Importing libraries

In [1]:

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

Loading dataset

In [2]:

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

Analysis of the dataset

In [3]:

```
app_data.shape
```

Out[3]:

```
(307511, 122)
```

In [4]:

```
#finding the columns that have null values more than 30%
nullcol=app_data.isnull().sum()
nullcol=nullcol[nullcol.values>(0.3*len(nullcol))]
len(nullcol)
```

Out[4]:

```
64
```

In [5]:

```
#removing the 64 columns that have more than 30% of null values
nullcol=list(nullcol[nullcol.values>=0.3].index)
app_data.drop(nullcol,axis=1,inplace=True)
```

In [6]:

```
#now checking the null values
app_data.isnull().sum()/len(app_data)*100
```

Out[6]:

```
SK_ID_CURR                    0.000000
TARGET                        0.000000
NAME_CONTRACT_TYPE            0.000000
CODE_GENDER                   0.000000
FLAG_OWN_CAR                  0.000000
FLAG_OWN_REALTY               0.000000
CNT_CHILDREN                  0.000000
AMT_INCOME_TOTAL              0.000000
AMT_CREDIT                    0.000000
AMT_ANNUITY                   0.003902
NAME_INCOME_TYPE              0.000000
NAME_EDUCATION_TYPE           0.000000
NAME_FAMILY_STATUS            0.000000
NAME_HOUSING_TYPE             0.000000
REGION_POPULATION_RELATIVE    0.000000
DAYS_BIRTH                    0.000000
DAYS_EMPLOYED                 0.000000
DAYS_REGISTRATION             0.000000
DAYS_ID_PUBLISH               0.000000
FLAG_MOBIL                    0.000000
```

```
FLAG_EMP_PHONE                 0.000000
FLAG_WORK_PHONE                0.000000
FLAG_CONT_MOBILE               0.000000
FLAG_PHONE                     0.000000
FLAG_EMAIL                     0.000000
CNT_FAM_MEMBERS                0.000650
REGION_RATING_CLIENT           0.000000
REGION_RATING_CLIENT_W_CITY    0.000000
WEEKDAY_APPR_PROCESS_START     0.000000
HOUR_APPR_PROCESS_START        0.000000
REG_REGION_NOT_LIVE_REGION     0.000000
REG_REGION_NOT_WORK_REGION     0.000000
LIVE_REGION_NOT_WORK_REGION    0.000000
REG_CITY_NOT_LIVE_CITY         0.000000
REG_CITY_NOT_WORK_CITY         0.000000
LIVE_CITY_NOT_WORK_CITY        0.000000
ORGANIZATION_TYPE              0.000000
DAYS_LAST_PHONE_CHANGE         0.000325
FLAG_DOCUMENT_2                0.000000
FLAG_DOCUMENT_3                0.000000
FLAG_DOCUMENT_4                0.000000
FLAG_DOCUMENT_5                0.000000
FLAG_DOCUMENT_6                0.000000
FLAG_DOCUMENT_7                0.000000
FLAG_DOCUMENT_8                0.000000
FLAG_DOCUMENT_9                0.000000
FLAG_DOCUMENT_10               0.000000
FLAG_DOCUMENT_11               0.000000
```

```
FLAG_DOCUMENT_12               0.000000
FLAG_DOCUMENT_13               0.000000
FLAG_DOCUMENT_14               0.000000
FLAG_DOCUMENT_15               0.000000
FLAG_DOCUMENT_16               0.000000
FLAG_DOCUMENT_17               0.000000
FLAG_DOCUMENT_18               0.000000
FLAG_DOCUMENT_19               0.000000
FLAG_DOCUMENT_20               0.000000
FLAG_DOCUMENT_21               0.000000
dtype: float64
```

we will impute the missing values in the AMT_ANNUITY column and fill the values using median

In [6]:

```
#filling missing values with median
value=app_data['AMT_ANNUITY'].median()
app_data.loc[app_data['AMT_ANNUITY'].isnull(),'AMT_ANNUITY']=value
```

In [7]:

```
#removing rows having null values >=30%
nullrow=app_data.isnull().sum(axis=1)
nullrow=list(nullrow[nullrow.values>=0.3*len(app_data)].index)
app_data.drop(nullrow,axis=0,inplace=True)
```

In [8]:

```
#removing unwanted columns from the dataset
unwanted=['FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE','FLAG_PHONE
       'REGION_RATING_CLIENT_W_CITY','DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_2', 'FLAG_DOC
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9','FLAG_DOCUMENT_10', 'FLAG_DO
       'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15','FLAG_DOCUMENT_16', 'FLAG
       'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21']

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

In [9]:

```
#finding columns with 'XNA' values
#for gender column
app_data[app_data['CODE_GENDER']=='XNA'].shape
```

Out[9]:

```
(4, 28)
```

In [10]:

```
#for organizational column
app_data[app_data['ORGANIZATION_TYPE']=='XNA'].shape
```

Out[10]:

```
(55374, 28)
```

In [11]:

```
#finding the number of males and females in the gender column
app_data['CODE_GENDER'].value_counts()
```

Out[11]:

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

In [12]:

```
#since majority of the values are 'F' we can update the 'XNA' values to 'F'
app_data.loc[app_data['CODE_GENDER']=='XNA','CODE_GENDER']='F'
app_data['CODE_GENDER'].value_counts()
```

Out[12]:

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

In [14]:

```
#finding the values in organization column
app_data['ORGANIZATION_TYPE'].value_counts()
```

Out[14]:

```
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
```

In [13]:

```
#55374 rows have the 'XNA' values. therfore we will drop these rows
app_data=app_data.drop(app_data.loc[app_data['ORGANIZATION_TYPE']=='XNA'].index)
app_data[app_data['ORGANIZATION_TYPE']=='XNA'].shape
```

Out[13]:

```
(0, 28)
```

In [14]:

```
#casting all variables into numeric types
numeric_col=['TARGET','CNT_CHILDREN','AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','REGION_
                'DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH','HOUR_APPR_PROCESS_ST
app_data[numeric_col]=app_data[numeric_col].apply(pd.to_numeric)
app_data.head()
```

Out[14]:

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

5 rows × 28 columns

Now we will create bins for continous variable category columns like 'AMT_INCOME_TOTAL' and 'AMT_CREDIT'

In [15]:

```
#creating bins for AMT_INCOME_TOTAL
bins=[0,25000,50000,75000,100000,125000,150000,175000,200000,225000,250000,275000,300000,32
slots=['0-25000', '25000-50000','50000-75000','75000,100000','100000-125000', '125000-15000
       '200000-225000','225000-250000','250000-275000','275000-300000','300000-325000','325
       '375000-400000','400000-425000','425000-450000','450000-475000','475000-500000','500
app_data['AMT_INCOME_RANGE']=pd.cut(app_data['AMT_INCOME_TOTAL'],bins,labels=slots)
```

In [18]:

```
#creating bins for AMT_CREDIT
bins = [0,150000,200000,250000,300000,350000,400000,450000,500000,550000,600000,650000,7000
slots = ['0-150000', '150000-200000','200000-250000', '250000-300000', '300000-350000', '35
         '450000-500000','500000-550000','550000-600000','600000-650000','650000-700000','70
         '800000-850000','850000-900000','900000 and above']

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

In [16]:

```
# Dividing the dataset into two dataset of  target=1(client with payment difficulties) and
target0_app_data=app_data.loc[app_data["TARGET"]==0]
target1_app_data=app_data.loc[app_data["TARGET"]==1]
```

In [17]:

```
#calculating the imbalance percentage
round(len(target0_app_data)/len(target1_app_data),2)
```

Out[17]:

```
10.55
```

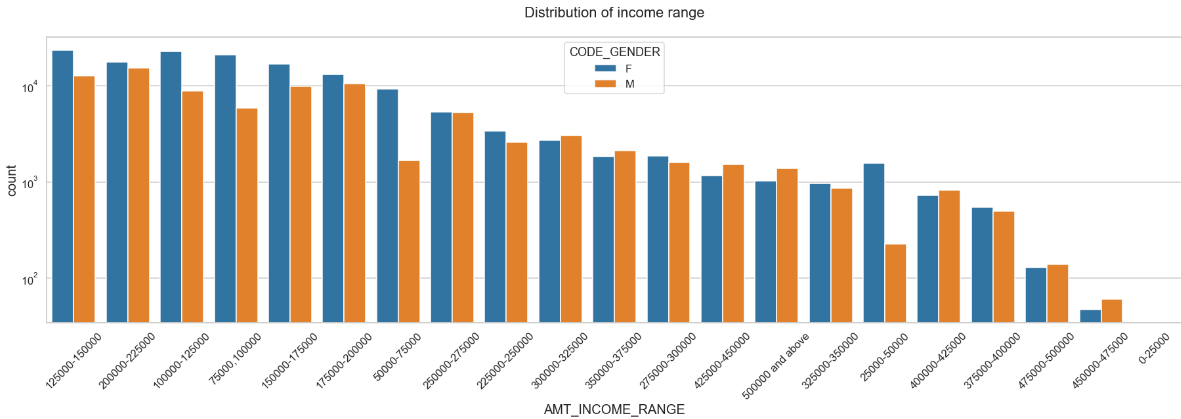
Univariate Analysis

In [22]:

```
#univariate analysis for target0 and plotting using loagrithmic scale
def uniplot(df,col,title,hue =None):

    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 [23]:

```
uniplot(target0_app_data,col='AMT_INCOME_RANGE',title='Distribution of income range',hue='C
```

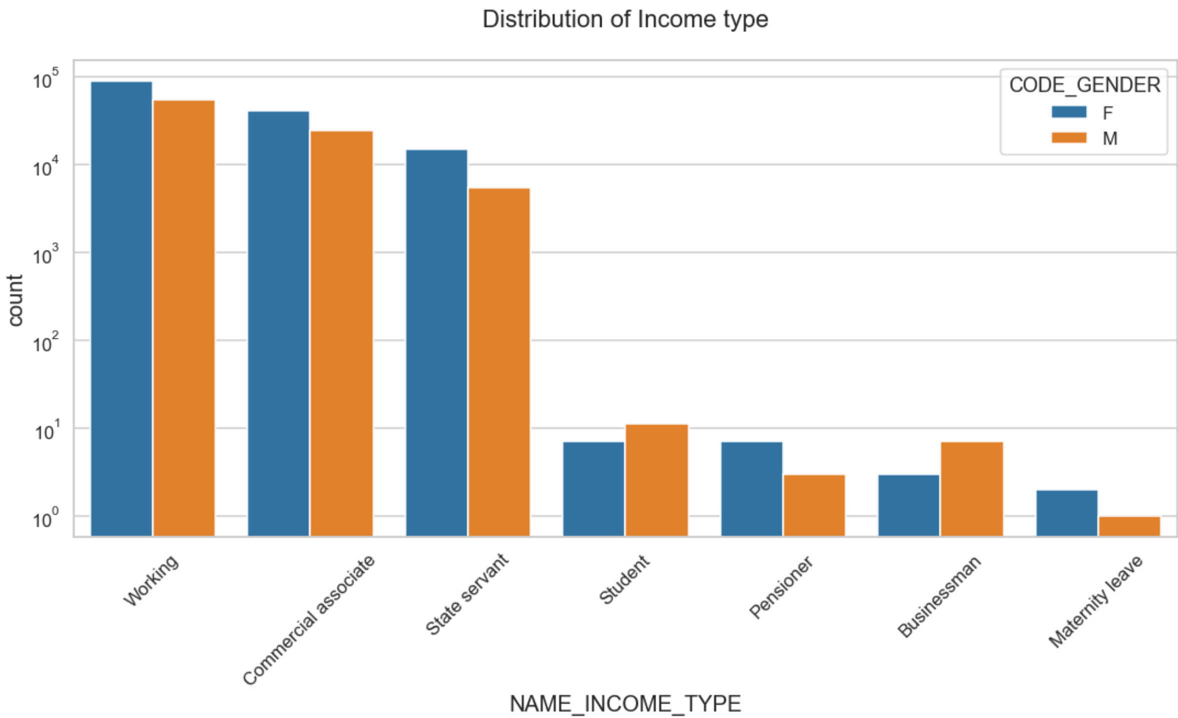


Insights:

1) Female count is more than male count 2) Income range between 100000-200000 has the highest number of credits 3) Count is considerably less beyond 400000

In [24]:

```
#plotting the graph for income type
uniplot(target0_app_data,col='NAME_INCOME_TYPE',title='Distribution of Income type',hue='CO
```

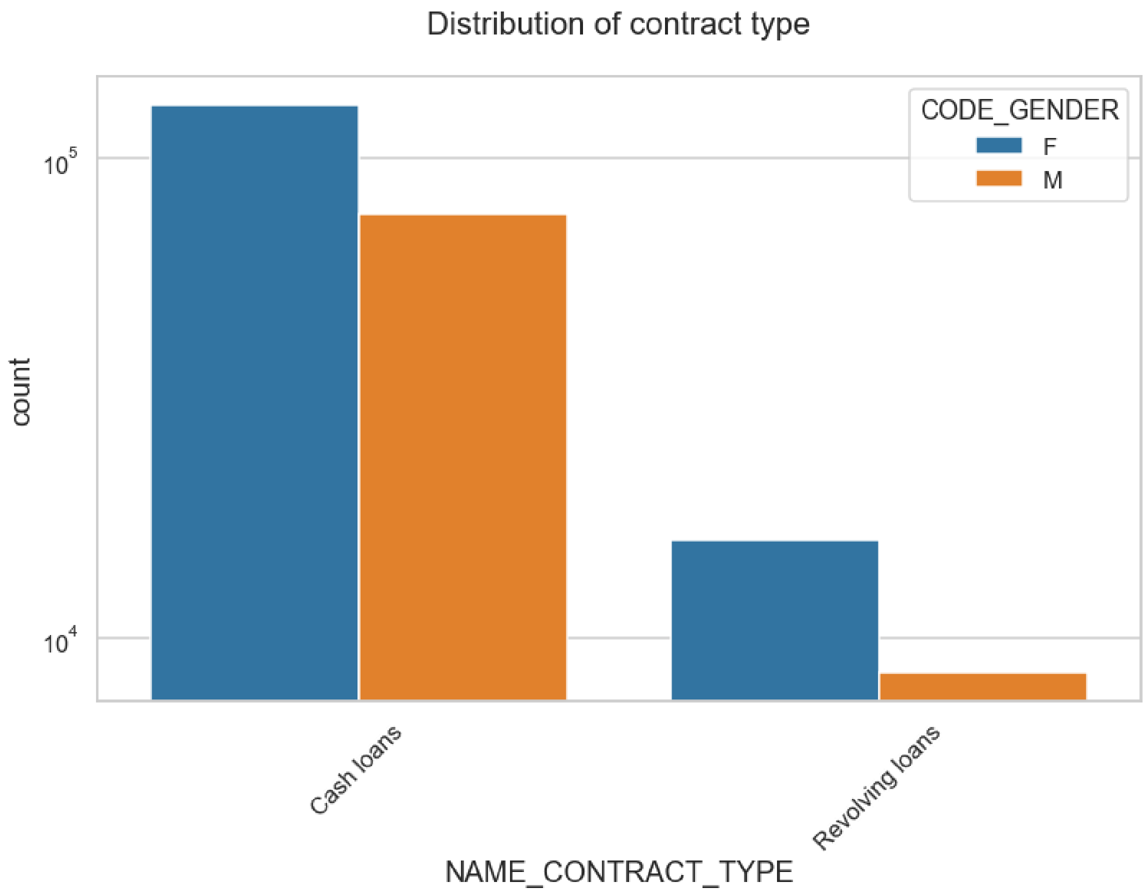


Insights:

1) Working, Commercial associate and state servant have the highest credit count 2) females have more credit counts than males

In [25]:

```
#plotting graph for contract type
uniplot(target0_app_data,col='NAME_CONTRACT_TYPE',title='Distribution of contract type',hue
```

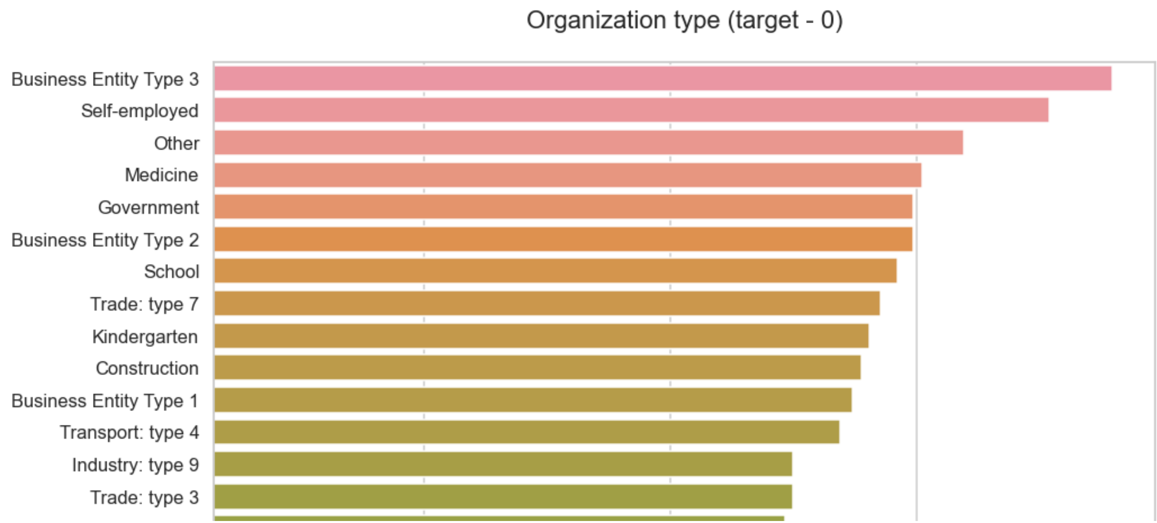

Distribution of contract type
CODE_GENDER
F
M
10^5
10^4
count
Cash loans
Revolving loans
NAME_CONTRACT_TYPE

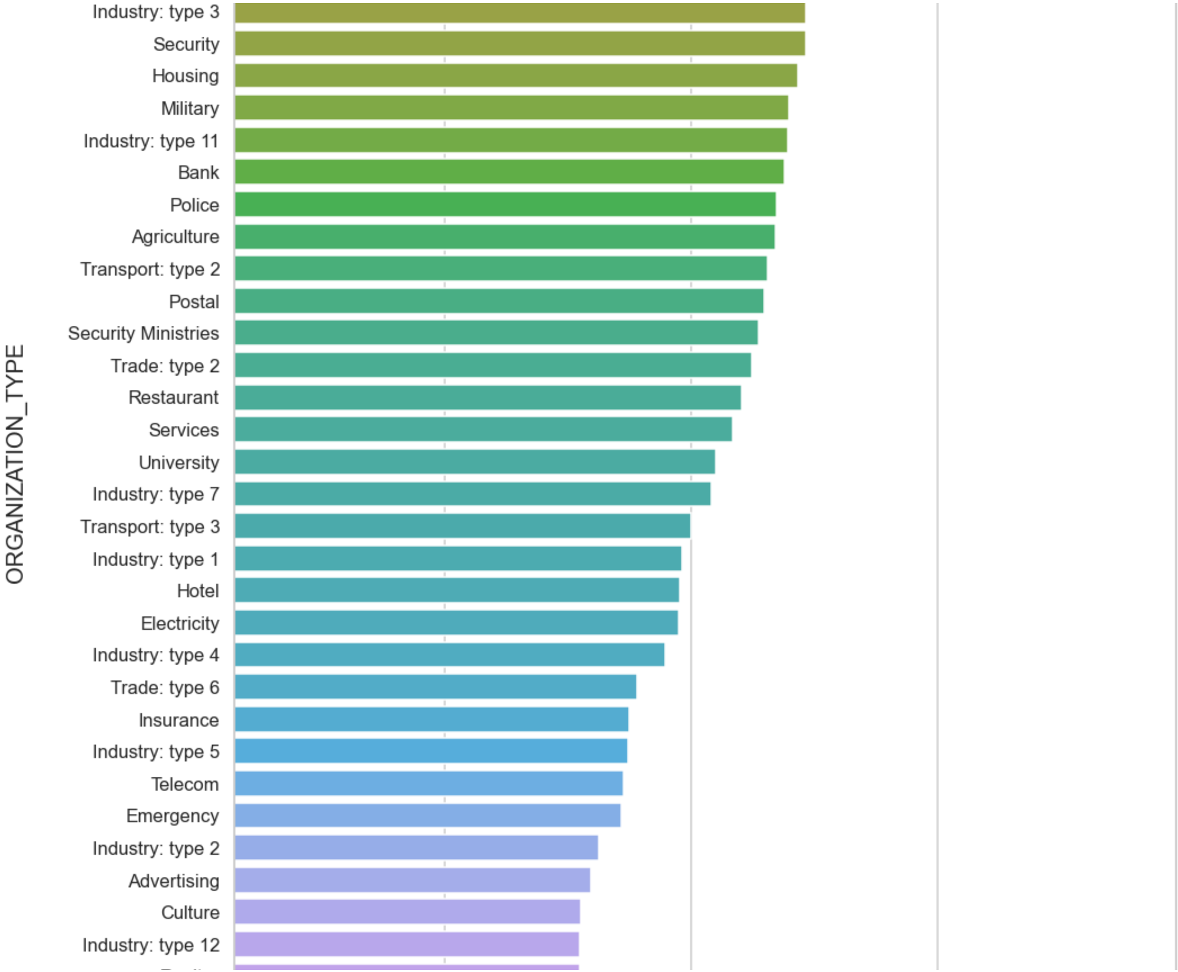
Insights:

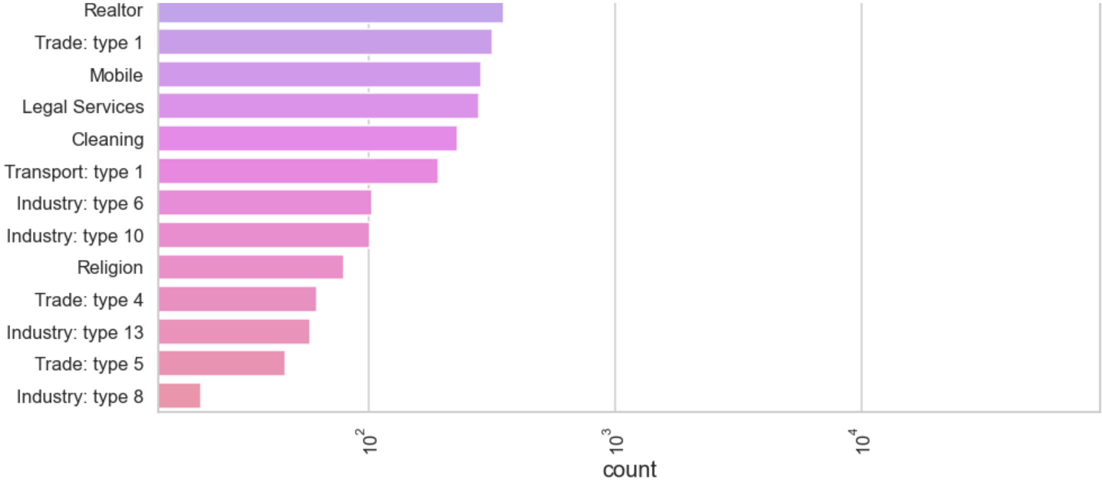
1) Cash loans have more credits 2) Females received more loans

In [27]:

