1. Import Python libraries:

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.style as style
import seaborn as sns
import itertools
%matplotlib inline

#setting up plot style
style.use('seaborn-poster')
style.use('fivethirtyeight')
```

2.Supressing Warnings:

In [2]:

```
import warnings
warnings.filterwarnings('ignore')
```

3. Adjust Jupyter Views:

In [3]:

```
pd.set_option('display.max_rows',500)
pd.set_option('display.max_columns',500)
pd.set_option('display.width',1000)
pd.set_option('display.expand_frame_repr', False)
```

4. Reading and Understanding the data

In [4]:

```
application_d = pd.read_csv(r'V:\Senapati Sir file\9. 11 oct\Resume projects\application
previous_d = pd.read_csv(r'V:\Senapati Sir file\10. 12 oct\11th\Resume projects\previous
```

In [5]:

```
previous_d
```

Out[5]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLIC
0	2030495	271877	Consumer loans	1730.430	17
1	2802425	108129	Cash loans	25188.615	607
2	2523466	122040	Cash loans	15060.735	112
3	2819243	176158	Cash loans	47041.335	450
4	1784265	202054	Cash loans	31924.395	337
...	...	...	...	...	
1670209	2300464	352015	Consumer loans	14704.290	267
1670210	2357031	334635	Consumer loans	6622.020	87
1670211	2659632	249544	Consumer loans	11520.855	105
1670212	2785582	400317	Cash loans	18821.520	180
1670213	2418762	261212	Cash loans	16431.300	360

1670214 rows × 37 columns

In [6]:

```
application_d
```

Out[6]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR
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

In [7]:

```
application_d.head()
```

Out[7]:

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

In [8]:

```
previous_d.head()
```

Out[8]:

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

In [9]:

```
# Database dimension
print("Database dimension - application_d : ", application_d.shape)
print("Database dimension - previous_d :", previous_d.shape)

#Database size
print("Database size - application_d  :",application_d.size)
print("Database size - previousDF_d   :",previous_d.size)
```

```
Database dimension - application_d :  (307511, 122)
Database dimension - previous_d : (1670214, 37)
Database size - application_d  : 37516342
Database size - previousDF_d   : 61797918
```

In [10]:

```
# Database column types
application_d.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 122 columns):
 #    Column                       Dtype
---   ------                       -----
 0    SK_ID_CURR                   int64
 1    TARGET                       int64
 2    NAME_CONTRACT_TYPE           object
 3    CODE_GENDER                  object
 4    FLAG_OWN_CAR                 object
 5    FLAG_OWN_REALTY              object
 6    CNT_CHILDREN                 int64
 7    AMT_INCOME_TOTAL             float64
 8    AMT_CREDIT                   float64
 9    AMT_ANNUITY                  float64
 10   AMT_GOODS_PRICE              float64
 11   NAME_TYPE_SUITE              object
 12   NAME_INCOME_TYPE             object
 13   NAME_EDUCATION_TYPE          object
```

In [11]:

```
previous_d.info(verbose=True)
```

```
<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
```

In [12]:

```
# Checking the numeric variables of the dataframes
application_d.describe()
```

Out[12]:

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

In [13]:

```
previous_d.describe()
```

Out[13]:

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

4. Data Cleaning & Manipulation

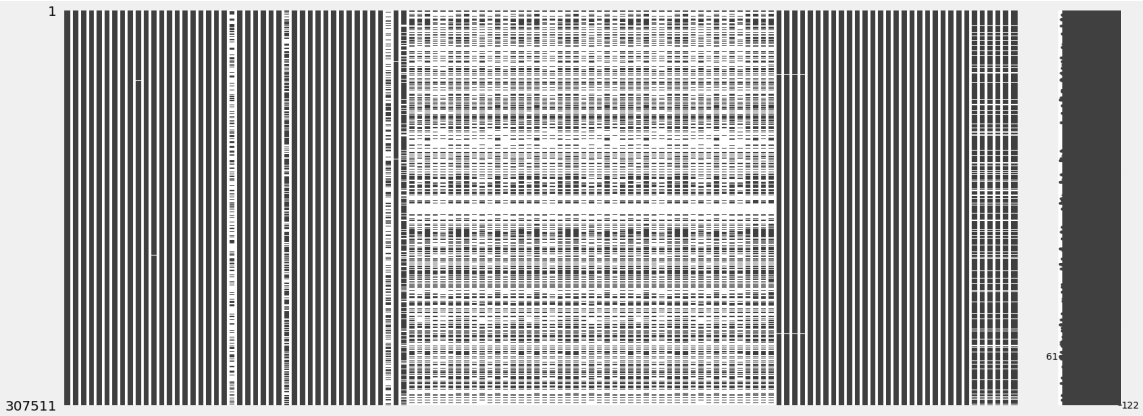
Null Value Calculation

In [14]:

```
import missingno as mn
mn.matrix(application_d)
```

Out[14]:

```
<AxesSubplot:>
```



In [15]:

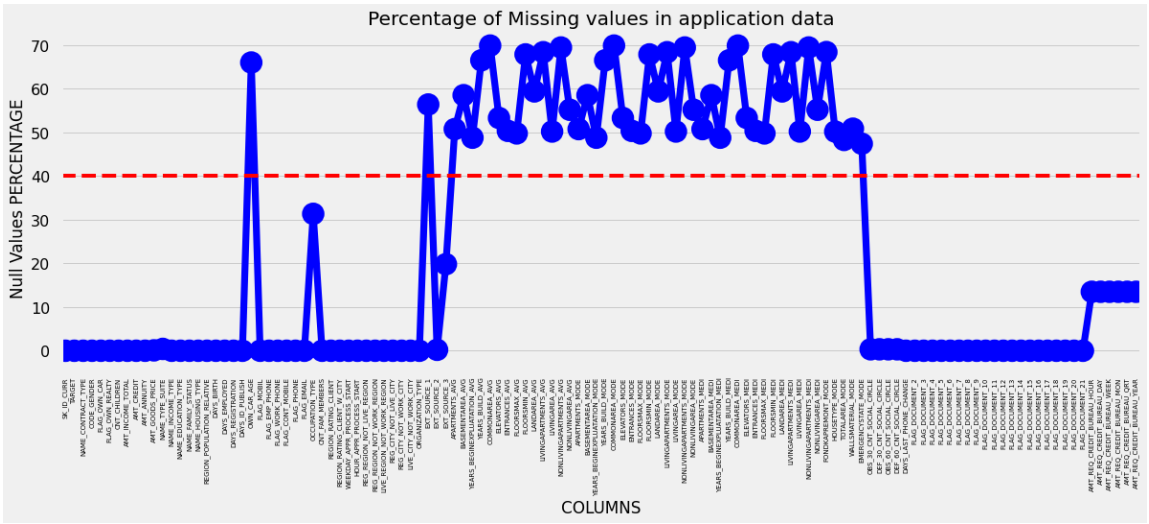
```
# % null value in each column
round(application_d.isnull().sum() / application_d.shape[0] * 100.00,2)
```

Out[15]:

```
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
```

In [16]:

```
null_application_d = pd.DataFrame((application_d.isnull().sum())*100/application_d.shape
null_application_d.columns = ['Column Name', 'Null Values Percentage']
fig = plt.figure(figsize=(18,6))
ax = sns.pointplot(x="Column Name",y="Null Values Percentage",data=null_application_d,co
plt.xticks(rotation =90,fontsize =7)
ax.axhline(40, ls='--',color='red')
plt.title("Percentage of Missing values in application data")
plt.ylabel("Null Values PERCENTAGE")
plt.xlabel("COLUMNS")
plt.show()
```



In [17]:

```
nullcol_40_application = null_application_d[null_application_d['Null Values Percentage']
nullcol_40_application    # here are more than 40% null values
```

Out[17]:

	Column Name	Null Values Percentage
21	OWN_CAR_AGE	65.990810
41	EXT_SOURCE_1	56.381073
44	APARTMENTS_AVG	50.749729
45	BASEMENTAREA_AVG	58.515956
46	YEARS_BEGINEXPLUATATION_AVG	48.781019
47	YEARS_BUILD_AVG	66.497784
48	COMMONAREA_AVG	69.872297
49	ELEVATORS_AVG	53.295980
50	ENTRANCES_AVG	50.348768
51	FLOORSMAX_AVG	49.760822
52	FLOORSMIN_AVG	67.848630
53	LANDAREA_AVG	59.376738
54	LIVINGAPARTMENTS_AVG	68.354953
55	LIVINGAREA_AVG	50.193326
56	NONLIVINGAPARTMENTS_AVG	69.432963
57	NONLIVINGAREA_AVG	55.179164
58	APARTMENTS_MODE	50.749729
59	BASEMENTAREA_MODE	58.515956
60	YEARS_BEGINEXPLUATATION_MODE	48.781019
61	YEARS_BUILD_MODE	66.497784
62	COMMONAREA_MODE	69.872297
63	ELEVATORS_MODE	53.295980
64	ENTRANCES_MODE	50.348768
65	FLOORSMAX_MODE	49.760822
66	FLOORSMIN_MODE	67.848630
67	LANDAREA_MODE	59.376738
68	LIVINGAPARTMENTS_MODE	68.354953
69	LIVINGAREA_MODE	50.193326
70	NONLIVINGAPARTMENTS_MODE	69.432963
71	NONLIVINGAREA_MODE	55.179164
72	APARTMENTS_MEDI	50.749729
73	BASEMENTAREA_MEDI	58.515956
74	YEARS_BEGINEXPLUATATION_MEDI	48.781019
75	YEARS_BUILD_MEDI	66.497784
76	COMMONAREA_MEDI	69.872297
77	ELEVATORS_MEDI	53.295980
78	ENTRANCES_MEDI	50.348768

	Column Name	Null Values Percentage
79	FLOORSMAX_MEDI	49.760822
80	FLOORSMIN_MEDI	67.848630
81	LANDAREA_MEDI	59.376738
82	LIVINGAPARTMENTS_MEDI	68.354953
83	LIVINGAREA_MEDI	50.193326
84	NONLIVINGAPARTMENTS_MEDI	69.432963
85	NONLIVINGAREA_MEDI	55.179164
86	FONDKAPREMONT_MODE	68.386172
87	HOUSETYPE_MODE	50.176091
88	TOTALAREA_MODE	48.268517
89	WALLSMATERIAL_MODE	50.840783
90	EMERGENCYSTATE_MODE	47.398304

In [18]:

```
len(nullcol_40_application)
```

Out[18]:

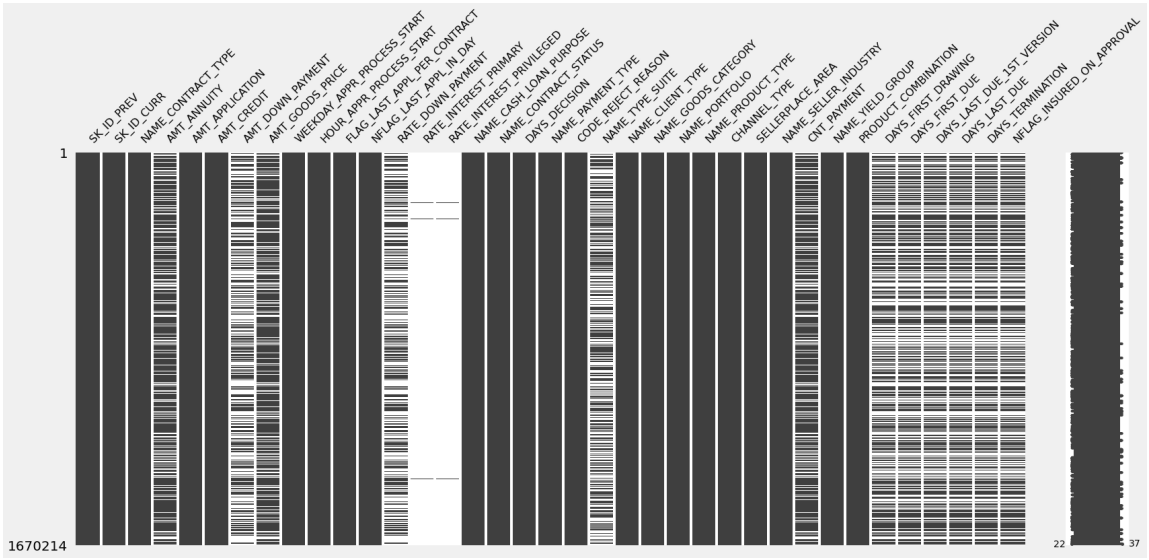
```
49
```

In [19]:

```
mn.matrix(previous_d)
```

Out[19]:

```
<AxesSubplot:>
```



In [20]:

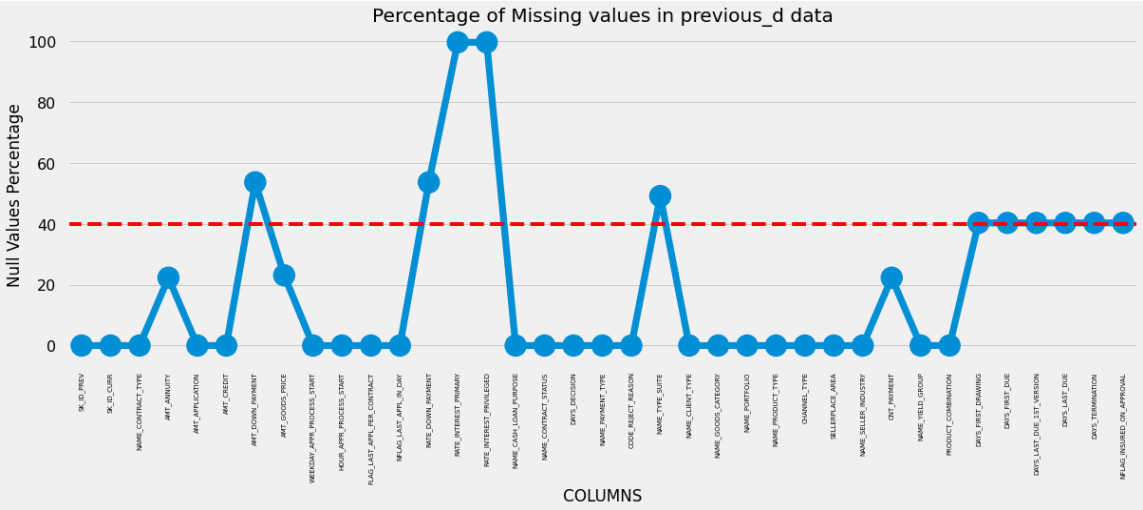
```
round(previous_d.isnull().sum() / previous_d.shape[0] * 100.00,2)
```

Out[20]:

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

In [21]:

```
null_previous_d = pd.DataFrame((previous_d.isnull().sum())*100/previous_d.shape[0]).rese
null_previous_d.columns = ['Column Name','Null Values Percentage']
fig = plt.figure(figsize=(18,6)),
ax = sns.pointplot(x="Column Name", y="Null Values Percentage", data=null_previous_d,col
plt.xticks(rotation=90, fontsize= 7)
ax.axhline(40,ls='--',color= 'red')
plt.title("Percentage of Missing values in previous_d data")
plt.ylabel("Null Values Percentage")
plt.xlabel("COLUMNS")
plt.show()
```



In [22]:

```
nullcol_40_previous = null_previous_d[null_previous_d["Null Values Percentage"]>=40]
nullcol_40_previous
```

Out[22]:

	Column Name	Null Values Percentage
6	AMT_DOWN_PAYMENT	53.636480
12	RATE_DOWN_PAYMENT	53.636480
13	RATE_INTEREST_PRIMARY	99.643698
14	RATE_INTEREST_PRIVILEGED	99.643698
20	NAME_TYPE_SUITE	49.119754
31	DAYS_FIRST_DRAWING	40.298129
32	DAYS_FIRST_DUE	40.298129
33	DAYS_LAST_DUE_1ST_VERSION	40.298129
34	DAYS_LAST_DUE	40.298129
35	DAYS_TERMINATION	40.298129
36	NFLAG_INSURED_ON_APPROVAL	40.298129

In [23]:

```
len(nullcol_40_previous)
```

Out[23]:

```
11
```

Analyze & Delete Unnecessary Columns in application_d

In [24]:

```
# Checking correlation of EXT_SOURCE_X columns vs target column (dependent)
Source = application_d[["EXT_SOURCE_1","EXT_SOURCE_2","EXT_SOURCE_3","TARGET"]]
source_corr = Source.corr()
ax = sns.heatmap(source_corr,
            xticklabels=source_corr.columns,
            yticklabels=source_corr.columns,
            annot = True,
            cmap ="RdYlGn")
```

In [25]:

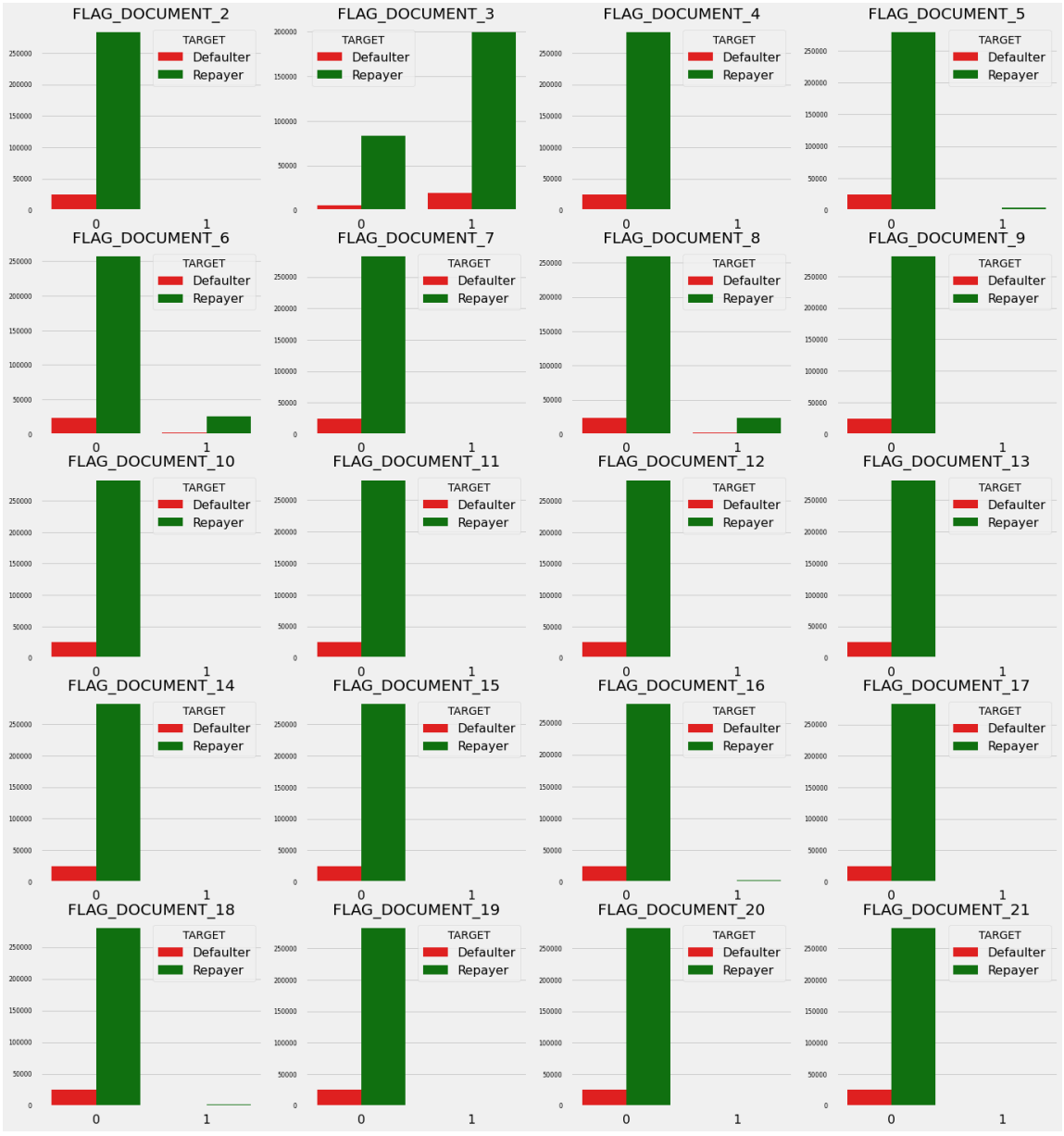
```
#list of columns that needs to be dropped including the columns with >40% null values
Unwanted_application = nullcol_40_application["Column Name"].tolist()+ ['EXT_SOURCE_2','
len(Unwanted_application)
```

Out[25]:

```
51
```

In [26]:

```
# Checking the relevance of Flag_Document and whether it has any relation with loan repay
col_Doc = [ 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3','FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', '
           'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9','FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11',
           'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15','FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17'
           'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21']
df_flag = application_d[col_Doc+["TARGET"]]
length = len(col_Doc)
df_flag["TARGET"] = df_flag["TARGET"].replace({1:"Defaulter",0:"Repayer"})
fig = plt.figure(figsize=(21,24))
for i,j in itertools.zip_longest(col_Doc,range(length)):
    plt.subplot(5,4,j+1)
    ax = sns.countplot(df_flag[i],hue=df_flag["TARGET"],palette=["r","g"])
    plt.yticks(fontsize=8)
    plt.xlabel("")
    plt.ylabel("")
    plt.title(i)
```



In [27]:

```
# Including the flag documents for dropping the Document columns
col_Doc.remove('FLAG_DOCUMENT_3')
Unwanted_application = Unwanted_application + col_Doc
len(Unwanted_application)
```

Out[27]:

```
70
```

In [28]:

```
# checking is there is any correlation between mobile phone, work phone etc, email, Fami
contact_col = ['FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE',
       'FLAG_PHONE', 'FLAG_EMAIL','TARGET']
Contact_corr = application_d[contact_col].corr()
fig = plt.figure(figsize=(8,8))
ax = sns.heatmap(Contact_corr,
            xticklabels=Contact_corr.columns,
            yticklabels=Contact_corr.columns,
            annot = True,
            cmap ="RdYlGn",
            linewidth=1)
```

In [29]:

