Credit EDA

In [14]:
```
# Importing all necessary libraries

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

%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [15]:
```
# Reading() data set

df=pd.read_csv("application_data.csv")
```

In [16]:
```
# Determining the shape of the dataset

df.shape
```

Out[16]:
```
(307511, 122)
```

In [17]:
```
# Identifying the variables dtypes

df.dtypes
```

Out[17]:
```
SK_ID_CURR                      int64
TARGET                          int64
NAME_CONTRACT_TYPE             object
CODE_GENDER                    object
FLAG_OWN_CAR                   object
                               ...
AMT_REQ_CREDIT_BUREAU_DAY     float64
AMT_REQ_CREDIT_BUREAU_WEEK    float64
AMT_REQ_CREDIT_BUREAU_MON     float64
AMT_REQ_CREDIT_BUREAU_QRT     float64
AMT_REQ_CREDIT_BUREAU_YEAR    float64
Length: 122, dtype: object
```

Data Cleaning And Manipulation

In [18]:
```
# Checking nul values in the data frames

df.isnull().sum()
```

Out[18]:
```
SK_ID_CURR                        0
TARGET                            0
NAME_CONTRACT_TYPE                0
CODE_GENDER                       0
FLAG_OWN_CAR                      0
                              ...
AMT_REQ_CREDIT_BUREAU_DAY     41519
AMT_REQ_CREDIT_BUREAU_WEEK    41519
AMT_REQ_CREDIT_BUREAU_MON     41519
AMT_REQ_CREDIT_BUREAU_QRT     41519
AMT_REQ_CREDIT_BUREAU_YEAR    41519
Length: 122, dtype: int64
```

In [23]:
```
# Identifying the column having more than 30% of null values

nullColumns = df.isnull().sum()
nullColumns = nullColumns[nullColumns.values > (0.3*len(nullColumns))]
len(nullColumns)
```

Out[23]:
```
64
```

There are 64columns in which there are more than 30% of null values. These will impact our analysis. we can drop this columns for better results.

In [26]:
```
# Removing the columns having more than 30% of null values

df.drop(labels=list(nullColumns.index),axis=1,inplace=True)

df.shape
```

Out[26]:
```
(307511, 58)
```

In [27]:
```
# Checkig the existance of null values in the remaining dataframe

df.isnull().sum()
```

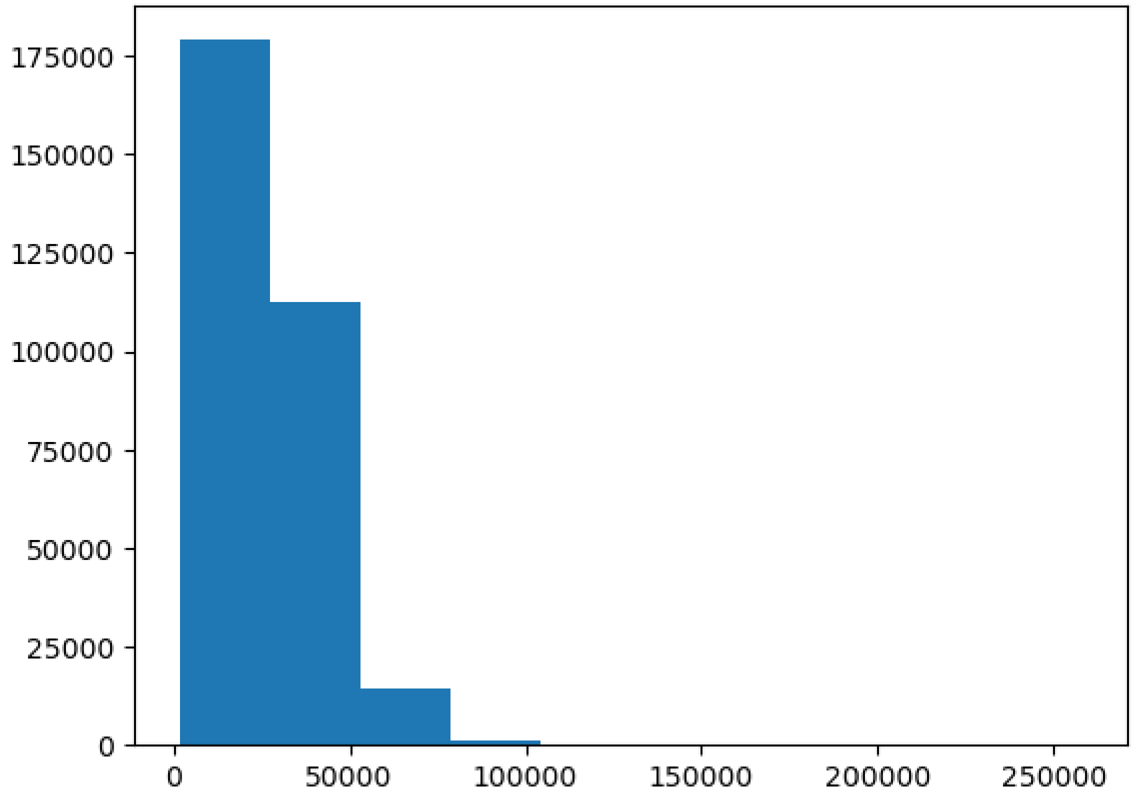
Out[27]:
```
SK_ID_CURR                      0
TARGET                          0
NAME_CONTRACT_TYPE              0
CODE_GENDER                     0
FLAG_OWN_CAR                    0
FLAG_OWN_REALTY                 0
CNT_CHILDREN                    0
AMT_INCOME_TOTAL                0
AMT_CREDIT                      0
AMT_ANNUITY                    12
NAME_INCOME_TYPE                0
NAME_EDUCATION_TYPE             0
NAME_FAMILY_STATUS              0
NAME_HOUSING_TYPE               0
REGION_POPULATION_RELATIVE      0
DAYS_BIRTH                      0
DAYS_EMPLOYED                   0
DAYS_REGISTRATION               0
DAYS_ID_PUBLISH                 0
FLAG_MOBIL                      0
FLAG_EMP_PHONE                  0
FLAG_WORK_PHONE                 0
FLAG_CONT_MOBILE                0
FLAG_PHONE                      0
FLAG_EMAIL                      0
CNT_FAM_MEMBERS                 2
REGION_RATING_CLIENT            0
REGION_RATING_CLIENT_W_CITY     0
WEEKDAY_APPR_PROCESS_START      0
HOUR_APPR_PROCESS_START         0
REG_REGION_NOT_LIVE_REGION      0
REG_REGION_NOT_WORK_REGION      0
LIVE_REGION_NOT_WORK_REGION     0
REG_CITY_NOT_LIVE_CITY          0
REG_CITY_NOT_WORK_CITY          0
LIVE_CITY_NOT_WORK_CITY         0
ORGANIZATION_TYPE               0
DAYS_LAST_PHONE_CHANGE          1
FLAG_DOCUMENT_2                 0
FLAG_DOCUMENT_3                 0
FLAG_DOCUMENT_4                 0
FLAG_DOCUMENT_5                 0
FLAG_DOCUMENT_6                 0
FLAG_DOCUMENT_7                 0
FLAG_DOCUMENT_8                 0
FLAG_DOCUMENT_9                 0
FLAG_DOCUMENT_10                0
FLAG_DOCUMENT_11                0
FLAG_DOCUMENT_12                0
FLAG_DOCUMENT_13                0
FLAG_DOCUMENT_14                0
FLAG_DOCUMENT_15                0
FLAG_DOCUMENT_16                0
FLAG_DOCUMENT_17                0
FLAG_DOCUMENT_18                0
FLAG_DOCUMENT_19                0
FLAG_DOCUMENT_20                0
FLAG_DOCUMENT_21                0
dtype: int64
```

"AMT ANNUITY" column has few null values, hence we try to impute them with the suitable value.

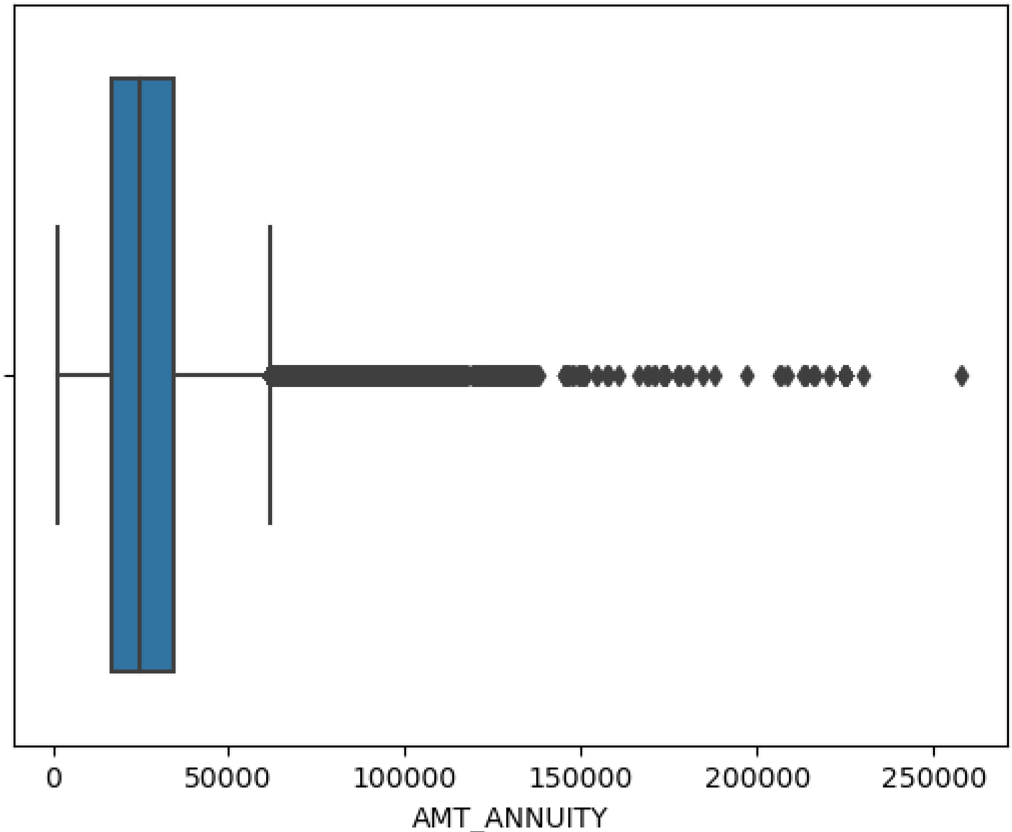
In [28]:
```
df.AMT_ANNUITY.describe()
```

Out[28]:
```
count    307499.000000
mean      27108.573909
std       14493.737315
min        1615.500000
25%       16524.000000
50%       24903.000000
75%       34596.000000
max      258025.500000
Name: AMT_ANNUITY, dtype: float64
```

In [29]:
```
# Ploting Histogram for AMT_ANNUITY column

plt.hist(df.AMT_ANNUITY)
plt.show()
```



In [31]:
```
# Plotting Box plot to identify outliers

sns.boxplot(df.AMT_ANNUITY)
plt.show()
```



Since "AMT ANNUITY" column is having an outliers which is very large, imputing missing values with mean will be inappropriate. Hence, Median comes to rescue for this and we will fill those missing values with median values.

In [32]:
```
# Calculating median and replace null values with median

medianvalue = df.AMT_ANNUITY.median()
df.loc[df['AMT_ANNUITY'].isnull(),'AMT_ANNUITY'] = medianvalue
```

In [33]:
```
# Checking the existance of null values in the remaining data frame (in percentage:

df.isnull().sum()
```

Out[33]:
```
SK_ID_CURR                     0
TARGET                         0
NAME_CONTRACT_TYPE             0
CODE_GENDER                    0
FLAG_OWN_CAR                   0
FLAG_OWN_REALTY                0
CNT_CHILDREN                   0
AMT_INCOME_TOTAL               0
AMT_CREDIT                     0
AMT_ANNUITY                    0
NAME_INCOME_TYPE               0
NAME_EDUCATION_TYPE            0
NAME_FAMILY_STATUS             0
NAME_HOUSING_TYPE              0
REGION_POPULATION_RELATIVE     0
DAYS_BIRTH                     0
DAYS_EMPLOYED                  0
DAYS_REGISTRATION              0
DAYS_ID_PUBLISH                0
FLAG_MOBIL                     0
FLAG_EMP_PHONE                 0
FLAG_WORK_PHONE                0
FLAG_CONT_MOBILE               0
FLAG_PHONE                     0
FLAG_EMAIL                     0
CNT_FAM_MEMBERS                2
REGION_RATING_CLIENT           0
REGION_RATING_CLIENT_W_CITY    0
WEEKDAY_APPR_PROCESS_START     0
HOUR_APPR_PROCESS_START        0
REG_REGION_NOT_LIVE_REGION     0
REG_REGION_NOT_WORK_REGION     0
LIVE_REGION_NOT_WORK_REGION    0
REG_CITY_NOT_LIVE_CITY         0
REG_CITY_NOT_WORK_CITY         0
LIVE_CITY_NOT_WORK_CITY        0
ORGANIZATION_TYPE              0
DAYS_LAST_PHONE_CHANGE         1
FLAG_DOCUMENT_2                0
FLAG_DOCUMENT_3                0
FLAG_DOCUMENT_4                0
FLAG_DOCUMENT_5                0
FLAG_DOCUMENT_6                0
FLAG_DOCUMENT_7                0
FLAG_DOCUMENT_8                0
FLAG_DOCUMENT_9                0
FLAG_DOCUMENT_10               0
FLAG_DOCUMENT_11               0
FLAG_DOCUMENT_12               0
FLAG_DOCUMENT_13               0
FLAG_DOCUMENT_14               0
FLAG_DOCUMENT_15               0
FLAG_DOCUMENT_16               0
FLAG_DOCUMENT_17               0
FLAG_DOCUMENT_18               0
FLAG_DOCUMENT_19               0
FLAG_DOCUMENT_20               0
FLAG_DOCUMENT_21               0
dtype: int64
```

In [34]:
```
# Reading the column names

df.columns
```

Out[34]:

```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE',
       'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH',
       'FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE',
       'FLAG_PHONE', 'FLAG_EMAIL', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START',
       'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION',
       'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION',
       'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY',
       'LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_TYPE',
       'DAYS_LAST_PHONE_CHANGE', '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'],
      dtype='object')
```

In [36]:

```python
# We will drop wanted columns from the data frame for the better analysis:

canDrop = ['FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE',
       'FLAG_PHONE', 'FLAG_EMAIL','CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY','DAYS_LAST_PHONE_CHANGE', '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_
       'FLAG_DOCUMENT_15','FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18
       'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21']


df.drop(labels=canDrop,axis=1,inplace =True)

df.shape
```

Out[36]: (307511, 28)

CHECKING THE DATATYPE OF THE COLUMN

In [37]:

```python
df.info(verbose= True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 28 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_CAR                 307511 non-null  object 
 5   FLAG_OWN_REALTY              307511 non-null  object 
 6   CNT_CHILDREN                 307511 non-null  int64  
 7   AMT_INCOME_TOTAL             307511 non-null  float64
 8   AMT_CREDIT                   307511 non-null  float64
 9   AMT_ANNUITY                  307511 non-null  float64
 10  NAME_INCOME_TYPE             307511 non-null  object 
 11  NAME_EDUCATION_TYPE          307511 non-null  object 
 12  NAME_FAMILY_STATUS           307511 non-null  object 
 13  NAME_HOUSING_TYPE            307511 non-null  object 
 14  REGION_POPULATION_RELATIVE   307511 non-null  float64
 15  DAYS_BIRTH                   307511 non-null  int64  
 16  DAYS_EMPLOYED                307511 non-null  int64  
 17  DAYS_REGISTRATION            307511 non-null  float64
 18  DAYS_ID_PUBLISH              307511 non-null  int64  
 19  WEEKDAY_APPR_PROCESS_START   307511 non-null  object 
 20  HOUR_APPR_PROCESS_START      307511 non-null  int64  
 21  REG_REGION_NOT_LIVE_REGION   307511 non-null  int64  
 22  REG_REGION_NOT_WORK_REGION   307511 non-null  int64  
 23  LIVE_REGION_NOT_WORK_REGION  307511 non-null  int64  
 24  REG_CITY_NOT_LIVE_CITY       307511 non-null  int64  
 25  REG_CITY_NOT_WORK_CITY       307511 non-null  int64  
 26  LIVE_CITY_NOT_WORK_CITY      307511 non-null  int64  
 27  ORGANIZATION_TYPE            307511 non-null  object 
dtypes: float64(5), int64(13), object(10)
memory usage: 65.7+ MB
```

verifying if the object type column are correct , if these columns are incorrect we will fix them first before our analysis.

In [40]:
```
# Name_ Contract_Type

df.NAME_CONTRACT_TYPE.head(10)
```

Out[40]:
```
0        Cash loans
1        Cash loans
2    Revolving loans
3        Cash loans
4        Cash loans
5        Cash loans
6        Cash loans
7        Cash loans
8        Cash loans
9    Revolving loans
Name: NAME_CONTRACT_TYPE, dtype: object
```

In [42]:
```
# CODE_GENDER
df.CODE_GENDER.value_counts()
```

Out[42]:
```
F      202448
M      105059
XNA         4
Name: CODE_GENDER, dtype: int64
```

There are XNA values in 4 columns which means they are not available. since there are more female we can impute them with "F", this will not have any impact on our analysis.

In [44]:
```
# Updating the column 'CODE_GENDER' with "F" in the dataframe

df.loc[df['CODE_GENDER']=='XNA', 'CODE_GENDER']='F'
df['CODE_GENDER'].value_counts()
```

Out[44]:
```
F    202452
M    105059
Name: CODE_GENDER, dtype: int64
```

In [45]:
```
df.ORGANIZATION_TYPE.value_counts(normalize=True)*100
```

Out[45]:
```
Business Entity Type 3    22.110429
XNA                       18.007161
Self-employed             12.491260
Other                      5.425172
Medicine                   3.639870
Business Entity Type 2     3.431747
Government                 3.383294
School                     2.891929
Trade: type 7              2.546576
Kindergarten               2.237318
Construction               2.185613
Business Entity Type 1     1.945947
Transport: type 4          1.755384
Trade: type 3              1.135569
Industry: type 9           1.095245
Industry: type 3           1.065978
Security                   1.055897
Housing                    0.961917
Industry: type 11          0.879318
Military                   0.856555
Bank                       0.815255
Agriculture                0.798020
Police                     0.761274
Transport: type 2          0.716722
Postal                     0.701438
Security Ministries        0.641928
Trade: type 2              0.617864
Restaurant                 0.588922
Services                   0.512177
University                 0.431529
Industry: type 7           0.425025
Transport: type 3          0.386002
Industry: type 1           0.337874
Hotel                      0.314135
Electricity                0.308932
Industry: type 4           0.285193
Trade: type 6              0.205196
Industry: type 5           0.194790
Insurance                  0.194139
Telecom                    0.187636
Emergency                  0.182107
Industry: type 2           0.148938
Advertising                0.139507
Realtor                    0.128776
Culture                    0.123248
Industry: type 12          0.119996
Trade: type 1              0.113167
Mobile                     0.103086
Legal Services             0.099183
Cleaning                   0.084550
Transport: type 1          0.065364
Industry: type 6           0.036421
Industry: type 10          0.035446
Religion                   0.027641
Industry: type 13          0.021788
Trade: type 4              0.020812
Trade: type 5              0.015934
Industry: type 8           0.007805
Name: ORGANIZATION_TYPE, dtype: float64
```

18% values in thr "ORGANIZATION_TYPE" column has XNA values, we can drop these rows from the dataframe causing no impact on analysis

In [46]:
```
# Dropping XNA rows for data frame im ORGANIZATION_TYPE column

df= df[~(df.ORGANIZATION_TYPE=="XNA")]
```

In [47]:
```
df.shape
```

Out[47]:
```
(252137, 28)
```

In [48]:
```
df.columns
```

Out[48]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE',
       'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY',
       'ORGANIZATION_TYPE'],
      dtype='object')
```

In [49]:
```
# Typecasting all the int/float variables to numeric in the dataset

toNumeric =['TARGET','CNT_CHILDREN','AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY',
                 'REGION_POPULATION_RELATIVE','DAYS_BIRTH','DAYS_EMPLOYED',
                 'DAYS_REGISTRATION','DAYS_ID_PUBLISH','HOUR_APPR_PROCESS_START',
                 'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
                 'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY']
df[toNumeric] = df[toNumeric].apply(pd.to_numeric)
```

In [50]:
```
df.head(10)
```

Out[50]:

	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	
5	100008	0	Cash loans	M	N	
6	100009	0	Cash loans	F	Y	
7	100010	0	Cash loans	M	Y	
9	100012	0	Revolving loans	M	N	
10	100014	0	Cash loans	F	N	

10 rows × 28 columns

Since we have cleaned the data set and handled the missing values, we will start our analysis

BINNING THE CATEGORICAL VALUE

Lets start with categorising based on annual income

In [52]:
```
# Creating bins for "AMT_INCOME_TOTAL"

bins = [0,50000,100000,150000,200000,250000,300000,350000,400000,450000,500000,100
xlabels = ['0-50000','50000-100000','100000-150000','150000-200000','200000-250000
           '250000-300000','300000-350000','350000-400000','400000-450000',
           '450000-500000','500000 and Above']

df['AMT_INCOME_RANGE'] = pd.cut(df['AMT_CREDIT'],bins=bins,labels=xlabels)
```

In [54]:
```
# Creating bins for "AMT_CREDIT"

bins = [0,100000,200000,300000,400000,500000,600000,700000,800000,900000,1000000]
xlabels = ['0-100000','100000-200000','200000-300000','300000-400000','400000-5000
           '600000-700000','700000-800000','800000-900000','900000 and Above']

df['AMT_CREDIT_RANGE'] = pd.cut(df['AMT_CREDIT'],bins=bins,labels=xlabels)
```

In [56]:
```
# Dividing the dataset into two datasets consisting of
# target=1 : client with payment difficulties
# Target=0 : others

df_target1 = df.loc[df["TARGET"] == 1]
df_target0 = df.loc[df["TARGET"] == 0]
```

In [57]:
```
print("Target 1 shape : ", df_target1.shape)
print("Target 0 shape : ",df_target0.shape)
```

```
Target 1 shape :  (21835, 30)
Target 0 shape :  (230302, 30)
```

There are less clients with payment difficulties(21835) compared to others (230302)

FINDING RHE IMBALANCE RATIO

In [58]:
```
# Calculating imbalance percentage
# since the majority is target 0  and minority is target1

print("The Data Imbalance ratio is:",round(len(df_target0)/len(df_target1),2))
```

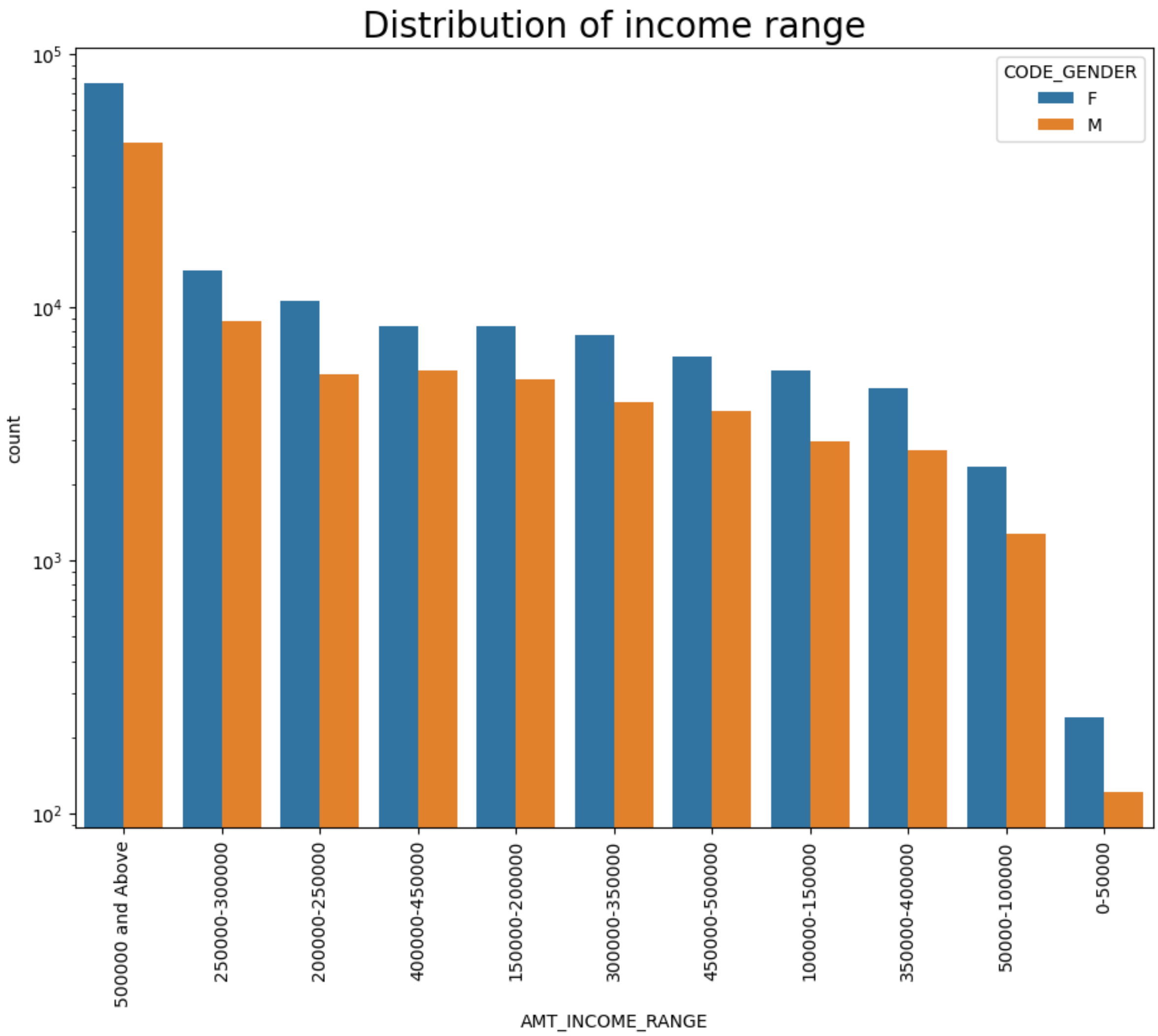
```
The Data Imbalance ratio is: 10.55
```

Categorical Univariate Analysis - Target 0

In [68]:
```
# Common method to plot count plot

def countPlotForUnivariateAnalysis(df,col,title,hue =None):

    plt.rcParams["axes.labelsize"] = 10
    plt.rcParams["axes.titlesize"] = 20
```

```
    temCol = pd.Series(data = hue)
    fig, ax = plt.subplots()
    width = len(df[col].unique())
    fig.set_size_inches(width , 8)
    plt.xticks(rotation=90)
    plt.yscale('log') # Using log scale to capture better  analysis

    plt.title(title)
    ax = sns.countplot(data = df, x= col, order = df[col].value_counts().index, hu

    plt.show()
```

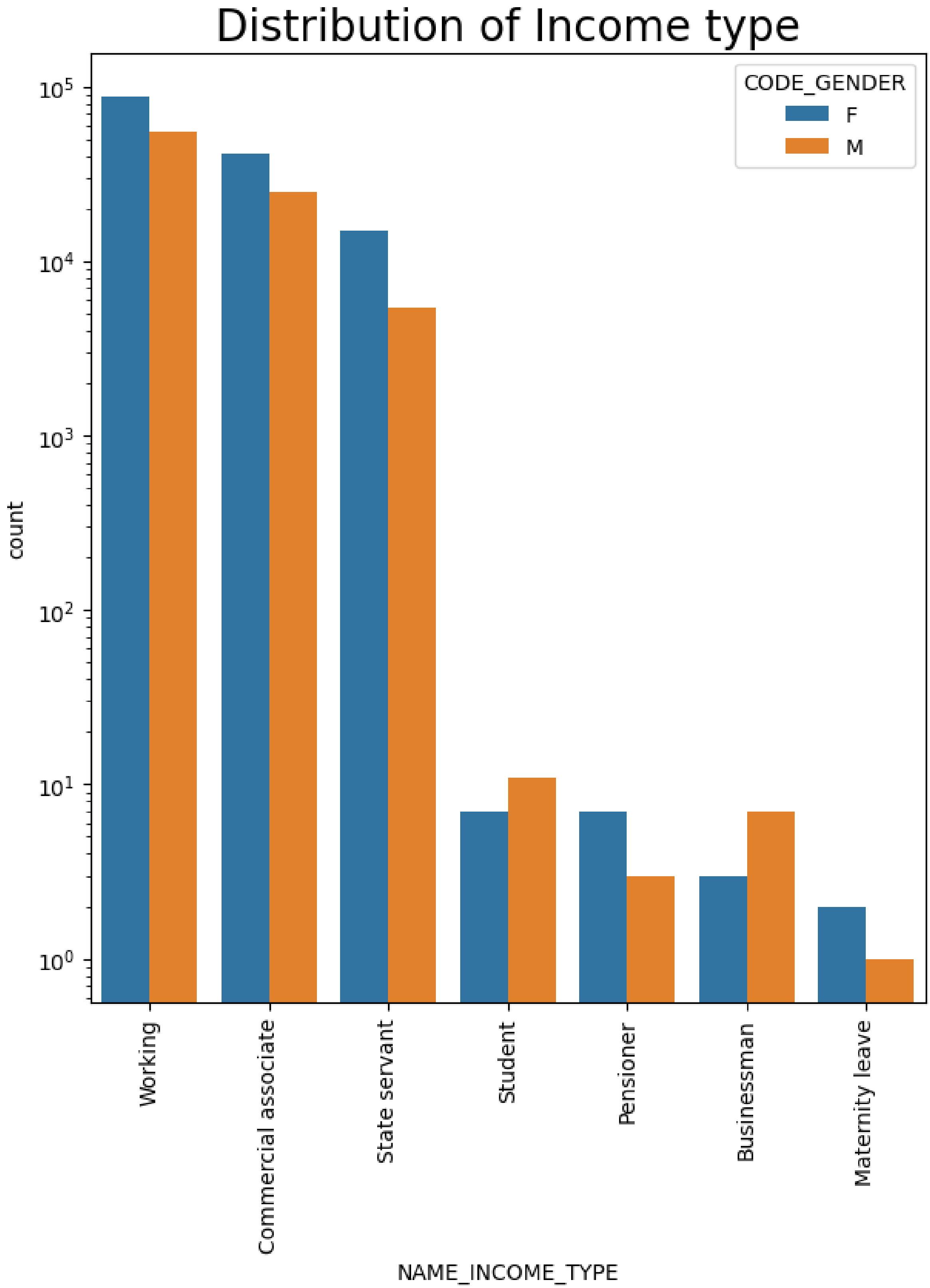
In [69]:
```
# plotting for income range

countPlotForUnivariateAnalysis(df_target0,col='AMT_INCOME_RANGE',title = 'Distribu
```



Insights from the above graph

1. Income range from 1,00,000 to 1,50,000 is having more number of credits.
2. Credit rating for females are more than male
3. For 4,50,000 and above count is very less compared to others.

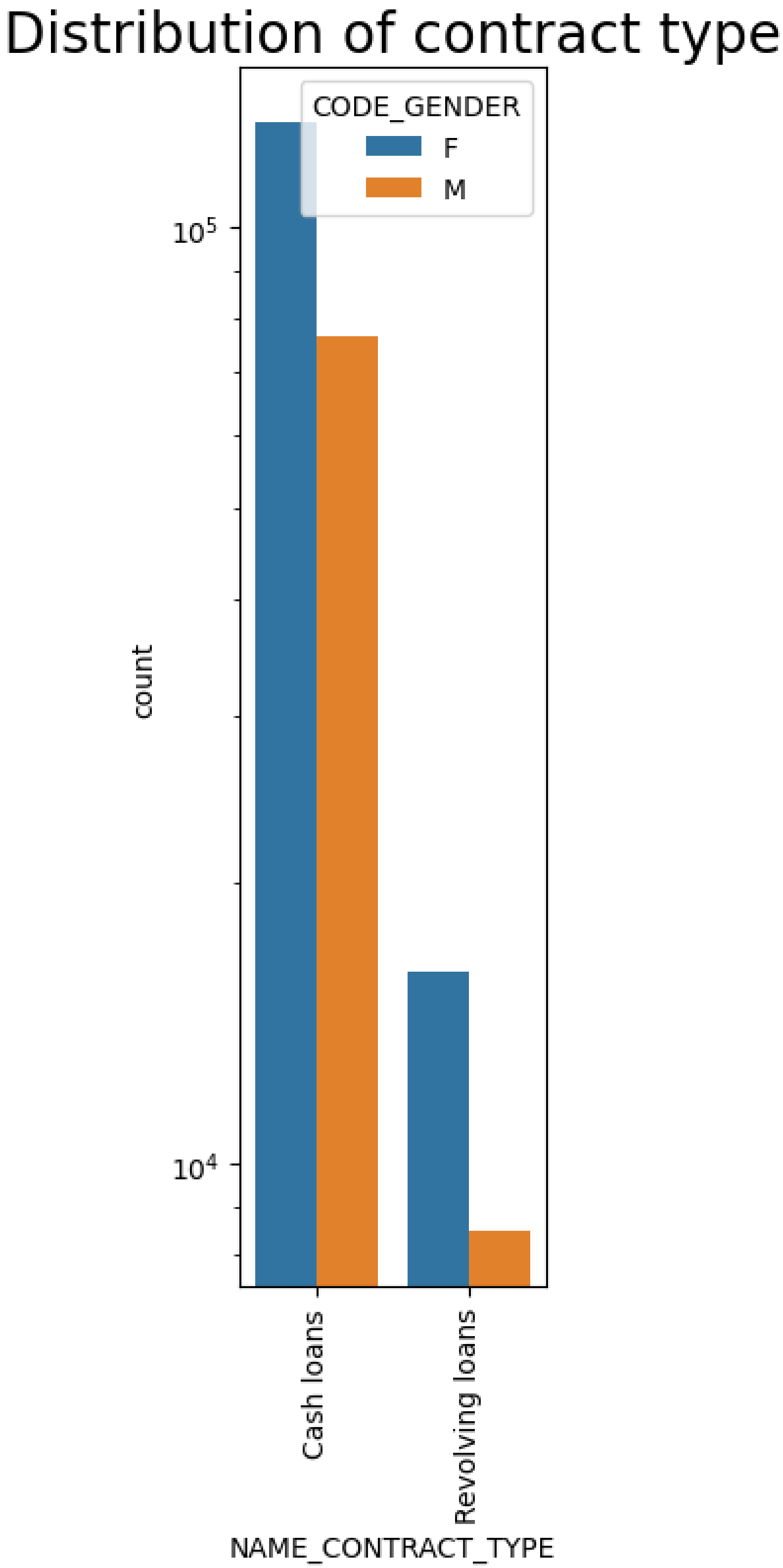
In [70]:
```
# Plotting for Income type

countPlotForUnivariateAnalysis(df_target0,col='NAME_INCOME_TYPE', title = 'Distribu
```



Insights from above graph

1. Working professionals have the highest numbers
2. Those awho are on Maternity leave are the least in numbers
3. Those who are employed in one way or the other have better results.

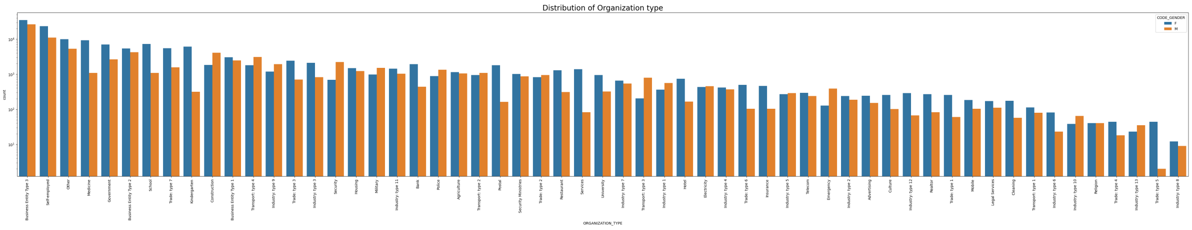
In [73]:
```
# plotting for contract type

countPlotForUnivariateAnalysis(df_target0,col='NAME_CONTRACT_TYPE',title='Distribu
```



Insights from above graph

1. Cash loans contracts have more credit rating than the revolving loans.
2. For this ,also Female is leading for applying credits.

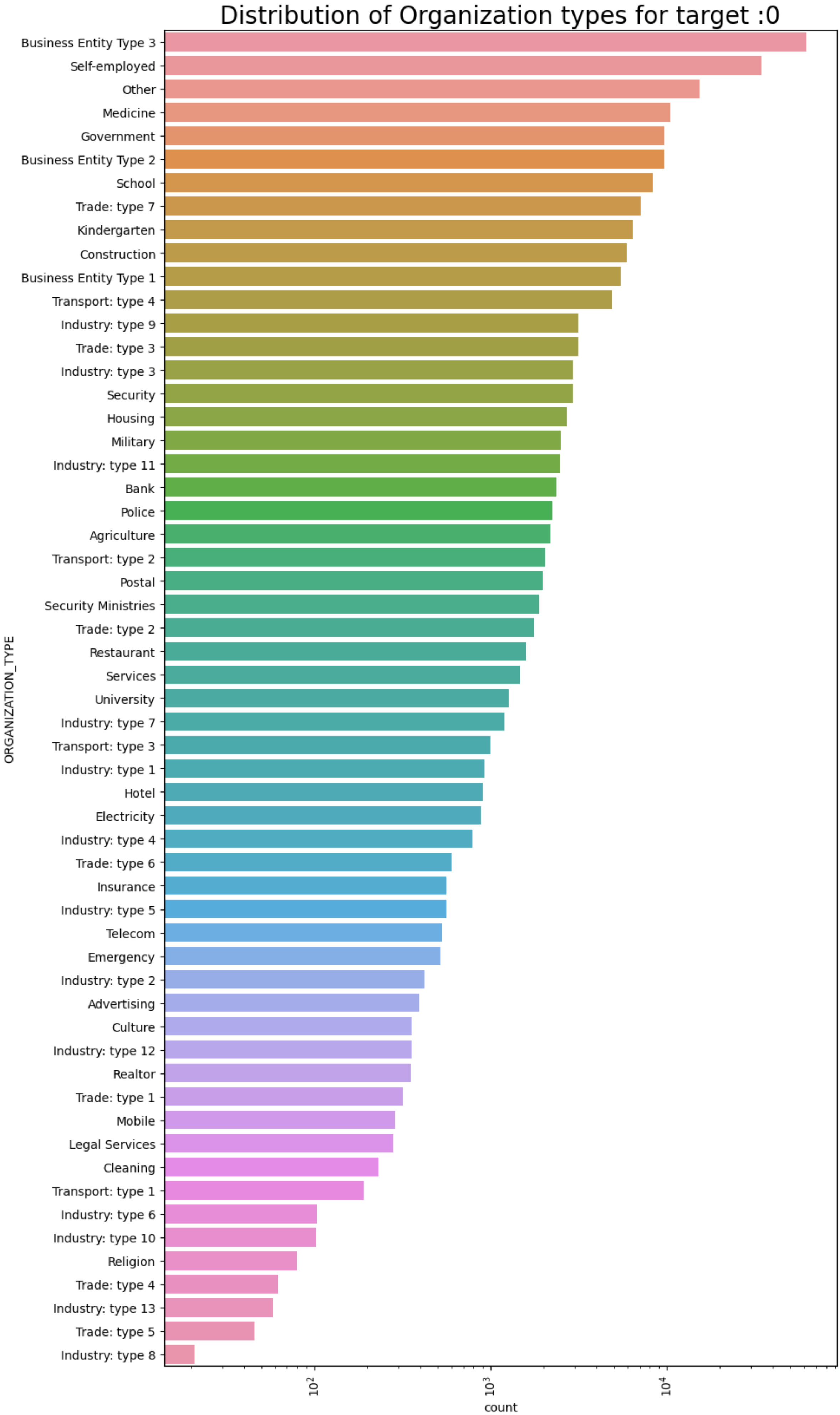
In [76]:
```
# PLOTTING FOR ORGANIZATION TYPE
countPlotForUnivariateAnalysis(df_target0,col='ORGANIZATION_TYPE', title='Distribu
```



Since it is difficult to interpret from the above graph we will create a graph for Organization type separately.

In [80]:
```
plt.figure(figsize=(10,20))
plt.rcParams["axes.labelsize"] = 10
plt.rcParams["axes.titlesize"] = 20
plt.title("Distribution of Organization types for target :0")
plt.xticks(rotation = 90)
plt.xscale('log')
sns.countplot(data=df_target0, y='ORGANIZATION_TYPE', order= df_target0['ORGANIZATI
```

Out[80]:
```
<AxesSubplot:title={'center':'Distribution of Organization types for target :0'},
xlabel='count', ylabel='ORGANIZATION_TYPE'>
```

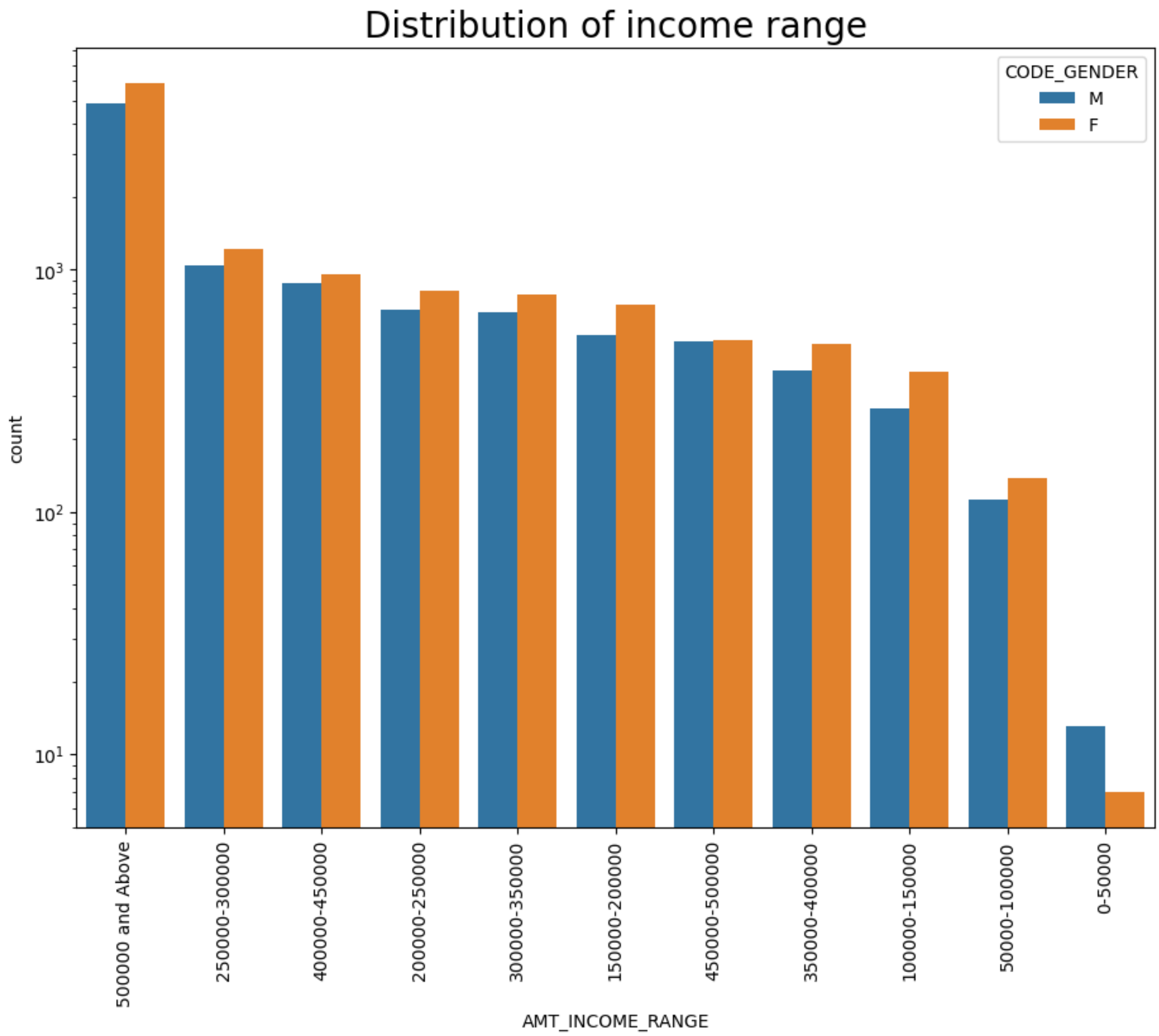


Insight from above graph

1. 'Business entity type 3' , 'Self employed', 'Other', 'Medicine' org types have applied for more credits compared to others.
2. There are few clients from 'Industry type 8', 'Trade type5'

Categorical Univariate Analysis - Target 1

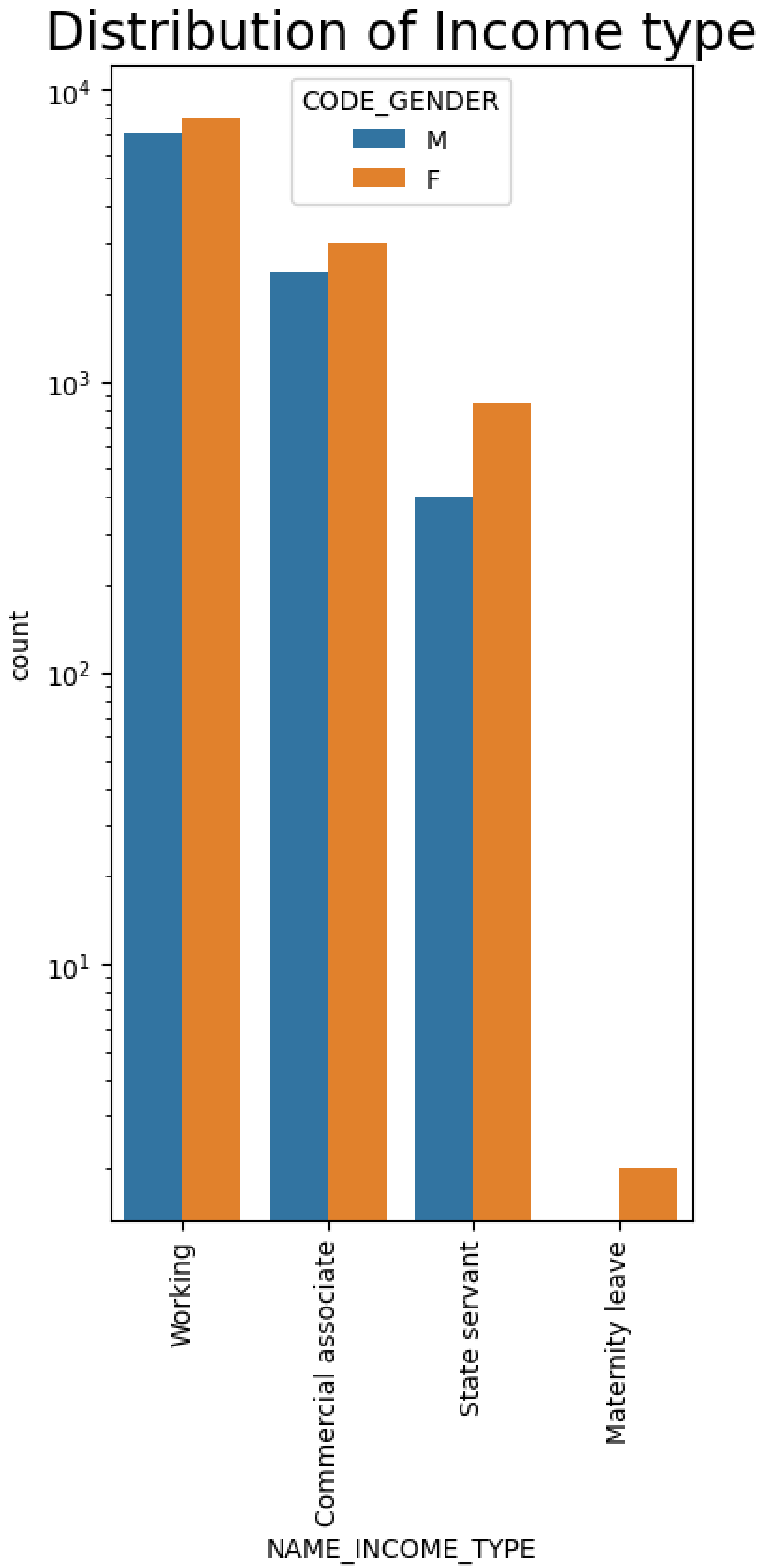
In [81]:
```
# Plotting for income range

countPlotForUnivariateAnalysis(df_target1,col="AMT_INCOME_RANGE", title= 'Distribu
```



Insights from above graph

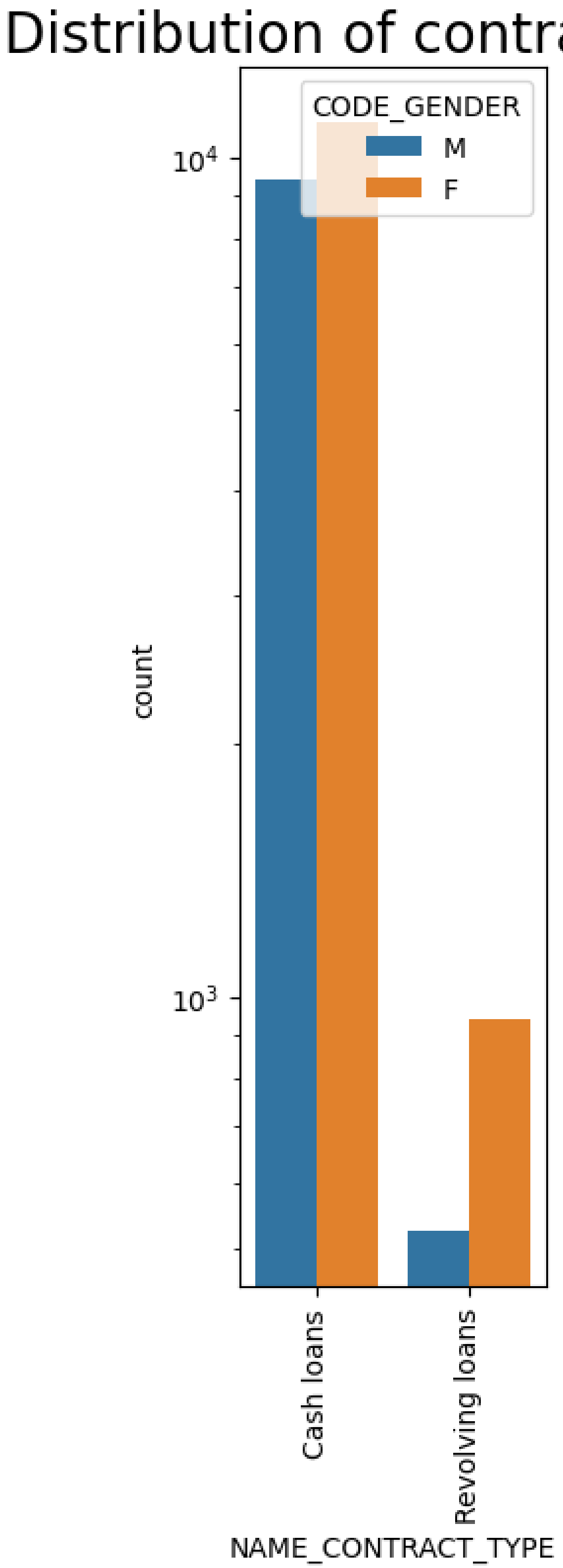
1. Female counts are higher than male .
2. Income range from 1,00,000 to 2,00,000 is having more number of credits.
3. This graoh show that females are more than male in having credits for that range.

In [83]:
```
# Plotting for income type

countPlotForUnivariateAnalysis(df_target1,col='NAME_INCOME_TYPE', title = 'Distribu
```



Insights from the graph

In [87]:
```
countPlotForUnivariateAnalysis(df_target1,col='NAME_CONTRACT_TYPE',title='Distribu
```



Insights from graph

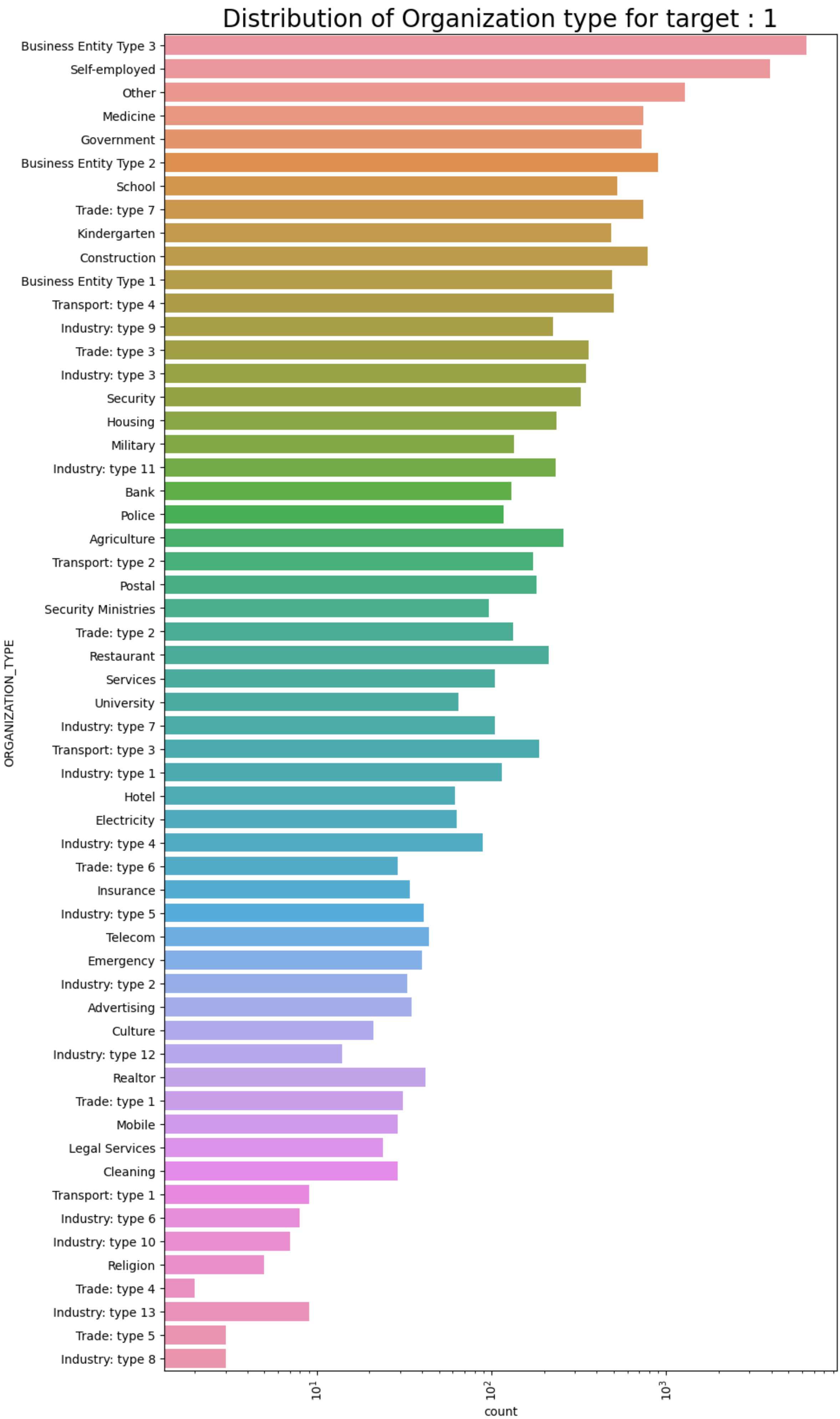
1. for contact type 'cash loans' is having a higher number of credits than 'Revolving loans' contract type.
2. For this reason , Women are also leading the way in applying for credits
3. For type 1 : there are only Female Revolving loans.

In [93]:
```
plt.figure(figsize=(10,20))
plt.rcParams["axes.labelsize"] = 10
```

```
plt.rcParams["axes.titlesize"] = 20
plt.title("Distribution of Organization type for target : 1")
plt.xticks(rotation=90)
plt.xscale('log')
sns.countplot(data=df_target1,y='ORGANIZATION_TYPE',order=df_target0['ORGANIZATION_
```

Out[93]:

```
<AxesSubplot:title={'center':'Distribution of Organization type for target : 1'},
xlabel='count', ylabel='ORGANIZATION_TYPE'>
```

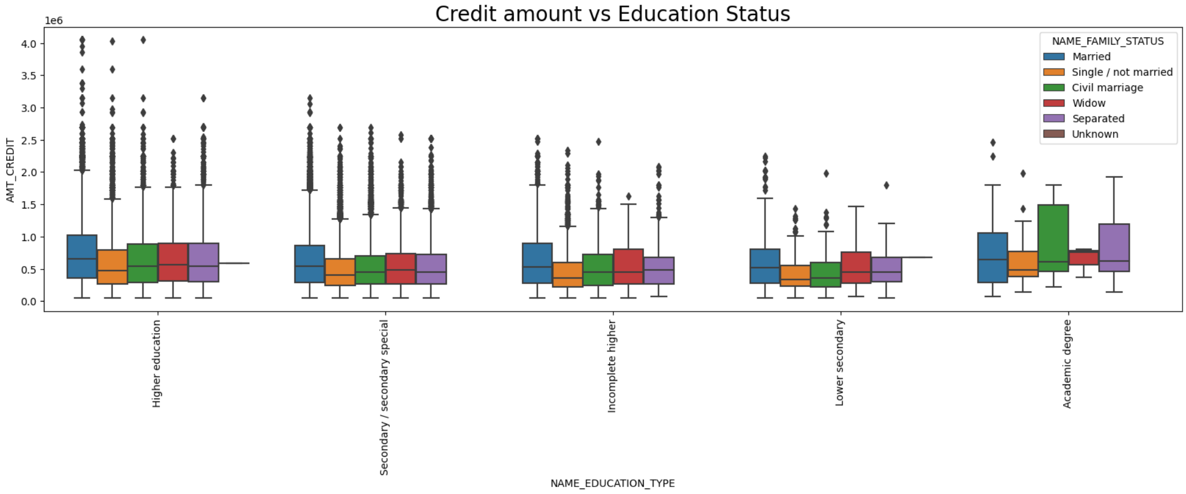


Insights from above graph

1. Clients which have applied for credits are from of the organisation type 'Business entity Type 3', 'Self employed', 'Other', 'Medicine' and 'Government'.
2. Less cients are from Industry type 8, 6 type, 10 religion and trade type 5, type 4.
3. Same as type 0 in distribution of organization type.

Bivariate Analysis --- Target0 ***

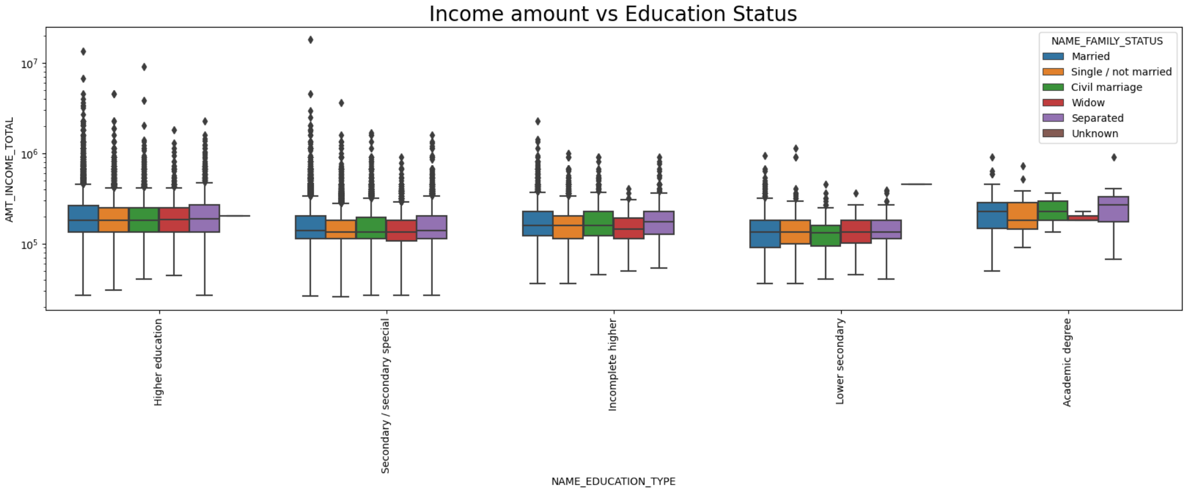
In [95]:
```
# Box plotting for credit amount

plt.figure(figsize=(20,5))
plt.xticks(rotation=90)
sns.boxplot(data= df_target0, x='NAME_EDUCATION_TYPE', y = 'AMT_CREDIT', hue = 'NA
plt.title('Credit amount vs Education Status')
plt.show()
```



Insight from above graph

From the above box plot we are able to conclude that family status of 'Civil marriage', 'MArriage',and 'Seprated' of academic degree education are having higher numbers of credits than other. Also , education of family status of 'marriage' , 'single' and 'civil marriage' are having more outliers. civil marriage for education degree is having most of the credtis within the third quartile.

In [96]:
```
# Box plotting for income amount in lagarithmic scale

plt.figure(figsize= (20,5))
plt.xticks(rotation = 90)
plt.yscale('log')
sns.boxplot(data = df_target0, x= 'NAME_EDUCATION_TYPE', y = 'AMT_INCOME_TOTAL', h
plt.title('Income amount vs Education Status')
plt.show()
```



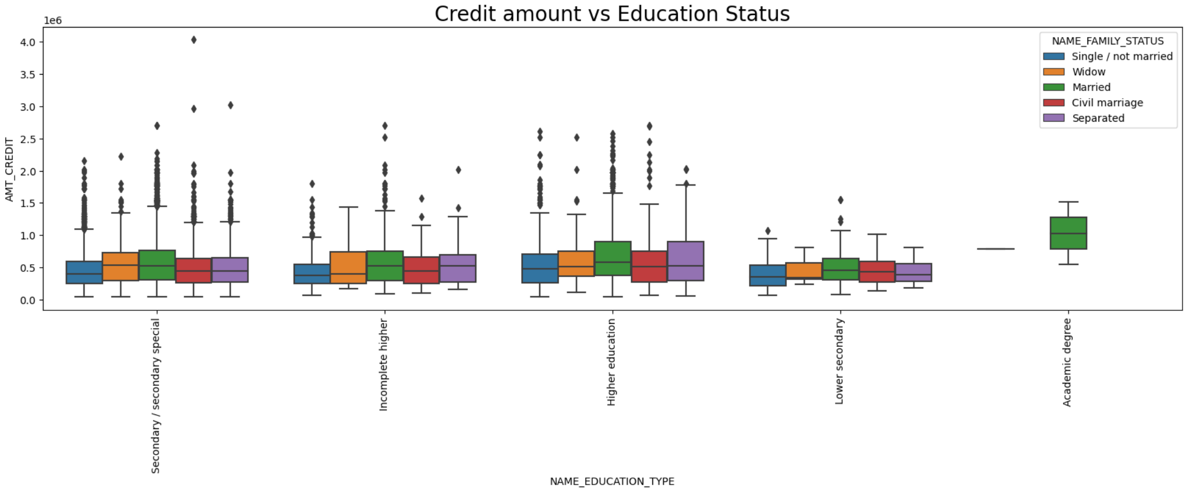
Insights from above graph

from above boxplot for education type ' Higher Education' on the income amount is usually equal with family status. it does contain many outliers. less outliers are having for academicsn degree but there income amount is little higher that higher education . Lower secondary of marriage family status are less income amount than others.

Bivariate Analysis - Target 1***

In [100…

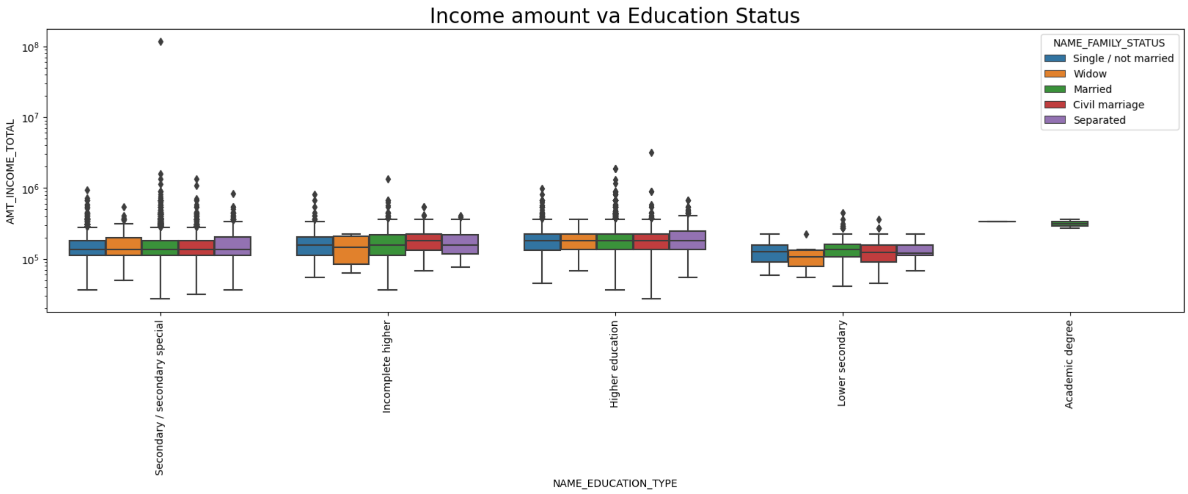
```
# Boxplotting for credit amount

plt.figure(figsize=(20,5))
plt.xticks(rotation =90)
sns.boxplot(data = df_target1, x='NAME_EDUCATION_TYPE', y = 'AMT_CREDIT', hue = 'N
plt.title('Credit amount vs Education Status')
plt.show()
```



Insight from above graph

From the above box plot we can say that family status of 'civil marriage' 'marriage',and 'separated' of academic degree education are having higher number of credits than others. Most of the outliers are from education type 'Higher education' and 'Secondary'. civil marriage for academics degree is having most of the credits in the third quartile.

In [102…
```
# Box plotting for income amount in logarithmic scale

plt.figure(figsize = (20, 5))
plt.xticks(rotation = 90)
plt.yscale('log')
sns.boxplot(data= df_target1, x = 'NAME_EDUCATION_TYPE', y = 'AMT_INCOME_TOTAL', hu
plt.title('Income amount va Education Status')
plt.show()
```



Insights from above graph

From above box plot for education type 'Higher education' the income amount is mostly equal with family status . less outliers are having for academics degree but there income amount is little higher that Higher education . Lower secondary are have less income amount than others.

CORRELATION FOR THE CLIENT WITH PAYMENT DIFFICULTIES AND ALL OTHER

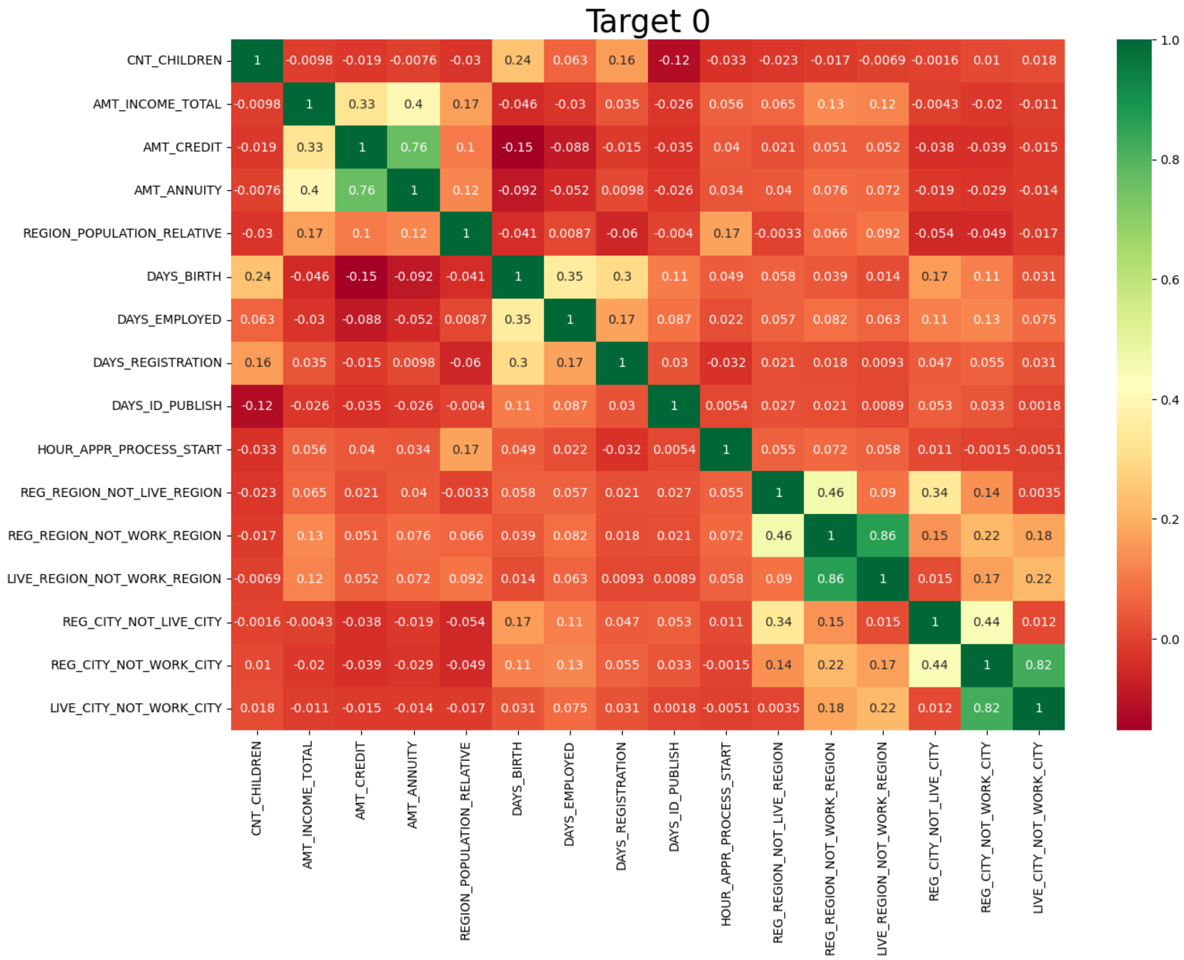
In [103…
```
# Find correlation between the numerical columns for target 0

df_target0_corr = df_target0.iloc[0:,2:]
target0=df_target0_corr.corr()
target0
```

Out[103]:

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNU
CNT_CHILDREN	1.000000	-0.009826	-0.018704	-0.00
AMT_INCOME_TOTAL	-0.009826	1.000000	0.326155	0.40
AMT_CREDIT	-0.018704	0.326155	1.000000	0.76
AMT_ANNUITY	-0.007612	0.400752	0.762103	1.00
REGION_POPULATION_RELATIVE	-0.030352	0.169306	0.103876	0.12
DAYS_BIRTH	0.242462	-0.045543	-0.152659	-0.09
DAYS_EMPLOYED	0.063036	-0.030102	-0.087500	-0.05
DAYS_REGISTRATION	0.162900	0.034508	-0.015180	0.00
DAYS_ID_PUBLISH	-0.117746	-0.026462	-0.034914	-0.02
HOUR_APPR_PROCESS_START	-0.033031	0.055934	0.040390	0.03
REG_REGION_NOT_LIVE_REGION	-0.023033	0.064868	0.020979	0.03
REG_REGION_NOT_WORK_REGION	-0.016798	0.129765	0.050597	0.07
LIVE_REGION_NOT_WORK_REGION	-0.006946	0.121288	0.052028	0.07
REG_CITY_NOT_LIVE_CITY	-0.001566	-0.004264	-0.037527	-0.01
REG_CITY_NOT_WORK_CITY	0.010369	-0.020260	-0.038517	-0.02
LIVE_CITY_NOT_WORK_CITY	0.018414	-0.011238	-0.014834	-0.01

In [105…

```python
# plotting Heatmap for above correlation

plt.figure(figsize=(15,10))
plt.rcParams['axes.titlesize'] = 25

sns.heatmap(target0, cmap = 'RdYlGn' , annot = True)

plt.title("Target 0")
plt.yticks(rotation = 0)
plt.show()
```



Insights from above graph

1. Credit amount is inversely proportional to the date of birth , which means Credit amount is higher for low age and vice versa.
2. Credit amount is inversely proportional to the number of children client have, means credit amount is higher for less children count client have and vice versa
3. Income amount is inversely proportional to the number of children client have , means more income for less children client have and vice versa.

In [107…

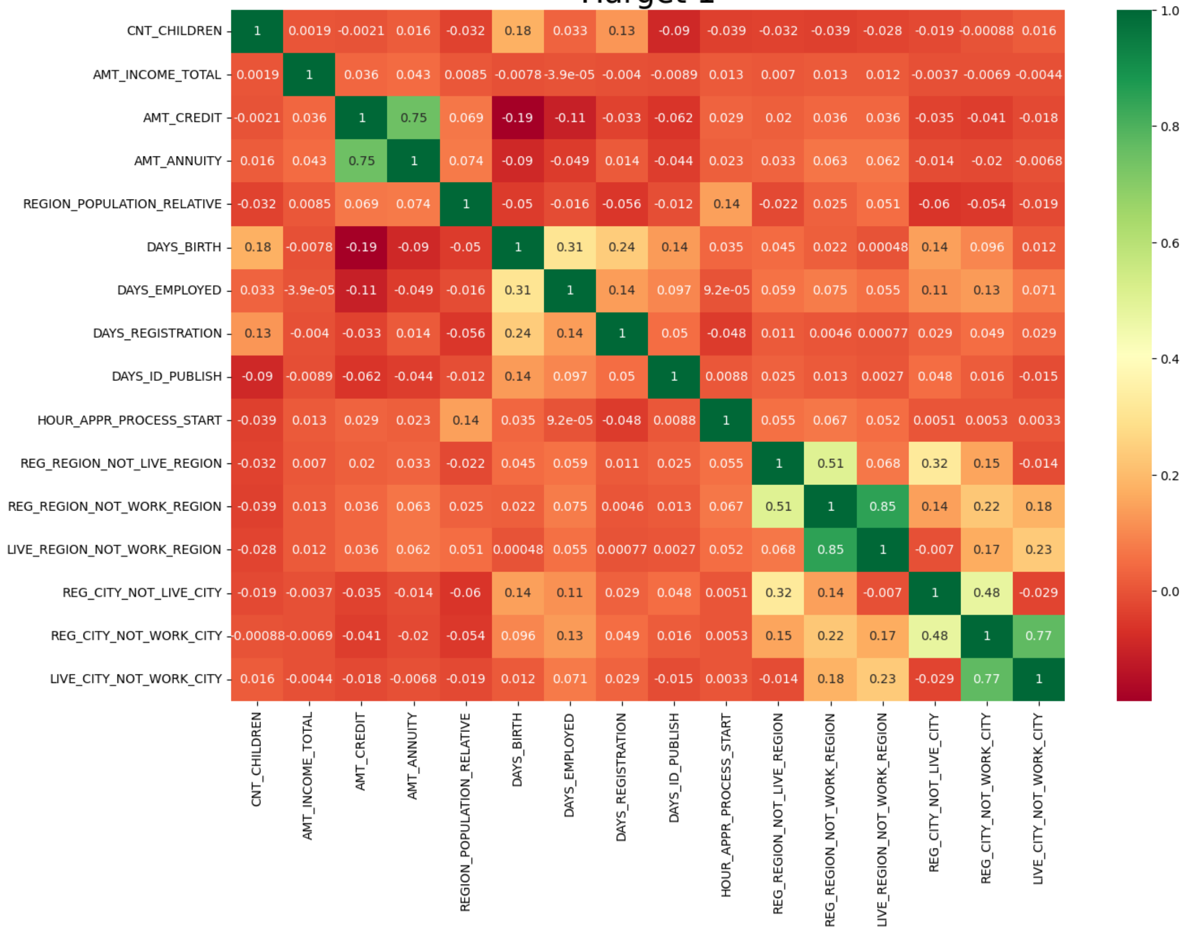
```
# Find correlation between the numerical columns for target1

df_target1_corr = df_target1.iloc[0:,2:]
target1 = df_target1_corr.corr()
target1
```

Out[107]:

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNU
CNT_CHILDREN	1.000000	0.001872	-0.002074	0.01
AMT_INCOME_TOTAL	0.001872	1.000000	0.036484	0.04
AMT_CREDIT	-0.002074	0.036484	1.000000	0.74
AMT_ANNUITY	0.015653	0.043358	0.748708	1.00
REGION_POPULATION_RELATIVE	-0.032019	0.008476	0.069220	0.07
DAYS_BIRTH	0.176563	-0.007822	-0.189512	-0.09
DAYS_EMPLOYED	0.032627	-0.000039	-0.106003	-0.04
DAYS_REGISTRATION	0.126411	-0.003959	-0.033250	0.01
DAYS_ID_PUBLISH	-0.089861	-0.008858	-0.062405	-0.04
HOUR_APPR_PROCESS_START	-0.038923	0.012520	0.029054	0.02
REG_REGION_NOT_LIVE_REGION	-0.032465	0.006951	0.020083	0.03
REG_REGION_NOT_WORK_REGION	-0.039498	0.013245	0.035695	0.06
LIVE_REGION_NOT_WORK_REGION	-0.028031	0.012287	0.035966	0.06
REG_CITY_NOT_LIVE_CITY	-0.019278	-0.003664	-0.035325	-0.01
REG_CITY_NOT_WORK_CITY	-0.000876	-0.006886	-0.041392	-0.01
LIVE_CITY_NOT_WORK_CITY	0.016332	-0.004401	-0.017875	-0.00

In [111…

```
# Plotting heatmap for above correlation

plt.figure(figsize =(15, 10))
plt.rcParams['axes.titlesize'] = 25

sns.heatmap(target1,cmap="RdYlGn" ,annot =True)

plt.title(":Target 1")
plt.yticks(rotation=0)
plt.show()
```



Insight from above graph

1. The client's permanent address doesnot match contact address are having less children and vice-versa.
2. The client's permanent address doesnot match work address are having less children and vice-versa.

PREVIOUS_DATA

This data is about whether the previous application had been approved, Cancelled, Refused or Unused offer.

By taking previous application into consideration for Analysis

In [112…
```
previous_df = pd.read_csv("E:\previous_application.csv")
```

In [113…
```
previous_df.shape
```

Out[113]:
```
(1670214, 37)
```

In [114…
```
# identifying and cleaning the missing values which are greater than 30%

nullColumns = previous_df.isnull().sum()
```

```
nullColumns = nullColumns[nullColumns.values > (0.3*len(nullColumns))]
len(nullColumns)
```

Out[114]:

```
15
```

In [115…

```
# removing 15 columns

previous_df.drop(labels=list(nullColumns.index), axis=1, inplace=True)

previous_df.shape
```

Out[115]:

```
(1670214, 22)
```

In [116…

```
previous_df.dtypes
```

Out[116]:

```
SK_ID_PREV                       int64
SK_ID_CURR                       int64
NAME_CONTRACT_TYPE              object
AMT_APPLICATION                float64
AMT_CREDIT                     float64
WEEKDAY_APPR_PROCESS_START      object
HOUR_APPR_PROCESS_START          int64
FLAG_LAST_APPL_PER_CONTRACT     object
NFLAG_LAST_APPL_IN_DAY           int64
NAME_CASH_LOAN_PURPOSE          object
NAME_CONTRACT_STATUS            object
DAYS_DECISION                    int64
NAME_PAYMENT_TYPE               object
CODE_REJECT_REASON              object
NAME_CLIENT_TYPE                object
NAME_GOODS_CATEGORY             object
NAME_PORTFOLIO                  object
NAME_PRODUCT_TYPE               object
CHANNEL_TYPE                    object
SELLERPLACE_AREA                 int64
NAME_SELLER_INDUSTRY            object
NAME_YIELD_GROUP                object
dtype: object
```

In [117…

```
previous_df.NAME_CASH_LOAN_PURPOSE.value_counts()
```

Out[117]:

```
XAP                                 922661
XNA                                 677918
Repairs                              23765
Other                                15608
Urgent needs                          8412
Buying a used car                     2888
Building a house or an annex          2693
Everyday expenses                     2416
Medicine                              2174
Payments on other loans               1931
Education                             1573
Journey                               1239
Purchase of electronic equipment      1061
Buying a new car                      1012
Wedding / gift / holiday               962
Buying a home                          865
Car repairs                            797
Furniture                              749
Buying a holiday home / land           533
Business development                   426
Gasification / water supply            300
Buying a garage                        136
Hobby                                   55
Money for a third person                25
Refusal to name the goal                15
Name: NAME_CASH_LOAN_PURPOSE, dtype: int64
```

In [119…

```
# Removing the column values of 'XNA' AND 'XAP'

previous_df= previous_df[~(previous_df['NAME_CASH_LOAN_PURPOSE']=='XNA')]
previous_df= previous_df[~(previous_df['NAME_CASH_LOAN_PURPOSE']=='XAP')]

previous_df.shape
```

Out[119]:

```
(69635, 22)
```

MERGING TWO DATAFRAMES

In [126…

```
# merging both the data frames

df = pd.merge(left = df,right = previous_df, how='inner', on = "SK_ID_CURR",suffix
df.shape
```

Out[126]:

```
(145854, 72)
```

In [127…

```
df.columns
```

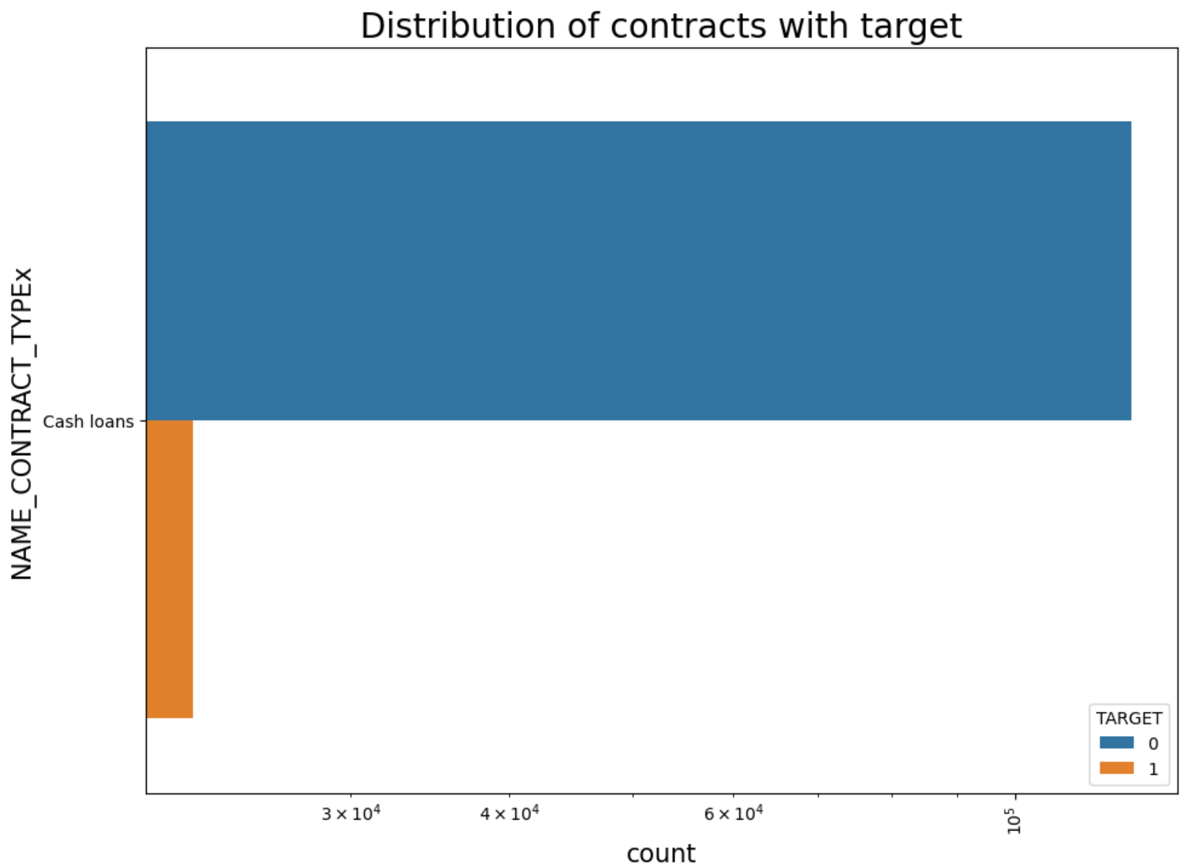
Out[127]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE_', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT_', 'AMT_ANNUITY', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE',
       'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH',
       'WEEKDAY_APPR_PROCESS_START_', 'HOUR_APPR_PROCESS_START_',
       'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY',
       'ORGANIZATION_TYPE', 'AMT_INCOME_RANGE', 'AMT_CREDIT_RANGE',
       'SK_ID_PREV_', 'NAME_CONTRACT_TYPEx', 'AMT_APPLICATION_', 'AMT_CREDITx',
       'WEEKDAY_APPR_PROCESS_STARTx', 'HOUR_APPR_PROCESS_STARTx',
       'FLAG_LAST_APPL_PER_CONTRACT_', 'NFLAG_LAST_APPL_IN_DAY_',
       'NAME_CASH_LOAN_PURPOSE_', 'NAME_CONTRACT_STATUS_', 'DAYS_DECISION_',
       'NAME_PAYMENT_TYPE_', 'CODE_REJECT_REASON_', 'NAME_CLIENT_TYPE_',
       'NAME_GOODS_CATEGORY_', 'NAME_PORTFOLIO_', 'NAME_PRODUCT_TYPE_',
       'CHANNEL_TYPE_', 'SELLERPLACE_AREA_', 'NAME_SELLER_INDUSTRY_',
       'NAME_YIELD_GROUP_', 'SK_ID_PREVx', 'NAME_CONTRACT_TYPE',
       'AMT_APPLICATIONx', 'AMT_CREDIT', 'WEEKDAY_APPR_PROCESS_START',
       'HOUR_APPR_PROCESS_START', 'FLAG_LAST_APPL_PER_CONTRACTx',
       'NFLAG_LAST_APPL_IN_DAYx', 'NAME_CASH_LOAN_PURPOSEx',
       'NAME_CONTRACT_STATUSx', 'DAYS_DECISIONx', 'NAME_PAYMENT_TYPEx',
       'CODE_REJECT_REASONx', 'NAME_CLIENT_TYPEx', 'NAME_GOODS_CATEGORYx',
       'NAME_PORTFOLIOx', 'NAME_PRODUCT_TYPEx', 'CHANNEL_TYPEx',
       'SELLERPLACE_AREAx', 'NAME_SELLER_INDUSTRYx', 'NAME_YIELD_GROUPx'],
      dtype='object')
```

In [128…
```
df.dtypes
```

Out[128]:
```
SK_ID_CURR                int64
TARGET                    int64
NAME_CONTRACT_TYPE_      object
CODE_GENDER              object
FLAG_OWN_CAR             object
                          ...
NAME_PRODUCT_TYPEx       object
CHANNEL_TYPEx            object
SELLERPLACE_AREAx         int64
NAME_SELLER_INDUSTRYx    object
NAME_YIELD_GROUPx        object
Length: 72, dtype: object
```

Univariate Analysis

In [131…
```
# Distribution of contract status

plt.figure(figsize=(11,8))
plt.rcParams["axes.labelsize"] =15
plt.rcParams['axes.titlesize'] = 20

plt.xticks(rotation = 90)
plt.xscale('log')
plt.title('Distribution of contracts with target')
ax = sns.countplot(data = df, y = 'NAME_CONTRACT_TYPEx', order = df['NAME_CONTRACT_
plt.show()
```

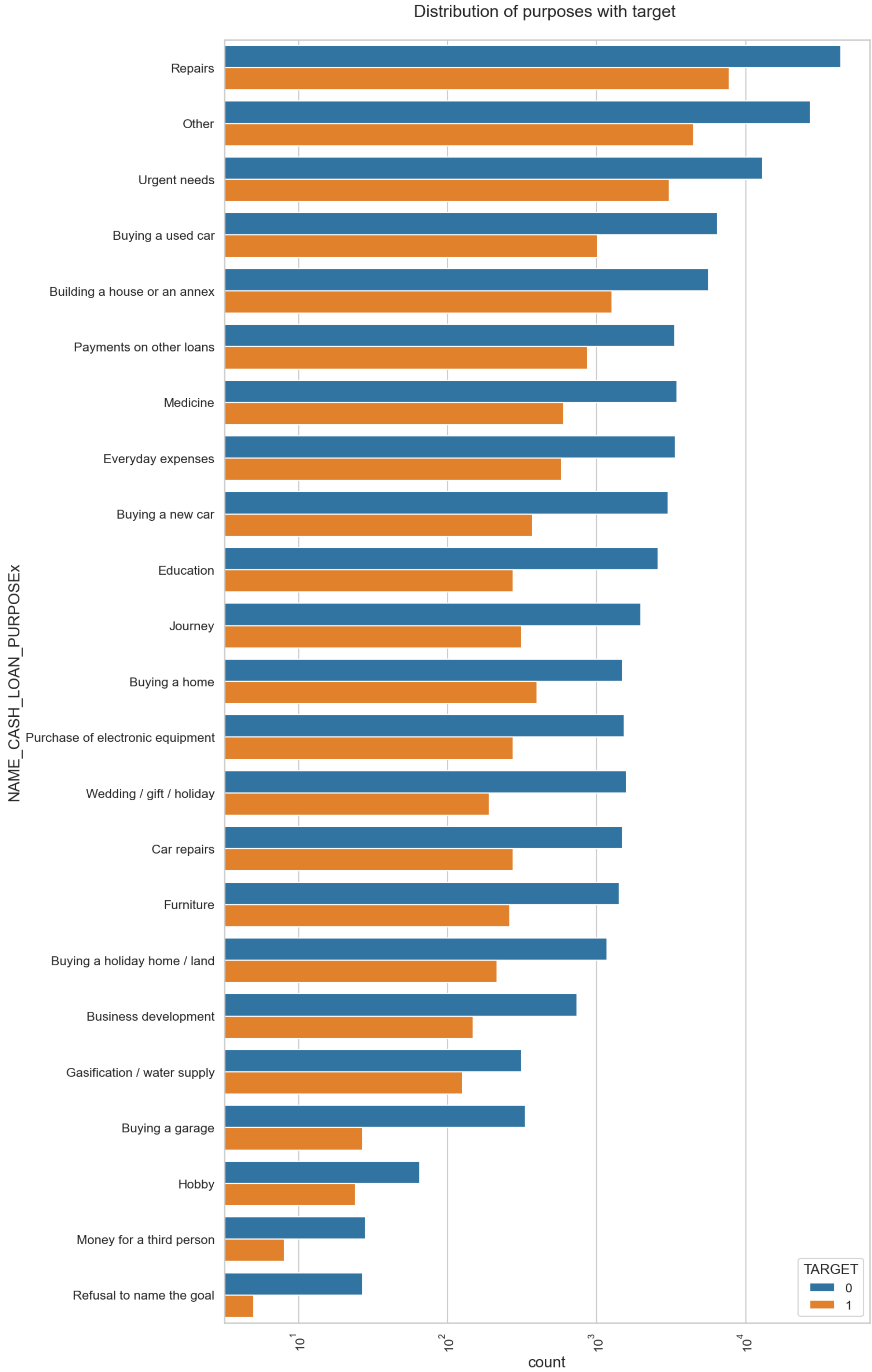


Insight from above graph

1. Most rejection of loans came from purpose 'repairs'.
2. For education purposes we have equal number of approves and rejection.
3. Paying other loans and buying a new car is having significant higher rejection than approves.

In [143…

```
# Distribution of contracts status:
sns.set_style('whitegrid')
sns.set_context('talk')

plt.figure(figsize=(15,30))
plt.rcParams["axes.labelsize"] = 20
plt.rcParams['axes.titlesize'] = 22
plt.rcParams['axes.titlepad'] = 30
plt.xticks(rotation=90)
plt.xscale('log')
plt.title('Distribution of purposes with target ')
ax = sns.countplot(data =df, y= 'NAME_CASH_LOAN_PURPOSEx',
                   order=df['NAME_CASH_LOAN_PURPOSEx'].value_counts().index,hue =
```



insights from above graph

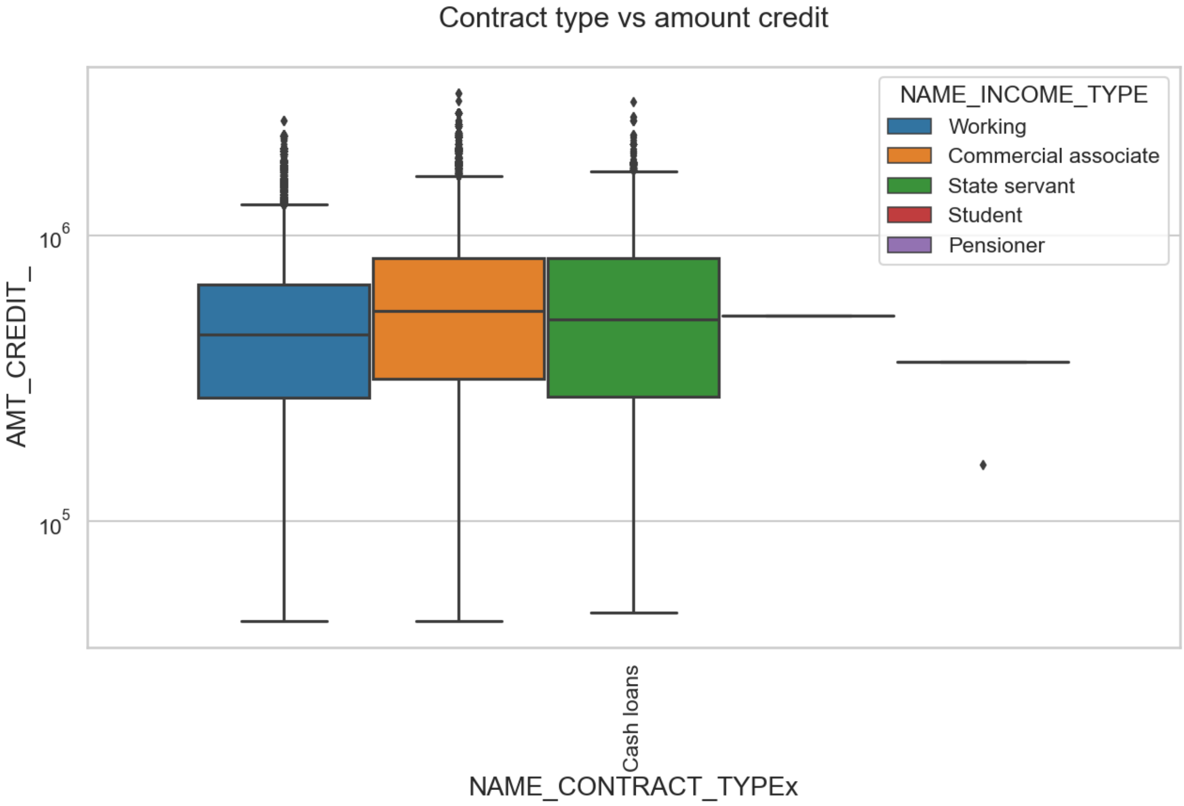
1. Loan purpose witb 'Repairs' are facing more difficulties in paymrnt on time.

2. There are few places where loan payment is significant higher than facing difficulties . They aer 'Buying a garage' ,' Business development', ' Buying a new car', and 'Education'.

Bivariate Analysis

In [138…

```
plt.figure(figsize=(15, 8))
plt.xticks(rotation=90)
plt.yscale('log')
sns.boxplot(data= df, x='NAME_CONTRACT_TYPEx', hue='NAME_INCOME_TYPE', y= 'AMT_CREI
plt.title('Contract type vs amount credit')
plt.show()
```

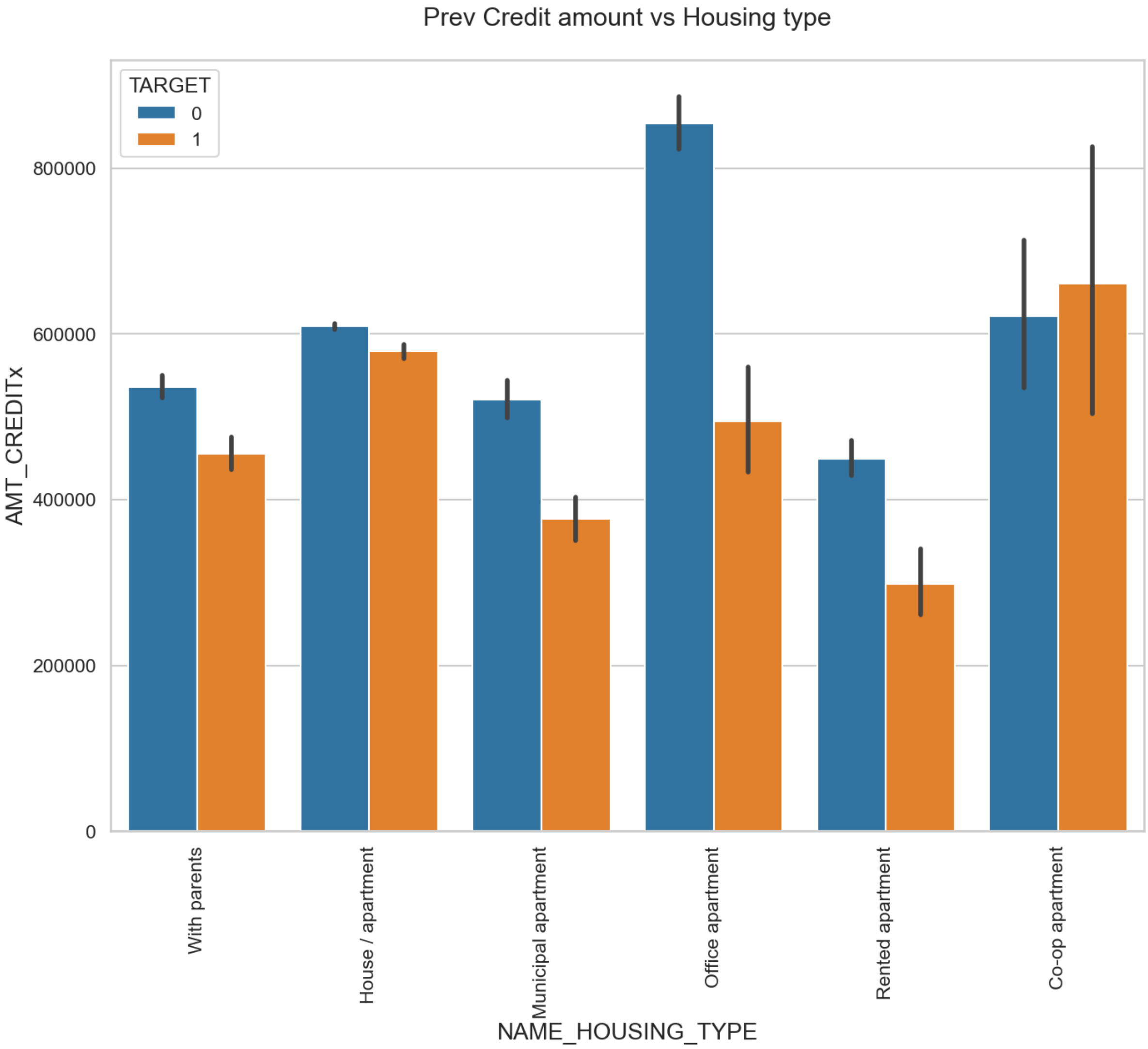


insights from above graph

1. The credit amount of loan purpose lije 'Buying a land', 'Buying a new car' and 'building a house' is higher.
2. Income type of state servants have a significant amount of credit applied.
3. Money for third person or a hobby is having less credits applied for.

In [139…

```
# Box plotting for credit amount prev vs Housing type in logarithmic sca;e

plt.figure(figsize=(16,12))
plt.xticks(rotation=90)
sns.barplot(data=df, y = 'AMT_CREDITx', hue= 'TARGET',x= 'NAME_HOUSING_TYPE')
plt.title('Prev Credit amount vs Housing type')
plt.show()
```



Insights from above graph

Here, for housing type , office apartment is having higher credit of target 0 and co-op apartment is having higher credit of target1. So we can conclude that the bank should avoid giving loans tot he housing type of co-op apartment as they are having difficulties in payment. Bank can focus mostly on housing type with parents or House/ apartment or municipal apartment for successful payments.

CONCLUSION

1. Banks should focus more on contract type 'Student', 'Pensioner',and 'Businessman' with housing type other than 'co-op apartment' for successful payments.
2. Banks should focus less on income type 'working' as they are having most number of unsuccessful payments.
3. Also with loan purpose 'Repair' is having higher number of unsucessful payment on time.
4. Get as much as clients from housing type 'with parents' as they are having least number of unsucessful payments.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: