Himanshu Jain
himjain567@gmail.com

Bank Loan Case Study

Description:

Bank earns their profits by providing loans to the customer. But when ever a person is not able to repay the loan bank suffer from the loss. To avoid this kind of situation bank try to analyse the past records of the customer which will help bank to undersetand the chances of a person to repay the loan.

Also some of the people who be the defaulter apply for the loan again which will cost the bank. Analysing the defaulter in bank is a major work of risk analyst. In this case study I will try to find out the relation between the defaulter and the different features of the data set, so that bank can earn profit.

The loss to the bank can be in two ways:

1. If a person is not likely to repay the loan then acception his loan request will lead to the loss of the bank.
2. If the person is likely to repay the loan but now approving the loan will lead to the loss of the bank.

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

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

Approach:

First understanding both the dataset and what is the key connection between them. Check for the null values and plotting some scatter plots and bar plots to understanding the

distribution. For each of the result the chart is the best approach to present it. So, I created chart for each of the result.

Tech used:

For the project I have used Google Colab. Google Colab in its free version provides 12 GB of working ram and 90 GB of storage to load and work with the dataset.

Insights:

This project helps me to understand the power of different platforms. I understood that for the large data set where the total number of data point higher than 10 Lac it is better to use Python. Also, with this project I learned how to deal with a lot of features in a data set.

I have also find worked out on the following problems:

- Present the overall approach of the analysis. Mention the problem statement and the analysis approach briefly
- Identify the missing data and use appropriate method to deal with it. (Remove columns/or replace it with an appropriate value) Hint: Note that in EDA, since it is not necessary to replace the missing value, but if you have to replace the missing value, what should be the approach. Clearly mention the approach.
- Identify if there are outliers in the dataset. Also, mention why do you think it is an outlier. Again, remember that for this exercise, it is not necessary to remove any data points.
- Identify if there is data imbalance in the data. Find the ratio of data imbalance. Hint: Since there are a lot of columns, you can run your analysis in loops for the appropriate columns and find the insights.
- Explain the results of univariate, segmented univariate, bivariate analysis, etc. in business terms.
- Find the top 10 correlation for the Client with payment difficulties and all other cases (Target variable). Note that you have to find the top correlation by segmenting the data frame w.r.t to the target variable and then find the top correlation for each of the segmented data and find if any insight is there. Say, there are 5+1(target) variables in a dataset: Var1, Var2, Var3, Var4, Var5, Target. And if you have to find top 3 correlation, it can be: Var1

& Var2, Var2 & Var3, Var1 & Var3. Target variable will not feature in this correlation as it is a categorical variable and not a continuous variable which is increasing or decreasing.

- Include visualizations and summarize the most important results in the presentation. You are free to choose the graphs which explain the numerical/categorical variables. Insights should explain why the variable is important for differentiating the clients with payment difficulties with all other cases.

Results:

1) Importing libraries:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme(style="whitegrid")
import numpy as np
```

2) Creating function to find the null values in ascending order

```
def count_null_values(df,per=0):
 length = len(df)
 df1 = df.isnull().sum()*100/length
 df2 = df1[df1>=per]
 return df2.sort_values(ascending=False)
```

3) Reading the First dataset (application.csv):

```
# application_data.csv
df1 = pd.read_csv('/content/drive/MyDrive/Trainity/Loan Case Study/application_data.csv')
df1.head()
```

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

5 rows × 122 columns

4) Find the description of the dataset:

```
df1.describe()
```

Himanshu Jain
himjain567@gmail.com

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_POPULATION_RELATIVE
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	307499.000000	3.072330e+05	307511.000000
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	27108.573909	5.383962e+05	0.020868
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	14493.737315	3.694465e+05	0.013831
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1615.500000	4.050000e+04	0.000290
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16524.000000	2.385000e+05	0.010006
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	24903.000000	4.500000e+05	0.018850
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.000000	6.795000e+05	0.028663
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	258025.500000	4.050000e+06	0.072508

8 rows × 106 columns

5) Dealing with the null values:

```
# Check for the null values greater than 50% in each column in df1
null50 = count_null_values(df1,50)
null50
```

```
['OWN_CAR_AGE',
 'EXT_SOURCE_1',
 'APARTMENTS_AVG',
 'BASEMENTAREA_AVG',
 'YEARS_BUILD_AVG',
 'COMMONAREA_AVG',
 'ELEVATORS_AVG',
 'ENTRANCES_AVG',
 'FLOORSMIN_AVG',
 'LANDAREA_AVG',
 'LIVINGAPARTMENTS_AVG',
 'LIVINGAREA_AVG',
 'NONLIVINGAPARTMENTS_AVG',
 'NONLIVINGAREA_AVG',
 'APARTMENTS_MODE',
 'BASEMENTAREA_MODE',
 'YEARS_BUILD_MODE',
 'COMMONAREA_MODE',
 'ELEVATORS_MODE',
 'ENTRANCES_MODE',
 'FLOORSMIN_MODE',
 'LANDAREA_MODE',
 'LIVINGAPARTMENTS_MODE',
 'LIVINGAREA_MODE',
 'NONLIVINGAPARTMENTS_MODE',
 'NONLIVINGAREA_MODE',
 'APARTMENTS_MEDI',
 'BASEMENTAREA_MEDI',
 'YEARS_BUILD_MEDI',
 'COMMONAREA_MEDI',
 'ELEVATORS_MEDI',
 'ENTRANCES_MEDI',
 'FLOORSMIN_MEDI',
 'LANDAREA_MEDI',
 'LIVINGAPARTMENTS_MEDI',
 'LIVINGAREA_MEDI',
 'NONLIVINGAPARTMENTS_MEDI',
 'NONLIVINGAREA_MEDI',
```

```
'FONDKAPREMONT_MODE',
'HOUSETYPE_MODE',
'WALLSMATERIAL_MODE']
```

Inference:

The null columns contain the values most related to the apartment or house, and one column contain data for car age, and one normalized score for external data source. So that can be removed because they cannot be useful for the target.

drop values with more than 50% null values

```
df1.drop(null50.index,axis=1,inplace=True)
df1.shape
```

(307511, 81)

Check for the null values greater than 15% in each column in df1

```
print('Percentage (%) null values in each column')
null15 = count_null_values(df1,15)
```

```
Percentage (%) null values in each column
FLOORSMAX_AVG                49.760822
FLOORSMAX_MODE               49.760822
FLOORSMAX_MEDI               49.760822
YEARS_BEGINEXPLUATATION_AVG    48.781019
YEARS_BEGINEXPLUATATION_MODE   48.781019
YEARS_BEGINEXPLUATATION_MEDI   48.781019
TOTALAREA_MODE               48.268517
EMERGENCYSTATE_MODE          47.398304
OCCUPATION_TYPE              31.345545
EXT_SOURCE_3                 19.825307
DAYS_EMPLOYED                18.007161
dtype: float64
```

Inference:

In this the except occupation_type and ext_souorce_3 looks familiar to target null values and those are related to the building. So that can be removed.

Remove those columns and drop remaining columns

```
columns_with_null = dict(null15)
del columns_with_null['EXT_SOURCE_3']
del columns_with_null['OCCUPATION_TYPE']
df1.shape
```

(307511, 72)

Himanshu Jain
himjain567@gmail.com

6) Feature selection and remove unnecessary columns

Treating the columns with the null values

```
null_count = count_null_values(df1)
null_count
```

OCCUPATION_TYPE 31.345545

EXT_SOURCE_3 19.825307

AMT_REQ_CREDIT_BUREAU_DAY 13.501631

AMT_REQ_CREDIT_BUREAU_HOUR 13.501631

AMT_REQ_CREDIT_BUREAU_YEAR 13.501631

...

REG_REGION_NOT_WORK_REGION 0.000000

LIVE_REGION_NOT_WORK_REGION 0.000000

REG_CITY_NOT_LIVE_CITY 0.000000

TARGET 0.000000

REG_CITY_NOT_WORK_CITY 0.000000

Length: 72, dtype: float64

check correlation for the columns EXT_SOURCE_3 and EXT_SOURCE_2 to the target column

```
sns.heatmap(df1[['EXT_SOURCE_3','EXT_SOURCE_2','TARGET']].corr(),annot=True)
```

Himanshu Jain
himjain567@gmail.com

Inference:

As it is clearly seen that the both columns do not show a good correlation with the Target. So, it can be removed.

```
df1.drop(['EXT_SOURCE_3','EXT_SOURCE_2'],axis=1,inplace=True)
df1.shape
```

(307511, 70)

 # Now there are some columns named *flag* which contain some true false values

```
flags = []
for i in df1.columns:
  if 'FLAG' in i:
    flags.append(i)
flags
```

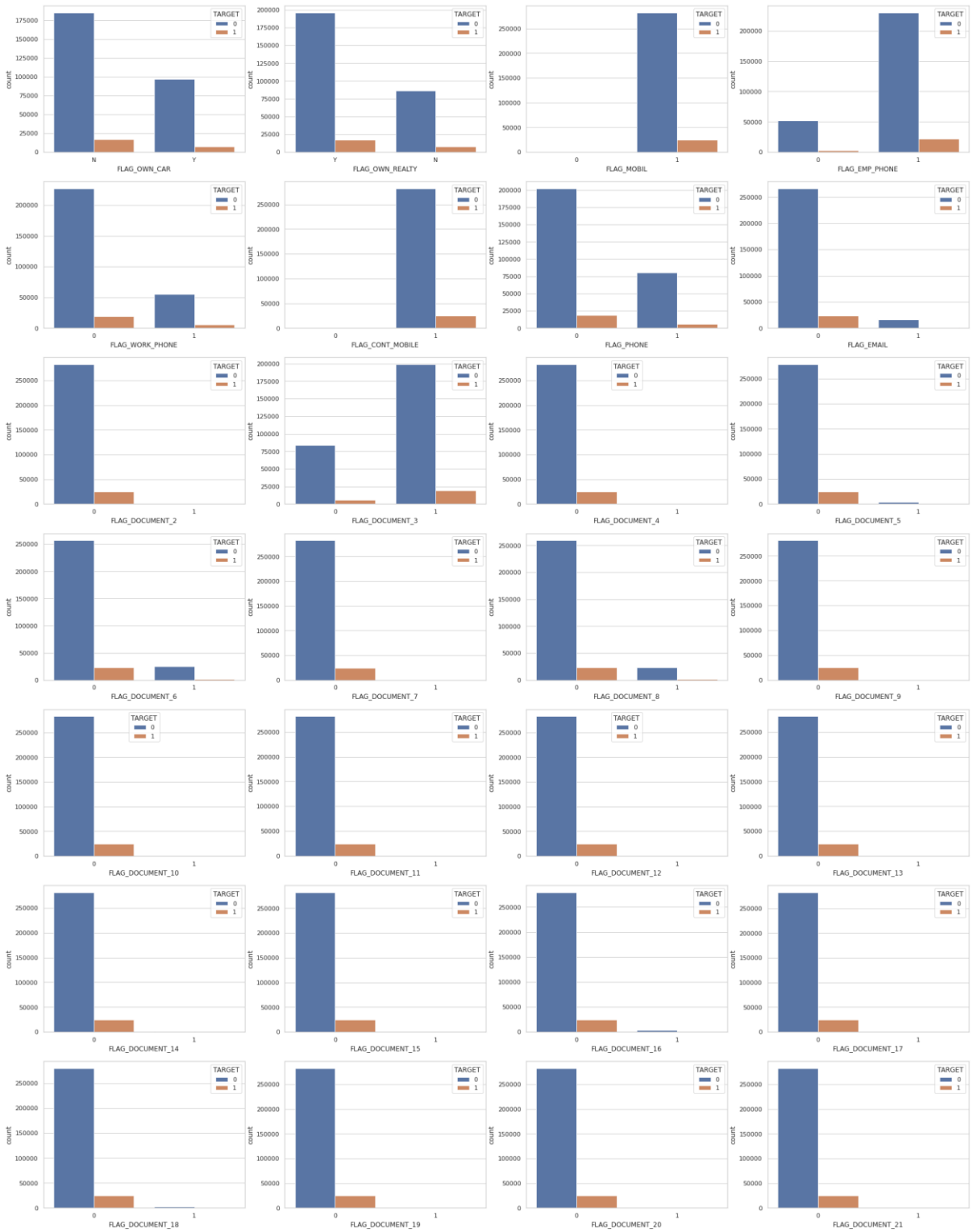
['FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3', 'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9', 'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12', 'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15', 'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21']

 # Plot each flag column vs target column

 # Plot will be of 28 plots with hue = Y/N

```
plt.figure(figsize=(30,40))
for i in range(len(flag_df.columns)-1):
  plt.subplot(7,4,i+1)
  sns.countplot(x=flag_df[flag_df.columns[i]],hue=flag_df[flag_df.columns[-1]])
```

Himanshu Jain
himjain567@gmail.com



Inference:

All the columns which have sufficient values in both 0 and 1 will be kept, except removed columns to kept FLAG_OWN_CAR, FLAG_OWN_REALTY, FLAG_PHONE, FLAG_WORK_PHONE.

 # Drop unnecessary flag columns:

Himanshu Jain
himjain567@gmail.com

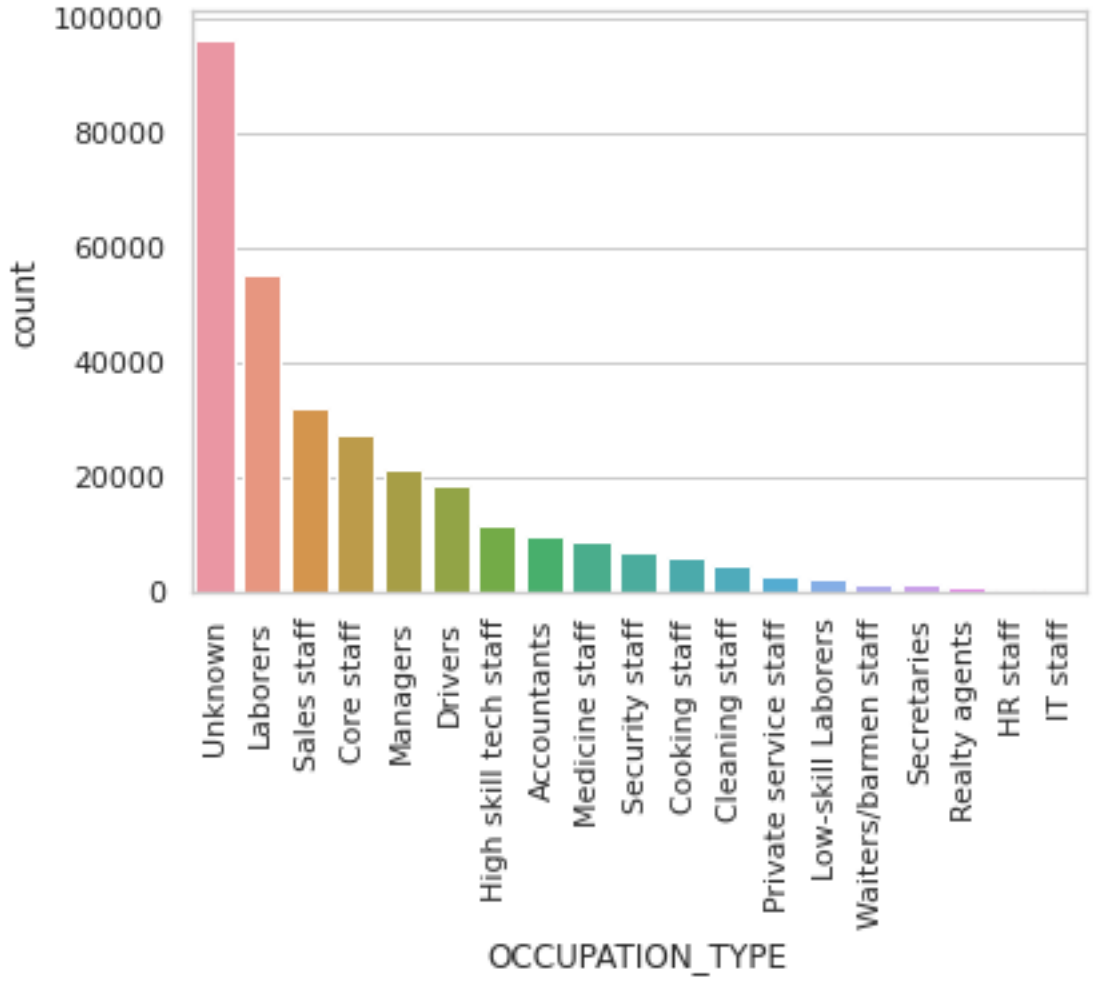
```
remove_list = ['FLAG_OWN_REALTY','FLAG_OWN_CAR','FLAG_PHONE','FLAG_WORK_PHONE'
]
flags = list(set(flags) - set(remove_list))
# finally remove those columns
df1.drop(flags,axis=1,inplace=True)
# Dataset after removing unnecessary flag values
df1.shape
```

(307511, 46)

7) Filling null values

Fill the null values in OCCUPATION_TYPE with the 'Unknown'

```
df1['OCCUPATION_TYPE'].fillna('Unknown',inplace=True)
sns.countplot(x =df1['OCCUPATION_TYPE'],order = df1['OCCUPATION_TYPE'].value_counts().index
)
plt.xticks(rotation=90)
```



Filling values of the column name starts with amt_req

```
amt_req = []
```

Himanshu Jain
himjain567@gmail.com

```
for i in df1.columns:
  if "AMT_REQ" in i:
    amt_req.append(i)
amt_req
```

['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']

Inference:

The columns represent the Number of enquiries to Credit Bureau about the client before different time periods. these values should be integer.

Mean

```
df1[amt_req].mean()
```

AMT_REQ_CREDIT_BUREAU_HOUR 0.006402
AMT_REQ_CREDIT_BUREAU_DAY 0.007000
AMT_REQ_CREDIT_BUREAU_WEEK 0.034362
AMT_REQ_CREDIT_BUREAU_MON 0.267395
AMT_REQ_CREDIT_BUREAU_QRT 0.265474
AMT_REQ_CREDIT_BUREAU_YEAR 1.899974
dtype: float64

Mode

```
df1[amt_req].mode()
```

	AMT_REQ_CREDIT_BUREAU_HOUR	AMT_REQ_CREDIT_BUREAU_DAY	AMT_REQ_CREDIT_BUREAU_WEEK	AMT_REQ_CREDIT_BUREAU_MON	AMT_F
0	0.0	0.0	0.0	0.0	

Median

```
df1[amt_req].median()
```

AMT_REQ_CREDIT_BUREAU_HOUR 0.0
AMT_REQ_CREDIT_BUREAU_DAY 0.0
AMT_REQ_CREDIT_BUREAU_WEEK 0.0
AMT_REQ_CREDIT_BUREAU_MON 0.0
AMT_REQ_CREDIT_BUREAU_QRT 0.0
AMT_REQ_CREDIT_BUREAU_YEAR 1.0
dtype: float64

Himanshu Jain
himjain567@gmail.com

Inference:

For mean the values are real numbers so it cannot bs used, most values in all the columns are 0 but for the column AMT_REQ_CREDIT_BUREAU_YEAR occurrence of all the values is quite comparable so mode will not a good approach for all the columns, but for median it is giving feasible values. So, I will replace all the values by median.

 # Filling the null values by Median

```
try1 = df1[amt_req].fillna(df1[amt_req].median())
```

 # For the column NAME_TYPE_SUITE I will replace the null values by Unknown

```
# replace null by Unknown
df1['NAME_TYPE_SUITE'].fillna('Unknown',inplace=True)
```

Fill the null values for the column social circle

```
social_circle = []
for i in df1.columns:
  if 'SOCIAL_CIRCLE' in i:
    social_circle.append(i)
social_circle
```

['OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',

'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE']

Mean

```
df1[social_circle].mean()
```

OBS_30_CNT_SOCIAL_CIRCLE 1.422245

DEF_30_CNT_SOCIAL_CIRCLE 0.143421

OBS_60_CNT_SOCIAL_CIRCLE 1.405292

DEF_60_CNT_SOCIAL_CIRCLE 0.100049

dtype: float64

Mode

```
df1[social_circle].mode()
```

	OBS_30_CNT_SOCIAL_CIRCLE	DEF_30_CNT_SOCIAL_CIRCLE	OBS_60_CNT_SOCIAL_CIRCLE	DEF_60_CNT_SOCIAL_CIRCLE
0	0.0	0.0	0.0	0.0

```
df1[social_circle].median()
```

OBS_30_CNT_SOCIAL_CIRCLE 0.0

DEF_30_CNT_SOCIAL_CIRCLE 0.0

OBS_60_CNT_SOCIAL_CIRCLE 0.0

Himanshu Jain
himjain567@gmail.com

DEF_60_CNT_SOCIAL_CIRCLE 0.0

dtype: float64

Inference:

This column represents how many observations of client's social surroundings day past due, which is integer. So, I will replace it by median.

