.000000	0.043130	0.055551	0.042457	0.013728	
AMT_CREDIT_x	0.043130	1.000000	0.745679	0.982912	0.066670	
AMT_ANNUITY_x	0.055551	0.745679	1.000000	0.745604	0.069390	
AMT_GOODS_PRICE_x	0.042457	0.982912	0.745604	1.000000	0.073608	
REGION_POPULATION_RELATIVE	0.013728	0.066670	0.069390	0.073608	1.000000	
DAYS_REGISTRATION	-0.000555	0.023541	-0.045499	0.024934	0.044030	
DAYS_ID_PUBLISH	0.002292	0.031287	-0.007674	0.034317	0.018088	
REG_REGION_NOT_LIVE_REGION	0.008832	0.010052	0.035162	0.011281	-0.031123	
EXT_SOURCE_2	0.013624	0.102965	0.096975	0.111788	0.170794	
EXT_SOURCE_3	-0.014286	0.083694	0.035382	0.085316	-0.007878	
OBS_30_CNT_SOCIAL_CIRCLE	-0.003187	0.019918	0.006909	0.022536	0.003023	
DEF_30_CNT_SOCIAL_CIRCLE	-0.004220	-0.023765	-0.019382	-0.018981	0.016511	
OBS_60_CNT_SOCIAL_CIRCLE	-0.003066	0.020453	0.008446	0.023150	0.003285	
DEF_60_CNT_SOCIAL_CIRCLE	-0.003285	-0.028387	-0.021699	-0.024320	0.012277	
DAYS_LAST_PHONE_CHANGE	0.005156	0.110605	0.059664	0.116375	0.042835	
Age	0.000144	0.125261	-0.002746	0.127664	0.053327	
YEARS_EMPLOYED	-0.015820	0.010687	-0.083693	0.018180	0.024116	
AMT_ANNUITY_y	0.033361	0.137121	0.164643	0.140492	0.050184	
AMT_APPLICATION	0.019538	0.119639	0.097688	0.125007	0.029503	
AMT_CREDIT_y	0.019166	0.118091	0.094217	0.121914	0.029167	
AMT_GOODS_PRICE_y	0.021025	0.133726	0.114429	0.138576	0.033661	
DAYS_DECISION	0.003064	0.103279	0.063261	0.106777	0.046432	
CNT_PAYMENT	0.005778	0.049580	0.005664	0.053896	-0.001428	
DAYS_FIRST_DRAWING	0.000774	0.014865	0.024823	0.020829	0.002184	

Heat map for defaulter

```
# Creating a heat map for defaulters on basis of correaltion variables
plt.figure(figsize=[22,14])
ax=sns.heatmap(data=df_loan_1.corr(numeric_only=True),annot= True,cmap='RdYlGn')
plt.show()
```

Heat map for defaulters showing correlation among different variables



Insights after comparing correlation matrix and heat maps for defaulters and non defaulters :-

1. Loan annuity correlation with AMT_INCOME_TOTAL(0.055),AMT_CREDIT(0.74) has reduced in defaulters when compared to non defaulters, AMT_INCOME_TOTAL(0.402),AMT_CREDIT(0.759).
2. We can also see that non defaulters have high correlation in number of days employed and age (0.64) when compared defaulters(0.59).
3. There is a severe drop in the correlation between total income of the client and the AMT_CREDIT(0.324), AMT_ANNUITY(0.402), AMT_GOODS_PRICE(0.328),REGION_POPULATION_RELATIVE(0.152) among non defaulters whereas it is AMT_CREDIT (0.043), AMT_ANNUITY (0.055),AMT_GOODS_PRICE (0.0424),REGION_POPULATION_RELATIVE(0.0137) among defaulters.
4. There is a slight increase in defaulted to observed count in social circle among defaulters when compared to non defaulters.
5. There is a severe drop in the correlation between CNT_PAYMENT & AMT_INCOME_TOTAL for defaulters (0.005) while for non defaulters it is (0.039).

CONCLUSION :-

For Defaulters we have following key observations :-

1.DAYS_REGISTRATION - for defaulters density is high for low value of days registration value which clearly shows most defaulters change registation recently.
2.DAYS_ID_PUBLISH - for defaulters density is high for low value of days id publish value which clearly shows most defaulters change ID recently.
3. DAYS_LAST_PHONE_CHANGE- More people from the Target 1 have changed their phone earlier than Target 1. Indicating intention issues in repaying loan.
4. AGE_IN_YEARS – Younger in age 0-30,30-40 are defaulting more.
5. CODE_GENDER - Ratio of F to M in Target 0 is 2.157 and F to M in Target 0 is 1.47 indicating that
MEN are defaulting more than Women
6. YEARS_EMPLOYED_BUCKET - for Defaulters 46 % comes from 0-2,2-4 category bank need to check while providing loans to less years clients.
7. NAME_CONTRACT_TYPE_y - for Defaulters Cash loans = 46.74 % so clearly cash loans need to be taken care of
8. for defaulters **YEARS_EMPLOYED_BUCKET 0-2,2-4 and Age_Group of 0-30,30-40** are more defaulters as compare to other YEARS_EMPLOYED_BUCKET and age group.
9. for defaulters **YEARS_EMPLOYED_BUCKET 0-2,2-4 with AMT_INCOME_BUCKET 100000-200000** are more as compare to other YEARS_EMPLOYED_BUCKET and AMT_INCOME_BUCKET.
10.for defaulters **ORGANIZATION_TYPE Business Entity type 3 , 0-2,2-4 YEARS_EMPLOYED_BUCKET** is having high values as compare to other .
11.for defaulters **ORGANIZATION_TYPE Transport-type 1 is having high AMT_INCOME_TOTAL** as compare to other .
12. for defaulters **NAME_CASH_LOAN_PURPOSE** Refuse to name the goal is having high **AMT_INCOME_TOTAL** as compare to other.