Team 5 - Advanced Analytics using Statistics Exam

EDA on Bank Loan Application Dataset

Team Members

1. Tushar Shirsath : 220940325083
2. Saurav Sharma : 220940325063
3. Srujak Gedam : 220940325078
4. Somesh Rewadkar : 220940325075
5. Ayush Singh : 220940325018

In [36]:

```
# Import libraries

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

Importing Dataset

In [37]:

```
df1 = pd.read_csv("application_data (1).csv")
df1.head()
```

Out[37]:

	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 × 122 columns

Understanding the Dataset

Inspect the dataframe for dimensions, null-values, and summary of different numeric columns.

In [38]:

```
# Check the number of rows and columns in the dataset
df1.shape
```

Out[38]:

```
(307511, 122)
```

- **The dataset has 307511 rows and 122 columns**

In [39]:

```
# Checking the numeric variables of the dataframes

df1.describe(include='all')
```

Out[39]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_C
count	307511.000000	307511.000000	307511	307511	307
unique	NaN	NaN	2	3	
top	NaN	NaN	Cash loans	F	
freq	NaN	NaN	278232	202448	202!
mean	278180.518577	0.080729	NaN	NaN	N
std	102790.175348	0.272419	NaN	NaN	N
min	100002.000000	0.000000	NaN	NaN	N
25%	189145.500000	0.000000	NaN	NaN	N
50%	278202.000000	0.000000	NaN	NaN	N
75%	367142.500000	0.000000	NaN	NaN	N
max	456255.000000	1.000000	NaN	NaN	N

11 rows × 122 columns

In [54]:

```
#Checking information of all the columns like data types

df1.info("all")
```

```
 67    LANDAREA_MODE                  float64
 68    LIVINGAPARTMENTS_MODE          float64
 69    LIVINGAREA_MODE                float64
 70    NONLIVINGAPARTMENTS_MODE       float64
 71    NONLIVINGAREA_MODE             float64
 72    APARTMENTS_MEDI                float64
 73    BASEMENTAREA_MEDI              float64
 74    YEARS_BEGINEXPLUATATION_MEDI   float64
 75    YEARS_BUILD_MEDI               float64
 76    COMMONAREA_MEDI                float64
 77    ELEVATORS_MEDI                 float64
 78    ENTRANCES_MEDI                 float64
 79    FLOORSMAX_MEDI                 float64
 80    FLOORSMIN_MEDI                 float64
 81    LANDAREA_MEDI                  float64
 82    LIVINGAPARTMENTS_MEDI          float64
 83    LIVINGAREA_MEDI                float64
 84    NONLIVINGAPARTMENTS_MEDI       float64
 85    NONLIVINGAREA_MEDI             float64
 86    FONDKAPREMONT_MODE             object
```

- **In dataset there are 122 columns having data-type object, int and float**

Handling NULL Values

In [40]:

```
#checking how many null values are present in each of the columns

#creating a function to find null values for the dataframe
def null_values(df):
    return round((df.isnull().sum()*100/len(df)).sort_values(ascending = False),2)
```

In [56]:

```
null_values(df1).head(30)
```

Out[56]:

```
COMMONAREA_MEDI             69.87
COMMONAREA_AVG              69.87
COMMONAREA_MODE             69.87
NONLIVINGAPARTMENTS_MODE    69.43
NONLIVINGAPARTMENTS_AVG     69.43
NONLIVINGAPARTMENTS_MEDI    69.43
FONDKAPREMONT_MODE          68.39
LIVINGAPARTMENTS_MODE       68.35
LIVINGAPARTMENTS_AVG        68.35
LIVINGAPARTMENTS_MEDI       68.35
FLOORSMIN_AVG               67.85
FLOORSMIN_MODE              67.85
FLOORSMIN_MEDI              67.85
YEARS_BUILD_MEDI            66.50
YEARS_BUILD_MODE            66.50
YEARS_BUILD_AVG             66.50
OWN_CAR_AGE                 65.99
LANDAREA_MEDI               59.38
LANDAREA_MODE               59.38
LANDAREA_AVG                59.38
BASEMENTAREA_MEDI           58.52
BASEMENTAREA_AVG            58.52
BASEMENTAREA_MODE           58.52
EXT_SOURCE_1                56.38
NONLIVINGAREA_MODE          55.18
NONLIVINGAREA_AVG           55.18
NONLIVINGAREA_MEDI          55.18
ELEVATORS_MEDI              53.30
ELEVATORS_AVG               53.30
ELEVATORS_MODE              53.30
dtype: float64
```

Dealing with Null values more than 40 %

In [41]:

```
#Displaying null columns having missing values more than 40%

null_40 = null_values(df1)[null_values(df1)>40]
print(null_40,'\n')
print("There are over",len(null_40),"columns having greater than 40%")
```

```
COMMONAREA_MEDI                 69.87
COMMONAREA_AVG                  69.87
COMMONAREA_MODE                 69.87
NONLIVINGAPARTMENTS_MODE        69.43
NONLIVINGAPARTMENTS_AVG         69.43
NONLIVINGAPARTMENTS_MEDI        69.43
FONDKAPREMONT_MODE              68.39
LIVINGAPARTMENTS_MODE           68.35
LIVINGAPARTMENTS_AVG            68.35
LIVINGAPARTMENTS_MEDI           68.35
FLOORSMIN_AVG                   67.85
FLOORSMIN_MODE                  67.85
FLOORSMIN_MEDI                  67.85
YEARS_BUILD_MEDI                66.50
YEARS_BUILD_MODE                66.50
YEARS_BUILD_AVG                 66.50
OWN_CAR_AGE                     65.99
LANDAREA_MEDI                   59.38
LANDAREA_MODE                   59.38
LANDAREA_AVG                    59.38
BASEMENTAREA_MEDI               58.52
BASEMENTAREA_AVG                58.52
BASEMENTAREA_MODE               58.52
EXT_SOURCE_1                    56.38
NONLIVINGAREA_MODE              55.18
NONLIVINGAREA_AVG               55.18
NONLIVINGAREA_MEDI              55.18
ELEVATORS_MEDI                  53.30
ELEVATORS_AVG                   53.30
ELEVATORS_MODE                  53.30
WALLSMATERIAL_MODE              50.84
APARTMENTS_MEDI                 50.75
APARTMENTS_AVG                  50.75
APARTMENTS_MODE                 50.75
ENTRANCES_MEDI                  50.35
ENTRANCES_AVG                   50.35
ENTRANCES_MODE                  50.35
LIVINGAREA_AVG                  50.19
LIVINGAREA_MODE                 50.19
LIVINGAREA_MEDI                 50.19
HOUSETYPE_MODE                  50.18
FLOORSMAX_MODE                  49.76
FLOORSMAX_MEDI                  49.76
FLOORSMAX_AVG                   49.76
YEARS_BEGINEXPLUATATION_MODE    48.78
YEARS_BEGINEXPLUATATION_MEDI    48.78
YEARS_BEGINEXPLUATATION_AVG     48.78
TOTALAREA_MODE                  48.27
EMERGENCYSTATE_MODE             47.40
dtype: float64

There are over 49 columns having greater than 40%
```

In [42]:

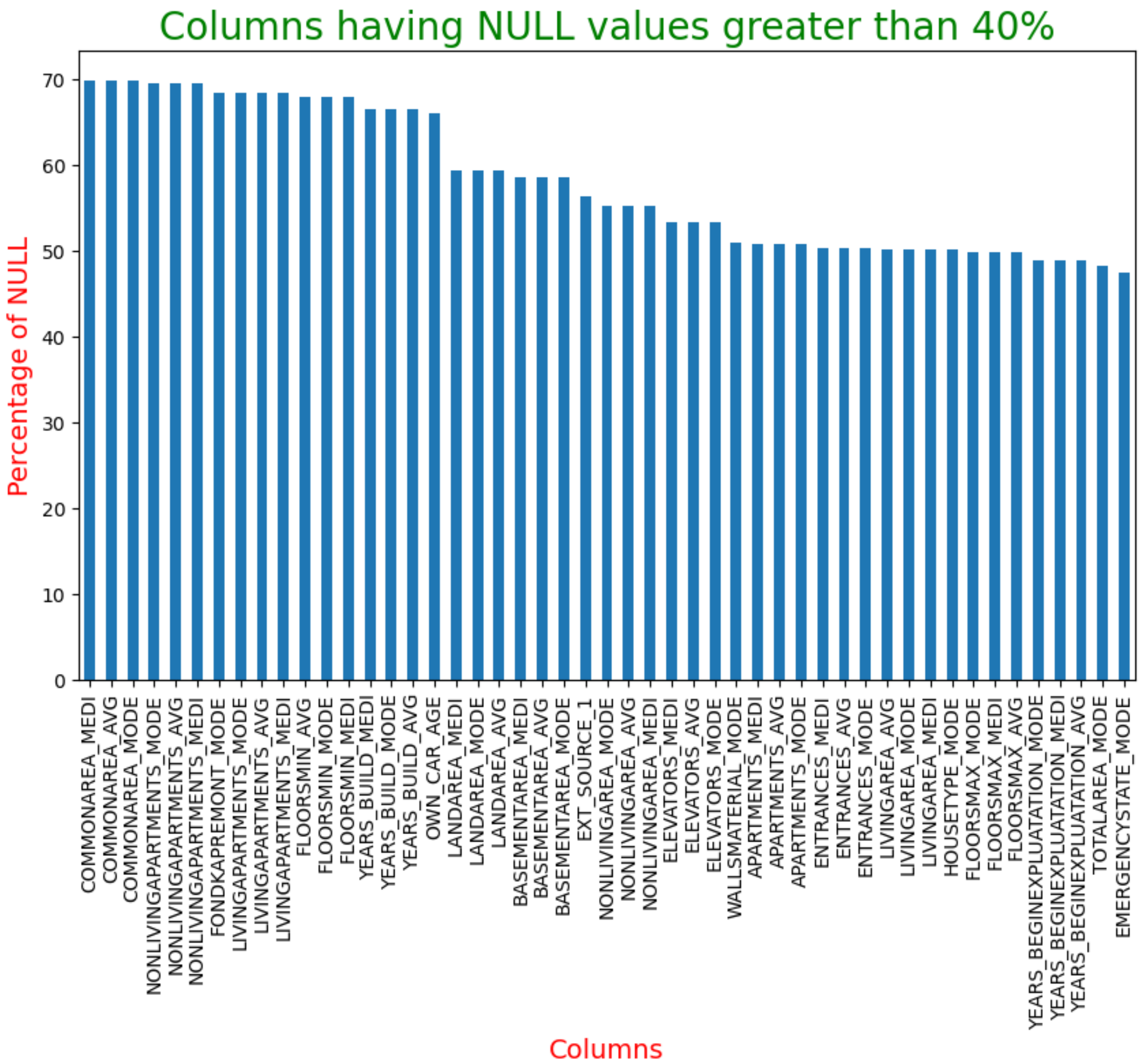
```
null_40.index
```

Out[42]:

```
Index(['COMMONAREA_MEDI', 'COMMONAREA_AVG', 'COMMONAREA_MODE',
       'NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAPARTMENTS_AVG',
       'NONLIVINGAPARTMENTS_MEDI', 'FONDKAPREMONT_MODE',
       'LIVINGAPARTMENTS_MODE', 'LIVINGAPARTMENTS_AVG',
       'LIVINGAPARTMENTS_MEDI', 'FLOORSMIN_AVG', 'FLOORSMIN_MODE',
       'FLOORSMIN_MEDI', 'YEARS_BUILD_MEDI', 'YEARS_BUILD_MODE',
       'YEARS_BUILD_AVG', 'OWN_CAR_AGE', 'LANDAREA_MEDI', 'LANDAREA_MODE',
       'LANDAREA_AVG', 'BASEMENTAREA_MEDI', 'BASEMENTAREA_AVG',
       'BASEMENTAREA_MODE', 'EXT_SOURCE_1', 'NONLIVINGAREA_MODE',
       'NONLIVINGAREA_AVG', 'NONLIVINGAREA_MEDI', 'ELEVATORS_MEDI',
       'ELEVATORS_AVG', 'ELEVATORS_MODE', 'WALLSMATERIAL_MODE',
       'APARTMENTS_MEDI', 'APARTMENTS_AVG', 'APARTMENTS_MODE',
       'ENTRANCES_MEDI', 'ENTRANCES_AVG', 'ENTRANCES_MODE', 'LIVINGAREA_AV
G',
       'LIVINGAREA_MODE', 'LIVINGAREA_MEDI', 'HOUSETYPE_MODE',
       'FLOORSMAX_MODE', 'FLOORSMAX_MEDI', 'FLOORSMAX_AVG',
       'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BEGINEXPLUATATION_MEDI',
       'YEARS_BEGINEXPLUATATION_AVG', 'TOTALAREA_MODE', 'EMERGENCYSTATE_MOD
E'],
      dtype='object')
```

In [44]:

```
#Bar Chart for null values
plt.figure(figsize = (10,6))
null_40.plot.bar()
plt.title('Columns having NULL values greater than 40%',fontdict={'fontsize':20,'color':'Gr
plt.xlabel('Columns',fontdict={'fontsize':14,'fontweight':7,'color':'Red'})
plt.ylabel('Percentage of NULL',fontdict={'fontsize':14,'fontweight':7,'color':'Red'})

plt.show()
```



In [45]:

```
#Dropping the columns
df1.drop(labels = null_40.index,axis=1,inplace=True)
```

In [46]:

```
df1.shape
```

Out[46]:

```
(307511, 73)
```

In [ ]:

```

```

In [47]:

```
# Checking the null percentage in new columns
new_null = df1.isnull().sum()/len(df1)*100
new_null[new_null.values>0]
```

Out[47]:

```
AMT_ANNUITY                    0.003902
AMT_GOODS_PRICE                0.090403
NAME_TYPE_SUITE                0.420148
OCCUPATION_TYPE               31.345545
CNT_FAM_MEMBERS                0.000650
EXT_SOURCE_2                   0.214626
EXT_SOURCE_3                  19.825307
OBS_30_CNT_SOCIAL_CIRCLE       0.332021
DEF_30_CNT_SOCIAL_CIRCLE       0.332021
OBS_60_CNT_SOCIAL_CIRCLE       0.332021
DEF_60_CNT_SOCIAL_CIRCLE       0.332021
DAYS_LAST_PHONE_CHANGE         0.000325
AMT_REQ_CREDIT_BUREAU_HOUR    13.501631
AMT_REQ_CREDIT_BUREAU_DAY     13.501631
AMT_REQ_CREDIT_BUREAU_WEEK    13.501631
AMT_REQ_CREDIT_BUREAU_MON     13.501631
AMT_REQ_CREDIT_BUREAU_QRT     13.501631
AMT_REQ_CREDIT_BUREAU_YEAR    13.501631
dtype: float64
```

In [48]:

```
df1.shape
```

Out[48]:

```
(307511, 73)
```

In [49]:

```
df1[df1.AMT_ANNUITY.isna()]
```

Out[49]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FI
47531	155054	0	Cash loans	M	N	
50035	157917	0	Cash loans	F	N	
51594	159744	0	Cash loans	F	N	
55025	163757	0	Cash loans	F	N	
59934	169487	0	Cash loans	M	Y	
75873	187985	0	Cash loans	M	Y	
89343	203726	0	Cash loans	F	Y	
123872	243648	0	Cash loans	F	N	
207186	340147	0	Cash loans	M	N	
227939	364022	0	Cash loans	F	N	
239329	377174	0	Cash loans	F	N	
241835	379997	0	Cash loans	F	N	

12 rows × 73 columns

In [50]:

```
#Droping records with AMT_ANNUITY as the percentage of records missing are very less
df2 = df1[~df1.AMT_ANNUITY.isna()].copy()
```

In [51]:

```
df2.shape
```

Out[51]:

```
(307499, 73)
```

In [52]:

```
new_null = df2.isnull().sum()/len(df2)*100
new_null[new_null.values>0]
```

Out[52]:

```
AMT_GOODS_PRICE               0.090407
NAME_TYPE_SUITE               0.420164
OCCUPATION_TYPE              31.346769
CNT_FAM_MEMBERS               0.000650
EXT_SOURCE_2                  0.214635
EXT_SOURCE_3                 19.825756
OBS_30_CNT_SOCIAL_CIRCLE      0.332034
DEF_30_CNT_SOCIAL_CIRCLE      0.332034
OBS_60_CNT_SOCIAL_CIRCLE      0.332034
DEF_60_CNT_SOCIAL_CIRCLE      0.332034
DAYS_LAST_PHONE_CHANGE        0.000325
AMT_REQ_CREDIT_BUREAU_HOUR   13.501833
AMT_REQ_CREDIT_BUREAU_DAY    13.501833
AMT_REQ_CREDIT_BUREAU_WEEK   13.501833
AMT_REQ_CREDIT_BUREAU_MON    13.501833
AMT_REQ_CREDIT_BUREAU_QRT    13.501833
AMT_REQ_CREDIT_BUREAU_YEAR   13.501833
dtype: float64
```

We can see that most of the values is equal to 0. Let's check the mode for it

In [53]:

```
df2.AMT_REQ_CREDIT_BUREAU_HOUR.value_counts()
```

Out[53]:

```
0.0    264355
1.0      1560
2.0        56
3.0         9
4.0         1
Name: AMT_REQ_CREDIT_BUREAU_HOUR, dtype: int64
```

In [54]:

```
df2.AMT_REQ_CREDIT_BUREAU_DAY.value_counts()
```

Out[54]:

```
0.0    264492
1.0      1292
2.0       106
3.0        45
4.0        26
5.0         9
6.0         8
9.0         2
8.0         1
Name: AMT_REQ_CREDIT_BUREAU_DAY, dtype: int64
```

In [55]:

```
df2.AMT_REQ_CREDIT_BUREAU_WEEK.value_counts()
```

Out[55]:

```
0.0    257448
1.0      8205
2.0       199
3.0        58
4.0        34
6.0        20
5.0        10
8.0         5
7.0         2
Name: AMT_REQ_CREDIT_BUREAU_WEEK, dtype: int64
```

In [56]:

```
df2.AMT_REQ_CREDIT_BUREAU_MON.value_counts()
```

Out[56]:

```
0.0     222227
1.0      33143
2.0       5385
3.0       1991
4.0       1076
5.0        602
6.0        343
7.0        298
9.0        206
8.0        185
10.0       132
11.0       119
12.0        77
13.0        72
14.0        40
15.0        35
16.0        23
17.0        14
18.0         6
19.0         3
24.0         1
23.0         1
27.0         1
22.0         1
Name: AMT_REQ_CREDIT_BUREAU_MON, dtype: int64
```

In [57]:

```
df2.AMT_REQ_CREDIT_BUREAU_QRT.value_counts()
```

Out[57]:

```
0.0      215409
1.0       33859
2.0       14412
3.0        1717
4.0         476
5.0          64
6.0          28
8.0           7
7.0           7
261.0         1
19.0          1
Name: AMT_REQ_CREDIT_BUREAU_QRT, dtype: int64
```

In [58]:

```
df2.AMT_REQ_CREDIT_BUREAU_YEAR.value_counts()
```

Out[58]:

```
0.0     71800
1.0     63401
2.0     50190
3.0     33628
4.0     20713
5.0     12051
6.0      6966
7.0      3869
8.0      2127
9.0      1096
12.0       30
11.0       30
10.0       22
13.0       19
14.0       10
17.0        7
15.0        6
19.0        4
18.0        4
16.0        3
25.0        1
23.0        1
22.0        1
21.0        1
20.0        1
Name: AMT_REQ_CREDIT_BUREAU_YEAR, dtype: int64
```

For all above columns we can replace it with 0

In [59]:

```
df2.AMT_REQ_CREDIT_BUREAU_HOUR.fillna(0,inplace=True)
df2.AMT_REQ_CREDIT_BUREAU_DAY.fillna(0,inplace=True)
df2.AMT_REQ_CREDIT_BUREAU_WEEK.fillna(0,inplace=True)
df2.AMT_REQ_CREDIT_BUREAU_MON.fillna(0,inplace=True)
df2.AMT_REQ_CREDIT_BUREAU_QRT.fillna(0,inplace=True)
df2.AMT_REQ_CREDIT_BUREAU_YEAR.fillna(0,inplace=True)
```

In [29]:

```
new_null = df2.isnull().sum()/len(df2)*100
new_null[new_null.values>0]
```

Out[29]:

```
AMT_GOODS_PRICE                  0.090407
NAME_TYPE_SUITE                  0.420164
OCCUPATION_TYPE                 31.346769
CNT_FAM_MEMBERS                  0.000650
EXT_SOURCE_2                     0.214635
EXT_SOURCE_3                    19.825756
YEARS_BEGINEXPLUATATION_AVG     48.780972
FLOORSMAX_AVG                   49.760812
YEARS_BEGINEXPLUATATION_MODE    48.780972
FLOORSMAX_MODE                  49.760812
YEARS_BEGINEXPLUATATION_MEDI    48.780972
FLOORSMAX_MEDI                  49.760812
TOTALAREA_MODE                  48.268450
EMERGENCYSTATE_MODE             47.398203
OBS_30_CNT_SOCIAL_CIRCLE         0.332034
DEF_30_CNT_SOCIAL_CIRCLE         0.332034
OBS_60_CNT_SOCIAL_CIRCLE         0.332034
DEF_60_CNT_SOCIAL_CIRCLE         0.332034
DAYS_LAST_PHONE_CHANGE           0.000325
dtype: float64
```

After seeing above result we can say that occupation has 31.34% null values

In [60]:

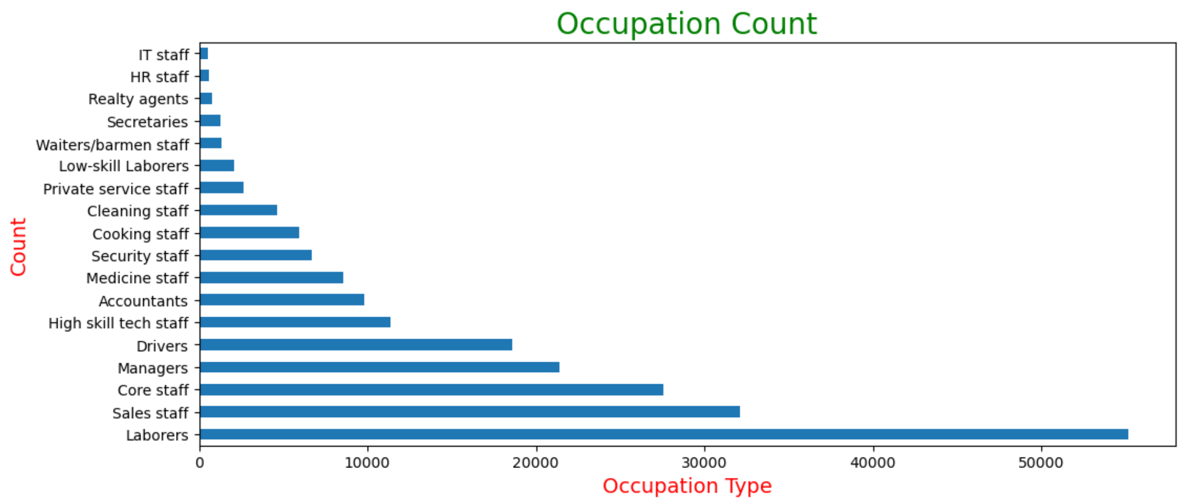
```
df2.OCCUPATION_TYPE.value_counts()
```

Out[60]:

```
Laborers                 55184
Sales staff              32101
Core staff               27569
Managers                 21370
Drivers                  18602
High skill tech staff    11379
Accountants               9812
Medicine staff            8536
Security staff            6720
Cooking staff             5945
Cleaning staff            4653
Private service staff     2652
Low-skill Laborers        2093
Waiters/barmen staff      1348
Secretaries               1304
Realty agents              751
HR staff                   563
IT staff                   526
Name: OCCUPATION_TYPE, dtype: int64
```

In [31]:

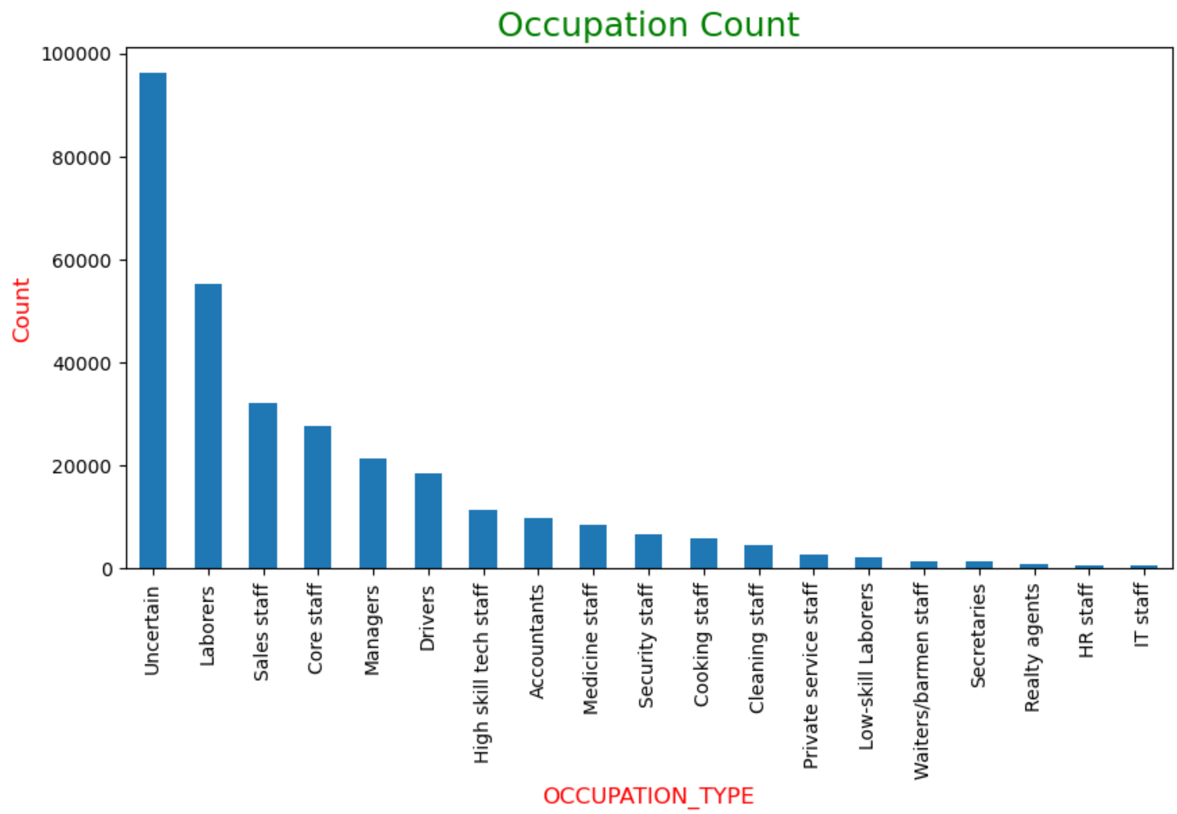
```
plt.figure(figsize = (12,5))
plt.title('Occupation Count',fontdict={'fontsize':20,'color':'Green'})
df2.OCCUPATION_TYPE.value_counts().plot.barh()
plt.xlabel('Occupation Type',fontdict={'fontsize':14,'fontweight':5,'color':'Red'})
plt.ylabel('Count',fontdict={'fontsize':14,'fontweight':5,'color':'Red'})
plt.show()
```



- **Labourers has the highest number of occupation replacing the missing data with most numbers of occupation is not sensible as this will make the data wrong. Therefore all the missing values are replaced with new group as "uncertain" entries**

In [61]:

```
df2.OCCUPATION_TYPE.fillna(value='Uncertain',inplace=True)
plt.figure(figsize = (10,5))
plt.title('Occupation Count',fontdict={'fontsize':18,'color':'Green'})
df2.OCCUPATION_TYPE.value_counts().plot.bar()
plt.xlabel('OCCUPATION_TYPE',fontdict={'fontsize':12,'fontweight':5,'color':'Red'})
plt.ylabel('Count',fontdict={'fontsize':12,'fontweight':5,'color':'Red'})
plt.show()
```



In [62]:

```
new_null = df2.isnull().sum()/len(df2)*100
new_null[new_null.values>0]
```

Out[62]:

```
AMT_GOODS_PRICE              0.090407
NAME_TYPE_SUITE              0.420164
CNT_FAM_MEMBERS              0.000650
EXT_SOURCE_2                 0.214635
EXT_SOURCE_3                19.825756
OBS_30_CNT_SOCIAL_CIRCLE     0.332034
DEF_30_CNT_SOCIAL_CIRCLE     0.332034
OBS_60_CNT_SOCIAL_CIRCLE     0.332034
DEF_60_CNT_SOCIAL_CIRCLE     0.332034
DAYS_LAST_PHONE_CHANGE       0.000325
dtype: float64
```

Working on EXT_SOURCE_3

In [63]:

```
df2.EXT_SOURCE_3.value_counts().head()
```

Out[63]:

```
0.746300    1460
0.713631    1315
0.694093    1276
0.670652    1191
0.652897    1154
Name: EXT_SOURCE_3, dtype: int64
```

Not changing any value of EXT_SOURCE as the missing percentage is very high.

In [66]:

```
#Columns having null values between 0 and 1

new_null = df2.isnull().sum()/len(df2)*100
new_null[(new_null.values>0) & (new_null.values<1) ]
```

Out[66]:

```
AMT_GOODS_PRICE             0.090407
NAME_TYPE_SUITE             0.420164
CNT_FAM_MEMBERS             0.000650
EXT_SOURCE_2                0.214635
OBS_30_CNT_SOCIAL_CIRCLE    0.332034
DEF_30_CNT_SOCIAL_CIRCLE    0.332034
OBS_60_CNT_SOCIAL_CIRCLE    0.332034
DEF_60_CNT_SOCIAL_CIRCLE    0.332034
DAYS_LAST_PHONE_CHANGE      0.000325
dtype: float64
```

Working with Box Plot

Handling null values in AMT_GOODS_PRICE

In [67]:

```
df2.AMT_GOODS_PRICE.value_counts().head()
```

Out[67]:

```
450000.0    26019
225000.0    25281
675000.0    24962
900000.0    15416
270000.0    11428
Name: AMT_GOODS_PRICE, dtype: int64
```

In [69]:

```
#Checking the percentile of AMT_GOODS_PRICE
df2.AMT_GOODS_PRICE.quantile(q=[0.25,0.5,0.75,0.95,0.99,1])
```

Out[69]:

```
0.25     238500.0
0.50     450000.0
0.75     679500.0
0.95    1305000.0
0.99    1800000.0
1.00    4050000.0
Name: AMT_GOODS_PRICE, dtype: float64
```

In [71]:

```
#Most occuring value
df2.AMT_GOODS_PRICE.mode()[0]
```

Out[71]:

```
450000.0
```

In [75]:

```
df2.AMT_GOODS_PRICE.median()
```

Out[75]:

```
450000.0
```

In [76]:

```
df2.AMT_GOODS_PRICE.mean()
```

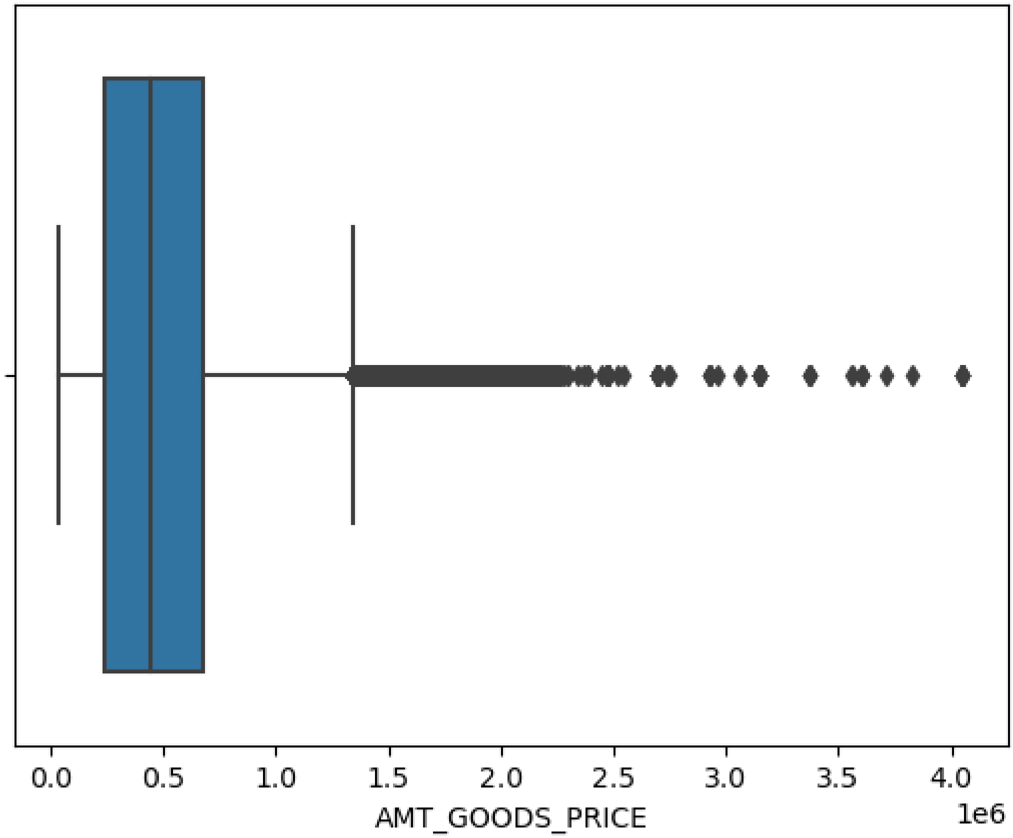
Out[76]:

```
538397.3458747937
```

In [70]:

```
sns.boxplot(df2.AMT_GOODS_PRICE)
plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Observations:

- Mean and median are very close to each other.
- Mode and Median are same i.e 45000.
- Since mode and median are similar we can use either of them.

In [77]:

```
#Replacing null values with mode
df2.AMT_GOODS_PRICE.fillna(df2.AMT_GOODS_PRICE.mode()[0],inplace=True)
```

Working with NAME_TYPE_SUITE

In [79]:

```
df2.NAME_TYPE_SUITE.value_counts()
```

Out[79]:

```
Unaccompanied      248515
Family              40148
Spouse, partner     11370
Children             3267
Other_B              1770
Other_A               866
Group of people       271
Name: NAME_TYPE_SUITE, dtype: int64
```

In [81]:

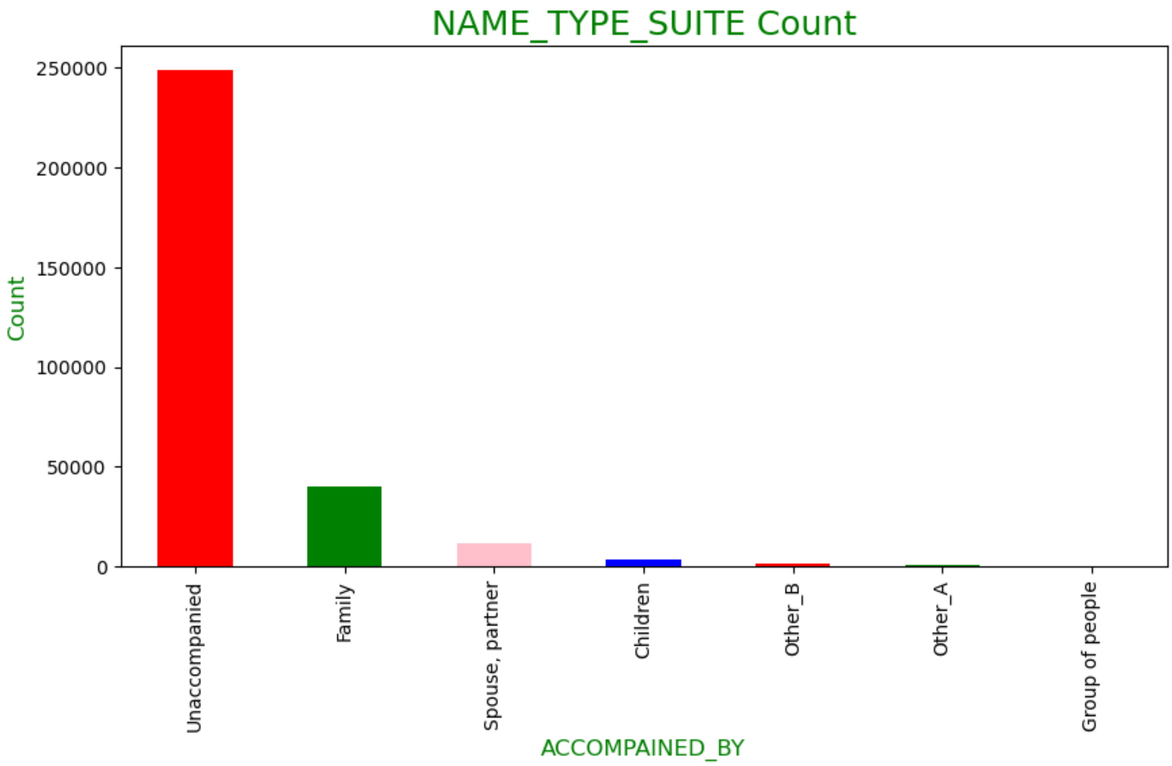
```
# Percentage of each category in NAME_TYPE_SUITE
df2.NAME_TYPE_SUITE.value_counts(normalize=True)*100
```

Out[81]:

```
Unaccompanied      81.159151
Family             13.111392
Spouse, partner     3.713174
Children            1.066925
Other_B             0.578040
Other_A             0.282815
Group of people     0.088502
Name: NAME_TYPE_SUITE, dtype: float64
```

In [87]:

```
plt.figure(figsize = (10,5))
df2.NAME_TYPE_SUITE.value_counts().plot.bar(color={'pink','green','red','blue'})
plt.title('NAME_TYPE_SUITE Count',fontdict={'fontsize':18,'color':'green'})
plt.xlabel('ACCOMPAINED_BY',fontdict={'fontsize':12,'fontweight':5,'color':'green'})
plt.ylabel('Count',fontdict={'fontsize':12,'fontweight':5,'color':'green'})
plt.show()
```



Obeservation:

1. It can be conclude that 80% of the loan applicant are Unaccompained.
2. Missing values can be replaced by unaccompained.

In [90]:

```
#Impute Null values in NAME_TYPE_SUITE by Unaccompained
df2.NAME_TYPE_SUITE.fillna('Unaccompanied',inplace=True)
```

Working with OBS_30_CNT_SOCIAL_CIRCLE

In [91]:

```
df2.OBS_30_CNT_SOCIAL_CIRCLE.value_counts().head()
```

Out[91]:

```
0.0    163901
1.0     48780
2.0     29808
3.0     20322
4.0     14143
Name: OBS_30_CNT_SOCIAL_CIRCLE, dtype: int64
```

In [92]:

```
#Percentile values for DEF_30_CNT_SOCIAL_CIRCLE
df2.OBS_30_CNT_SOCIAL_CIRCLE.quantile(q = [0.25,0.5,0.75,0.99,1])
```

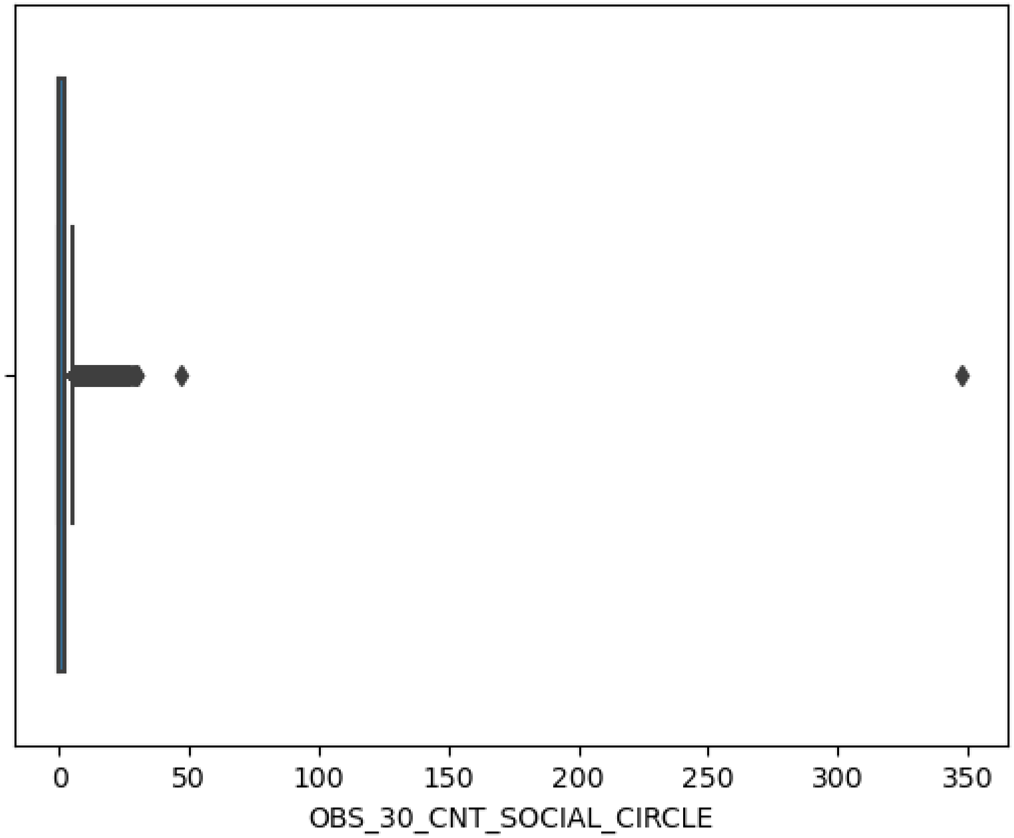
Out[92]:

```
0.25      0.0
0.50      0.0
0.75      2.0
0.99     10.0
1.00    348.0
Name: OBS_30_CNT_SOCIAL_CIRCLE, dtype: float64
```

In [93]:

```
sns.boxplot(df2.OBS_30_CNT_SOCIAL_CIRCLE)
plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [94]:

```
#Checking the mode of DEF_30_CNT_SOCIAL_CIRCLE
df2.OBS_30_CNT_SOCIAL_CIRCLE.mode()[0]
```

Out[94]:

```
0.0
```

In [97]:

```
df2.OBS_30_CNT_SOCIAL_CIRCLE.median()
```

Out[97]:

```
0.0
```

In [95]:

```
df2.OBS_30_CNT_SOCIAL_CIRCLE.mean()
```

Out[95]:

```
1.4222913227050555
```

Observation:

- Till 50th percentile all the values are 0, above it there are two outliers
- Mean and mode are closer, so we can use it for imputation.
- Median and mode are same i.e '0'

In [71]:

```
# Replacing missing values in OBS_30_CNT_SOCIAL_CIRCLE with 0
df2.OBS_30_CNT_SOCIAL_CIRCLE.fillna(0,inplace=True)
```

Working with DEF_30_CNT_SOCIAL_CIRCLE

In [98]:

```
df2.DEF_30_CNT_SOCIAL_CIRCLE.value_counts()
```

Out[98]:

```
0.0     271312
1.0      28328
2.0       5323
3.0       1192
4.0        253
5.0         56
6.0         11
7.0          1
34.0         1
8.0          1
Name: DEF_30_CNT_SOCIAL_CIRCLE, dtype: int64
```

In [99]:

```
#Percentile values for DEF_30_CNT_SOCIAL_CIRCLE
df2.DEF_30_CNT_SOCIAL_CIRCLE.quantile(q = [0.25,0.5,0.75,0.99,1])
```

Out[99]:

```
0.25     0.0
0.50     0.0
0.75     0.0
0.99     2.0
1.00    34.0
Name: DEF_30_CNT_SOCIAL_CIRCLE, dtype: float64
```

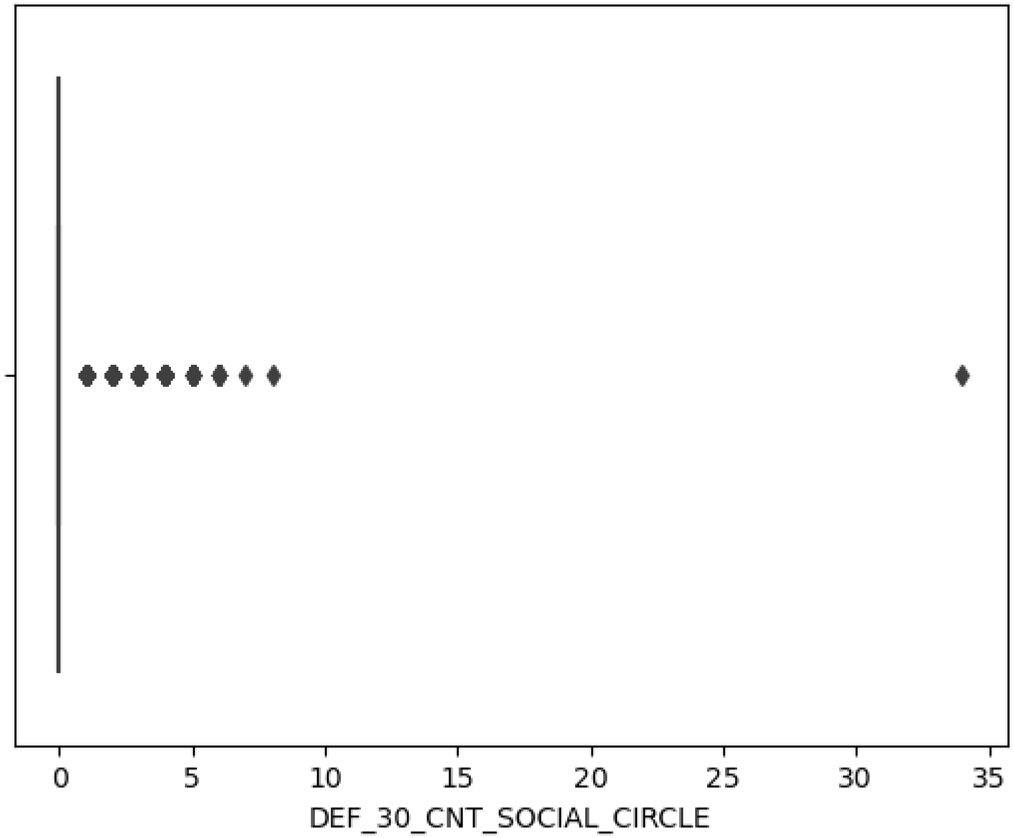
In [100]:

```
sns.boxplot(df2.DEF_30_CNT_SOCIAL_CIRCLE)
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[100]:

```
<AxesSubplot:xlabel='DEF_30_CNT_SOCIAL_CIRCLE'>
```



In [101]:

```
#Checking the most recurring value in DEF_30_CNT_SOCIAL_CIRCLE
df2.DEF_30_CNT_SOCIAL_CIRCLE.mode()[0]
```

Out[101]:

```
0.0
```

In [103]:

```
df2.DEF_30_CNT_SOCIAL_CIRCLE.median()
```

Out[103]:

```
0.0
```

In [102]:

```
df2.DEF_30_CNT_SOCIAL_CIRCLE.mean()
```

Out[102]:

```
0.1434262818212074
```

Observation:

- Median and mode are same i.e '0'
- Till 75th percentile all the values are 0, above it there are 9 outliers.
- Mean and mode are closer, so we can use it for replaceing with missing values.

In [104]:

```
# Replacing missing values in DEF_30_CNT_SOCIAL_CIRCLE with 0
df2.DEF_30_CNT_SOCIAL_CIRCLE.fillna(0,inplace=True)
```

Working with DEF_60_CNT_SOCIAL_CIRCLE

In [106]:

```
df2.DEF_60_CNT_SOCIAL_CIRCLE.value_counts()
```

Out[106]:

```
0.0     280709
1.0      21841
2.0       3170
3.0        598
4.0        135
5.0         20
6.0          3
7.0          1
24.0         1
Name: DEF_60_CNT_SOCIAL_CIRCLE, dtype: int64
```

In [109]:

```
#Percentile values for DEF_60_CNT_SOCIAL_CIRCLE
df2.DEF_60_CNT_SOCIAL_CIRCLE.quantile(q = [0.25,0.5,0.75,0.99,1])
```

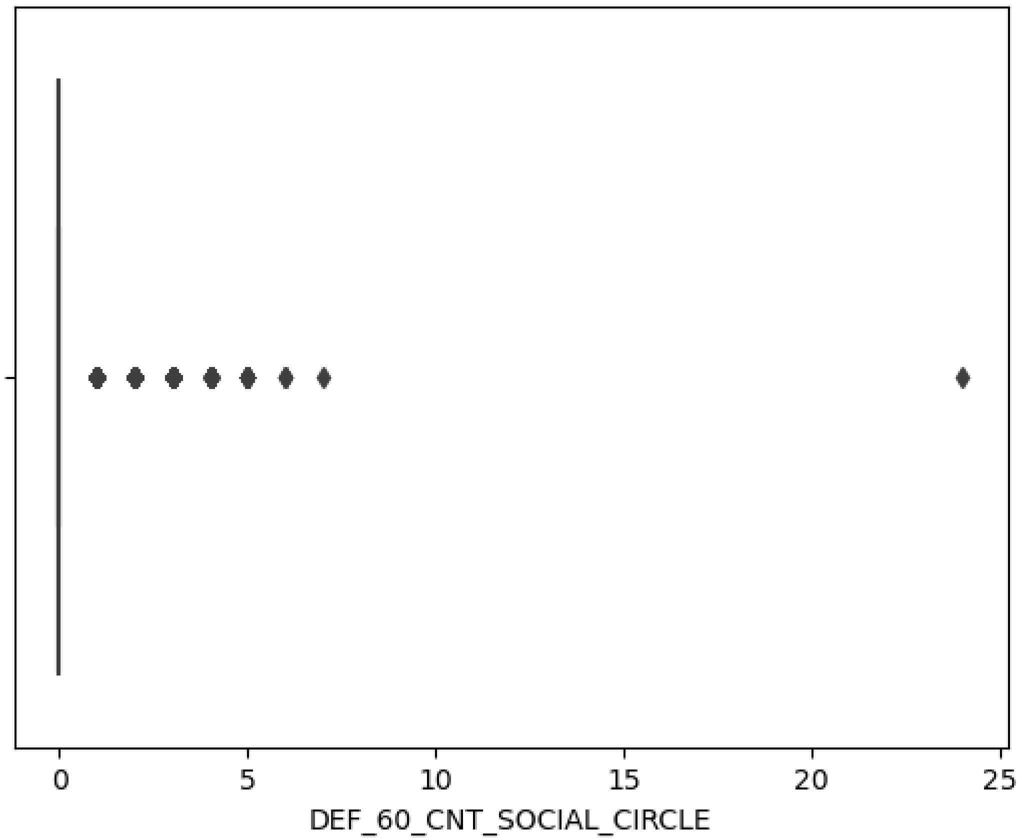
Out[109]:

```
0.25     0.0
0.50     0.0
0.75     0.0
0.99     2.0
1.00    24.0
Name: DEF_60_CNT_SOCIAL_CIRCLE, dtype: float64
```

In [110]:

```
sns.boxplot(df2.DEF_60_CNT_SOCIAL_CIRCLE)
plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [111]:

```
#Checking the most recurring value in DEF_60_CNT_SOCIAL_CIRCLE
df2.DEF_60_CNT_SOCIAL_CIRCLE.mode()[0]
```

Out[111]:

```
0.0
```

In [113]:

```
df2.DEF_60_CNT_SOCIAL_CIRCLE.median()
```

Out[113]:

```
0.0
```

In [114]:

```
df2.DEF_60_CNT_SOCIAL_CIRCLE.mean()
```

Out[114]:

```
0.10005285860649052
```

Observation:

- Median and mode are same i.e '0'
- Till 75th percentile all the values are 0, above it there are 8 outliers
- Mean and mode are closer, so we can use it for imputation.

In [115]:

```
# Replacing missing values in DEF_60_CNT_SOCIAL_CIRCLE with 0
df2.DEF_60_CNT_SOCIAL_CIRCLE.fillna(0,inplace=True)
```

In [116]:

```
df2.DEF_60_CNT_SOCIAL_CIRCLE.isnull().sum()
```

Out[116]:

```
0
```

Wokring OBS_60_CNT_SOCIAL_CIRCLE

In [117]:

```
df2.OBS_60_CNT_SOCIAL_CIRCLE.value_counts().head()
```

Out[117]:

```
0.0    164657
1.0     48867
2.0     29766
3.0     20215
4.0     13946
Name: OBS_60_CNT_SOCIAL_CIRCLE, dtype: int64
```

In [118]:

```
#Percentile values for OBS_60_CNT_SOCIAL_CIRCLE
df2.OBS_60_CNT_SOCIAL_CIRCLE.quantile(q = [0.25,0.5,0.75,0.99,1])
```

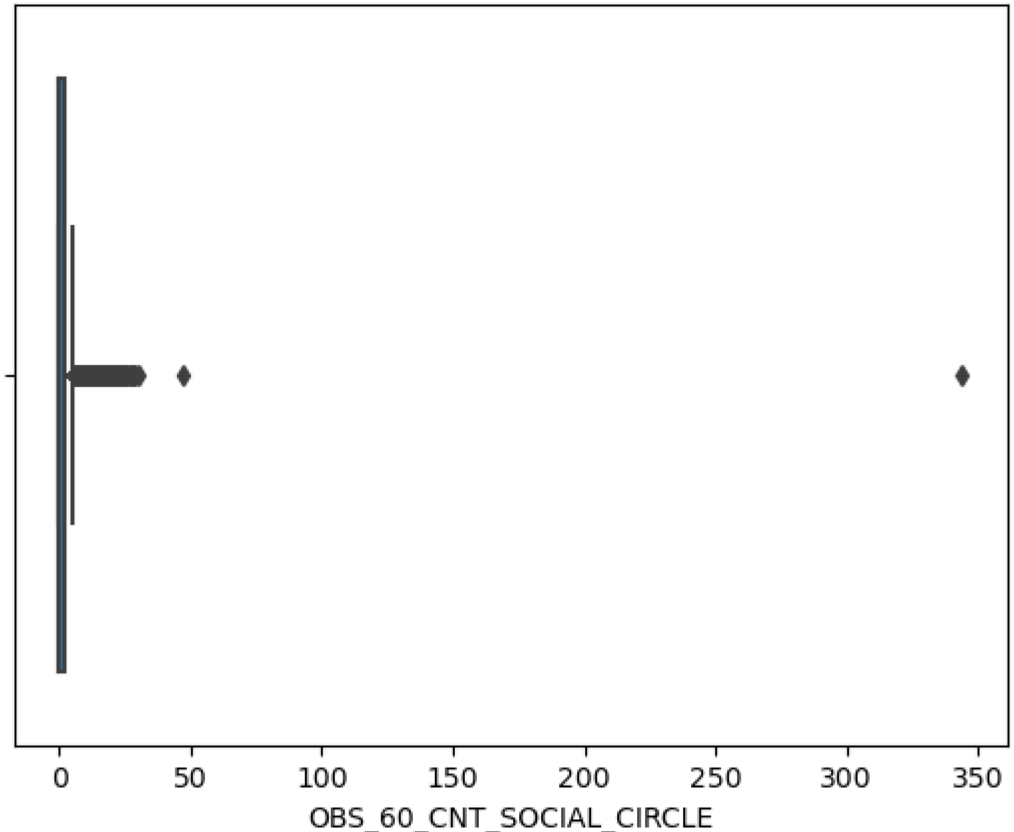
Out[118]:

```
0.25      0.0
0.50      0.0
0.75      2.0
0.99     10.0
1.00    344.0
Name: OBS_60_CNT_SOCIAL_CIRCLE, dtype: float64
```

In [119]:

```
sns.boxplot(df2.OBS_60_CNT_SOCIAL_CIRCLE)
plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [90]:

```
#Checking the most recurring value in OBS_60_CNT_SOCIAL_CIRCLE
df2.OBS_60_CNT_SOCIAL_CIRCLE.mode()[0]
```

Out[90]:

```
0.0
```

In [120]:

```
df2.OBS_60_CNT_SOCIAL_CIRCLE.median()
```

Out[120]:

```
0.0
```

In [121]:

```
df2.OBS_60_CNT_SOCIAL_CIRCLE.mean()
```

Out[121]:

```
1.4053374141047645
```

Observation:

- Median and mode are same is 0.
- Till 50th percentile all the values are 0, above it there are 2 outliers
- Mean and mode are closer, so we can use it for imputation.

In [125]:

```
# Replacing missing values in OBS_30_CNT_SOCIAL_CIRCLE with 0
df2.OBS_60_CNT_SOCIAL_CIRCLE.fillna(0,inplace=True)
```

In [126]:

```
df2.OBS_60_CNT_SOCIAL_CIRCLE.isnull().sum()
```

Out[126]:

```
0
```

In [127]:

```
#Checking the null values in all columns
new_null = df2.isnull().sum()/len(df2)*100
new_null[(new_null.values>0) & (new_null.values<1)]
```

Out[127]:

```
CNT_FAM_MEMBERS             0.000650
EXT_SOURCE_2                0.214635
OBS_30_CNT_SOCIAL_CIRCLE    0.332034
DAYS_LAST_PHONE_CHANGE      0.000325
dtype: float64
```

NULL values are very low so we are ignoring

Treating errors in Data types and data

In [137]:

```
#Lets analyze data types and values of other columns
df2.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 307499 entries, 0 to 307510
Data columns (total 73 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   SK_ID_CURR                   307499 non-null  int64  
 1   TARGET                       307499 non-null  int64  
 2   NAME_CONTRACT_TYPE           307499 non-null  object 
 3   CODE_GENDER                  307499 non-null  object 
 4   FLAG_OWN_CAR                 307499 non-null  object 
 5   FLAG_OWN_REALTY              307499 non-null  object 
 6   CNT_CHILDREN                 307499 non-null  int64  
 7   AMT_INCOME_TOTAL             307499 non-null  float64
 8   AMT_CREDIT                   307499 non-null  float64
 9   AMT_ANNUITY                  307499 non-null  float64
 10  AMT_GOODS_PRICE              307499 non-null  float64
 11  NAME_TYPE_SUITE              307499 non-null  object 
 12  NAME_INCOME_TYPE             307499 non-null  object 
 13  NAME_EDUCATION_TYPE          307499 non-null  object 
 14  NAME_FAMILY_STATUS           307499 non-null  object 
 15  NAME_HOUSING_TYPE            307499 non-null  object 
 16  REGION_POPULATION_RELATIVE   307499 non-null  float64
 17  DAYS_BIRTH                   307499 non-null  int64  
 18  DAYS_EMPLOYED                307499 non-null  int64  
 19  DAYS_REGISTRATION            307499 non-null  float64
 20  DAYS_ID_PUBLISH              307499 non-null  int64  
 21  FLAG_MOBIL                   307499 non-null  int64  
 22  FLAG_EMP_PHONE               307499 non-null  int64  
 23  FLAG_WORK_PHONE              307499 non-null  int64  
 24  FLAG_CONT_MOBILE             307499 non-null  int64  
 25  FLAG_PHONE                   307499 non-null  int64  
 26  FLAG_EMAIL                   307499 non-null  int64  
 27  OCCUPATION_TYPE              307499 non-null  object 
 28  CNT_FAM_MEMBERS              307497 non-null  float64
 29  REGION_RATING_CLIENT         307499 non-null  int64  
 30  REGION_RATING_CLIENT_W_CITY  307499 non-null  int64  
 31  WEEKDAY_APPR_PROCESS_START   307499 non-null  object 
 32  HOUR_APPR_PROCESS_START      307499 non-null  int64  
 33  REG_REGION_NOT_LIVE_REGION   307499 non-null  int64  
 34  REG_REGION_NOT_WORK_REGION   307499 non-null  int64  
 35  LIVE_REGION_NOT_WORK_REGION  307499 non-null  int64  
 36  REG_CITY_NOT_LIVE_CITY       307499 non-null  int64  
 37  REG_CITY_NOT_WORK_CITY       307499 non-null  int64  
 38  LIVE_CITY_NOT_WORK_CITY      307499 non-null  int64  
 39  ORGANIZATION_TYPE            307499 non-null  object 
 40  EXT_SOURCE_2                 306839 non-null  float64
 41  EXT_SOURCE_3                 246535 non-null  float64
 42  OBS_30_CNT_SOCIAL_CIRCLE     306478 non-null  float64
 43  DEF_30_CNT_SOCIAL_CIRCLE     307499 non-null  float64
 44  OBS_60_CNT_SOCIAL_CIRCLE     307499 non-null  float64
 45  DEF_60_CNT_SOCIAL_CIRCLE     307499 non-null  float64
 46  DAYS_LAST_PHONE_CHANGE       307498 non-null  float64
 47  FLAG_DOCUMENT_2              307499 non-null  int64  
 48  FLAG_DOCUMENT_3              307499 non-null  int64  
 49  FLAG_DOCUMENT_4              307499 non-null  int64  
```

```
 50  FLAG_DOCUMENT_5              307499 non-null  int64
 51  FLAG_DOCUMENT_6              307499 non-null  int64
 52  FLAG_DOCUMENT_7              307499 non-null  int64
 53  FLAG_DOCUMENT_8              307499 non-null  int64
 54  FLAG_DOCUMENT_9              307499 non-null  int64
 55  FLAG_DOCUMENT_10             307499 non-null  int64
 56  FLAG_DOCUMENT_11             307499 non-null  int64
 57  FLAG_DOCUMENT_12             307499 non-null  int64
 58  FLAG_DOCUMENT_13             307499 non-null  int64
 59  FLAG_DOCUMENT_14             307499 non-null  int64
 60  FLAG_DOCUMENT_15             307499 non-null  int64
 61  FLAG_DOCUMENT_16             307499 non-null  int64
 62  FLAG_DOCUMENT_17             307499 non-null  int64
 63  FLAG_DOCUMENT_18             307499 non-null  int64
 64  FLAG_DOCUMENT_19             307499 non-null  int64
 65  FLAG_DOCUMENT_20             307499 non-null  int64
 66  FLAG_DOCUMENT_21             307499 non-null  int64
 67  AMT_REQ_CREDIT_BUREAU_HOUR   307499 non-null  float64
 68  AMT_REQ_CREDIT_BUREAU_DAY    307499 non-null  float64
 69  AMT_REQ_CREDIT_BUREAU_WEEK   307499 non-null  float64
 70  AMT_REQ_CREDIT_BUREAU_MON    307499 non-null  float64
 71  AMT_REQ_CREDIT_BUREAU_QRT    307499 non-null  float64
 72  AMT_REQ_CREDIT_BUREAU_YEAR   307499 non-null  float64
dtypes: float64(20), int64(41), object(12)
memory usage: 173.6+ MB
```

Lets analysze all the columns

In [138]:

```
df2.CODE_GENDER.value_counts()
```

Out[138]:

```
F      202440
M      105055
XNA         4
Name: CODE_GENDER, dtype: int64
```

In [139]:

```
df2.CODE_GENDER.value_counts(normalize=True)
```

Out[139]:

```
F      0.658344
M      0.341643
XNA    0.000013
Name: CODE_GENDER, dtype: float64
```

We need to replace XNA with some value. If we observe value count percentage, aroung 65% loan applicants are female. Hence we can impute it with 'F'

In [140]:

```
#Replacing XNA with 'F' by using function

def replace(Gender):
    if Gender == 'XNA':
        return 'F'
    else:
        return Gender
```

In [141]:

```
df2.CODE_GENDER = df2.CODE_GENDER.apply(replace)
```

In [142]:

```
df2.CODE_GENDER.value_counts()
```

Out[142]:

```
F    202444
M    105055
Name: CODE_GENDER, dtype: int64
```

In [143]:

```
# Data in below mentioned columns is negative
print(df2['DAYS_BIRTH'].unique())
print(df2['DAYS_EMPLOYED'].unique())
print(df2['DAYS_REGISTRATION'].unique())
print(df2['DAYS_ID_PUBLISH'].unique())
print(df2['DAYS_LAST_PHONE_CHANGE'].unique())
```

```
[ -9461 -16765 -19046 ...  -7951  -7857 -25061]
[  -637  -1188   -225 ... -12971 -11084  -8694]
[ -3648.  -1186.  -4260. ... -16396. -14558. -14798.]
[-2120  -291 -2531 ... -6194 -5854 -6211]
[-1134.  -828.  -815. ... -3988. -3899. -3538.]
```

In [144]:

```
#Creating series for columns which has days, and converting them to positive value because
Days_negative = [column for column in df2 if column.startswith('DAYS')]
Days_negative
```

Out[144]:

```
['DAYS_BIRTH',
 'DAYS_EMPLOYED',
 'DAYS_REGISTRATION',
 'DAYS_ID_PUBLISH',
 'DAYS_LAST_PHONE_CHANGE']
```

In [145]:

```
#Converting to absolute values
df2[Days_negative] = abs(df2[Days_negative])
```

In [146]:

```
# After converting
print(df2['DAYS_BIRTH'].unique())
print(df2['DAYS_EMPLOYED'].unique())
print(df2['DAYS_REGISTRATION'].unique())
print(df2['DAYS_ID_PUBLISH'].unique())
print(df2['DAYS_LAST_PHONE_CHANGE'].unique())
```

```
[ 9461 16765 19046 ...  7951  7857 25061]
[  637  1188   225 ... 12971 11084  8694]
[ 3648.  1186.  4260. ... 16396. 14558. 14798.]
[2120  291 2531 ... 6194 5854 6211]
[1134.  828.  815. ... 3988. 3899. 3538.]
```

In [147]:

```python
# Analyze Organization Type column
df2.ORGANIZATION_TYPE.value_counts()
```

Out[147]:

```
Business Entity Type 3    67989
XNA                       55374
Self-employed             38409
Other                     16681
Medicine                  11192
Business Entity Type 2    10553
Government                10403
School                     8893
Trade: type 7              7831
Kindergarten               6880
Construction               6721
Business Entity Type 1     5983
Transport: type 4          5398
Trade: type 3              3492
Industry: type 9           3368
Industry: type 3           3278
Security                   3246
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 [148]:

```
(df2.ORGANIZATION_TYPE.value_counts(normalize=True)*100).head()
```

Out[148]:

```
Business Entity Type 3    22.110316
XNA                       18.007863
Self-employed             12.490772
Other                      5.424733
Medicine                   3.639687
Name: ORGANIZATION_TYPE, dtype: float64
```

We can observe 18% of the data is XNA, we cannot impute by mean/mode because then the result may get biased. We will create new category as unknown

In [149]:

```
df2.ORGANIZATION_TYPE = df2.ORGANIZATION_TYPE.apply(lambda x: 'Unknown' if x == 'XNA' else
```

In [150]:

```
(df2.ORGANIZATION_TYPE.value_counts(normalize=True)*100).head()
```

Out[150]:

```
Business Entity Type 3    22.110316
Unknown                   18.007863
Self-employed             12.490772
Other                      5.424733
Medicine                   3.639687
Name: ORGANIZATION_TYPE, dtype: float64
```

Converting all the Days column to Year

In [151]:

```
Days_column = [column for column in df2 if column.startswith('DAYS')]
df2[Days_column] = df2[Days_column]/365
df2[Days_column].describe()
```

Out[151]:

	DAYS_BIRTH	DAYS_EMPLOYED	DAYS_REGISTRATION	DAYS_ID_PUBLISH	DAYS_LAST
count	307499.000000	307499.000000	307499.000000	307499.000000	
mean	43.937135	185.554239	13.660574	8.203322	
std	11.956166	382.043486	9.651664	4.135487	
min	20.517808	0.000000	0.000000	0.000000	
25%	34.008219	2.556164	5.506849	4.712329	
50%	43.150685	6.079452	12.339726	8.915068	
75%	53.923288	15.635616	20.490411	11.778082	
max	69.120548	1000.665753	67.594521	19.717808	

In [152]:

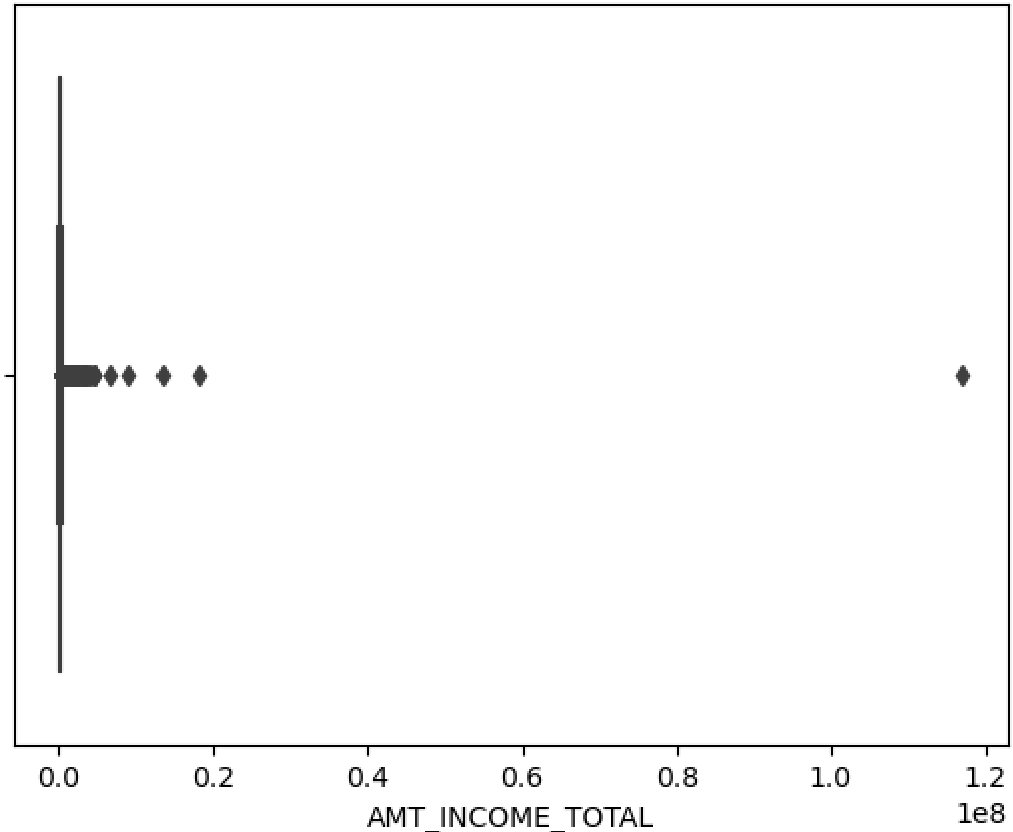
```
#We need to rename columns
df2.rename(columns={'DAYS_BIRTH':'AGE' ,'DAYS_EMPLOYED':'WORK_EXPERIENCE' ,
    'DAYS_REGISTRATION':'YEARS_REGISTRATION' ,'DAYS_ID_PUBLISH':'ID_CHANGE_YEAR','DAYS_LAST
```

Checking Outliers

In [153]:

```
# Finding outlier on AMT_INCOME_TOTAL column
sns.boxplot(df2.AMT_INCOME_TOTAL)
plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [154]:

```
#Let's check the spread of AMT_INCOME_TOTAL
df2.AMT_INCOME_TOTAL.quantile(q=[0.25,0.5,0.75,0.95,0.99,1])
```

Out[154]:

```
0.25       112500.0
0.50       146997.0
0.75       202500.0
0.95       337500.0
0.99       472500.0
1.00    117000000.0
Name: AMT_INCOME_TOTAL, dtype: float64
```

In [155]:

```
#In the above case we can see a large value, let's check the data
Max_Annual_income =df2[df2.AMT_INCOME_TOTAL == df2.AMT_INCOME_TOTAL.max()]
Max_Annual_income
```

Out[155]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FL
12840	114967	1	Cash loans	F	N	

1 rows × 73 columns

In [156]:

```
Max_Annual_income.OCCUPATION_TYPE
```

Out[156]:

```
12840    Laborers
Name: OCCUPATION_TYPE, dtype: object
```

In [157]:

```
#Before deleting
df2.shape
```

Out[157]:

```
(307499, 73)
```

In [158]:

```
#From above two results we can infer that this a outlier since the person is a labourer and
#We will remove this row as we don't want this data to impact our analysis
df3 = df2[df2.index!=12840]
```

In [159]:

```
#After deleting
df3.shape
```

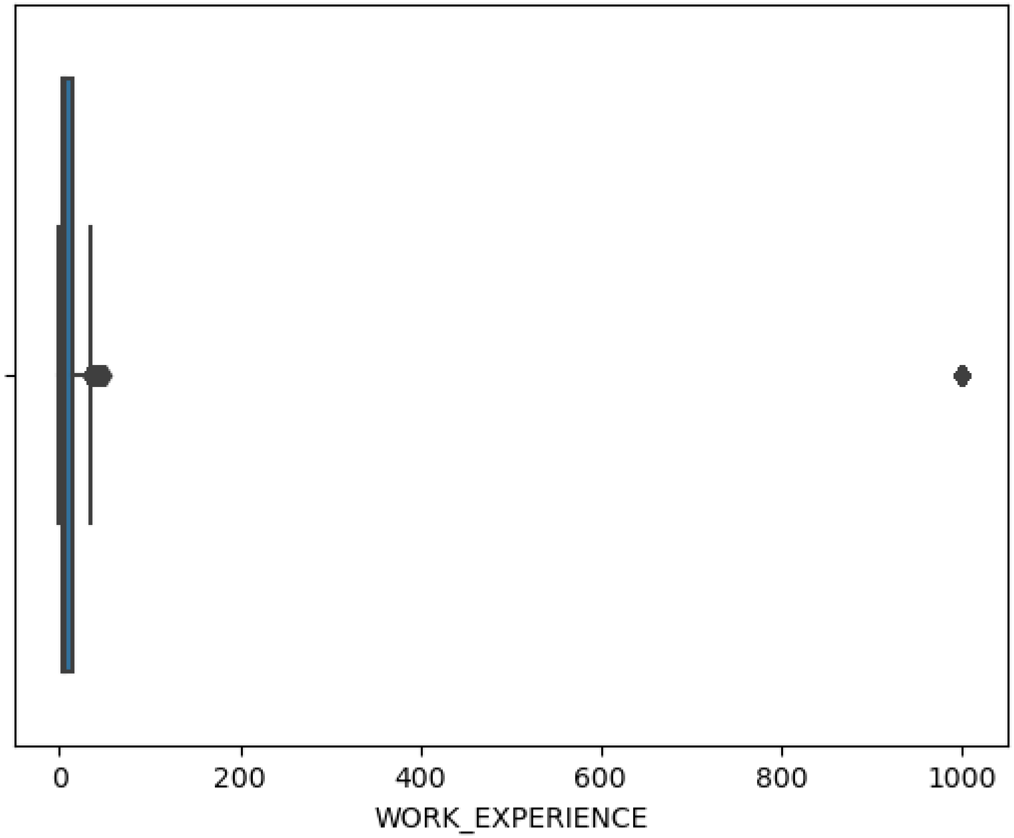
Out[159]:

```
(307498, 73)
```

In [160]:

```
# Finding outlier on WORK_EXPERIENCE column
sns.boxplot(df3.WORK_EXPERIENCE)
plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [163]:

```
#In the above boxplot we can see work_experience as more than 1000 which is not possible.
df3[df3.WORK_EXPERIENCE>999]
```

Out[163]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FI
8	100011	0	Cash loans	F	N	
11	100015	0	Cash loans	F	N	
23	100027	0	Cash loans	F	N	
38	100045	0	Cash loans	F	N	
43	100050	0	Cash loans	F	N	
...	...	...	...	...	...	
307469	456209	0	Cash loans	F	N	
307483	456227	0	Cash loans	F	N	
307487	456231	0	Cash loans	M	N	
307505	456249	0	Cash loans	F	N	
307507	456252	0	Cash loans	F	N	

55374 rows × 73 columns

In [164]:

```
#We can see there are 55374 records we cannot drop the data completetly, let's check the in
df3.WORK_EXPERIENCE.quantile([0.25,0.5,0.75,0.8,0.9,0.95,0.99])
# 80% data is below 25 years
```

Out[164]:

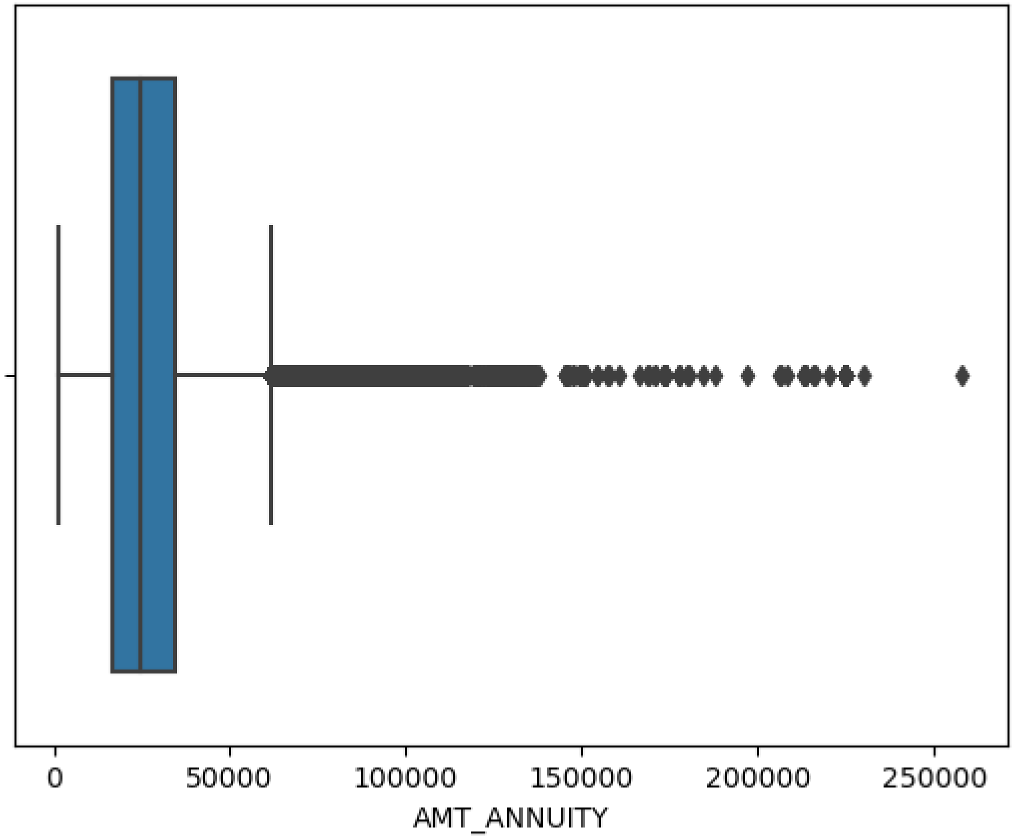
```
0.25       2.556164
0.50       6.079452
0.75      15.635616
0.80      25.178082
0.90    1000.665753
0.95    1000.665753
0.99    1000.665753
Name: WORK_EXPERIENCE, dtype: float64
```

The values above 1000 WORK_EXPERIENCE are outliers

In [165]:

```
# Finding outlier on AMT_ANNUITY column
sns.boxplot(df3['AMT_ANNUITY'])
plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [166]:

```
#In the above boxplot we can observe there is one value which is above 2500000, let's check
df3[df3.AMT_ANNUITY>250000]
#By observing the AMT_CREDIT and AMT_INCOME_TOTAL we can say that AMT_ANNUITY is not a outl
```

Out[166]:

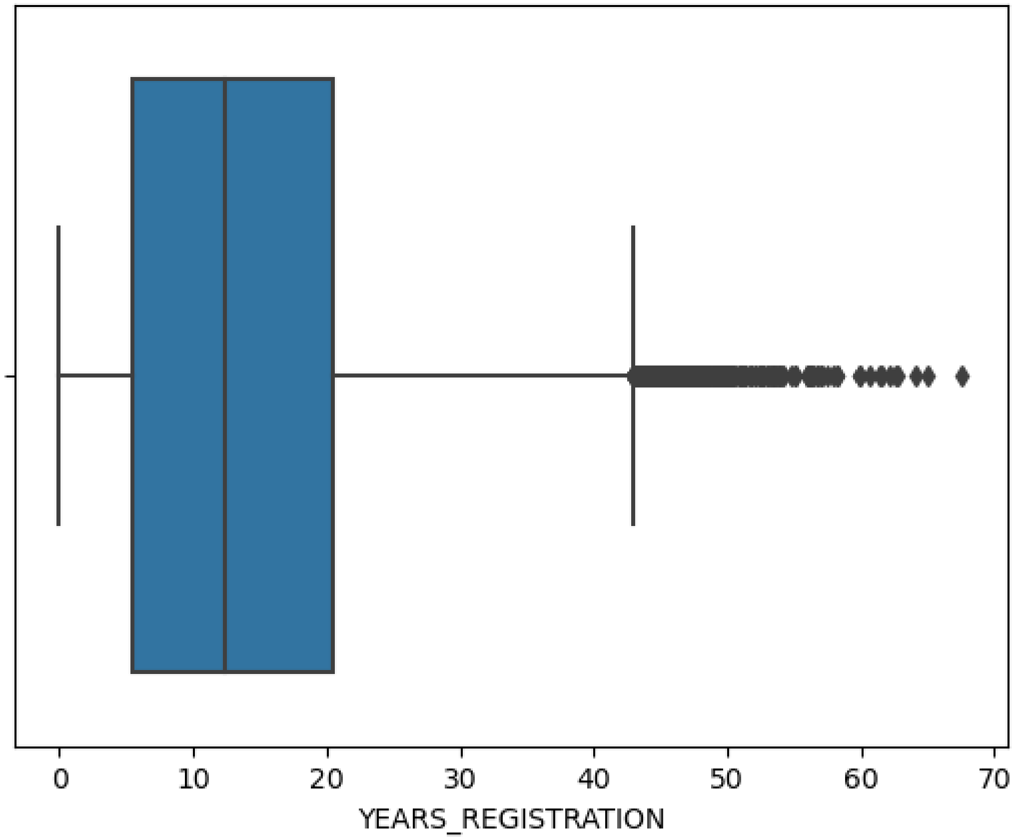
	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FL
17948	120926	0	Cash loans	M	Y	

1 rows × 73 columns

In [167]:

```
# Finding outlier on YEARS_REGISTRATION column
sns.boxplot(df3['YEARS_REGISTRATION'])
plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [168]:

```
#Let's check the value were YEARS_REGISTRATION is greater than 65
df3[df3.YEARS_REGISTRATION>65]
#These values can be considered as outliers
```

Out[168]:

EQ_CREDIT_BUREAU_MON	AMT_REQ_CREDIT_BUREAU_QRT	AMT_REQ_CREDIT_BUREAU_YEAR
0.0	0.0	0.0
1.0	0.0	2.0

Binning the values for AMT_INCOME_RANGE

In [169]:

```
#Binning makes it easier to analyze continuous variables
# Binning 'AMT_INCOME_RANGE'
# 0-0.2 as Extremly Low
# 0.2-0.5 as Low
# 0.5-0.8 as Medium
# 0.8-0.95 as High
# 0.95 - 1 as Extremly High
df3['AMT_INCOME_RANGE'] = pd.qcut(df3.AMT_INCOME_TOTAL, q=[0, 0.2, 0.5, 0.8, 0.95, 1], labe
df3['AMT_INCOME_RANGE'].head()
```

```
C:\Users\somes\AppData\Local\Temp\ipykernel_12308\3074512057.py:8: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df3['AMT_INCOME_RANGE'] = pd.qcut(df3.AMT_INCOME_TOTAL, q=[0, 0.2, 0.5, 0.
8, 0.95, 1], labels=['EXTREMLY_LOW', 'LOW', "MEDIUM", 'HIGH', 'EXTREMLY_HIG
H'])
```

Out[169]:

```
0          MEDIUM
1            HIGH
2    EXTREMLY_LOW
3             LOW
4             LOW
Name: AMT_INCOME_RANGE, dtype: category
Categories (5, object): ['EXTREMLY_LOW' < 'LOW' < 'MEDIUM' < 'HIGH' < 'EXTRE
MLY_HIGH']
```

In [170]:

```
#Binning Amount credit range
df3['AMT_CREDIT_RANGE'] = pd.qcut(df3.AMT_CREDIT, q=[0, 0.2, 0.5, 0.8, 0.95, 1], labels=['V
df3['AMT_CREDIT_RANGE'].head()
```

```
C:\Users\somes\AppData\Local\Temp\ipykernel_12308\3726798758.py:2: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df3['AMT_CREDIT_RANGE'] = pd.qcut(df3.AMT_CREDIT, q=[0, 0.2, 0.5, 0.8, 0.9
5, 1], labels=['VERY_LOW', 'LOW', "MEDIUM", 'HIGH', 'VERY_HIGH'])
```

Out[170]:

```
0          LOW
1         HIGH
2     VERY_LOW
3          LOW
4          LOW
Name: AMT_CREDIT_RANGE, dtype: category
Categories (5, object): ['VERY_LOW' < 'LOW' < 'MEDIUM' < 'HIGH' < 'VERY_HIG
H']
```

Data Imbalance for target variable

In [171]:

```
df3.TARGET.value_counts()
```

Out[171]:

```
0    282674
1     24824
Name: TARGET, dtype: int64
```

In [172]:

```
df3.TARGET.value_counts(normalize=True)*100
```

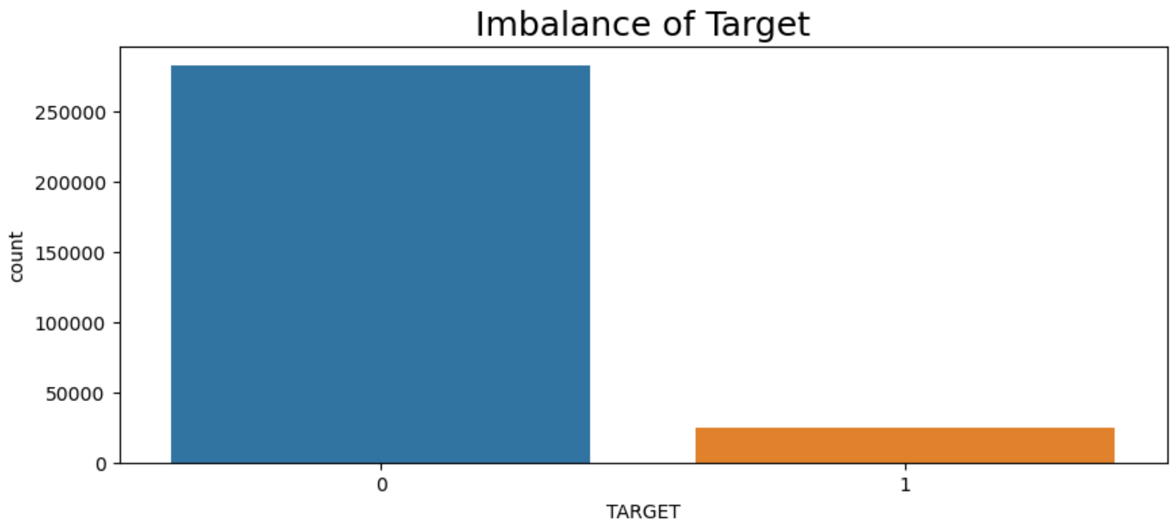
Out[172]:

```
0    91.927102
1     8.072898
Name: TARGET, dtype: float64
```

We can see clearly that there is imbalance in TARGET variable, let's plot a bar chart to support our inference

In [173]:

```
plt.figure(figsize=[10,4])
sns.countplot(x = df3['TARGET'], data = df3)
plt.title('Imbalance of Target',fontdict={'fontsize':18})
plt.show()
```



In [175]:

```
#We will divide the dataset into two dataframe, one for client's with payment difficulties(
df_target0 = df3[df3.TARGET==0]
df_target1 = df3[df3.TARGET==1]
```

In [176]:

```
df_target0.shape
```

Out[176]:

```
(282674, 75)
```

In [177]:

```
df_target1.shape
```

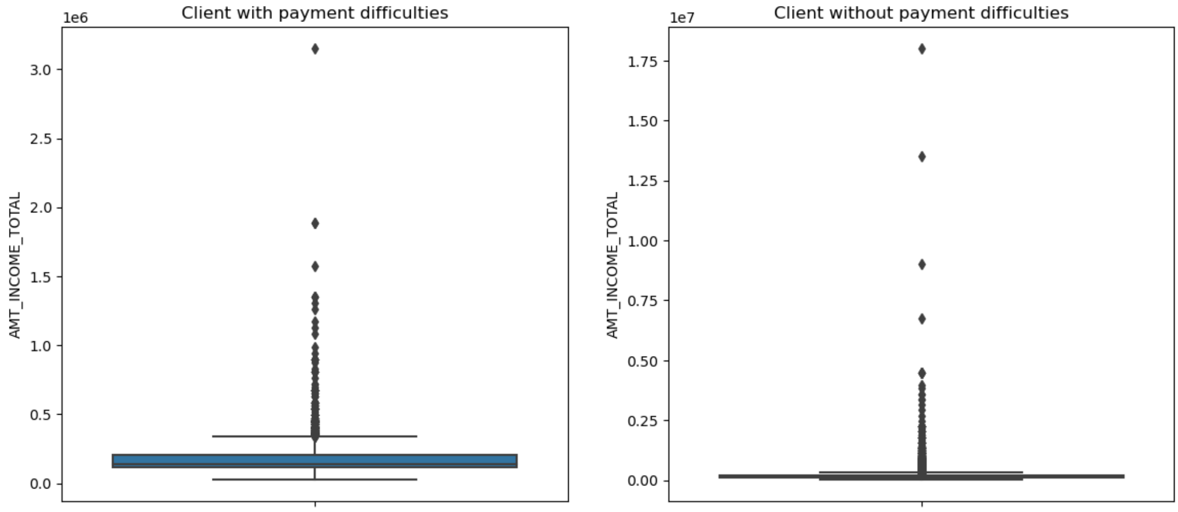
Out[177]:

```
(24824, 75)
```

Univariate Analysis

In [179]:

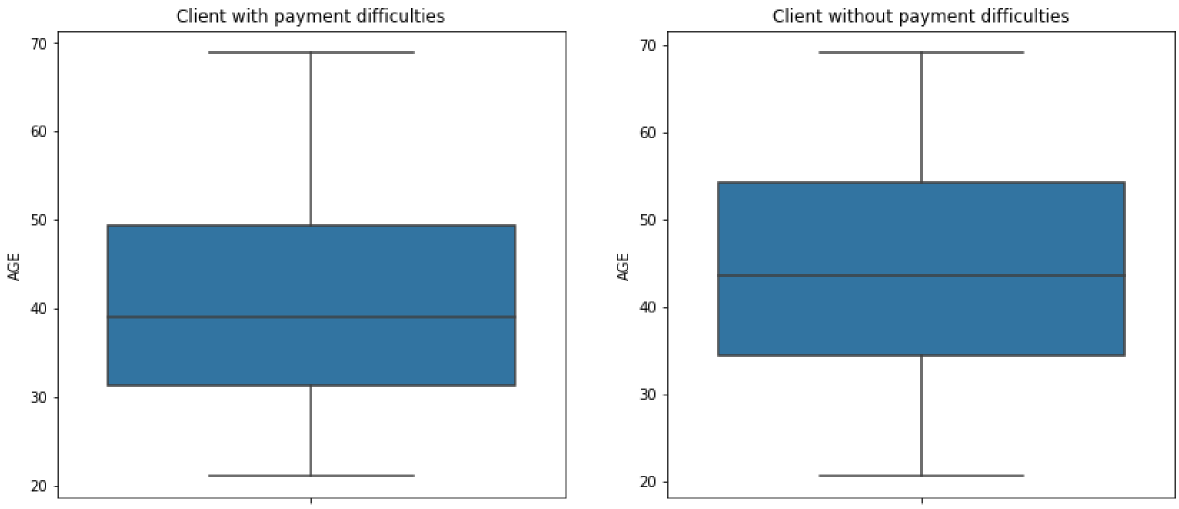
```
#The most important column in deciding target variable could be Total income of applicant.
plt.figure(figsize=[14,6])

plt.subplot(1,2,1)
ax = sns.boxplot(y=df_target1['AMT_INCOME_TOTAL'])
plt.title('Client with payment difficulties')

plt.subplot(1,2,2)
ax = sns.boxplot(y=df_target0['AMT_INCOME_TOTAL'])
plt.title('Client without payment difficulties')
plt.show()
```



Inference: Total income is higher for client without payment difficulties as compared to client with payment difficulties

In [133]:

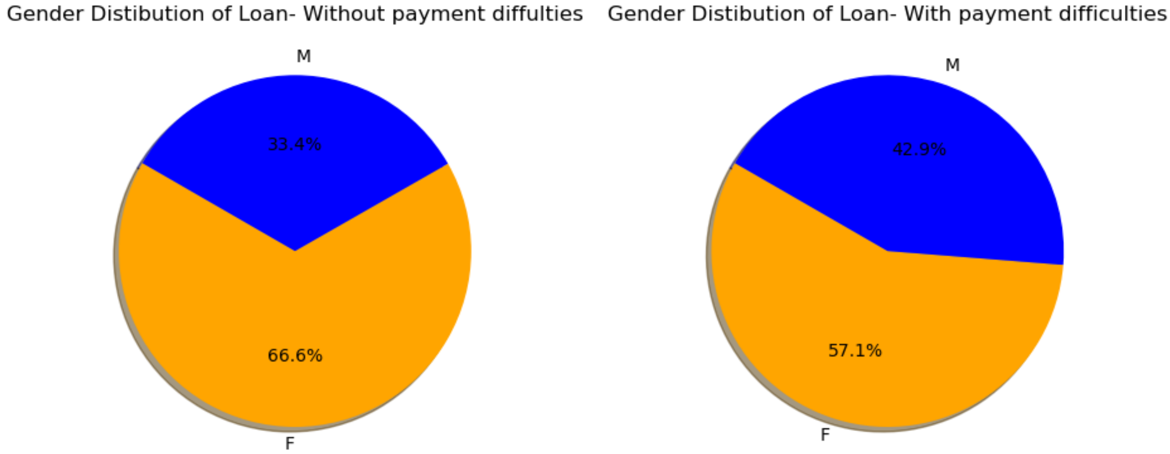
```
#Let's analyze the age group of applicants
plt.figure(figsize=[14,6])

plt.subplot(1,2,1)
ax = sns.boxplot(y=df_target1.AGE)
plt.title('Client with payment difficulties')

plt.subplot(1,2,2)
ax = sns.boxplot(y=df_target0.AGE)
plt.title('Client without payment difficulties')
plt.show()

#By observing the boxplot, we can infer that Client with payment difficulties are in range
#client without payment difficulties are in range 34-54
```



Categorical Variable

In [180]:

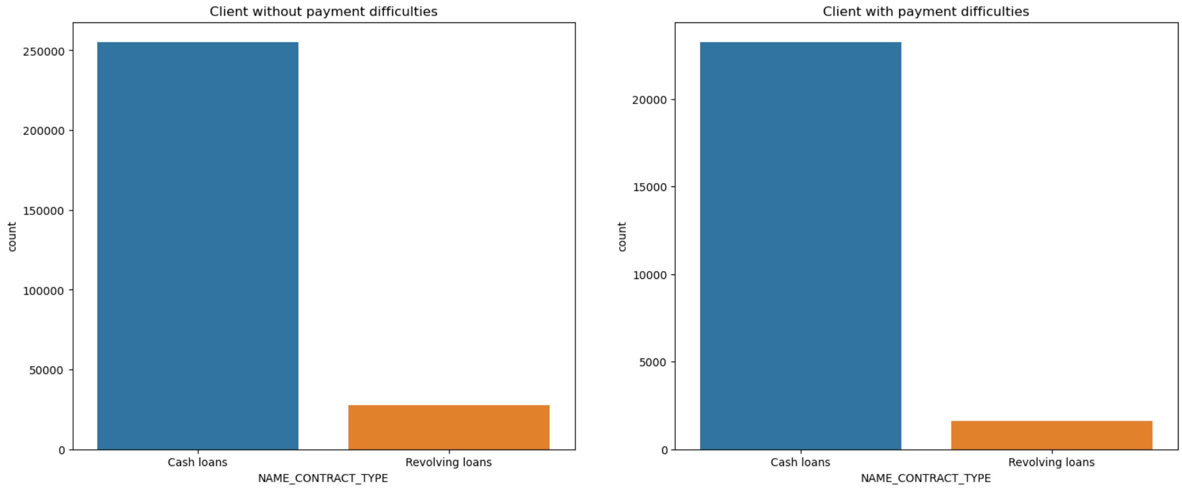
```
g_count = df_target0.CODE_GENDER.value_counts()
f_count = df_target1.CODE_GENDER.value_counts()

plt.figure(figsize=[10,4])
plt.subplot(1,2,1)
plt.pie(g_count.values,labels=g_count.index,colors=['orange','blue'], autopct='%1.1f%%',sha
plt.title("Gender Distibution of Loan- Without payment diffulties")

plt.subplot(1,2,2)
plt.pie(f_count.values,labels=f_count.index,colors=['orange','blue'], autopct='%1.1f%%',sha
plt.title("Gender Distibution of Loan- With payment difficulties")
plt.tight_layout()
plt.show()
# Here we observe that Females have more Loan payment difficulties as compared to Male's.
```



In [181]:

```
#Based on NAME_CONTRACT_TYPE
plt.figure(figsize=(18,7))

plt.subplot(1,2,1)
ax = sns.countplot(df_target0['NAME_CONTRACT_TYPE'])
plt.title('Client without payment difficulties')

plt.subplot(1,2,2)
ax = sns.countplot(df_target1['NAME_CONTRACT_TYPE'])
plt.title('Client with payment difficulties')
plt.show()

#Observing the countplot, we don't see significant difference in NAME_CONTRACT_TYPE
```

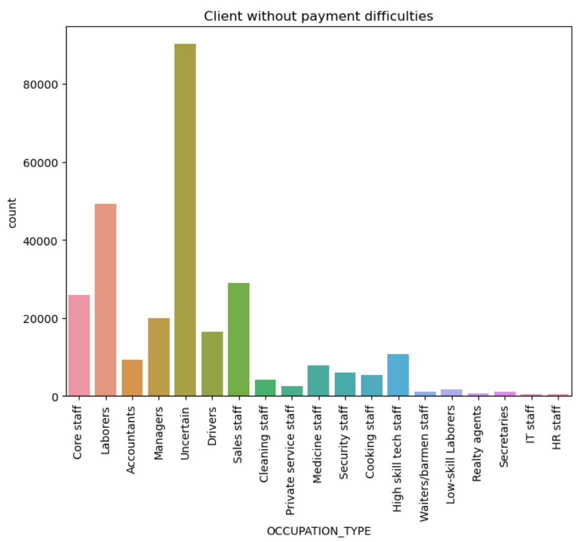
```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

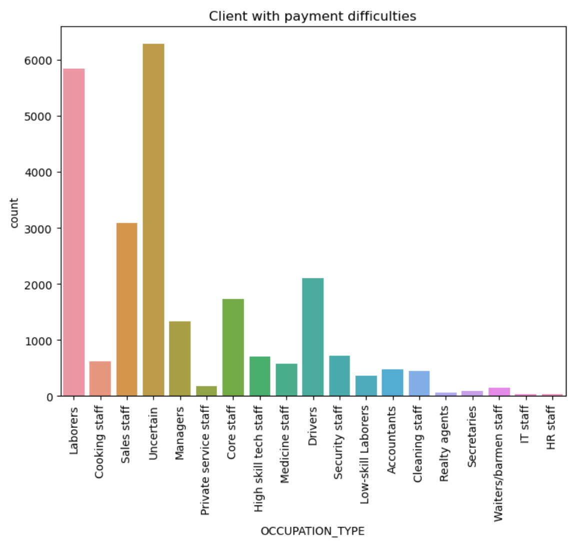


In [182]:

```
# Based on the OCCUPATION_TYPE
plt.figure(figsize=(18,6))

plt.subplot(1,2,1)
ax = sns.countplot(df_target0['OCCUPATION_TYPE'])
plt.title('Client without payment difficulties')
plt.xticks(rotation=90)

plt.subplot(1,2,2)
ax = sns.countplot(df_target1['OCCUPATION_TYPE'])
plt.title('Client with payment difficulties')
plt.xticks(rotation=90)
plt.show()

#We can clearly observe that labourers are the highest in both categories i.e client with a
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

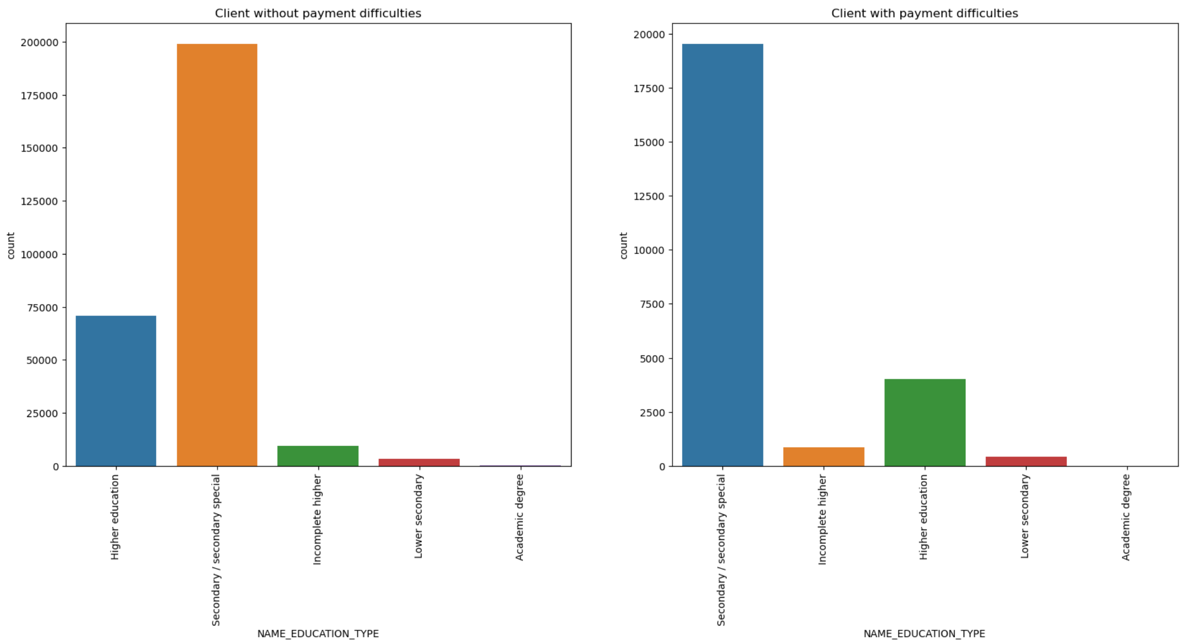




In [183]:

```
# Based on NAME_EDUCATION_TYPE
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
ax = sns.countplot(df_target0['NAME_EDUCATION_TYPE'])
plt.title('Client without payment difficulties')
plt.xticks(rotation=90)

plt.subplot(1,2,2)
ax = sns.countplot(df_target1['NAME_EDUCATION_TYPE'])
plt.title('Client with payment difficulties')
plt.xticks(rotation=90)
plt.show()

# .
```

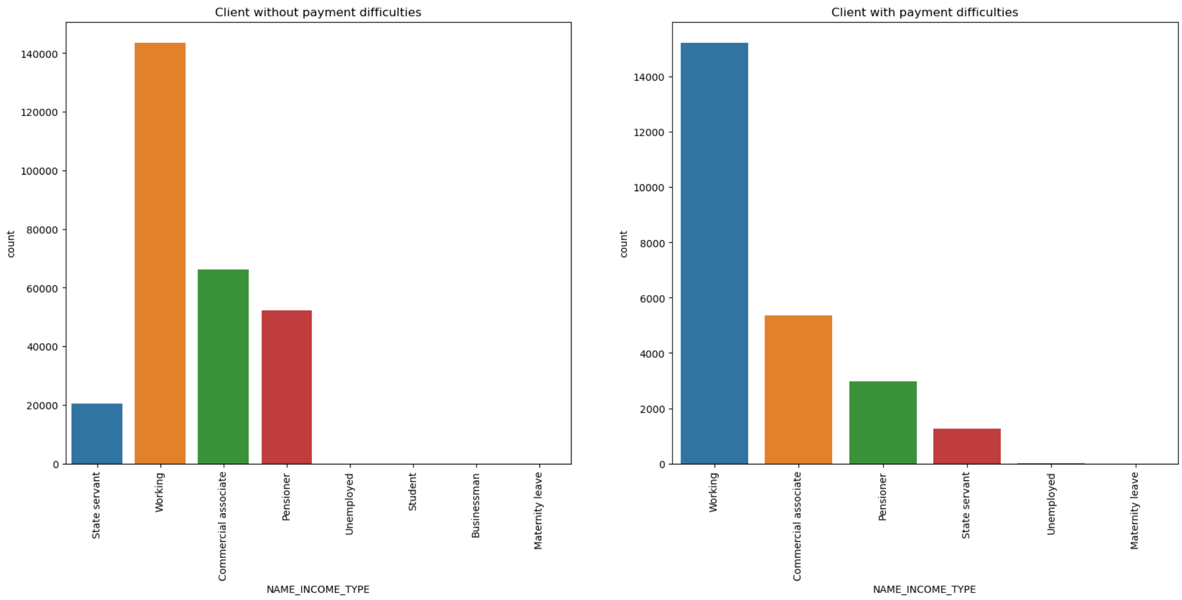
```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [184]:

```
# Based upon the NAME_INCOME_TYPE
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
ax = sns.countplot(df_target0['NAME_INCOME_TYPE'])
plt.title('Client without payment difficulties')
plt.xticks(rotation=90)

plt.subplot(1,2,2)
ax = sns.countplot(df_target1['NAME_INCOME_TYPE'])
plt.title('Client with payment difficulties')
plt.xticks(rotation=90)
plt.show()

#Pensioner and Govt Employees have better on-time payments.
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [188]:

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

Cleaning the previous_app Dataframe

In [189]:

```
previous_app.head(5)
```

Out[189]:

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

5 rows × 37 columns

In [191]:

```
#check for missing value percentage
nul=(previous_app.isnull().sum()*100)/len(previous_app)
nul
```

Out[191]:

```
SK_ID_PREV                      0.000000
SK_ID_CURR                      0.000000
NAME_CONTRACT_TYPE              0.000000
AMT_ANNUITY                    22.286665
AMT_APPLICATION                 0.000000
AMT_CREDIT                      0.000060
AMT_DOWN_PAYMENT               53.636480
AMT_GOODS_PRICE                23.081773
WEEKDAY_APPR_PROCESS_START      0.000000
HOUR_APPR_PROCESS_START         0.000000
FLAG_LAST_APPL_PER_CONTRACT     0.000000
NFLAG_LAST_APPL_IN_DAY          0.000000
RATE_DOWN_PAYMENT              53.636480
RATE_INTEREST_PRIMARY          99.643698
RATE_INTEREST_PRIVILEGED       99.643698
NAME_CASH_LOAN_PURPOSE          0.000000
NAME_CONTRACT_STATUS            0.000000
DAYS_DECISION                   0.000000
NAME_PAYMENT_TYPE               0.000000
CODE_REJECT_REASON              0.000000
NAME_TYPE_SUITE                49.119754
NAME_CLIENT_TYPE                0.000000
NAME_GOODS_CATEGORY             0.000000
NAME_PORTFOLIO                  0.000000
NAME_PRODUCT_TYPE               0.000000
CHANNEL_TYPE                    0.000000
SELLERPLACE_AREA                0.000000
NAME_SELLER_INDUSTRY            0.000000
CNT_PAYMENT                    22.286366
NAME_YIELD_GROUP                0.000000
PRODUCT_COMBINATION             0.020716
DAYS_FIRST_DRAWING             40.298129
DAYS_FIRST_DUE                 40.298129
DAYS_LAST_DUE_1ST_VERSION      40.298129
DAYS_LAST_DUE                  40.298129
DAYS_TERMINATION               40.298129
NFLAG_INSURED_ON_APPROVAL      40.298129
dtype: float64
```

In [192]:

```
#Check for the columns which have missing values more than 40%
nul=nul[nul>40]

nul
```

Out[192]:

```
AMT_DOWN_PAYMENT             53.636480
RATE_DOWN_PAYMENT            53.636480
RATE_INTEREST_PRIMARY        99.643698
RATE_INTEREST_PRIVILEGED     99.643698
NAME_TYPE_SUITE              49.119754
DAYS_FIRST_DRAWING           40.298129
DAYS_FIRST_DUE               40.298129
DAYS_LAST_DUE_1ST_VERSION    40.298129
DAYS_LAST_DUE                40.298129
DAYS_TERMINATION             40.298129
NFLAG_INSURED_ON_APPROVAL    40.298129
dtype: float64
```

In [193]:

```
#Getting the indexes where null value percentage is more than 40%
col_Drop=nul[nul.values>40]

col_Drop.index
```

Out[193]:

```
Index(['AMT_DOWN_PAYMENT', 'RATE_DOWN_PAYMENT', 'RATE_INTEREST_PRIMARY',
       'RATE_INTEREST_PRIVILEGED', 'NAME_TYPE_SUITE', 'DAYS_FIRST_DRAWING',
       'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION', 'DAYS_LAST_DUE',
       'DAYS_TERMINATION', 'NFLAG_INSURED_ON_APPROVAL'],
      dtype='object')
```

In [194]:

```
#dropping the columns which have null values more than 40%
previous_app.drop(['AMT_DOWN_PAYMENT', 'RATE_DOWN_PAYMENT', 'RATE_INTEREST_PRIMARY','RATE_I
       'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION', 'DAYS_LAST_DUE',
       'DAYS_TERMINATION', 'NFLAG_INSURED_ON_APPROVAL'],axis=1,inplace=True)
```

In [195]:

```
#Dropping the unused columns from Dataframe
previous_app.drop(["WEEKDAY_APPR_PROCESS_START","HOUR_APPR_PROCESS_START","NAME_CASH_LOAN_P

previous_app.head(2)
```

Out[195]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_APPLICATION	AMT_CREDIT	FLA
0	2030495	271877	Consumer loans	17145.0	17145.0	
1	2802425	108129	Cash loans	607500.0	679671.0	

Handling null values in AMT_APPLICATION

In [196]:

```
#Checking out the values percentage in AMT_APPLICATION

previous_app["AMT_APPLICATION"].value_counts(normalize=True)*100
```

Out[196]:

```
0.00         23.494115
45000.00      2.863765
225000.00     2.607031
135000.00     2.435496
450000.00     2.329342
               ...
185292.00     0.000060
225054.00     0.000060
156212.55     0.000060
99896.31      0.000060
267295.50     0.000060
Name: AMT_APPLICATION, Length: 93885, dtype: float64
```

Observation

1. As we can see there are approx 23% values which are 0 in AMT_APPLICATION column.We need to remove it as it may affect the EDA process

In [197]:

```
#Checking out the percentage contribution in AMT_APPLICATION using quantile
previous_app["AMT_APPLICATION"].quantile([0,.25,.50,.75,.99,1])
```

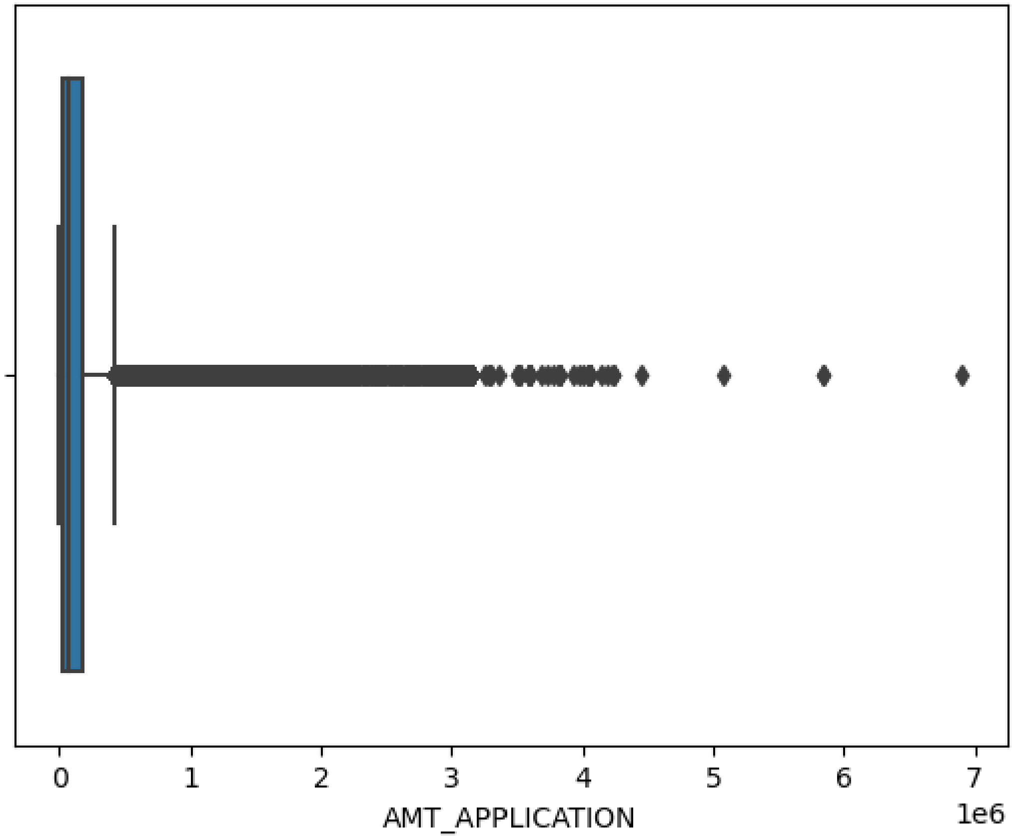
Out[197]:

```
0.00          0.0
0.25      18720.0
0.50      71046.0
0.75     180360.0
0.99    1350000.0
1.00    6905160.0
Name: AMT_APPLICATION, dtype: float64
```

In [198]:

```
#Checking out the outliers using the boxplot
sns.boxplot(previous_app["AMT_APPLICATION"])

plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Observation

1. As we have seen there is no null values in AMT_APPLICATION column but there are some values which are 0. we need to replace them with another value.

2. Also there are some outliers also present which can affect the analysis. it is better approach to remove them

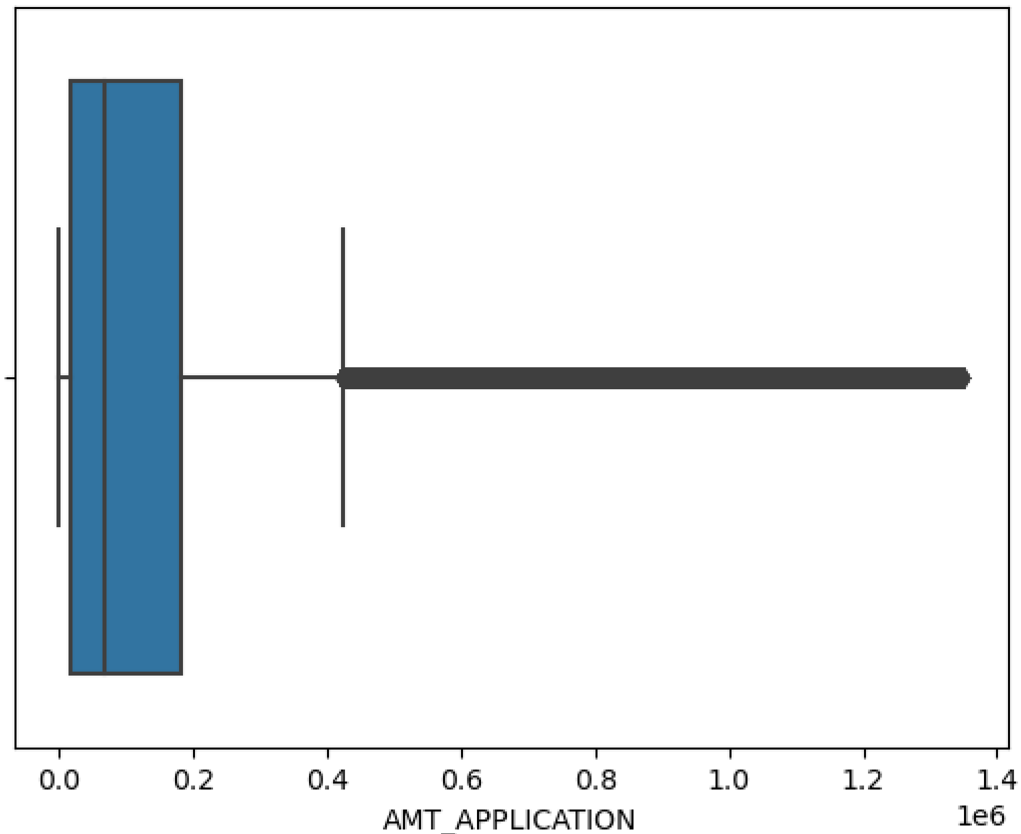
In [199]:

```
#Handling the outliers by taking first 99th percentile values
previous_app["AMT_APPLICATION"]=(previous_app[previous_app["AMT_APPLICATION"]<previous_app[
```

In [200]:

```
sns.boxplot(previous_app["AMT_APPLICATION"])
plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



After removing the outliers and considering the 99% of AMT_APPLICATION. we can remove those rows which have **AMT_APPLICATION** value as 0.

In [201]:

```
previous_app=previous_app[~(previous_app["AMT_APPLICATION"]==0.0)]
```

In [202]:

```
previous_app.shape
```

Out[202]:

```
(1277812, 9)
```

Cleaning of **NAME_CONTRACT_TYPE** column

In [203]:

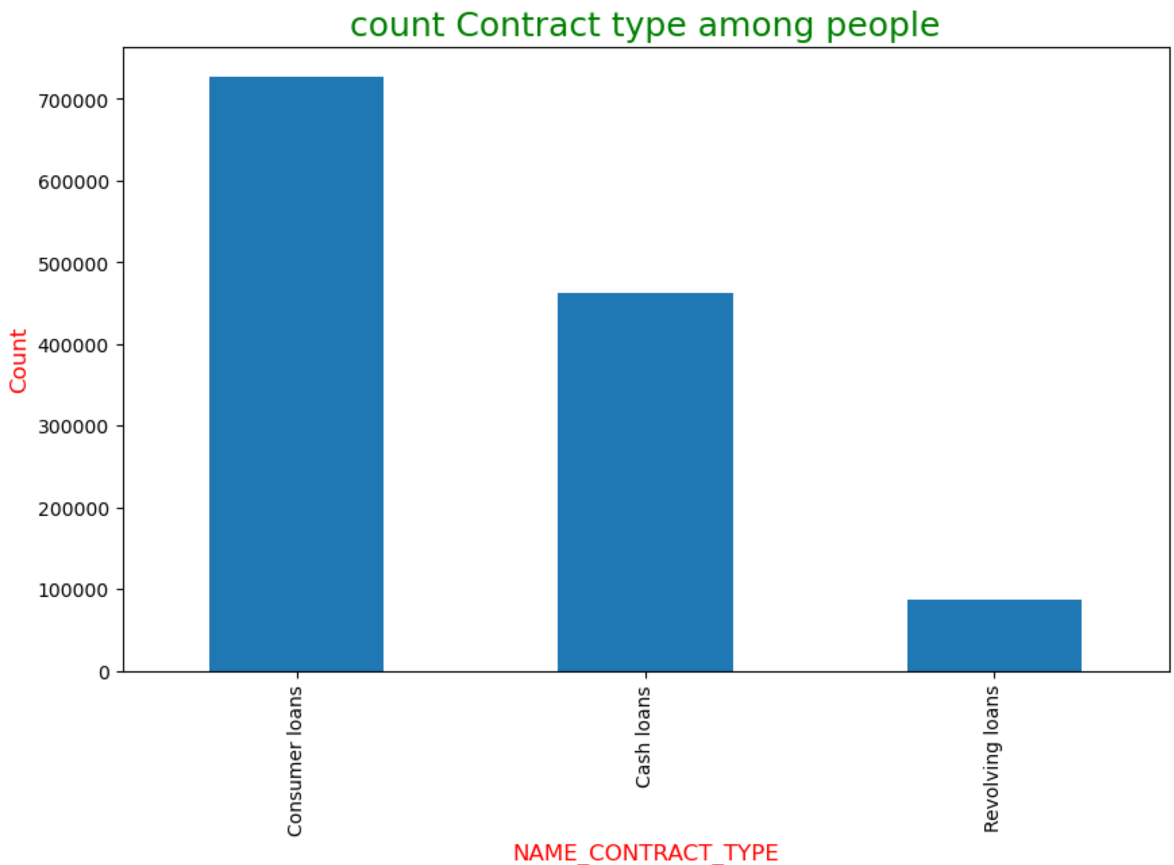
```
#Checking the percentage value counts in NAME_CONTRACT_TYPE column
previous_app["NAME_CONTRACT_TYPE"].value_counts(normalize=True)*100
```

Out[203]:

```
Consumer loans     56.955640
Cash loans         36.174336
Revolving loans     6.870025
Name: NAME_CONTRACT_TYPE, dtype: float64
```

In [206]:

```
plt.figure(figsize = (10,6))
plt.title('count Contract type among people',fontdict={'fontsize':18,'color':'Green'})
previous_app.NAME_CONTRACT_TYPE.value_counts().plot.bar()
plt.xlabel('NAME_CONTRACT_TYPE',fontdict={'fontsize':12,'fontweight':5,'color':'Red'})
plt.ylabel('Count',fontdict={'fontsize':12,'fontweight':5,'color':'Red'})
plt.show()
```



OBSERVATION

- There is no null value in **NAME_CONTRACT_TYPE** column

Cleaning process in **AMT_CREDIT** Column

In [207]:

```
#Checking out the value percentage in DataFrame
previous_app["AMT_CREDIT"].value_counts(normalize=True)*100
```

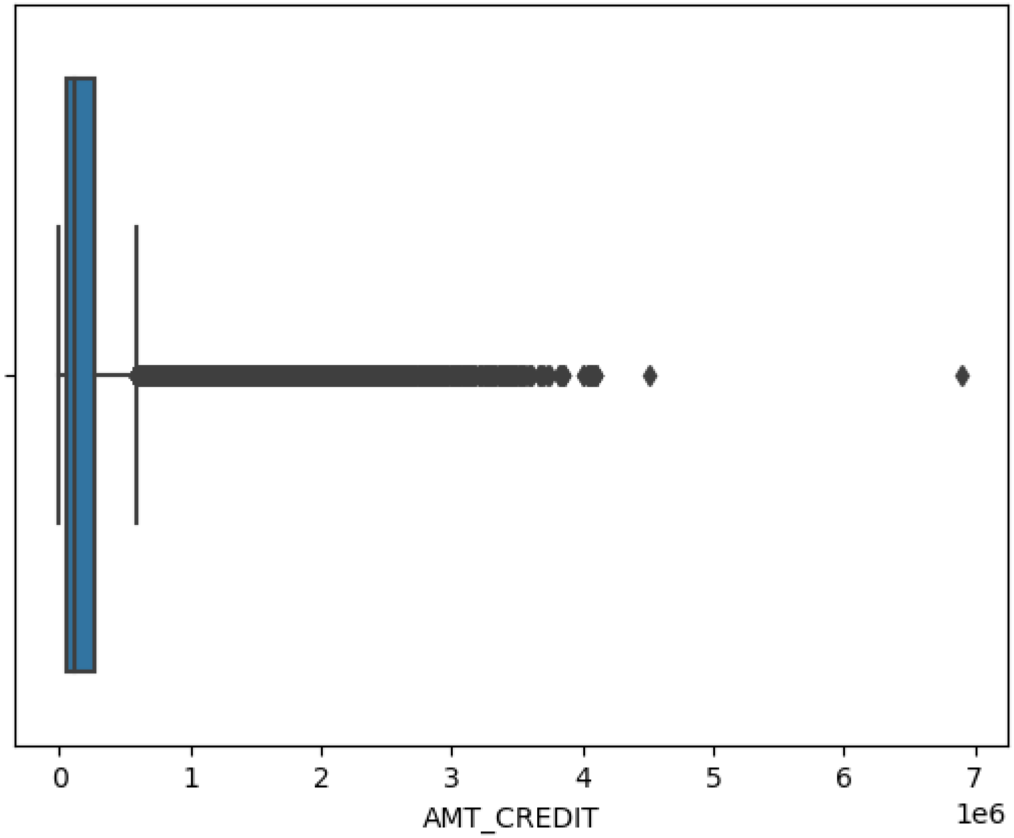
Out[207]:

```
45000.00     2.518133
225000.00    1.327895
135000.00    1.016034
450000.00    0.893793
180000.00    0.755432
               ...
337315.50    0.000078
412110.00    0.000078
331731.00    0.000078
338301.00    0.000078
436370.22    0.000078
Name: AMT_CREDIT, Length: 86803, dtype: float64
```

In [208]:

```
#Checking out the outliers in DataFrame using boxplot
sns.boxplot(previous_app["AMT_CREDIT"])

plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



OBSERVATION

- As we can see from the above boxplot there are some outliers present in the data.
- There are some null value present in data.

In [209]:

```
#Checking out the percentage composition on the data
previous_app["AMT_CREDIT"].quantile([0,.25,.50,.75,.99,1])
```

Out[209]:

```
0.00          0.00
0.25      50553.00
0.50     112500.00
0.75     269550.00
0.99    1557135.54
1.00    6905160.00
Name: AMT_CREDIT, dtype: float64
```

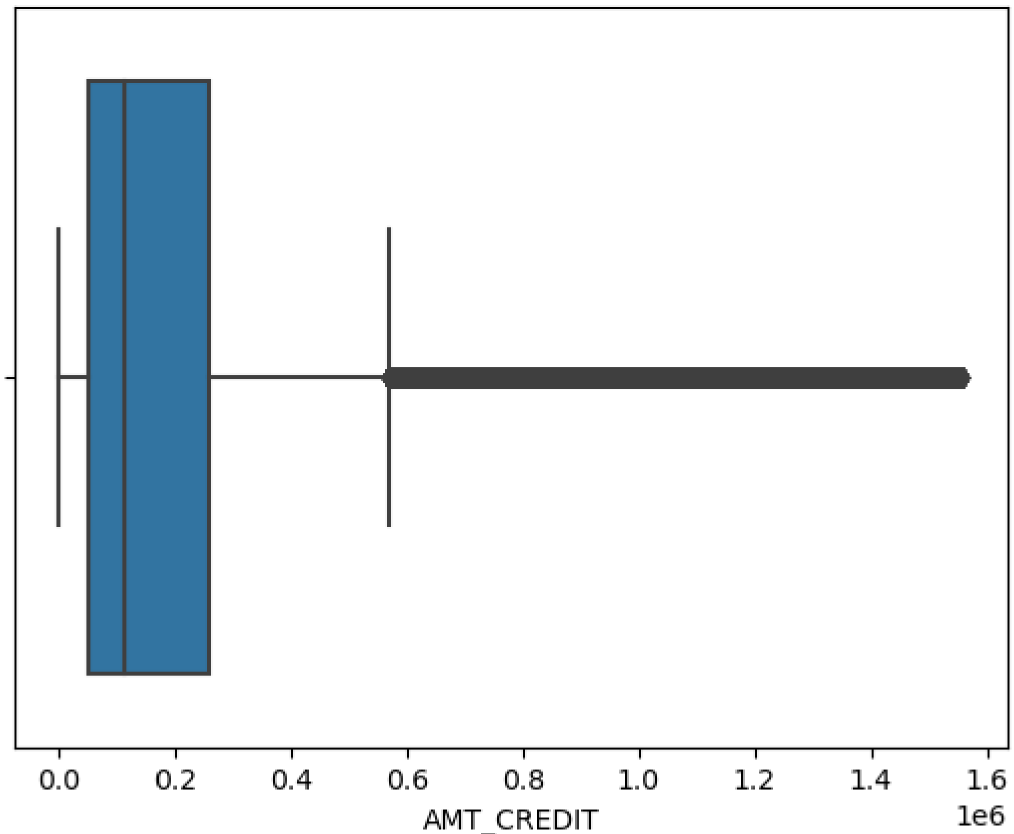
In [210]:

```
#Outliers handling by taking the 99th percentile of the data
previous_app["AMT_CREDIT"]=(previous_app[previous_app["AMT_CREDIT"]<previous_app["AMT_CREDI
```

In [211]:

```
sns.boxplot(previous_app["AMT_CREDIT"])
plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [212]:

```
#Checking out the null values present in the data
previous_app["AMT_CREDIT"].isnull().sum()
```

Out[212]:

12779

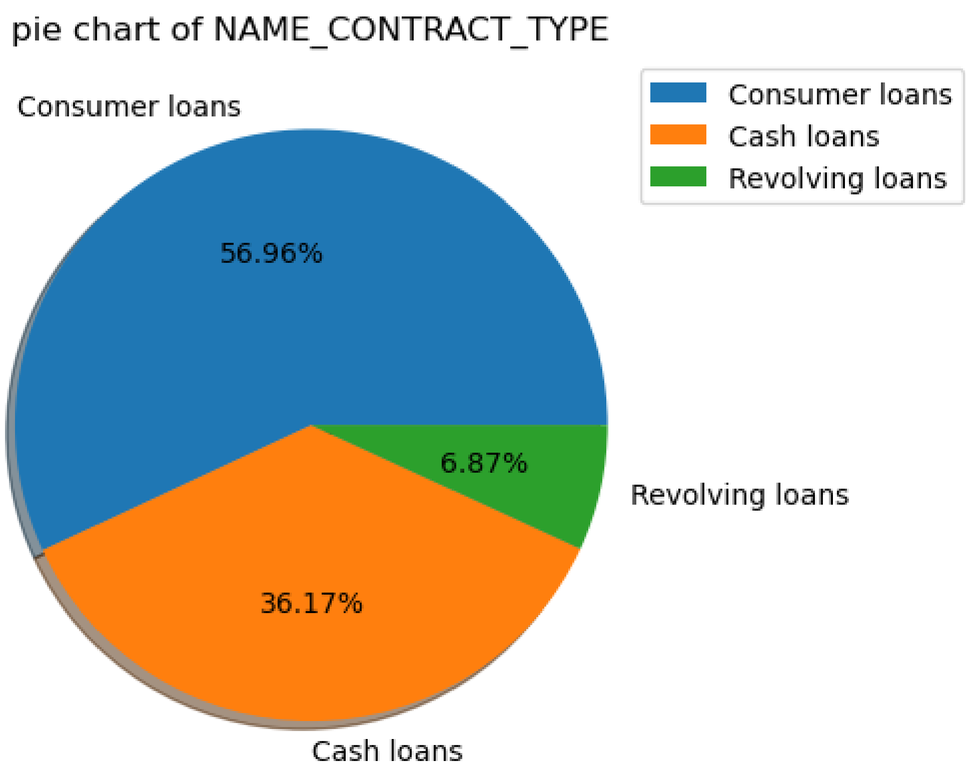
In [213]:

```
previous_app["AMT_CREDIT"]=previous_app["AMT_CREDIT"].fillna(previous_app["AMT_CREDIT"].med
```

Univariate Analysis

In [218]:

```
#Univariate Analysis of NAME_CONTRACT_TYPE column
previous_app["NAME_CONTRACT_TYPE"].value_counts(normalize=True).plot.pie(autopct='%1.02f%%'
plt.legend(loc="upper right",bbox_to_anchor=(1.4,1))

plt.ylabel('')
plt.show()
```

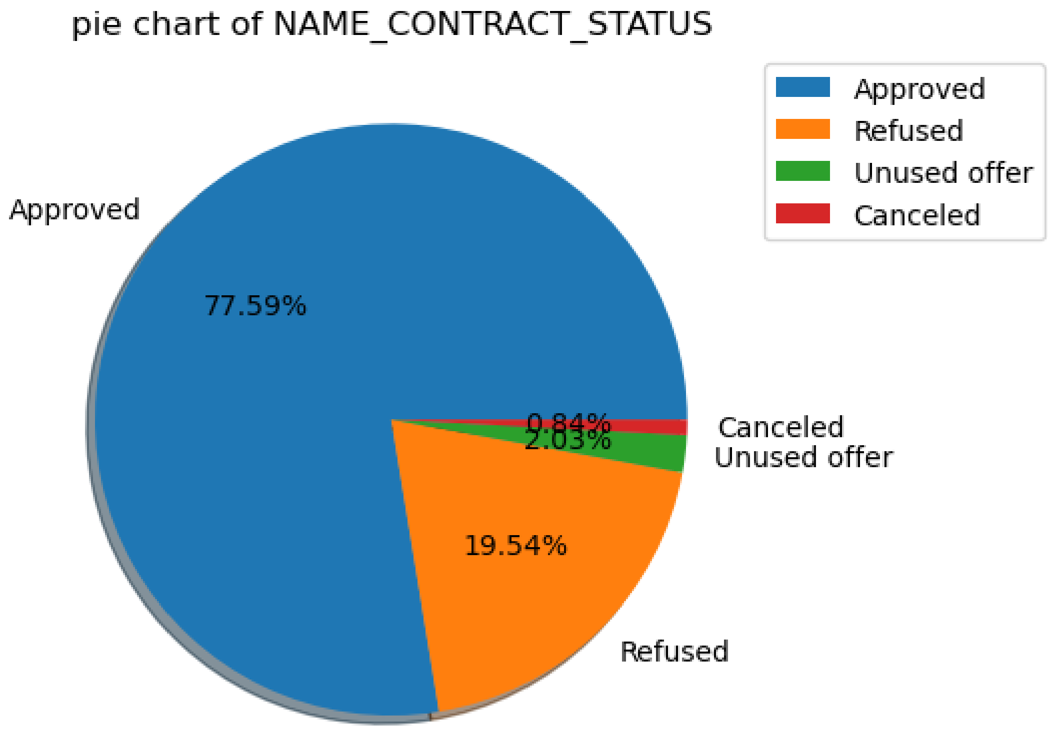


Observation

1. Maximum people approx 57% demands for Consumer Loans
2. only 7% people demands for Revolving Loans which is the least percentage

In [219]:

```
#Univariate Analysis of NAME_CONTRACT_STATUS column
previous_app["NAME_CONTRACT_STATUS"].value_counts(normalize=True).plot.pie(autopct='%1.02f%
plt.legend(loc="upper right",bbox_to_anchor=(1.4,1))

plt.ylabel('')
plt.show()
```



In [ ]:

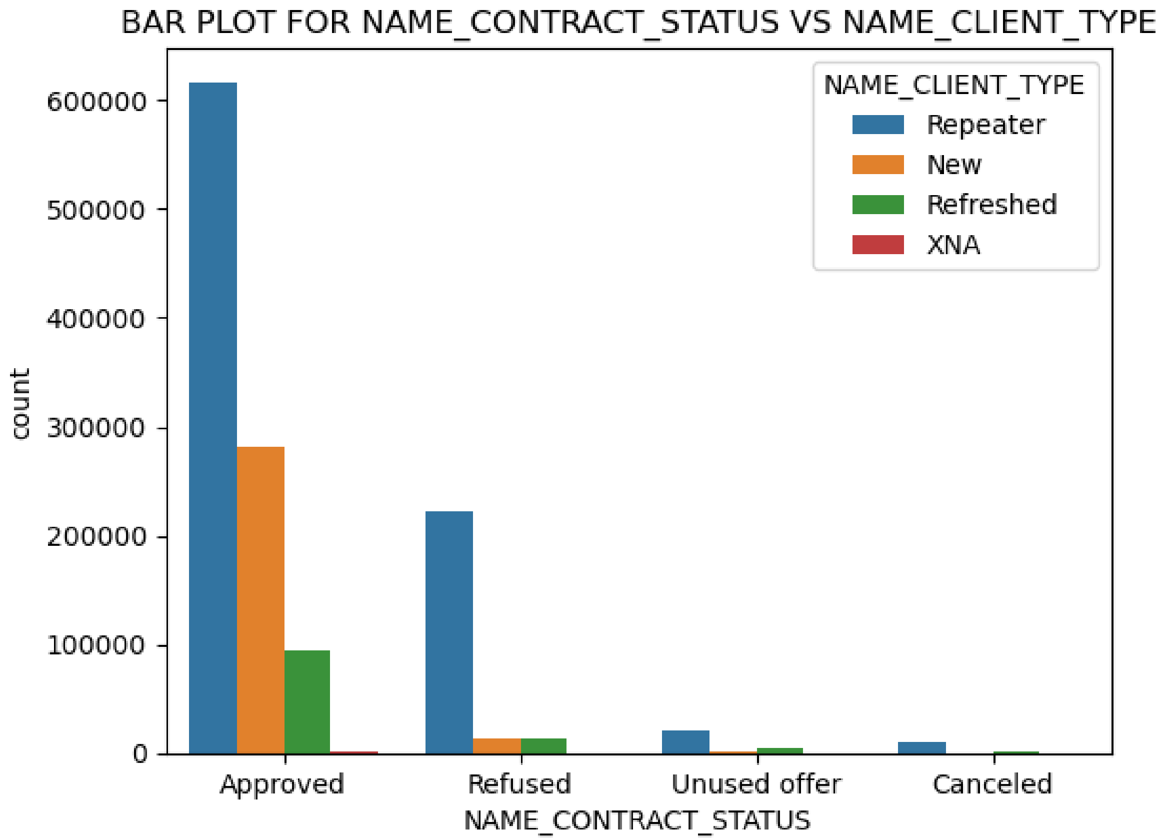
Observation

1. Maximum loans has been approved by the banks(approx=78%)
2. Canceled loans percentage is very less approx 1%
3. Loans Refused percentage is approx 20%

Bi-VARIATE ANALYSIS

In [224]:

```
#Bi-Variate Analysis of NAME_CONTRACT_STATUS vs NAME_CLIENT_TYPE
sns.countplot(data=previous_app,x="NAME_CONTRACT_STATUS",hue="NAME_CLIENT_TYPE")
plt.title("BAR PLOT FOR NAME_CONTRACT_STATUS VS NAME_CLIENT_TYPE")

plt.show()
```



Observation

1. The loan approval and refused rate for the repeaters is much higher than any other client types

MERGING BOTH THE DATAFRAMES

In [225]:

```
#Merging both the DataFrames
final=pd.merge(df2,previous_app,how='left',on='SK_ID_CURR')

final.head(5)
```

Out[225]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100003	0	Cash loans	F	N	
3	100003	0	Cash loans	F	N	
4	100004	0	Revolving loans	M	Y	

5 rows × 82 columns

In [226]:

```
#Checking the column values for the merged DataFrame
final.columns
```

Out[226]:

```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE_x', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTA
L',
       'AMT_CREDIT_x', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'AGE',
       'WORK_EXPERIENCE', 'YEARS_REGISTRATION', 'ID_CHANGE_YEAR', 'FLAG_MOBI
L',
       'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHON
E',
       'FLAG_EMAIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS',
       'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY',
       'ORGANIZATION_TYPE', 'EXT_SOURCE_2', 'EXT_SOURCE_3',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'YEARS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3',
       'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6',
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9',
       'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12',
       'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15',
       'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18',
       'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR', 'SK_ID_PRE
V',
       'NAME_CONTRACT_TYPE_y', 'AMT_APPLICATION', 'AMT_CREDIT_y',
       'FLAG_LAST_APPL_PER_CONTRACT', 'NAME_CONTRACT_STATUS', 'DAYS_DECISIO
N',
       'NAME_CLIENT_TYPE', 'YEAR_DECISION'],
      dtype='object')
```

In [227]:

```
#Dropping the Unused columns from final DataFrame
final.drop(['YEARS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3',
       'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6',
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9',
       'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12',
       'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15',
       'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18',
       'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR','ID_CHANGE_YEAR', 'FLAG_MO
       'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE','FLAG_EMAIL','
       'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY','REGION_POPULATION_RELATIVE'],ax
```

In [228]:

```
final.head(10)
```

Out[228]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100003	0	Cash loans	F	N	
3	100003	0	Cash loans	F	N	
4	100004	0	Revolving loans	M	Y	
5	100006	0	Cash loans	F	N	
6	100006	0	Cash loans	F	N	
7	100006	0	Cash loans	F	N	
8	100006	0	Cash loans	F	N	
9	100006	0	Cash loans	F	N	

10 rows × 39 columns

UNIVARIATE ANALYSIS ON CATAGORICAL COLUMNS

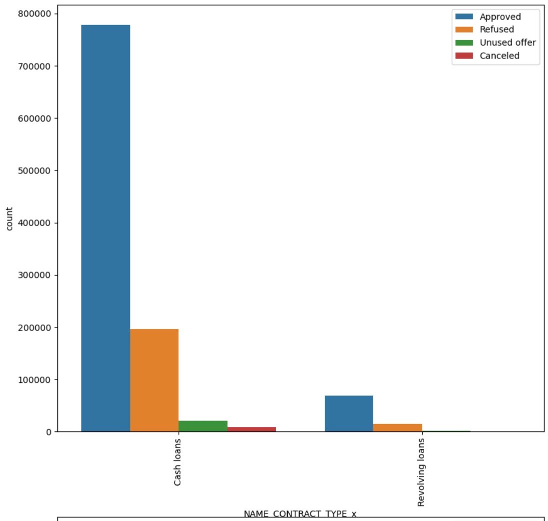
In [229]:

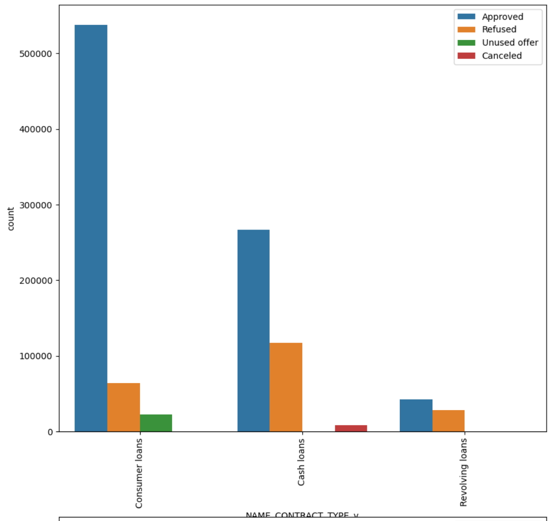
```
final.info()
```

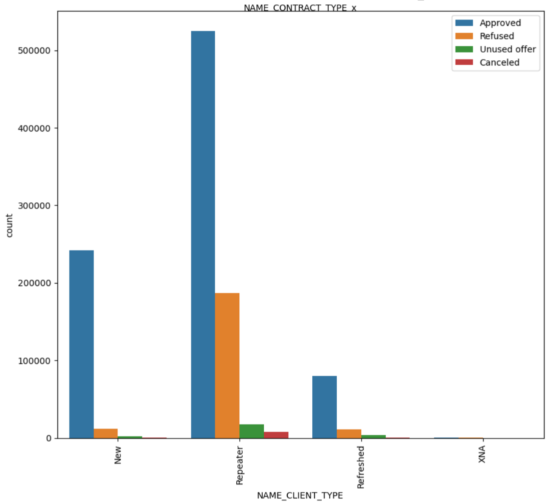
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1105806 entries, 0 to 1105805
Data columns (total 39 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   SK_ID_CURR                   1105806 non-null  int64  
 1   TARGET                       1105806 non-null  int64  
 2   NAME_CONTRACT_TYPE_x         1105806 non-null  object 
 3   CODE_GENDER                  1105806 non-null  object 
 4   FLAG_OWN_CAR                 1105806 non-null  object 
 5   FLAG_OWN_REALTY              1105806 non-null  object 
 6   CNT_CHILDREN                 1105806 non-null  int64  
 7   AMT_INCOME_TOTAL             1105806 non-null  float64
 8   AMT_CREDIT_x                 1105806 non-null  float64
 9   AMT_ANNUITY                  1105806 non-null  float64
 10  AMT_GOODS_PRICE              1105806 non-null  float64
 11  NAME_TYPE_SUITE              1105806 non-null  object 
 12  NAME_INCOME_TYPE             1105806 non-null  object 
 13  NAME_EDUCATION_TYPE          1105806 non-null  object 
 14  NAME_FAMILY_STATUS           1105806 non-null  object 
 15  NAME_HOUSING_TYPE            1105806 non-null  object 
 16  AGE                          1105806 non-null  float64
 17  WORK_EXPERIENCE              1105806 non-null  float64
 18  YEARS_REGISTRATION           1105806 non-null  float64
 19  OCCUPATION_TYPE              1105806 non-null  object 
 20  CNT_FAM_MEMBERS              1105804 non-null  float64
 21  REGION_RATING_CLIENT         1105806 non-null  int64  
 22  REGION_RATING_CLIENT_W_CITY  1105806 non-null  int64  
 23  ORGANIZATION_TYPE            1105806 non-null  object 
 24  EXT_SOURCE_2                 1104113 non-null  float64
 25  EXT_SOURCE_3                 916911 non-null   float64
 26  OBS_30_CNT_SOCIAL_CIRCLE     1103079 non-null  float64
 27  DEF_30_CNT_SOCIAL_CIRCLE     1105806 non-null  float64
 28  OBS_60_CNT_SOCIAL_CIRCLE     1105806 non-null  float64
 29  DEF_60_CNT_SOCIAL_CIRCLE     1105806 non-null  float64
 30  SK_ID_PREV                   1088337 non-null  float64
 31  NAME_CONTRACT_TYPE_y         1088337 non-null  object 
 32  AMT_APPLICATION              1067450 non-null  float64
 33  AMT_CREDIT_y                 1088337 non-null  float64
 34  FLAG_LAST_APPL_PER_CONTRACT  1088337 non-null  object 
 35  NAME_CONTRACT_STATUS         1088337 non-null  object 
 36  DAYS_DECISION                1088337 non-null  float64
 37  NAME_CLIENT_TYPE             1088337 non-null  object 
 38  YEAR_DECISION                1088337 non-null  float64
dtypes: float64(19), int64(5), object(15)
memory usage: 337.5+ MB
```

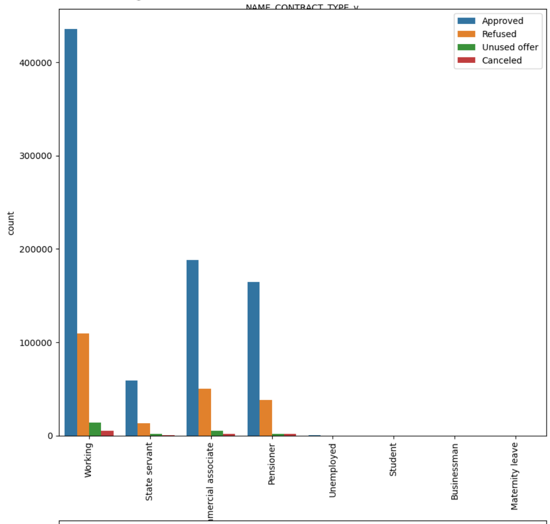
In [230]:

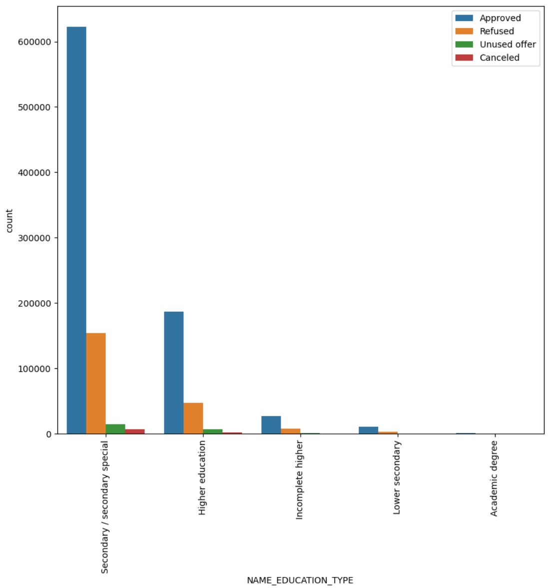
```
categorical=["NAME_CONTRACT_TYPE_x","NAME_CONTRACT_TYPE_y","NAME_CLIENT_TYPE","NAME_INCOME_

plt.figure(figsize=(22,30))


for i in (enumerate(categorical)):
    plt.subplot(len(categorical)//2,2,i[0]+1)
    sns.countplot(x=i[1],hue='NAME_CONTRACT_STATUS',data=final)
    plt.xticks(rotation=90)
    plt.legend(loc="upper right")
plt.show()
```

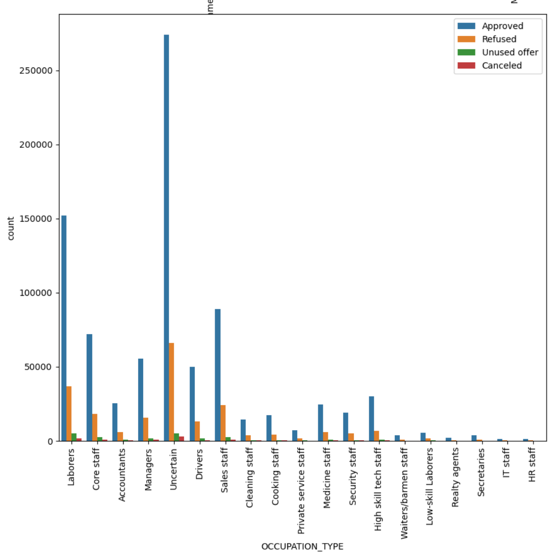












Observation

1. Loan aprroval rates for **Consumer Loans** is much higher than any other loan.
2. Banks like to give loans to the **Repeaters**.
3. People with **Secondary Education or more** receives loan approval easily.
4. Occupation_type **Laborers** get the more loans then others.
5. **Working** class people receives more loan approvals than any other Income_type

UNIVARIATE ANALYSIS ON NUMERICAL

COLUMNS

In [231]:

```
Numerical=["CNT_CHILDREN","CNT_FAM_MEMBERS","AMT_CREDIT_x","AMT_CREDIT_y","AMT_GOODS_PRICE"

plt.figure(figsize=(22,25))
for i in (enumerate(Numerical)):
    plt.subplot(len(Numerical)//2,2,i[0]+1)
    sns.distplot(final.loc[final.NAME_CONTRACT_STATUS=='Approved',:][i[1]].dropna(),hist=Fa
    sns.distplot(final.loc[final.NAME_CONTRACT_STATUS=='Canceled',:][i[1]].dropna(),hist=Fa
    sns.distplot(final.loc[final.NAME_CONTRACT_STATUS=='Refused',:][i[1]].dropna(),hist=Fal
    # we added kde_kws={'bw':0.1} in parameter to overcome bandwidth limitation.
    sns.distplot(final.loc[final.NAME_CONTRACT_STATUS=='Unused offer',:][i[1]].dropna(),his
    plt.legend(["Approved", "Canceled","Refused","Unused offer"], loc ="upper right")
plt.show()
```

```
C:\Users\somes\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figur
e-level function with similar flexibility) or `kdeplot` (an axes-level fu
nction for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\somes\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figur
e-level function with similar flexibility) or `kdeplot` (an axes-level fu
nction for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\somes\anaconda3\lib\site-packages\seaborn\distributions.py:1699:
FutureWarning: The `bw` parameter is deprecated in favor of `bw_method` a
nd `bw_adjust`. Using 0.1 for `bw_method`, but please see the docs for th
e new parameters and update your code.
  warnings.warn(msg, FutureWarning)
C:\Users\somes\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in
```

Observation

1. Loan cancellation rate is higher for a person if he has less number of child or no child.
2. Loan approval rate is higher for the family which has family member more than 2.
3. Previously bank has more **AMT_CREDIT** for unused offers but now it has more **AMT_CREDIT** for approved loans.
4. Loan approval rate is high for the loans which has **AMT_GOODS_PRICE** less than 1 lacs.
5. **AMT_APPLICATION** is high for unused offers.

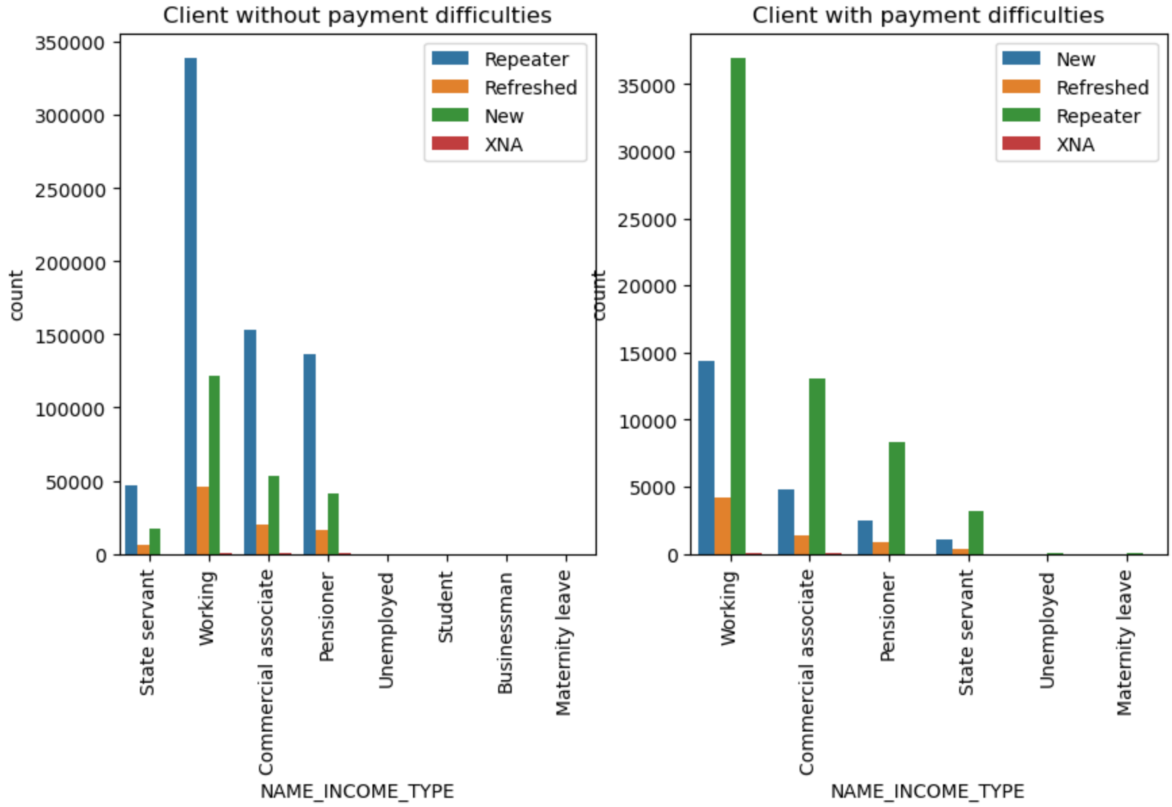
BI-VARIATE ANALYSIS ON FINAL DATAFRAME

In [238]:

```
Target_0 = final[final.TARGET==0]
Target_1 = final[final.TARGET==1]
```

In [233]:

```
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
ax = sns.countplot(data=Target_0,x='NAME_INCOME_TYPE',hue='NAME_CLIENT_TYPE')
plt.title('Client without payment difficulties')
plt.xticks(rotation=90)
plt.legend(loc="upper right")

plt.subplot(1,2,2)
ax = sns.countplot(data=Target_1,x='NAME_INCOME_TYPE',hue='NAME_CLIENT_TYPE')
plt.title('Client with payment difficulties')
plt.xticks(rotation=90)
plt.legend(loc="upper right")

plt.show()
```

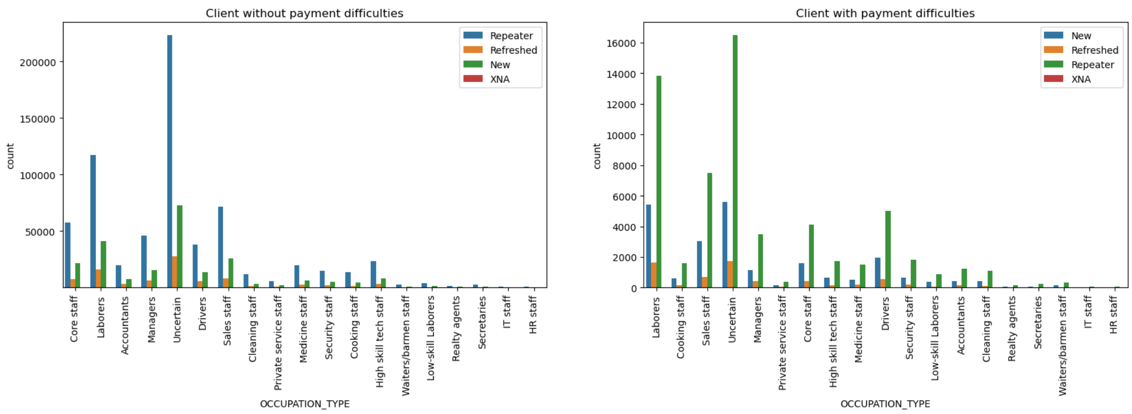


Obervations

1. Working Class people take most of the loans
2. Unemployed, Students, Businessman, Matrnity leaves people don't take loans

In [234]:

```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
ax = sns.countplot(data=Target_0,x='OCCUPATION_TYPE',hue='NAME_CLIENT_TYPE')
plt.title('Client without payment difficulties')
plt.xticks(rotation=90)
plt.legend(loc="upper right")
plt.ylim(10)

plt.subplot(1,2,2)
ax = sns.countplot(data=Target_1,x='OCCUPATION_TYPE',hue='NAME_CLIENT_TYPE')
plt.title('Client with payment difficulties')
plt.xticks(rotation=90)
plt.legend(loc="upper right")

plt.show()
```

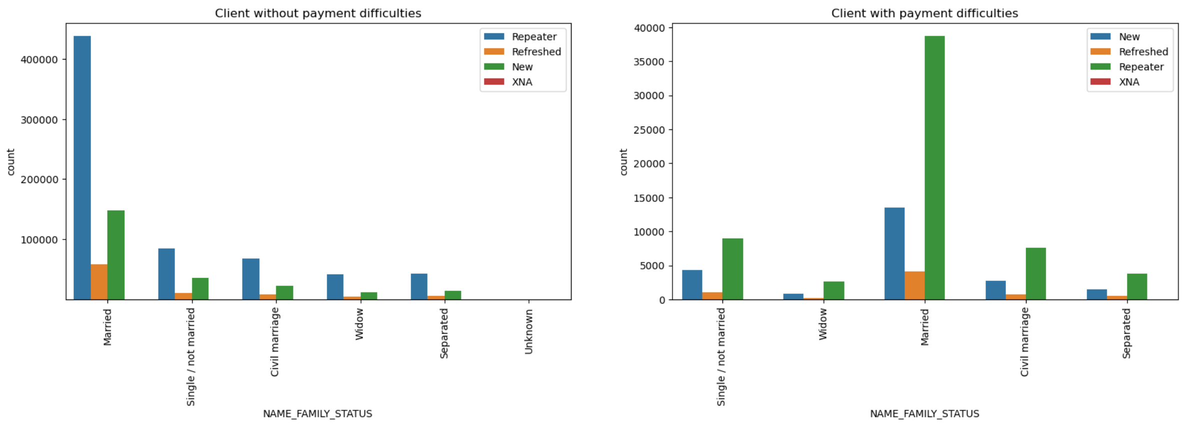


Observation

1. Number of Repeaters are very large in both the payment with difficulties and payment without difficulties.
2. IT and HR staff are very less which apply for the loans.

In [235]:

```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
ax = sns.countplot(data=Target_0,x='NAME_FAMILY_STATUS',hue='NAME_CLIENT_TYPE')
plt.title('Client without payment difficulties')
plt.xticks(rotation=90)
plt.legend(loc="upper right")
plt.ylim(10)

plt.subplot(1,2,2)
ax = sns.countplot(data=Target_1,x='NAME_FAMILY_STATUS',hue='NAME_CLIENT_TYPE')
plt.title('Client with payment difficulties')
plt.xticks(rotation=90)
plt.legend(loc="upper right")

plt.show()
```



In [ ]:

Observations

1. Married people are the repeators of the loans.
2. Widows and Separated people don't apply much for the loans.

Conclusion

1. Top most category which is facing the highest payment difficulties is married and working class people.

2. Loan cancellation rate is higher for a person if he has less number of child or no child.

3. These are some major variable which can be considered as loan predictors:-

- NAME_FAMILY_STATUS
- AMT_CREDIT
- OCCUPATION_TYPE
- NAME_INCOME_TYPE
- CNT_FAM_MEMBERS
- CNT_GOODS_PRICE

```
In [ ]:
```