```
# Filling values by median
df1[social_circle] = df1[social_circle].fillna(df1[social_circle].median())

# Fill the null values for AMT_GOODS_PRICE with the mean value
df1['AMT_GOODS_PRICE'].fillna(df1['AMT_GOODS_PRICE'].mean(),inplace=True)

# Fill the null values for AMT_ANNUITY with the mean value
df1['AMT_ANNUITY'].fillna(df1['AMT_ANNUITY'].mean(),inplace=True)

# Filling CNT_FAM_MEMBERS with the median
df1['CNT_FAM_MEMBERS'].fillna(df1['CNT_FAM_MEMBERS'].median(),inplace=True)

# Filling CNT_FAM_MEMBERS with the median
df1['DAYS_LAST_PHONE_CHANGE'].fillna(df1['DAYS_LAST_PHONE_CHANGE'].median(),inplace
=True)

# find the columns with days

days = []
for i in df1.columns:
 if 'DAYS' in i:
   days.append(i)
days
```

['DAYS_BIRTH', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH',

'DAYS_LAST_PHONE_CHANGE']

```
#There are 5 columns with the days change it in years

# change days to years
df1[days] = abs(df1[days])/365
# Rename the columns
df1.rename(columns = {'DAYS_BIRTH':'AGE','DAYS_EMPLOYED':'YEARS_EMPLOYED','DAYS_R
EGISTRATION':'YEARS_REGISTRATION','DAYS_ID_PUBLISH':'YEARS_ID_PUBLISH','DAYS_L
AST_PHONE_CHANGE':'YEARS_LAST_PHONE_CHANGE'},inplace=True)
```

Himanshu Jain
himjain567@gmail.com

8) Reading the dataset previous_applications.csv

```
df = pd.read_csv('/content/drive/MyDrive/Trainity/Loan Case Study/previous_application.csv')
df.info()
```

```
 0   SK_ID_PREV                   1670214 non-null  int64
 1   SK_ID_CURR                   1670214 non-null  int64
 2   NAME_CONTRACT_TYPE           1670214 non-null  object
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_DOWN_PAYMENT             774370 non-null   float64
 7   AMT_GOODS_PRICE              1284699 non-null  float64
 8   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object
 9   HOUR_APPR_PROCESS_START      1670214 non-null  int64
 10  FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object
 16  NAME_CONTRACT_STATUS         1670214 non-null  object
 17  DAYS_DECISION                1670214 non-null  int64
 18  NAME_PAYMENT_TYPE            1670214 non-null  object
 19  CODE_REJECT_REASON           1670214 non-null  object
 20  NAME_TYPE_SUITE              849809 non-null   object
 21  NAME_CLIENT_TYPE             1670214 non-null  object
 22  NAME_GOODS_CATEGORY          1670214 non-null  object
 23  NAME_PORTFOLIO               1670214 non-null  object
 24  NAME_PRODUCT_TYPE            1670214 non-null  object
 25  CHANNEL_TYPE                 1670214 non-null  object
 26  SELLERPLACE_AREA             1670214 non-null  int64
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object
 28  CNT_PAYMENT                  1297984 non-null  float64
```

Description of the data set

```
df.describe()
```

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

8 rows × 21 columns

Himanshu Jain
himjain567@gmail.com

9) Dealing with null values

Count null values

```
count_null_values(df)
```

```
RATE_INTEREST_PRIVILEGED       99.643698
RATE_INTEREST_PRIMARY          99.643698
AMT_DOWN_PAYMENT               53.636480
RATE_DOWN_PAYMENT              53.636480
NAME_TYPE_SUITE                49.119754
NFLAG_INSURED_ON_APPROVAL      40.298129
DAYS_TERMINATION               40.298129
DAYS_LAST_DUE                  40.298129
DAYS_LAST_DUE_1ST_VERSION      40.298129
DAYS_FIRST_DUE                 40.298129
DAYS_FIRST_DRAWING             40.298129
AMT_GOODS_PRICE                23.081773
AMT_ANNUITY                    22.286665
CNT_PAYMENT                    22.286366
PRODUCT_COMBINATION             0.020716
AMT_CREDIT                      0.000060
NAME_YIELD_GROUP                0.000000
NAME_PORTFOLIO                  0.000000
NAME_SELLER_INDUSTRY            0.000000
SELLERPLACE_AREA                0.000000
CHANNEL_TYPE                    0.000000
NAME_PRODUCT_TYPE               0.000000
SK_ID_PREV                      0.000000
NAME_GOODS_CATEGORY             0.000000
NAME_CLIENT_TYPE                0.000000
CODE_REJECT_REASON              0.000000
SK_ID_CURR                      0.000000
DAYS_DECISION                   0.000000
NAME_CONTRACT_STATUS            0.000000
NAME_CASH_LOAN_PURPOSE          0.000000
NFLAG_LAST_APPL_IN_DAY          0.000000
FLAG_LAST_APPL_PER_CONTRACT     0.000000
HOUR_APPR_PROCESS_START         0.000000
WEEKDAY_APPR_PROCESS_START      0.000000
AMT_APPLICATION                 0.000000
NAME_CONTRACT_TYPE              0.000000
NAME_PAYMENT_TYPE               0.000000
dtype: float64
```

Shape of the data set

```
df.shape
```

```
(1670214, 37)
```

Find the columns with more than 40% of null values

```
# find columns with more than 40% of null values
#print('Percentage (%) null values in each column')
null40 = count_null_values(df,40)
# this columns will be removed because of greater than 50% of null values
null40
```

```
RATE_INTEREST_PRIMARY 99.643698
```

Himanshu Jain
himjain567@gmail.com

RATE_INTEREST_PRIVILEGED 99.643698

AMT_DOWN_PAYMENT 53.636480

RATE_DOWN_PAYMENT 53.636480

NAME_TYPE_SUITE 49.119754

DAYS_FIRST_DRAWING 40.298129

DAYS_FIRST_DUE 40.298129

DAYS_LAST_DUE_1ST_VERSION 40.298129

DAYS_LAST_DUE 40.298129

DAYS_TERMINATION 40.298129

NFLAG_INSURED_ON_APPROVAL 40.298129

dtype: float64

Dop all the values with more than 40% of null values

```
df.drop(null40,axis=1,inplace=True)
df.shape
```

(1670214, 22)

Remove the columns which are unnecessary

```
df.drop(['WEEKDAY_APPR_PROCESS_START','HOUR_APPR_PROCESS_START','FLAG_LAST_APPL_PER_CONTRACT','NFLAG_LAST_APPL_IN_DAY'],axis=1,inplace=True)
df.shape
```

(1670214, 22)

Now find the columns with missing values more than 15%

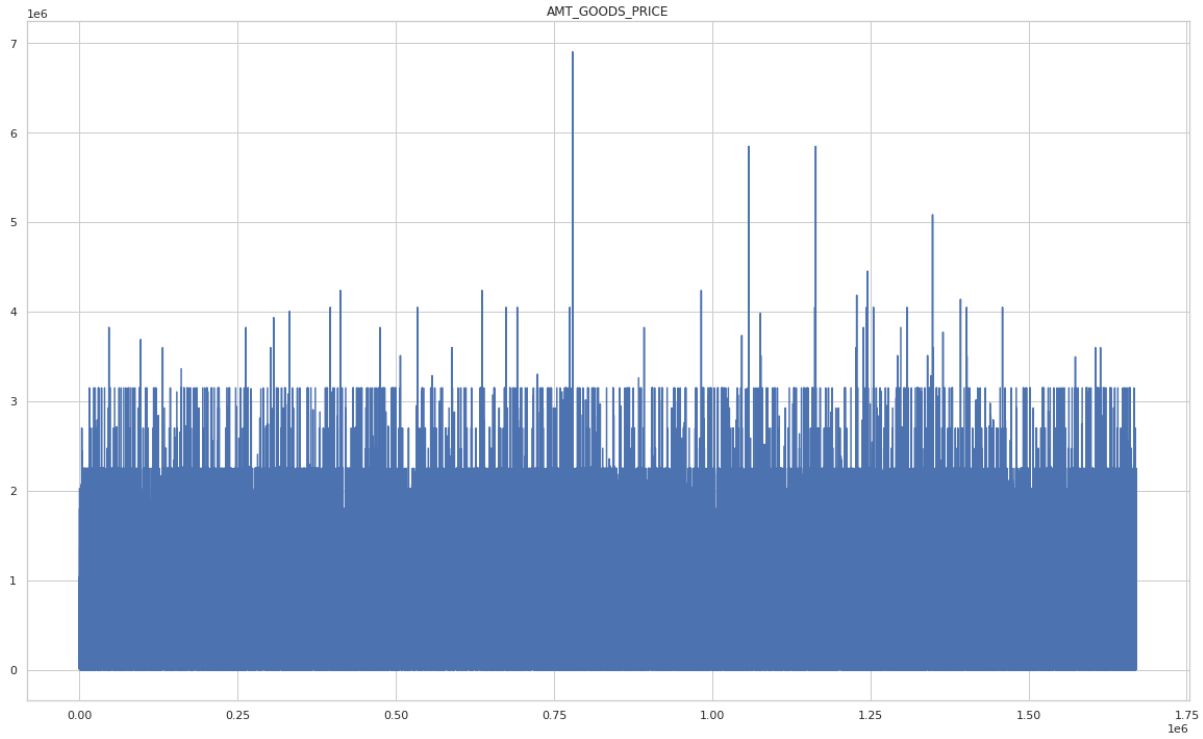
```
null15 = count_null_values(df,15)
# 10 columns with null values greater than 15%
null15
```

['AMT_ANNUITY', 'AMT_GOODS_PRICE', 'CNT_PAYMENT']

Plot goods price to check distributions

```
plt.figure(figsize=(20,12))
plt.plot(df['AMT_GOODS_PRICE'])
```

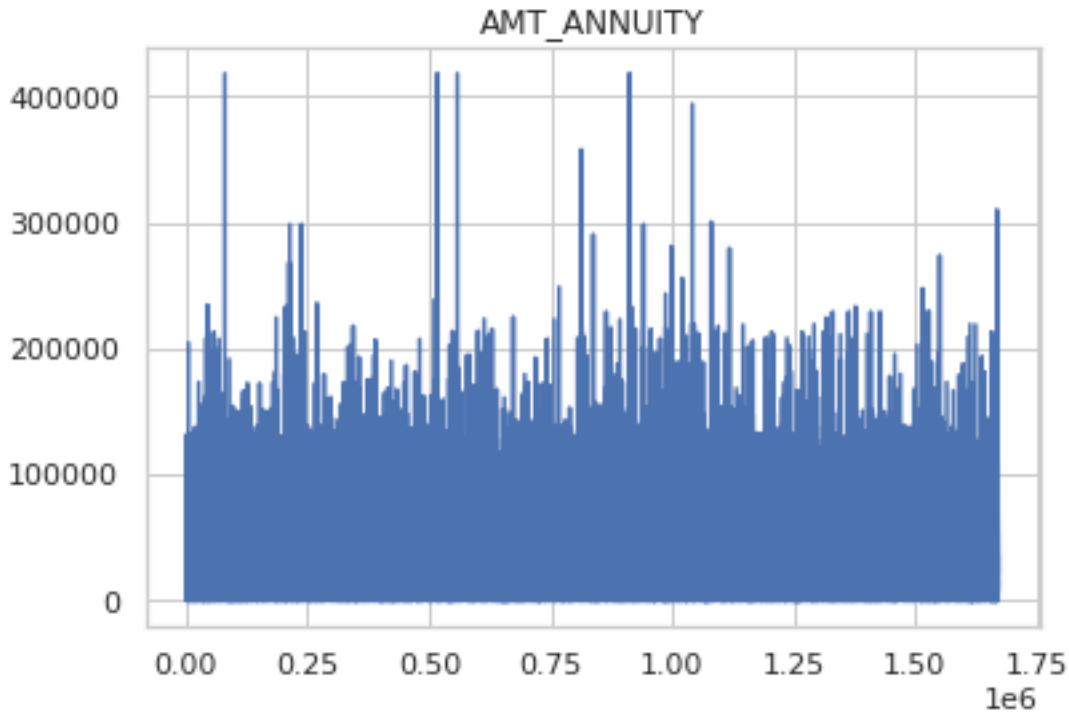
Himanshu Jain
himjain567@gmail.com




```
# Replace null in AMT_GOODS_PRICE by its mean as mean shows good replacement by the plot
df['AMT_GOODS_PRICE'].fillna(df['AMT_GOODS_PRICE'].mean(),inplace=True)
```

Plot amt annuity

```
plt.plot(df['AMT_ANNUITY'])
plt.title('AMT_ANNUITY')
```




```
# replace null values in ANT_ANNUITY by the mean
```

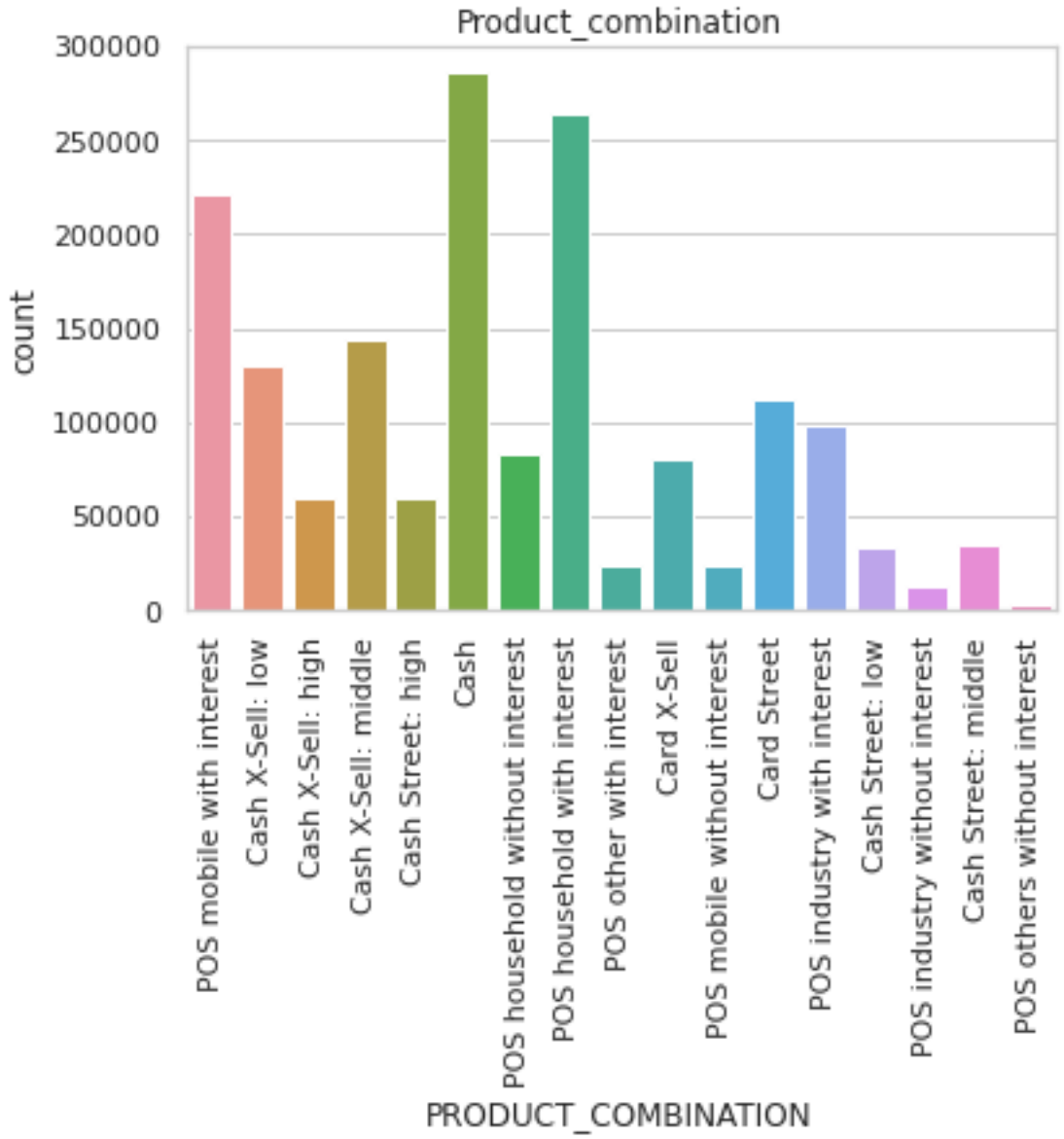
Himanshu Jain
himjain567@gmail.com

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

```
# replace null values in CNT_PAYMENT by its median
df['CNT_PAYMENT'].fillna(df['CNT_PAYMENT'].median(),inplace=True)
```

Plot product combination

```
sns.countplot(df['PRODUCT_COMBINATION'])
plt.xticks(rotation=90)
plt.title('Product  combination')
```




```
# replace null with the Unknown in PRODUCT_COMBINATION
df['PRODUCT_COMBINATION'].fillna('Unknown',inplace=True)
```

```
# Replace null in AMT_CREDIT by its mean
df['AMT_CREDIT'].fillna(df['AMT_CREDIT'].mean(),inplace=True)
# convert days to years for DAYS_DECISION column
```

Himanshu Jain
himjain567@gmail.com

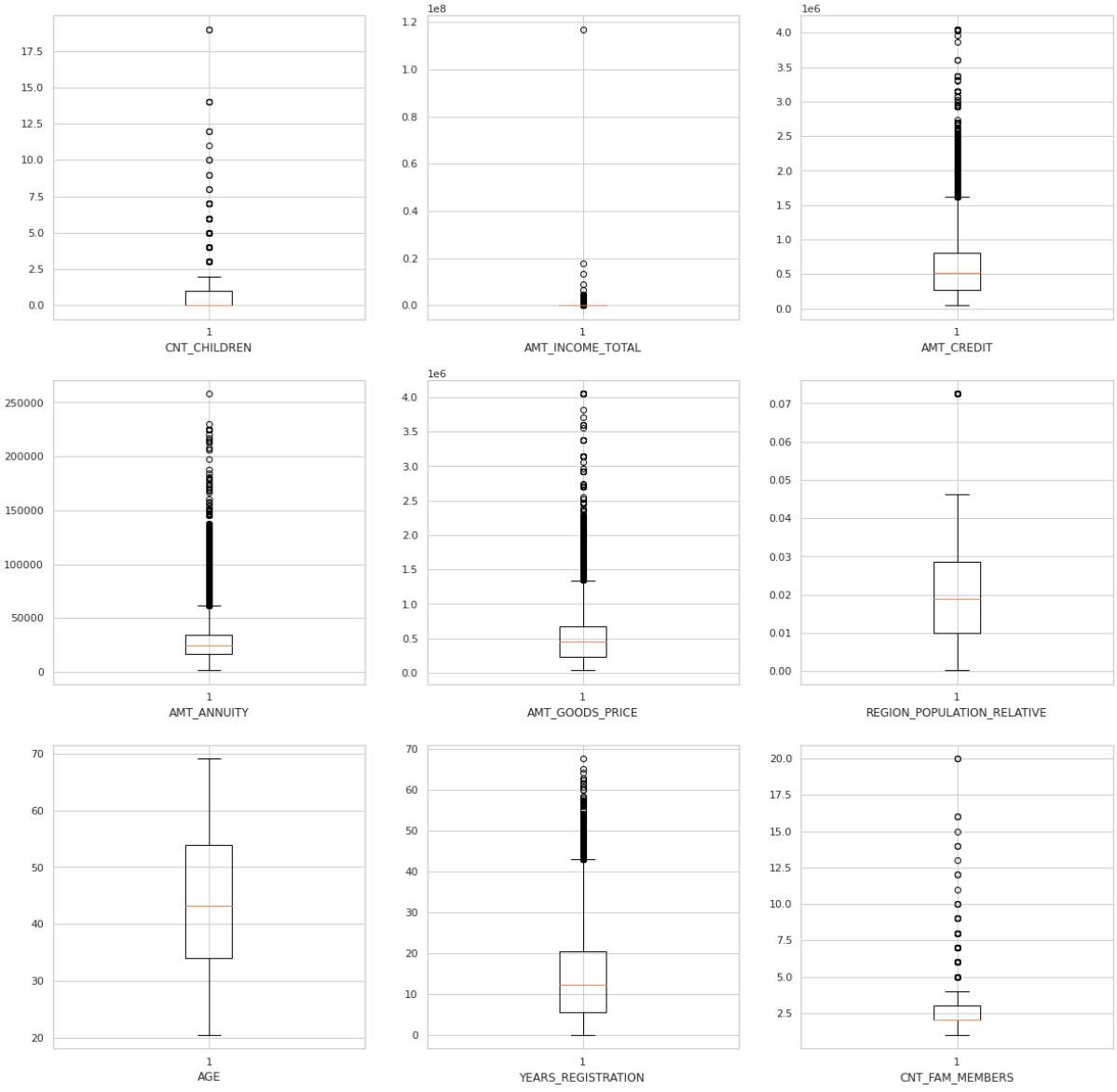
```
df['DAYS_DECISION'] = abs(df['DAYS_DECISION'])/365
```

10) Check Outliers

1) application_data.csv

Find outliers

```
out_check = 'CNT_CHILDREN,AMT_INCOME_TOTAL,AMT_CREDIT,AMT_ANNUITY,AMT_GOODS_PRICE,REGION_POPULATION_RELATIVE,AGE,YEARS_REGISTRATION,CNT_FAM_MEMBERS'.split(',')
plt.figure(figsize=(20,20))
#df1.boxplot(column = out_check, grid=False, rot=90, fontsize=15)
for i in range(len(out_check)):
  plt.subplot(3,3,i+1)
  plt.boxplot(df1[out_check[i]])
  plt.xlabel(out_check[i])
```



Himanshu Jain
himjain567@gmail.com

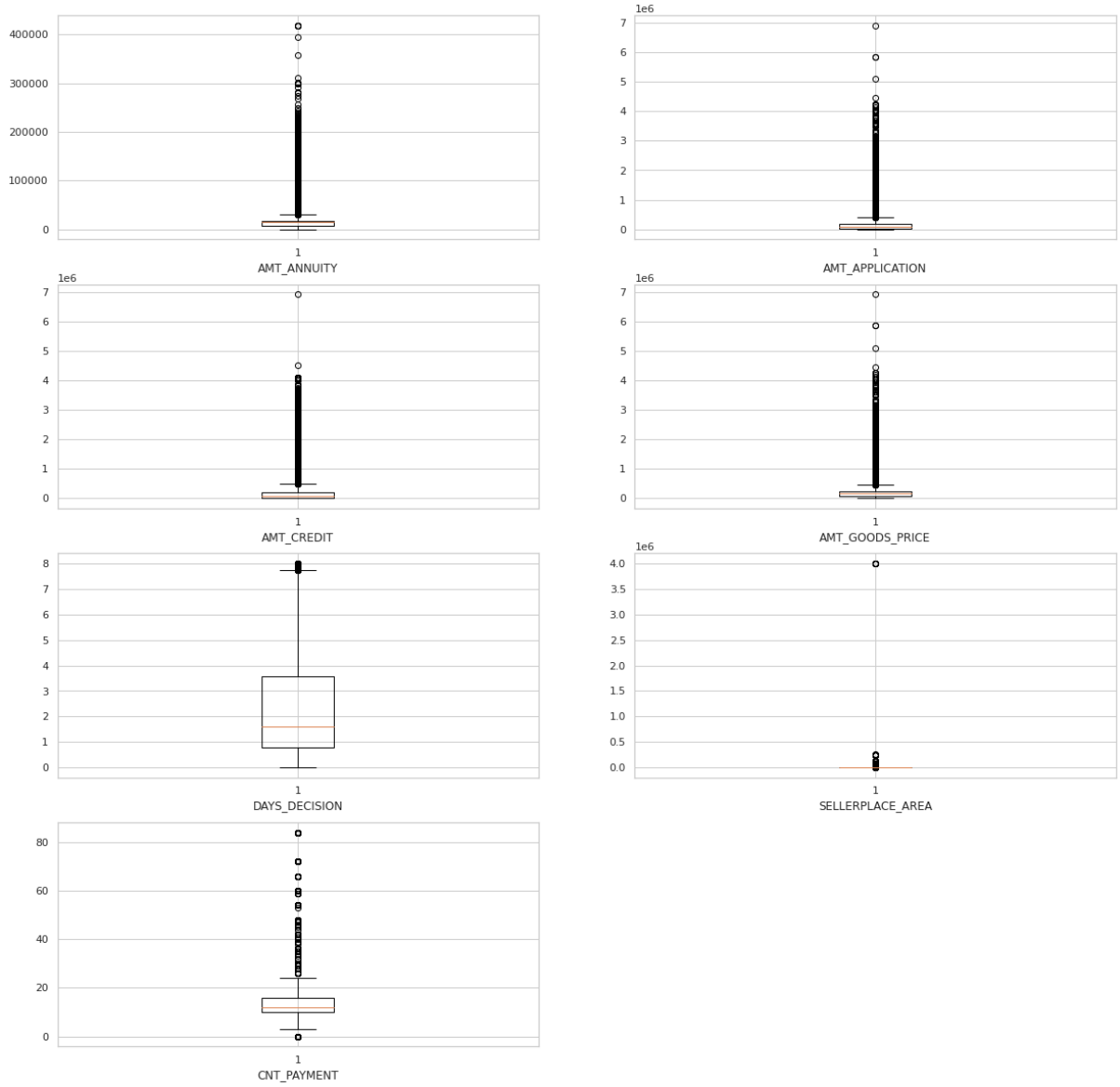
Inference:

There are outliers in each of the column except AGE. In the REGION_POPULATION_RELATIVE there are few outliers.

2) Previous_application.csv

 # Plot for outlier's check