```
contact_col.remove('TARGET')
Unwanted_application = Unwanted_application + contact_col
len(Unwanted_application)
```

Out[29]:

```
76
```

#Total 76 columns can be deleted from applicationDF

In [30]:

```
application_d.drop(labels=Unwanted_application,axis=1,inplace=True)
```

In [31]:

```
application_d.shape
```

Out[31]:

```
(307511, 46)
```

In [32]:

```
application_d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 46 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   SK_ID_CURR                   307511 non-null  int64  
 1   TARGET                       307511 non-null  int64  
 2   NAME_CONTRACT_TYPE           307511 non-null  object 
 3   CODE_GENDER                  307511 non-null  object 
 4   FLAG_OWN_CAR                 307511 non-null  object 
 5   FLAG_OWN_REALTY              307511 non-null  object 
 6   CNT_CHILDREN                 307511 non-null  int64  
 7   AMT_INCOME_TOTAL             307511 non-null  float64
 8   AMT_CREDIT                   307511 non-null  float64
 9   AMT_ANNUITY                  307499 non-null  float64
 10  AMT_GOODS_PRICE              307233 non-null  float64
 11  NAME_TYPE_SUITE              306219 non-null  object 
 12  NAME_INCOME_TYPE             307511 non-null  object 
 13  NAME_EDUCATION_TYPE          307511 non-null  object 
 14  NAME_FAMILY_STATUS           307511 non-null  object 
 15  NAME_HOUSING_TYPE            307511 non-null  object 
 16  REGION_POPULATION_RELATIVE   307511 non-null  float64
 17  DAYS_BIRTH                   307511 non-null  int64  
 18  DAYS_EMPLOYED                307511 non-null  int64  
 19  DAYS_REGISTRATION            307511 non-null  float64
 20  DAYS_ID_PUBLISH              307511 non-null  int64  
 21  OCCUPATION_TYPE              211120 non-null  object 
 22  CNT_FAM_MEMBERS              307509 non-null  float64
 23  REGION_RATING_CLIENT         307511 non-null  int64  
 24  REGION_RATING_CLIENT_W_CITY  307511 non-null  int64  
 25  WEEKDAY_APPR_PROCESS_START   307511 non-null  object 
 26  HOUR_APPR_PROCESS_START      307511 non-null  int64  
 27  REG_REGION_NOT_LIVE_REGION   307511 non-null  int64  
 28  REG_REGION_NOT_WORK_REGION   307511 non-null  int64  
 29  LIVE_REGION_NOT_WORK_REGION  307511 non-null  int64  
 30  REG_CITY_NOT_LIVE_CITY       307511 non-null  int64  
 31  REG_CITY_NOT_WORK_CITY       307511 non-null  int64  
 32  LIVE_CITY_NOT_WORK_CITY      307511 non-null  int64  
 33  ORGANIZATION_TYPE            307511 non-null  object 
 34  OBS_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 35  DEF_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 36  OBS_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 37  DEF_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 38  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64
 39  FLAG_DOCUMENT_3              307511 non-null  int64  
 40  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64
 41  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64
 42  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64
 43  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64
 44  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64
 45  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64
dtypes: float64(18), int64(16), object(12)
memory usage: 107.9+ MB
```

Analyze & Delete Unnecessary Columns in previous_d

In [33]:

```
Unwanted_previous = nullcol_40_previous["Column Name"].tolist()
Unwanted_previous
```

Out[33]:

```
['AMT_DOWN_PAYMENT',
 'RATE_DOWN_PAYMENT',
 'RATE_INTEREST_PRIMARY',
 'RATE_INTEREST_PRIVILEGED',
 'NAME_TYPE_SUITE',
 'DAYS_FIRST_DRAWING',
 'DAYS_FIRST_DUE',
 'DAYS_LAST_DUE_1ST_VERSION',
 'DAYS_LAST_DUE',
 'DAYS_TERMINATION',
 'NFLAG_INSURED_ON_APPROVAL']
```

In [34]:

```
Unnecessary_previous = ['WEEKDAY_APPR_PROCESS_START','HOUR_APPR_PROCESS_START',
                        'FLAG_LAST_APPL_PER_CONTRACT','NFLAG_LAST_APPL_IN_DAY']
```

In [35]:

```
Unwanted_previous = Unwanted_previous + Unnecessary_previous
len(Unwanted_previous)
```

Out[35]:

```
15
```

In [36]:

```
#Deleting the unwanted columns
previous_d.drop(labels=Unwanted_previous,axis=1,inplace=True)
previous_d.shape
```

Out[36]:

```
(1670214, 22)
```

In [37]:

```
previous_d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 22 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_GOODS_PRICE         1284699 non-null  float64
 7   NAME_CASH_LOAN_PURPOSE  1670214 non-null  object
 8   NAME_CONTRACT_STATUS    1670214 non-null  object
 9   DAYS_DECISION           1670214 non-null  int64
 10  NAME_PAYMENT_TYPE       1670214 non-null  object
 11  CODE_REJECT_REASON      1670214 non-null  object
 12  NAME_CLIENT_TYPE        1670214 non-null  object
 13  NAME_GOODS_CATEGORY     1670214 non-null  object
 14  NAME_PORTFOLIO          1670214 non-null  object
 15  NAME_PRODUCT_TYPE       1670214 non-null  object
 16  CHANNEL_TYPE            1670214 non-null  object
 17  SELLERPLACE_AREA        1670214 non-null  int64
 18  NAME_SELLER_INDUSTRY    1670214 non-null  object
 19  CNT_PAYMENT             1297984 non-null  float64
 20  NAME_YIELD_GROUP        1670214 non-null  object
 21  PRODUCT_COMBINATION     1669868 non-null  object
dtypes: float64(5), int64(4), object(13)
memory usage: 280.3+ MB
```

4. Standardize Values

In [38]:

```
# Converting Negative days to positive days

date_col = ['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH']

for col in date_col:
    application_d[col] = abs(application_d[col])
```

In [39]:

```
# Binning Numerical Columns to create a categorical column

# Creating bins for income amount
application_d['AMT_INCOME_TOTAL']=application_d['AMT_INCOME_TOTAL']/100000

bins = [0,1,2,3,4,5,6,7,8,9,10,11]
slot = ['0-100K','100K-200K', '200k-300k','300k-400k','400k-500k','500k-600k','600k-700k

application_d['AMT_INCOME_RANGE']=pd.cut(application_d['AMT_INCOME_TOTAL'],bins,labels=s
```

In [40]:

```
application_d['AMT_INCOME_RANGE'].value_counts(normalize=True)*100
```

Out[40]:

```
100K-200K    50.735000
200k-300k    21.210691
0-100K       20.729695
300k-400k     4.776116
400k-500k     1.744669
500k-600k     0.356354
600k-700k     0.282805
800k-900k     0.096980
700k-800k     0.052721
900k-1M       0.009112
1M Above      0.005858
Name: AMT_INCOME_RANGE, dtype: float64
```

In [41]:

```
# Creating bins for Credit amount (bank indentification number)
application_d['AMT_CREDIT']=application_d['AMT_CREDIT']/100000

bins = [0,1,2,3,4,5,6,7,8,9,10,100]
slots = ['0-100K','100K-200K', '200k-300k','300k-400k','400k-500k','500k-600k','600k-700
       '800k-900k','900k-1M', '1M Above']

application_d['AMT_CREDIT_RANGE']=pd.cut(application_d['AMT_CREDIT'],bins=bins,labels=sl
```

In [42]:

```
#checking the bin number and % of data in each category
application_d['AMT_CREDIT_RANGE'].value_counts(normalize=True)*100
```

Out[42]:

```
200k-300k    17.824728
1M Above     16.254703
500k-600k    11.131960
400k-500k    10.418489
100K-200K     9.801275
300k-400k     8.564897
600k-700k     7.820533
800k-900k     7.086576
700k-800k     6.241403
900k-1M       2.902986
0-100K        1.952450
Name: AMT_CREDIT_RANGE, dtype: float64
```

In [43]:

```
# Creating bins for Age
application_d['AGE'] = application_d['DAYS_BIRTH'] // 365
bins = [0,20,30,40,50,100]
slots = ['0-20','20-30','30-40','40-50','50 above']
application_d['AGE_GROUP']=pd.cut(application_d['AGE'],bins=bins,labels=slots)
```

In [44]:

```
application_d['AGE_GROUP'].value_counts(normalize=True)*100
```

Out[44]:

```
50 above    31.604398
30-40       27.028952
40-50       24.194582
20-30       17.171743
0-20         0.000325
Name: AGE_GROUP, dtype: float64
```

In [45]:

```
# Creating bins for Employement Time
application_d['YEARS_EMPLOYED'] = application_d['DAYS_EMPLOYED'] // 365
bins = [0,5,10,20,30,40,50,60,150]
slots = ['0-5','5-10','10-20','20-30','30-40','40-50','50-60','60 above']

application_d['EMPLOYMENT_YEAR']=pd.cut(application_d['YEARS_EMPLOYED'],bins=bins,labels
```

In [46]:

```
#checking the binning of data and % of data in each category
application_d['EMPLOYMENT_YEAR'].value_counts(normalize=True)*100
```

Out[46]:

```
0-5         55.582363
5-10        24.966441
10-20       14.564315
20-30        3.750117
30-40        1.058720
40-50        0.078044
50-60        0.000000
60 above     0.000000
Name: EMPLOYMENT_YEAR, dtype: float64
```

In [47]:

```
#Checking the number of unique values each column possess to identify categorical column
application_d.nunique().sort_values()
```

Out[47]:

```
LIVE_CITY_NOT_WORK_CITY             2
TARGET                              2
NAME_CONTRACT_TYPE                  2
REG_REGION_NOT_LIVE_REGION          2
FLAG_OWN_CAR                        2
FLAG_OWN_REALTY                     2
REG_REGION_NOT_WORK_REGION          2
LIVE_REGION_NOT_WORK_REGION         2
FLAG_DOCUMENT_3                     2
REG_CITY_NOT_LIVE_CITY              2
REG_CITY_NOT_WORK_CITY              2
REGION_RATING_CLIENT                3
CODE_GENDER                         3
REGION_RATING_CLIENT_W_CITY         3
AMT_REQ_CREDIT_BUREAU_HOUR          5
NAME_EDUCATION_TYPE                 5
AGE_GROUP                           5
NAME_FAMILY_STATUS                  6
NAME_HOUSING_TYPE                   6
EMPLOYMENT_YEAR                     6
WEEKDAY_APPR_PROCESS_START          7
NAME_TYPE_SUITE                     7
NAME_INCOME_TYPE                    8
AMT_REQ_CREDIT_BUREAU_WEEK          9
AMT_REQ_CREDIT_BUREAU_DAY           9
DEF_60_CNT_SOCIAL_CIRCLE            9
DEF_30_CNT_SOCIAL_CIRCLE           10
AMT_CREDIT_RANGE                   11
AMT_INCOME_RANGE                   11
AMT_REQ_CREDIT_BUREAU_QRT          11
CNT_CHILDREN                       15
CNT_FAM_MEMBERS                    17
OCCUPATION_TYPE                    18
HOUR_APPR_PROCESS_START            24
AMT_REQ_CREDIT_BUREAU_MON          24
AMT_REQ_CREDIT_BUREAU_YEAR         25
OBS_60_CNT_SOCIAL_CIRCLE           33
OBS_30_CNT_SOCIAL_CIRCLE           33
AGE                                50
YEARS_EMPLOYED                     51
ORGANIZATION_TYPE                  58
REGION_POPULATION_RELATIVE         81
AMT_GOODS_PRICE                  1002
AMT_INCOME_TOTAL                 2548
DAYS_LAST_PHONE_CHANGE           3773
AMT_CREDIT                       5603
DAYS_ID_PUBLISH                  6168
DAYS_EMPLOYED                   12574
AMT_ANNUITY                     13672
DAYS_REGISTRATION               15688
DAYS_BIRTH                      17460
SK_ID_CURR                     307511
dtype: int64
```

In [48]:

```
#Data Type Conversion

# inspecting the column types if they are in correct data type using the above result.
application_d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 52 columns):
 #   Column                       Non-Null Count   Dtype   
---  ------                       --------------   -----   
 0   SK_ID_CURR                   307511 non-null  int64   
 1   TARGET                       307511 non-null  int64   
 2   NAME_CONTRACT_TYPE           307511 non-null  object  
 3   CODE_GENDER                  307511 non-null  object  
 4   FLAG_OWN_CAR                 307511 non-null  object  
 5   FLAG_OWN_REALTY              307511 non-null  object  
 6   CNT_CHILDREN                 307511 non-null  int64   
 7   AMT_INCOME_TOTAL             307511 non-null  float64 
 8   AMT_CREDIT                   307511 non-null  float64 
 9   AMT_ANNUITY                  307499 non-null  float64 
 10  AMT_GOODS_PRICE              307233 non-null  float64 
 11  NAME_TYPE_SUITE              306219 non-null  object  
 12  NAME_INCOME_TYPE             307511 non-null  object  
 13  NAME_EDUCATION_TYPE          307511 non-null  object  
 14  NAME_FAMILY_STATUS           307511 non-null  object  
 15  NAME_HOUSING_TYPE            307511 non-null  object  
 16  REGION_POPULATION_RELATIVE   307511 non-null  float64 
 17  DAYS_BIRTH                   307511 non-null  int64   
 18  DAYS_EMPLOYED                307511 non-null  int64   
 19  DAYS_REGISTRATION            307511 non-null  float64 
 20  DAYS_ID_PUBLISH              307511 non-null  int64   
 21  OCCUPATION_TYPE              211120 non-null  object  
 22  CNT_FAM_MEMBERS              307509 non-null  float64 
 23  REGION_RATING_CLIENT         307511 non-null  int64   
 24  REGION_RATING_CLIENT_W_CITY  307511 non-null  int64   
 25  WEEKDAY_APPR_PROCESS_START   307511 non-null  object  
 26  HOUR_APPR_PROCESS_START      307511 non-null  int64   
 27  REG_REGION_NOT_LIVE_REGION   307511 non-null  int64   
 28  REG_REGION_NOT_WORK_REGION   307511 non-null  int64   
 29  LIVE_REGION_NOT_WORK_REGION  307511 non-null  int64   
 30  REG_CITY_NOT_LIVE_CITY       307511 non-null  int64   
 31  REG_CITY_NOT_WORK_CITY       307511 non-null  int64   
 32  LIVE_CITY_NOT_WORK_CITY      307511 non-null  int64   
 33  ORGANIZATION_TYPE            307511 non-null  object  
 34  OBS_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64 
 35  DEF_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64 
 36  OBS_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64 
 37  DEF_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64 
 38  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64 
 39  FLAG_DOCUMENT_3              307511 non-null  int64   
 40  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64 
 41  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64 
 42  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64 
 43  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64 
 44  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64 
 45  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64 
 46  AMT_INCOME_RANGE             307279 non-null  category
 47  AMT_CREDIT_RANGE             307511 non-null  category
 48  AGE                          307511 non-null  int64   
 49  AGE_GROUP                    307511 non-null  category
 50  YEARS_EMPLOYED               307511 non-null  int64   
 51  EMPLOYMENT_YEAR              224233 non-null  category
dtypes: category(4), float64(18), int64(18), object(12)
memory usage: 113.8+ MB
```

In [49]:

```
#Converting from Object and Numerical columns to Categorical Columns
Categorical_columns = ['NAME_CONTRACT_TYPE','CODE_GENDER','NAME_TYPE_SUITE','NAME_INCOME_
                       'NAME_FAMILY_STATUS','NAME_HOUSING_TYPE','OCCUPATION_TYPE','WEEKD
                       'ORGANIZATION_TYPE','FLAG_OWN_CAR','FLAG_OWN_REALTY','LIVE_CITY_N
                       'REG_CITY_NOT_LIVE_CITY','REG_CITY_NOT_WORK_CITY','REG_REGION_NOT
                       'LIVE_REGION_NOT_WORK_REGION','REGION_RATING_CLIENT','WEEKDAY_APP
                       'REGION_RATING_CLIENT_W_CITY'
                      ]
for col in Categorical_columns:
    application_d[col] =pd.Categorical(application_d[col])
```

In [50]:

```
application_d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 52 columns):
 #   Column                       Non-Null Count   Dtype   
---  ------                       --------------   -----   
 0   SK_ID_CURR                   307511 non-null  int64   
 1   TARGET                       307511 non-null  int64   
 2   NAME_CONTRACT_TYPE           307511 non-null  category
 3   CODE_GENDER                  307511 non-null  category
 4   FLAG_OWN_CAR                 307511 non-null  category
 5   FLAG_OWN_REALTY              307511 non-null  category
 6   CNT_CHILDREN                 307511 non-null  int64   
 7   AMT_INCOME_TOTAL             307511 non-null  float64 
 8   AMT_CREDIT                   307511 non-null  float64 
 9   AMT_ANNUITY                  307499 non-null  float64 
 10  AMT_GOODS_PRICE              307233 non-null  float64 
 11  NAME_TYPE_SUITE              306219 non-null  category
 12  NAME_INCOME_TYPE             307511 non-null  category
 13  NAME_EDUCATION_TYPE          307511 non-null  category
 14  NAME_FAMILY_STATUS           307511 non-null  category
 15  NAME_HOUSING_TYPE            307511 non-null  category
 16  REGION_POPULATION_RELATIVE   307511 non-null  float64 
 17  DAYS_BIRTH                   307511 non-null  int64   
 18  DAYS_EMPLOYED                307511 non-null  int64   
 19  DAYS_REGISTRATION            307511 non-null  float64 
 20  DAYS_ID_PUBLISH              307511 non-null  int64   
 21  OCCUPATION_TYPE              211120 non-null  category
 22  CNT_FAM_MEMBERS              307509 non-null  float64 
 23  REGION_RATING_CLIENT         307511 non-null  category
 24  REGION_RATING_CLIENT_W_CITY  307511 non-null  category
 25  WEEKDAY_APPR_PROCESS_START   307511 non-null  category
 26  HOUR_APPR_PROCESS_START      307511 non-null  int64   
 27  REG_REGION_NOT_LIVE_REGION   307511 non-null  int64   
 28  REG_REGION_NOT_WORK_REGION   307511 non-null  category
 29  LIVE_REGION_NOT_WORK_REGION  307511 non-null  category
 30  REG_CITY_NOT_LIVE_CITY       307511 non-null  category
 31  REG_CITY_NOT_WORK_CITY       307511 non-null  category
 32  LIVE_CITY_NOT_WORK_CITY      307511 non-null  category
 33  ORGANIZATION_TYPE            307511 non-null  category
 34  OBS_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64 
 35  DEF_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64 
 36  OBS_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64 
 37  DEF_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64 
 38  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64 
 39  FLAG_DOCUMENT_3              307511 non-null  int64   
 40  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64 
 41  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64 
 42  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64 
 43  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64 
 44  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64 
 45  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64 
 46  AMT_INCOME_RANGE             307279 non-null  category
 47  AMT_CREDIT_RANGE             307511 non-null  category
 48  AGE                          307511 non-null  int64   
 49  AGE_GROUP                    307511 non-null  category
 50  YEARS_EMPLOYED               307511 non-null  int64   
 51  EMPLOYMENT_YEAR              224233 non-null  category
dtypes: category(23), float64(18), int64(11)
memory usage: 74.8 MB
```

Standardize Values for previousDF

In [51]:

```
previous_d.nunique().sort_values()
```

Out[51]:

```
NAME_PRODUCT_TYPE               3
NAME_PAYMENT_TYPE               4
NAME_CONTRACT_TYPE              4
NAME_CLIENT_TYPE                4
NAME_CONTRACT_STATUS            4
NAME_PORTFOLIO                  5
NAME_YIELD_GROUP                5
CHANNEL_TYPE                    8
CODE_REJECT_REASON              9
NAME_SELLER_INDUSTRY           11
PRODUCT_COMBINATION            17
NAME_CASH_LOAN_PURPOSE         25
NAME_GOODS_CATEGORY            28
CNT_PAYMENT                    49
SELLERPLACE_AREA             2097
DAYS_DECISION                2922
AMT_CREDIT                  86803
AMT_GOODS_PRICE             93885
AMT_APPLICATION             93885
SK_ID_CURR                 338857
AMT_ANNUITY                357959
SK_ID_PREV                1670214
dtype: int64
```

In [52]:

```
previous_d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 22 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_GOODS_PRICE         1284699 non-null  float64
 7   NAME_CASH_LOAN_PURPOSE  1670214 non-null  object
 8   NAME_CONTRACT_STATUS    1670214 non-null  object
 9   DAYS_DECISION           1670214 non-null  int64
 10  NAME_PAYMENT_TYPE       1670214 non-null  object
 11  CODE_REJECT_REASON      1670214 non-null  object
 12  NAME_CLIENT_TYPE        1670214 non-null  object
 13  NAME_GOODS_CATEGORY     1670214 non-null  object
 14  NAME_PORTFOLIO          1670214 non-null  object
 15  NAME_PRODUCT_TYPE       1670214 non-null  object
 16  CHANNEL_TYPE            1670214 non-null  object
 17  SELLERPLACE_AREA        1670214 non-null  int64
 18  NAME_SELLER_INDUSTRY    1670214 non-null  object
 19  CNT_PAYMENT             1297984 non-null  float64
 20  NAME_YIELD_GROUP        1670214 non-null  object
 21  PRODUCT_COMBINATION     1669868 non-null  object
dtypes: float64(5), int64(4), object(13)
memory usage: 280.3+ MB
```

In [53]:

```
#Converting negative days to positive days
previous_d['DAYS_DECISION'] = abs(previous_d['DAYS_DECISION'])
```

In [54]:

```
#age group calculation e.g. 388 will be grouped as 300-400
previous_d['DAYS_DECISION_GROUP'] = (previous_d['DAYS_DECISION']-(previous_d['DAYS_DECIS
```

In [55]:

```
previous_d['DAYS_DECISION_GROUP'].value_counts(normalize=True)*100
```

Out[55]:

```
0-400        37.490525
400-800      22.944724
800-1200     12.444753
1200-1600     7.904556
2400-2800     6.297456
1600-2000     5.795784
2000-2400     5.684960
2800-3200     1.437241
Name: DAYS_DECISION_GROUP, dtype: float64
```

Almost 37% loan applicatants have applied for a new loan within 0-400 days of previous loan decision

In [56]:

```
#Converting Categorical columns from Object to categorical
Catgorical_col_p = ['NAME_CASH_LOAN_PURPOSE','NAME_CONTRACT_STATUS','NAME_PAYMENT_TYPE',
                    'CODE_REJECT_REASON','NAME_CLIENT_TYPE','NAME_GOODS_CATEGORY','NAME_
                   'NAME_PRODUCT_TYPE','CHANNEL_TYPE','NAME_SELLER_INDUSTRY','NAME_YIELD
                    'NAME_CONTRACT_TYPE','DAYS_DECISION_GROUP']

for col in Catgorical_col_p:
    previous_d[col] = pd.Categorical(previous_d[col])
```

In [57]:

```
# inspecting the column types after conversion
previous_d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 23 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  category
 3   AMT_ANNUITY             1297979 non-null  float64
 4   AMT_APPLICATION         1670214 non-null  float64
 5   AMT_CREDIT              1670213 non-null  float64
 6   AMT_GOODS_PRICE         1284699 non-null  float64
 7   NAME_CASH_LOAN_PURPOSE  1670214 non-null  category
 8   NAME_CONTRACT_STATUS    1670214 non-null  category
 9   DAYS_DECISION           1670214 non-null  int64
 10  NAME_PAYMENT_TYPE       1670214 non-null  category
 11  CODE_REJECT_REASON      1670214 non-null  category
 12  NAME_CLIENT_TYPE        1670214 non-null  category
 13  NAME_GOODS_CATEGORY     1670214 non-null  category
 14  NAME_PORTFOLIO          1670214 non-null  category
 15  NAME_PRODUCT_TYPE       1670214 non-null  category
 16  CHANNEL_TYPE            1670214 non-null  category
 17  SELLERPLACE_AREA        1670214 non-null  int64
 18  NAME_SELLER_INDUSTRY    1670214 non-null  category
 19  CNT_PAYMENT             1297984 non-null  float64
 20  NAME_YIELD_GROUP        1670214 non-null  category
 21  PRODUCT_COMBINATION     1669868 non-null  category
 22  DAYS_DECISION_GROUP     1670214 non-null  category
dtypes: category(14), float64(5), int64(4)
memory usage: 137.0 MB
```

Null Value Data Imputation

Imputing Null Values in applicationDF

In [58]:

```
# checking the null value % of each column in applicationDF dataframe
round(application_d.isnull().sum() / application_d.shape[0] * 100.00,2)
```

Out[58]:

```
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
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
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_3                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
AMT_INCOME_RANGE               0.08
AMT_CREDIT_RANGE               0.00
AGE                            0.00
AGE_GROUP                      0.00
YEARS_EMPLOYED                 0.00
EMPLOYMENT_YEAR               27.08
dtype: float64
```

In [59]:

```
application_d['NAME_TYPE_SUITE'].describe()
```

Out[59]:

```
count            306219
unique                7
top       Unaccompanied
freq             248526
Name: NAME_TYPE_SUITE, dtype: object
```

In [60]:

```
application_d['NAME_TYPE_SUITE'].fillna((application_d['NAME_TYPE_SUITE'].mode()[0]),inp
```

In [61]:

```
application_d['OCCUPATION_TYPE'] = application_d['OCCUPATION_TYPE'].cat.add_categories('
application_d['OCCUPATION_TYPE'].fillna('Unknown', inplace = True)
```

In [62]:

```
application_d[['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']].describe()
```

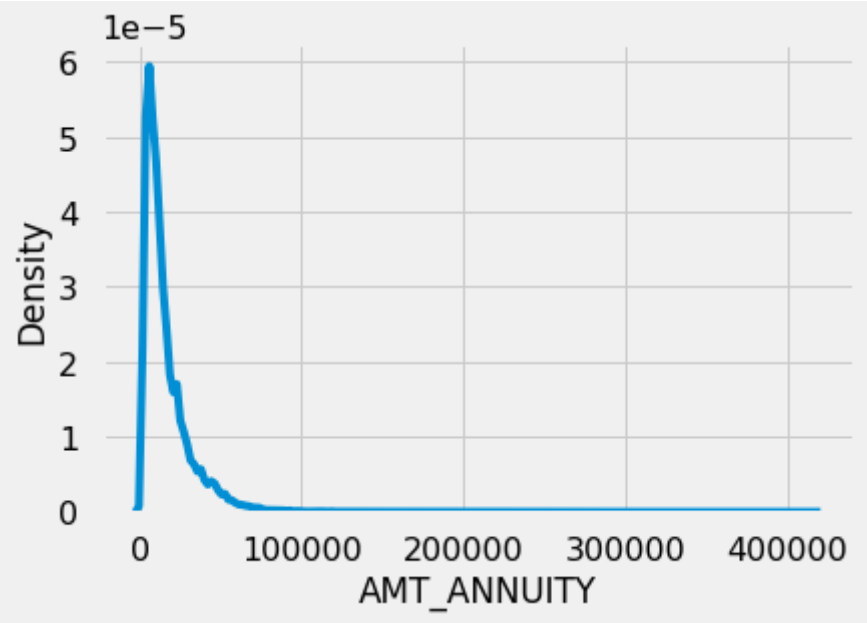
Out[62]:

	AMT_REQ_CREDIT_BUREAU_HOUR	AMT_REQ_CREDIT_BUREAU_DAY	AMT_REQ_CREI
count	265992.000000	265992.000000	
mean	0.006402	0.007000	
std	0.083849	0.110757	
min	0.000000	0.000000	
25%	0.000000	0.000000	
50%	0.000000	0.000000	
75%	0.000000	0.000000	
max	4.000000	9.000000	

#Impute AMT_ANNUITY with median as the distribution is greatly skewed:

In [63]:

```
plt.figure(figsize=(6,4))
sns.kdeplot(previous_d['AMT_ANNUITY'])
plt.show()
```



#There is a single peak at the left side of the distribution and it indicates the presence of outliers and hence imputing with mean would not be the right approach and hence imputing with median.

In [64]:

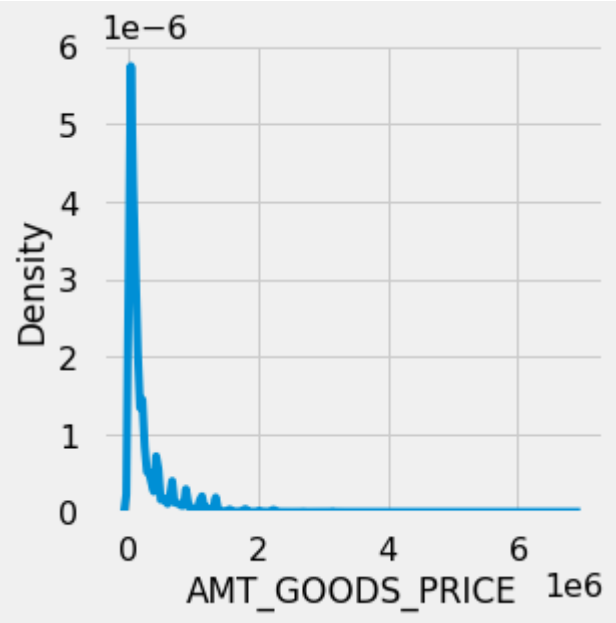
```
previous_d['AMT_ANNUITY'].fillna(previous_d['AMT_ANNUITY'].median(),inplace = True)
```

In [ ]:

```
#Impute AMT_GOODS_PRICE with mode as the distribution is closely similar:
```

In [65]:

```
plt.figure(figsize=(4,4))
sns.kdeplot(previous_d['AMT_GOODS_PRICE'][pd.notnull(previous_d['AMT_GOODS_PRICE'])])
plt.show()
```

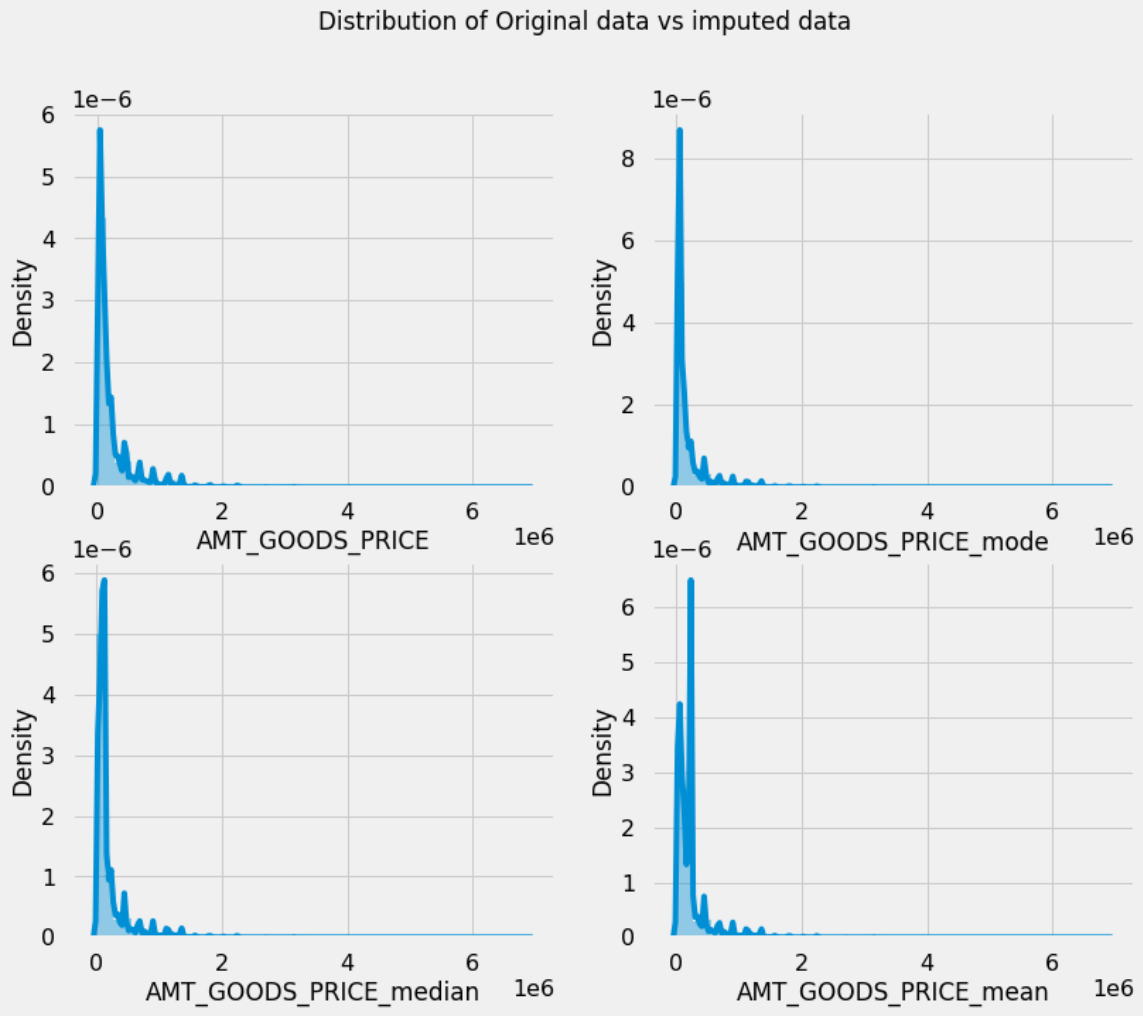


In [ ]:

```
# There are several peaks along the distribution. Let's impute using the mode, mean and
```

In [66]:

```
stats_d = pd.DataFrame() # columns imputed with mode, median and mean
stats_d['AMT_GOODS_PRICE_mode'] = previous_d['AMT_GOODS_PRICE'].fillna(previous_d['AMT_G
stats_d['AMT_GOODS_PRICE_median'] = previous_d['AMT_GOODS_PRICE'].fillna(previous_d['AMT
stats_d['AMT_GOODS_PRICE_mean'] = previous_d['AMT_GOODS_PRICE'].fillna(previous_d['AMT_G

cols_d = ['AMT_GOODS_PRICE_mode', 'AMT_GOODS_PRICE_median','AMT_GOODS_PRICE_mean']

plt.figure(figsize=(12,10))
plt.suptitle('Distribution of Original data vs imputed data')
plt.subplot(221)
sns.distplot(previous_d['AMT_GOODS_PRICE'][pd.notnull(previous_d['AMT_GOODS_PRICE'])]);
for i in enumerate(cols_d):
    plt.subplot(2,2,i[0]+2)
    sns.distplot(stats_d[i[1]])
```



The original distribution is closer with the distribution of data imputed with mode in this case

In [67]:

```
previous_d['AMT_GOODS_PRICE'].fillna(previous_d['AMT_GOODS_PRICE'].mode()[0], inplace=Tr
```

#Impute CNT_PAYMENT with 0 as the NAME_CONTRACT_STATUS for these indicate that most of these loans were not started:

In [68]:

```
previous_d.loc[previous_d['CNT_PAYMENT'].isnull(),'NAME_CONTRACT_STATUS'].value_counts()
```

Out[68]:

```
Canceled        305805
Refused          40897
Unused offer     25524
Approved             4
Name: NAME_CONTRACT_STATUS, dtype: int64
```

In [69]:

```
previous_d['CNT_PAYMENT'].fillna(0,inplace = True)
```

In [70]:

```
# checking the null value % of each column in previousDF dataframe
round(previous_d.isnull().sum() / previous_d.shape[0] * 100.0,2)
```

Out[70]:

```
SK_ID_PREV                0.00
SK_ID_CURR                0.00
NAME_CONTRACT_TYPE        0.00
AMT_ANNUITY               0.00
AMT_APPLICATION           0.00
AMT_CREDIT                0.00
AMT_GOODS_PRICE           0.00
NAME_CASH_LOAN_PURPOSE    0.00
NAME_CONTRACT_STATUS      0.00
DAYS_DECISION             0.00
NAME_PAYMENT_TYPE         0.00
CODE_REJECT_REASON        0.00
NAME_CLIENT_TYPE          0.00
NAME_GOODS_CATEGORY       0.00
NAME_PORTFOLIO            0.00
NAME_PRODUCT_TYPE         0.00
CHANNEL_TYPE              0.00
SELLERPLACE_AREA          0.00
```

We still have few null values in the PRODUCT_COMBINATION column. We can ignore as this percentage is very less.

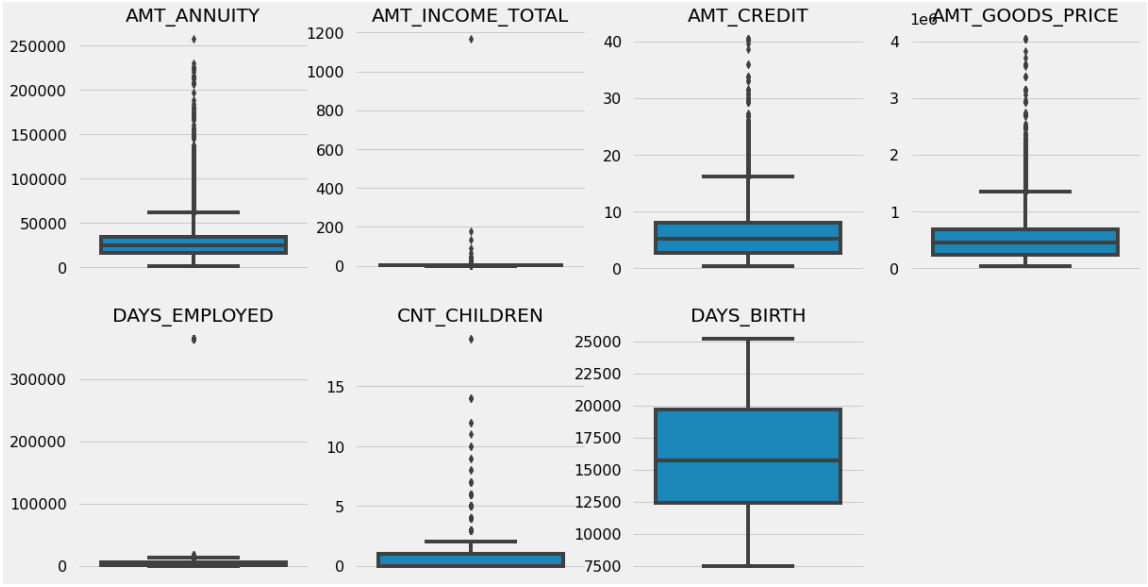
Identifying the outliers

In [ ]:

```
#Finding outlier information in applicationDF
```

In [71]:

```
plt.figure(figsize = (18,10))
app_outlier_1 = ['AMT_ANNUITY','AMT_INCOME_TOTAL','AMT_CREDIT','AMT_GOODS_PRICE','DAYS_E
app_outlier_2 = ['CNT_CHILDREN','DAYS_BIRTH']
for i in enumerate(app_outlier_1):
    plt.subplot(2,4,i[0]+1)
    sns.boxplot(y =application_d[i[1]])
    plt.title(i[1])
    plt.ylabel('')
for i in enumerate(app_outlier_2):
    plt.subplot(2,4,i[0]+6)
    sns.boxplot(y=application_d[i[1]])
    plt.title(i[1])
    plt.ylabel('')
```



It can be seen that in current application data AMT_ANNUITY, AMT_CREDIT, AMT_GOODS_PRICE,CNT_CHILDREN have some number of outliers. AMT_INCOME_TOTAL has huge number of outliers which indicate that few of the loan applicants have high income when compared to the others. DAYS_BIRTH has no outliers which means the data available is reliable. DAYS_EMPLOYED has outlier values around 350000(days) which is around 958 years which is impossible and hence this has to be incorrect entry.

In [72]:

```
application_d[['AMT_ANNUITY', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_GOODS_PRICE', 'DAYS
```

Out[72]:

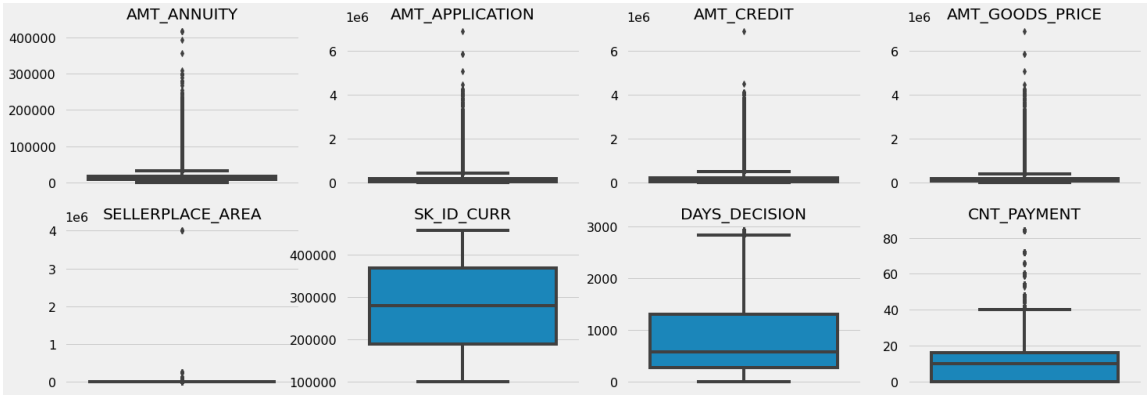
	AMT_ANNUITY	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_GOODS_PRICE	DAYS_BIF
count	307499.000000	307511.000000	307511.000000	3.072330e+05	307511.000
mean	27108.573909	1.687979	5.990260	5.383962e+05	16036.995
std	14493.737315	2.371231	4.024908	3.694465e+05	4363.988
min	1615.500000	0.256500	0.450000	4.050000e+04	7489.000
25%	16524.000000	1.125000	2.700000	2.385000e+05	12413.000
50%	24903.000000	1.471500	5.135310	4.500000e+05	15750.000
75%	34596.000000	2.025000	8.086500	6.795000e+05	19682.000
max	258025.500000	1170.000000	40.500000	4.050000e+06	25229.000

In [ ]:

```
#Finding outlier information in previousDF
```

In [73]:

```
plt.figure(figsize=(22,8))

prev_outlier_1 = ['AMT_ANNUITY','AMT_APPLICATION','AMT_CREDIT','AMT_GOODS_PRICE','SELLER
prev_outlier_2 = ['SK_ID_CURR','DAYS_DECISION','CNT_PAYMENT']
for i in enumerate(prev_outlier_1):
    plt.subplot(2,4,i[0]+1)
    sns.boxplot(y=previous_d[i[1]])
    plt.title(i[1])
    plt.ylabel("")

for i in enumerate(prev_outlier_2):
    plt.subplot(2,4,i[0]+6)
    sns.boxplot(y=previous_d[i[1]])
    plt.title(i[1])
    plt.ylabel("")
```



It can be seen that in previous application data AMT_ANNUITY, AMT_APPLICATION, AMT_CREDIT, AMT_GOODS_PRICE, SELLERPLACE_AREA have huge number of outliers. CNT_PAYMENT has few outlier values. SK_ID_CURR is an ID column and hence no outliers. DAYS_DECISION has little number of outliers indicating that these previous applications decisions were taken long back.

In [74]:

```
previous_d[['AMT_ANNUITY', 'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_GOODS_PRICE', 'SELLERPL
```

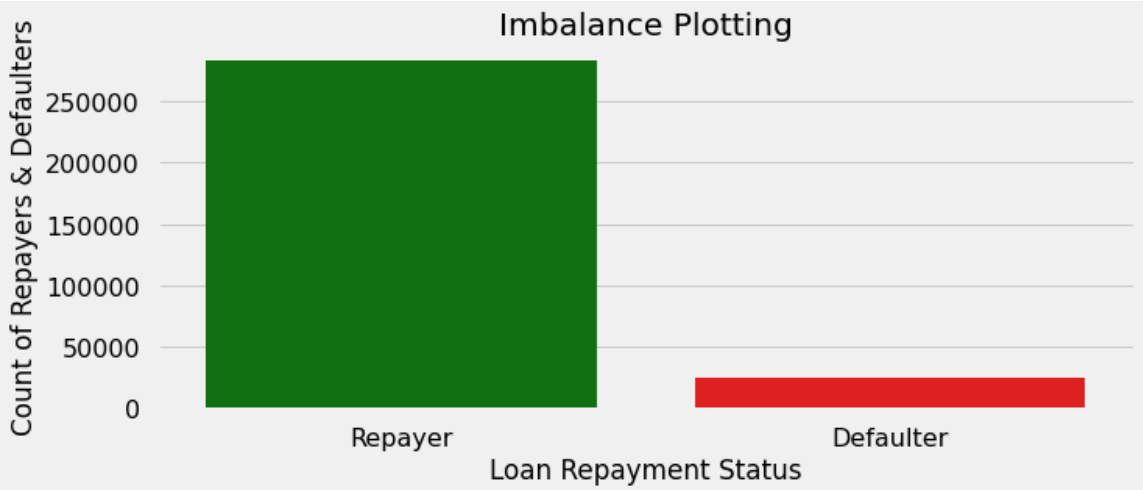
Out[74]:

	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_GOODS_PRICE	SELLERPLAC
count	1.670214e+06	1.670214e+06	1.670213e+06	1.670214e+06	1.670
mean	1.490651e+04	1.752339e+05	1.961140e+05	1.856429e+05	3.139
std	1.317751e+04	2.927798e+05	3.185746e+05	2.871413e+05	7.127
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-1.000
25%	7.547096e+03	1.872000e+04	2.416050e+04	4.500000e+04	-1.000
50%	1.125000e+04	7.104600e+04	8.054100e+04	7.105050e+04	3.000
75%	1.682403e+04	1.803600e+05	2.164185e+05	1.804050e+05	8.200
max	4.180581e+05	6.905160e+06	6.905160e+06	6.905160e+06	4.000

#Imbalance Analysis

In [75]:

```
Imbalance = application_d["TARGET"].value_counts().reset_index()

plt.figure(figsize=(10,4))
x= ['Repayer','Defaulter']
sns.barplot(x,"TARGET",data = Imbalance,palette= ['g','r'])
plt.xlabel("Loan Repayment Status")
plt.ylabel("Count of Repayers & Defaulters")
plt.title("Imbalance Plotting")
plt.show()
```



In [76]:

```
count_0 = Imbalance.iloc[0]["TARGET"]
count_1 = Imbalance.iloc[1]["TARGET"]
count_0_perc = round(count_0/(count_0+count_1)*100,2)
count_1_perc = round(count_1/(count_0+count_1)*100,2)

print('Ratios of imbalance in percentage with respect to Repayer and Defaulter datas are
print('Ratios of imbalance in relative with respect to Repayer and Defaulter datas is %.
```

```
Ratios of imbalance in percentage with respect to Repayer and Defaulter da
tas are: 91.93 and 8.07
Ratios of imbalance in relative with respect to Repayer and Defaulter data
s is 11.39 : 1 (approx)
```

In [ ]:

```
#  Plotting Functions
```

In [102]:

```
def univariate_categorical(feature,ylog=False,label_rotation=False,horizontal_layout=Tru
    temp = application_d[feature].value_counts()
    df1 = pd.DataFrame({feature: temp.index,'Number of contracts': temp.values})

    # Calculate the percentage of target=1 per category value
    cat_perc = application_d[[feature, 'TARGET']].groupby([feature],as_index=False).mean
    cat_perc["TARGET"] = cat_perc["TARGET"]*100
    cat_perc.sort_values(by='TARGET', ascending=False, inplace=True)

    if(horizontal_layout):
        fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12,6))
    else:
        fig, (ax1, ax2) = plt.subplots(nrows=2, figsize=(20,24))

    # 1. Subplot 1: Count plot of categorical column
    # sns.set_palette("Set2")
    s = sns.countplot(ax=ax1,
                      x = feature,
                      data=application_d,
                      hue ="TARGET",
                      order=cat_perc[feature],
                      palette=['g','r'])

 # Define common styling
    ax1.set_title(feature, fontdict={'fontsize' : 10, 'fontweight' : 3, 'color' : 'Blue'
    ax1.legend(['Repayer','Defaulter'])

    # If the plot is not readable, use the log scale.
    if ylog:
        ax1.set_yscale('log')
        ax1.set_ylabel("Count (log)",fontdict={'fontsize' : 10, 'fontweight' : 3, 'color


    if(label_rotation):
        s.set_xticklabels(s.get_xticklabels(),rotation=90)

    # 2. Subplot 2: Percentage of defaulters within the categorical column
    s = sns.barplot(ax=ax2,
                      x = feature,
                      y='TARGET',
                      order=cat_perc[feature],
                      data=cat_perc,
                      palette='Set2')
    if(label_rotation):
        s.set_xticklabels(s.get_xticklabels(),rotation=90)
    plt.ylabel('Percent of Defaulters [%]', fontsize=10)
    plt.tick_params(axis='both', which='major', labelsize=10)
    ax2.set_title(feature + " Defaulter %", fontdict={'fontsize' : 15, 'fontweight' : 5,
    plt.show();
```

In [106]:

```
# function for plotting repetitive countplots in bivariate categorical analysis

def bivariate_bar(x,y,df,hue,figsize):

    plt.figure(figsize=figsize)
    sns.barplot(x=x,
                y=y,
                data=application_d,
                hue=hue,
                palette =['g','r'])

    # Defining aesthetics of Labels and Title of the plot using style dictionaries
    plt.xlabel(x,fontdict={'fontsize' : 10, 'fontweight' : 3, 'color' : 'Blue'})
    plt.ylabel(y,fontdict={'fontsize' : 10, 'fontweight' : 3, 'color' : 'Blue'})
    plt.title(col, fontdict={'fontsize' : 15, 'fontweight' : 5, 'color' : 'Blue'})
    plt.xticks(rotation=90, ha='right')
    plt.legend(labels = ['Repayer','Defaulter'])
    plt.show()
```

In [79]:

```
# function for plotting repetitive rel plots in bivaritae numerical analysis on applicat

def bivariate_rel(x,y,data, hue, kind, palette, legend,figsize):

    plt.figure(figsize=figsize)
    sns.relplot(x=x,
                y=y,
                data=application_d,
                hue="TARGET",
                kind=kind,
                palette = ['g','r'],
                legend = False)
    plt.legend(['Repayer','Defaulter'])
    plt.xticks(rotation=90, ha='right')
    plt.show()
```

In [133]:

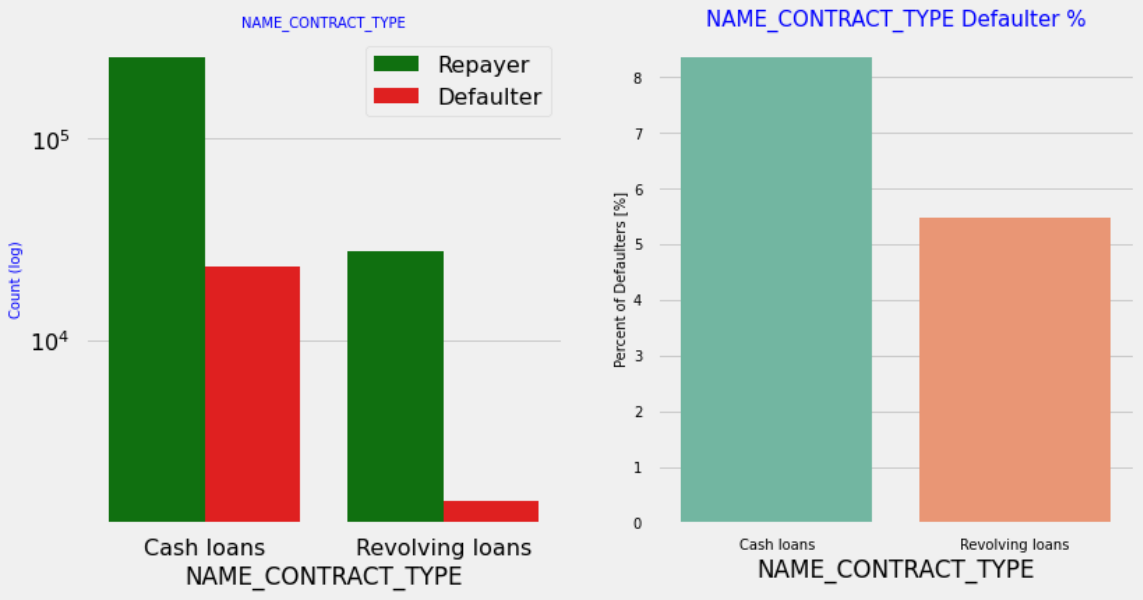
```
#function for plotting repetitive countplots in univariate categorical analysis on the m

def univariate_merged(col,df,hue,palette,ylog,figsize):
    plt.figure(figsize=figsize)
    ax=sns.countplot(x=col,
                  data=loan_process,
                  hue= hue,
                  palette= palette,
                  order=df[col].value_counts().index)


    if ylog:
        plt.yscale('log')
        plt.ylabel("Count (log)",fontdict={'fontsize' : 10, 'fontweight' : 3, 'color' :
    else:
        plt.ylabel("Count",fontdict={'fontsize' : 10, 'fontweight' : 3, 'color' : 'Blue'

    plt.title(col , fontdict={'fontsize' : 15, 'fontweight' : 5, 'color' : 'Blue'})
    plt.legend(loc = "upper right")
    plt.xticks(rotation=90, ha='right')

    plt.show()
```

In [130]:

```
# Function to plot point plots on merged dataframe

def merged_pointplot(x,y):
    plt.figure(figsize=(8,4))
    sns.pointplot(x=x,
                  y=y,
                  hue="TARGET",
                  data=loan_process_d,
                  palette =['g','r'])
   # plt.legend(['Repayer','Defaulter'])
```

In [82]:

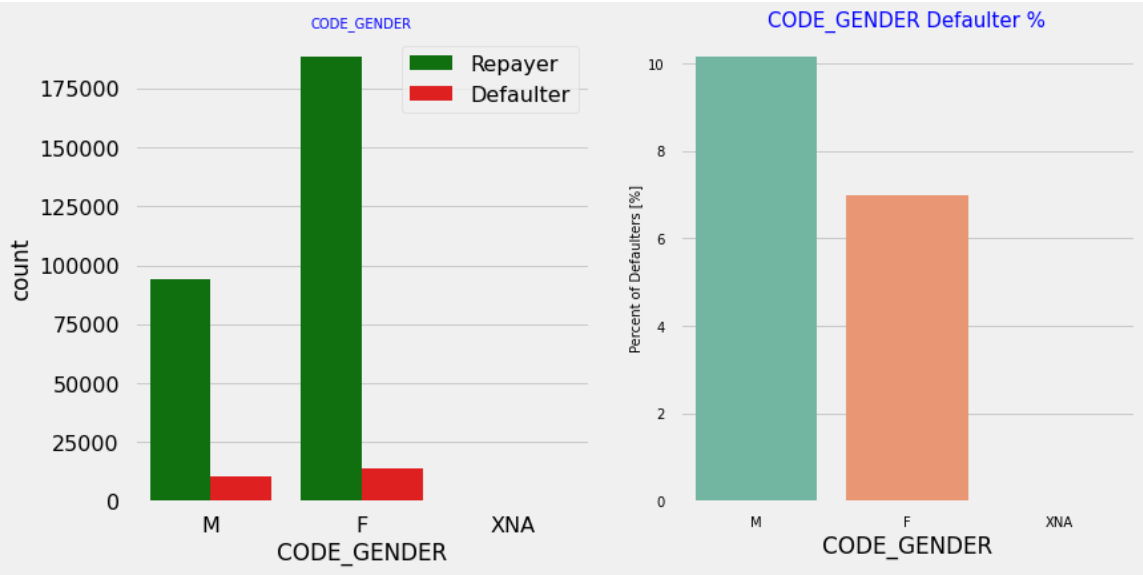
```
# Checking the contract type based on loan repayment status
univariate_categorical('NAME_CONTRACT_TYPE',True)
```



Inferences: Contract type: Revolving loans are just a small fraction (10%) from the total number of loans; in the same time, a larger amount of Revolving loans, comparing with their frequency, are not repaid.

In [83]:

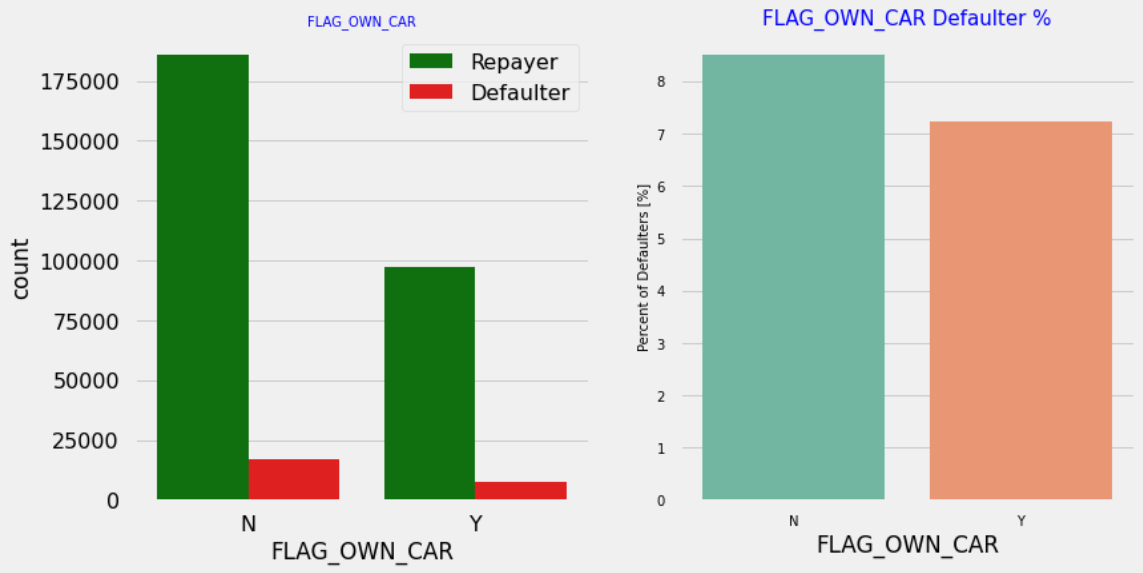
```
# Checking the type of Gender on loan repayment status
univariate_categorical('CODE_GENDER')
```



Inferences: The number of female clients is almost double the number of male clients. Based on the percentage of defaulted credits, males have a higher chance of not returning their loans (~~10%), comparing with women~~ (7%)

In [84]:

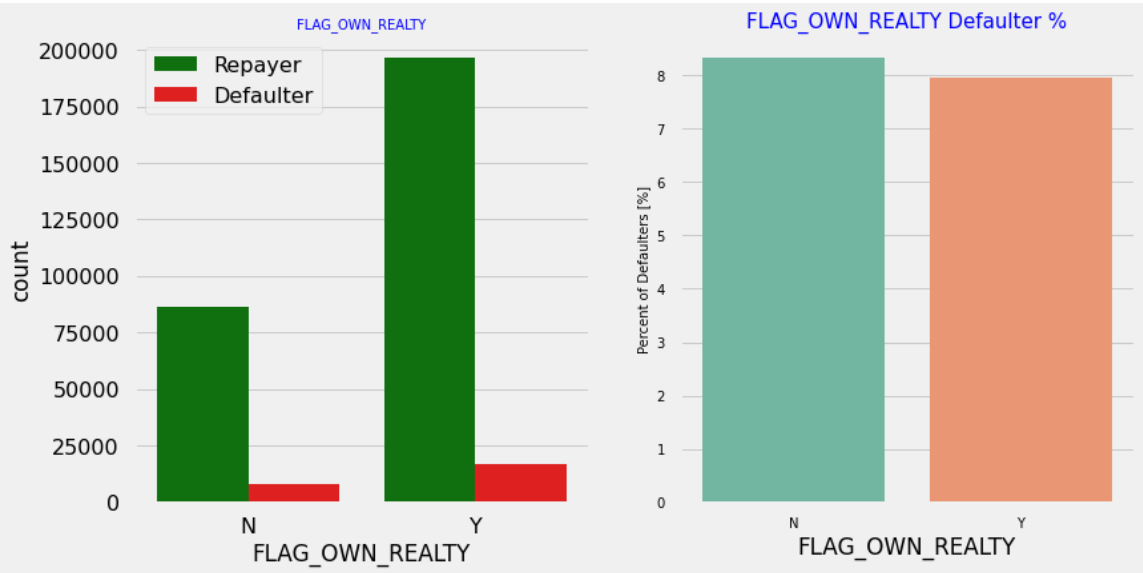
```
# Checking if owning a car is related to loan repayment status
univariate_categorical('FLAG_OWN_CAR')
```



Inferences: Clients who own a car are half in number of the clients who dont own a car. But based on the percentage of deault, there is no correlation between owning a car and loan repayment as in both cases the default percentage is almost same.

In [85]:

```
# Checking if owning a realty is related to loan repayment status
univariate_categorical('FLAG_OWN_REALTY')
```



Inferences: The clients who own real estate are more than double of the ones that don't own. But the defaulting rate of both categories are around the same (~8%). Thus there is no correlation between owning a reality and defaulting the loan.

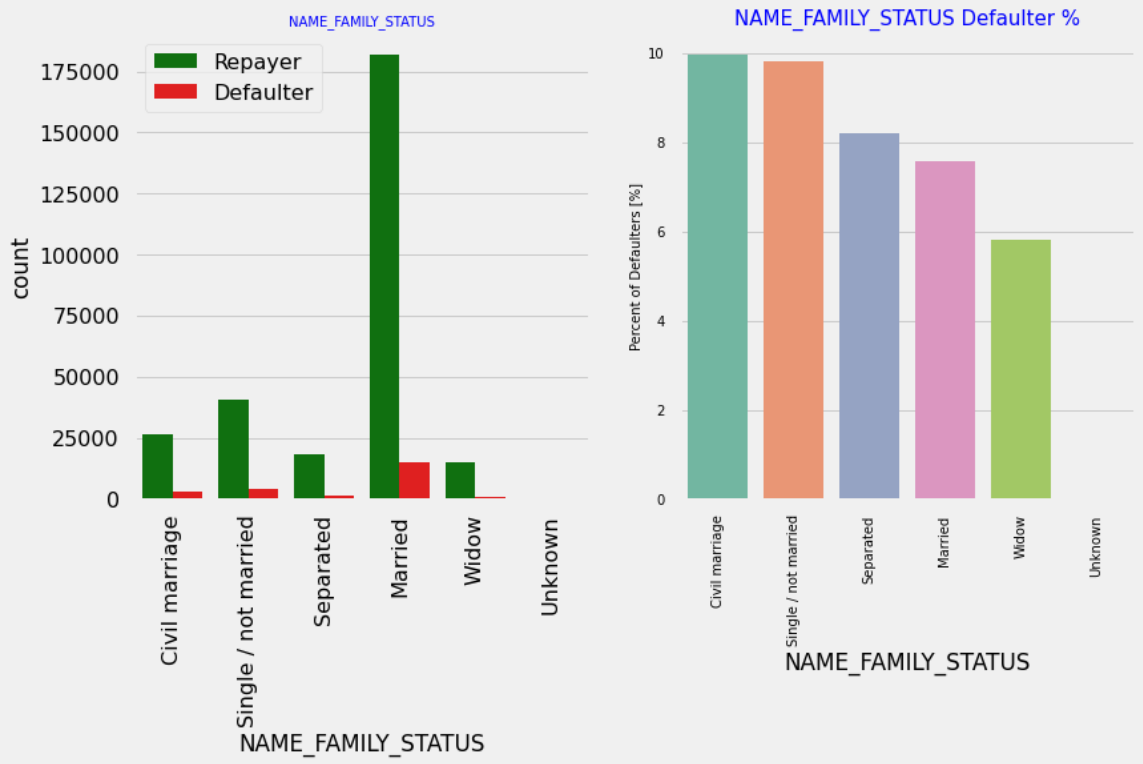
In [86]:

```
# Analyzing Housing Type based on loan repayment status
univariate_categorical("NAME_HOUSING_TYPE",True,True,True)
```

Inferences: Majority of people live in House/apartment People living in office apartments have lowest default rate People living with parents (~11.5%) and living in rented apartments(>12%) have higher probability of defaulting

In [87]:

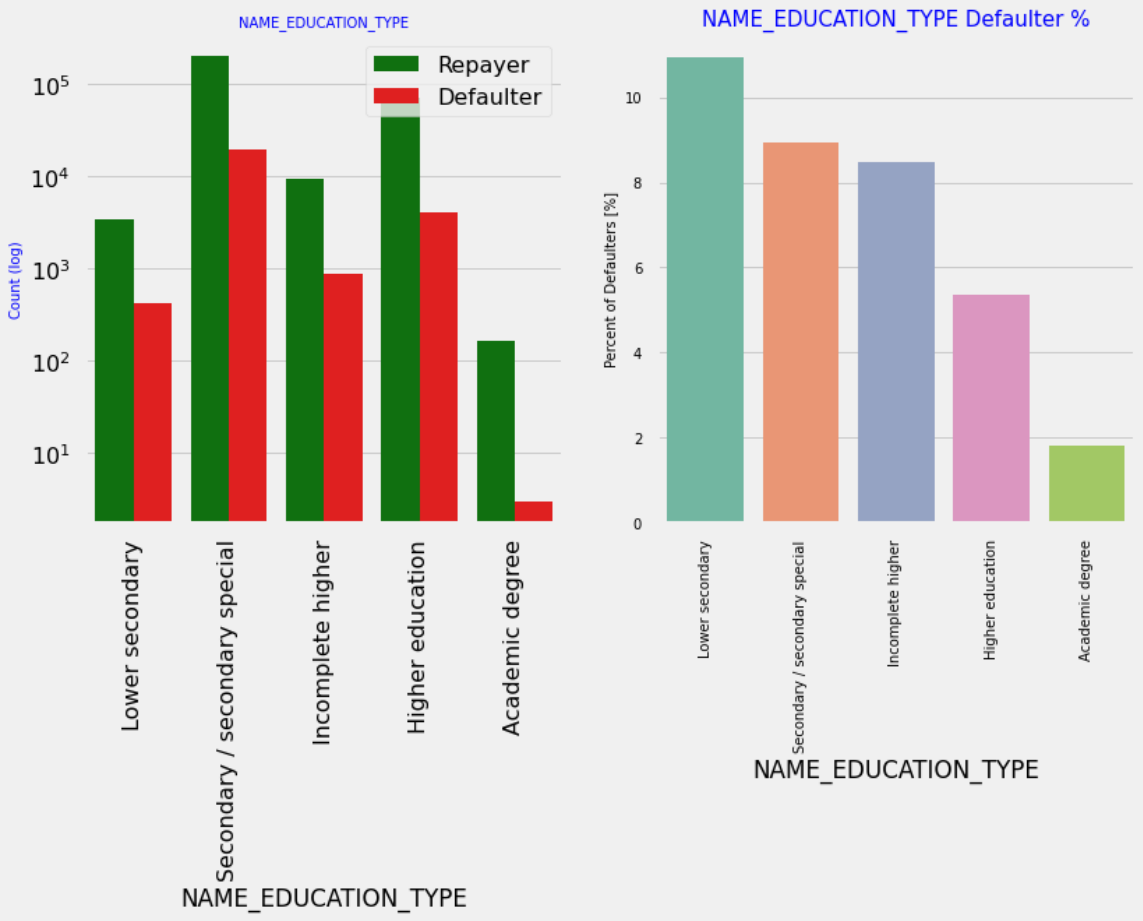
```
# Analyzing Family status based on loan repayment status
univariate_categorical("NAME_FAMILY_STATUS",False,True,True)
```



Inferences: Most of the people who have taken loan are married, followed by Single/not married and civil marriage In terms of percentage of not repayment of loan, Civil marriage has the highest percent of not repayment (10%), with Widow the lowest (exception being Unknown).

In [88]:

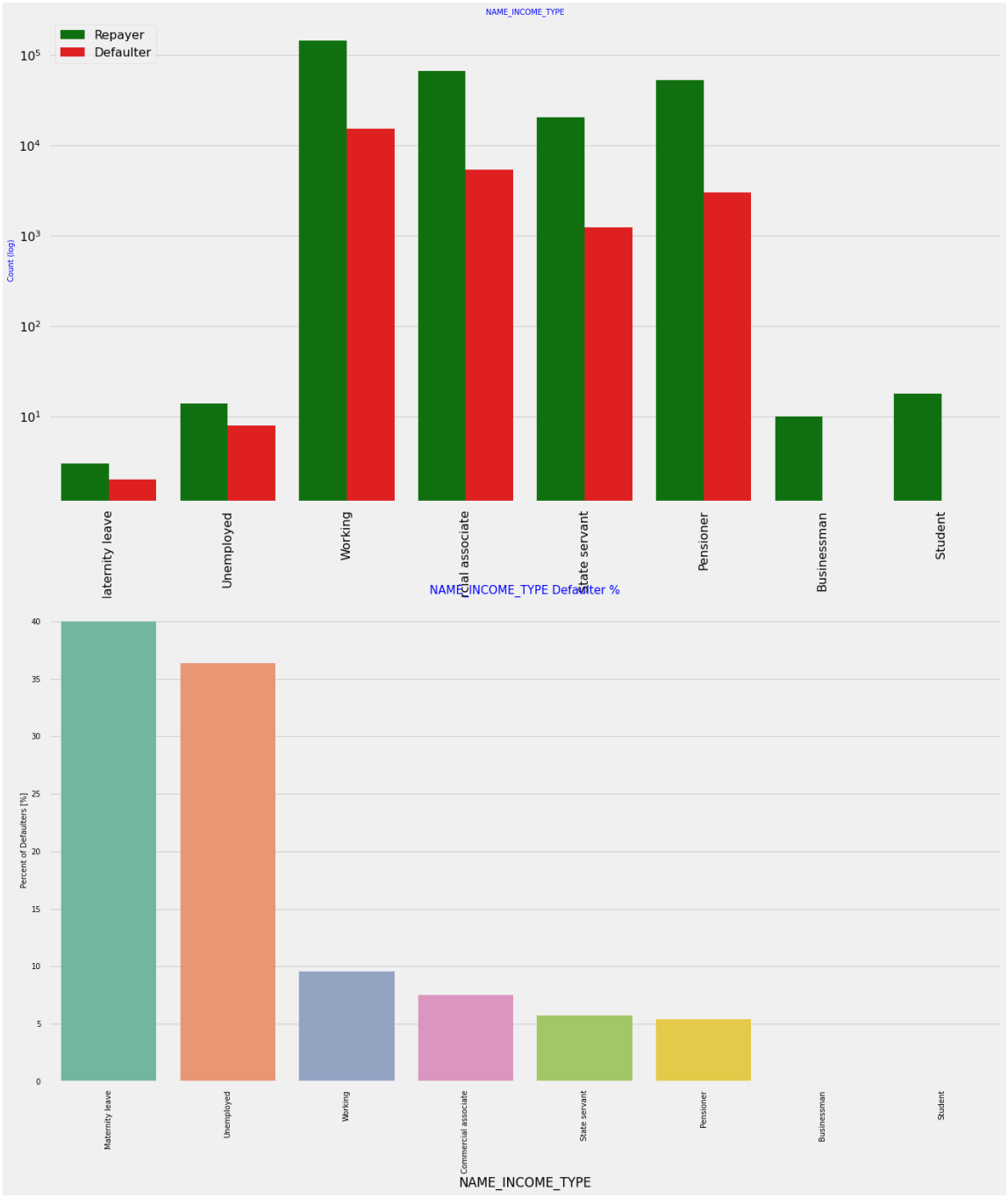
```
# Analyzing Education Type based on loan repayment status
univariate_categorical("NAME_EDUCATION_TYPE",True,True,True)
```



Inferences: Majority of the clients have Secondary / secondary special education, followed by clients with Higher education. Only a very small number having an academic degree The Lower secondary category, although rare, have the largest rate of not returning the loan (11%). The people with Academic degree have less than 2% defaulting rate.

In [89]:

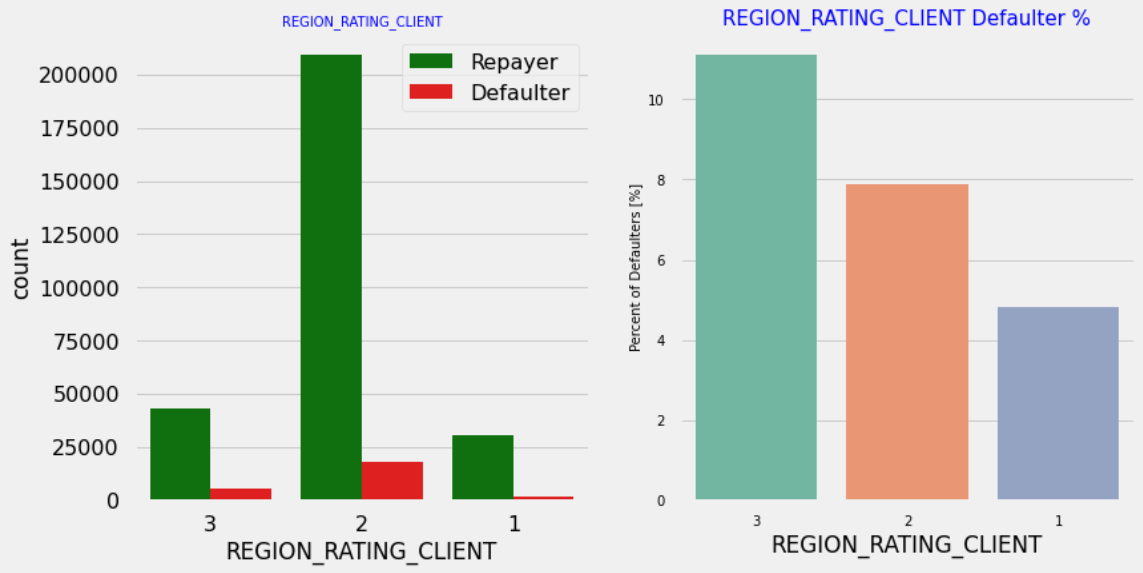
```
# Analyzing Income Type based on loan repayment status
univariate_categorical("NAME_INCOME_TYPE",True,True,False)
```



Inferences: Most of applicants for loans have income type as Working, followed by Commercial associate, Pensioner and State servant. The applicants with the type of income Maternity leave have almost 40% ratio of not returning loans, followed by Unemployed (37%). The rest of types of incomes are under the average of 10% for not returning loans. Student and Businessmen, though less in numbers do not have any default record. Thus these two category are safest for providing loan.

In [90]:

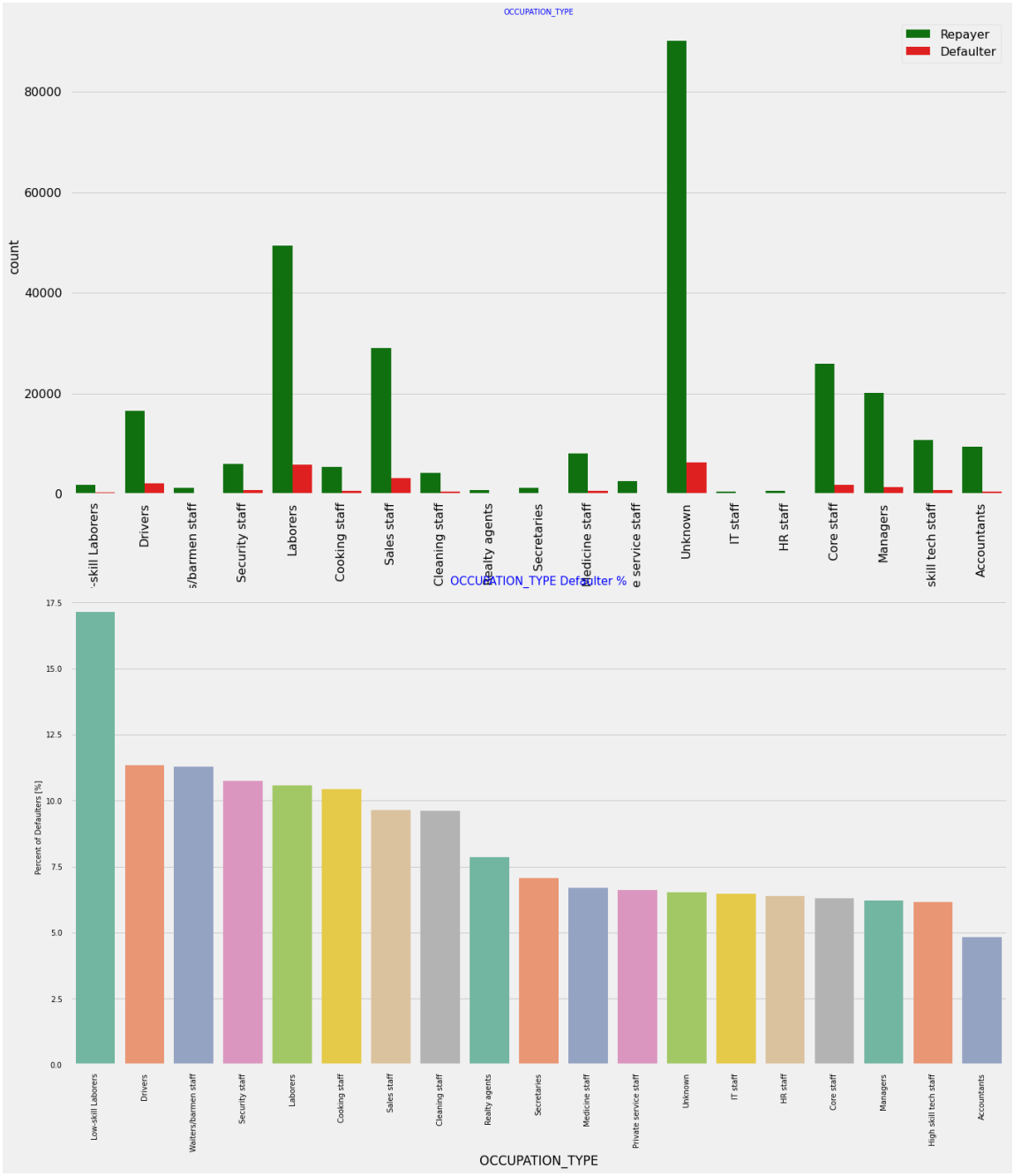
```
# Analyzing Region rating where applicant lives based on loan repayment status
univariate_categorical("REGION_RATING_CLIENT",False,False,True)
```



Inferences: Most of the applicants are living in Region_Rating 2 place. Region Rating 3 has the highest default rate (11%) Applicant living in Region_Rating 1 has the lowest probability of defaulting, thus safer for approving loans

In [91]:

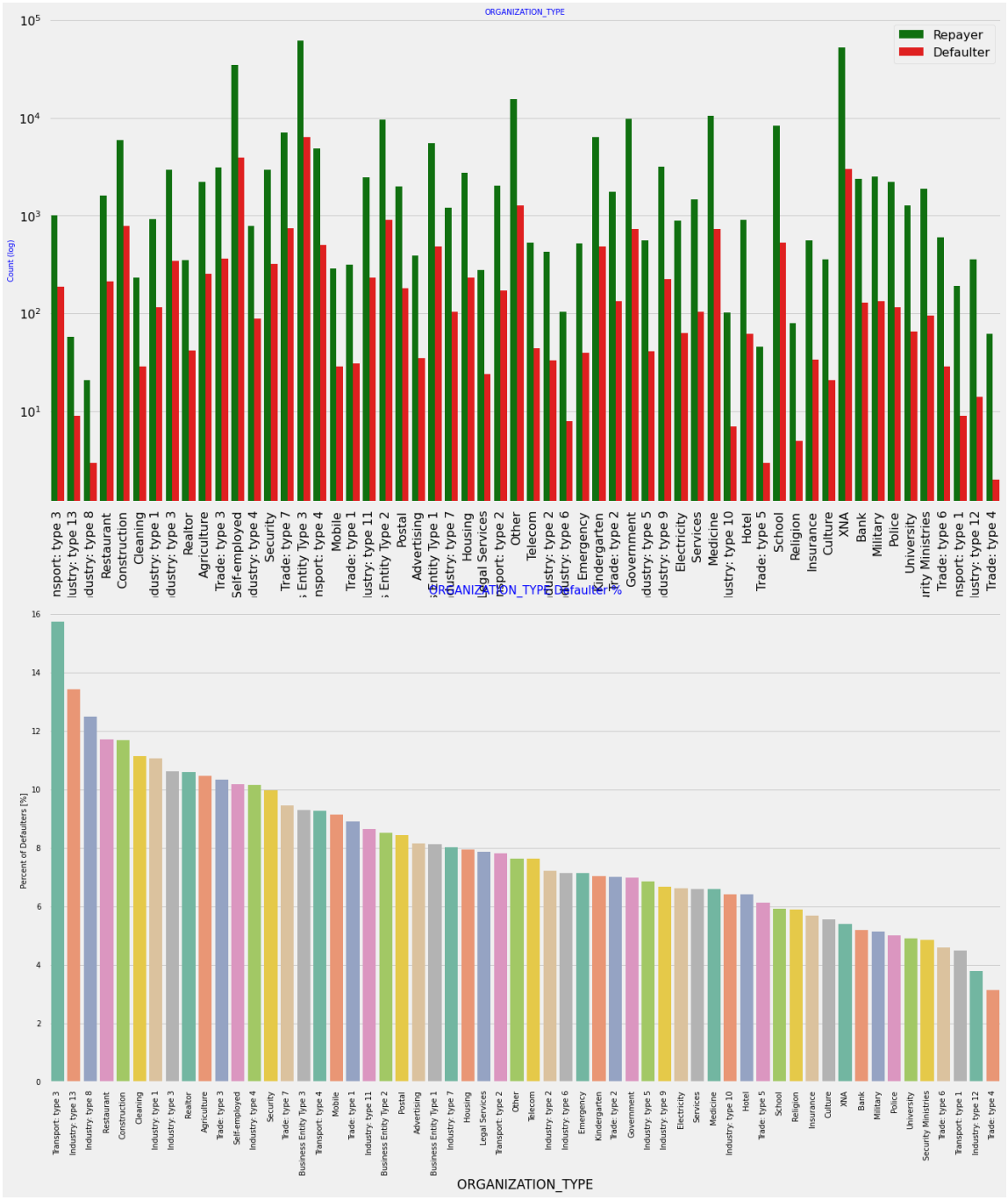
```
# Analyzing Occupation Type where applicant lives based on loan repayment status
univariate_categorical("OCCUPATION_TYPE",False,True,False)
```



Inferences: Most of the loans are taken by Laborers, followed by Sales staff. IT staff take the lowest amount of loans. The category with highest percent of not repaid loans are Low-skill Laborers (above 17%), followed by Drivers and Waiters/barmen staff, Security staff, Laborers and Cooking staff.

In [92]:

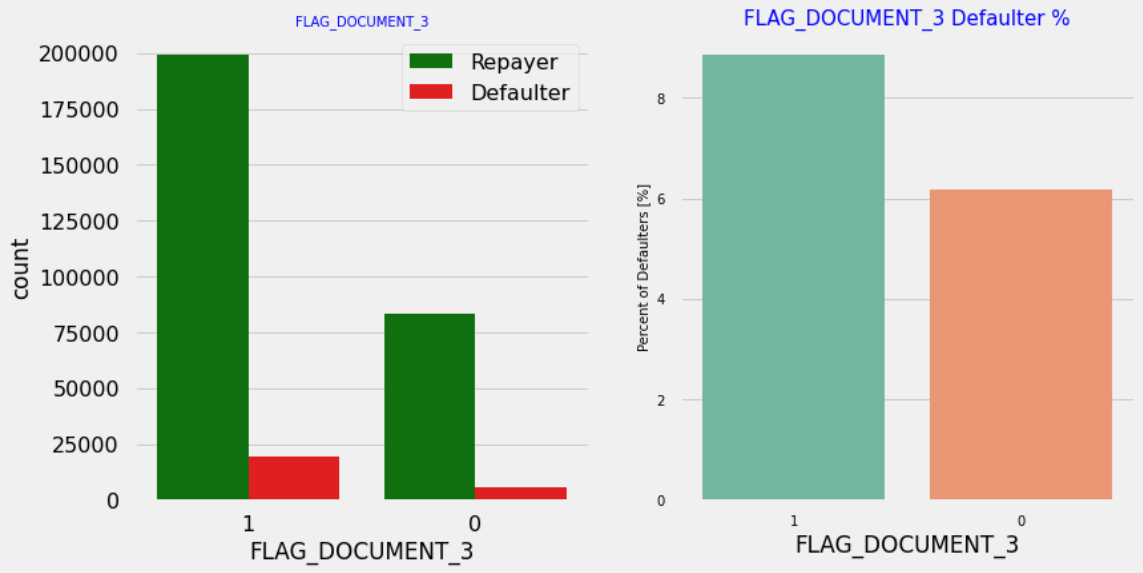
```
# Checking Loan repayment status based on Organization type
univariate_categorical("ORGANIZATION_TYPE",True,True,False)
```



Inferences: Organizations with highest percent of loans not repaid are Transport: type 3 (16%), Industry: type 13 (13.5%), Industry: type 8 (12.5%) and Restaurant (less than 12%). Self employed people have relative high defaulting rate, and thus should be avoided to be approved for loan or provide loan with higher interest rate to mitigate the risk of defaulting. Most of the people application for loan are from Business Entity Type 3 For a very high number of applications, Organization type information is unavailable(XNA) It can be seen that following category of organization type has lesser defaulters thus safer for providing loans: Trade Type 4 and 5 Industry type 8

In [93]:

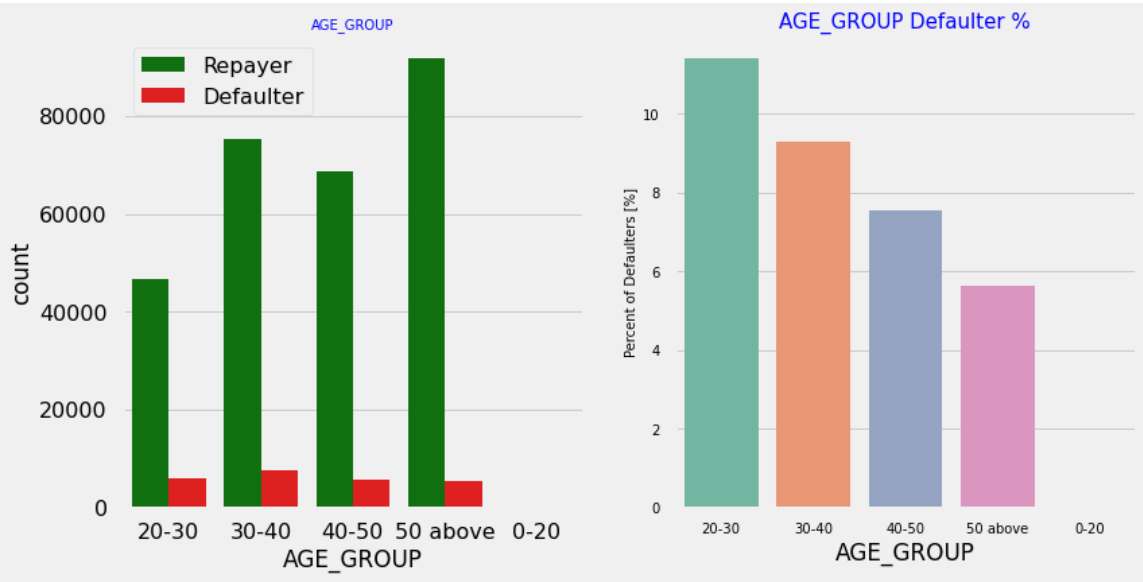
```
# Analyzing Flag_Doc_3 submission status based on loan repayment status
univariate_categorical("FLAG_DOCUMENT_3",False,False,True)
```



Inferences: There is no significant correlation between repayers and defaulters in terms of submitting document 3 as we see even if applicants have submitted the document, they have defaulted a slightly more (~9%) than who have not submitted the document (6%)

In [94]:

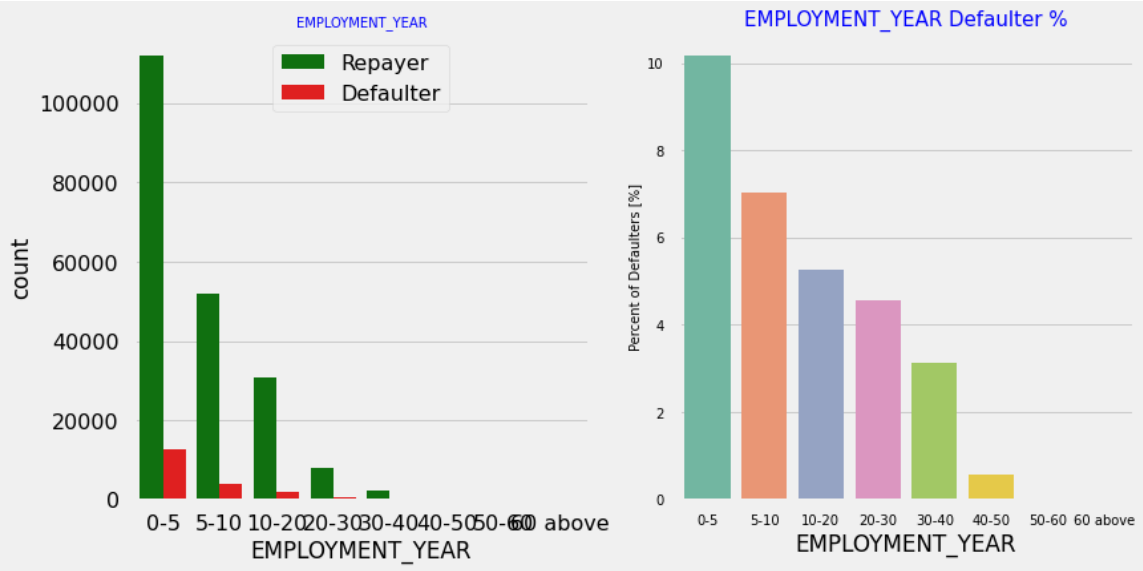
```
# Analyzing Age Group based on loan repayment status
univariate_categorical("AGE_GROUP",False,False,True)
```



Inferences: People in the age group range 20-40 have higher probability of defaulting People above age of 50 have low probability of defailting

In [95]:

```
# Analyzing Employment_Year based on loan repayment status
univariate_categorical("EMPLOYMENT_YEAR",False,False,True)
```



Inferences: Majority of the applicants have been employeed in between 0-5 years. The defaulting rating of this group is also the highest which is 10% With increase of employment year, defaulting rate is gradually decreasing with people having 40+ year experience having less than 1% default rate

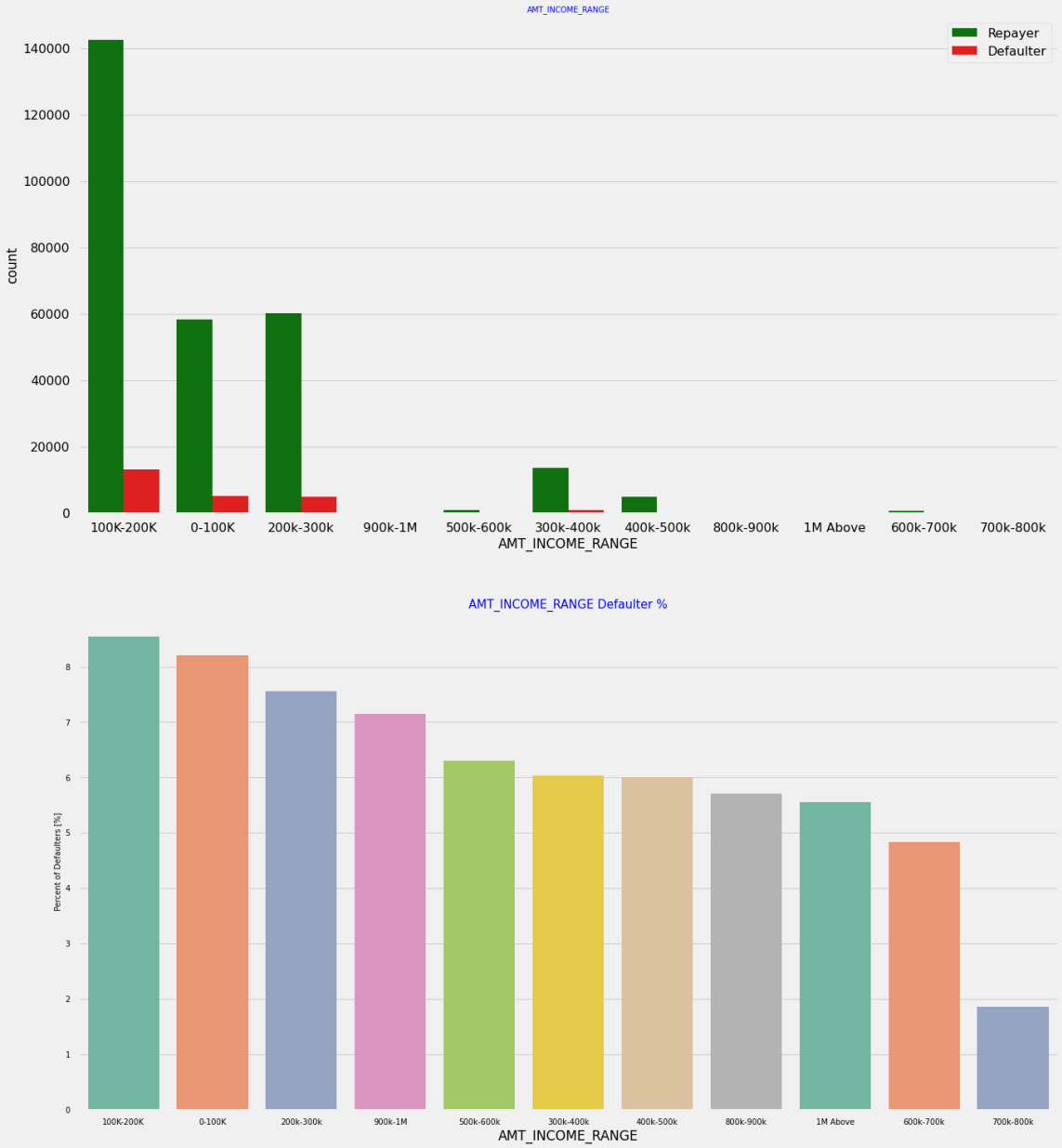
In [96]:

```
# Analyzing Amount_Credit based on loan repayment status
univariate_categorical("AMT_CREDIT_RANGE",False,False,False)
```

Inferences: More than 80% of the loan provided are for amount less than 900,000 People who get loan for 300-600k tend to default more than others.

In [97]:

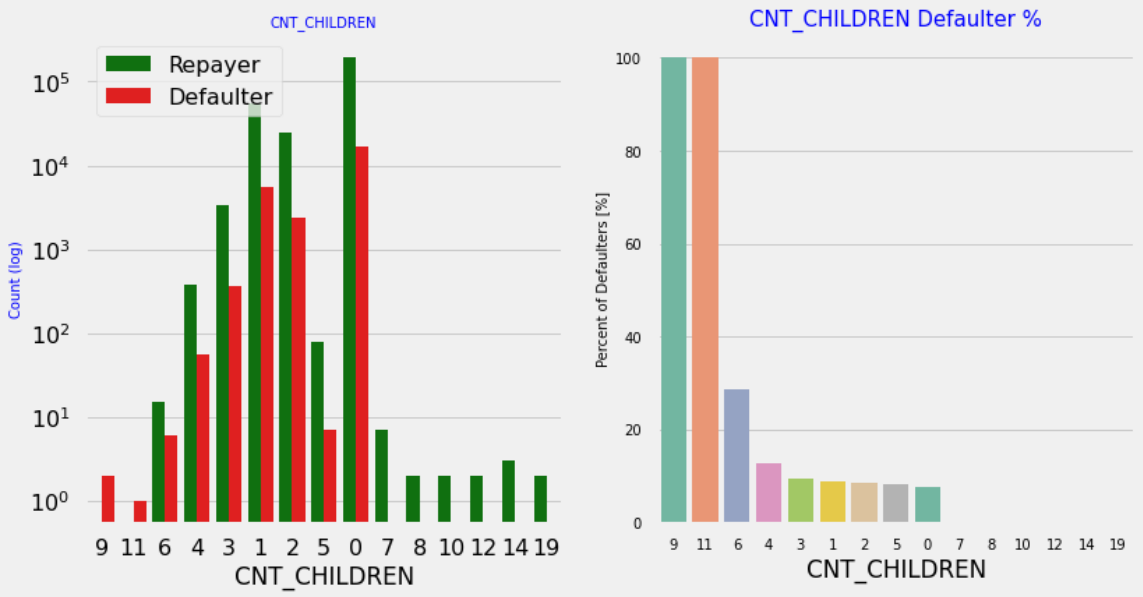
```
# Analyzing Amount_Income Range based on loan repayment status
univariate_categorical("AMT_INCOME_RANGE",False,False,False)
```



Inferences: 90% of the applications have Income total less than 300,000 Application with Income less than 300,000 has high probability of defaulting Applicant with Income more than 700,000 are less likely to default

In [98]:

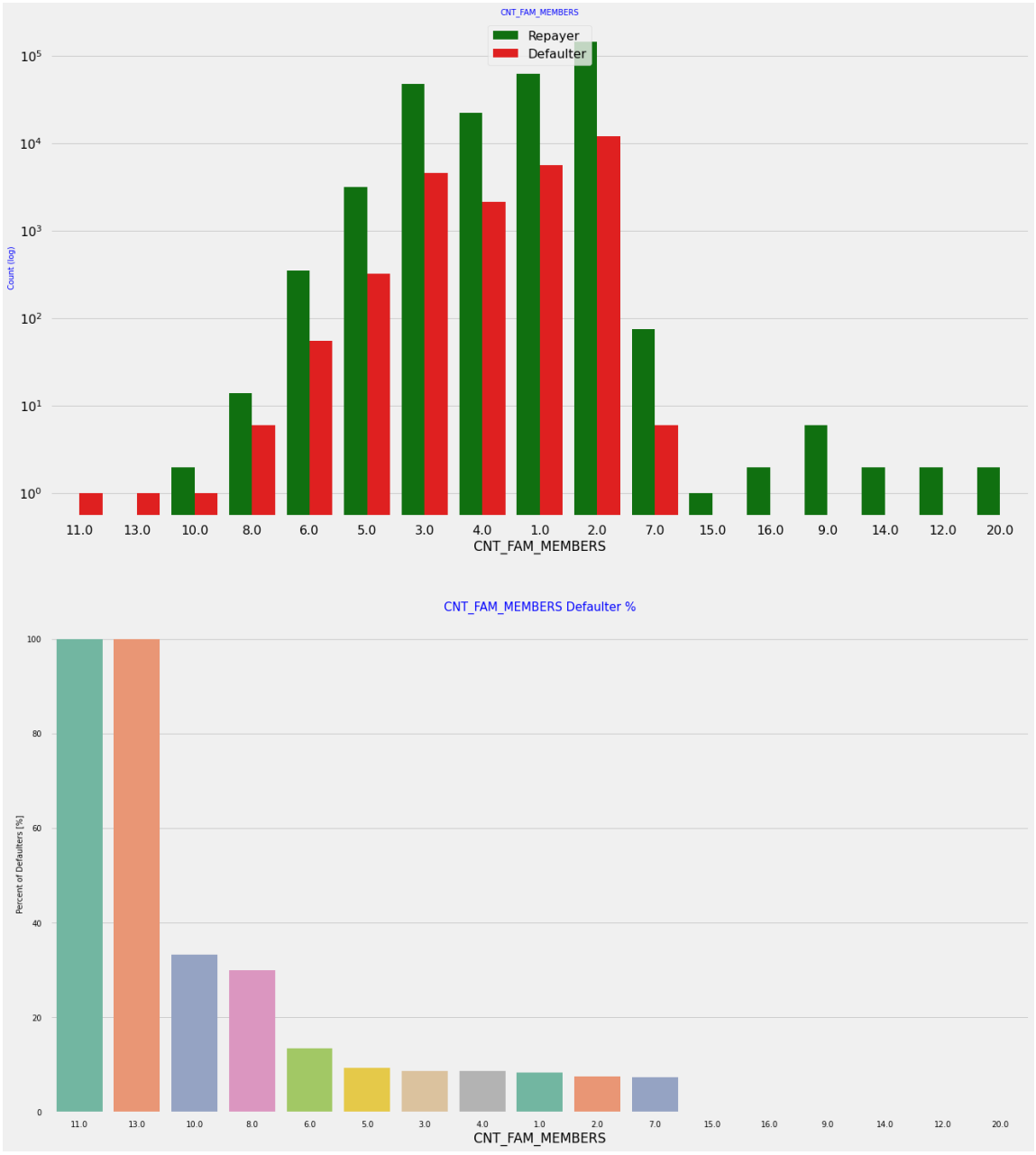
```
# Analyzing Number of children based on loan repayment status
univariate_categorical("CNT_CHILDREN",True)
```



Inferences: Most of the applicants do not have children Very few clients have more than 3 children. Client who have more than 4 children has a very high default rate with child count 9 and 11 showing 100% default rate

In [99]:

```
# Analyzing Number of family members based on loan repayment status
univariate_categorical("CNT_FAM_MEMBERS",True, False, False)
```



Inferences: Family member follows the same trend as children where having more family members increases the risk of defaulting

In [100]:

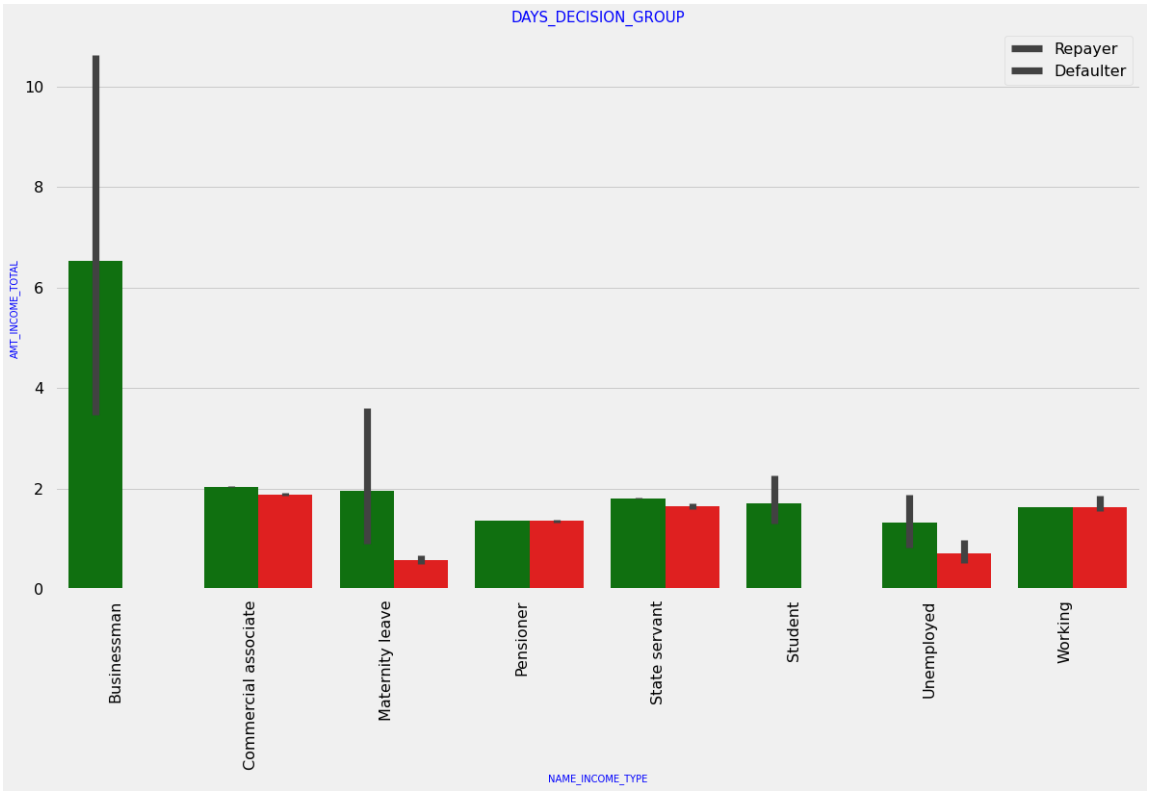
```
application_d.groupby('NAME_INCOME_TYPE')['AMT_INCOME_TOTAL'].describe()
```

Out[100]:

NAME_INCOME_TYPE	count	mean	std	min	25%	50%	75%	ma
Businessman	10.0	6.525000	6.272260	1.8000	2.250	4.9500	8.43750	22.500
Commercial associate	71617.0	2.029553	1.479742	0.2655	1.350	1.8000	2.25000	180.000
Maternity leave	5.0	1.404000	1.268569	0.4950	0.675	0.9000	1.35000	3.600
Pensioner	55362.0	1.364013	0.766503	0.2565	0.900	1.1700	1.66500	22.500
State servant	21703.0	1.797380	1.008806	0.2700	1.125	1.5750	2.25000	31.500
Student	18.0	1.705000	1.066447	0.8100	1.125	1.5750	1.78875	5.625
Unemployed	22.0	1.105364	0.880551	0.2655	0.540	0.7875	1.35000	3.375
Working	158774.0	1.631699	3.075777	0.2565	1.125	1.3500	2.02500	1170.000

In [107]:

```
# Income type vs Income Amount Range
bivariate_bar("NAME_INCOME_TYPE","AMT_INCOME_TOTAL",application_d,"TARGET",(18,10))
```



Inferences: It can be seen that business man's income is the highest and the estimated range with default 95% confidence level seem to indicate that the income of a business man could be in the range of slightly close to 4 lakhs and slightly above 10 lakhs

In [ ]:

```
# Numeric Variables Analysis
```

In [ ]:

```
# Bifurcating the applicationDF dataframe based on Target value 0 and 1 for correlation
```

In [108]:

```
application_d.columns
```

Out[108]:

```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_
OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CRED
IT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYP
E', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'REG
ION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_REGISTRATIO
N', 'DAYS_ID_PUBLISH', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'REGION_RATIN
G_CLIENT', 'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START', 'H
OUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WOR
K_REGION', 'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY', 'REG_C
ITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_TYPE', 'OBS_3
0_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCL
E', 'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_
3', 'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CR
EDIT_BUREAU_WEEK',
       'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_
CREDIT_BUREAU_YEAR', 'AMT_INCOME_RANGE', 'AMT_CREDIT_RANGE', 'AGE', 'AGE_G
ROUP', 'YEARS_EMPLOYED', 'EMPLOYMENT_YEAR'],
      dtype='object')
```

In [109]:

```
# Bifurcating the applicationDF dataframe based on Target value 0 and 1 for correlation
cols_for_correlation = ['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_R
                        'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY',
                        'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'N
                        'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
                        'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'OCCUPATION_TYPE', 'CNT_
                        'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START', 'HO
                        'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION', 'LIV
                        'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_N
                        'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LA
                        'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_
                        'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_R


Repayer_d = application_d.loc[application_d['TARGET']==0, cols_for_correlation] # Repaye
Defaulter_d = application_d.loc[application_d['TARGET']==1, cols_for_correlation] # Defa
```

In [ ]:

```
#Correlation between numeric variable
```

In [110]:

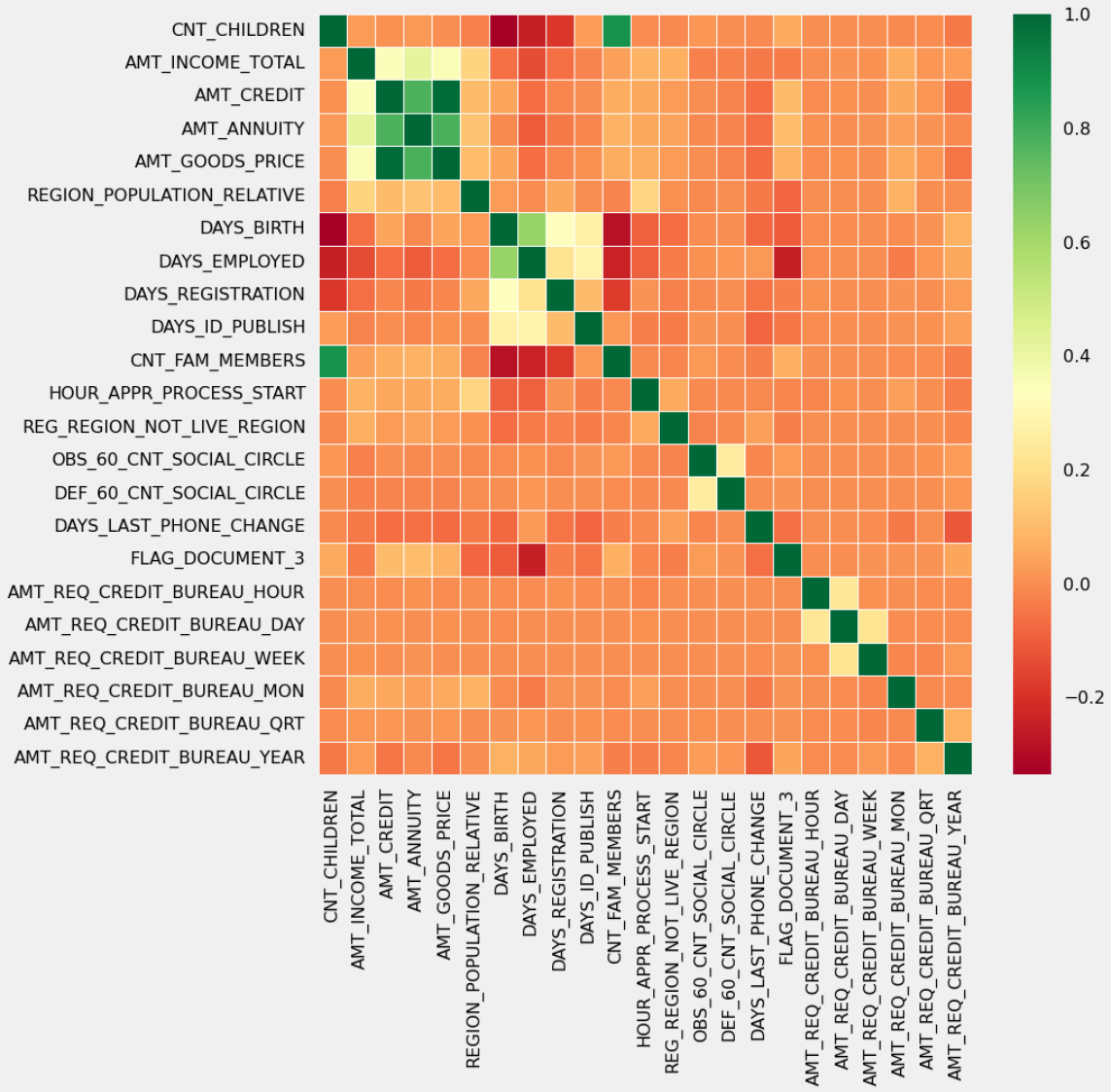
```
# Getting the top 10 correlation for the Repayers data
corr_repayer = Repayer_d.corr()
corr_repayer = corr_repayer.where(np.triu(np.ones(corr_repayer.shape),k=1).astype(np.boo
corr_df_repayer = corr_repayer.unstack().reset_index()
corr_df_repayer.columns =['VAR1','VAR2','Correlation']
corr_df_repayer.dropna(subset = ["Correlation"], inplace = True)
corr_df_repayer["Correlation"]=corr_df_repayer["Correlation"].abs()
corr_df_repayer.sort_values(by='Correlation', ascending=False, inplace=True)
corr_df_repayer.head(10)
```

Out[110]:

	VAR1	VAR2	Correlation
94	AMT_GOODS_PRICE	AMT_CREDIT	0.987250
230	CNT_FAM_MEMBERS	CNT_CHILDREN	0.878571
95	AMT_GOODS_PRICE	AMT_ANNUITY	0.776686
71	AMT_ANNUITY	AMT_CREDIT	0.771309
167	DAYS_EMPLOYED	DAYS_BIRTH	0.626114
70	AMT_ANNUITY	AMT_INCOME_TOTAL	0.418953
93	AMT_GOODS_PRICE	AMT_INCOME_TOTAL	0.349462
47	AMT_CREDIT	AMT_INCOME_TOTAL	0.342799
138	DAYS_BIRTH	CNT_CHILDREN	0.336966
190	DAYS_REGISTRATION	DAYS_BIRTH	0.333151

In [111]:

```
fig = plt.figure(figsize=(12,12))
ax = sns.heatmap(Repayer_d.corr(), cmap="RdYlGn",annot=False,linewidth =1)
```



Inferences: Correlating factors amongst repayers: Credit amount is highly correlated with amount of goods price loan annuity total income We can also see that repayers have high correlation in number of days employed.

In [112]:

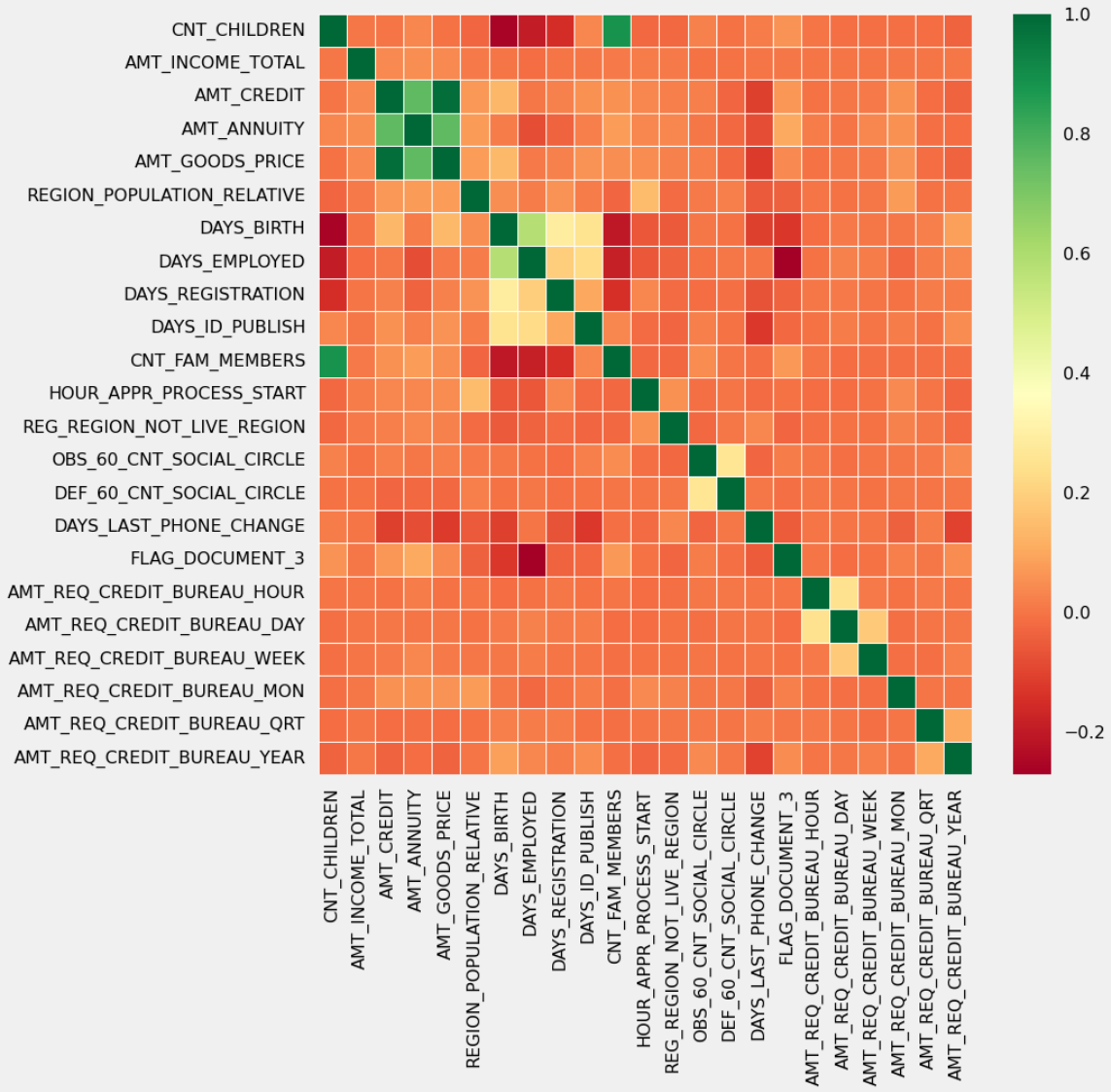
```
# Getting the top 10 correlation for the Defaulter data
corr_Defaulter = Defaulter_d.corr()
corr_Defaulter = corr_Defaulter.where(np.triu(np.ones(corr_Defaulter.shape),k=1).astype(
corr_df_Defaulter = corr_Defaulter.unstack().reset_index()
corr_df_Defaulter.columns =['VAR1','VAR2','Correlation']
corr_df_Defaulter.dropna(subset = ["Correlation"], inplace = True)
corr_df_Defaulter["Correlation"]=corr_df_Defaulter["Correlation"].abs()
corr_df_Defaulter.sort_values(by='Correlation', ascending=False, inplace=True)
corr_df_Defaulter.head(10)
```

Out[112]:

	VAR1	VAR2	Correlation
94	AMT_GOODS_PRICE	AMT_CREDIT	0.983103
230	CNT_FAM_MEMBERS	CNT_CHILDREN	0.885484
95	AMT_GOODS_PRICE	AMT_ANNUITY	0.752699
71	AMT_ANNUITY	AMT_CREDIT	0.752195
167	DAYS_EMPLOYED	DAYS_BIRTH	0.582185
190	DAYS_REGISTRATION	DAYS_BIRTH	0.289114
375	FLAG_DOCUMENT_3	DAYS_EMPLOYED	0.272169
335	DEF_60_CNT_SOCIAL_CIRCLE	OBS_60_CNT_SOCIAL_CIRCLE	0.264159
138	DAYS_BIRTH	CNT_CHILDREN	0.259109
213	DAYS_ID_PUBLISH	DAYS_BIRTH	0.252863

In [113]:

```
fig = plt.figure(figsize=(12,12,))
ax = sns.heatmap(Defaulter_d.corr(),cmap ="RdYlGn",annot =False,linewidth =1)
```



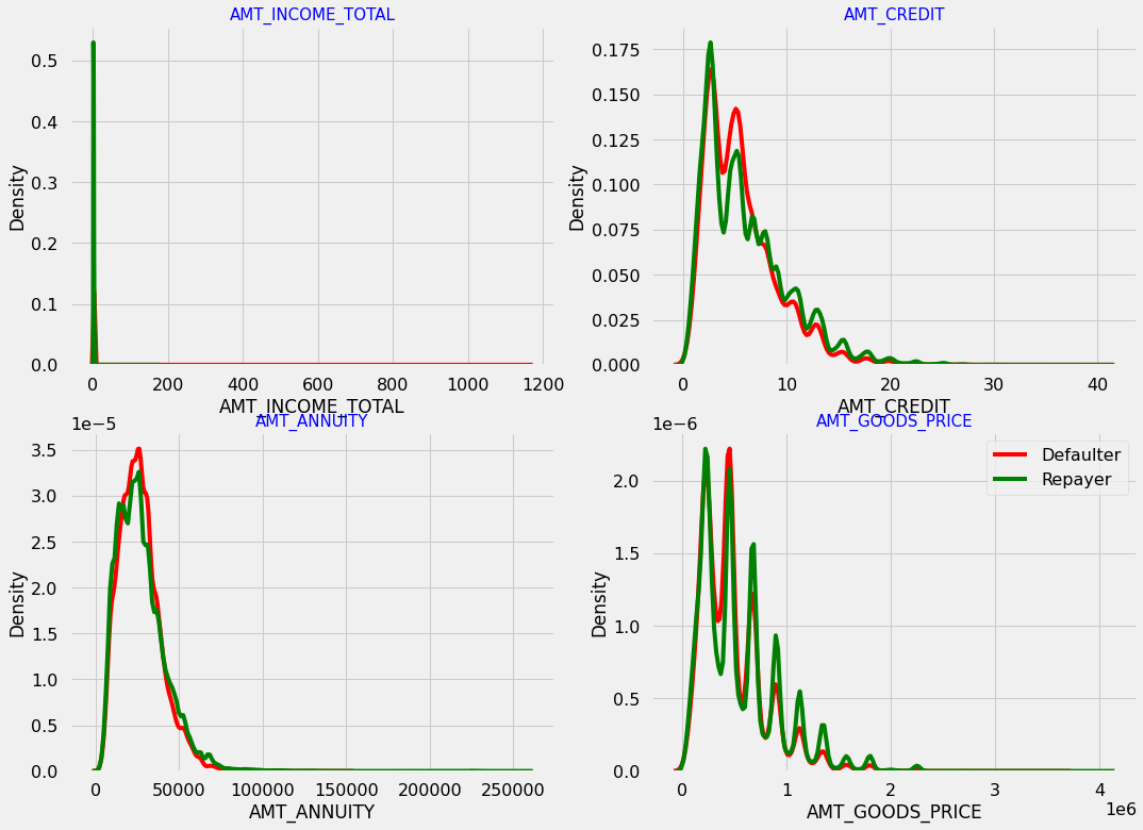
Inferences: Credit amount is highly correlated with amount of goods price which is same as repayers. But the loan annuity correlation with credit amount has slightly reduced in defaulters(0.75) when compared to repayers(0.77) We can also see that repayers have high correlation in number of days employed(0.62) when compared to defaulters(0.58). There is a severe drop in the correlation between total income of the client and the credit amount(0.038) amongst defaulters whereas it is 0.342 among repayers. Days_birth and number of children correlation has reduced to 0.259 in defaulters when compared to 0.337 in repayers. There is a slight increase in defaulted to observed count in social circle among defaulters(0.264) when compared to repayers(0.254)

In [ ]:

```
# Numerical Univariate Analysis
```

In [114]:

```
# Plotting the numerical columns related to amount as distribution plot to see density
amount = application_d[[ 'AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY', 'AMT_GOODS_PRICE

fig = plt.figure(figsize=(16,12))

for i in enumerate(amount):
    plt.subplot(2,2,i[0]+1)
    sns.distplot(Defaulter_d[i[1]], hist=False, color='r',label ="Defaulter")
    sns.distplot(Repayer_d[i[1]], hist=False, color='g', label ="Repayer")
    plt.title(i[1], fontdict={'fontsize' : 15, 'fontweight' : 5, 'color' : 'Blue'})

plt.legend()

plt.show()
```



Inferences: Most no of loans are given for goods price below 10 lakhs Most people pay annuity below 50000 for the credit loan Credit amount of the loan is mostly less then 10 lakhs The repayers and defaulters distribution overlap in all the plots and hence we cannot use any of these variables in isolation to make a decision

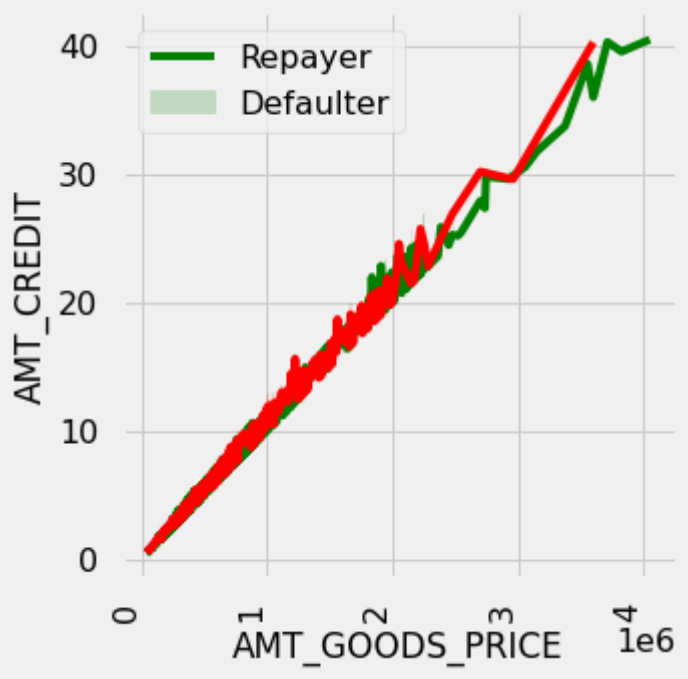
In [ ]:

```
# Numerical Bivariate Analysis
```

In [115]:

```
# Checking the relationship between Goods price and credit and comparing with loan repayr
bivariate_rel('AMT_GOODS_PRICE','AMT_CREDIT',application_d,"TARGET", "line", ['g','r'],
```

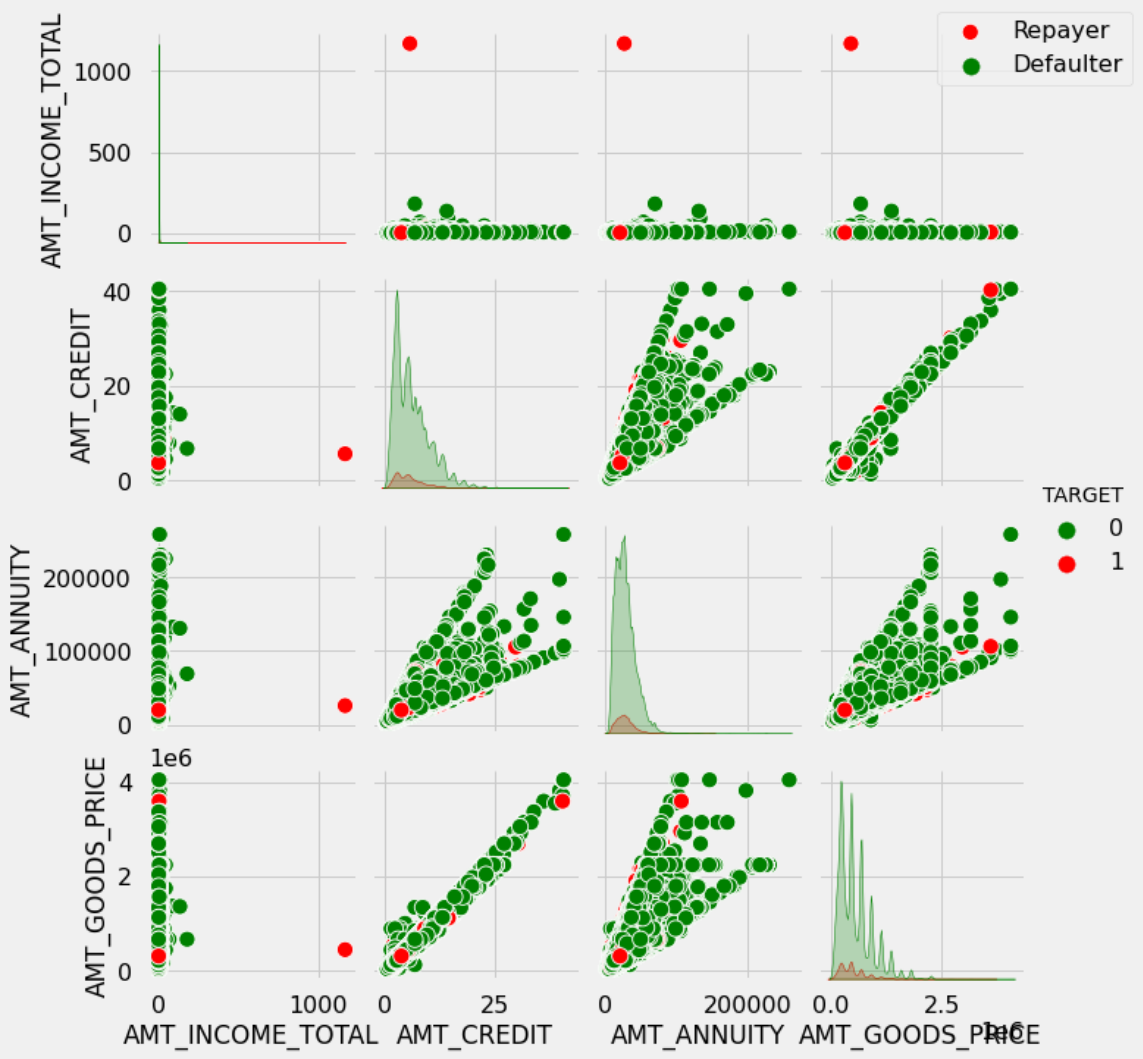
```
<Figure size 1080x432 with 0 Axes>
```



Inferences: When the credit amount goes beyond 3M, there is an increase in defaulters.

In [116]:

```
# Plotting pairplot between amount variable to draw reference against loan repayment sta
amount = application_d[[ 'AMT_INCOME_TOTAL','AMT_CREDIT',
                         'AMT_ANNUITY', 'AMT_GOODS_PRICE','TARGET']]
amount = amount[(amount["AMT_GOODS_PRICE"].notnull()) & (amount["AMT_ANNUITY"].notnull()
ax= sns.pairplot(amount,hue="TARGET",palette=["g","r"])
ax.fig.legend(labels=['Repayer','Defaulter'])
plt.show()
```



Inferences:

When amt_annuity >15000 amt_goods_price> 3M, there is a lesser chance of defaulters AMT_CREDIT and AMT_GOODS_PRICE are highly correlated as based on the scatterplot where most of the data are consolidated in form of a line There are very less defaulters for AMT_CREDIT >3M Inferences related to distribution plot has been already mentioned in previous distplot graphs inferences section

In [ ]:

```
# Merged Dataframes Analysis
```

In [117]:

```
#merge both the dataframe on SK_ID_CURR with Inner Joins
loan_process_d = pd.merge(application_d, previous_d, how='inner', on='SK_ID_CURR')
loan_process_d.head()
```

Out[117]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FL
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	

In [118]:

```
#Checking the details of the merged dataframe
loan_process_d.shape
```

Out[118]:

```
(1413701, 74)
```

In [119]:

```
# Checking the element count of the dataframe
loan_process_d.size
```

Out[119]:

```
104613874
```

In [120]:

```
# checking the columns and column types of the dataframe
loan_process_d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1413701 entries, 0 to 1413700
Data columns (total 74 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   SK_ID_CURR                   1413701 non-null  int64
 1   TARGET                       1413701 non-null  int64
 2   NAME_CONTRACT_TYPE_x         1413701 non-null  category
 3   CODE_GENDER                  1413701 non-null  category
 4   FLAG_OWN_CAR                 1413701 non-null  category
 5   FLAG_OWN_REALTY              1413701 non-null  category
 6   CNT_CHILDREN                 1413701 non-null  int64
 7   AMT_INCOME_TOTAL             1413701 non-null  float64
 8   AMT_CREDIT_x                 1413701 non-null  float64
 9   AMT_ANNUITY_x                1413608 non-null  float64
 10  AMT_GOODS_PRICE_x            1412493 non-null  float64
 11  NAME_TYPE_SUITE              1413701 non-null  category
 12  NAME_INCOME_TYPE             1413701 non-null  category
 13  NAME_EDUCATION_TYPE          1413701 non-null  category
 14  NAME_FAMILY_STATUS           1413701 non-null  category
 15  NAME_HOUSING_TYPE            1413701 non-null  category
 16  REGION_POPULATION_RELATIVE   1413701 non-null  float64
 17  DAYS_BIRTH                   1413701 non-null  int64
 18  DAYS_EMPLOYED                1413701 non-null  int64
 19  DAYS_REGISTRATION            1413701 non-null  float64
 20  DAYS_ID_PUBLISH              1413701 non-null  int64
 21  OCCUPATION_TYPE              1413701 non-null  category
 22  CNT_FAM_MEMBERS              1413701 non-null  float64
 23  REGION_RATING_CLIENT         1413701 non-null  category
 24  REGION_RATING_CLIENT_W_CITY  1413701 non-null  category
 25  WEEKDAY_APPR_PROCESS_START   1413701 non-null  category
 26  HOUR_APPR_PROCESS_START      1413701 non-null  int64
 27  REG_REGION_NOT_LIVE_REGION   1413701 non-null  int64
 28  REG_REGION_NOT_WORK_REGION   1413701 non-null  category
 29  LIVE_REGION_NOT_WORK_REGION  1413701 non-null  category
 30  REG_CITY_NOT_LIVE_CITY       1413701 non-null  category
 31  REG_CITY_NOT_WORK_CITY       1413701 non-null  category
 32  LIVE_CITY_NOT_WORK_CITY      1413701 non-null  category
 33  ORGANIZATION_TYPE            1413701 non-null  category
 34  OBS_30_CNT_SOCIAL_CIRCLE     1410555 non-null  float64
 35  DEF_30_CNT_SOCIAL_CIRCLE     1410555 non-null  float64
 36  OBS_60_CNT_SOCIAL_CIRCLE     1410555 non-null  float64
 37  DEF_60_CNT_SOCIAL_CIRCLE     1410555 non-null  float64
 38  DAYS_LAST_PHONE_CHANGE       1413701 non-null  float64
 39  FLAG_DOCUMENT_3              1413701 non-null  int64
 40  AMT_REQ_CREDIT_BUREAU_HOUR   1250074 non-null  float64
 41  AMT_REQ_CREDIT_BUREAU_DAY    1250074 non-null  float64
 42  AMT_REQ_CREDIT_BUREAU_WEEK   1250074 non-null  float64
 43  AMT_REQ_CREDIT_BUREAU_MON    1250074 non-null  float64
 44  AMT_REQ_CREDIT_BUREAU_QRT    1250074 non-null  float64
 45  AMT_REQ_CREDIT_BUREAU_YEAR   1250074 non-null  float64
 46  AMT_INCOME_RANGE             1413024 non-null  category
 47  AMT_CREDIT_RANGE             1413701 non-null  category
 48  AGE                          1413701 non-null  int64
 49  AGE_GROUP                    1413701 non-null  category
 50  YEARS_EMPLOYED               1413701 non-null  int64
 51  EMPLOYMENT_YEAR              1032756 non-null  category
 52  SK_ID_PREV                   1413701 non-null  int64
 53  NAME_CONTRACT_TYPE_y         1413701 non-null  category
 54  AMT_ANNUITY_y                1413701 non-null  float64
 55  AMT_APPLICATION              1413701 non-null  float64
```

```
 56  AMT_CREDIT_y                 1413700 non-null  float64
 57  AMT_GOODS_PRICE_y            1413701 non-null  float64
 58  NAME_CASH_LOAN_PURPOSE       1413701 non-null  category
 59  NAME_CONTRACT_STATUS         1413701 non-null  category
 60  DAYS_DECISION                1413701 non-null  int64
 61  NAME_PAYMENT_TYPE            1413701 non-null  category
 62  CODE_REJECT_REASON           1413701 non-null  category
 63  NAME_CLIENT_TYPE             1413701 non-null  category
 64  NAME_GOODS_CATEGORY          1413701 non-null  category
 65  NAME_PORTFOLIO               1413701 non-null  category
 66  NAME_PRODUCT_TYPE            1413701 non-null  category
 67  CHANNEL_TYPE                 1413701 non-null  category
 68  SELLERPLACE_AREA             1413701 non-null  int64
 69  NAME_SELLER_INDUSTRY         1413701 non-null  category
 70  CNT_PAYMENT                  1413701 non-null  float64
 71  NAME_YIELD_GROUP             1413701 non-null  category
 72  PRODUCT_COMBINATION          1413388 non-null  category
 73  DAYS_DECISION_GROUP          1413701 non-null  category
dtypes: category(37), float64(23), int64(14)
memory usage: 459.8 MB
```

In [121]:

```
# Checking merged dataframe numerical columns statistics
loan_process_d.describe()
```

Out[121]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT_x
count	1.413701e+06	1.413701e+06	1.413701e+06	1.413701e+06	1.413701e+06
mean	2.784813e+05	8.655296e-02	4.048933e-01	1.733160e+00	5.875537e+00
std	1.028118e+05	2.811789e-01	7.173454e-01	1.985734e+00	3.849173e+00
min	1.000020e+05	0.000000e+00	0.000000e+00	2.565000e-01	4.500000e-01
25%	1.893640e+05	0.000000e+00	0.000000e+00	1.125000e+00	2.700000e+00
50%	2.789920e+05	0.000000e+00	0.000000e+00	1.575000e+00	5.084955e+00
75%	3.675560e+05	0.000000e+00	1.000000e+00	2.070000e+00	8.079840e+00
max	4.562550e+05	1.000000e+00	1.900000e+01	1.170000e+03	4.050000e+01

In [122]:

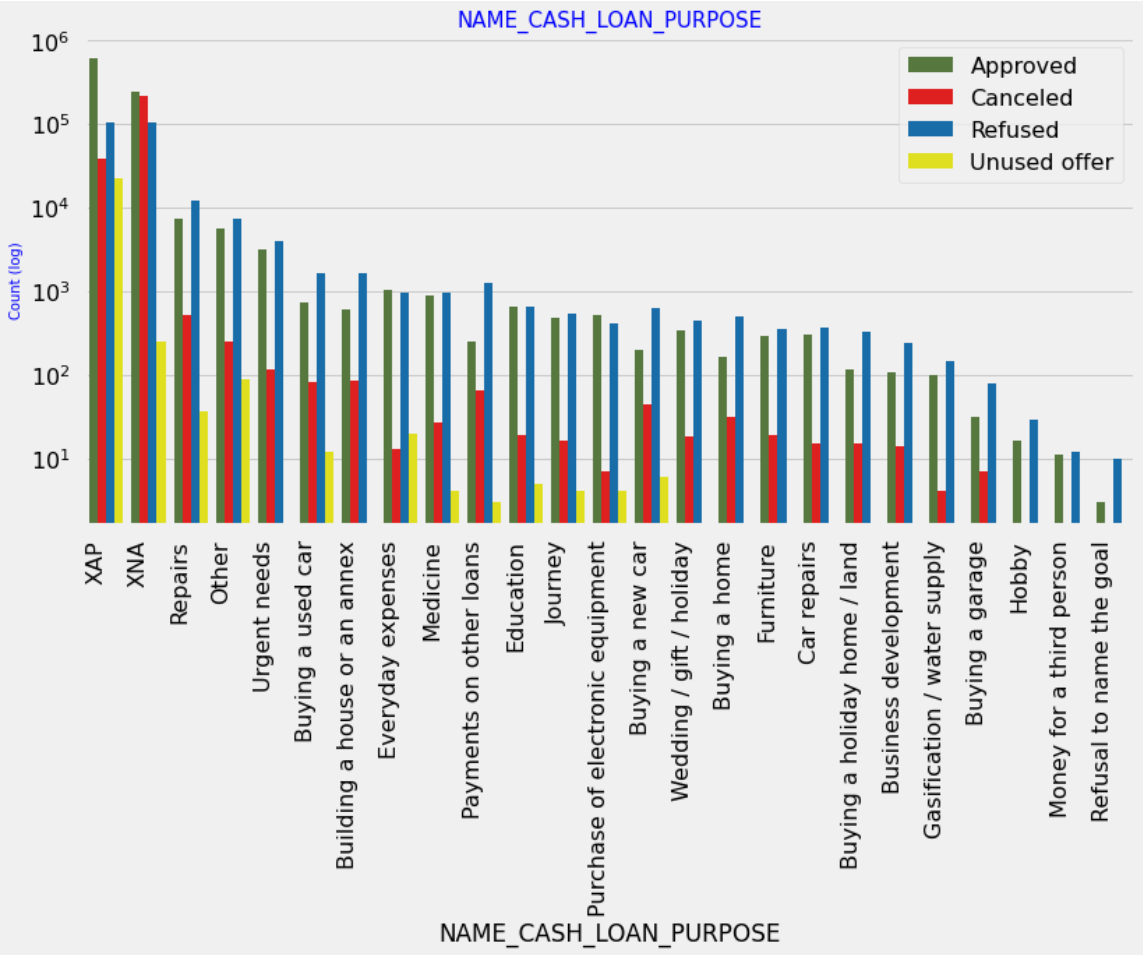
```
# Bifurcating the applicationDF dataframe based on Target value 0 and 1 for correlation

L0 = loan_process_d[loan_process_d['TARGET']==0] # Repayers
L1 = loan_process_d[loan_process_d['TARGET']==1] # Defaulters
```

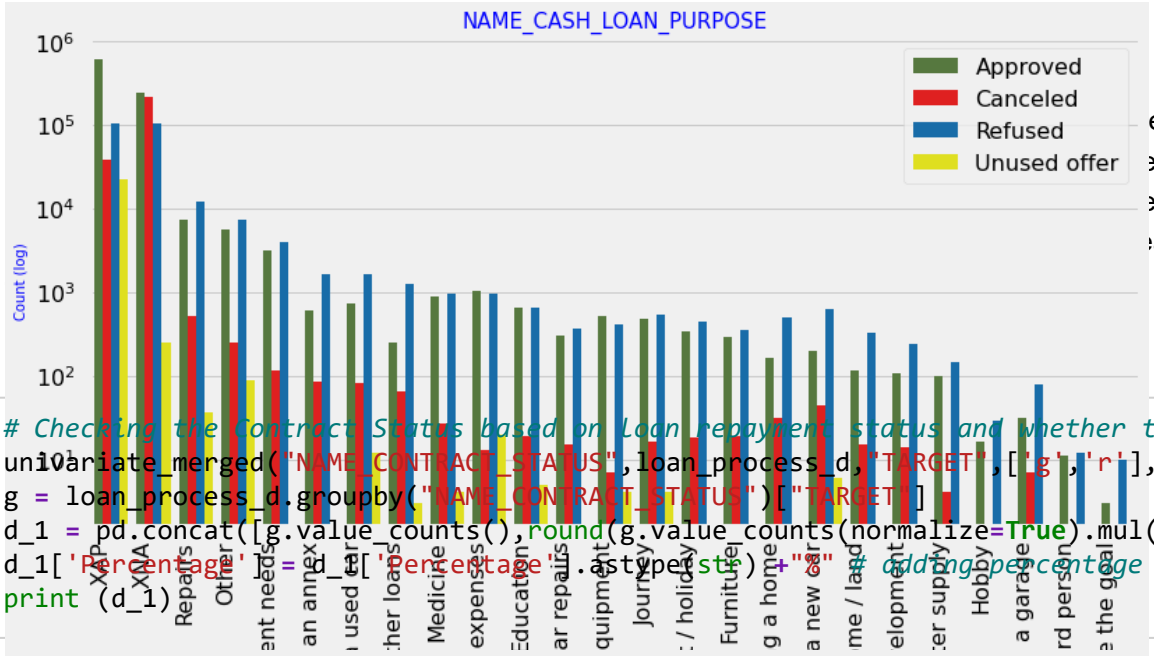
In [ ]:

```
#Plotting Contract Status vs purpose of the loan:
```

In [139]:

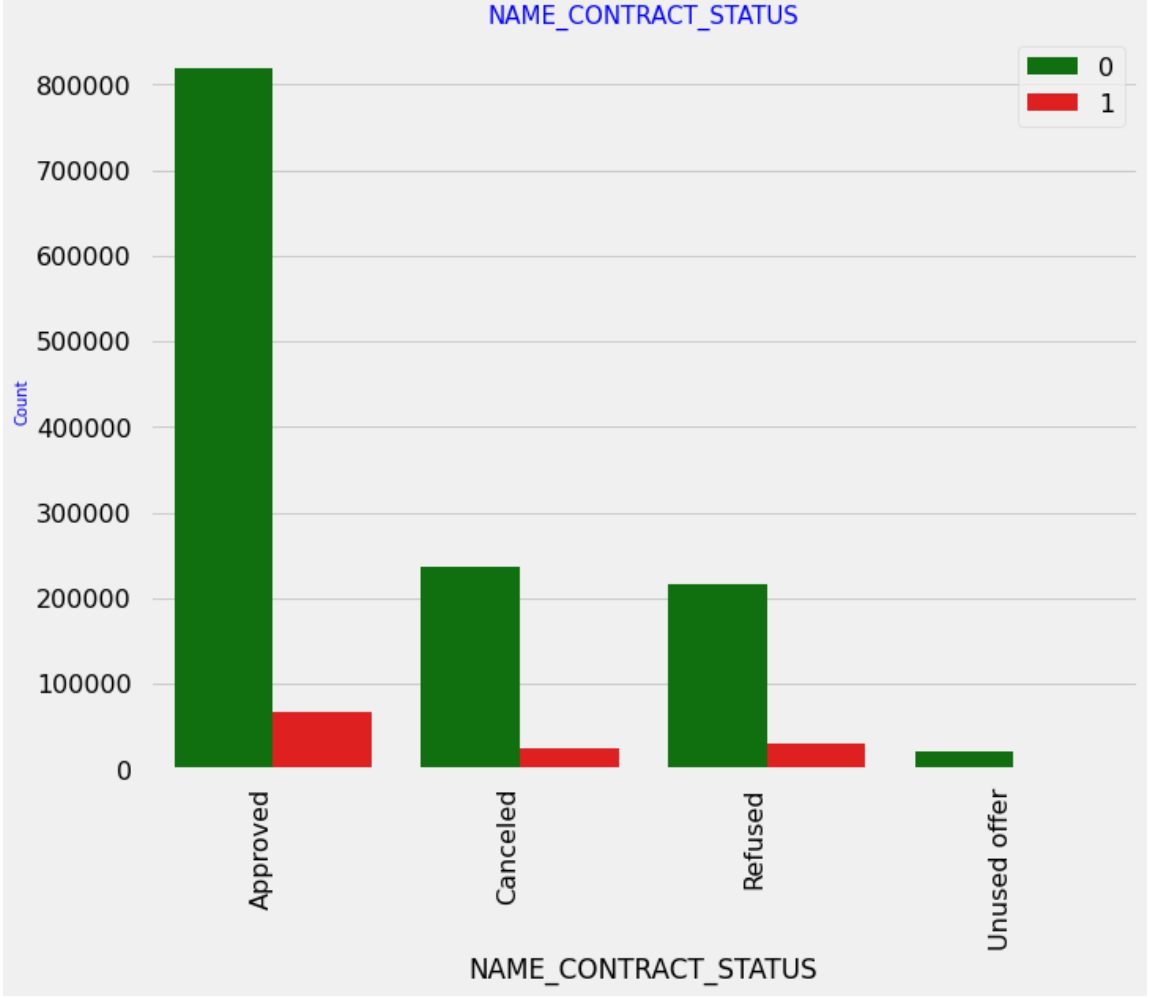
```
univariate_merged("NAME_CASH_LOAN_PURPOSE",L0,"NAME_CONTRACT_STATUS",["#548235","#FF0000

univariate_merged("NAME_CASH_LOAN_PURPOSE",L1,"NAME_CONTRACT_STATUS",["#548235","#FF0000
```





e purpose of
ed by bank or
en as high risk
easible by the

```
# Checking the Contract Status based on loan repayment status and whether there is any bu
univariate_merged("NAME_CONTRACT_STATUS",loan_process_d,"TARGET",['g','r'],False,(10,8))
g = loan_process_d.groupby("NAME_CONTRACT_STATUS")["TARGET"]
d_1 = pd.concat([g.value_counts(),round(g.value_counts(normalize=True).mul(100),2)],axis
d_1['Percentage'] = d_1['Percentage'].astype(str) +"%" # adding percentage symbol in the
print (d_1)
```

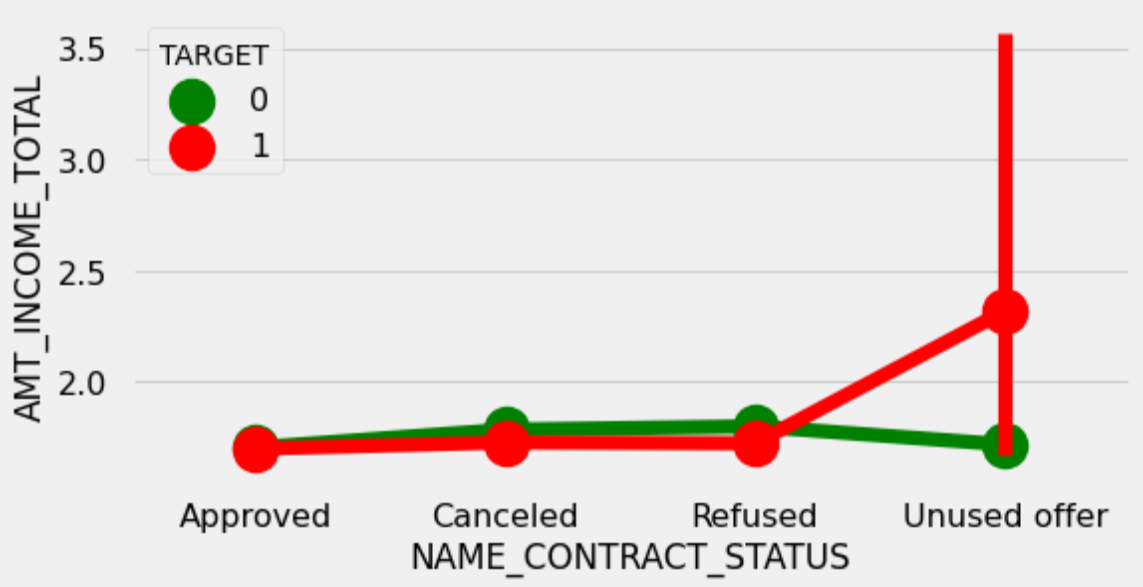


		Counts	Percentage
NAME_CONTRACT_STATUS	TARGET		
Approved	0	818856	92.41%
	1	67243	7.59%
Canceled	0	235641	90.83%
	1	23800	9.17%
Refused	0	215952	88.0%
	1	29438	12.0%
Unused offer	0	20892	91.75%
	1	1879	8.25%

Inferences: 90% of the previously cancelled client have actually repayed the loan. Revisiting the interest rates would increase business opoortunity for these clients 88% of the clients who have been previously refused a loan has payed back the loan in current case. Refual reason should be recorded for further analysis as these clients would turn into potential repaying customer.

In [136]:

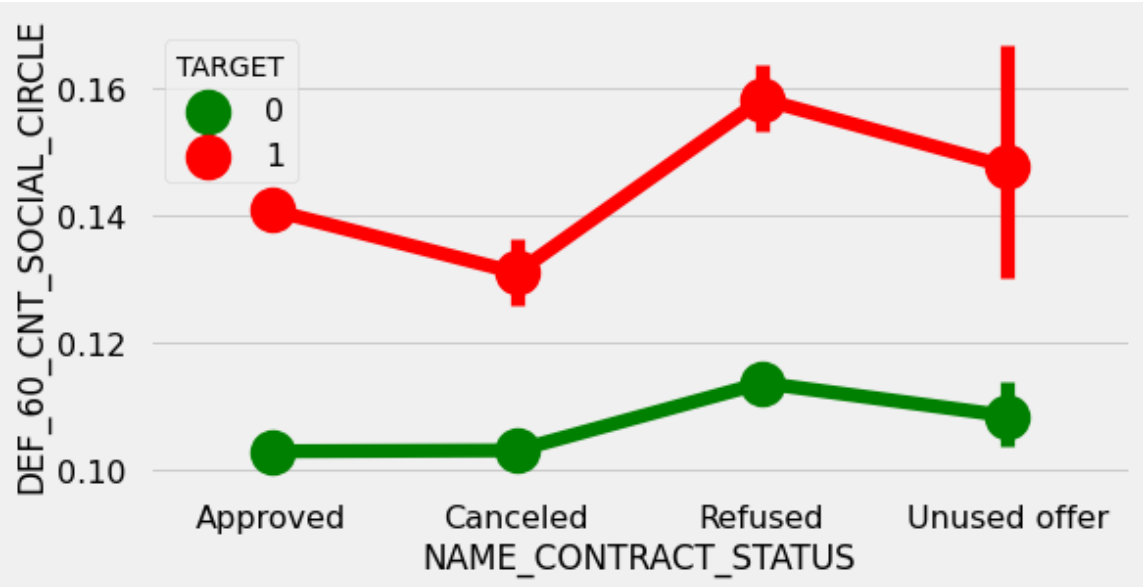
```
# plotting the relationship between income total and contact status
merged_pointplot("NAME_CONTRACT_STATUS",'AMT_INCOME_TOTAL')
```



Inferences: The point plot show that the people who have not used offer earlier have defaulted even when there average income is higher than others

In [137]:

```
merged_pointplot("NAME_CONTRACT_STATUS",'DEF_60_CNT_SOCIAL_CIRCLE')
```



Inferences: Clients who have average of 0.13 or higher DEF_60_CNT_SOCIAL_CIRCLE score tend to default more and hence client's social circle has to be analysed before providing the loan.

Conclusions

After analysing the datasets, there are few attributes of a client with which the bank would be able to identify if they will repay the loan or not. The analysis is consised as below with the contributing factors and categorization:

Decisive Factor whether an applicant will be Repayer:

1. NAME_EDUCATION_TYPE: Academic degree has less defaults.
2. NAME_INCOME_TYPE: Student and Businessmen have no defaults.
3. REGION_RATING_CLIENT: RATING 1 is safer.
4. ORGANIZATION_TYPE: Clients with Trade Type 4 and 5 and Industry type 8 have defaulted less than 3%
5. DAYS_BIRTH: People above age of 50 have low probability of defaulting
6. DAYS_EMPLOYED: Clients with 40+ year experience having less than 1% default rate
7. AMT_INCOME_TOTAL:Applicant with Income more than 700,000 are less likely to default
8. NAME_CASH_LOAN_PURPOSE: Loans bought for Hobby, Buying garage are being repayed mostly.
9. CNT_CHILDREN: People with zero to two children tend to repay the loans.

In [ ]: