In [15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import math
warnings.filterwarnings("ignore")
pd.set_option('display.max_columns',125)
pd.set_option('display.max_rows',125)
```

Loading the Dataset

In [16]:
```
df_application_current = pd.read_csv('C:/Users/SAGNICK/TrainityBankLoneEDA/LoanCaseSt
df_application_current.head()
```

Out[16]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REAL
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	

Getting the Dataframe dimensions

In [17]:
```
df_application_current.shape
```

Out[17]: (307511, 122)

Basic Info of the Data Frame

In [18]:
```
df_application_current.info()
```

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

Getting the statistical information

In [19]:
```
df_application_current.describe()
```

Out[19]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANN

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANN
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	307499.00
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	27108.57
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	14493.73
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1615.50
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16524.00
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	24903.00
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.00

Getting the column names

In [14]:

```
column_names=[]
column_names=df_application_current.columns
for cols in column_names:
    print(cols)
```

```
SK_ID_CURR
TARGET
NAME_CONTRACT_TYPE
CODE_GENDER
FLAG_OWN_CAR
FLAG_OWN_REALTY
CNT_CHILDREN
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
NAME_TYPE_SUITE
NAME_INCOME_TYPE
NAME_EDUCATION_TYPE
NAME_FAMILY_STATUS
NAME_HOUSING_TYPE
REGION_POPULATION_RELATIVE
DAYS_BIRTH
DAYS_EMPLOYED
DAYS_REGISTRATION
DAYS_ID_PUBLISH
OWN_CAR_AGE
FLAG_MOBIL
FLAG_EMP_PHONE
FLAG_WORK_PHONE
FLAG_CONT_MOBILE
FLAG_PHONE
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_1
EXT_SOURCE_2
EXT_SOURCE_3
APARTMENTS_AVG
BASEMENTAREA_AVG
YEARS_BEGINEXPLUATATION_AVG
YEARS_BUILD_AVG
COMMONAREA_AVG
ELEVATORS_AVG
ENTRANCES_AVG
FLOORSMAX_AVG
FLOORSMIN_AVG
LANDAREA_AVG
LIVINGAPARTMENTS_AVG
LIVINGAREA_AVG
NONLIVINGAPARTMENTS_AVG
NONLIVINGAREA_AVG
APARTMENTS_MODE
BASEMENTAREA_MODE
YEARS_BEGINEXPLUATATION_MODE
YEARS_BUILD_MODE
COMMONAREA_MODE
ELEVATORS_MODE
ENTRANCES_MODE
FLOORSMAX_MODE
FLOORSMIN_MODE
LANDAREA_MODE
LIVINGAPARTMENTS_MODE
LIVINGAREA_MODE
NONLIVINGAPARTMENTS_MODE
NONLIVINGAREA_MODE
APARTMENTS_MEDI
BASEMENTAREA_MEDI
YEARS_BEGINEXPLUATATION_MEDI
YEARS_BUILD_MEDI
COMMONAREA_MEDI
ELEVATORS_MEDI
ENTRANCES_MEDI
FLOORSMAX_MEDI
FLOORSMIN_MEDI
LANDAREA_MEDI
LIVINGAPARTMENTS_MEDI
LIVINGAREA_MEDI
NONLIVINGAPARTMENTS_MEDI
NONLIVINGAREA_MEDI
FONDKAPREMONT_MODE
HOUSETYPE_MODE
TOTALAREA_MODE
WALLSMATERIAL_MODE
EMERGENCYSTATE_MODE
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE
DEF_60_CNT_SOCIAL_CIRCLE
DAYS_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
```

Handling missing values in columns

Creating Copy of the DataFrame

In [22]:
```
df_application_current_copy=df_application_current.copy(deep=True)
df_application_current_copy
```

Out[22]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_
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	
...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	
307507	456252	0	Cash loans	F	N	
307508	456253	0	Cash loans	F	N	
307509	456254	1	Cash loans	F	N	
307510	456255	0	Cash loans	F	N	

307511 rows × 122 columns

Count missing values column wise

In [23]:
```
df_application_current_copy.isnull().sum()
```

Out[23]:
```
SK_ID_CURR                        0
TARGET                            0
NAME_CONTRACT_TYPE                0
CODE_GENDER                       0
FLAG_OWN_CAR                      0
FLAG_OWN_REALTY                   0
CNT_CHILDREN                      0
AMT_INCOME_TOTAL                  0
AMT_CREDIT                        0
AMT_ANNUITY                      12
AMT_GOODS_PRICE                 278
NAME_TYPE_SUITE                1292
NAME_INCOME_TYPE                  0
NAME_EDUCATION_TYPE               0
NAME_FAMILY_STATUS                0
NAME_HOUSING_TYPE                 0
REGION_POPULATION_RELATIVE        0
DAYS_BIRTH                        0
DAYS_EMPLOYED                     0
```

```
DAYS_REGISTRATION                  0
DAYS_ID_PUBLISH                    0
OWN_CAR_AGE                   202929
FLAG_MOBIL                         0
FLAG_EMP_PHONE                     0
FLAG_WORK_PHONE                    0
FLAG_CONT_MOBILE                   0
FLAG_PHONE                         0
FLAG_EMAIL                         0
OCCUPATION_TYPE                96391
CNT_FAM_MEMBERS                    2
REGION_RATING_CLIENT               0
REGION_RATING_CLIENT_W_CITY        0
WEEKDAY_APPR_PROCESS_START         0
HOUR_APPR_PROCESS_START            0
REG_REGION_NOT_LIVE_REGION         0
REG_REGION_NOT_WORK_REGION         0
LIVE_REGION_NOT_WORK_REGION        0
REG_CITY_NOT_LIVE_CITY             0
REG_CITY_NOT_WORK_CITY             0
LIVE_CITY_NOT_WORK_CITY            0
ORGANIZATION_TYPE                  0
EXT_SOURCE_1                  173378
EXT_SOURCE_2                     660
EXT_SOURCE_3                   60965
APARTMENTS_AVG                156061
BASEMENTAREA_AVG              179943
YEARS_BEGINEXPLUATATION_AVG   150007
YEARS_BUILD_AVG               204488
COMMONAREA_AVG                214865
ELEVATORS_AVG                 163891
ENTRANCES_AVG                 154828
FLOORSMAX_AVG                 153020
FLOORSMIN_AVG                 208642
LANDAREA_AVG                  182590
LIVINGAPARTMENTS_AVG          210199
LIVINGAREA_AVG                154350
NONLIVINGAPARTMENTS_AVG       213514
NONLIVINGAREA_AVG             169682
APARTMENTS_MODE               156061
BASEMENTAREA_MODE             179943
YEARS_BEGINEXPLUATATION_MODE  150007
YEARS_BUILD_MODE              204488
COMMONAREA_MODE               214865
ELEVATORS_MODE                163891
ENTRANCES_MODE                154828
FLOORSMAX_MODE                153020
FLOORSMIN_MODE                208642
LANDAREA_MODE                 182590
LIVINGAPARTMENTS_MODE         210199
LIVINGAREA_MODE               154350
NONLIVINGAPARTMENTS_MODE      213514
NONLIVINGAREA_MODE            169682
APARTMENTS_MEDI               156061
BASEMENTAREA_MEDI             179943
YEARS_BEGINEXPLUATATION_MEDI  150007
YEARS_BUILD_MEDI              204488
COMMONAREA_MEDI               214865
ELEVATORS_MEDI                163891
ENTRANCES_MEDI                154828
FLOORSMAX_MEDI                153020
FLOORSMIN_MEDI                208642
```

```
LANDAREA_MEDI                   182590
LIVINGAPARTMENTS_MEDI           210199
LIVINGAREA_MEDI                 154350
NONLIVINGAPARTMENTS_MEDI        213514
NONLIVINGAREA_MEDI              169682
FONDKAPREMONT_MODE              210295
HOUSETYPE_MODE                  154297
TOTALAREA_MODE                  148431
WALLSMATERIAL_MODE              156341
EMERGENCYSTATE_MODE             145755
OBS_30_CNT_SOCIAL_CIRCLE          1021
DEF_30_CNT_SOCIAL_CIRCLE          1021
OBS_60_CNT_SOCIAL_CIRCLE          1021
DEF_60_CNT_SOCIAL_CIRCLE          1021
DAYS_LAST_PHONE_CHANGE               1
FLAG_DOCUMENT_2                      0
FLAG_DOCUMENT_3                      0
FLAG_DOCUMENT_4                      0
FLAG_DOCUMENT_5                      0
FLAG_DOCUMENT_6                      0
FLAG_DOCUMENT_7                      0
FLAG_DOCUMENT_8                      0
FLAG_DOCUMENT_9                      0
FLAG_DOCUMENT_10                     0
FLAG_DOCUMENT_11                     0
FLAG_DOCUMENT_12                     0
FLAG_DOCUMENT_13                     0
FLAG_DOCUMENT_14                     0
FLAG_DOCUMENT_15                     0
FLAG_DOCUMENT_16                     0
FLAG_DOCUMENT_17                     0
FLAG_DOCUMENT_18                     0
FLAG_DOCUMENT_19                     0
FLAG_DOCUMENT_20                     0
FLAG_DOCUMENT_21                     0
AMT_REQ_CREDIT_BUREAU_HOUR       41519
AMT_REQ_CREDIT_BUREAU_DAY        41519
AMT_REQ_CREDIT_BUREAU_WEEK       41519
AMT_REQ_CREDIT_BUREAU_MON        41519
AMT_REQ_CREDIT_BUREAU_QRT        41519
AMT_REQ_CREDIT_BUREAU_YEAR       41519
```

In [24]:

```
rows,cols=df_application_current_copy.shape
rows
```

Out[24]: 307511

Calculating Percetage of NA values in each column

In [42]:

```
Perc_Of_NA_Columns=round(df_application_current_copy.isnull().sum()/rows*100,2)
Perc_Of_NA_Columns
```

Out[42]:

```
SK_ID_CURR                      0.00
TARGET                          0.00
NAME_CONTRACT_TYPE              0.00
CODE_GENDER                     0.00
FLAG_OWN_CAR                    0.00
FLAG_OWN_REALTY                 0.00
CNT_CHILDREN                    0.00
```

```
AMT_INCOME_TOTAL                 0.00
AMT_CREDIT                       0.00
AMT_ANNUITY                      0.00
AMT_GOODS_PRICE                  0.09
NAME_TYPE_SUITE                  0.42
NAME_INCOME_TYPE                 0.00
NAME_EDUCATION_TYPE              0.00
NAME_FAMILY_STATUS               0.00
NAME_HOUSING_TYPE                0.00
REGION_POPULATION_RELATIVE       0.00
DAYS_BIRTH                       0.00
DAYS_EMPLOYED                    0.00
DAYS_REGISTRATION                0.00
DAYS_ID_PUBLISH                  0.00
OWN_CAR_AGE                     65.99
FLAG_MOBIL                       0.00
FLAG_EMP_PHONE                   0.00
FLAG_WORK_PHONE                  0.00
FLAG_CONT_MOBILE                 0.00
FLAG_PHONE                       0.00
FLAG_EMAIL                       0.00
OCCUPATION_TYPE                 31.35
CNT_FAM_MEMBERS                  0.00
REGION_RATING_CLIENT             0.00
REGION_RATING_CLIENT_W_CITY      0.00
WEEKDAY_APPR_PROCESS_START       0.00
HOUR_APPR_PROCESS_START          0.00
REG_REGION_NOT_LIVE_REGION       0.00
REG_REGION_NOT_WORK_REGION       0.00
LIVE_REGION_NOT_WORK_REGION      0.00
REG_CITY_NOT_LIVE_CITY           0.00
REG_CITY_NOT_WORK_CITY           0.00
LIVE_CITY_NOT_WORK_CITY          0.00
ORGANIZATION_TYPE                0.00
EXT_SOURCE_1                    56.38
EXT_SOURCE_2                     0.21
EXT_SOURCE_3                    19.83
APARTMENTS_AVG                  50.75
BASEMENTAREA_AVG                58.52
YEARS_BEGINEXPLUATATION_AVG     48.78
YEARS_BUILD_AVG                 66.50
COMMONAREA_AVG                  69.87
ELEVATORS_AVG                   53.30
ENTRANCES_AVG                   50.35
FLOORSMAX_AVG                   49.76
FLOORSMIN_AVG                   67.85
LANDAREA_AVG                    59.38
LIVINGAPARTMENTS_AVG            68.35
LIVINGAREA_AVG                  50.19
NONLIVINGAPARTMENTS_AVG         69.43
NONLIVINGAREA_AVG               55.18
APARTMENTS_MODE                 50.75
BASEMENTAREA_MODE               58.52
YEARS_BEGINEXPLUATATION_MODE    48.78
YEARS_BUILD_MODE                66.50
COMMONAREA_MODE                 69.87
ELEVATORS_MODE                  53.30
ENTRANCES_MODE                  50.35
FLOORSMAX_MODE                  49.76
FLOORSMIN_MODE                  67.85
LANDAREA_MODE                   59.38
LIVINGAPARTMENTS_MODE           68.35
```

```
LIVINGAREA_MODE                 50.19
NONLIVINGAPARTMENTS_MODE        69.43
NONLIVINGAREA_MODE              55.18
APARTMENTS_MEDI                 50.75
BASEMENTAREA_MEDI               58.52
YEARS_BEGINEXPLUATATION_MEDI    48.78
YEARS_BUILD_MEDI                66.50
COMMONAREA_MEDI                 69.87
ELEVATORS_MEDI                  53.30
ENTRANCES_MEDI                  50.35
FLOORSMAX_MEDI                  49.76
FLOORSMIN_MEDI                  67.85
LANDAREA_MEDI                   59.38
LIVINGAPARTMENTS_MEDI           68.35
LIVINGAREA_MEDI                 50.19
NONLIVINGAPARTMENTS_MEDI        69.43
NONLIVINGAREA_MEDI              55.18
FONDKAPREMONT_MODE              68.39
HOUSETYPE_MODE                  50.18
TOTALAREA_MODE                  48.27
WALLSMATERIAL_MODE              50.84
EMERGENCYSTATE_MODE             47.40
OBS_30_CNT_SOCIAL_CIRCLE         0.33
DEF_30_CNT_SOCIAL_CIRCLE         0.33
OBS_60_CNT_SOCIAL_CIRCLE         0.33
DEF_60_CNT_SOCIAL_CIRCLE         0.33
DAYS_LAST_PHONE_CHANGE           0.00
FLAG_DOCUMENT_2                  0.00
FLAG_DOCUMENT_3                  0.00
FLAG_DOCUMENT_4                  0.00
FLAG_DOCUMENT_5                  0.00
FLAG_DOCUMENT_6                  0.00
FLAG_DOCUMENT_7                  0.00
FLAG_DOCUMENT_8                  0.00
FLAG_DOCUMENT_9                  0.00
FLAG_DOCUMENT_10                 0.00
FLAG_DOCUMENT_11                 0.00
FLAG_DOCUMENT_12                 0.00
FLAG_DOCUMENT_13                 0.00
FLAG_DOCUMENT_14                 0.00
FLAG_DOCUMENT_15                 0.00
FLAG_DOCUMENT_16                 0.00
FLAG_DOCUMENT_17                 0.00
FLAG_DOCUMENT_18                 0.00
FLAG_DOCUMENT_19                 0.00
FLAG_DOCUMENT_20                 0.00
FLAG_DOCUMENT_21                 0.00
AMT_REQ_CREDIT_BUREAU_HOUR      13.50
AMT_REQ_CREDIT_BUREAU_DAY       13.50
AMT_REQ_CREDIT_BUREAU_WEEK      13.50
AMT_REQ_CREDIT_BUREAU_MON       13.50
AMT_REQ_CREDIT_BUREAU_QRT       13.50
AMT_REQ_CREDIT_BUREAU_YEAR      13.50
```

Name of Columns having atleast 50% NA values with Percentage

In [43]:

```
Columns_with_min50_NA_Values=Perc_Of_NA_Columns[Perc_Of_NA_Columns>50]
Columns_with_min50_NA_Values
```

Out[43]:

```
OWN_CAR_AGE                 65.99
EXT_SOURCE_1                56.38
APARTMENTS_AVG              50.75
```

```
BASEMENTAREA_AVG            58.52
YEARS_BUILD_AVG             66.50
COMMONAREA_AVG              69.87
ELEVATORS_AVG               53.30
ENTRANCES_AVG               50.35
FLOORSMIN_AVG               67.85
LANDAREA_AVG                59.38
LIVINGAPARTMENTS_AVG        68.35
LIVINGAREA_AVG              50.19
NONLIVINGAPARTMENTS_AVG     69.43
NONLIVINGAREA_AVG           55.18
APARTMENTS_MODE             50.75
BASEMENTAREA_MODE           58.52
YEARS_BUILD_MODE            66.50
COMMONAREA_MODE             69.87
ELEVATORS_MODE              53.30
ENTRANCES_MODE              50.35
FLOORSMIN_MODE              67.85
LANDAREA_MODE               59.38
LIVINGAPARTMENTS_MODE       68.35
LIVINGAREA_MODE             50.19
NONLIVINGAPARTMENTS_MODE    69.43
NONLIVINGAREA_MODE          55.18
APARTMENTS_MEDI             50.75
BASEMENTAREA_MEDI           58.52
YEARS_BUILD_MEDI            66.50
COMMONAREA_MEDI             69.87
ELEVATORS_MEDI              53.30
ENTRANCES_MEDI              50.35
FLOORSMIN_MEDI              67.85
LANDAREA_MEDI               59.38
LIVINGAPARTMENTS_MEDI       68.35
LIVINGAREA_MEDI             50.19
NONLIVINGAPARTMENTS_MEDI    69.43
NONLIVINGAREA_MEDI          55.18
FONDKAPREMONT_MODE          68.39
HOUSETYPE_MODE              50.18
WALLSMATERIAL_MODE          50.84
dtype: float64
```

Number of Columns having atleast 50% NA values

In [44]:

```
len(Columns_with_min50_NA_Values)
```

Out[44]:

```
41
```

Drop all the columns with 50% NA values

In [48]:

```
df_application_current_copy=df_application_current_copy.drop(Columns_with_min50_NA_Va
df_application_current_copy.shape
```

Out[48]:

```
(307511, 81)
```

In [60]:

```
round(100.0* df_application_current_copy.isnull().sum()/len(df_application_current_co
```

Out[60]:

```
SK_ID_CURR                    0.00
WEEKDAY_APPR_PROCESS_START    0.00
HOUR_APPR_PROCESS_START       0.00
```

```
REG_REGION_NOT_LIVE_REGION      0.00
REG_REGION_NOT_WORK_REGION      0.00
LIVE_REGION_NOT_WORK_REGION     0.00
REG_CITY_NOT_LIVE_CITY          0.00
REG_CITY_NOT_WORK_CITY          0.00
LIVE_CITY_NOT_WORK_CITY         0.00
ORGANIZATION_TYPE               0.00
FLAG_DOCUMENT_19                0.00
FLAG_DOCUMENT_18                0.00
FLAG_DOCUMENT_17                0.00
REGION_RATING_CLIENT_W_CITY     0.00
FLAG_DOCUMENT_16                0.00
FLAG_DOCUMENT_14                0.00
FLAG_DOCUMENT_13                0.00
FLAG_DOCUMENT_12                0.00
FLAG_DOCUMENT_11                0.00
FLAG_DOCUMENT_10                0.00
FLAG_DOCUMENT_9                 0.00
FLAG_DOCUMENT_8                 0.00
FLAG_DOCUMENT_7                 0.00
FLAG_DOCUMENT_6                 0.00
DAYS_LAST_PHONE_CHANGE          0.00
FLAG_DOCUMENT_2                 0.00
FLAG_DOCUMENT_3                 0.00
FLAG_DOCUMENT_15                0.00
FLAG_DOCUMENT_4                 0.00
REGION_RATING_CLIENT            0.00
FLAG_DOCUMENT_20                0.00
TARGET                          0.00
NAME_CONTRACT_TYPE              0.00
CODE_GENDER                     0.00
FLAG_OWN_CAR                    0.00
FLAG_OWN_REALTY                 0.00
CNT_CHILDREN                    0.00
AMT_INCOME_TOTAL                0.00
AMT_CREDIT                      0.00
AMT_ANNUITY                     0.00
FLAG_DOCUMENT_21                0.00
NAME_INCOME_TYPE                0.00
CNT_FAM_MEMBERS                 0.00
NAME_EDUCATION_TYPE             0.00
NAME_HOUSING_TYPE               0.00
REGION_POPULATION_RELATIVE      0.00
DAYS_BIRTH                      0.00
DAYS_EMPLOYED                   0.00
DAYS_REGISTRATION               0.00
DAYS_ID_PUBLISH                 0.00
FLAG_MOBIL                      0.00
FLAG_EMP_PHONE                  0.00
FLAG_WORK_PHONE                 0.00
FLAG_CONT_MOBILE                0.00
FLAG_PHONE                      0.00
FLAG_EMAIL                      0.00
NAME_FAMILY_STATUS              0.00
FLAG_DOCUMENT_5                 0.00
AMT_GOODS_PRICE                 0.09
EXT_SOURCE_2                    0.21
OBS_60_CNT_SOCIAL_CIRCLE        0.33
DEF_30_CNT_SOCIAL_CIRCLE        0.33
OBS_30_CNT_SOCIAL_CIRCLE        0.33
DEF_60_CNT_SOCIAL_CIRCLE        0.33
NAME_TYPE_SUITE                 0.42
```

```
AMT_REQ_CREDIT_BUREAU_MON      13.50
AMT_REQ_CREDIT_BUREAU_WEEK     13.50
AMT_REQ_CREDIT_BUREAU_DAY      13.50
AMT_REQ_CREDIT_BUREAU_HOUR     13.50
AMT_REQ_CREDIT_BUREAU_QRT      13.50
AMT_REQ_CREDIT_BUREAU_YEAR     13.50
EXT_SOURCE_3                   19.83
OCCUPATION_TYPE                31.35
EMERGENCYSTATE_MODE            47.40
TOTALAREA_MODE                 48.27
YEARS_BEGINEXPLUATATION_MODE   48.78
YEARS_BEGINEXPLUATATION_MEDI   48.78
YEARS_BEGINEXPLUATATION_AVG    48.78
FLOORSMAX_MODE                 49.76
FLOORSMAX_MEDI                 49.76
FLOORSMAX_AVG                  49.76
```

Columns which does not have any missing values

In [61]:

```
cols_without_missingValues=Perc_Of_NA_Columns[Perc_Of_NA_Columns==0]
print("Number of columns having null value less than 15% :", len(cols_without_missing
print(cols_without_missingValues)
```

```
Number of columns having null value less than 15% : 58
SK_ID_CURR                    0.0
TARGET                        0.0
NAME_CONTRACT_TYPE            0.0
CODE_GENDER                   0.0
FLAG_OWN_CAR                  0.0
FLAG_OWN_REALTY               0.0
CNT_CHILDREN                  0.0
AMT_INCOME_TOTAL              0.0
AMT_CREDIT                    0.0
AMT_ANNUITY                   0.0
NAME_INCOME_TYPE              0.0
NAME_EDUCATION_TYPE           0.0
NAME_FAMILY_STATUS            0.0
NAME_HOUSING_TYPE             0.0
REGION_POPULATION_RELATIVE    0.0
DAYS_BIRTH                    0.0
DAYS_EMPLOYED                 0.0
DAYS_REGISTRATION             0.0
DAYS_ID_PUBLISH               0.0
FLAG_MOBIL                    0.0
FLAG_EMP_PHONE                0.0
FLAG_WORK_PHONE               0.0
FLAG_CONT_MOBILE              0.0
FLAG_PHONE                    0.0
FLAG_EMAIL                    0.0
CNT_FAM_MEMBERS               0.0
REGION_RATING_CLIENT          0.0
REGION_RATING_CLIENT_W_CITY   0.0
WEEKDAY_APPR_PROCESS_START    0.0
HOUR_APPR_PROCESS_START       0.0
REG_REGION_NOT_LIVE_REGION    0.0
REG_REGION_NOT_WORK_REGION    0.0
LIVE_REGION_NOT_WORK_REGION   0.0
REG_CITY_NOT_LIVE_CITY        0.0
REG_CITY_NOT_WORK_CITY        0.0
LIVE_CITY_NOT_WORK_CITY       0.0
ORGANIZATION_TYPE             0.0
```

```
DAYS_LAST_PHONE_CHANGE          0.0
FLAG_DOCUMENT_2                 0.0
FLAG_DOCUMENT_3                 0.0
FLAG_DOCUMENT_4                 0.0
FLAG_DOCUMENT_5                 0.0
FLAG_DOCUMENT_6                 0.0
FLAG_DOCUMENT_7                 0.0
FLAG_DOCUMENT_8                 0.0
FLAG_DOCUMENT_9                 0.0
FLAG_DOCUMENT_10                0.0
FLAG_DOCUMENT_11                0.0
FLAG_DOCUMENT_12                0.0
FLAG_DOCUMENT_13                0.0
FLAG_DOCUMENT_14                0.0
FLAG_DOCUMENT_15                0.0
FLAG_DOCUMENT_16                0.0
FLAG_DOCUMENT_17                0.0
FLAG_DOCUMENT_18                0.0
FLAG_DOCUMENT_19                0.0
FLAG_DOCUMENT_20                0.0
FLAG_DOCUMENT_21                0.0
dtype: float64
```

Columns having <15% NA values

In [63]:

```python
Perc_Of_NA_Columns=Perc_Of_NA_Columns[Perc_Of_NA_Columns>0]
Cols_with_less_than_15Perc_NA_Values = Perc_Of_NA_Columns[Perc_Of_NA_Columns<15]
print("Number of columns having null value less than 15% :", len(Cols_with_less_than_
print(Cols_with_less_than_15Perc_NA_Values)
```

```
Number of columns having null value less than 15% : 13
AMT_GOODS_PRICE                  0.09
NAME_TYPE_SUITE                  0.42
EXT_SOURCE_2                     0.21
OBS_30_CNT_SOCIAL_CIRCLE         0.33
DEF_30_CNT_SOCIAL_CIRCLE         0.33
OBS_60_CNT_SOCIAL_CIRCLE         0.33
DEF_60_CNT_SOCIAL_CIRCLE         0.33
AMT_REQ_CREDIT_BUREAU_HOUR      13.50
AMT_REQ_CREDIT_BUREAU_DAY       13.50
AMT_REQ_CREDIT_BUREAU_WEEK      13.50
AMT_REQ_CREDIT_BUREAU_MON       13.50
AMT_REQ_CREDIT_BUREAU_QRT       13.50
AMT_REQ_CREDIT_BUREAU_YEAR      13.50
dtype: float64
```

In [67]:

```python
Cols_with_less_than_15Perc_NA_Values.index
```

Out[67]:

```
Index(['AMT_GOODS_PRICE', 'NAME_TYPE_SUITE', 'EXT_SOURCE_2',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       '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'],
      dtype='object')
```

Understand the insight of missing columns having <15% null values

In [69]:

```python
df_application_current_copy[Cols_with_less_than_15Perc_NA_Values.index].describe()
```

Out[69]:

	AMT_GOODS_PRICE	EXT_SOURCE_2	OBS_30_CNT_SOCIAL_CIRCLE	DEF_30_CNT_SOCIAL_CIRCLE
count	3.072330e+05	3.068510e+05	306490.000000	306490.000000
mean	5.383962e+05	5.143927e-01	1.422245	0.143421
std	3.694465e+05	1.910602e-01	2.400989	0.446698
min	4.050000e+04	8.173617e-08	0.000000	0.000000
25%	2.385000e+05	3.924574e-01	0.000000	0.000000
50%	4.500000e+05	5.659614e-01	0.000000	0.000000
75%	6.795000e+05	6.636171e-01	2.000000	0.000000
max	4.050000e+06	8.549997e-01	348.000000	34.000000

Identify unique values in the colums having <15% null value

In [70]:

```
df_application_current_copy[Cols_with_less_than_15Perc_NA_Values.index].nunique().sor
```

Out[70]:

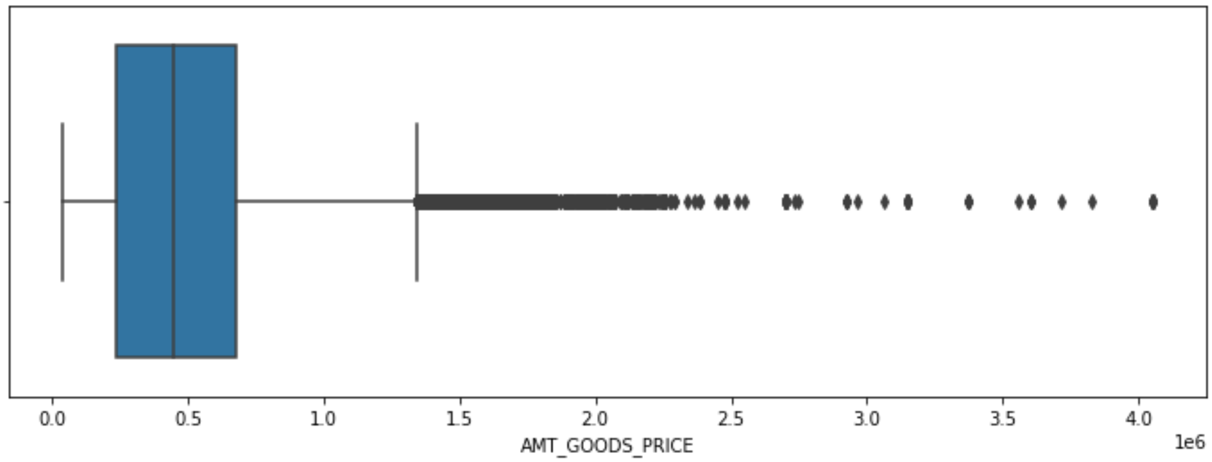
```
EXT_SOURCE_2                  119831
AMT_GOODS_PRICE                 1002
OBS_30_CNT_SOCIAL_CIRCLE          33
OBS_60_CNT_SOCIAL_CIRCLE          33
AMT_REQ_CREDIT_BUREAU_YEAR        25
AMT_REQ_CREDIT_BUREAU_MON         24
AMT_REQ_CREDIT_BUREAU_QRT         11
DEF_30_CNT_SOCIAL_CIRCLE          10
DEF_60_CNT_SOCIAL_CIRCLE           9
AMT_REQ_CREDIT_BUREAU_DAY          9
AMT_REQ_CREDIT_BUREAU_WEEK         9
NAME_TYPE_SUITE                    7
AMT_REQ_CREDIT_BUREAU_HOUR         5
dtype: int64
```

Imputing NA Values

Impute NA Values for 'AMT_GOODS_PRICE'

In [72]:

```
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['AMT_GOODS_PRICE'])
plt.show()
```



There are significant number of outliers present in the data. So we have to impute the data

In [71]:

```
df_application_current_copy['AMT_GOODS_PRICE'].describe()
```

Out[71]:

```
count    3.072330e+05
mean     5.383962e+05
std      3.694465e+05
min      4.050000e+04
25%      2.385000e+05
50%      4.500000e+05
75%      6.795000e+05
max      4.050000e+06
Name: AMT_GOODS_PRICE, dtype: float64
```

In [74]:

```
print(df_application_current_copy['AMT_GOODS_PRICE'].median())
print(df_application_current_copy['AMT_GOODS_PRICE'].mean())
```

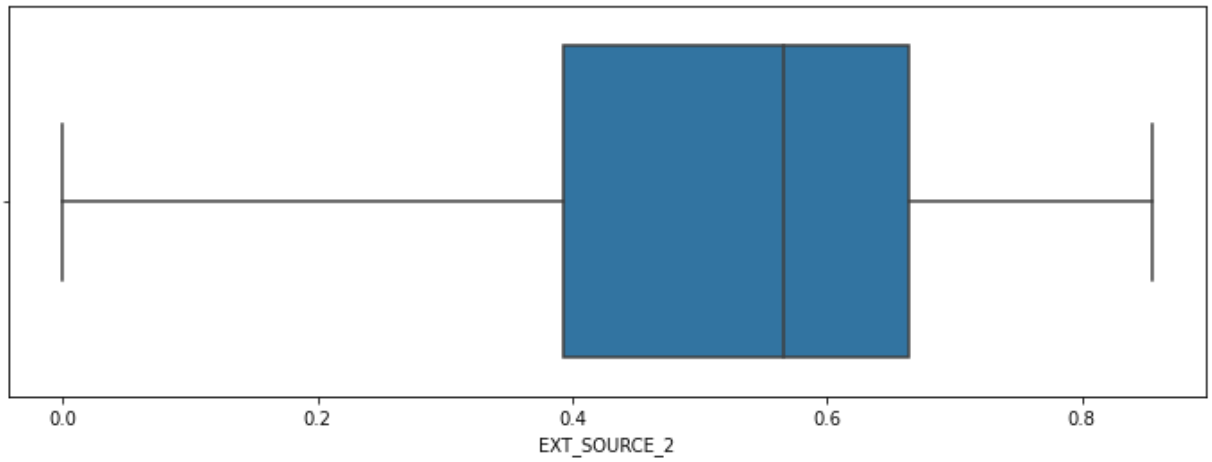
```
450000.0
538396.2074288895
```

Mean is higher than median so data should be imputed with median value:45000

Impute NA Values for 'EXT_SOURCE_2'

In [75]:

```
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['EXT_SOURCE_2'])
plt.show()
```



There is no outliers present.

In [76]:
```
df_application_current_copy['EXT_SOURCE_2'].describe()
```

Out[76]:
```
count    3.068510e+05
mean     5.143927e-01
std      1.910602e-01
min      8.173617e-08
25%      3.924574e-01
50%      5.659614e-01
75%      6.636171e-01
max      8.549997e-01
Name: EXT_SOURCE_2, dtype: float64
```

In [77]:
```
print(df_application_current_copy['EXT_SOURCE_2'].median())
print(df_application_current_copy['EXT_SOURCE_2'].mean())
```

```
0.5659614260608526
0.5143926741308463
```

There is no significant diffence observed between mean and median. However data look to be right skewed. So missing values can be imputed with median value: 0.5659614260608526

Impute NA Values for 'NAME_TYPE_SUITE' Categorical Variable

In [82]:
```
df_application_current_copy['NAME_TYPE_SUITE'].value_counts()
```

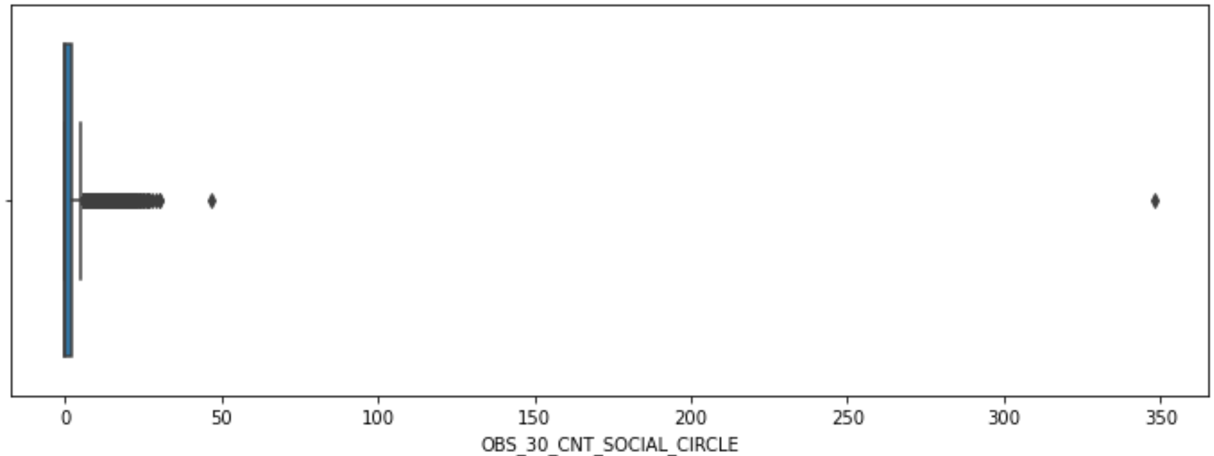
Out[82]:
```
Unaccompanied      248526
Family              40149
Spouse, partner     11370
Children             3267
Other_B              1770
Other_A               866
Group of people       271
Name: NAME_TYPE_SUITE, dtype: int64
```

For categorical vriable the value which should be imputed with maximum in frequency. So the value to be imputed is: Unaccompanied

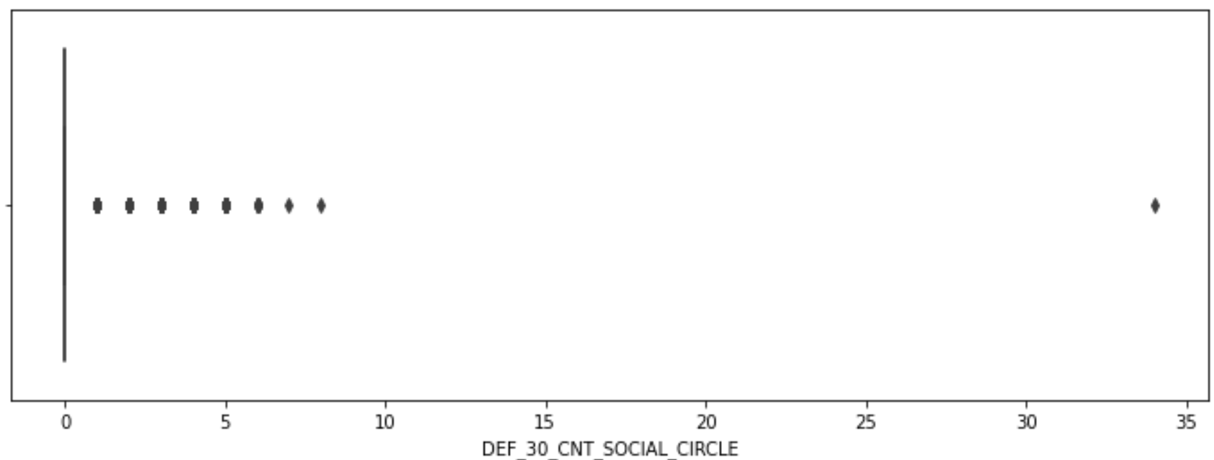
Impute NA Values for 'OBS_30_CNT_SOCIAL_CIRCLE','DEF_30_CNT_SOCIAL_CIRCLE','OBS_60_CNT_SOCIAL_CIRCLE','DE

Categorical Variables

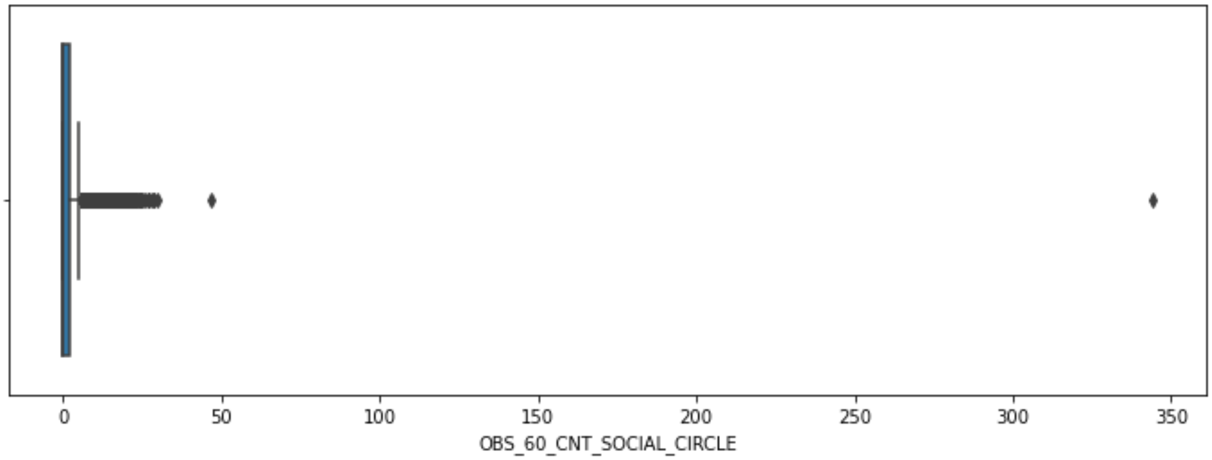
In [94]:
```
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['OBS_30_CNT_SOCIAL_CIRCLE'])
plt.show()
```



In [95]:
```
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['DEF_30_CNT_SOCIAL_CIRCLE'])
plt.show()
```

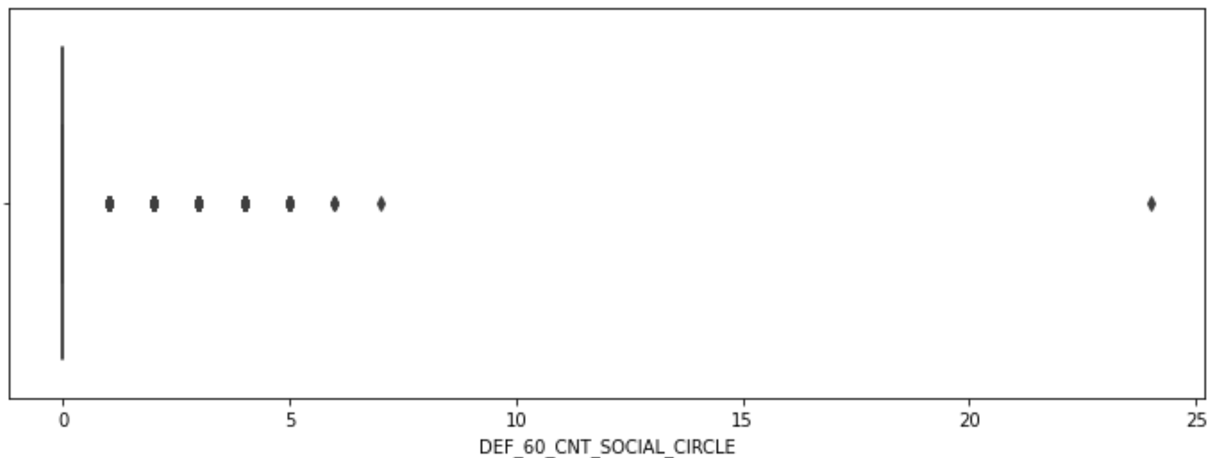


In [96]:
```
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['OBS_60_CNT_SOCIAL_CIRCLE'])
plt.show()
```



In [97]:

```
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['DEF_60_CNT_SOCIAL_CIRCLE'])
plt.show()
```



In [92]:

```
print('OBS_30_CNT_SOCIAL_CIRCLE:', df_application_current_copy['OBS_30_CNT_SOCIAL_CIF
print('DEF_30_CNT_SOCIAL_CIRCLE:', df_application_current_copy['DEF_30_CNT_SOCIAL_CIF
print('OBS_60_CNT_SOCIAL_CIRCLE:', df_application_current_copy['OBS_60_CNT_SOCIAL_CIF
print('DEF_60_CNT_SOCIAL_CIRCLE:', df_application_current_copy['DEF_60_CNT_SOCIAL_CIF
```

```
OBS_30_CNT_SOCIAL_CIRCLE: 0.0
DEF_30_CNT_SOCIAL_CIRCLE: 0.0
OBS_60_CNT_SOCIAL_CIRCLE: 0.0
DEF_60_CNT_SOCIAL_CIRCLE: 0.0
```

For categorical vriable the value which should be imputed with maximum in frequency.

So the value to be imputed are:

OBS_30_CNT_SOCIAL_CIRCLE: 0.0

DEF_30_CNT_SOCIAL_CIRCLE: 0.0

OBS_60_CNT_SOCIAL_CIRCLE: 0.0

DEF_60_CNT_SOCIAL_CIRCLE: 0.0

Since the maximum occurring values for these columns is 0 so all the missing values should be imputed with 0

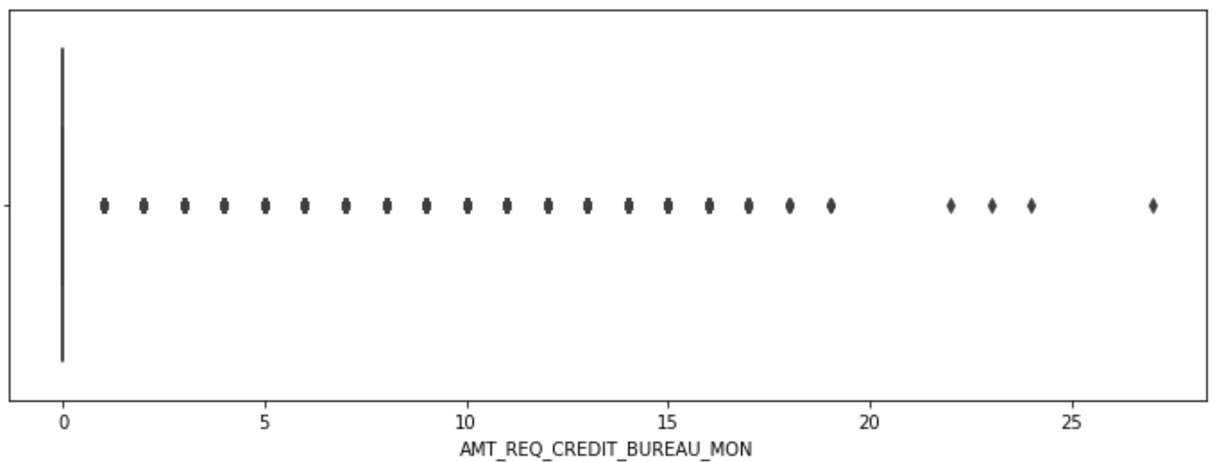
In [98]:

```
print('AMT_REQ_CREDIT_BUREAU_MON:', df_application_current_copy['AMT_REQ_CREDIT_BUREA
print('AMT_REQ_CREDIT_BUREAU_WEEK:', df_application_current_copy['AMT_REQ_CREDIT_BURE
print('AMT_REQ_CREDIT_BUREAU_DAY:', df_application_current_copy['AMT_REQ_CREDIT_BUREA
print('AMT_REQ_CREDIT_BUREAU_HOUR:', df_application_current_copy['AMT_REQ_CREDIT_BURE
print('AMT_REQ_CREDIT_BUREAU_QRT:', df_application_current_copy['AMT_REQ_CREDIT_BUREA
print('AMT_REQ_CREDIT_BUREAU_YEAR:', df_application_current_copy['AMT_REQ_CREDIT_BURE
```

```
AMT_REQ_CREDIT_BUREAU_MON: 0.0
AMT_REQ_CREDIT_BUREAU_WEEK: 0.0
AMT_REQ_CREDIT_BUREAU_DAY: 0.0
AMT_REQ_CREDIT_BUREAU_HOUR: 0.0
AMT_REQ_CREDIT_BUREAU_QRT: 0.0
AMT_REQ_CREDIT_BUREAU_YEAR: 0.0
```

In [101…

```
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_MON'])
plt.show()
```



In [104…

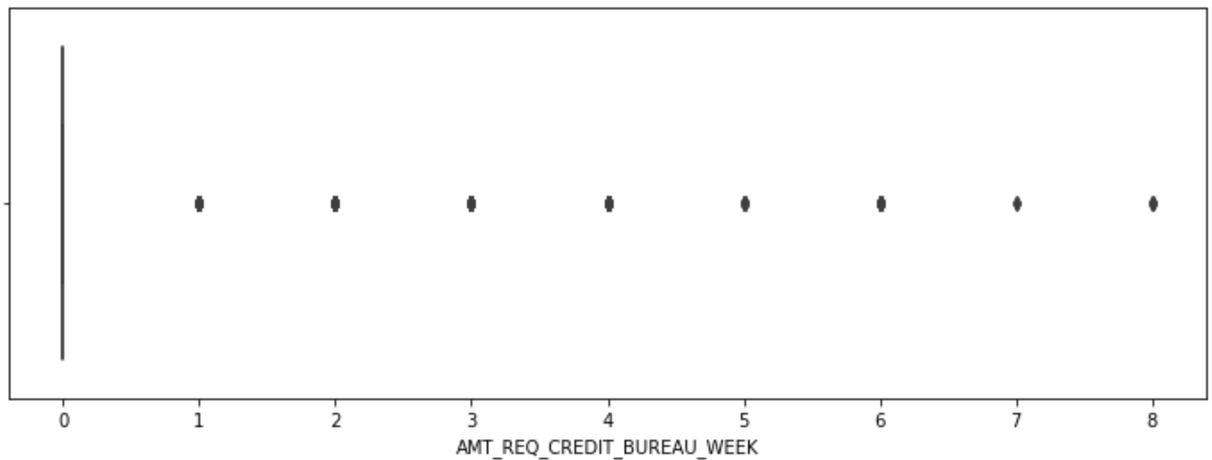
```
print(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_MON'].value_counts())
print(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_MON'].describe())
```

```
0.0     222233
1.0      33147
2.0       5386
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
27.0         1
```

```
22.0          1
23.0          1
24.0          1
Name: AMT_REQ_CREDIT_BUREAU_MON, dtype: int64
count    265992.000000
mean          0.267395
std           0.916002
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max          27.000000
Name: AMT_REQ_CREDIT_BUREAU_MON, dtype: float64
```

In [102…

```python
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_WEEK'])
plt.show()
```



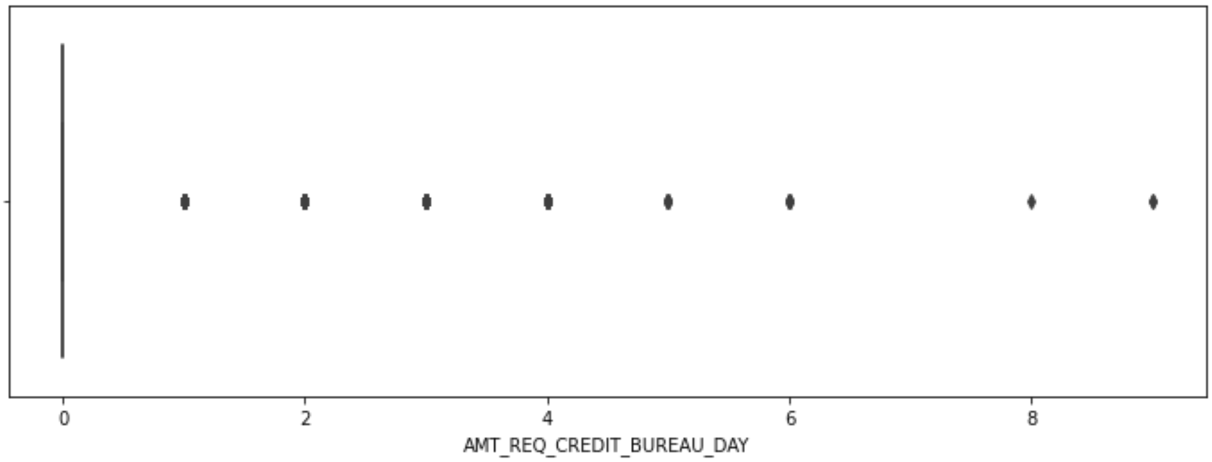
In [105…

```python
print(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_WEEK'].value_counts())
print(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_WEEK'].describe())
```

```
0.0    257456
1.0      8208
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
count    265992.000000
mean          0.034362
std           0.204685
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           8.000000
Name: AMT_REQ_CREDIT_BUREAU_WEEK, dtype: float64
```

In [106…

```
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_DAY'])
plt.show()
```



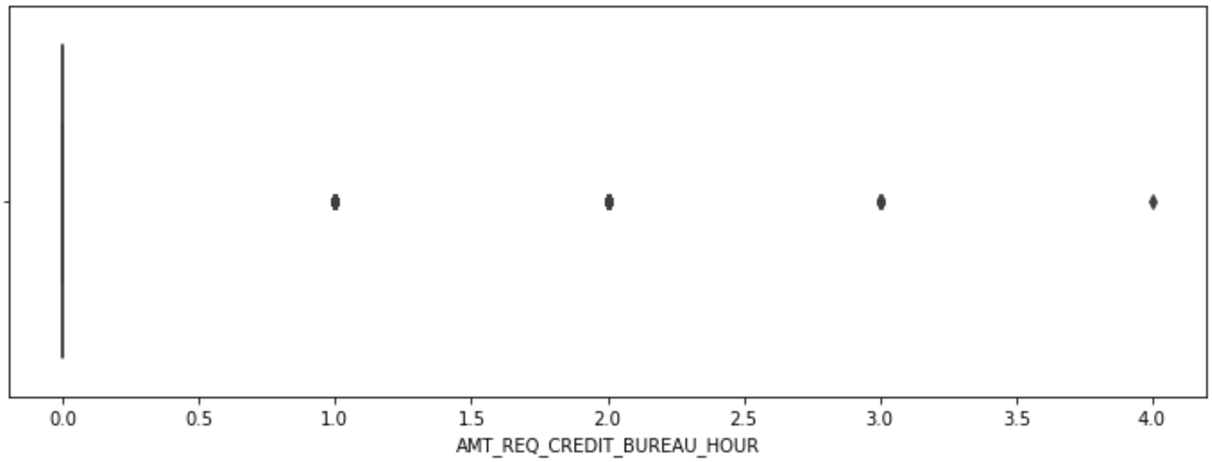
In [107…

```
print(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_DAY'].value_counts())
print(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_DAY'].describe())
```

```
0.0    264503
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
count    265992.000000
mean          0.007000
std           0.110757
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           9.000000
Name: AMT_REQ_CREDIT_BUREAU_DAY, dtype: float64
```

In [108…

```
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_HOUR'])
plt.show()
```



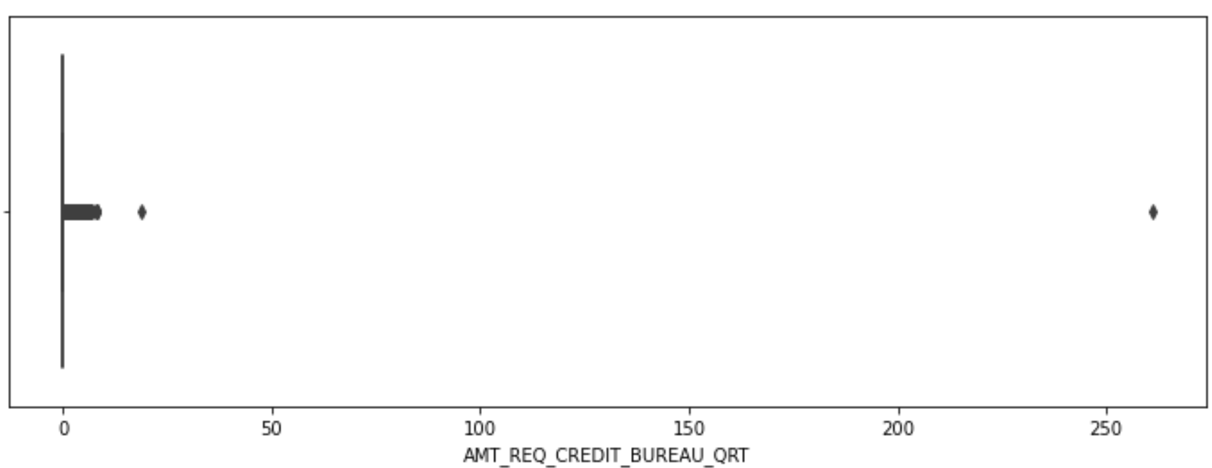
In [109…

```
print(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_HOUR'].value_counts())
print(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_HOUR'].describe())
```

```
0.0    264366
1.0      1560
2.0        56
3.0         9
4.0         1
Name: AMT_REQ_CREDIT_BUREAU_HOUR, dtype: int64
count    265992.000000
mean          0.006402
std           0.083849
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           4.000000
Name: AMT_REQ_CREDIT_BUREAU_HOUR, dtype: float64
```

In [110…

```
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_QRT'])
plt.show()
```



In [111…

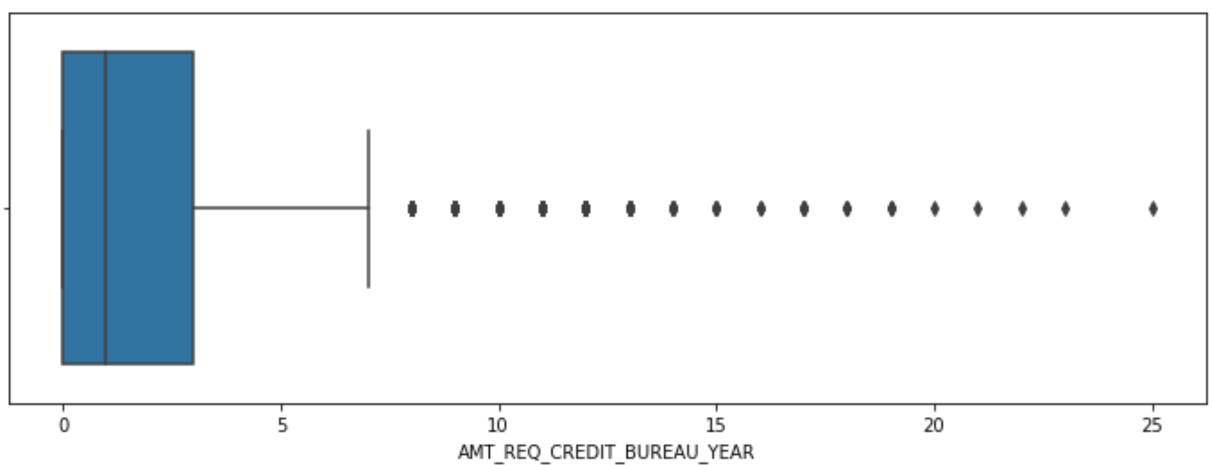
```
print(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_QRT'].value_counts())
print(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_QRT'].describe())
```

```
0.0      215417
1.0       33862
2.0       14412
3.0        1717
4.0         476
5.0          64
6.0          28
8.0           7
7.0           7
19.0          1
261.0         1
Name: AMT_REQ_CREDIT_BUREAU_QRT, dtype: int64
count    265992.000000
mean          0.265474
std           0.794056
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max         261.000000
Name: AMT_REQ_CREDIT_BUREAU_QRT, dtype: float64
```

For column AMT_REQ_CREDIT_BUREAU_MON, AMT_REQ_CREDIT_BUREAU_WEEK, AMT_REQ_CREDIT_BUREAU_DAY, AMT_REQ_CREDIT_BUREAU_HOUR, AMT_REQ_CREDIT_BUREAU_QRT we have two approaches either exclude missing values or impute the column with value 0 which is present in maximum rows. Hence, the recommended imputation technique is replacing null by the mode which is 0

In [114…

```
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_YEAR'])
plt.show()
```



In [113…

```
print(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_YEAR'].value_counts())
print(df_application_current_copy['AMT_REQ_CREDIT_BUREAU_YEAR'].describe())
```

```
0.0     71801
1.0     63405
2.0     50192
3.0     33628
4.0     20714
5.0     12052
6.0      6967
```

```
7.0      3869
8.0      2127
9.0      1096
11.0       31
12.0       30
10.0       22
13.0       19
14.0       10
17.0        7
15.0        6
18.0        4
19.0        4
16.0        3
25.0        1
21.0        1
22.0        1
20.0        1
23.0        1
Name: AMT_REQ_CREDIT_BUREAU_YEAR, dtype: int64
count    265992.000000
mean          1.899974
std           1.869295
min           0.000000
25%           0.000000
50%           1.000000
75%           3.000000
max          25.000000
Name: AMT_REQ_CREDIT_BUREAU_YEAR, dtype: float64
```

For the column AMT_REQ_CREDIT_BUREAU_YEAR :Values - 0,1,2,3,4,5 are present in a significant number. Hence, imputing null values can significantly change column statistics and hence, the best approach would be to remove these rows

In [115…

```
df_application_current_copy.shape
```

Out[115…

```
(307511, 81)
```

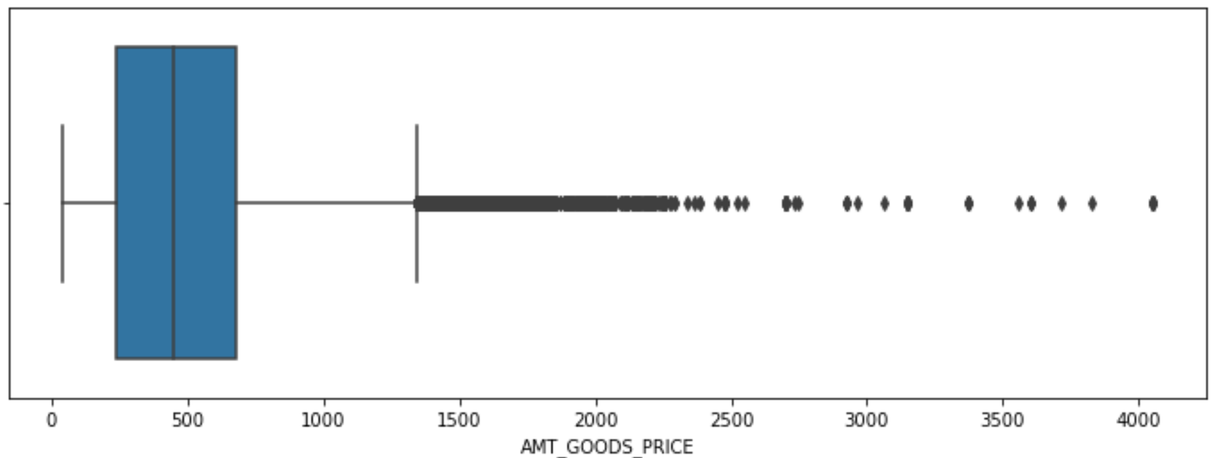
Identifying Outliers for Numerical columns

For the outlier analysis of numerical columns, we will focus on
AMT_GOODS_PRICE
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
FLOORSMAX_AVG

Outlier analysis for AMT_GOODS_PRICE

In [126…

```
#Dividing the amount by 1000 for ease of calculation
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['AMT_GOODS_PRICE']/1000.0)
plt.show()
# checking column statistics
print((df_application_current_copy['AMT_GOODS_PRICE']/1000.0).describe())

# Maximum value for boxplot
IQR_AMT_GOODS_PRICE = (df_application_current_copy['AMT_GOODS_PRICE']/1000).quantile(
Upper_limit_IQR_AMT_GOODS_PRICE = (df_application_current_copy['AMT_GOODS_PRICE']/100
print('Upper Limit',Upper_limit_IQR_AMT_GOODS_PRICE)

# percentage of outliers in AMT_GOODS_PRICE
print('Outlier %',round(100.0 * len(df_application_current_copy[(df_application_curre
```

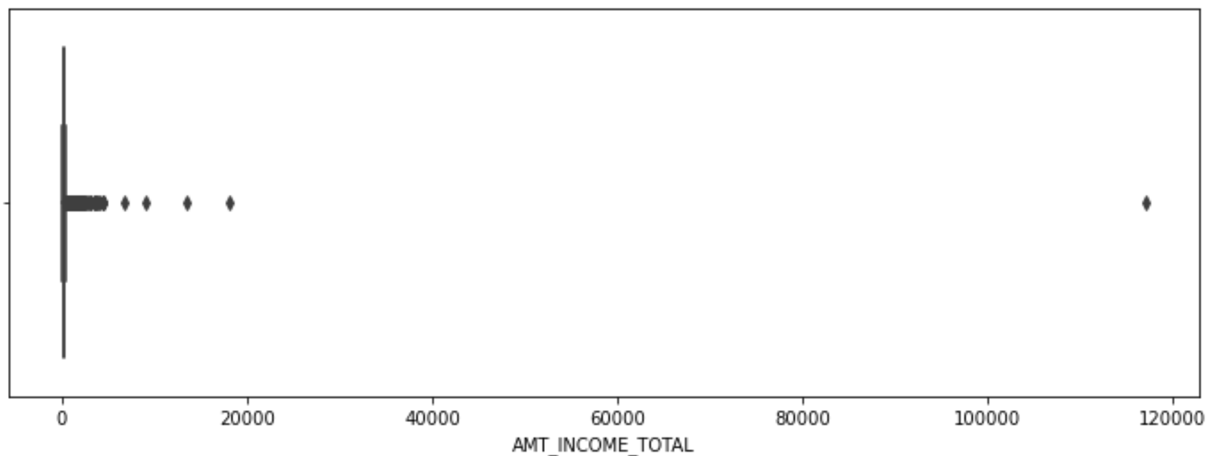



```
count    307233.000000
mean        538.396207
std         369.446461
min          40.500000
25%         238.500000
50%         450.000000
75%         679.500000
max        4050.000000
Name: AMT_GOODS_PRICE, dtype: float64
Upper Limit 1341.0
Outlier % 4.79
```

Outlier analysis for AMT_INCOME_TOTAL

In [127…

```
#Dividing the amount by 1000 for ease of calculation
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['AMT_INCOME_TOTAL']/1000.0)
plt.show()
# checking column statistics
print((df_application_current_copy['AMT_INCOME_TOTAL']/1000.0).describe())

# Maximum value for boxplot
IQR_AMT_INCOME_TOTAL = (df_application_current_copy['AMT_INCOME_TOTAL']/1000).quantil
Upper_limit_IQR_AMT_INCOME_TOTAL = (df_application_current_copy['AMT_INCOME_TOTAL']/1
print('Upper Limit',Upper_limit_IQR_AMT_INCOME_TOTAL)

# percentage of outliers in AMT_INCOME_TOTAL
print('Outlier %',round(100.0 * len(df_application_current_copy[(df_application_curre
```

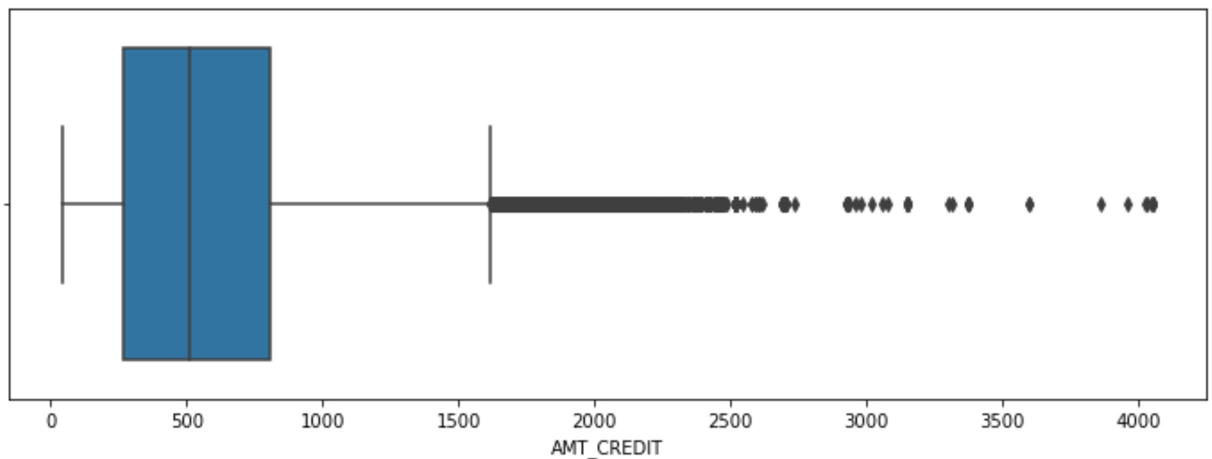



```
count    307511.000000
mean        168.797919
std         237.123146
min          25.650000
25%         112.500000
50%         147.150000
75%         202.500000
max      117000.000000
Name: AMT_INCOME_TOTAL, dtype: float64
Upper Limit 337.5
Outlier % 4.56
```

Outlier analysis for AMT_CREDIT

In [128…

```
#Dividing the amount by 1000 for ease of calculation
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['AMT_CREDIT']/1000.0)
plt.show()
# checking column statistics
print((df_application_current_copy['AMT_CREDIT']/1000.0).describe())

# Maximum value for boxplot
IQR_AMT_CREDIT = (df_application_current_copy['AMT_CREDIT']/1000).quantile(0.75) - (d
Upper_limit_IQR_AMT_CREDIT = (df_application_current_copy['AMT_CREDIT']/1000).quantil
print('Upper Limit',Upper_limit_IQR_AMT_CREDIT)

# percentage of outliers in AMT_CREDIT
print('Outlier %',round(100.0 * len(df_application_current_copy[(df_application_curre
```

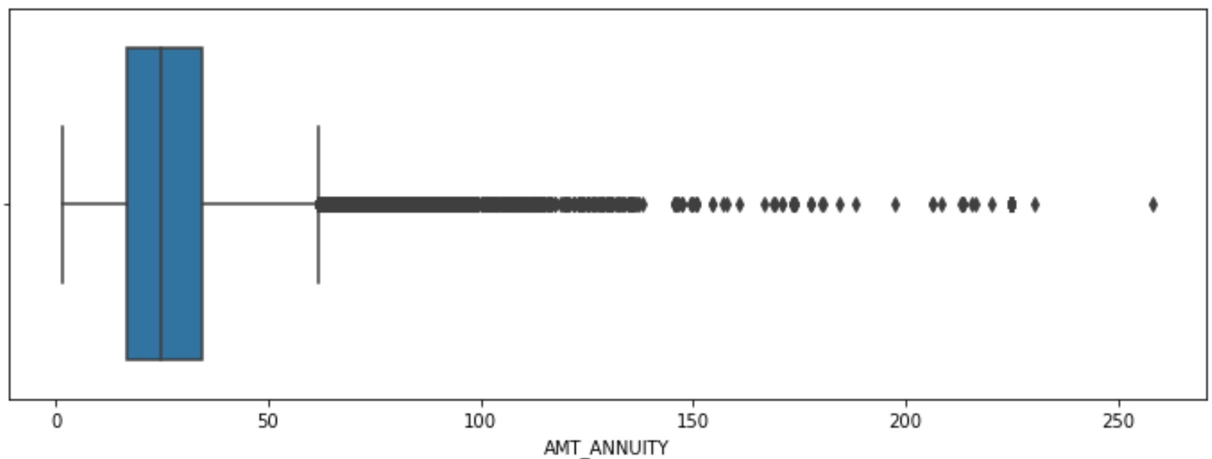



```
count    307511.000000
mean        599.026000
std         402.490777
min          45.000000
25%         270.000000
50%         513.531000
75%         808.650000
max        4050.000000
Name: AMT_CREDIT, dtype: float64
Upper Limit 1616.625
Outlier % 2.13
```

Outlier analysis for AMT_ANNUITY

In [129…

```
#Dividing the amount by 1000 for ease of calculation
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['AMT_ANNUITY']/1000.0)
plt.show()
# checking column statistics
print((df_application_current_copy['AMT_ANNUITY']/1000.0).describe())

# Maximum value for boxplot
IQR_AMT_ANNUITY = (df_application_current_copy['AMT_ANNUITY']/1000).quantile(0.75) -
Upper_limit_IQR_AMT_ANNUITY = (df_application_current_copy['AMT_ANNUITY']/1000).quant
print('Upper Limit',Upper_limit_IQR_AMT_ANNUITY)

# percentage of outliers in AMT_ANNUITY
print('Outlier %',round(100.0 * len(df_application_current_copy[(df_application_curre
```

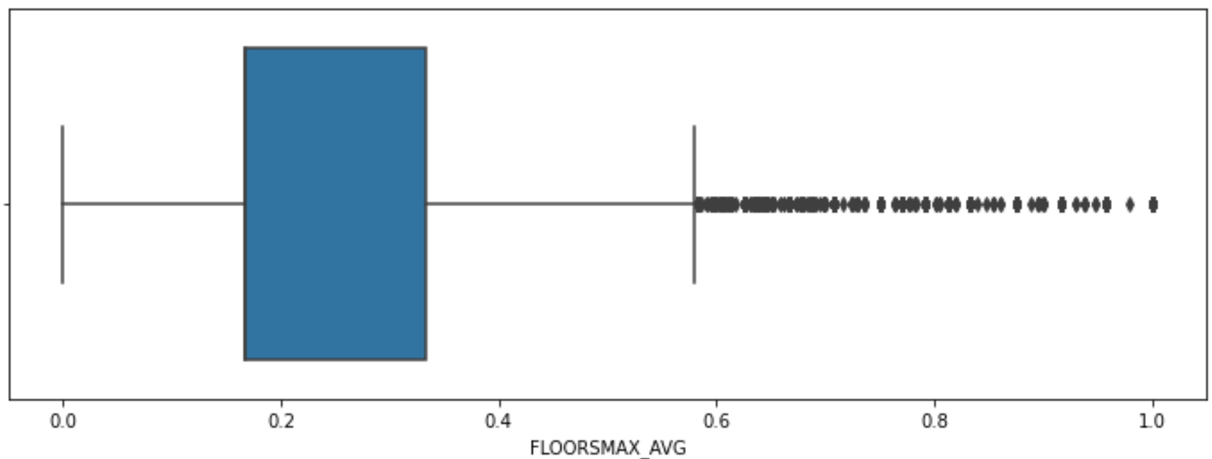



```
count    307499.000000
mean         27.108574
std          14.493737
min           1.615500
25%          16.524000
50%          24.903000
75%          34.596000
max         258.025500
Name: AMT_ANNUITY, dtype: float64
Upper Limit 61.70399999999999
Outlier % 2.44
```

Outlier analysis for FLOORSMAX_AVG

In [130…

```
#Dividing the amount by 1000 for ease of calculation
plt.figure(figsize=(12,4))
sns.boxplot(df_application_current_copy['FLOORSMAX_AVG'])
plt.show()
# checking column statistics
print((df_application_current_copy['FLOORSMAX_AVG']).describe())

# Maximum value for boxplot
IQR_FLOORSMAX_AVG = (df_application_current_copy['FLOORSMAX_AVG']).quantile(0.75) - (
Upper_limit_IQR_FLOORSMAX_AVG = (df_application_current_copy['FLOORSMAX_AVG']).quant:
print('Upper Limit',Upper_limit_IQR_FLOORSMAX_AVG)

# percentage of outliers in FLOORSMAX_AVG
print('Outlier %',round(100.0 * len(df_application_current_copy[(df_application_curre
```




```
count    154491.000000
mean          0.226282
std           0.144641
min           0.000000
25%           0.166700
50%           0.166700
75%           0.333300
max           1.000000
Name: FLOORSMAX_AVG, dtype: float64
Upper Limit 0.5831999999999999
Outlier % 1.7
```

Binning of AGE, AMT_INCOME_TOTAL columns

'DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH','DAYS_LAST_PHONE_CHA columns are having -ve value, which needs to converted to +ve value.

In [134…

```
df_application_current_copy['DAYS_BIRTH'] = df_application_current_copy['DAYS_BIRTH']
df_application_current_copy['DAYS_EMPLOYED'] = df_application_current_copy['DAYS_EMPI
df_application_current_copy['DAYS_REGISTRATION'] = df_application_current_copy['DAYS_
df_application_current_copy['DAYS_ID_PUBLISH'] = df_application_current_copy['DAYS_II
df_application_current_copy['DAYS_LAST_PHONE_CHANGE'] = df_application_current_copy[
```

All the dates are in days so dividing it with 365 to convert to years

In [135…

```
df_application_current_copy['AGE'] = abs(df_application_current_copy['DAYS_BIRTH']//3
df_application_current_copy['YEARS_EMPLOYED'] = abs(df_application_current_copy['DAYS
df_application_current_copy['YEARS_REGISTRATION'] = abs(df_application_current_copy[
df_application_current_copy['YEARS_ID_PUBLISH'] = abs(df_application_current_copy['DA
df_application_current_copy['YEARS_LAST_PHONE_CHANGE'] = abs(df_application_current_c
```

In [136…

```
df_application_current_copy.head(5)
```

Out[136…

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REAL
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	

Now 'DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH','DAYS_LAST_PHONE_CHA columns are no more required so dropping these columns

In [143…

```
drop_date_cols=['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH','D
df_application_current_copy=df_application_current_copy.drop(columns=drop_date_cols,a
df_application_current_copy.shape
```

Out[143…

```
(307511, 81)
```

In [144…

```
df_application_current_copy.head(5)
```

Out[144…

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REAL
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	

Create bins for AGE

In [145…

```
# 1. AGE column can be binned 0-10,10-20,20-30,30-40, 40-50 and so on
df_application_current_copy['AGE_GROUP'] = pd.cut(x=df_application_current_copy.AGE,
df_application_current_copy.head()
```

Out[145…

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REAL
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	

Create bins for AMT_INCOME_TOTAL

In [146…

```
# 2. AMT_INCOME_TOTAL column can be binned 'Low','Average', 'Good', 'Best' ,'High',
df_application_current_copy['AMT_CATEGORY'] = pd.cut(x=df_application_current_copy.AM
df_application_current_copy.head()
```

Out[146…

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REAL
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	

In [155…

```
df_application_current_copy.shape
```

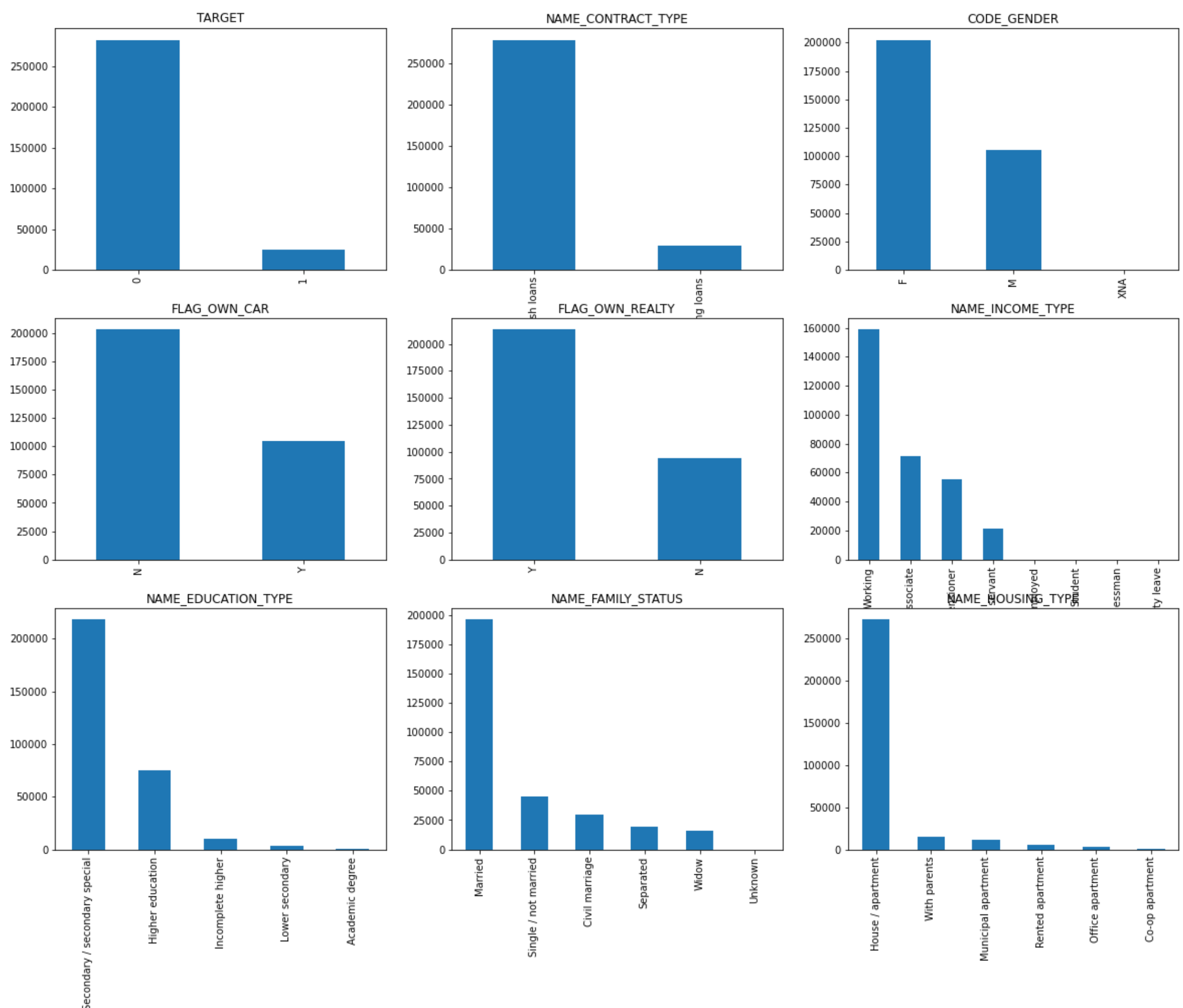
Out[155…

```
(307511, 83)
```

Checking Data Imbalence

In [156…

```
# Listing columns for checking data imbalance and plotting them
col_list = ['TARGET','NAME_CONTRACT_TYPE','CODE_GENDER','FLAG_OWN_CAR','FLAG_OWN_REA
            'NAME_EDUCATION_TYPE','NAME_FAMILY_STATUS','NAME_HOUSING_TYPE']
k=0
plt.figure(figsize=(20,15))
for col in col_list:
    k=k+1
    plt.subplot(3, 3,k)
    df_application_current_copy[col].value_counts().plot(kind='bar');
    plt.title(col)
```



We can see that there is data imbalance in below columns:-

TARGET - There are very few defaulters(1) compare to non defaulters(0)

NAME_CONTRACT_TYPE - There are very few Revolving loans than Cash loans

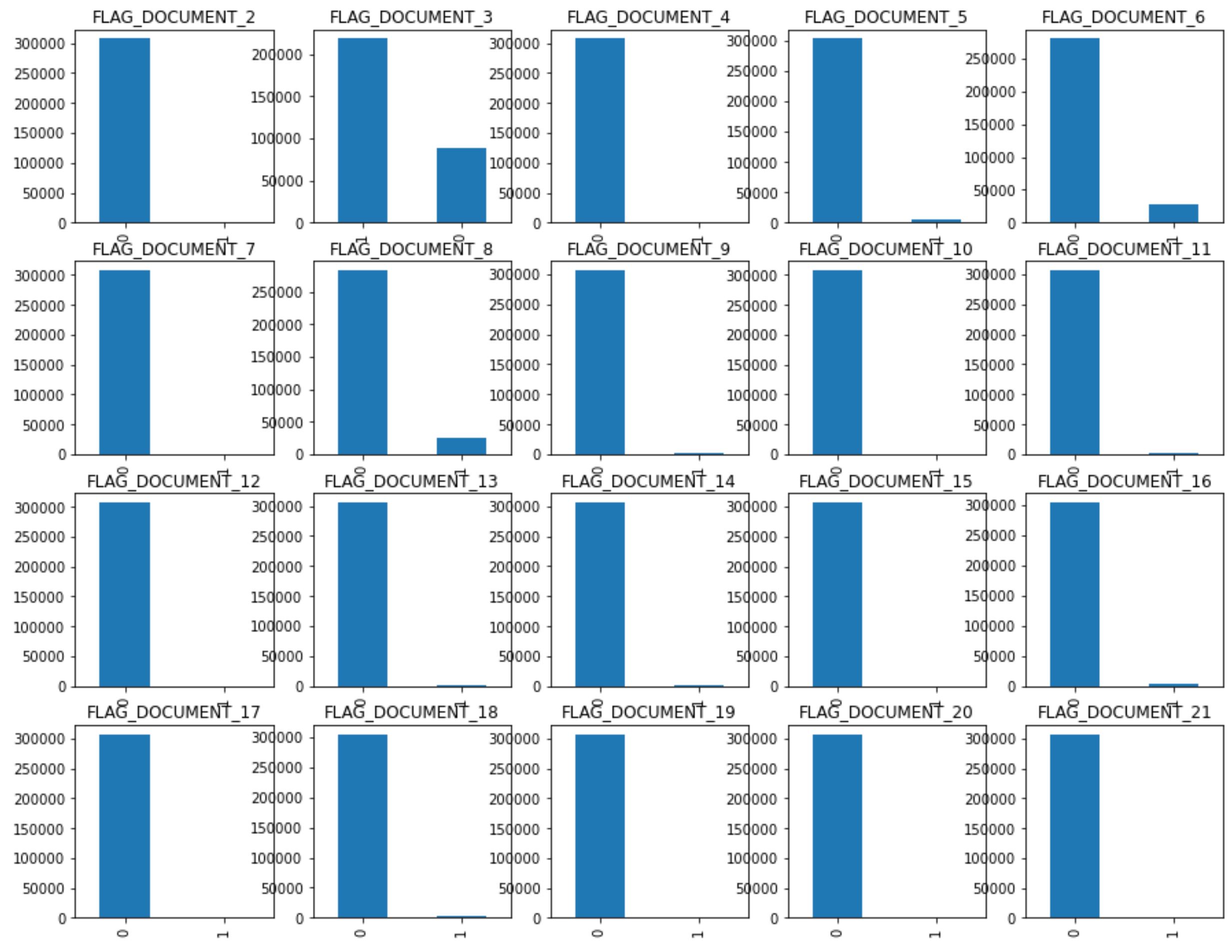
NAME_EDUCATION_TYPE - Most of the loans applied by Secondary/Secondary special educated people

NAME_FAMILY_STATUS - Most of the loans applied by Married people.

NAME_HOUSING_TYPE - Most of the application came from Home/appartment owner

In [157…

```
# Plotting all the FLAG_DOCUMENT columns to check data imbalance
k=0
plt.figure(figsize=(15,15))
for i in range(2,22) :
    k=k+1
    plt.subplot(5, 5,k)
    col_name = 'FLAG_DOCUMENT_'+str(i)
    df_application_current_copy[col_name].value_counts().plot(kind='bar');
    plt.title(col_name)
```



As we can see that except FLAG_DOCUMENT_3 all the columns have negligible count of 1s. So we are removing all the FLAG_DOCUMENT columns except FLAG_DOCUMENT_3

In [162…

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

target0_df=df_application_current_copy.loc[df_application_current_copy["TARGET"]==0]
target1_df=df_application_current_copy.loc[df_application_current_copy["TARGET"]==1]


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

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

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


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

```
Count of target0_df: 282686
Count of target1_df: 24825
Percentage of people who paid their loan are:  91.93 %
Percentage of people who did not paid their loan are:  8.07 %
```

As the percentage of Target =0 and Target =1 are different, there is an imbalance

Imbalence Ratio

In [253…

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

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

```
Imbalance Ratio: 11.39
```

For further analysis, we will remove irrelevant columns and continue analysis with a few selected columns

In [159…

```
# list of columns to be dropped
drop_columns = ['FLAG_CONT_MOBILE',
                'FLAG_MOBIL',
                'FLAG_EMP_PHONE',
                'FLAG_WORK_PHONE',
                'FLAG_PHONE',
                'FLAG_EMAIL',
                'HOUR_APPR_PROCESS_START',
                'WEEKDAY_APPR_PROCESS_START',
                'FLOORSMAX_AVG',
                'EXT_SOURCE_2',
                'EXT_SOURCE_3',
                'FLOORSMAX_AVG',
                'FLOORSMAX_MODE',
                'FLOORSMAX_MEDI',
                'TOTALAREA_MODE',
                'EMERGENCYSTATE_MODE',
                'REGION_POPULATION_RELATIVE',
                'YEARS_BEGINEXPLUATATION_AVG',
                'YEARS_BEGINEXPLUATATION_MEDI',
                'YEARS_BEGINEXPLUATATION_MODE',
                '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',
                'FLAG_DOCUMENT_2',
                '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'
               ]
```

In [ ]:

Dropping all the unwanted columns from Target1 data

In [169…

```
target1_df_2 = target1_df.drop(columns=drop_columns, axis=1)
target1_df_2.shape
```

Out[169…
```
(24825, 39)
```

Dropping all the unwanted columns from Target0 data

In [171…
```
target0_df_2 = target0_df.drop(columns=drop_columns, axis=1)
target0_df_2.shape
```

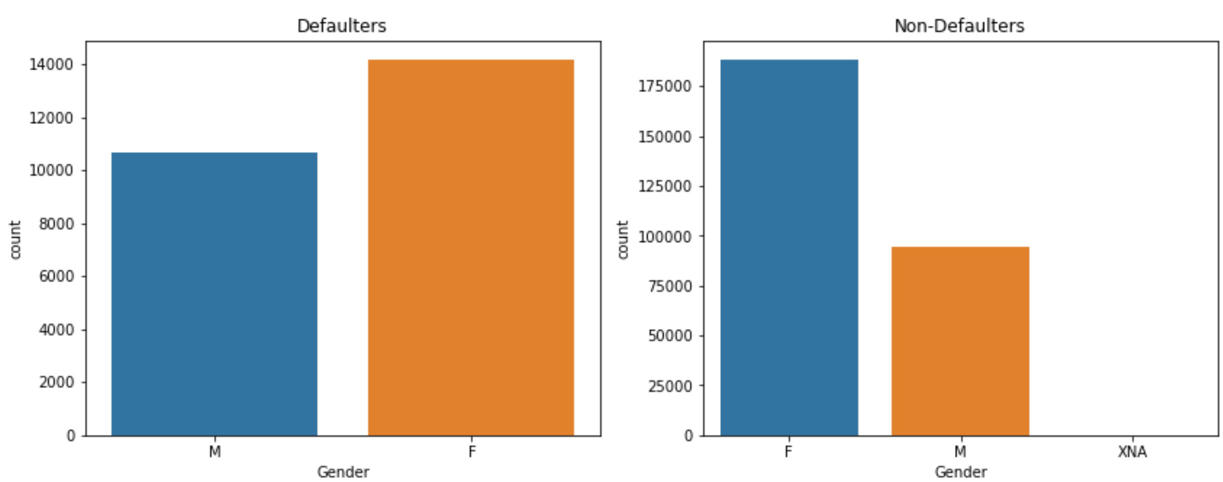
Out[171…
```
(282686, 39)
```

Analysis

Comparison of defaulters and non-defaulters on the basis of gender

In [172…
```
# Plotting two plots for delaulters and non defaulters on basis of gender
plt.figure(figsize=(14,5))

plt.subplot(1,2,1)
ax = sns.countplot(x = 'CODE_GENDER',data=target1_df_2)
plt.title('Defaulters')
ax.set(xlabel='Gender')

plt.subplot(1,2,2)
ax = sns.countplot(x = 'CODE_GENDER',data=target0_df_2)
plt.title('Non-Defaulters')
ax.set(xlabel='Gender')
```

Out[172…
```
[Text(0.5, 0, 'Gender')]
```



Insights

Defaluters - We can see that females are more in number of defaulters than male.

Non-defaluters - The same pattern continues for non-defaluters as well. The females are more in number here than male.

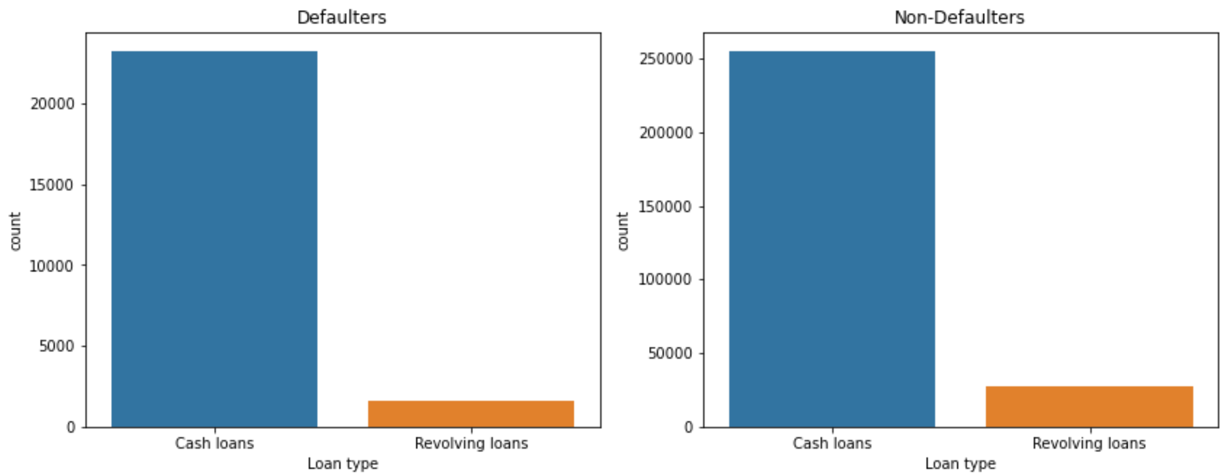
Comparison of defaulters and non-defaulters on the basis of Loan Type

In [177…

```
# Plotting two plots for delaulters and non defaulters on basis of gender
plt.figure(figsize=(14,5))

plt.subplot(1,2,1)
ax = sns.countplot(x = 'NAME_CONTRACT_TYPE',data=target1_df_2)
plt.title('Defaulters')
ax.set(xlabel='Loan type')

plt.subplot(1,2,2)
ax = sns.countplot(x = 'NAME_CONTRACT_TYPE',data=target0_df_2)
plt.title('Non-Defaulters')
ax.set(xlabel='Loan type')
```

Out[177…

```
[Text(0.5, 0, 'Loan type')]
```



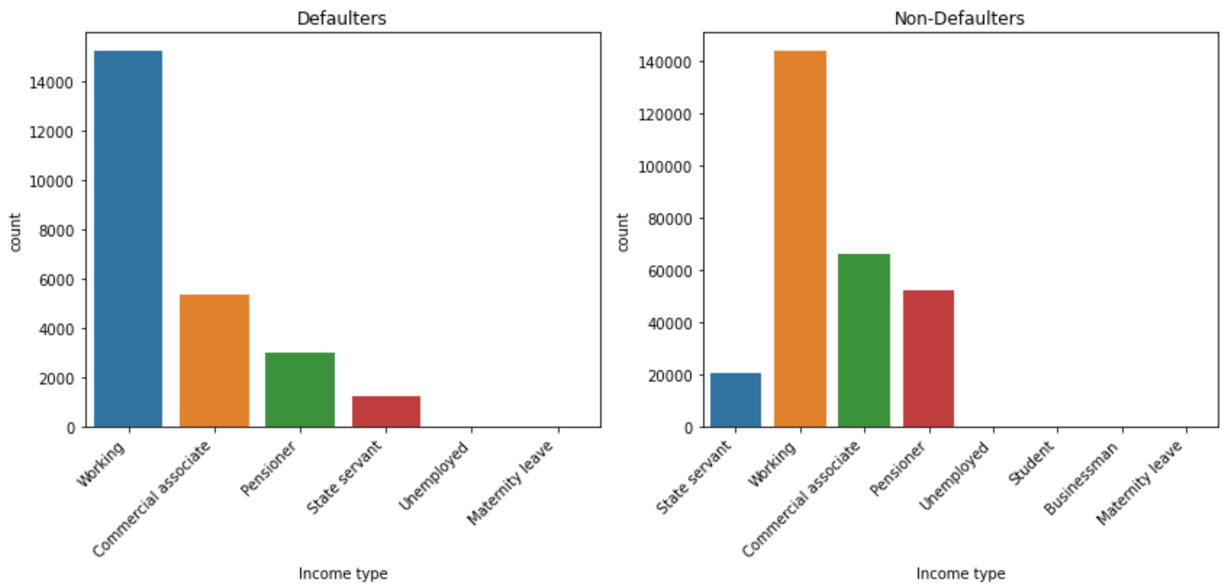
Insights

We see in both the cases that Revolving loans are very less in number compared to Cash loans.

Comparison of Defaulters and non-defaulters on the basis of Income type

In [178…

```
# Plotting two plots for delaulters and non defaulters on basis of gender
plt.figure(figsize=(14,5))

plt.subplot(1,2,1)
ax = sns.countplot(x = 'NAME_INCOME_TYPE',data=target1_df_2)
plt.title('Defaulters')
ax.set(xlabel='Income type')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r


plt.subplot(1,2,2)
ax = sns.countplot(x = 'NAME_INCOME_TYPE',data=target0_df_2)
plt.title('Non-Defaulters')
ax.set(xlabel='Income type')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r
```



Insights

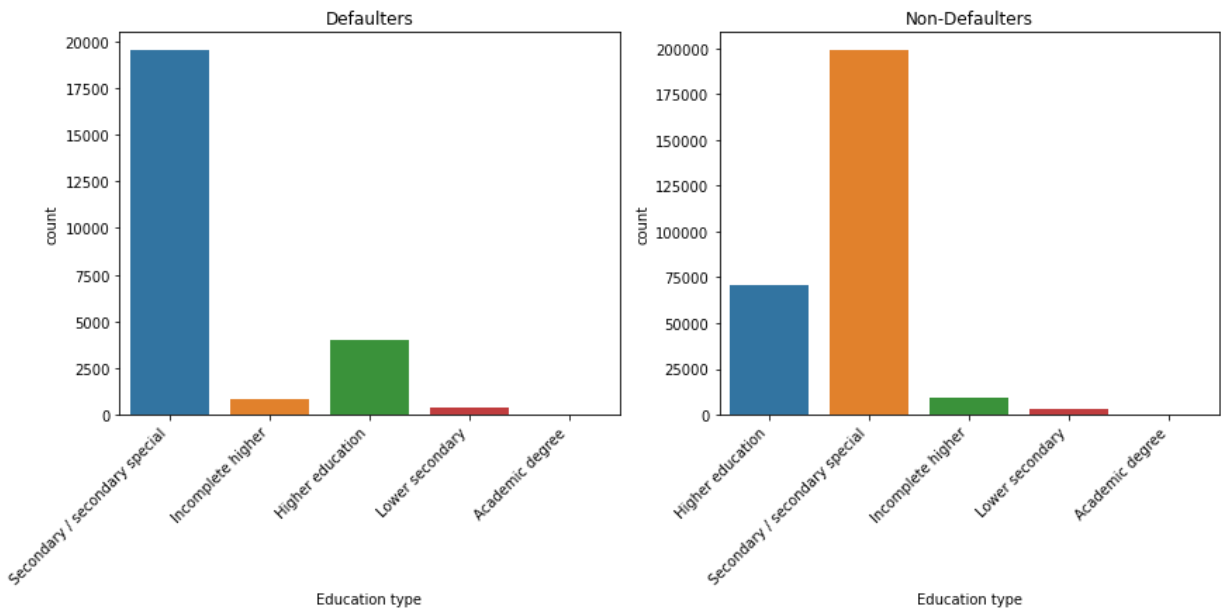
Defaulters - Working people are mostly defaulted as their numbers are high with compare to other professions.

Non-defaulters - Similarly here also working people are more in number who are not defaulted.

Comparison of Defaulters and non-defaulters on the basis of Education type

In [179…

```
# Plotting two plots for delaulters and non defaulters on basis of gender
plt.figure(figsize=(14,5))

plt.subplot(1,2,1)
ax = sns.countplot(x = 'NAME_EDUCATION_TYPE',data=target1_df_2)
plt.title('Defaulters')
ax.set(xlabel='Education type')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r


plt.subplot(1,2,2)
ax = sns.countplot(x = 'NAME_EDUCATION_TYPE',data=target0_df_2)
plt.title('Non-Defaulters')
ax.set(xlabel='Education type')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r
```



Insights

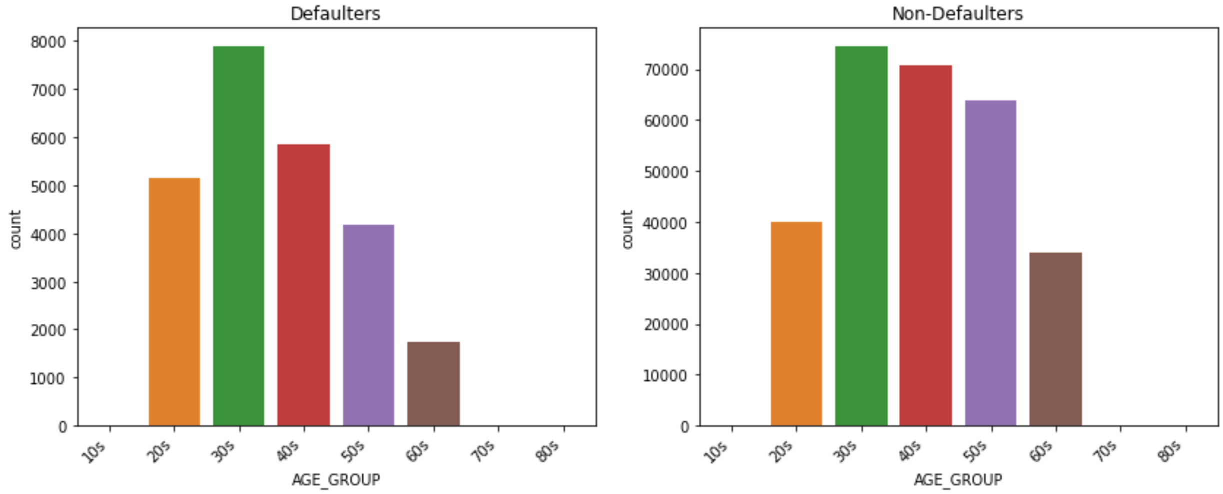
Defaulters - Education with Secondary/Secondary sepcial customers are more number in defaulters comapre with other level of eduacted poeple.

Non defaulters - Here also Secondary/Secondary sepcial are more in numbers.

Comparison of Defaulters and non-defaulters on the basis of AGE_GROUP

In [182…

```
# Plotting two plots for delaulters and non defaulters on basis of gender
plt.figure(figsize=(14,5))

plt.subplot(1,2,1)
ax = sns.countplot(x = 'AGE_GROUP',data=target1_df_2)
plt.title('Defaulters')
ax.set(xlabel='AGE_GROUP')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r


plt.subplot(1,2,2)
ax = sns.countplot(x = 'AGE_GROUP',data=target0_df_2)
plt.title('Non-Defaulters')
ax.set(xlabel='AGE_GROUP')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r
```



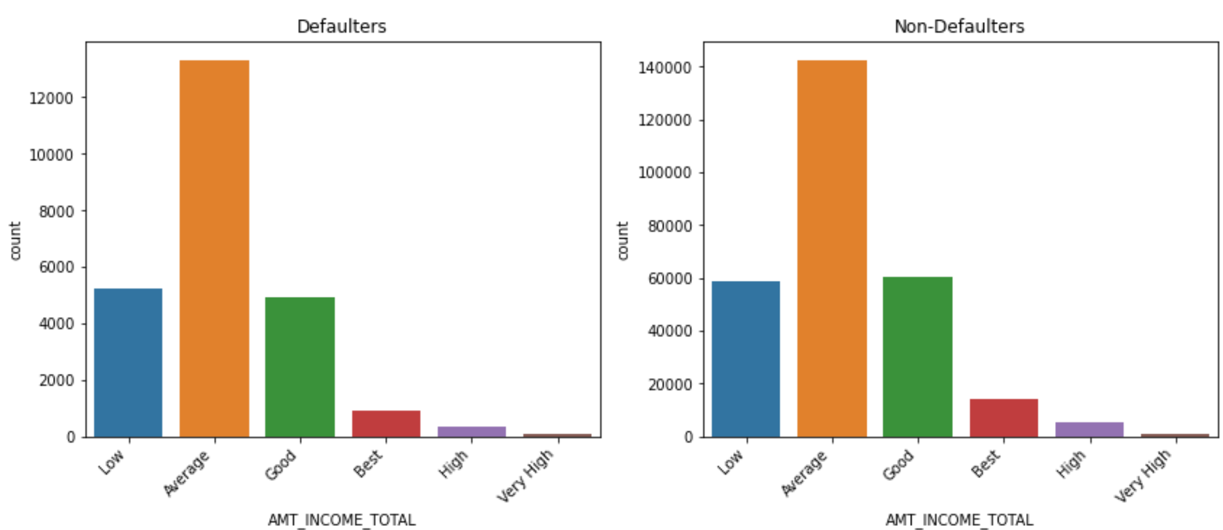
Insights

Defaulters - Customers who are in their early days in career(specially in 30's) are more number in defaulters comapre with other Age group. Older people are less likely to be defaulter
Non defaulters - Here also the same trend is followed.

Comparison of Defaulters and non-defaulters on the basis of AMT_INCOME_TOTAL

In [185…

```
# Plotting two plots for delaulters and non defaulters on basis of gender
plt.figure(figsize=(14,5))

plt.subplot(1,2,1)
ax = sns.countplot(x = 'AMT_CATEGORY',data=target1_df_2)
plt.title('Defaulters')
ax.set(xlabel='AMT_INCOME_TOTAL')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r


plt.subplot(1,2,2)
ax = sns.countplot(x = 'AMT_CATEGORY',data=target0_df_2)
plt.title('Non-Defaulters')
ax.set(xlabel='AMT_INCOME_TOTAL')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r
```



Insights

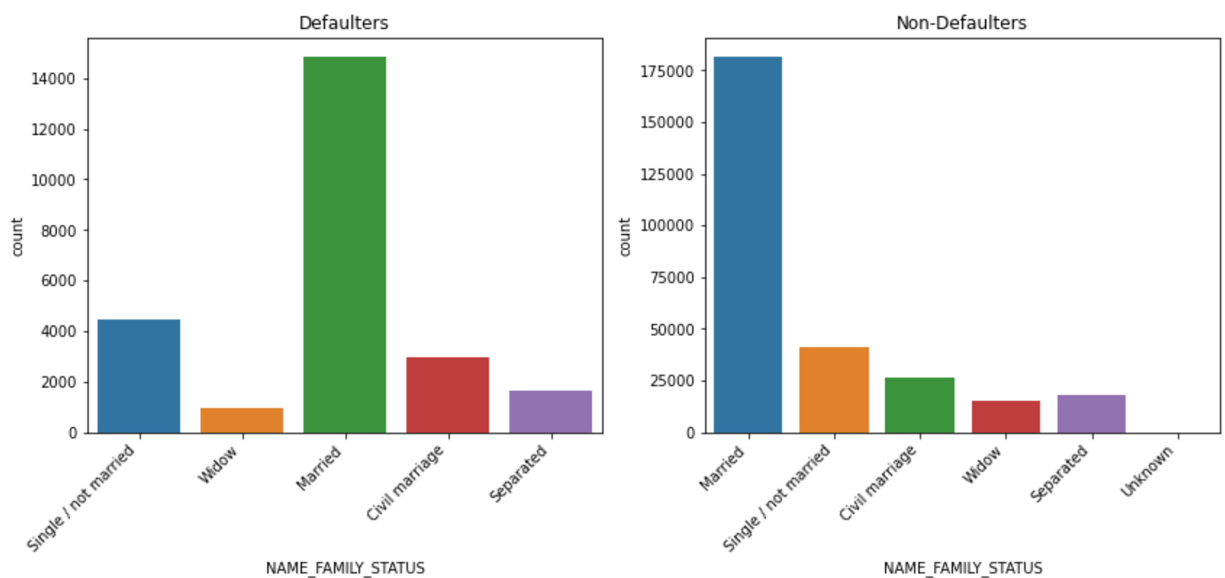
Defaulters - Customers who are average income group are most likely to be defaulter and who are in best,high and very high income group are less likely defaulter

Non defaulters - Here also the same trend is followed.

Comparison of Defaulters and non-defaulters on the basis of NAME_FAMILY_STATUS

In [186…

```
# Plotting two plots for delaulters and non defaulters on basis of gender
plt.figure(figsize=(14,5))

plt.subplot(1,2,1)
ax = sns.countplot(x = 'NAME_FAMILY_STATUS',data=target1_df_2)
plt.title('Defaulters')
ax.set(xlabel='NAME_FAMILY_STATUS')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r


plt.subplot(1,2,2)
ax = sns.countplot(x = 'NAME_FAMILY_STATUS',data=target0_df_2)
plt.title('Non-Defaulters')
ax.set(xlabel='NAME_FAMILY_STATUS')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r
```



Insights

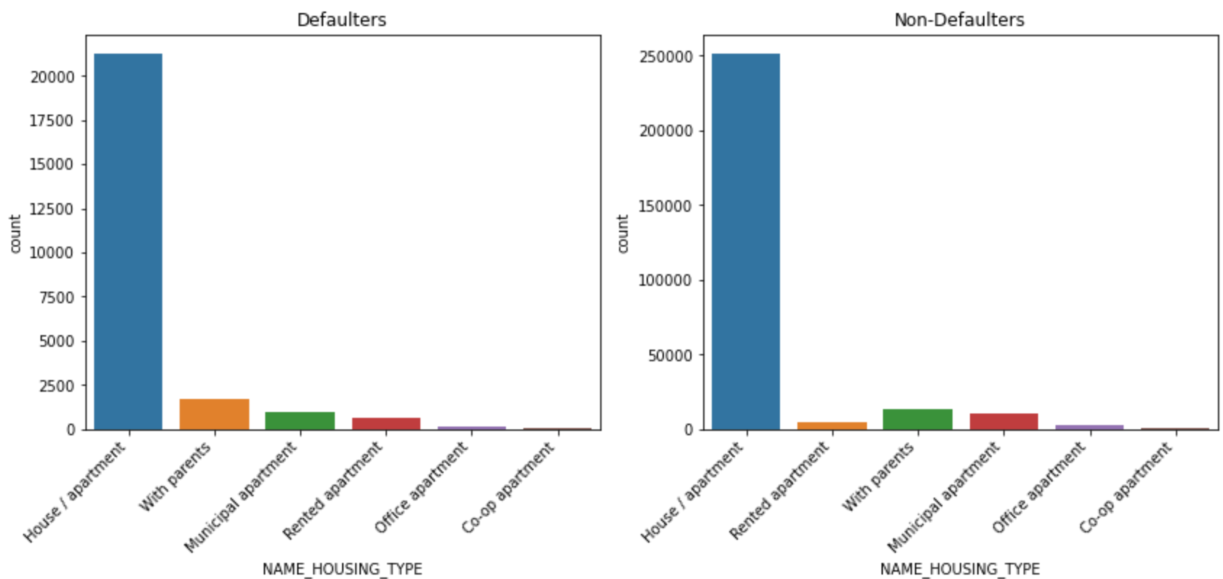
Defaulters - Customers who are married are most likely to be defaulter

Non defaulters - Here also the same trend is followed.

Comparison of Defaulters and non-defaulters on the basis of NAME_HOUSING_TYPE

In [188…

```
# Plotting two plots for delaulters and non defaulters on basis of gender
plt.figure(figsize=(14,5))

plt.subplot(1,2,1)
ax = sns.countplot(x = 'NAME_HOUSING_TYPE',data=target1_df_2)
plt.title('Defaulters')
ax.set(xlabel='NAME_HOUSING_TYPE')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r


plt.subplot(1,2,2)
ax = sns.countplot(x = 'NAME_HOUSING_TYPE',data=target0_df_2)
plt.title('Non-Defaulters')
ax.set(xlabel='NAME_HOUSING_TYPE')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r
```



Insights

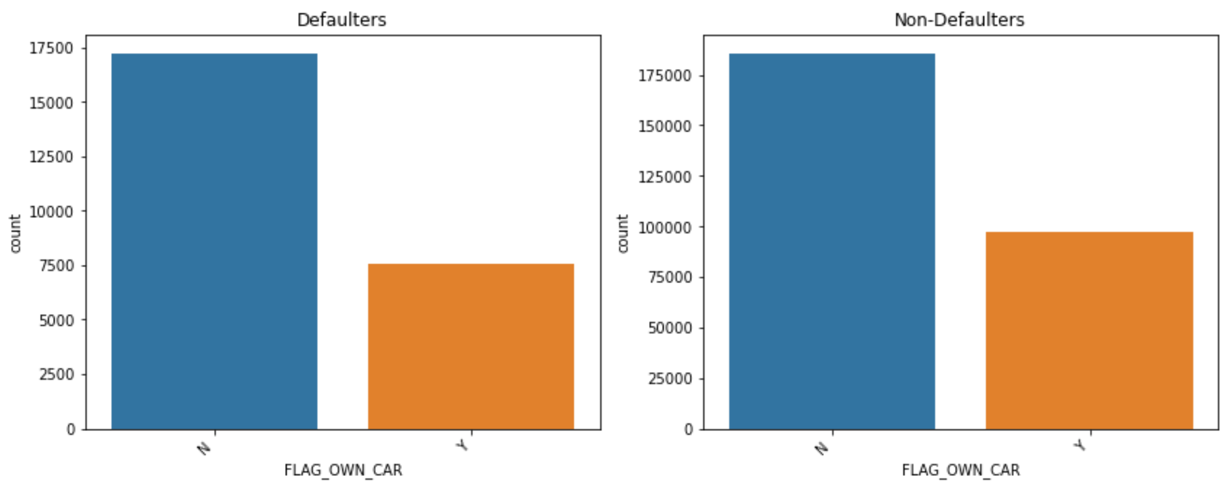
Defaulters - Customers who are owining a house/apartment are most likely to be defaulter

Non defaulters - Here also the same trend is followed.

Comparison of Defaulters and non-defaulters on the basis of FLAG_OWN_CAR

In [189…

```
# Plotting two plots for delaulters and non defaulters on basis of gender
plt.figure(figsize=(14,5))

plt.subplot(1,2,1)
ax = sns.countplot(x = 'FLAG_OWN_CAR',data=target1_df_2)
plt.title('Defaulters')
ax.set(xlabel='FLAG_OWN_CAR')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r


plt.subplot(1,2,2)
ax = sns.countplot(x = 'FLAG_OWN_CAR',data=target0_df_2)
plt.title('Non-Defaulters')
ax.set(xlabel='FLAG_OWN_CAR')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r
```



Insights

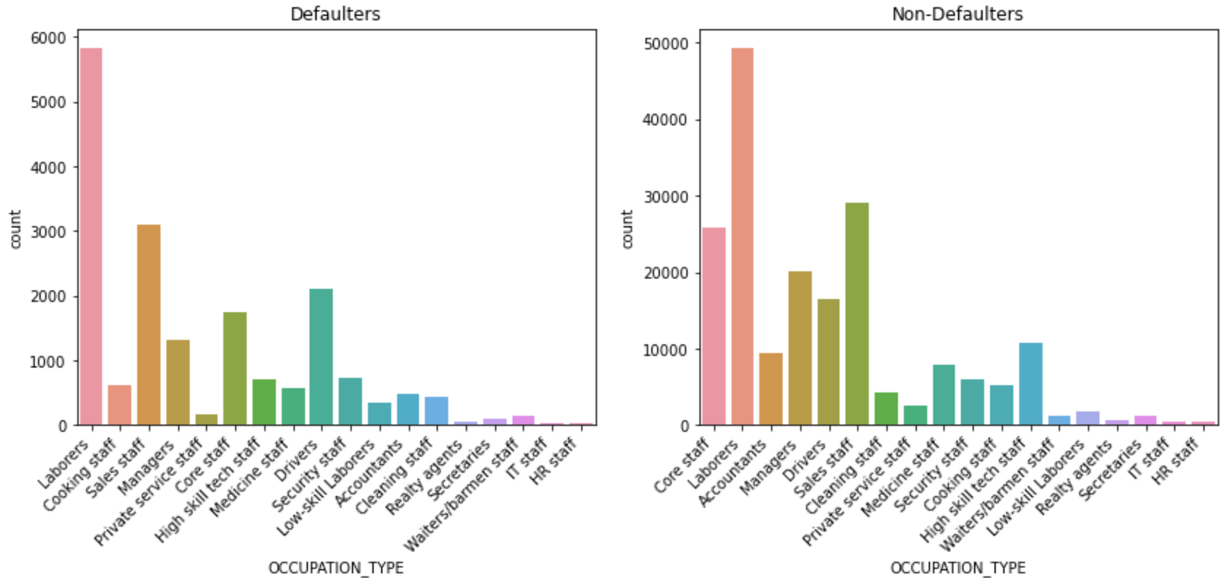
Defaulters - Customers who are not owining a car are most likely to be defaulter

Non defaulters - Here also the same trend is followed.

Comparison of Defaulters and non-defaulters on the basis of OCCUPATION_TYPE

In [190…

```
# Plotting two plots for delaulters and non defaulters on basis of gender
plt.figure(figsize=(14,5))

plt.subplot(1,2,1)
ax = sns.countplot(x = 'OCCUPATION_TYPE',data=target1_df_2)
plt.title('Defaulters')
ax.set(xlabel='OCCUPATION_TYPE')
```
data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAA0oAAA(
```
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r


plt.subplot(1,2,2)
ax = sns.countplot(x = 'OCCUPATION_TYPE',data=target0_df_2)
plt.title('Non-Defaulters')
ax.set(xlabel='OCCUPATION_TYPE')
temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment='r
```



Insights

Defaulters - Customers who are having OCCUPATION_TYPE as laborers,Sales Staff,Drivers,Core Staff are most likely to be defaulter

Non defaulters - Customers who are having OCCUPATION_TYPE as laborers,Sales Staff,Managers,Core Staff are most likely to be non-defaulter

Correlation of relevant numerical columns for defaulters and non defaulters

Selecting the correlation Columns

In [195…

```
corr_cols = ['AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','AMT_GOODS_PRICE','AGE','R
```

Corelation of defaulters

In [196…

```
df_corr_target_1 = target1_df_2[corr_cols]
df_corr_target_1.head()
```

Out[196…

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	AGE	REGION_RATING_C
0	202500.0	406597.5	24700.5	351000.0	25	
26	112500.0	979992.0	27076.5	702000.0	51	
40	202500.0	1193580.0	35028.0	855000.0	47	
42	135000.0	288873.0	16258.5	238500.0	36	
81	81000.0	252000.0	14593.5	252000.0	67	

Correlation matrix for target 1

In [198…

```
df_corr_target_1.corr()
```

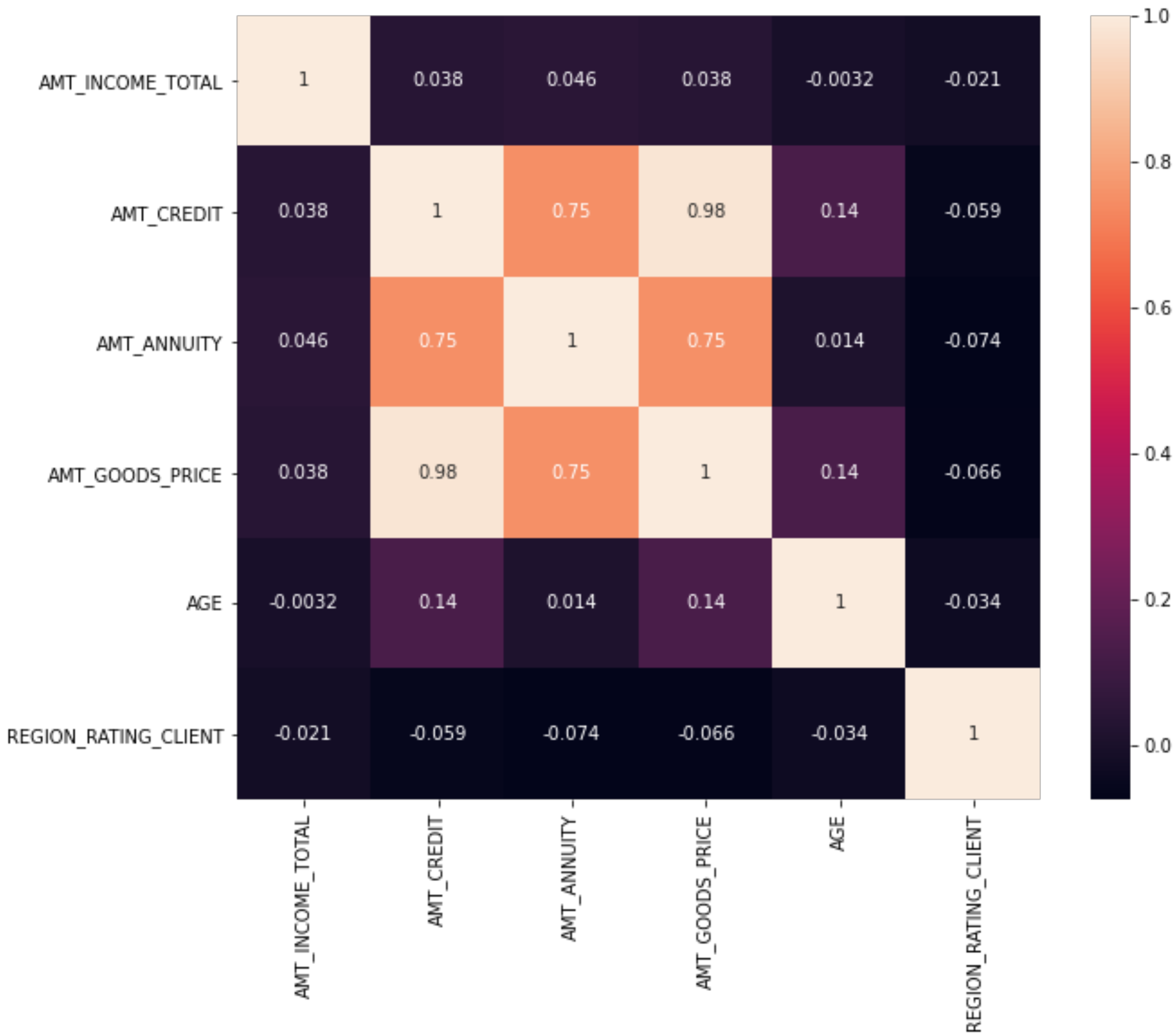
Out[198…

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	
AMT_INCOME_TOTAL	1.000000	0.038131	0.046421	0.037583	-0.(
AMT_CREDIT	0.038131	1.000000	0.752195	0.983103	0.1
AMT_ANNUITY	0.046421	0.752195	1.000000	0.752699	0.(
AMT_GOODS_PRICE	0.037583	0.983103	0.752699	1.000000	0.1
AGE	-0.003154	0.135070	0.014028	0.135603	1.(
REGION_RATING_CLIENT	-0.021486	-0.059193	-0.073784	-0.066390	-0.(

In [200…

```
plt.figure(figsize=(10,8))
sns.heatmap(df_corr_target_1.corr(),annot=True)
```

Out[200… <AxesSubplot:>



Highly correlate columns for defaulters

AMT_CREDIT and AMT_ANNUITY (0.75)

AMT_CREDIT and AMT_GOODS_PRICE (0.98)

AMT_ANNUITY and AMT_GOODS_PRICE (0.75)

Corelation of Non-defaulters

In [201…

```
df_corr_target_0 = target0_df_2[corr_cols]
df_corr_target_0.head()
```

Out[201…

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	AGE	REGION_RATING_CLI
1	270000.0	1293502.5	35698.5	1129500.0	45	
2	67500.0	135000.0	6750.0	135000.0	52	
3	135000.0	312682.5	29686.5	297000.0	52	
4	121500.0	513000.0	21865.5	513000.0	54	
5	99000.0	490495.5	27517.5	454500.0	46	

Correlation matrix for target 0

In [202…

```
df_corr_target_0.corr()
```

Out[202…

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	
AMT_INCOME_TOTAL	1.000000	0.342799	0.418953	0.349462	-0.(
AMT_CREDIT	0.342799	1.000000	0.771309	0.987250	0.(
AMT_ANNUITY	0.418953	0.771309	1.000000	0.776686	-0.(
AMT_GOODS_PRICE	0.349462	0.987250	0.776686	1.000000	0.(
AGE	-0.062494	0.047366	-0.012254	0.044552	1.(
REGION_RATING_CLIENT	-0.186573	-0.103337	-0.132128	-0.104382	-0.(

In [203…

```
plt.figure(figsize=(10,8))
sns.heatmap(df_corr_target_0.corr(),annot=True)
```

Out[203…

```
<AxesSubplot:>
```

Highly corelate columns for non defaulters

AMT_CREDIT and AMT_ANNUITY (0.77)

AMT_CREDIT and AMT_GOODS_PRICE (0.99)

AMT_ANNUITY and AMT_GOODS_PRICE (0.78)

Bivariate analysis on categorical variable

Credit amount of the loan of various categories

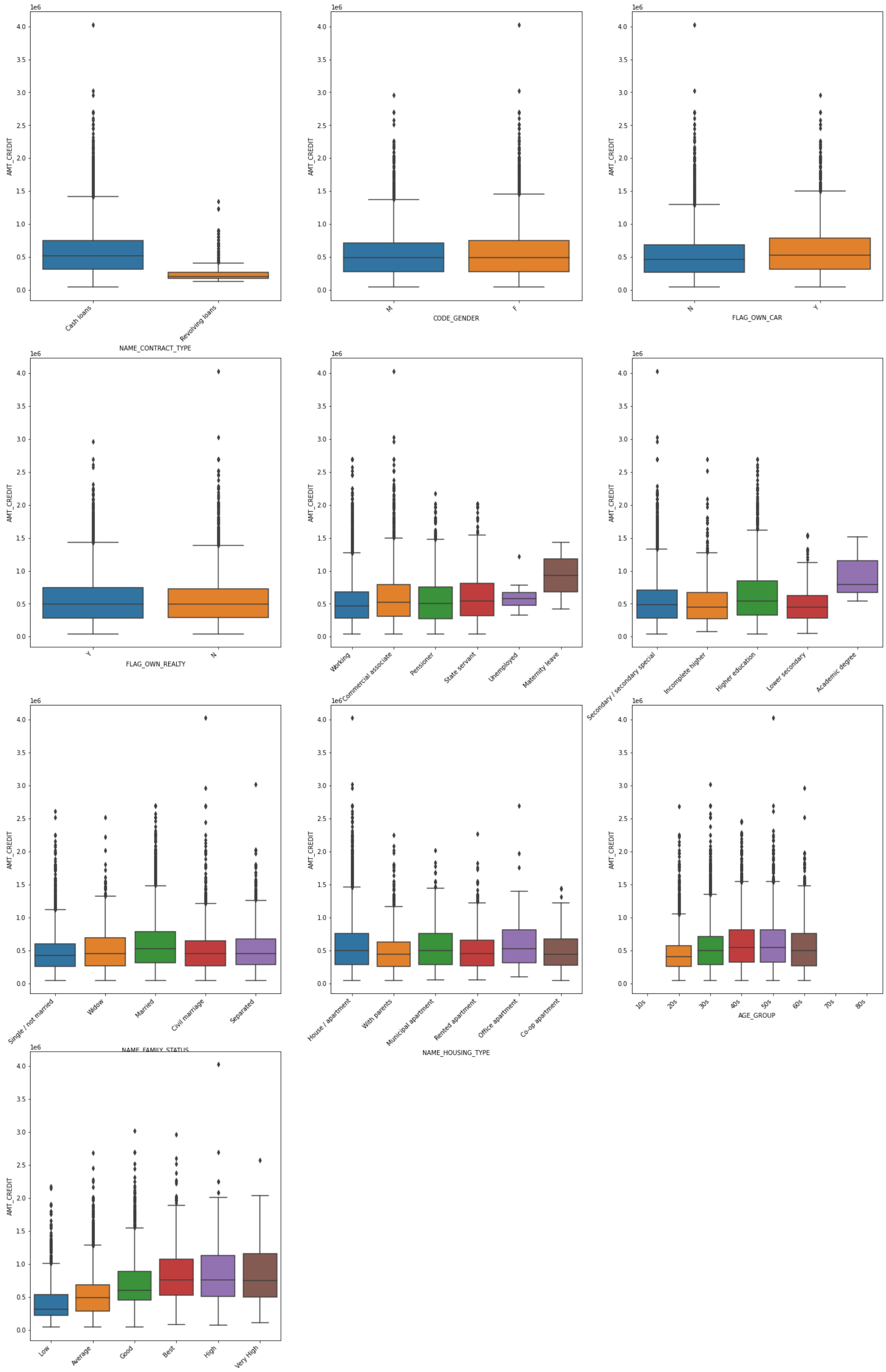
In [206…

```
categories = ['NAME_CONTRACT_TYPE','CODE_GENDER','FLAG_OWN_CAR','FLAG_OWN_REALTY','NA
              'NAME_FAMILY_STATUS','NAME_HOUSING_TYPE','AGE_GROUP','AMT_CATEGORY']
```

Analysis for Defaulters

In [254…

```
plt.figure(figsize=(25,40))
k=0
for category in categories:
    k = k+1
    ax = plt.subplot(4,3,k)
    sns.boxplot(x = category, y = 'AMT_CREDIT', data=target1_df_2)
    temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignmen

    plt.savefig('bivariate_defaulters.png')
```


1e6
AMT_CREDIT
Cash loans
Revolving loans
NAME_CONTRACT_TYPE
M
F
CODE_GENDER
N
Y
FLAG_OWN_CAR
Y
N
FLAG_OWN_REALTY
Working
Commercial associate
Pensioner
State servant
Unemployed
Maternity leave
Secondary / secondary special
Incomplete higher
Higher education
Lower secondary
Academic degree
Single / not married
Widow
Married
Civil marriage
Separated
NAME_FAMILY_STATUS
House / apartment
With parents
Municipal apartment
Rented apartment
Office apartment
Co-op apartment
NAME_HOUSING_TYPE
10s
20s
30s
40s
50s
60s
70s
80s
AGE_GROUP
Low
Average
Good
Best
High
Very High
AMT_CATEGORY

Analysis

Credit amount of the loans are very low for Revolving loans

There is no credit amount difference between genders, client owning cars or realty.

The Young age group got less amount of loan credited compared to mid age and senior citizen.

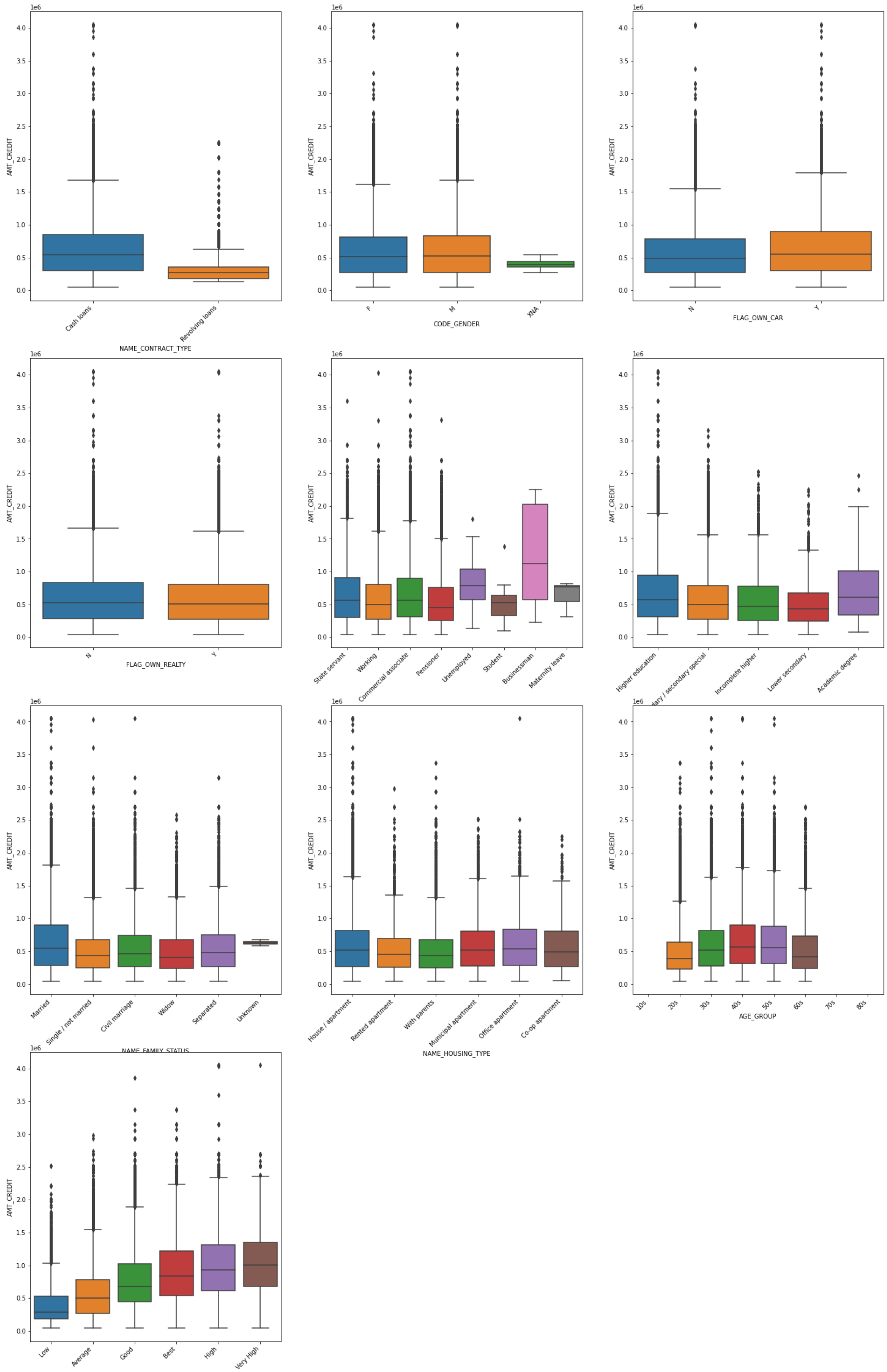
Higher income group have more loan amount credited.

Clients having higher external score have more loan amount.

Analysis for Non-Defaulters

In [255…

```
plt.figure(figsize=(25,40))
k=0
for category in categories:
    k = k+1
    ax = plt.subplot(4,3,k)
    sns.boxplot(x = category, y = 'AMT_CREDIT', data=target0_df_2)
    temp = ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignmen
    plt.savefig('bivariate_non_defaulters.png')
```


AMT_CREDIT
1e6
Cash loans
Revolving loans
NAME_CONTRACT_TYPE
F
M
XNA
CODE_GENDER
N
Y
FLAG_OWN_CAR
FLAG_OWN_REALTY
State servant
Working
Commercial associate
Pensioner
Unemployed
Student
Businessman
Maternity leave
Higher education
Incomplete higher
Lower secondary
Academic degree
Married
Single / not married
Civil marriage
Widow
Separated
Unknown
NAME_FAMILY_STATUS
House / apartment
Rented apartment
With parents
Municipal apartment
Office apartment
Co-op apartment
NAME_HOUSING_TYPE
10s
20s
30s
40s
50s
60s
70s
80s
AGE_GROUP
Low
Average
Good
Best
High
Very High
AMT_CATEGORY

Analysis

Credit amount of the loans are very low for Revolving loans
There is no credit amount differnce between genders, client owning cars or realty.
The mid age group got more amount of loan credited compared to young and senior citizen.
Higher income group have more loan amount credited and lower the lowest.
Clients having higher external score have more loan amount.
Surprisingly the unemployed people have spike in credit amount of loan
The Married people have more loan amount credited.

Previous application Data

Reading the Previous Application in Pandas Dataframe

In [209...

```
df_application_previous = pd.read_csv('C:/Users/SAGNICK/TrainityBankLoneEDA/LoanCase$
df_application_previous.head()
```

Out[209...

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CRED
0	2030495	271877	Consumer loans	1730.430	17145.0	1714!
1	2802425	108129	Cash loans	25188.615	607500.0	67967'
2	2523466	122040	Cash loans	15060.735	112500.0	13644
3	2819243	176158	Cash loans	47041.335	450000.0	47079(
4	1784265	202054	Cash loans	31924.395	337500.0	40405!

In [210...

```
df_application_previous.shape
```

Out[210...

```
(1670214, 37)
```

In [211...

```
df_application_previous.describe()
```

Out[211...

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_P
count	1.670214e+06	1.670214e+06	1.297979e+06	1.670214e+06	1.670213e+06	7.74:
mean	1.923089e+06	2.783572e+05	1.595512e+04	1.752339e+05	1.961140e+05	6.697
std	5.325980e+05	1.028148e+05	1.478214e+04	2.927798e+05	3.185746e+05	2.09:
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00	0.000000e+00	-9.00

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_P
25%	1.461857e+06	1.893290e+05	6.321780e+03	1.872000e+04	2.416050e+04	0.000
50%	1.923110e+06	2.787145e+05	1.125000e+04	7.104600e+04	8.054100e+04	1.638
75%	2.384280e+06	3.675140e+05	2.065842e+04	1.803600e+05	2.164185e+05	7.740

In [212…

```
df_application_previous.info()
```

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

Creating copy of the dataset

In [213…

```
df_application_previous_copy=df_application_previous.copy(deep=True)
df_application_previous_copy
```

Out[213...

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AM
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	
...	...	...	...	...	...	
1670209	2300464	352015	Consumer loans	14704.290	267295.5	
1670210	2357031	334635	Consumer loans	6622.020	87750.0	
1670211	2659632	249544	Consumer loans	11520.855	105237.0	
1670212	2785582	400317	Cash loans	18821.520	180000.0	
1670213	2418762	261212	Cash loans	16431.300	360000.0	

1670214 rows × 37 columns

Data Quality Check And Missing Values

In [215...

```
Perc_Of_NA_Columns_prev=round(df_application_previous_copy.isnull().sum()/len(df_appl
Perc_Of_NA_Columns_prev
```

Out[215...

```
SK_ID_PREV                    0.00
SK_ID_CURR                    0.00
NAME_CONTRACT_TYPE            0.00
AMT_ANNUITY                  22.29
AMT_APPLICATION               0.00
AMT_CREDIT                    0.00
AMT_DOWN_PAYMENT             53.64
AMT_GOODS_PRICE              23.08
WEEKDAY_APPR_PROCESS_START    0.00
HOUR_APPR_PROCESS_START       0.00
FLAG_LAST_APPL_PER_CONTRACT   0.00
NFLAG_LAST_APPL_IN_DAY        0.00
RATE_DOWN_PAYMENT            53.64
RATE_INTEREST_PRIMARY        99.64
RATE_INTEREST_PRIVILEGED     99.64
NAME_CASH_LOAN_PURPOSE        0.00
NAME_CONTRACT_STATUS          0.00
DAYS_DECISION                 0.00
NAME_PAYMENT_TYPE             0.00
CODE_REJECT_REASON            0.00
NAME_TYPE_SUITE              49.12
NAME_CLIENT_TYPE              0.00
NAME_GOODS_CATEGORY           0.00
NAME_PORTFOLIO                0.00
NAME_PRODUCT_TYPE             0.00
CHANNEL_TYPE                  0.00
```

```
SELLERPLACE_AREA              0.00
NAME_SELLER_INDUSTRY          0.00
CNT_PAYMENT                  22.29
NAME_YIELD_GROUP              0.00
PRODUCT_COMBINATION           0.02
DAYS_FIRST_DRAWING           40.30
DAYS_FIRST_DUE               40.30
DAYS_LAST_DUE_1ST_VERSION    40.30
DAYS_LAST_DUE                40.30
DAYS_TERMINATION             40.30
NFLAG_INSURED_ON_APPROVAL    40.30
dtype: float64
```

Checking for the columns where NA values are >50%

In [216…

```
Columns_with_min50_NA_Values_prev=Perc_Of_NA_Columns_prev[Perc_Of_NA_Columns_prev>50]
Columns_with_min50_NA_Values_prev
```

Out[216…

```
AMT_DOWN_PAYMENT            53.64
RATE_DOWN_PAYMENT           53.64
RATE_INTEREST_PRIMARY       99.64
RATE_INTEREST_PRIVILEGED    99.64
dtype: float64
```

Dropping the columns where >50% data are NA

In [217…

```
df_application_previous_copy=df_application_previous_copy.drop(Columns_with_min50_NA_
df_application_previous_copy.shape
```

Out[217…

```
(1670214, 33)
```

Merging the Application data with witht Previous Application Data on SK_ID_CURR column

In [218…

```
df_application_current_copy.shape
```

Out[218…

```
(307511, 83)
```

In [219…

```
drop_columns
```

Out[219…

```
['FLAG_CONT_MOBILE',
 'FLAG_MOBIL',
 'FLAG_EMP_PHONE',
 'FLAG_WORK_PHONE',
 'FLAG_PHONE',
 'FLAG_EMAIL',
 'HOUR_APPR_PROCESS_START',
 'WEEKDAY_APPR_PROCESS_START',
 'FLOORSMAX_AVG',
 'EXT_SOURCE_2',
 'EXT_SOURCE_3',
 'FLOORSMAX_AVG',
 'FLOORSMAX_MODE',
 'FLOORSMAX_MEDI',
 'TOTALAREA_MODE',
 'EMERGENCYSTATE_MODE',
```

```
 'REGION_POPULATION_RELATIVE',
 'YEARS_BEGINEXPLUATATION_AVG',
 'YEARS_BEGINEXPLUATATION_MEDI',
 'YEARS_BEGINEXPLUATATION_MODE',
 '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',
 'FLAG_DOCUMENT_2',
 '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']
```

Dropping all the irrelevant columns from Current data

In [220…

```
df_application_current_copy=df_application_current_copy.drop(drop_columns,axis=1)
df_application_current_copy.shape
```

Out[220…

```
(307511, 39)
```

Merging both the datasets

In [221…

```
combined_df = pd.merge(left =df_application_current_copy, right =df_application_prev:
combined_df.shape
```

Out[221…

```
(1413701, 71)
```

Getting the different Types of contract statuses:

In [222…

```
combined_df.NAME_CONTRACT_STATUS.unique()
```

Out[222…

```
array(['Approved', 'Canceled', 'Refused', 'Unused offer'], dtype=object)
```

Doing analysis on People with Contract Status as Approved

In [226…

```
combined_approved_df = combined_df[combined_df.NAME_CONTRACT_STATUS == 'Approved']
print(combined_approved_df.shape)
combined_approved_df.head(5)
```

(886099, 71)

Out[226…

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REA
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	

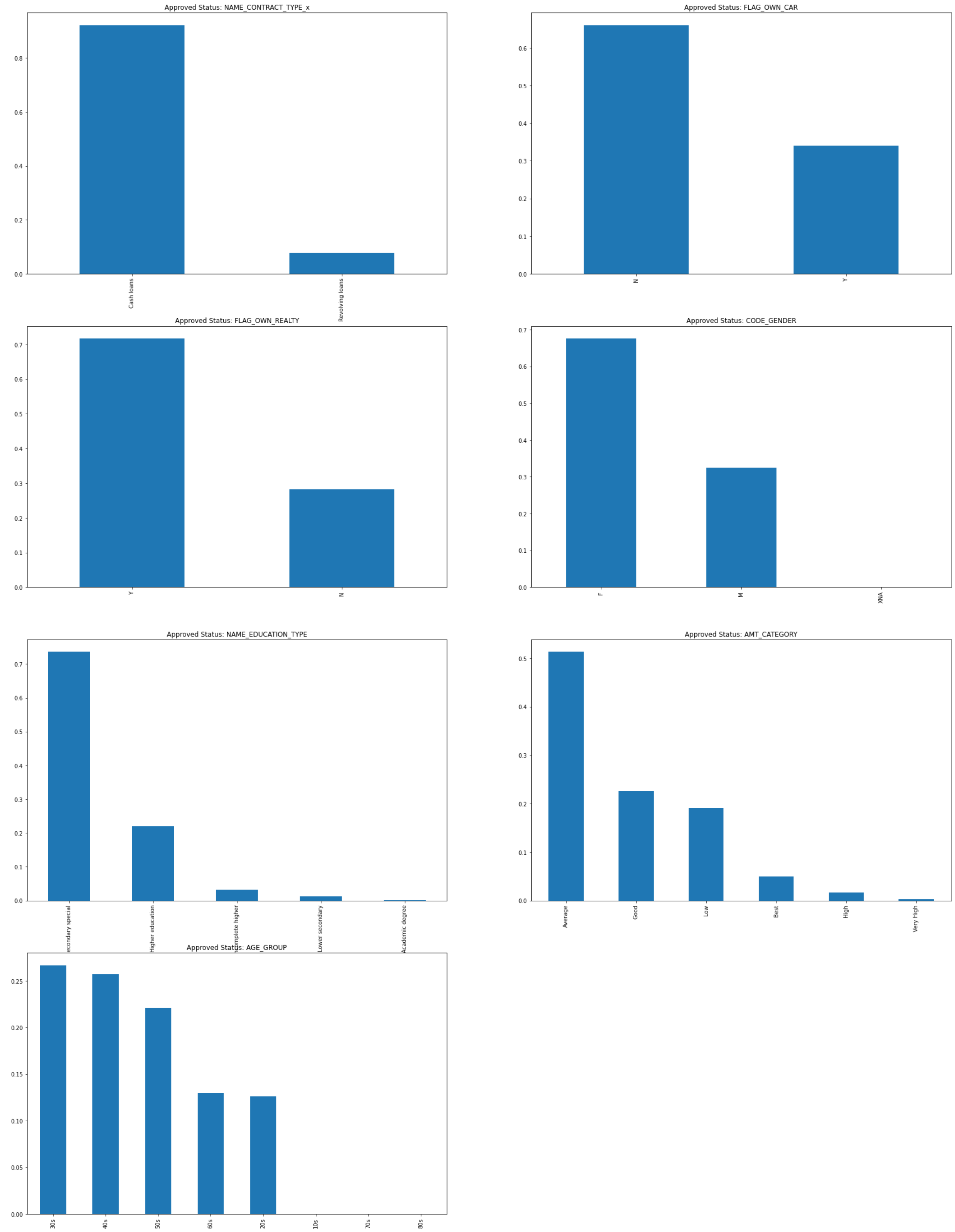
Univariate Analysis for few categorical columns in combined dataframe

In [227…

```
combined_categorical_columns = ['NAME_CONTRACT_TYPE_x',
                               'FLAG_OWN_CAR',
                               'FLAG_OWN_REALTY',
                               'CODE_GENDER',
                               'NAME_EDUCATION_TYPE',
                               'AMT_CATEGORY',
                               'AGE_GROUP']
```

In [271…

```
plt.figure(figsize=(30,40))
k=0
for i in combined_categorical_columns:
    k+=1
    ax=plt.subplot(4,2,k)
    combined_approved_df[i].value_counts(normalize=True).plot.bar()
    plt.title("Approved Status: "+ i)
```



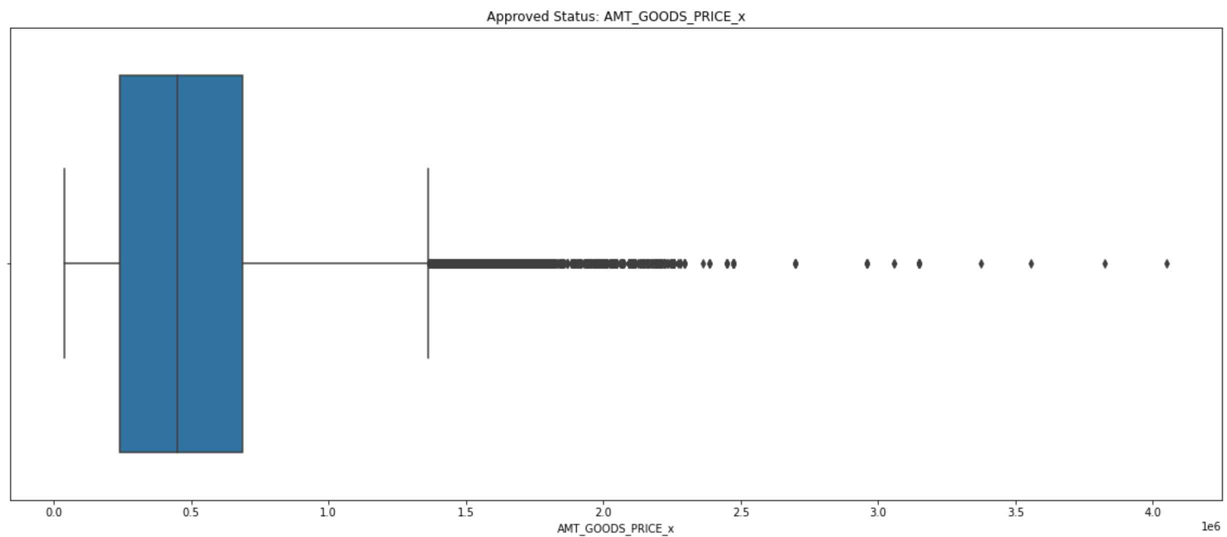
Univariate Analysis for few numerical columns in combined dataframe

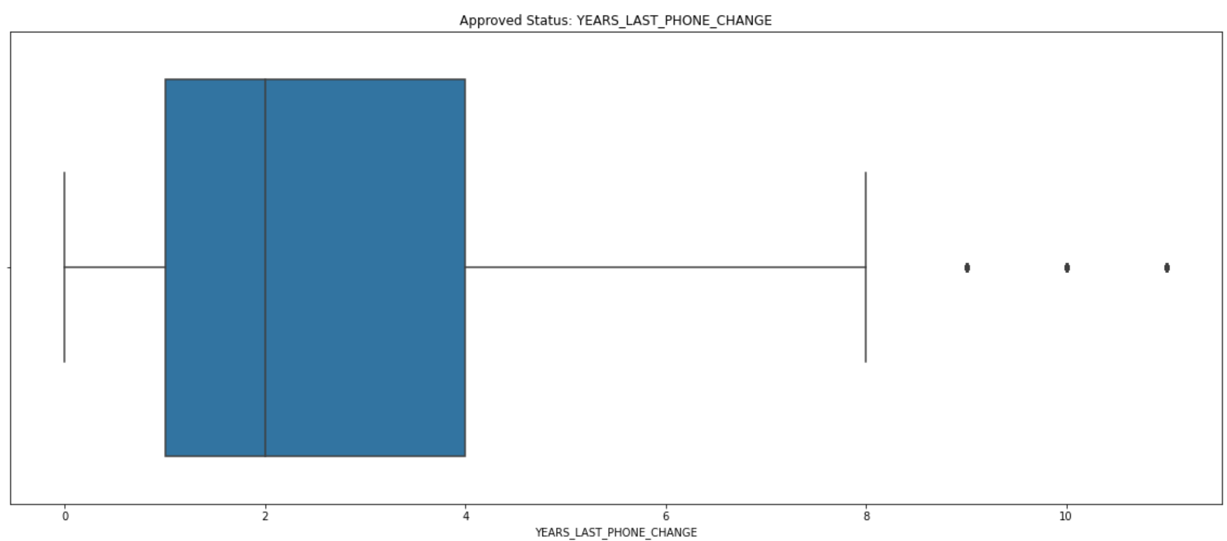
In [233…

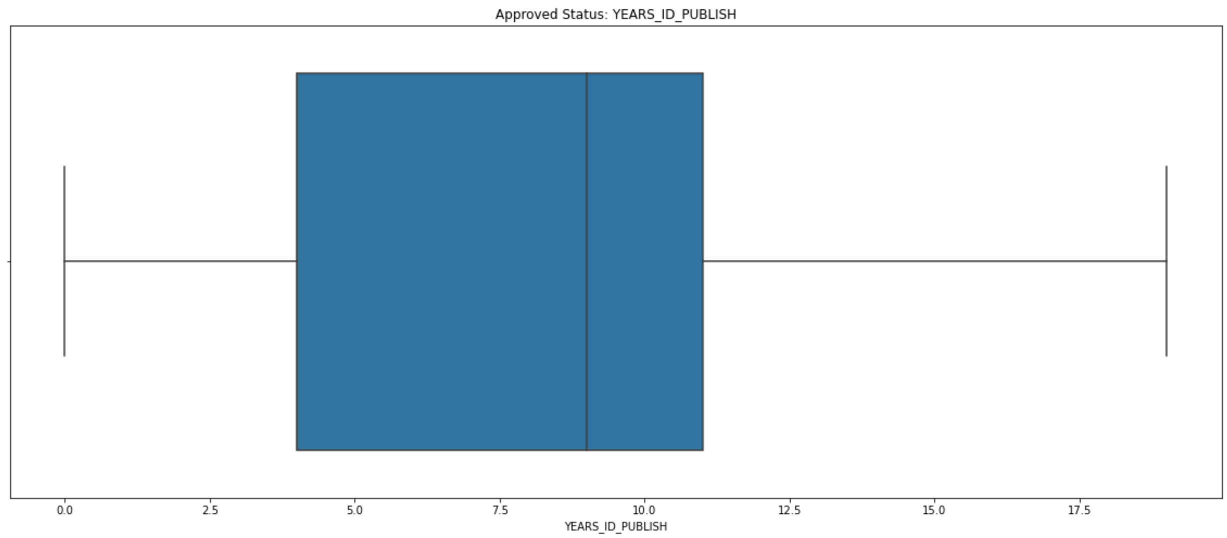
```
combined_numerical_columns= ['AMT_GOODS_PRICE_x',
                    'YEARS_LAST_PHONE_CHANGE',
                    'YEARS_ID_PUBLISH',
                    'AMT_INCOME_TOTAL',
                    'YEARS_EMPLOYED',
                    'YEARS_REGISTRATION',
                    'AGE',
                    'AMT_CREDIT_x',
                    'AMT_ANNUITY_x'
                   ]
```

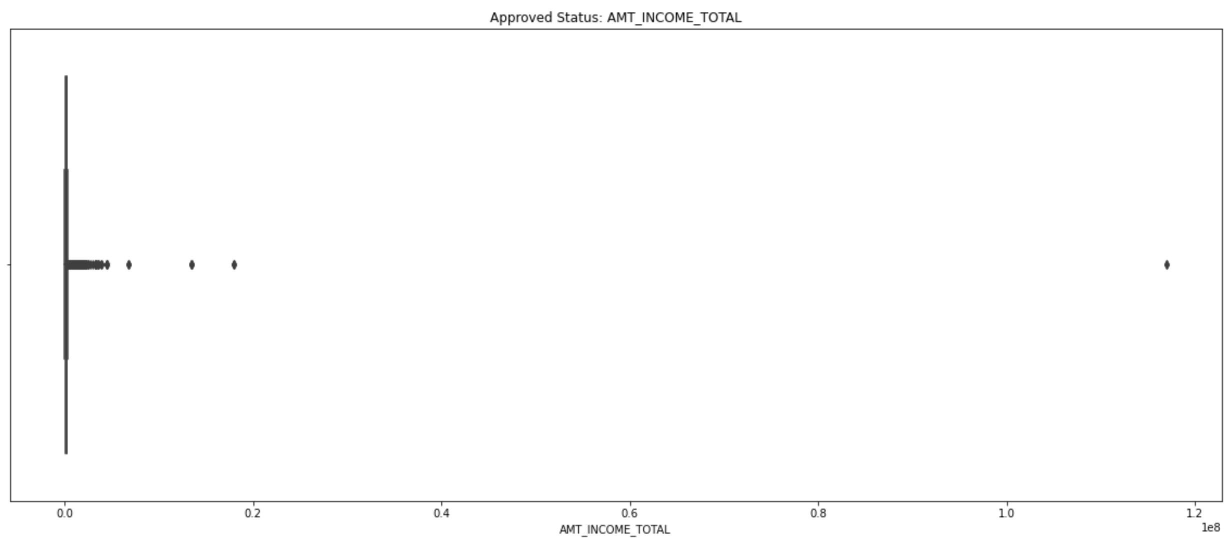
In [234…

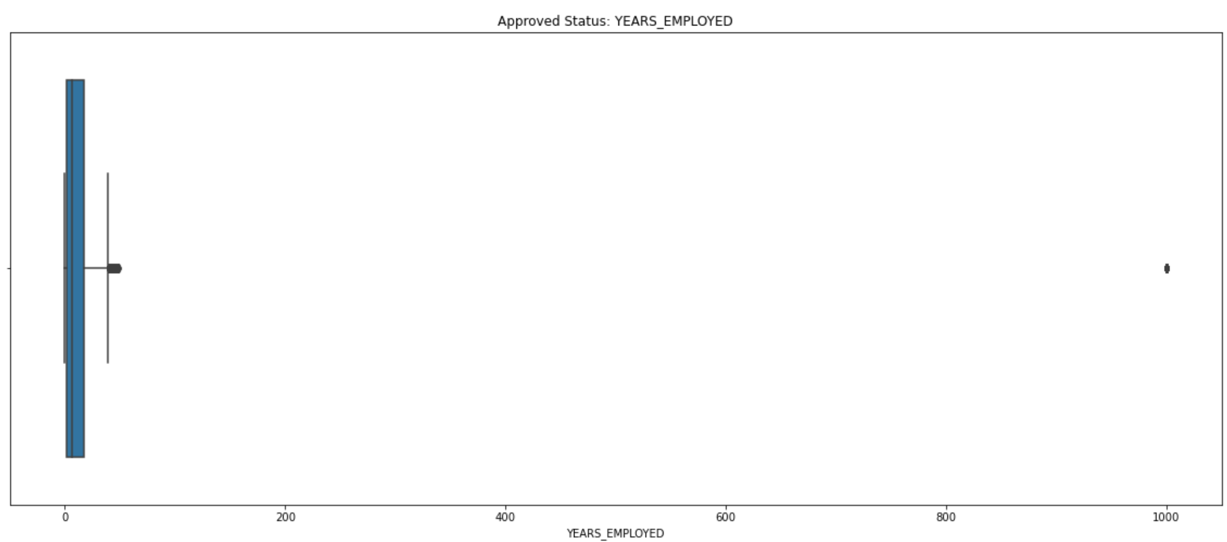
```
for i in combined_numerical_columns:
    plt.figure(figsize=(20,8))
    sns.boxplot(combined_approved_df[i])
    plt.title("Approved Status: "+i)
```

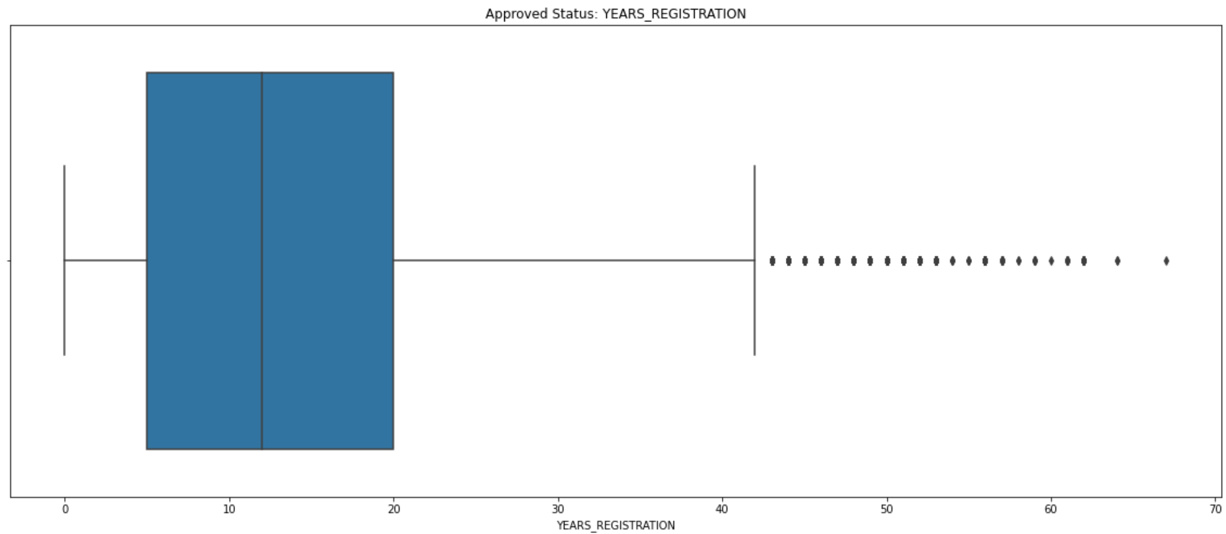


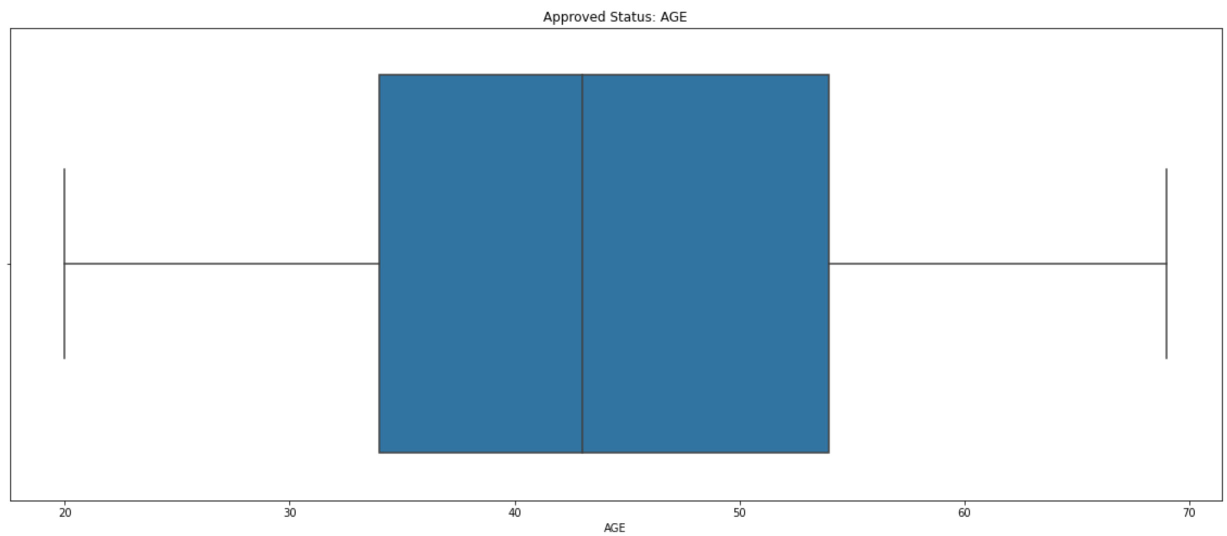


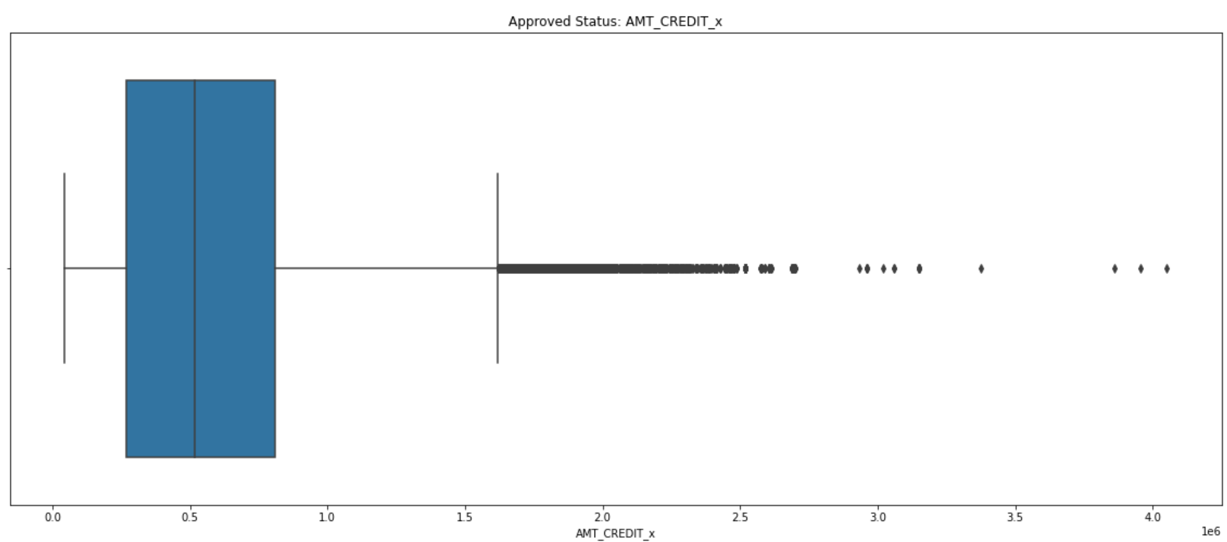

Approved Status: YEARS_ID_PUBLISH
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
YEARS_ID_PUBLISH

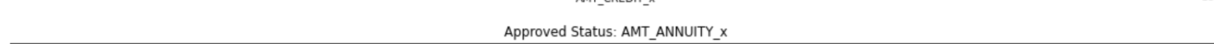

Approved Status: AMT_INCOME_TOTAL
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e8
AMT_INCOME_TOTAL


Approved Status: YEARS_EMPLOYED
0
200
400
600
800
1000
YEARS_EMPLOYED





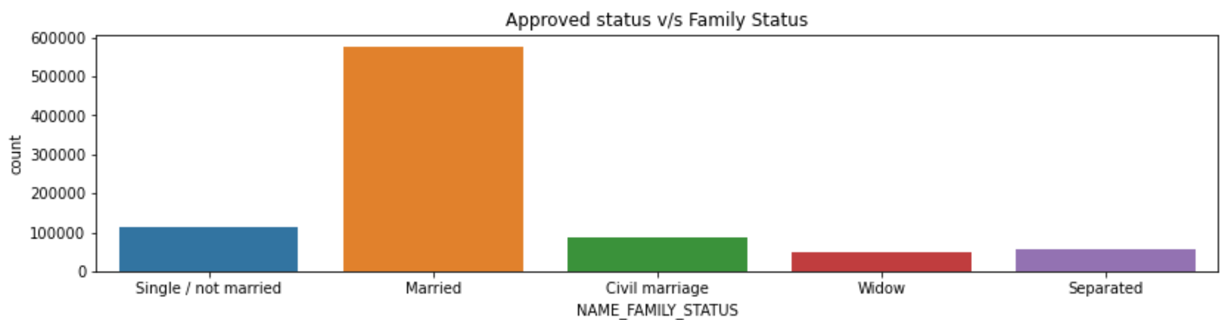


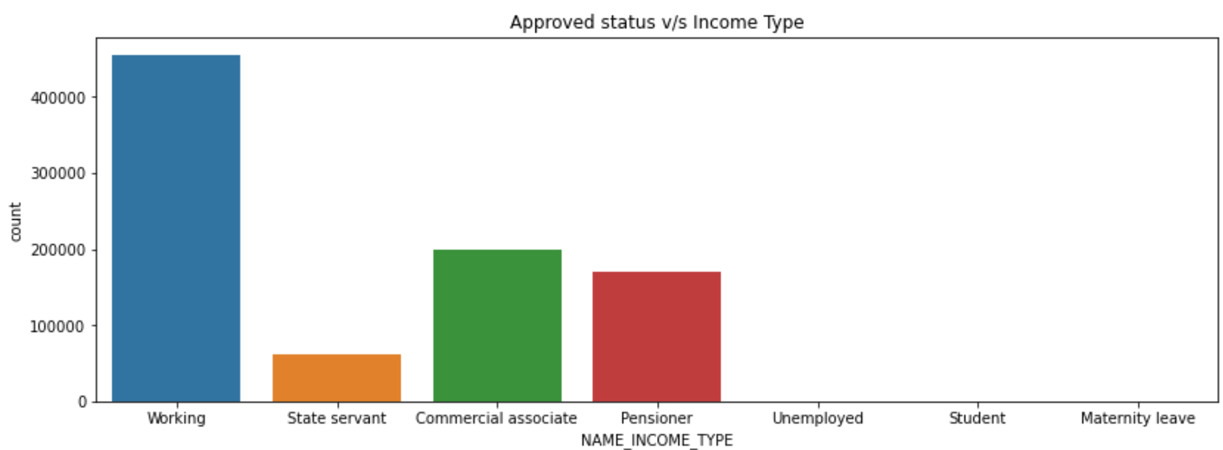


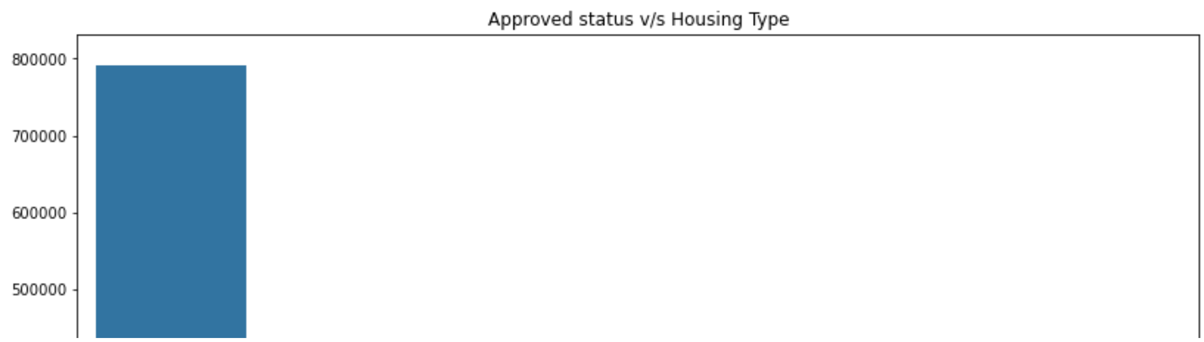
Bivariate Analysis for few Categorical Variable of combined dataframe

In [235…

```
# 1. Second Variable : Family Status
plt.figure(figsize=(30,10))
plt.subplot(3,2,2)
sns.countplot(x = "NAME_FAMILY_STATUS", data = combined_approved_df)
plt.title("Approved status v/s Family Status")
plt.show()

# 2. Second Variable : Income Type
plt.figure(figsize=(30,10))
plt.subplot(2,2,1)
sns.countplot(x = "NAME_INCOME_TYPE", data = combined_approved_df)
plt.title("Approved status v/s Income Type")
plt.show()

# 3. Second Variable : Housing type
plt.figure(figsize=(30,8))
plt.subplot(1,2,1)
sns.countplot(x = "NAME_HOUSING_TYPE", data = combined_approved_df)
plt.title("Approved status v/s Housing Type")
plt.show()
```







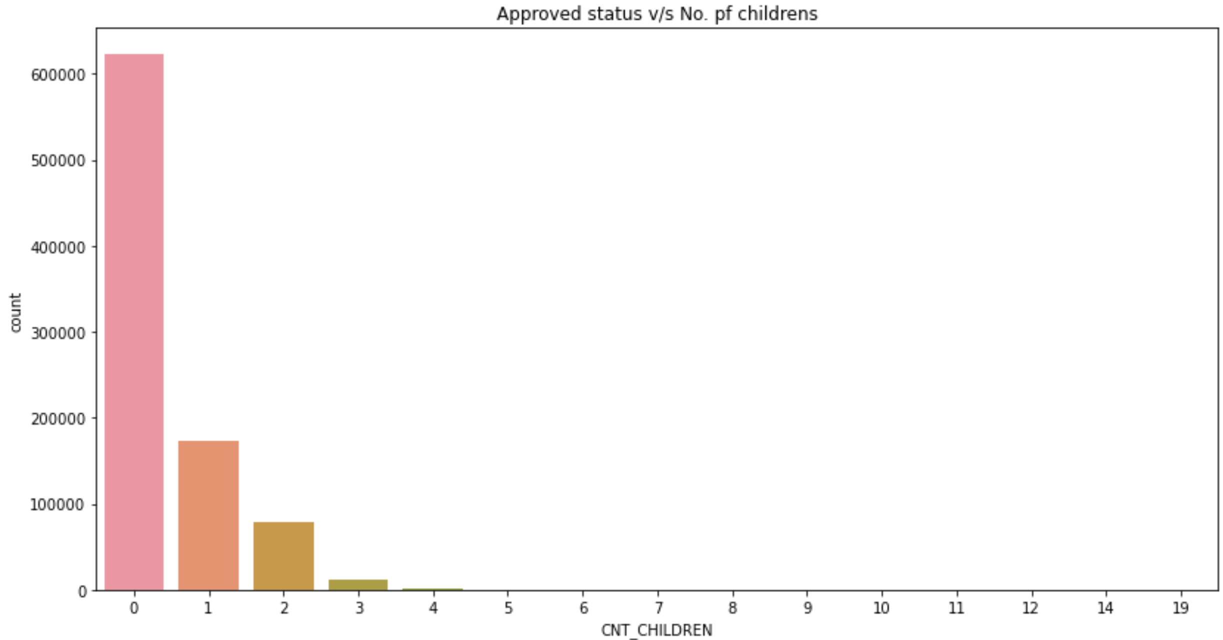
Insights from the above Bivariant analysis for categorical data is as below:

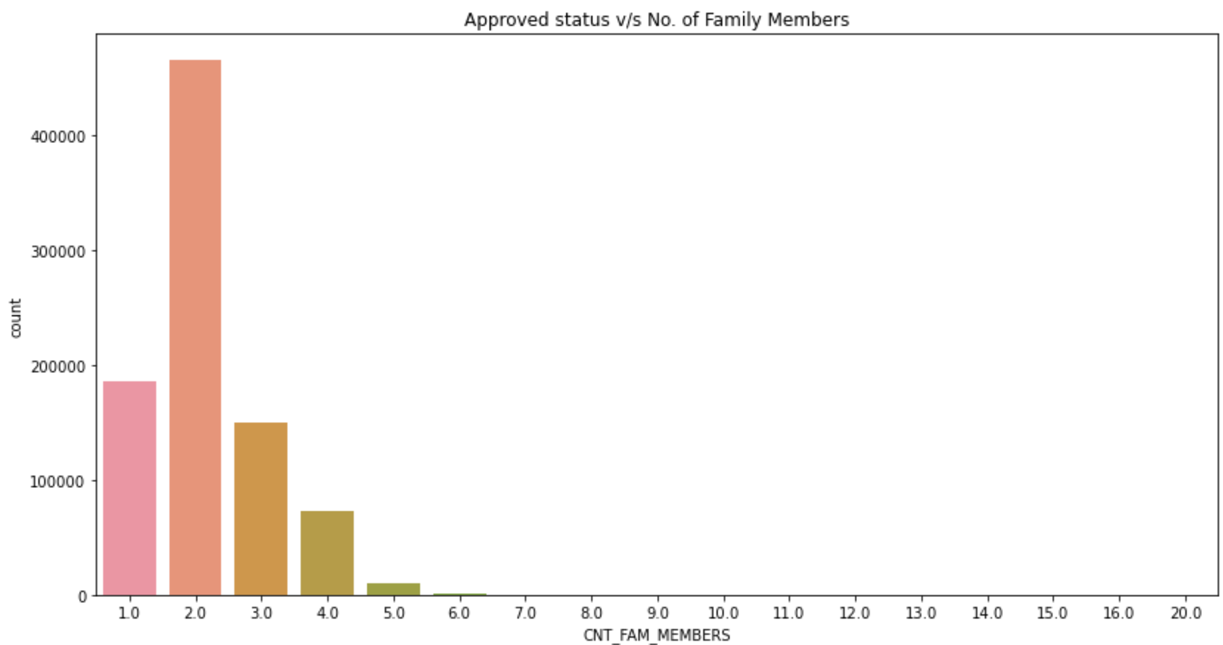
1. Approved status v/s Family Status: People who are married are more likely to get loan approved
2. Approved status v/s Income Type: People who are working are more likely to get loan approved compared to students who are least likely to get loan approved
3. Approved status v/s Housing_type: People who own House/apartment are more likely to get loan approved then compared to rented appartments/ co-op apartment types

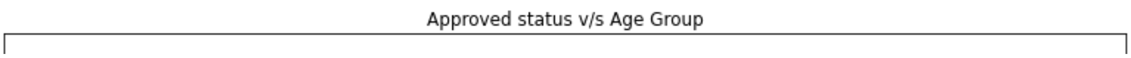
Bivariant analysis for Numerical Data.

In [236…

```
# 1. Second Variable : CNT_CHILDREN
plt.figure(figsize=(30,7))
plt.subplot(1,2,1)
sns.countplot(x = "CNT_CHILDREN", data = combined_approved_df)
plt.title("Approved status v/s No. pf childrens")
plt.show()

# 2. Second Variable : CNT_FAM_MEMBERS
plt.figure(figsize=(30,7))
plt.subplot(1,2,1)
sns.countplot(x = "CNT_FAM_MEMBERS", data = combined_approved_df)
plt.title("Approved status v/s No. of Family Members")
plt.show()

# 3. Second Variable : Age
plt.figure(figsize=(30,7))
plt.subplot(1,2,1)
sns.countplot(x="AGE_GROUP", data = combined_approved_df)
plt.title("Approved status v/s Age Group")
plt.show()
```


Approved status v/s No. pf childrens
count
600000
500000
400000
300000
200000
100000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 14 19
CNT_CHILDREN


Approved status v/s No. of Family Members
count
400000
300000
200000
100000
0
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0 20.0
CNT_FAM_MEMBERS



Inference from the above Bivariant analysis for numerical data is as below:

1. Approved status v/s No. of children: People with 0 children are more likely to get loan approved
2. Approved status v/s No. of family members: If the number of people in a family is 2 they are more likely to get loan approved.
3. Approved status v/s Age: People with age in between 30 -50 years are more likely to get loan approved compared to the people in 20s and 60s

Analysis on People with Contract Status as Refused

In [237…

```
combined_refused_df = combined_df[combined_df.NAME_CONTRACT_STATUS == 'Refused']
print(combined_refused_df.shape)
combined_refused_df.head(5)
```

```
(245390, 71)
```

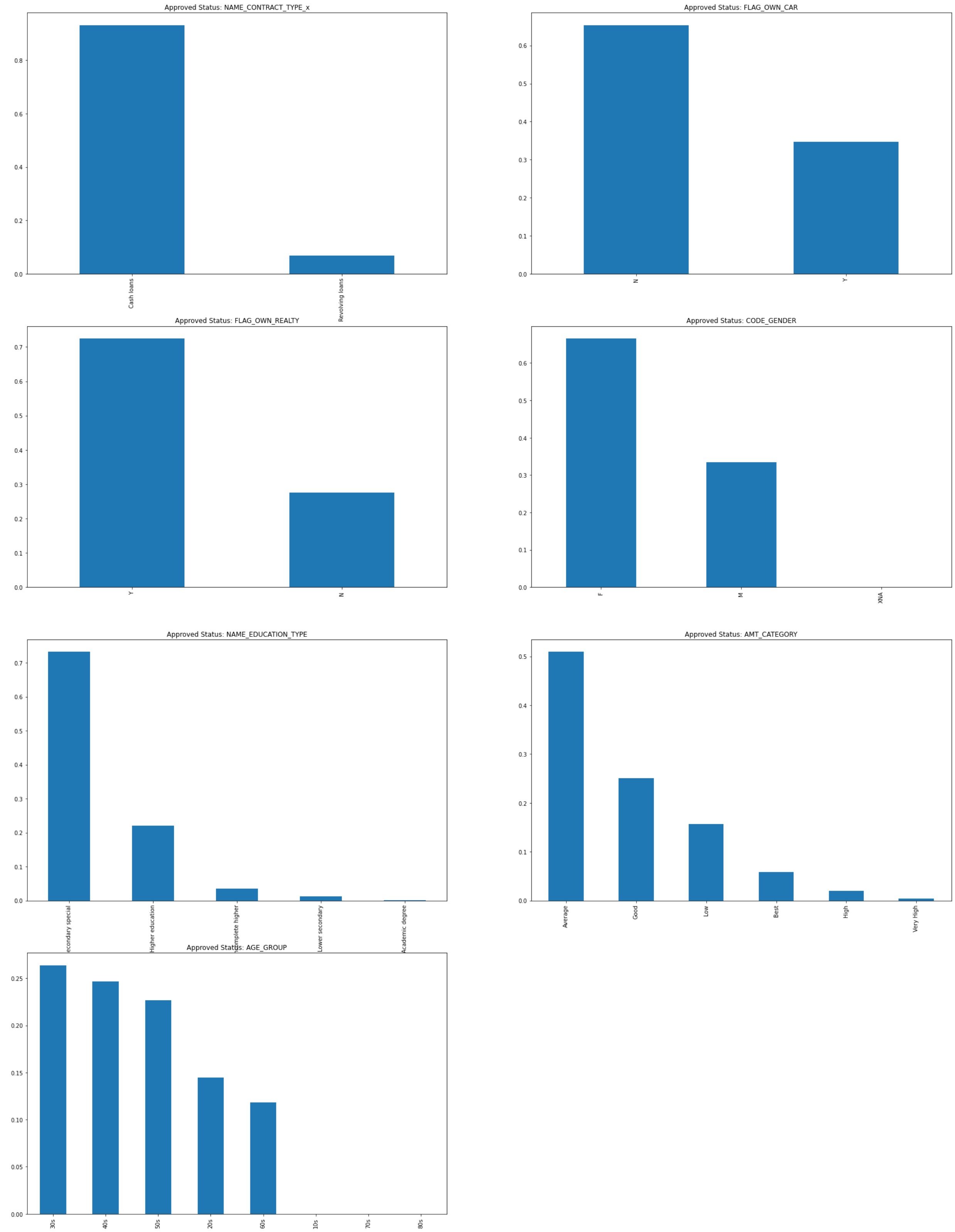
Out[237…

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_RE
13	100006	0	Cash loans	F	N	
33	100011	0	Cash loans	F	N	
79	100027	0	Cash loans	F	N	
84	100030	0	Cash loans	F	N	
85	100030	0	Cash loans	F	N	

Univariate Analysis for few categorical columns in combined dataframe

In [272…

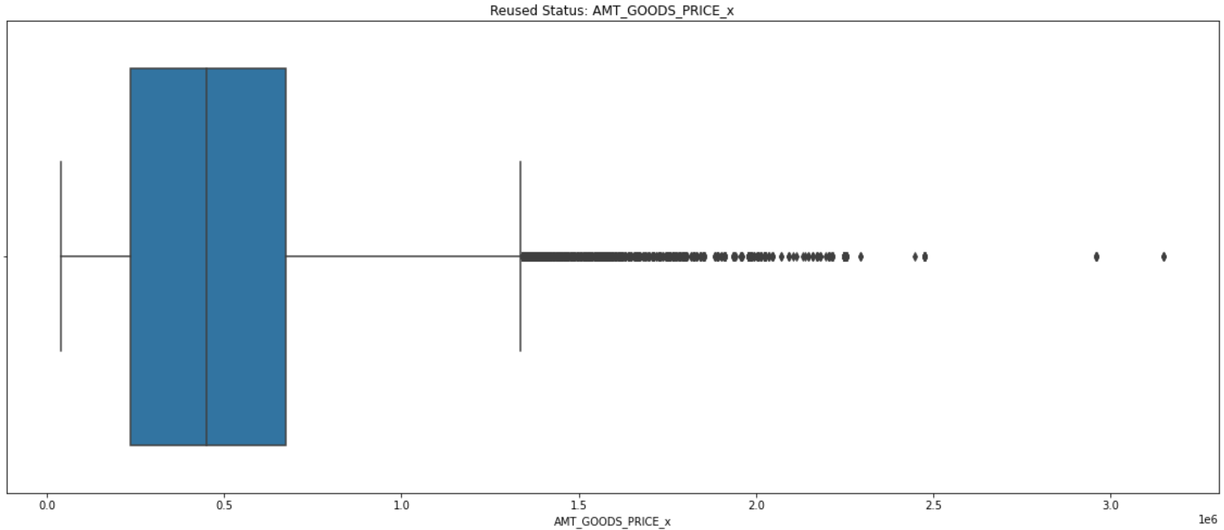
```
plt.figure(figsize=(30,40))
k=0
for i in combined_categorical_columns:
    k+=1
    ax=plt.subplot(4,2,k)
    combined_refused_df[i].value_counts(normalize=True).plot.bar()
    plt.title("Approved Status: "+ i)
```

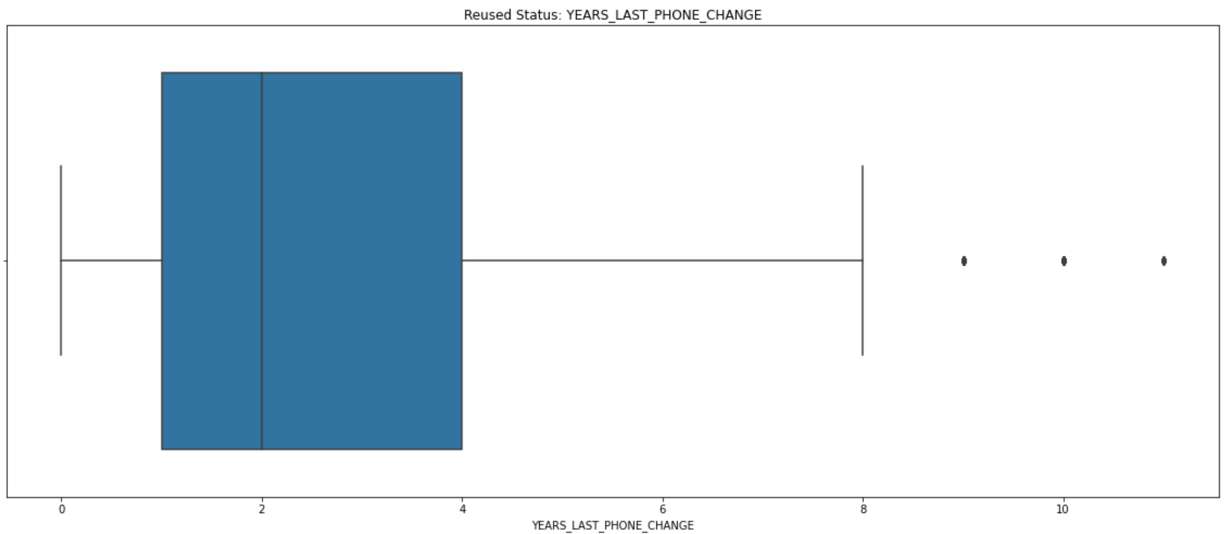


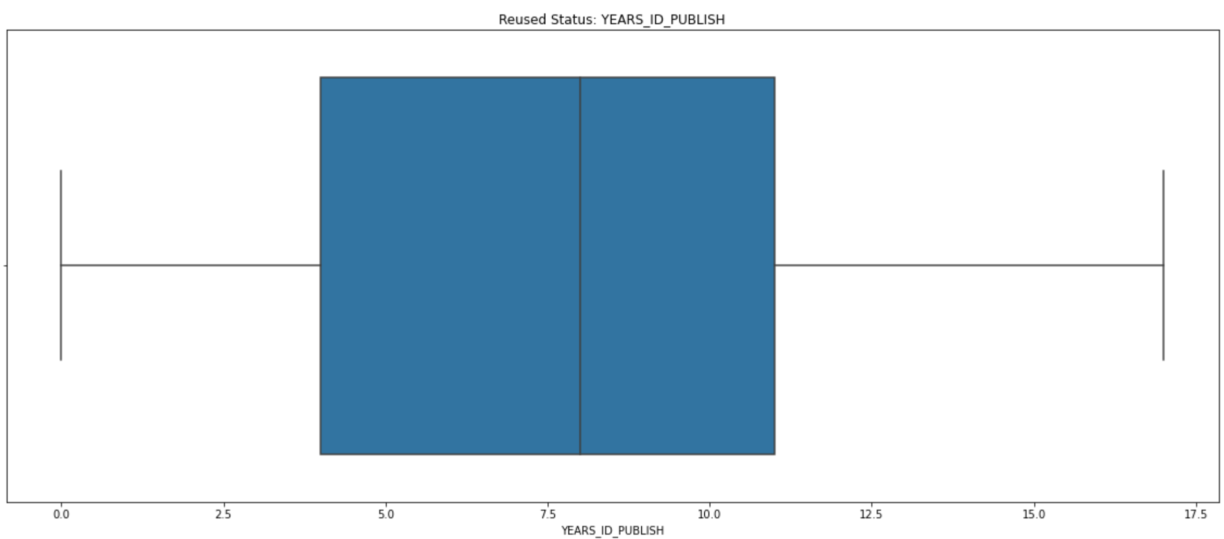
Univariate Analysis for few numerical columns in combined dataframe

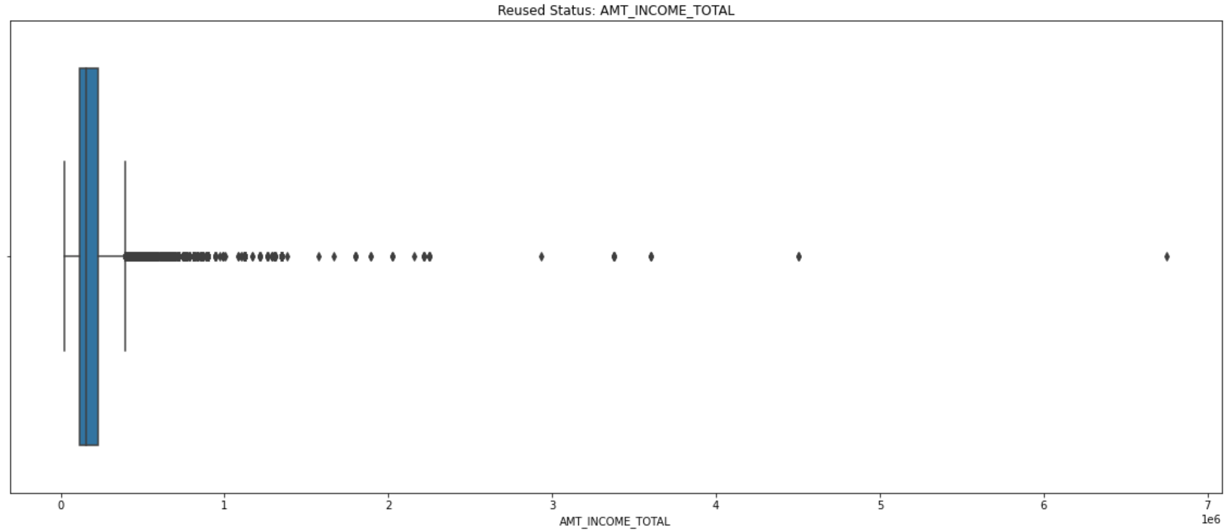
In [239…

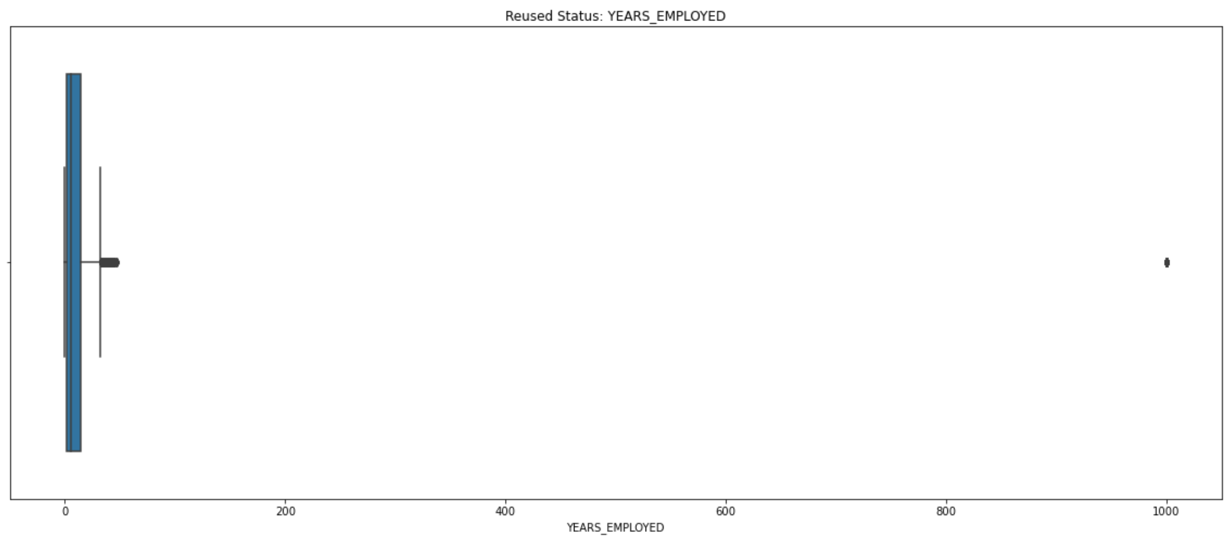
```
for i in combined_numerical_columns:
    plt.figure(figsize=(20,8))
    sns.boxplot(combined_refused_df[i])
    plt.title("Reused Status: "+i)
```

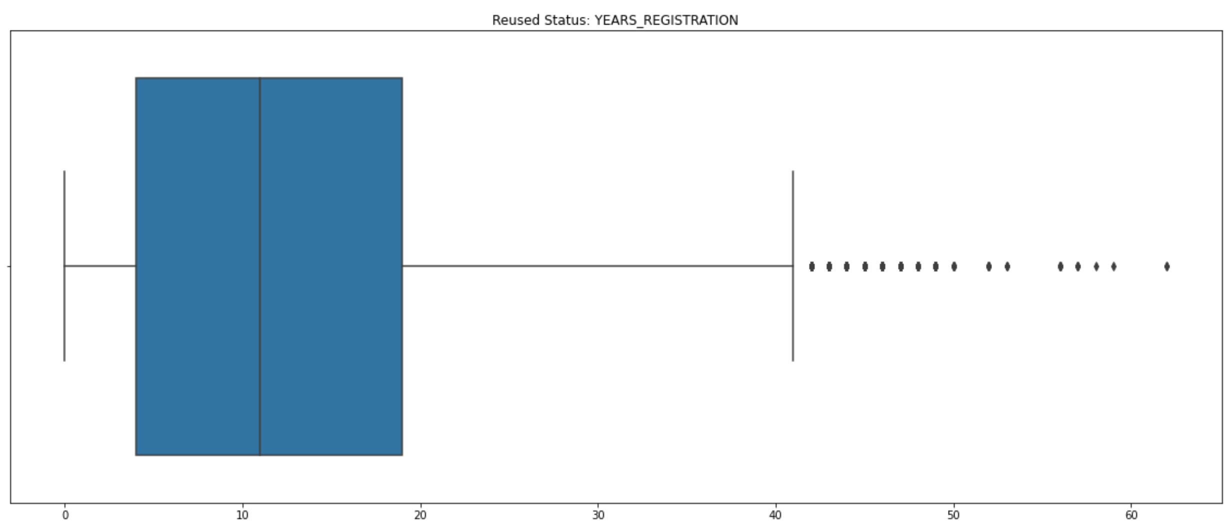


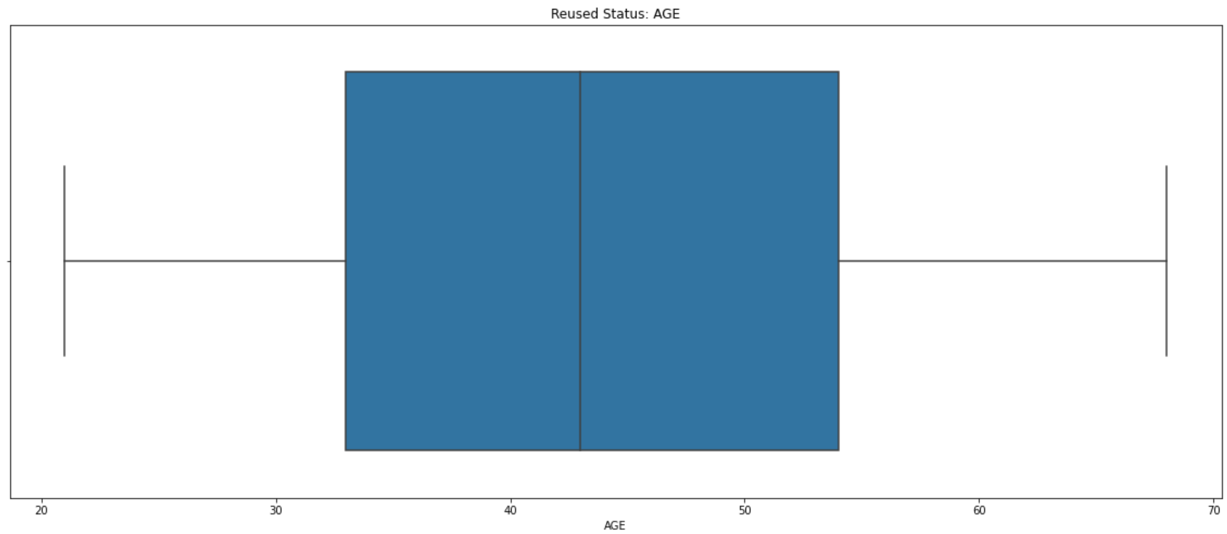


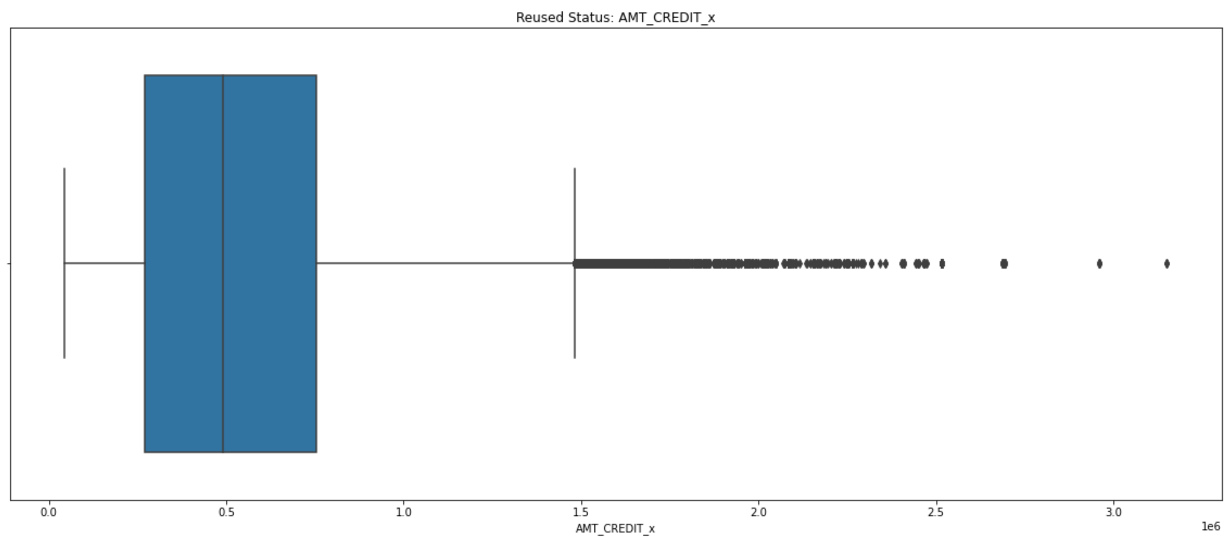


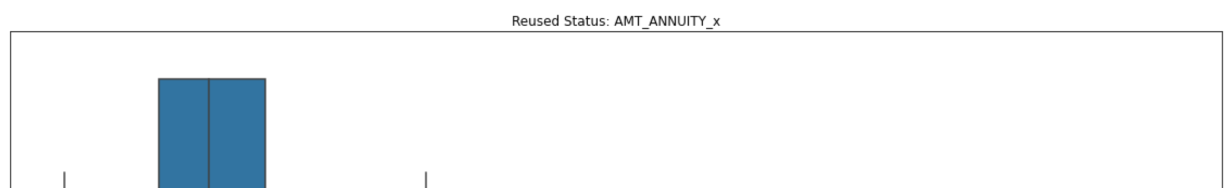

Reused Status: AMT_INCOME_TOTAL
0
1
2
3
4
5
6
7
1e6
AMT_INCOME_TOTAL


Reused Status: YEARS_EMPLOYED
0
200
400
600
800
1000
YEARS_EMPLOYED


Reused Status: YEARS_REGISTRATION
0
10
20
30
40
50
60
YEARS_REGISTRATION



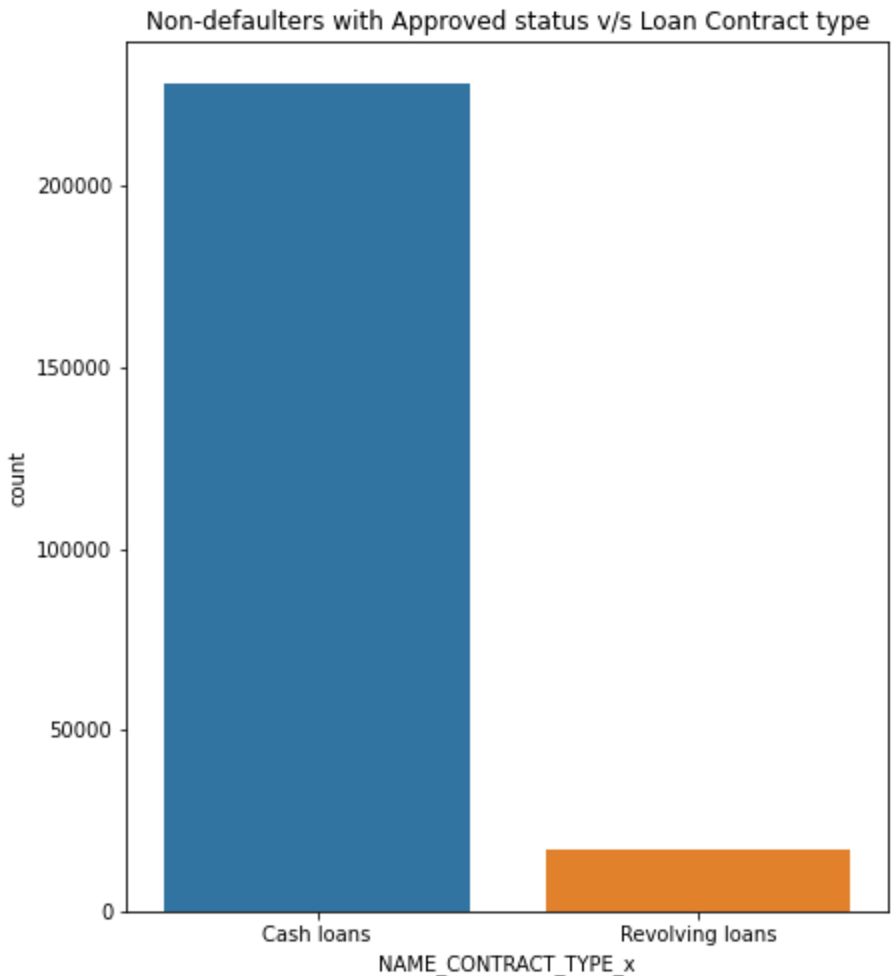




Bivariant analysis for Categorical Data for the refused applications

In [241…

```
plt.figure(figsize=(15,8))
plt.subplot(1,2,1)
sns.countplot(x = "NAME_CONTRACT_TYPE_x", data = combined_refused_df)
plt.title("Non-defaulters with Approved status v/s Loan Contract type")
plt.show()
```



Doing analysis on People with Contract Status as Canceled

In [243…

```
combined_cancelled_df=combined_df[combined_df.NAME_CONTRACT_STATUS == 'Canceled']
print(combined_cancelled_df.shape)
combined_cancelled_df
```

(259441, 71)

Out[243…

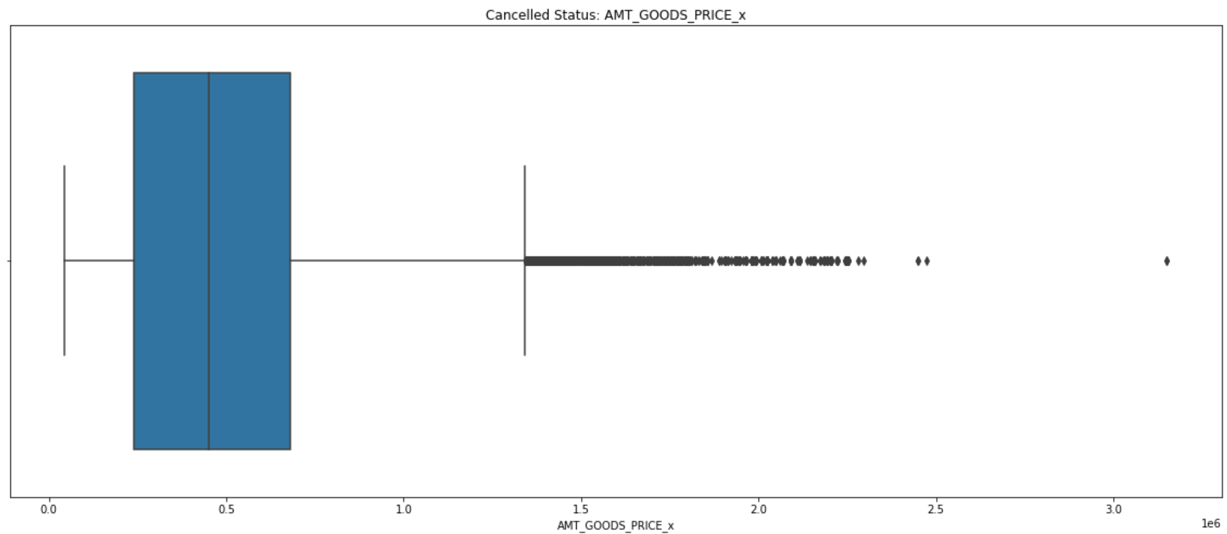
	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG_O\
6	100006	0	Cash loans	F	N	
10	100006	0	Cash loans	F	N	
12	100006	0	Cash loans	F	N	
21	100008	0	Cash loans	M	N	
40	100012	0	Revolving loans	M	N	
...	...	...	...	...	...	
1413658	456244	0	Cash loans	F	N	

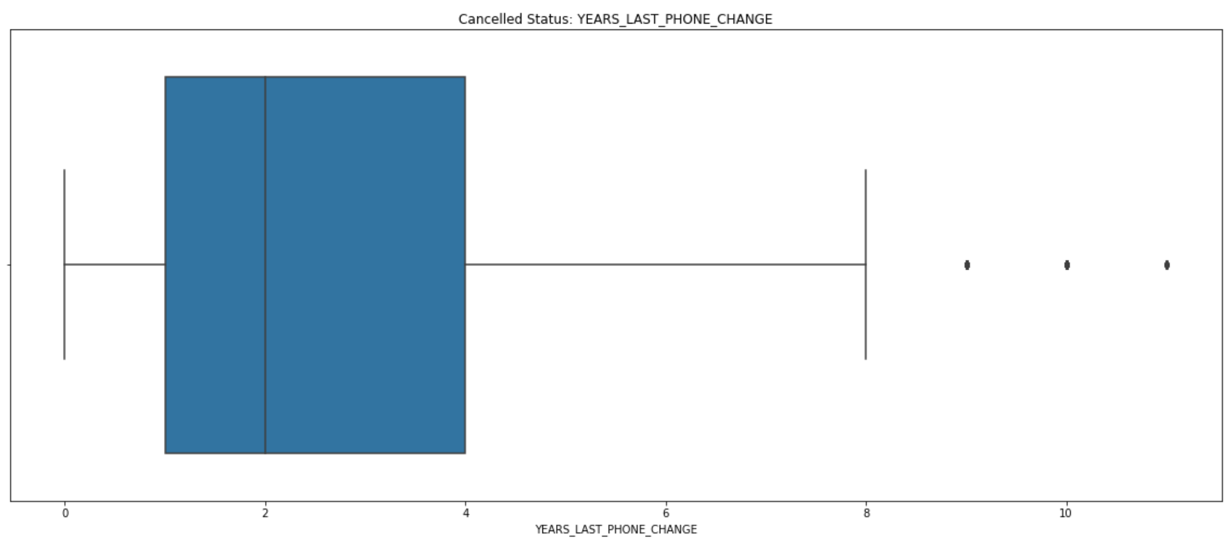
	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG_OV
1413661	456244	0	Cash loans	F	N	
1413663	456244	0	Cash loans	F	N	
1413666	456244	0	Cash loans	F	N	
1413667	456244	0	Cash loans	F	N	

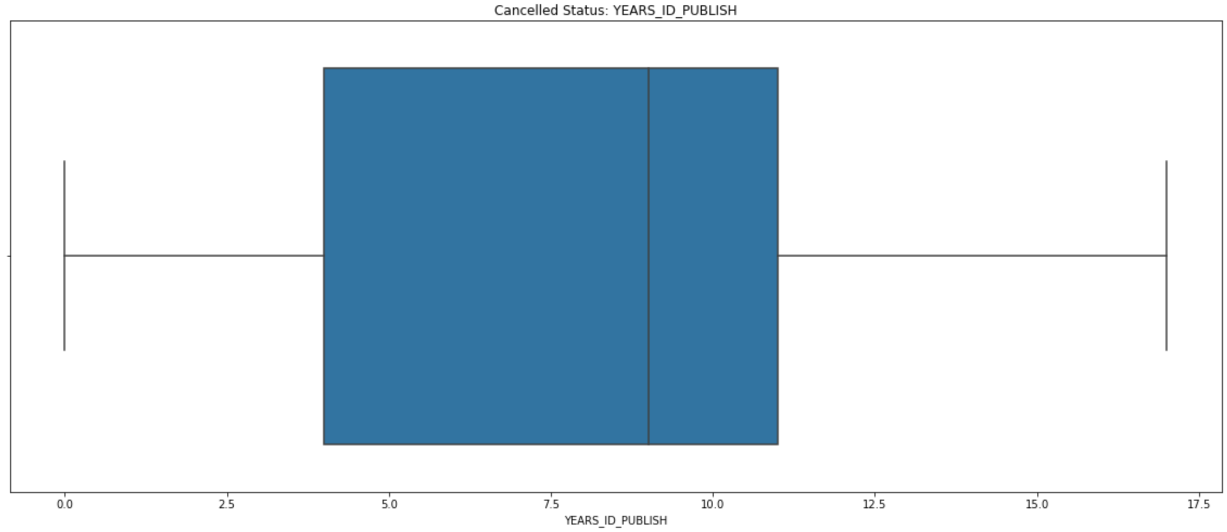
Univariate Analysis for few numerical columns in combined dataframe

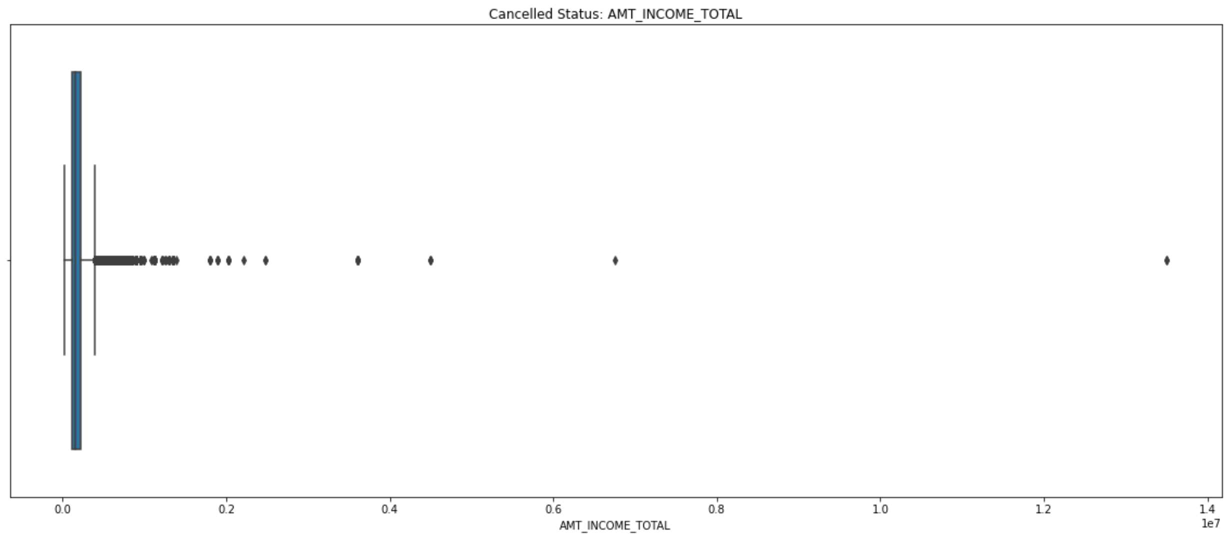
In [244...

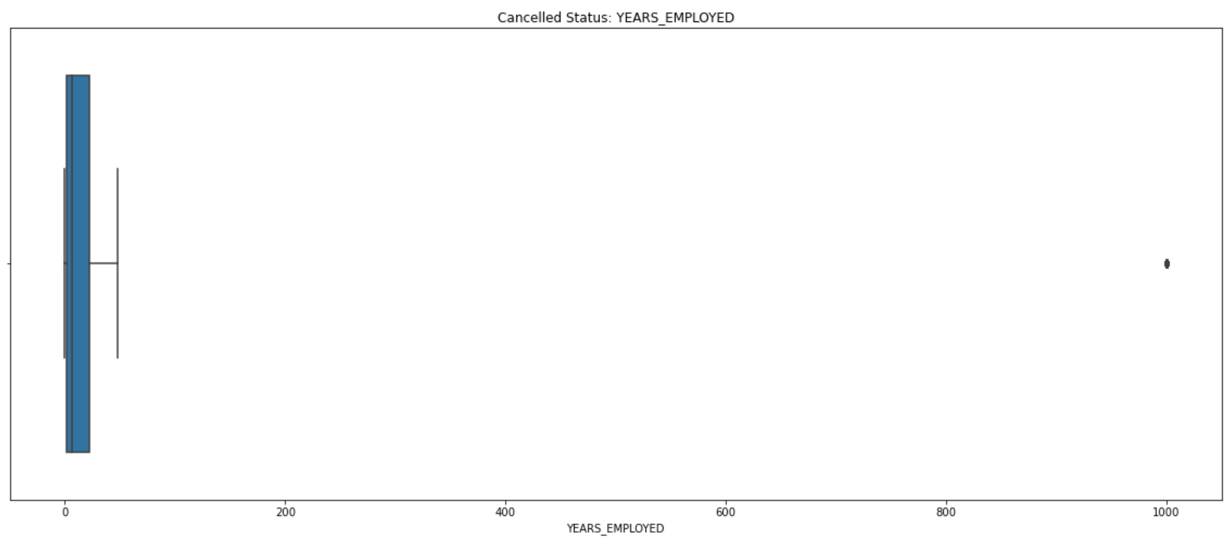
```
for i in combined_numerical_columns:
    plt.figure(figsize=(20,8))
    sns.boxplot(combined_cancelled_df[i])
    plt.title("Cancelled Status: "+i)
```

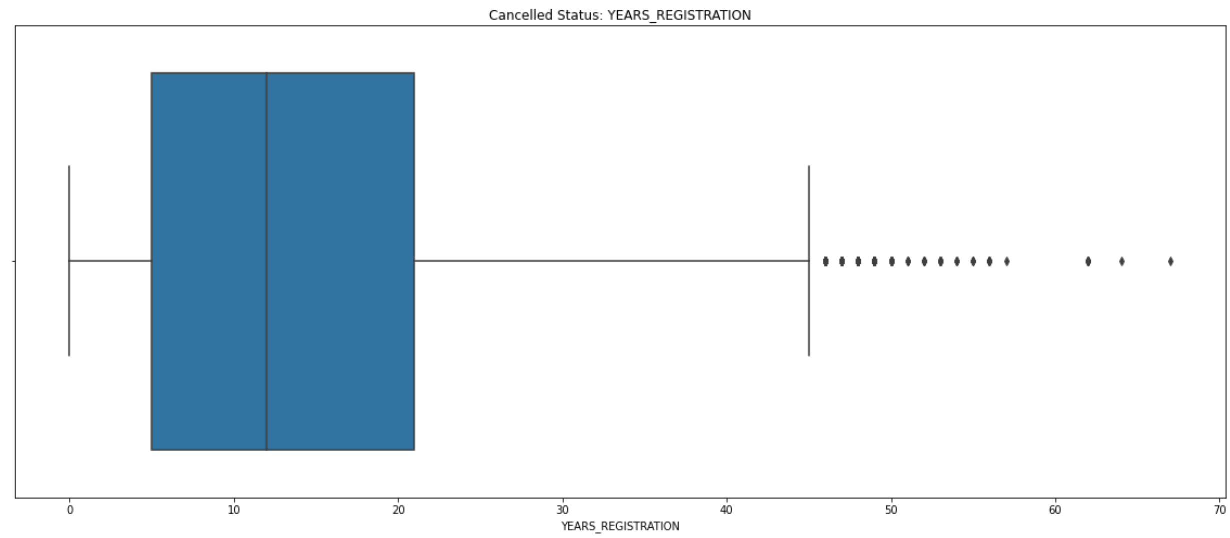


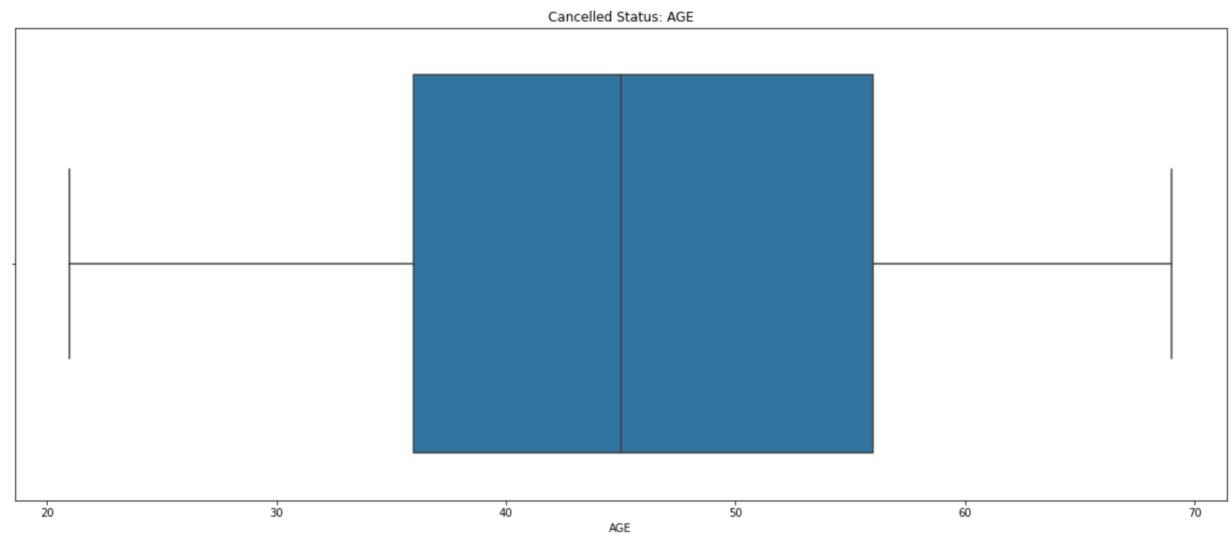


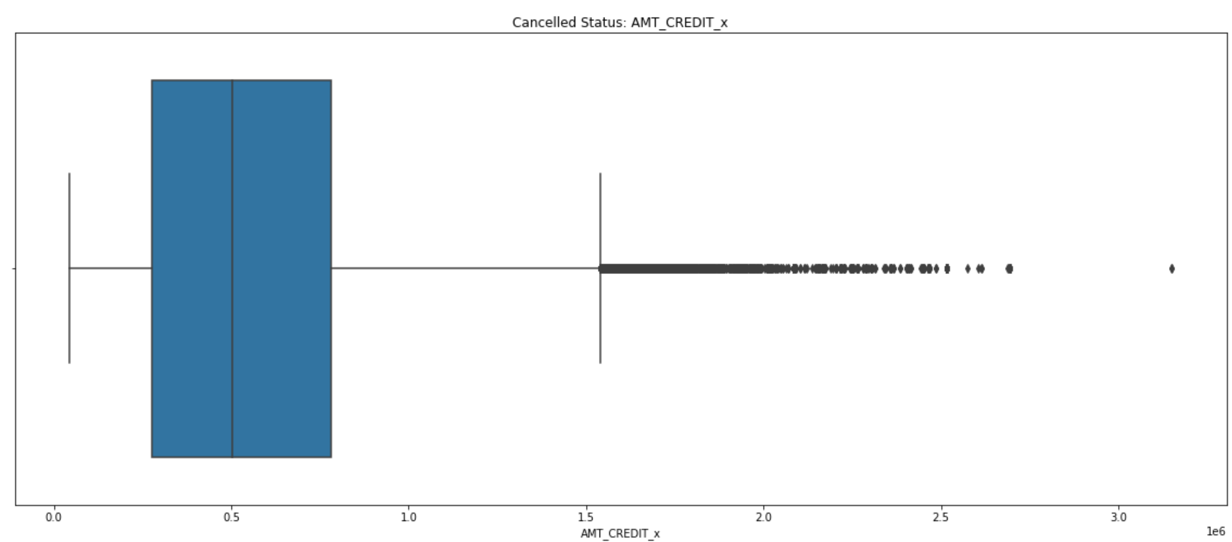

Cancelled Status: YEARS_ID_PUBLISH
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
YEARS_ID_PUBLISH


Cancelled Status: AMT_INCOME_TOTAL
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e7
AMT_INCOME_TOTAL


Cancelled Status: YEARS_EMPLOYED
0
200
400
600
800
1000
YEARS_EMPLOYED





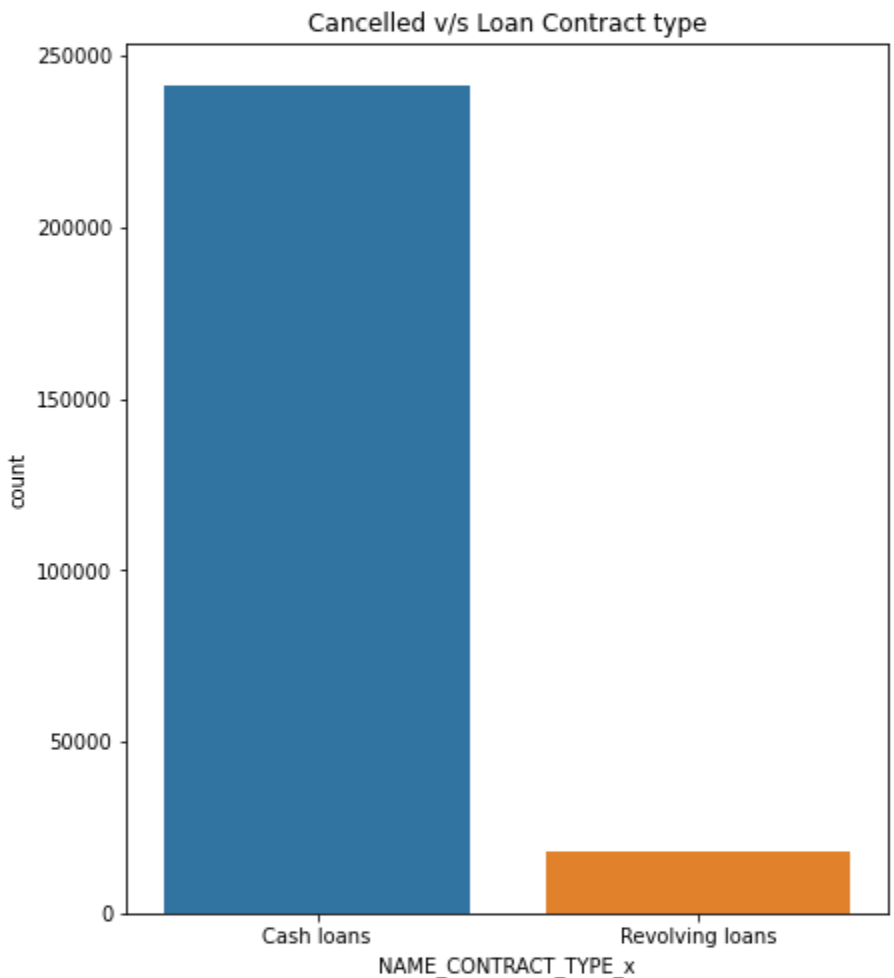


Cancelled Status: AMT_ANNUITY_x

Bivariant analysis for Categorical Data for the Canceled applications

In [246…

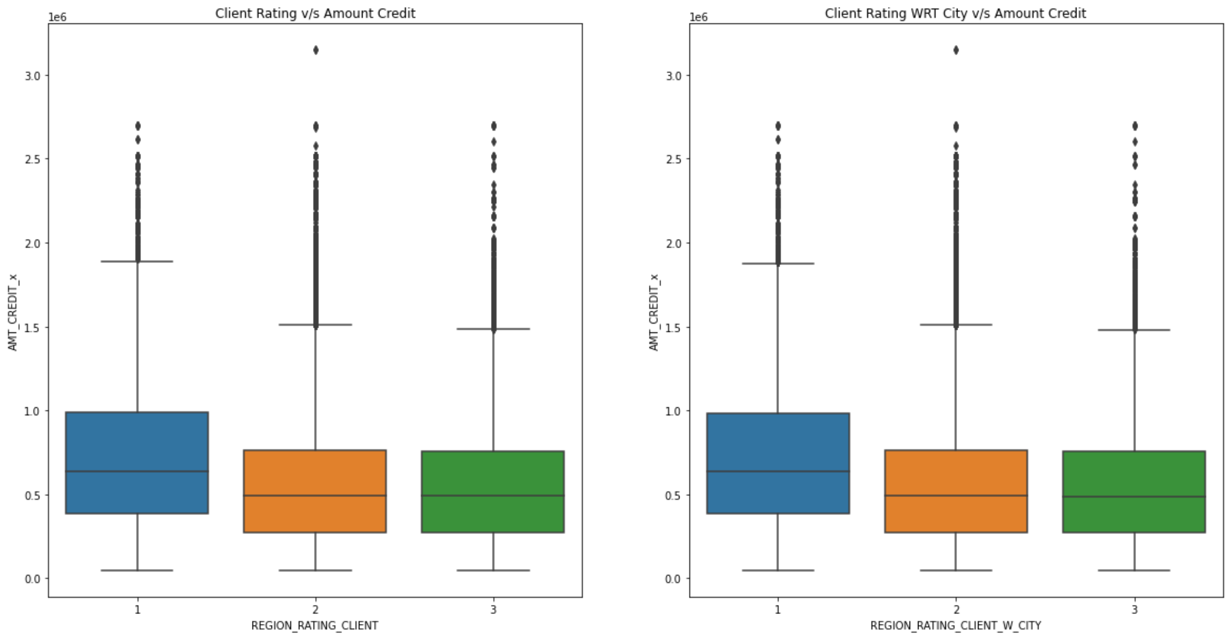
```
plt.figure(figsize=(15,8))
plt.subplot(1,2,1)
sns.countplot(x = "NAME_CONTRACT_TYPE_x", data = combined_cancelled_df)
plt.title("Cancelled v/s Loan Contract type")
plt.show()
```



Bivariant analysis for Numerical Data for the Cancelled applications

In [247…

```
# First Variable: Client Rating
# Second Variable: AMT_CREDIT
plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
sns.boxplot(x='REGION_RATING_CLIENT', y='AMT_CREDIT_x', data = combined_cancelled_df
plt.title("Client Rating v/s Amount Credit")

# First Variable: Client Rating With Respect to City
# Second Variable: AMT_CREDIT
plt.subplot(1,2,2)
sns.boxplot(x='REGION_RATING_CLIENT_W_CITY',y='AMT_CREDIT_x', data = combined_cancel
plt.title("Client Rating WRT City v/s Amount Credit")
plt.show()
```



Doing analysis on People with Contract Status as Un-used offer

Preparing data for people with Unused offer status

In [248...

```
combined_unused_df = combined_df[combined_df.NAME_CONTRACT_STATUS == 'Unused offer']
print(combined_unused_df.shape)
combined_unused_df
```

(22771, 71)

Out[248...

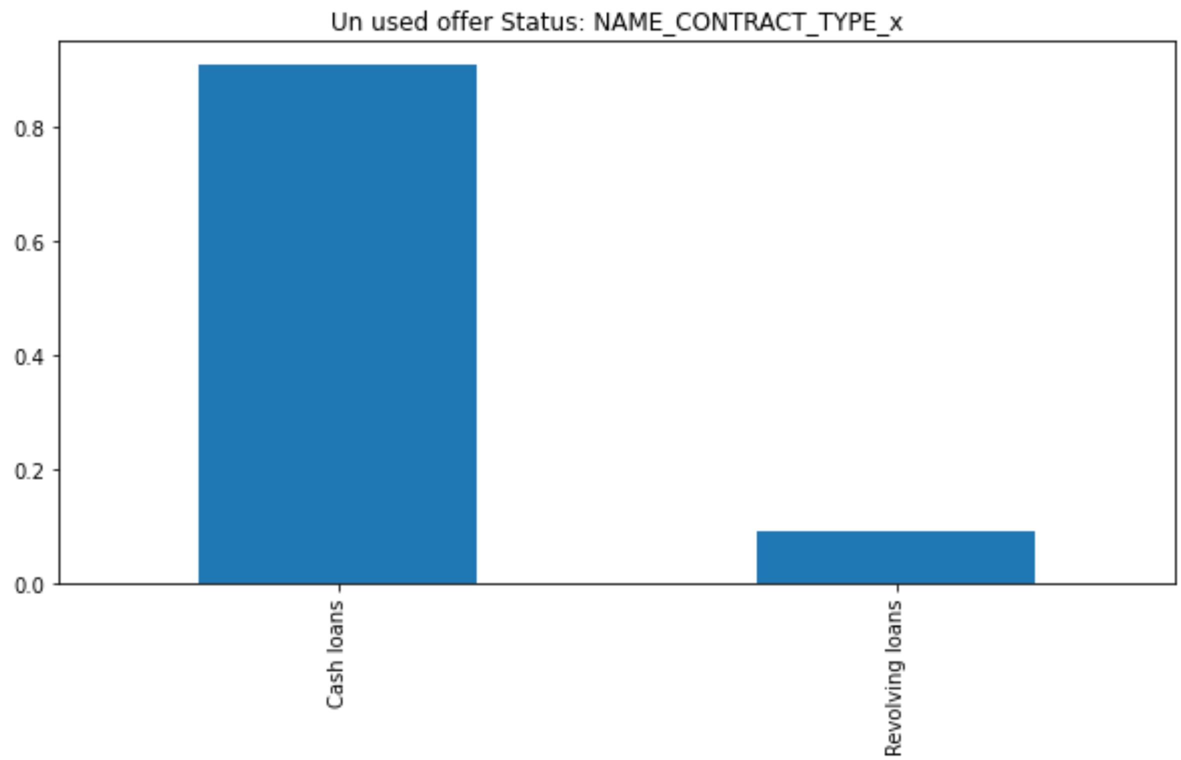
	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG_OV
222	100061	0	Cash loans	F	N	
358	100086	0	Cash loans	F	N	
383	100093	0	Cash loans	F	N	
463	100116	0	Cash loans	F	N	
465	100116	0	Cash loans	F	N	
...	...	...	...	...	...	
1413551	456220	0	Cash loans	F	N	
1413581	456230	0	Cash loans	F	Y	
1413582	456230	0	Cash loans	F	Y	
1413603	456234	0	Cash loans	M	N	
1413626	456238	0	Cash loans	M	Y	

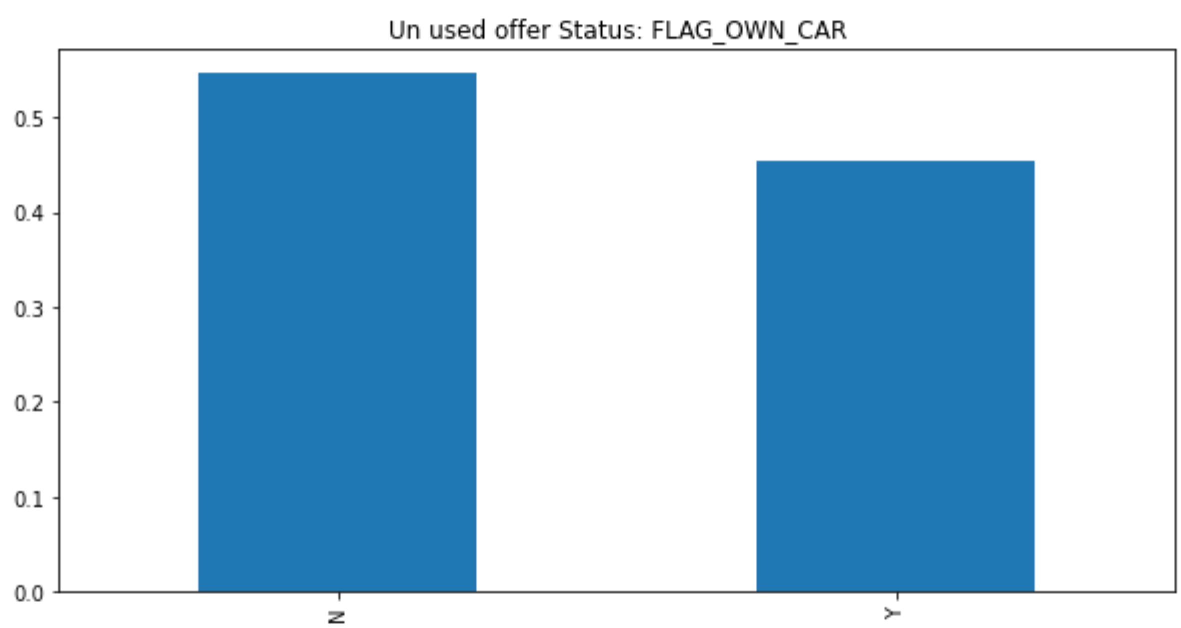
22771 rows × 71 columns

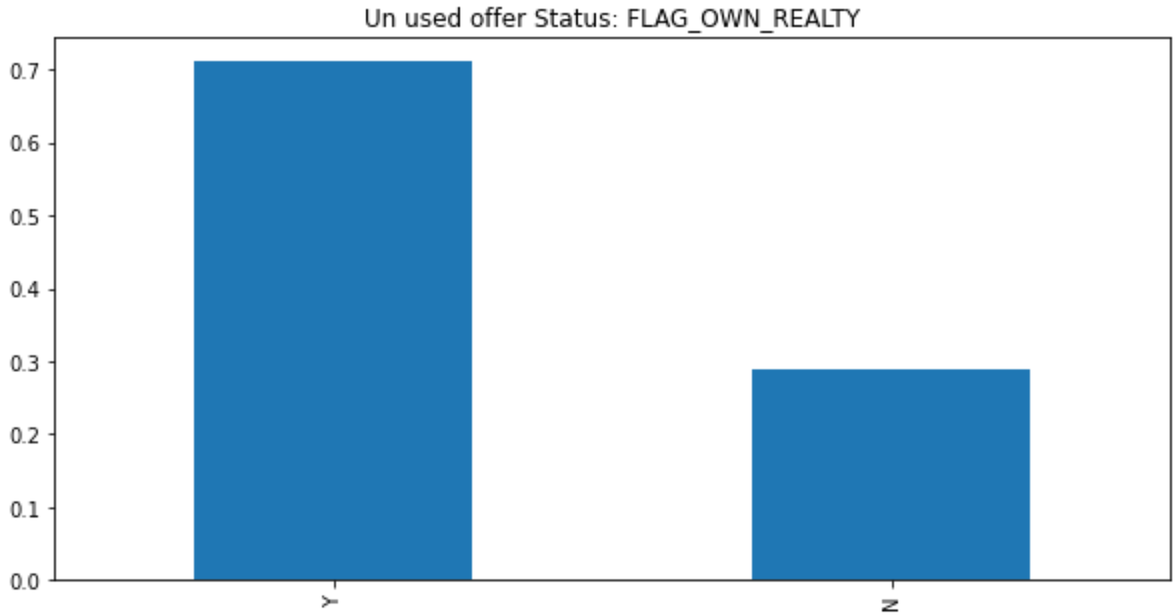
Univariate Analysis for few categorical columns in combined dataframe

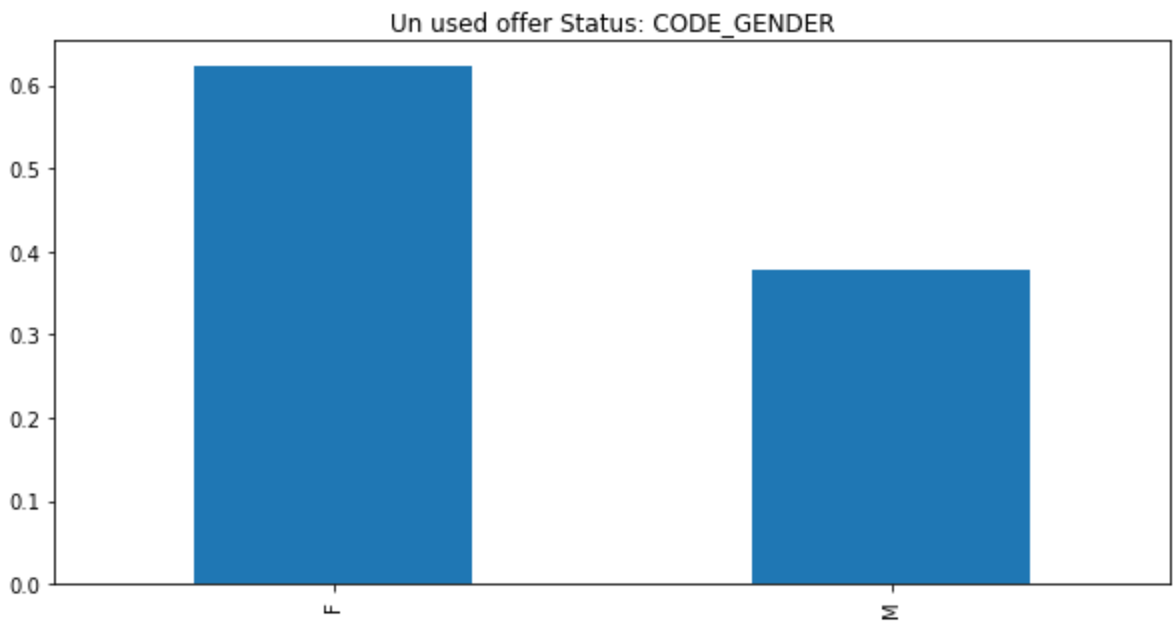
In [249…

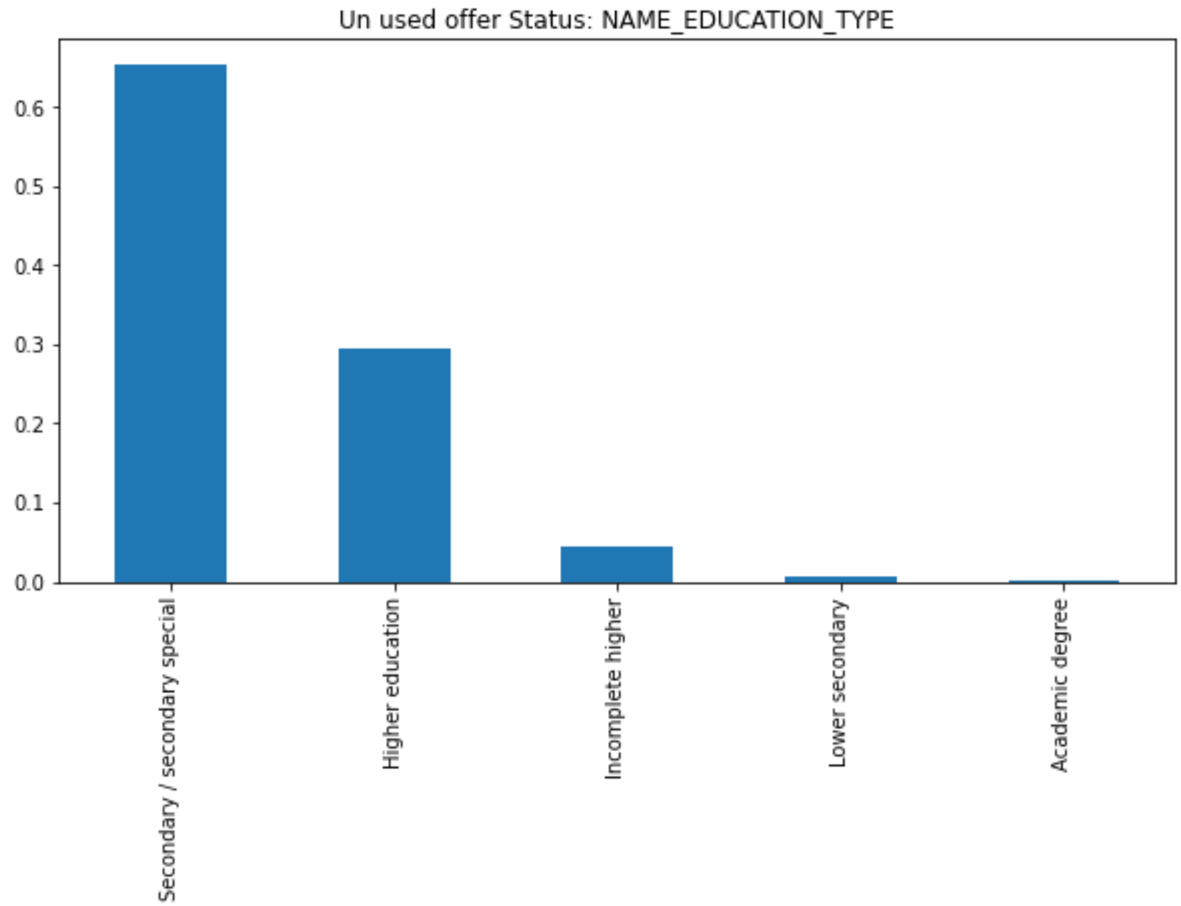
```
for i in combined_categorical_columns:
    plt.figure(figsize=(10,5))
    combined_unused_df[i].value_counts(normalize=True).plot.bar()
    plt.title("Un used offer Status: "+ i)
```

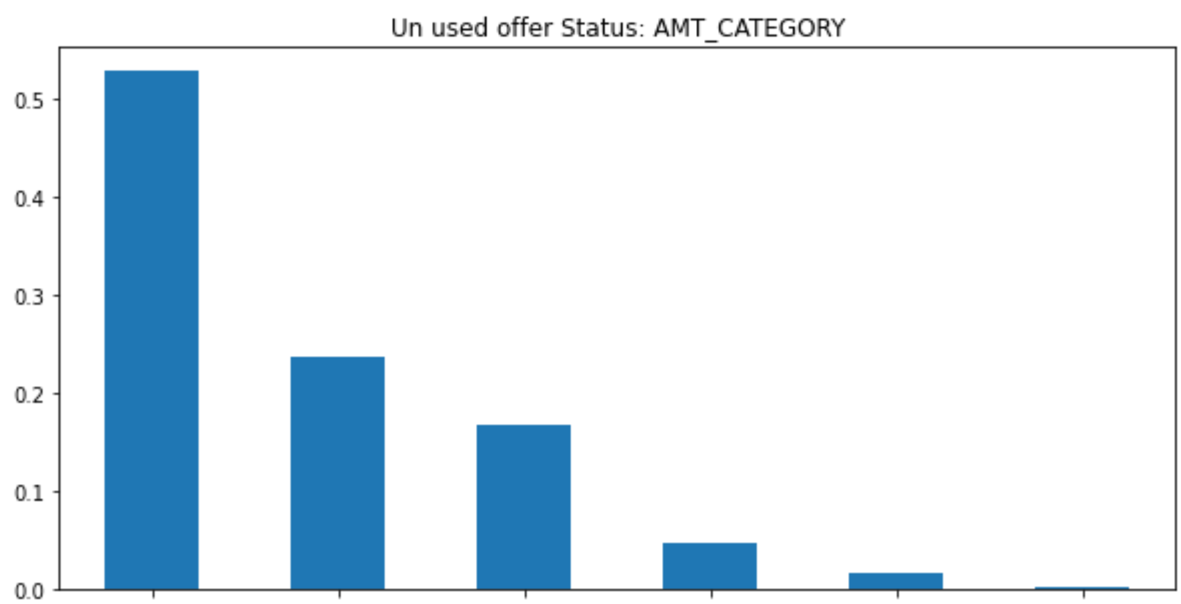





Un used offer Status: FLAG_OWN_REALTY
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Y
N


Un used offer Status: CODE_GENDER
0.0
0.1
0.2
0.3
0.4
0.5
0.6
F
M

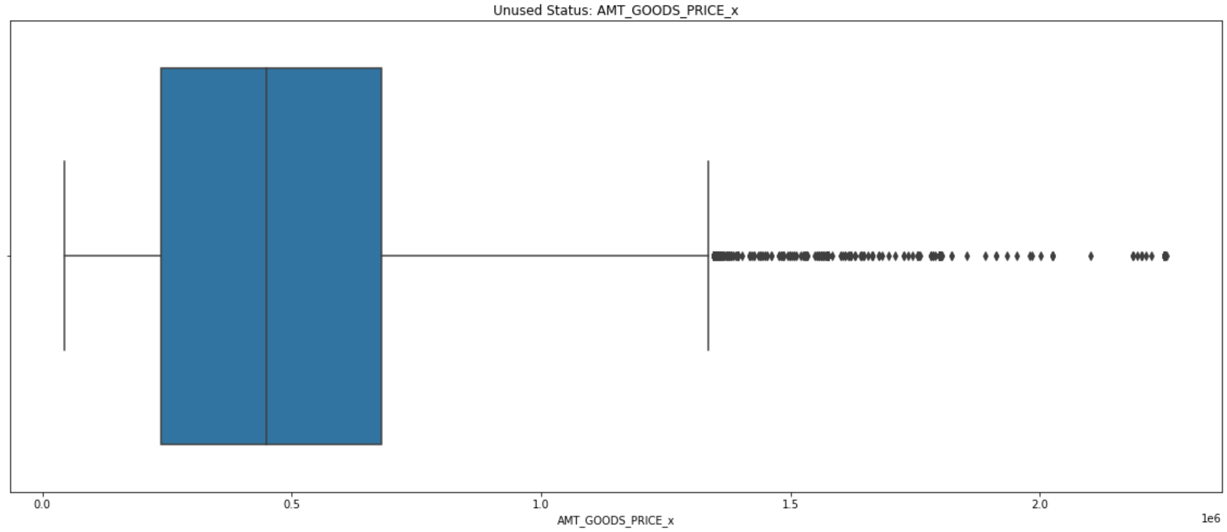


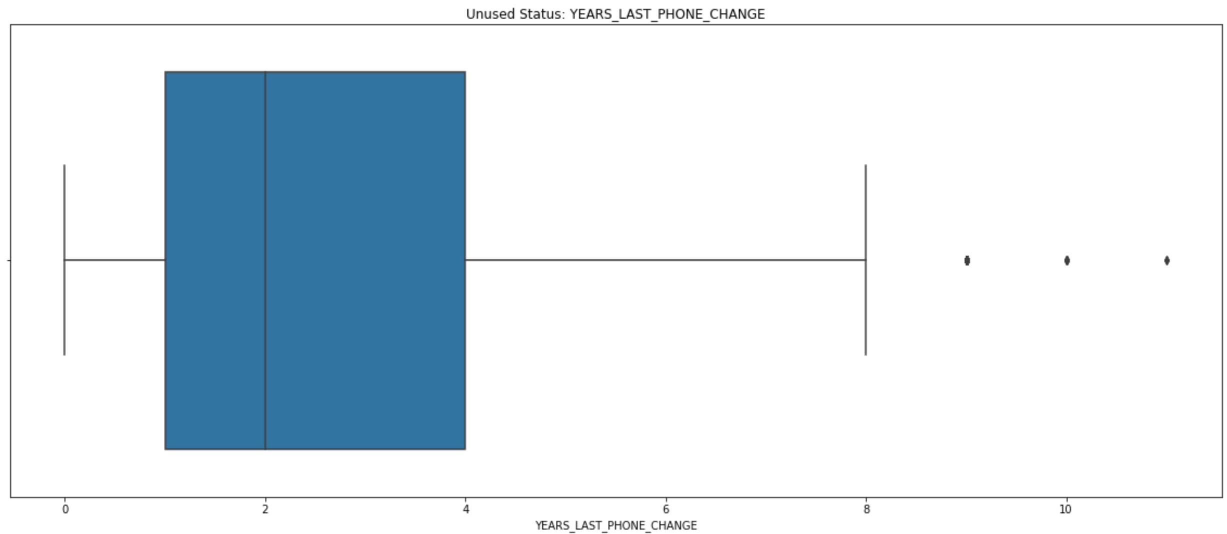


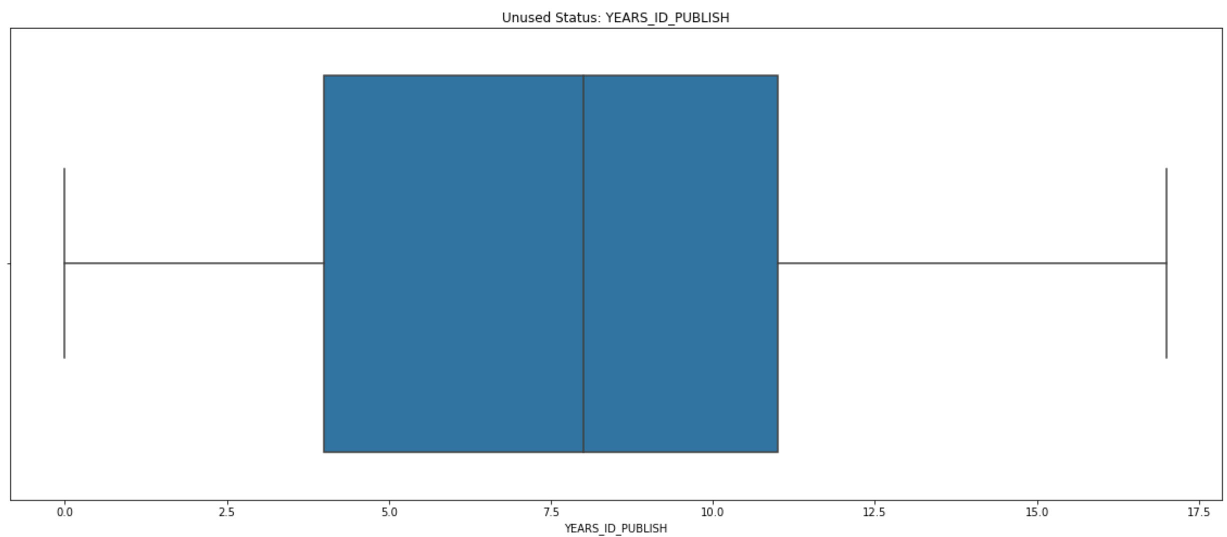
Univariate Analysis for few numerical columns in combined dataframe

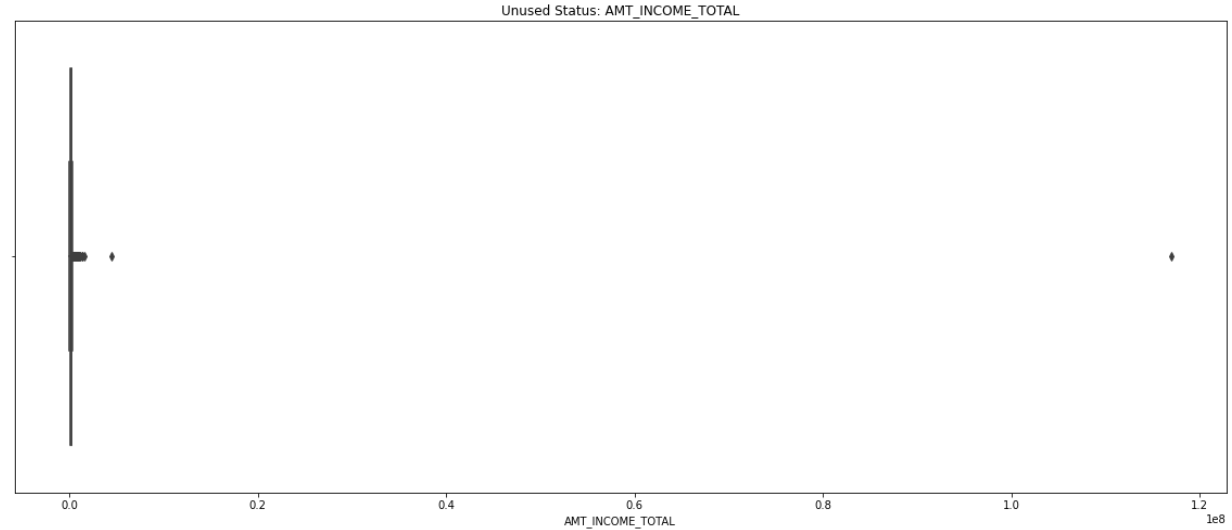
In [250…

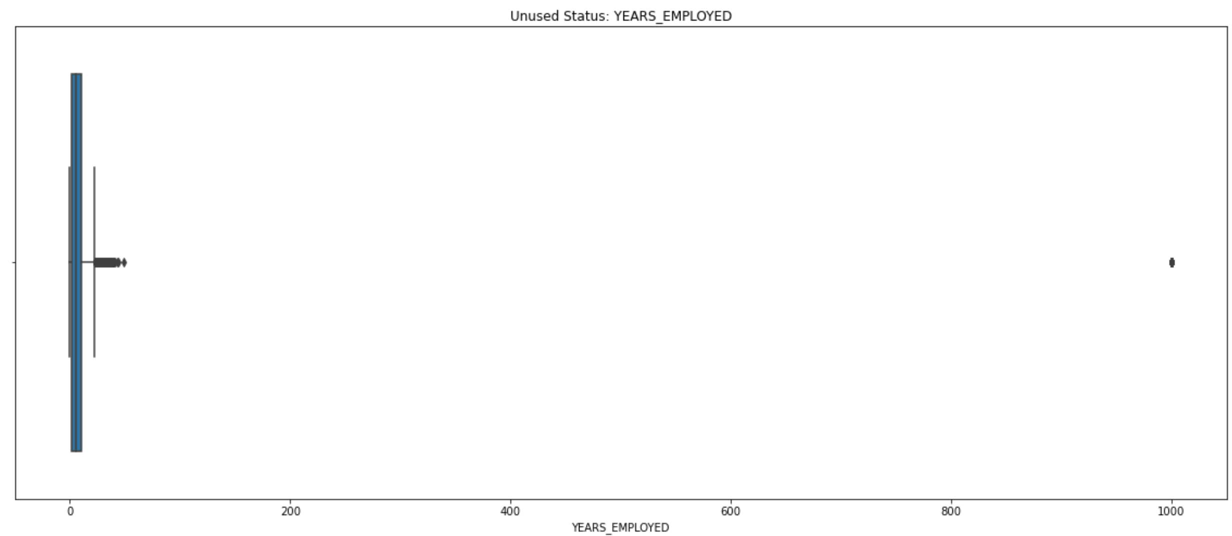
```
for i in combined_numerical_columns:
    plt.figure(figsize=(20,8))
    sns.boxplot(combined_unused_df[i])
    plt.title("Unused Status: "+i)
```

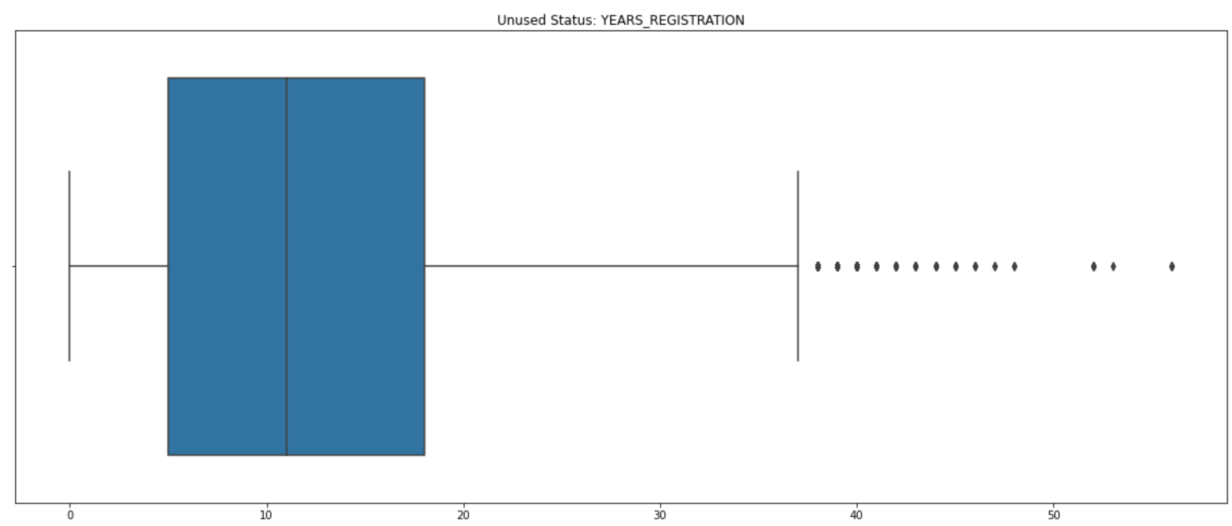

Unused Status: AMT_GOODS_PRICE_x
0.0
0.5
1.0
1.5
2.0
1e6
AMT_GOODS_PRICE_x

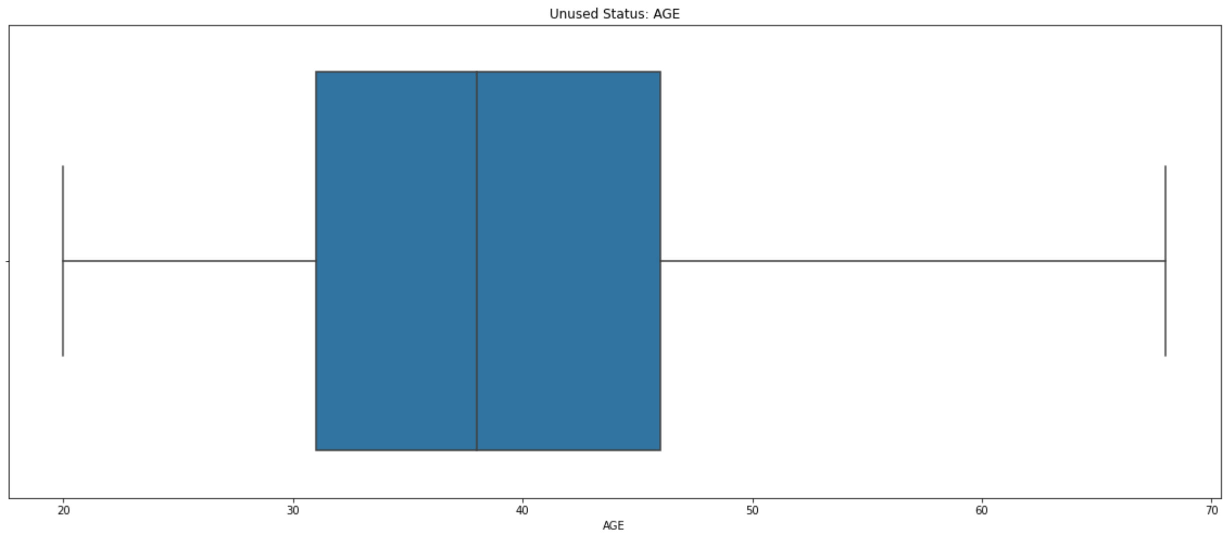

Unused Status: YEARS_LAST_PHONE_CHANGE
0
2
4
6
8
10
YEARS_LAST_PHONE_CHANGE

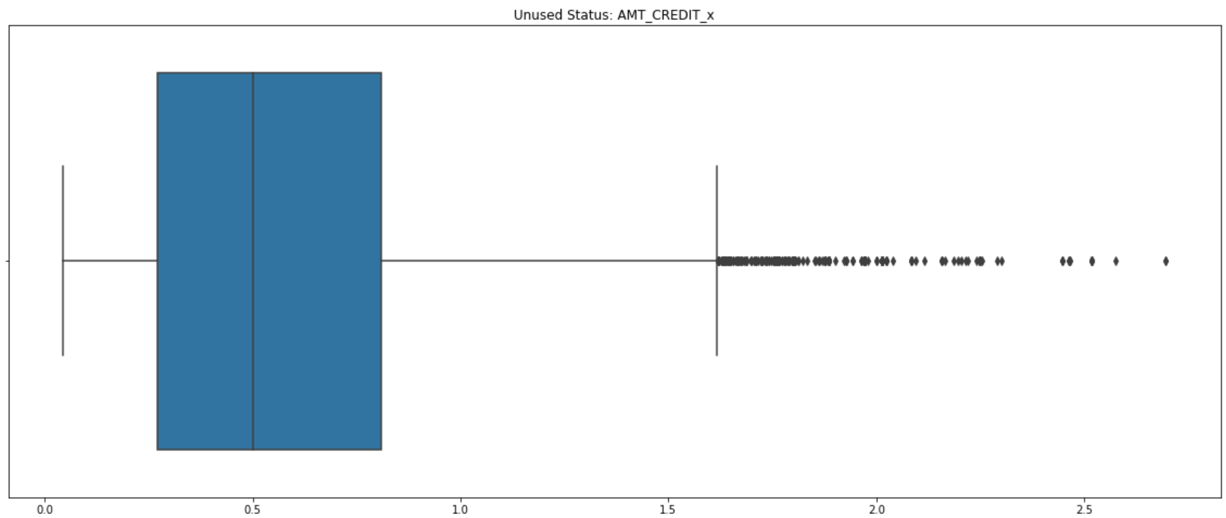

Unused Status: YEARS_ID_PUBLISH
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
YEARS_ID_PUBLISH


Unused Status: AMT_INCOME_TOTAL
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e8
AMT_INCOME_TOTAL


Unused Status: YEARS_EMPLOYED
0
200
400
600
800
1000
YEARS_EMPLOYED


Unused Status: YEARS_REGISTRATION
0
10
20
30
40
50
YEARS_REGISTRATION

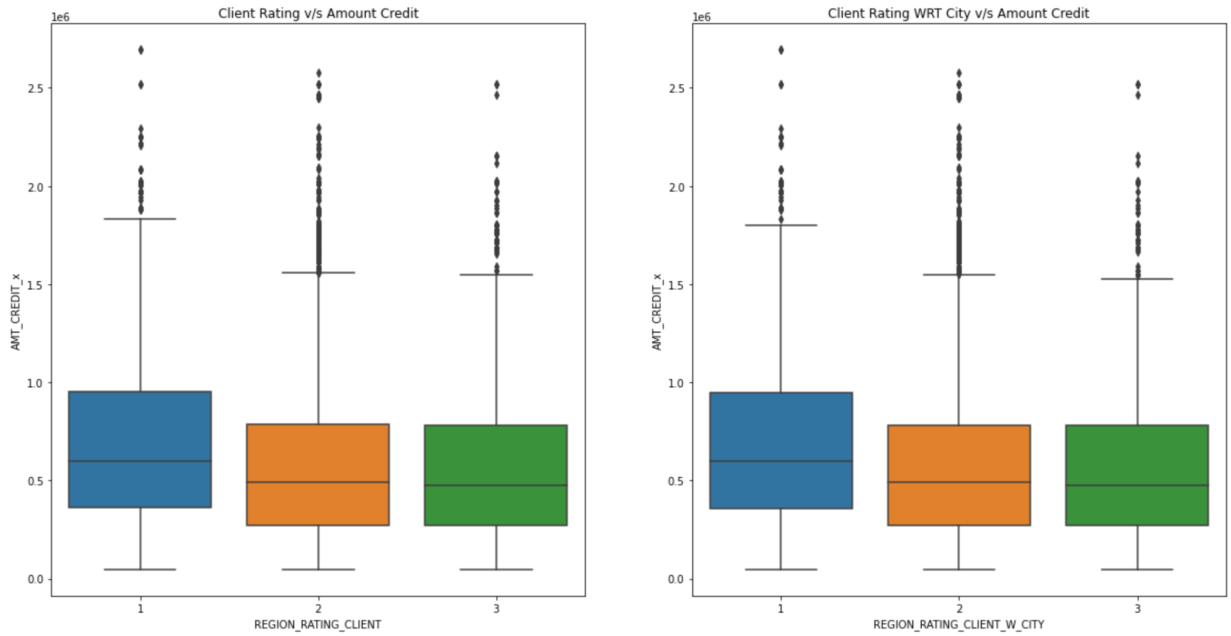




Bivariant analysis for Numerical Data for the Unused applications

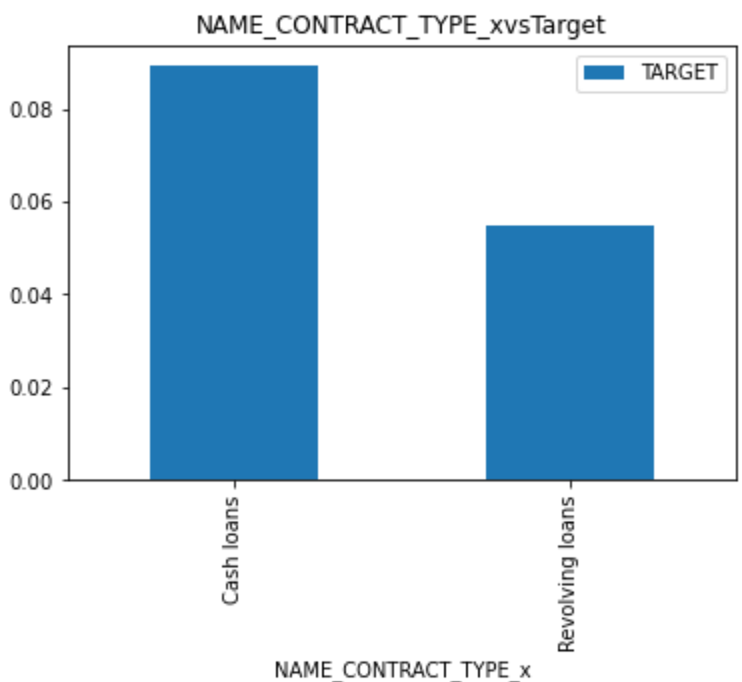
In [251…

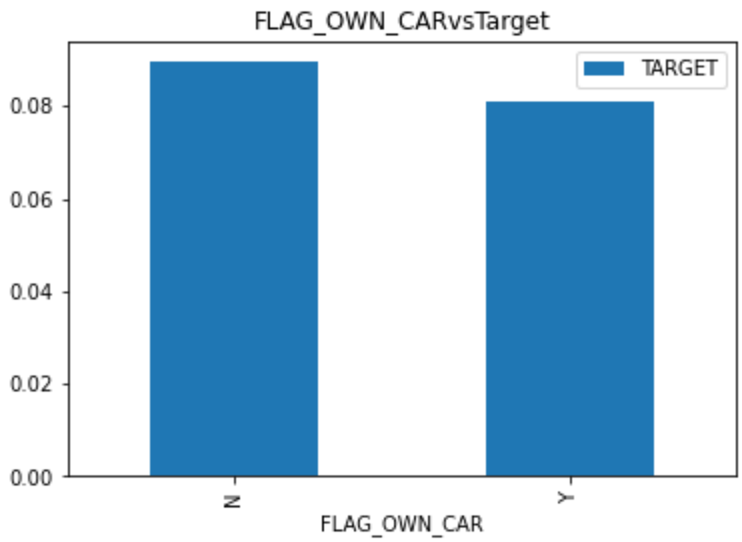
```
plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
sns.boxplot(x='REGION_RATING_CLIENT', y='AMT_CREDIT_x', data = combined_unused_df)
plt.title("Client Rating v/s Amount Credit")

# First Variable: Client Rating With Respect to City
# Second Variable: AMT_CREDIT
plt.subplot(1,2,2)
sns.boxplot(x='REGION_RATING_CLIENT_W_CITY',y='AMT_CREDIT_x', data = combined_unused_
plt.title("Client Rating WRT City v/s Amount Credit")
plt.show()
```

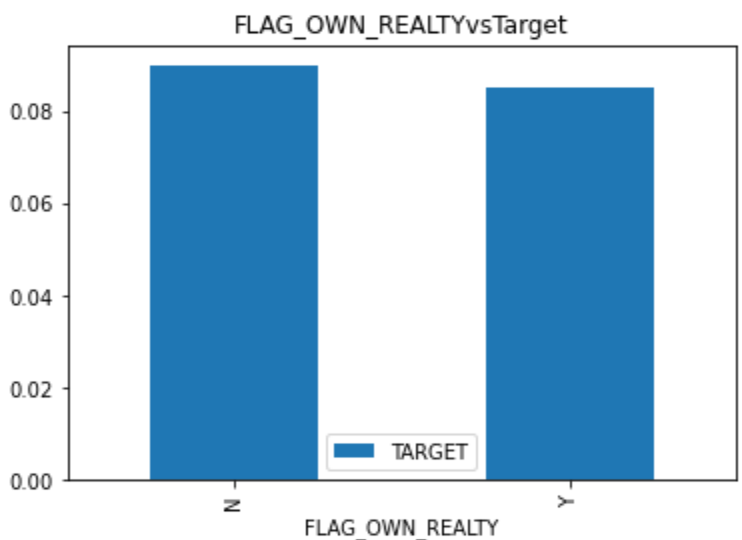


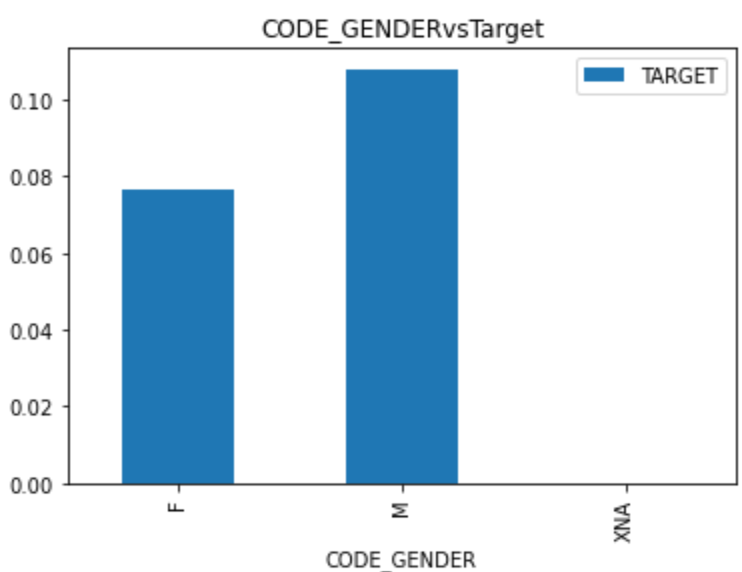
In [252…

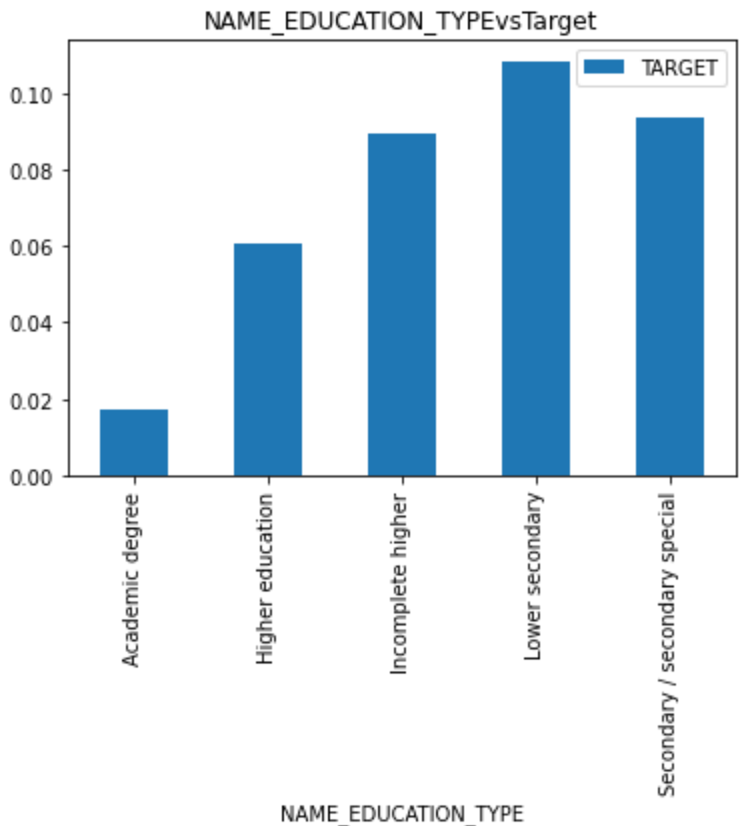
```
for i in combined_categorical_columns:
    (combined_df.groupby(i)['TARGET', 'NAME_CONTRACT_STATUS'].mean()).plot.bar()
    plt.title(i+ 'vs' +'Target')
    plt.show()
```

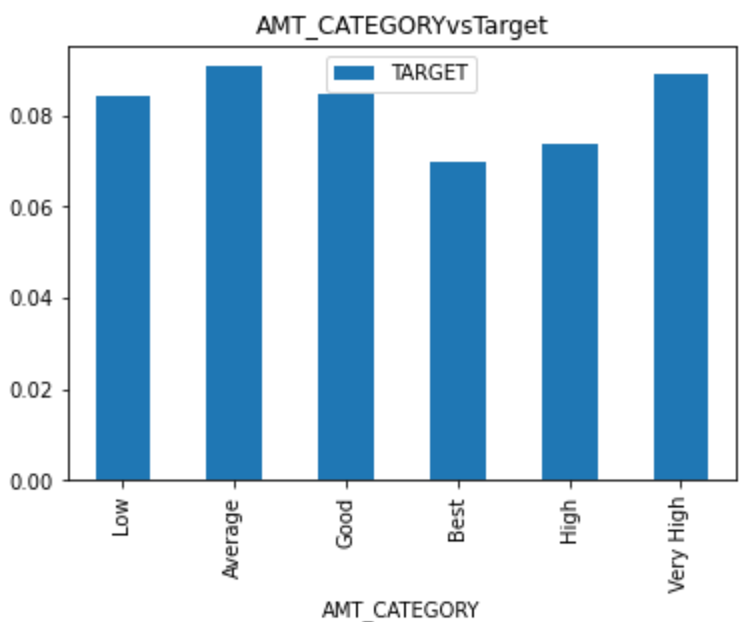



FLAG_OWN_CARvsTarget
TARGET
0.08
0.06
0.04
0.02
0.00
N
Y
FLAG_OWN_CAR


FLAG_OWN_REALTYvsTarget
TARGET
0.08
0.06
0.04
0.02
0.00
N
Y
FLAG_OWN_REALTY


CODE_GENDERvsTarget
TARGET
0.10
0.08
0.06
0.04
0.02
0.00
F
M
XNA
CODE_GENDER





Analysis to understand the relation between NAME_CONTRACT_STATUS AND TARGET

In [273…

```
combined_df.groupby('NAME_CONTRACT_STATUS')['TARGET'].mean()
```

Out[273…

```
NAME_CONTRACT_STATUS
Approved        0.075887
Canceled        0.091736
Refused         0.119964
Unused offer    0.082517
```

In [ ]:

```

```