Introduction

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

Business Understanding

The loan providing companies find it hard to give loans to the people due to their insufficient or non-existent credit history. Because of that, some consumers use it as their advantage by becoming a defaulter. Suppose you work for a consumer finance company which specialises in lending various types of loans to urban customers. You have to use EDA to analyse the patterns present in the data. This will ensure that the applicants capable of repaying the loan are not rejected.

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

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

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

- The client with payment difficulties: he/she had late payment more than X days on at least one of the first Y instalments of the loan in our sample,
- All other cases: All other cases when the payment is paid on time.

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

- **Approved:** The Company has approved loan Application
- **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.
- **Refused:** The company had rejected the loan (because the client does not meet their requirements etc.).
- **Unused offer:** Loan has been cancelled by the client but on different stages of the process.

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

Business Objectives

This case study aims to identify patterns which indicate if a client has difficulty paying their installments which may be used for taking actions such as denying the loan, reducing the amount of loan, lending (to risky applicants) at a higher interest rate, etc. This will ensure that the consumers capable of repaying the loan are not rejected. Identification of such applicants using EDA is the aim of this case study.

In other words, the company wants to understand the driving factors (or driver variables) behind loan default, i.e. the variables which are strong indicators of default. The company can utilise this knowledge for its portfolio and risk assessment.

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

Data Understanding

This dataset has 3 files as explained below:

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

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

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

Results Expected by Learners

- **Present the overall approach of the analysis in a presentation. 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: *How will you analyse the data in case of data imbalance? You can plot more than one type of plot to analyse the different aspects due to data imbalance. For example, you can choose your own scale for the graphs, i.e. one can plot in terms of percentage or absolute value. Do this analysis for the ‘Target variable’ in the dataset (clients with payment difficulties and all other cases). Use a mix of univariate and bivariate analysis etc.*

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 visualisations and summarise 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.**

You need to submit one/two Ipython notebook which clearly explains the thought process behind your analysis (either in comments of markdown text), code and relevant plots. The presentation file needs to be in PDF format and should contain the points discussed above with the necessary visualisations. Also, all the visualisations and plots must be done in Python(should be present in the Ipython notebook), though they may be recreated in Tableau for better aesthetics in the PPT file.

IMPORTING ALL THE NECCESSARY MODULES

In [1]:

```
#importing all the important libraries like numpy. pandas, matlplolib, and warnings t

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

In [2]:

```
# to suppress warnings

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

In [3]:

```
#notebook setting to display all the rowns and columns to have better clearity on the

pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
pd.set_option('display.expand_frame_repr', False)
```

Dataset 1"application_data.csv"

reading and understanding the data

Importing the dataset

In [ ]:

```
appl_data=pd.read_csv(r"C:\Users\ARCHANA\application_data.csv")
```

Understanding the dataset

In [5]:

```
#checking the rows and columns of the raw dataset

appl_data.shape
```

Out[5]:

```
(307511, 122)
```

In [6]:

```
#Checking information of all the columns like data types
appl_data.info("all")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 122 columns):
 #   Column                        Dtype
---  ------                        -----
 0   SK_ID_CURR                    int64
 1   TARGET                        int64
 2   NAME_CONTRACT_TYPE            object
 3   CODE_GENDER                   object
 4   FLAG_OWN_CAR                  object
 5   FLAG_OWN_REALTY               object
 6   CNT_CHILDREN                  int64
 7   AMT_INCOME_TOTAL              float64
 8   AMT_CREDIT                    float64
 9   AMT_ANNUITY                   float64
 10  AMT_GOODS_PRICE               float64
 11  NAME_TYPE_SUITE               object
 12  NAME_INCOME_TYPE              object
 13  NAME_EDUCATION_TYPE           object
 14  NAME_FAMILY_STATUS            object
 15  NAME_HOUSING_TYPE             object
 16  REGION_POPULATION_RELATIVE    float64
 17  DAYS_BIRTH                    int64
 18  DAYS_EMPLOYED                 int64
 19  DAYS_REGISTRATION             float64
 20  DAYS_ID_PUBLISH               int64
 21  OWN_CAR_AGE                   float64
 22  FLAG_MOBIL                    int64
 23  FLAG_EMP_PHONE                int64
 24  FLAG_WORK_PHONE               int64
 25  FLAG_CONT_MOBILE              int64
 26  FLAG_PHONE                    int64
 27  FLAG_EMAIL                    int64
 28  OCCUPATION_TYPE               object
 29  CNT_FAM_MEMBERS               float64
 30  REGION_RATING_CLIENT          int64
 31  REGION_RATING_CLIENT_W_CITY   int64
 32  WEEKDAY_APPR_PROCESS_START    object
 33  HOUR_APPR_PROCESS_START       int64
 34  REG_REGION_NOT_LIVE_REGION    int64
 35  REG_REGION_NOT_WORK_REGION    int64
 36  LIVE_REGION_NOT_WORK_REGION   int64
```

```
 37  REG_CITY_NOT_LIVE_CITY        int64
 38  REG_CITY_NOT_WORK_CITY        int64
 39  LIVE_CITY_NOT_WORK_CITY       int64
 40  ORGANIZATION_TYPE             object
 41  EXT_SOURCE_1                  float64
 42  EXT_SOURCE_2                  float64
 43  EXT_SOURCE_3                  float64
 44  APARTMENTS_AVG                float64
 45  BASEMENTAREA_AVG              float64
 46  YEARS_BEGINEXPLUATATION_AVG   float64
 47  YEARS_BUILD_AVG               float64
 48  COMMONAREA_AVG                float64
 49  ELEVATORS_AVG                 float64
 50  ENTRANCES_AVG                 float64
 51  FLOORSMAX_AVG                 float64
 52  FLOORSMIN_AVG                 float64
 53  LANDAREA_AVG                  float64
 54  LIVINGAPARTMENTS_AVG          float64
 55  LIVINGAREA_AVG                float64
 56  NONLIVINGAPARTMENTS_AVG       float64
 57  NONLIVINGAREA_AVG             float64
 58  APARTMENTS_MODE               float64
 59  BASEMENTAREA_MODE             float64
 60  YEARS_BEGINEXPLUATATION_MODE  float64
 61  YEARS_BUILD_MODE              float64
 62  COMMONAREA_MODE               float64
 63  ELEVATORS_MODE                float64
 64  ENTRANCES_MODE                float64
 65  FLOORSMAX_MODE                float64
 66  FLOORSMIN_MODE                float64
 67  LANDAREA_MODE                 float64
 68  LIVINGAPARTMENTS_MODE         float64
 69  LIVINGAREA_MODE               float64
 70  NONLIVINGAPARTMENTS_MODE      float64
 71  NONLIVINGAREA_MODE            float64
 72  APARTMENTS_MEDI               float64
 73  BASEMENTAREA_MEDI             float64
 74  YEARS_BEGINEXPLUATATION_MEDI  float64
 75  YEARS_BUILD_MEDI              float64
 76  COMMONAREA_MEDI               float64
 77  ELEVATORS_MEDI                float64
 78  ENTRANCES_MEDI                float64
 79  FLOORSMAX_MEDI                float64
 80  FLOORSMIN_MEDI                float64
 81  LANDAREA_MEDI                 float64
 82  LIVINGAPARTMENTS_MEDI         float64
 83  LIVINGAREA_MEDI               float64
 84  NONLIVINGAPARTMENTS_MEDI      float64
 85  NONLIVINGAREA_MEDI            float64
 86  FONDKAPREMONT_MODE            object
 87  HOUSETYPE_MODE                object
 88  TOTALAREA_MODE                float64
 89  WALLSMATERIAL_MODE            object
 90  EMERGENCYSTATE_MODE           object
 91  OBS_30_CNT_SOCIAL_CIRCLE      float64
 92  DEF_30_CNT_SOCIAL_CIRCLE      float64
 93  OBS_60_CNT_SOCIAL_CIRCLE      float64
 94  DEF_60_CNT_SOCIAL_CIRCLE      float64
 95  DAYS_LAST_PHONE_CHANGE        float64
 96  FLAG_DOCUMENT_2               int64
 97  FLAG_DOCUMENT_3               int64
 98  FLAG_DOCUMENT_4               int64
 99  FLAG_DOCUMENT_5               int64
 100 FLAG_DOCUMENT_6               int64
```

```
 101 FLAG_DOCUMENT_7              int64
 102 FLAG_DOCUMENT_8              int64
 103 FLAG_DOCUMENT_9              int64
 104 FLAG_DOCUMENT_10             int64
 105 FLAG_DOCUMENT_11             int64
 106 FLAG_DOCUMENT_12             int64
 107 FLAG_DOCUMENT_13             int64
 108 FLAG_DOCUMENT_14             int64
 109 FLAG_DOCUMENT_15             int64
 110 FLAG_DOCUMENT_16             int64
 111 FLAG_DOCUMENT_17             int64
 112 FLAG_DOCUMENT_18             int64
 113 FLAG_DOCUMENT_19             int64
 114 FLAG_DOCUMENT_20             int64
 115 FLAG_DOCUMENT_21             int64
 116 AMT_REQ_CREDIT_BUREAU_HOUR   float64
 117 AMT_REQ_CREDIT_BUREAU_DAY    float64
 118 AMT_REQ_CREDIT_BUREAU_WEEK   float64
 119 AMT_REQ_CREDIT_BUREAU_MON    float64
 120 AMT_REQ_CREDIT_BUREAU_QRT    float64
 121 AMT_REQ_CREDIT_BUREAU_YEAR   float64
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

There are 122 columns having various data types like object, int, float and 305711 rows.

In [7]:

```
appl_data.head()
```

Out[7]:

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

In [8]:

```
# Checking the numeric variables of the dataframes
appl_data.describe()
```

Out[8]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	30749
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	2710
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	1449
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	161
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	1652
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	2490
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	3459
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	25802

- **there are 122 columns and 307511 rows.**

Data Cleaning

Null Values

In [9]:
```
#checking how many null values are present in each of the columns

#creating a function to find null values for the dataframe
def null_values(df):
    return round((df.isnull().sum()*100/len(df)).sort_values(ascending = False),2)
```

In [10]:
```
null_values(appl_data)
```

Out[10]:
```
COMMONAREA_MEDI                 69.87
COMMONAREA_AVG                  69.87
COMMONAREA_MODE                 69.87
NONLIVINGAPARTMENTS_MODE        69.43
NONLIVINGAPARTMENTS_MEDI        69.43
NONLIVINGAPARTMENTS_AVG         69.43
FONDKAPREMONT_MODE              68.39
LIVINGAPARTMENTS_MEDI           68.35
LIVINGAPARTMENTS_MODE           68.35
LIVINGAPARTMENTS_AVG            68.35
FLOORSMIN_MEDI                  67.85
FLOORSMIN_MODE                  67.85
FLOORSMIN_AVG                   67.85
YEARS_BUILD_MEDI                66.50
YEARS_BUILD_AVG                 66.50
YEARS_BUILD_MODE                66.50
OWN_CAR_AGE                     65.99
LANDAREA_MODE                   59.38
LANDAREA_AVG                    59.38
LANDAREA_MEDI                   59.38
BASEMENTAREA_MEDI               58.52
BASEMENTAREA_AVG                58.52
BASEMENTAREA_MODE               58.52
EXT_SOURCE_1                    56.38
NONLIVINGAREA_MEDI              55.18
NONLIVINGAREA_AVG               55.18
NONLIVINGAREA_MODE              55.18
ELEVATORS_MODE                  53.30
ELEVATORS_AVG                   53.30
ELEVATORS_MEDI                  53.30
WALLSMATERIAL_MODE              50.84
APARTMENTS_MODE                 50.75
APARTMENTS_AVG                  50.75
APARTMENTS_MEDI                 50.75
ENTRANCES_MEDI                  50.35
ENTRANCES_MODE                  50.35
ENTRANCES_AVG                   50.35
LIVINGAREA_MEDI                 50.19
LIVINGAREA_MODE                 50.19
LIVINGAREA_AVG                  50.19
HOUSETYPE_MODE                  50.18
FLOORSMAX_MODE                  49.76
FLOORSMAX_MEDI                  49.76
FLOORSMAX_AVG                   49.76
YEARS_BEGINEXPLUATATION_MEDI    48.78
YEARS_BEGINEXPLUATATION_AVG     48.78
YEARS_BEGINEXPLUATATION_MODE    48.78
TOTALAREA_MODE                  48.27
```

```
EMERGENCYSTATE_MODE             47.40
OCCUPATION_TYPE                 31.35
EXT_SOURCE_3                    19.83
AMT_REQ_CREDIT_BUREAU_QRT       13.50
AMT_REQ_CREDIT_BUREAU_YEAR      13.50
AMT_REQ_CREDIT_BUREAU_WEEK      13.50
AMT_REQ_CREDIT_BUREAU_MON       13.50
AMT_REQ_CREDIT_BUREAU_DAY       13.50
AMT_REQ_CREDIT_BUREAU_HOUR      13.50
NAME_TYPE_SUITE                  0.42
OBS_30_CNT_SOCIAL_CIRCLE         0.33
OBS_60_CNT_SOCIAL_CIRCLE         0.33
DEF_60_CNT_SOCIAL_CIRCLE         0.33
DEF_30_CNT_SOCIAL_CIRCLE         0.33
EXT_SOURCE_2                     0.21
AMT_GOODS_PRICE                  0.09
AMT_ANNUITY                      0.00
CNT_FAM_MEMBERS                  0.00
DAYS_LAST_PHONE_CHANGE           0.00
AMT_CREDIT                       0.00
FLAG_OWN_CAR                     0.00
FLAG_EMAIL                       0.00
TARGET                           0.00
FLAG_PHONE                       0.00
FLAG_CONT_MOBILE                 0.00
FLAG_WORK_PHONE                  0.00
FLAG_EMP_PHONE                   0.00
FLAG_MOBIL                       0.00
NAME_CONTRACT_TYPE               0.00
CODE_GENDER                      0.00
FLAG_OWN_REALTY                  0.00
AMT_INCOME_TOTAL                 0.00
DAYS_ID_PUBLISH                  0.00
DAYS_REGISTRATION                0.00
DAYS_EMPLOYED                    0.00
DAYS_BIRTH                       0.00
REGION_POPULATION_RELATIVE       0.00
REGION_RATING_CLIENT             0.00
NAME_FAMILY_STATUS               0.00
NAME_EDUCATION_TYPE              0.00
NAME_INCOME_TYPE                 0.00
CNT_CHILDREN                     0.00
NAME_HOUSING_TYPE                0.00
REG_REGION_NOT_LIVE_REGION       0.00
REGION_RATING_CLIENT_W_CITY      0.00
WEEKDAY_APPR_PROCESS_START       0.00
FLAG_DOCUMENT_2                  0.00
FLAG_DOCUMENT_3                  0.00
FLAG_DOCUMENT_4                  0.00
FLAG_DOCUMENT_5                  0.00
FLAG_DOCUMENT_6                  0.00
FLAG_DOCUMENT_7                  0.00
FLAG_DOCUMENT_8                  0.00
FLAG_DOCUMENT_9                  0.00
FLAG_DOCUMENT_10                 0.00
FLAG_DOCUMENT_11                 0.00
FLAG_DOCUMENT_12                 0.00
FLAG_DOCUMENT_13                 0.00
FLAG_DOCUMENT_14                 0.00
FLAG_DOCUMENT_15                 0.00
FLAG_DOCUMENT_16                 0.00
FLAG_DOCUMENT_17                 0.00
FLAG_DOCUMENT_18                 0.00
FLAG_DOCUMENT_19                 0.00
```

```
FLAG_DOCUMENT_20               0.00
FLAG_DOCUMENT_21               0.00
ORGANIZATION_TYPE              0.00
LIVE_CITY_NOT_WORK_CITY        0.00
REG_CITY_NOT_WORK_CITY         0.00
REG_CITY_NOT_LIVE_CITY         0.00
LIVE_REGION_NOT_WORK_REGION    0.00
REG_REGION_NOT_WORK_REGION     0.00
HOUR_APPR_PROCESS_START        0.00
SK_ID_CURR                     0.00
dtype: float64
```

Dealing with Null values more than 50 %

In [11]:

```
#creating a variable null_col_50 for storing null columns having missing values more

null_col_50 = null_values(appl_data)[null_values(appl_data)>50]
```

In [12]:

```
#revieving null_col_50

print(null_col_50)
print()
print("Num of columns having missing values more than 50% :",len(null_col_50))
```

```
COMMONAREA_MEDI             69.87
COMMONAREA_AVG              69.87
COMMONAREA_MODE             69.87
NONLIVINGAPARTMENTS_MODE    69.43
NONLIVINGAPARTMENTS_MEDI    69.43
NONLIVINGAPARTMENTS_AVG     69.43
FONDKAPREMONT_MODE          68.39
LIVINGAPARTMENTS_MEDI       68.35
LIVINGAPARTMENTS_MODE       68.35
LIVINGAPARTMENTS_AVG        68.35
FLOORSMIN_MEDI              67.85
FLOORSMIN_MODE              67.85
FLOORSMIN_AVG               67.85
YEARS_BUILD_MEDI            66.50
YEARS_BUILD_AVG             66.50
YEARS_BUILD_MODE            66.50
OWN_CAR_AGE                 65.99
LANDAREA_MODE               59.38
LANDAREA_AVG                59.38
LANDAREA_MEDI               59.38
BASEMENTAREA_MEDI           58.52
BASEMENTAREA_AVG            58.52
BASEMENTAREA_MODE           58.52
EXT_SOURCE_1                56.38
NONLIVINGAREA_MEDI          55.18
NONLIVINGAREA_AVG           55.18
NONLIVINGAREA_MODE          55.18
ELEVATORS_MODE              53.30
ELEVATORS_AVG               53.30
ELEVATORS_MEDI              53.30
WALLSMATERIAL_MODE          50.84
APARTMENTS_MODE             50.75
APARTMENTS_AVG              50.75
APARTMENTS_MEDI             50.75
ENTRANCES_MEDI              50.35
ENTRANCES_MODE              50.35
ENTRANCES_AVG               50.35
LIVINGAREA_MEDI             50.19
```

```
LIVINGAREA_MODE             50.19
LIVINGAREA_AVG              50.19
HOUSETYPE_MODE              50.18
dtype: float64

Num of columns having missing values more than 50% : 41
```

There are 41 columns having null values more than 50%

In [13]:
```python
null_col_50.index
```

Out[13]:
```
Index(['COMMONAREA_MEDI', 'COMMONAREA_AVG', 'COMMONAREA_MODE', 'NONLIVINGAPARTMENTS_M
ODE', 'NONLIVINGAPARTMENTS_MEDI', 'NONLIVINGAPARTMENTS_AVG', 'FONDKAPREMONT_MODE', 'L
IVINGAPARTMENTS_MEDI', 'LIVINGAPARTMENTS_MODE', 'LIVINGAPARTMENTS_AVG', 'FLOORSMIN_ME
DI', 'FLOORSMIN_MODE', 'FLOORSMIN_AVG', 'YEARS_BUILD_MEDI', 'YEARS_BUILD_AVG', 'YEARS
_BUILD_MODE', 'OWN_CAR_AGE', 'LANDAREA_MODE', 'LANDAREA_AVG', 'LANDAREA_MEDI', 'BASEM
ENTAREA_MEDI', 'BASEMENTAREA_AVG', 'BASEMENTAREA_MODE', 'EXT_SOURCE_1', 'NONLIVINGARE
A_MEDI', 'NONLIVINGAREA_AVG', 'NONLIVINGAREA_MODE', 'ELEVATORS_MODE', 'ELEVATORS_AV
G', 'ELEVATORS_MEDI', 'WALLSMATERIAL_MODE', 'APARTMENTS_MODE', 'APARTMENTS_AVG', 'APA
RTMENTS_MEDI', 'ENTRANCES_MEDI', 'ENTRANCES_MODE', 'ENTRANCES_AVG', 'LIVINGAREA_MED
I', 'LIVINGAREA_MODE', 'LIVINGAREA_AVG', 'HOUSETYPE_MODE'], dtype='object')
```

In [14]:
```python
# Now lets drop all the columns having missing values more than 50% that is 41 colum

appl_data.drop(columns = null_col_50.index, inplace = True)
```

In [15]:
```python
appl_data.shape  # Now there are 81 columns remaining
```

Out[15]:
```
(307511, 81)
```

After after dropping 41 columns we are left with 81 columns

Dealing with null values more than 15%

In [16]:
```python
# now we will deal with null values more than 15%

null_col_15 = null_values(appl_data)[null_values(appl_data)>15]
```

In [17]:
```python
null_col_15
```

Out[17]:
```
FLOORSMAX_AVG                   49.76
FLOORSMAX_MEDI                  49.76
FLOORSMAX_MODE                  49.76
YEARS_BEGINEXPLUATATION_AVG     48.78
YEARS_BEGINEXPLUATATION_MEDI    48.78
YEARS_BEGINEXPLUATATION_MODE    48.78
TOTALAREA_MODE                  48.27
EMERGENCYSTATE_MODE             47.40
OCCUPATION_TYPE                 31.35
EXT_SOURCE_3                    19.83
dtype: float64
```

- **from the columns dictionary we can conclude that only 'OCCUPATION_TYPE', 'EXT_SOURCE_3 looks relevant to TARGET column.thus dropping all other columns except 'OCCUPATION_TYPE','EXT_SOURCE_3**

In [18]:
```python
#removing 'OCCUPATION_TYPE', 'EXT_SOURCE_3' from "null_col_15" so that we can drop a
```

```
null_col_15.drop(["OCCUPATION_TYPE","EXT_SOURCE_3"], inplace = True)
```

In [19]:
```
print(null_col_15)
print()
print("No of columns having missing values more than 15% and are not reletable:",len(
```

```
FLOORSMAX_AVG                   49.76
FLOORSMAX_MEDI                  49.76
FLOORSMAX_MODE                  49.76
YEARS_BEGINEXPLUATATION_AVG     48.78
YEARS_BEGINEXPLUATATION_MEDI    48.78
YEARS_BEGINEXPLUATATION_MODE    48.78
TOTALAREA_MODE                  48.27
EMERGENCYSTATE_MODE             47.40
dtype: float64

No of columns having missing values more than 15% and are not reletable: 8
```

In [20]:
```
#thus removing columns having missing values more than 15% and which are not reletabl

appl_data.drop(null_col_15.index,axis=1, inplace = True)
```

In [21]:
```
appl_data.shape  # After dropping null_col_15, we have left with 73 columns
```

Out[21]:
```
(307511, 73)
```

- **After after dropping 8 columns we are left with 73 columns**
- **There are 2 more Columns with missing values more than 15%**

In [22]:
```
null_values(appl_data).head(10)
```

Out[22]:
```
OCCUPATION_TYPE               31.35
EXT_SOURCE_3                  19.83
AMT_REQ_CREDIT_BUREAU_YEAR    13.50
AMT_REQ_CREDIT_BUREAU_MON     13.50
AMT_REQ_CREDIT_BUREAU_WEEK    13.50
AMT_REQ_CREDIT_BUREAU_DAY     13.50
AMT_REQ_CREDIT_BUREAU_HOUR    13.50
AMT_REQ_CREDIT_BUREAU_QRT     13.50
NAME_TYPE_SUITE                0.42
OBS_30_CNT_SOCIAL_CIRCLE       0.33
dtype: float64
```

In [ ]:

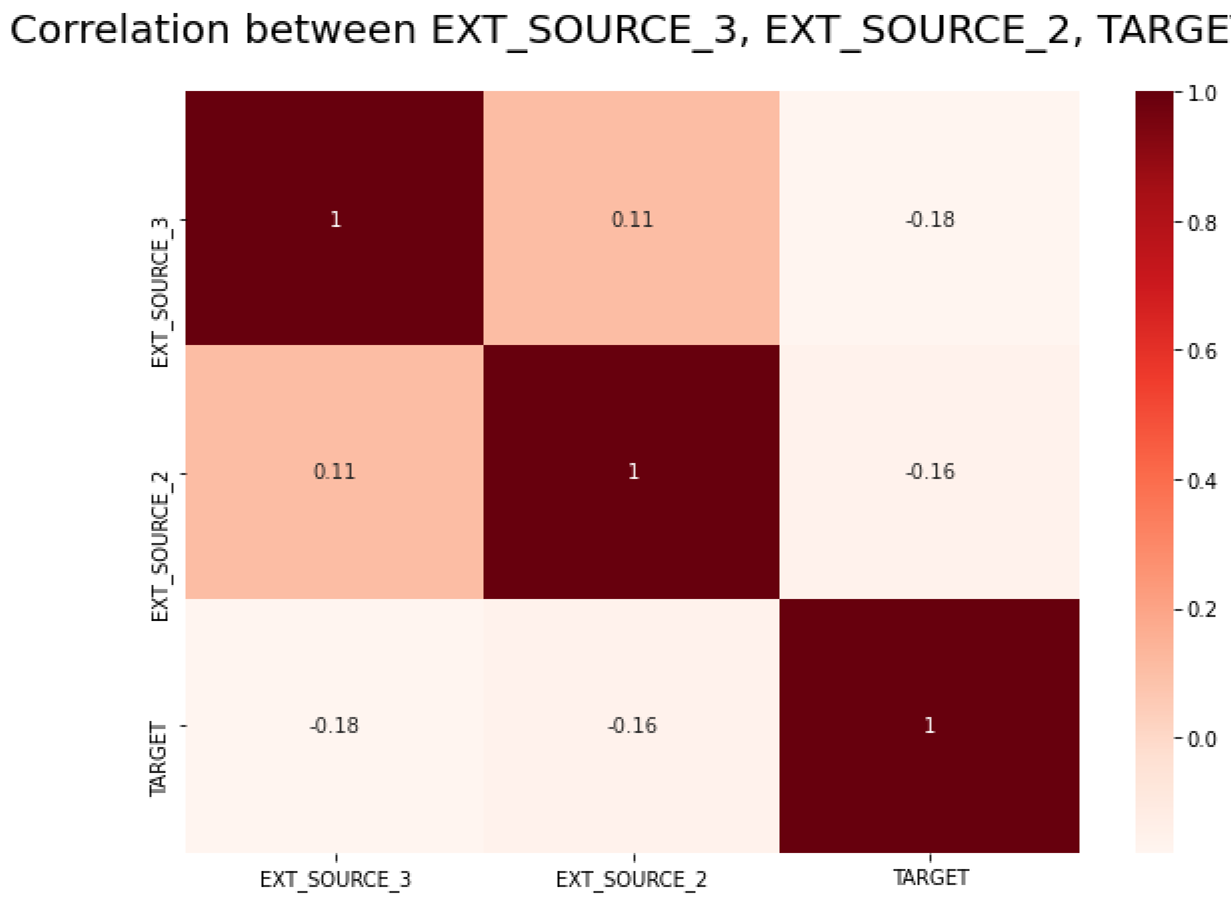
Analyse & Removing Unneccsary Columns

2Starting with EXT_SOURCE_3 , EXT_SOURCE_2. As they have normalised values, now we will understand the relation between these columns with TARGET column using a heatmap

In [23]:
```
irrev = ["EXT_SOURCE_3","EXT_SOURCE_2"]  # putting irrevlent columns in varibale "irr
```

In [24]:
```
plt.figure(figsize= [10,7])
```

```
sns.heatmap(appl_data[irrev+["TARGET"]].corr(), cmap="Reds",annot=True)

plt.title("Correlation between EXT_SOURCE_3, EXT_SOURCE_2, TARGET", fontdict={"fontsi
plt.show()
```



- **There seems to be no linear correlation and also from columns description we decided to remove these columns.**
- **Also we are aware correation doesn't cause causation**

In [25]:
```
#dropping above columns as decided

appl_data.drop(irrev, axis=1, inplace= True)
```

In [26]:
```
appl_data.shape  # Now we are left with 71 columns
```

Out[26]:
```
(307511, 71)
```

In [27]:
```
null_values(appl_data).head(10)
```

Out[27]:
```
OCCUPATION_TYPE               31.35
AMT_REQ_CREDIT_BUREAU_YEAR    13.50
AMT_REQ_CREDIT_BUREAU_MON     13.50
AMT_REQ_CREDIT_BUREAU_WEEK    13.50
AMT_REQ_CREDIT_BUREAU_DAY     13.50
AMT_REQ_CREDIT_BUREAU_HOUR    13.50
AMT_REQ_CREDIT_BUREAU_QRT     13.50
NAME_TYPE_SUITE                0.42
OBS_30_CNT_SOCIAL_CIRCLE       0.33
DEF_30_CNT_SOCIAL_CIRCLE       0.33
dtype: float64
```

Now we will check columns with FLAGS and their relation with TARGET columns to remove irrelevant ones

In [28]:
```
# adding all flags coloumns in variable "flag_columns"

flag_columns = [col for col in appl_data.columns if "FLAG" in col]

flag_columns  # Viewing all FLAG columns
```

Out[28]:
```
['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']
```

In [29]:
```
# creating flag_df dataframe having all FLAG columns and TARGET column

flag_df = appl_data[flag_columns+["TARGET"]]
```

In [30]:
```
# replacing "0" as repayer and "1" as defaulter for TARGET column

flag_df["TARGET"] = flag_df["TARGET"].replace({1:"Defaulter", 0:"Repayer"})
```

In [31]:
```
# as stated in columnn description replacing "1" as Y being TRUE and "0" as N being F

for i in flag_df:
    if i!= "TARGET":
        flag_df[i] = flag_df[i].replace({1:"Y", 0:"N"})
```

In [32]:
```
flag_df.head()
```

Out[32]:

	FLAG_OWN_CAR	FLAG_OWN_REALTY	FLAG_MOBIL	FLAG_EMP_PHONE	FLAG_WORK_PHONE	I
0	N	Y	Y	Y	N	

	FLAG_OWN_CAR	FLAG_OWN_REALTY	FLAG_MOBIL	FLAG_EMP_PHONE	FLAG_WORK_PHONE
1	N	N	Y	Y	N
2	Y	Y	Y	Y	Y
3	N	Y	Y	Y	N
4	N	Y	Y	Y	N

In [33]:

```
import itertools # using itertools for efficient looping plotting subplots

# Plotting all the graph to find the relation and evaluting for dropping such columns

plt.figure(figsize = [20,24])

for i,j in itertools.zip_longest(flag_columns,range(len(flag_columns))):
    plt.subplot(7,4,j+1)
    ax = sns.countplot(flag_df[i], hue = flag_df["TARGET"], palette = ["r","b"])
    #plt.yticks(fontsize=8)
    plt.xlabel("")
    plt.ylabel("")
    plt.title(i)
```

Columns (FLAG_OWN_REALTY, FLAG_MOBIL ,FLAG_EMP_PHONE, FLAG_CONT_MOBILE, FLAG_DOCUMENT_3) have more repayers than defaulter and from these keeping FLAG_DOCUMENT_3,FLAG_OWN_REALTY, FLAG_MOBIL more sense thus we can include these columns and remove all other FLAG columns for furhter analysis.

In [34]:
```
# removing required columns from "flag_df" such that we can remove the irrelevent co

flag_df.drop(["TARGET","FLAG_OWN_REALTY","FLAG_MOBIL","FLAG_DOCUMENT_3"], axis=1 , ir
```

In [35]:
```
len(flag_df.columns)
```

Out[35]:
```
25
```

In [36]:
```
# dropping the columns of "flag_df" dataframe that is removing more 25 columns from '

appl_data.drop(flag_df.columns, axis=1, inplace= True)
```

In [37]:
```
appl_data.shape    # Now we are left 46 revelent columns
```

Out[37]:
```
(307511, 46)
```

After removing uneccsarry, irrelevent and missing columns. We are left with 46 columns**

Now that we have removed all the unneccesarry columns, we will proced with imputing values for relevent missing columns whereever required

In [38]:
```
null_values(appl_data).head(10)
```

Out[38]:
```
OCCUPATION_TYPE               31.35
AMT_REQ_CREDIT_BUREAU_YEAR    13.50
AMT_REQ_CREDIT_BUREAU_QRT     13.50
AMT_REQ_CREDIT_BUREAU_MON     13.50
AMT_REQ_CREDIT_BUREAU_WEEK    13.50
AMT_REQ_CREDIT_BUREAU_DAY     13.50
AMT_REQ_CREDIT_BUREAU_HOUR    13.50
NAME_TYPE_SUITE                0.42
DEF_60_CNT_SOCIAL_CIRCLE       0.33
OBS_60_CNT_SOCIAL_CIRCLE       0.33
dtype: float64
```

Now we have only 7 columns which have missing values more than 1%.

In [39]:
```
#Percentage of each category present in "OCCUPATION_TYPE"

appl_data["OCCUPATION_TYPE"].value_counts(normalize=True)*100
```

Out[39]:
```
Laborers                 26.139636
Sales staff              15.205570
Core staff               13.058924
Managers                 10.122679
Drivers                   8.811576
High skill tech staff     5.390299
Accountants               4.648067
Medicine staff            4.043672
Security staff            3.183498
Cooking staff             2.816408
Cleaning staff            2.203960
Private service staff     1.256158
Low-skill Laborers        0.991379
Waiters/barmen staff      0.638499
Secretaries               0.618132
Realty agents             0.355722
HR staff                  0.266673
IT staff                  0.249147
Name: OCCUPATION_TYPE, dtype: float64
```

Insight:

- **from above it looks like this columnn is categorical one and have missing values of 31.35%. to fix this we will impute another category as "Unknown" for the missing values.**

In [40]:
```
# imputing null values with "Unknown"
```

```python
appl_data["OCCUPATION_TYPE"] = appl_data["OCCUPATION_TYPE"].fillna("Unknown")
```

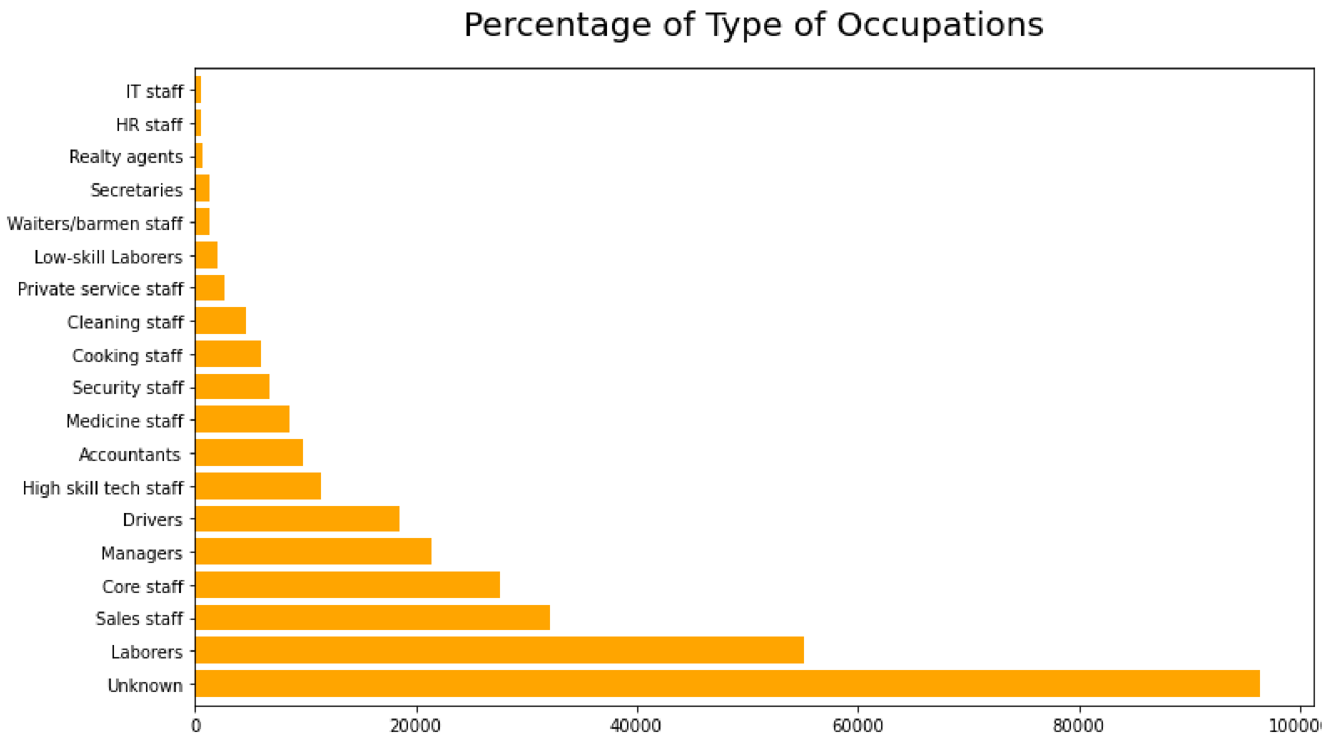
In [41]:

```python
appl_data["OCCUPATION_TYPE"].isnull().sum() # Now we have zero null values
```

Out[41]: 0

In [42]:

```python
# Plotting a percentage graph having each category of "OCCUPATION_TYPE"

plt.figure(figsize = [12,7])
(appl_data["OCCUPATION_TYPE"].value_counts()).plot.barh(color= "orange",width = .8)
plt.title("Percentage of Type of Occupations", fontdict={"fontsize":20}, pad =20)
plt.show()
```



- **Highest percentage of values belongs to Unknown group and Secons belongs to Laborers**

Now let's move to other 6 columns :

AMT_REQ_CREDIT_BUREAU_YEAR", "AMT_REQ_CREDIT_BUREAU_QRT","AMT_REQ_CREDIT_BUREAU_MON", "AMT_REQ_CREDIT_BUREAU_WEEK","AMT_REQ_CREDIT_BUREAU_DAY", "AMT_REQ_CREDIT_BUREAU_HOUR"

In [43]:

```python
appl_data[["AMT_REQ_CREDIT_BUREAU_YEAR","AMT_REQ_CREDIT_BUREAU_QRT","AMT_REQ_CREDIT_E
"AMT_REQ_CREDIT_BUREAU_DAY","AMT_REQ_CREDIT_BUREAU_HOUR"]].describe()
```

Out[43]:

	AMT_REQ_CREDIT_BUREAU_YEAR	AMT_REQ_CREDIT_BUREAU_QRT	AMT_REQ_CREDIT_BURE
count	265992.000000	265992.000000	26599
mean	1.899974	0.265474	
std	1.869295	0.794056	

	AMT_REQ_CREDIT_BUREAU_YEAR	AMT_REQ_CREDIT_BUREAU_QRT	AMT_REQ_CREDIT_BURE
min	0.000000	0.000000	
25%	0.000000	0.000000	
50%	1.000000	0.000000	
75%	3.000000	0.000000	
max	25.000000	261.000000	2

These above columns represent number of enquries made for the customer(which should be discrete and not continous). from above describe results we see that all values are numerical and can conclude that for imputing missing we should not use mean as it is in decimal form, hence for imputing purpose we will use median for all these columns.

In [44]:

```
These above columns represent number of enquries made for the customer
```

In [45]:

```
#filling missing values with median values

appl_data.fillna(appl_data[amt_credit].median(),inplace = True)
```

In [46]:

```
null_values(appl_data).head(10)
```

Out[46]:

```
NAME_TYPE_SUITE              0.42
DEF_60_CNT_SOCIAL_CIRCLE     0.33
OBS_60_CNT_SOCIAL_CIRCLE     0.33
DEF_30_CNT_SOCIAL_CIRCLE     0.33
OBS_30_CNT_SOCIAL_CIRCLE     0.33
AMT_GOODS_PRICE              0.09
AMT_ANNUITY                  0.00
CNT_FAM_MEMBERS              0.00
DAYS_LAST_PHONE_CHANGE       0.00
DAYS_EMPLOYED                0.00
dtype: float64
```

Still there some missing value coloumns but we will not impute them as the missing value count very less.

4. Standardising values

In [47]:

```
appl_data.describe()
```

Out[47]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	30749
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	2710
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	1449
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	161
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	1652
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	2490
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	3459
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	25802

columns DAYS_BIRTH, DAYS_EMPLOYED, DAYS_REGISTRATION, DAYS_ID_PUBLISH, DAYS_LAST_PHONE_CHANGE which counts days have negative values. thus will correct those values convert DAYS_BIRTH to AGE in years , DAYS_EMPLOYED to YEARS EMPLOYED

Taking care of columns: AMT_INCOME_TOTAL, AMT_CREDIT, AMT_GOODS_PRICE

In [48]:

```
# Binning Numerical Columns to create a categorical column

# Creating bins for income amount in term of Lakhs
appl_data['AMT_INCOME_TOTAL']=appl_data['AMT_INCOME_TOTAL']/100000

bins = [0,1,2,3,4,5,6,7,8,9,10,11]
slot = ['0-1L','1L-2L', '2L-3L','3L-4L','4L-5L','5L-6L','6L-7L','7L-8L','8L-9L','9L-1

appl_data['AMT_INCOME_RANGE']=pd.cut(appl_data['AMT_INCOME_TOTAL'],bins,labels=slot)
```

In [49]:

```
round((appl_data["AMT_INCOME_RANGE"].value_counts(normalize = True)*100),2)
```

Out[49]:

```
1L-2L        50.73
2L-3L        21.21
0-1L         20.73
3L-4L         4.78
4L-5L         1.74
5L-6L         0.36
6L-7L         0.28
8L-9L         0.10
7L-8L         0.05
9L-10L        0.01
10L Above     0.01
Name: AMT_INCOME_RANGE, dtype: float64
```

In [50]:

```
# Creating bins for Credit amount in term of Lakhs
appl_data['AMT_CREDIT']=appl_data['AMT_CREDIT']/100000

bins = [0,1,2,3,4,5,6,7,8,9,10,100]
slots = ['0-1L','1L-2L', '2L-3L','3L-4L','4L-5L','5L-6L','6L-7L','7L-8L','8L-9L','9L-

appl_data['AMT_CREDIT_RANGE']=pd.cut(appl_data['AMT_CREDIT'],bins=bins,labels=slots)
```

In [51]:

```
round((appl_data["AMT_CREDIT_RANGE"].value_counts(normalize = True)*100),2)
```

Out[51]:

```
2L-3L        17.82
10L Above    16.25
5L-6L        11.13
4L-5L        10.42
1L-2L         9.80
3L-4L         8.56
6L-7L         7.82
8L-9L         7.09
7L-8L         6.24
9L-10L        2.90
0-1L          1.95
Name: AMT_CREDIT_RANGE, dtype: float64
```

In [52]:

```
# Creating bins for Price of Goods in term of Lakhs
appl_data['AMT_GOODS_PRICE']=appl_data['AMT_GOODS_PRICE']/100000
```

```
bins = [0,1,2,3,4,5,6,7,8,9,10,100]
slots = ['0-1L','1L-2L', '2L-3L','3L-4L','4L-5L','5L-6L','6L-7L','7L-8L','8L-9L','9L-

appl_data['AMT_GOODS_PRICE_RANGE']=pd.cut(appl_data['AMT_GOODS_PRICE'],bins=bins,labe
```

In [53]:

```
round((appl_data["AMT_GOODS_PRICE_RANGE"].value_counts(normalize = True)*100),2)
```

Out[53]:

```
2L-3L        20.43
4L-5L        18.54
6L-7L        13.03
10L Above    11.11
1L-2L        10.73
8L-9L         6.99
3L-4L         6.91
5L-6L         4.27
0-1L          2.83
7L-8L         2.64
9L-10L        2.53
Name: AMT_GOODS_PRICE_RANGE, dtype: float64
```

Dealing with columns :

In [54]:

```
# creating "days_col" varibale to store all days columns
days_col = ["DAYS_BIRTH", "DAYS_EMPLOYED", "DAYS_REGISTRATION", "DAYS_ID_PUBLISH", "D

appl_data[days_col].describe()
```

Out[54]:

	DAYS_BIRTH	DAYS_EMPLOYED	DAYS_REGISTRATION	DAYS_ID_PUBLISH	DAYS_LAST_PH
count	307511.000000	307511.000000	307511.000000	307511.000000	
mean	-16036.995067	63815.045904	-4986.120328	-2994.202373	
std	4363.988632	141275.766519	3522.886321	1509.450419	
min	-25229.000000	-17912.000000	-24672.000000	-7197.000000	
25%	-19682.000000	-2760.000000	-7479.500000	-4299.000000	
50%	-15750.000000	-1213.000000	-4504.000000	-3254.000000	
75%	-12413.000000	-289.000000	-2010.000000	-1720.000000	
max	-7489.000000	365243.000000	0.000000	0.000000	

- **from describe we get that days are in negative that is not usual, so to correct it we use absolute function as below**

In [55]:

```
#using abs() function to correct the days values

appl_data[days_col]= abs(appl_data[days_col])
```

In [56]:

```
# Data is correct now
appl_data[days_col]= abs(appl_data[days_col])
```

Out[56]:

	DAYS_BIRTH	DAYS_EMPLOYED	DAYS_REGISTRATION	DAYS_ID_PUBLISH	DAYS_LAST_PH
count	307511.000000	307511.000000	307511.000000	307511.000000	
mean	16036.995067	67724.742149	4986.120328	2994.202373	

	DAYS_BIRTH	DAYS_EMPLOYED	DAYS_REGISTRATION	DAYS_ID_PUBLISH	DAYS_LAST_PH
std	4363.988632	139443.751806	3522.886321	1509.450419	
min	7489.000000	0.000000	0.000000	0.000000	
25%	12413.000000	933.000000	2010.000000	1720.000000	
50%	15750.000000	2219.000000	4504.000000	3254.000000	
75%	19682.000000	5707.000000	7479.500000	4299.000000	
max	25229.000000	365243.000000	24672.000000	7197.000000	

4.3. now convert DAYS_BIRTH, DAYS_EMPLOYED columns in terms of Years and binning years for better understanding, that is adding two more categorical column

In [57]:

```
appl_data["AGE"] = appl_data["DAYS_BIRTH"]/365
bins = [0,20,25,30,35,40,45,50,55,60,100]
slots = ["0-20","20-25","25-30","30-35","35-40","40-45","45-50","50-55","55-60","60 A

appl_data["AGE_GROUP"] = pd.cut(appl_data["AGE"], bins=bins, labels=slots)
```

In [58]:

```
appl_data["AGE_GROUP"].value_counts(normalize= True)*100
```

Out[58]:

```
35-40       13.940314
40-45       13.464884
30-35       12.825557
60 Above    11.569993
45-50       11.425608
50-55       11.362846
55-60       10.770346
25-30       10.686447
20-25        3.954005
0-20         0.000000
Name: AGE_GROUP, dtype: float64
```

In [59]:

```
#creating column "EMPLOYEMENT_YEARS" from "DAYS_EMPLOYED"

appl_data["YEARS_EMPLOYED"] = appl_data["DAYS_EMPLOYED"]/365
bins = [0,5,10,15,20,25,30,50]
slots = ["0-5","5-10","10-15","15-20","20-25","25-30","30 Above"]

appl_data["EMPLOYEMENT_YEARS"] = pd.cut(appl_data["YEARS_EMPLOYED"], bins=bins, label
```

In [60]:

```
appl_data["EMPLOYEMENT_YEARS"].value_counts(normalize= True)*100
```

Out[60]:

```
0-5         54.061911
5-10        25.729074
10-15       10.926289
15-20        4.302854
20-25        2.476054
25-30        1.311996
30 Above     1.191822
Name: EMPLOYEMENT_YEARS, dtype: float64
```

Identifying Outliers

In [61]:

```
appl_data.describe()
```

Out[61]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT
count	307511.000000	307511.000000	307511.000000	307511.000000	307511.000000	3074
mean	278180.518577	0.080729	0.417052	1.687979	5.990260	271
std	102790.175348	0.272419	0.722121	2.371231	4.024908	144
min	100002.000000	0.000000	0.000000	0.256500	0.450000	16
25%	189145.500000	0.000000	0.000000	1.125000	2.700000	165
50%	278202.000000	0.000000	0.000000	1.471500	5.135310	249
75%	367142.500000	0.000000	1.000000	2.025000	8.086500	345
max	456255.000000	1.000000	19.000000	1170.000000	40.500000	2580

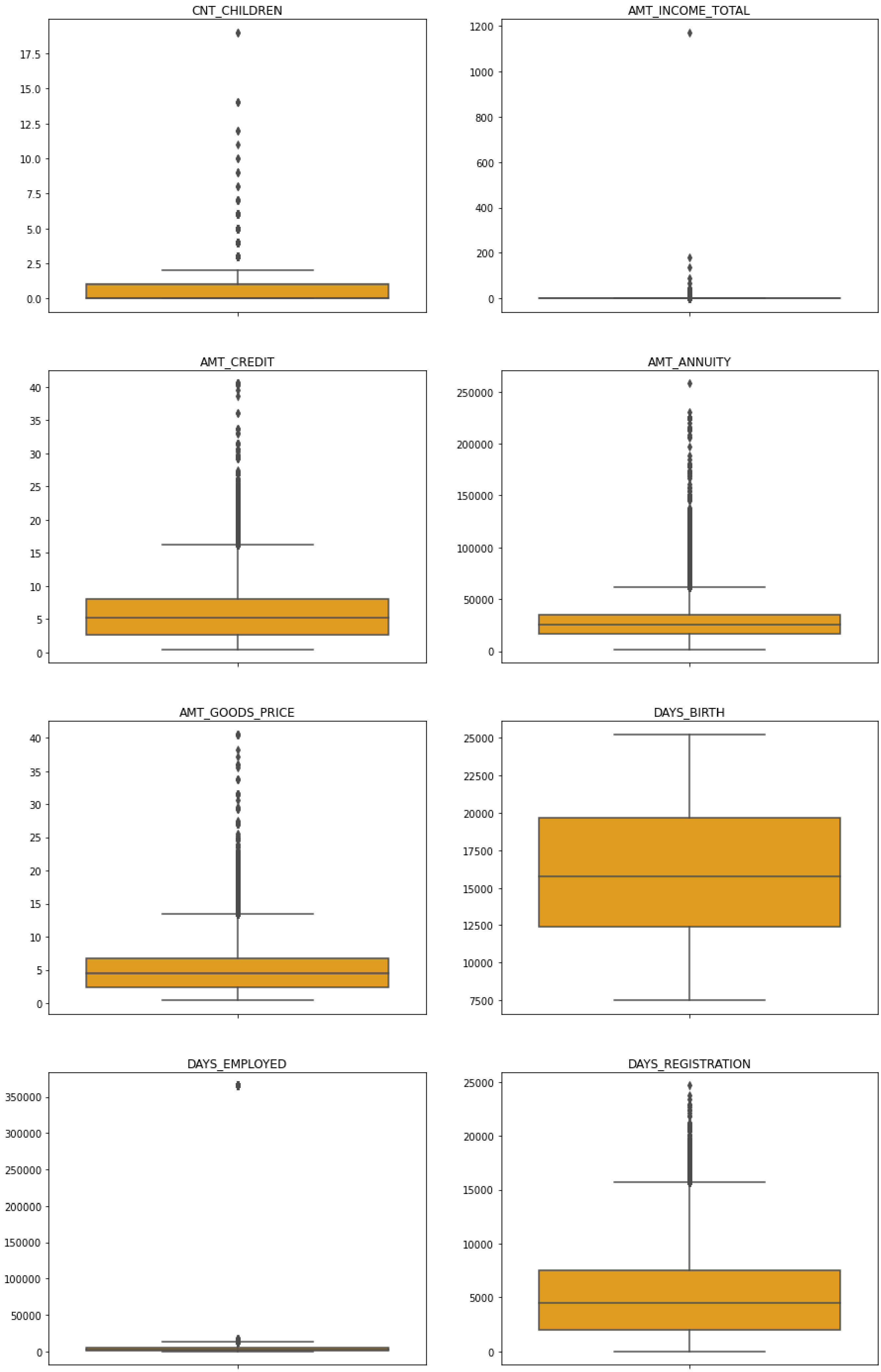
- **from describe we could find all the columns those wo have high difference between max and 75 percentile and the ones which makes no sense having max value to be so high are captured below:**

In [62]:

```
outlier_col = ["CNT_CHILDREN","AMT_INCOME_TOTAL", "AMT_CREDIT", "AMT_ANNUITY", "AMT_G
               "DAYS_BIRTH", "DAYS_EMPLOYED", "DAYS_REGISTRATION"]
```

In [63]:

```
plt.figure(figsize=[15,25])
for i,j in itertools.zip_longest(outlier_col, range(len(outlier_col))):
    plt.subplot(4,2,j+1)
    sns.boxplot(y = appl_data[i], orient = "h", color = "orange")
    #plt.yticks(fontsize=8)
    plt.xlabel("")
    plt.ylabel("")
    plt.title(i)
```


CNT_CHILDREN
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
AMT_INCOME_TOTAL
0
200
400
600
800
1000
1200
AMT_CREDIT
0
5
10
15
20
25
30
35
40
AMT_ANNUITY
0
50000
100000
150000
200000
250000
AMT_GOODS_PRICE
0
5
10
15
20
25
30
35
40
DAYS_BIRTH
7500
10000
12500
15000
17500
20000
22500
25000
DAYS_EMPLOYED
0
50000
100000
150000
200000
250000
300000
350000
DAYS_REGISTRATION
0
5000
10000
15000
20000
25000

Insight:

It can be seen that in current application data

- **AMT_ANNUITY, AMT_CREDIT, AMT_GOODS_PRICE,CNT_CHILDREN have some number of outliers.**
- **AMT_INCOME_TOTAL has huge number of outliers which indicate that few of the loan applicants have high income when compared to the others.**
- **DAYS_BIRTH has no outliers which means the data available is reliable.**
- **DAYS_EMPLOYED has outlier values around 350000(days) which is around 958 years which is impossible and hence this has to be incorrect entry.**

In [64]:

```
appl_data.nunique().sort_values()
```

Out[64]:

```
LIVE_REGION_NOT_WORK_REGION         2
TARGET                              2
NAME_CONTRACT_TYPE                  2
REG_REGION_NOT_LIVE_REGION          2
FLAG_OWN_REALTY                     2
LIVE_CITY_NOT_WORK_CITY             2
REG_CITY_NOT_WORK_CITY              2
REG_CITY_NOT_LIVE_CITY              2
FLAG_DOCUMENT_3                     2
REG_REGION_NOT_WORK_REGION          2
FLAG_MOBIL                          2
REGION_RATING_CLIENT                3
CODE_GENDER                         3
REGION_RATING_CLIENT_W_CITY         3
NAME_EDUCATION_TYPE                 5
AMT_REQ_CREDIT_BUREAU_HOUR          5
NAME_FAMILY_STATUS                  6
NAME_HOUSING_TYPE                   6
EMPLOYEMENT_YEARS                   7
WEEKDAY_APPR_PROCESS_START          7
NAME_TYPE_SUITE                     7
NAME_INCOME_TYPE                    8
AMT_REQ_CREDIT_BUREAU_WEEK          9
AMT_REQ_CREDIT_BUREAU_DAY           9
DEF_60_CNT_SOCIAL_CIRCLE            9
AGE_GROUP                           9
DEF_30_CNT_SOCIAL_CIRCLE           10
AMT_REQ_CREDIT_BUREAU_QRT          11
AMT_INCOME_RANGE                   11
AMT_CREDIT_RANGE                   11
AMT_GOODS_PRICE_RANGE              11
CNT_CHILDREN                       15
CNT_FAM_MEMBERS                    17
OCCUPATION_TYPE                    19
AMT_REQ_CREDIT_BUREAU_MON          24
HOUR_APPR_PROCESS_START            24
AMT_REQ_CREDIT_BUREAU_YEAR         25
OBS_60_CNT_SOCIAL_CIRCLE           33
OBS_30_CNT_SOCIAL_CIRCLE           33
ORGANIZATION_TYPE                  58
REGION_POPULATION_RELATIVE         81
AMT_GOODS_PRICE                  1002
AMT_INCOME_TOTAL                 2548
DAYS_LAST_PHONE_CHANGE           3773
AMT_CREDIT                       5603
DAYS_ID_PUBLISH                  6168
DAYS_EMPLOYED                   12574
```

```
YEARS_EMPLOYED                  12574
AMT_ANNUITY                     13672
DAYS_REGISTRATION               15688
DAYS_BIRTH                      17460
AGE                             17460
SK_ID_CURR                     307511
dtype: int64
```

In [65]:

```
#Checking the number of unique values each column possess to identify categorical co

appl_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 53 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   SK_ID_CURR                   307511 non-null  int64
 1   TARGET                       307511 non-null  int64
 2   NAME_CONTRACT_TYPE           307511 non-null  object
 3   CODE_GENDER                  307511 non-null  object
 4   FLAG_OWN_REALTY              307511 non-null  object
 5   CNT_CHILDREN                 307511 non-null  int64
 6   AMT_INCOME_TOTAL             307511 non-null  float64
 7   AMT_CREDIT                   307511 non-null  float64
 8   AMT_ANNUITY                  307499 non-null  float64
 9   AMT_GOODS_PRICE              307233 non-null  float64
 10  NAME_TYPE_SUITE              306219 non-null  object
 11  NAME_INCOME_TYPE             307511 non-null  object
 12  NAME_EDUCATION_TYPE          307511 non-null  object
 13  NAME_FAMILY_STATUS           307511 non-null  object
 14  NAME_HOUSING_TYPE            307511 non-null  object
 15  REGION_POPULATION_RELATIVE   307511 non-null  float64
 16  DAYS_BIRTH                   307511 non-null  float64
 17  DAYS_EMPLOYED                307511 non-null  float64
 18  DAYS_REGISTRATION            307511 non-null  float64
 19  DAYS_ID_PUBLISH              307511 non-null  float64
 20  FLAG_MOBIL                   307511 non-null  int64
 21  OCCUPATION_TYPE              307511 non-null  object
 22  CNT_FAM_MEMBERS              307509 non-null  float64
 23  REGION_RATING_CLIENT         307511 non-null  int64
 24  REGION_RATING_CLIENT_W_CITY  307511 non-null  int64
 25  WEEKDAY_APPR_PROCESS_START   307511 non-null  object
 26  HOUR_APPR_PROCESS_START      307511 non-null  int64
 27  REG_REGION_NOT_LIVE_REGION   307511 non-null  int64
 28  REG_REGION_NOT_WORK_REGION   307511 non-null  int64
 29  LIVE_REGION_NOT_WORK_REGION  307511 non-null  int64
 30  REG_CITY_NOT_LIVE_CITY       307511 non-null  int64
 31  REG_CITY_NOT_WORK_CITY       307511 non-null  int64
 32  LIVE_CITY_NOT_WORK_CITY      307511 non-null  int64
 33  ORGANIZATION_TYPE            307511 non-null  object
 34  OBS_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 35  DEF_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 36  OBS_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 37  DEF_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 38  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64
 39  FLAG_DOCUMENT_3              307511 non-null  int64
 40  AMT_REQ_CREDIT_BUREAU_HOUR   307511 non-null  float64
 41  AMT_REQ_CREDIT_BUREAU_DAY    307511 non-null  float64
 42  AMT_REQ_CREDIT_BUREAU_WEEK   307511 non-null  float64
 43  AMT_REQ_CREDIT_BUREAU_MON    307511 non-null  float64
 44  AMT_REQ_CREDIT_BUREAU_QRT    307511 non-null  float64
 45  AMT_REQ_CREDIT_BUREAU_YEAR   307511 non-null  float64
 46  AMT_INCOME_RANGE             307279 non-null  category
```

```
 47  AMT_CREDIT_RANGE            307511 non-null  category
 48  AMT_GOODS_PRICE_RANGE       307233 non-null  category
 49  AGE                         307511 non-null  float64
 50  AGE_GROUP                   307511 non-null  category
 51  YEARS_EMPLOYED              307511 non-null  float64
 52  EMPLOYEMENT_YEARS           252135 non-null  category
dtypes: category(5), float64(23), int64(14), object(11)
memory usage: 114.1+ MB
```

Converting Desired columns from Object to categorical column

In [66]:

```
appl_data.columns
```

Out[66]:

```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_REALT
Y', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRIC
E', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATU
S', 'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED',
'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'FLAG_MOBIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEM
BERS', 'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_S
TART', '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_30_CNT_SOCIAL_CIRCLE', 'D
EF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'DA
YS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_3', 'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CRED
IT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
       'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BURE
AU_YEAR', 'AMT_INCOME_RANGE', 'AMT_CREDIT_RANGE', 'AMT_GOODS_PRICE_RANGE', 'AGE', 'AG
E_GROUP', 'YEARS_EMPLOYED', 'EMPLOYEMENT_YEARS'],
      dtype='object')
```

In [67]:

```
#from the list, we have taken out the desired columns for conversion

categorical_columns = ['NAME_CONTRACT_TYPE','CODE_GENDER','NAME_TYPE_SUITE','NAME_INC
                       'NAME_FAMILY_STATUS','NAME_HOUSING_TYPE','OCCUPATION_TYPE','WE
                       'ORGANIZATION_TYPE','FLAG_OWN_REALTY','LIVE_CITY_NOT_WORK_CITY
                       'REG_CITY_NOT_LIVE_CITY','REG_CITY_NOT_WORK_CITY','REG_REGION_
                       'LIVE_REGION_NOT_WORK_REGION','REGION_RATING_CLIENT','WEEKDAY_
                       'REGION_RATING_CLIENT_W_CITY','CNT_CHILDREN','CNT_FAM_MEMBERS'

for col in categorical_columns:
    appl_data[col] = pd.Categorical(appl_data[col])
```

In [68]:

```
len(categorical_columns) # Converting total of 21 columns to categorical one
```

Out[68]:

```
21
```

In [69]:

```
appl_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 53 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   SK_ID_CURR                  307511 non-null  int64
 1   TARGET                      307511 non-null  int64
 2   NAME_CONTRACT_TYPE          307511 non-null  category
 3   CODE_GENDER                 307511 non-null  category
 4   FLAG_OWN_REALTY             307511 non-null  category
```

```
 5   CNT_CHILDREN                 307511 non-null  category
 6   AMT_INCOME_TOTAL             307511 non-null  float64
 7   AMT_CREDIT                   307511 non-null  float64
 8   AMT_ANNUITY                  307499 non-null  float64
 9   AMT_GOODS_PRICE              307233 non-null  float64
 10  NAME_TYPE_SUITE              306219 non-null  category
 11  NAME_INCOME_TYPE             307511 non-null  category
 12  NAME_EDUCATION_TYPE          307511 non-null  category
 13  NAME_FAMILY_STATUS           307511 non-null  category
 14  NAME_HOUSING_TYPE            307511 non-null  category
 15  REGION_POPULATION_RELATIVE   307511 non-null  float64
 16  DAYS_BIRTH                   307511 non-null  float64
 17  DAYS_EMPLOYED                307511 non-null  float64
 18  DAYS_REGISTRATION            307511 non-null  float64
 19  DAYS_ID_PUBLISH              307511 non-null  float64
 20  FLAG_MOBIL                   307511 non-null  int64
 21  OCCUPATION_TYPE              307511 non-null  category
 22  CNT_FAM_MEMBERS              307509 non-null  category
 23  REGION_RATING_CLIENT         307511 non-null  category
 24  REGION_RATING_CLIENT_W_CITY  307511 non-null  category
 25  WEEKDAY_APPR_PROCESS_START   307511 non-null  category
 26  HOUR_APPR_PROCESS_START      307511 non-null  int64
 27  REG_REGION_NOT_LIVE_REGION   307511 non-null  int64
 28  REG_REGION_NOT_WORK_REGION   307511 non-null  category
 29  LIVE_REGION_NOT_WORK_REGION  307511 non-null  category
 30  REG_CITY_NOT_LIVE_CITY       307511 non-null  category
 31  REG_CITY_NOT_WORK_CITY       307511 non-null  category
 32  LIVE_CITY_NOT_WORK_CITY      307511 non-null  category
 33  ORGANIZATION_TYPE            307511 non-null  category
 34  OBS_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 35  DEF_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 36  OBS_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 37  DEF_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 38  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64
 39  FLAG_DOCUMENT_3              307511 non-null  int64
 40  AMT_REQ_CREDIT_BUREAU_HOUR   307511 non-null  float64
 41  AMT_REQ_CREDIT_BUREAU_DAY    307511 non-null  float64
 42  AMT_REQ_CREDIT_BUREAU_WEEK   307511 non-null  float64
 43  AMT_REQ_CREDIT_BUREAU_MON    307511 non-null  float64
 44  AMT_REQ_CREDIT_BUREAU_QRT    307511 non-null  float64
 45  AMT_REQ_CREDIT_BUREAU_YEAR   307511 non-null  float64
 46  AMT_INCOME_RANGE             307279 non-null  category
 47  AMT_CREDIT_RANGE             307511 non-null  category
 48  AMT_GOODS_PRICE_RANGE        307233 non-null  category
 49  AGE                          307511 non-null  float64
 50  AGE_GROUP                    307511 non-null  category
 51  YEARS_EMPLOYED               307511 non-null  float64
 52  EMPLOYEMENT_YEARS            252135 non-null  category
dtypes: category(25), float64(22), int64(6)
memory usage: 73.0 MB
```

Insight

- **After imputing we have 53 columns and we will move ahead with Data Analysis on these columns**

Dataset 2 - "previous_application.csv"

In [70]:
```
# importing previous_application.csv

prev_appl = pd.read_csv(r"C:\Users\ARCHANA\previous_application.csv")
```

In [71]:
```
prev_appl.head()
```

Out[71]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_C
0	2030495	271877	Consumer loans	1730.430	17145.0	1
1	2802425	108129	Cash loans	25188.615	607500.0	67
2	2523466	122040	Cash loans	15060.735	112500.0	13
3	2819243	176158	Cash loans	47041.335	450000.0	47
4	1784265	202054	Cash loans	31924.395	337500.0	40

In [72]:
```
#Checking rows and columns of the raw data
prev_appl.shape
```

Out[72]:
```
(1670214, 37)
```

In [73]:
```
#Checking information of all the columns like data types
prev_appl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 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
 29  NAME_YIELD_GROUP             1670214 non-null  object
 30  PRODUCT_COMBINATION          1669868 non-null  object
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
```

```
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
```

- **There are 37 columns having various data types like object, int, float and 1670214 rows.**

In [74]:
```
v
```

Out[74]:

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOW
count	1.670214e+06	1.670214e+06	1.297979e+06	1.670214e+06	1.670213e+06	7
mean	1.923089e+06	2.783572e+05	1.595512e+04	1.752339e+05	1.961140e+05	6
std	5.325980e+05	1.028148e+05	1.478214e+04	2.927798e+05	3.185746e+05	2
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00	0.000000e+00	-
25%	1.461857e+06	1.893290e+05	6.321780e+03	1.872000e+04	2.416050e+04	0
50%	1.923110e+06	2.787145e+05	1.125000e+04	7.104600e+04	8.054100e+04	1
75%	2.384280e+06	3.675140e+05	2.065842e+04	1.803600e+05	2.164185e+05	7
max	2.845382e+06	4.562550e+05	4.180581e+05	6.905160e+06	6.905160e+06	3

Insight

- **there are 37 columns and 1679214 rows.**
- **there columns having negative, postive values which includes days. fixing is required**

In [75]:
```
#checking how many null values are present in each of the columns in percentage
null_values(prev_appl)
```

Out[75]:
```
RATE_INTEREST_PRIVILEGED       99.64
RATE_INTEREST_PRIMARY          99.64
RATE_DOWN_PAYMENT              53.64
AMT_DOWN_PAYMENT               53.64
NAME_TYPE_SUITE                49.12
DAYS_TERMINATION               40.30
NFLAG_INSURED_ON_APPROVAL      40.30
DAYS_FIRST_DRAWING             40.30
DAYS_FIRST_DUE                 40.30
DAYS_LAST_DUE_1ST_VERSION      40.30
DAYS_LAST_DUE                  40.30
AMT_GOODS_PRICE                23.08
AMT_ANNUITY                    22.29
CNT_PAYMENT                    22.29
PRODUCT_COMBINATION             0.02
AMT_CREDIT                      0.00
SK_ID_CURR                      0.00
NAME_CONTRACT_TYPE              0.00
WEEKDAY_APPR_PROCESS_START      0.00
HOUR_APPR_PROCESS_START         0.00
FLAG_LAST_APPL_PER_CONTRACT     0.00
NFLAG_LAST_APPL_IN_DAY          0.00
AMT_APPLICATION                 0.00
NAME_PAYMENT_TYPE               0.00
NAME_CASH_LOAN_PURPOSE          0.00
```

```
NAME_CONTRACT_STATUS          0.00
DAYS_DECISION                 0.00
CODE_REJECT_REASON            0.00
NAME_CLIENT_TYPE              0.00
NAME_GOODS_CATEGORY           0.00
NAME_PORTFOLIO                0.00
NAME_PRODUCT_TYPE             0.00
CHANNEL_TYPE                  0.00
SELLERPLACE_AREA              0.00
NAME_SELLER_INDUSTRY          0.00
NAME_YIELD_GROUP              0.00
SK_ID_PREV                    0.00
dtype: float64
```

In [76]:

```
#creating a variable p_null_col_50 for storing null columns having missing values mor

p_null_col_50 = null_values(prev_appl)[null_values(prev_appl)>50]
```

In [77]:

```
p_null_col_50 # There only 4 columns with missing valus more than 50%
```

Out[77]:

```
RATE_INTEREST_PRIVILEGED    99.64
RATE_INTEREST_PRIMARY       99.64
RATE_DOWN_PAYMENT           53.64
AMT_DOWN_PAYMENT            53.64
dtype: float64
```

In [78]:

```
 # There only 4 columns with missing valus more than 50%
```

In [79]:

```
#creating a variable p_null_col_15 for storing null columns having missing values mor

p_null_col_15 = null_values(prev_appl)[null_values(prev_appl)>15]
```

In [80]:

```
p_null_col_15
```

Out[80]:

```
NAME_TYPE_SUITE              49.12
DAYS_FIRST_DUE               40.30
DAYS_TERMINATION             40.30
DAYS_FIRST_DRAWING           40.30
NFLAG_INSURED_ON_APPROVAL    40.30
DAYS_LAST_DUE_1ST_VERSION    40.30
DAYS_LAST_DUE                40.30
AMT_GOODS_PRICE              23.08
AMT_ANNUITY                  22.29
CNT_PAYMENT                  22.29
dtype: float64
```

In [81]:

```
prev_appl[p_null_col_15.index]
```

Out[81]:

	NAME_TYPE_SUITE	DAYS_FIRST_DUE	DAYS_TERMINATION	DAYS_FIRST_DRAWING	NFI
0	NaN	-42.0	-37.0	365243.0	
1	Unaccompanied	-134.0	365243.0	365243.0	
2	Spouse, partner	-271.0	365243.0	365243.0	
3	NaN	-482.0	-177.0	365243.0	
4	NaN	NaN	NaN	NaN	

	NAME_TYPE_SUITE	DAYS_FIRST_DUE	DAYS_TERMINATION	DAYS_FIRST_DRAWING	NFI
...	...	...	...	...	
1670209	NaN	-508.0	-351.0	365243.0	
1670210	Unaccompanied	-1604.0	-1297.0	365243.0	
1670211	Spouse, partner	-1457.0	-1181.0	365243.0	
1670212	Family	-1155.0	-817.0	365243.0	
1670213	Family	-1163.0	-423.0	365243.0	

1670214 rows × 10 columns

In [82]:
```
prev_appl.columns
```

Out[82]:
```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_ANNUITY', 'AMT_APPLICAT
ION', 'AMT_CREDIT', 'AMT_GOODS_PRICE', 'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCE
SS_START', 'FLAG_LAST_APPL_PER_CONTRACT', 'NFLAG_LAST_APPL_IN_DAY', 'NAME_CASH_LOAN_P
URPOSE', 'NAME_CONTRACT_STATUS', 'DAYS_DECISION', 'NAME_PAYMENT_TYPE', 'CODE_REJECT_R
EASON', 'NAME_TYPE_SUITE', 'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY', 'NAME_PORTFOLI
O', 'NAME_PRODUCT_TYPE', 'CHANNEL_TYPE', 'SELLERPLACE_AREA', 'NAME_SELLER_INDUSTRY',
'CNT_PAYMENT', 'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION', 'DAYS_FIRST_DRAWING', 'DAYS
_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION', 'DAYS_LAST_DUE', 'DAYS_TERMINATION', 'NFLAG
_INSURED_ON_APPROVAL'], dtype='object')
```

In [83]:
```
# Listing down columns which are not needed
Unnecessary_prev = ['WEEKDAY_APPR_PROCESS_START','HOUR_APPR_PROCESS_START','FLAG_LAST

prev_appl.drop(Unnecessary_prev,axis =1, inplace = True)

prev_appl.shape
```

Out[83]:
```
(1670214, 29)
```

In [84]:
```
# IMputing values "Unknown" as this a categorical column
prev_appl["NAME_TYPE_SUITE"] = prev_appl["NAME_TYPE_SUITE"].fillna("Unknown")

null_values(prev_appl)
```

Out[84]:
```
NFLAG_INSURED_ON_APPROVAL    40.30
DAYS_LAST_DUE                40.30
DAYS_LAST_DUE_1ST_VERSION    40.30
DAYS_FIRST_DUE               40.30
DAYS_FIRST_DRAWING           40.30
DAYS_TERMINATION             40.30
AMT_GOODS_PRICE              23.08
AMT_ANNUITY                  22.29
CNT_PAYMENT                  22.29
PRODUCT_COMBINATION           0.02
AMT_CREDIT                    0.00
NAME_CONTRACT_STATUS          0.00
NAME_CASH_LOAN_PURPOSE        0.00
NAME_CONTRACT_TYPE            0.00
AMT_APPLICATION               0.00
NAME_PAYMENT_TYPE             0.00
SK_ID_CURR                    0.00
DAYS_DECISION                 0.00
NAME_GOODS_CATEGORY           0.00
CODE_REJECT_REASON            0.00
NAME_TYPE_SUITE               0.00
```

```
NAME_CLIENT_TYPE              0.00
NAME_PORTFOLIO                0.00
NAME_PRODUCT_TYPE             0.00
CHANNEL_TYPE                  0.00
SELLERPLACE_AREA              0.00
NAME_SELLER_INDUSTRY          0.00
NAME_YIELD_GROUP              0.00
SK_ID_PREV                    0.00
dtype: float64
```

- **There are missing values in columns 'DAYS_FIRST_DUE', 'DAYS_TERMINATION', 'DAYS_FIRST_DRAWING','DAYS_LAST_DUE_1ST_VERSION', 'DAYS_LAST_DUE' and these columns count days thus will keeping null values as they are**

In [85]:

```
#Analying numerical columns using describe

prev_appl[p_null_col_15.index].describe()
```

Out[85]:

	DAYS_FIRST_DUE	DAYS_TERMINATION	DAYS_FIRST_DRAWING	NFLAG_INSURED_ON_APPR
count	997149.000000	997149.000000	997149.000000	997149.00
mean	13826.269337	81992.343838	342209.855039	0.3
std	72444.869708	153303.516729	88916.115834	0.4
min	-2892.000000	-2874.000000	-2922.000000	0.0
25%	-1628.000000	-1270.000000	365243.000000	0.0
50%	-831.000000	-499.000000	365243.000000	0.0
75%	-411.000000	-44.000000	365243.000000	1.0
max	365243.000000	365243.000000	365243.000000	1.0

In [86]:

```
# To convert negative days to postive days creating a varaible "p_days_col"

p_days_col = ['DAYS_DECISION','DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_

prev_appl[p_days_col].describe() # Analysis before conversion
```

Out[86]:

	DAYS_DECISION	DAYS_FIRST_DRAWING	DAYS_FIRST_DUE	DAYS_LAST_DUE_1ST_VERSION
count	1.670214e+06	997149.000000	997149.000000	997149.000000
mean	-8.806797e+02	342209.855039	13826.269337	33767.774054
std	7.790997e+02	88916.115834	72444.869708	106857.034789
min	-2.922000e+03	-2922.000000	-2892.000000	-2801.000000
25%	-1.300000e+03	365243.000000	-1628.000000	-1242.000000
50%	-5.810000e+02	365243.000000	-831.000000	-361.000000
75%	-2.800000e+02	365243.000000	-411.000000	129.000000
max	-1.000000e+00	365243.000000	365243.000000	365243.000000

In [87]:

```
# Converting Negative days to positive days

prev_appl[p_days_col] = abs(prev_appl[p_days_col])

prev_appl[p_null_col_15.index].describe() # analysing after conversion
```

Out[87]:

	DAYS_FIRST_DUE	DAYS_TERMINATION	DAYS_FIRST_DRAWING	NFLAG_INSURED_ON_APPR
count	997149.000000	997149.000000	997149.000000	997149.00
mean	15949.224065	83505.775017	342340.056543	0.3
std	72007.270877	152484.418802	88413.495220	0.4
min	2.000000	2.000000	2.000000	0.0
25%	475.000000	447.000000	365243.000000	0.0
50%	921.000000	1171.000000	365243.000000	0.0
75%	1825.000000	2501.000000	365243.000000	1.0
max	365243.000000	365243.000000	365243.000000	1.0

In [88]:

```
#days group calculation e.g. 369 will be grouped as with in 2 years

bins = [0,1*365,2*365,3*365,4*365,5*365,6*365,7*365,10*365]
slots = ["1","2","3","4","5","6","7","7 above"]
prev_appl['YEARLY_DECISION'] = pd.cut(prev_appl['DAYS_DECISION'],bins,labels=slots)
```

In [89]:

```
prev_appl['YEARLY_DECISION'].value_counts(normalize=True)*100
```

Out[89]:

```
1          34.351287
2          23.056806
3          12.855598
4           7.883181
5           6.128556
7           5.813806
7 above     5.060729
6           4.850037
Name: YEARLY_DECISION, dtype: float64
```

Insight:

- **Almost 35% loan applicatants have applied for a new loan within 1 year of previous loan decision**

In [90]:

```
prev_appl.nunique()
```

Out[90]:

```
SK_ID_PREV                     1670214
SK_ID_CURR                      338857
NAME_CONTRACT_TYPE                   4
AMT_ANNUITY                     357959
AMT_APPLICATION                  93885
AMT_CREDIT                       86803
AMT_GOODS_PRICE                  93885
NAME_CASH_LOAN_PURPOSE              25
NAME_CONTRACT_STATUS                 4
DAYS_DECISION                     2922
NAME_PAYMENT_TYPE                    4
CODE_REJECT_REASON                   9
NAME_TYPE_SUITE                      8
NAME_CLIENT_TYPE                     4
NAME_GOODS_CATEGORY                 28
NAME_PORTFOLIO                       5
NAME_PRODUCT_TYPE                    3
CHANNEL_TYPE                         8
SELLERPLACE_AREA                  2097
NAME_SELLER_INDUSTRY                11
```

```
CNT_PAYMENT                       49
NAME_YIELD_GROUP                   5
PRODUCT_COMBINATION               17
DAYS_FIRST_DRAWING              2838
DAYS_FIRST_DUE                  2892
DAYS_LAST_DUE_1ST_VERSION       2803
DAYS_LAST_DUE                   2873
DAYS_TERMINATION                2830
NFLAG_INSURED_ON_APPROVAL          2
YEARLY_DECISION                    8
dtype: int64
```

In [91]:

```python
prev_appl.nunique()
```

Out[91]:

```
DAYS_TERMINATION             40.30
DAYS_LAST_DUE                40.30
DAYS_LAST_DUE_1ST_VERSION    40.30
DAYS_FIRST_DUE               40.30
DAYS_FIRST_DRAWING           40.30
NFLAG_INSURED_ON_APPROVAL    40.30
AMT_GOODS_PRICE              23.08
AMT_ANNUITY                  22.29
CNT_PAYMENT                  22.29
PRODUCT_COMBINATION           0.02
AMT_CREDIT                    0.00
NAME_CONTRACT_STATUS          0.00
NAME_CASH_LOAN_PURPOSE        0.00
YEARLY_DECISION               0.00
AMT_APPLICATION               0.00
NAME_CONTRACT_TYPE            0.00
NAME_PAYMENT_TYPE             0.00
SK_ID_CURR                    0.00
DAYS_DECISION                 0.00
NAME_GOODS_CATEGORY           0.00
CODE_REJECT_REASON            0.00
NAME_TYPE_SUITE               0.00
NAME_CLIENT_TYPE              0.00
NAME_PORTFOLIO                0.00
NAME_PRODUCT_TYPE             0.00
CHANNEL_TYPE                  0.00
SELLERPLACE_AREA              0.00
NAME_SELLER_INDUSTRY          0.00
NAME_YIELD_GROUP              0.00
SK_ID_PREV                    0.00
dtype: float64
```

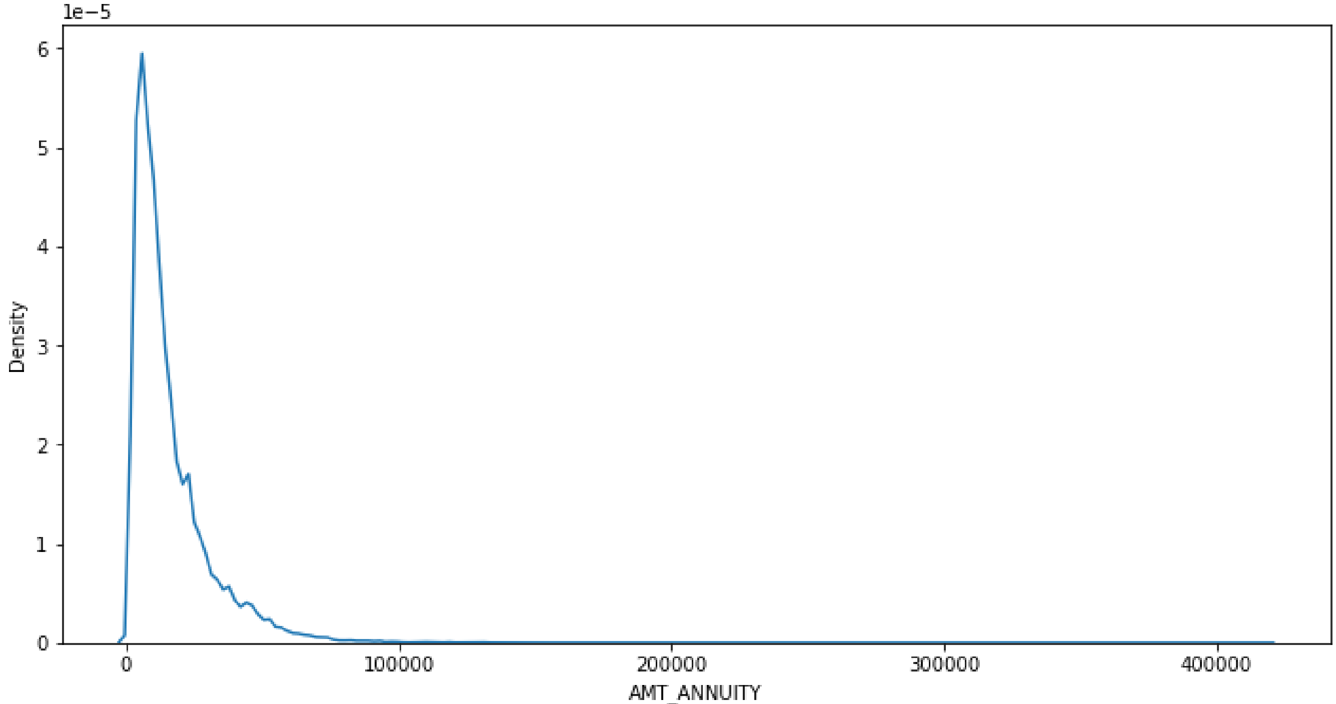
Now dealing with continuos variables "AMT_ANNUITY", "AMT_GOODS_PRICE"

To impute null values in continuous variables, we plotted the distribution of the columns and used

- **median if the distribution is skewed**
- **mode if the distribution pattern is preserved.**

In [92]:

```python
#plotting a kdeplot to understand distribution of "AMT_ANNUITY"

plt.figure(figsize=(12,6))
sns.kdeplot(prev_appl['AMT_ANNUITY'])
plt.show()
```

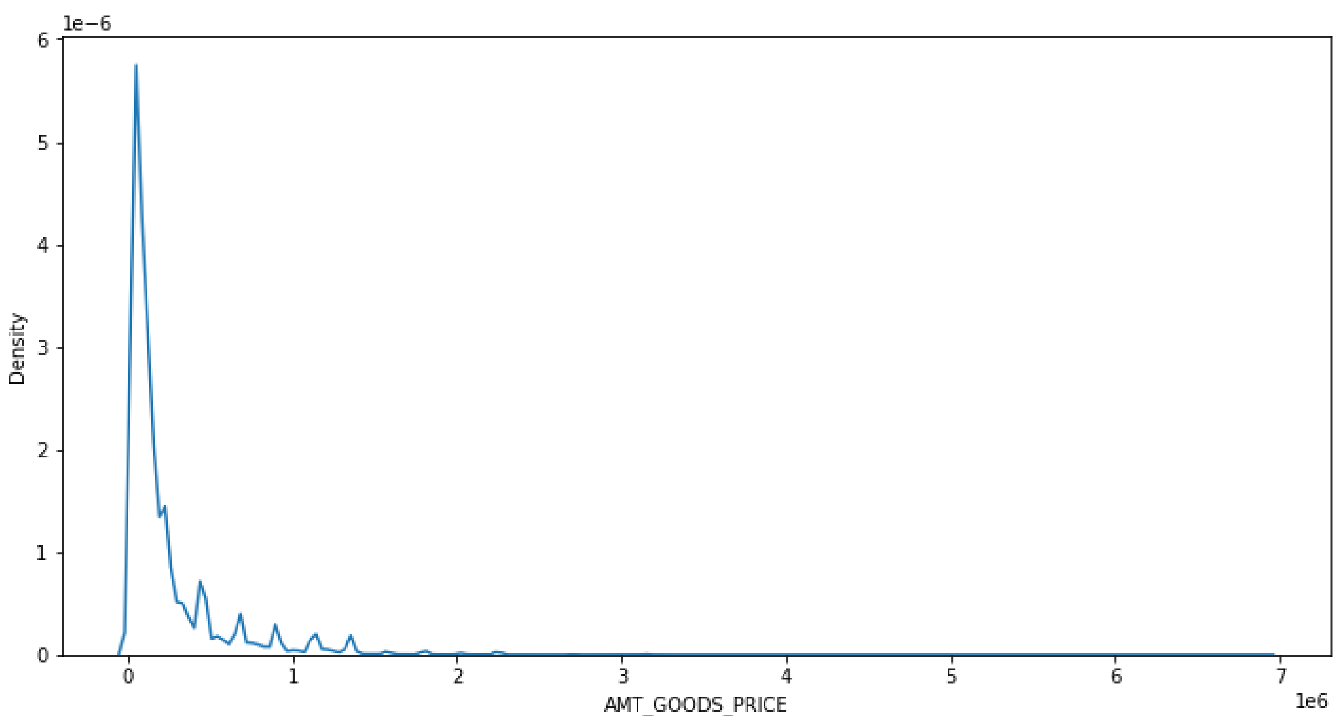


Insight:

- **There is a single peak at the left side of the distribution and it indicates the presence of outliers and hence imputing with mean would not be the right approach and hence imputing with median.**

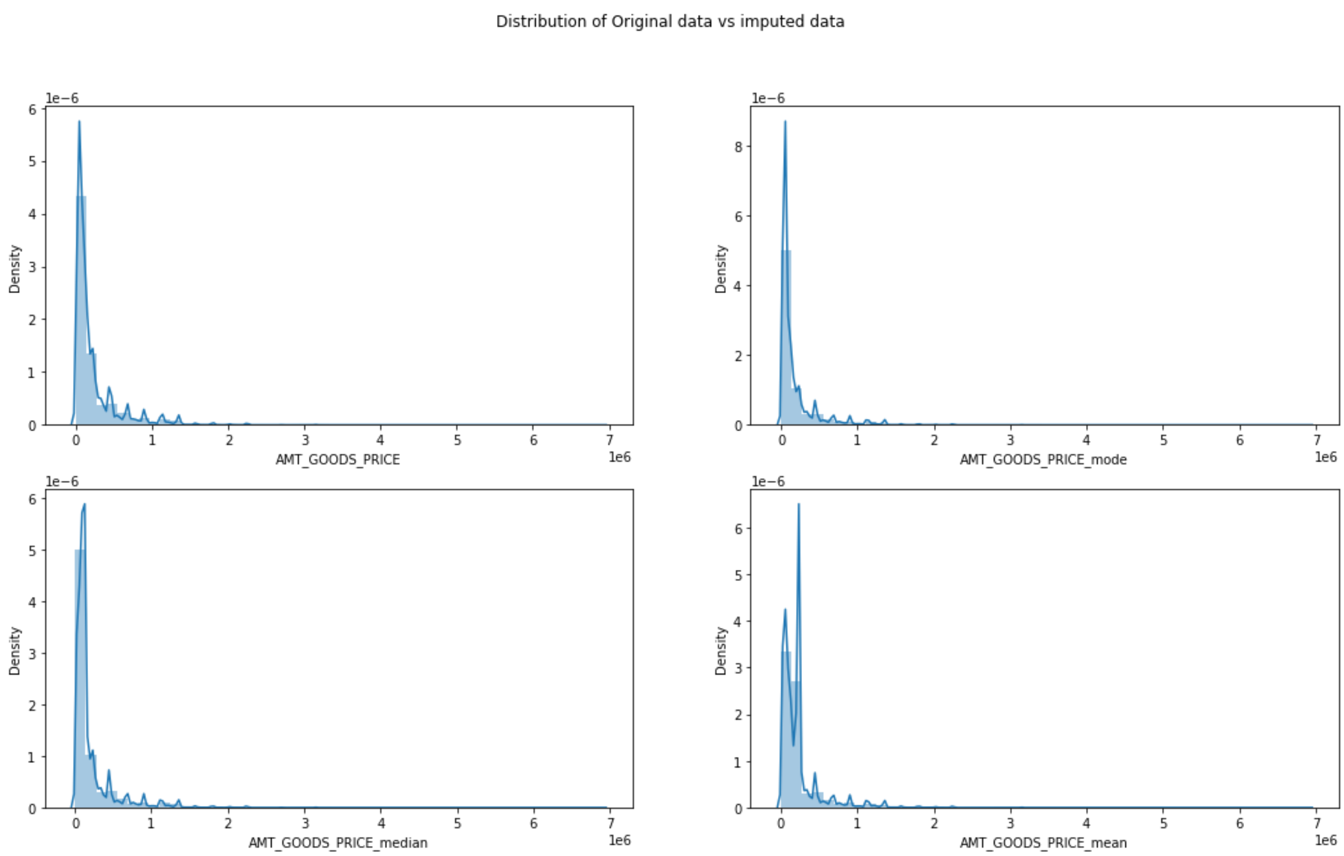
In [93]:
```
#imputing missing values with median

prev_appl['AMT_ANNUITY'].fillna(prev_appl['AMT_ANNUITY'].median(),inplace = True)
```

In [94]:
```
# Plotting kde plot for "AMT_GOODS_PRICE" to understand the distribution

plt.figure(figsize=(12,6))
sns.kdeplot(prev_appl['AMT_GOODS_PRICE'])
plt.show()
```



- **There are several peaks along the distribution. Let's impute using the mode, mean and median and see if the distribution is still about the same.**

In [95]:
```
# Creating new dataframe for "AMT_GOODS_PRICE" with columns imputed with mode, mediar

statsDF = pd.DataFrame()
statsDF['AMT_GOODS_PRICE_mode'] = prev_appl['AMT_GOODS_PRICE'].fillna(prev_appl['AMT_
statsDF['AMT_GOODS_PRICE_median'] = prev_appl['AMT_GOODS_PRICE'].fillna(prev_appl['AM
statsDF['AMT_GOODS_PRICE_mean'] = prev_appl['AMT_GOODS_PRICE'].fillna(prev_appl['AMT_

cols = ['AMT_GOODS_PRICE_mode', 'AMT_GOODS_PRICE_median','AMT_GOODS_PRICE_mean']

plt.figure(figsize=(18,10))
plt.suptitle('Distribution of Original data vs imputed data')
plt.subplot(221)
sns.distplot(prev_appl['AMT_GOODS_PRICE'][pd.notnull(prev_appl['AMT_GOODS_PRICE'])]);
for i in enumerate(cols):
    plt.subplot(2,2,i[0]+2)
    sns.distplot(statsDF[i[1]])
```



- **The original distribution is closer with the distribution of data imputed with mode in this case, thus will impute mode for missing values**

In [96]:
```
# Imputing null values with mode

prev_appl['AMT_GOODS_PRICE'].fillna(prev_appl['AMT_GOODS_PRICE'].mode()[0], inplace=T
```

Imputing CNT_PAYMENT with 0 as the NAME_CONTRACT_STATUS for these indicate that most of these loans were not started:

In [97]:
```
#taking out values count for NAME_CONTRACT_STATUS categories where CNT_PAYMENT have r
```

```
prev_appl.loc[prev_appl['CNT_PAYMENT'].isnull(),'NAME_CONTRACT_STATUS'].value_counts(
```

Out[97]:

```
Canceled        305805
Refused          40897
Unused offer     25524
Approved             4
Name: NAME_CONTRACT_STATUS, dtype: int64
```

In [98]:

```
#imputing null values as 0

prev_appl['CNT_PAYMENT'].fillna(0,inplace = True)
```

In [99]:

```
prev_appl.columns
```

Out[99]:

```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_ANNUITY', 'AMT_APPLICAT
ION', 'AMT_CREDIT', 'AMT_GOODS_PRICE', 'NAME_CASH_LOAN_PURPOSE', 'NAME_CONTRACT_STATU
S', 'DAYS_DECISION', 'NAME_PAYMENT_TYPE', 'CODE_REJECT_REASON', 'NAME_TYPE_SUITE', 'N
AME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO', 'NAME_PRODUCT_TYPE', 'CHAN
NEL_TYPE', 'SELLERPLACE_AREA', 'NAME_SELLER_INDUSTRY', 'CNT_PAYMENT', 'NAME_YIELD_GRO
UP', 'PRODUCT_COMBINATION', 'DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1S
T_VERSION', 'DAYS_LAST_DUE', 'DAYS_TERMINATION', 'NFLAG_INSURED_ON_APPROVAL', 'YEARLY
_DECISION'], dtype='object')
```

In [100]:

```
#Converting required categoical columns from Object to categorical

p_catgorical_col = ['NAME_CASH_LOAN_PURPOSE','NAME_CONTRACT_STATUS','NAME_PAYMENT_TYF
                     'CODE_REJECT_REASON','NAME_CLIENT_TYPE','NAME_GOODS_CATEGORY','NA
                    'NAME_PRODUCT_TYPE','CHANNEL_TYPE','NAME_SELLER_INDUSTRY','NAME_YI
                     'NAME_CONTRACT_TYPE']

for col in p_catgorical_col:
    prev_appl[col] =pd.Categorical(prev_appl[col])
```

Finding outliers

In [101]:

```
prev_appl.describe()
```

Out[101]:

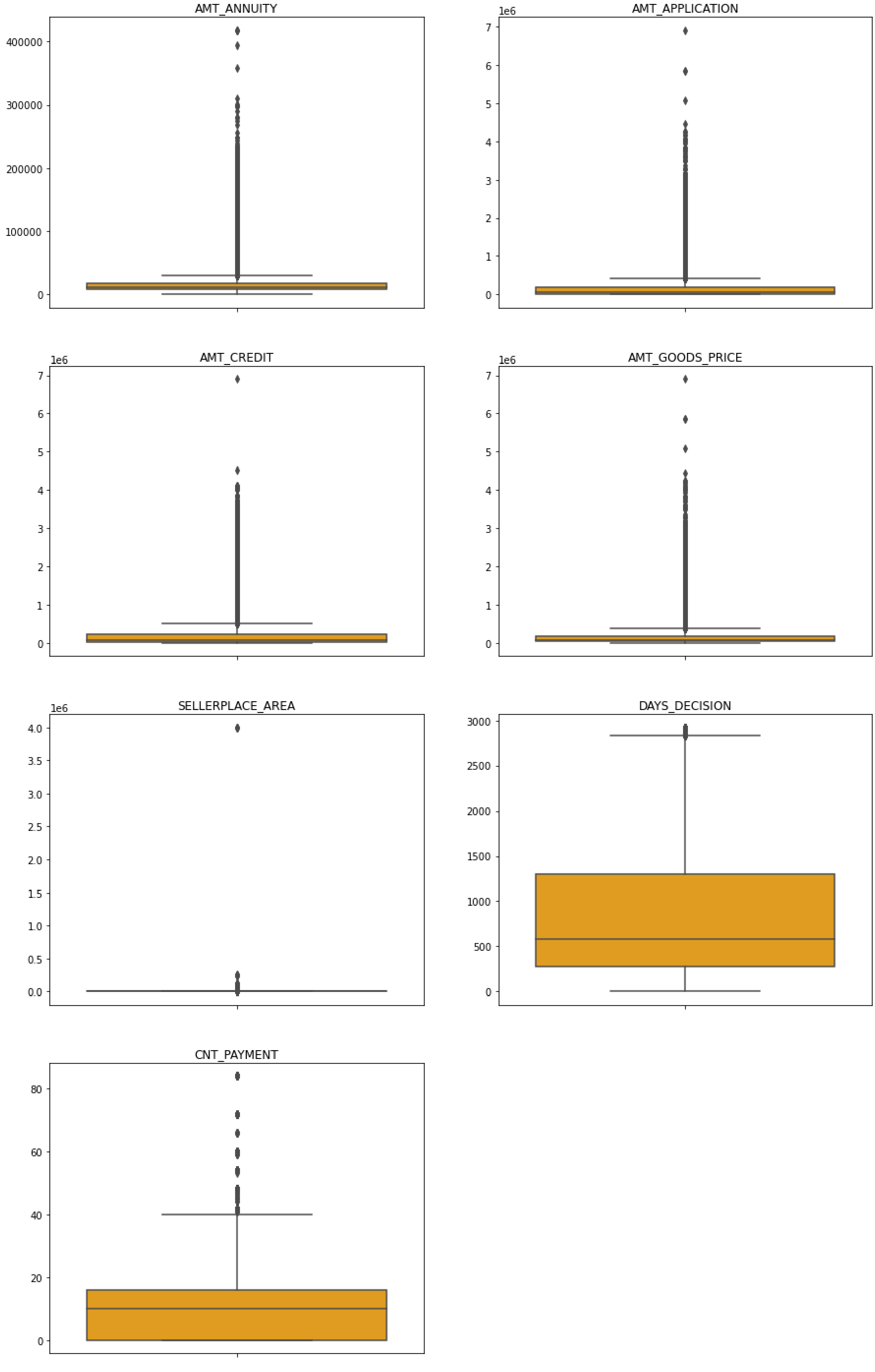
	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_GOO
count	1.670214e+06	1.670214e+06	1.670214e+06	1.670214e+06	1.670213e+06	1.67
mean	1.923089e+06	2.783572e+05	1.490651e+04	1.752339e+05	1.961140e+05	1.85
std	5.325980e+05	1.028148e+05	1.317751e+04	2.927798e+05	3.185746e+05	2.87
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00	0.000000e+00	0.00
25%	1.461857e+06	1.893290e+05	7.547096e+03	1.872000e+04	2.416050e+04	4.50
50%	1.923110e+06	2.787145e+05	1.125000e+04	7.104600e+04	8.054100e+04	7.10
75%	2.384280e+06	3.675140e+05	1.682403e+04	1.803600e+05	2.164185e+05	1.80
max	2.845382e+06	4.562550e+05	4.180581e+05	6.905160e+06	6.905160e+06	6.90

- **from describe we could find all the columns those wo have high difference between max and 75 percentile and the ones which makes no sense having max value to be so high are captured below**

In [102]:

```
p_outlier_col = ['AMT_ANNUITY','AMT_APPLICATION','AMT_CREDIT','AMT_GOODS_PRICE',
```

```
                 'SELLERPLACE_AREA','DAYS_DECISION','CNT_PAYMENT']

plt.figure(figsize=[15,25])
for i,j in itertools.zip_longest(p_outlier_col, range(len(p_outlier_col))):
    plt.subplot(4,2,j+1)
    sns.boxplot(y = prev_appl[i], orient = "h", color = "orange")
    #plt.yticks(fontsize=8)
    plt.xlabel("")
    plt.ylabel("")
    plt.title(i)
```



AMT_ANNUITY
0
100000
200000
300000
400000
AMT_APPLICATION
1e6
0
1
2
3
4
5
6
7
AMT_CREDIT
1e6
0
1
2
3
4
5
6
7
AMT_GOODS_PRICE
1e6
0
1
2
3
4
5
6
7
SELLERPLACE_AREA
1e6
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
DAYS_DECISION
0
500
1000
1500
2000
2500
3000
CNT_PAYMENT
0
20
40
60
80

Insight:

It can be seen that in previous application data

- **AMT_ANNUITY, AMT_APPLICATION, AMT_CREDIT, AMT_GOODS_PRICE, SELLERPLACE_AREA have huge number of outliers.**
- **CNT_PAYMENT has few outlier values.**
- **DAYS_DECISION has little number of outliers indicating that these previous applications decisions were taken long back.**

Data Analysis Time

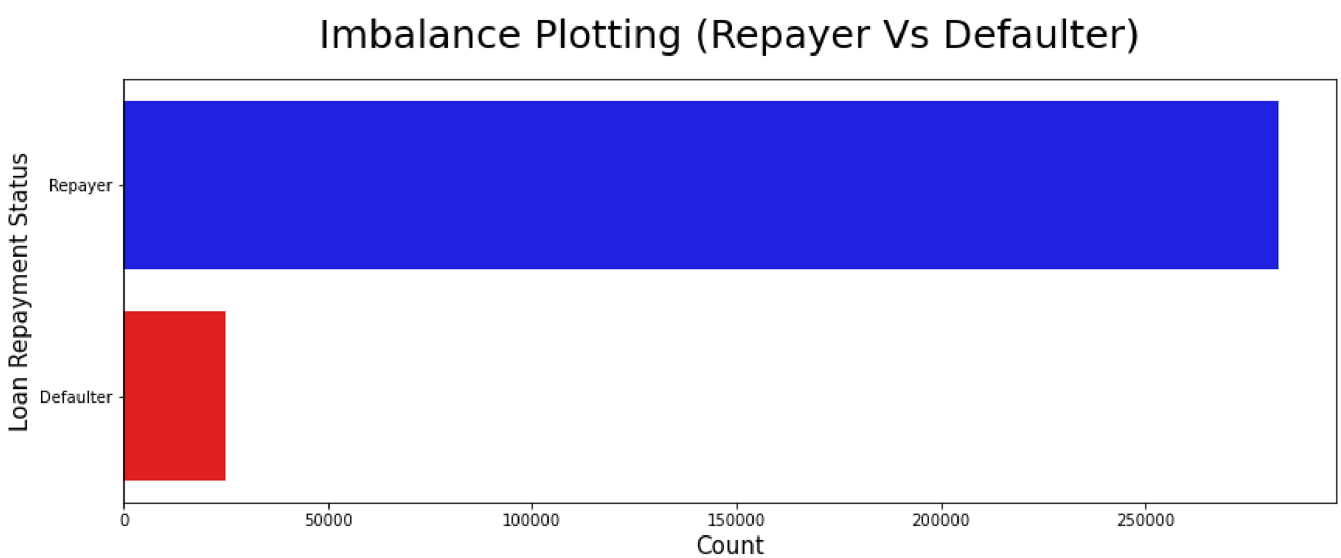
Strategy: The data analysis flow has been planned in following way :

- Imbalance in Data
- Categorical Data Analysis
- Categorical segmented Univariate Analysis
- Categorical Bi/Multivariate analysis
- Numeric Data Analysis
- Bi-furcation of databased based on TARGET data
- Correlation Matrix
- Numerical segmented Univariate Analysis
- Numerical Bi/Multivariate analysis

Imbalance Data

In [103]:

```
plt.figure(figsize= [14,5])
sns.barplot(y=["Repayer","Defaulter"], x = appl_data["TARGET"].value_counts(), palett
plt.ylabel("Loan Repayment Status",fontdict = {"fontsize":15})
plt.xlabel("Count",fontdict = {"fontsize":15})
plt.title("Imbalance Plotting (Repayer Vs Defaulter)", fontdict = {"fontsize":25}, pa
plt.show()
```



In [104]:

```
#Ratio of imbalance percentage with respect to defaulter and repayer is given below
repayer = round((appl_data["TARGET"].value_counts()[0]/len(appl_data)* 100),2)
print("Repayer Percentage is {}%".format(repayer))
defaluter = round((appl_data["TARGET"].value_counts()[1]/len(appl_data)* 100),2)
print("Defaulter Percentage is {}%".format(defaluter))
print("Imbalance Ratio with respect to Repayer and Defaulter is given: {0:.2f}/1 (app
```

```
Repayer Percentage is 91.93%
Defaulter Percentage is 8.07%
Imbalance Ratio with respect to Repayer and Defaulter is given: 11.39/1 (approx)
```

Plotting Functions

Important Function for Univariate analysis

Creating a function for plotting Variables to do univariate analysis. This function will create two plots

1. Count plot of given column w.r.t TARGET column
2. Percentage of defaulters within that column

The function is taking 6 arguments

1. dataset : to put the dataset we want to use
2. col : column name for which we need to the analysis
3. target_col : column name for with which we will be comparing
4. ylog : to have y-axis in log10 terms, in case the plot is not readable
5. x_label_angle : to maintain the orientation of x-axis labels
6. h_layout : to give horizontal layout of the subplots

In [106]:

```
# Creating a function to find if the column is categorical or numerical

def data_type(dataset,col):
    if dataset[col].dtype == np.int64 or dataset[col].dtype == np.float64:
        return "numerical"
    if dataset[col].dtype == "category":
        return "categorical"

# Creating a function "univariate" to perform analysis one single variable with respe

def univariate(dataset,col,target_col,ylog=False,x_label_angle=False,h_layout=True):
    if data_type(dataset,col) == "numerical":
        sns.distplot(dataset[col],hist=False)


    elif data_type(dataset,col) == "categorical":
        val_count = dataset[col].value_counts()
        df1 = pd.DataFrame({col: val_count.index,'count': val_count.values})


        target_1_percentage = dataset[[col, target_col]].groupby([col],as_index=False
        target_1_percentage[target_col] = target_1_percentage[target_col]*100
        target_1_percentage.sort_values(by=target_col,inplace = True)


# If the plot is not readable, use the log scale

        if(h_layout):
            fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(15,7))
        else:
            fig, (ax1, ax2) = plt.subplots(nrows=2, figsize=(25,35))


# 1. Subplot 1: Count plot of the column

        s = sns.countplot(ax=ax1, x=col, data=dataset, hue=target_col)
```

```
        ax1.set_title(col, fontsize = 20)
        ax1.legend(['Repayer','Defaulter'])
        ax1.set_xlabel(col,fontdict={'fontsize' : 15, 'fontweight' : 3})

        if(x_label_angle):
            s.set_xticklabels(s.get_xticklabels(),rotation=75)

# 2. Subplot 2: Percentage of defaulters within the column

        s = sns.barplot(ax=ax2, x = col, y=target_col, data=target_1_percentage)
        ax2.set_title("Defaulters % in "+col, fontsize = 20)
        ax2.set_xlabel(col,fontdict={'fontsize' : 15, 'fontweight' : 3})
        ax2.set_ylabel(target_col,fontdict={'fontsize' : 15, 'fontweight' : 3})

        if(x_label_angle):
            s.set_xticklabels(s.get_xticklabels(),rotation=75)


# If the plot is not readable, use the log scale

        if ylog:
            ax1.set_yscale('log')
            ax1.set_ylabel("Count (log)",fontdict={'fontsize' : 15, 'fontweight' : 3}
        else:
            ax1.set_ylabel("Count",fontdict={'fontsize' : 15, 'fontweight' : 3})


        plt.show()
```

In [107]:

```
# function for plotting repetitive rel plots in bivaritae numerical analysis

def bivariate_n(x,y,df,hue,kind,labels):
    plt.figure(figsize=[15,15])
    sns.relplot(x=x, y=y, data=df, hue=hue,kind=kind,legend = False)
    plt.legend(labels=labels)
    plt.xticks(rotation=45, ha='right')
    plt.show()
```

In [108]:

```
# function for plotting repetitive barplots in bivariate categorical analysis

def bivariate_c(x,y,df,hue,figsize,labels):

    plt.figure(figsize=figsize)
    sns.barplot(x=x,y=y,data=df, hue=hue)

    # Defining aesthetics of Labels and Title of the plot using style dictionaries
    plt.xlabel(x,fontsize = 15)
    plt.ylabel(y,fontsize = 15)
    plt.title(col,fontsize = 20)
    plt.xticks(rotation=45, ha='right')
    plt.legend(labels = labels )
    plt.show()
```

In [109]:

```
#function for plotting repetitive countplots in univariate categorical analysis on th

def univariate_c_merged(col,df,hue,palette,ylog,figsize):
    plt.figure(figsize=figsize)
    ax=sns.countplot(x=col, data=df,hue= hue,palette= palette,order=df[col].value_cou
```

```python
    if ylog:
        plt.yscale('log')
        plt.ylabel("Count (log)",fontsize=15)
    else:
        plt.ylabel("Count",fontsize=15)

    plt.title(col , fontsize=20)
    plt.legend(loc = "upper right")
    plt.xticks(rotation=45, ha='right')

    plt.show()
```

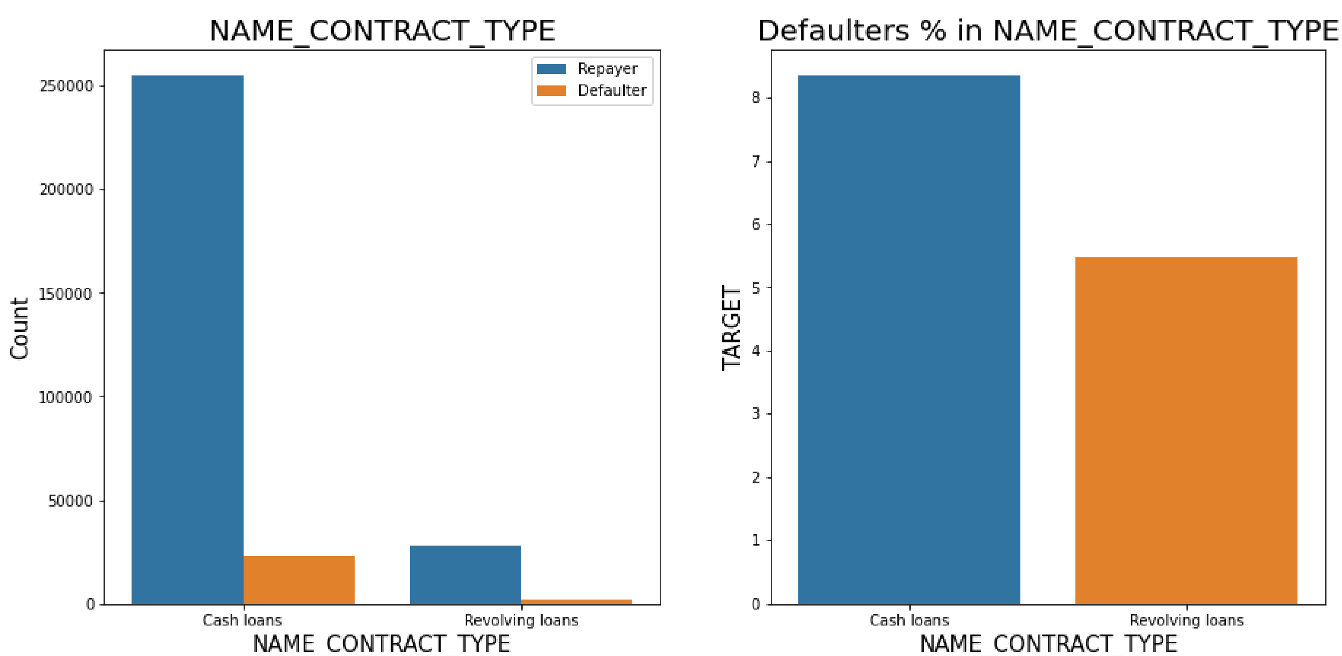
In [110]:
```python
# Function to plot point plots

def pointplot(df,hue,x,y):
    plt.figure(figsize=(12,6))
    sns.pointplot(x=x, y=y, hue=hue, data=df)
    plt.title(x+" VS "+y,fontsize = 15)
```

In [111]:
```python
# storing numnercial and categorical columns as list in belows varibles

cat_col = list(appl_data.select_dtypes(["category"]).columns)   # Categorical columns
num_col = list(appl_data.select_dtypes(["int","float"]).columns)  #N Numerical Colum
```

Categorical Variables Analysis

Segmented Univariate Analysis

In [112]:
```python
#1 Checking the contract type based on loan repayment status
univariate(appl_data,"NAME_CONTRACT_TYPE","TARGET",False,False,True)
```

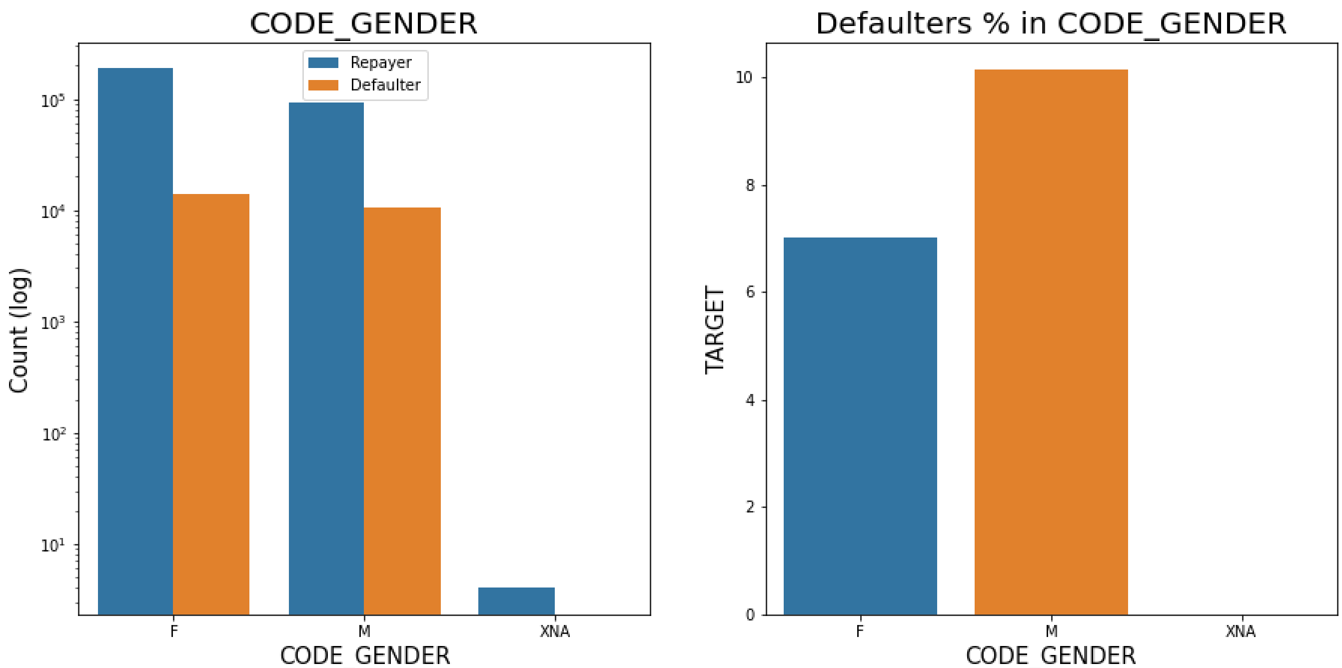


Inferences: Contract type

- **Revolving loans are just a small fraction (10%) from the total number of loans**

- **Around 8-9% Cash loan applicants and 5-6% Revolving loan applicant are in defaulters**

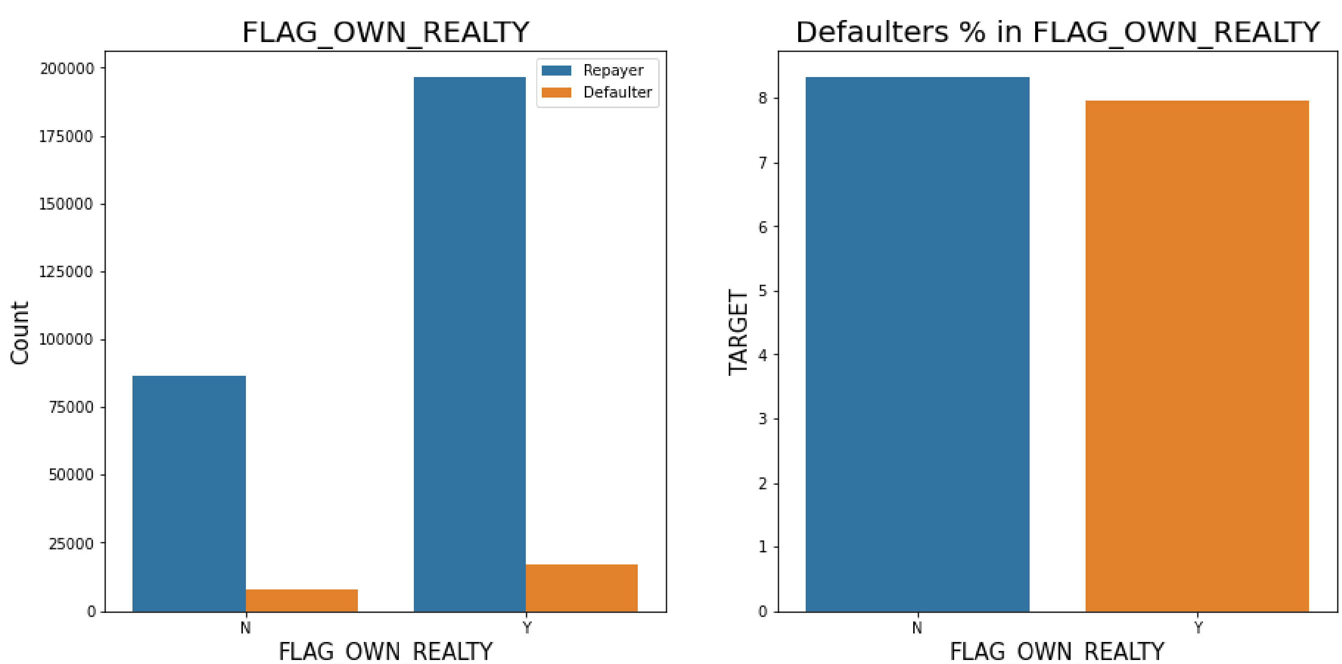
In [113]:
```
#2 Checking the type of Gender on loan repayment status
univariate(appl_data,"CODE_GENDER","TARGET",True,False,True)
```



Inferences: Gender Type

- **The number of female clients is almost double the number of male clients.**
- **Based on the percentage of defaulted credits, males have a higher chance of not returning their loans about 10%, comparing with women about 7%**

In [114]:
```
#3 Checking if owning a real estate is related to loan repayment status
univariate(appl_data,"FLAG_OWN_REALTY","TARGET",False,False,True)
```

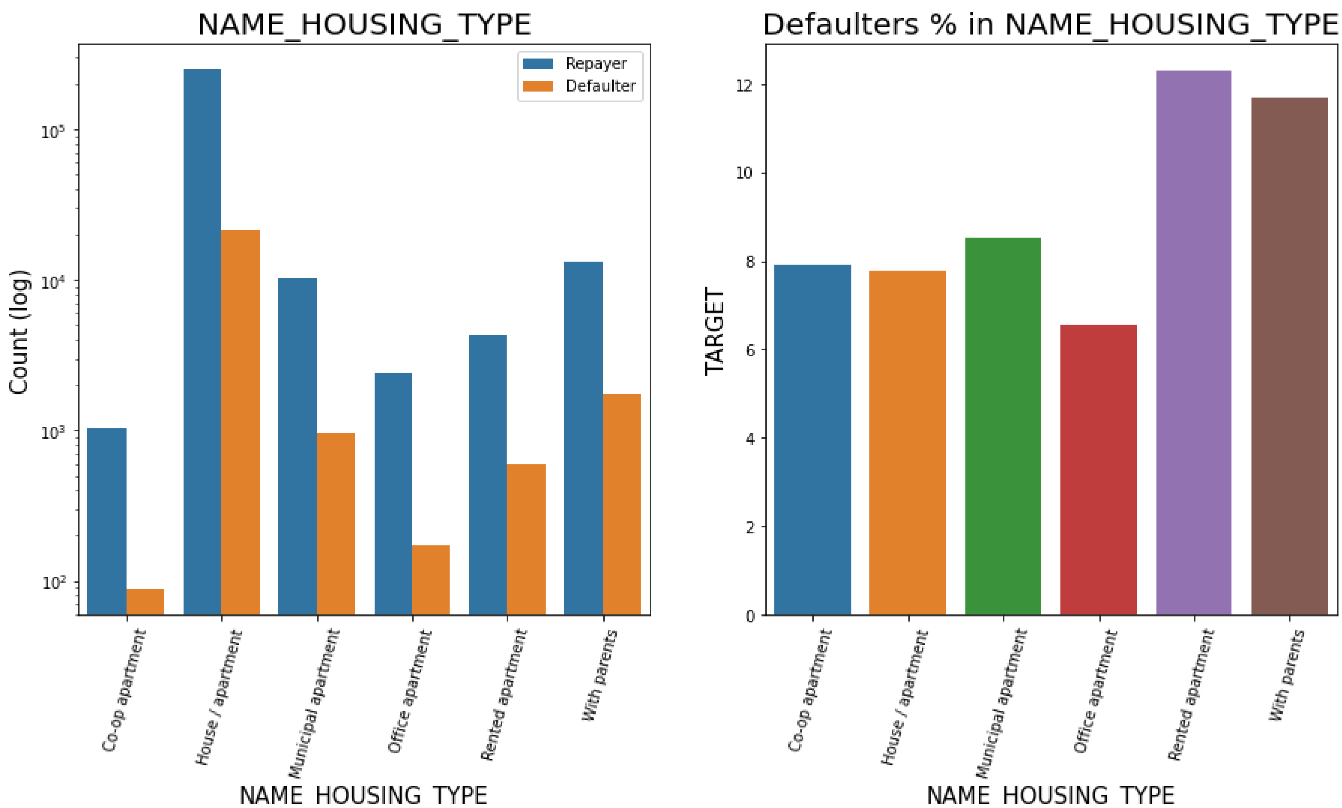


Inferences:

- **The clients who own real estate are more than double of the ones that don't own.**
- **The defaulting rate of both categories are around the same (~8%). Thus we can infer that there is no correlation between owning a reality and defaulting the loan.**

In [115]:

```
#4 Analyzing Housing Type based on loan repayment status
univariate(appl_data,"NAME_HOUSING_TYPE","TARGET",True,True,True)
```

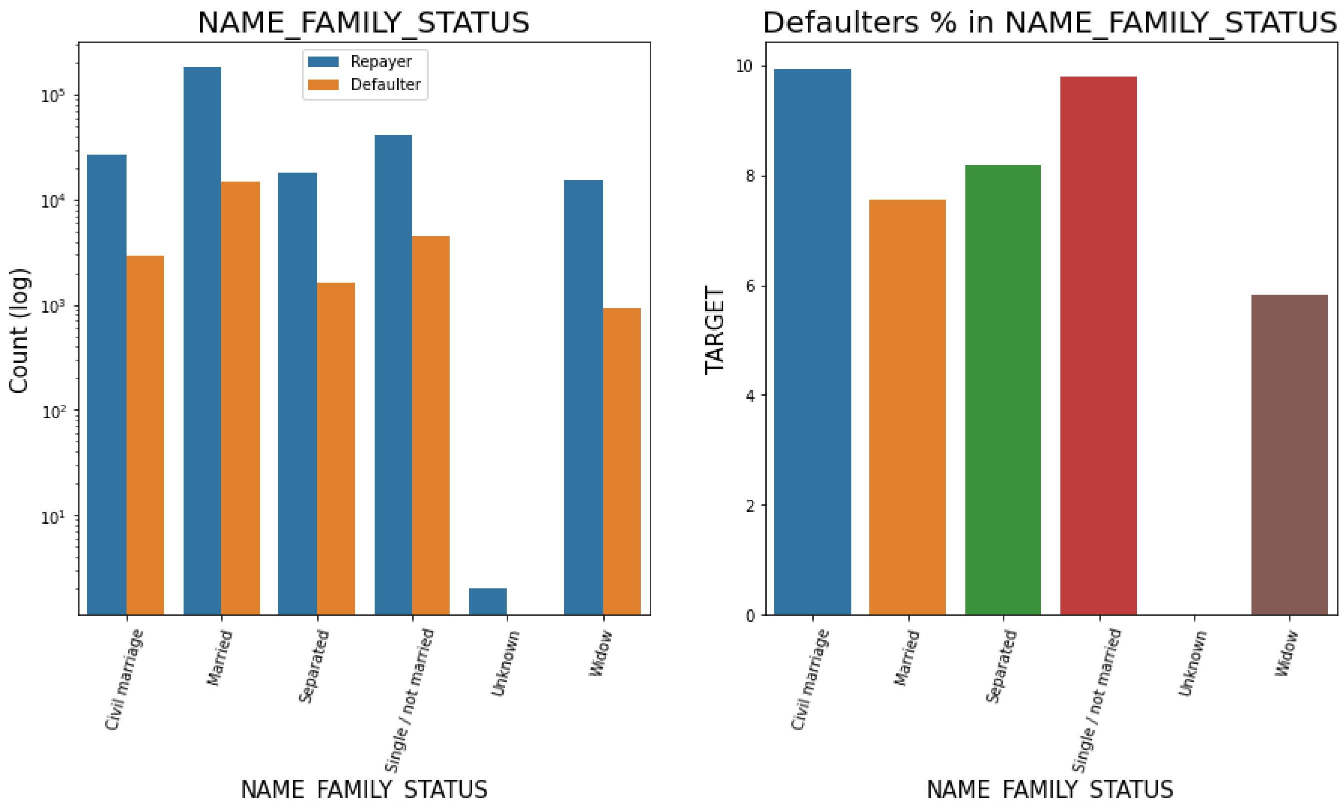


Inferences: Applicant House type

- **Majority of people live in House/apartment**
- **People living in office apartments have lowest default rate**
- **People living with parents (~11.5%) and living in rented apartments(>12%) have higher probability of defaulting**

In [116]:

```
#5 Analyzing Family status based on loan repayment status
univariate(appl_data,"NAME_FAMILY_STATUS","TARGET",True,True,True)
```

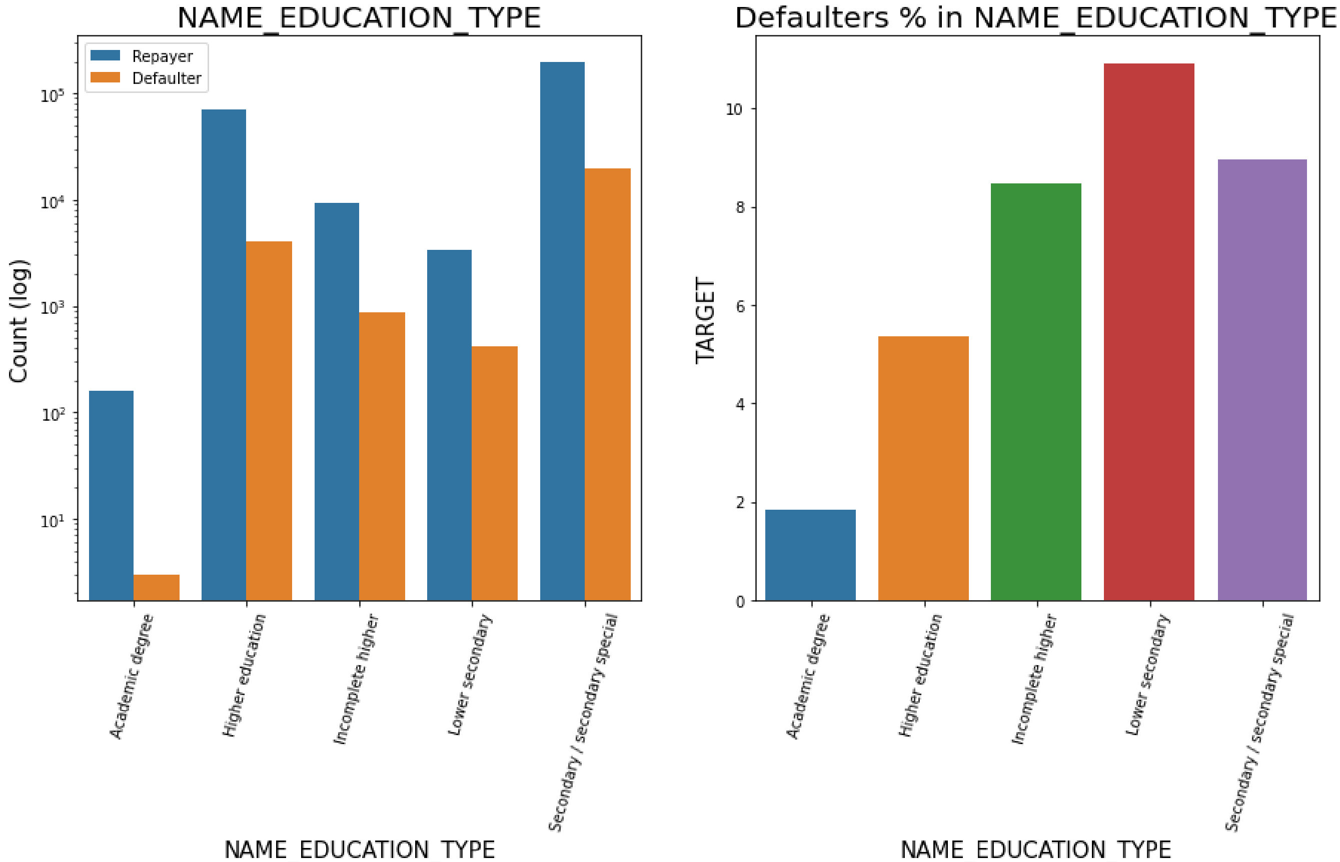


Inferences:

- **Most of the people who have taken loan are married, followed by Single/not married and civil marriage**
- **In Percentage of defaulters, Civil marriage has the highest percent around (10%) and widow has the lowest around 6% (exception being Unknown).**

In [117]:

```
#6 Analyzing Education Type based on loan repayment status
univariate(appl_data,"NAME_EDUCATION_TYPE","TARGET",True,True,True)
```

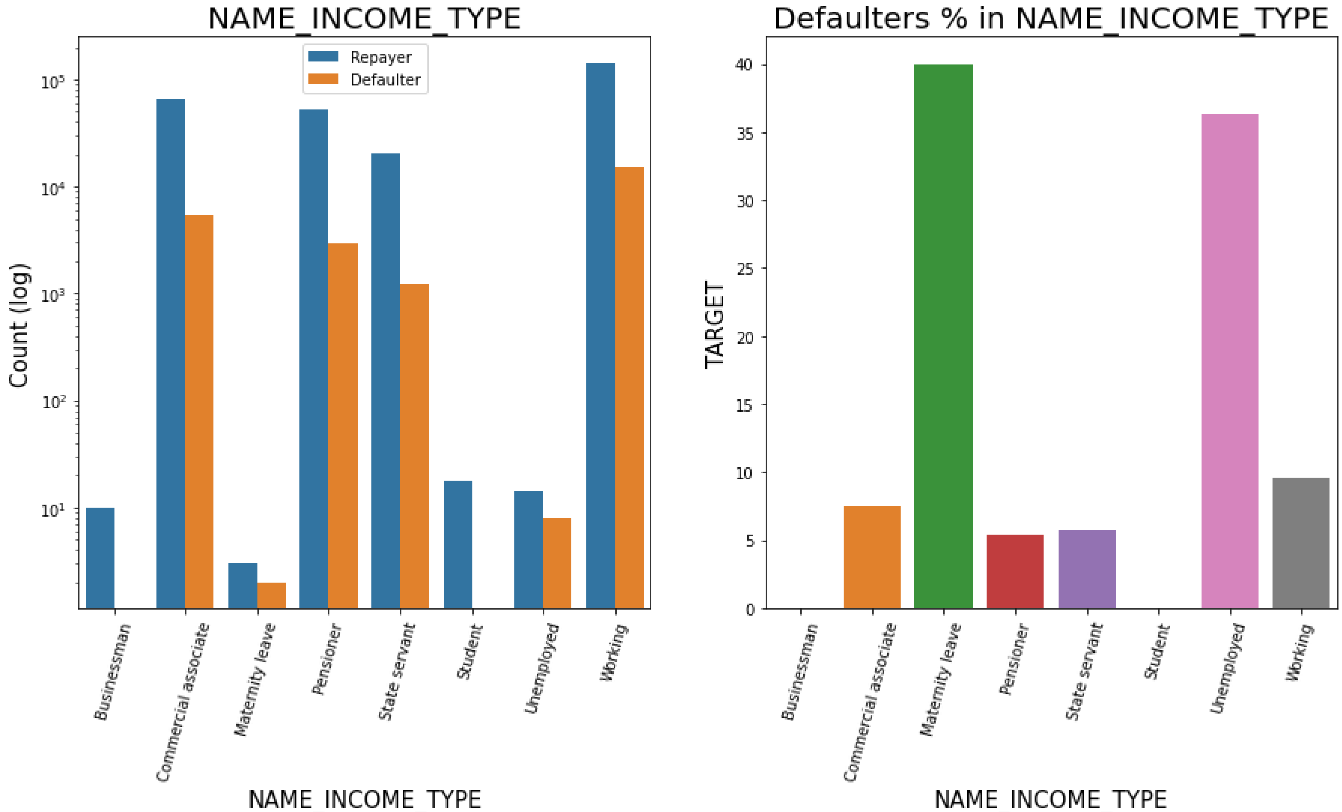


Inferences: Education Type

- **Majority of clients have Secondary/secondary special education, followed by clients with Higher education.**
- **Very few clients have an academic degree**
- **Lower secondary category have highest rate of defaulting around 11%.**
- **People with Academic degree are least likely to default.**

In [118]:

```
#7 Analyzing Income Type based on loan repayment status
univariate(appl_data,"NAME_INCOME_TYPE","TARGET",True,True,True)
```

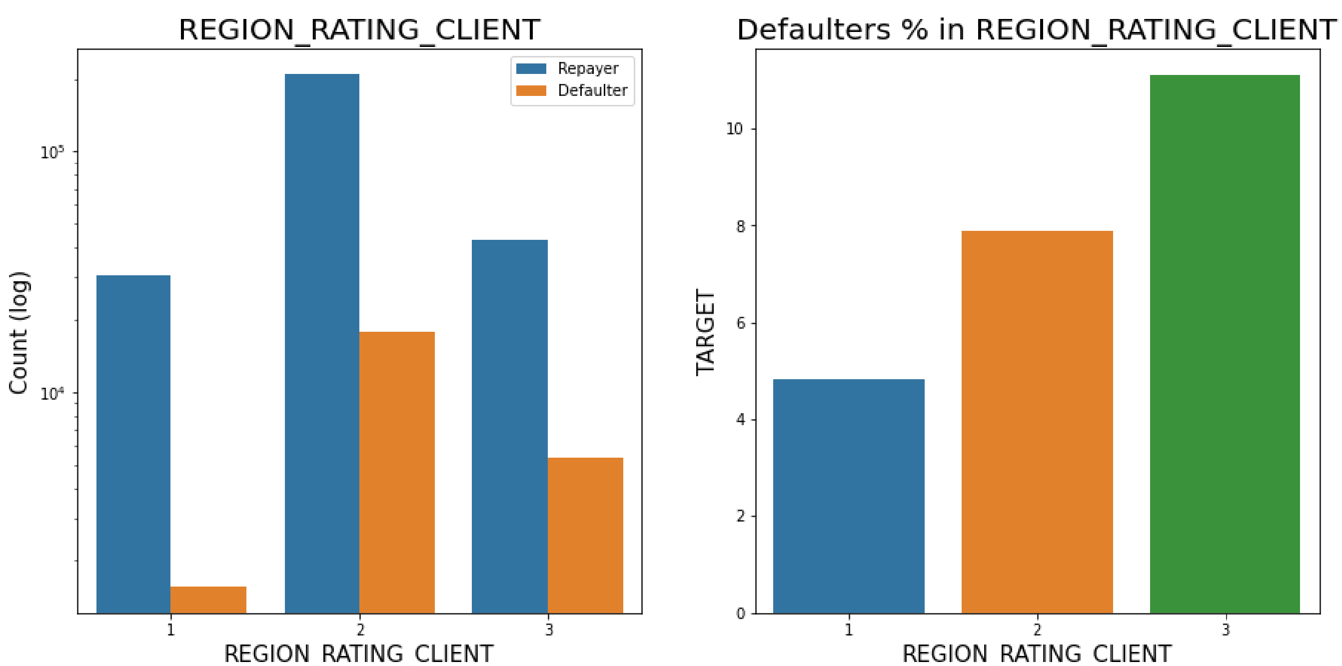


Inferences:

- **Most of applicants for loans income type is Working, followed by Commercial associate, Pensioner and State servant.**
- **The applicants who are on Maternity leave have defaulting percentage of 40% which is the highest, followed by Unemployed (37%). The rest under average around 10% defaultees.**
- **Student and Businessmen though less in numbers, do not have default record. Safest two categories for providing loan.**

In [119]:

```
#8 Analyzing Region rating where applicant lives based on loan repayment status
univariate(appl_data,"REGION_RATING_CLIENT","TARGET",True,False,True)
```

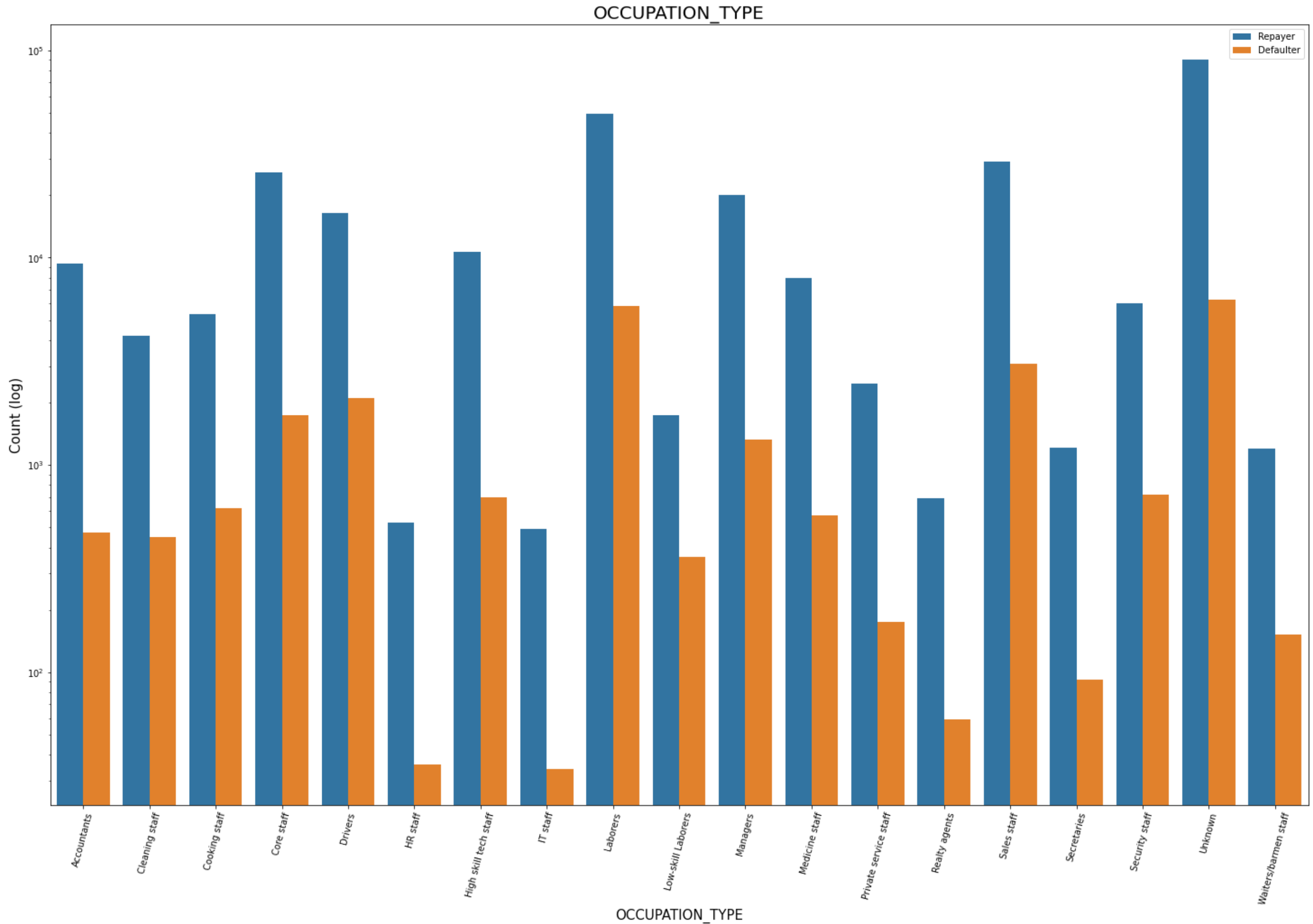


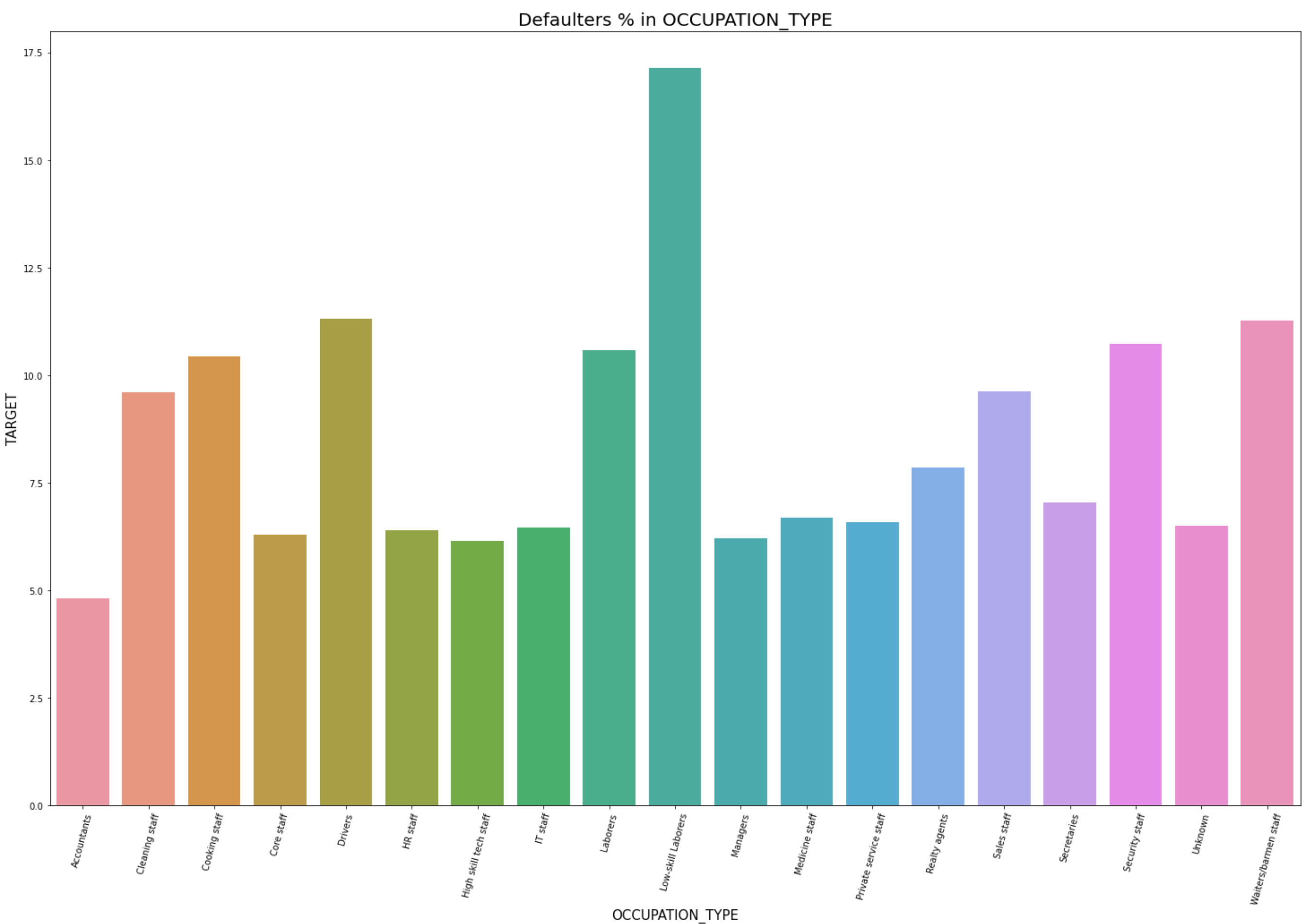
Inferences:Client Region Rating

- **Most of the applicants are living in Region with Rating 2 place.**
- **Region Rating 3 has the highest default rate (11%)**
- **Applicant living in Region_Rating 1 has the lowest probability of defaulting, thus safer for approving loans**

In [120]:

```
#9 Analyzing Occupation Type where applicant lives based on loan repayment status
univariate(appl_data,"OCCUPATION_TYPE","TARGET",True,True,False)
```





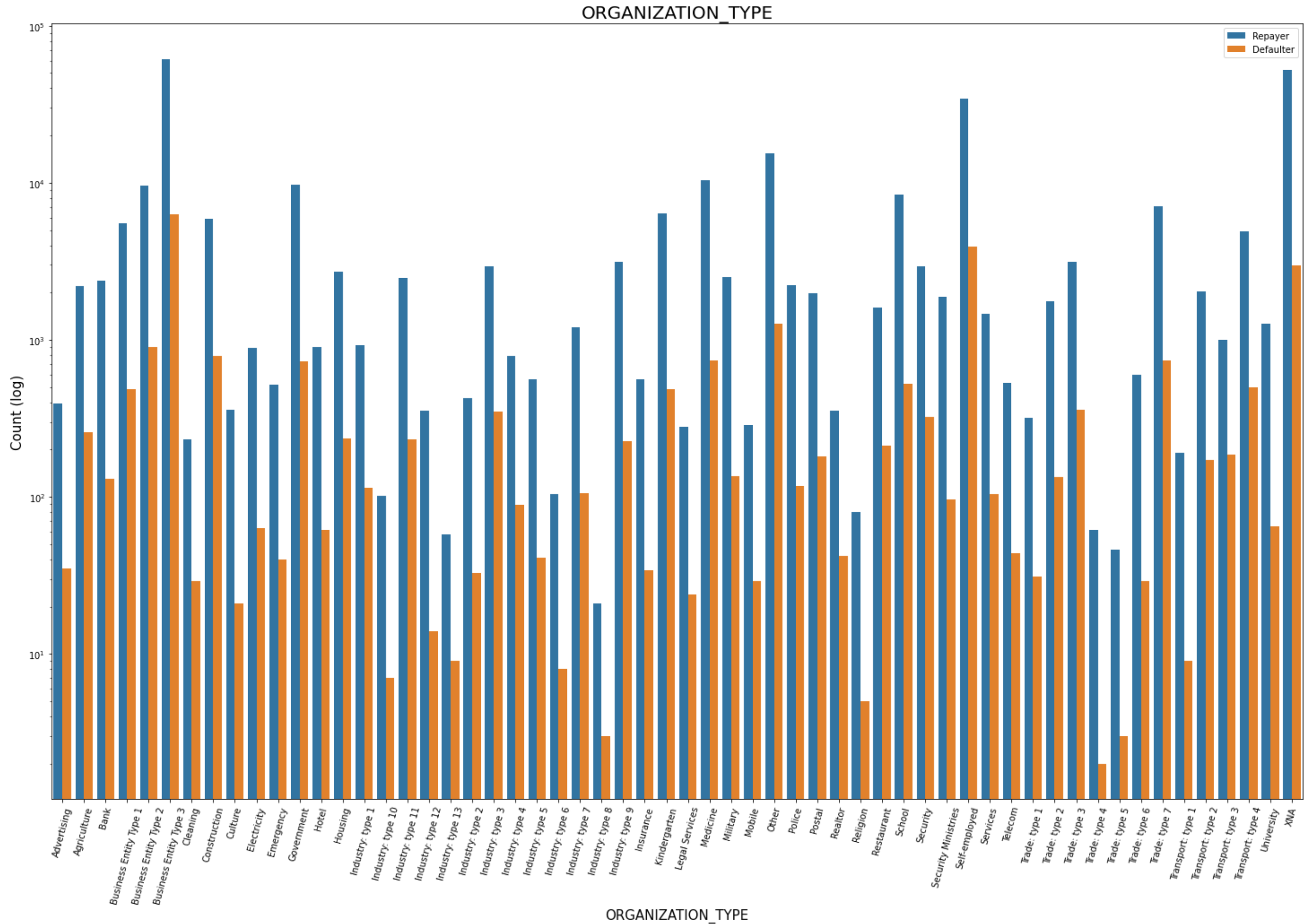
Inferences:

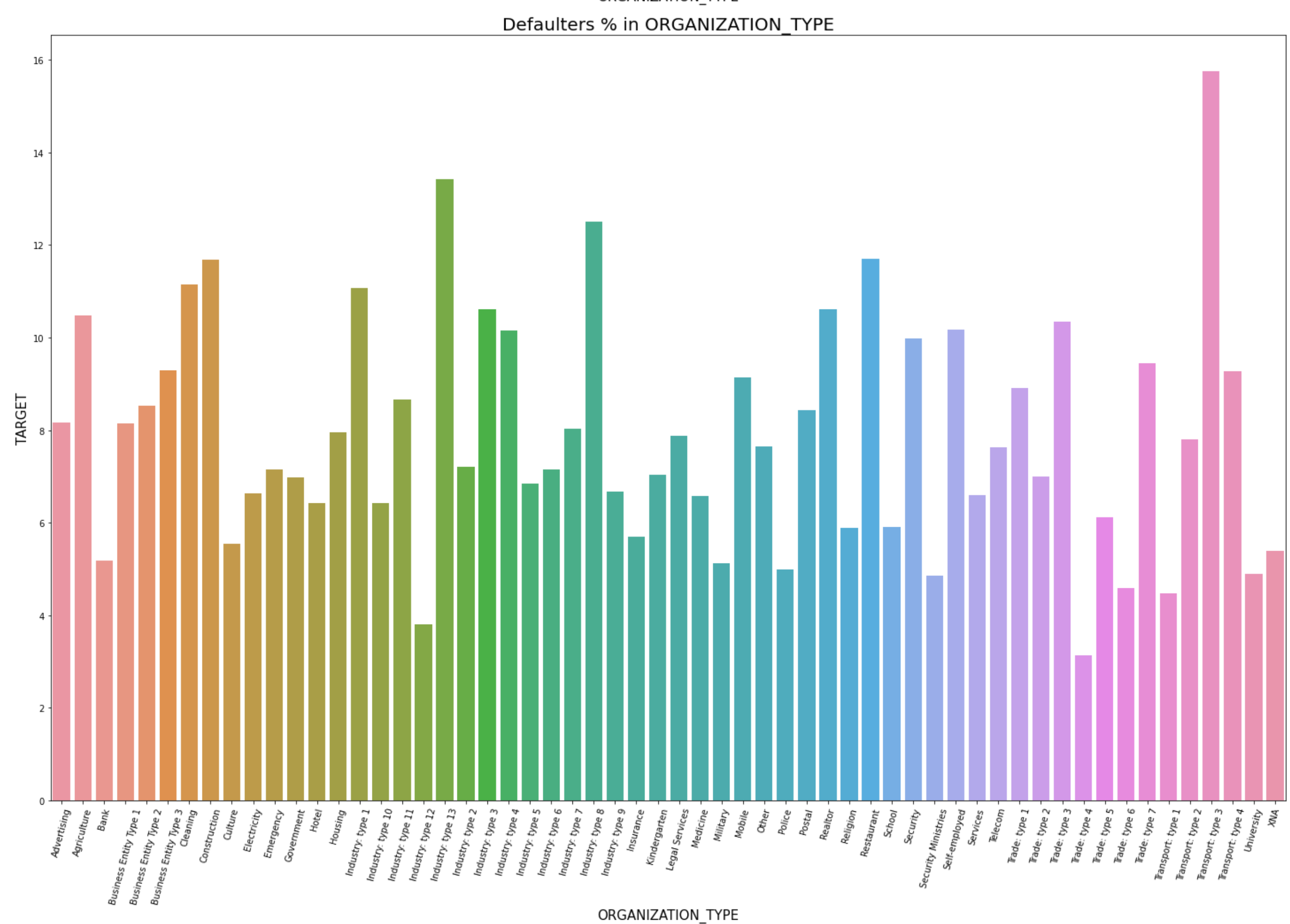
- **Most of the loans are taken by Laborers, followed by Sales staff.**
- **IT staff are less likely to apply for Loan.**

- **Category with highest percent of defautess are Low-skill Laborers (above 17%), followed by Drivers and Waiters/barmen staff, Security staff, Laborers and Cooking staff**

In [121]:

```
#10 Checking Loan repayment status based on Organization type

univariate(appl_data,"ORGANIZATION_TYPE","TARGET",True,True,False)
```





Inferences: Organization Type

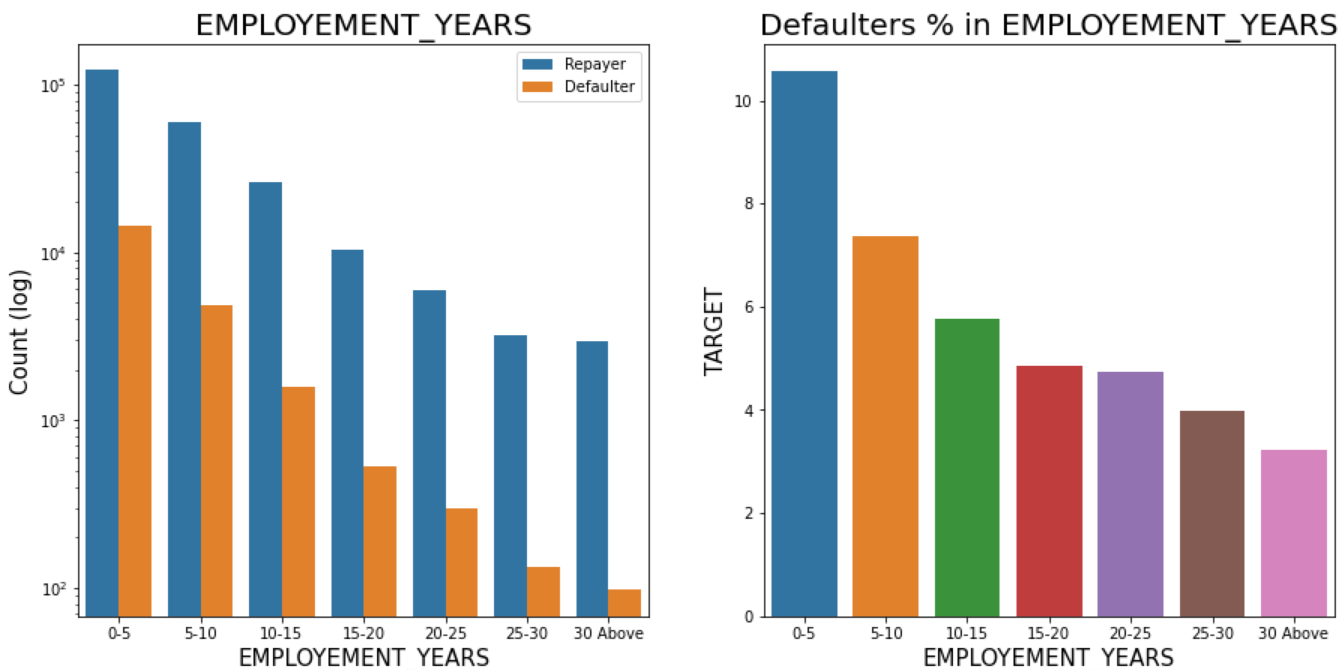
- **Organizations with highest percent of defaultess are Transport: type 3 (16%), Industry: type 13 (13.5%), Industry: type 8 (12.5%) and Restaurant (less than**

12%).

- **Self employed people have relative high defaulting rate,to be safer side loan disbursement should be avoided or provide loan with higher interest rate to mitigate the risk of defaulting.**
- **Most of the people application for loan are from Business Entity Type 3**
- **For a very high number of applications, Organization type information is unavailable(XNA)**
- **It can be seen that following category of organization type has lesser defaulters thus safer for providing loans: Trade Type 4 and 5, Industry type 8**

In [122]:

```
#11 Analyzing Employment_Year based on loan repayment status
univariate(appl_data,"EMPLOYEMENT_YEARS","TARGET",True,False,True)
```

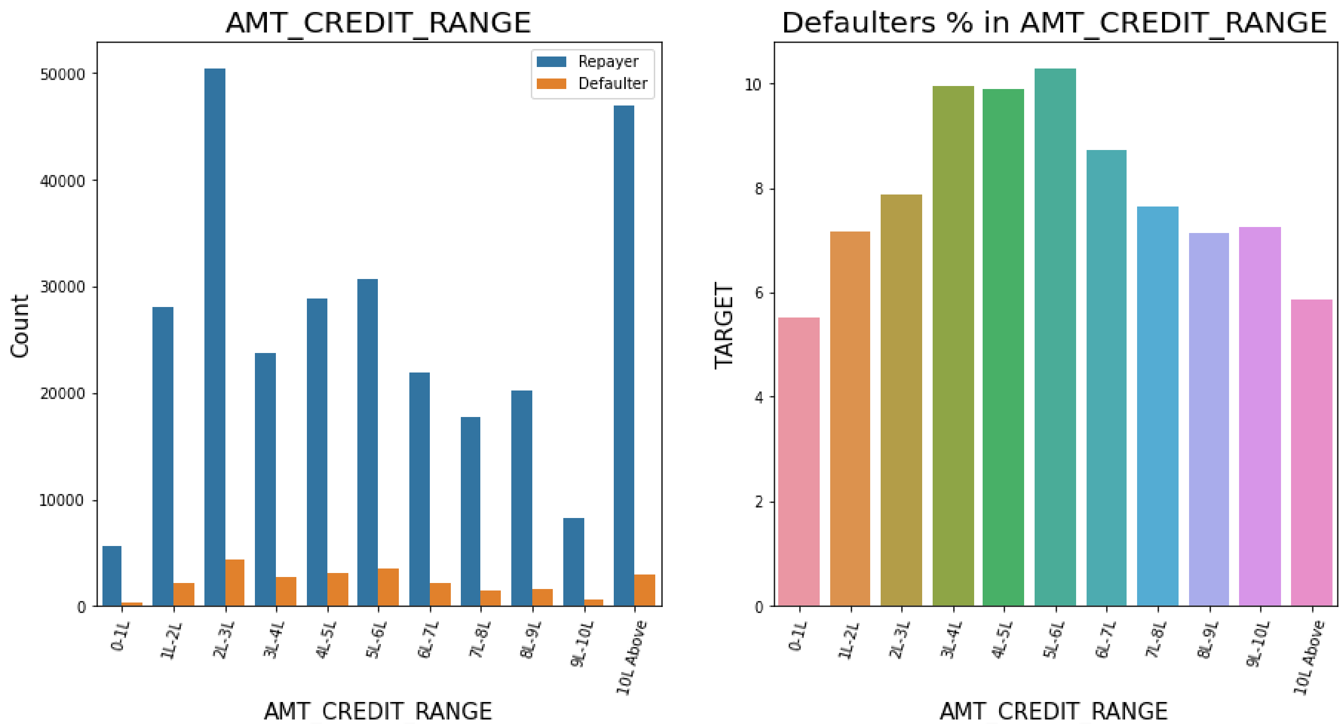


Inferences: Employment in Years

- **Majority of the applicants having working experience between 0-5 years are defaultees. The defaulting rating of this group is also the highest which is around 10%**
- **With increase of employment year, defaulting rate is radually decreasing.**
- **with people having 40+ year experience have less than 1% default rate**

In [123]:

```
#12 Analyzing Amount_Credit based on loan repayment status

univariate(appl_data,"AMT_CREDIT_RANGE","TARGET",False,True,True)
```

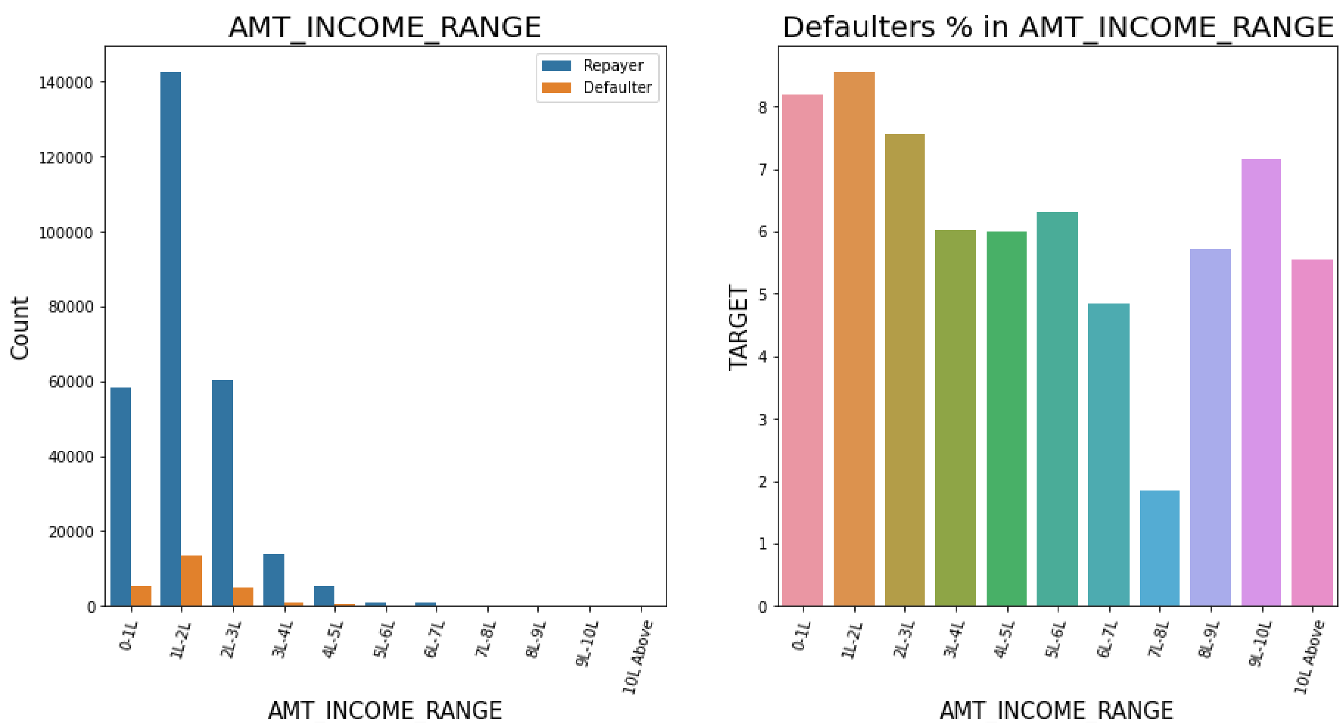


Inferences: Loan Amount

- **there are high number of applicants have loan in range of 2-3 Lakhs followed by 10 Lakh above range**
- **People who get loan for 3-6 Lakhs have most number of defaulters than other loan range.**

In [124]:

```
#13  Analyzing Amount_Income Range based on loan repayment status

univariate(appl_data,"AMT_INCOME_RANGE","TARGET",False,True,True)
```

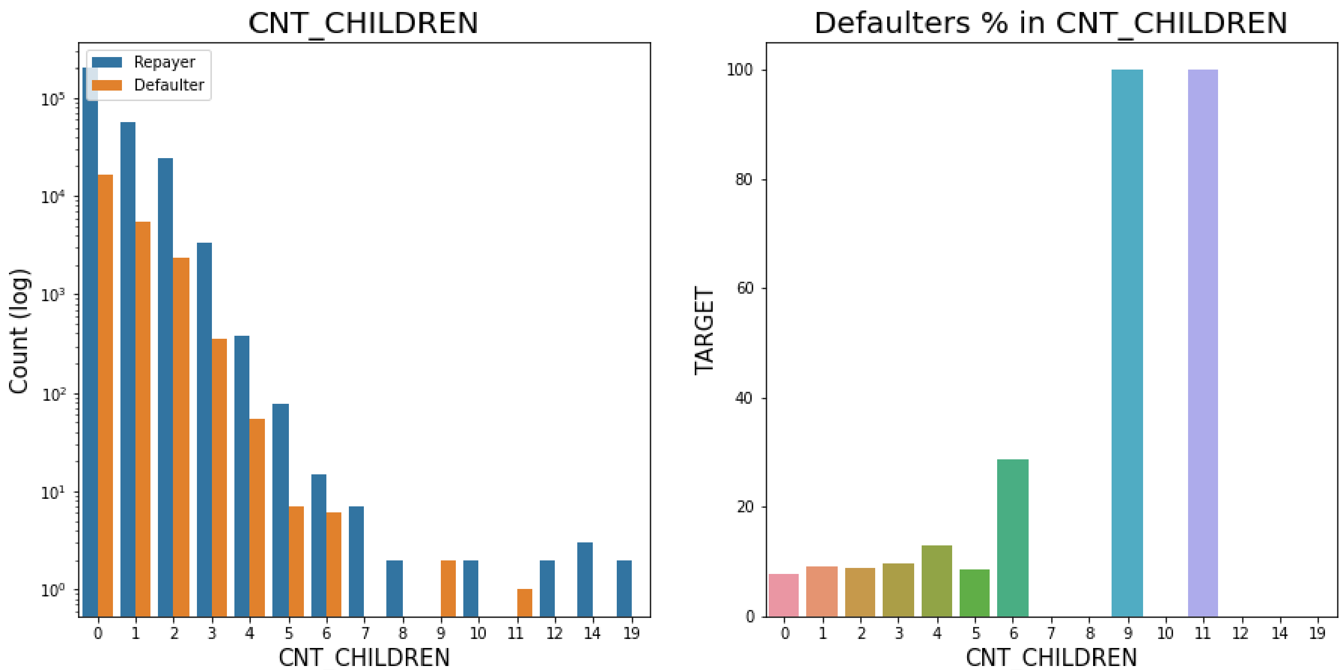


Inferences: Applicant Income

- **Majority of the applications have Income total less than 3 Lakhs.**
- **Application with Income less than 3 Lakhs has high probability of defaulting**

- **Applicant with Income 7-8 Lakhas are less likely to default.**

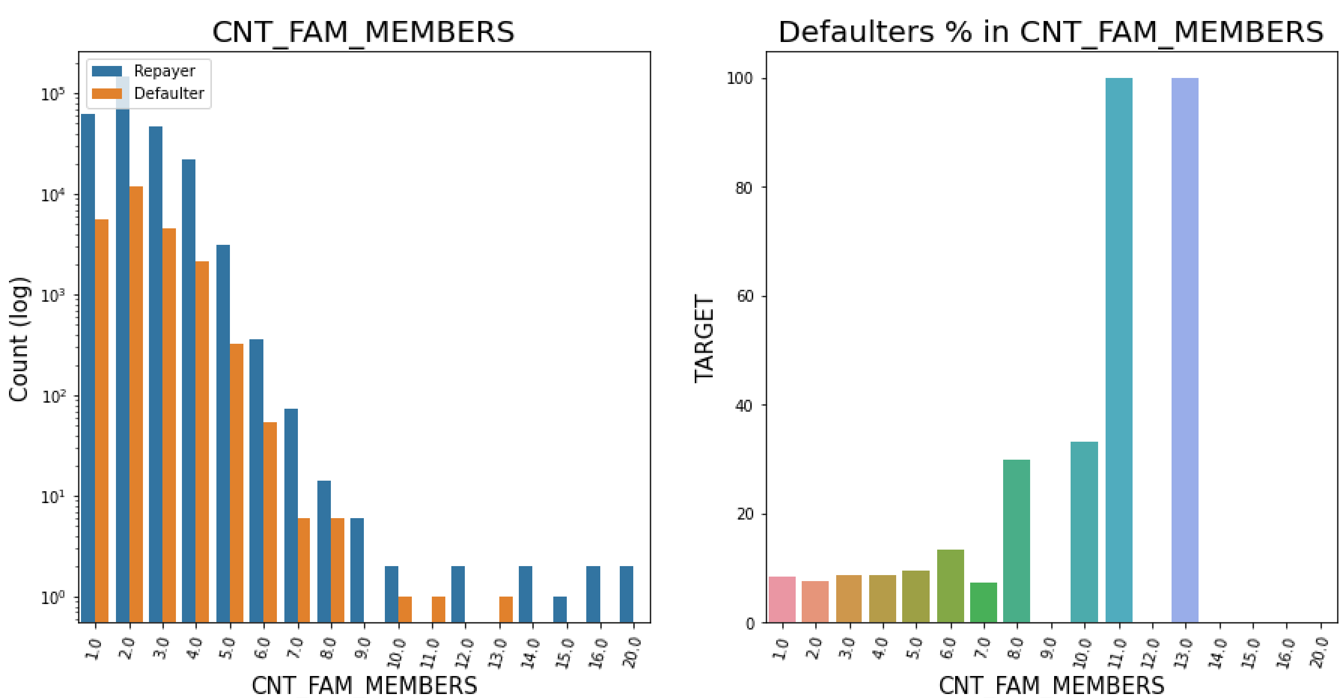
In [125]:
```
#14  Analyzing Number of children based on loan repayment status

univariate(appl_data,"CNT_CHILDREN","TARGET",True,False,True)
```



Inferences: Client Children's Count

- **Most of the applicants do not have children**
- **Very few clients have more than 3 children.**
- **Client who have more than 4 children has a very high default rate with child count 9 and 11 showing 100% default rate**

In [126]:
```
#15 Analyzing Number of family members based on loan repayment status
univariate(appl_data,"CNT_FAM_MEMBERS","TARGET",True,True,True)
```



Inferences: Family Memembers Count

- **Family member follows the same trend as children where having more family members increases the risk of defaulting**

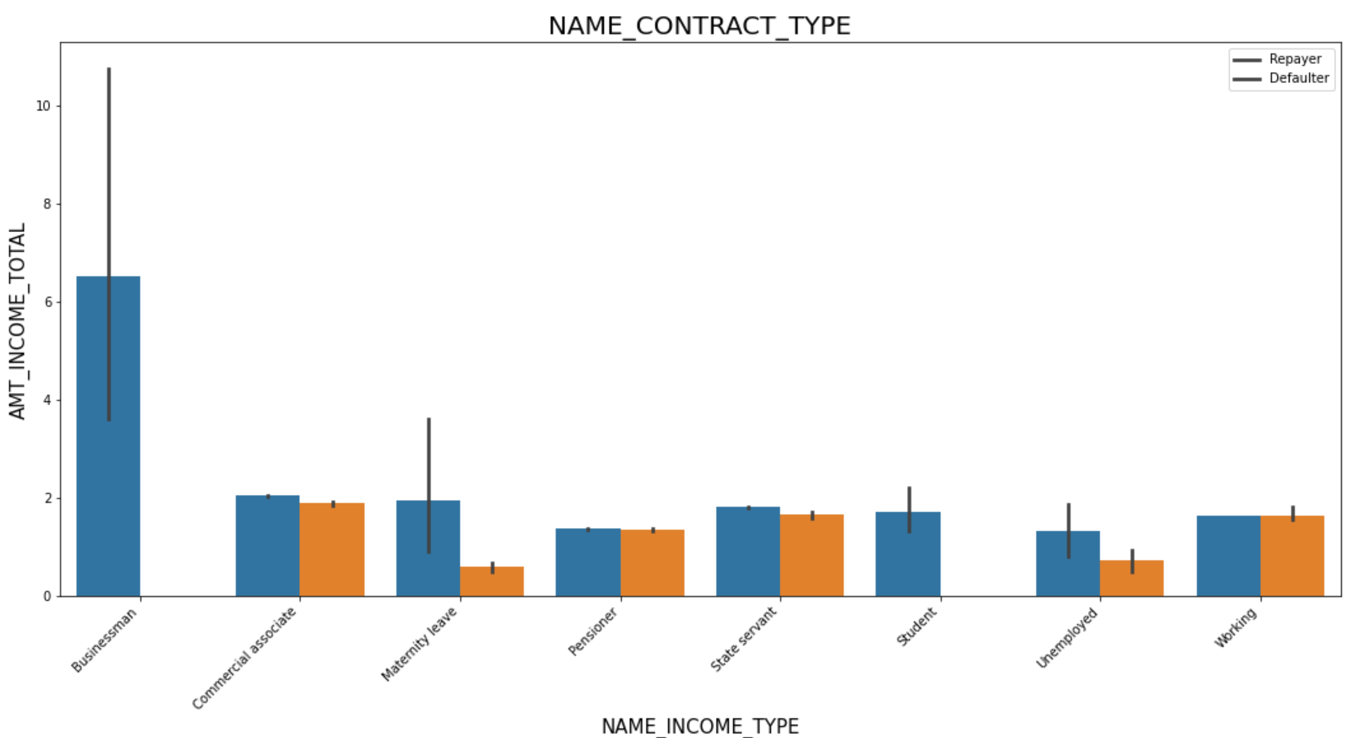
Categorical Bivariate or Multivariate Analysis

In [127]:
```
appl_data.groupby('NAME_INCOME_TYPE')['AMT_INCOME_TOTAL'].describe()
```

Out[127]:

NAME_INCOME_TYPE	count	mean	std	min	25%	50%	75%	max
Businessman	10.0	6.525000	6.272260	1.8000	2.250	4.9500	8.43750	22.5000
Commercial associate	71617.0	2.029553	1.479742	0.2655	1.350	1.8000	2.25000	180.0009
Maternity leave	5.0	1.404000	1.268569	0.4950	0.675	0.9000	1.35000	3.6000
Pensioner	55362.0	1.364013	0.766503	0.2565	0.900	1.1700	1.66500	22.5000
State servant	21703.0	1.797380	1.008806	0.2700	1.125	1.5750	2.25000	31.5000
Student	18.0	1.705000	1.066447	0.8100	1.125	1.5750	1.78875	5.6250
Unemployed	22.0	1.105364	0.880551	0.2655	0.540	0.7875	1.35000	3.3750
Working	158774.0	1.631699	3.075777	0.2565	1.125	1.3500	2.02500	1170.0000

In [128]:
```
# Income type vs Income Amount Range on a Seaborn Barplot
bivariate_c("NAME_INCOME_TYPE","AMT_INCOME_TOTAL",appl_data,"TARGET",(18,8),['Repayer
```



Inferences:

- **It can be seen that Businessman income is the highest and the estimated range with default 95% confidence level seem to indicate that the income of a Businessman could be in the range of slightly close to 4 lakhs and slightly above 10 lakhs**

Numeric Variables Analysis

Bisecting the app_data dataframe based on Target value 0 and 1 for correlation and other analysis

In [129]:

```
#Listing all the columnns of dataframe "appl_data"
appl_data.columns
```

Out[129]:

```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_REALT
Y', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRIC
E', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATU
S', 'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED',
'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'FLAG_MOBIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEM
BERS', 'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_S
TART', '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_30_CNT_SOCIAL_CIRCLE', 'D
EF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'DA
YS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_3', 'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CRED
IT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
       'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BURE
AU_YEAR', 'AMT_INCOME_RANGE', 'AMT_CREDIT_RANGE', 'AMT_GOODS_PRICE_RANGE', 'AGE', 'AG
E_GROUP', 'YEARS_EMPLOYED', 'EMPLOYEMENT_YEARS'],
      dtype='object')
```

In [130]:

```
# bisecting the app_data dataframe based on Target value 0 and 1 for correlation and

cols_for_correlation = ['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_REALTY',
                        'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUIT
                        'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
                        'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRT
                        'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'OCCUPATION_TYPE', 'C
                        'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START',
                        'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION', '
                        'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CIT
                        'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS
                        'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY', 'A
                        'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT', 'AM


# Repayers dataframe
Repayer_df = appl_data.loc[appl_data['TARGET']==0, cols_for_correlation]

# Defaulters dataframe
Defaulter_df = appl_data.loc[appl_data['TARGET']==1, cols_for_correlation]
```

In [131]:

```
len(cols_for_correlation)
```

Out[131]: 41

Correlation between numeric variable

In [132]:

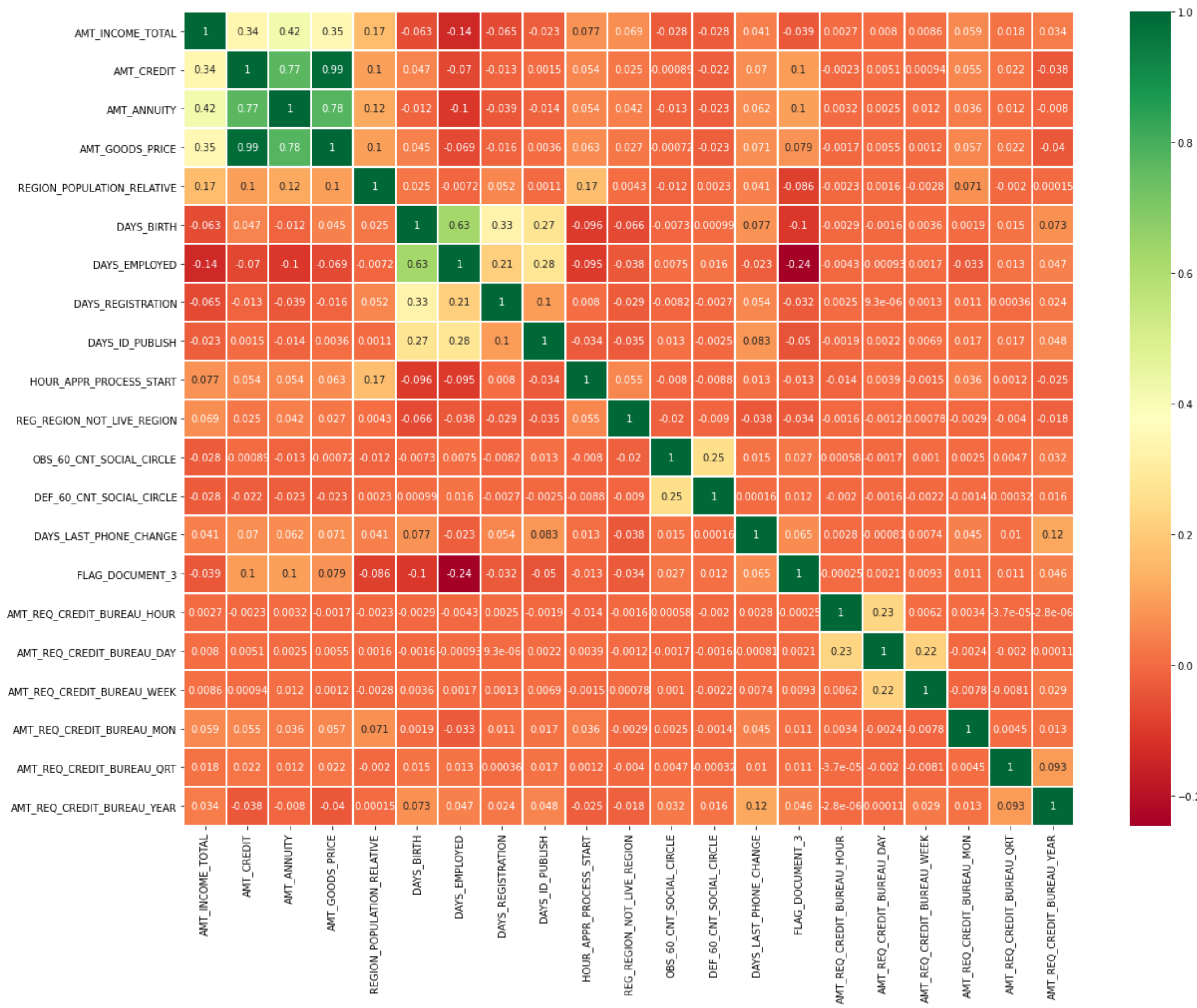
```
# Getting  top 10 correlation for the Repayers dataframe

corr_repayer = Repayer_df.corr()
corr_df_repayer = corr_repayer.where(np.triu(np.ones(corr_repayer.shape),k=1).astype(
corr_df_repayer.columns =['VAR1','VAR2','Correlation']
corr_df_repayer.dropna(subset = ["Correlation"], inplace = True)
corr_df_repayer["Correlation"]=corr_df_repayer["Correlation"].abs()
corr_df_repayer.sort_values(by='Correlation', ascending=False, inplace=True)
corr_df_repayer.head(10)
```

Out[132]:

	VAR1	VAR2	Correlation
64	AMT_GOODS_PRICE	AMT_CREDIT	0.987250
65	AMT_GOODS_PRICE	AMT_ANNUITY	0.776686
43	AMT_ANNUITY	AMT_CREDIT	0.771309
131	DAYS_EMPLOYED	DAYS_BIRTH	0.626114
42	AMT_ANNUITY	AMT_INCOME_TOTAL	0.418953
63	AMT_GOODS_PRICE	AMT_INCOME_TOTAL	0.349462
21	AMT_CREDIT	AMT_INCOME_TOTAL	0.342799
152	DAYS_REGISTRATION	DAYS_BIRTH	0.333151
174	DAYS_ID_PUBLISH	DAYS_EMPLOYED	0.276663
173	DAYS_ID_PUBLISH	DAYS_BIRTH	0.271314

In [133]:

```
#plotting heatmap to see linear correlation amoung Repayers

fig = plt.figure(figsize=(20,15))
ax = sns.heatmap(Repayer_df.corr(), cmap="RdYlGn",annot=True,linewidth =1)
```



Inferences: Correlating factors amongst repayers

1. Credit amount is highly correlated with:

- **Goods Price Amount**
- **Loan Annuity**
- **Total Income**

2. We can also see that repayers have high correlation in number of days employed.

In [134]:

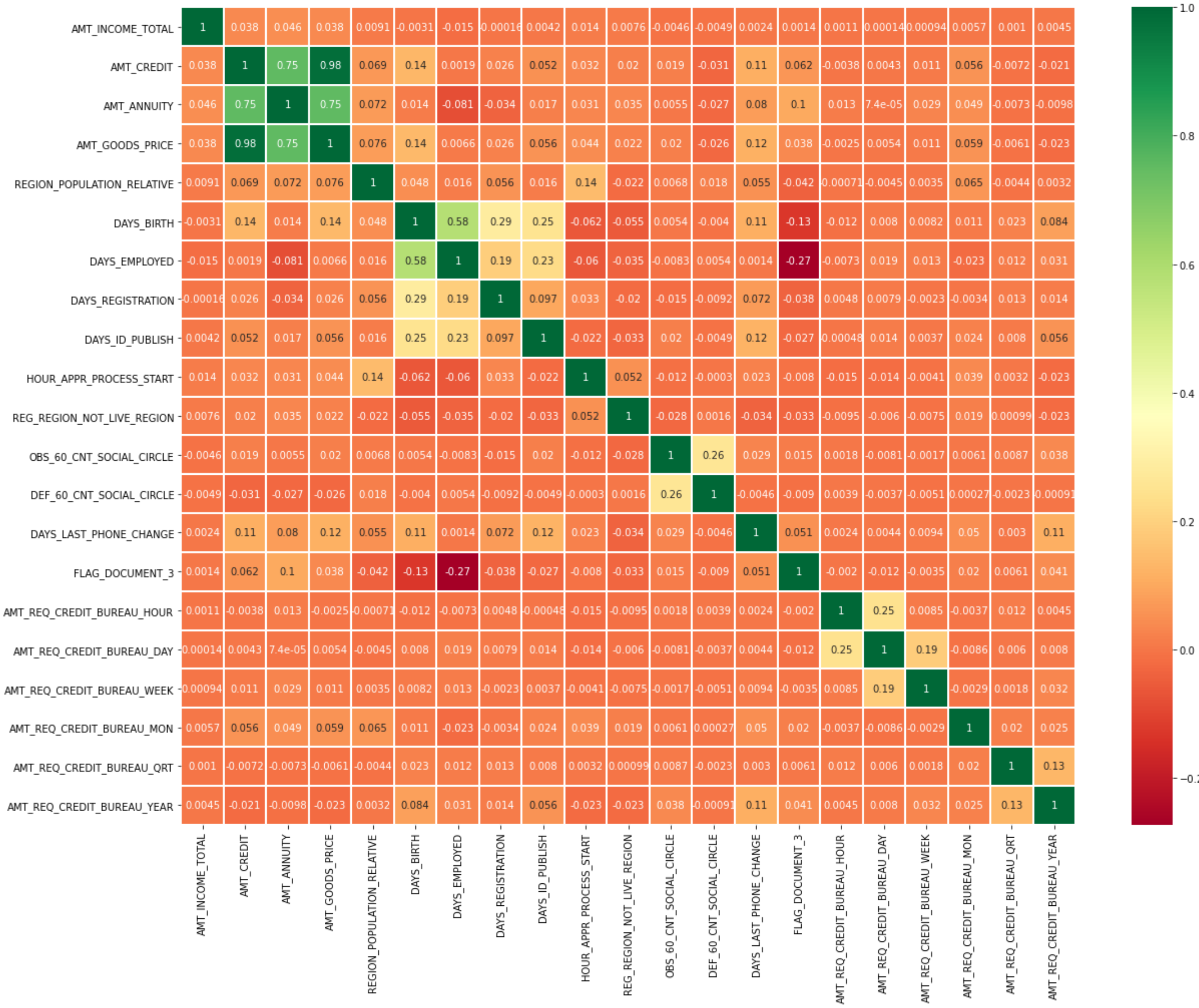
```
# Getting the top 10 correlation for the Defaulter data
corr_Defaulter = Defaulter_df.corr()
corr_Defaulter = corr_Defaulter.where(np.triu(np.ones(corr_Defaulter.shape),k=1).asty
corr_df_Defaulter = corr_Defaulter.unstack().reset_index()
corr_df_Defaulter.columns =['VAR1','VAR2','Correlation']
corr_df_Defaulter.dropna(subset = ["Correlation"], inplace = True)
corr_df_Defaulter["Correlation"]=corr_df_Defaulter["Correlation"].abs()
corr_df_Defaulter.sort_values(by='Correlation', ascending=False, inplace=True)
corr_df_Defaulter.head(10)
```

Out[134]:

	VAR1	VAR2	Correlation
64	AMT_GOODS_PRICE	AMT_CREDIT	0.983103
65	AMT_GOODS_PRICE	AMT_ANNUITY	0.752699
43	AMT_ANNUITY	AMT_CREDIT	0.752195
131	DAYS_EMPLOYED	DAYS_BIRTH	0.582185
152	DAYS_REGISTRATION	DAYS_BIRTH	0.289114
300	FLAG_DOCUMENT_3	DAYS_EMPLOYED	0.272169
263	DEF_60_CNT_SOCIAL_CIRCLE	OBS_60_CNT_SOCIAL_CIRCLE	0.264159
173	DAYS_ID_PUBLISH	DAYS_BIRTH	0.252863
351	AMT_REQ_CREDIT_BUREAU_DAY	AMT_REQ_CREDIT_BUREAU_HOUR	0.247511
174	DAYS_ID_PUBLISH	DAYS_EMPLOYED	0.229090

In [135]:

```
fig = plt.figure(figsize=(20,15))
ax = sns.heatmap(Defaulter_df.corr(), cmap="RdYlGn",annot=True,linewidth =1)
```



Inferences: Correlating factors amongst repayers

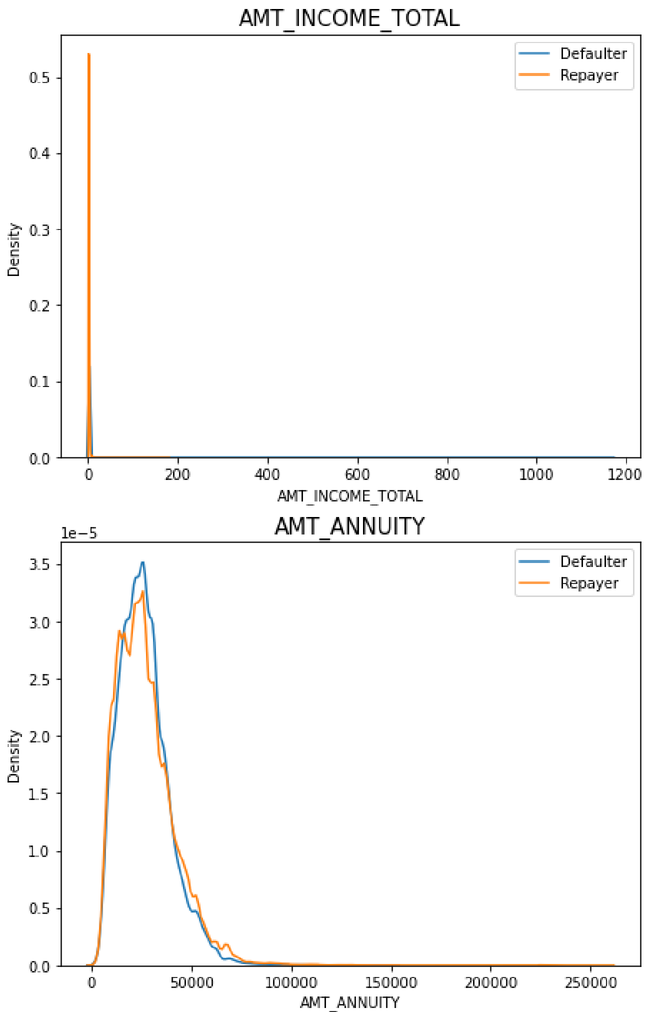
- **Credit amount is highly correlated with good price amount which is same as repayers.**
- **Loan annuity correlation with credit amount has slightly reduced in defaulters(0.75) when compared to repayers(0.77)**
- **We can also see that repayers have high correlation in number of days employed(0.62) when compared to defaulters(0.58).**
- **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 repayers.**
- **Days_birth and number of children correlation has reduced to 0.259 in defaulters when compared to 0.337 in repayers.**
- **There is a slight increase in defaulted to observed count in social circle among defaulters(0.264) when compared to repayers(0.254)**

Numerical Univariate Analysis

In [136]:

```
# Plotting the numerical columns related to amount as distribution plot to see densi
amount = appl_data[[ 'AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY', 'AMT_GOODS_PRICE'

fig = plt.figure(figsize=(16,12))

for i in enumerate(amount):
```

```
    plt.subplot(2,2,i[0]+1)
    sns.distplot(Defaulter_df[i[1]], hist=False,label ="Defaulter")
    sns.distplot(Repayer_df[i[1]], hist=False, label ="Repayer")
    plt.title(i[1], fontdict={'fontsize' : 15, 'fontweight' : 5})
    plt.legend()


plt.show()
```



Inferences:

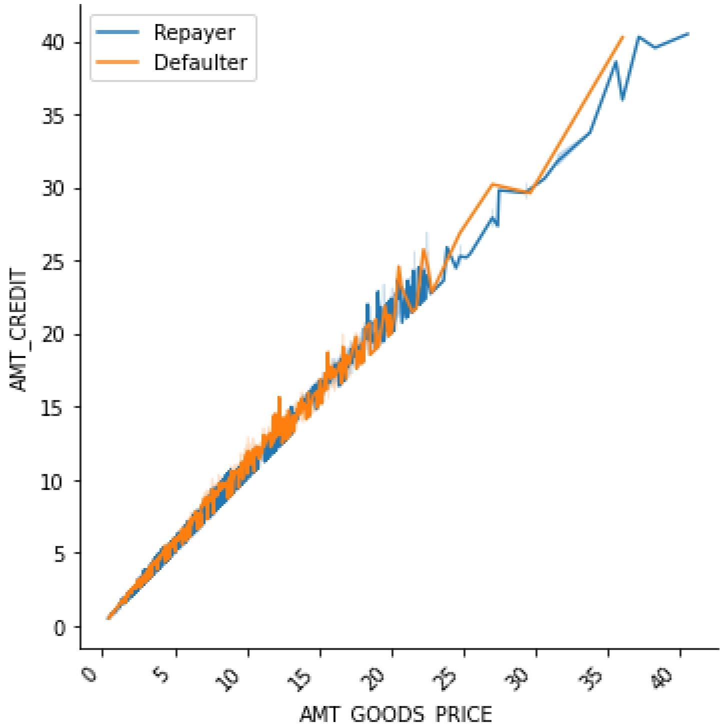
- **Most no of loans are given for goods price below 10 lakhs**
- **Most people pay annuity below 50K for the credit loan**
- **Credit amount of the loan is mostly less then 10 lakhs**
- **The repayers and defaulters distribution overlap in all the plots and hence we cannot use any of these variables in isolation to make a decision**

Numerical Bivariate Analysis

In [137]:

```
# Checking the relationship between Goods price and credit and comparing with loan re
bivariate_n('AMT_GOODS_PRICE','AMT_CREDIT',appl_data,"TARGET", "line",['Repayer','Def
```

<Figure size 1080x1080 with 0 Axes>

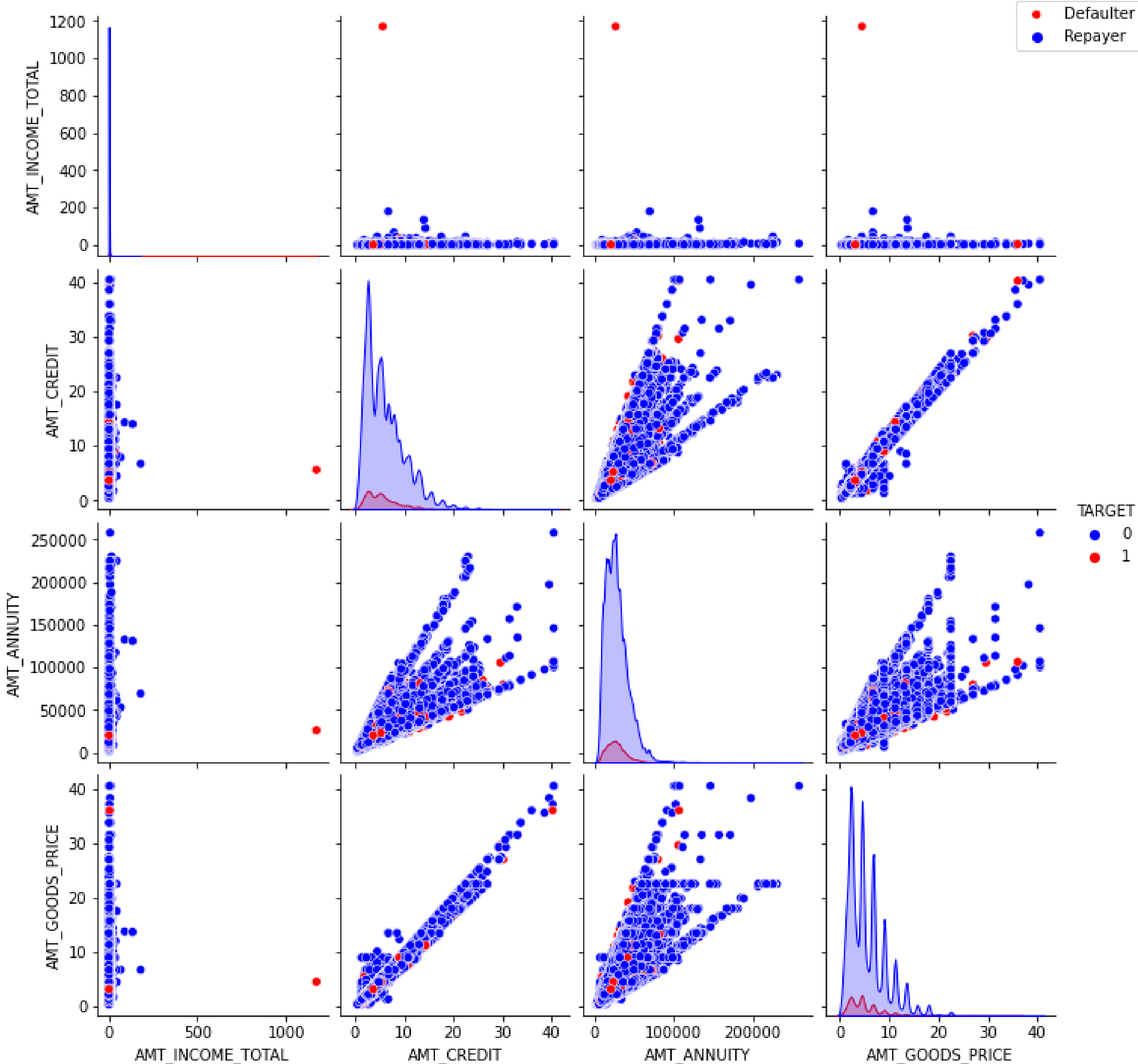


Inferences:

- **When the credit amount goes beyond 30 Lakhs, there is an increase in defaulters.**

In [138]:

```
# Plotting pairplot between amount variable to draw reference against loan repayment

amount = appl_data[[ 'AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY', 'AMT_GOODS_PRICE'
amount = amount[(amount["AMT_GOODS_PRICE"].notnull()) & (amount["AMT_ANNUITY"].notnul

ax= sns.pairplot(amount,hue="TARGET",palette=["b","r"])
ax.fig.legend(labels=['Defaulter','Repayer'])
plt.show()
```



Inferences:

- **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 as based on the scatterplot where most of the data are consolidated in form of a line**
- **There are very less defaulters for AMT_CREDIT >20 Lakhs**

Merged Dataframes Analysis

In [139]:

```
# merge both the dataframe on SK_ID_CURR with Inner Joins
loan_df = pd.merge(appl_data, prev_appl, how='inner', on='SK_ID_CURR')
loan_df.head()
```

Out[139]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_REALTY	CNT_CH
0	100002	1	Cash loans	M	Y	
1	100003	0	Cash loans	F	N	

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_REALTY	CNT_CH
2	100003	0	Cash loans	F	N	
3	100003	0	Cash loans	F	N	
4	100004	0	Revolving loans	M	Y	

In [140]:

```
#Checking the details of the merged dataframe
loan_df.shape
```

Out[140]: (1413701, 82)

In [141]:

```
# checking the columns and column types of the dataframe
loan_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1413701 entries, 0 to 1413700
Data columns (total 82 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
 3   CODE_GENDER                   1413701 non-null  category
 4   FLAG_OWN_REALTY               1413701 non-null  category
 5   CNT_CHILDREN                  1413701 non-null  category
 6   AMT_INCOME_TOTAL              1413701 non-null  float64
 7   AMT_CREDIT_x                  1413701 non-null  float64
 8   AMT_ANNUITY_x                 1413608 non-null  float64
 9   AMT_GOODS_PRICE_x             1412493 non-null  float64
 10  NAME_TYPE_SUITE_x             1410175 non-null  category
 11  NAME_INCOME_TYPE              1413701 non-null  category
 12  NAME_EDUCATION_TYPE           1413701 non-null  category
 13  NAME_FAMILY_STATUS            1413701 non-null  category
 14  NAME_HOUSING_TYPE             1413701 non-null  category
 15  REGION_POPULATION_RELATIVE    1413701 non-null  float64
 16  DAYS_BIRTH                    1413701 non-null  float64
 17  DAYS_EMPLOYED                 1413701 non-null  float64
 18  DAYS_REGISTRATION             1413701 non-null  float64
 19  DAYS_ID_PUBLISH               1413701 non-null  float64
 20  FLAG_MOBIL                    1413701 non-null  int64
 21  OCCUPATION_TYPE               1413701 non-null  category
 22  CNT_FAM_MEMBERS               1413701 non-null  category
 23  REGION_RATING_CLIENT          1413701 non-null  category
 24  REGION_RATING_CLIENT_W_CITY   1413701 non-null  category
 25  WEEKDAY_APPR_PROCESS_START    1413701 non-null  category
 26  HOUR_APPR_PROCESS_START       1413701 non-null  int64
 27  REG_REGION_NOT_LIVE_REGION    1413701 non-null  int64
 28  REG_REGION_NOT_WORK_REGION    1413701 non-null  category
 29  LIVE_REGION_NOT_WORK_REGION   1413701 non-null  category
 30  REG_CITY_NOT_LIVE_CITY        1413701 non-null  category
 31  REG_CITY_NOT_WORK_CITY        1413701 non-null  category
 32  LIVE_CITY_NOT_WORK_CITY       1413701 non-null  category
 33  ORGANIZATION_TYPE             1413701 non-null  category
 34  OBS_30_CNT_SOCIAL_CIRCLE      1410555 non-null  float64
 35  DEF_30_CNT_SOCIAL_CIRCLE      1410555 non-null  float64
 36  OBS_60_CNT_SOCIAL_CIRCLE      1410555 non-null  float64
 37  DEF_60_CNT_SOCIAL_CIRCLE      1410555 non-null  float64
 38  DAYS_LAST_PHONE_CHANGE        1413701 non-null  float64
 39  FLAG_DOCUMENT_3               1413701 non-null  int64
```

```
 40  AMT_REQ_CREDIT_BUREAU_HOUR   1413701 non-null  float64
 41  AMT_REQ_CREDIT_BUREAU_DAY    1413701 non-null  float64
 42  AMT_REQ_CREDIT_BUREAU_WEEK   1413701 non-null  float64
 43  AMT_REQ_CREDIT_BUREAU_MON    1413701 non-null  float64
 44  AMT_REQ_CREDIT_BUREAU_QRT    1413701 non-null  float64
 45  AMT_REQ_CREDIT_BUREAU_YEAR   1413701 non-null  float64
 46  AMT_INCOME_RANGE             1413024 non-null  category
 47  AMT_CREDIT_RANGE             1413701 non-null  category
 48  AMT_GOODS_PRICE_RANGE        1412493 non-null  category
 49  AGE                          1413701 non-null  float64
 50  AGE_GROUP                    1413701 non-null  category
 51  YEARS_EMPLOYED               1413701 non-null  float64
 52  EMPLOYEMENT_YEARS            1140109 non-null  category
 53  SK_ID_PREV                   1413701 non-null  int64
 54  NAME_CONTRACT_TYPE_y         1413701 non-null  category
 55  AMT_ANNUITY_y                1413701 non-null  float64
 56  AMT_APPLICATION              1413701 non-null  float64
 57  AMT_CREDIT_y                 1413700 non-null  float64
 58  AMT_GOODS_PRICE_y            1413701 non-null  float64
 59  NAME_CASH_LOAN_PURPOSE       1413701 non-null  category
 60  NAME_CONTRACT_STATUS         1413701 non-null  category
 61  DAYS_DECISION                1413701 non-null  float64
 62  NAME_PAYMENT_TYPE            1413701 non-null  category
 63  CODE_REJECT_REASON           1413701 non-null  category
 64  NAME_TYPE_SUITE_y            1413701 non-null  object
 65  NAME_CLIENT_TYPE             1413701 non-null  category
 66  NAME_GOODS_CATEGORY          1413701 non-null  category
 67  NAME_PORTFOLIO               1413701 non-null  category
 68  NAME_PRODUCT_TYPE            1413701 non-null  category
 69  CHANNEL_TYPE                 1413701 non-null  category
 70  SELLERPLACE_AREA             1413701 non-null  int64
 71  NAME_SELLER_INDUSTRY         1413701 non-null  category
 72  CNT_PAYMENT                  1413701 non-null  float64
 73  NAME_YIELD_GROUP             1413701 non-null  category
 74  PRODUCT_COMBINATION          1413388 non-null  category
 75  DAYS_FIRST_DRAWING           852595 non-null   float64
 76  DAYS_FIRST_DUE               852595 non-null   float64
 77  DAYS_LAST_DUE_1ST_VERSION    852595 non-null   float64
 78  DAYS_LAST_DUE                852595 non-null   float64
 79  DAYS_TERMINATION             852595 non-null   float64
 80  NFLAG_INSURED_ON_APPROVAL    852595 non-null   float64
 81  YEARLY_DECISION              1413701 non-null  category
dtypes: category(39), float64(34), int64(8), object(1)
memory usage: 527.2+ MB
```

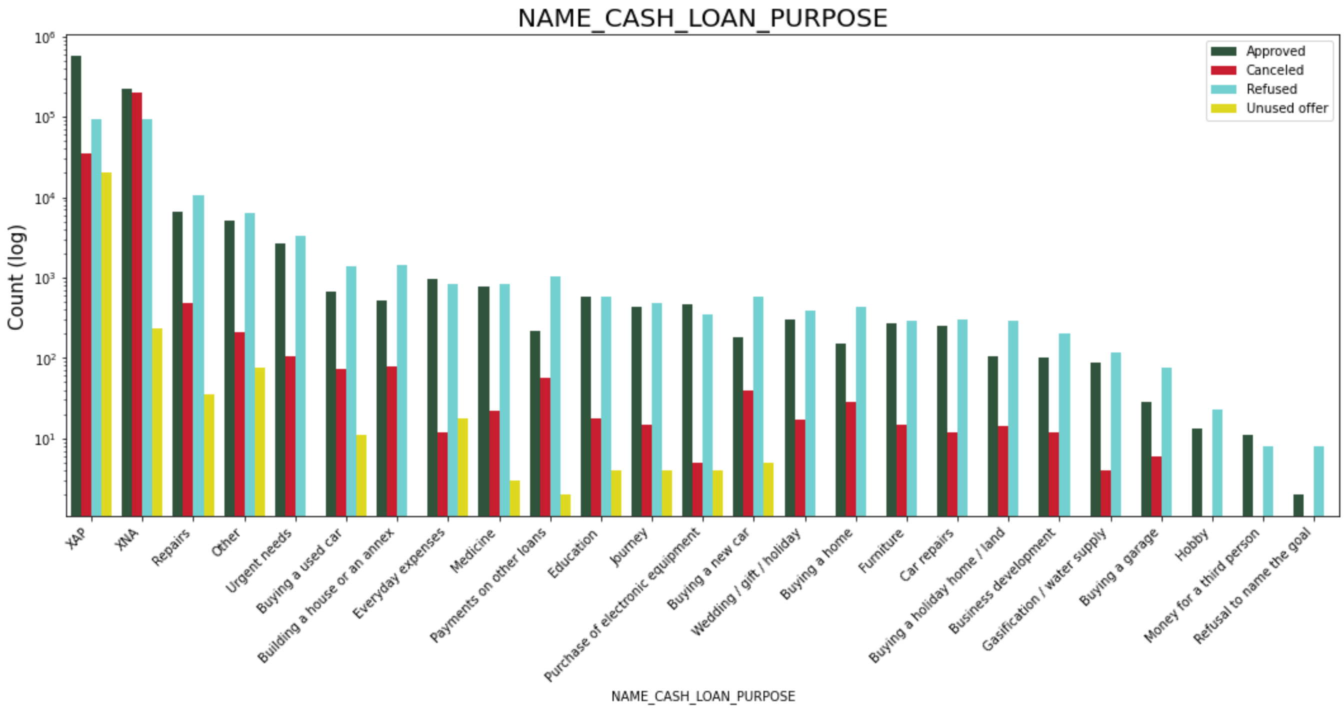
In [142]:

```
# Bisecting the "loan_df" dataframe based on Target value 0 and 1 for correlation and

L0 = loan_df[loan_df['TARGET']==0] # Repayers
L1 = loan_df[loan_df['TARGET']==1] # Defaulters
```

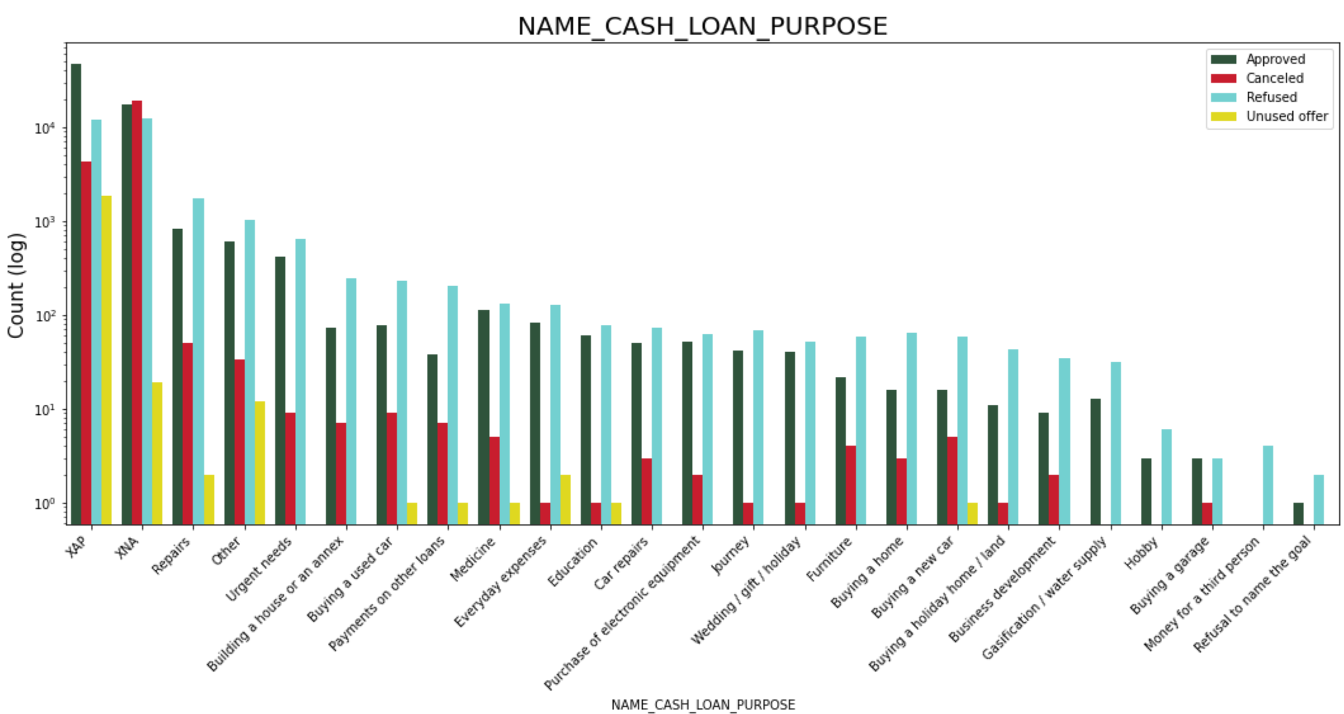
Plotting Contract Status vs purpose of the loan

In [143]:

```
univariate_c_merged("NAME_CASH_LOAN_PURPOSE",L0,"NAME_CONTRACT_STATUS",["#295939","#e
```



In [144]:
```
univariate_c_merged("NAME_CASH_LOAN_PURPOSE",L1,"NAME_CONTRACT_STATUS",["#295939","#e
```

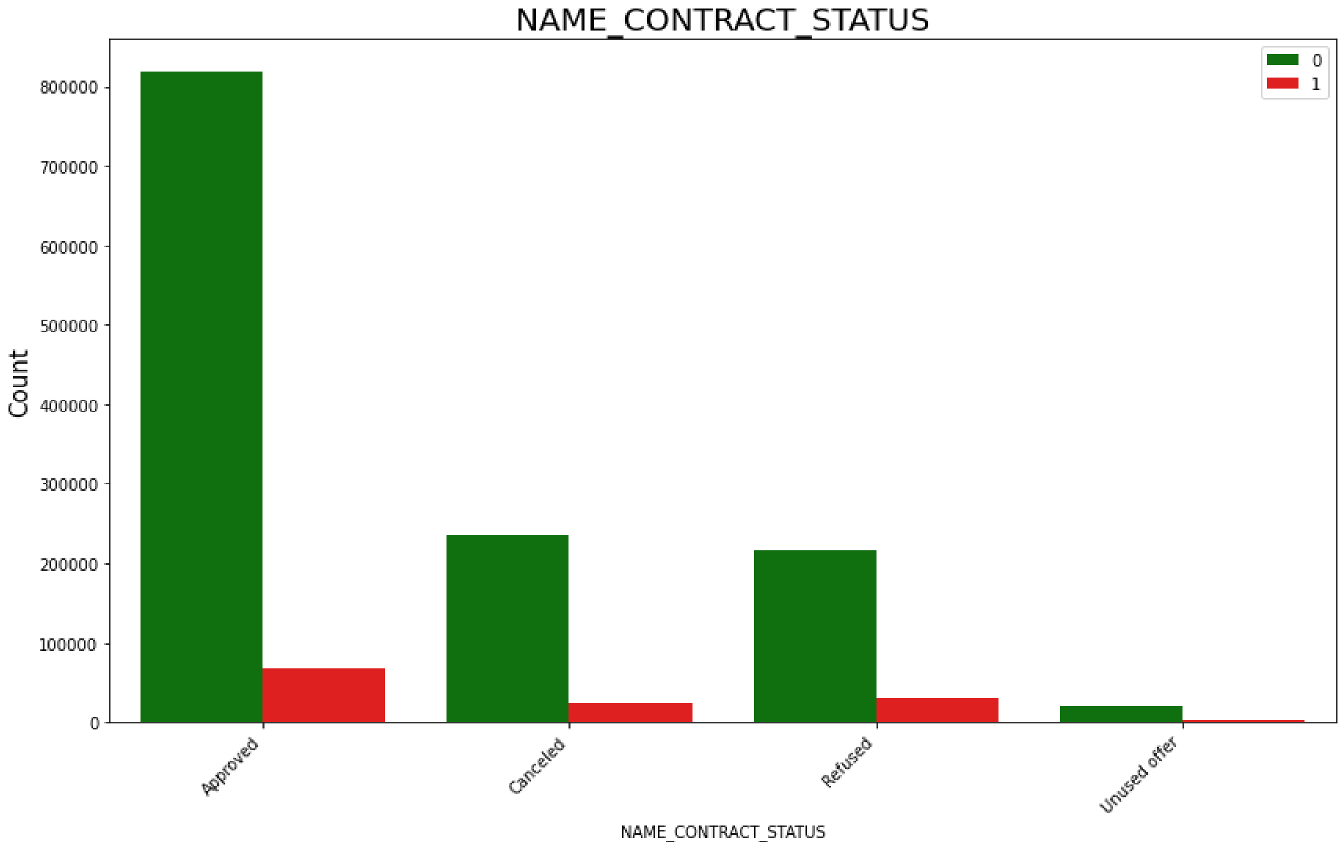


Inferences:

- **Loan purpose has high number of unknown values (XAP, XNA)**
- **Loan taken for the purpose of Repairs looks to have highest default rate**
- **Huge number application have been rejected by bank or refused by client which are applied for Repair or Other. from this we can infer that repair is considered high risk by bank. Also, either they are rejected or bank offers loan on high interest rate which is not feasible by the clients and they refuse the loan.**

In [146]:
```
# Checking Contract Status based on loan repayment status whether there is any busine

univariate_c_merged("NAME_CONTRACT_STATUS",loan_df,"TARGET",['g','r'],False,(14,8))
```

```
r = loan_df.groupby("NAME_CONTRACT_STATUS")["TARGET"]
df1 = pd.concat([r.value_counts(),round(r.value_counts(normalize=True).mul(100),2)],a
df1['Percentage'] = df1['Percentage'].astype(str) +"%" # adding percentage symbol in
df1
```



Out[146]:

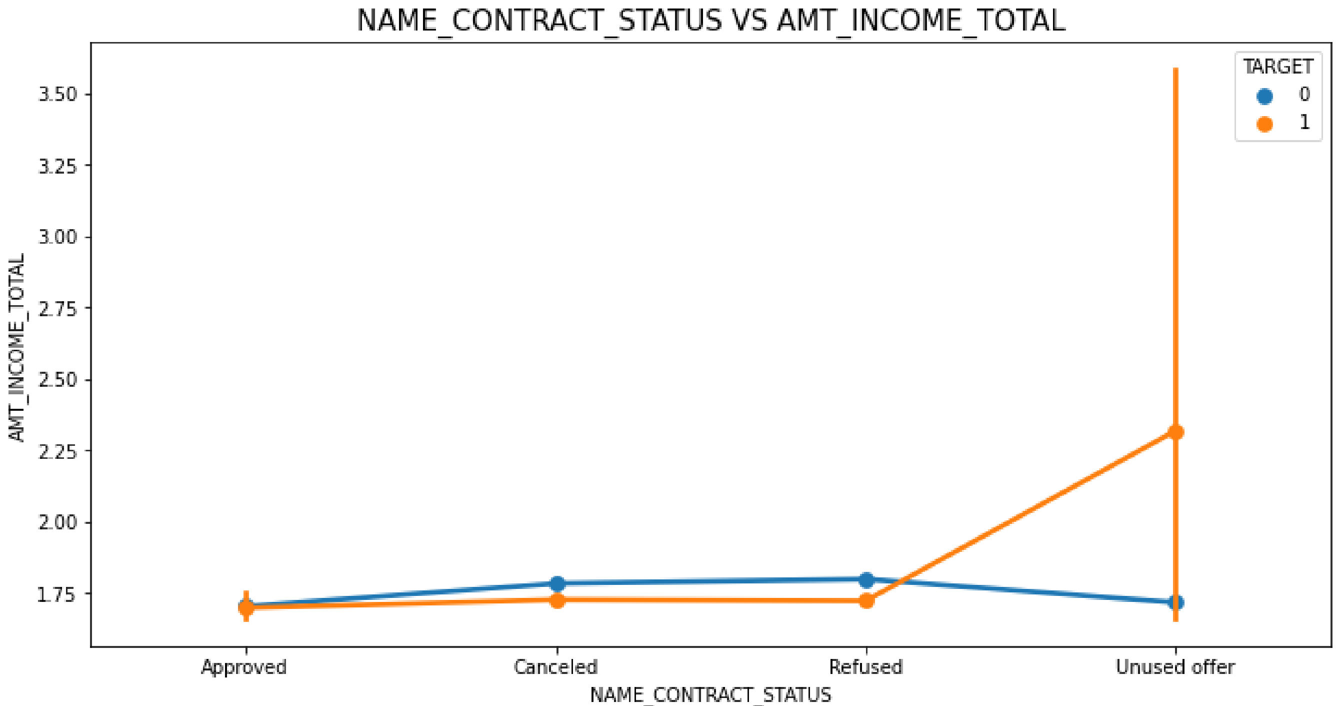
NAME_CONTRACT_STATUS	TARGET	Counts	Percentage
Approved	**0**	818856	92.41%
	1	67243	7.59%
Canceled	**0**	235641	90.83%
	1	23800	9.17%
Refused	**0**	215952	88.0%
	1	29438	12.0%
Unused offer	**0**	20892	91.75%
	1	1879	8.25%

Inferences:

- **90% of the previously cancelled client have actually repayed the loan. Revising the interest rates would increase business opportunity for these clients**
- **88% of the clients who have been previously refused a loan has payed back the loan in current case.**
- **Refusal reason should be recorded for further analysis as these clients could turn into potential repaying customer.**

In [147]:

```
# plotting the relationship between income total and contact status

pointplot(loan_df,"TARGET","NAME_CONTRACT_STATUS",'AMT_INCOME_TOTAL')
```

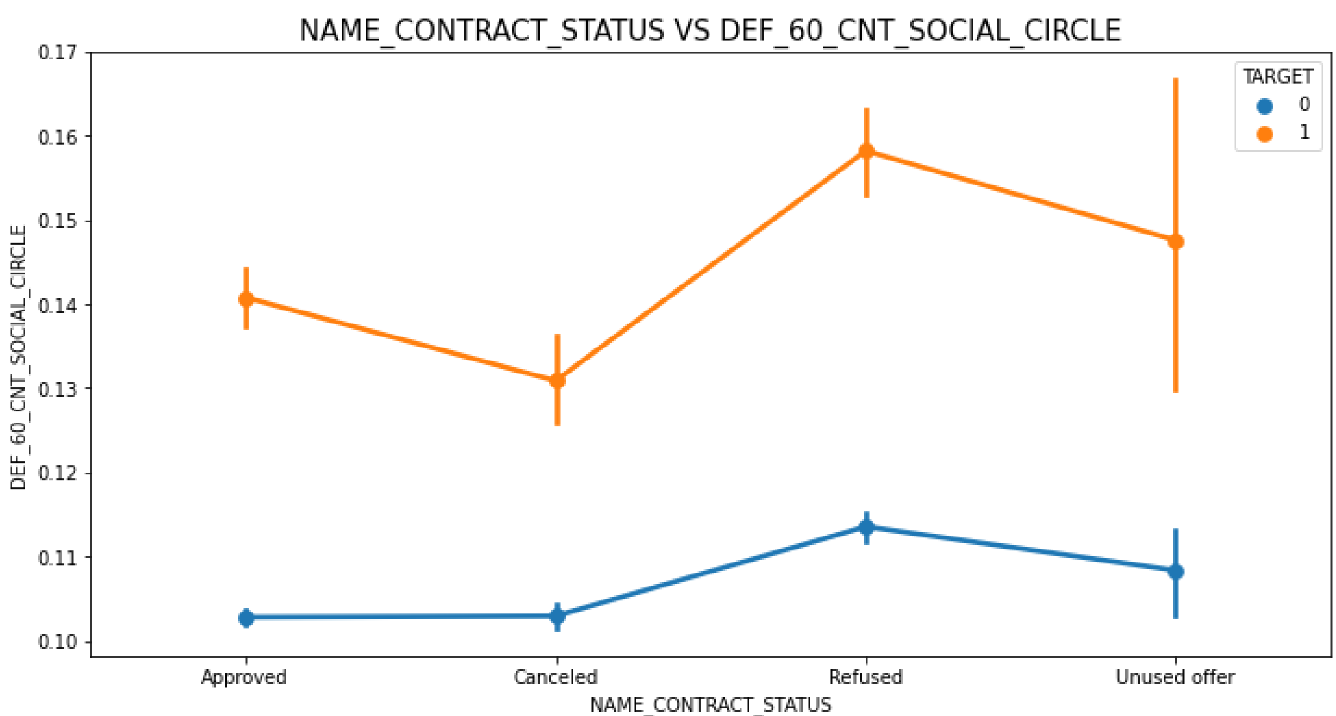


Inferences:

- **The point plot show that the people who have not used offer earlier have defaulted even when there average income is higher than others**

In [148]:

```
# plotting the relationship between people who defaulted in last 60 days being in cl

pointplot(loan_df,"TARGET","NAME_CONTRACT_STATUS",'DEF_60_CNT_SOCIAL_CIRCLE')
```



Inferences:

- **Clients who have average of 0.13 or higher their DEF_60_CNT_SOCIAL_CIRCLE score tend to default more and thus analysing client's social circle could help in disbursment of the loan.**

Conclusions

After analysing the datasets, there are few attributes of a client with which the bank would be able to identify if they will repay the loan or not. The analysis is consised as below with the contributing factors and categorization:

A. Decisive Factor whether an applicant will be Repayer: 1.**NAME_EDUCATION_TYPE: Academic degree has less defaults.**

1. **NAME_INCOME_TYPE: Student and Businessmen have no defaults.**
2. **REGION_RATING_CLIENT: RATING 1 is safer.**
3. **ORGANIZATION_TYPE: Clients with Trade Type 4 and 5 and Industry type 8 have defaulted less than 3%**
4. **DAYS_BIRTH: People above age of 50 have low probability of defaulting**
5. **DAYS_EMPLOYED: Clients with 40+ year experience having less than 1% default rate**
6. **AMT_INCOME_TOTAL:Applicant with Income more than 700,000 are less likely to default**
7. **NAME_CASH_LOAN_PURPOSE: Loans bought for Hobby, Buying garage are being repayed mostly.**
8. **CNT_CHILDREN: People with zero to two children tend to repay the loans.**

B.Decisive Factor whether an applicant will be Defaulter:

1. **CODE_GENDER: Men are at relatively higher default rate**
2. **NAME_FAMILY_STATUS : People who have civil marriage or who are single default a lot.**
3. **NAME_EDUCATION_TYPE: People with Lower Secondary & Secondary education**
4. **NAME_INCOME_TYPE: Clients who are either at Maternity leave OR Unemployed default a lot.**
5. **REGION_RATING_CLIENT: People who live in Rating 3 has highest defaults.**
6. **OCCUPATION_TYPE: Avoid Low-skill Laborers, Drivers and Waiters/barmen staff, Security staff, Laborers and Cooking staff as their default rate is huge.**
7. **ORGANIZATION_TYPE: Organizations with highest percent of loans not repaid are Transport: type 3 (16%), Industry: type 13 (13.5%), Industry: type 8 (12.5%) and Restaurant (less than 12%). Self-employed people have relative high defaulting rate, and thus should be avoided to be approved for loan or provide loan with higher interest rate to mitigate the risk of defaulting.**
8. **DAYS_BIRTH: Avoid young people who are in age group of 20-40 as they have higher probability of defaulting**
9. **DAYS_EMPLOYED: People who have less than 5 years of employment have high default rate.**
10. **CNT_CHILDREN & CNT_FAM_MEMBERS: Client who have children equal to or more than 9 default 100% and hence their applications are to be rejected.**
11. **AMT_GOODS_PRICE: When the credit amount goes beyond 3lakhs, there is an increase in defaulters.**

C. Factors that Loan can be given on Condition of High Interest rate to mitigate any default risk leading to business loss:

1. **NAME_HOUSING_TYPE: High number of loan applications are from the category of people who live in Rented apartments & living with parents and hence offering the loan would mitigate the loss if any of those default.**

2. **AMT_CREDIT: People who get loan for 3-6 Lakhs tend to default more than others and hence having higher interest specifically for this credit range would be ideal.**
3. **AMT_INCOME: Since 90% of the applications have Income total less than 3Lakhs and they have high probability of defaulting, they could be offered loan with higher interest compared to other income category.**
4. **CNT_CHILDREN & CNT_FAM_MEMBERS: Clients who have 4 to 8 children has a very high default rate and hence higher interest should be imposed on their loans.**

D. Suggestions:

- **90% of the previously cancelled client have actually repayed the loan. Record the reason for cancellation which might help the bank to determine and negotiate terms with these repaying customers in future for increase business opportunity.**

In [ ]:

In [ ]:

In [ ]:

In [ ]: