CREDIT EDA CASE STUDY

Business Objectives

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.

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

1.Importing the Libraries

In [1]:

```
#Ignoring filterwarnings
import warnings

warnings.filterwarnings('ignore')
```

In [2]:

```
#Importing essential libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
#To view large outputs
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

2. Check the structure of data

2.1. Reading the Files

In [4]:

```
A_D = pd.read_csv('application_data.csv')
P_A=pd.read_csv('previous_application.csv')
```

2.2. Examining application data

In [5]:

```
#Checking the shape of the data
A_D.shape
```

Out[5]:

```
(307511, 122)
```

In [6]:
```
# Check the column-wise info of the dataframe
A_D.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 [7]:
```
#Check the summary for the numeric columns
A_D.describe()
```

Out[7]:

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

In [8]:
```
#Check the first five entries of the data
A_D.head(5)
```

Out[8]:

	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	

2.2. Examining previous application data

In [9]:
```
#Checking the shape of the data
P_A.shape
```

Out[9]: (1670214, 37)

In [10]:
```
# Check the column-wise info of the dataframe
P_A.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
```

```
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   SK_ID_PREV                   1670214 non-null  int64  
 1   SK_ID_CURR                   1670214 non-null  int64  
 2   NAME_CONTRACT_TYPE           1670214 non-null  object 
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_DOWN_PAYMENT             774370 non-null   float64
 7   AMT_GOODS_PRICE              1284699 non-null  float64
 8   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object 
 9   HOUR_APPR_PROCESS_START      1670214 non-null  int64  
 10  FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object 
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64  
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object 
 16  NAME_CONTRACT_STATUS         1670214 non-null  object 
 17  DAYS_DECISION                1670214 non-null  int64  
 18  NAME_PAYMENT_TYPE            1670214 non-null  object 
 19  CODE_REJECT_REASON           1670214 non-null  object 
 20  NAME_TYPE_SUITE              849809 non-null   object 
 21  NAME_CLIENT_TYPE             1670214 non-null  object 
 22  NAME_GOODS_CATEGORY          1670214 non-null  object 
 23  NAME_PORTFOLIO               1670214 non-null  object 
 24  NAME_PRODUCT_TYPE            1670214 non-null  object 
 25  CHANNEL_TYPE                 1670214 non-null  object 
 26  SELLERPLACE_AREA             1670214 non-null  int64  
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object 
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object 
 30  PRODUCT_COMBINATION          1669868 non-null  object 
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
```

In [11]:

```
#Check the summary for the numeric columns
P_A.describe()
```

Out[11]:

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

In [12]:

```
#Check the first five entries of the data
P_A.head(5)
```

Out[12]:

	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

3. Data quality check and Missing Values

3.1 Checking missing values in Application data

In [13]:

```
#Checking missing values in Application data
null_1=(100*A_D.isnull().sum()/len(A_D)).round(2)
null_1
```

Out[13]:

```
SK_ID_CURR                      0.00
TARGET                          0.00
NAME_CONTRACT_TYPE              0.00
CODE_GENDER                     0.00
FLAG_OWN_CAR                    0.00
FLAG_OWN_REALTY                 0.00
CNT_CHILDREN                    0.00
AMT_INCOME_TOTAL                0.00
AMT_CREDIT                      0.00
AMT_ANNUITY                     0.00
AMT_GOODS_PRICE                 0.09
NAME_TYPE_SUITE                 0.42
NAME_INCOME_TYPE                0.00
NAME_EDUCATION_TYPE             0.00
NAME_FAMILY_STATUS              0.00
NAME_HOUSING_TYPE               0.00
REGION_POPULATION_RELATIVE      0.00
DAYS_BIRTH                      0.00
DAYS_EMPLOYED                   0.00
DAYS_REGISTRATION               0.00
DAYS_ID_PUBLISH                 0.00
OWN_CAR_AGE                    65.99
FLAG_MOBIL                      0.00
FLAG_EMP_PHONE                  0.00
FLAG_WORK_PHONE                 0.00
FLAG_CONT_MOBILE                0.00
FLAG_PHONE                      0.00
FLAG_EMAIL                      0.00
OCCUPATION_TYPE                31.35
CNT_FAM_MEMBERS                 0.00
REGION_RATING_CLIENT            0.00
REGION_RATING_CLIENT_W_CITY     0.00
WEEKDAY_APPR_PROCESS_START      0.00
HOUR_APPR_PROCESS_START         0.00
REG_REGION_NOT_LIVE_REGION      0.00
REG_REGION_NOT_WORK_REGION      0.00
LIVE_REGION_NOT_WORK_REGION     0.00
REG_CITY_NOT_LIVE_CITY          0.00
REG_CITY_NOT_WORK_CITY          0.00
LIVE_CITY_NOT_WORK_CITY         0.00
```

ORGANIZATION_TYPE	0.00
EXT_SOURCE_1	56.38
EXT_SOURCE_2	0.21
EXT_SOURCE_3	19.83
APARTMENTS_AVG	50.75
BASEMENTAREA_AVG	58.52
YEARS_BEGINEXPLUATATION_AVG	48.78
YEARS_BUILD_AVG	66.50
COMMONAREA_AVG	69.87
ELEVATORS_AVG	53.30
ENTRANCES_AVG	50.35
FLOORSMAX_AVG	49.76
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_BEGINEXPLUATATION_MODE	48.78
YEARS_BUILD_MODE	66.50
COMMONAREA_MODE	69.87
ELEVATORS_MODE	53.30
ENTRANCES_MODE	50.35
FLOORSMAX_MODE	49.76
FLOORSMIN_MODE	67.85
LANDAREA_MODE	59.38
LIVINGAPARTMENTS_MODE	68.35
LIVINGAREA_MODE	50.19
NONLIVINGAPARTMENTS_MODE	69.43
NONLIVINGAREA_MODE	55.18
APARTMENTS_MEDI	50.75
BASEMENTAREA_MEDI	58.52
YEARS_BEGINEXPLUATATION_MEDI	48.78
YEARS_BUILD_MEDI	66.50
COMMONAREA_MEDI	69.87
ELEVATORS_MEDI	53.30
ENTRANCES_MEDI	50.35
FLOORSMAX_MEDI	49.76
FLOORSMIN_MEDI	67.85
LANDAREA_MEDI	59.38
LIVINGAPARTMENTS_MEDI	68.35
LIVINGAREA_MEDI	50.19
NONLIVINGAPARTMENTS_MEDI	69.43
NONLIVINGAREA_MEDI	55.18
FONDKAPREMONT_MODE	68.39
HOUSETYPE_MODE	50.18
TOTALAREA_MODE	48.27
WALLSMATERIAL_MODE	50.84
EMERGENCYSTATE_MODE	47.40
OBS_30_CNT_SOCIAL_CIRCLE	0.33
DEF_30_CNT_SOCIAL_CIRCLE	0.33
OBS_60_CNT_SOCIAL_CIRCLE	0.33
DEF_60_CNT_SOCIAL_CIRCLE	0.33
DAYS_LAST_PHONE_CHANGE	0.00
FLAG_DOCUMENT_2	0.00
FLAG_DOCUMENT_3	0.00
FLAG_DOCUMENT_4	0.00
FLAG_DOCUMENT_5	0.00
FLAG_DOCUMENT_6	0.00
FLAG_DOCUMENT_7	0.00
FLAG_DOCUMENT_8	0.00
FLAG_DOCUMENT_9	0.00
FLAG_DOCUMENT_10	0.00
FLAG_DOCUMENT_11	0.00
FLAG_DOCUMENT_12	0.00
FLAG_DOCUMENT_13	0.00

```
FLAG_DOCUMENT_14                0.00
FLAG_DOCUMENT_15                0.00
FLAG_DOCUMENT_16                0.00
FLAG_DOCUMENT_17                0.00
FLAG_DOCUMENT_18                0.00
FLAG_DOCUMENT_19                0.00
FLAG_DOCUMENT_20                0.00
FLAG_DOCUMENT_21                0.00
AMT_REQ_CREDIT_BUREAU_HOUR     13.50
AMT_REQ_CREDIT_BUREAU_DAY      13.50
AMT_REQ_CREDIT_BUREAU_WEEK     13.50
AMT_REQ_CREDIT_BUREAU_MON      13.50
AMT_REQ_CREDIT_BUREAU_QRT      13.50
AMT_REQ_CREDIT_BUREAU_YEAR     13.50
dtype: float64
```

In [14]:

```
null_1.describe()
```

Out[14]:

```
count    122.000000
mean      24.395902
std       28.446741
min        0.000000
25%        0.000000
50%        0.330000
75%       50.817500
max       69.870000
dtype: float64
```

3.2 Cleaning in Application data

In [15]:

```
#Cleaning the rows which have null value percentage Standard Deviation
A_D=A_D.loc[:, null_1<=29]
```

In [16]:

```
#Checking null value percent for the cleaned data
(100*A_D.isnull().sum()/len(A_D)).round(2)
```

Out[16]:

```
SK_ID_CURR                     0.00
TARGET                         0.00
NAME_CONTRACT_TYPE             0.00
CODE_GENDER                    0.00
FLAG_OWN_CAR                   0.00
FLAG_OWN_REALTY                0.00
CNT_CHILDREN                   0.00
AMT_INCOME_TOTAL               0.00
AMT_CREDIT                     0.00
AMT_ANNUITY                    0.00
AMT_GOODS_PRICE                0.09
NAME_TYPE_SUITE                0.42
NAME_INCOME_TYPE               0.00
NAME_EDUCATION_TYPE            0.00
NAME_FAMILY_STATUS             0.00
NAME_HOUSING_TYPE              0.00
REGION_POPULATION_RELATIVE     0.00
DAYS_BIRTH                     0.00
DAYS_EMPLOYED                  0.00
DAYS_REGISTRATION              0.00
DAYS_ID_PUBLISH                0.00
FLAG_MOBIL                     0.00
FLAG_EMP_PHONE                 0.00
FLAG_WORK_PHONE                0.00
FLAG_CONT_MOBILE               0.00
FLAG_PHONE                     0.00
FLAG_EMAIL                     0.00
```

```
CNT_FAM_MEMBERS                  0.00
REGION_RATING_CLIENT             0.00
REGION_RATING_CLIENT_W_CITY      0.00
WEEKDAY_APPR_PROCESS_START       0.00
HOUR_APPR_PROCESS_START          0.00
REG_REGION_NOT_LIVE_REGION       0.00
REG_REGION_NOT_WORK_REGION       0.00
LIVE_REGION_NOT_WORK_REGION      0.00
REG_CITY_NOT_LIVE_CITY           0.00
REG_CITY_NOT_WORK_CITY           0.00
LIVE_CITY_NOT_WORK_CITY          0.00
ORGANIZATION_TYPE                0.00
EXT_SOURCE_2                     0.21
EXT_SOURCE_3                    19.83
OBS_30_CNT_SOCIAL_CIRCLE         0.33
DEF_30_CNT_SOCIAL_CIRCLE         0.33
OBS_60_CNT_SOCIAL_CIRCLE         0.33
DEF_60_CNT_SOCIAL_CIRCLE         0.33
DAYS_LAST_PHONE_CHANGE           0.00
FLAG_DOCUMENT_2                  0.00
FLAG_DOCUMENT_3                  0.00
FLAG_DOCUMENT_4                  0.00
FLAG_DOCUMENT_5                  0.00
FLAG_DOCUMENT_6                  0.00
FLAG_DOCUMENT_7                  0.00
FLAG_DOCUMENT_8                  0.00
FLAG_DOCUMENT_9                  0.00
FLAG_DOCUMENT_10                 0.00
FLAG_DOCUMENT_11                 0.00
FLAG_DOCUMENT_12                 0.00
FLAG_DOCUMENT_13                 0.00
FLAG_DOCUMENT_14                 0.00
FLAG_DOCUMENT_15                 0.00
FLAG_DOCUMENT_16                 0.00
FLAG_DOCUMENT_17                 0.00
FLAG_DOCUMENT_18                 0.00
FLAG_DOCUMENT_19                 0.00
FLAG_DOCUMENT_20                 0.00
FLAG_DOCUMENT_21                 0.00
AMT_REQ_CREDIT_BUREAU_HOUR      13.50
AMT_REQ_CREDIT_BUREAU_DAY       13.50
AMT_REQ_CREDIT_BUREAU_WEEK      13.50
AMT_REQ_CREDIT_BUREAU_MON       13.50
AMT_REQ_CREDIT_BUREAU_QRT       13.50
AMT_REQ_CREDIT_BUREAU_YEAR      13.50
dtype: float64
```

In [17]:

```
#Shape after cleaning
A_D.shape
```

Out[17]:

```
(307511, 72)
```

3.3 Checking missing values in Previous Application data

In [18]:

```
#Checking the missing values in previous application data.
null_2=(100*P_A.isnull().sum()/len(P_A)).round(2)
null_2
```

Out[18]:

```
SK_ID_PREV                       0.00
SK_ID_CURR                       0.00
NAME_CONTRACT_TYPE               0.00
AMT_ANNUITY                     22.29
AMT_APPLICATION                  0.00
AMT_CREDIT                       0.00
```

```
AMT_DOWN_PAYMENT                53.64
AMT_GOODS_PRICE                 23.08
WEEKDAY_APPR_PROCESS_START       0.00
HOUR_APPR_PROCESS_START          0.00
FLAG_LAST_APPL_PER_CONTRACT      0.00
NFLAG_LAST_APPL_IN_DAY           0.00
RATE_DOWN_PAYMENT               53.64
RATE_INTEREST_PRIMARY           99.64
RATE_INTEREST_PRIVILEGED        99.64
NAME_CASH_LOAN_PURPOSE           0.00
NAME_CONTRACT_STATUS             0.00
DAYS_DECISION                    0.00
NAME_PAYMENT_TYPE                0.00
CODE_REJECT_REASON               0.00
NAME_TYPE_SUITE                 49.12
NAME_CLIENT_TYPE                 0.00
NAME_GOODS_CATEGORY              0.00
NAME_PORTFOLIO                   0.00
NAME_PRODUCT_TYPE                0.00
CHANNEL_TYPE                     0.00
SELLERPLACE_AREA                 0.00
NAME_SELLER_INDUSTRY             0.00
CNT_PAYMENT                     22.29
NAME_YIELD_GROUP                 0.00
PRODUCT_COMBINATION              0.02
DAYS_FIRST_DRAWING              40.30
DAYS_FIRST_DUE                  40.30
DAYS_LAST_DUE_1ST_VERSION       40.30
DAYS_LAST_DUE                   40.30
DAYS_TERMINATION                40.30
NFLAG_INSURED_ON_APPROVAL       40.30
dtype: float64
```

In [19]:

```
null_2.describe()
```

Out[19]:

```
count    37.000000
mean     17.977297
std      27.556341
min       0.000000
25%       0.000000
50%       0.000000
75%      40.300000
max      99.640000
dtype: float64
```

3.4 Cleaning in Previous Application data

In [20]:

```
#Cleaning the rows which have null value percentage Standard Deviation
P_A=P_A.loc[:, null_2<=28]
```

In [21]:

```
#Checking null value percent for the cleaned data
(100*P_A.isnull().sum()/len(P_A)).round(2)
```

Out[21]:

```
SK_ID_PREV                       0.00
SK_ID_CURR                       0.00
NAME_CONTRACT_TYPE               0.00
AMT_ANNUITY                     22.29
AMT_APPLICATION                  0.00
AMT_CREDIT                       0.00
AMT_GOODS_PRICE                 23.08
WEEKDAY_APPR_PROCESS_START       0.00
HOUR_APPR_PROCESS_START          0.00
FLAG_LAST_APPL_PER_CONTRACT      0.00
```

```
NFLAG_LAST_APPL_IN_DAY          0.00
NAME_CASH_LOAN_PURPOSE          0.00
NAME_CONTRACT_STATUS            0.00
DAYS_DECISION                   0.00
NAME_PAYMENT_TYPE               0.00
CODE_REJECT_REASON              0.00
NAME_CLIENT_TYPE                0.00
NAME_GOODS_CATEGORY             0.00
NAME_PORTFOLIO                  0.00
NAME_PRODUCT_TYPE               0.00
CHANNEL_TYPE                    0.00
SELLERPLACE_AREA                0.00
NAME_SELLER_INDUSTRY            0.00
CNT_PAYMENT                    22.29
NAME_YIELD_GROUP                0.00
PRODUCT_COMBINATION             0.02
dtype: float64
```

In [22]:

```
#Shape after cleaning
P_A.shape
```

Out[22]:

```
(1670214, 26)
```

4.Checking the data-types of the columns in Application Data

In [23]:

```
A_D.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 72 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                  307499 non-null  float64
 10  AMT_GOODS_PRICE              307233 non-null  float64
 11  NAME_TYPE_SUITE              306219 non-null  object 
 12  NAME_INCOME_TYPE             307511 non-null  object 
 13  NAME_EDUCATION_TYPE          307511 non-null  object 
 14  NAME_FAMILY_STATUS           307511 non-null  object 
 15  NAME_HOUSING_TYPE            307511 non-null  object 
 16  REGION_POPULATION_RELATIVE   307511 non-null  float64
 17  DAYS_BIRTH                   307511 non-null  int64  
 18  DAYS_EMPLOYED                307511 non-null  int64  
 19  DAYS_REGISTRATION            307511 non-null  float64
 20  DAYS_ID_PUBLISH              307511 non-null  int64  
 21  FLAG_MOBIL                   307511 non-null  int64  
 22  FLAG_EMP_PHONE               307511 non-null  int64  
 23  FLAG_WORK_PHONE              307511 non-null  int64  
 24  FLAG_CONT_MOBILE             307511 non-null  int64  
 25  FLAG_PHONE                   307511 non-null  int64  
 26  FLAG_EMAIL                   307511 non-null  int64  
 27  CNT_FAM_MEMBERS              307509 non-null  float64
 28  REGION_RATING_CLIENT         307511 non-null  int64  
```

```
 29  REGION_RATING_CLIENT_W_CITY  307511 non-null  int64
 30  WEEKDAY_APPR_PROCESS_START   307511 non-null  object
 31  HOUR_APPR_PROCESS_START      307511 non-null  int64
 32  REG_REGION_NOT_LIVE_REGION   307511 non-null  int64
 33  REG_REGION_NOT_WORK_REGION   307511 non-null  int64
 34  LIVE_REGION_NOT_WORK_REGION  307511 non-null  int64
 35  REG_CITY_NOT_LIVE_CITY       307511 non-null  int64
 36  REG_CITY_NOT_WORK_CITY       307511 non-null  int64
 37  LIVE_CITY_NOT_WORK_CITY      307511 non-null  int64
 38  ORGANIZATION_TYPE            307511 non-null  object
 39  EXT_SOURCE_2                 306851 non-null  float64
 40  EXT_SOURCE_3                 246546 non-null  float64
 41  OBS_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 42  DEF_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 43  OBS_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 44  DEF_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 45  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64
 46  FLAG_DOCUMENT_2              307511 non-null  int64
 47  FLAG_DOCUMENT_3              307511 non-null  int64
 48  FLAG_DOCUMENT_4              307511 non-null  int64
 49  FLAG_DOCUMENT_5              307511 non-null  int64
 50  FLAG_DOCUMENT_6              307511 non-null  int64
 51  FLAG_DOCUMENT_7              307511 non-null  int64
 52  FLAG_DOCUMENT_8              307511 non-null  int64
 53  FLAG_DOCUMENT_9              307511 non-null  int64
 54  FLAG_DOCUMENT_10             307511 non-null  int64
 55  FLAG_DOCUMENT_11             307511 non-null  int64
 56  FLAG_DOCUMENT_12             307511 non-null  int64
 57  FLAG_DOCUMENT_13             307511 non-null  int64
 58  FLAG_DOCUMENT_14             307511 non-null  int64
 59  FLAG_DOCUMENT_15             307511 non-null  int64
 60  FLAG_DOCUMENT_16             307511 non-null  int64
 61  FLAG_DOCUMENT_17             307511 non-null  int64
 62  FLAG_DOCUMENT_18             307511 non-null  int64
 63  FLAG_DOCUMENT_19             307511 non-null  int64
 64  FLAG_DOCUMENT_20             307511 non-null  int64
 65  FLAG_DOCUMENT_21             307511 non-null  int64
 66  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64
 67  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64
 68  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64
 69  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64
 70  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64
 71  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64
dtypes: float64(20), int64(41), object(11)
memory usage: 168.9+ MB
```

In [24]:

```
# Converting the data types of some of the columns that shouldn't be flo
A_D['DAYS_REGISTRATION'] = A_D['DAYS_REGISTRATION'].astype(int,errors='i
A_D['CNT_FAM_MEMBERS'] = A_D['CNT_FAM_MEMBERS'].astype(int,errors='ignore
A_D['OBS_30_CNT_SOCIAL_CIRCLE'] = A_D['OBS_30_CNT_SOCIAL_CIRCLE'].astype
A_D['DEF_30_CNT_SOCIAL_CIRCLE'] = A_D['DEF_30_CNT_SOCIAL_CIRCLE'].astype
A_D['OBS_60_CNT_SOCIAL_CIRCLE'] = A_D['OBS_60_CNT_SOCIAL_CIRCLE'].astype
A_D['DEF_60_CNT_SOCIAL_CIRCLE'] = A_D['DEF_60_CNT_SOCIAL_CIRCLE'].astype
A_D['AMT_REQ_CREDIT_BUREAU_HOUR'] = A_D['AMT_REQ_CREDIT_BUREAU_HOUR'].as
A_D['AMT_REQ_CREDIT_BUREAU_DAY'] = A_D['AMT_REQ_CREDIT_BUREAU_DAY'].asty
A_D['AMT_REQ_CREDIT_BUREAU_WEEK'] = A_D['AMT_REQ_CREDIT_BUREAU_WEEK'].as
A_D['AMT_REQ_CREDIT_BUREAU_MON'] = A_D['AMT_REQ_CREDIT_BUREAU_MON'].asty
A_D['AMT_REQ_CREDIT_BUREAU_QRT'] = A_D['AMT_REQ_CREDIT_BUREAU_QRT'].asty
A_D['AMT_REQ_CREDIT_BUREAU_YEAR'] = A_D['AMT_REQ_CREDIT_BUREAU_YEAR'].as
```

5.Removing the unwanted columns from the application_dataset

In [25]:
```
# We will remove the unwanted columns from the application_dataset

unwanted = ['FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CON
       'FLAG_PHONE', 'FLAG_EMAIL','REGION_RATING_CLIENT','REGION_RATING_
       'REGION_RATING_CLIENT_W_CITY','DAYS_LAST_PHONE_CHANGE', 'FLAG_DOC
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9','FLAG_DOC
       'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15','FLAG_
       'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21']

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

6.Checking the Gender and Organization column for any error

In [26]:
```
# Checking the Gender column first,

A_D.CODE_GENDER.value_counts()
```

Out[26]:
```
F      202448
M      105059
XNA         4
Name: CODE_GENDER, dtype: int64
```

In [27]:
```
# Replacing the 'XNA' values with the Females as majority is Females & i

A_D.CODE_GENDER.replace(to_replace = 'XNA', value = 'F', inplace = True)
```

In [28]:
```
# Confirmation of changes in the Gender column
A_D.CODE_GENDER.value_counts()
```

Out[28]:
```
F    202452
M    105059
Name: CODE_GENDER, dtype: int64
```

In [29]:
```
# Checking the Organization column second,

A_D.ORGANIZATION_TYPE.value_counts()
```

Out[29]:
```
Business Entity Type 3    67992
XNA                       55374
Self-employed             38412
Other                     16683
Medicine                  11193
Business Entity Type 2    10553
Government                10404
School                     8893
Trade: type 7              7831
Kindergarten               6880
Construction               6721
Business Entity Type 1     5984
Transport: type 4          5398
Trade: type 3              3492
Industry: type 9           3368
Industry: type 3           3278
Security                   3247
Housing                    2958
Industry: type 11          2704
Military                   2634
Bank                       2507
```

```
Agriculture                 2454
Police                      2341
Transport: type 2           2204
Postal                      2157
Security Ministries         1974
Trade: type 2               1900
Restaurant                  1811
Services                    1575
University                  1327
Industry: type 7            1307
Transport: type 3           1187
Industry: type 1            1039
Hotel                        966
Electricity                  950
Industry: type 4             877
Trade: type 6                631
Industry: type 5             599
Insurance                    597
Telecom                      577
Emergency                    560
Industry: type 2             458
Advertising                  429
Realtor                      396
Culture                      379
Industry: type 12            369
Trade: type 1                348
Mobile                       317
Legal Services               305
Cleaning                     260
Transport: type 1            201
Industry: type 6             112
Industry: type 10            109
Religion                      85
Industry: type 13             67
Trade: type 4                 64
Trade: type 5                 49
Industry: type 8              24
Name: ORGANIZATION_TYPE, dtype: int64
```

7.Creating bins for the 'AMT_INCOME_TOTAL' and 'AMT_CREDIT'

In [30]:

```
# Creating bins for the Income column i.e. "AMT_INCOME_TOTAL"

bins_income = [0,25000,50000,75000,100000,125000,150000,175000,200000,22

slot_income = ['0-25000', '25000-50000','50000-75000','75000,100000','10
       '200000-225000','225000-250000','250000-275000','275000-300000','
       '375000-400000','400000-425000','425000-450000','450000-475000','

A_D['AMT_INCOME_RANGE'] = pd.cut(A_D['AMT_INCOME_TOTAL'], bins_income, l
```

In [31]:

```
# Creating bins for the Credit column i.e. "AMT_CREDIT"

bins_credit = [0,150000,200000,250000,300000,350000,400000,450000,500000

slots_credit = ['0-150000', '150000-200000','200000-250000', '250000-300
        '450000-500000','500000-550000','550000-600000','600000-650000',
        '800000-850000','850000-900000','900000 and above']

A_D['AMT_CREDIT_RANGE']=pd.cut(A_D['AMT_CREDIT'], bins_credit, labels=sl
```

8.Checking for the Imbalance Ratio

In [32]:

```
# Have a look into the TARGET data,

A_D.TARGET.value_counts()
```

Out[32]:

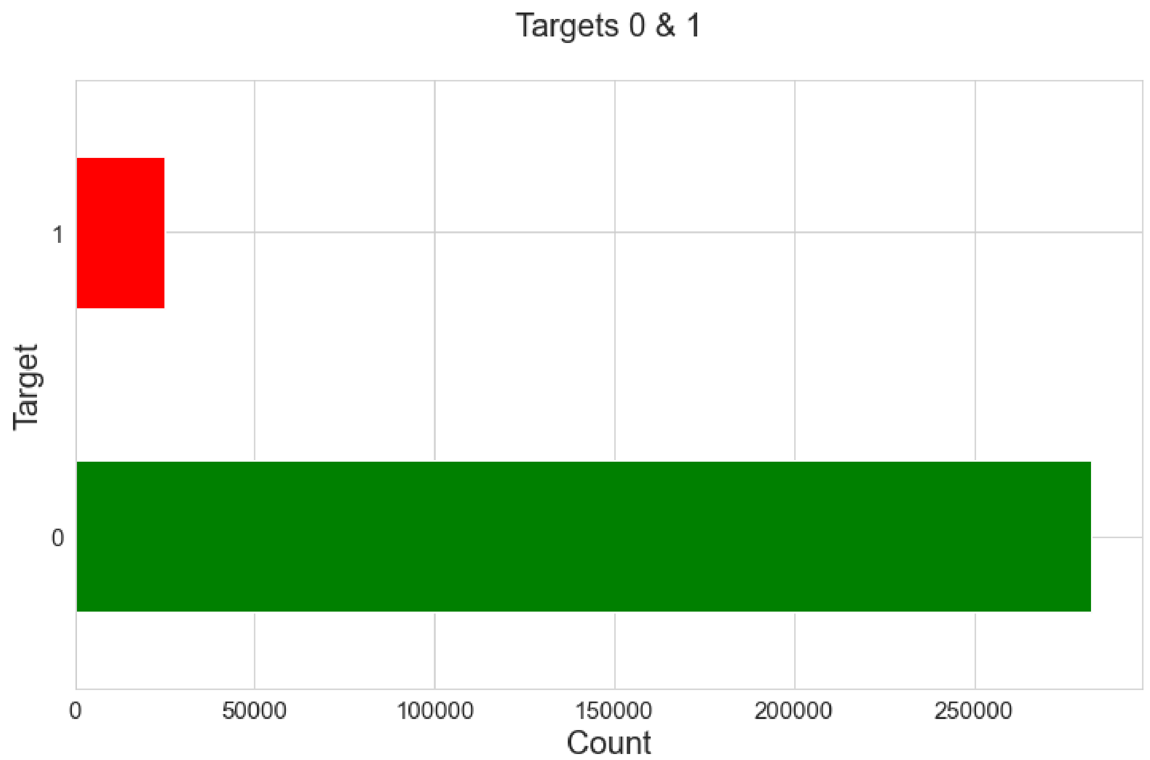
```
0    282686
1     24825
Name: TARGET, dtype: int64
```

Here, 'Target = 0' means the people those who are non-defaulters.

And, 'Target=1' means the people those who are defaulters.

In [72]:

```
# Plotting for the Targets,

plt.figure(figsize=[12,7])

A_D.TARGET.value_counts().plot.barh(color=['Green','Red'])

plt.title('Targets 0 & 1\n', fontsize=20)
plt.xlabel('Count', fontsize=20)
plt.ylabel('Target', fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)

plt.show()
```



In [34]:

```
# Checking the imbalance ratio for the Target column

target_0 = A_D.loc[A_D["TARGET"]==0]
target_1 = A_D.loc[A_D["TARGET"]==1]

round(len(target_0)/len(target_1),2)
```

```
11.39
```

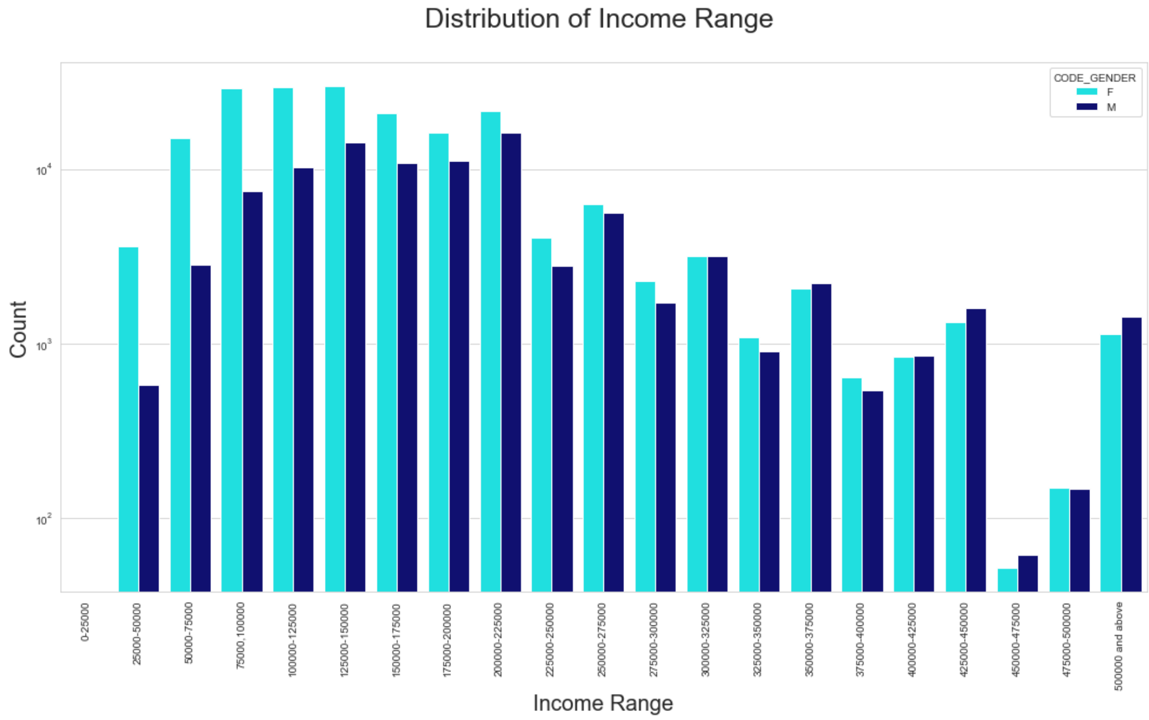
Out[34]:

10.Univariate Analysis

Plotting a bar chart for those having no difficulties in re-paying the loan i.e. the Target = 0 people.

In [73]:

```
# Plotting for Income Range across various Gender.

plt.figure(figsize=[18,9])
sns.set_style('whitegrid')

sns.countplot(data=target_0, x='AMT_INCOME_RANGE', hue='CODE_GENDER', pa

plt.xticks(rotation=90)
plt.title('Distribution of Income Range \n', fontsize=25)
plt.xlabel('Income Range', fontsize=20)
plt.ylabel('Count', fontsize=20)
plt.yscale('log')

plt.show()
```



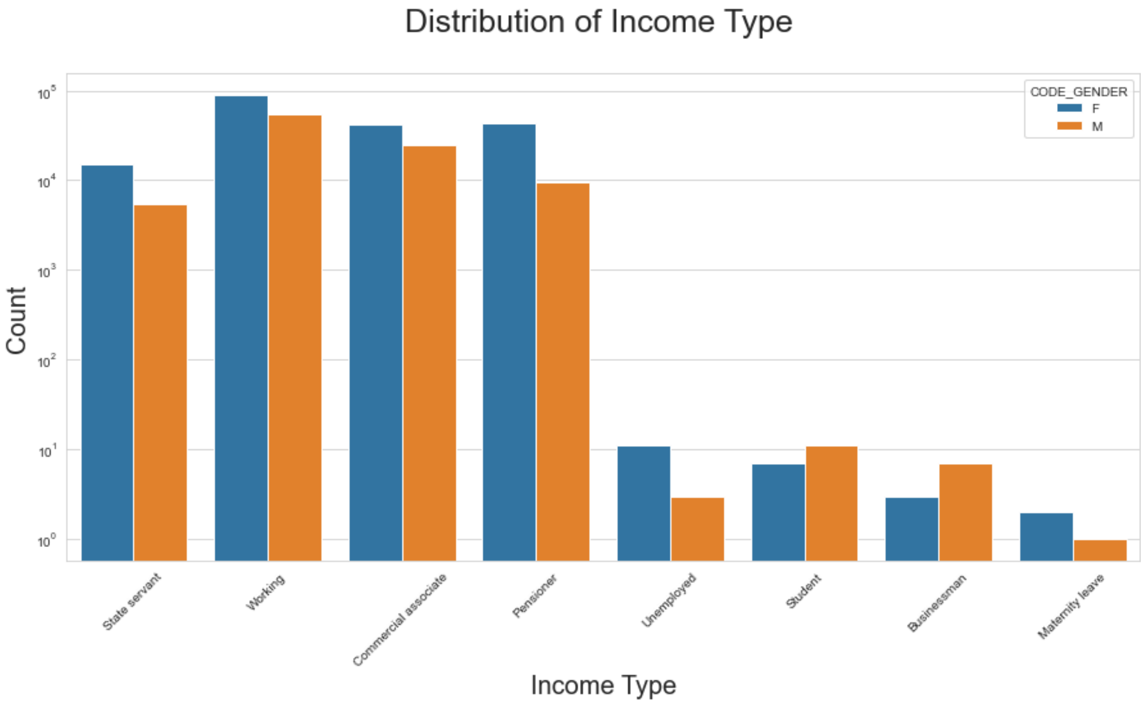
Conclusion from the graph:

> 1. Income range from 125000 to 150000 is having more number of credits.
> 2. Very less count from range 450000-475000.
> 3. It seems that the females are more than male in having credit.

In [36]:

```
# Plotting for the various Income types across various Gender.

plt.figure(figsize=[15,7])
sns.set_style('whitegrid')

sns.countplot(data=target_0, x='NAME_INCOME_TYPE', hue='CODE_GENDER')
```

```
plt.xticks(rotation=45)
plt.title('Distribution of Income Type \n', fontsize=25)
plt.xlabel('Income Type', fontsize=20)
plt.ylabel('Count', fontsize=20)
plt.yscale('log')

plt.show()
```

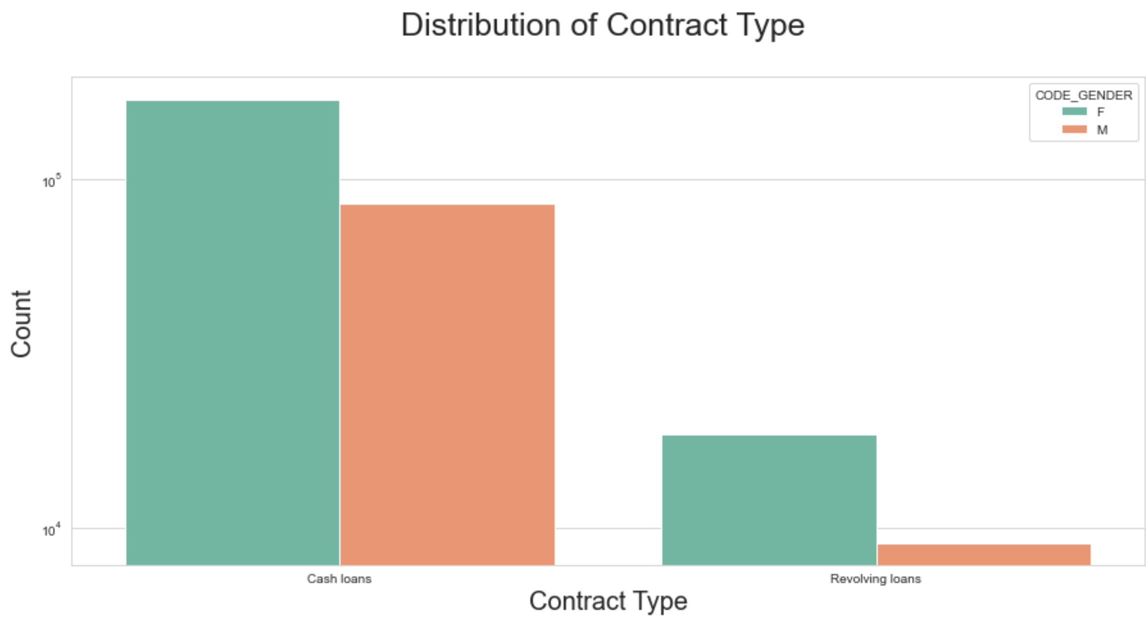


Conclusion from the graph:

> 1. It seems that working women have most credit than others.
> 2. It seems that 'State Servant','Working' and 'Commercial Associate'have more credit counts compared to others.
> 3. It seems Women in 'Maternity leave' has less credit in comparison to others.

In [37]:

```
# Plotting for the Contract type across various Genders.

plt.figure(figsize=[15,7])
sns.set_style('whitegrid')

sns.countplot(data=target_0, x='NAME_CONTRACT_TYPE', hue='CODE_GENDER', ]

plt.xticks
plt.title('Distribution of Contract Type \n', fontsize=25)
plt.xlabel('Contract Type', fontsize=20)
plt.ylabel('Count', fontsize=20)
plt.yscale('log')

plt.show()
```

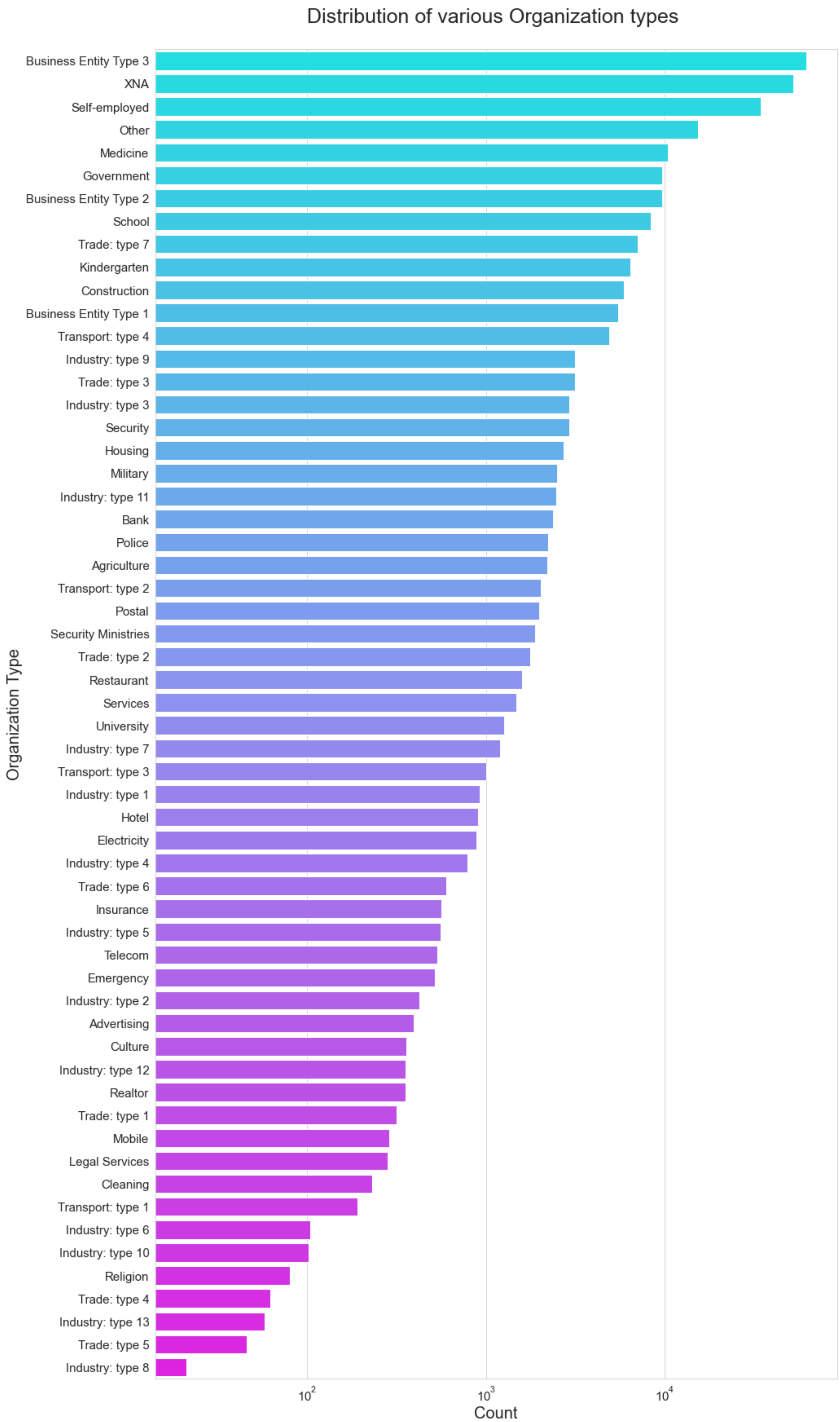


Conclusion from the graph:

> 1. It seems that cash loans' is having higher number of credits than 'Revolving loans' contract type.
> 2. Also, female applies more for Credit.

In [38]:

```
# Plotting for the various Organization Types

plt.figure(figsize=[15, 30])

sns.countplot(data=target_0, y='ORGANIZATION_TYPE', order=target_0['ORGA

plt.title("Distribution of various Organization types \n", fontsize=25)
plt.xticks( fontsize=15)
plt.yticks(fontsize=15)
plt.xscale('log')
plt.xlabel('Count', fontsize=20)
plt.ylabel('Organization Type', fontsize=20)

plt.show()
```



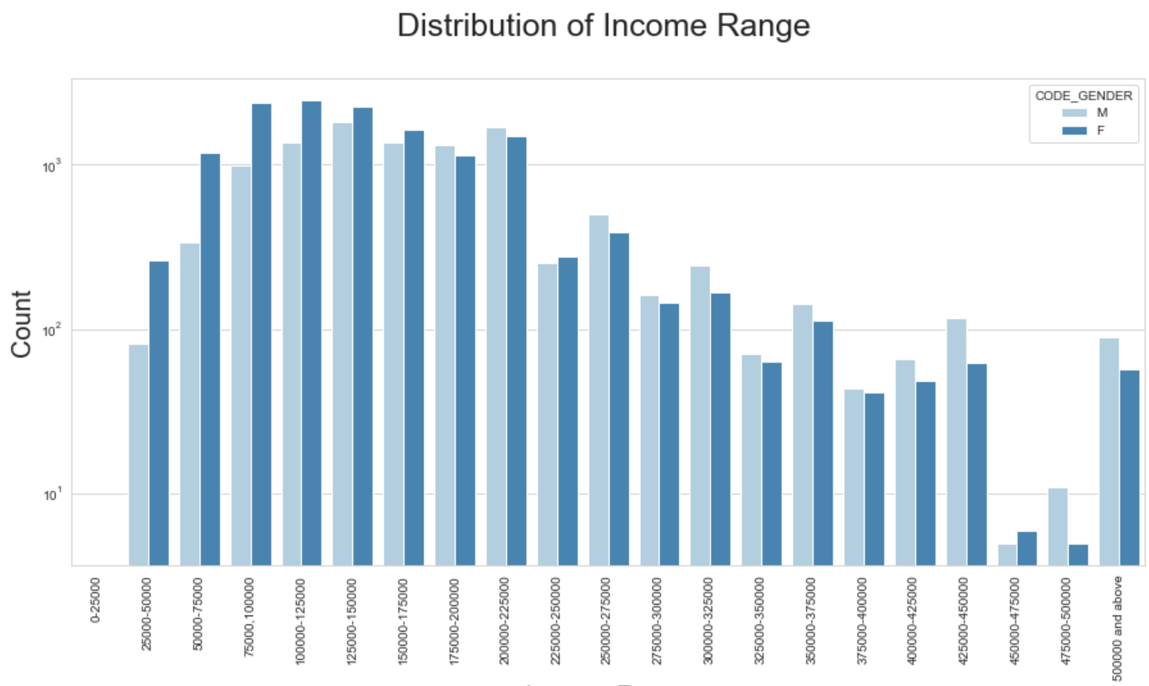
Conclusions from the graph:

> 1. Clients which have applied for credits are from most of the organization type ‘Business entity Type 3’ , ‘Self employed’, ‘Other’ , ‘Medicine’ and ‘Government’.
> 2. Less clients are from Industry type 8,type 6, type 10, religion and trade type 5, type 4.

Plotting for those having difficulty in re-paying the loan i.e. Target = 1 people.

In [39]:

```
# Plotting for Income Range across various Gender.

plt.figure(figsize=[15,7])
sns.set_style('whitegrid')

sns.countplot(data=target_1, x='AMT_INCOME_RANGE', hue='CODE_GENDER', pa

plt.xticks(rotation=90)
plt.title('Distribution of Income Range \n', fontsize=25)
plt.xlabel('Income Range', fontsize=20)
plt.ylabel('Count', fontsize=20)
plt.yscale('log')

plt.show()
```



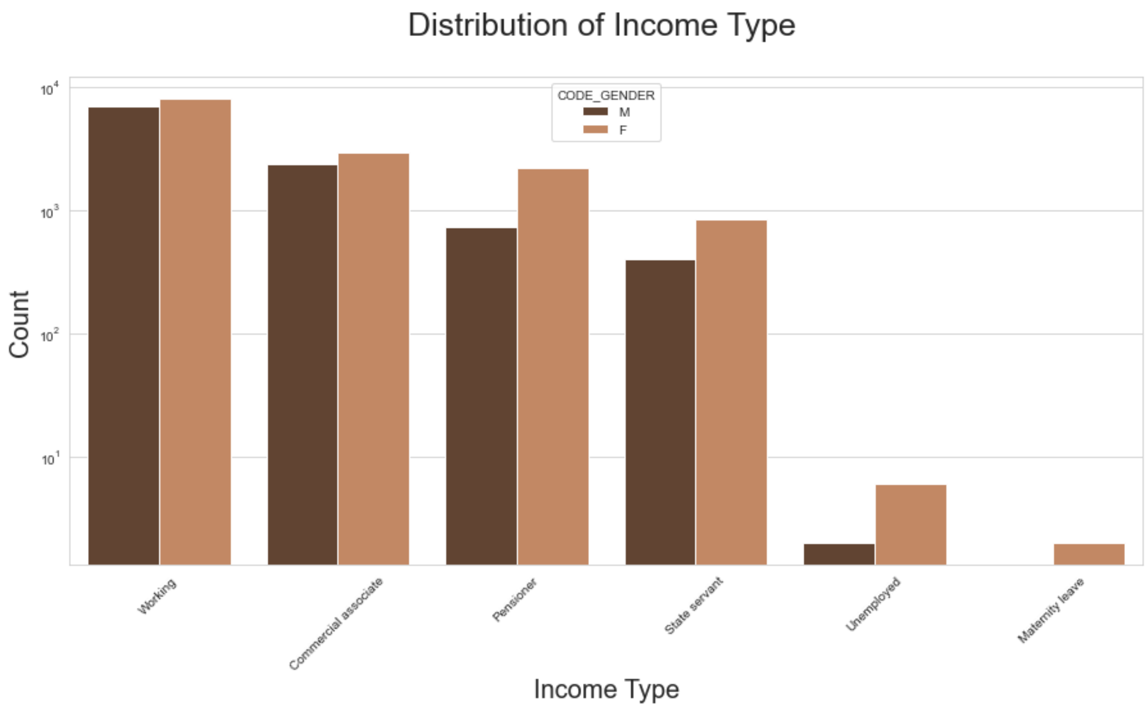
Conclusions from the graph:

> 1. Male Counts are higher.
> 2. Income range from 100000 to 200000 is having more number of credits.
> 3. Less count for income range 450000-475000.

In [40]:

```
# Plotting for the various Income types across various Gender.

plt.figure(figsize=[15,7])
```

```
sns.set_style('whitegrid')

sns.countplot(data=target_1, x='NAME_INCOME_TYPE', hue='CODE_GENDER', pa

plt.xticks(rotation=45)
plt.title('Distribution of Income Type \n', fontsize=25)
plt.xlabel('Income Type', fontsize=20)
plt.ylabel('Count', fontsize=20)
plt.yscale('log')

plt.show()
```

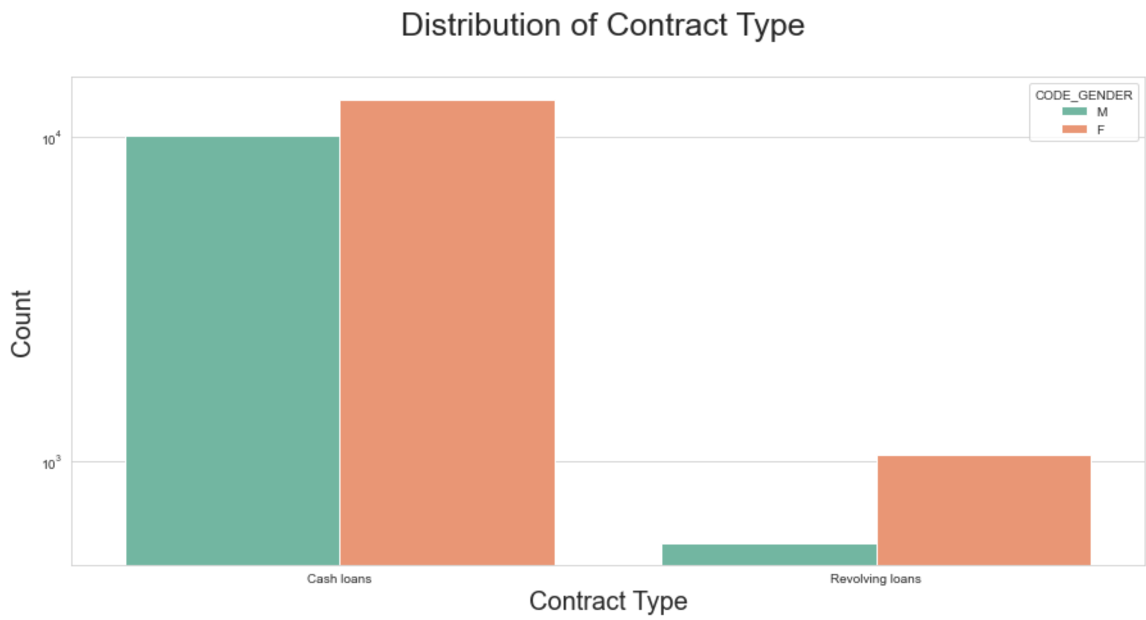


Conclusions from the graph:

> 1. For income type ‘working’, ’commercial associate’, and ‘State Servant’ the number of credits are higher than other i.e. ‘Maternity leave.
> 2. For this Females are having more number of credits than male.
> 3. Less number of credits for income type ‘Maternity leave’.

In [41]:

```
# Plotting for the Contract type across various Genders.

plt.figure(figsize=[15,7])
sns.set_style('whitegrid')

sns.countplot(data=target_1, x='NAME_CONTRACT_TYPE', hue='CODE_GENDER', ]


plt.title('Distribution of Contract Type \n', fontsize=25)
plt.xlabel('Contract Type', fontsize=20)
plt.ylabel('Count', fontsize=20)
plt.yscale('log')

plt.show()
```

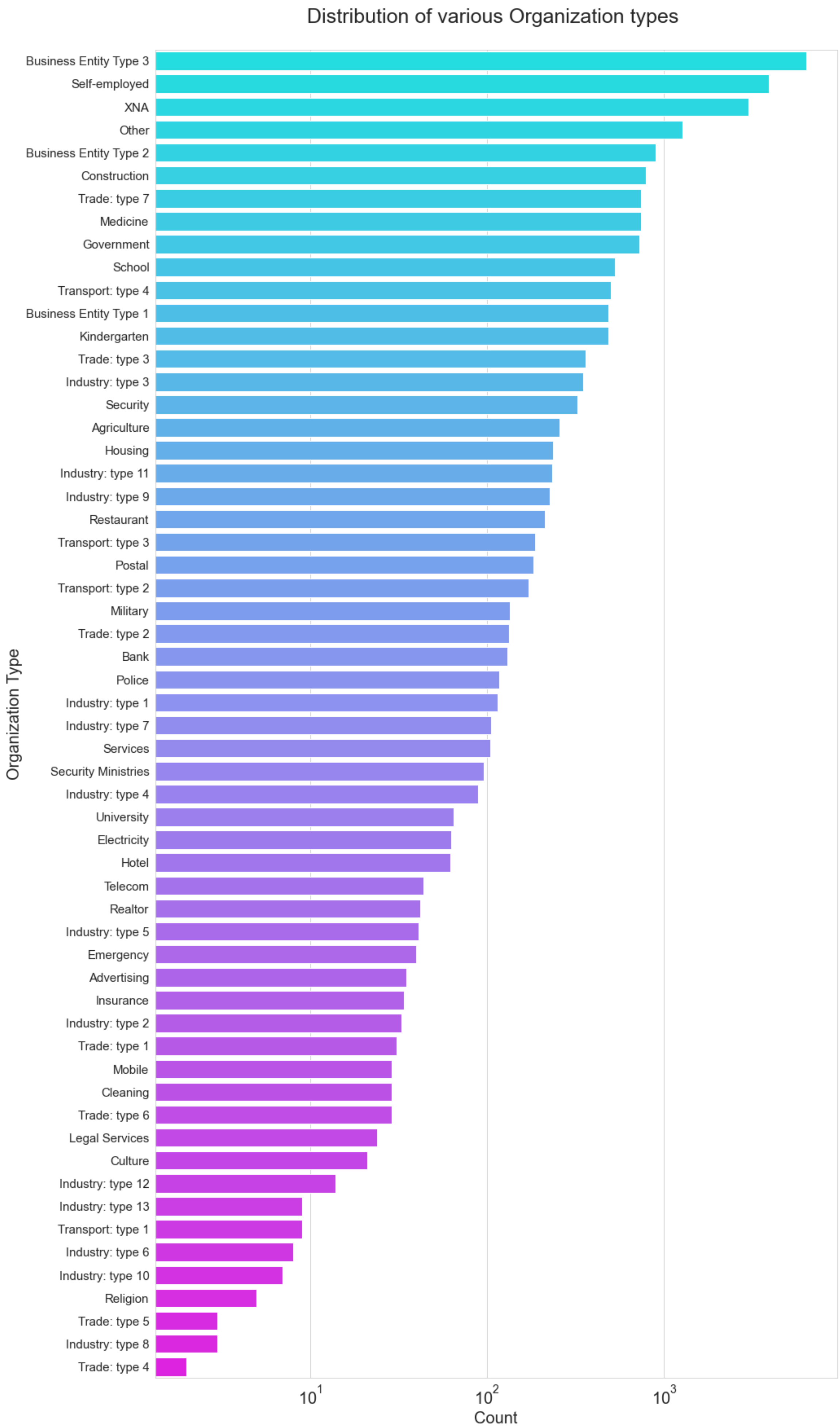


Conclusions from the graph:

> 1. For contract type ‘cash loans’ is having higher number of credits than ‘Revolving loans’ contract type.
> 2. For this also Female is leading for applying credits.

In [42]:

```
# Plotting for the various Organization Types

plt.figure(figsize=[15, 30])

sns.countplot(data=target_1, y='ORGANIZATION_TYPE', order=target_1['ORGA

plt.title("Distribution of various Organization types \n", fontsize=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=15)
plt.xscale('log')
plt.xlabel('Count', fontsize=20)
plt.ylabel('Organization Type', fontsize=20)

plt.show()
```



Conclusions from the graph:

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

Defining the Correlation

In [43]:

```
# Calculating the correlation among the target_0 people

target_0_corr = target_0.iloc[0:, 2:].corr()
target_0_corr
```

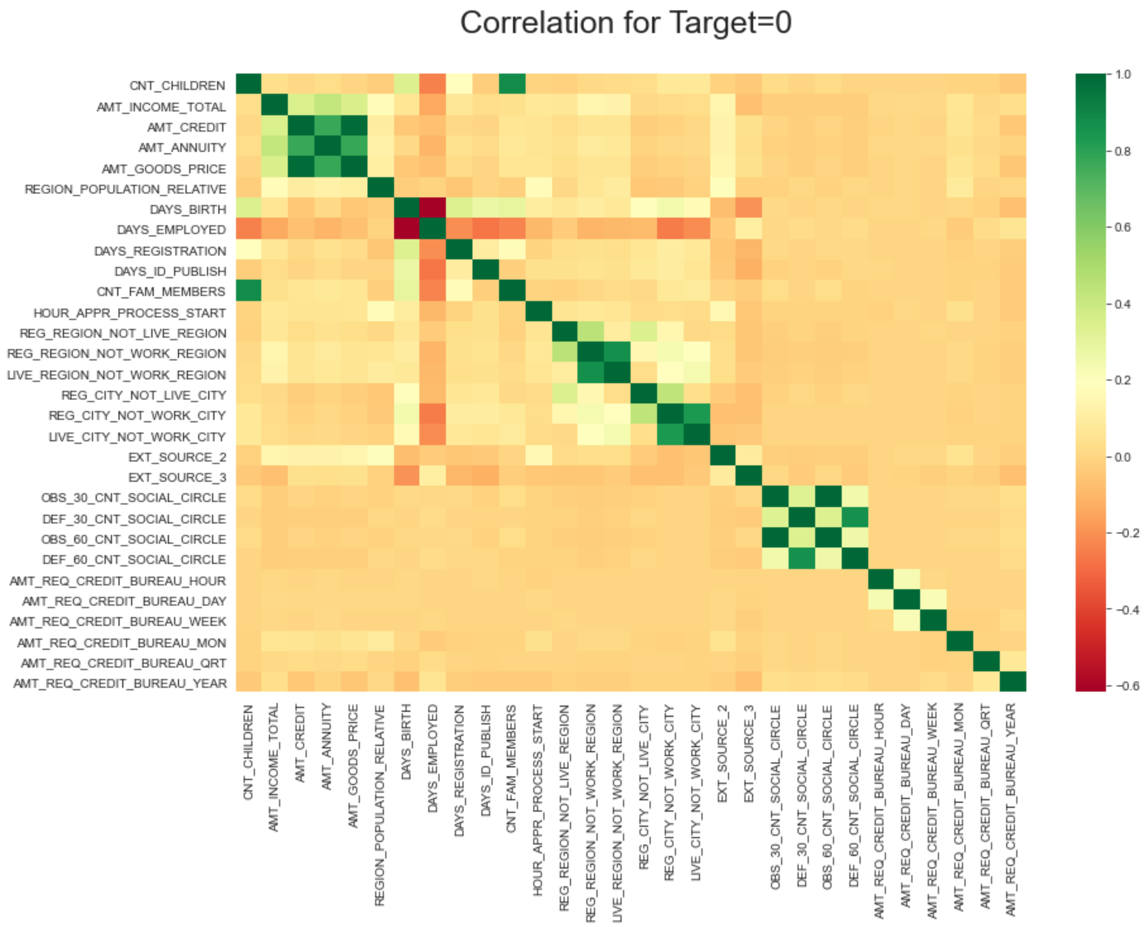
Out[43]:

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AI
CNT_CHILDREN	1.000000	0.027397	0.003081	
AMT_INCOME_TOTAL	0.027397	1.000000	0.342799	
AMT_CREDIT	0.003081	0.342799	1.000000	
AMT_ANNUITY	0.020905	0.418953	0.771309	
AMT_GOODS_PRICE	-0.000525	0.349462	0.987250	
REGION_POPULATION_RELATIVE	-0.024363	0.167851	0.100604	
DAYS_BIRTH	0.336966	0.062609	-0.047378	
DAYS_EMPLOYED	-0.243356	-0.141250	-0.072515	
DAYS_REGISTRATION	0.185792	0.064937	0.013477	
DAYS_ID_PUBLISH	-0.028751	0.022896	-0.001464	
CNT_FAM_MEMBERS	0.878571	0.034256	0.064536	
HOUR_APPR_PROCESS_START	-0.005244	0.076743	0.053619	
REG_REGION_NOT_LIVE_REGION	-0.012342	0.068510	0.024617	
REG_REGION_NOT_WORK_REGION	0.010857	0.137174	0.053735	
LIVE_REGION_NOT_WORK_REGION	0.017326	0.127701	0.054250	
REG_CITY_NOT_LIVE_CITY	0.021587	0.010567	-0.025036	
REG_CITY_NOT_WORK_CITY	0.072193	0.017618	-0.015703	
LIVE_CITY_NOT_WORK_CITY	0.070988	0.020684	0.002506	
EXT_SOURCE_2	-0.015455	0.139598	0.129140	
EXT_SOURCE_3	-0.041729	-0.072401	0.036085	
OBS_30_CNT_SOCIAL_CIRCLE	0.014471	-0.027828	-0.000914	
DEF_30_CNT_SOCIAL_CIRCLE	-0.002246	-0.027621	-0.019851	
OBS_60_CNT_SOCIAL_CIRCLE	0.014137	-0.027690	-0.000892	
DEF_60_CNT_SOCIAL_CIRCLE	-0.002172	-0.027593	-0.022225	
AMT_REQ_CREDIT_BUREAU_HOUR	-0.000432	0.001417	-0.003734	
AMT_REQ_CREDIT_BUREAU_DAY	0.000648	0.007862	0.004409	
AMT_REQ_CREDIT_BUREAU_WEEK	-0.001632	0.006234	-0.001883	

AMT_REQ_CREDIT_BUREAU_MON	-0.010455	0.061470	0.054071
AMT_REQ_CREDIT_BUREAU_QRT	-0.007087	0.013128	0.017767
AMT_REQ_CREDIT_BUREAU_YEAR	-0.042547	0.029536	-0.048866

In [44]:

```
# Plotting the correlation for the Target_0.

plt.figure(figsize=[14,9])
sns.heatmap(target_0_corr, annot=False, cmap='RdYlGn')
plt.title('Correlation for Target=0 \n', fontsize=25)
plt.show()
```



Conclusions from the 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.
> 4. Less children client have in densely populated area.
> 5. Credit amount is higher to densely populated area.
> 6. The income is also higher in densely populated area.

In [45]:

```
# Calculating the correlation among the target_1 people
```

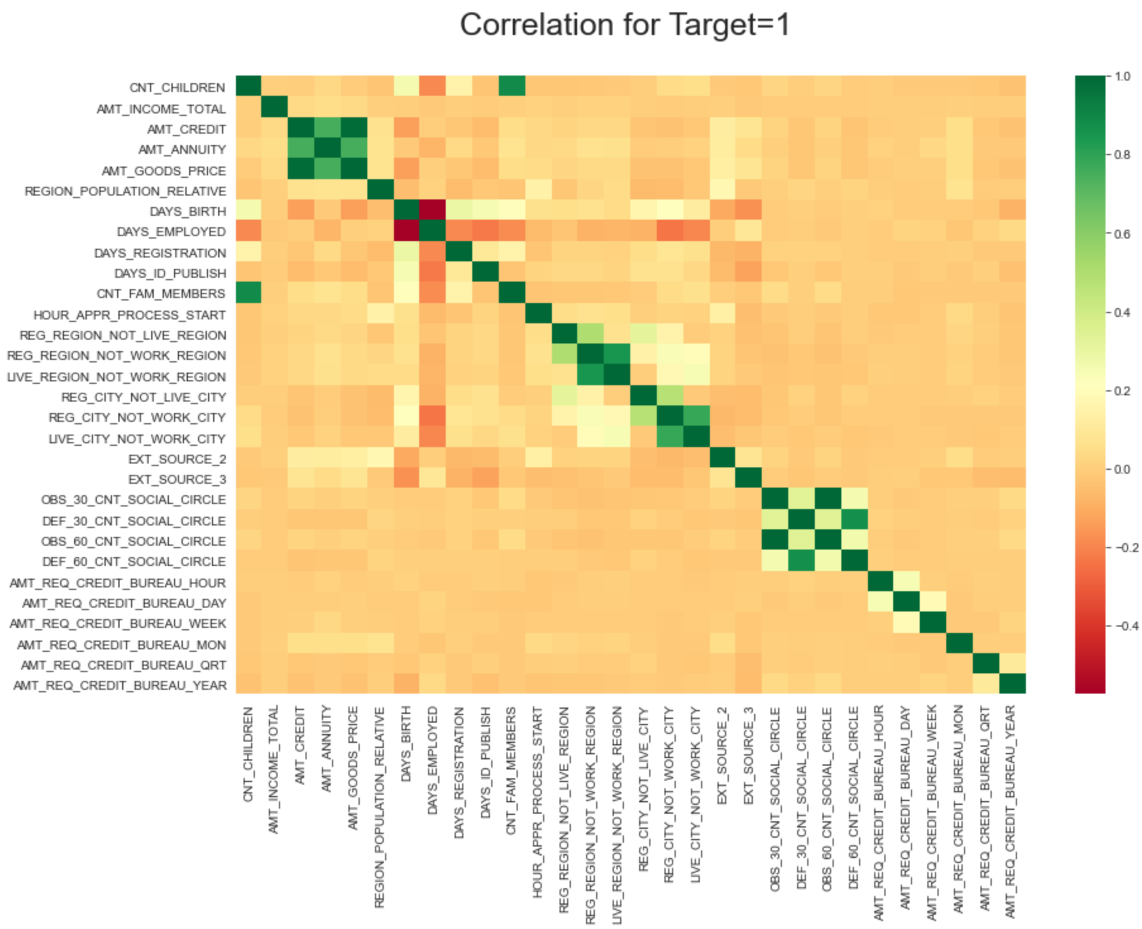
```
target_1_corr = target_1.iloc[0:, 2:].corr()
target_1_corr
```

Out[45]:

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AI
CNT_CHILDREN	1.000000	0.004796	-0.001675	
AMT_INCOME_TOTAL	0.004796	1.000000	0.038131	
AMT_CREDIT	-0.001675	0.038131	1.000000	
AMT_ANNUITY	0.031257	0.046421	0.752195	
AMT_GOODS_PRICE	-0.008112	0.037583	0.983103	
REGION_POPULATION_RELATIVE	-0.031975	0.009135	0.069161	
DAYS_BIRTH	0.259109	0.003096	-0.135316	
DAYS_EMPLOYED	-0.191942	-0.014979	-0.000968	
DAYS_REGISTRATION	0.149154	0.000158	-0.025854	
DAYS_ID_PUBLISH	-0.032299	-0.004215	-0.052329	
CNT_FAM_MEMBERS	0.885484	0.006654	0.051224	
HOUR_APPR_PROCESS_START	-0.023899	0.013775	0.031782	
REG_REGION_NOT_LIVE_REGION	-0.024322	0.007577	0.019540	
REG_REGION_NOT_WORK_REGION	-0.020793	0.014531	0.033260	
LIVE_REGION_NOT_WORK_REGION	-0.012073	0.013409	0.033554	
REG_CITY_NOT_LIVE_CITY	-0.001174	-0.002223	-0.033034	
REG_CITY_NOT_WORK_CITY	0.046115	-0.003019	-0.037720	
LIVE_CITY_NOT_WORK_CITY	0.053515	-0.001353	-0.016509	
EXT_SOURCE_2	-0.012260	0.007154	0.120848	
EXT_SOURCE_3	-0.020268	-0.015110	0.077698	
OBS_30_CNT_SOCIAL_CIRCLE	0.025804	-0.004709	0.019098	
DEF_30_CNT_SOCIAL_CIRCLE	0.001448	-0.005186	-0.025979	
OBS_60_CNT_SOCIAL_CIRCLE	0.025180	-0.004616	0.019487	
DEF_60_CNT_SOCIAL_CIRCLE	-0.005106	-0.004866	-0.030880	
AMT_REQ_CREDIT_BUREAU_HOUR	-0.000382	0.000656	-0.005981	
AMT_REQ_CREDIT_BUREAU_DAY	-0.013004	-0.000272	0.003008	
AMT_REQ_CREDIT_BUREAU_WEEK	-0.011792	0.000018	0.007650	
AMT_REQ_CREDIT_BUREAU_MON	-0.012583	0.004114	0.055038	
AMT_REQ_CREDIT_BUREAU_QRT	-0.018174	-0.001133	-0.017467	
AMT_REQ_CREDIT_BUREAU_YEAR	-0.035427	0.001752	-0.035719	

In [46]:

```
# Plotting the correlation for the Target_0.

plt.figure(figsize=[14,9])
sns.heatmap(target_1_corr, annot=False, cmap='RdYlGn')
plt.title('Correlation for Target=1 \n', fontsize=25)
plt.show()
```



Conclusions from the graph:

> Same like the target=0 heatmap above, adding some other points from this heatmap.
>
> 1. The client's permanent address does not match contact address are having less children and vice-versa
> 2. The client's permanent address does not match work address are having less children and vice-versa

Finding the top 10 correlations for Target 0 and Target 1

In [47]:

```
# Converting the negative values to postive values and sorting the value
corr_0 = target_0_corr.abs().unstack().sort_values(kind='quicksort').dro
corr_0 = corr_0[corr_0 != 1.0]
corr_0
```

Out[47]:

```
DAYS_REGISTRATION             AMT_REQ_CREDIT_BUREAU_DAY      0.000035
AMT_REQ_CREDIT_BUREAU_DAY     DAYS_REGISTRATION              0.000035
REG_REGION_NOT_WORK_REGION    AMT_REQ_CREDIT_BUREAU_WEEK     0.000125
AMT_REQ_CREDIT_BUREAU_WEEK    REG_REGION_NOT_WORK_REGION     0.000125
LIVE_CITY_NOT_WORK_CITY       AMT_REQ_CREDIT_BUREAU_HOUR     0.000149
                                                               ...
CNT_CHILDREN                  CNT_FAM_MEMBERS                0.878571
AMT_CREDIT                    AMT_GOODS_PRICE                0.987250
AMT_GOODS_PRICE               AMT_CREDIT                     0.987250
OBS_30_CNT_SOCIAL_CIRCLE      OBS_60_CNT_SOCIAL_CIRCLE       0.998508
```

```
OBS_60_CNT_SOCIAL_CIRCLE    OBS_30_CNT_SOCIAL_CIRCLE      0.998508
Length: 870, dtype: float64
```

In [48]:
```
# Top 10 correlation for the Target = 0,

corr_0.tail(10)
```

Out[48]:
```
DEF_60_CNT_SOCIAL_CIRCLE     DEF_30_CNT_SOCIAL_CIRCLE       0.859332
DEF_30_CNT_SOCIAL_CIRCLE     DEF_60_CNT_SOCIAL_CIRCLE       0.859332
REG_REGION_NOT_WORK_REGION   LIVE_REGION_NOT_WORK_REGION    0.861861
LIVE_REGION_NOT_WORK_REGION  REG_REGION_NOT_WORK_REGION     0.861861
CNT_FAM_MEMBERS              CNT_CHILDREN                   0.878571
CNT_CHILDREN                 CNT_FAM_MEMBERS                0.878571
AMT_CREDIT                   AMT_GOODS_PRICE                0.987250
AMT_GOODS_PRICE              AMT_CREDIT                     0.987250
OBS_30_CNT_SOCIAL_CIRCLE     OBS_60_CNT_SOCIAL_CIRCLE       0.998508
OBS_60_CNT_SOCIAL_CIRCLE     OBS_30_CNT_SOCIAL_CIRCLE       0.998508
dtype: float64
```

In [49]:
```
# Converting the negative values to postive values and sorting the value

corr_1 = target_1_corr.abs().unstack().sort_values(kind='quicksort').dro
corr_1 = corr_1[corr_1 != 1.0]
corr_1
```

Out[49]:
```
LIVE_REGION_NOT_WORK_REGION  REG_CITY_NOT_LIVE_CITY         0.000011
REG_CITY_NOT_LIVE_CITY       LIVE_REGION_NOT_WORK_REGION    0.000011
AMT_INCOME_TOTAL             AMT_REQ_CREDIT_BUREAU_WEEK     0.000018
AMT_REQ_CREDIT_BUREAU_WEEK   AMT_INCOME_TOTAL               0.000018
AMT_INCOME_TOTAL             DAYS_REGISTRATION              0.000158
                                                              ...
CNT_CHILDREN                 CNT_FAM_MEMBERS                0.885484
AMT_CREDIT                   AMT_GOODS_PRICE                0.983103
AMT_GOODS_PRICE              AMT_CREDIT                     0.983103
OBS_30_CNT_SOCIAL_CIRCLE     OBS_60_CNT_SOCIAL_CIRCLE       0.998269
OBS_60_CNT_SOCIAL_CIRCLE     OBS_30_CNT_SOCIAL_CIRCLE       0.998269
Length: 870, dtype: float64
```

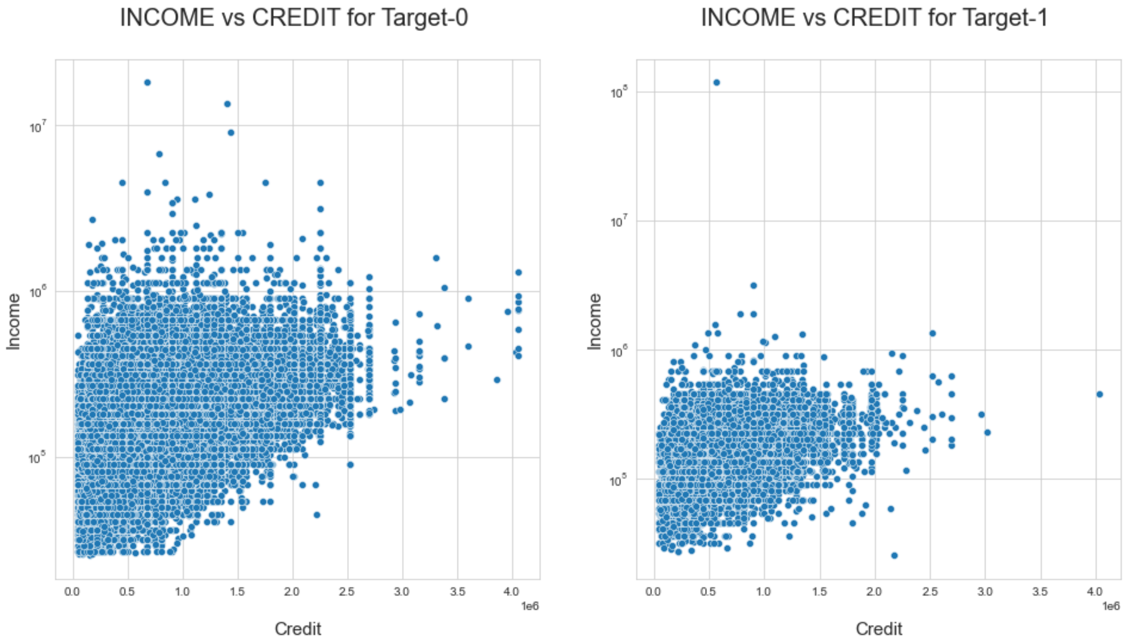
In [50]:
```
# Top 10 correlation for the Target = 1,

corr_1.tail(10)
```

Out[50]:
```
REG_REGION_NOT_WORK_REGION   LIVE_REGION_NOT_WORK_REGION    0.847885
LIVE_REGION_NOT_WORK_REGION  REG_REGION_NOT_WORK_REGION     0.847885
DEF_30_CNT_SOCIAL_CIRCLE     DEF_60_CNT_SOCIAL_CIRCLE       0.868994
DEF_60_CNT_SOCIAL_CIRCLE     DEF_30_CNT_SOCIAL_CIRCLE       0.868994
CNT_FAM_MEMBERS              CNT_CHILDREN                   0.885484
CNT_CHILDREN                 CNT_FAM_MEMBERS                0.885484
AMT_CREDIT                   AMT_GOODS_PRICE                0.983103
AMT_GOODS_PRICE              AMT_CREDIT                     0.983103
OBS_30_CNT_SOCIAL_CIRCLE     OBS_60_CNT_SOCIAL_CIRCLE       0.998269
OBS_60_CNT_SOCIAL_CIRCLE     OBS_30_CNT_SOCIAL_CIRCLE       0.998269
dtype: float64
```

11.Bivariate Analysis of the numerical columns

In [51]:
```
# Plotting scatterplot to find any correlations and to check the trends

plt.figure(figsize=[16,8])

plt.subplot(1,2,1)
sns.scatterplot(target_0.AMT_CREDIT, target_0.AMT_INCOME_TOTAL)
```

```
plt.title('INCOME vs CREDIT for Target-0 \n', fontsize=20)
plt.yscale('log')
plt.xlabel('\nCredit', fontsize=15)
plt.ylabel('\nIncome', fontsize=15)

plt.subplot(1,2,2)
sns.scatterplot(target_1.AMT_CREDIT, target_1.AMT_INCOME_TOTAL)
plt.title('INCOME vs CREDIT for Target-1 \n', fontsize=20)
plt.yscale('log')
plt.xlabel('\nCredit', fontsize=15)
plt.ylabel('\nIncome', fontsize=15)

plt.show()
```



In [52]:

```
# Plotting scatterplot to find any correlations and to check the trends


plt.figure(figsize=[16,8])

plt.subplot(1,2,1)
sns.scatterplot(x='AMT_CREDIT',y='AMT_GOODS_PRICE',data=target_0)
plt.title('CREDIT vs GOODS PRICE for Target-0 \n', fontsize=20)
plt.xlabel('\nCredit', fontsize=15)
plt.ylabel('\nGoods Price', fontsize=15)

plt.subplot(1,2,2)
sns.scatterplot(x='AMT_CREDIT',y='AMT_GOODS_PRICE',data=target_1)
plt.title('CREDIT vs GOODS PRICE for Target-1 \n', fontsize=20)
plt.xlabel('\nCredit', fontsize=15)
plt.ylabel('\nGoods Price', fontsize=15)

plt.show()
```

Conclusions from the graph:

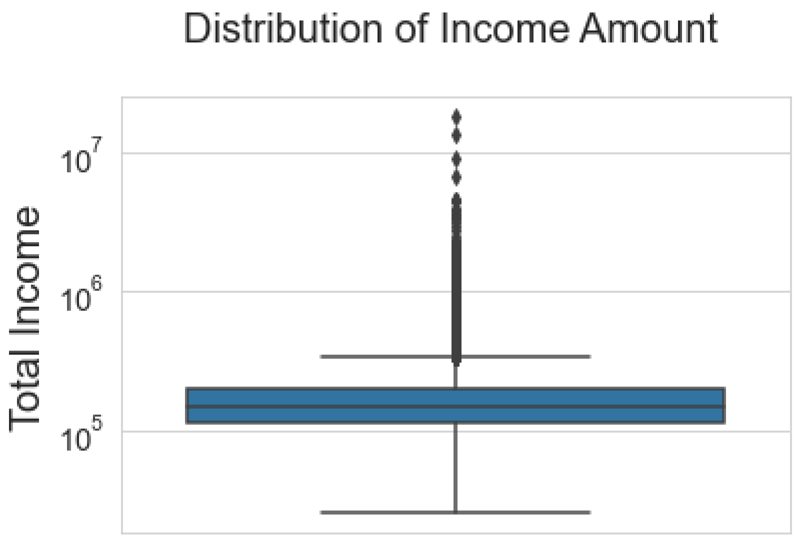
> With the scatter plot,we can determine that AMT CREDIT and AMT GOODS PRICE are highly correlated,which means if increase in goods price,the credit increased directly and vice versa.

Finding Outliers

Univariate Analysis

For Target=0

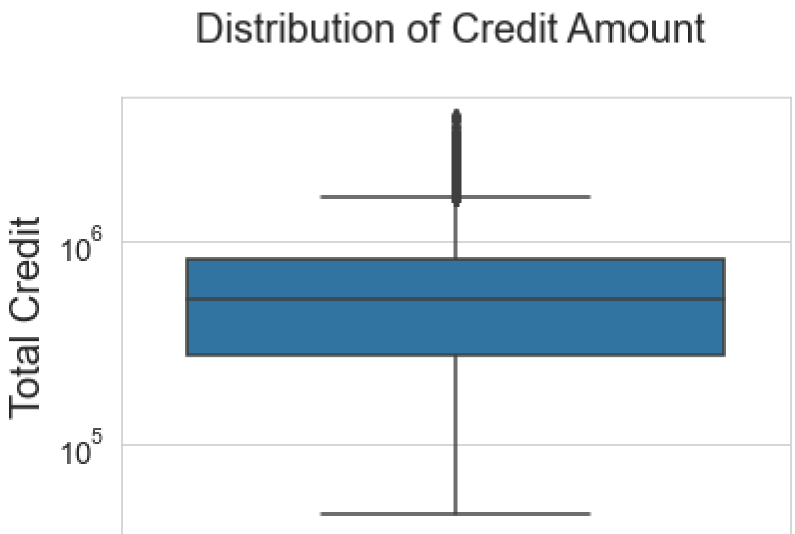
In [53]:
```
# Distribution of Income Amount,

sns.set_style('whitegrid')

sns.boxplot(data=target_0, y='AMT_INCOME_TOTAL')
plt.yscale('log')
plt.yticks(fontsize=15)
plt.ylabel('Total Income', fontsize=20)
plt.title('Distribution of Income Amount \n', fontsize=20)

plt.show()
```



Conclusions from the graph:

> 1. There seems to be an equal distribution of the Income amount of the clients.
> 2. Also some of the outliers present in the dataset.

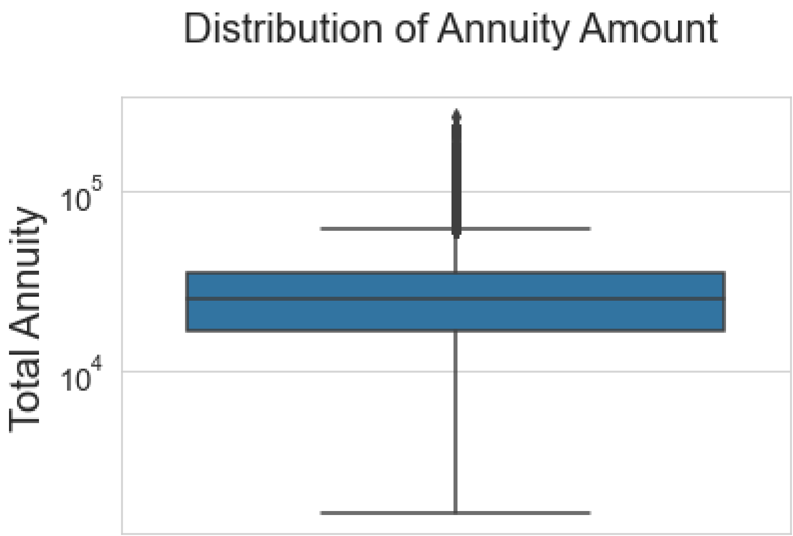
In [54]:
```
# Distribution of Credit Amount,

sns.set_style('whitegrid')

sns.boxplot(data=target_0, y='AMT_CREDIT')
plt.yscale('log')
plt.yticks(fontsize=15)
plt.ylabel('Total Credit', fontsize=20)
plt.title('Distribution of Credit Amount \n', fontsize=20)

plt.show()
```



Conclusions from the graph:

> 1. The first quartile is bigger than the third quartile, that means most of the client credit lies in the first quartile.
> 2. There seems some outliers in the Credit boxplot.

In [55]:
```
# Distribution of Annuity,

sns.set_style('whitegrid')

sns.boxplot(data=target_0, y='AMT_ANNUITY')
plt.yscale('log')
plt.yticks(fontsize=15)
plt.ylabel('Total Annuity', fontsize=20)
plt.title('Distribution of Annuity Amount \n', fontsize=20)

plt.show()
```

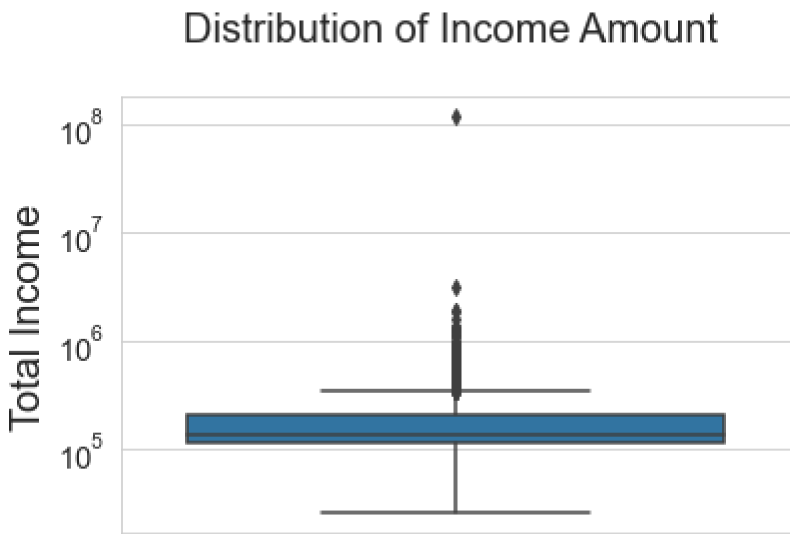


Conclusions from the graph:

> 1. The first quartile is bigger than the third quartile.
> 2. There seems some outliers in the Anuuity boxplot.

For Target=1

In [56]:
```
# Distribution of Income Amount,

sns.set_style('whitegrid')

sns.boxplot(data=target_1, y='AMT_INCOME_TOTAL')
plt.yscale('log')
plt.yticks(fontsize=15)
plt.ylabel('Total Income', fontsize=20)
plt.title('Distribution of Income Amount \n', fontsize=20)

plt.show()
```

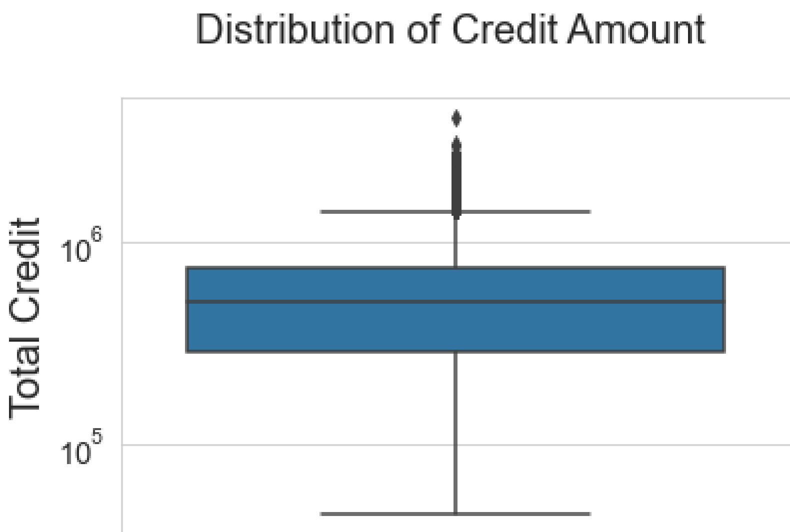


Conclusions from the graph:

> 1. There seems a significant outlier in the Income dataset.
> 2. Most of the income of the client lies in the third quartile.

In [57]:

```
# Distribution of Credit Amount,

sns.set_style('whitegrid')

sns.boxplot(data=target_1, y='AMT_CREDIT')
plt.yscale('log')
plt.yticks(fontsize=15)
plt.ylabel('Total Credit', fontsize=20)
plt.title('Distribution of Credit Amount \n', fontsize=20)

plt.show()
```



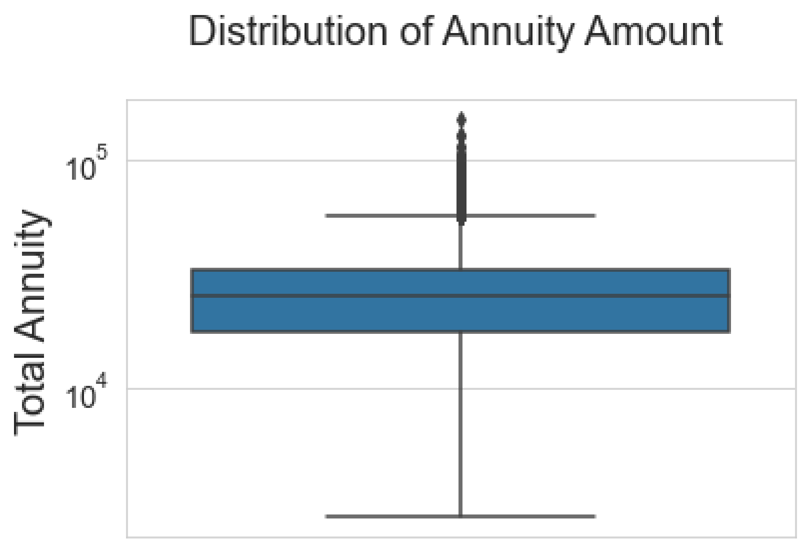
Conclusions from the graph:

> 1. The first quartile is bigger than the third quartile, that means most of the client credit lies in the first quartile.
> 2. There seems some outliers in the Credit boxplot.

In [58]:

```
# Distribution of Annuity,
```

```
sns.set_style('whitegrid')

sns.boxplot(data=target_1, y='AMT_ANNUITY')
plt.yscale('log')
plt.yticks(fontsize=15)
plt.ylabel('Total Annuity', fontsize=20)
plt.title('Distribution of Annuity Amount \n', fontsize=20)

plt.show()
```



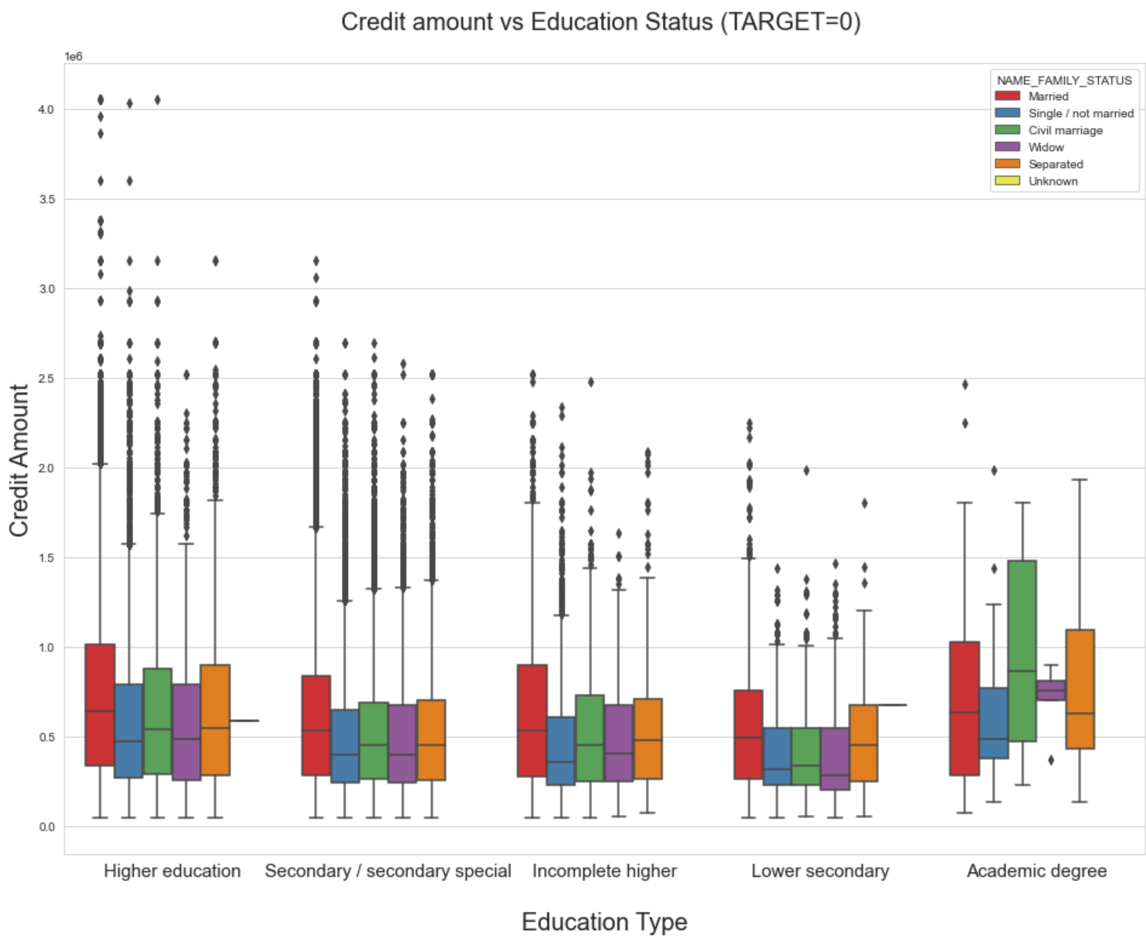
Conclusions from the graph:

> 1. The first quartile is bigger than the third quartile.
> 2. There seems some outliers in the Anuuity boxplot.

Multivariate Analysis

Target = 0

In [59]:
```
# Box Plotting for the Target = 0, Credit Amount

plt.figure(figsize=[16,12])

sns.boxplot(data =target_0, x='NAME_EDUCATION_TYPE',y='AMT_CREDIT', hue =

plt.xticks(rotation=0,fontsize=15)
plt.xlabel('\nEducation Type', fontsize=20)
plt.ylabel('Credit Amount', fontsize=20)
plt.title('Credit amount vs Education Status (TARGET=0) \n', fontsize=20
plt.show()
```

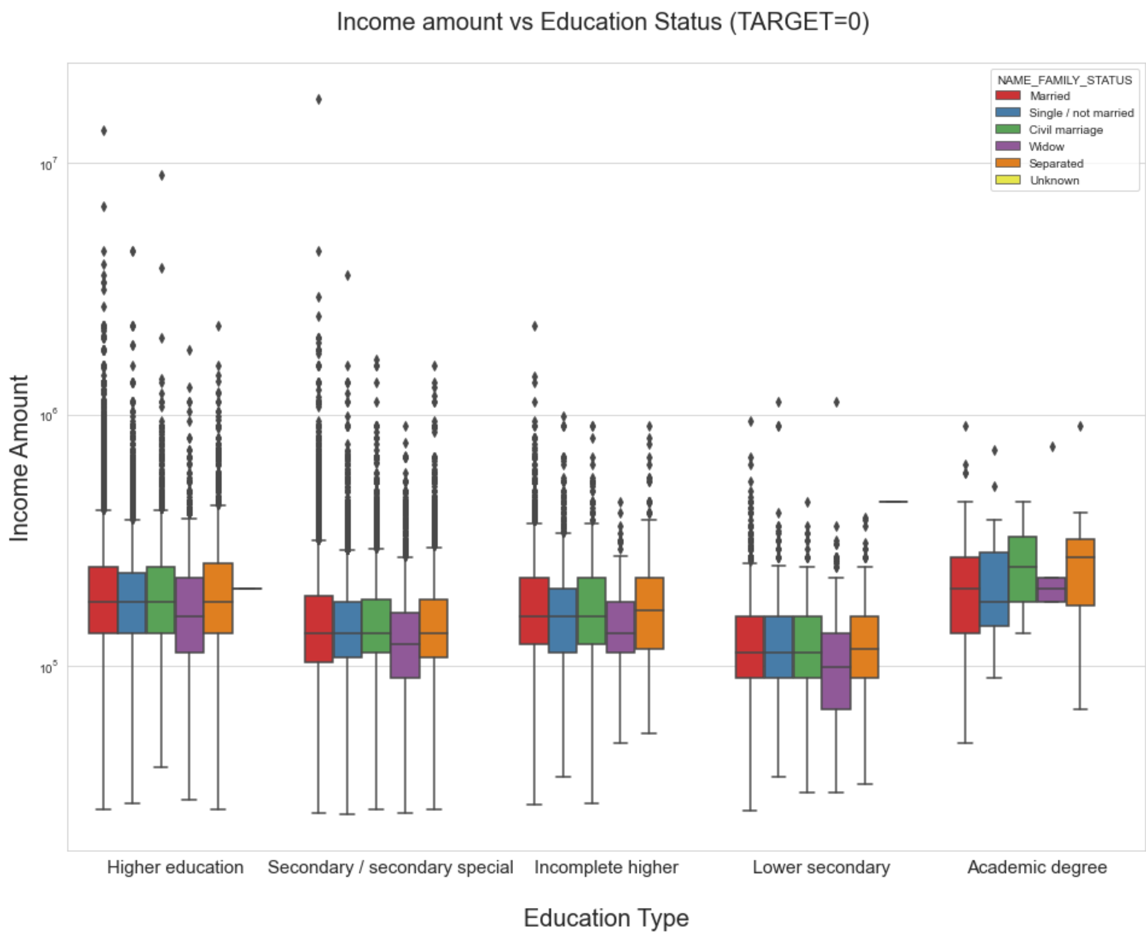


Conclusions from the graph:

> From the above box plot we can conclude that 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' are having more outliers. Civil marriage for Academic degree is having most of the credits in the third quartile.

In [60]:

```
# Box Plotting for the Target = 0, Income Amount

plt.figure(figsize=[16,12])

sns.boxplot(data =target_0, x='NAME_EDUCATION_TYPE',y='AMT_INCOME_TOTAL'

plt.xticks(fontsize=15)
plt.xlabel('\nEducation Type', fontsize=20)
plt.ylabel('Income Amount', fontsize=20)
plt.yscale('log')
plt.title('Income amount vs Education Status (TARGET=0) \n', fontsize=20

plt.show()
```



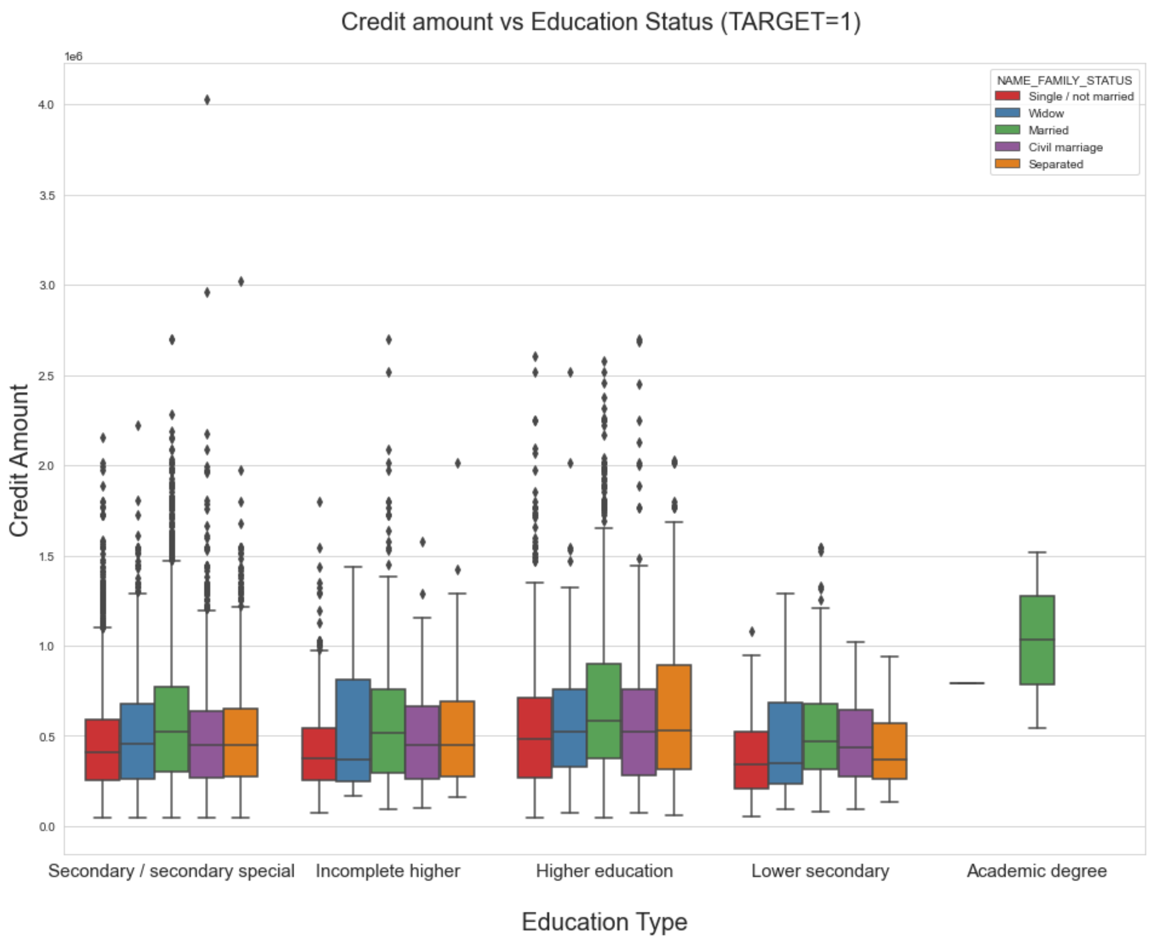
Conclusions from the graph:

> From above boxplot for 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.

Target = 1

In [61]:

```
# Box Plotting for the Target = 1, Credit Amount

plt.figure(figsize=[16,12])

sns.boxplot(data =target_1, x='NAME_EDUCATION_TYPE', y='AMT_CREDIT', hue

plt.xticks(fontsize=15)
plt.xlabel('\nEducation Type', fontsize=20)
plt.ylabel('Credit Amount', fontsize=20)
plt.title('Credit amount vs Education Status (TARGET=1) \n', fontsize=20
plt.show()
```

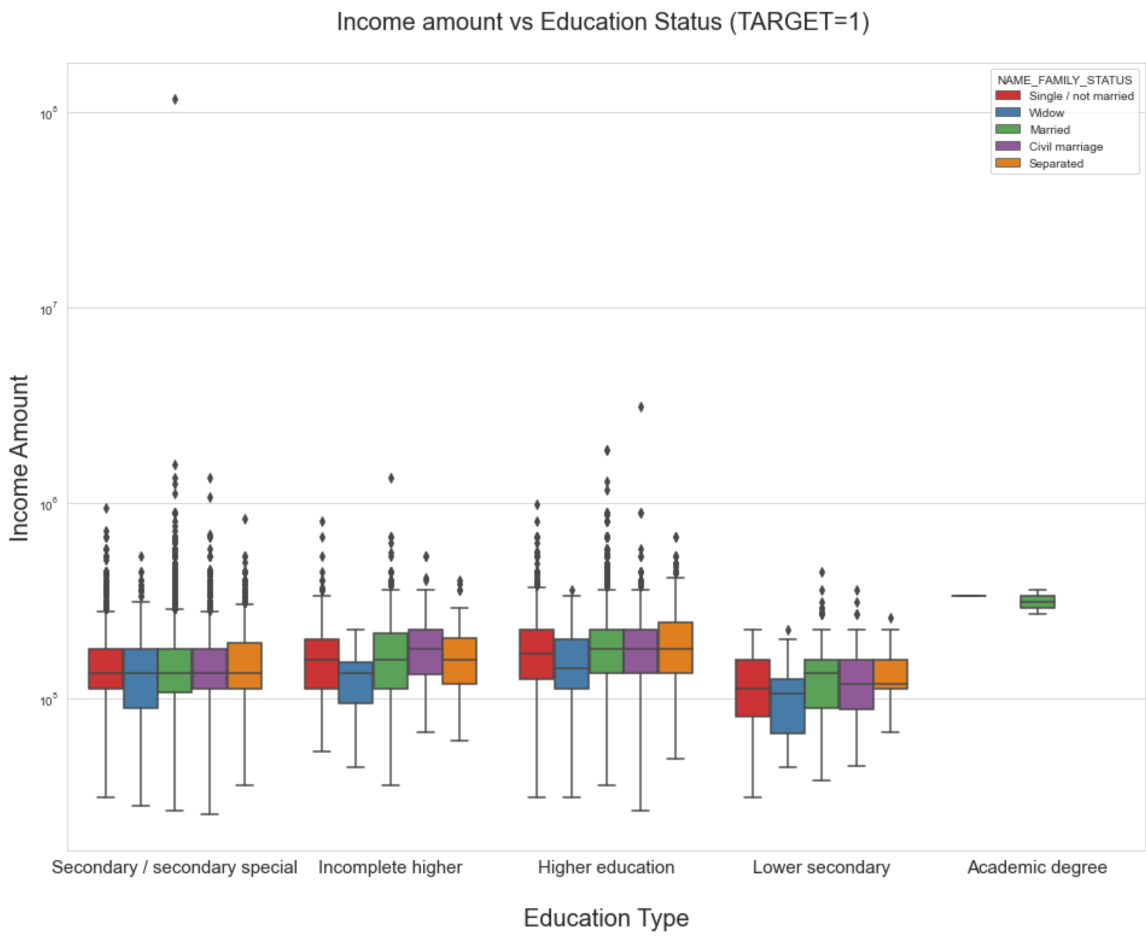


Conclusions from the graph:

> 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 Academic degree is having most of the credits in the third quartile.

In [62]:

```
# Box Plotting for the Target = 1, Income Amount

plt.figure(figsize=[16,12])

sns.boxplot(data =target_1, x='NAME_EDUCATION_TYPE',y='AMT_INCOME_TOTAL'

plt.xticks(fontsize=15)
plt.xlabel('\nEducation Type', fontsize=20)
plt.ylabel('Income Amount', fontsize=20)
plt.yscale('log')
plt.title('Income amount vs Education Status (TARGET=1) \n', fontsize=20
plt.show()
```



Conclusions from the graph:

> 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.

12.Work on previous_application dataset

In [63]:

```
# Removing the 'XNA' and 'XAP' column values from the column,

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

In [64]:

```
# Rechecking the NAME_CASH_LOAN_PURPOSE for the values.

P_A.NAME_CASH_LOAN_PURPOSE.value_counts()
```

Out[64]:

```
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
```

13.Merging the two datasets, i.e. application_dataset and previous_application

In [65]:

```
# Merging of the two datasets,

loan_merg = pd.merge(left = A_D, right = P_A, how = 'inner', on = 'SK_ID_
loan_merg.head()
```

Out[65]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_	CODE_GENDER	FLAG_OWN_CAR	FL
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	

In [66]:

```
# Renaming the columns in the loan_merg datasets,

loan_merg = loan_merg.rename({'NAME_CONTRACT_TYPE_' : 'NAME_CONTRACT_TYP
                         'WEEKDAY_APPR_PROCESS_START_' : 'WEEKDAY_APPR_P
                         'HOUR_APPR_PROCESS_START_':'HOUR_APPR_PROCESS_S
                         'AMT_CREDITx':'AMT_CREDIT_PREV','AMT_ANNUITYx':
                         'WEEKDAY_APPR_PROCESS_STARTx':'WEEKDAY_APPR_PRO
                         'HOUR_APPR_PROCESS_STARTx':'HOUR_APPR_PROCESS_S
```

In [67]:

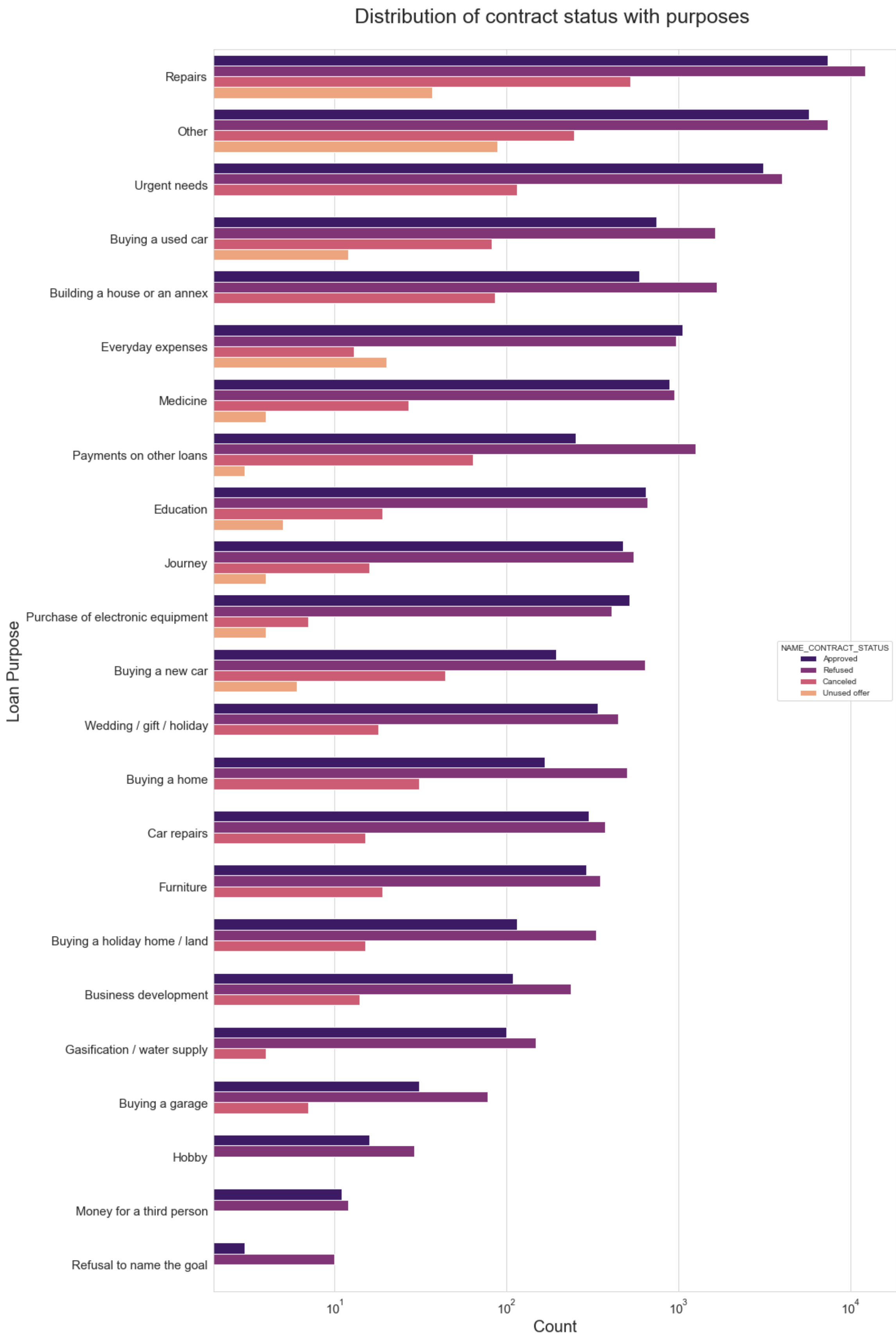
```
# Removing the unwanted columns from the dataset for the ease of analysi

loan_merg.drop(['SK_ID_CURR','WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PR
              'REG_REGION_NOT_WORK_REGION','LIVE_REGION_NOT_WORK_REGION'
              'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY','WEEKD
              'HOUR_APPR_PROCESS_START_PREV', 'FLAG_LAST_APPL_PER_CONTRA
```

14.Performing the Univariate analysis

In [68]:

```
# Plotting for the Contract Status,
```

```
plt.figure(figsize=[15,28])

sns.countplot(data = loan_merg, y= 'NAME_CASH_LOAN_PURPOSE',
                   order=loan_merg['NAME_CASH_LOAN_PURPOSE'].value_count

plt.title('Distribution of contract status with purposes \n', fontsize=2
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.xlabel('Count', fontsize=20)
plt.ylabel('Loan Purpose', fontsize=20)
plt.xscale('log')

plt.show()
```

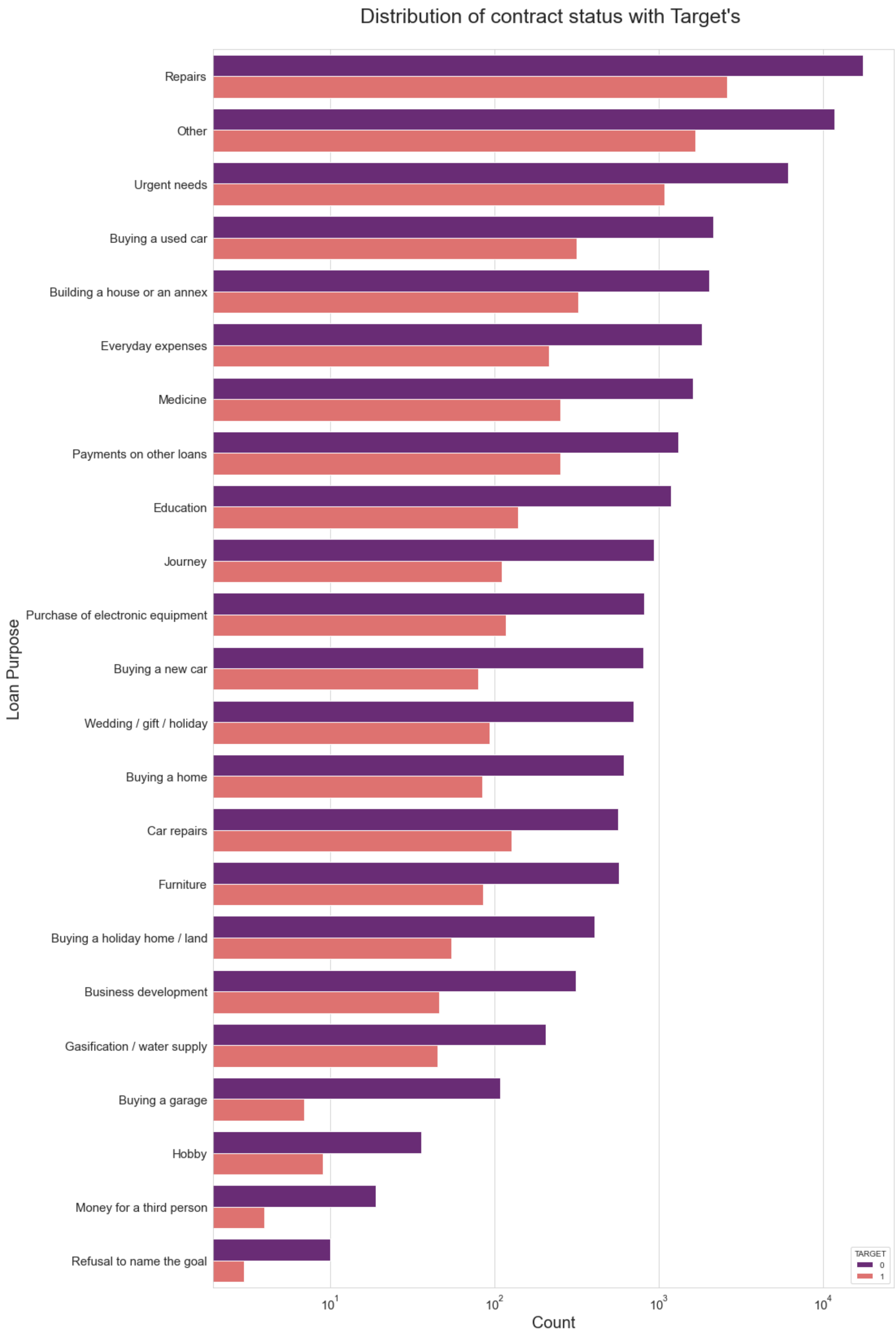


Conclusions from the graph:

> 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 [69]:

```
# Plotting for the Contract Status,

plt.figure(figsize=[15,28])

sns.countplot(data = loan_merg, y= 'NAME_CASH_LOAN_PURPOSE',
                   order=loan_merg['NAME_CASH_LOAN_PURPOSE'].value_count

plt.title('Distribution of contract status with Target\'s \n', fontsize=
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.xlabel('Count', fontsize=20)
plt.ylabel('Loan Purpose', fontsize=20)
plt.xscale('log')

plt.show()
```



Conclusions from the graph:

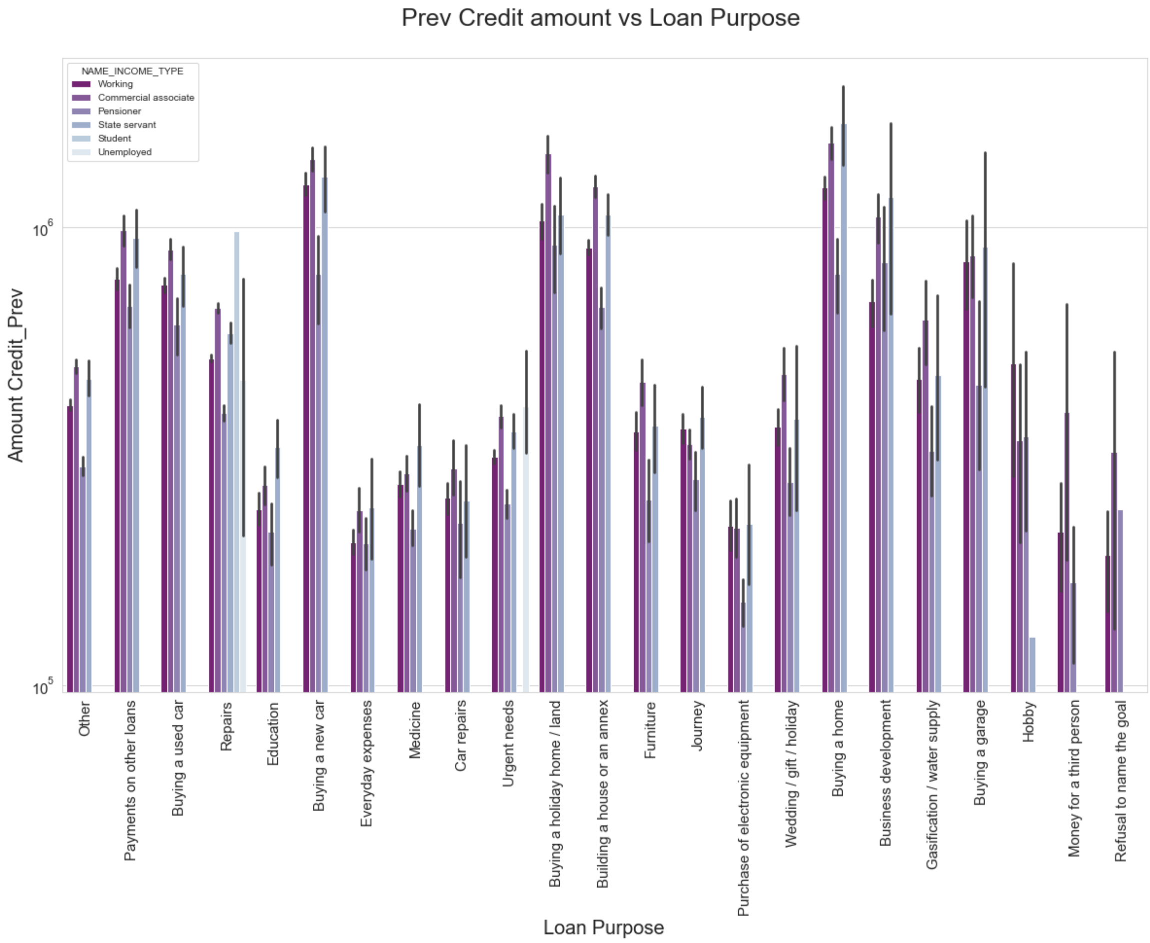
> 1. Loan purposes with 'Repairs' are facing more difficulites in payment on time.
> 2. There are few places where loan payment is significant higher than facing difficulties. They are 'Buying a garage', 'Business developemt', 'Buying land','Buying a new car' and 'Education' Hence we can focus

on these purposes for which the client is having for minimal payment difficulties.

15.Performing the bivariate analysis

In [70]:

```
# Plotting for Credit amount in logarithmic scale

plt.figure(figsize=(20,12))

sns.barplot(data = loan_merg, x='NAME_CASH_LOAN_PURPOSE', hue='NAME_INCO
plt.xticks(rotation=90)
plt.ylabel('Amount Credit_Prev', fontsize=20)
plt.xlabel('Loan Purpose', fontsize=20)
plt.yscale('log')
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.title('Prev Credit amount vs Loan Purpose \n', fontsize=25)

plt.show()
```

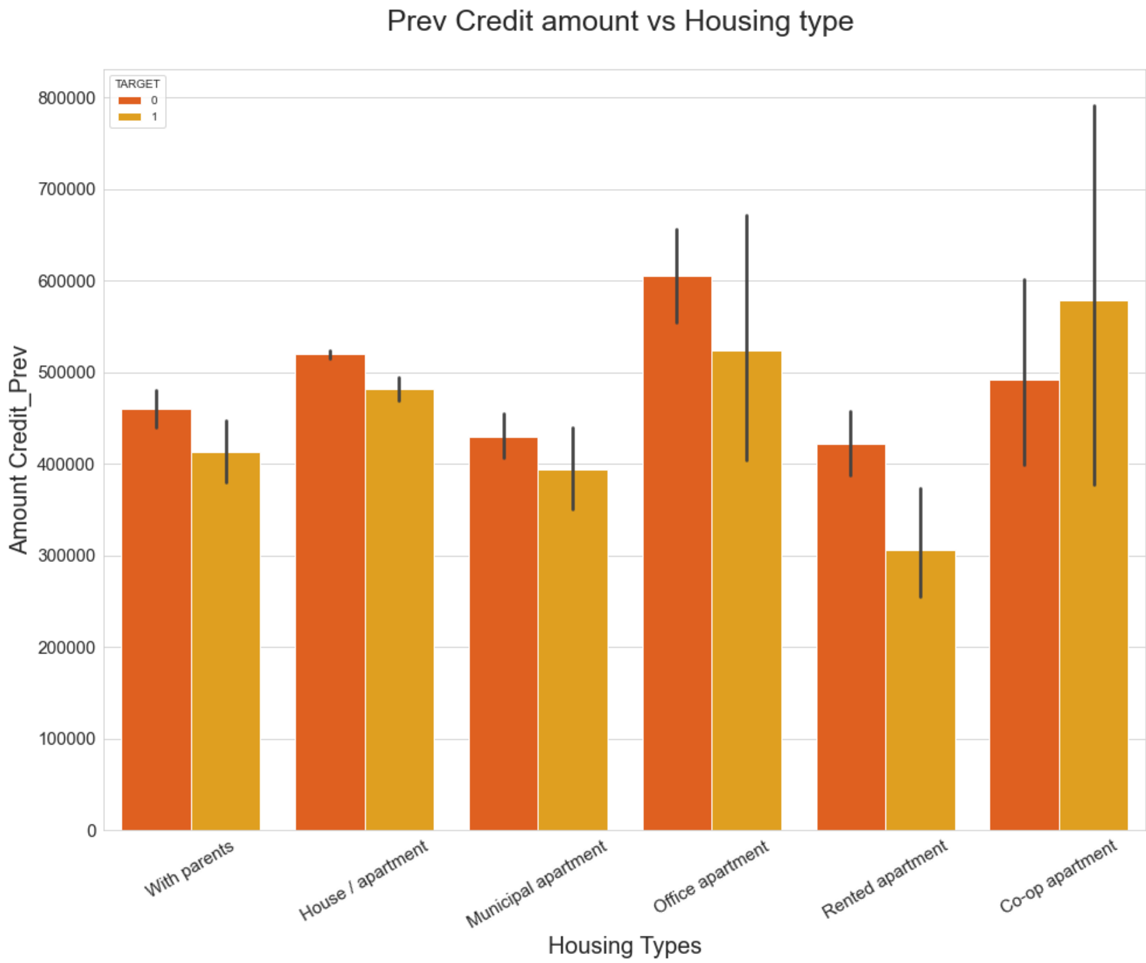


Conclusions from the graph:

> 1. The credit amount of Loan purposes like 'Buying a home','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 [71]:

```
# Plotting for Credit amount prev vs Housing type,

plt.figure(figsize=(16,12))
plt.xticks(rotation=30)
sns.barplot(data =loan_merg, y='AMT_CREDIT_PREV',hue='TARGET',x='NAME_HO
plt.title('Prev Credit amount vs Housing type \n', fontsize=25)
plt.ylabel('Amount Credit_Prev', fontsize=20)
plt.xlabel('Housing Types', fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)

plt.show()
```



Conclusions from the graph:

> 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.

Conclusions of this loan EDA Analysis:

> Banks should approve loans more for Office apartment, Co-Op apartment housing type as there are less payment difficulties.
>
> Banks should provide loans to ‘Repairs’ & ‘Others’ purposes.

> Banks should provide loans to the ‘Business Entity Type-3’ and ‘Self-Employed’ persons.
>
> ‘Working’ people especially female employers are the best to target for the loans.