```
#plotting for organization type
plt.figure(figsize=(15,30))
plt.title("Organization type (target - 0)")

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

sns.countplot(data=target0_app_data,y='ORGANIZATION_TYPE',order=target0_app_data['ORGANIZAT

plt.show()
```



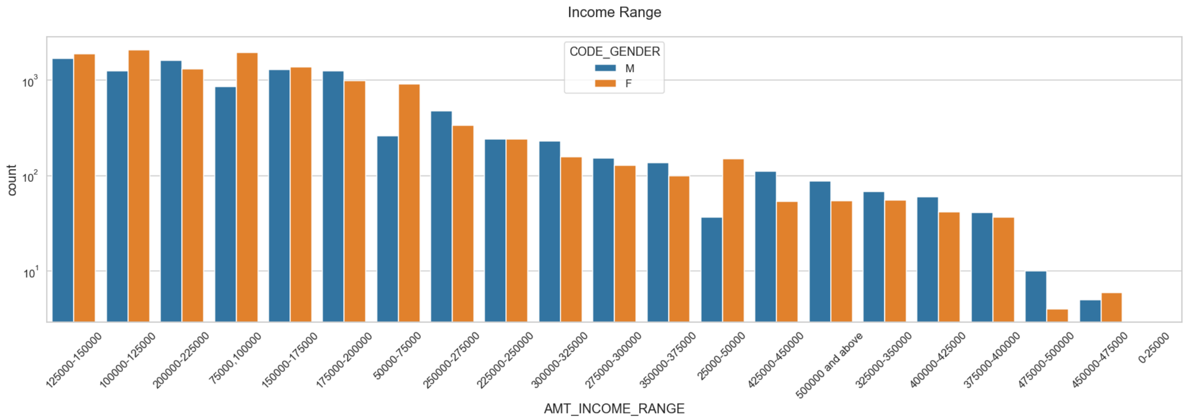

ORGANIZATION_TYPE
Industry: type 3
Security
Housing
Military
Industry: type 11
Bank
Police
Agriculture
Transport: type 2
Postal
Security Ministries
Trade: type 2
Restaurant
Services
University
Industry: type 7
Transport: type 3
Industry: type 1
Hotel
Electricity
Industry: type 4
Trade: type 6
Insurance
Industry: type 5
Telecom
Emergency
Industry: type 2
Advertising
Culture
Industry: type 12



Insights: Clients are mostly from ‘Business entity Type 3’ , ‘Self employed’, ‘Other’ , ‘Medicine’ and ‘Government’

In [28]:

```
#plotting for income range(target1)
uniplot(target1_app_data,col='AMT_INCOME_RANGE',title='Income Range',hue='CODE_GENDER')
```

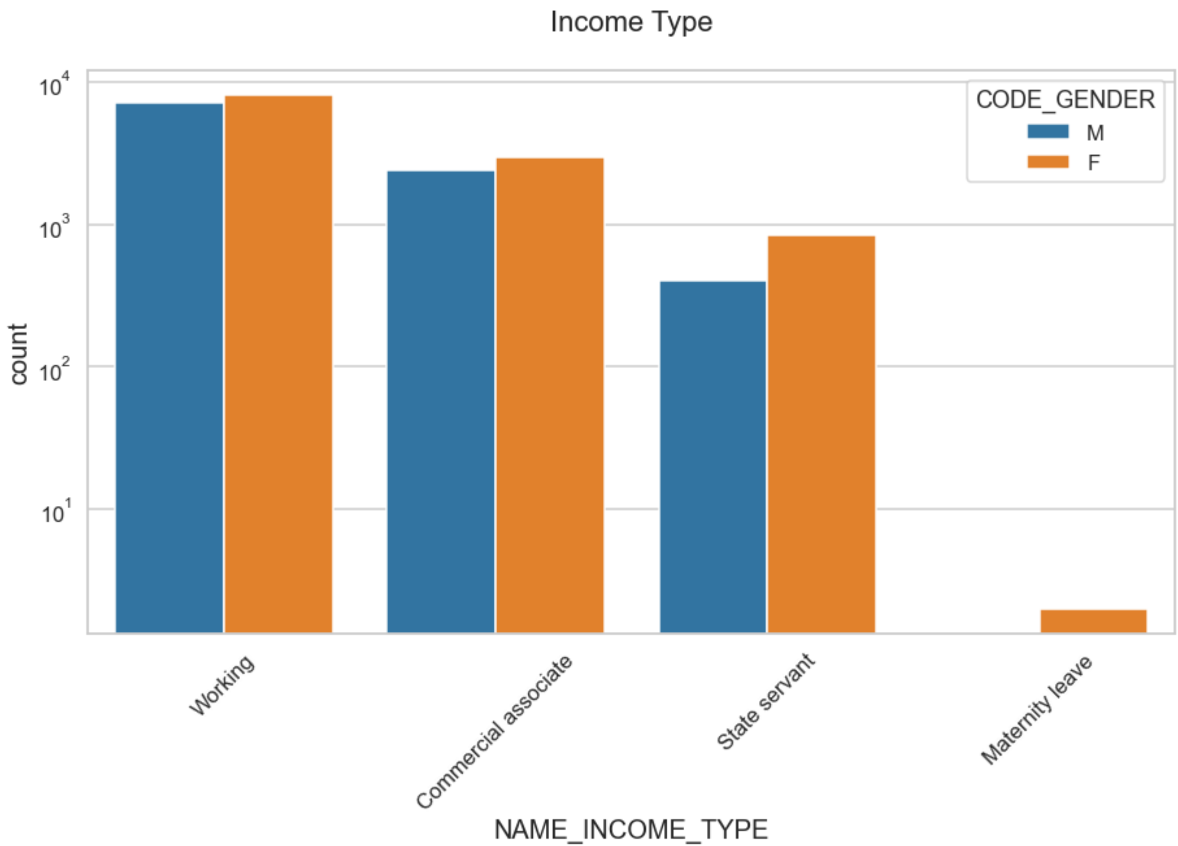


Insights:

1) Male credit counts are more than female 2) The income range 100000-200000 has highest number of credits

In [29]:

```
#plotting for income type
uniplot(target1_app_data,col='NAME_INCOME_TYPE',title='Income Type',hue='CODE_GENDER')
```

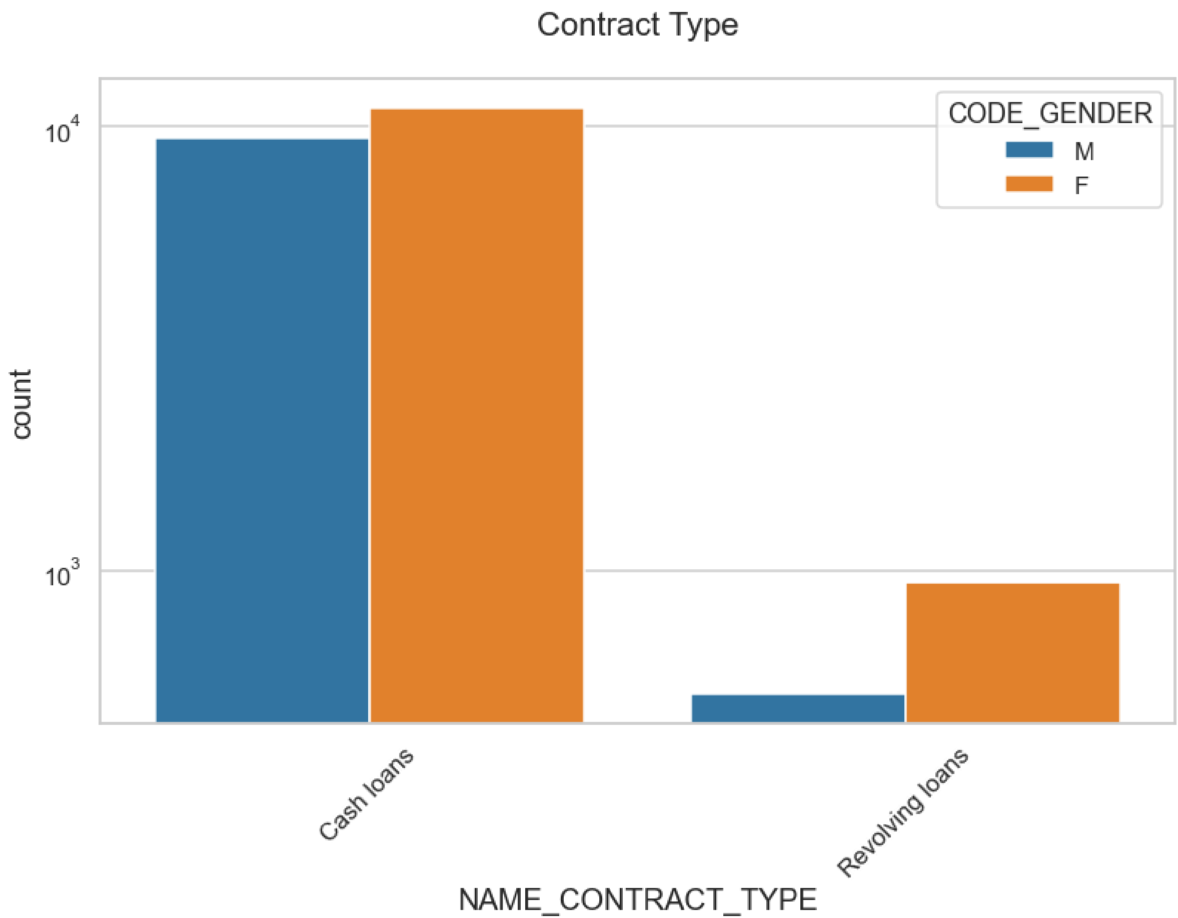

Income Type
count
10^4
10^3
10^2
10^1
CODE_GENDER
M
F
Working
Commercial associate
State servant
Maternity leave
NAME_INCOME_TYPE

Insights:

1) working, commercial associate and state servant have highest number of credits 2) females have more credit counts than males 3) the columns student, businessman and pensioner are not present hence we can conclude that they are not defaulters

In [30]:

```
#plotting for contract type
uniplot(target1_app_data,col='NAME_CONTRACT_TYPE',title='Contract Type',hue='CODE_GENDER')
```

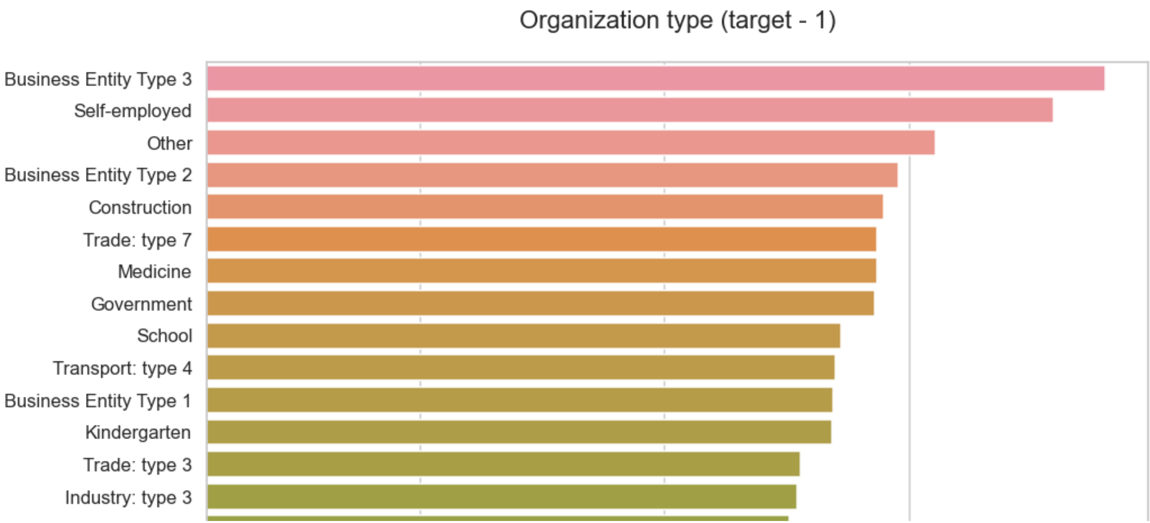

Contract Type
CODE_GENDER
M
F
10^4
10^3
count
Cash loans
Revolving loans
NAME_CONTRACT_TYPE

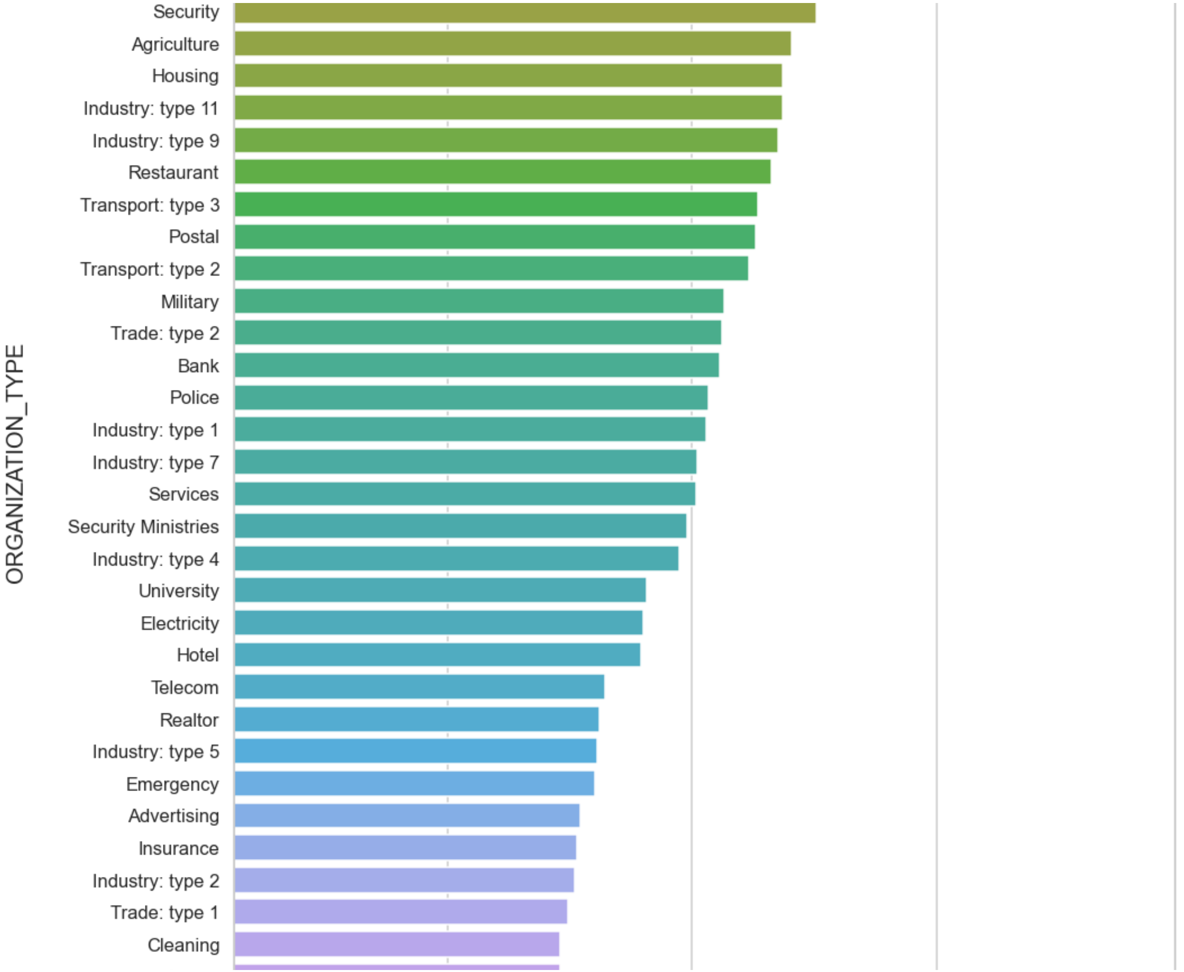
Insights:

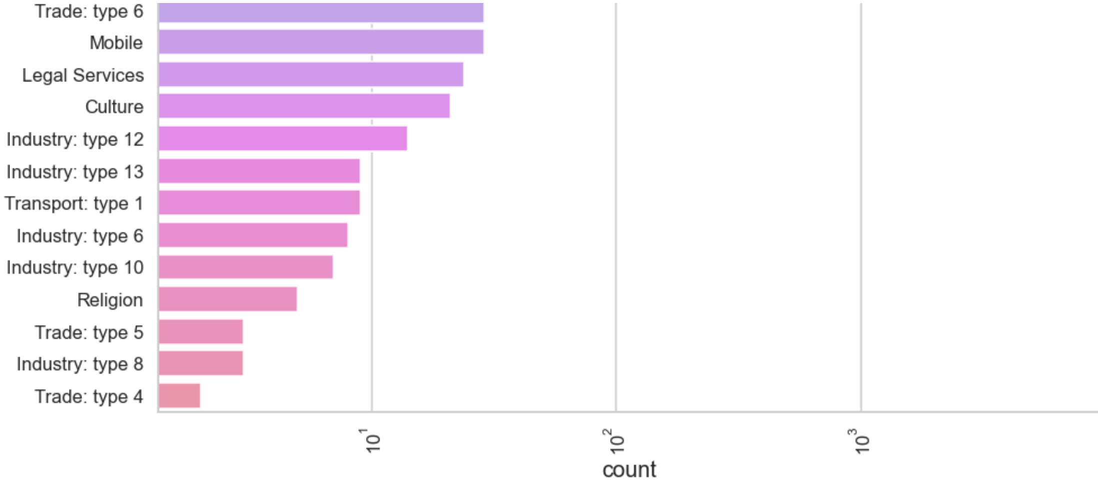
1) cash loans have the highest number of credits 2) females are given more loans

In [31]:

```
#plotting for oragnization type
plt.figure(figsize=(15,30))
plt.title("Organization type (target - 1)")

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

sns.countplot(data=target1_app_data,y='ORGANIZATION_TYPE',order=target1_app_data['ORGANIZAT

plt.show()
```




ORGANIZATION_TYPE
Security
Agriculture
Housing
Industry: type 11
Industry: type 9
Restaurant
Transport: type 3
Postal
Transport: type 2
Military
Trade: type 2
Bank
Police
Industry: type 1
Industry: type 7
Services
Security Ministries
Industry: type 4
University
Electricity
Hotel
Telecom
Realtor
Industry: type 5
Emergency
Advertising
Insurance
Industry: type 2
Trade: type 1
Cleaning



Insight: Clients are mostly from ‘Business entity Type 3’ , ‘Self employed’ , ‘Other’

In [33]:

```
#correlation between target 0 and target1
target0_corr=target0_app_data.iloc[0:,2:]
target1_corr=target1_app_data.iloc[0:,2:]

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

In [34]:

```
target0
```

Out[34]:

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_
CNT_CHILDREN	1.000000	-0.009826	-0.018704	
AMT_INCOME_TOTAL	-0.009826	1.000000	0.326155	
AMT_CREDIT	-0.018704	0.326155	1.000000	
AMT_ANNUITY	-0.007612	0.400752	0.762103	
REGION_POPULATION_RELATIVE	-0.030352	0.169306	0.103876	
DAYS_BIRTH	0.242462	-0.045543	-0.152659	
DAYS_EMPLOYED	0.063036	-0.030102	-0.087500	
DAYS_REGISTRATION	0.162900	0.034508	-0.015180	
DAYS_ID_PUBLISH	-0.117746	-0.026462	-0.034914	
HOUR_APPR_PROCESS_START	-0.033031	0.055934	0.040390	
REG_REGION_NOT_LIVE_REGION	-0.023033	0.064868	0.020979	
REG_REGION_NOT_WORK_REGION	-0.016798	0.129765	0.050597	
LIVE_REGION_NOT_WORK_REGION	-0.006946	0.121288	0.052028	

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_
REG_CITY_NOT_LIVE_CITY	-0.001566	-0.004264	-0.037527	
REG_CITY_NOT_WORK_CITY	0.010369	-0.020260	-0.038517	
LIVE_CITY_NOT_WORK_CITY	0.018414	-0.011238	-0.014834	

In [35]:

```
target1
```

Out[35]:

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AM
CNT_CHILDREN	1.000000	0.001872	-0.002074	
AMT_INCOME_TOTAL	0.001872	1.000000	0.036484	
AMT_CREDIT	-0.002074	0.036484	1.000000	
AMT_ANNUITY	0.015653	0.043358	0.748708	
REGION_POPULATION_RELATIVE	-0.032019	0.008476	0.069220	
DAYS_BIRTH	0.176563	-0.007822	-0.189512	
DAYS_EMPLOYED	0.032627	-0.000039	-0.106003	
DAYS_REGISTRATION	0.126411	-0.003959	-0.033250	
DAYS_ID_PUBLISH	-0.089861	-0.008858	-0.062405	
HOUR_APPR_PROCESS_START	-0.038923	0.012520	0.029054	
REG_REGION_NOT_LIVE_REGION	-0.032465	0.006951	0.020083	
REG_REGION_NOT_WORK_REGION	-0.039498	0.013245	0.035695	
LIVE_REGION_NOT_WORK_REGION	-0.028031	0.012287	0.035966	

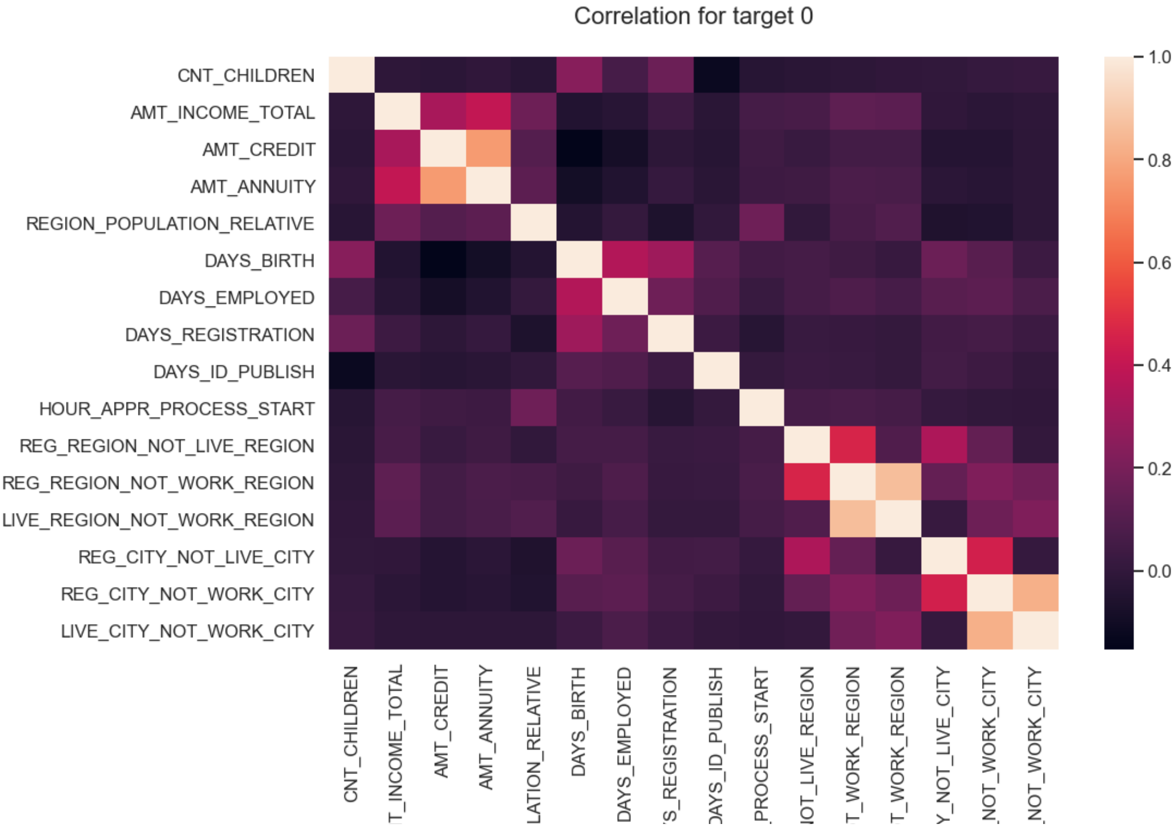
	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AM
REG_CITY_NOT_LIVE_CITY	-0.019278	-0.003664	-0.035325	
REG_CITY_NOT_WORK_CITY	-0.000876	-0.006886	-0.041392	
LIVE_CITY_NOT_WORK_CITY	0.016332	-0.004401	-0.017875	

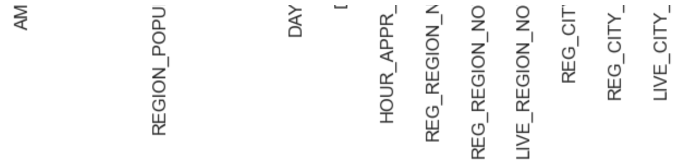
In [36]:

```
#plotting the above correlation
def target_corr(df1,title):
    plt.figure(figsize=(15, 10))
    sns.heatmap(df1,annot=False)

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

In [37]:

```
target_corr(df1=target0,title='Correlation for target 0')
```



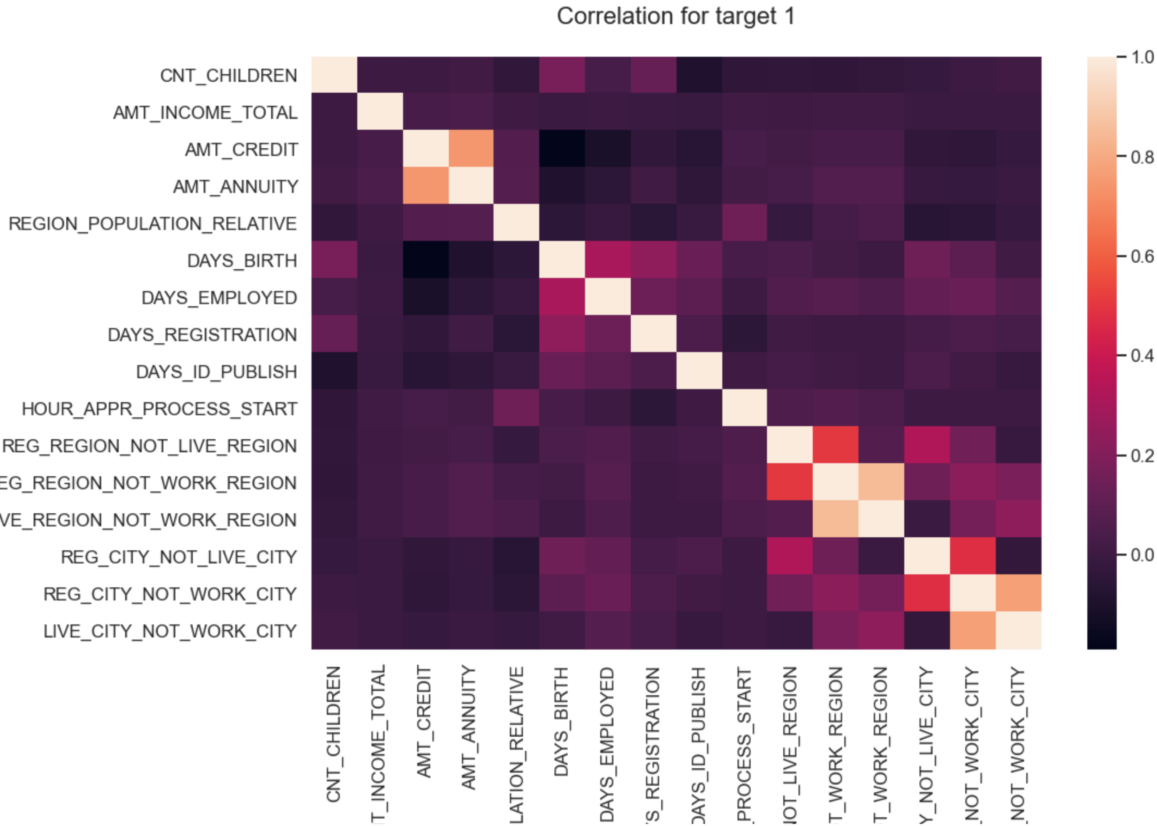


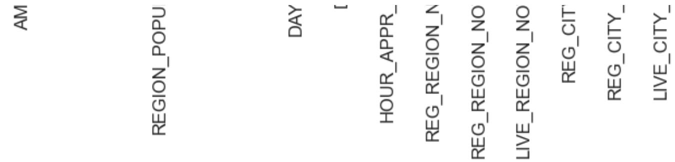
Insights:

1) Credit amount is inversely proportional to the number of children a client has 2) Credit amount is inversely proportional to the date of birth 3) Income amount is inversely proportional to the number of children a client has

In [38]:

```
target_corr(df1=target1,title='Correlation for target 1')
```





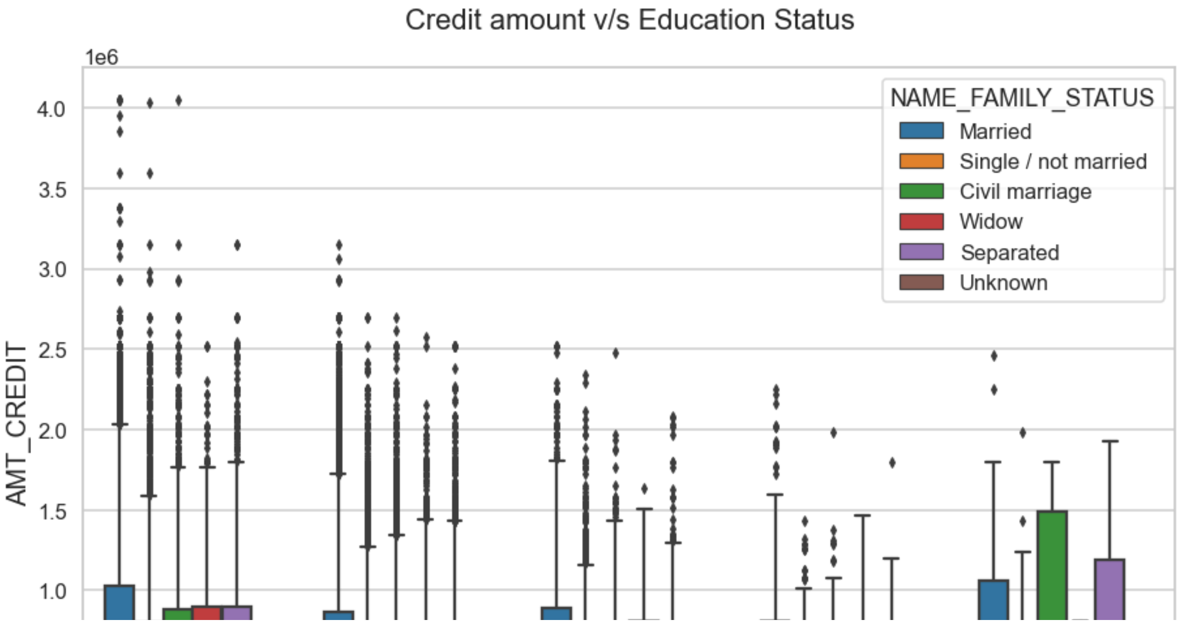
Insights:

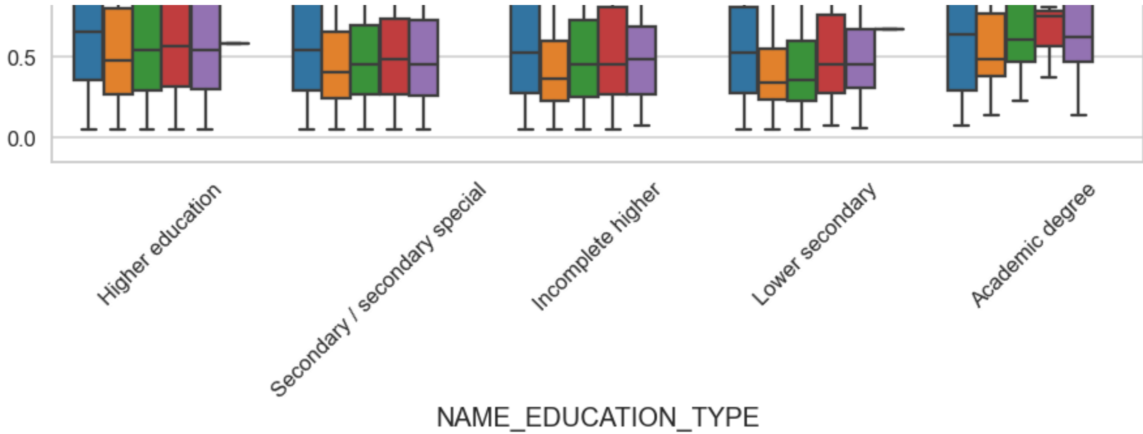
1) The client's whose permanent address that does not match work address have less number of children 2) The client's whose permanent address that does not match contact address have less number of children

Bivariate Analysis for variables

In [57]:

```
#for target0
#plotting credit amount
plt.figure(figsize=(15,10))
plt.xticks(rotation=45)
sns.boxplot(data =target0_app_data, x='NAME_EDUCATION_TYPE',y='AMT_CREDIT', hue ='NAME_FAMI
plt.title('Credit amount v/s Education Status')
plt.show()
```

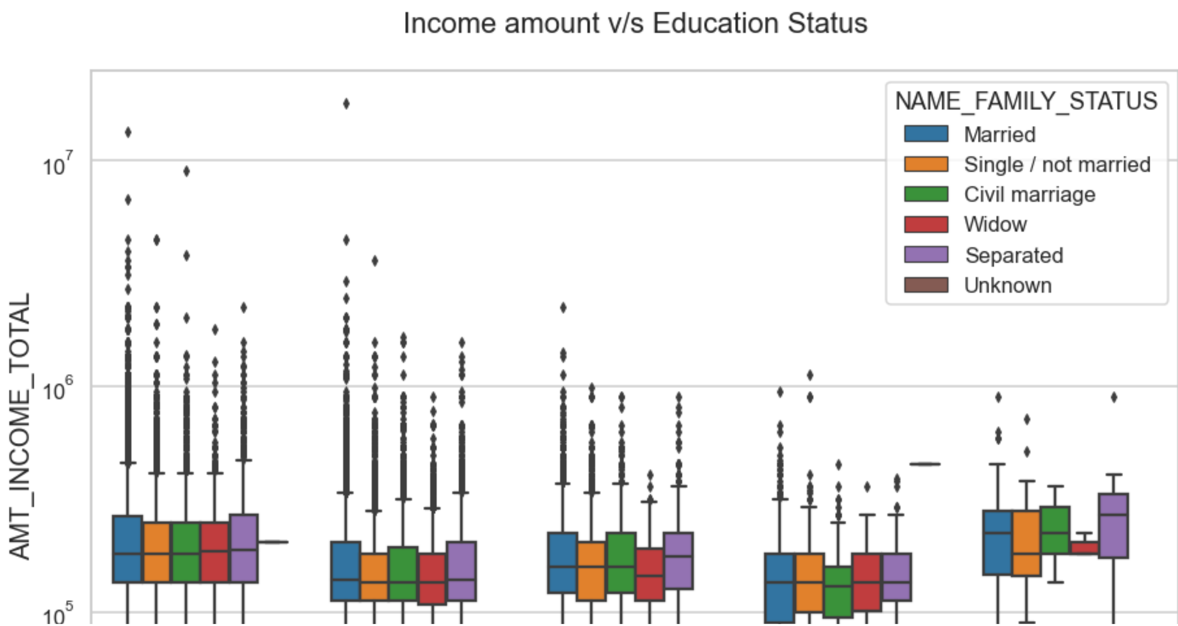


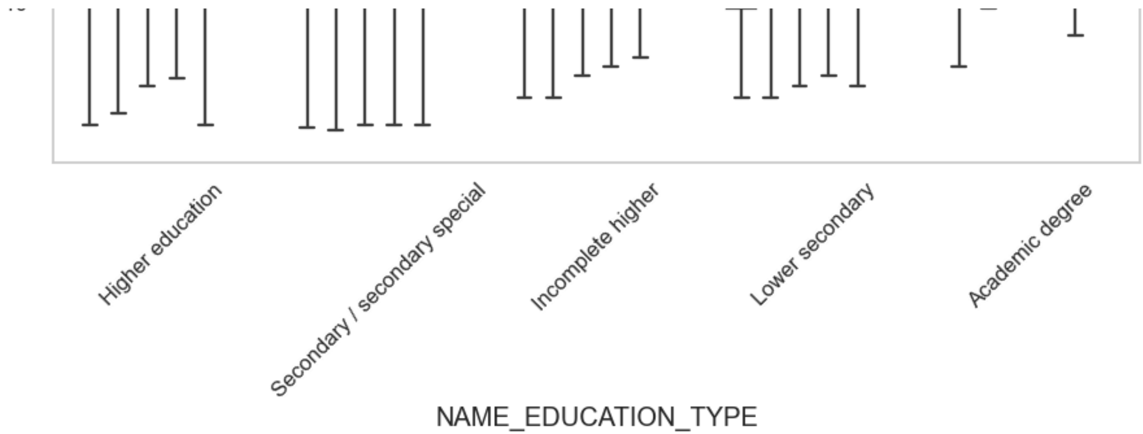


Insight: Family status like 'civil marriage', 'married' and 'separated' of Academic degree education have higher number of credits. Family status of 'marriage', 'single' and 'civil marriage' with higher education has more number of outliers

In [58]:

```
#plotting income amount
plt.figure(figsize=(15,10))
plt.xticks(rotation=45)
plt.yscale('log')
sns.boxplot(data =target0_app_data, x='NAME_EDUCATION_TYPE',y='AMT_INCOME_TOTAL', hue ='NAM
plt.title('Income amount v/s Education Status')
plt.show()
```

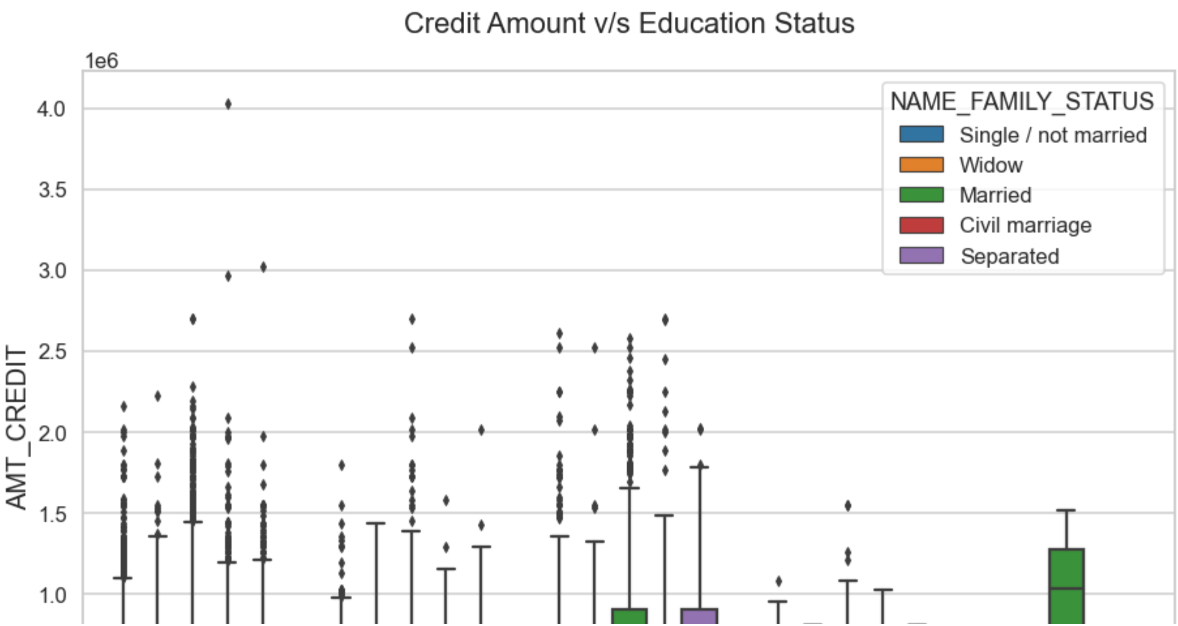


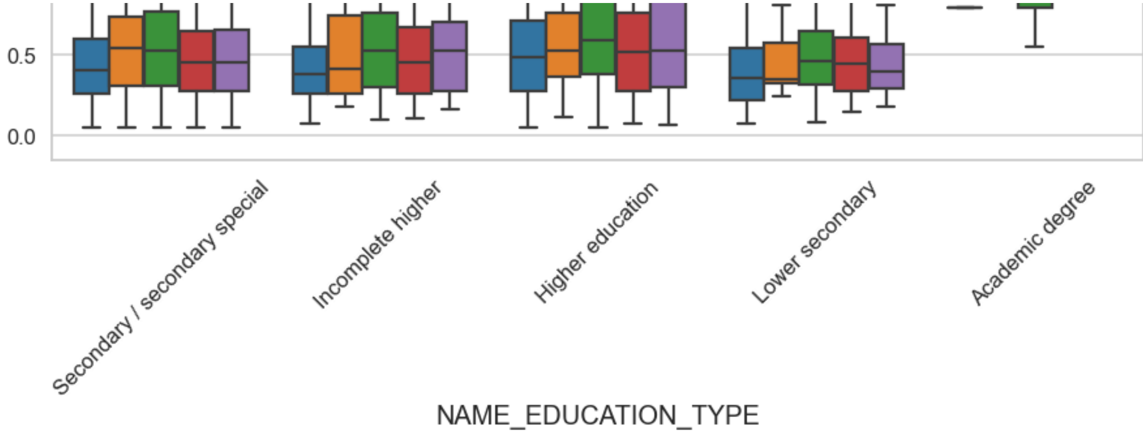


Insights: This plot has similar inferences as the one before

In [59]:

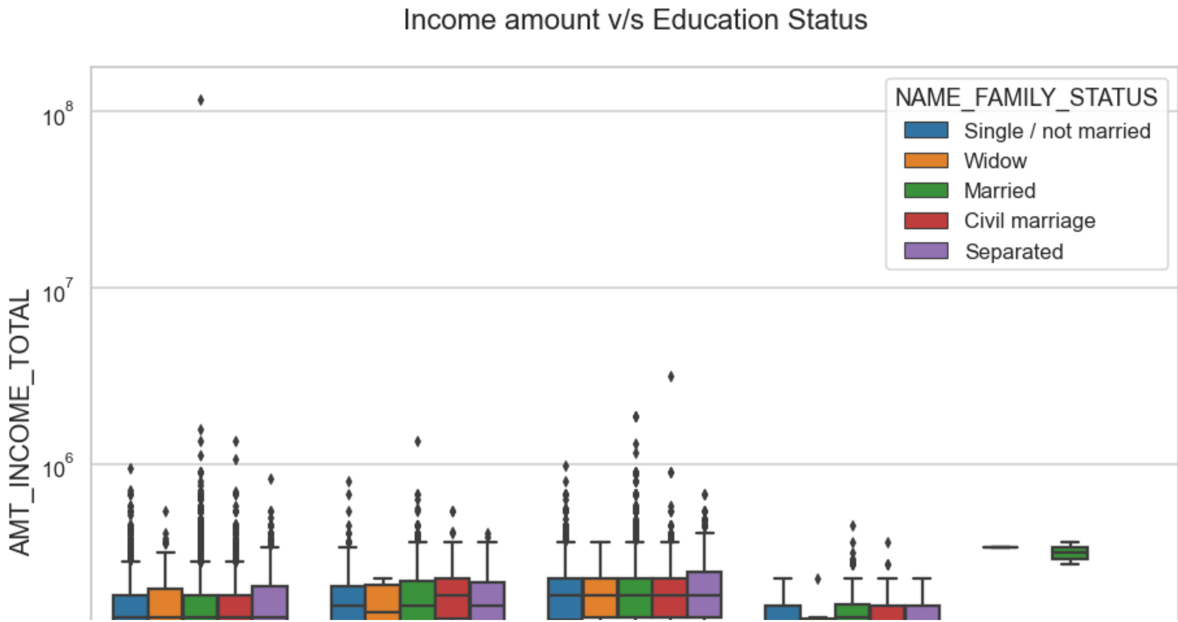
```
#for target1
#plotting for credit amount
plt.figure(figsize=(15,10))
plt.xticks(rotation=45)
sns.boxplot(data =target1_app_data, x='NAME_EDUCATION_TYPE',y='AMT_CREDIT', hue ='NAME_FAMI
plt.title('Credit Amount v/s Education Status')
plt.show()
```

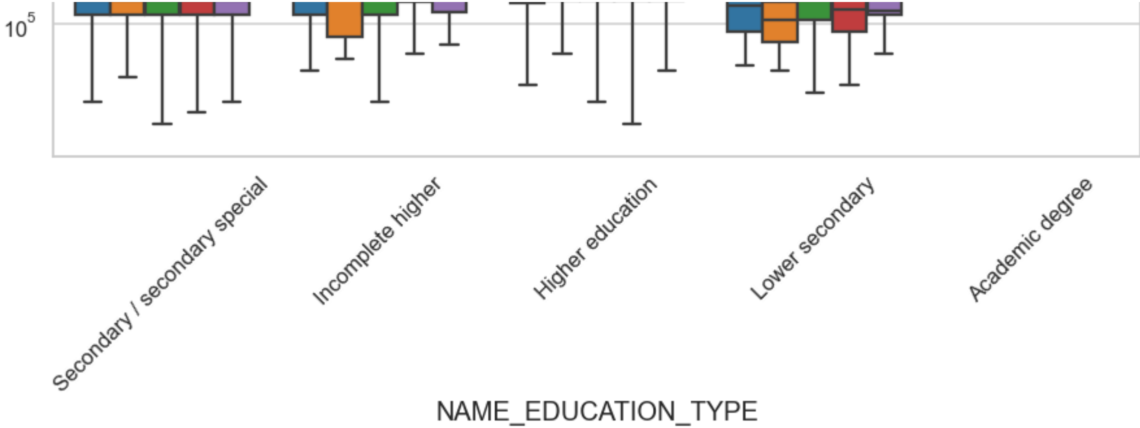



0.5
0.0
Secondary / secondary special
Incomplete higher
Higher education
Lower secondary
Academic degree
NAME_EDUCATION_TYPE

In [61]:

```
#plotting for income amount
plt.figure(figsize=(15,10))
plt.xticks(rotation=45)
plt.yscale('log')
sns.boxplot(data =target1_app_data, x='NAME_EDUCATION_TYPE',y='AMT_INCOME_TOTAL', hue ='NAM
plt.title('Income amount v/s Education Status')
plt.show()
```





Loading Previous Application dataset

In [62]:

```
prev_app=pd.read_csv('previous_application.csv')
```

In [63]:

```
#finding columns with more than 30% null values
nullcol1=prev_app.isnull().sum()
nullcol1=nullcol1[nullcol1.values>(0.3*len(nullcol1))]
len(nullcol1)
```

Out[63]:

```
15
```

In [64]:

```
#removing those 15 columns
nullcol1 = list(nullcol1[nullcol1.values>=0.3].index)
prev_app.drop(labels=nullcol1,axis=1,inplace=True)
```

In [66]:

```
prev_app.shape
```

Out[66]:

```
(1670214, 22)
```

In [67]:

```
#removing the columns with the values 'XNA' and 'XPA'
prev_app=prev_app.drop(prev_app[prev_app['NAME_CASH_LOAN_PURPOSE']=='XNA'].index)
prev_app=prev_app.drop(prev_app[prev_app['NAME_CASH_LOAN_PURPOSE']=='XAP'].index)
```

In [68]:

```
prev_app.shape
```

Out[68]:

```
(69635, 22)
```

In [74]:

```
#merging both the datasets
df=pd.merge(left=app_data,right=prev_app,how='inner',on='SK_ID_CURR',suffixes='_x' )
```

```
<ipython-input-74-e773199dd44b>:2: FutureWarning: Passing 'suffixes' as a <c
lass 'str'>, is not supported and may give unexpected results. Provide 'suff
ixes' as a tuple instead. In the future a 'TypeError' will be raised.
  df=pd.merge(left=app_data,right=prev_app,how='inner',on='SK_ID_CURR',suffi
xes='_x' )
```

In [75]:

```
# Renaming the column names after merging

df = df.rename({'NAME_CONTRACT_TYPE_' : 'NAME_CONTRACT_TYPE','AMT_CREDIT_':'AMT_CREDIT','AM
                         'WEEKDAY_APPR_PROCESS_START_' : 'WEEKDAY_APPR_PROCESS_START',
                         'HOUR_APPR_PROCESS_START_':'HOUR_APPR_PROCESS_START','NAME_CONTRAC
                         'AMT_CREDITx':'AMT_CREDIT_PREV','AMT_ANNUITYx':'AMT_ANNUITY_PREV',
                         'WEEKDAY_APPR_PROCESS_STARTx':'WEEKDAY_APPR_PROCESS_START_PREV',
                         'HOUR_APPR_PROCESS_STARTx':'HOUR_APPR_PROCESS_START_PREV'}, axis=1
```

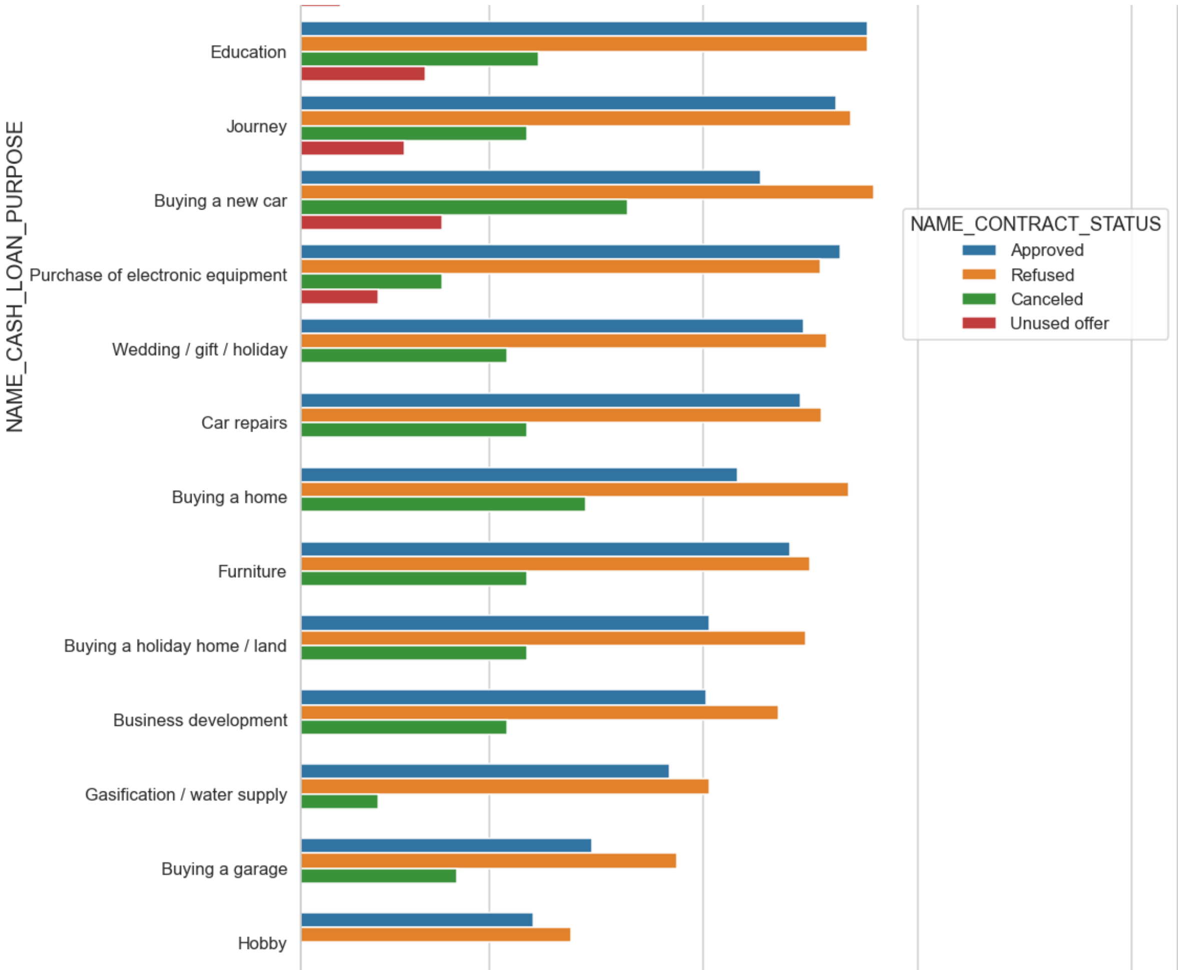
In [77]:

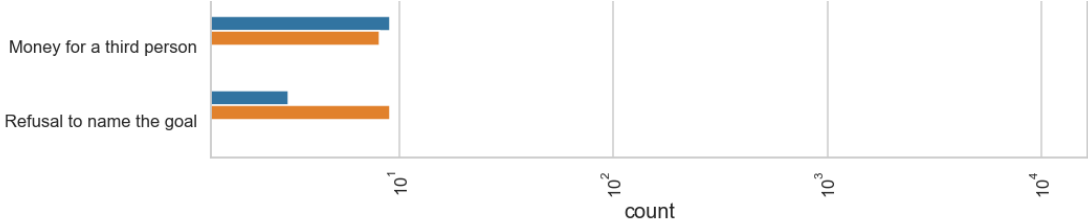
```
#removing unwanted columns
df.drop(['SK_ID_CURR','WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START','REG_REGION_N
              'REG_REGION_NOT_WORK_REGION','LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIV
              'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY','WEEKDAY_APPR_PROCESS_STA
              'HOUR_APPR_PROCESS_START_PREV', 'FLAG_LAST_APPL_PER_CONTRACT','NFLAG_LAST_APP
```

Univariate analysis

In [79]:

```
plt.figure(figsize=(15,30))
plt.xticks(rotation=90)
plt.xscale('log')
plt.title('Contract Status')
ax = sns.countplot(data = df, y= 'NAME_CASH_LOAN_PURPOSE',
    order=df['NAME_CASH_LOAN_PURPOSE'].value_counts().index,hue = 'NAME_CONTRACT_STATUS')
```


NAME_CASH_LOAN_PURPOSE
Education
Journey
Buying a new car
Purchase of electronic equipment
Wedding / gift / holiday
Car repairs
Buying a home
Furniture
Buying a holiday home / land
Business development
Gasification / water supply
Buying a garage
Hobby
NAME_CONTRACT_STATUS
Approved
Refused
Canceled
Unused offer

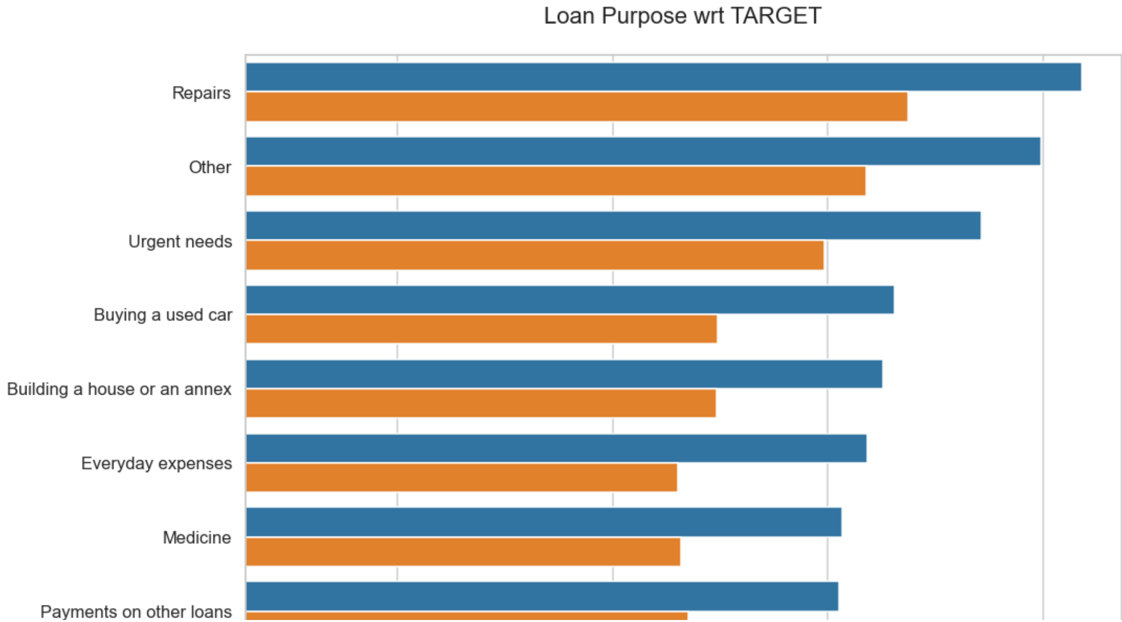


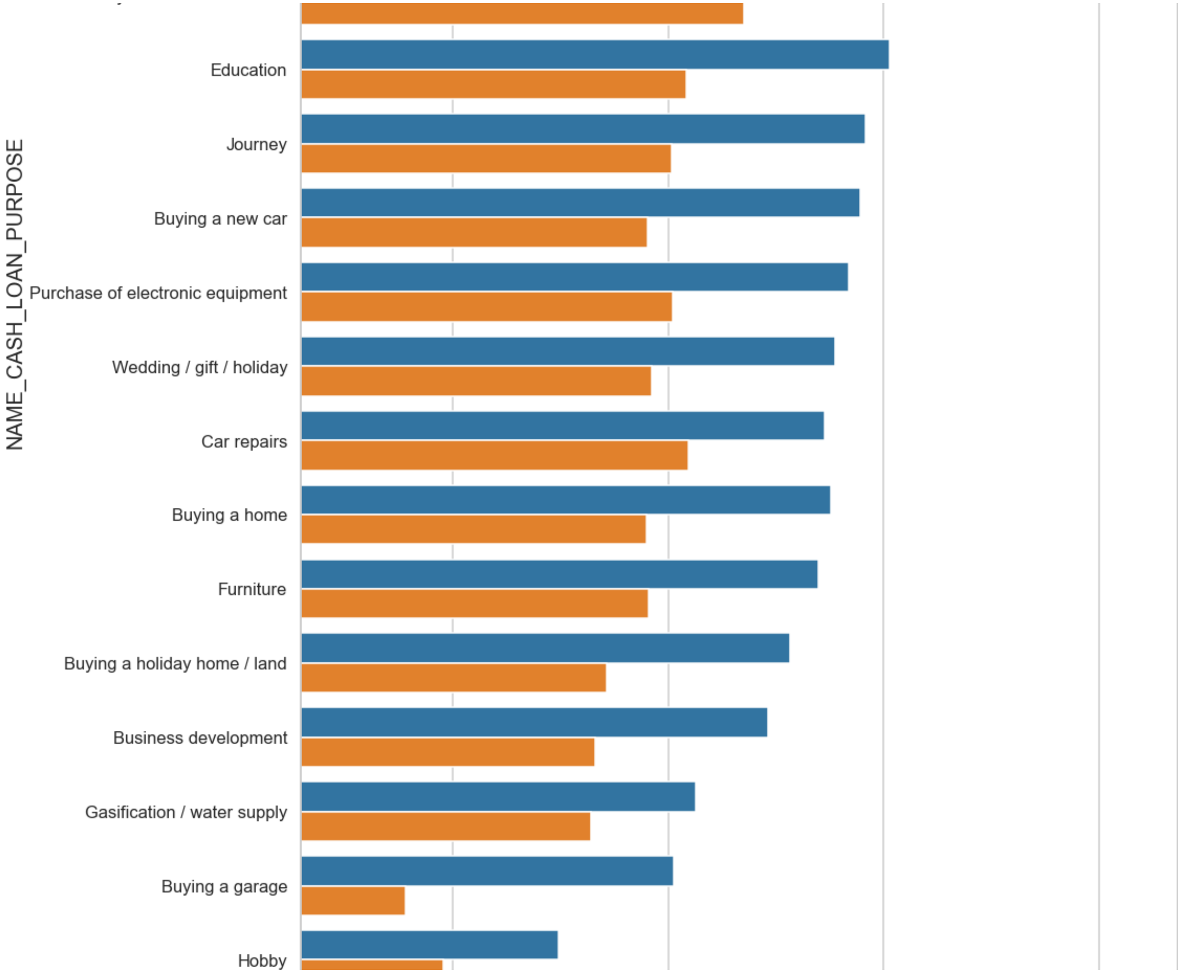
Insights:

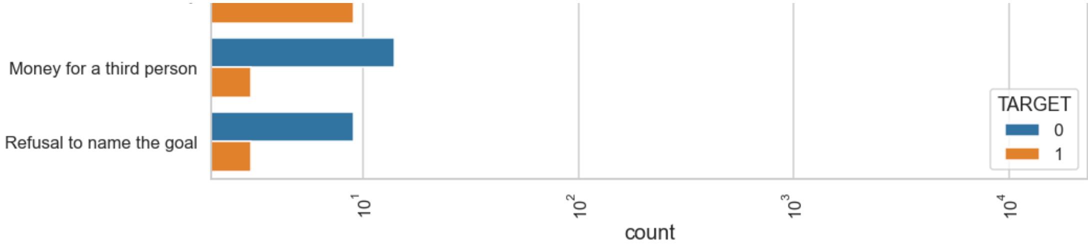
1) Most of the loans were rejected due to repairs

In [80]:

```
#loan purpose
plt.figure(figsize=(15,30))
plt.xticks(rotation=90)
plt.xscale('log')
plt.title('Loan Purpose wrt TARGET')
ax = sns.countplot(data = df, y= 'NAME_CASH_LOAN_PURPOSE',
                   order=df['NAME_CASH_LOAN_PURPOSE'].value_counts().index,hue = 'TARGET')
```




NAME_CASH_LOAN_PURPOSE
Education
Journey
Buying a new car
Purchase of electronic equipment
Wedding / gift / holiday
Car repairs
Buying a home
Furniture
Buying a holiday home / land
Business development
Gasification / water supply
Buying a garage
Hobby

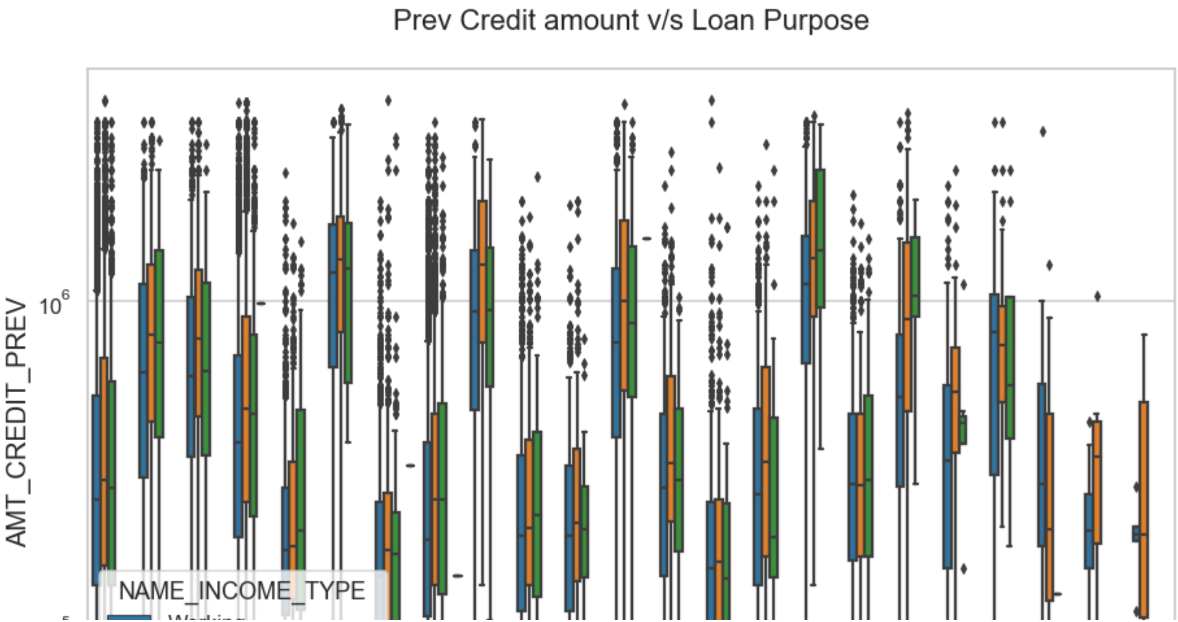


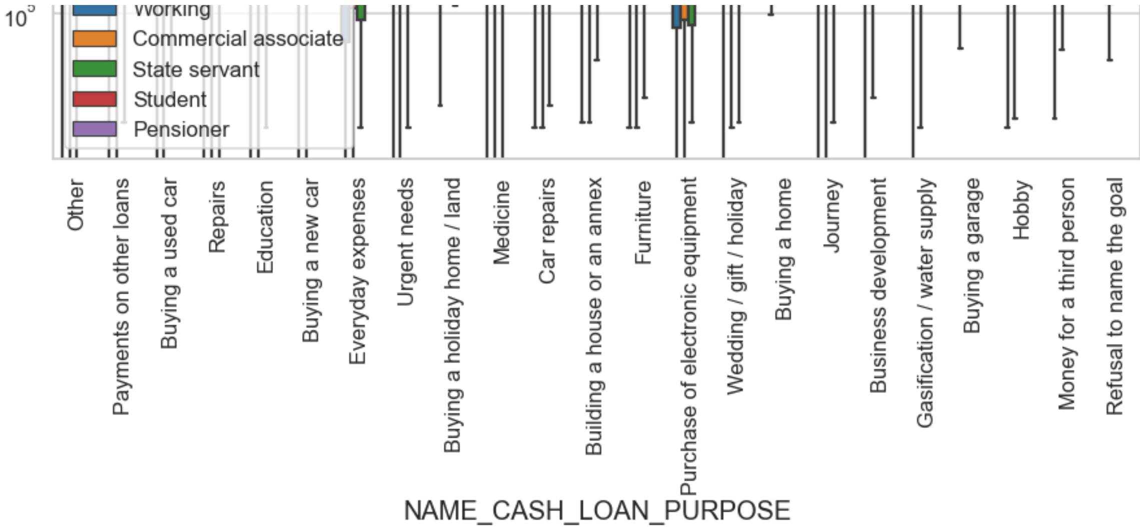
Insights: loans which have the purpose as 'repairs' end up having a delay in payment. Clients with purposes such as 'Buying a garage', 'Business development', 'Buying land','Buying a new car' and 'Education' have less difficulty in repaying the loan on time hence such loans can be approved

Bivariate Analysis

In [81]:

```
#for credit amount vs loan purpose
plt.figure(figsize=(15,10))
plt.xticks(rotation=90)
plt.yscale('log')
sns.boxplot(data =df, x='NAME_CASH_LOAN_PURPOSE',hue='NAME_INCOME_TYPE',y='AMT_CREDIT_PREV'
plt.title('Prev Credit amount v/s Loan Purpose')
plt.show()
```



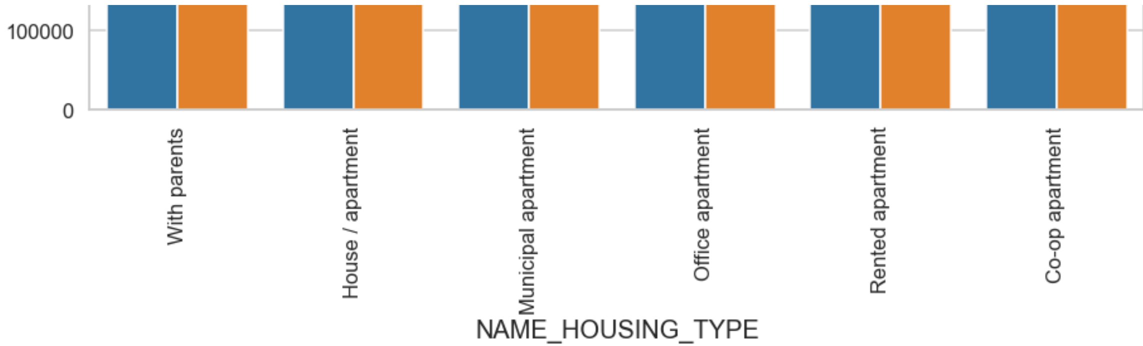


Insights:

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) Credit count is significantly high for income type of 'state servants'

In [83]:

```
#for prev credit amount vs housing type
plt.figure(figsize=(15,10))
plt.xticks(rotation=90)
sns.barplot(data =df, y='AMT_CREDIT_PREV',hue='TARGET',x='NAME_HOUSING_TYPE')
plt.title('Prev Credit amount v/s Housing type')
plt.show()
```



'office apartment' has higher credit wrt target0 and 'co-op apartment' has higher credit wrt to target1. Hence we can conclude that avoid giving loans for housing type='co-op apartment' as they have a delay in payment.

In [ ]: