Credit EDA Case Study

Uncovering insights hidden in data.

Overview

This assignment aims to give an idea of applying EDA in a real business scenario. In this assignment, apart from applying the techniques of the EDA, I also developed a basic understanding of risk analytics in banking and financial services and understand how data is used to minimize the risk of losing money while lending to customers.

Business Understanding

Building a foundation of knowledge for informed decisions.

The loan-providing companies find it hard to give loans to people due to their insufficient or non-existent credit history. Because of that, some consumers use it to their advantage by becoming defaulters. Suppose you work for a consumer finance company that specializes in lending various types of loans to urban customers. You have to use EDA to analyze 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 on 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 for 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.

Data Understanding

Discovering the story behind your data

The dataset I used 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.**

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

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

Prepared by: Rishabh Tiwari

Date: 19/07/2022

Credit EDA Case Study

Problem Statement

This case study aims to identify patterns which indicate if a client has difficulty paying their instalments 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.

Reading and Understanding the Data

In [1]:

```
# Importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
# Reading application_data

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

In [3]:

```
df.head()
```

Out[3]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLA
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 rows × 122 columns

In [4]:

```
# Checking number of columns and rows

df.shape
```

Out[4]:

```
(307511, 122)
```

In [5]:

```
df.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
```

Null Value Treatment

In [6]:

```
null_val = df.isnull().mean() *100
```

In [7]:

```
null_val
```

Out[7]:

```
SK_ID_CURR                     0.000000
TARGET                         0.000000
NAME_CONTRACT_TYPE             0.000000
CODE_GENDER                    0.000000
FLAG_OWN_CAR                   0.000000
                                  ...
AMT_REQ_CREDIT_BUREAU_DAY     13.501631
AMT_REQ_CREDIT_BUREAU_WEEK    13.501631
AMT_REQ_CREDIT_BUREAU_MON     13.501631
AMT_REQ_CREDIT_BUREAU_QRT     13.501631
AMT_REQ_CREDIT_BUREAU_YEAR    13.501631
Length: 122, dtype: float64
```

In [8]:

```
# Finding the columns which are having null values more than 40%

col_null = df.isnull().sum()
col_null= col_null[col_null>0.4*(len(col_null))]
len(col_null)
```

Out[8]:

```
64
```

In [9]:

```
# Dropping the columns with null values more than 40%

col_null = list(col_null[col_null.values>=0.4].index)
df.drop(labels= col_null,axis =1, inplace= True)
```

In [10]:

```
# Checking the percentage of null values present after dropping

df.isnull().sum()/len(df)*100
```

Out[10]:

```
SK_ID_CURR                     0.000000
TARGET                         0.000000
NAME_CONTRACT_TYPE             0.000000
CODE_GENDER                    0.000000
FLAG_OWN_CAR                   0.000000
FLAG_OWN_REALTY                0.000000
CNT_CHILDREN                   0.000000
AMT_INCOME_TOTAL               0.000000
AMT_CREDIT                     0.000000
AMT_ANNUITY                    0.003902
NAME_INCOME_TYPE               0.000000
NAME_EDUCATION_TYPE            0.000000
NAME_FAMILY_STATUS             0.000000
NAME_HOUSING_TYPE              0.000000
REGION_POPULATION_RELATIVE     0.000000
DAYS_BIRTH                     0.000000
DAYS_EMPLOYED                  0.000000
DAYS_REGISTRATION              0.000000
DAYS_ID_PUBLISH                0.000000
FLAG_MOBIL                     0.000000
FLAG_EMP_PHONE                 0.000000
FLAG_WORK_PHONE                0.000000
FLAG_CONT_MOBILE               0.000000
FLAG_PHONE                     0.000000
FLAG_EMAIL                     0.000000
CNT_FAM_MEMBERS                0.000650
REGION_RATING_CLIENT           0.000000
REGION_RATING_CLIENT_W_CITY    0.000000
WEEKDAY_APPR_PROCESS_START     0.000000
HOUR_APPR_PROCESS_START        0.000000
REG_REGION_NOT_LIVE_REGION     0.000000
REG_REGION_NOT_WORK_REGION     0.000000
LIVE_REGION_NOT_WORK_REGION    0.000000
REG_CITY_NOT_LIVE_CITY         0.000000
REG_CITY_NOT_WORK_CITY         0.000000
LIVE_CITY_NOT_WORK_CITY        0.000000
ORGANIZATION_TYPE              0.000000
DAYS_LAST_PHONE_CHANGE         0.000325
FLAG_DOCUMENT_2                0.000000
FLAG_DOCUMENT_3                0.000000
FLAG_DOCUMENT_4                0.000000
FLAG_DOCUMENT_5                0.000000
FLAG_DOCUMENT_6                0.000000
FLAG_DOCUMENT_7                0.000000
FLAG_DOCUMENT_8                0.000000
FLAG_DOCUMENT_9                0.000000
FLAG_DOCUMENT_10               0.000000
FLAG_DOCUMENT_11               0.000000
FLAG_DOCUMENT_12               0.000000
FLAG_DOCUMENT_13               0.000000
FLAG_DOCUMENT_14               0.000000
FLAG_DOCUMENT_15               0.000000
FLAG_DOCUMENT_16               0.000000
FLAG_DOCUMENT_17               0.000000
FLAG_DOCUMENT_18               0.000000
FLAG_DOCUMENT_19               0.000000
FLAG_DOCUMENT_20               0.000000
FLAG_DOCUMENT_21               0.000000
dtype: float64
```

Points to be concluded:

- application_data has a shape of 307511 rows and 122 columns.
- It contains float64(65), int64(41), object(16).
- There are 64 columns with null values more than 40%

In [11]:

```
# Checking for the null values of column

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

Out[11]:

```
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
```

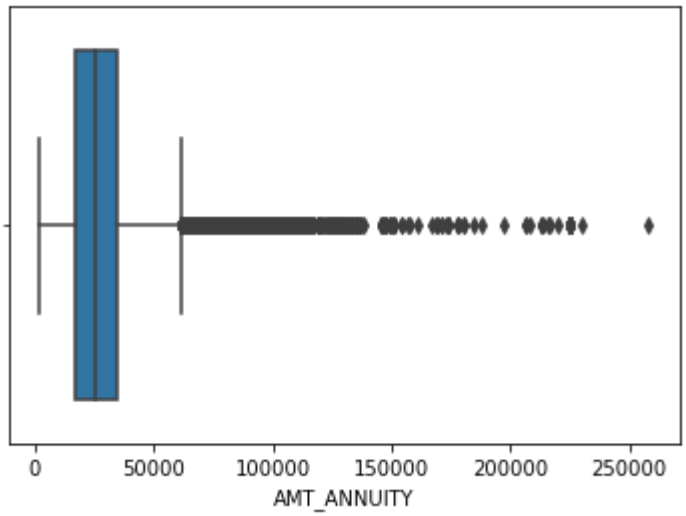
AMT_ANNUITY has 12 null values, CNT_FAM_MEMBERS has 2 null values and DAYS_LAST_PHONE_CHANGE has 1 null value.

In [12]:

```
# Treatment of AMT_ANNUITY null values

sns.boxplot(df['AMT_ANNUITY'])
```

Out[12]:

```
<AxesSubplot:xlabel='AMT_ANNUITY'>
```



As there are some outliers in the AMT_ANNUITY which may cause inaccuracy, so it will be better to treat those null values with the median but not mean.

In [13]:

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

In [14]:

```
# Checking the null values list again for verification of treatment

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

Out[14]:

```
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
```

Now there is no null values, except CNT_FAM_MEMBERS, DAYS_LAST_PHONE_CHANGE which has very less amount and it could not cause any problem in analysis.

In [15]:

```
# Removing rows having null values greater than or equal to 40%

emptyrow = df.isnull().sum(axis=1)
emptyrow = list(emptyrow[emptyrow.values>=0.4*len(df)].index)
df.drop(labels=emptyrow, axis=0,inplace=True)
print(len(emptyrow))
```

```
0
```

In [16]:

```
# Removing unwanted columns from dataset

unwant_col = ['FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE',
       'FLAG_PHONE', 'FLAG_EMAIL','REGION_RATING_CLIENT','REGION_RATING_CLIENT_W_CITY','
       'REGION_RATING_CLIENT_W_CITY','DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_2', 'FLAG_
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9','FLAG_DOCUMENT_10', 'FLAG
       'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15','FLAG_DOCUMENT_16', 'F
       'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21']

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

There are some columns where the value is mentioned as 'XNA' which means 'Not Available'. So we have to find the number of rows and columns and implement suitable techniques on them to fill those missing values or to delete them.

In [17]:

```
# let's find these categorical columns having these 'XNA' values

# For Gender column

df[df['CODE_GENDER']=='XNA'].shape
```

Out[17]:

```
(4, 28)
```

In [18]:

```
# For Organization column

df[df['ORGANIZATION_TYPE']=='XNA'].shape
```

Out[18]:

```
(55374, 28)
```

So, there are 4 rows from Gender column and 55374 rows from Organization type column

In [19]:

```
# Describing the Gender column to check the number of females and males

df['CODE_GENDER'].value_counts()
```

Out[19]:

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

Since, Female is having the majority and only 4 rows are having NA values, we can update those columns with Gender 'F' as there will be no impact on the dataset.

In [20]:

```
# Updating the column 'CODE_GENDER' with "F" for the dataset

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

Out[20]:

```
F    202452
M    105059
Name: CODE_GENDER, dtype: int64
```

In [21]:

```
# Describing the organization type column

df['ORGANIZATION_TYPE'].describe()
```

Out[21]:

```
count                    307511
unique                       58
top       Business Entity Type 3
freq                      67992
Name: ORGANIZATION_TYPE, dtype: object
```

So, for column 'ORGANIZATION_TYPE', we have total count of 307511 rows of which 55374 rows are having 'XNA' values. Which means 18% of the column is having this values. Hence if we drop the rows of total 55374, will not have any major impact on our dataset.

In [22]:

```
# Hence, dropping the rows of total 55374 have 'XNA' values in the organization type col

df=df.drop(df.loc[df['ORGANIZATION_TYPE']=='XNA'].index)
df[df['ORGANIZATION_TYPE']=='XNA'].shape
```

Out[22]:

```
(0, 28)
```

In [23]:

```
# Casting all variable into numeric in the dataset

numeric_columns = ['TARGET','CNT_CHILDREN','AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY'
                'DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH','HOUR_APPR_PROCESS_
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY']

df[numeric_columns]=df[numeric_columns].apply(pd.to_numeric)
df.head(5)
```

Out[23]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLA(
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 rows × 28 columns

Derived Metrics

Now, Creating bins for continous variable categories column 'AMT_INCOME_TOTAL' and 'AMT_CREDIT'

In [24]:

```
# Creating bins for income amount

bins = [0,25000,50000,75000,100000,125000,150000,175000,200000,225000,250000,275000,3000
slot = ['0-25000', '25000-50000','50000-75000','75000,100000','100000-125000', '125000-1
       '200000-225000','225000-250000','250000-275000','275000-300000','300000-325000','
       '375000-400000','400000-425000','425000-450000','450000-475000','475000-500000','

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

In [25]:

```
# Creating bins for Credit amount

bins = [0,150000,200000,250000,300000,350000,400000,450000,500000,550000,600000,650000,7
slots = ['0-150000', '150000-200000','200000-250000', '250000-300000', '300000-350000',
        '450000-500000','500000-550000','550000-600000','600000-650000','650000-700000',
        '800000-850000','850000-900000','900000 and above']

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

In [26]:

```
# Dividing the dataset into two dataset of  target=1(client with payment difficulties) a

target0_df=df.loc[df["TARGET"]==0]
target1_df=df.loc[df["TARGET"]==1]
```

In [27]:

```
# Calculating Imbalance percentage
# Since the majority is target0 and minority is target1

round(len(target0_df)/len(target1_df),2)
```

Out[27]:

```
10.55
```

The imbalance ratio is 10.55

Univariate Analysis for Categories

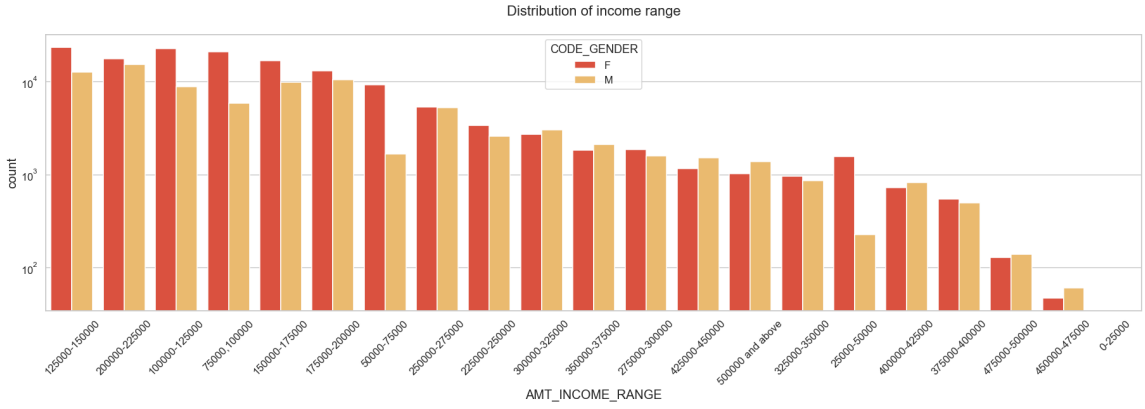
Categorical Univariate Analysis in logarithmic scale for target= 0 (client with no payment difficulties)

In [28]:

```
# Count plotting in logarithmic scale

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

    sns.set_style('whitegrid')
    sns.set_context('talk')
    plt.rcParams["axes.labelsize"] = 20
    plt.rcParams['axes.titlesize'] = 22
    plt.rcParams['axes.titlepad'] = 30


    temp = pd.Series(data = hue)
    fig, ax = plt.subplots()
    width = len(df[col].unique()) + 7 + 4*len(temp.unique())
    fig.set_size_inches(width , 8)
    plt.xticks(rotation=45)
    plt.yscale('log')
    plt.title(title)
    ax = sns.countplot(data = df, x= col, order=df[col].value_counts().index,hue = hue,p

    plt.show()
```

In [29]:

```
# PLotting for income range

uniplot(target0_df,col='AMT_INCOME_RANGE',title='Distribution of income range',hue='CODE_
```

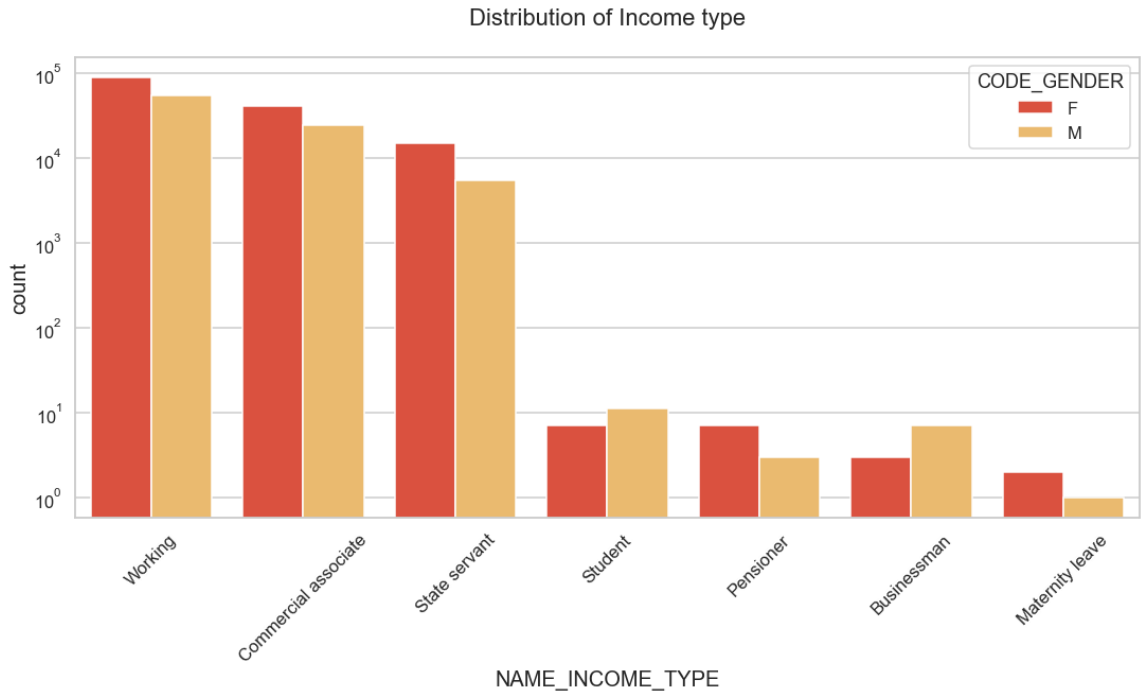


Points to be concluded from the above graph:

- Females counts are higher than male.
- Very less count for income range 400000 and above.
- Income range from 100000 to 200000 is having more number of credits.
- This graph show that females are more than male in having credits for that range.

In [30]:

```
# Plotting for Income type

uniplot(target0_df,col='NAME_INCOME_TYPE',title='Distribution of Income type',hue='CODE_
```

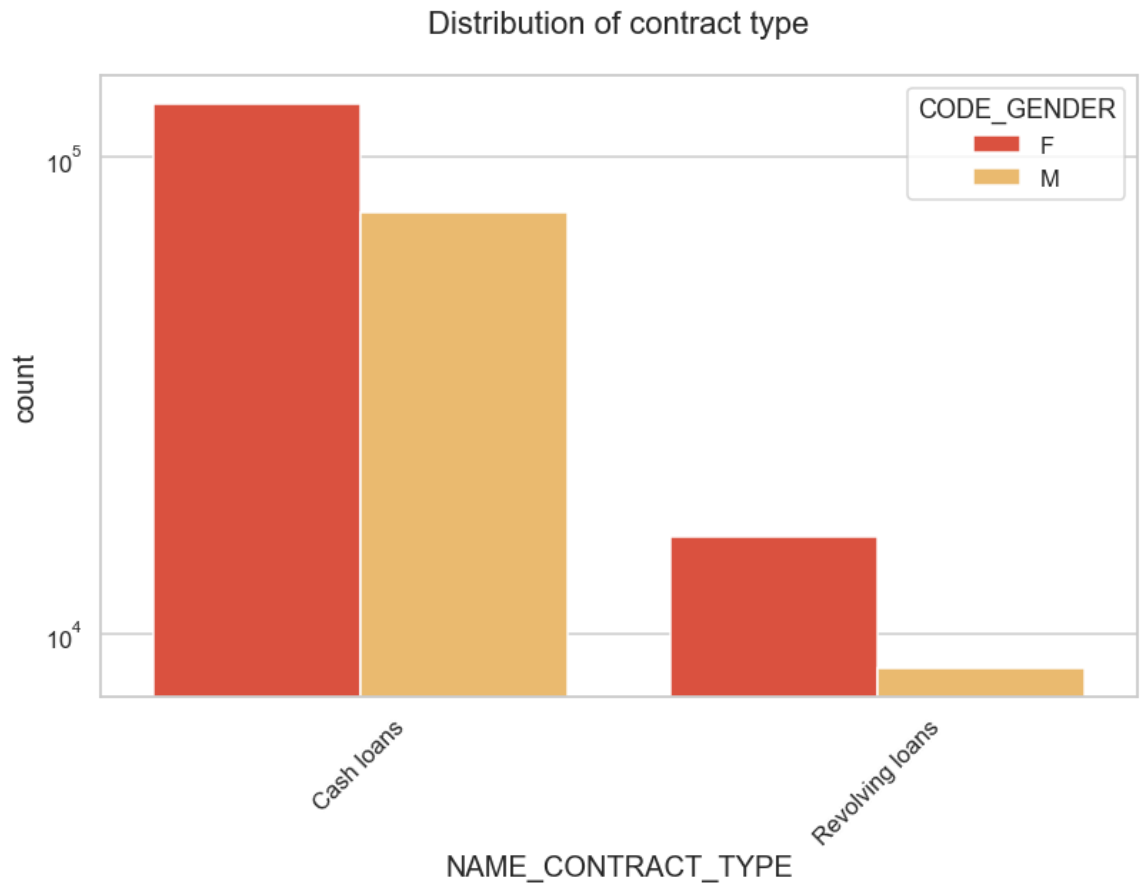


Points to be concluded from the above graph:

- For income type ‘working’, ’commercial associate’, and ‘State Servant’ the number of credits are higher than others.
- For this Females are having more number of credits than male.
- Less number of credits for income type ‘student’ ,’pensioner’, ‘Businessman’ and ‘Maternity leave’.

In [31]:

```
# Plotting for Contract type

uniplot(target0_df,col='NAME_CONTRACT_TYPE',title='Distribution of contract type',hue='C
```

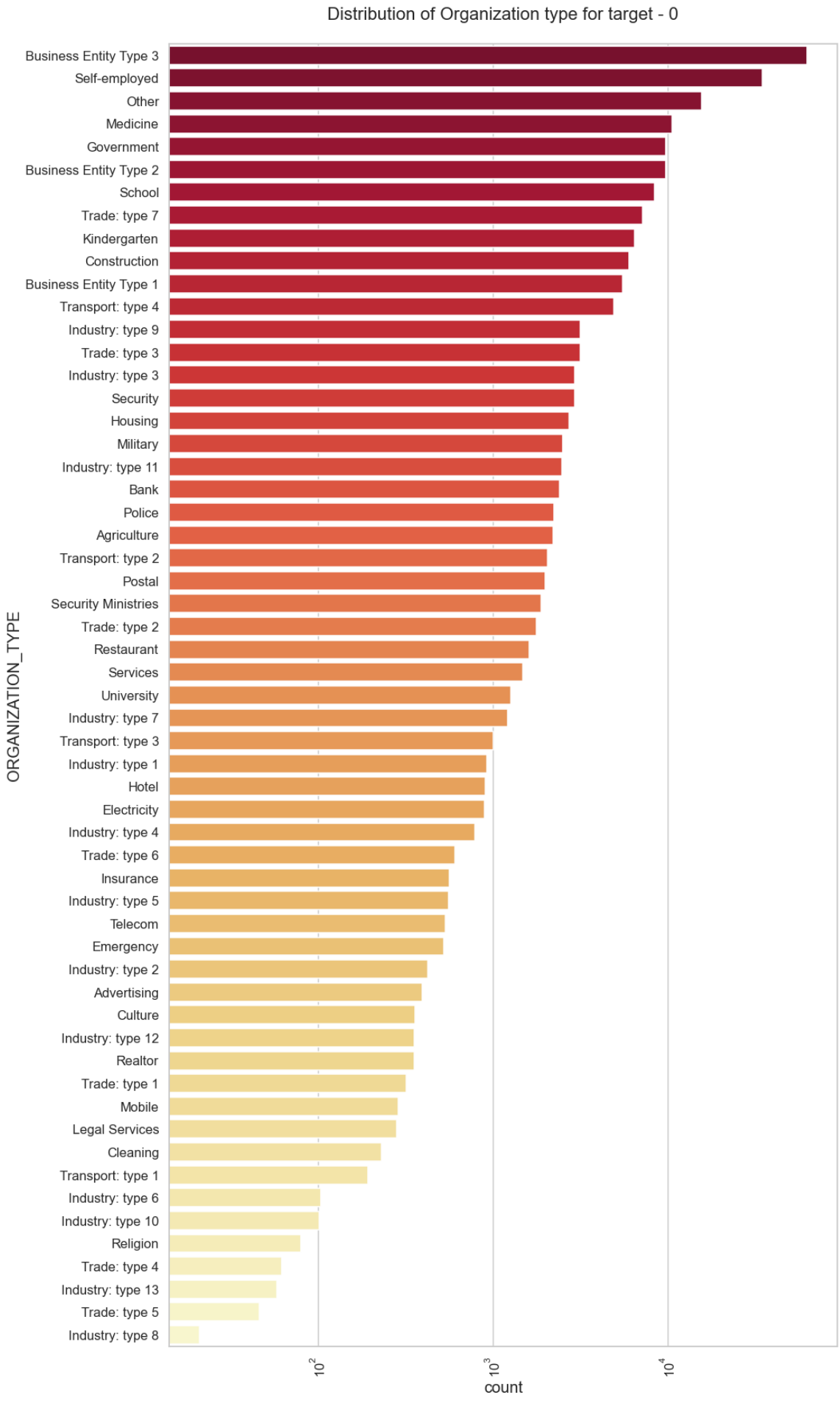


Points to be concluded from the above graph:

- For contract type ‘cash loans’ is having higher number of credits than ‘Revolving loans’ contract type.
- For this also Female is leading for applying credits.

In [32]:

```
# Plotting for Organization type in logarithmic scale

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.title("Distribution of Organization type for target - 0")

plt.xticks(rotation=90)
plt.xscale('log')

sns.countplot(data=target0_df,y='ORGANIZATION_TYPE',order=target0_df['ORGANIZATION_TYPE'

plt.show()
```



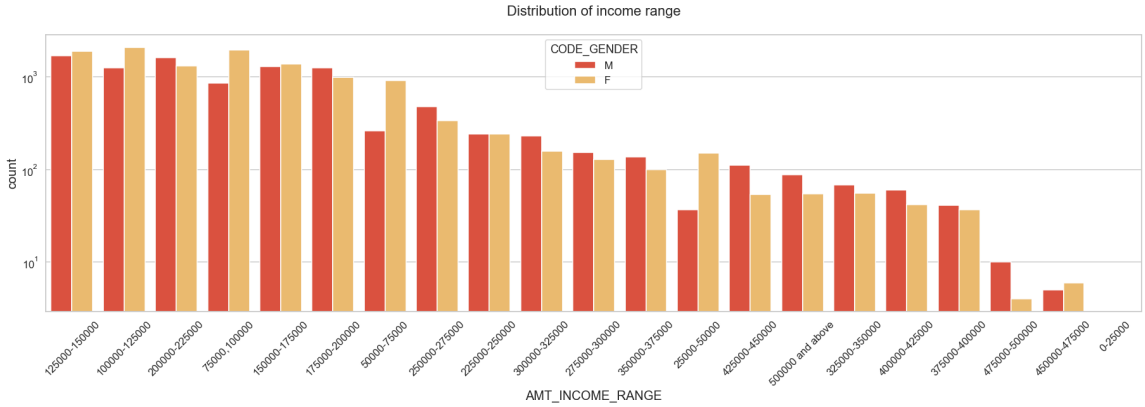
Points to be concluded from the above graph:

- Clients which have applied for credits are from most of the organization type ‘Business entity Type 3’ , ‘Self employed’, ‘Other’ , ‘Medicine’ and ‘Government’.

Categorical Univariate Analysis in logarithmic scale for target= 1 (client with payment difficulties)

In [33]:

```
# PLotting for income range

uniplot(target1_df,col='AMT_INCOME_RANGE',title='Distribution of income range',hue='CODE
```

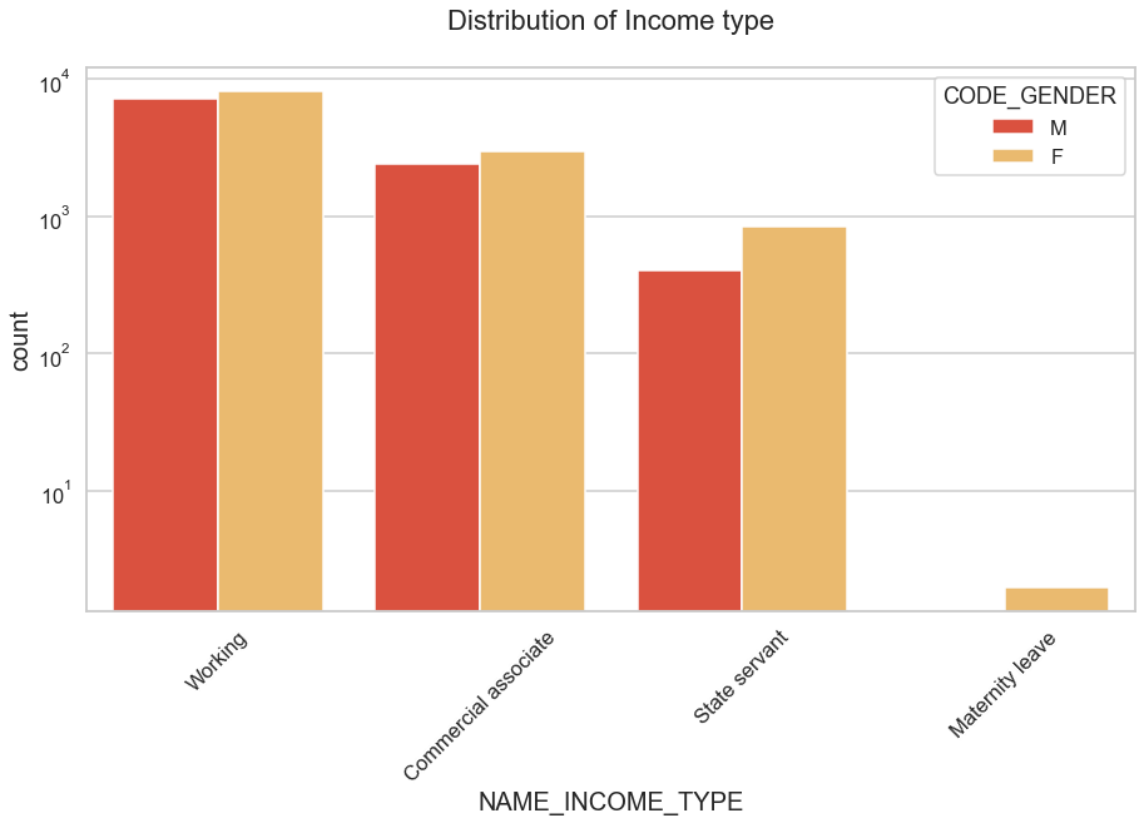


Points to be concluded from the above graph:

- Male counts are higher than female.
- Income range from 100000 to 200000 is having more number of credits.
- This graph show that males are more than female in having credits for that range.
- Very less count for income range 400000 and above.

In [34]:

```
# Plotting for Income type

uniplot(target1_df,col='NAME_INCOME_TYPE',title='Distribution of Income type',hue='CODE_
```

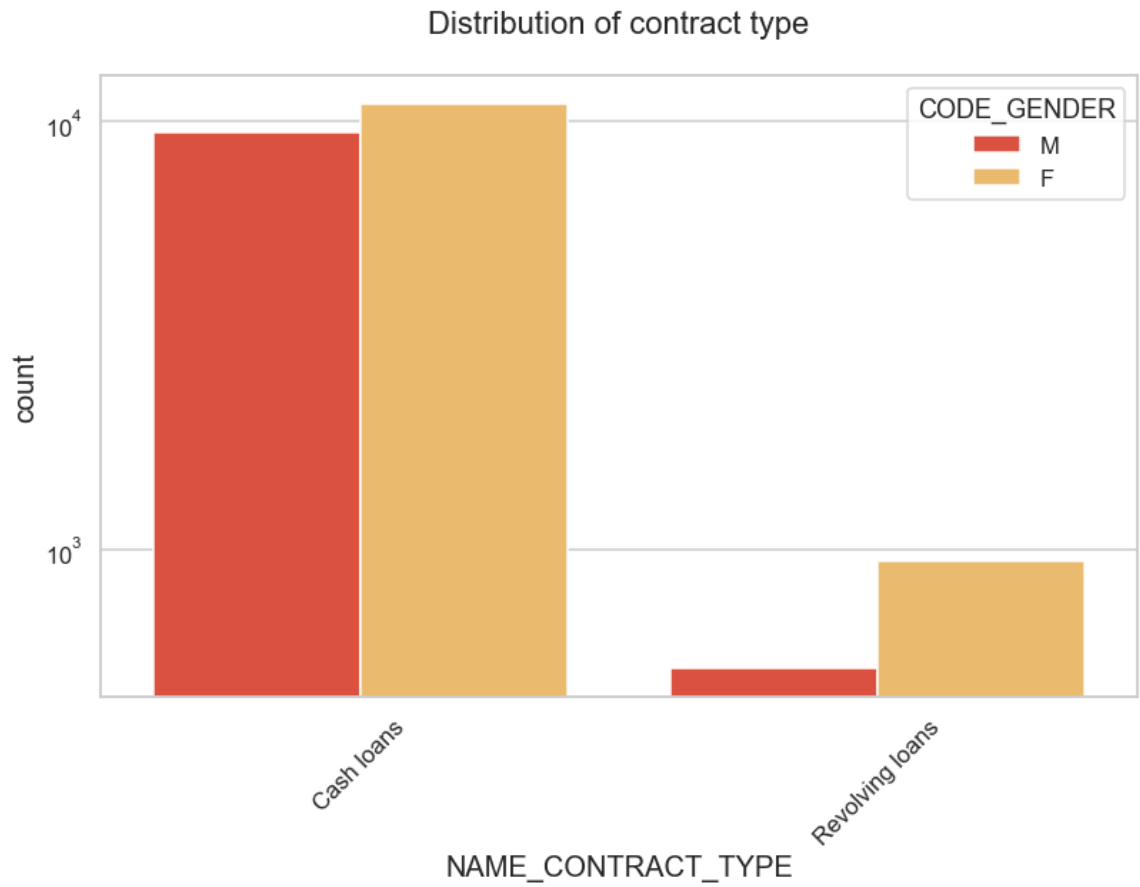


Points to be concluded from the above graph:

- For income type ‘working’, ’commercial associate’, and ‘State Servant’ the number of credits are higher than other i.e. ‘Maternity leave.
- For this Females are having more number of credits than male.
- Less number of credits for income type ‘Maternity leave’.
- For type 1: There is no income type for ‘student’ , ’pensioner’ and ‘Businessman’ which means they don’t do any late payments.

In [35]:

```
# Plotting for Contract type

uniplot(target1_df,col='NAME_CONTRACT_TYPE',title='Distribution of contract type',hue='C
```

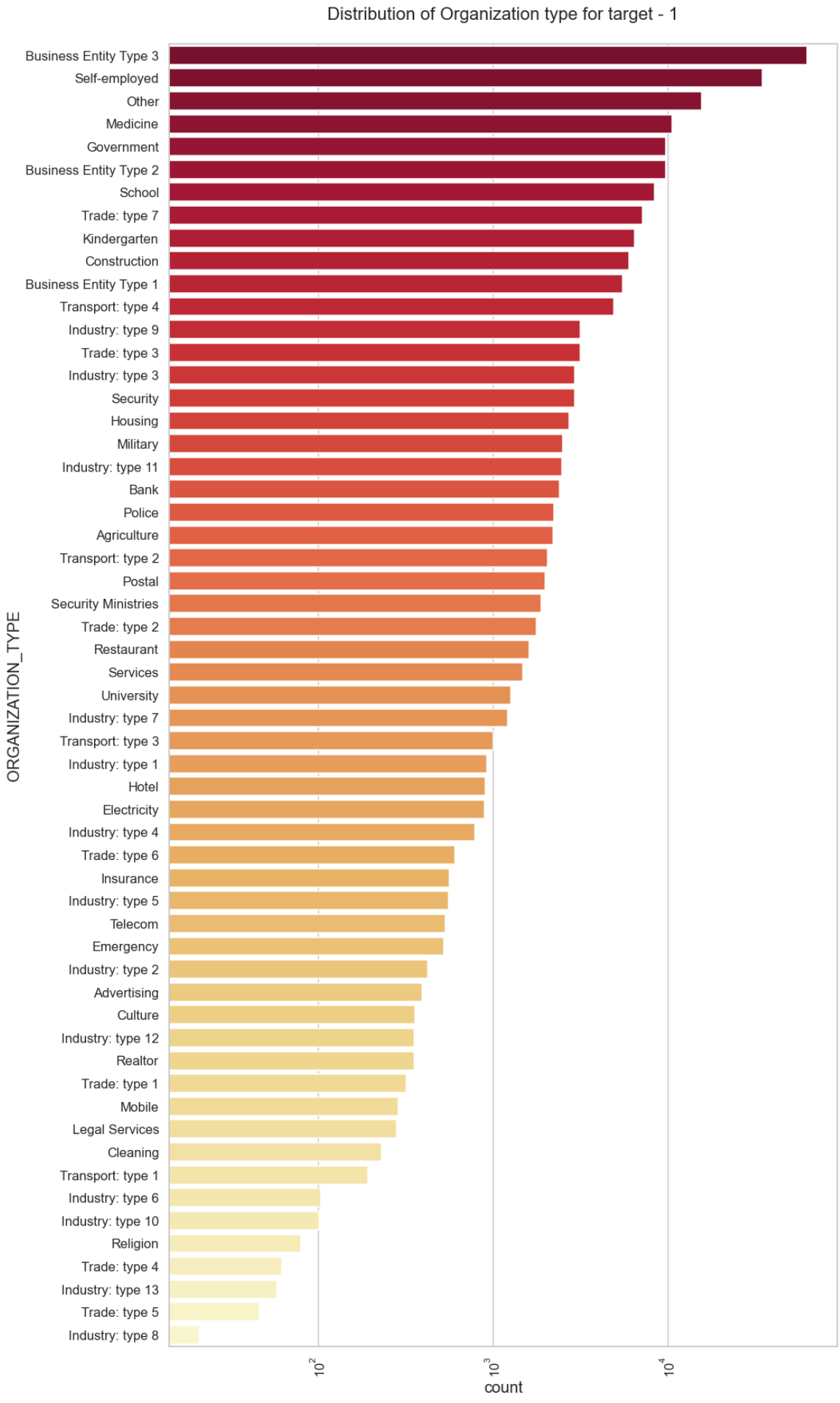


Points to be concluded from the above graph:

- For contract type ‘cash loans’ is having higher number of credits than ‘Revolving loans’ contract type.
- For this also Female is leading for applying credits.

In [36]:

```
# Plotting for Organization type

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.title("Distribution of Organization type for target - 1")

plt.xticks(rotation=90)
plt.xscale('log')

sns.countplot(data=target0_df,y='ORGANIZATION_TYPE',order=target0_df['ORGANIZATION_TYPE'

plt.show()
```



Points to be concluded from the above graph:

- Clients which have applied for credits are from most of the organization type ‘Business entity Type 3’ , ‘Self employed’ , ‘Other’ , ‘Medicine’ and ‘Government’.
- Less clients are from Industry type 8,type 6, type 10, religion and trade type 5, type 4.

Correlation

In [37]:

```
# Finding some correlation for numerical columns for both target 0 and 1

target0_corr=target0_df.iloc[0:,2:]
target1_corr=target1_df.iloc[0:,2:]

target0=target0_corr.corr(method='spearman')
target1=target1_corr.corr(method='spearman')
```

In [38]:

```
# Correlation for target 0

target0
```

Out[38]:

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AM
CNT_CHILDREN	1.000000	-0.021950	-0.023652	
AMT_INCOME_TOTAL	-0.021950	1.000000	0.403876	
AMT_CREDIT	-0.023652	0.403876	1.000000	
AMT_ANNUITY	-0.010795	0.472204	0.826689	
REGION_POPULATION_RELATIVE	-0.030579	0.110074	0.060706	
DAYS_BIRTH	0.266534	-0.054666	-0.169030	
DAYS_EMPLOYED	0.030948	-0.060868	-0.104251	
DAYS_REGISTRATION	0.155518	0.040559	-0.015318	
DAYS_ID_PUBLISH	-0.119164	-0.036702	-0.038197	
HOUR_APPR_PROCESS_START	-0.030162	0.073503	0.036923	
REG_REGION_NOT_LIVE_REGION	-0.022813	0.077634	0.015118	
REG_REGION_NOT_WORK_REGION	-0.015475	0.159962	0.041693	
LIVE_REGION_NOT_WORK_REGION	-0.005576	0.148281	0.045175	
REG_CITY_NOT_LIVE_CITY	0.002344	-0.001023	-0.040616	
REG_CITY_NOT_WORK_CITY	0.007487	-0.013856	-0.037000	
LIVE_CITY_NOT_WORK_CITY	0.013295	-0.004758	-0.011194	

In [39]:

```
# Correlation for target 1

target1
```

Out[39]:

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AM
CNT_CHILDREN	1.000000	-0.039123	0.000427	
AMT_INCOME_TOTAL	-0.039123	1.000000	0.364559	
AMT_CREDIT	0.000427	0.364559	1.000000	
AMT_ANNUITY	0.015133	0.428947	0.812093	
REGION_POPULATION_RELATIVE	-0.029682	0.058005	0.043545	
DAYS_BIRTH	0.175025	-0.103026	-0.200718	
DAYS_EMPLOYED	0.006823	-0.053798	-0.107605	
DAYS_REGISTRATION	0.110854	0.011378	-0.021973	
DAYS_ID_PUBLISH	-0.091042	-0.051113	-0.065143	
HOUR_APPR_PROCESS_START	-0.040338	0.078779	0.024616	
REG_REGION_NOT_LIVE_REGION	-0.035213	0.075615	0.015043	
REG_REGION_NOT_WORK_REGION	-0.040853	0.156374	0.032536	
LIVE_REGION_NOT_WORK_REGION	-0.027993	0.145982	0.034861	
REG_CITY_NOT_LIVE_CITY	-0.016072	-0.003813	-0.030974	
REG_CITY_NOT_WORK_CITY	-0.005444	-0.006241	-0.032882	
LIVE_CITY_NOT_WORK_CITY	0.009557	0.004230	-0.012465	

In [40]:

```
# Now, plotting the above correlation with heat map as it is the best choice to visulaiz

# figure size

def targets_corr(data,title):
    plt.figure(figsize=(15, 10))
    plt.rcParams['axes.titlesize'] = 25
    plt.rcParams['axes.titlepad'] = 70

# heatmap with a color map of choice


    sns.heatmap(data, cmap="RdYlGn",annot=False)

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

In [41]:

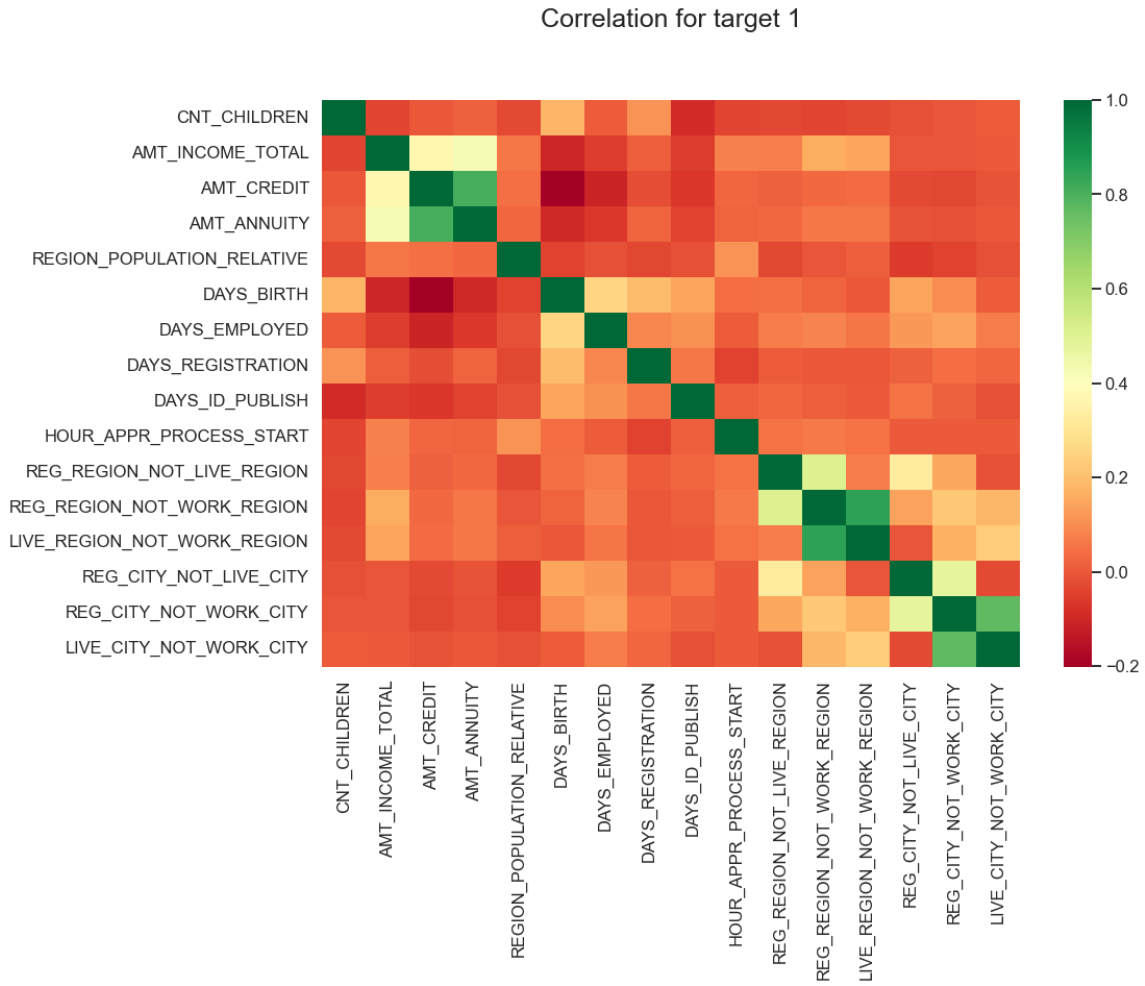
```
# For Target 0

targets_corr(data=target0,title='Correlation for target 0')
```

As we can see from above correlation heatmap, There are number of observation we can point out:

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

In [42]:

```
# For Target 1

targets_corr(data=target1,title='Correlation for target 1')
```



This heat map for Target 1 is also having quite a same observation just like Target 0. But for few points are different. They are listed below:

- The client's permanent address does not match contact address are having less children and vice-versa
- The client's permanent address does not match work address are having less children and vice-versa

Univariate Analysis for Variables

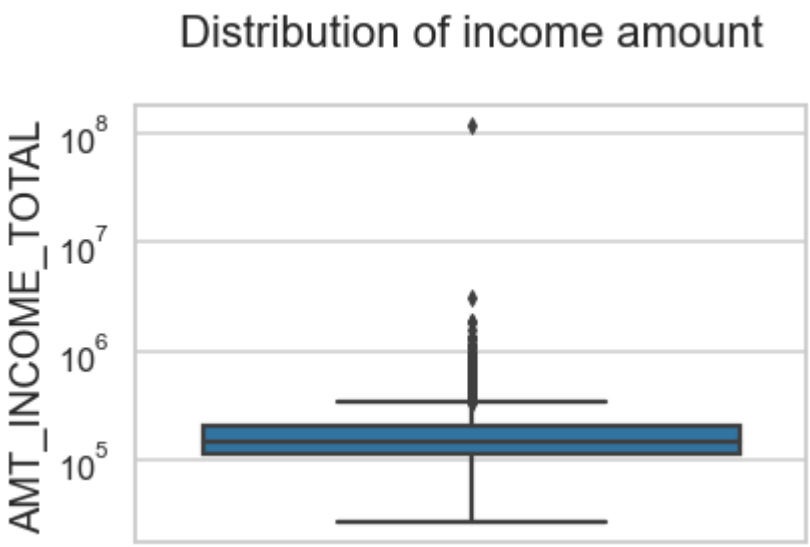
In [43]:

```
def univariate_numerical(data,col,title):
    sns.set_style('whitegrid')
    sns.set_context('talk')
    plt.rcParams["axes.labelsize"] = 20
    plt.rcParams['axes.titlesize'] = 22
    plt.rcParams['axes.titlepad'] = 30

    plt.title(title)
    plt.yscale('log')
    sns.boxplot(data =target1_df, y= col)
    plt.show()
```

Finding Outliers : Target 0

In [44]:

```
# Distribution of income amount

univariate_numerical(data=target0_df,col='AMT_INCOME_TOTAL',title='Distribution of incom
```

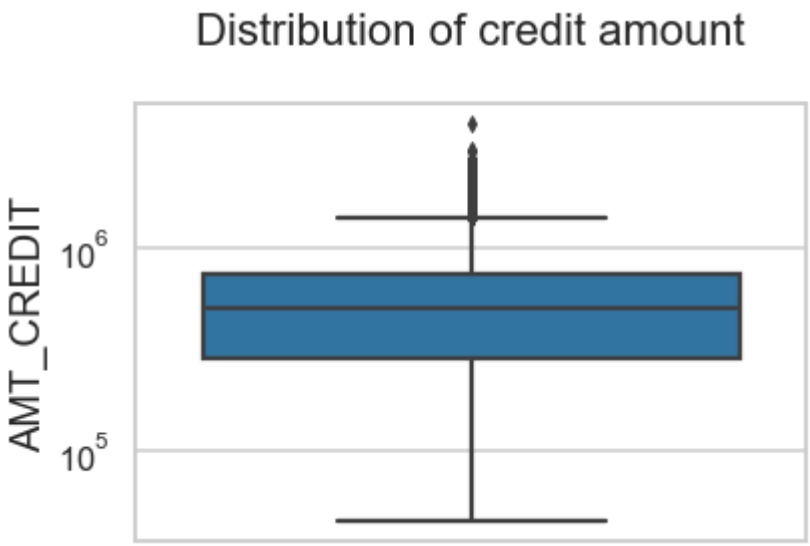


Few points can be concluded from the graph above:

- Some outliers are noticed in income amount.
- The third quartiles is very slim for income amount.

In [45]:

```
# Disrtibution of credit amount

univariate_numerical(data=target0_df,col='AMT_CREDIT',title='Distribution of credit amou
```

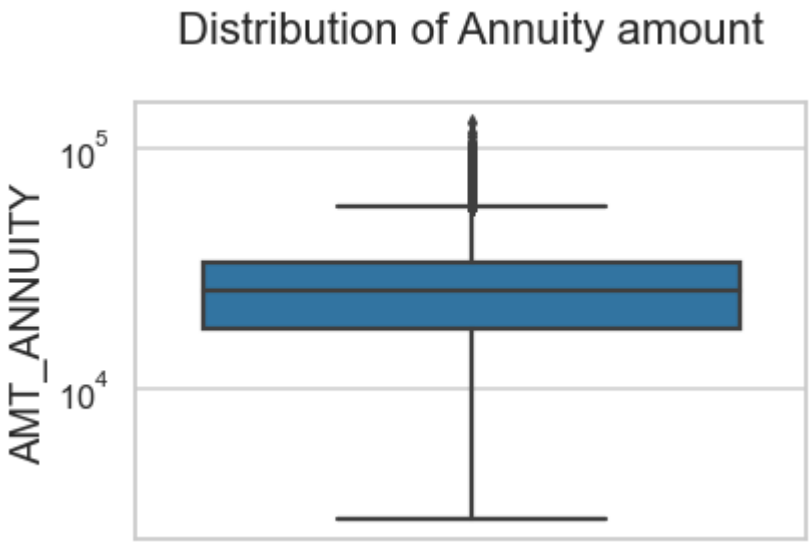


Few points can be concluded from the graph above:

- Some outliers are noticed in credit amount.
- The first quartile is bigger than third quartile for credit amount which means most of the credits of clients are present in the first quartile.

In [46]:

```
# Distribution of anuuity amount

univariate_numerical(data=target0_df,col='AMT_ANNUITY',title='Distribution of Annuity am
```



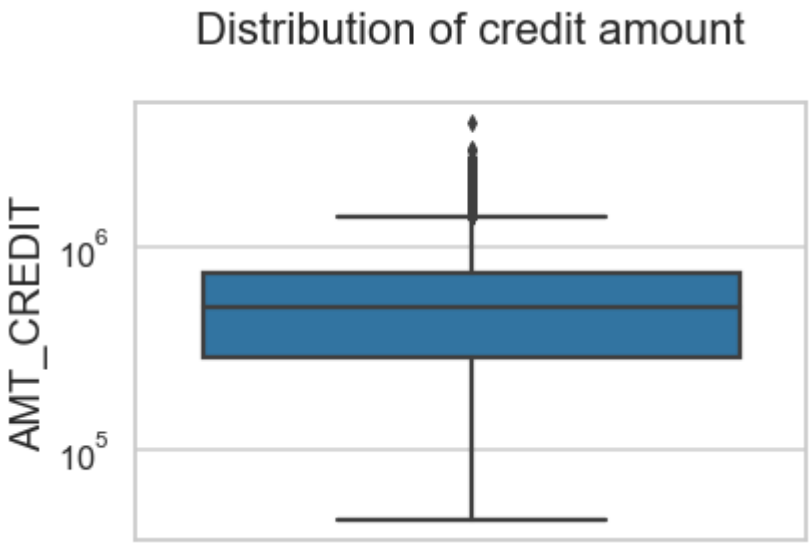
Few points can be concluded from the graph above:

- Some outliers are noticed in annuity amount.

Finding Outliers : Target 1

In [47]:

```
# Distribution of credit amount

univariate_numerical(data=target1_df,col='AMT_CREDIT',title='Distribution of credit amou
```

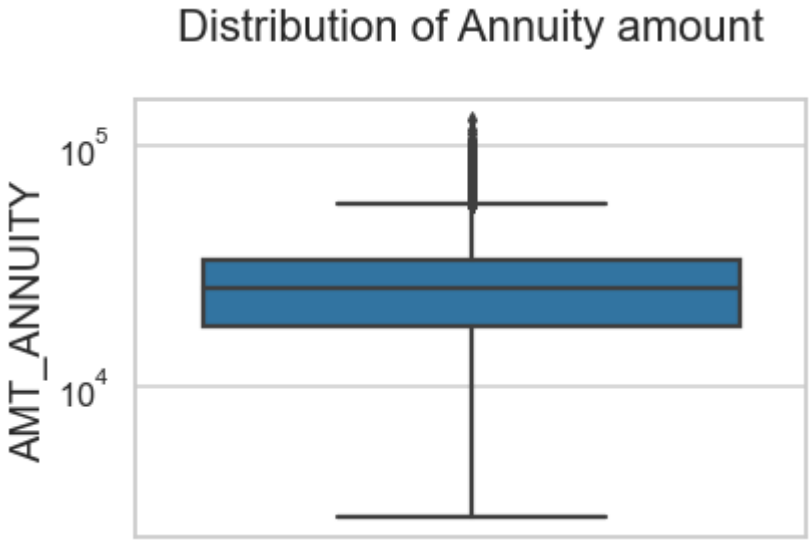


Few points can be concluded from the graph above:

- Some outliers are noticed in credit amount.
- The first quartile is bigger than third quartile for credit amount which means most of the credits of clients are present in the first quartile.

In [48]:

```
# Distribution of Annuity amount

univariate_numerical(data=target1_df,col='AMT_ANNUITY',title='Distribution of Annuity am
```



Few points can be concluded from the graph above:

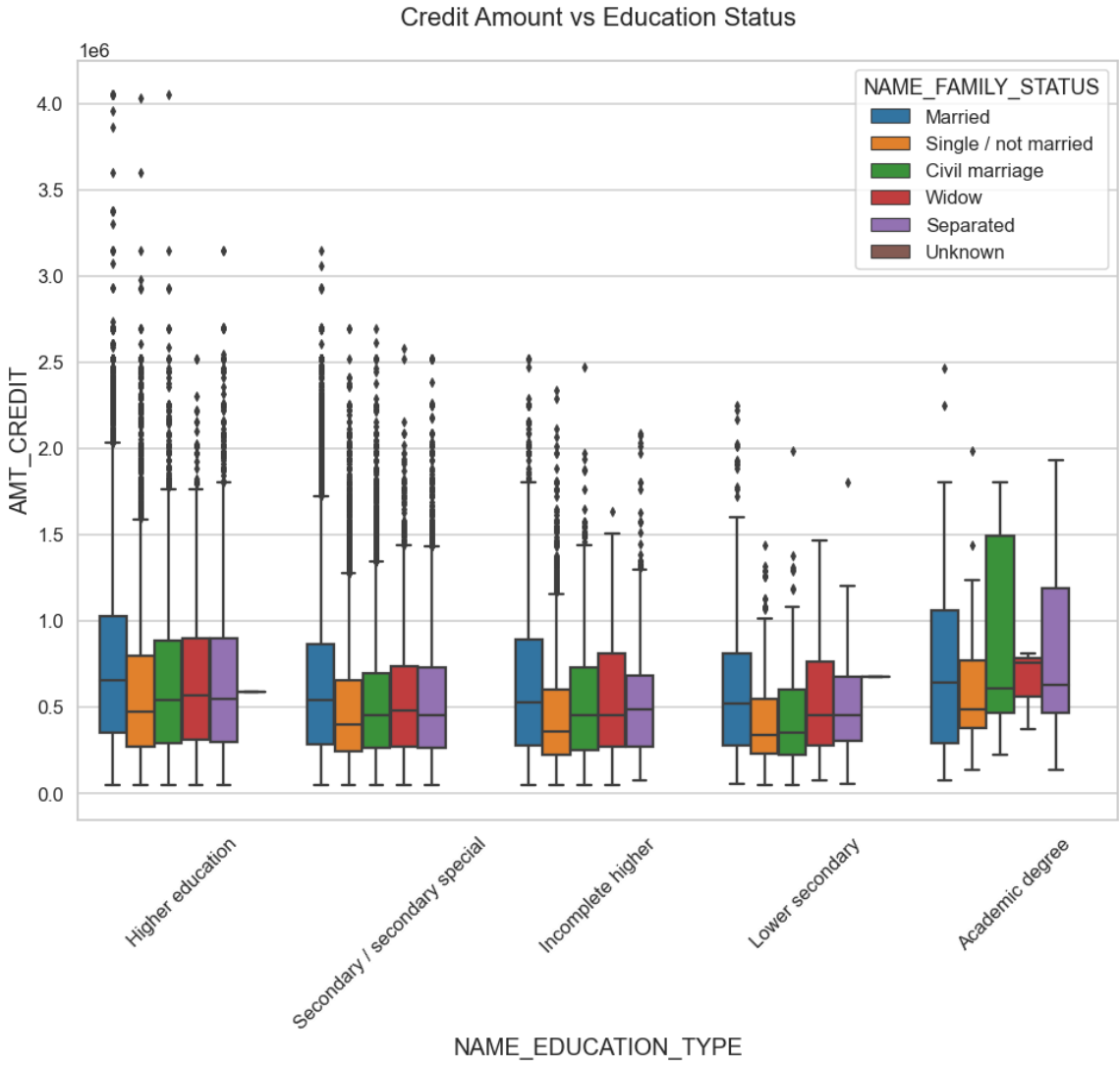
- Some outliers are noticed in annuity amount.
- The first quartile is bigger than third quartile for annuity amount which means most of the annuity clients are from first quartile

Bivariate Analysis for Numerical Variables

For Target 0

In [49]:

```
plt.figure(figsize=(16,12))
plt.xticks(rotation=45)
sns.boxplot(data =target0_df, x='NAME_EDUCATION_TYPE',y='AMT_CREDIT', hue ='NAME_FAMILY_
plt.title('Credit Amount vs Education Status')
plt.show()
```

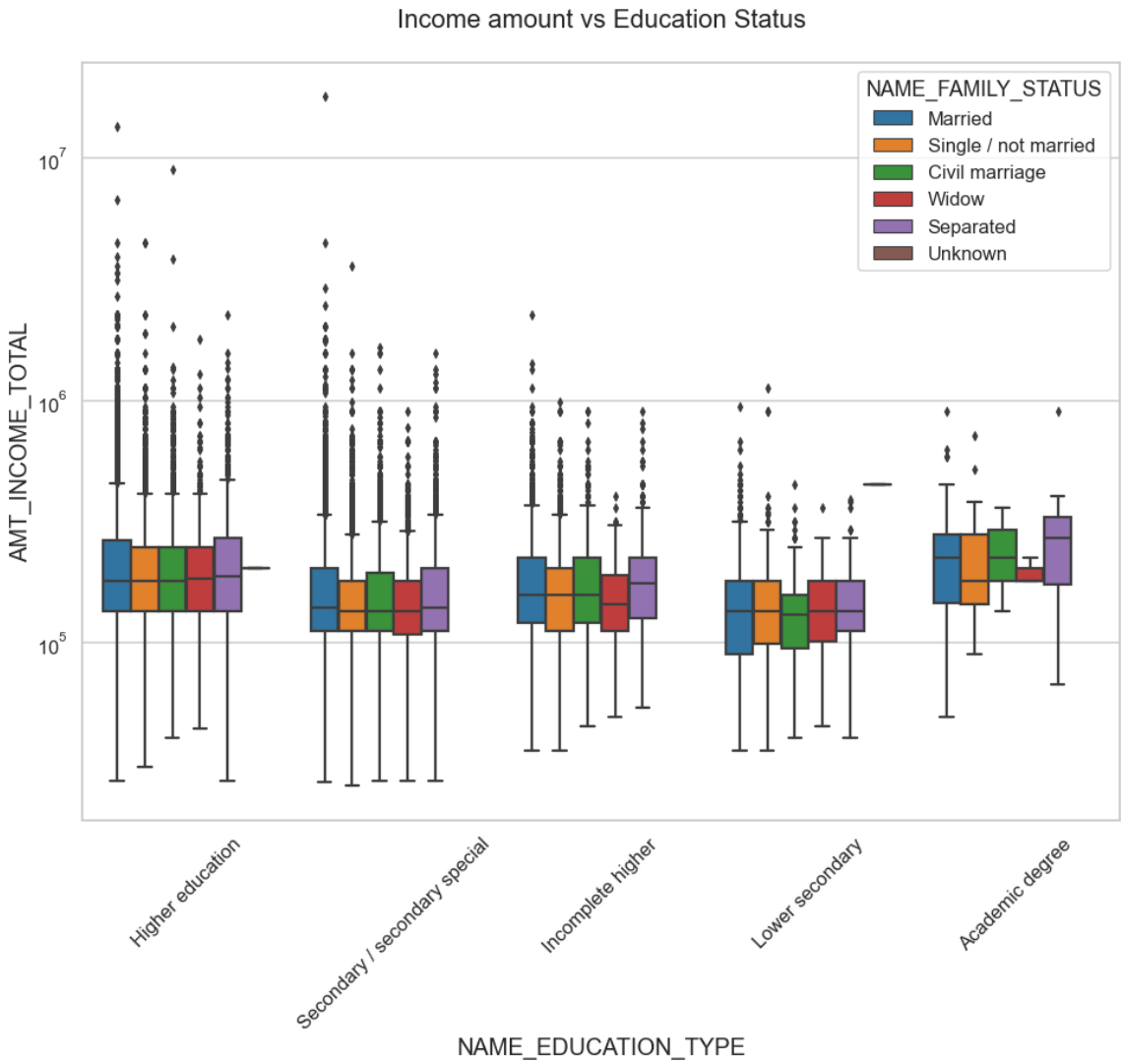


Few points can be concluded from the graph above:

- Family status of 'civil marriage', 'marriage' and 'separated' of Academic degree education are having higher number of credits than others.
- Higher education of family status of 'marriage', 'single' and 'civil marriage' are having more outliers.
- Civil marriage for Academic degree is having most of the credits in the third quartile.

In [51]:

```
# Box plotting for Income amount in logarithmic scale

plt.figure(figsize=(16,12))
plt.xticks(rotation=45)
plt.yscale('log')
sns.boxplot(data =target0_df, x='NAME_EDUCATION_TYPE',y='AMT_INCOME_TOTAL', hue ='NAME_F
plt.title('Income amount vs Education Status')
plt.show()
```



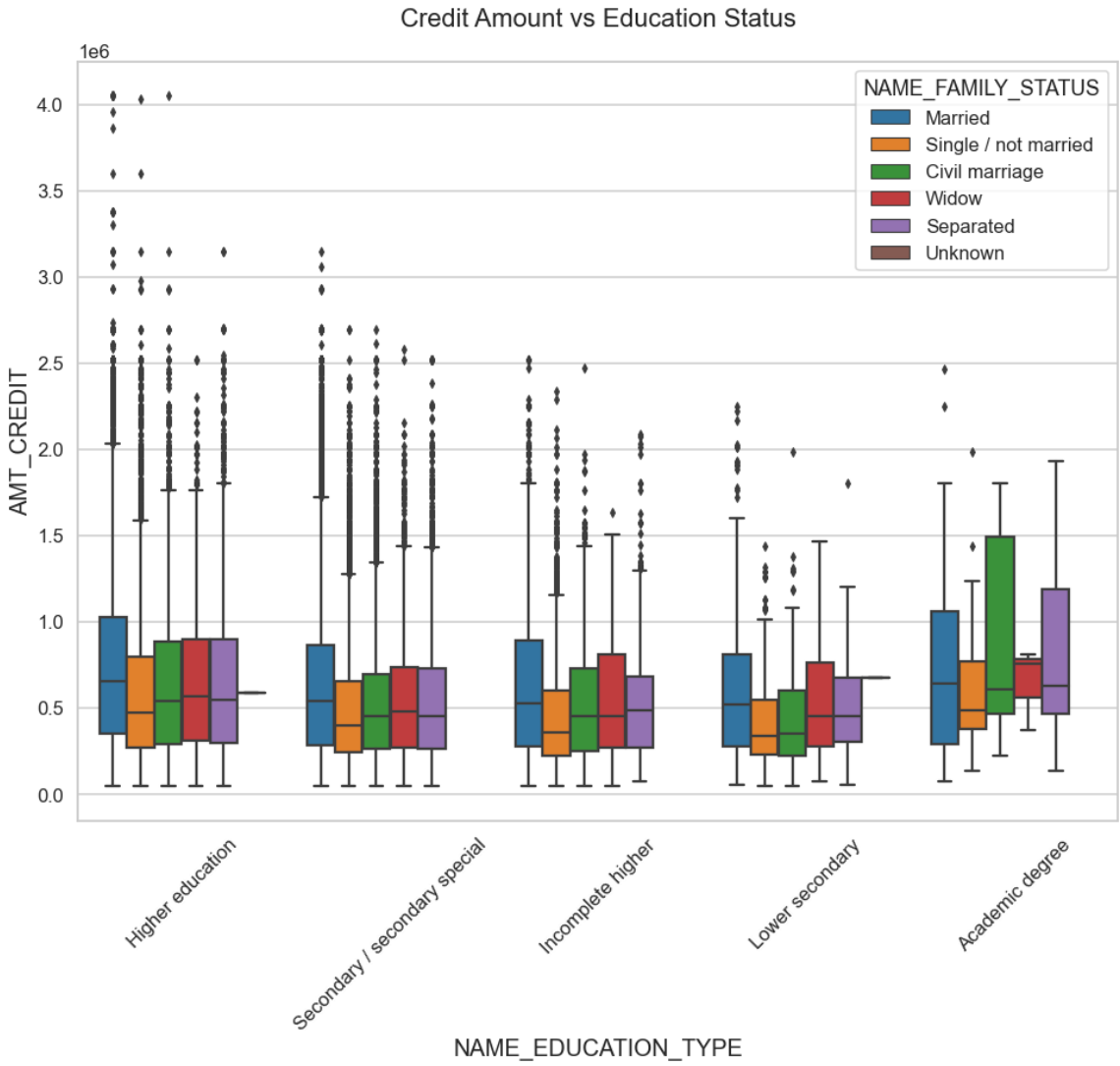
Few points can be concluded from the graph above:

- Education type 'Higher education' the income amount is mostly equal with family status. It does contain many outliers.
- Less outlier are having for Academic degree but there income amount is little higher that Higher education.
- Lower secondary of civil marriage family status are have less income amount than others.

For Target 1

In [52]:

```
# Box plotting for credit amount

plt.figure(figsize=(16,12))
plt.xticks(rotation=45)
sns.boxplot(data =target0_df, x='NAME_EDUCATION_TYPE',y='AMT_CREDIT', hue ='NAME_FAMILY_
plt.title('Credit Amount vs Education Status')
plt.show()
```

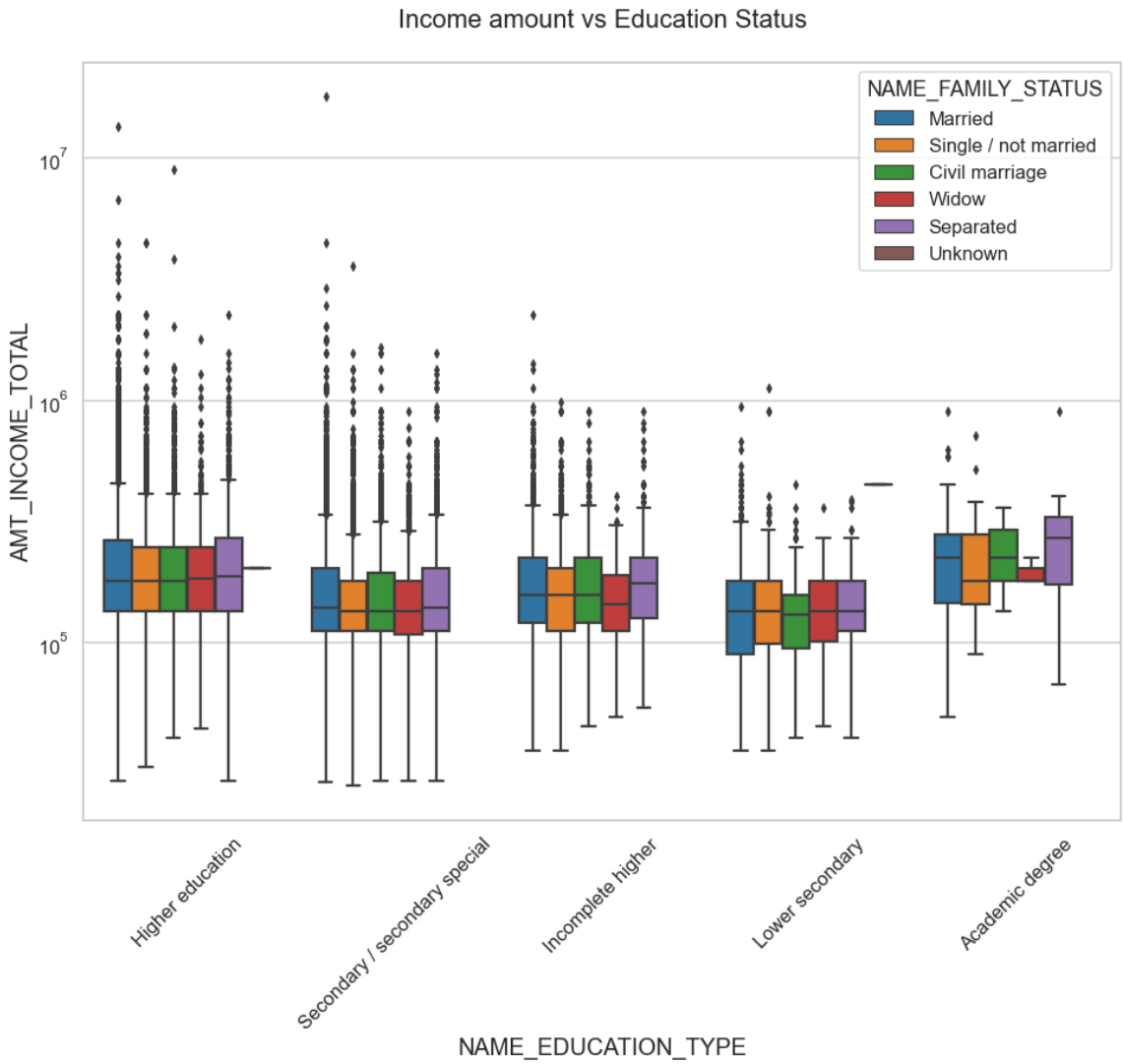


Few points can be concluded from the graph above:

- Quite similar with Target 0 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 Academic degree is having most of the credits in the third quartile.

In [53]:

```
# Box plotting for Income amount in logarithmic scale

plt.figure(figsize=(16,12))
plt.xticks(rotation=45)
plt.yscale('log')
sns.boxplot(data =target0_df, x='NAME_EDUCATION_TYPE',y='AMT_INCOME_TOTAL', hue ='NAME_F
plt.title('Income amount vs Education Status')
plt.show()
```



Few points can be concluded from the graph above:

- Have some similarity with Target0, From above boxplot for Education type 'Higher education' the income amount is mostly equal with family status.
- Less outlier are having for Academic degree but there income amount is little higher that Higher education. Lower secondary are have less income amount than others.

Reading Previous_Application

In [55]:

```
# Reading the dataset of previous application

df1=pd.read_csv("previous_application.csv")
df1.head()
```

Out[55]:

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

5 rows × 37 columns

In [56]:

```
# Cleaning the missing data

# listing the null values columns having more than 40%

emptycol1=df1.isnull().sum()
emptycol1=emptycol1[emptycol1.values>(0.4*len(emptycol1))]
len(emptycol1)
```

Out[56]:

```
15
```

In [57]:

```
# Removing those 15 columns

emptycol1 = list(emptycol1[emptycol1.values>=0.4].index)
df1.drop(labels=emptycol1,axis=1,inplace=True)

df1.shape
```

Out[57]:

```
(1670214, 22)
```

In [59]:

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

df1=df1.drop(df1[df1['NAME_CASH_LOAN_PURPOSE']=='XNA'].index)
df1=df1.drop(df1[df1['NAME_CASH_LOAN_PURPOSE']=='XNA'].index)
df1=df1.drop(df1[df1['NAME_CASH_LOAN_PURPOSE']=='XAP'].index)

df1.shape
```

Out[59]:

```
(69635, 22)
```

In [60]:

```
# Now merging the Application dataset with previous appliaction dataset

new_df=pd.merge(left=df,right=df1,how='inner',on='SK_ID_CURR',suffixes='_x')
```

In [62]:

```
new_df.head()
```

Out[62]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_	CODE_GENDER	FLAG_OWN_CAR	FLA
0	100034	0	Revolving loans	M	N	
1	100035	0	Cash loans	F	N	
2	100039	0	Cash loans	M	Y	
3	100046	0	Revolving loans	M	Y	
4	100046	0	Revolving loans	M	Y	

5 rows × 51 columns

In [63]:

```
# Renaming the column names after merging

new_df1 = new_df.rename({'NAME_CONTRACT_TYPE_' : 'NAME_CONTRACT_TYPE','AMT_CREDIT_':'AMT
                         'WEEKDAY_APPR_PROCESS_START_' : 'WEEKDAY_APPR_PROCESS_START',
                         'HOUR_APPR_PROCESS_START_':'HOUR_APPR_PROCESS_START','NAME_CONT
                         'AMT_CREDITx':'AMT_CREDIT_PREV','AMT_ANNUITYx':'AMT_ANNUITY_PRE
                         'WEEKDAY_APPR_PROCESS_STARTx':'WEEKDAY_APPR_PROCESS_START_PREV'
                         'HOUR_APPR_PROCESS_STARTx':'HOUR_APPR_PROCESS_START_PREV'}, axi
```

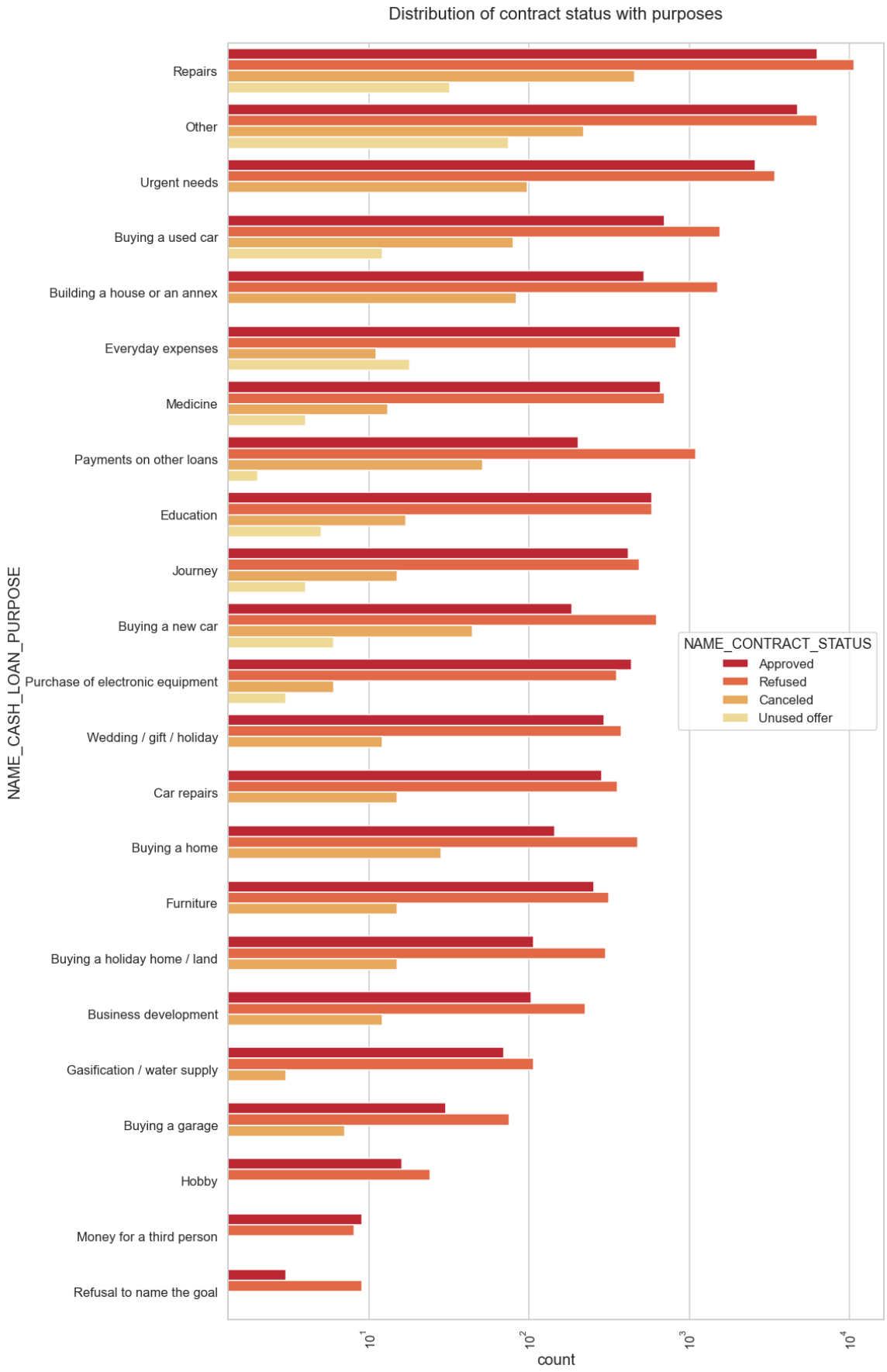
In [64]:

```
# Removing unwanted columns for analysis

new_df1.drop(['SK_ID_CURR','WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START','REG_
              'REG_REGION_NOT_WORK_REGION','LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_
              'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY','WEEKDAY_APPR_PROCESS_
              'HOUR_APPR_PROCESS_START_PREV', 'FLAG_LAST_APPL_PER_CONTRACT','NFLAG_LAST_
```

Performing Univariate Analysis

In [66]:

```
# Distribution of contract status in logarithmic scale

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 contract status with purposes')
ax = sns.countplot(data = new_df1, y= 'NAME_CASH_LOAN_PURPOSE',
                   order=new_df1['NAME_CASH_LOAN_PURPOSE'].value_counts().index,hue = 'N
```

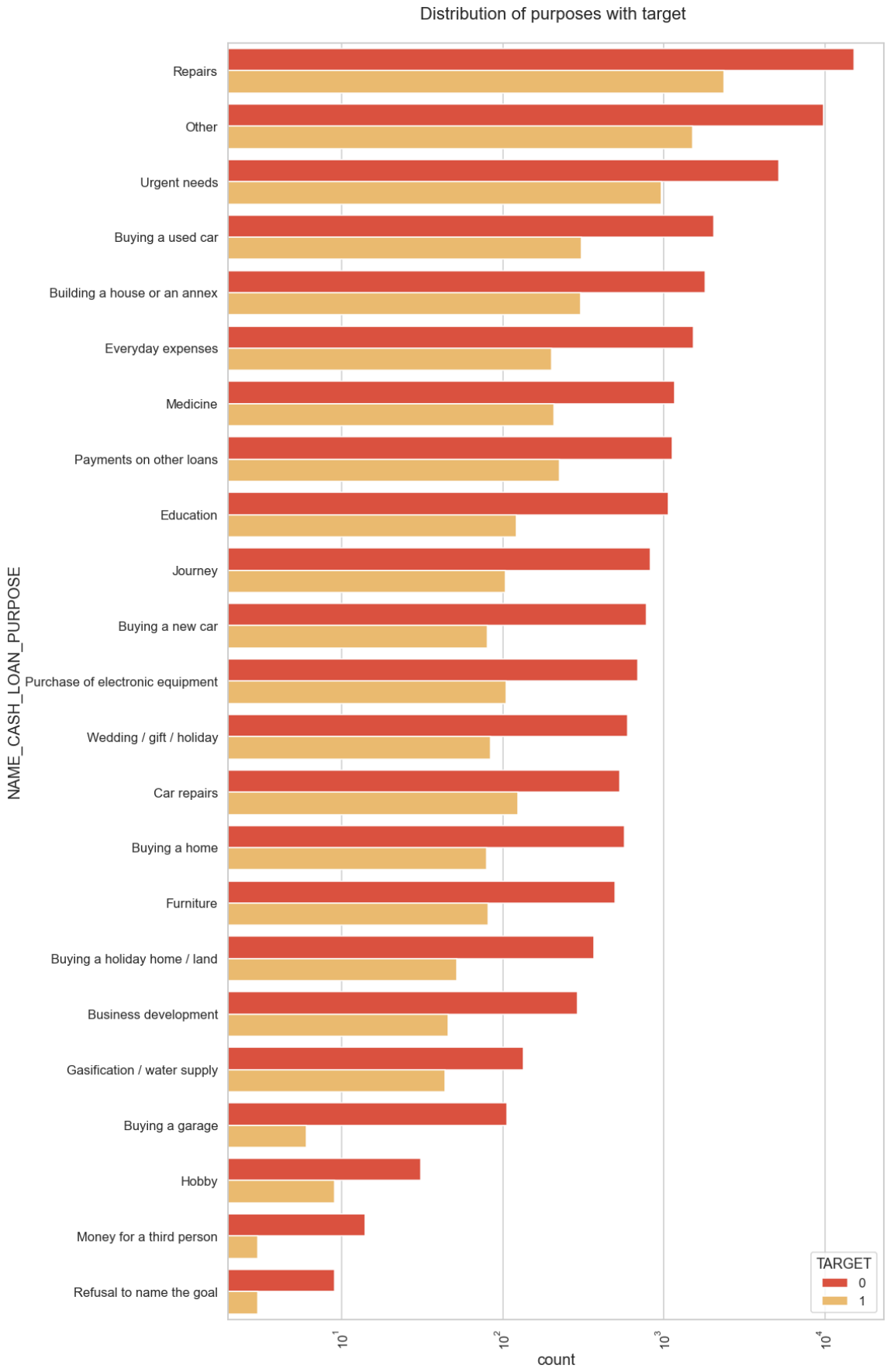


Points to be concluded from above plot:

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

In [68]:

```
# Distribution of contract 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 = new_df1, y= 'NAME_CASH_LOAN_PURPOSE',
                   order=new_df1['NAME_CASH_LOAN_PURPOSE'].value_counts().index,hue = 'T
```



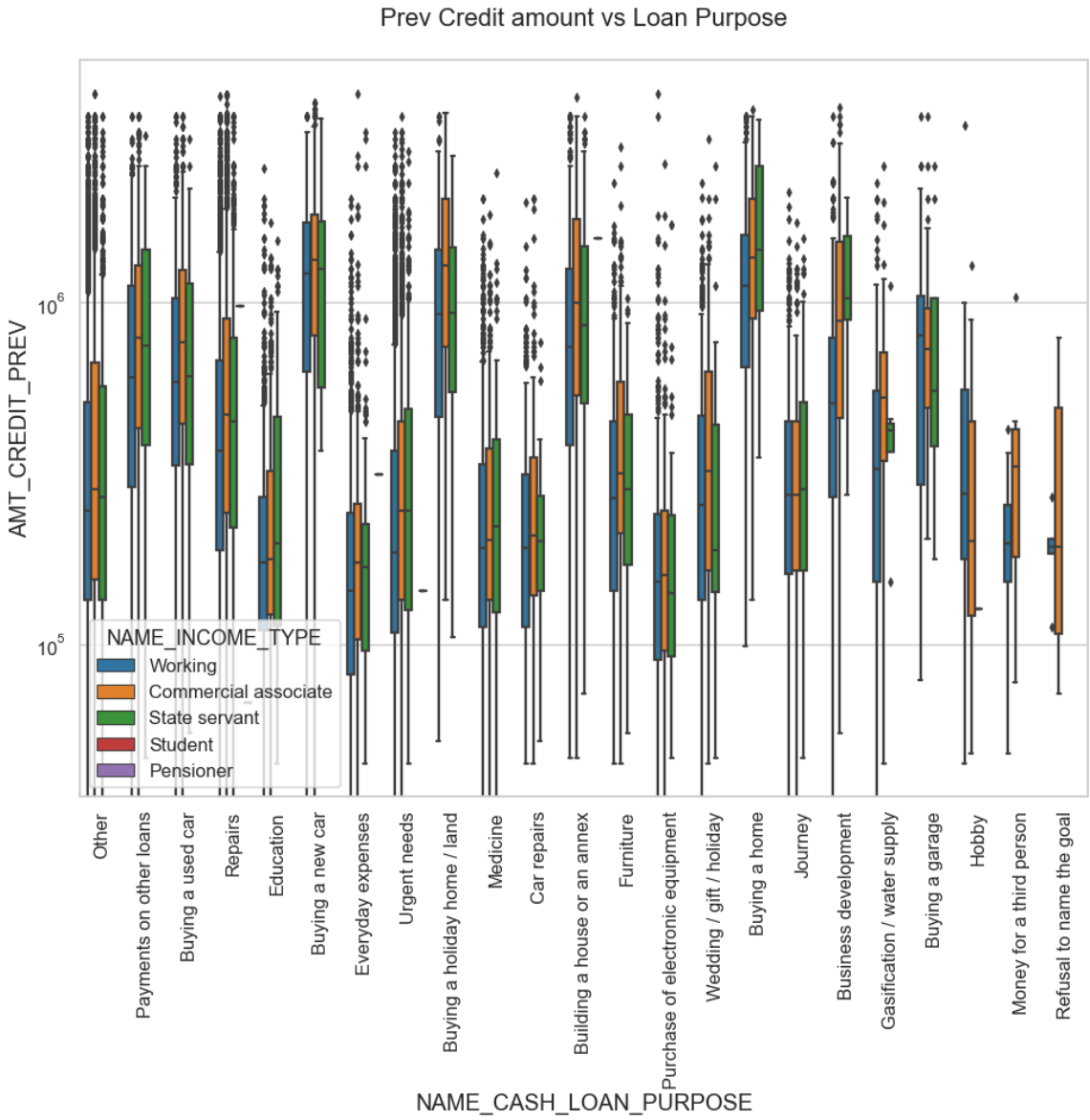
Few points we can conclude from above plot:

- Loan purposes with 'Repairs' are facing more difficulites in payment on time.
- There are few places where loan payment is significant higher than facing difficulties. They are 'Buying a garage', 'Business developemt', 'Buying land','Buying a new car' and 'Education'

Performing Bivariate Analysis

In [69]:

```
# Box plotting for Credit amount in logarithmic scale

plt.figure(figsize=(16,12))
plt.xticks(rotation=90)
plt.yscale('log')
sns.boxplot(data =new_df1, x='NAME_CASH_LOAN_PURPOSE',hue='NAME_INCOME_TYPE',y='AMT_CRED
plt.title('Prev Credit amount vs Loan Purpose')
plt.show()
```

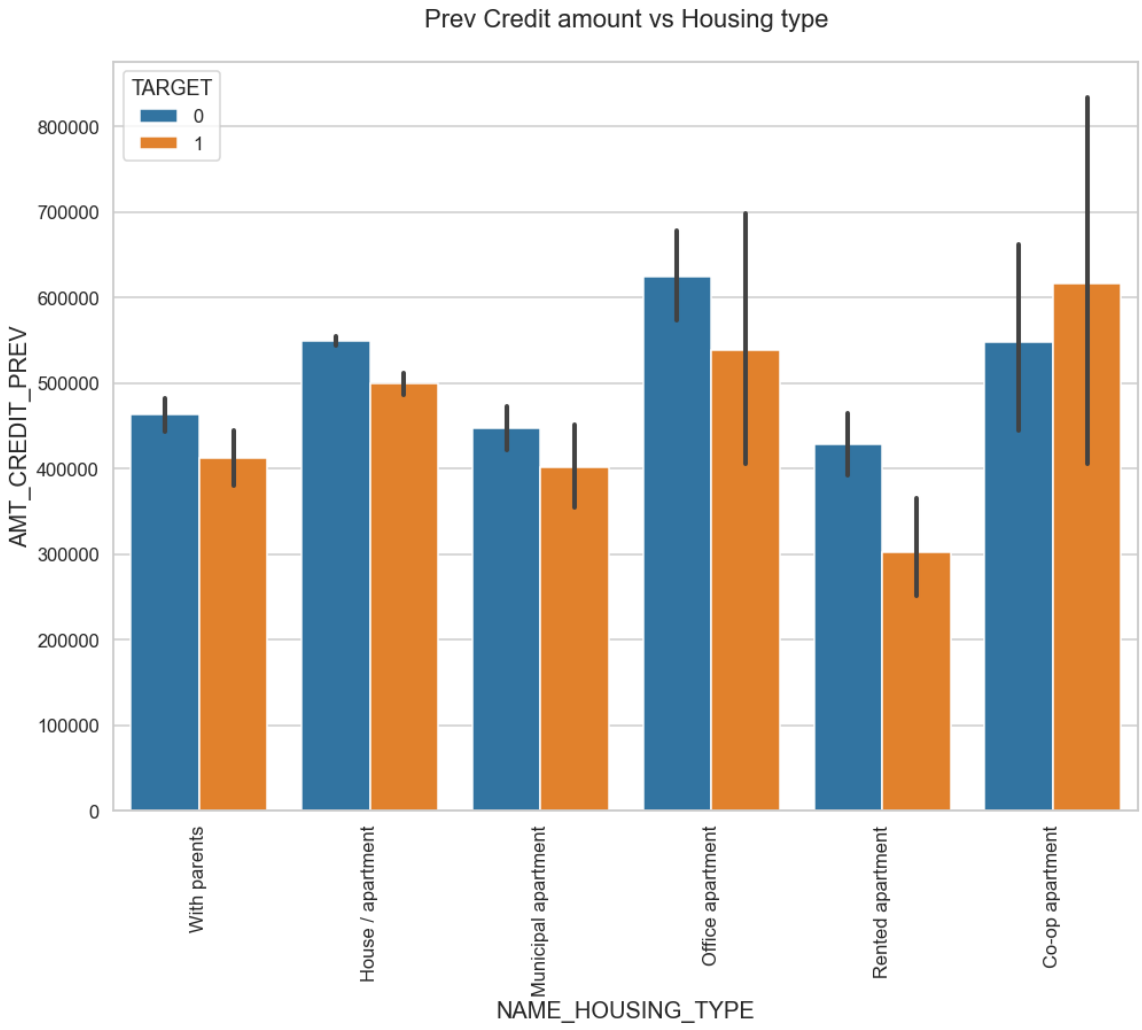


From the above plot we can conclude some points:

- The credit amount of Loan purposes like 'Buying a home','Buying a land','Buying a new car' and'Building a house' is higher.
- Income type of state servants have a significant amount of credit applied
- Money for third person or a Hobby is having less credits applied for.

In [70]:

```
# Box plotting for Credit amount prev vs Housing type in logarithmic scale

plt.figure(figsize=(16,12))
plt.xticks(rotation=90)
sns.barplot(data =new_df1, y='AMT_CREDIT_PREV',hue='TARGET',x='NAME_HOUSING_TYPE')
plt.title('Prev Credit amount vs Housing type')
plt.show()
```



From the above plot we can conclude:

Here for Housing type, office appartment is having higher credit of target 0 and co-op apartment is having higher credit of target 1. So, we can conclude that bank should avoid giving loans to the housing type of co-op apartment as they are having difficulties in payment. Bank can focus mostly on housing type with parents or House\appartment or miuncipal appartment for successful payments.

CONCLUSION

- Banks should focus less on income type ‘Working’ as they are having most number of unsuccessful payments.
- Banks should focus more on contract type ‘Student’ ,’pensioner’ and ‘Businessman’ with housing ‘type other than ‘Co-op apartment’ for successful payments.
- Also with loan purpose ‘Repair’ is having higher number of unsuccessful payments on time.

- Get as much as clients from housing type ‘With parents’ as they are having least number of unsuccessful payments.

In [ ]:

Thanks

for taking the time to read this case study. I hope it provided valuable insights and information