```
out_check2 = 'AMT_ANNUITY,AMT_APPLICATION,AMT_CREDIT,AMT_GOODS_PRICE,DAYS_DECISION,SELLERPLACE_AREA,CNT_PAYMENT'.split(',')
plt.figure(figsize=(20,20))
#df1.boxplot(column = out_check, grid=False, rot=90, fontsize=15)
for i in range(len(out_check2)):
  plt.subplot(4,2,i+1)
  plt.boxplot(df[out_check2[i]])
  plt.xlabel(out_check2[i])
```



Himanshu Jain
himjain567@gmail.com

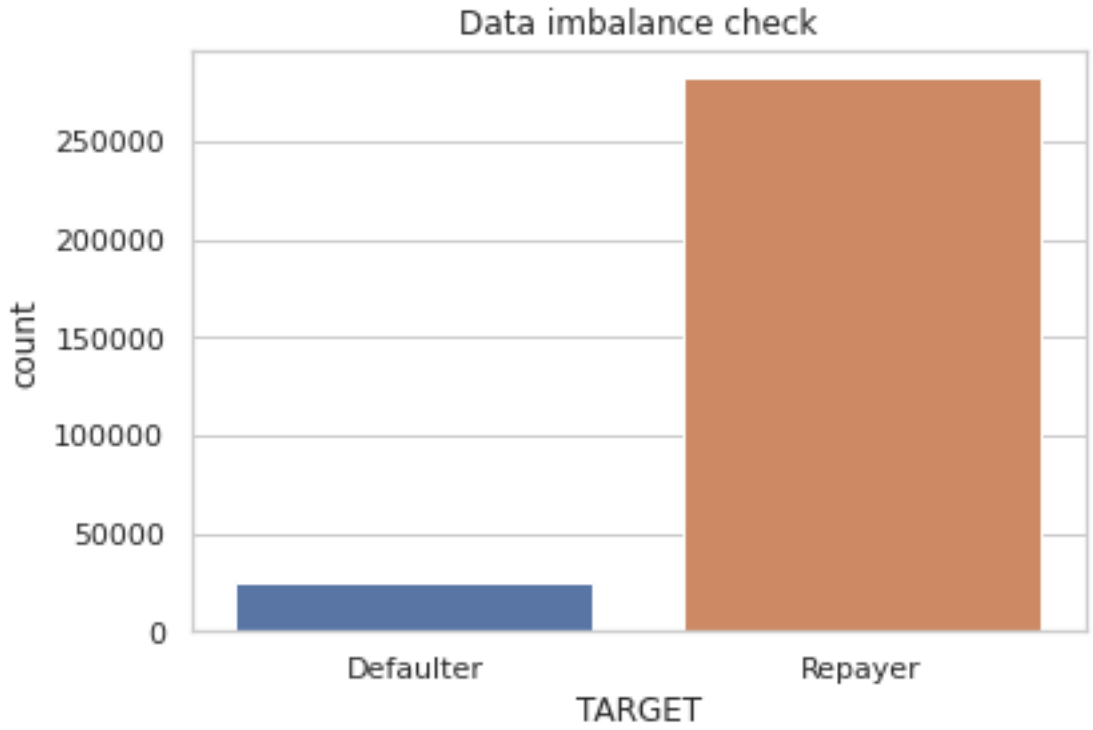
Inference:

Each of the numerical column contain outliers.

11) Data Imbalance

```
# before moving to the data imbalance rename the data values 0 to repayer and 1 to defaulter for better und
erstanding
df1["TARGET"] = df1["TARGET"].replace({1:"Defaulter",0:"Repayer"})
```

 # Plot Target column



 # percentage value count of defaulter and repayer

```
df1['TARGET'].value_counts()*100/len(df1)
```

Repayer 91.927118

Defaulter 8.072882

Name: TARGET, dtype: float64

12) Univariate Analysis for application_data.csv

 # Function to write labels in the bar plot

```
def addlabels(x,y):
    for i in range(len(x)):
        plt.text(i, y[i]//2, y[i], ha = 'center')
```

Himanshu Jain
himjain567@gmail.com

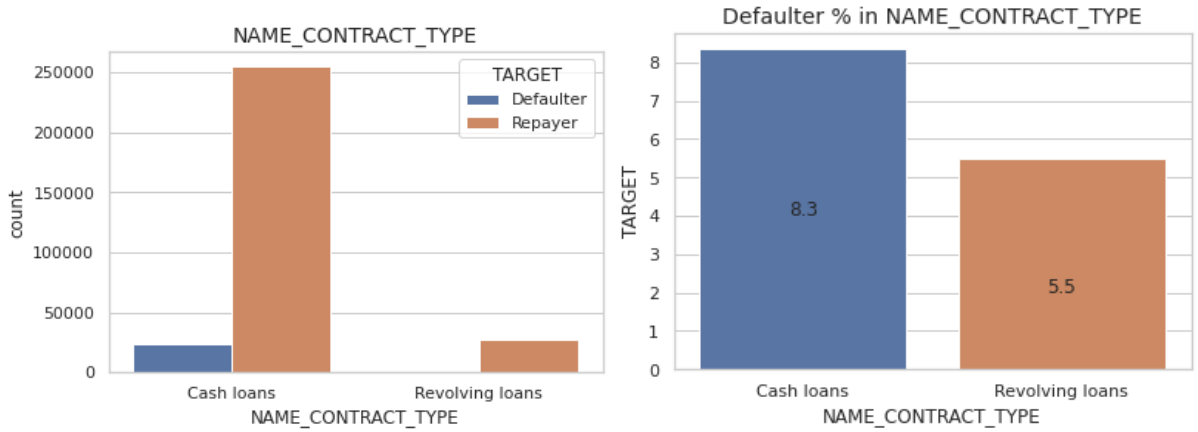
```
# function for univariate analysis
# Cateriogical and numerical
def univariate(df,data,target):
  col = df[data]
  tar = df[target]
  types = col.dtypes

  ex = col.value_counts()
  if len(ex)>8:
    plt.figure(figsize = (20,12))
    sns.countplot(x = col, hue = tar)
    plt.xticks(rotation = 90)
    plt.title(data,fontdict={'fontsize': 20})
  elif len(ex)>4:
    plt.figure(figsize = (15,8))
    sns.countplot(x = col, hue = tar)
    plt.xticks(rotation = 45)
    plt.title(data,fontdict={'fontsize': 20})
  else:
    plt.figure()
    sns.countplot(x = col, hue = tar)
    plt.title(data,fontdict={'fontsize': 14})
  plt.xlabel(data)

  # Calculate defaulter % for the data
  pf = df[[data,target]].value_counts().reset_index()
  percent= []
  for i in pf[data].unique():
    try:
      percent.append(pf[(pf[data]==i)&(pf[target]=='Defaulter')][0].values[0]*100/(pf[(pf[data]==i)&(pf[tar
get]=='Defaulter')][0].values[0]+pf[(pf[data]==i)&(pf[target]=='Repayer')][0].values[0]))
    except:
      percent.append(0)
  if len(percent)>8:
    plt.figure(figsize = (20,12))
    sns.barplot(y = percent, x = pf[data].unique())
    plt.xticks(rotation = 90)
    plt.title('Defaulter % in '+data,fontdict={'fontsize': 20})
  elif len(percent)>4:
    plt.figure(figsize = (15,8))
    sns.barplot(y = percent, x = pf[data].unique())
    plt.xticks(rotation = 45)
    plt.title('Defaulter % in '+data,fontdict={'fontsize': 20})
  else:
    plt.figure()
    sns.barplot(y = percent, x = pf[data].unique())
    plt.title('Defaulter % in '+data,fontdict={'fontsize': 14})
```

Himanshu Jain
himjain567@gmail.com

```
plt.xlabel(data)
plt.ylabel(target)
addlabels(x = pf[data].unique(),y=np.round(percent,1))
```

 # Check for NAME_CONTRACT_TYPE based on loan repay status

```
univariate(df1,'NAME_CONTRACT_TYPE','TARGET')
```



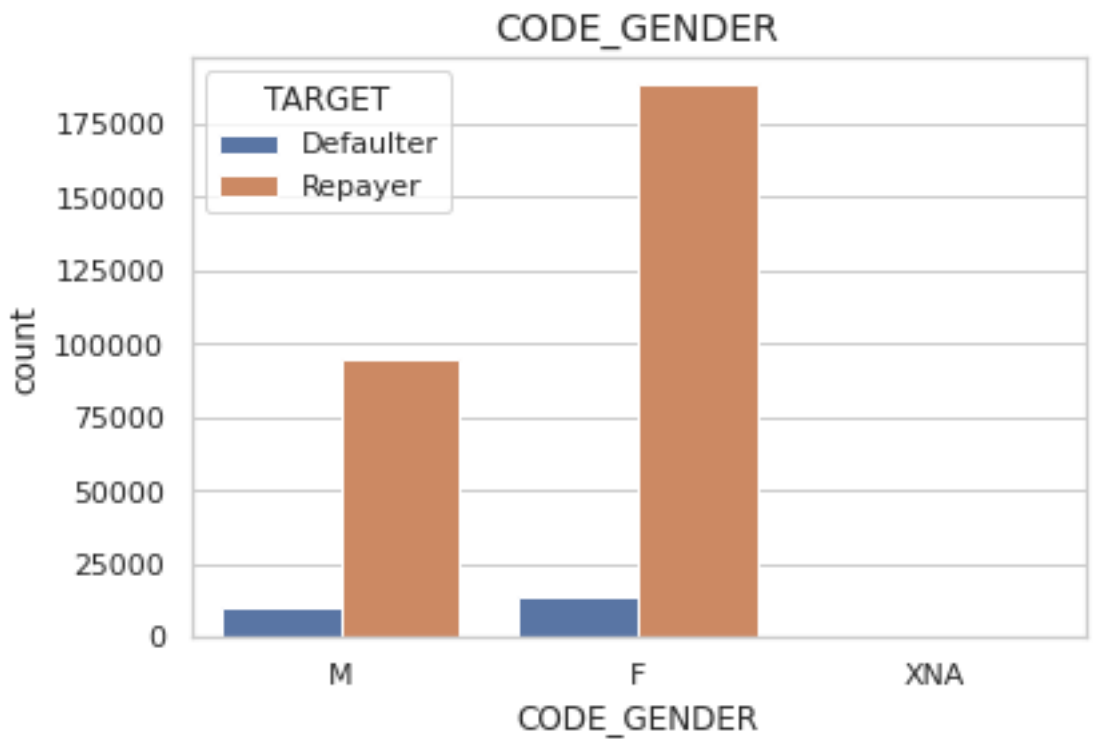
Inference:

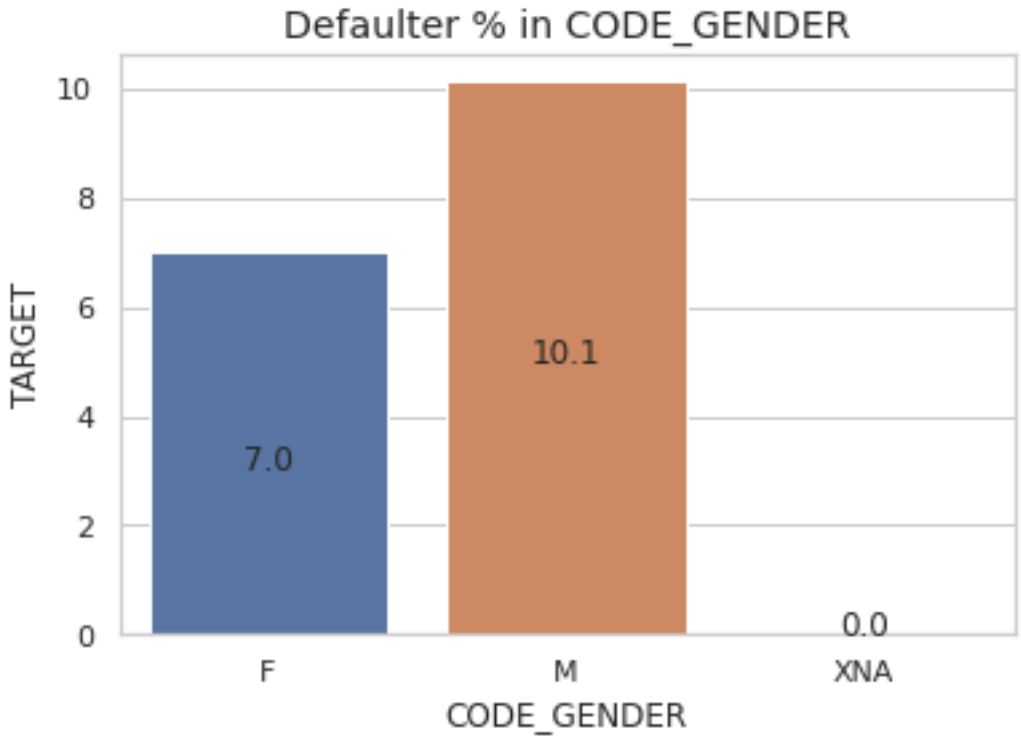
- Revolving loans are only 8-12% compared to cash loans.
- Revolving loans have less defaulter%

 # Check for CODE_GENDER based on loan repay status

```
univariate(df1,'CODE_GENDER','TARGET')
```

Himanshu Jain
himjain567@gmail.com





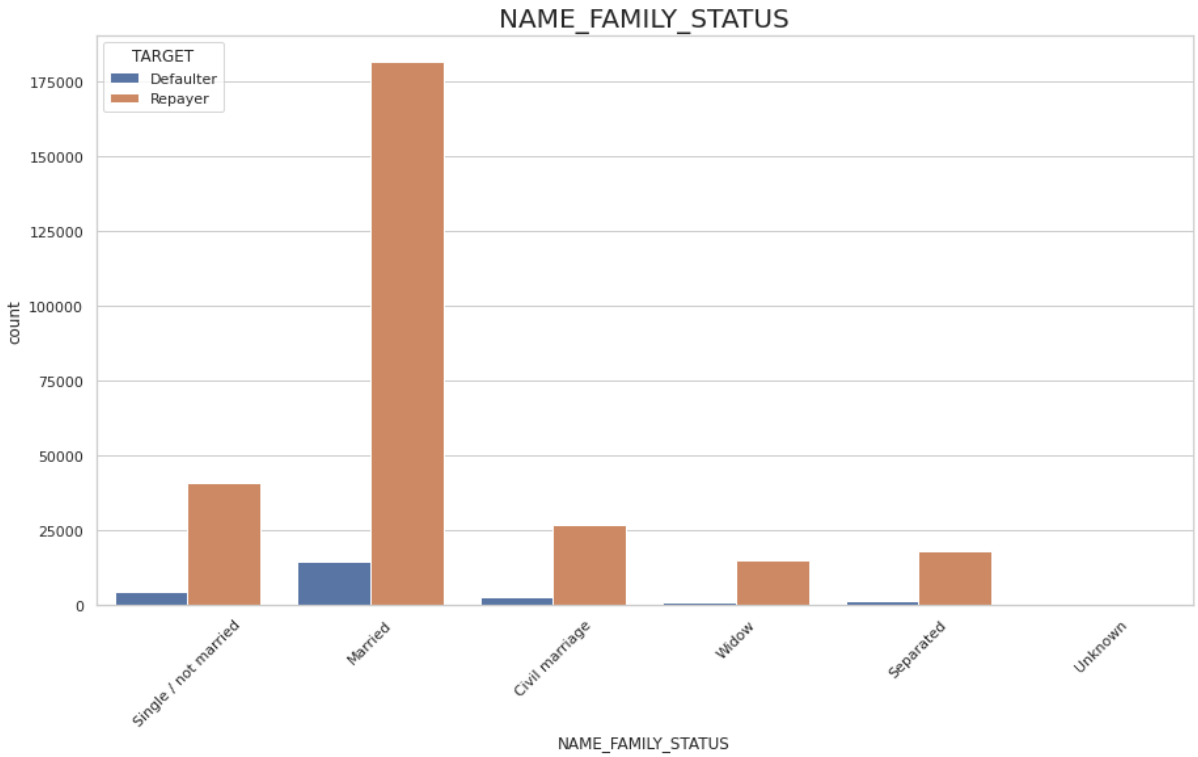
Inference:

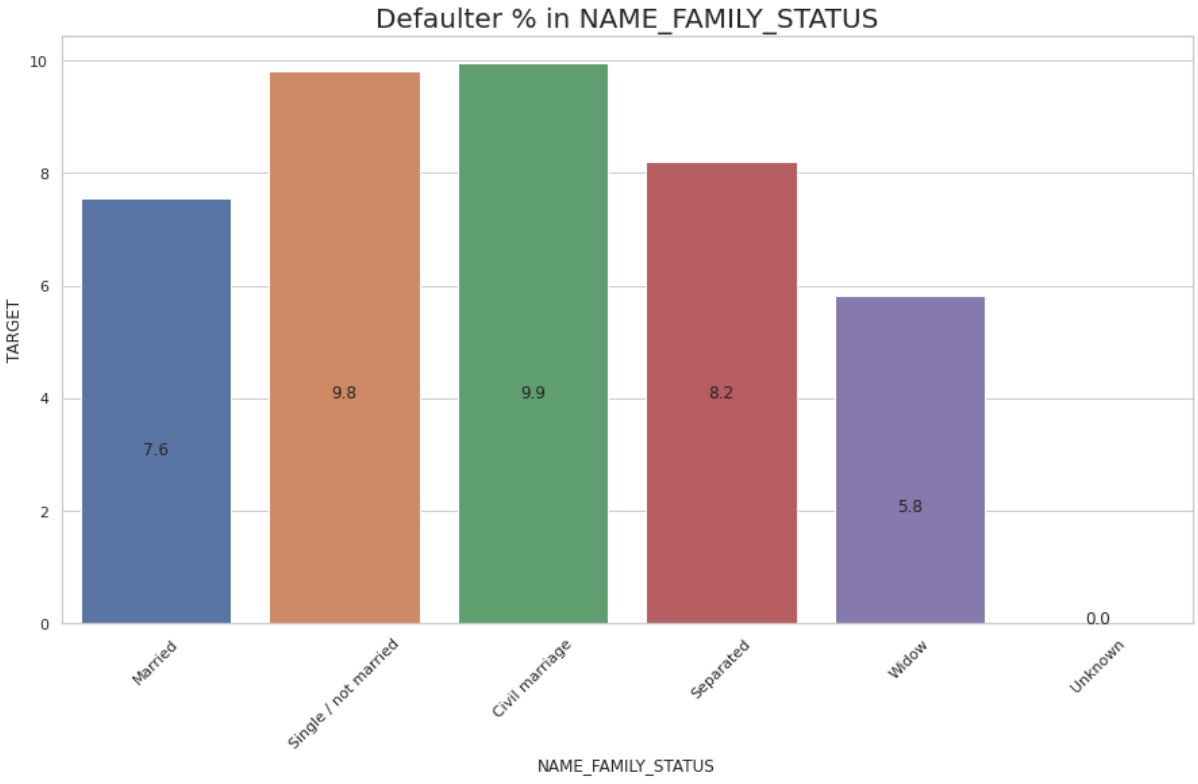
- Females are most likely to take loans compared to males almost twice.
- Male have high defaulter than female.
- There are only few cases where people have not shared the gender but they paid all the loan.

Himanshu Jain
himjain567@gmail.com

 # Check for NAME_FAMILY_STATUS based on loan repay status

```
univariate(df1,'NAME_FAMILY_STATUS','TARGET')
```





Inference:

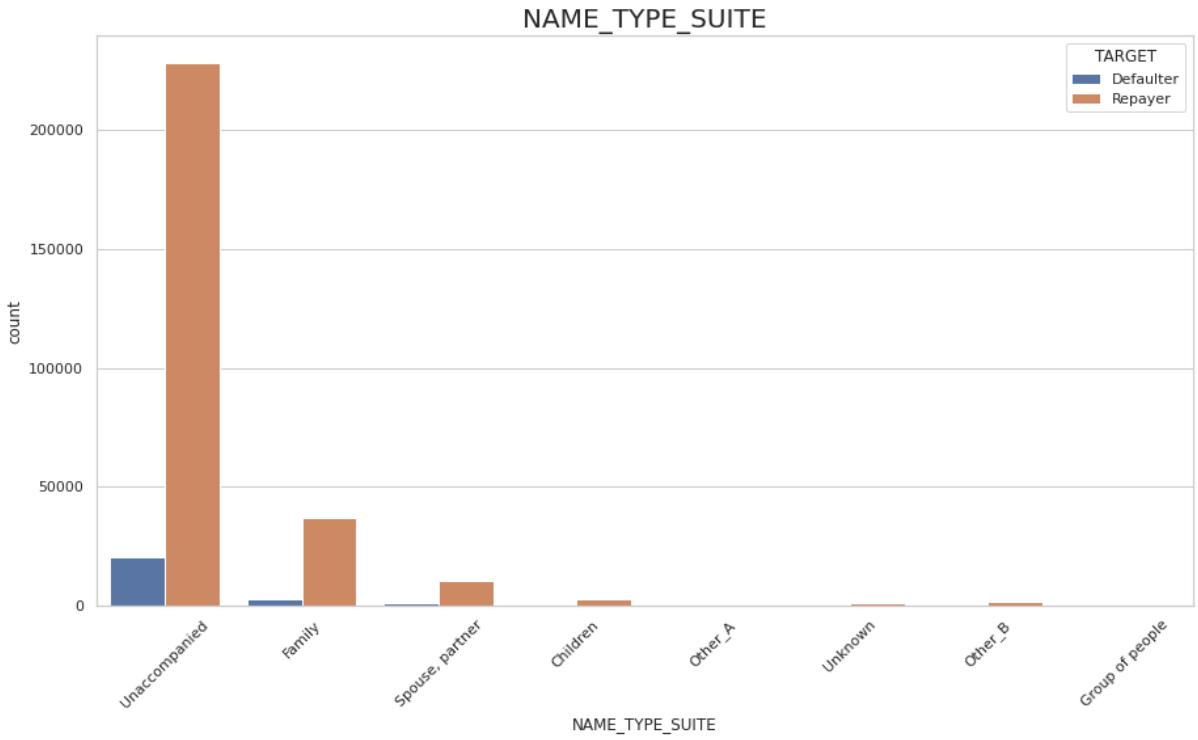
- Married Family has taken more loans compared to all the category.

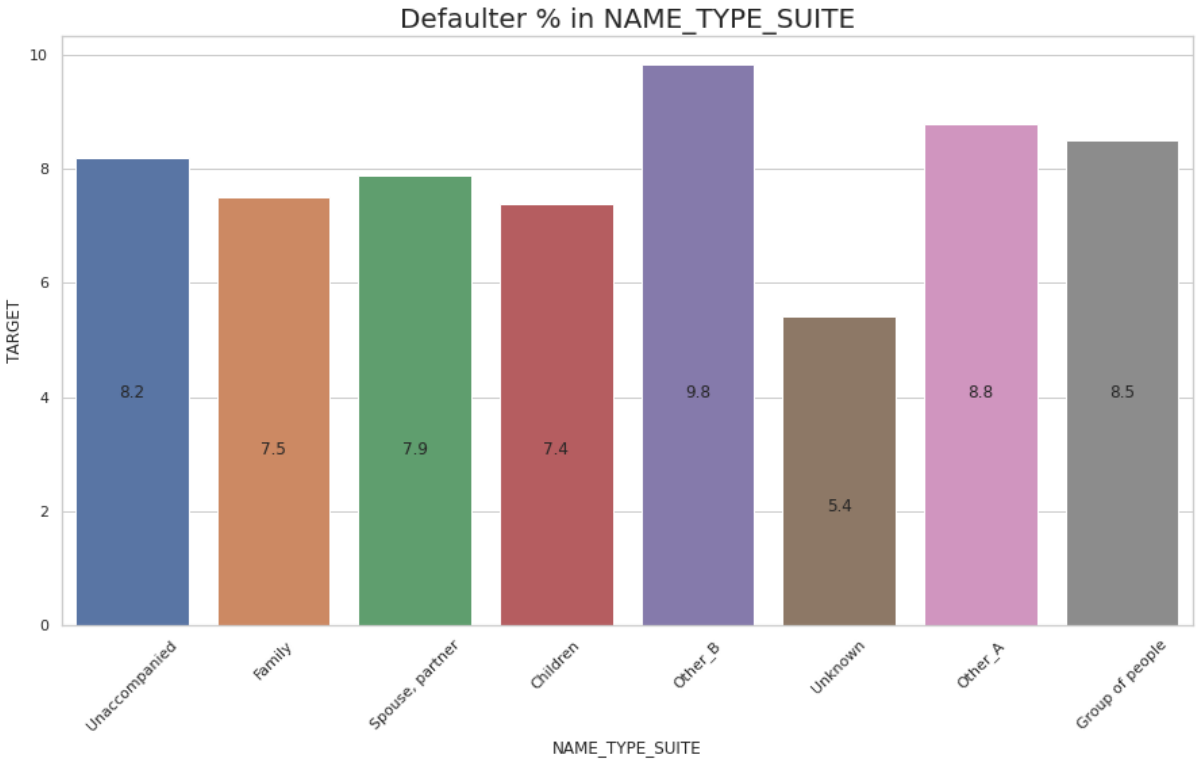
Himanshu Jain
himjain567@gmail.com

- In all the categories civil marriage have the highest default % followed by single / not married, separated, Married, and widow.
- Widow has the lowest defaulter value.

 # Check for NAME_TYPE_SUITE based on loan repay status

```
univariate(df1,'NAME_TYPE_SUITE','TARGET')
```





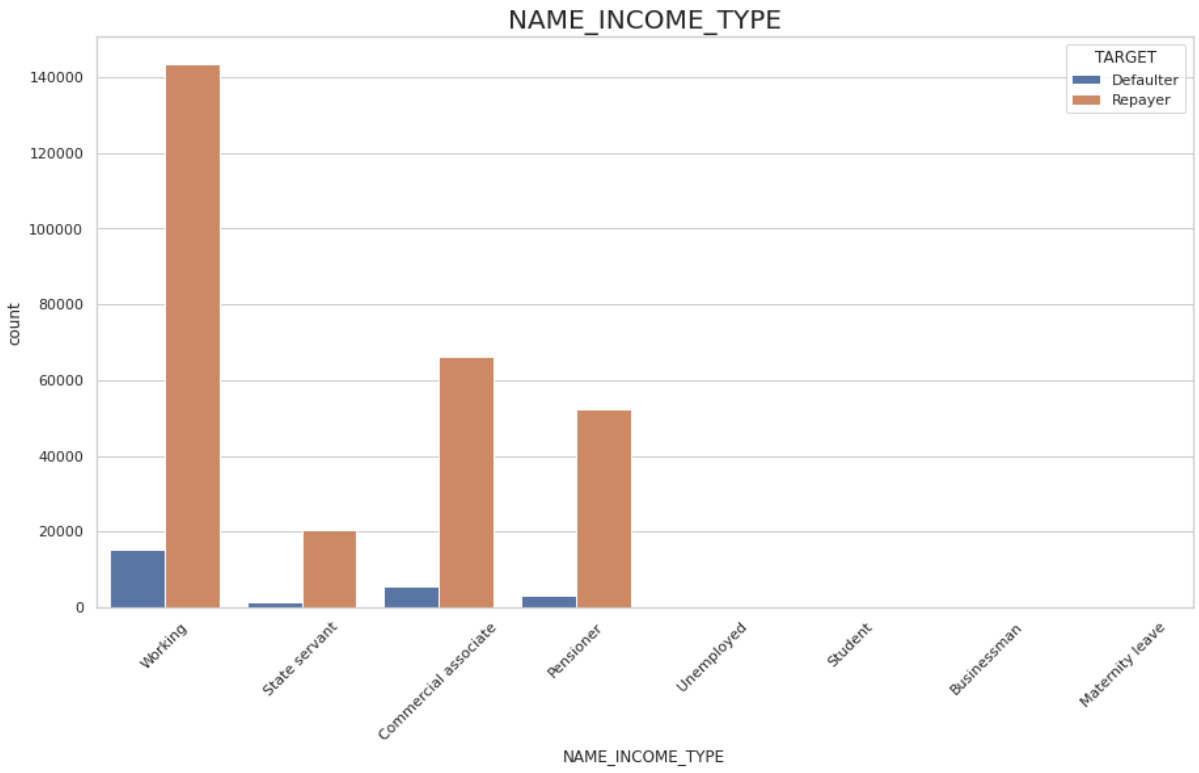
Himanshu Jain
himjain567@gmail.com

Inference:

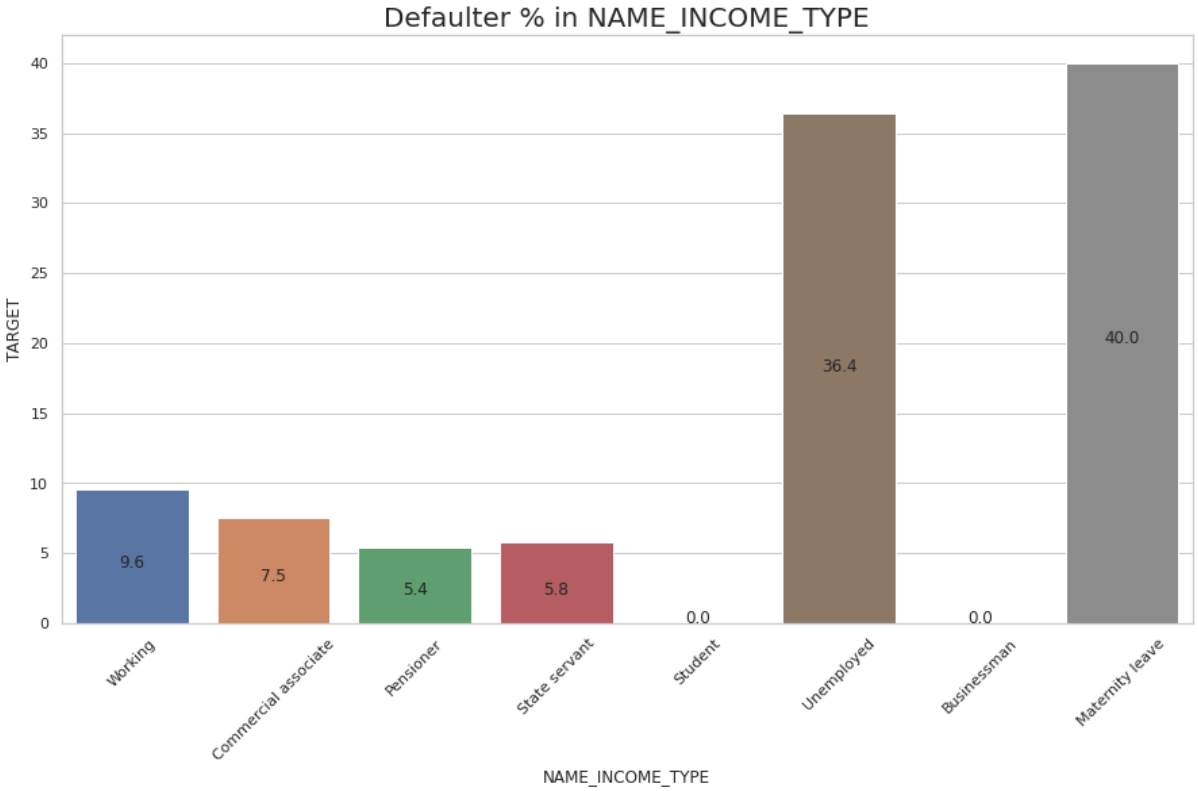
- Accompanied category has the highest loans.
- The maximum and minimum defaulter% are Other_B (9.8%) and Unknown (5.4%).

 # Check for NAME_INCOME_TYPE based on loan repay status

```
univariate(df1,'NAME_INCOME_TYPE','TARGET')
```



Himanshu Jain
himjain567@gmail.com



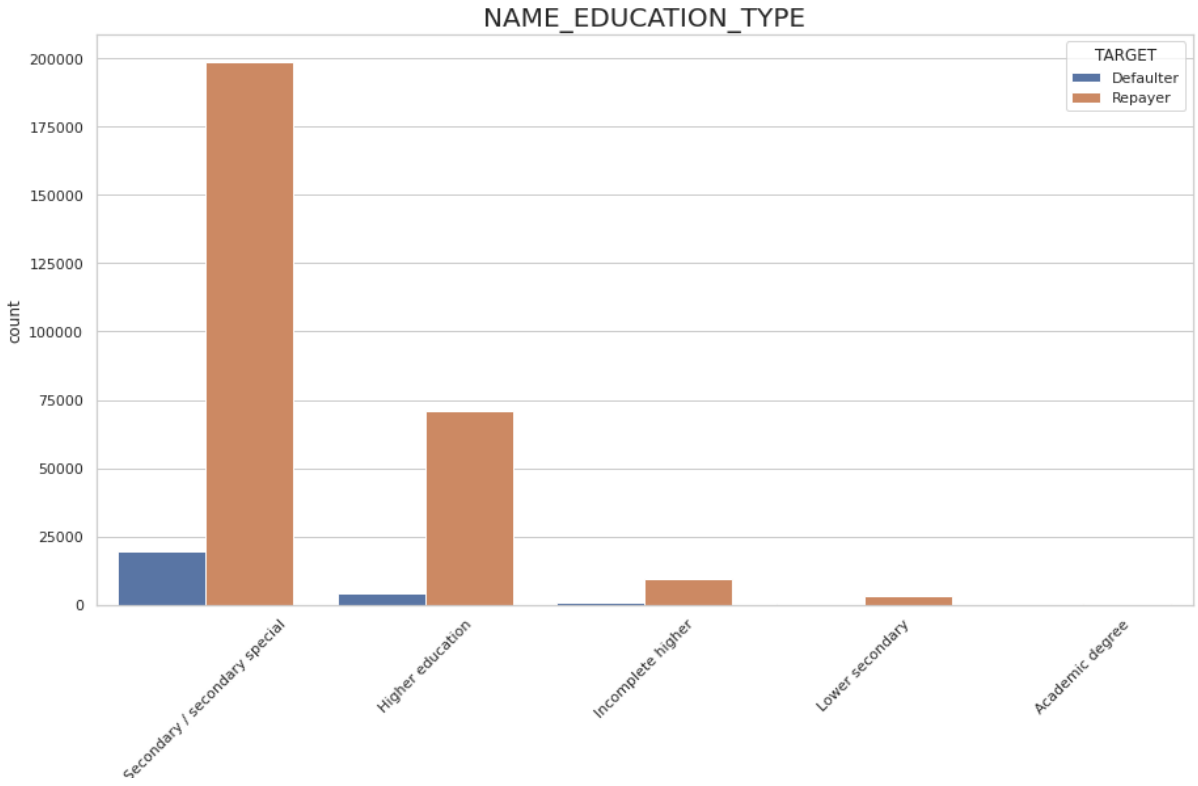
Inference:

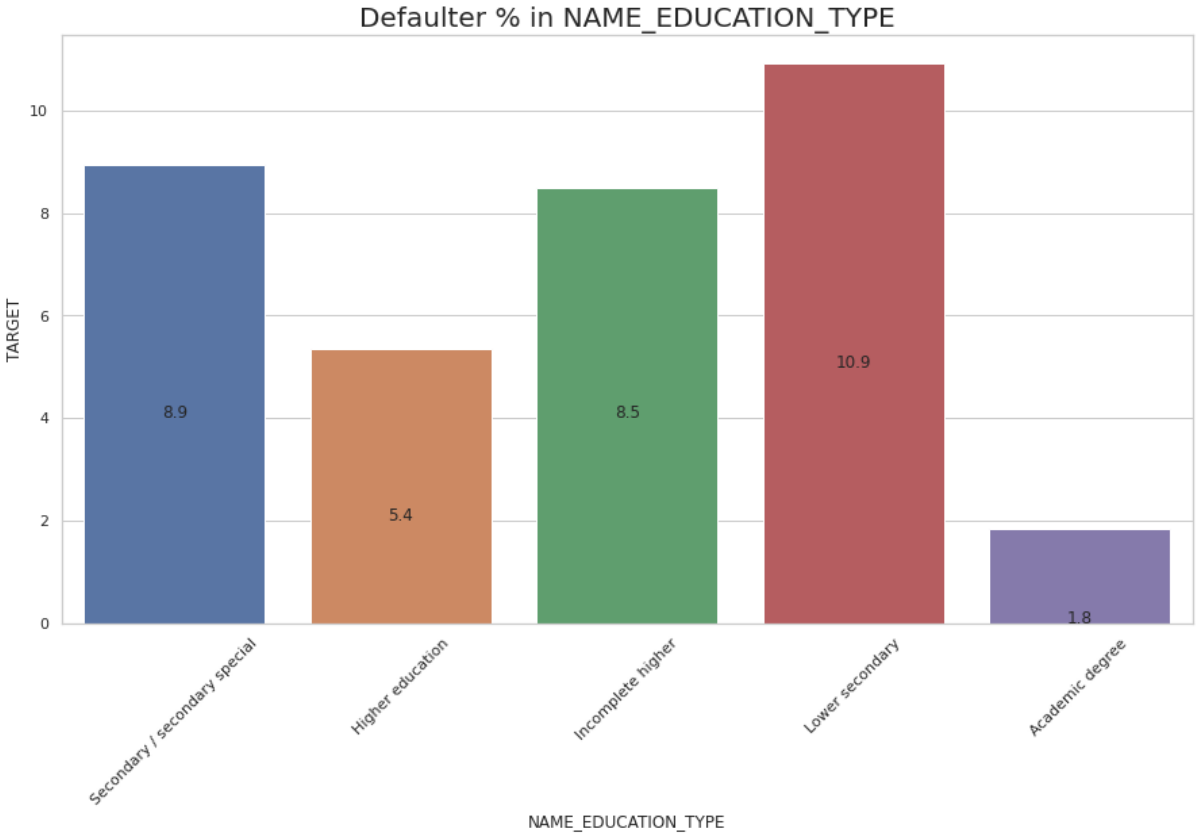
- Working is more likely to take a loan, followed by commercial assistant, pensioner, etc.
- People on maternity leave is the riskiest category followed by unemployed, rest lies in less than 10%.
- Student and business are less in number but no records of defaulter hence it is the safest category to provide the loan.

 # Check for NAME_EDUCATION_TYPE based on loan repay status

```
univariate(df1,'NAME_EDUCATION_TYPE','TARGET')
```

Himanshu Jain
himjain567@gmail.com





Inference:

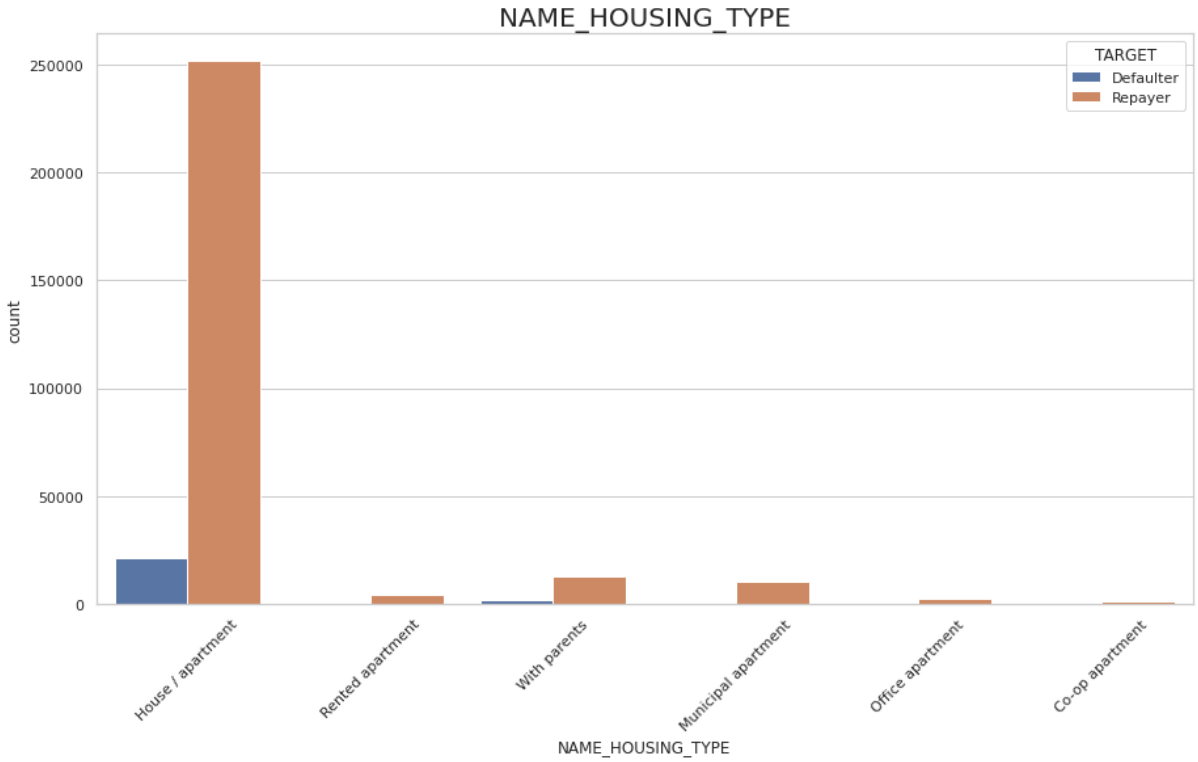
- People with the secondary / secondary special category has the hights loan application.

Himanshu Jain
himjain567@gmail.com

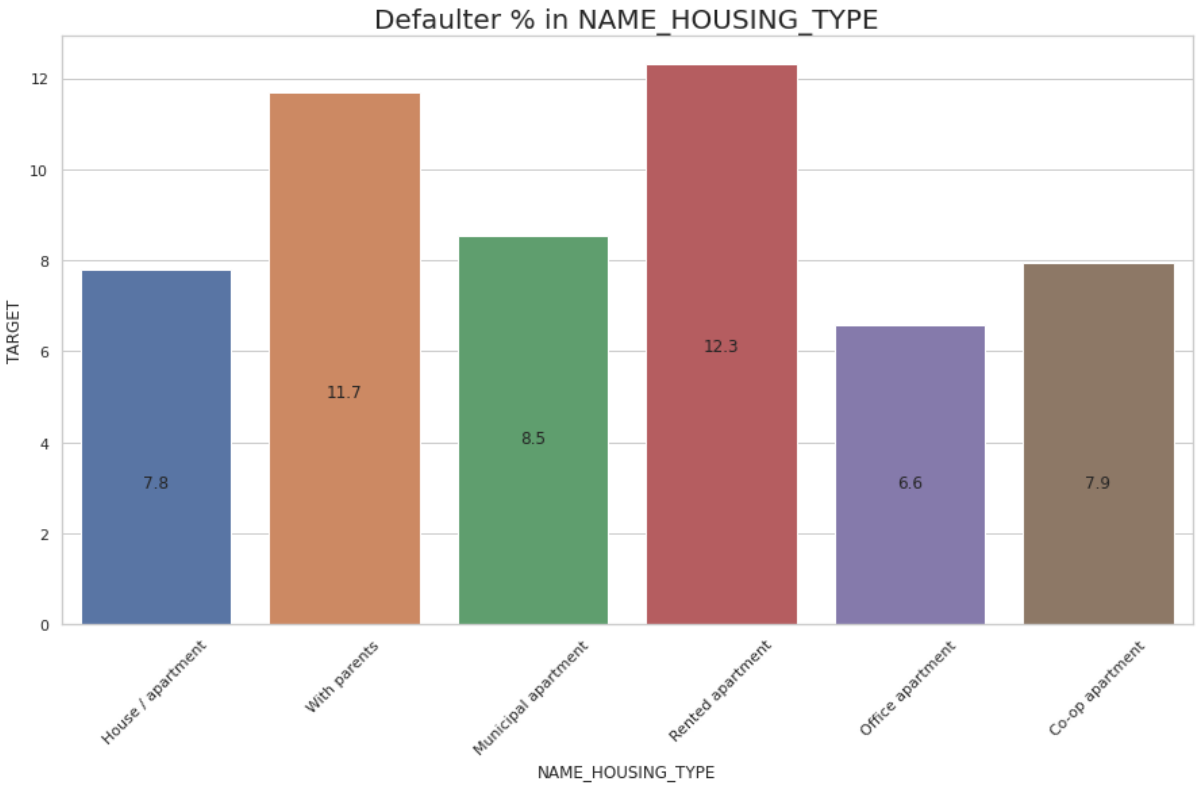
- lower secondary has the highest defaulter%.
- Academic degree has the lowest defaulter% among all the categories.

 # Check for NAME_HOUSING_TYPE based on loan repay status

```
univariate(df1,'NAME_HOUSING_TYPE','TARGET')
```



Himanshu Jain
himjain567@gmail.com



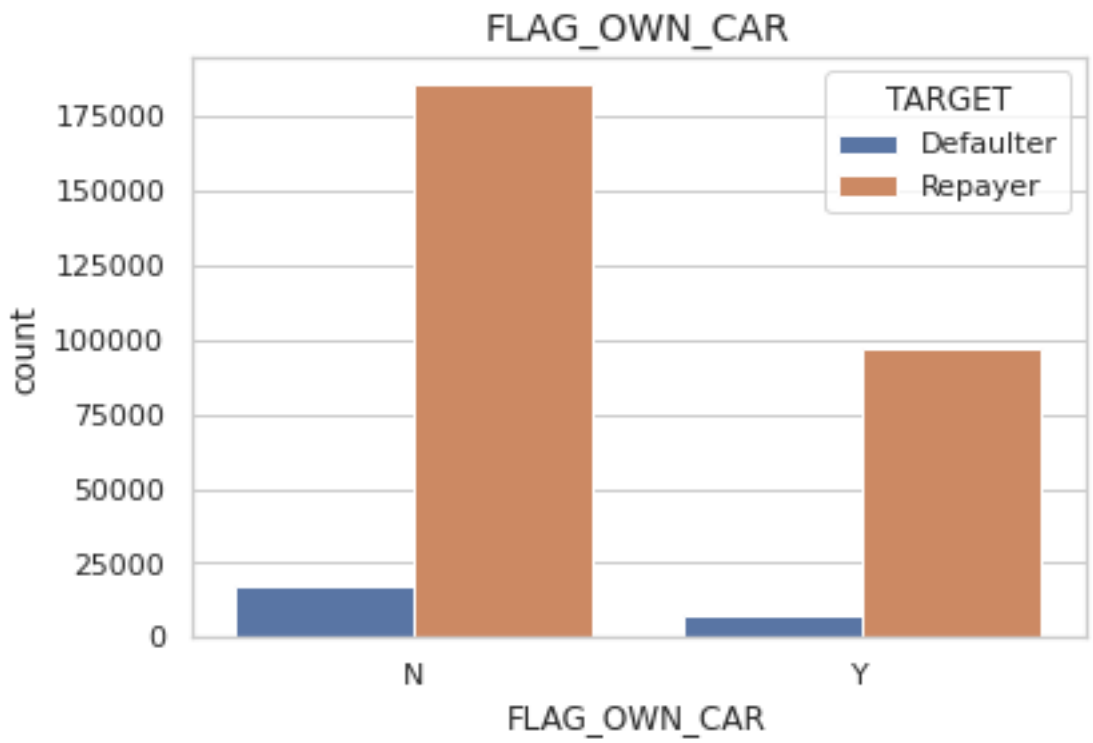
Inference:

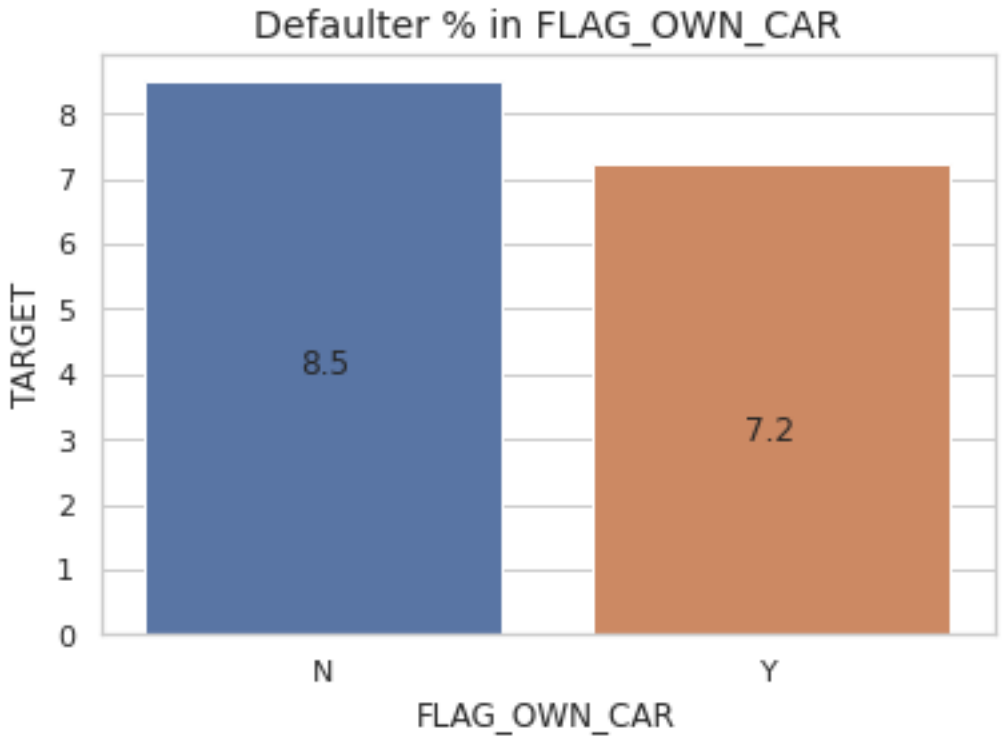
- People with the house / apartment have the highest application.
- Rented apartment is the riskiest category among all.

 # Check for FLAG_OWN_CAR based on loan repay status

```
univariate(df1,'FLAG_OWN_CAR','TARGET')
```

Himanshu Jain
himjain567@gmail.com





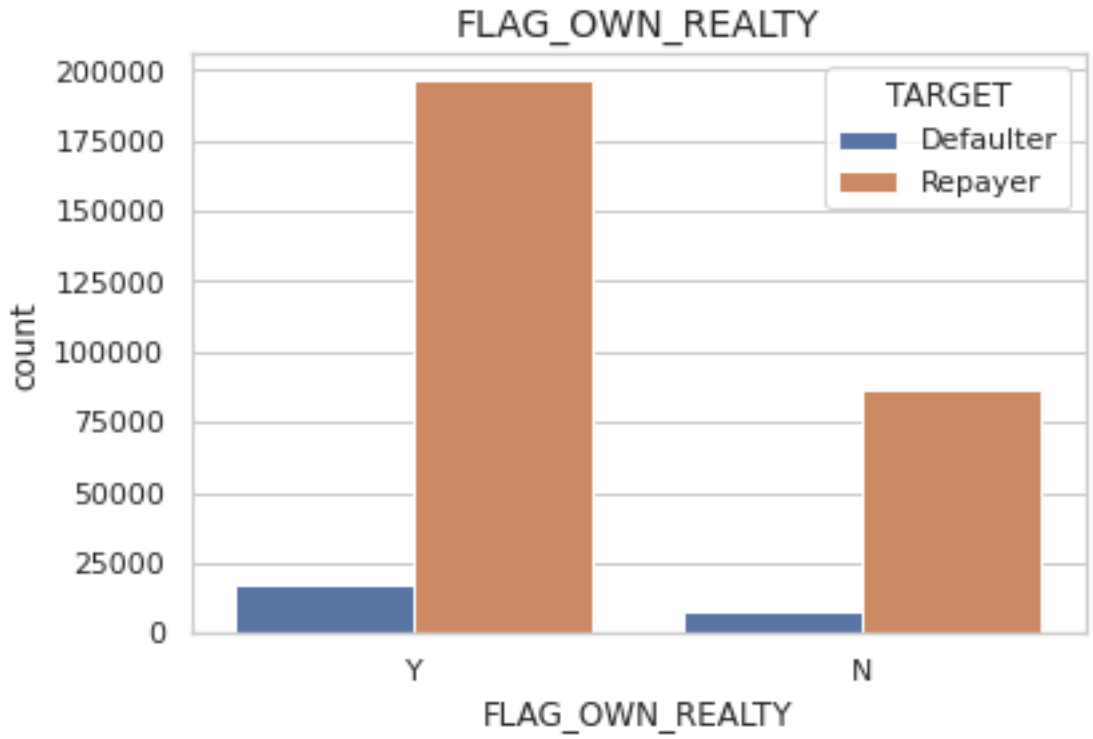
Inference:

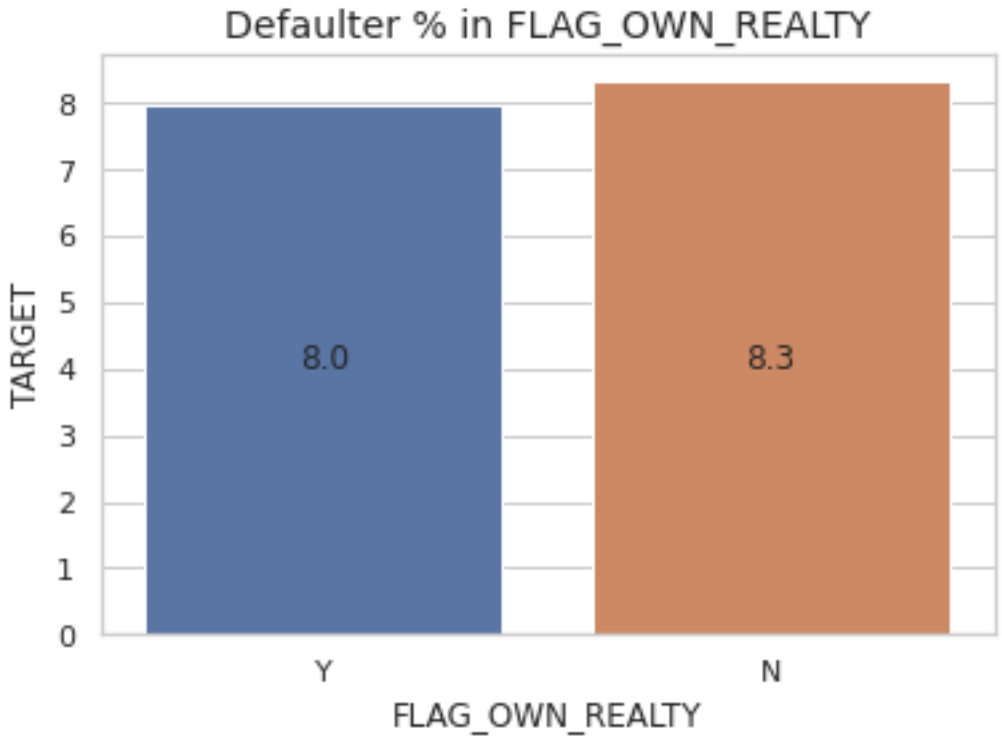
- People who do not own car have high loan applications as well as high defaulter%.

 # Check for FLAG_OWN__REALTY type based on loan repay status

Himanshu Jain
himjain567@gmail.com

```
univariate(df1,'FLAG_OWN_REALTY','TARGET')
```





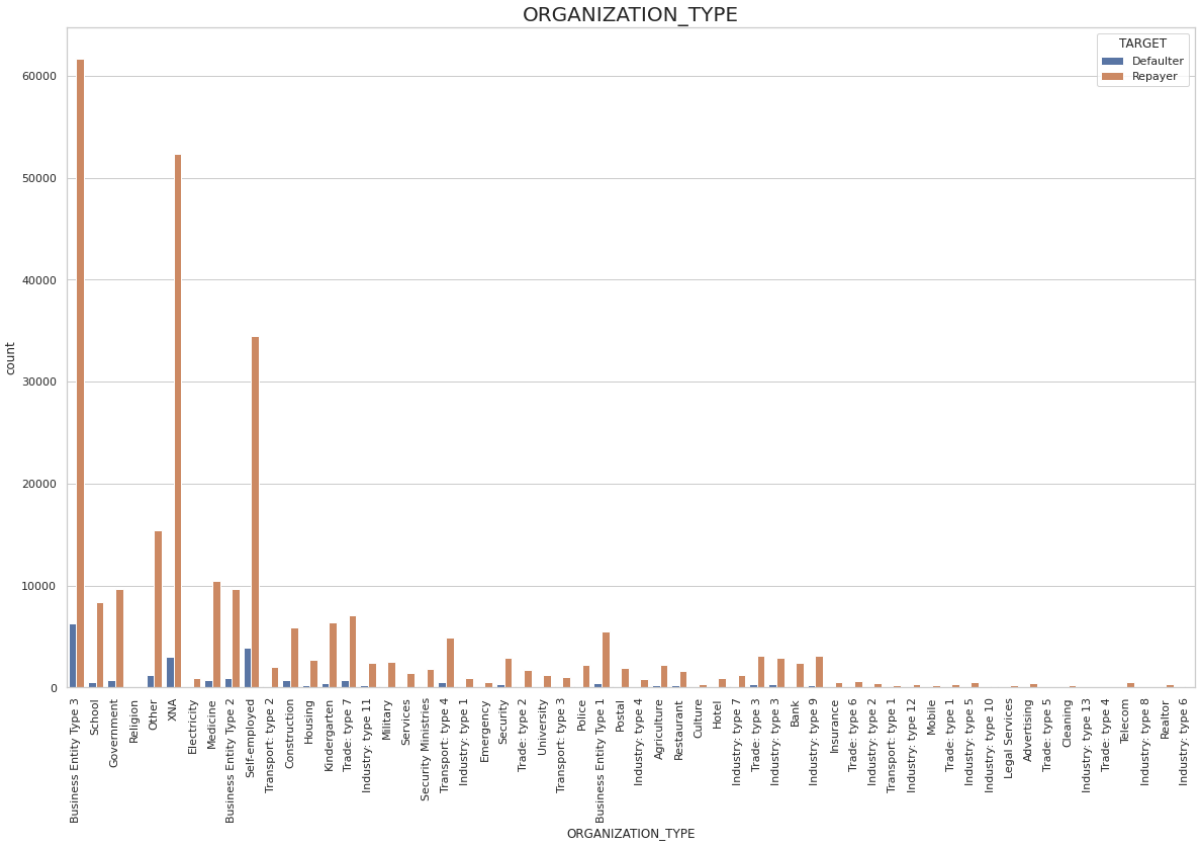
Inference:

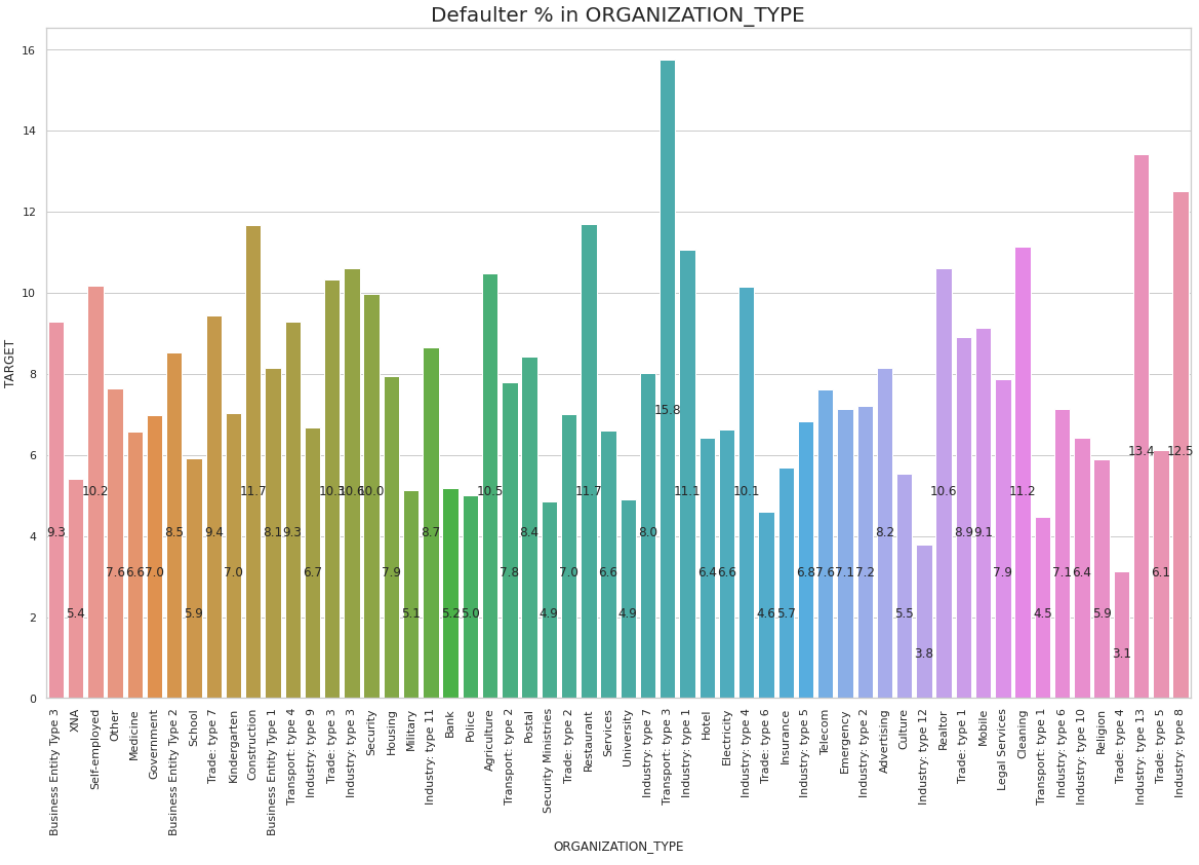
- people who own realty have high loan application and low defaulter% than who does not own realty.

Himanshu Jain
himjain567@gmail.com

Check for ORGANIZATION_TYPE based on loan repay status

```
univariate(df1,'ORGANIZATION_TYPE','TARGET')
```





Himanshu Jain
himjain567@gmail.com

Inference:

- Business entity type 3, XNA and self employed have the highest loan application others are less than 15%.
- riskiest category is transport: type 3 (15.8%)

 # Convert AMT_INCOME_TOTAL to bins

```
incomebins=[0,25000,50000,75000,100000,125000,150000,175000,200000,225000,250000,275000,300000,325000,350000,375000,400000,425000,450000,475000,500000,10000000000]
incomeslots = ['0-25000','25000-50000','50000-75000','75000,100000','100000-125000', '125000-150000', '150000-175000','175000-200000','200000-225000','225000-250000','250000-275000','275000-300000','300000-325000','325000-350000','350000-375000','375000-400000','400000-425000','425000-450000','450000-475000','475000-500000','500000 and above']
df1['AMT_INCOME_TOTAL_RANGE']=pd.cut(df1['AMT_INCOME_TOTAL'],bins=incomebins,labels=incomeslots)
```

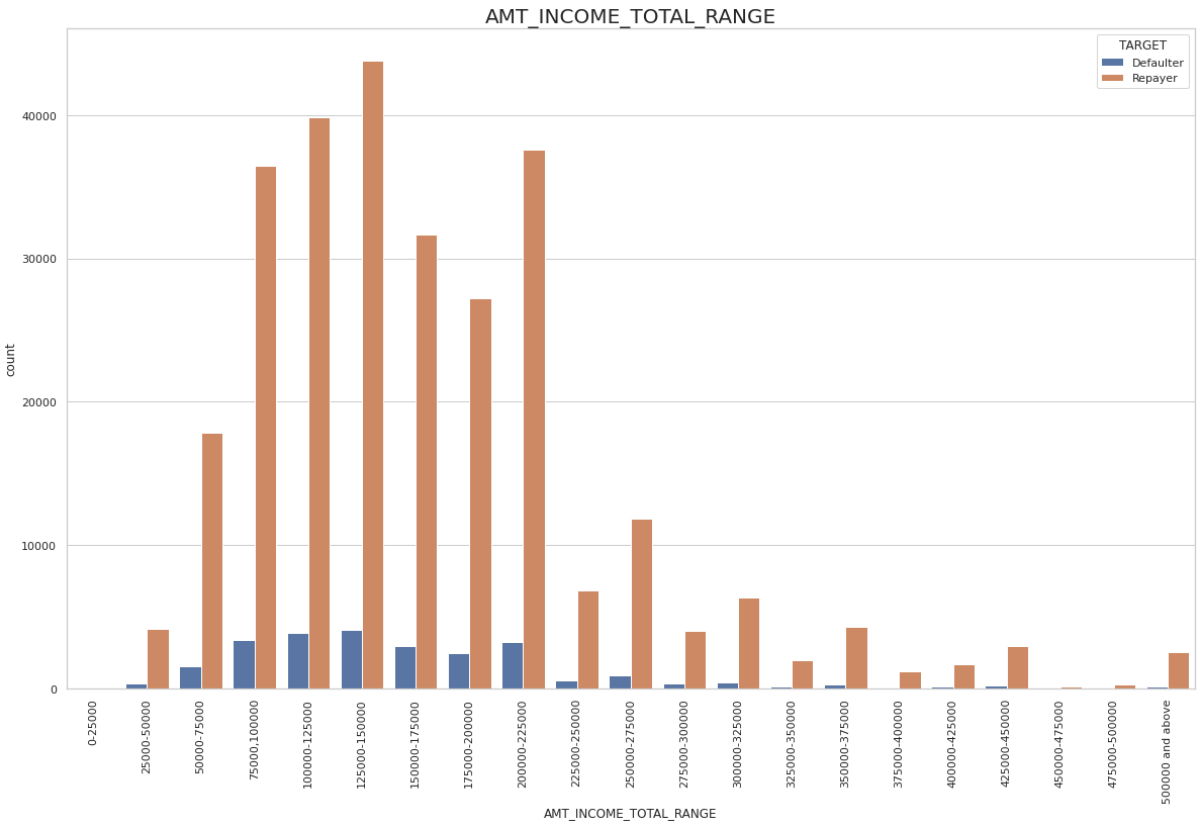
 # convert AMT_CREDIT to bins

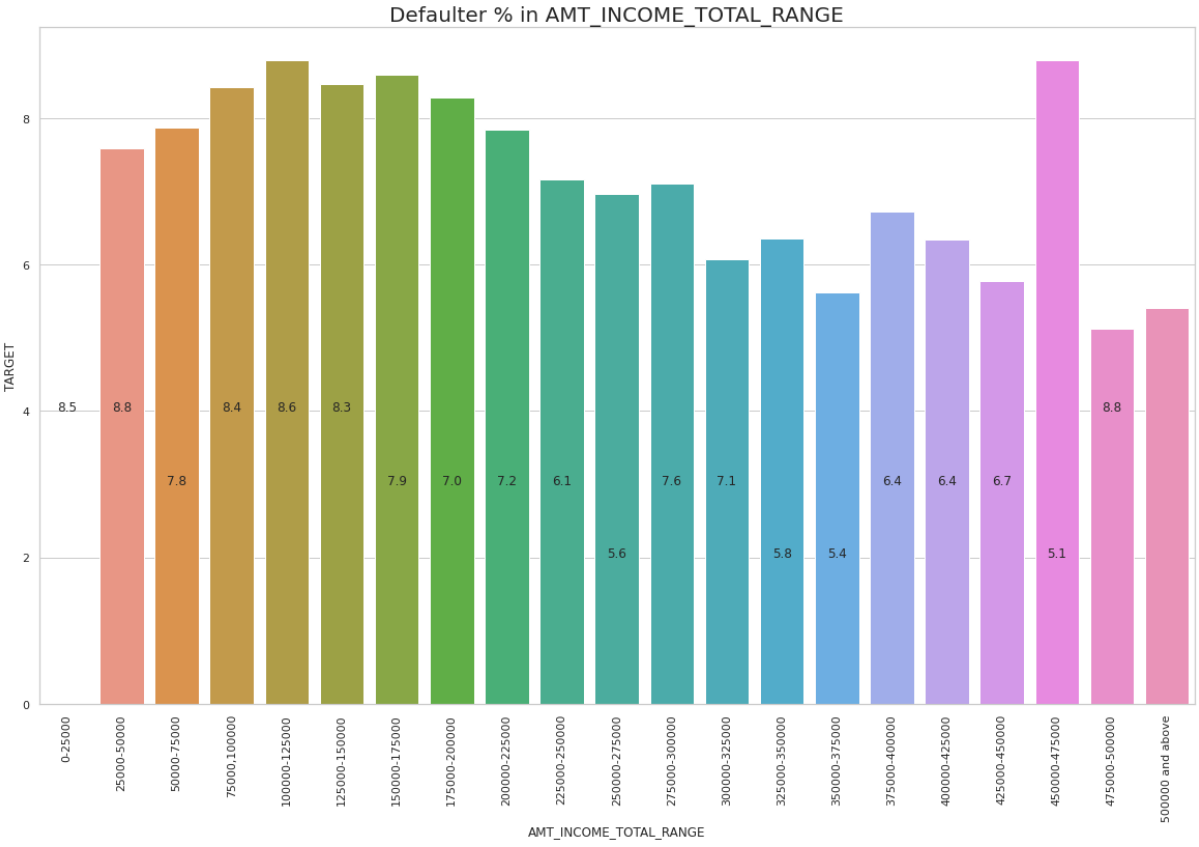
```
creditbins = [0,150000,200000,250000,300000,350000,400000,450000,500000,550000,600000,650000,700000,750000,800000,850000,900000,1000000000]
creditslots = ['0-150000', '150000-200000','200000-250000', '250000-300000', '300000-350000', '350000-400000','400000-450000','450000-500000','500000-550000','550000-600000','600000-650000','650000-700000','700000-750000','750000-800000','800000-850000','850000-900000','900000 and above']

df1['AMT_CREDIT_RANGE'] = pd.cut(df1.AMT_CREDIT,bins=creditbins,labels=creditslots)
```

 # univariate analysis for AMT_INCOME_TOTAL_RANGE

```
univariate(df1,'AMT_INCOME_TOTAL_RANGE','TARGET')
```

Himanshu Jain
himjain567@gmail.com





Inference:

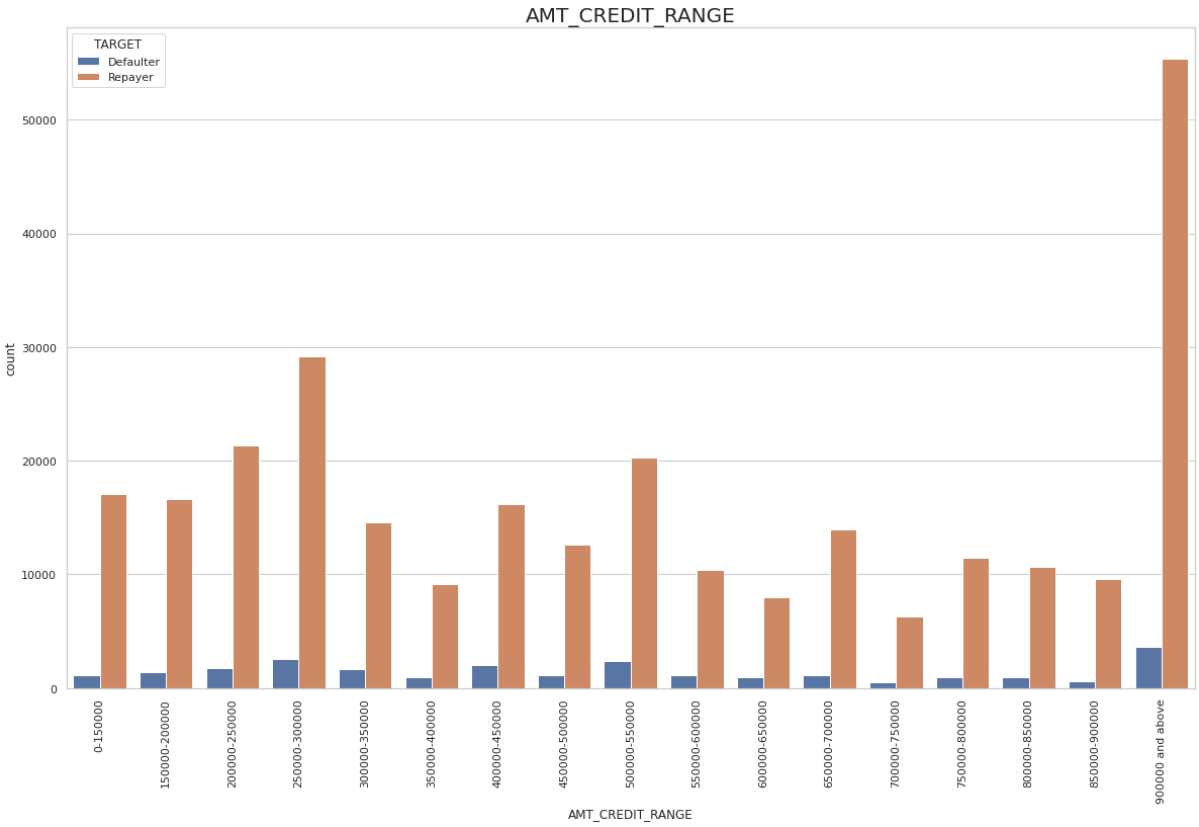
- Majority of the people who apply for the loan lies between 75000 to 225000.

Himanshu Jain
himjain567@gmail.com

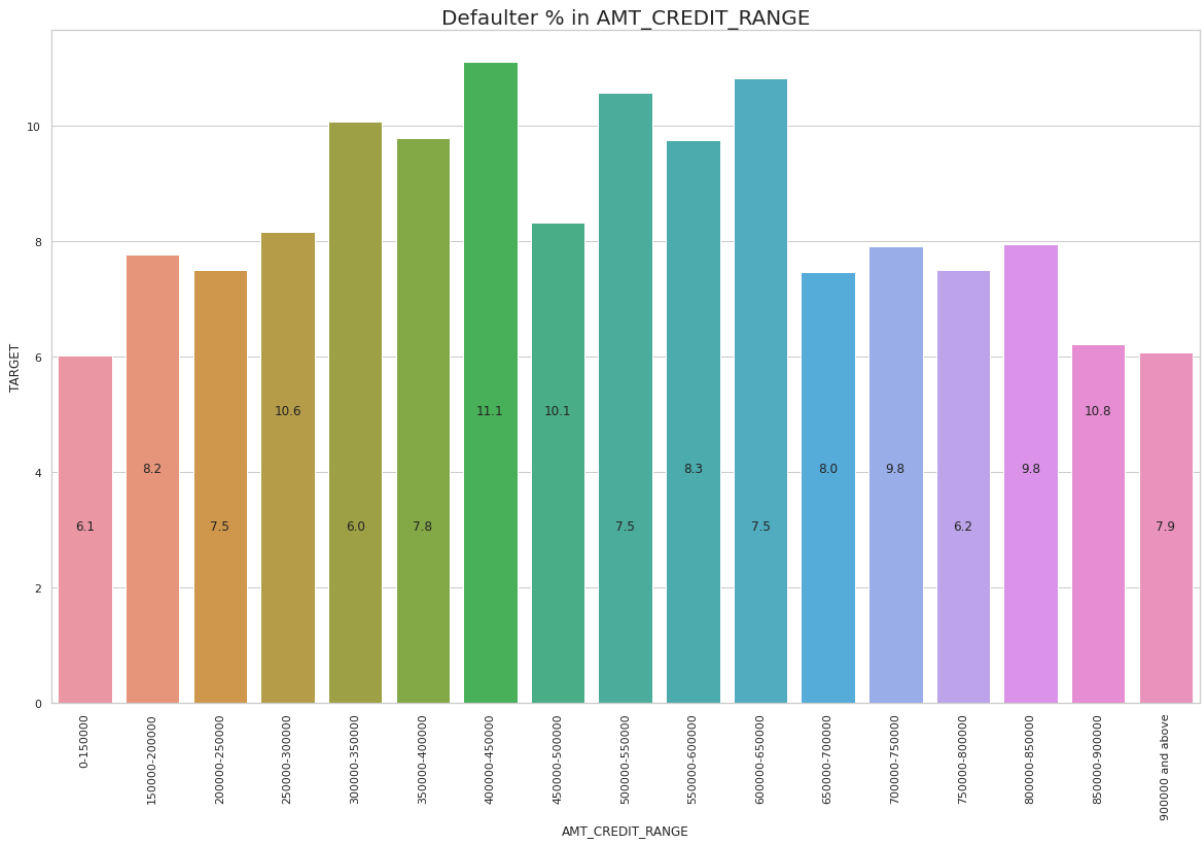
- people with low salary have higher default rate.
- exception to the above point salary from 475000 to 50000 have the higher default rate.

 # Univariate analysis for AMT_CREDIT

```
univariate(df1,'AMT_CREDIT_RANGE','TARGET')
```



Himanshu Jain
himjain567@gmail.com



Inference:

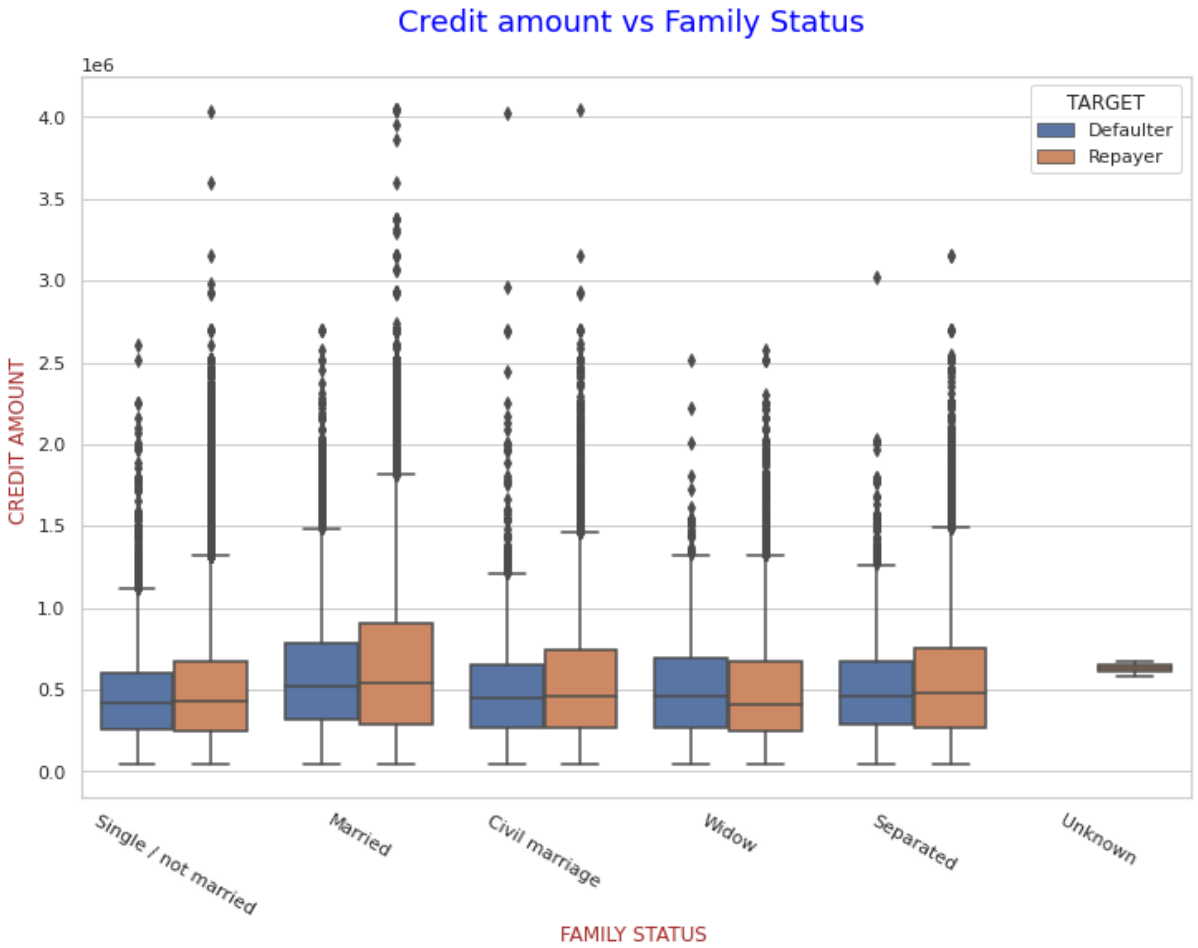
- Maximum loans are given in the range of 90000 and above.
- credit loan 30000 to 65000 have the higher default rate.

13) Bivariate analysis for application_data.csv

#Checking for Credit amount provided to the customers based on their Family type and plotted according TO target

```
plt.figure(figsize=(12,8))
scale_factor=5
sns.boxplot(data=df1,x=df1.NAME_FAMILY_STATUS,y=df1.AMT_CREDIT,hue=df1.TARGET)
plt.xticks(rotation=-30)
plt.xlabel("FAMILY STATUS ", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Brown'})
plt.ylabel("CREDIT AMOUNT ", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Brown'})
plt.title('Credit amount vs Family Status \n',fontdict={'fontsize': 18, 'fontweight' : 10, 'color' : 'Blue'})
plt.show()
```

Himanshu Jain
himjain567@gmail.com



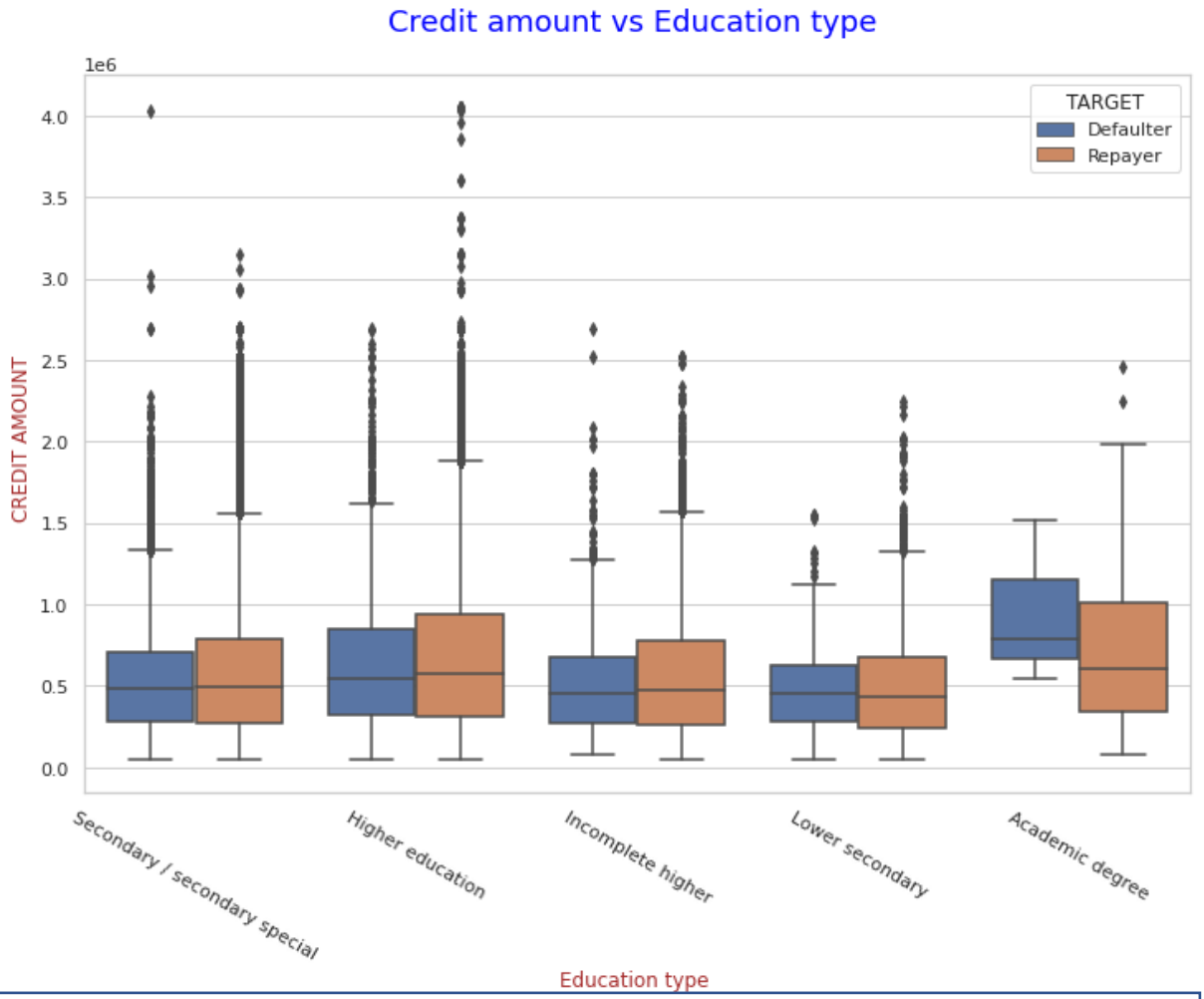
Inference:

- Civil Marriage has the highest defaulter credit amount.
- The credit amount for the repayers is high except in the case of civil marriage.

#Checking for Credit amount provided to the customers based on their Education type and plotted according TO target

```
plt.figure(figsize=(12,8))
scale_factor=5
sns.boxplot(data=df1,x=df1.NAME_EDUCATION_TYPE,y=df1.AMT_CREDIT,hue=df1.TARGET)
plt.xticks(rotation=-30)
plt.xlabel("Education type ", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Brown'})
plt.ylabel("CREDIT AMOUNT ", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Brown'})
plt.title('Credit amount vs Education type \n',fontdict={'fontsize': 18, 'fontweight' : 10, 'color' : 'Blue'})
plt.show()
```

Himanshu Jain
himjain567@gmail.com



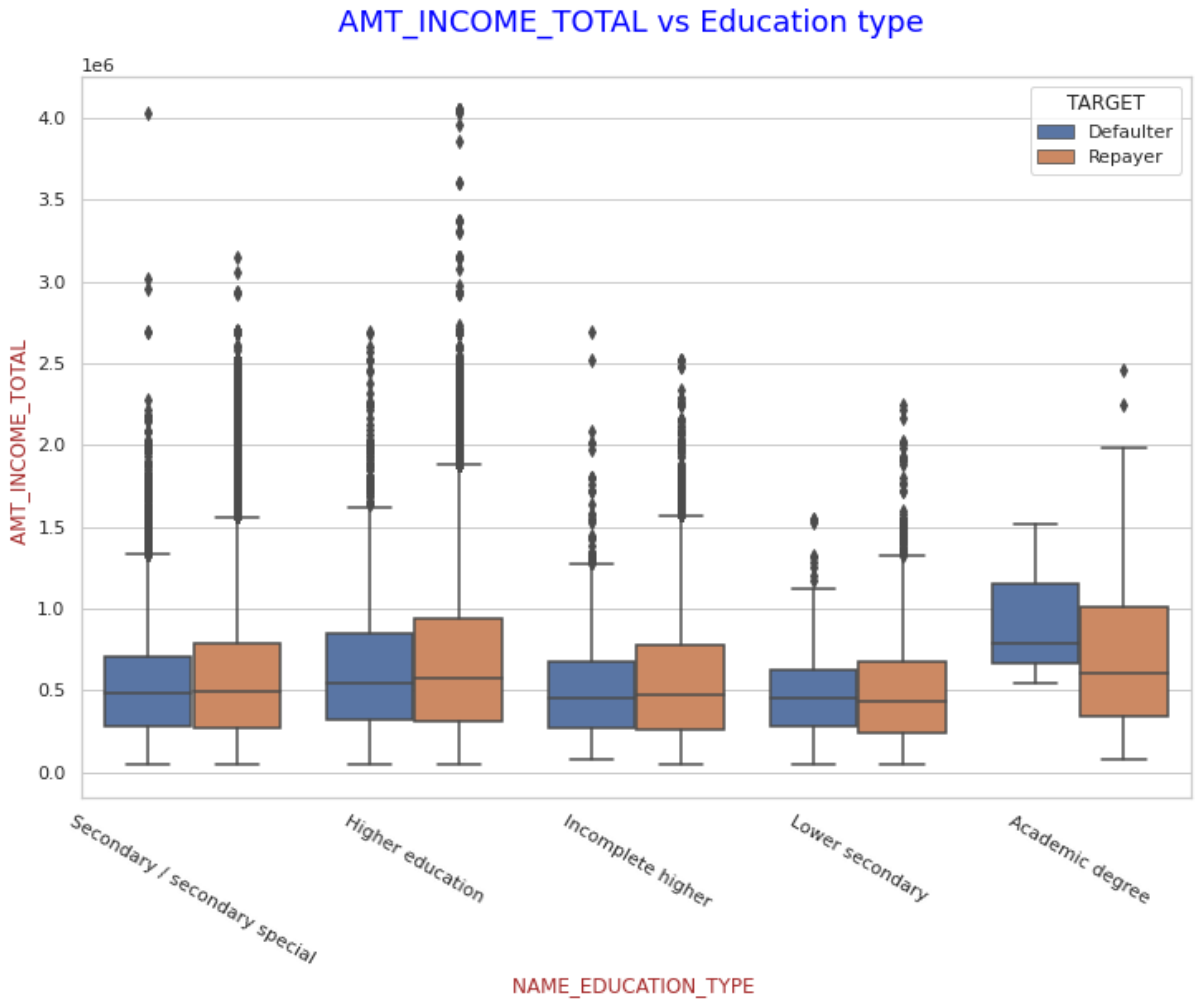
Inference:

- Higher education people are high credit seekers.
- Secondary / secondary special has high defaulter credit amount.

#Checking for AMT_INCOME_TOTAL provided to the customers based on their education type and plotted according to target

```
plt.figure(figsize=(12,8))
scale_factor=5
sns.boxplot(data=df1,x=df1.NAME_EDUCATION_TYPE,y=df1.AMT_CREDIT,hue=df1.TARGET)
plt.xticks(rotation=-30)
plt.xlabel("NAME_EDUCATION_TYPE ", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Brown'})
plt.ylabel("AMT_INCOME_TOTAL ", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Brown'})
plt.title('AMT_INCOME_TOTAL vs Education type \n',fontdict={'fontsize': 18, 'fontweight' : 10, 'color' : 'Blue'})
plt.show()
```

Himanshu Jain
himjain567@gmail.com



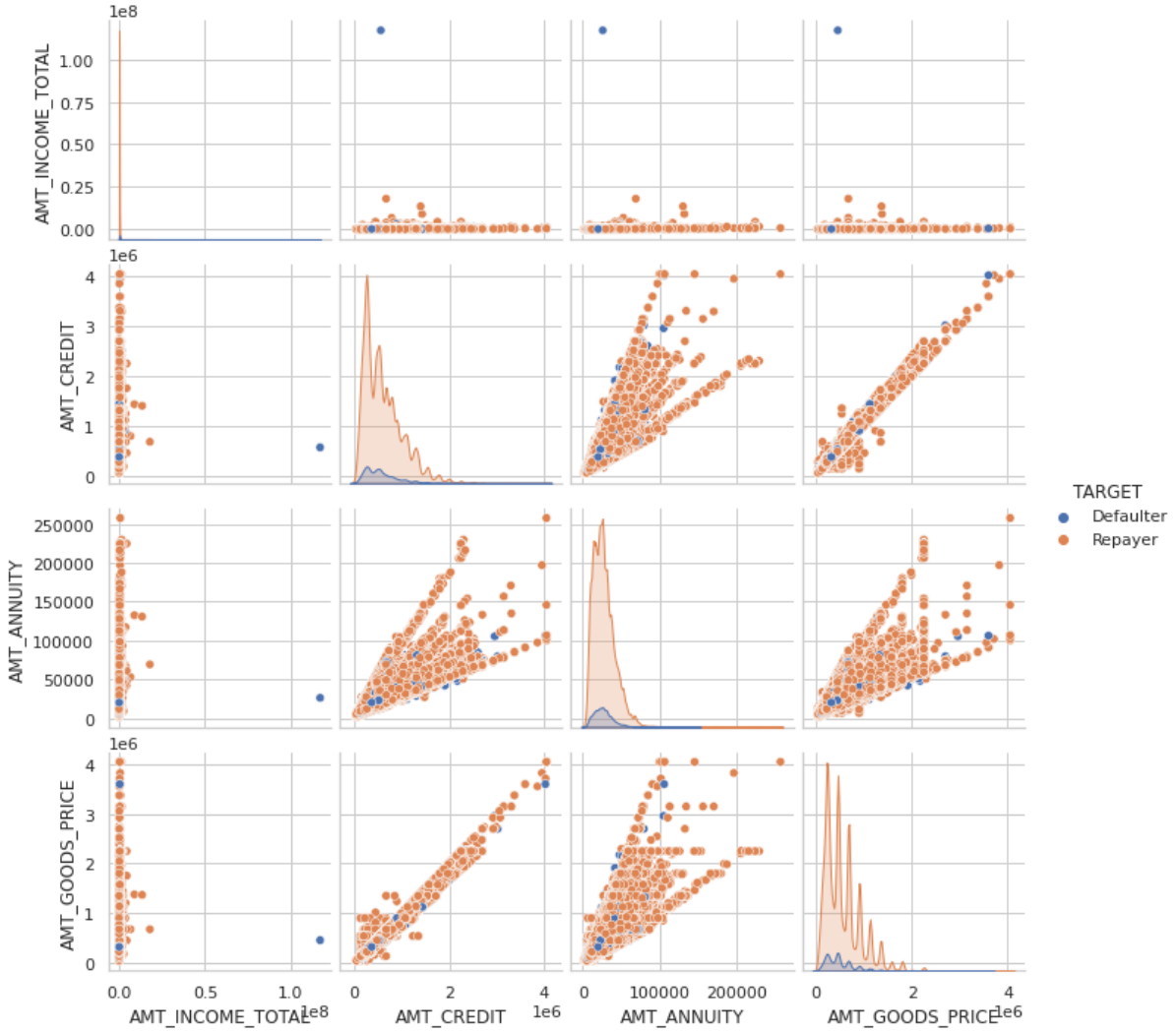
Inference:

- higher education is earning more than any education type.
- secondary / secondary special has the higher defaulter percentage.

 # Pair plots for the numeric column

```
amount = df1[['AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','AMT_GOODS_PRICE','TARGET']]
sns.pairplot(amount,hue = 'TARGET')
```

Himanshu Jain
himjain567@gmail.com



Inference:

- When Annuity Amount > 15K and Good Price Amount > 20 Lakhs, there is a lesser chance of defaulters
- Loan Amount (AMT_CREDIT) and Goods price(AMT_GOODS_PRICE) are highly correlated. It forms a linear relation.
- There are very less defaulters for AMT_CREDIT >20 Lakhs

14) Correlation check for the defaulter in application_data.csv

 # Divide the data based on the target column

```
defaulter_df = df1[df1['TARGET']==1]
```

 # Correlation check

```
corr2 = defaulter_df[['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',
```

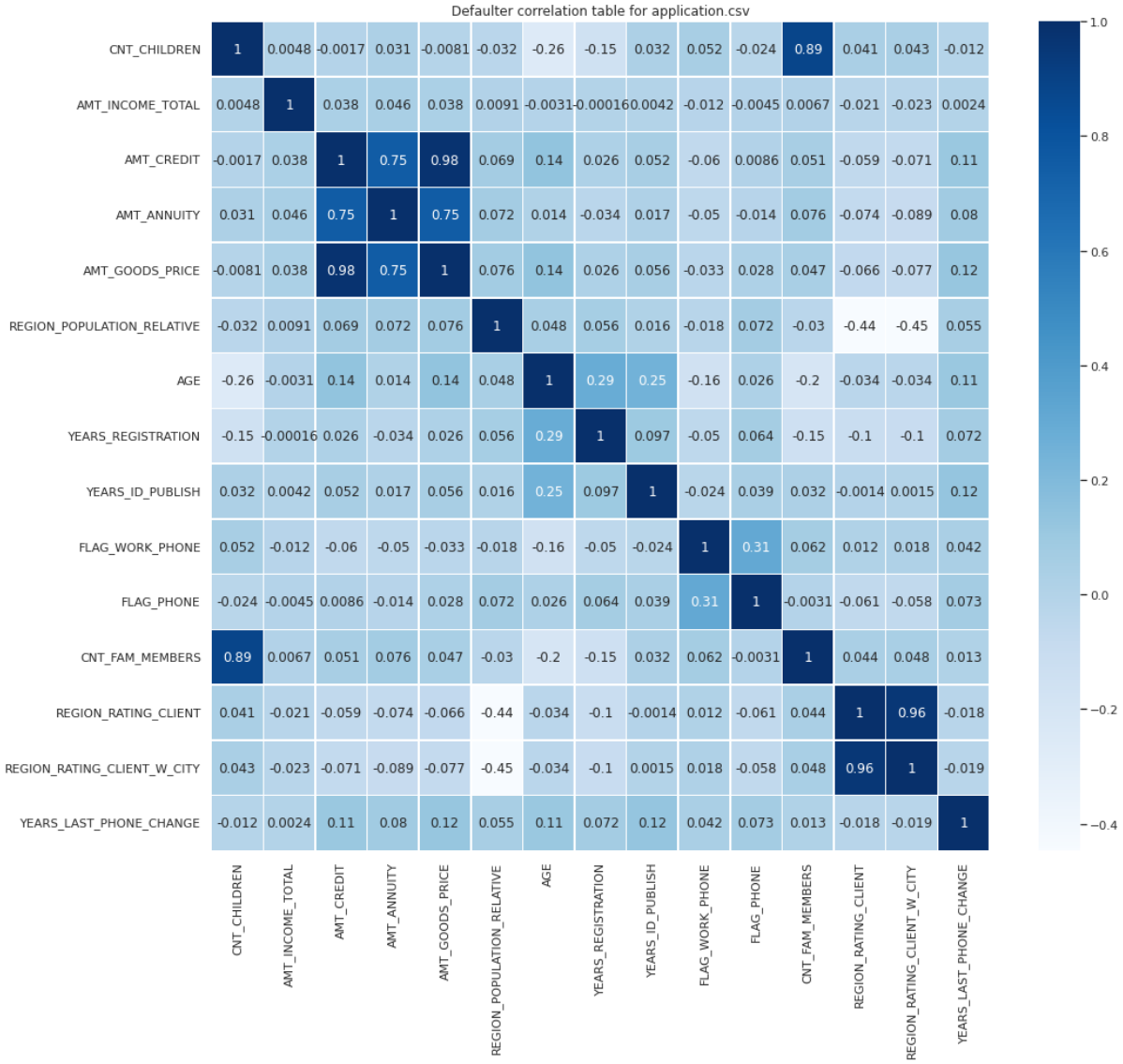
Himanshu Jain
himjain567@gmail.com

```
    'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'AGE',
    'YEARS_REGISTRATION', 'YEARS_ID_PUBLISH', 'FLAG_WORK_PHONE',
    'FLAG_PHONE', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS',
    'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY','ORGANIZATION_TYPE',
'YEARS_LAST_PHONE_CHANGE']].corr()

plt.figure(figsize=(16,14))
sns.heatmap(corr2,cmap='Blues',annot=True,linewidth=0.5)
plt.title('Defaulter correlation table for application.csv')
corr2 = corr2.where(np.triu(np.ones(corr2.shape), k=1).astype(bool))
corr_mat = corr2.unstack().sort_values(ascending=False).reset_index()
corr_mat.rename(columns={'level_0':'Var1','level_1':'Var2',0:'Correlation'},inplace=True )
print(corr_mat.head(10))
```

Top 10 correlation

	Var1	Var2	Correlation
0	AMT_GOODS_PRICE	AMT_CREDIT	0.982566
1	REGION_RATING_CLIENT_W_CITY	REGION_RATING_CLIENT	0.956637
2	CNT_FAM_MEMBERS	CNT_CHILDREN	0.885484
3	AMT_ANNUITY	AMT_CREDIT	0.752195
4	AMT_GOODS_PRICE	AMT_ANNUITY	0.752022
5	FLAG_PHONE	FLAG_WORK_PHONE	0.311035
6	YEARS_REGISTRATION	AGE	0.289114
7	YEARS_ID_PUBLISH	AGE	0.252863
8	AGE	AMT_GOODS_PRICE	0.135754
9	AGE	AMT_CREDIT	0.135316

Himanshu Jain
himjain567@gmail.com



15) Univariate analysis of merged data frame

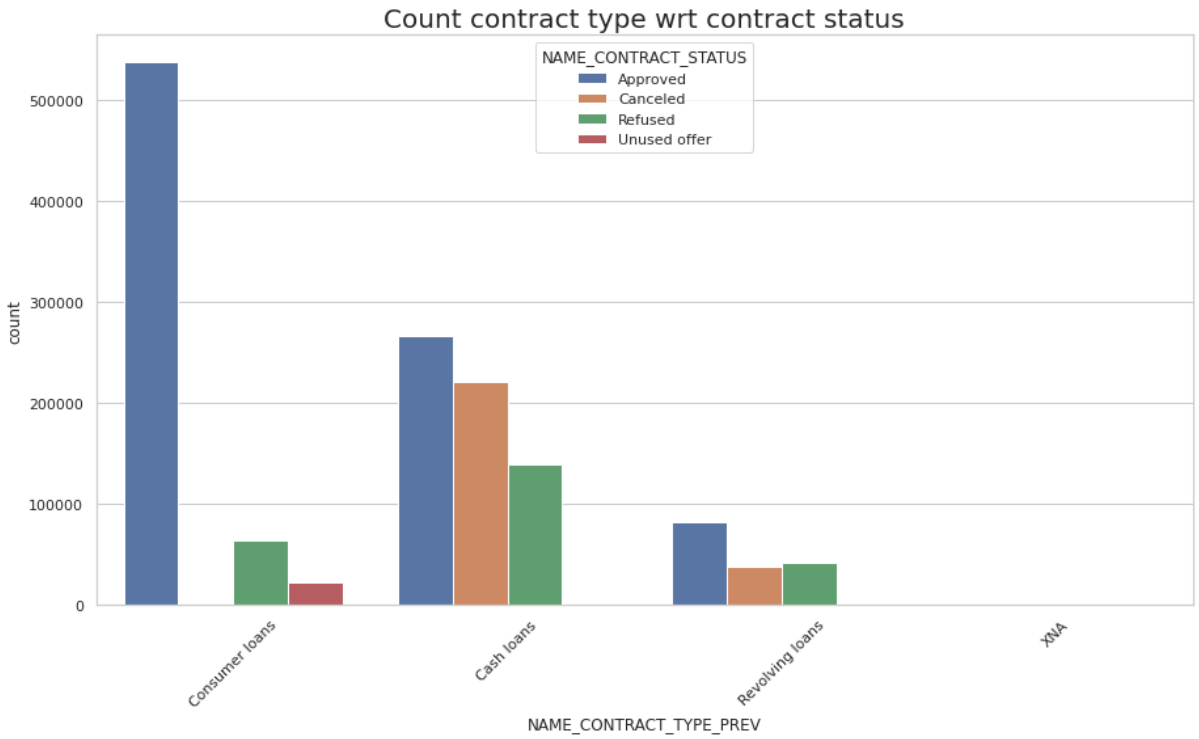
Merge the data set on SK_ID_CURR

```
mergedf = df1.merge(df, on='SK_ID_CURR')
```

Renaming the columns as convention

```
mergedf = mergedf.rename({'NAME_CONTRACT_TYPE_x' : 'NAME_CONTRACT_TYPE','AMT_CREDIT_x':'AMT_CREDIT','AMT_ANNUITY_x':'AMT_ANNUITY',
                'WEEKDAY_APPR_PROCESS_START_x' : 'WEEKDAY_APPR_PROCESS_START','AMT_GOODS_PRICE_x':'AMT_GOODS_PRICE',
                'HOUR_APPR_PROCESS_START_x':'HOUR_APPR_PROCESS_START','NAME_CONTRACT_TYPE_y':'NAME_CONTRACT_TYPE_PREV',
                'AMT_CREDIT_y':'AMT_CREDIT_PREV','AMT_ANNUITY_y':'AMT_ANNUITY_PREV','AMT_GOODS_PRICE_y':'AMT_GOODS_PRICE_PREV',
                'WEEKDAY_APPR_PROCESS_START_y':'WEEKDAY_APPR_PROCESS_START_PREV',
```

```
                    'HOUR_APPR_PROCESS_START_y':'HOUR_APPR_PROCESS_START_PREV'}, axis
=1)


plt.figure(figsize = (15,8))
sns.countplot(x = mergedf['NAME_CONTRACT_TYPE_PREV'], hue = mergedf['NAME_CONTRACT_
STATUS'])
plt.xticks(rotation = 45)
plt.title('Count contract type wrt contract status',fontdict={'fontsize': 20})
```



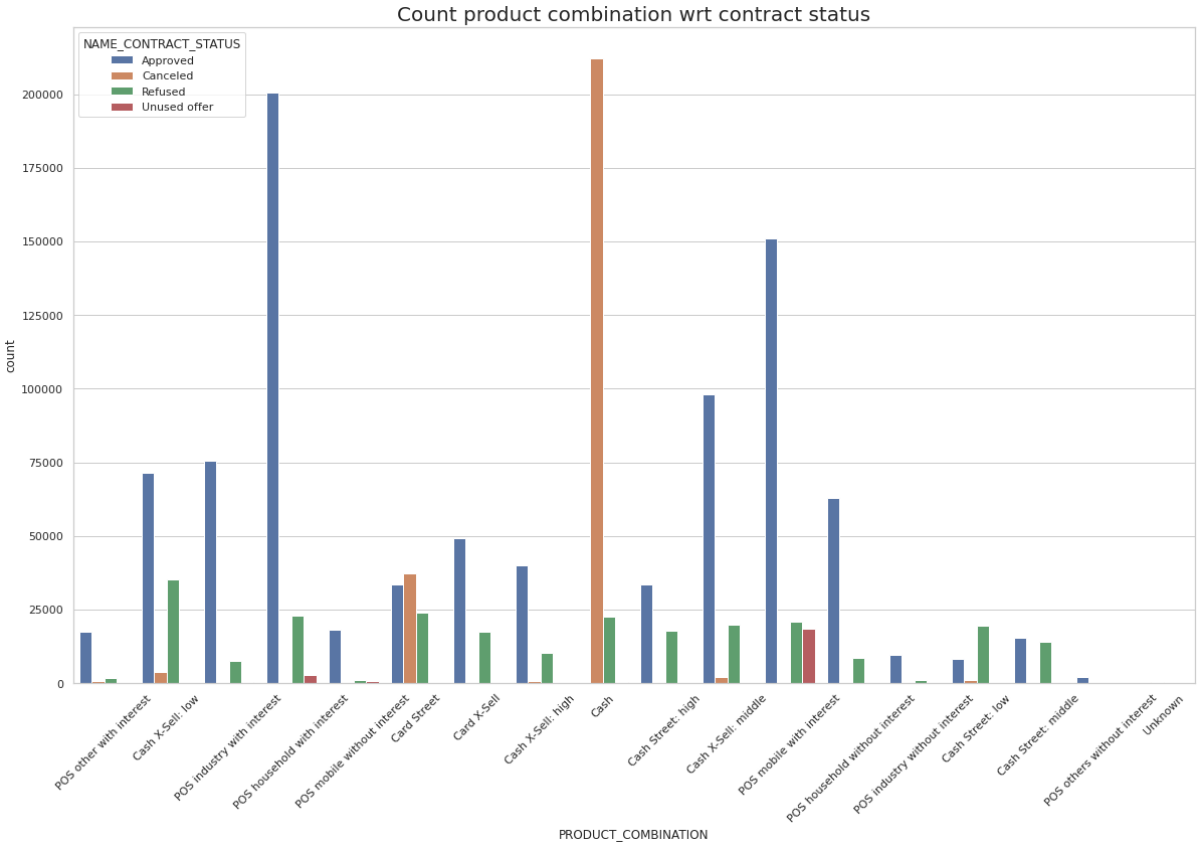
Inference:

- Consumer loans are highly applied and highly approved.
- There are only few cancelled loans in consumer loans.
- Revolving loans are less applied.
- More than 70% of cash loans are cancelled.

 # univariate analysis of PRODUCT_COMBINATION with hue NAME_CONTRACT_STATUS

```
plt.figure(figsize = (20,12))
sns.countplot(mergedf['PRODUCT_COMBINATION'],hue=mergedf['NAME_CONTRACT_STATUS'])
plt.xticks(rotation = 45)
plt.title('Count product combination wrt contract status',fontdict={'fontsize': 20})
```

Himanshu Jain
himjain567@gmail.com



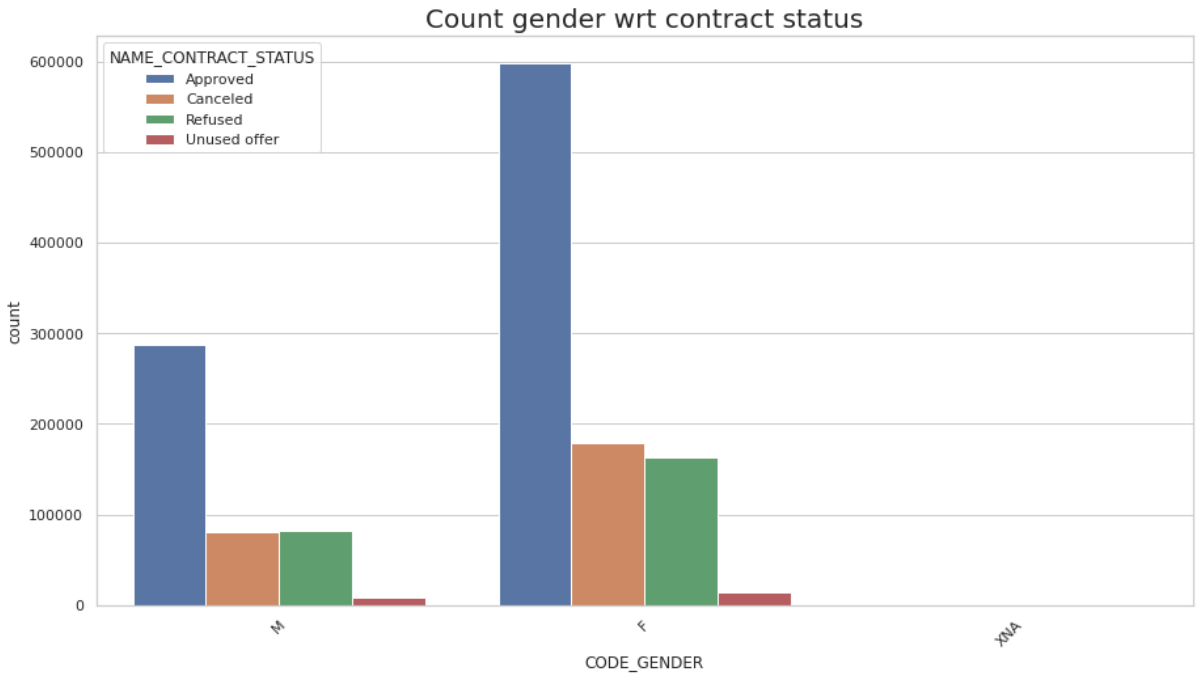
Inference:

- POS industry with interest has the highest approved loans.
- Cash loans has the highest cancelled loans.

 # Univariate analysis of CODE_GENDER with hue NAME_CONTRACT_STATUS

```
plt.figure(figsize = (15,8))
sns.countplot(mergedf['CODE_GENDER'],hue=mergedf['NAME_CONTRACT_STATUS'])
plt.xticks(rotation = 45)
plt.title('Count gender wrt contract status',fontdict={'fontsize': 20})
```

Himanshu Jain
himjain567@gmail.com



Inference:

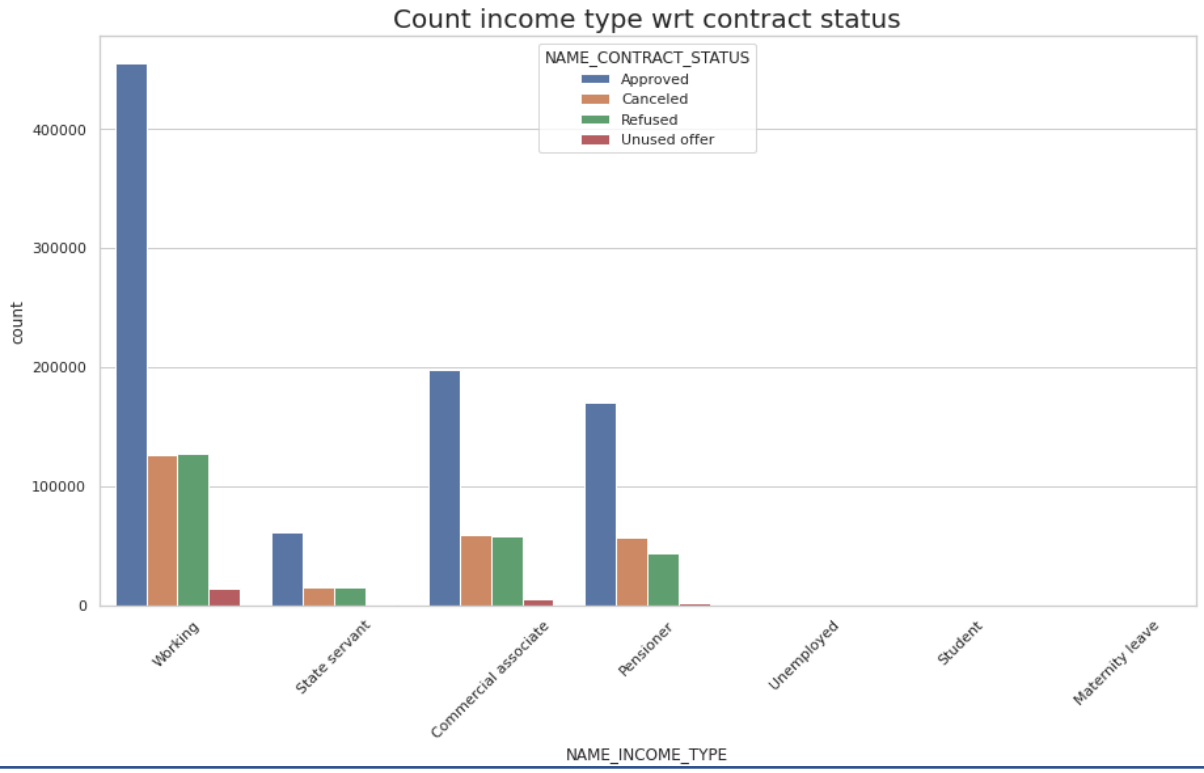
- Female loans are Approved more than male.
- Female loans have high unused offers than male.

 # Univariate analysis of NAME_INCOME_TYPE with hue

NAME_CONTRACT_STATUS

```
plt.figure(figsize = (15,8))
sns.countplot(mergedf['NAME_INCOME_TYPE'],hue=mergedf['NAME_CONTRACT_STATUS'])
plt.xticks(rotation = 45)
plt.title('Count income type wrt contract status',fontdict={'fontsize': 20})
```

Himanshu Jain
himjain567@gmail.com



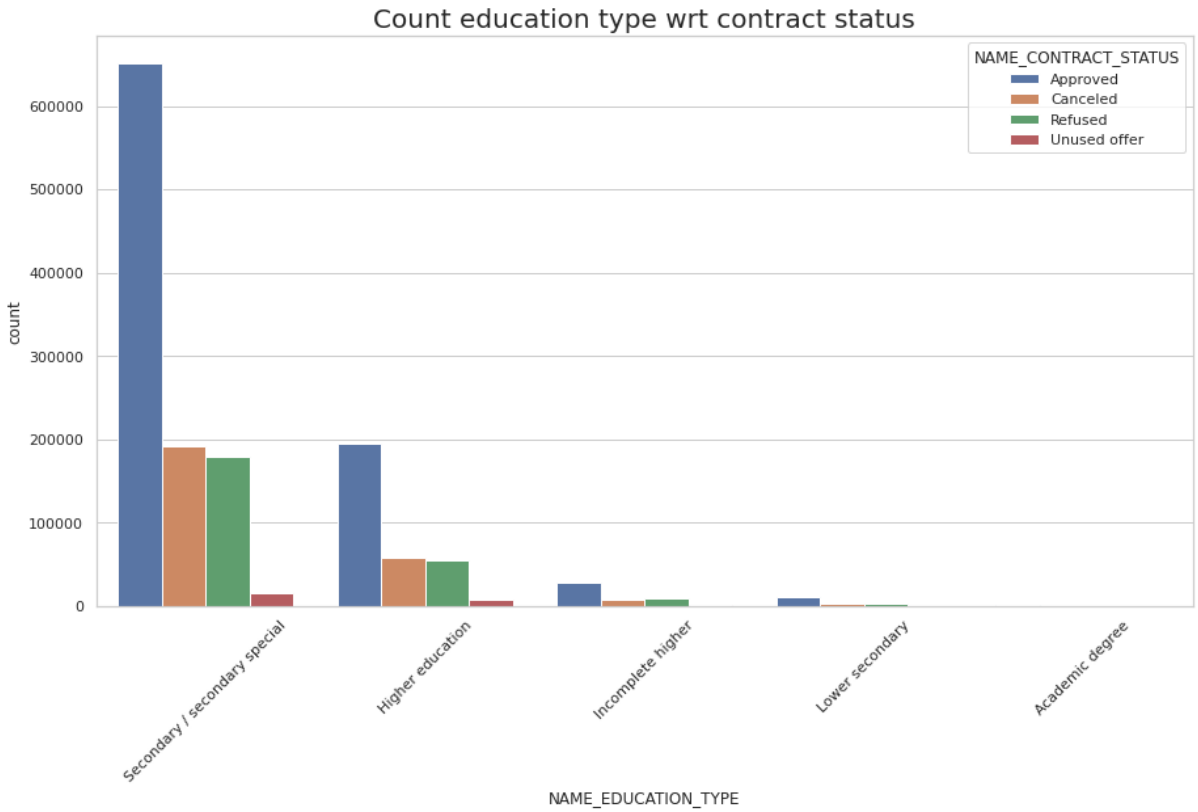
Inference:

- loans for working people are approved more.
- unemployed, student and maternity leave have very few records.

 # Univariate analysis of NAME_EDUCATION_TYPE with hue NAME_CONTRACT_STATUS

```
plt.figure(figsize = (15,8))
sns.countplot(mergedf['NAME_EDUCATION_TYPE'],hue=mergedf['NAME_CONTRACT_STATUS'])
plt.xticks(rotation = 45)
plt.title('Count education type wrt contract status',fontdict={'fontsize': 20})
```

Himanshu Jain
himjain567@gmail.com



Inference:

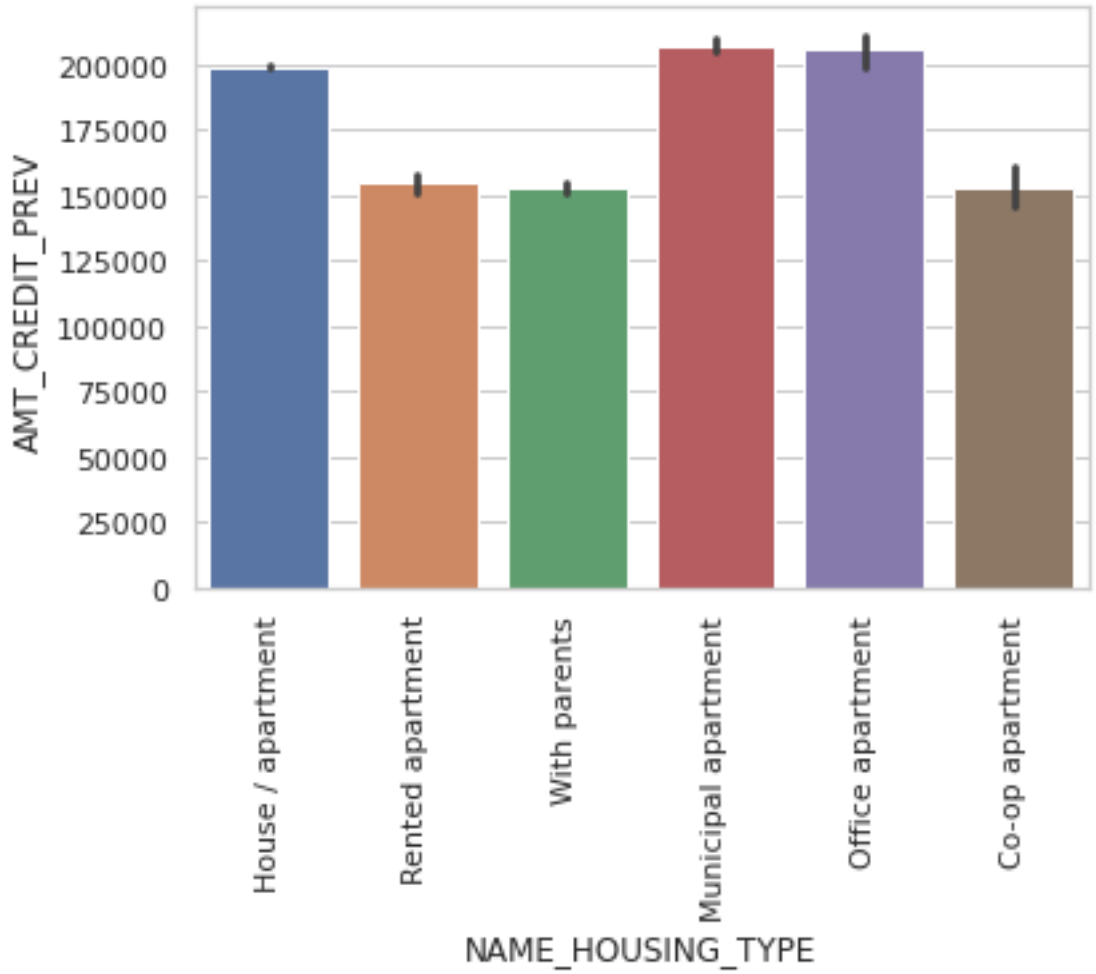
- secondary education has the highest loan approved.
- Academic degree has very few records

16) Bivariate analysis of merged data frame

 # Bar plot between NAME_HOUSING_TYPE and AMT_CREDIT_PREV

```
sns.barplot(data=mergedf,x = 'NAME_HOUSING_TYPE',y = 'AMT_CREDIT_PREV' )
plt.xticks(rotation=90)
```

Himanshu Jain
himjain567@gmail.com



Inference:

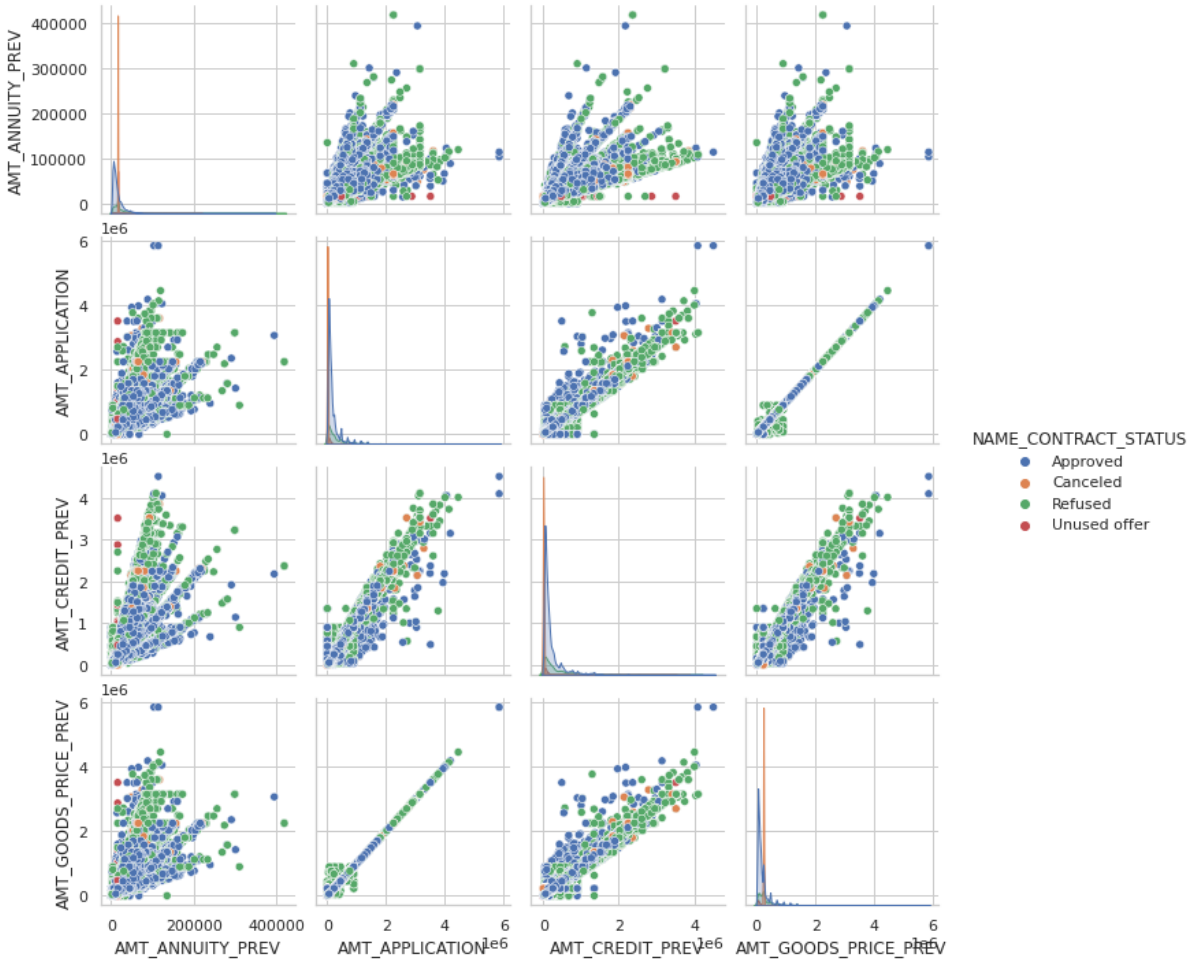
- Municipal apartment and credit apartment have the highest amount credit.

17) Correlation check for previous_application.csv

 # Pair plots for the numeric column

```
plt.figure(figsize=(20,14))
amount = mergedf[['AMT_ANNUITY_PREV', 'AMT_APPLICATION',
       'AMT_CREDIT_PREV', 'AMT_GOODS_PRICE_PREV','NAME_CONTRACT_STATUS']]
sns.pairplot(amount,hue = 'NAME_CONTRACT_STATUS')
```

Himanshu Jain
himjain567@gmail.com



Inference:

- AMT_GOODS_PRICE_PREV and AMT_APPLICATION shows too high linear relation.

 # Extracting columns for the correlation check

```
corr4 = mergedf[['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',
    'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'AGE',
    'YEARS_REGISTRATION', 'YEARS_ID_PUBLISH', 'FLAG_WORK_PHONE',
    'FLAG_PHONE', '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_CITY',
    'ORGANIZATION_TYPE', 'OBS_30_CNT_SOCIAL_CIRCLE',
    'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE',
    'DEF_60_CNT_SOCIAL_CIRCLE', 'YEARS_LAST_PHONE_CHANGE',
    'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
    'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
```

Himanshu Jain
himjain567@gmail.com

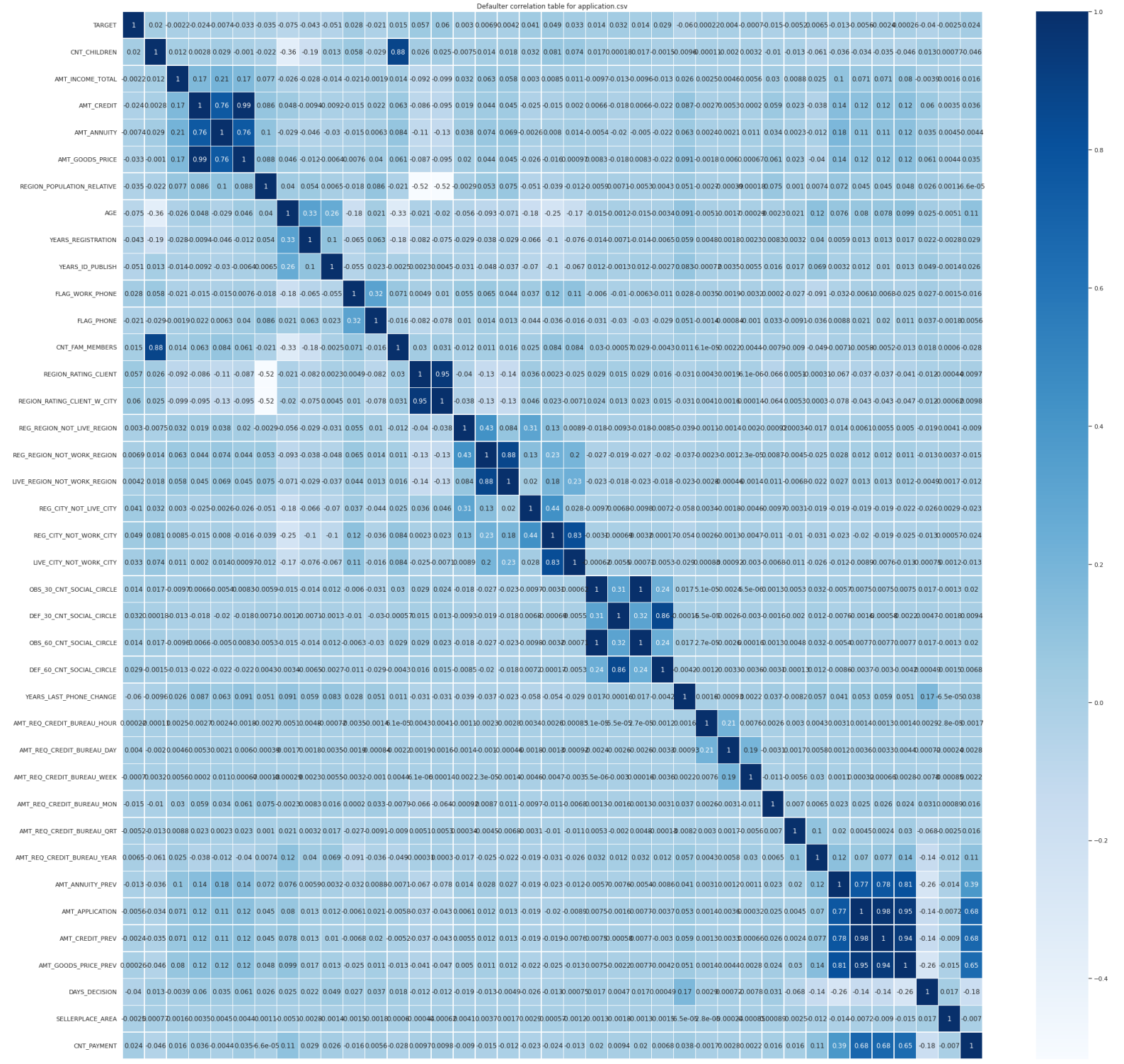
```
    'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR',
    'AMT_INCOME_TOTAL_RANGE', 'AMT_CREDIT_RANGE',
    'NAME_CONTRACT_TYPE_PREV', 'AMT_ANNUITY_PREV', 'AMT_APPLICATION',
    'AMT_CREDIT_PREV', 'AMT_GOODS_PRICE_PREV', 'NAME_CASH_LOAN_PURPOSE',
    'NAME_CONTRACT_STATUS', 'DAYS_DECISION', 'NAME_PAYMENT_TYPE',
    'CODE_REJECT_REASON', 'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY',
    'NAME_PORTFOLIO', 'NAME_PRODUCT_TYPE', 'CHANNEL_TYPE',
    'SELLERPLACE_AREA', 'NAME_SELLER_INDUSTRY', 'CNT_PAYMENT',
    'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION']].corr()
```

 # Plot correlation map

```
plt.figure(figsize=(35,35))
sns.heatmap(corr4,cmap='Blues',annot=True,linewidth=0.5)
plt.title('Defaulter correlation table for application.csv')
corr5 = corr4.where(np.triu(np.ones(corr4.shape), k=1).astype(bool))
corr_mat = corr5.unstack().sort_values(ascending=False).reset_index()
corr_mat.rename(columns={'level_0':'Var1','level_1':'Var2',0:'Correlation'},inplace=True )
print(corr_mat.head(10))
```

```
                Var1                    Var2        Correlation
0    OBS_60_CNT_SOCIAL_CIRCLE    OBS_30_CNT_SOCIAL_CIRCLE     0.998561
1          AMT_GOODS_PRICE              AMT_CREDIT     0.985959
2          AMT_CREDIT_PREV          AMT_APPLICATION     0.975683
3      AMT_GOODS_PRICE_PREV          AMT_APPLICATION     0.945759
4 REGION_RATING_CLIENT_W_CITY       REGION_RATING_CLIENT     0.945596
5      AMT_GOODS_PRICE_PREV          AMT_CREDIT_PREV     0.939147
6          CNT_FAM_MEMBERS             CNT_CHILDREN     0.879224
7 LIVE_REGION_NOT_WORK_REGION  REG_REGION_NOT_WORK_REGION     0.875505
8    DEF_60_CNT_SOCIAL_CIRCLE    DEF_30_CNT_SOCIAL_CIRCLE     0.862698
9     LIVE_CITY_NOT_WORK_CITY      REG_CITY_NOT_WORK_CITY     0.831422
```

Himanshu Jain
himjain567@gmail.com



Himanshu Jain
himjain567@gmail.com

Conclusion:

• The data is highly imbalance with almost 92% repayor and 8% defaulter.
• It is found that revolving loans has less defaulter (5.5%) than cash loans (8.2%).
• Unemployed and Maternity leave, income types have the highest defaulter 36.5% and 40% respectively. Loans giving to this category should be avoided.
• Academic degree in education type has the minimum defaulter rate. Loan distribution to this category can be increased.
• Person who owns car have the less defaulter rate than who does not own car.
• Banks should focus more on contract type ‘Student’, ‘pensioner’ and ‘Businessman’ with housing ‘type other than ‘Co-op apartment’ and 'office apartment' for successful payments.
• OCCUPATION_TYPE: Avoid Low-skill Laborers, Drivers and Waiters/barmen staff, Security staff, Laborers and Cooking staff as their default rate is huge.
• AMT_GOODS_PRICE: When the credit amount goes beyond 3lakhs, there is an increase in defaulters.

Challenges:

- The Dataset contains 122 features in application_data.csv and 32 features in previous_application.csv which makes it complicated to find the relation between the dataset.
- The processing time while making pair plots and correlation plots is quite high because of huge dataset.