BANK LOAN CASE STUDY

Name : Aniket Ashok Ubale
Class : B. Tech
Year : 3rd
Branch : Artificial intelligence and data science
Project : Bank Loan Case Study
Date : 25-05-2023

Bank loan case study

-Project Description
-Approach
-Tech Stack Used
-Insights
-Results

Project description

This case study aims to give you an idea of applying EDA in a real business scenario.
Through this case study, we will gain insights into the practical implementation of EDA techniques. Furthermore, develop a foundational understanding of risk analytics in the banking and financial services industry (BFSI).
During the case study, it'll be shown on how data can be utilized to minimize the risk of losing money while lending to customers. Thus, appreciating the significance of data analysis in decision-making processes within the financial sector

Business understanding

Problem Statement: Loan companies struggle to lend to people with insufficient or non-existent credit history, which can lead to defaulting.
Analysis Approach: As a worker in a consumer finance company, we need to use EDA to analyze data patterns and ensure that loan applicants who can repay the loan are not rejected. The company has to make loan approval decisions based on the applicant's profile, which entails two risks.
Risk 1: If the applicant can repay the loan, not approving it results in a loss of business for the company.
Risk 2: If the applicant is likely to default, approving the loan may lead to a financial loss for the company

Approach

Analysis Approach: The analysis will include identifying missing data and using appropriate methods to handle it, identifying outliers and data imbalances, and performing univariate, segmentedunivariate, and bivariate analyses to identify important variables. The top 10 correlations for clients with payment difficulties and all othercases will be identified by segmenting the data frame with respect to the target variable.

Missing Data: Missing data will be identified and replaced with an appropriate value or removed, depending on the context. Outliers: Outliers will be identified and explained in business terms, but will not necessarily be removed.

Visualization: The analysis will include visualizations to summarize important results, and insights will explain why variables are important for differentiating clients with payment difficulties from all other cases.

Tech stack used

➢ **Python**

python is used to write a code and performing EDA on the given data. I have used python inbuild packages like pandas, numpy, seaborn etc.

➢ **Jupyter notebook**

I have used jupyter notebook environment for doing analysis. It is more userfriendly and highly interactive.

➢ **Microsoft powerpoint**

Microsoft powerpoint is used for making a project report and chart to visualize the results.

Insights

Import python packages and reading data

In [1]:
```python
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```python
import warnings
warnings.filterwarnings('ignore')
```

reading datasets using pandas

In [3]:
```python
df_ad = pd.read_csv('application_data.csv')
```

In [4]:
```python
df_ad
```

Out[4]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_
0	100002	1	Cash loans	M	N	Y	0	202500.0	
1	100003	0	Cash loans	F	N	N	0	270000.0	1
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	
3	100006	0	Cash loans	F	N	Y	0	135000.0	
4	100007	0	Cash loans	M	N	Y	0	121500.0	
...	...	...	...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	N	0	157500.0	
307507	456252	0	Cash loans	F	N	Y	0	72000.0	
307508	456253	0	Cash loans	F	N	Y	0	153000.0	
307509	456254	1	Cash loans	F	N	Y	0	171000.0	
307510	456255	0	Cash loans	F	N	N	0	157500.0	

Reading the file size and dimensions and analyzing the type of Null values present in the file for further Data Cleaning.

```
In [9]: df_ad.shape

Out[9]: (307511, 122)

In [10]: df_pa.shape

Out[10]: (1670214, 37)

In [11]: df_ad.size

Out[11]: 37516342

In [12]: df_pa.size

Out[12]: 61797918

In [13]: df_ad.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB

In [14]: df_pa.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
```

Finding the file shapes and data types of each column header for further Data Cleaning.

```
In [15]: df_columns_description = pd.read_csv('columns_description.csv', encoding='latin-1')
         df_columns_description.head()
```

Out[15]:

	Unnamed: 0	Table	Row	Description	Special
0	1	application_data	SK_ID_CURR	ID of loan in our sample	NaN
1	2	application_data	TARGET	Target variable (1 - client with payment diffi...	NaN
2	5	application_data	NAME_CONTRACT_TYPE	Identification if loan is cash or revolving	NaN
3	6	application_data	CODE_GENDER	Gender of the client	NaN
4	7	application_data	FLAG_OWN_CAR	Flag if the client owns a car	NaN

```
In [16]: print(df_ad.dtypes)
```

```
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

Cleaning data by firstly counting the total number of null values present in each column of the dataset

```
In [18]: null_count =df_ad.isnull().sum()
         print(null_count)

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
```

funcion to get null value percentage

Using the "column_wise_null_perecntage" function to identify which columns have missing data and how much data is missing in each column.

In [19]:
```
def column_wise_null_percentage(df):
    output = round(df.isnull().sum()/len(df.index)*100,2)
    return output
```

In [20]:
```
null_col = column_wise_null_percentage(df_ad)
null_col
```

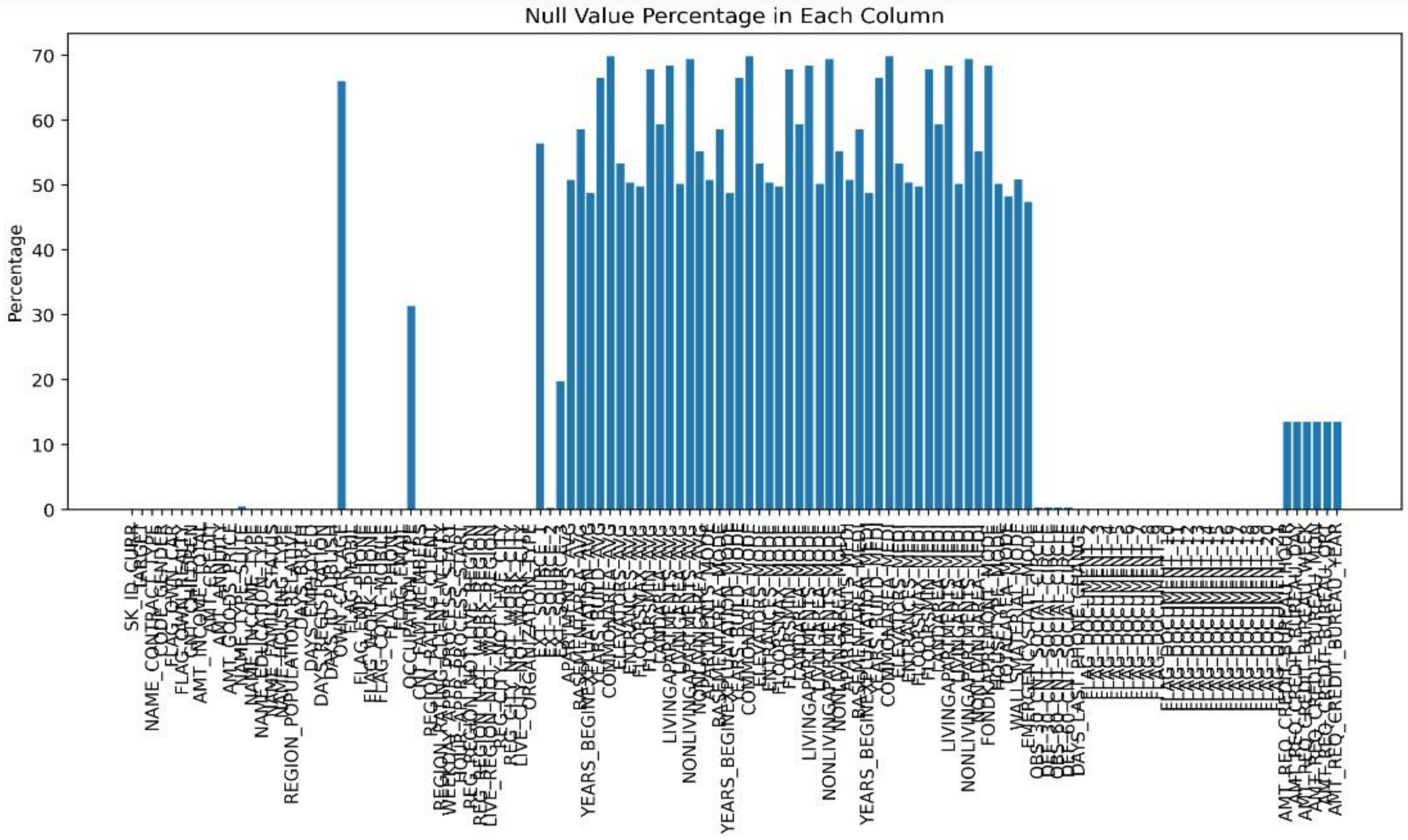
Out[20]:
```
SK_ID_CURR                   0.0
TARGET                       0.0
NAME_CONTRACT_TYPE           0.0
CODE_GENDER                  0.0
FLAG_OWN_CAR                 0.0
                            ... 
AMT_REQ_CREDIT_BUREAU_DAY   13.5
AMT_REQ_CREDIT_BUREAU_WEEK  13.5
AMT_REQ_CREDIT_BUREAU_MON   13.5
AMT_REQ_CREDIT_BUREAU_QRT   13.5
AMT_REQ_CREDIT_BUREAU_YEAR  13.5
Length: 122, dtype: float64
```

Calculating the percentage of null values in each column of the DataFrame df_application_data, and store the results in the variable null_50

```
In [22]: # Percentage of null values in each column of Application_Data
         null_50=round(df_ad.isnull().sum() / df_ad.shape[0] * 100.00,2)
         null_50

Out[22]: SK_ID_CURR                     0.0
         TARGET                         0.0
         NAME_CONTRACT_TYPE             0.0
         CODE_GENDER                    0.0
         FLAG_OWN_CAR                   0.0
                                       ...
         AMT_REQ_CREDIT_BUREAU_DAY     13.5
         AMT_REQ_CREDIT_BUREAU_WEEK    13.5
         AMT_REQ_CREDIT_BUREAU_MON     13.5
         AMT_REQ_CREDIT_BUREAU_QRT     13.5
         AMT_REQ_CREDIT_BUREAU_YEAR    13.5
         Length: 122, dtype: float64
```

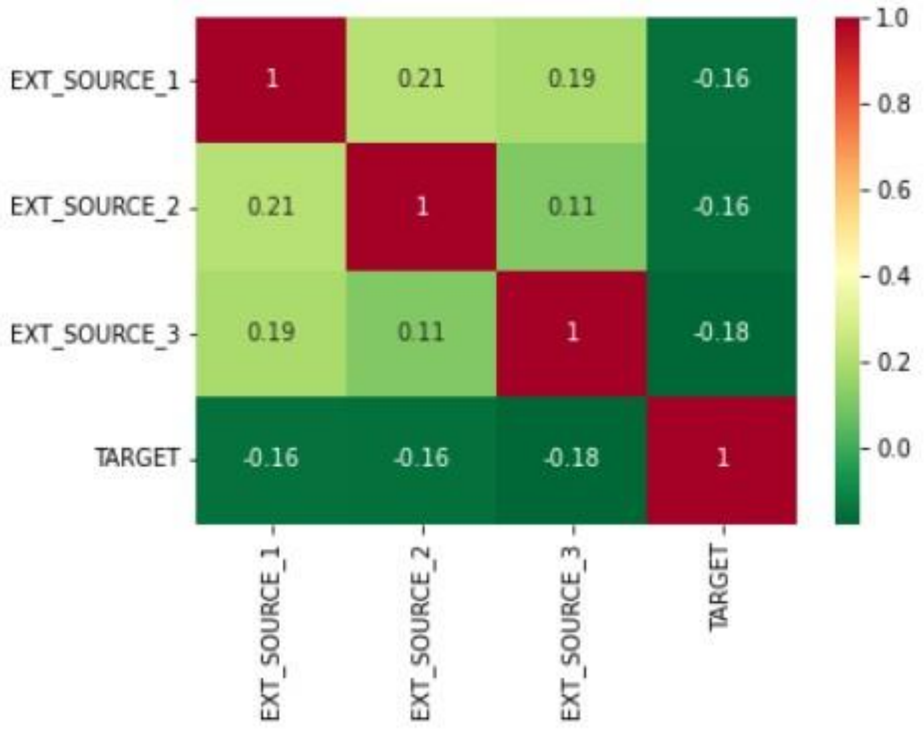
Visualization the graph between null value percentage in each column versus Null value perc entage

```
In [23]: null_percentage = df_ad.isnull().sum() * 100 / df_ad.shape[0]
         null_df = pd.DataFrame({'Column Name': null_percentage.index, 'Null Value Percentage': null_percentage.values})

         # Plot the percentage of null values for each column
         plt.figure(figsize=(13, 5), dpi=400)
         plt.bar(x=null_df['Column Name'], height=null_df['Null Value Percentage'])
         plt.xticks(rotation=90)
         plt.title('Null Value Percentage in Each Column')
         plt.ylabel('Percentage')
         plt.show()
```


Null Value Percentage in Each Column
Percentage
0
10
20
30
40
50
60
70

Checking correlation between External source columns and target columns using heatmap for better overall view

```
In [24]: #correlation of EXT_SOURCE_X columns vs TARGET column
         Source = df_ad[["EXT_SOURCE_1","EXT_SOURCE_2","EXT_SOURCE_3","TARGET"]]
         source_corr = Source.corr()
         ax = sns.heatmap(source_corr,
                     xticklabels=source_corr.columns,
                     yticklabels=source_corr.columns,
                     annot = True,
                     cmap ="RdYlGn_r")
```



Defining Missing Values, this function is used for quickly identifying which columns of a pandas dataframe have the highest percentage of missing data, which can be helpful for subsequent data cleaning and imputation processes.
Showing number of columns having null values with more than 50% using index function, by Filtering out the columns with null values more than 50% and displays the list of data.frame those columns. creating a new dataframe null_50 and storing those values on the column.

```
In [25]: null_50 = null_50[null_50>50]
         print("Number of columns having null value more than 50% :", len(null_50.index))
         print(null_50)
```

```
Number of columns having null value more than 50% : 41
OWN_CAR_AGE                 65.99
EXT_SOURCE_1                56.38
APARTMENTS_AVG              50.75
BASEMENTAREA_AVG            58.52
YEARS_BUILD_AVG             66.50
COMMONAREA_AVG              69.87
ELEVATORS_AVG               53.30
ENTRANCES_AVG               50.35
FLOORSMIN_AVG               67.85
LANDAREA_AVG                59.38
LIVINGAPARTMENTS_AVG        68.35
LIVINGAREA_AVG              50.19
NONLIVINGAPARTMENTS_AVG     69.43
NONLIVINGAREA_AVG           55.18
APARTMENTS_MODE             50.75
BASEMENTAREA_MODE           58.52
YEARS_BUILD_MODE            66.50
COMMONAREA_MODE             69.87
```

Identifying the columns with 15% or less than missing values and displaying them Listing them out from the application dataset having null values lesss than or equal to 15%.

```
In [26]: #to removed 41 columns having null percentage more than 50%.
         df_ad = df_ad.drop(null_50.index, axis =1)
         df_ad.shape

Out[26]: (307511, 81)

In [27]: #columns having <15% null values
         Null_15 = null_col[null_col<15]
         print("Number of columns having null value less than 15% :", len(Null_15.index))
         print(Null_15)

         Number of columns having null value less than 15% : 71
         SK_ID_CURR                     0.0
         TARGET                         0.0
         NAME_CONTRACT_TYPE             0.0
         CODE_GENDER                    0.0
         FLAG_OWN_CAR                   0.0
                                       ...
         AMT_REQ_CREDIT_BUREAU_DAY     13.5
         AMT_REQ_CREDIT_BUREAU_WEEK    13.5
         AMT_REQ_CREDIT_BUREAU_MON     13.5
         AMT_REQ_CREDIT_BUREAU_QRT     13.5
         AMT_REQ_CREDIT_BUREAU_YEAR    13.5
         Length: 71, dtype: float64
```

In [29]:
```
df_ad[Null_15.index].describe()
```

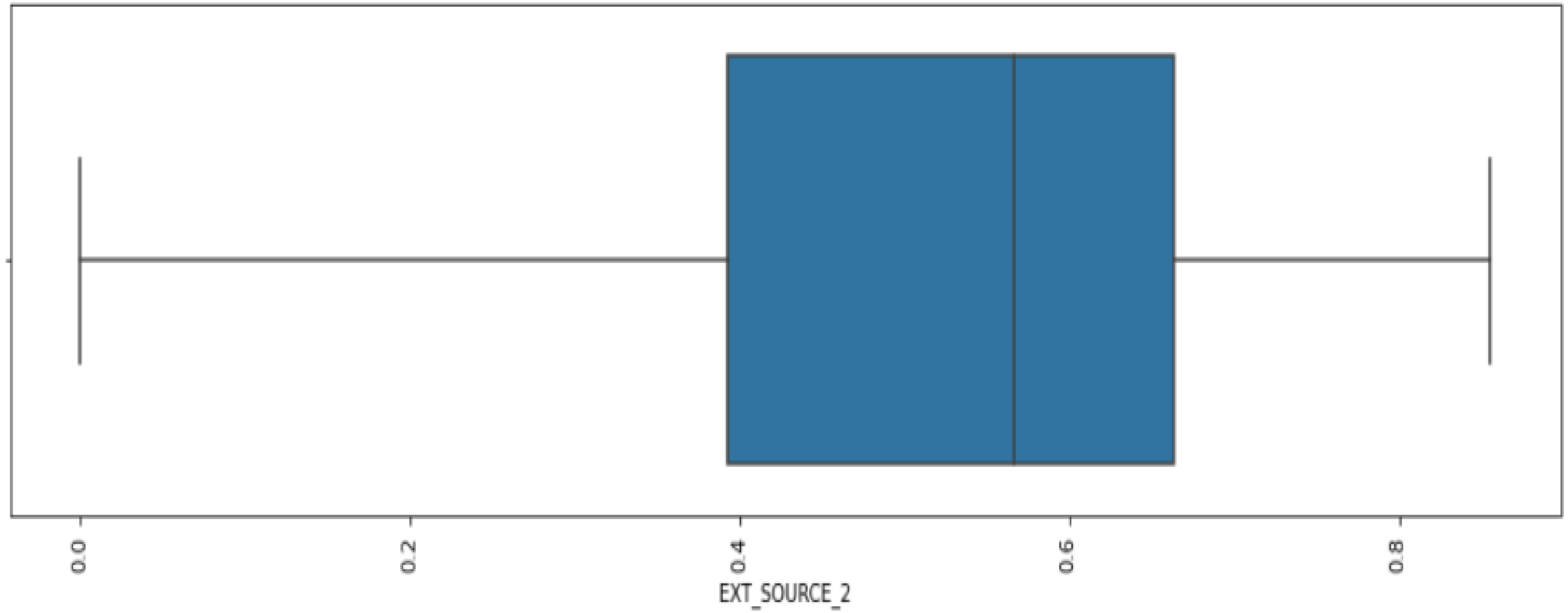
Out[29]:

SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_POPULATION_RELATIVE	DA
307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	307499.000000	3.072330e+05	307511.000000	3075
278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	27108.573909	5.383962e+05	0.020868	-160
102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	14493.737315	3.694465e+05	0.013831	43
100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1615.500000	4.050000e+04	0.000290	-252
189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16524.000000	2.385000e+05	0.010006	-196
278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	24903.000000	4.500000e+05	0.018850	-157
367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.000000	6.795000e+05	0.028663	-124
456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	258025.500000	4.050000e+06	0.072508	-74

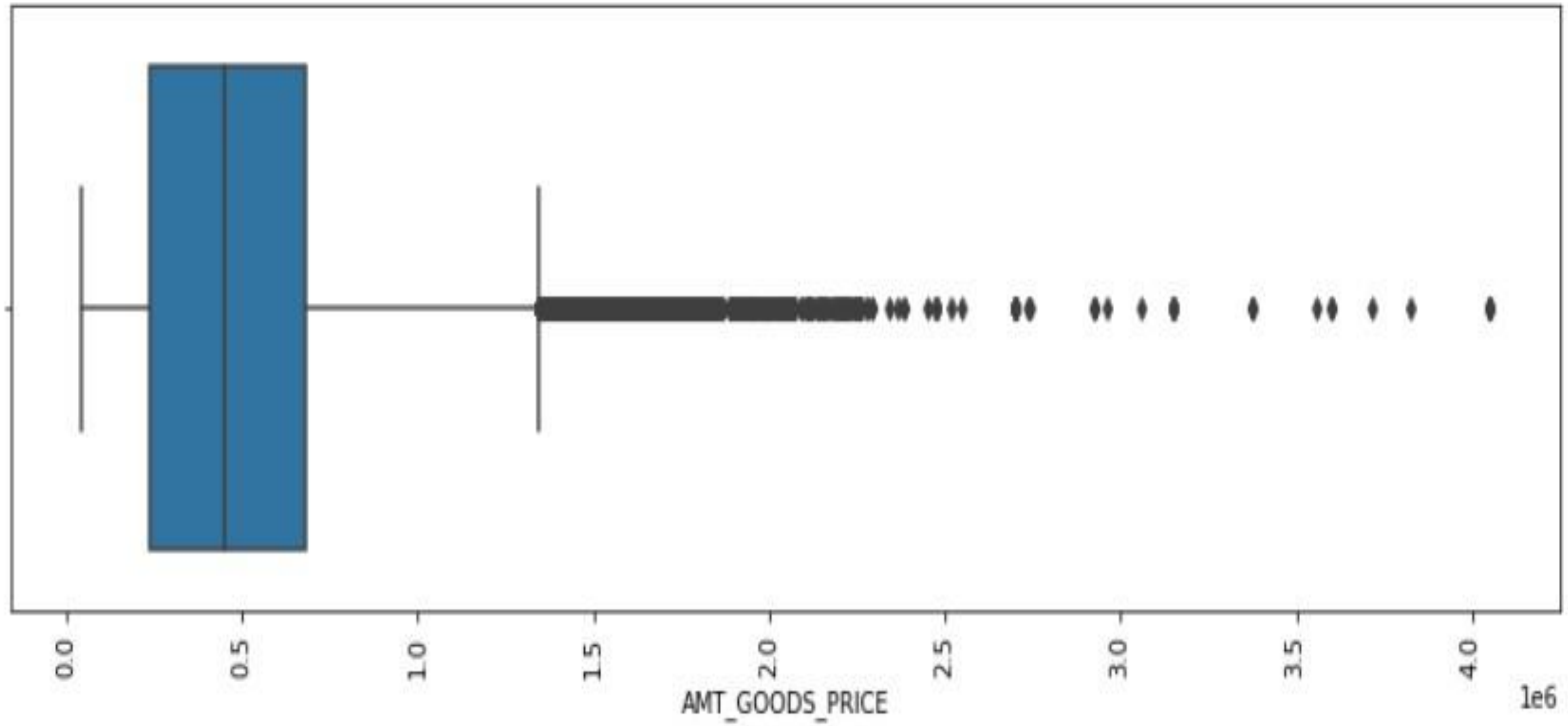
× 60 columns

Continuous variables and categorical variables using box plot to visualise the outliers identify potential relationships between variables.

```
In [31]: # Box plot for continuous variables
plt.figure(figsize=(16,4))
sns.boxplot(x=df_ad['EXT_SOURCE_2'], orient='h')
plt.xticks(rotation=90)
plt.show()
```



In [32]:
```
plt.figure(figsize=(13,4))
sns.boxplot(x=df_ad['AMT_GOODS_PRICE'])
plt.xticks(rotation=90)
plt.show()
```



for 'EXT_SOURCE_2' there is no outliers present. for 'AMT_GOODS_PRICE' there is significant number of outlier present in the data. SO data should be imputed with median value: 450000

Listingout the maximum frequency an dremoving unwanted columns

```
In [33]: unwanted=['FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE','FLAG_PHONE', 'FLAG_EMAIL',
                   'REGION_RATING_CLIENT','REGION_RATING_CLIENT_W_CITY','FLAG_EMAIL','CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
                   'REGION_RATING_CLIENT_W_CITY','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','EXT_SOURCE_2','EXT_SOURCE_3','YEARS_BEGINEXPLUATATION_AVG','FLOORSMAX_AVG','YEARS_BEGINEXPLUATATION
                   'FLOORSMAX_MODE','YEARS_BEGINEXPLUATATION_MEDI','FLOORSMAX_MEDI','TOTALAREA_MODE','EMERGENCYSTATE_MODE']

         df_ad.drop(labels=unwanted,axis=1,inplace=True)
```

```
In [34]: df_ad.shape
```

```
Out[34]: (307511, 42)
```

```
In [35]: df_ad.head()
```

Out[35]:

TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY
1	Cash loans	M	N	Y	0	202500.0	406597.5	24700.5
0	Cash loans	F	N	N	0	270000.0	1293502.5	35698.5
0	Revolving loans	M	Y	Y	0	67500.0	135000.0	6750.0
0	Cash loans	F	N	Y	0	135000.0	312682.5	29686.5
0	Cash loans	M	N	Y	0	121500.0	513000.0	21865.5

we have to find not avaliable values in rows and columns

In [36]:
```
print('CODE_GENDER: ',df_ad['CODE_GENDER'].unique())
print('No of values: ',df_ad[df_ad['CODE_GENDER']=='XNA'].shape[0])

XNA_count = df_ad[df_ad['CODE_GENDER']=='XNA'].shape[0]
per_XNA = round(XNA_count/len(df_ad.index)*100,3)

print('% of XNA Values:',  per_XNA)

print('maximum frequency data :', df_ad['CODE_GENDER'].describe().top)
```

```
CODE_GENDER:  ['M' 'F' 'XNA']
No of values:  4
% of XNA Values: 0.001
maximum frequency data : F
```

there are only 2 rows having not avaliable values

In [37]:
```
# Dropping the NA value in column
df_ad = df_ad.drop(df_ad.loc[df_ad['CODE_GENDER']=='XNA'].index)
df_ad[df_ad['CODE_GENDER']=='XNA'].shape
```

Out[37]:
```
(0, 42)
```

In [37]:
```
# Dropping the NA value in column
df_ad = df_ad.drop(df_ad.loc[df_ad['CODE_GENDER']=='XNA'].index)
df_ad[df_ad['CODE_GENDER']=='XNA'].shape
```
Out[37]:
```
(0, 42)
```
In [38]:
```
print('No of XNA values: ', df_ad[df_ad['ORGANIZATION_TYPE']=='XNA'].shape[0])

XNA_count = df_ad[df_ad['ORGANIZATION_TYPE']=='XNA'].shape[0]
per_XNA = round(XNA_count/len(df_ad.index)*100,3)

print('% of XNA Values:',  per_XNA)

df_ad['ORGANIZATION_TYPE'].describe()
```
```
No of XNA values:  55374
% of XNA Values: 18.007
```
Out[38]:
```
count                       307507
unique                          58
top       Business Entity Type 3
freq                         67992
Name: ORGANIZATION_TYPE, dtype: object
```
In [39]:
```
df_ad.head()
```
Out[39]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREI
0	100002	1	Cash loans	M	N	Y	0	202500.0	40659
1	100003	0	Cash loans	F	N	N	0	270000.0	129350
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	13500
3	100006	0	Cash loans	F	N	Y	0	135000.0	31268
4	100007	0	Cash loans	M	N	Y	0	121500.0	51300

converting the followingage / days columns having - value to +value

```
In [41]: # Converting '-' values into '+' Values
df_ad['DAYS_BIRTH'] = df_ad['DAYS_BIRTH'].abs()
df_ad['DAYS_EMPLOYED'] = df_ad['DAYS_EMPLOYED'].abs()
df_ad['DAYS_REGISTRATION'] = df_ad['DAYS_REGISTRATION'].abs()
df_ad['DAYS_ID_PUBLISH'] = df_ad['DAYS_ID_PUBLISH'].abs()
df_ad['DAYS_LAST_PHONE_CHANGE'] = df_ad['DAYS_LAST_PHONE_CHANGE'].abs()
```

to find outlier ¶

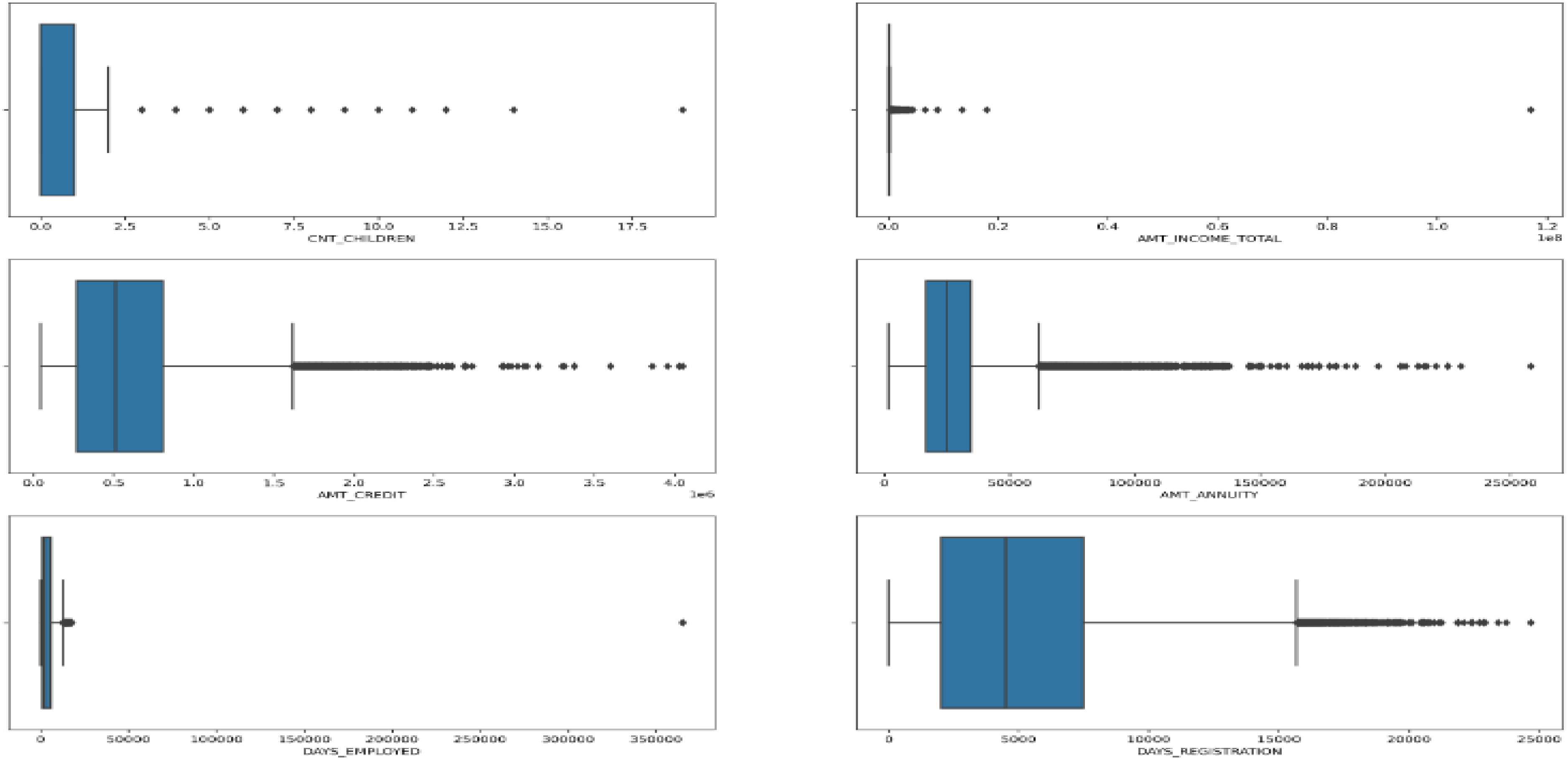
```
In [42]: df_ad[numeric_columns].describe()
```

Out[42]:

	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	REGION_POPULATION_RELATIVE	DAYS_BIRTH	DAYS_EMPLOYED
count	307507.00000	307507.000000	3.075070e+05	3.075070e+05	307495.000000	307507.000000	307507.000000	307507.000000
mean	0.08073	0.417047	1.687977e+05	5.990286e+05	27108.666786	0.020868	16037.027271	67725.569893
std	0.27242	0.722119	2.371246e+05	4.024926e+05	14493.798379	0.013831	4363.982424	139444.469301
min	0.00000	0.000000	2.565000e+04	4.500000e+04	1615.500000	0.000290	7489.000000	0.000000
25%	0.00000	0.000000	1.125000e+05	2.700000e+05	16524.000000	0.010006	12413.000000	933.000000
50%	0.00000	0.000000	1.471500e+05	5.135310e+05	24903.000000	0.018850	15750.000000	2219.000000
75%	0.00000	1.000000	2.025000e+05	8.086500e+05	34596.000000	0.028663	19682.000000	5707.000000
max	1.00000	19.000000	1.170000e+08	4.050000e+06	258025.500000	0.072508	25229.000000	365243.000000

Detecting outliers using Box plot for the selected columns.

```
In [43]: # Box plot for selected columns
features = ['CNT_CHILDREN', 'AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','DAYS_EMPLOYED', 'DAYS_REGISTRATION']

plt.figure(figsize = (20, 15), dpi=300)
for i in enumerate(features):
    plt.subplot(3, 2, i[0]+1)
    sns.boxplot(x = i[1], data = df_ad)
plt.show()
```



inference most of data present in first quartile in CNT_CHILDREN there is one single high value present in AMT_INCOME_TOTAL

Creating bins for continuous categorical variables so as to reduce the number of unique values in a variables.

In [44]:
```
bins = [0,100000,200000,300000,400000,500000,10000000000]
slot = ['<100000', '100000-200000','200000-300000','300000-400000','400000-500000', '500000 and above']

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

In [45]:
```
bins = [0,100000,200000,300000,400000,500000,600000,700000,800000,900000,10000000000]
slot = ['<100000', '100000-200000','200000-300000','300000-400000','400000-500000', '500000-600000',
        '600000-700000','700000-800000','850000-900000','900000 and above']

df_ad['AMT_CREDIT_RANGE']=pd.cut(df_ad['AMT_CREDIT'],bins,labels=slot)
```

In [46]:
```
# Dividing the dataset into two dataset of  target=1(client with payment difficulties) and target=0(all other)

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

Dividing the targets for an alysis in to 2 for percentage defaulter so f people who did pay and did n ot pay their loan

```
In [47]: # insights from number of target values

percentage_defaulters= round(100*len(target1_df)/(len(target0_df)+len(target1_df)),2)

percentage_nondefaulters=round(100*len(target0_df)/(len(target0_df)+len(target1_df)),2)

print('Count of target0_df:', len(target0_df))
print('Count of target1_df:', len(target1_df))


print('Percentage of people who paid their loan are: ', percentage_nondefaulters, '%' )
print('Percentage of people who did not paid their loan are: ', percentage_defaulters, '%' )
```

```
Count of target0_df: 282682
Count of target1_df: 24825
Percentage of people who paid their loan are:  91.93 %
Percentage of people who did not paid their loan are:  8.07 %
```

```
In [48]: imb_ratio = round(len(target0_df)/len(target1_df),2)

print('imbalance Ratio:', imb_ratio)
```

```
imbalance Ratio: 11.39
```

Univariate Analysis

UNIVARIATE ANALYSIS ¶

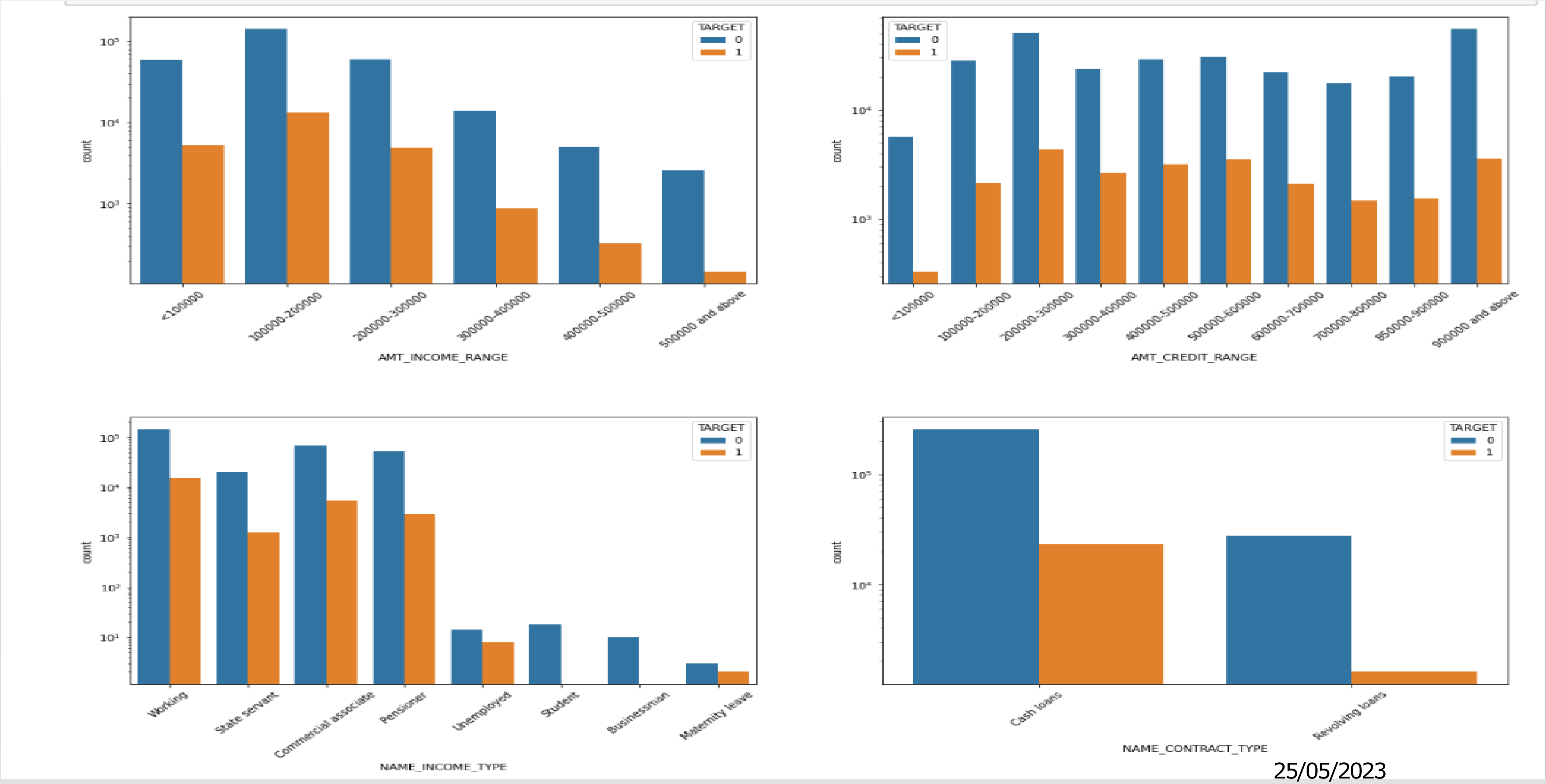
In [49]:
```
# Count plotting in logarithmic scale
def uniplot(df,col,title,hue =None):

    sns.set_style('whitegrid')
    sns.set_context('talk')
    plt.rcParams["axes.labelsize"] = 14
    plt.rcParams['axes.titlesize'] = 16
    plt.rcParams['axes.titlepad'] = 14


    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)

    plt.show()
```

In [50]:
```
# Categoroical Univariate Analysis in logarithmic scale

features = ['AMT_INCOME_RANGE', 'AMT_CREDIT_RANGE','NAME_INCOME_TYPE','NAME_CONTRACT_TYPE']
plt.figure(figsize = (20, 15))

for i in enumerate(features):
    plt.subplot(2, 2, i[0]+1)
    plt.subplots_adjust(hspace=0.5)
    sns.countplot(x = i[1], hue = 'TARGET', data = df_ad)

    plt.rcParams['axes.titlesize'] = 16

    plt.xticks(rotation = 45)
    plt.yscale('log')
```

-The people having 100000-200000 are having higher number of loan and also having higher value in defaulter The income segment having >500000 are having less defaulter.

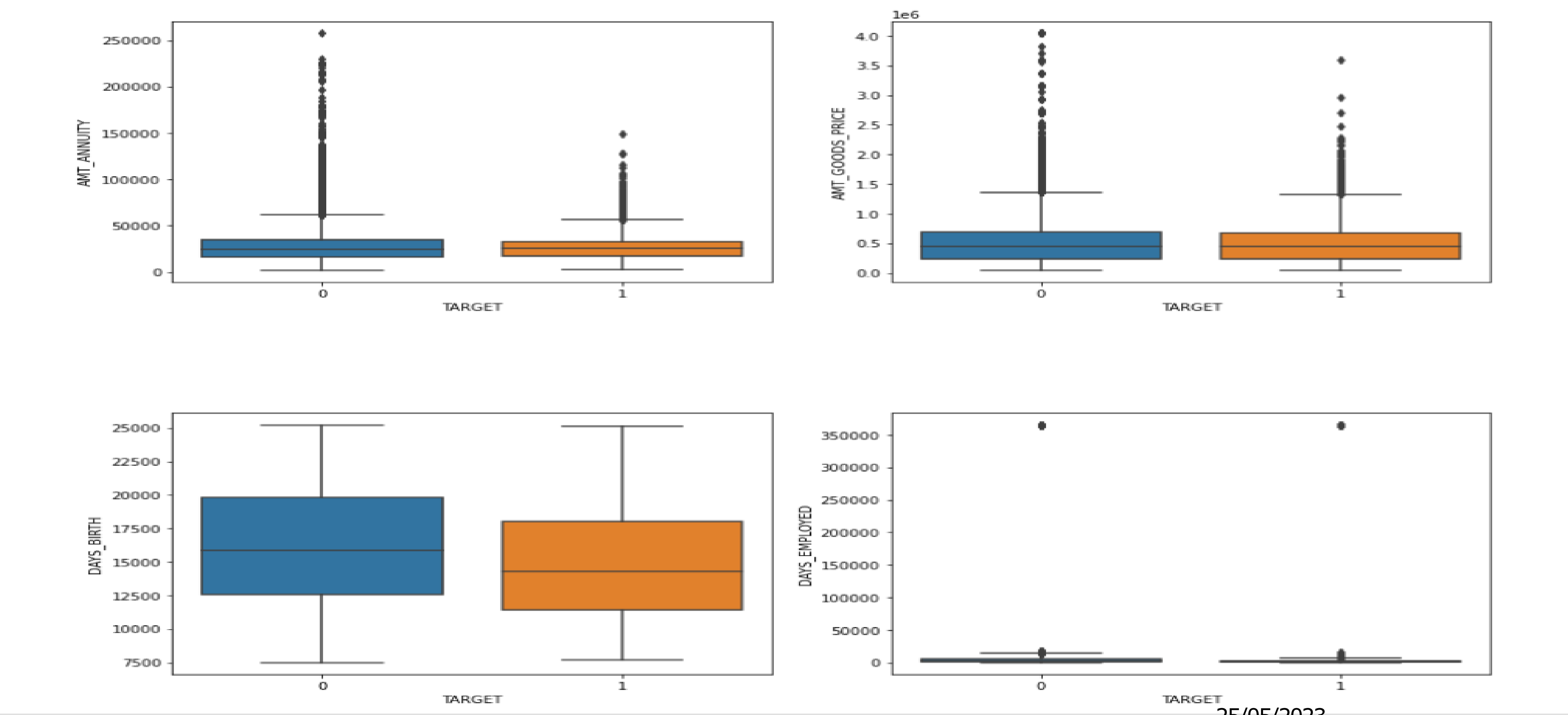
-Student pensioner and business have higher percentage of loan repayment.

-Income having more than >100000 are almost equal %to loan defaulter



Univariate Analysis for continuous variables

Less outlier observed in Days_Birth and DAYS_ID_PUBLISH Days_Birth: The people having higher age are having higher probability of repayment. 1st quartile is smaller than third quartile in In 'AMT_ANNUITY','AMT_GOODS_PRICE', DAYS_LAST_PHONE_CHANGE. In DAYS_ID_PUBLISH: people changing ID in recent days are relativelty prone to be default

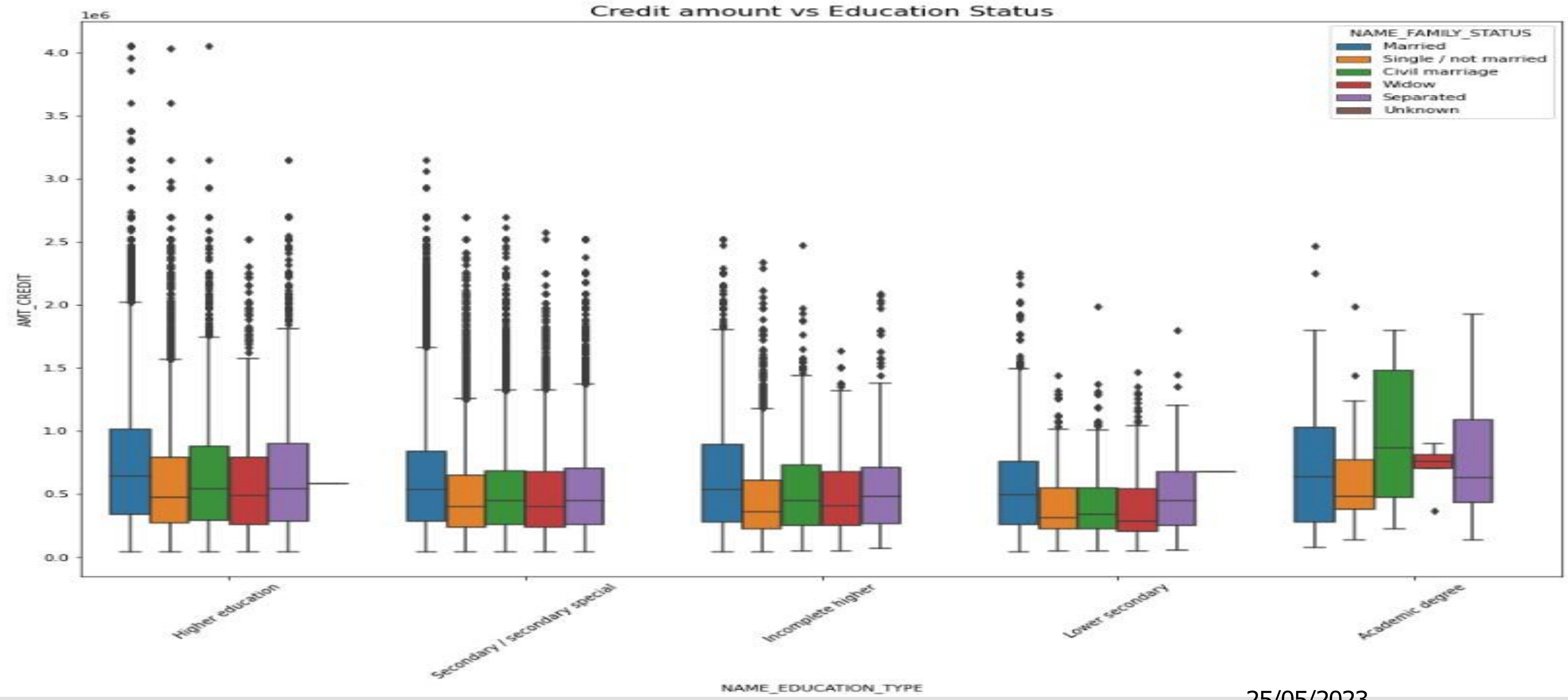


BIVARIATE ANALYSIS

For target 0

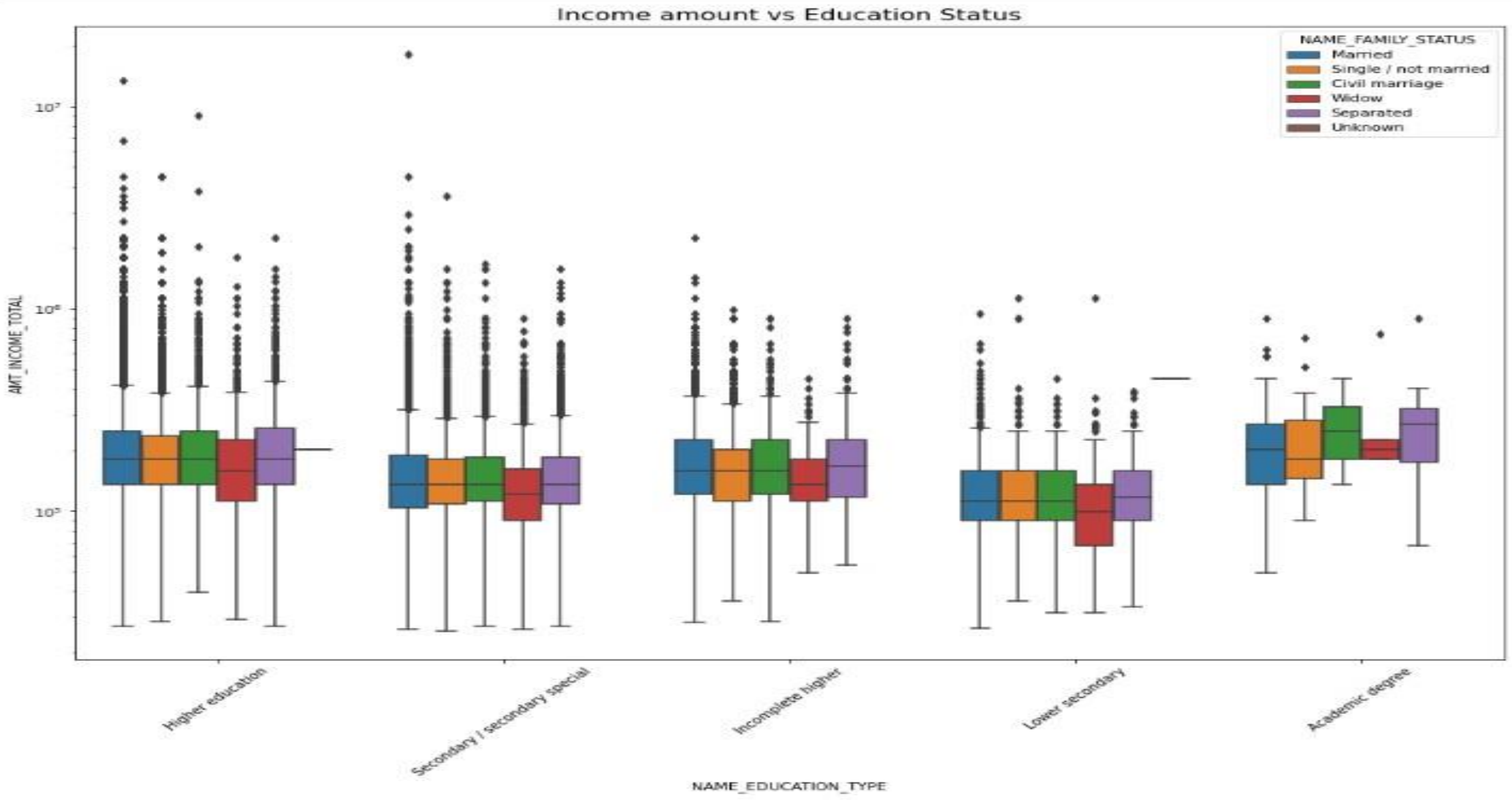
Family status of 'civil marriage', 'marriage' and 'separated' of Academic degree education are having higher number of credits than others. Also, higher education of family status of 'marriage', 'single' and 'civil marriage' has more outliers. Civil marriage for Academic degree has most of the credits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

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



In Education type 'Higher education' the income amount is mostly equal to family status. And contain many outliers. Less outlier are present for Academic degree although their income amount is bit higher than Higher education. Lower secondary of civil marriage family status are having less income amount than others.

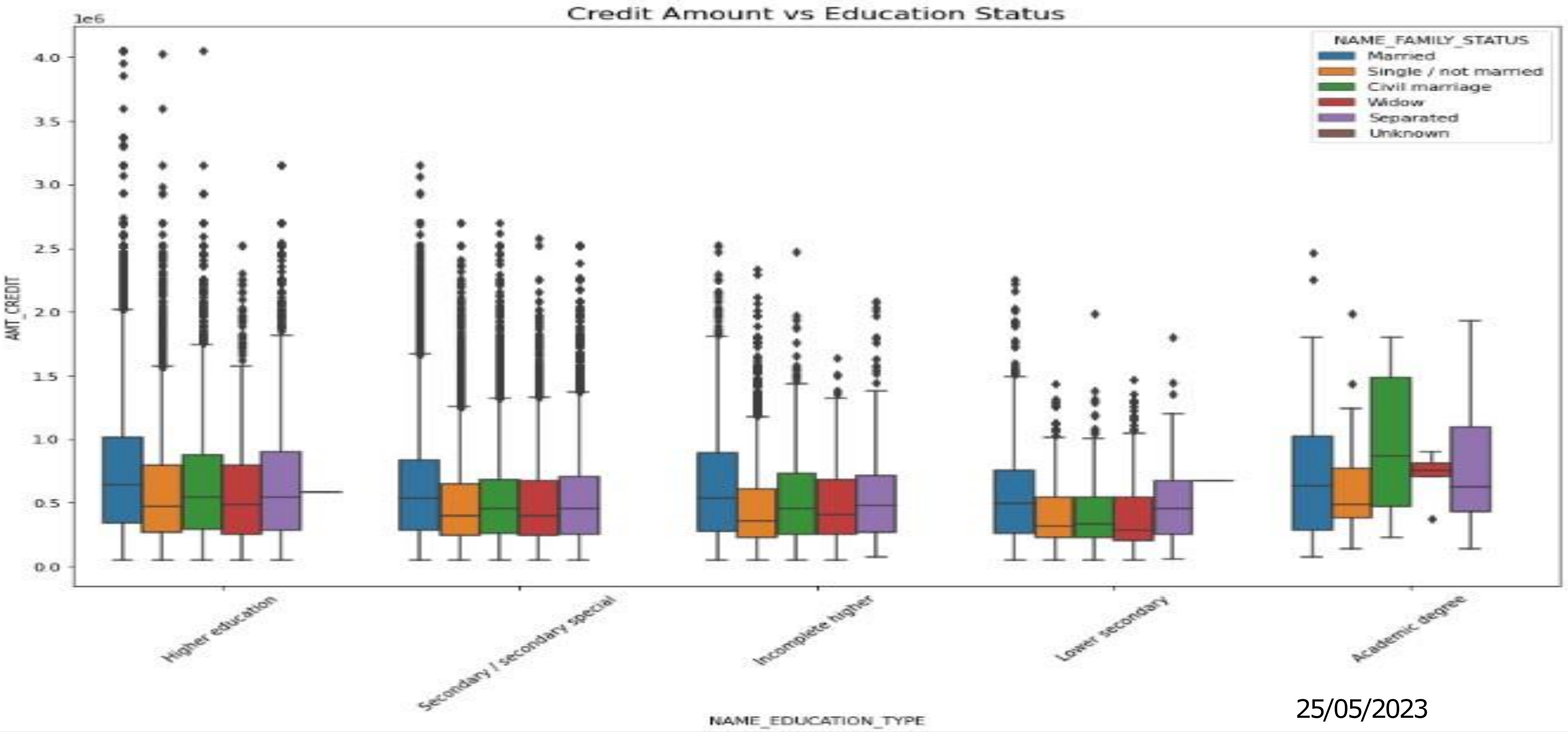
```
plt.figure(figsize=(16,12))
plt.xticks(rotation=45)
plt.yscale('log')
sns.boxplot(data =target0_df, x='NAME_EDUCATION_TYPE',
            y='AMT_INCOME_TOTAL', hue ='NAME_FAMILY_STATUS',orient='v')
plt.title('Income amount vs Education Status')
plt.show()
```



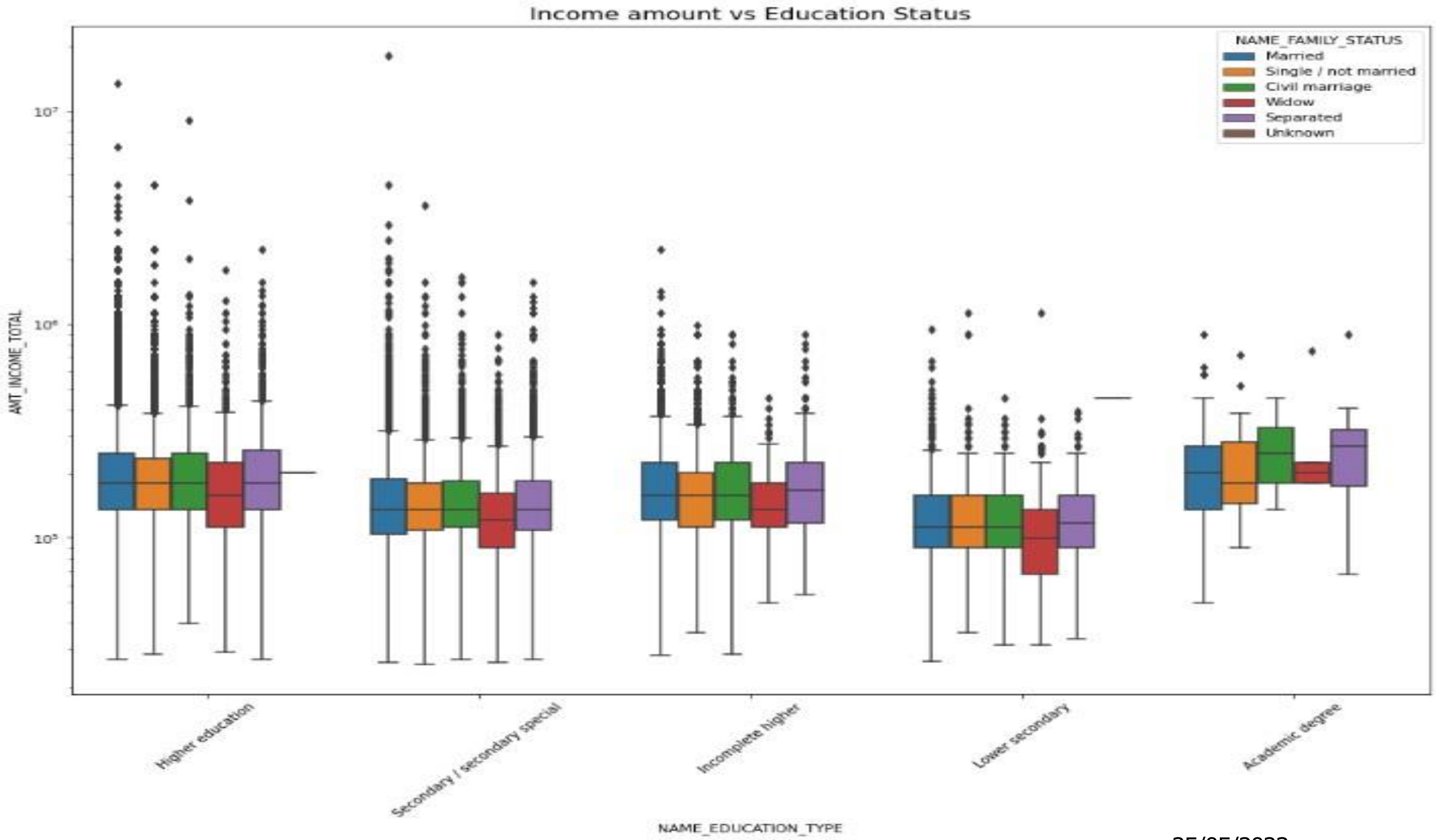
for target 1

It can be inferred that they are very similar to Target 0 Family status of 'civil marriage', 'marriage' and 'separated' of Academic degree education as they have higher number of credits than others. Most of the outliers are from Education type 'Higher education' and 'Secondary'. Civil marriage for Academic degree has most of the credits in the third quartile.

In [54]:
```
# Box plotting for credit amount

plt.figure(figsize=(15,10))
plt.xticks(rotation=45)
sns.boxplot(data =target0_df, x='NAME_EDUCATION_TYPE',
            y='AMT_CREDIT', hue ='NAME_FAMILY_STATUS',orient='v')
plt.title('Credit Amount vs Education Status')
plt.show()
```



```python
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_FAMILY_STATUS',orient='v')
plt.title('Income amount vs Education Status')
plt.show()
```



Correlation

```
In [54]: # Top 10 correlated variables: target 0 dataframe

corr = target0_df.corr(numeric_only=True)
corrdf = corr.where(np.triu(np.ones(corr.shape), k=1).astype(bool))
corrdf = corrdf.unstack().reset_index()
corrdf.columns = ['Var1', 'Var2', 'Correlation']
corrdf.dropna(subset = ['Correlation'], inplace = True)
corrdf['Correlation'] = round(corrdf['Correlation'], 2)
corrdf['Correlation'] = abs(corrdf['Correlation'])
corrdf.sort_values(by = 'Correlation', ascending = False).head(10)
```

Out[54]:

	Var1	Var2	Correlation
649	OBS_60_CNT_SOCIAL_CIRCLE	OBS_30_CNT_SOCIAL_CIRCLE	1.00
184	AMT_GOODS_PRICE	AMT_CREDIT	0.99
680	DEF_60_CNT_SOCIAL_CIRCLE	DEF_30_CNT_SOCIAL_CIRCLE	0.86
464	LIVE_REGION_NOT_WORK_REGION	REG_REGION_NOT_WORK_REGION	0.86
557	LIVE_CITY_NOT_WORK_CITY	REG_CITY_NOT_WORK_CITY	0.83
185	AMT_GOODS_PRICE	AMT_ANNUITY	0.78
154	AMT_ANNUITY	AMT_CREDIT	0.77
278	DAYS_EMPLOYED	DAYS_BIRTH	0.63
433	REG_REGION_NOT_WORK_REGION	REG_REGION_NOT_LIVE_REGION	0.45
526	REG_CITY_NOT_WORK_CITY	REG_CITY_NOT_LIVE_CITY	0.44

In [55]:
```python
# Top 10 correlated variables: target 1 dataframe

corr = target1_df.corr(numeric_only=True)
corrdf = corr.where(np.triu(np.ones(corr.shape), k=1).astype(bool))
corrdf = corrdf.unstack().reset_index()
corrdf.columns = ['Var1', 'Var2', 'Correlation']
corrdf.dropna(subset = ['Correlation'], inplace = True)
corrdf['Correlation'] = round(corrdf['Correlation'], 2)
corrdf['Correlation'] = abs(corrdf['Correlation'])
corrdf.sort_values(by = 'Correlation', ascending = False).head(10)
```

Out[55]:

	Var1	Var2	Correlation
649	OBS_60_CNT_SOCIAL_CIRCLE	OBS_30_CNT_SOCIAL_CIRCLE	1.00
184	AMT_GOODS_PRICE	AMT_CREDIT	0.98
680	DEF_60_CNT_SOCIAL_CIRCLE	DEF_30_CNT_SOCIAL_CIRCLE	0.87
464	LIVE_REGION_NOT_WORK_REGION	REG_REGION_NOT_WORK_REGION	0.85
557	LIVE_CITY_NOT_WORK_CITY	REG_CITY_NOT_WORK_CITY	0.78
185	AMT_GOODS_PRICE	AMT_ANNUITY	0.75
154	AMT_ANNUITY	AMT_CREDIT	0.75
278	DAYS_EMPLOYED	DAYS_BIRTH	0.58
433	REG_REGION_NOT_WORK_REGION	REG_REGION_NOT_LIVE_REGION	0.50
526	REG_CITY_NOT_WORK_CITY	REG_CITY_NOT_LIVE_CITY	0.47

Read Previous Application data and merging with application data

```
In [57]: df_pa = pd.read_csv('previous_application.csv')
```

```
In [58]: df_pa
```

Out[58]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	WEE
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	0.0	17145.0	
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	NaN	607500.0	
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	NaN	112500.0	
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	NaN	450000.0	
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	NaN	337500.0	
...	...	...	...	...	...	...	...	...	
'0209	2300464	352015	Consumer loans	14704.290	267295.5	311400.0	0.0	267295.5	
'0210	2357031	334635	Consumer loans	6622.020	87750.0	64291.5	29250.0	87750.0	
'0211	2659632	249544	Consumer loans	11520.855	105237.0	102523.5	10525.5	105237.0	
'0212	2785582	400317	Cash loans	18821.520	180000.0	191880.0	NaN	180000.0	
'0213	2418762	261212	Cash loans	16431.300	360000.0	360000.0	NaN	360000.0	

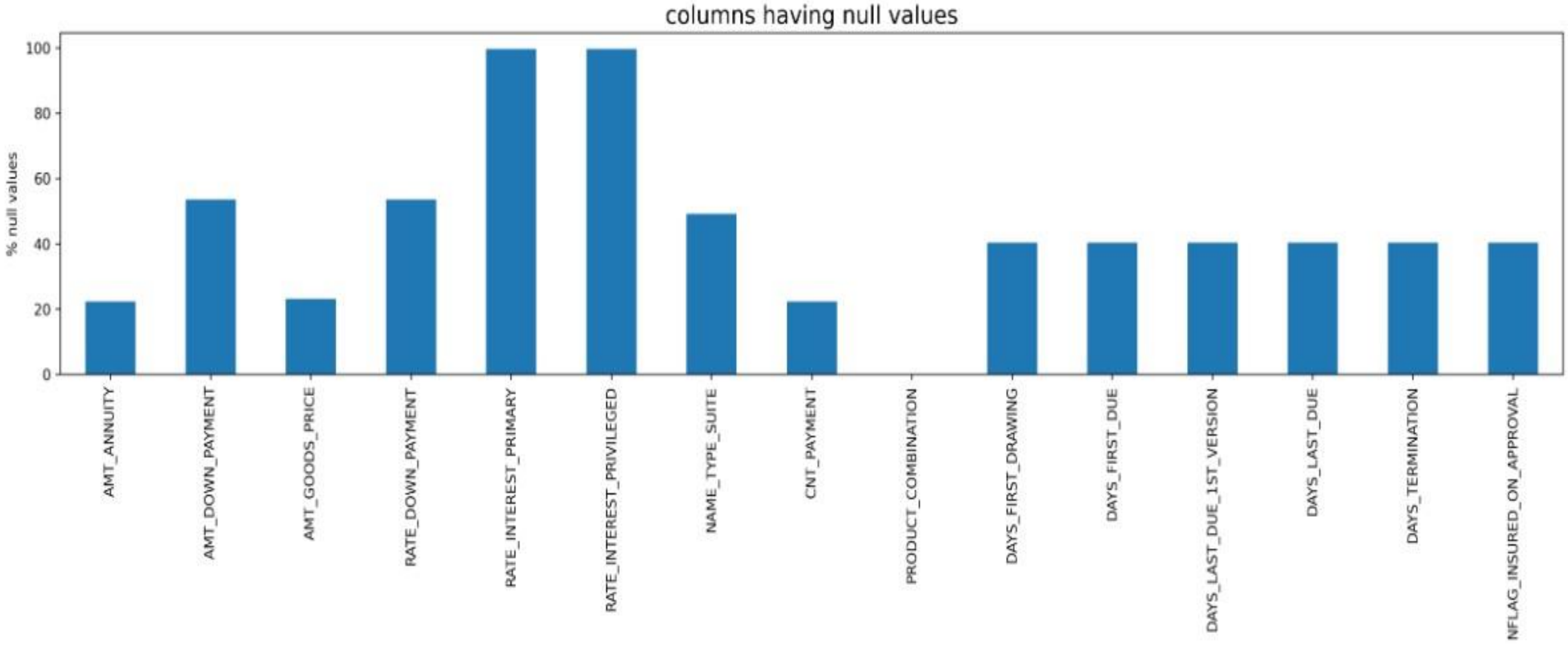
0214 rows × 37 columns

```
In [62]: df_pa.columns
```

```
Out[62]: Index(['SK_ID_PREV', 'SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_ANNUITY',
       'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_DOWN_PAYMENT', 'AMT_GOODS_PRICE',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       'FLAG_LAST_APPL_PER_CONTRACT', 'NFLAG_LAST_APPL_IN_DAY',
```

In [68]:

```
# graphical representation of columns having % null values
plt.figure(figsize= (20,4),dpi=300)
Null_prev.plot(kind = 'bar')
plt.title ('columns having null values')
plt.ylabel('% null values')
plt.show()
```



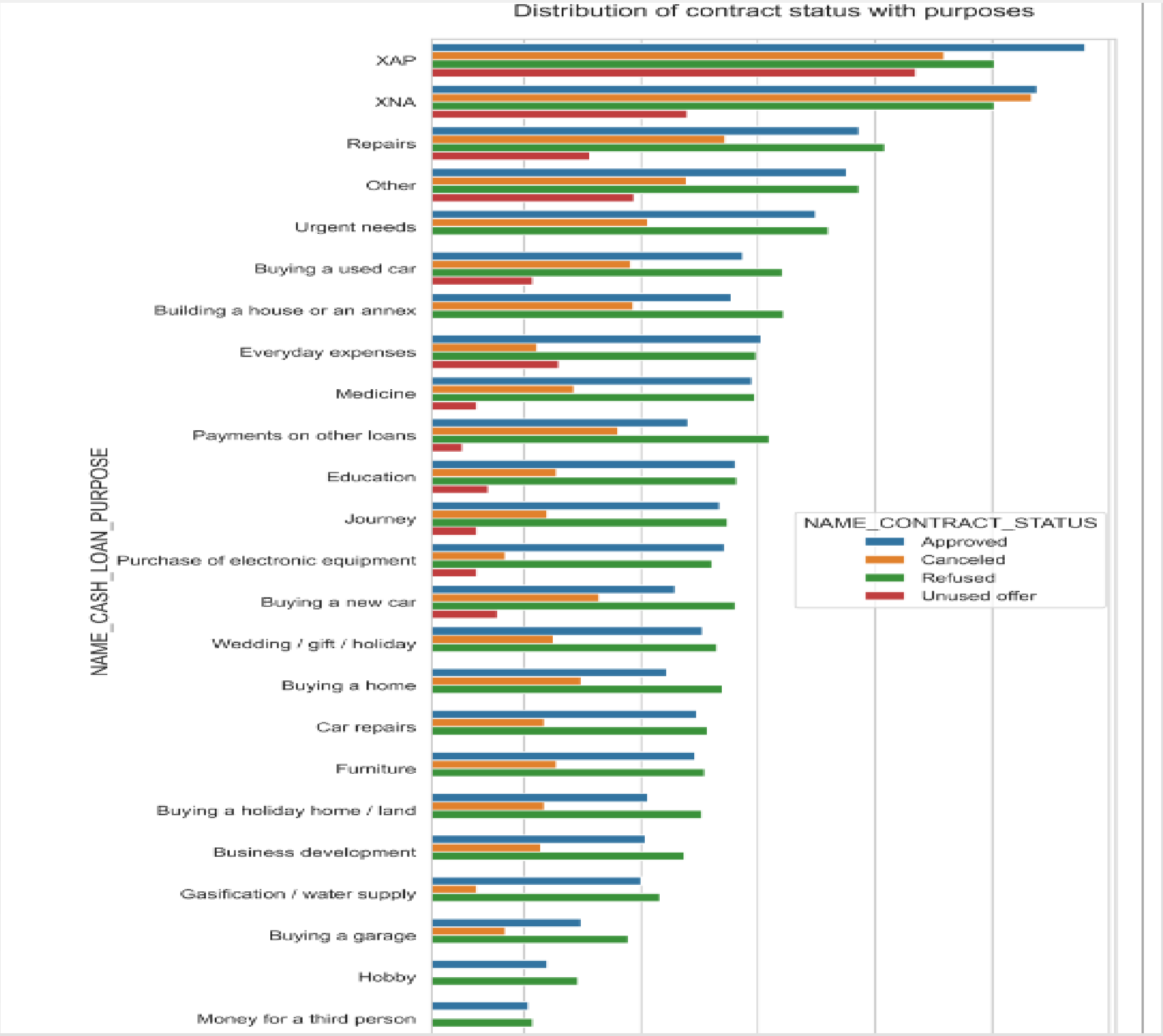
Extracting columns with null values over 50%

```
In [69]: # Get the column with null values more than 50%
         Null_prev = Null_prev[Null_prev>50]
         print("Number of columns having null value more than 50% :", len(Null_prev.index))
         print(Null_prev)

Number of columns having null value more than 50% : 4
AMT_DOWN_PAYMENT             53.64
RATE_DOWN_PAYMENT            53.64
RATE_INTEREST_PRIMARY        99.64
RATE_INTEREST_PRIVILEGED     99.64
dtype: float64

In [70]: # removed 4 columns having null percentage more than 50%.
         df_pa = df_pa.drop(Null_prev.index, axis =1)
         df_pa.shape

Out[70]: (1670214, 33)

In [71]: # Merging the Application dataset with previous appliaction dataset
         df_combine = pd.merge(left=df_ad, right=df_pa, how='inner', on='SK_ID_CURR', suffixes=('_x', '_y'))
         df_combine.shape

Out[71]: (1413646, 76)

In [72]: df_combine.head()

Out[72]:
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CR
0	100002	1	Cash loans	M	N	Y	0	202500.0	4
1	100003	0	Cash loans	F	N	N	0	270000.0	12
2	100003	0	Cash loans	F	N	N	0	270000.0	12
3	100003	0	Cash loans	F	N	N	0	270000.0	12
4	100004	0	Revolving loans	M	Y	Y	0	67500.0	1

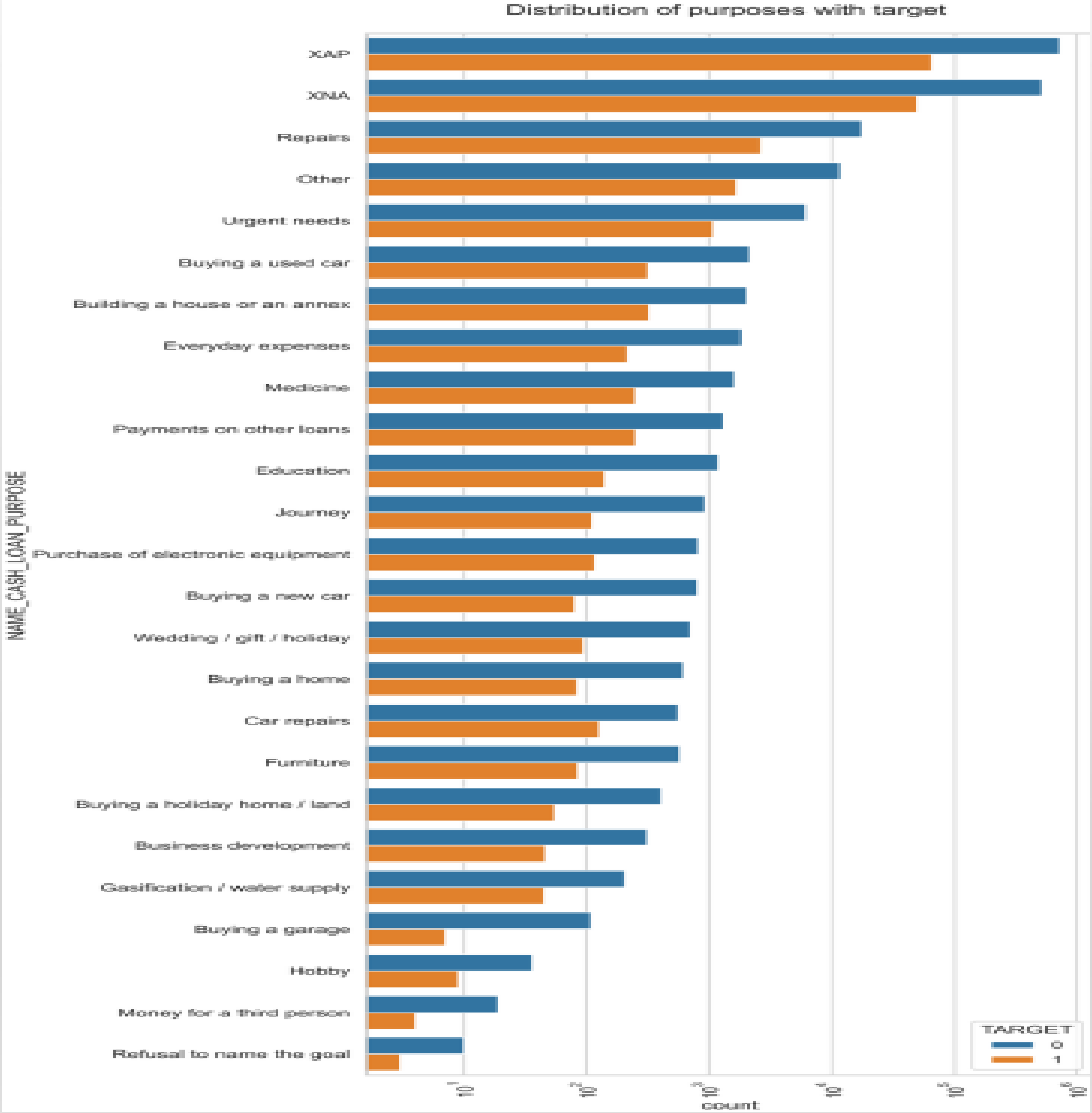
Univariate Analysis

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 [75]: sns.set_style('whitegrid')
         sns.set_context('talk')

         plt.figure(figsize=(10,25),dpi = 300)
         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 = df_combine, y= 'NAME_CASH_LOAN_PURPOSE',
                   order=df_combine['NAME_CASH_LOAN_PURPOSE'].value_counts().index,hue = 'NAME_CONTRACT_STATUS')
```

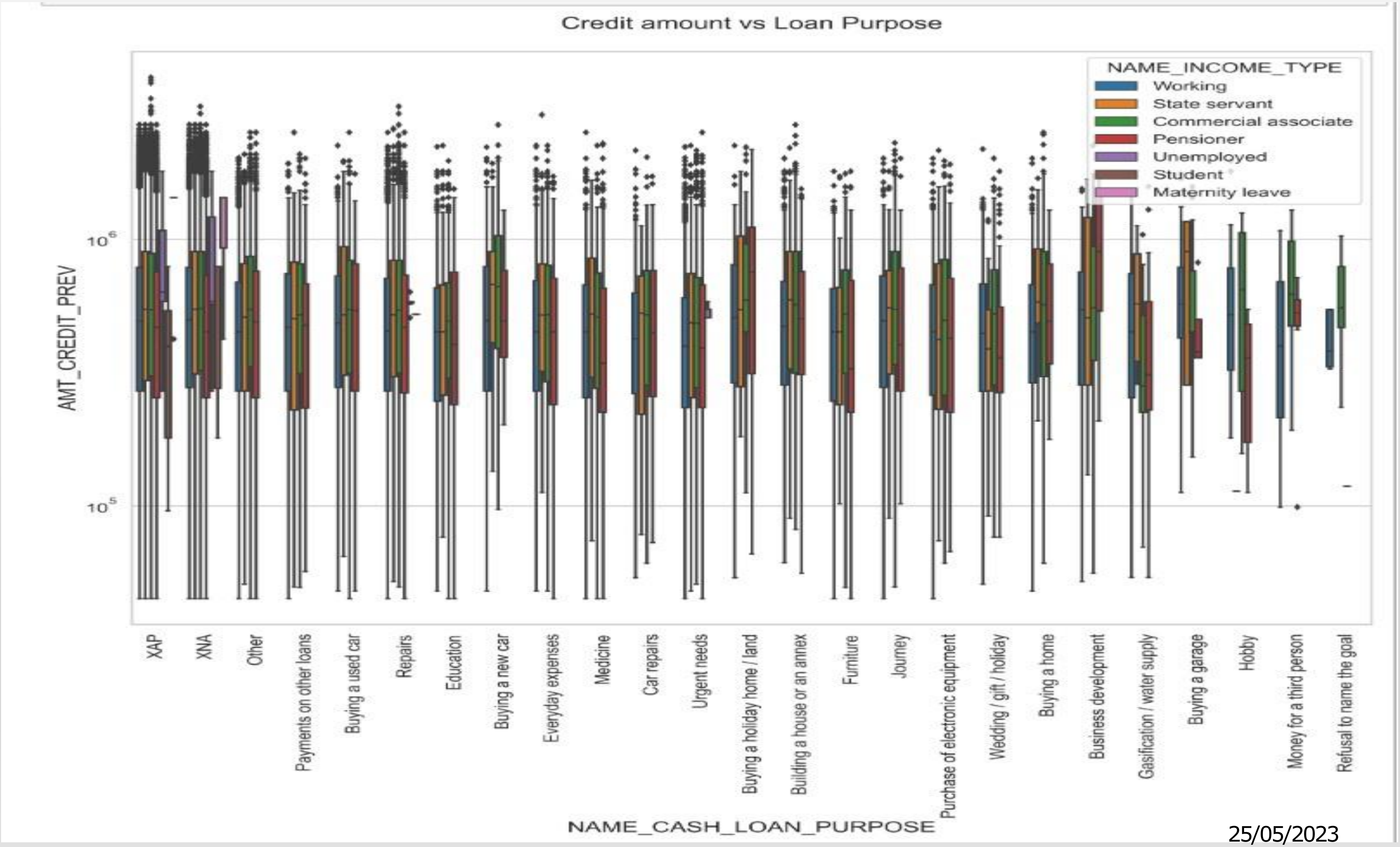
There are few places where loan payment is significant higher than facing difficulties. They are 'Buying a garage', 'Business developemt', 'Buying land','Buying a new car' and 'Education' Hence we can focus on these purposes for which the client is having for minimal payment difficulties




```
In [76]: # Distribution of contract status

sns.set_style('whitegrid')
sns.set_context('talk')

plt.figure(figsize=(10,30),dpi = 300)
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_combine, y= 'NAME_CASH_LOAN_PURPOSE',
                   order=df_combine['NAME_CASH_LOAN_PURPOSE'].value_counts().index,hue = 'TARGET')
```

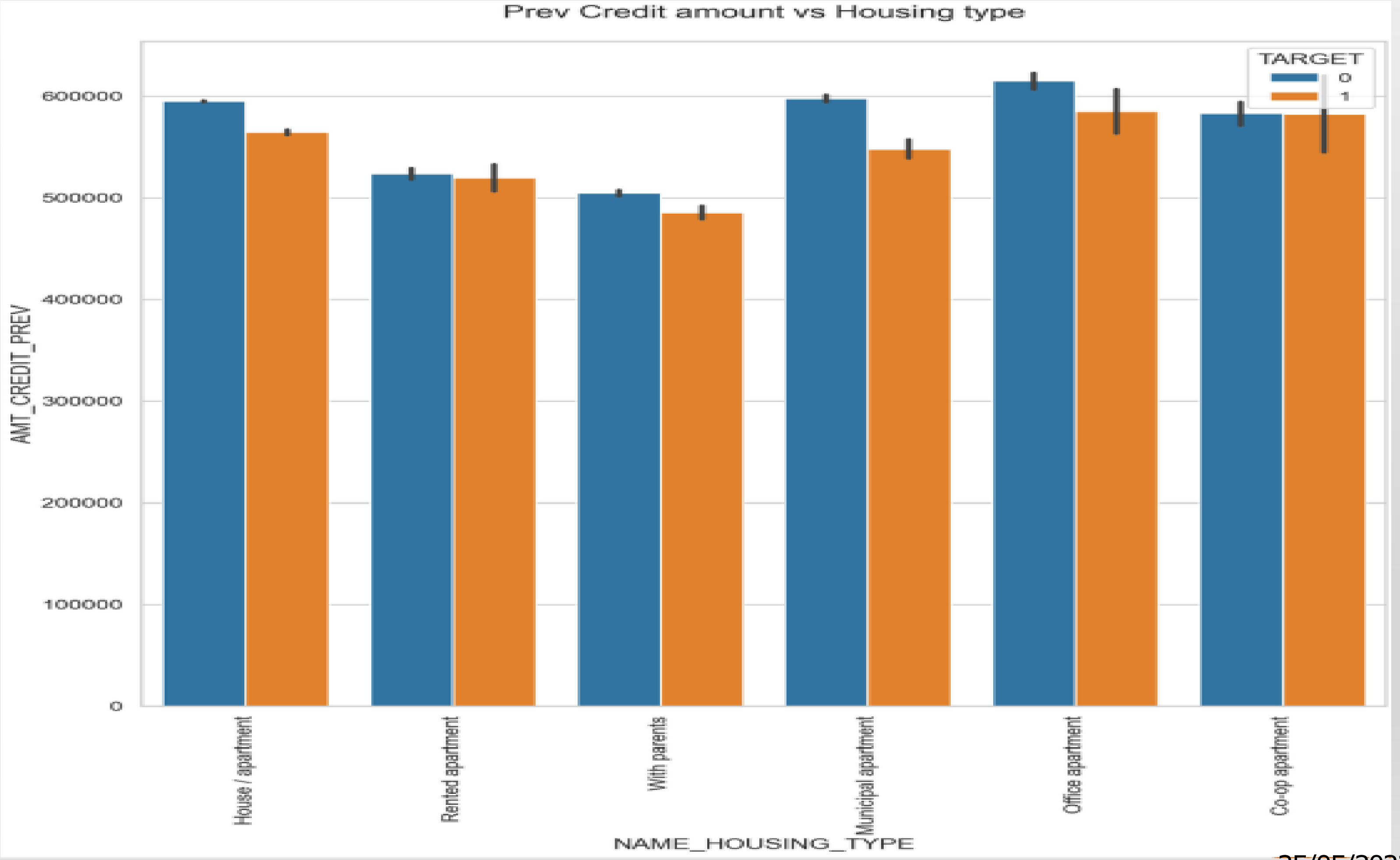
Bivariate Analysis

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



Bank can focus mostly on housing type with parents or House\apartment or municipal apartment for successful payments

```
In [78]: # Box plotting for Credit amount prev vs Housing type in logarithmic scale

         plt.figure(figsize=(15,112),dpi = 120)
         plt.xticks(rotation=90)
         sns.barplot(data =df_combine, y='AMT_CREDIT_PREV',hue='TARGET',x='NAME_HOUSING_TYPE')
         plt.title('Credit amount vs Housing type')
         plt.show()
```



Conclusion

- Banks should reduce their focus on clients who are categorized as " working " since they have the highest rate of unsuccessful payments.
- Banks should avoid granting loans for co-op apartments, as these clients have difficulties making payments on time

Result-

In this case study, I applied the EDA in the real business case scenario.

• I learned basic of risk analytics in banking and financial services and understood how data is used to minimize the risk of losing money while lending to customers.

• This case study helped me in learning how to summarize a huge dataset to gain the valuable insights.

• This project was very challenging. I implemented the study of correlation between different variables to extract the necessary insights for the clients.

• I learned about data imbalance, outliers, driving factors for the datasets.

• It helped me in visualizing the huge dataset and summarizing the most important results helpful to the client

Dataset and Analysis file : link