In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import itertools
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
pd.set_option('display.max_rows',500)
pd.set_option('display.max_columns',500)
pd.set_option('display.width',100)
```

In [3]:
```
app_data=pd.read_csv(r'S:\DOCS\dataset\application_data.csv')
```

In [4]:
```
app_data
```

Out[4]:

	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 [5]:
```
app_data.shape
```

Out[5]:
```
(307511, 122)
```

In [6]:
```
1 app_data.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 [7]:
```
1 app_data.describe()
```

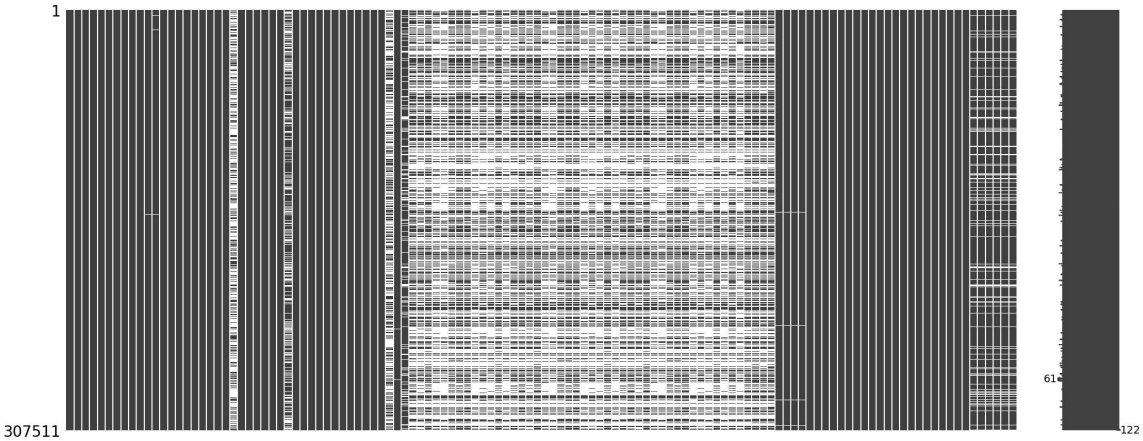
Out[7]:

	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	3
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	2

In [8]:
```
1 import missingno as mn
```

In [9]:
```
1 mn.matrix(app_data)
```

Out[9]: <Axes: >



In [10]:
```
1 # many values are missing in dataset
```

In [11]:
```
1 round(app_data.isnull().sum()/app_data.shape[0] *100)
```

Out[11]:
```
SK_ID_CURR                    0.0
TARGET                        0.0
NAME_CONTRACT_TYPE            0.0
CODE_GENDER                   0.0
FLAG_OWN_CAR                  0.0
FLAG_OWN_REALTY               0.0
CNT_CHILDREN                  0.0
AMT_INCOME_TOTAL              0.0
AMT_CREDIT                    0.0
AMT_ANNUITY                   0.0
AMT_GOODS_PRICE               0.0
NAME_TYPE_SUITE               0.0
NAME_INCOME_TYPE              0.0
NAME_EDUCATION_TYPE           0.0
NAME_FAMILY_STATUS            0.0
NAME_HOUSING_TYPE             0.0
REGION_POPULATION_RELATIVE    0.0
DAYS_BIRTH                    0.0
DAYS_EMPLOYED                 0.0
DAYS_REGISTRATION             0.0
```

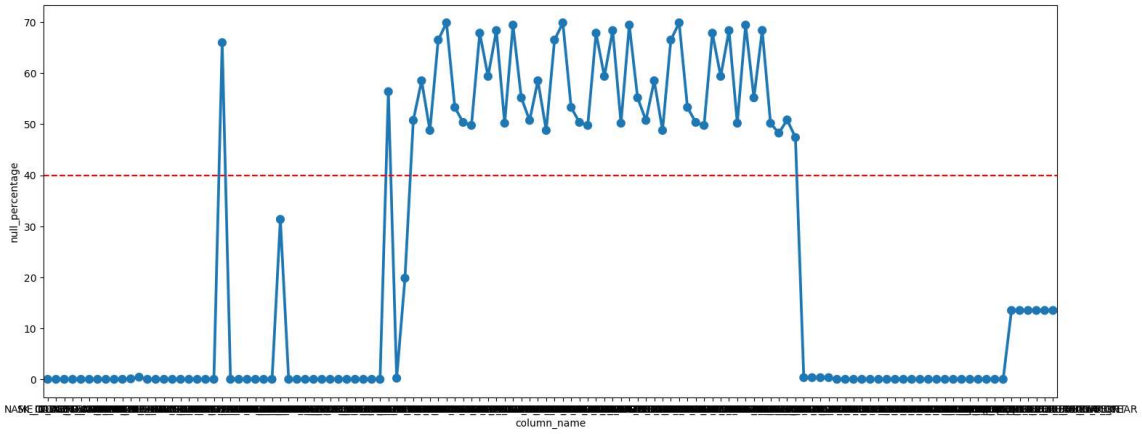
In [12]:
```
1 # many variables have missing value nore than 40 %
```

In [13]:
```
1 null_data= (app_data.isnull().sum()/app_data.shape[0] *100).reset_index
```

In [14]:
```
1 null_data.columns=['column_name','null_percentage']
```

In [15]:
```
fig=plt.figure(figsize=(18,7))
ax=sns.pointplot(data=null_data, x='column_name', y='null_percentage')
ax.axhline(40, ls='--',color='red')
```

Out[15]: <matplotlib.lines.Line2D at 0x166806e82d0>



In [16]:
```
# the column point above red line have missing more than 40%
```

In [17]:
```
null_value_40 = null_data[null_data['null_percentage']>=40]
```

In [18]:
```
len(null_value_40)
```

Out[18]: 49

In [19]:
```
source=app_data[['EXT_SOURCE_1','EXT_SOURCE_2','EXT_SOURCE_3','TARGET']
```

In [20]:
```
1 ax=sns.heatmap(source.corr(),annot=True)
```

In [21]:
```
1 # no correlation between ext_sources and target
```

In [22]:
```
1 unwanted_app= null_value_40['column_name'].tolist()+['EXT_SOURCE_2','EX
```

In [23]:
```
1 len(unwanted_app)
```

Out[23]: 51

In [24]:
```
flag_doc=['FLAG_DOCUMENT_2','FLAG_DOCUMENT_3','FLAG_DOCUMENT_4','FLAG_D
'FLAG_DOCUMENT_6','FLAG_DOCUMENT_7','FLAG_DOCUMENT_8','FLAG_DOCUMENT_9'
'FLAG_DOCUMENT_11','FLAG_DOCUMENT_12','FLAG_DOCUMENT_13','FLAG_DOCUMENT
'FLAG_DOCUMENT_16','FLAG_DOCUMENT_17','FLAG_DOCUMENT_18','FLAG_DOCUMENT
'FLAG_DOCUMENT_21' ]
flag_df=app_data[flag_doc + ['TARGET']]
flag_df['TARGET']=flag_df['TARGET'].replace({1:'defaulter', 0:'repayer'
length=len(flag_doc)
fig=plt.figure(figsize=(21,24))

for i,j in itertools.zip_longest(flag_doc,range(length)):
    plt.subplot(5,4,j+1)
    ax = sns.countplot(data=flag_df,x=i, hue=flag_df["TARGET"],palette=
```

The above graph shows that in most of the loan application cases, clients who applied for loans has not submitted FLAG_DOCUMENT_X except FLAG_DOCUMENT_3. Thus, Except for FLAG_DOCUMENT_3, we can delete rest of the columns.Data shows if borrower has submitted FLAG_DOCUMENT_3 then there is a less chance of defaulting the loan.

In [25]:
```
1 flag_doc.remove('FLAG_DOCUMENT_3')
```

In [26]:

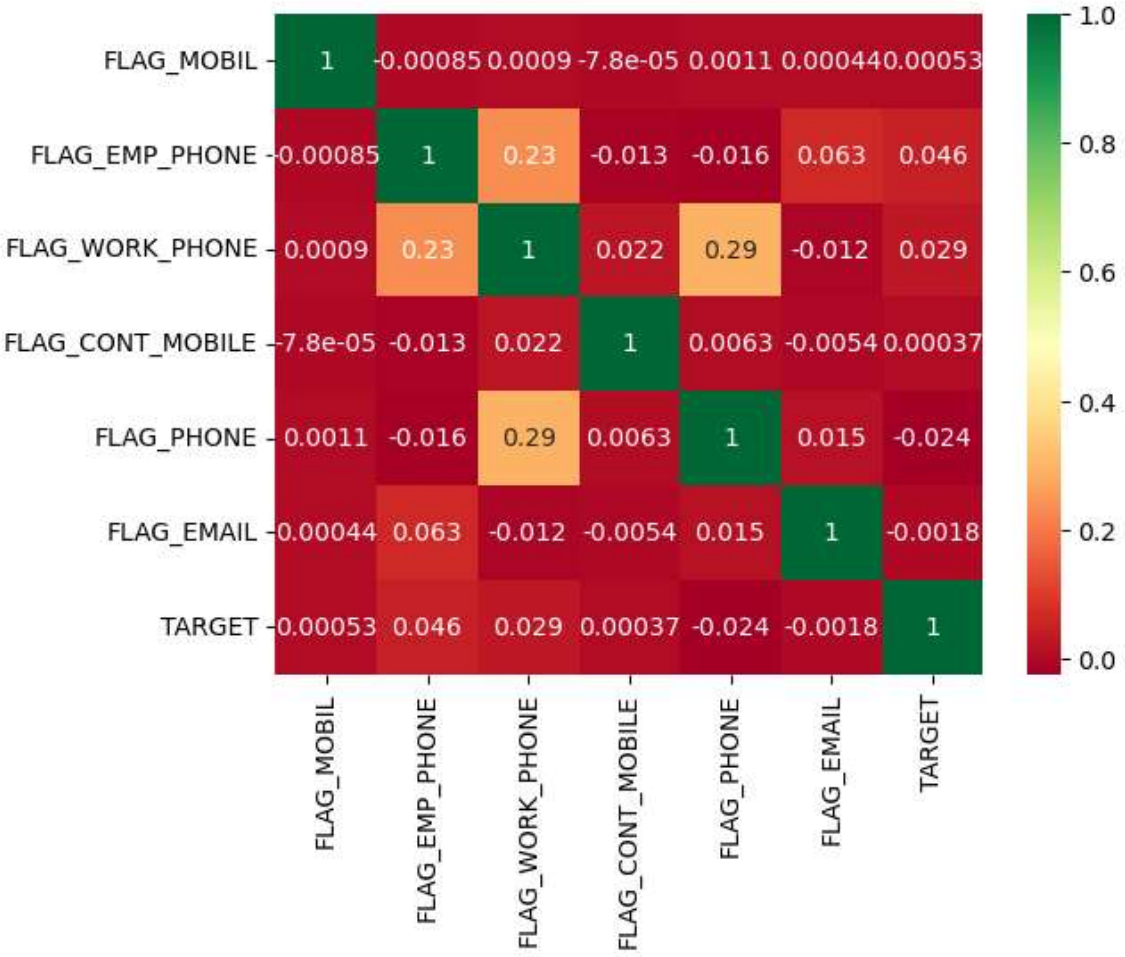
```
unwanted_app=unwanted_app + flag_doc
unwanted_app
```

Out[26]:
```
['OWN_CAR_AGE',
 'EXT_SOURCE_1',
 'APARTMENTS_AVG',
 'BASEMENTAREA_AVG',
 'YEARS_BEGINEXPLUATATION_AVG',
 'YEARS_BUILD_AVG',
 'COMMONAREA_AVG',
 'ELEVATORS_AVG',
 'ENTRANCES_AVG',
 'FLOORSMAX_AVG',
 'FLOORSMIN_AVG',
 'LANDAREA_AVG',
 'LIVINGAPARTMENTS_AVG',
 'LIVINGAREA_AVG',
 'NONLIVINGAPARTMENTS_AVG',
 'NONLIVINGAREA_AVG',
 'APARTMENTS_MODE',
 'BASEMENTAREA_MODE',
 'YEARS_BEGINEXPLUATATION_MODE',
 'YEARS_BUILD_MODE',
 'COMMONAREA_MODE',
 'ELEVATORS_MODE',
 'ENTRANCES_MODE',
 'FLOORSMAX_MODE',
 'FLOORSMIN_MODE',
 'LANDAREA_MODE',
 'LIVINGAPARTMENTS_MODE',
 'LIVINGAREA_MODE',
 'NONLIVINGAPARTMENTS_MODE',
 'NONLIVINGAREA_MODE',
 'APARTMENTS_MEDI',
 'BASEMENTAREA_MEDI',
 'YEARS_BEGINEXPLUATATION_MEDI',
 'YEARS_BUILD_MEDI',
 'COMMONAREA_MEDI',
 'ELEVATORS_MEDI',
 'ENTRANCES_MEDI',
 'FLOORSMAX_MEDI',
 'FLOORSMIN_MEDI',
 'LANDAREA_MEDI',
 'LIVINGAPARTMENTS_MEDI',
 'LIVINGAREA_MEDI',
 'NONLIVINGAPARTMENTS_MEDI',
 'NONLIVINGAREA_MEDI',
 'FONDKAPREMONT_MODE',
 'HOUSETYPE_MODE',
 'TOTALAREA_MODE',
 'WALLSMATERIAL_MODE',
 'EMERGENCYSTATE_MODE',
 'EXT_SOURCE_2',
 'EXT_SOURCE_3',
 'FLAG_DOCUMENT_2',
 'FLAG_DOCUMENT_4',
 'FLAG_DOCUMENT_5',
 'FLAG_DOCUMENT_6',
 'FLAG_DOCUMENT_7',
 'FLAG_DOCUMENT_8',
 'FLAG_DOCUMENT_9',
 'FLAG_DOCUMENT_10',
 'FLAG_DOCUMENT_11',
 'FLAG_DOCUMENT_12',
```

```
'FLAG_DOCUMENT_13',
'FLAG_DOCUMENT_14',
'FLAG_DOCUMENT_15',
'FLAG_DOCUMENT_16',
'FLAG_DOCUMENT_17',
'FLAG_DOCUMENT_18',
'FLAG_DOCUMENT_19',
'FLAG_DOCUMENT_20',
'FLAG_DOCUMENT_21']
```

In [27]:
```
contact_col=['FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_C
       'FLAG_PHONE', 'FLAG_EMAIL','TARGET']

contact_corr=app_data[contact_col].corr()
ax=sns.heatmap(contact_corr,annot=True,cmap ="RdYlGn")
```



There is no correlation between flags of mobile phone, email etc with loan repayment; thus these columns can be deleted

In [28]:
```
contact_col.remove('TARGET')
```

In [29]:
```
unwanted_app=unwanted_app + contact_col
```

In [30]:
```
app_data.drop(unwanted_app,axis=1, inplace=True)
```

In [31]:
```
1 app_data.shape
```

Out[31]:
```
(307511, 46)
```

In [32]:
```
1 app_data.info(verbose=True)
```

```
<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
```

Standardize Values

Convert DAYS_DECISION,DAYS_EMPLOYED, DAYS_REGISTRATION,DAYS_ID_PUBLISH from negative to positive as days cannot be negative.

In [33]:
```
date_col=['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUB
```

In [34]:
```
for i in date_col:
    app_data[i] = abs(app_data[i])
```

In [35]:
```
# Binning Numerical Columns to create a categorical column
```

In [36]:
```
# Creating bins for income amount
app_data['AMT_INCOME_TOTAL'] = app_data['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-60
     ,'800k-900k','900k-1M','above 1M']

app_data['AMT_INCOME_RANGE']= pd.cut(app_data['AMT_INCOME_TOTAL'], bins
```

In [37]:
```
app_data['AMT_INCOME_RANGE'].value_counts(normalize=True)*100
```

Out[37]:
```
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
above 1M      0.005858
Name: AMT_INCOME_RANGE, dtype: float64
```

In [38]:
```
# Creating bins for Credit amount
```

In [39]:
```
app_data['AMT_CREDIT']=app_data['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','500
       '800k-900k','900k-1M', '1M Above']

app_data['AMT_CREDIT_RANGE']=pd.cut(app_data['AMT_CREDIT'],bins=bins,la

```

In [40]:
```
app_data['AMT_CREDIT_RANGE'].value_counts(normalize=True)*100
```

Out[40]:
```
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 [41]:
```
# 16 % people have taken loan more than 1 million
```

In [42]:
```
# Creating bins for Age

app_data['AGE'] = app_data['DAYS_BIRTH'] // 365

bins=[0,20,30,40,50,100]
slots=['0-20','20-30','30-40','40-50','above 50']

app_data['AGE_GROUP']=pd.cut(app_data['AGE'],bins=bins,labels=slots)
```

In [43]:
```
app_data['AGE_GROUP'].value_counts(normalize=True)*100
```

Out[43]:
```
above 50    31.604398
30-40       27.028952
40-50       24.194582
20-30       17.171743
0-20         0.000325
Name: AGE_GROUP, dtype: float64
```

31% loan applicants have age above 50 years. More than 55% of loan applicants have age over 40 years.

In [44]:
```
# Creating bins for Employement Time
app_data['YEARS_EMPLOYED'] = app_data['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
app_data['EMPLOYMENT_YEAR']=pd.cut(app_data['YEARS_EMPLOYED'],bins=bins
```

In [45]:
```
1 app_data['EMPLOYMENT_YEAR'].value_counts(normalize=True)*100
```

Out[45]:
```
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
```

More than 55% of the loan applicants have work experience within 0-5 years and almost 80% of them have less than 10 years of work experience

In [46]:
```
1 #Checking the number of unique values each column possess to identify c
```

In [47]:
```
1 app_data.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
```

data type conversion

In [48]:
```
1 app_data.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]:
```
#Numeric columns are already in int64 and float64 format. Hence proceed
```

In [50]:
```
categorical_columns = ['NAME_CONTRACT_TYPE','CODE_GENDER','NAME_TYPE_SU
                       'NAME_FAMILY_STATUS','NAME_HOUSING_TYPE','OCCUPA
                       'ORGANIZATION_TYPE','FLAG_OWN_CAR','FLAG_OWN_REA
                       'REG_CITY_NOT_LIVE_CITY','REG_CITY_NOT_WORK_CITY
                       'LIVE_REGION_NOT_WORK_REGION','REGION_RATING_CLI
                       'REGION_RATING_CLIENT_W_CITY'
                      ]
for col in categorical_columns:
    app_data[col] =pd.Categorical(app_data[col])
```

In [51]:
```
1 app_data.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
```

In [52]:
```
# null value imputation
```

In [53]:
```
# checking% of null value

round(app_data.isnull().sum() / app_data.shape[0] *100,2)
```

Out[53]:
```
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 [54]:
```
# impute name_type_suit
```

In [55]:
```
app_data['NAME_TYPE_SUITE'].describe()
```

Out[55]:
```
count            306219
unique                7
top       Unaccompanied
freq             248526
Name: NAME_TYPE_SUITE, dtype: object
```

In [56]:
```
app_data['NAME_TYPE_SUITE'].fillna(app_data['NAME_TYPE_SUITE'].mode()[0
```

In [57]:
```
round(app_data.isnull().sum() / app_data.shape[0] *100,2)
```

Out[57]:

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	0.00
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 [58]:
```
app_data['OCCUPATION_TYPE'].describe()
```

Out[58]:
```
count       211120
unique          18
top       Laborers
freq         55186
Name: OCCUPATION_TYPE, dtype: object
```

In [59]:
```
# Impute categorical variable 'OCCUPATION_TYPE' which has higher null p
# with a new category as assigning to any existing category

app_data['OCCUPATION_TYPE'] = app_data['OCCUPATION_TYPE'].cat.add_categ
app_data['OCCUPATION_TYPE'].fillna('Unknown', inplace =True)
```

Impute numerical variables with the median as there are no outliers that can be seen from results of describe() and mean() returns decimal values and these columns represent number of enquiries made which cannot be decimal:

In [60]:
```
app_data[['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'
```

Out[60]:

	AMT_REQ_CREDIT_BUREAU_HOUR	AMT_REQ_CREDIT_BUREAU_DAY	AMT_REQ_CRED
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	

In [61]:
```
amount = ['AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY','AM
          'AMT_REQ_CREDIT_BUREAU_QRT','AMT_REQ_CREDIT_BUREAU_YEAR']
```

In [62]:
```
for i in amount:
    app_data[i].fillna(app_data[i].median(),inplace=True)
```

In [63]:
```
round(app_data.isnull().sum() / app_data.shape[0] *100,2)
```

Out[63]:
```
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                0.00
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                0.00
CNT_FAM_MEMBERS                0.00
REGION_RATING_CLIENT           0.00
REGION_RATING_CLIENT_W_CITY    0.00
WEEKDAY_APPR_PROCESS_START     0.00
HOUR_APPR_PROCESS_START        0.00
REG_REGION_NOT_LIVE_REGION     0.00
REG_REGION_NOT_WORK_REGION     0.00
LIVE_REGION_NOT_WORK_REGION    0.00
REG_CITY_NOT_LIVE_CITY         0.00
REG_CITY_NOT_WORK_CITY         0.00
LIVE_CITY_NOT_WORK_CITY        0.00
ORGANIZATION_TYPE              0.00
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     0.00
AMT_REQ_CREDIT_BUREAU_DAY      0.00
AMT_REQ_CREDIT_BUREAU_WEEK     0.00
AMT_REQ_CREDIT_BUREAU_MON      0.00
AMT_REQ_CREDIT_BUREAU_QRT      0.00
AMT_REQ_CREDIT_BUREAU_YEAR     0.00
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
```

We still have few null values in the columns: AMT_GOODS_PRICE, OBS_30_CNT_SOCIAL_CIRCLE, DEF_30_CNT_SOCIAL_CIRCLE, OBS_60_CNT_SOCIAL_CIRCLE, DEF_60_CNT_SOCIAL_CIRCLE. We can ignore as this

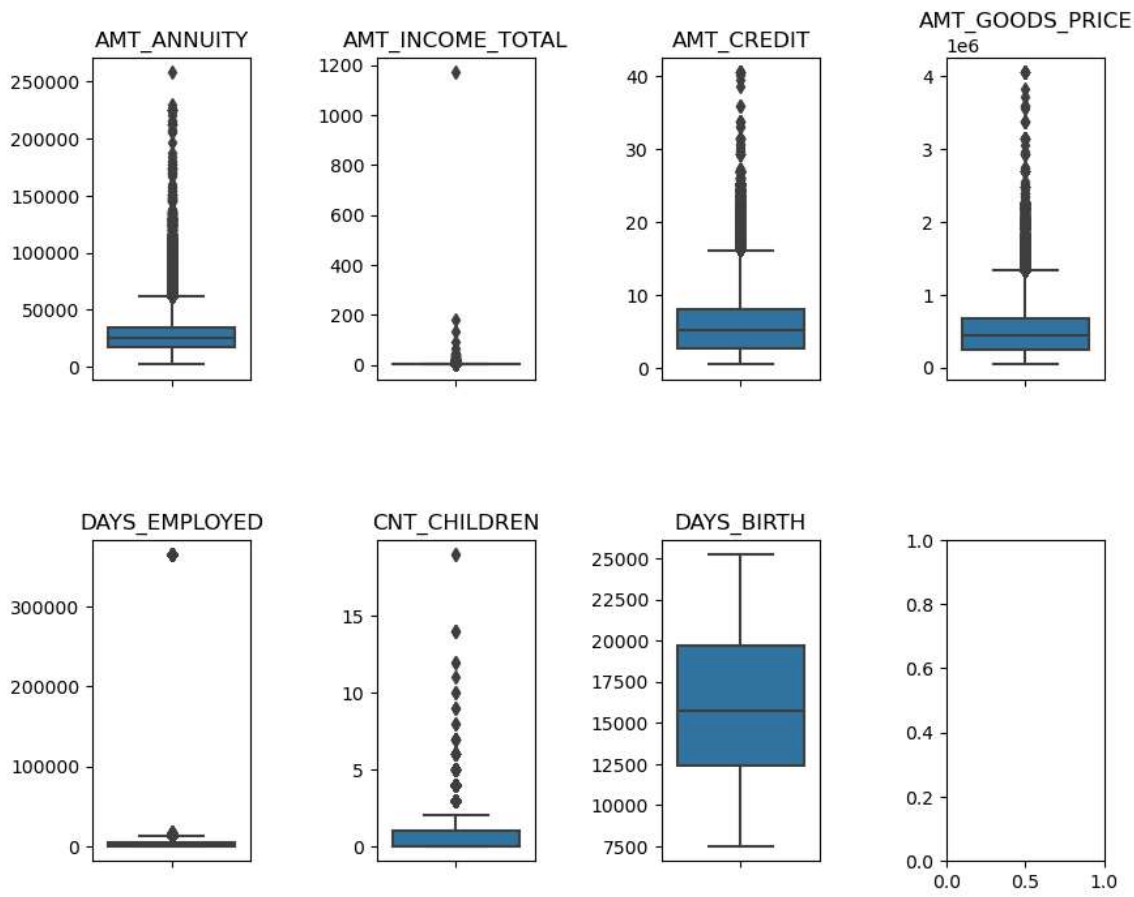
percentage is very less.

In [64]:
```
# identifying outliers
```

In [65]:
```
app_outlier_col_1 = ['AMT_ANNUITY','AMT_INCOME_TOTAL','AMT_CREDIT','AMT
app_outlier_col_2 = ['CNT_CHILDREN','DAYS_BIRTH']
```

In [77]:
```
fig, axes = plt.subplots(2, 4 , figsize=(10,8))
for i in  enumerate(app_outlier_col_1):
    plt.subplot(2,4,i[0]+1)
    plt.subplots_adjust(wspace=0.8, hspace=0.5)
    sns.boxplot(y=app_data[i[1]])
    plt.title(i[1])
    plt.ylabel('')

for i in enumerate(app_outlier_col_2):
    plt.subplot(2,4,i[0]+6)
    sns.boxplot(y=app_data[i[1]])
    plt.title(i[1])
    plt.ylabel("")
```



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 [79]:
```
app_data[['AMT_ANNUITY', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_GOODS_P
'DAYS_BIRTH','CNT_CHILDREN','DAYS_EMPLOYED']].describe()
```

Out[79]:

	AMT_ANNUITY	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_GOODS_PRICE	DAYS_BIR
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(

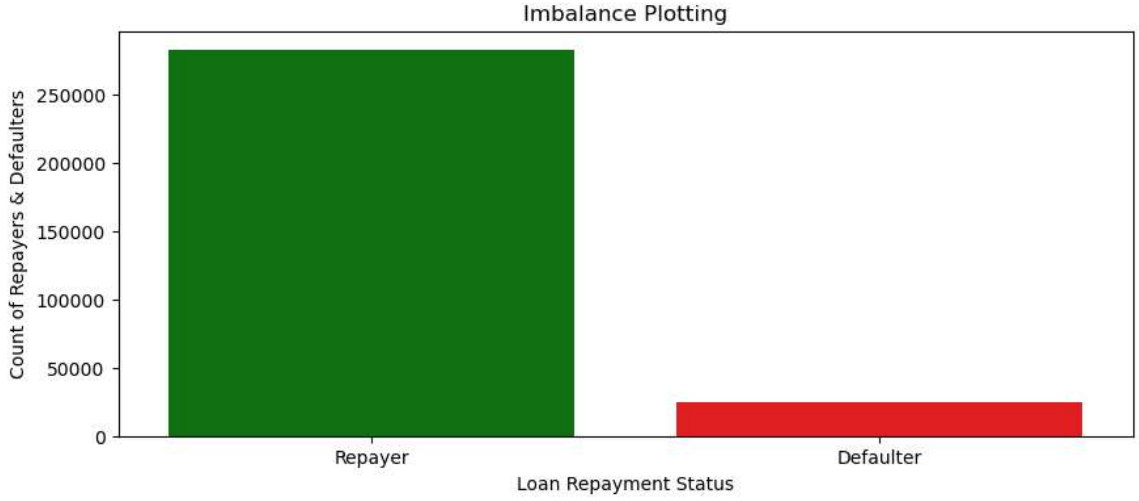
Strategy: The data analysis flow has been planned in following way : Imbalance in Data Categorical Data Analysis Categorical segmented Univariate Analysis Categorical Bi/Multivariate analysis

Numeric Data Analysis Bi-furcation of databased based on TARGET data Correlation Matrix Numerical segmented Univariate Analysis Numerical Bi/Multivariate analysis

Imbalance Analysis

In [82]:
```
Imbalance = app_data["TARGET"].value_counts().reset_index()

plt.figure(figsize=(10,4))
x= ['Repayer','Defaulter']
sns.barplot(x=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 [86]:
```
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 De
print('Ratios of imbalance in relative with respect to Repayer and Defa

```

```
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)
```

Plotting Functions

In [87]:

```
# function for plotting repetitive countplots in univariate categorical
# This function will create two subplots:
# 1. Count plot of categorical column w.r.t TARGET;
# 2. Percentage of defaulters within column

def univariate_categorical(feature,ylog=False,label_rotation=False,hori
    temp = app_data[feature].value_counts()
    df1 = pd.DataFrame({feature: temp.index,'Number of contracts': temp

    # Calculate the percentage of target=1 per category value
    cat_perc = app_data[[feature, 'TARGET']].groupby([feature],as_index
    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=app_data,
                      hue ="TARGET",
                      order=cat_perc[feature],
                      palette=['g','r'])

    # Define common styling
    ax1.set_title(feature, fontdict={'fontsize' : 10, 'fontweight' : 3,
    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, 'fontwe


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

    # 2. Subplot 2: Percentage of defaulters within the categorical col
    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,

    plt.show();
```

In [88]:
```
# function for plotting repetitive countplots in bivariate categorical

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

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

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

In [89]:
```
# function for plotting repetitive rel plots in bivaritae numerical ana

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

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

In [90]:
```
#function for plotting repetitive countplots in univariate categorical

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


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

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

    plt.show()
```

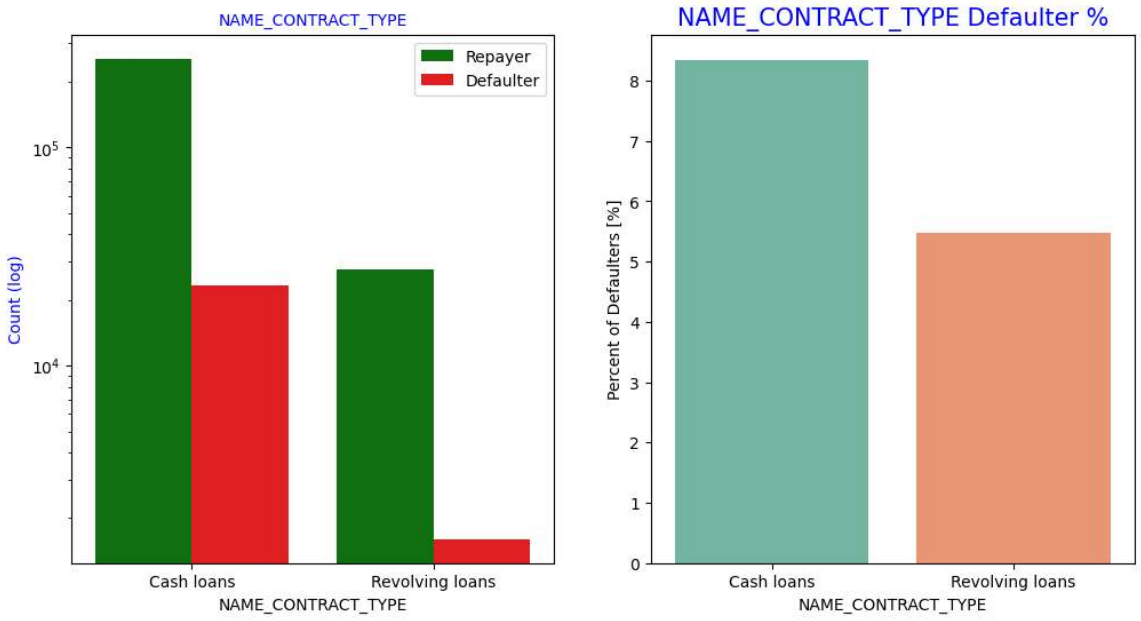
In [91]:
```
# Function to plot point plots on merged dataframe

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

Categorical Variables Analysis

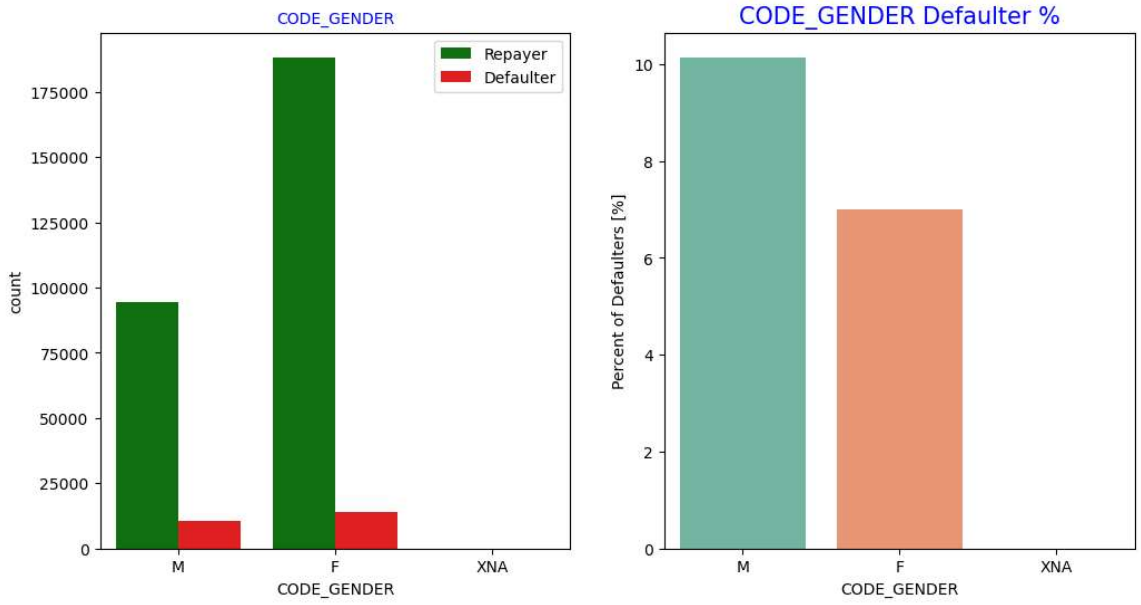
In [92]:
```
 # Segmented Univariate Analysis
```

In [93]:
```
# Checking the contract type based on loan repayment status
univariate_categorical('NAME_CONTRACT_TYPE',True)
```



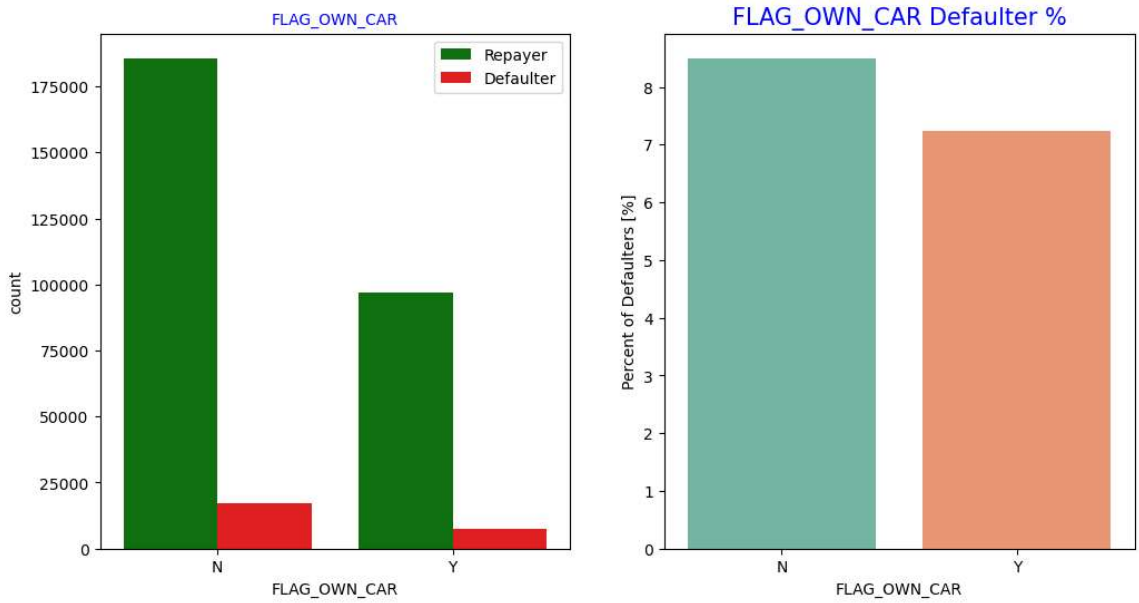
Contract type: Revolving loans are just a small fraction (10%) from the total number of loans; in the same time, a larger amount of Revolving loans, comparing with their frequency, are not repaid.

In [94]:
```
# Checking the type of Gender on loan repayment status
univariate_categorical('CODE_GENDER')
```



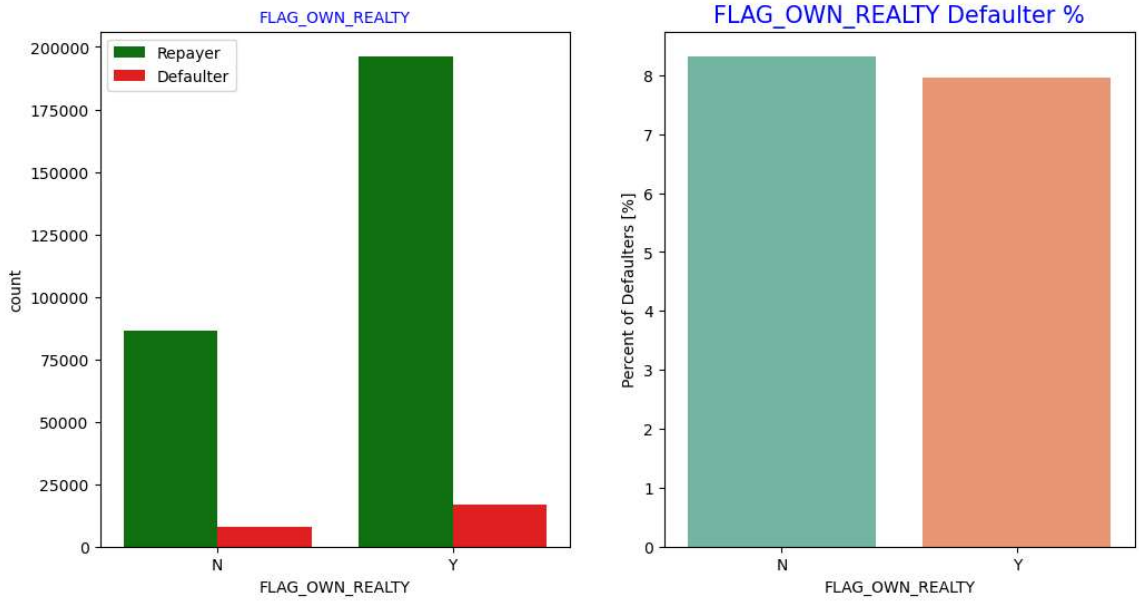
Contract type: Revolving loans are just a small fraction (10%) from the total number of loans;in the same time, a larger amount of Revolving loans, comparing with their frequency, are not repaid.

In [95]:
```
# Checking if owning a car is related to loan repayment status
univariate_categorical('FLAG_OWN_CAR')
```



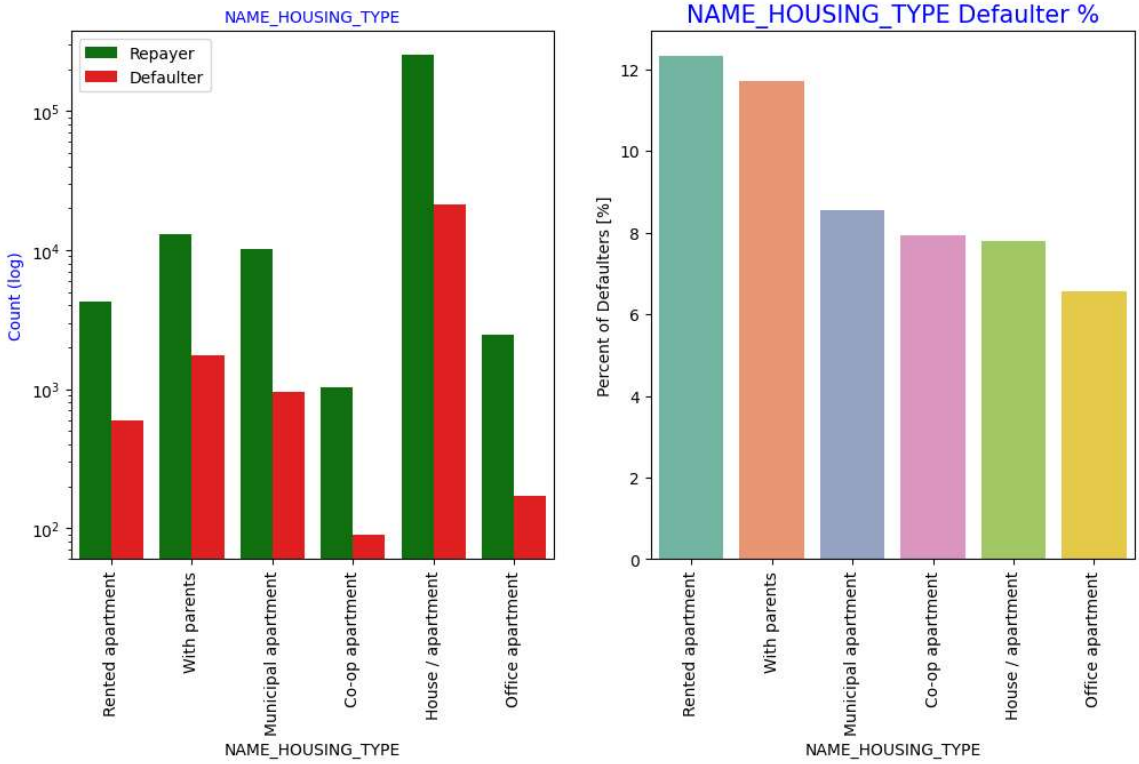
Clients who own a car are half in number of the clients who dont own a car. But based on the percentage of deault, there is no correlation between owning a car and loan repayment as in both cases the default percentage is almost same.

In [96]:
```
# Checking if owning a realty is related to loan repayment status
univariate_categorical('FLAG_OWN_REALTY')
```



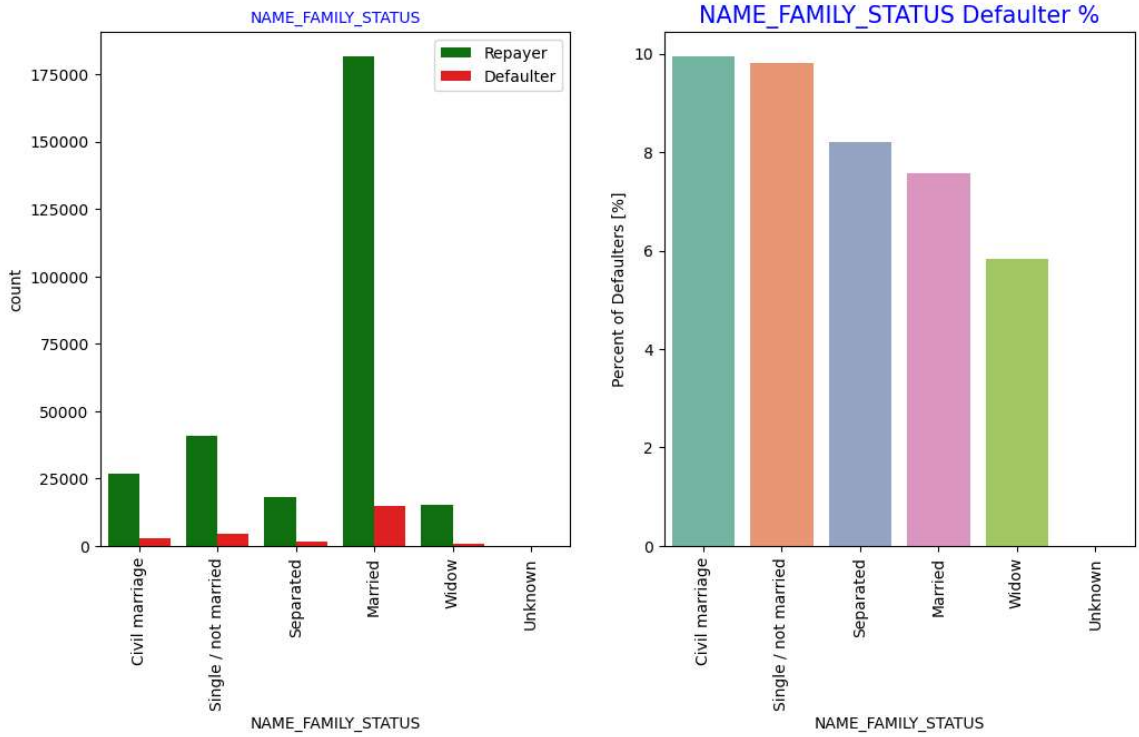
The clients who own real estate are more than double of the ones that don't own. But the defaulting rate of both categories are around the same (~8%). Thus there is no correlation between owning a reality and defaulting the loan.

In [97]:
```
# Analyzing Housing Type based on loan repayment status
univariate_categorical("NAME_HOUSING_TYPE",True,True,True)
```



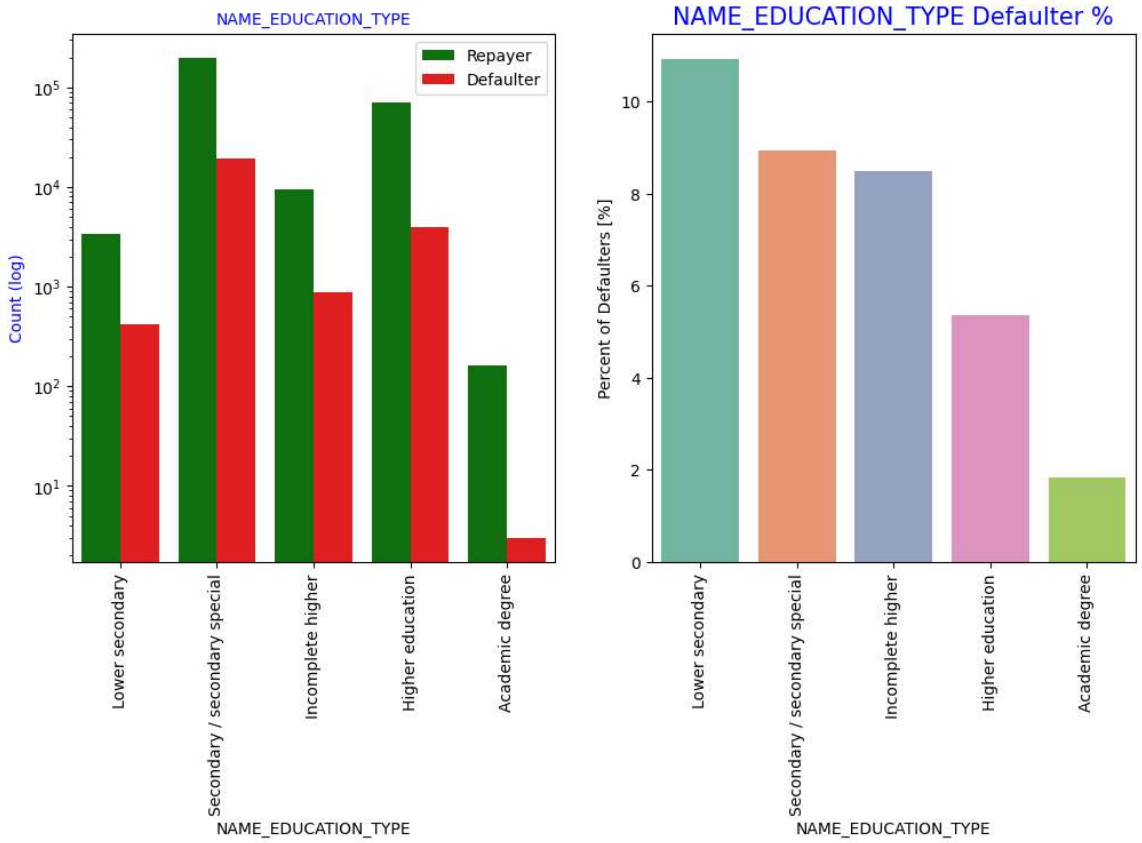
Majority of people live in House/apartment People living in office apartments have lowest default rate People living with parents (~11.5%) and living in rented apartments(>12%) have higher probability of defaulting

In [98]:
```
# Analyzing Family status based on loan repayment status
univariate_categorical("NAME_FAMILY_STATUS",False,True,True)
```



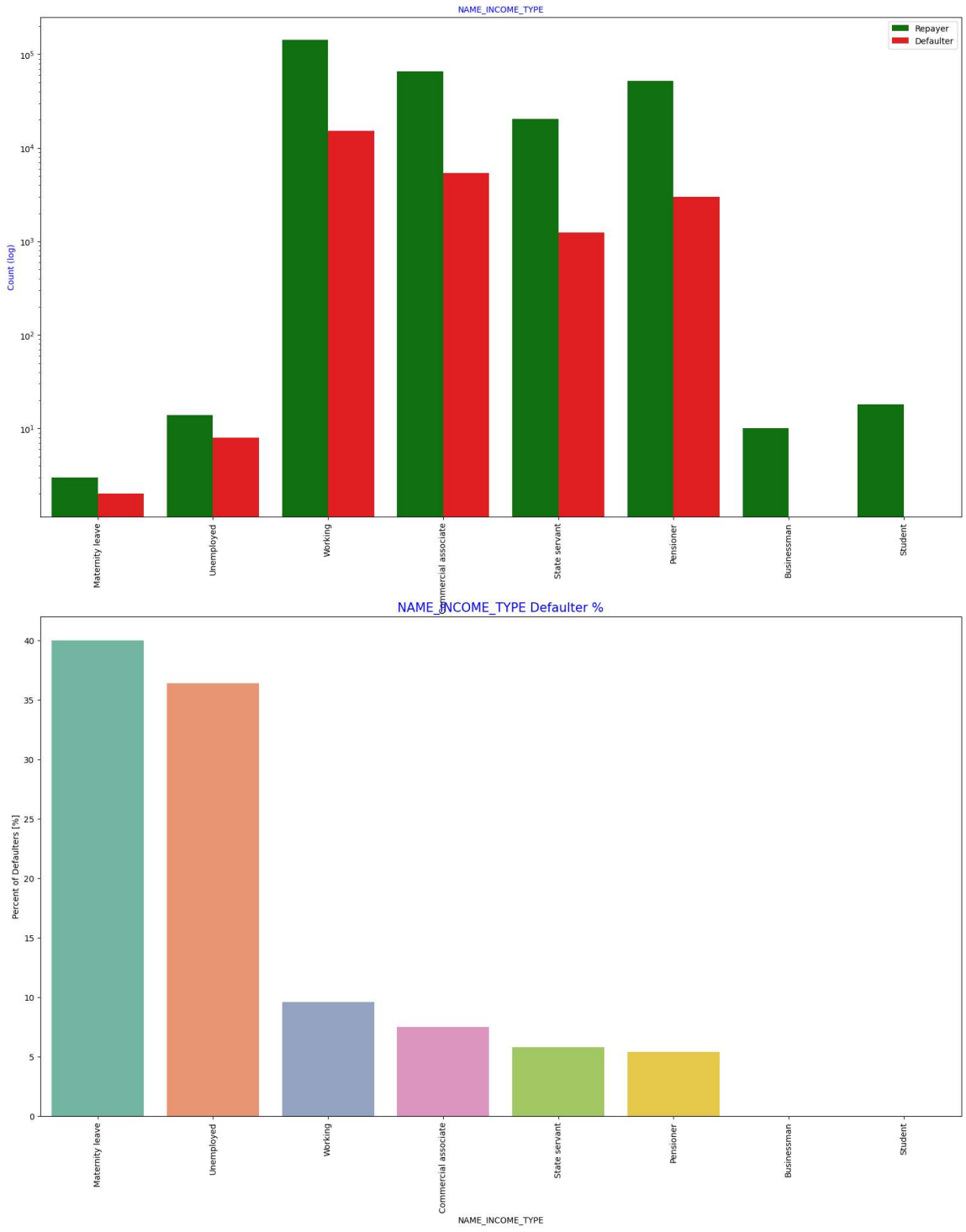
Most of the people who have taken loan are married, followed by Single/not married and civil marriage In terms of percentage of not repayment of loan, Civil marriage has the highest percent of not repayment (10%), with Widow the lowest (exception being Unknown).

In [99]:
```
# Analyzing Education Type based on loan repayment status
univariate_categorical("NAME_EDUCATION_TYPE",True,True,True)
```



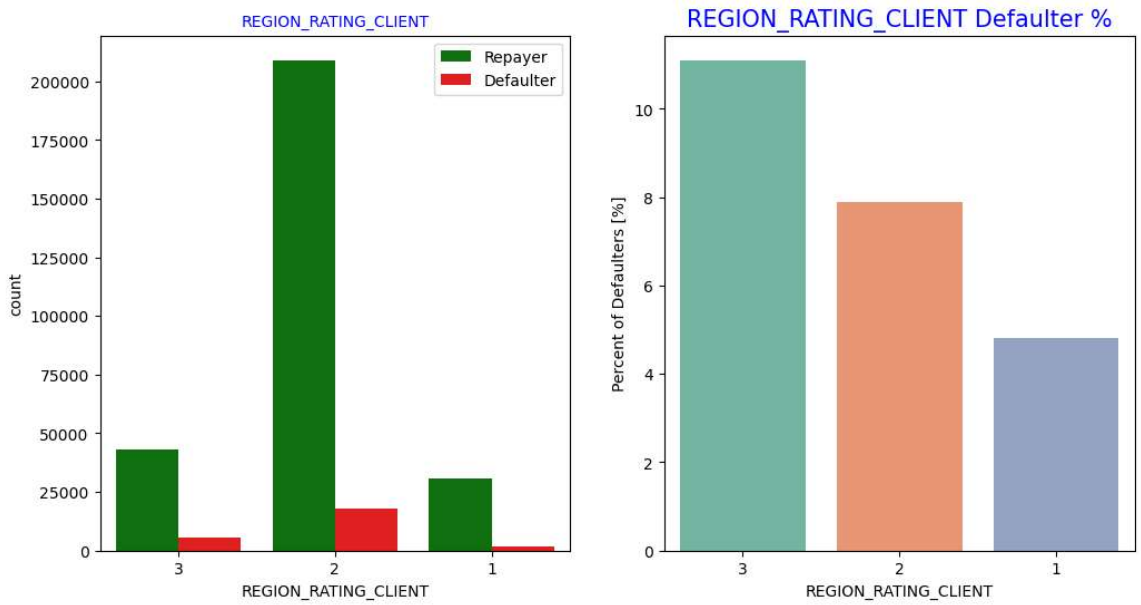
Majority of the clients have Secondary / secondary special education, followed by clients with Higher education. Only a very small number having an academic degree The Lower secondary category, although rare, have the largest rate of not returning the loan (11%). The people with Academic degree have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

In [100]:
```
# Analyzing Income Type based on loan repayment status
univariate_categorical("NAME_INCOME_TYPE",True,True,False)
```



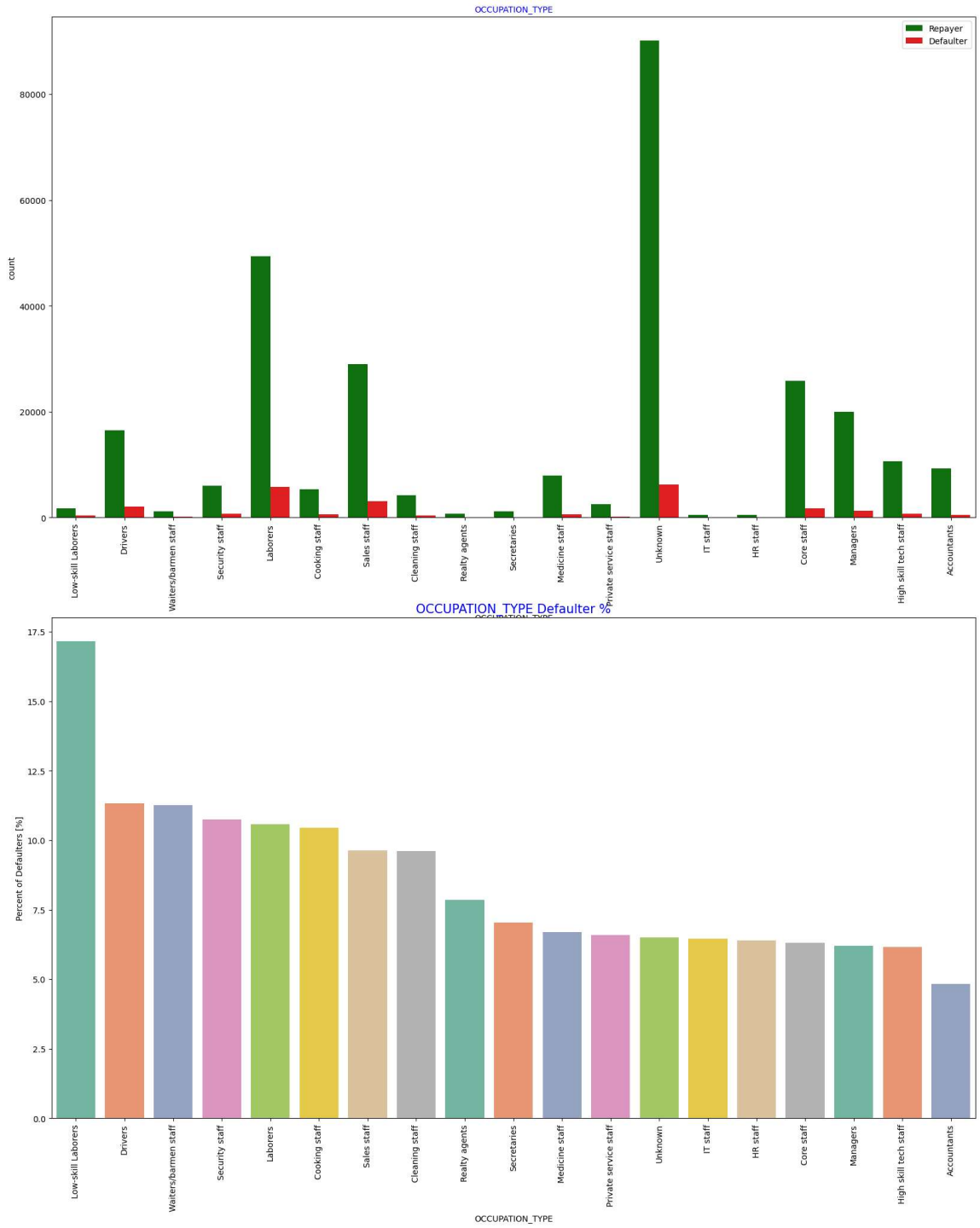
Most of applicants for loans have income type as Working, followed by Commercial associate, Pensioner and State servant. The applicants with the type of income Maternity leave have almost 40% ratio of not returning loans, followed by Unemployed (37%). The rest of types of incomes are under the average 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 [101]:
```
# Analyzing Region rating where applicant lives based on loan repayment
univariate_categorical("REGION_RATING_CLIENT",False,False,True)
```



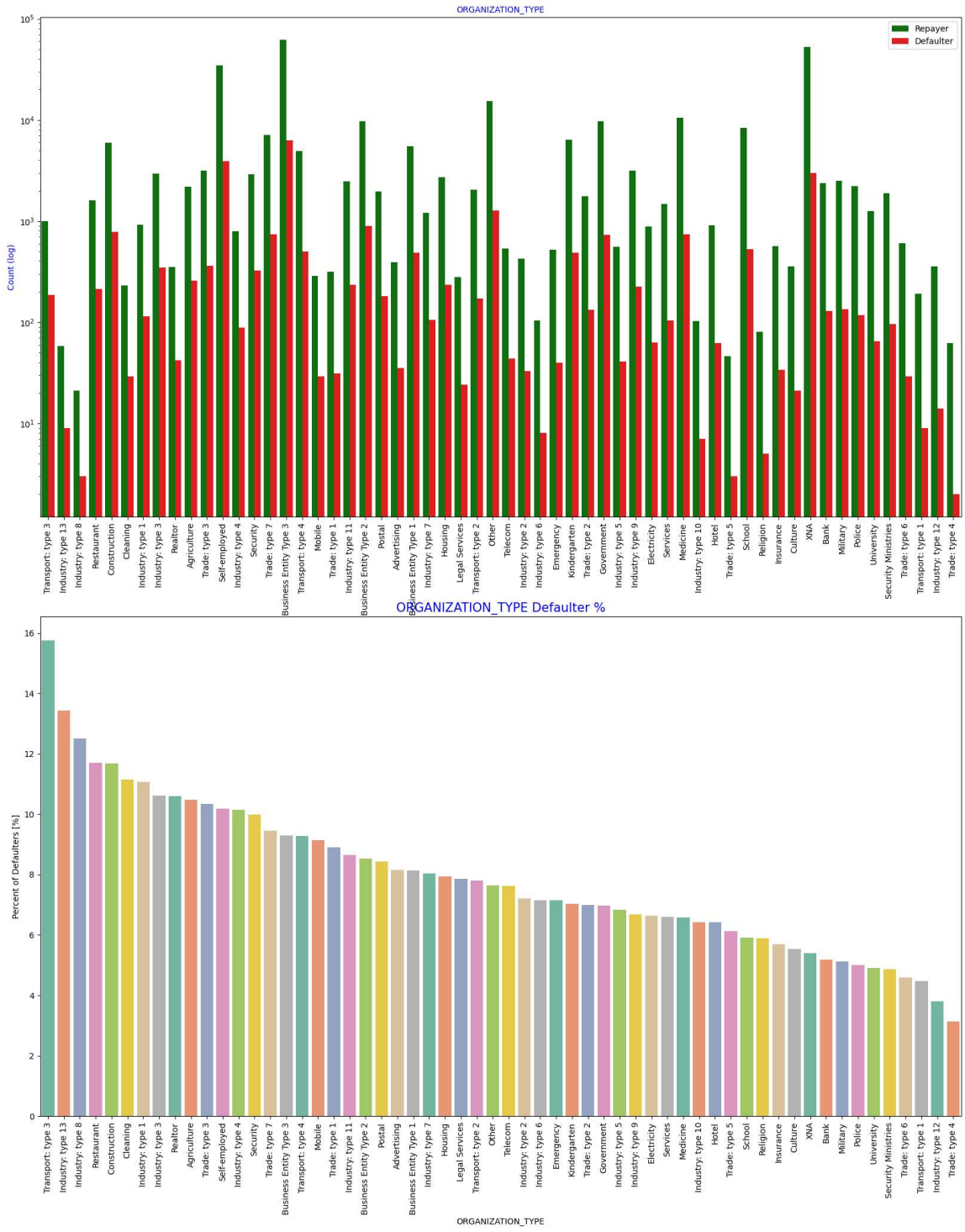
Most of the applicants are living in Region_Rating 2 place. Region Rating 3 has the highest default rate (11%) Applicant living in Region_Rating 1 has the lowest probability of defaulting, thus safer for approving loans

In [102]:
```
# Analyzing Occupation Type where applicant lives based on loan repayme
univariate_categorical("OCCUPATION_TYPE",False,True,False)
```



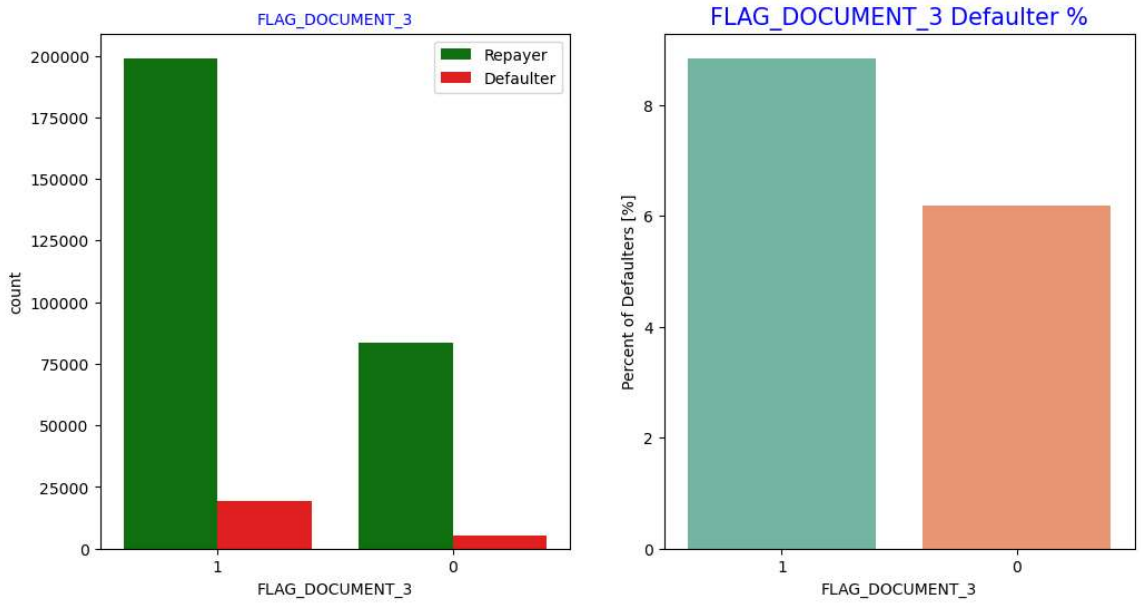
Most of the loans are taken by Laborers, followed by Sales staff. IT staff take the lowest amount of loans. The category with highest percent of not repaid loans are Low-skill Laborers (above 17%), followed by Drivers and Waiters/barmen staff, Security staff, Laborers and Cooking staff.

In [103]:
```
# Checking Loan repayment status based on Organization type
univariate_categorical("ORGANIZATION_TYPE",True,True,False)
```



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

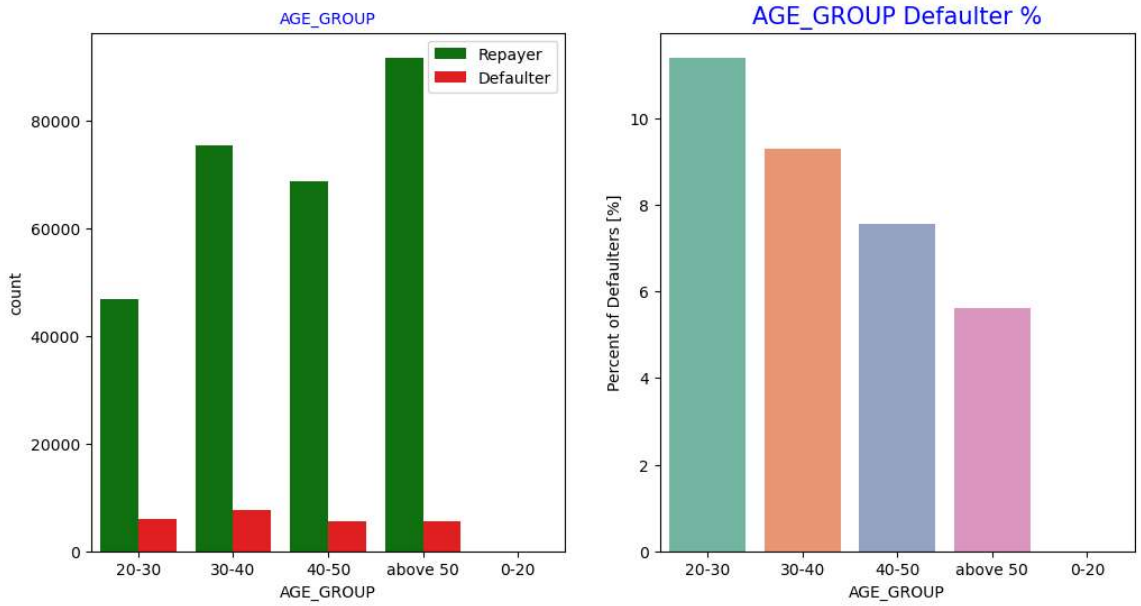
In [105]:
```
# Analyzing Flag_Doc_3 submission status based on loan repayment status
univariate_categorical("FLAG_DOCUMENT_3",False,False,True)
```



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%)

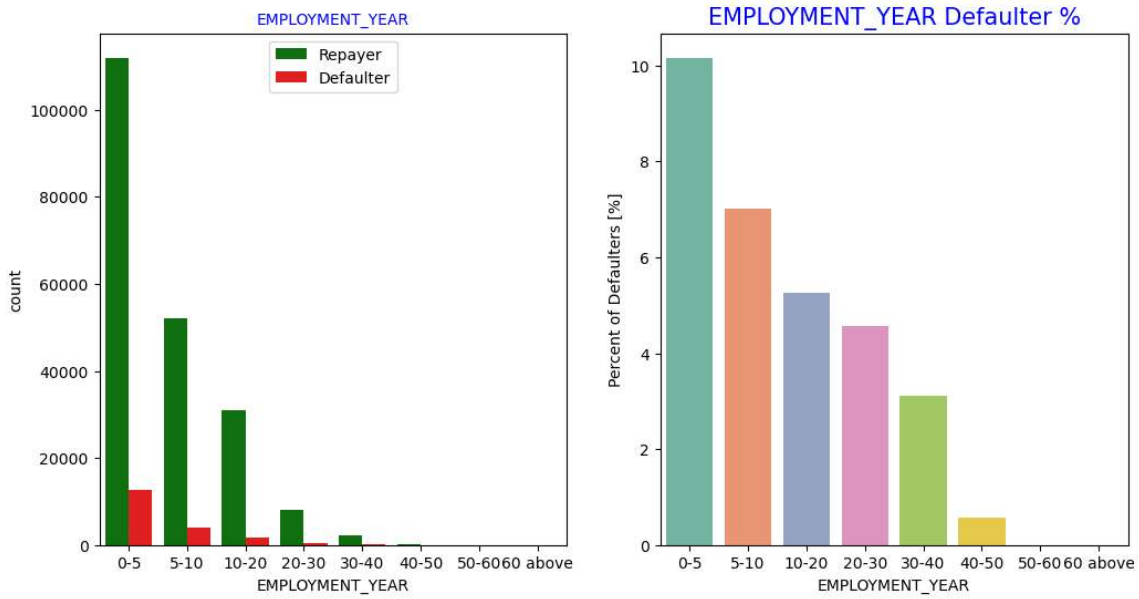
In [106]:
```
# Analyzing Age Group based on loan repayment status
univariate_categorical("AGE_GROUP",False,False,True)

```



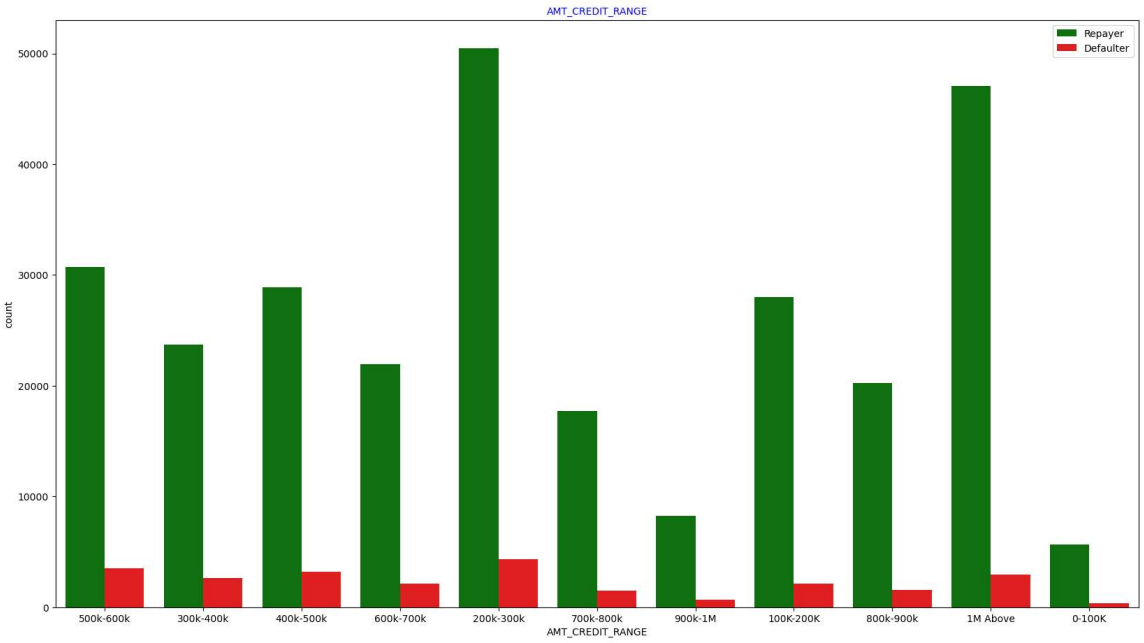
People in the age group range 20-40 have higher probability of defaulting People above age of 50 have low probability of defailting

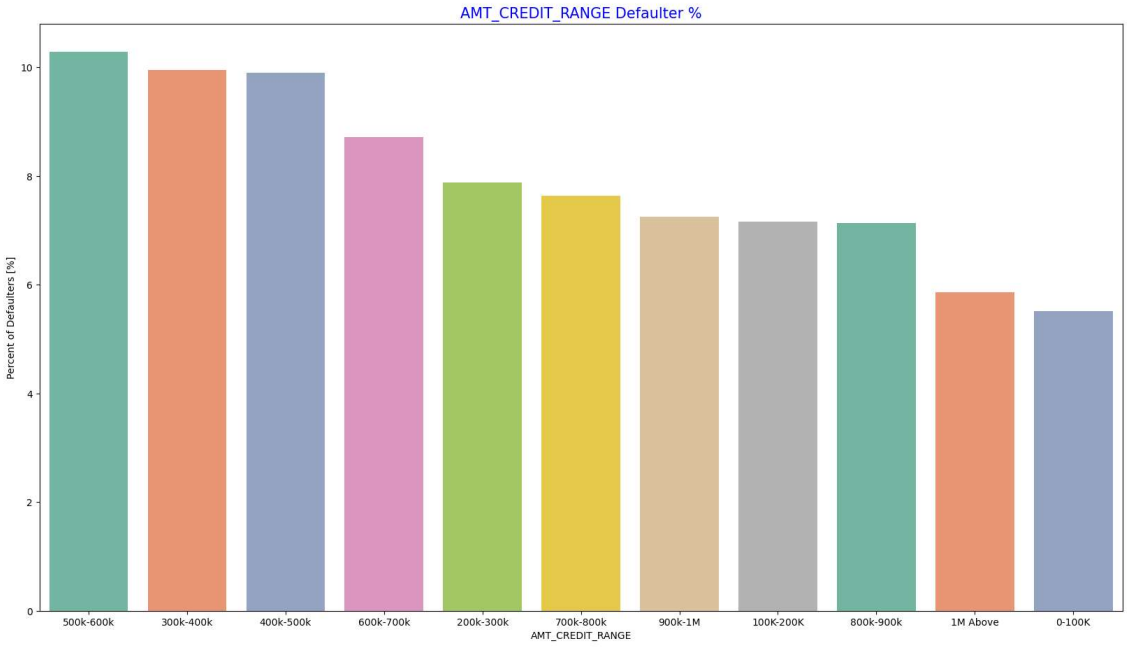
In [107]:
```
# Analyzing Employment_Year based on loan repayment status
univariate_categorical("EMPLOYMENT_YEAR",False,False,True)
```



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

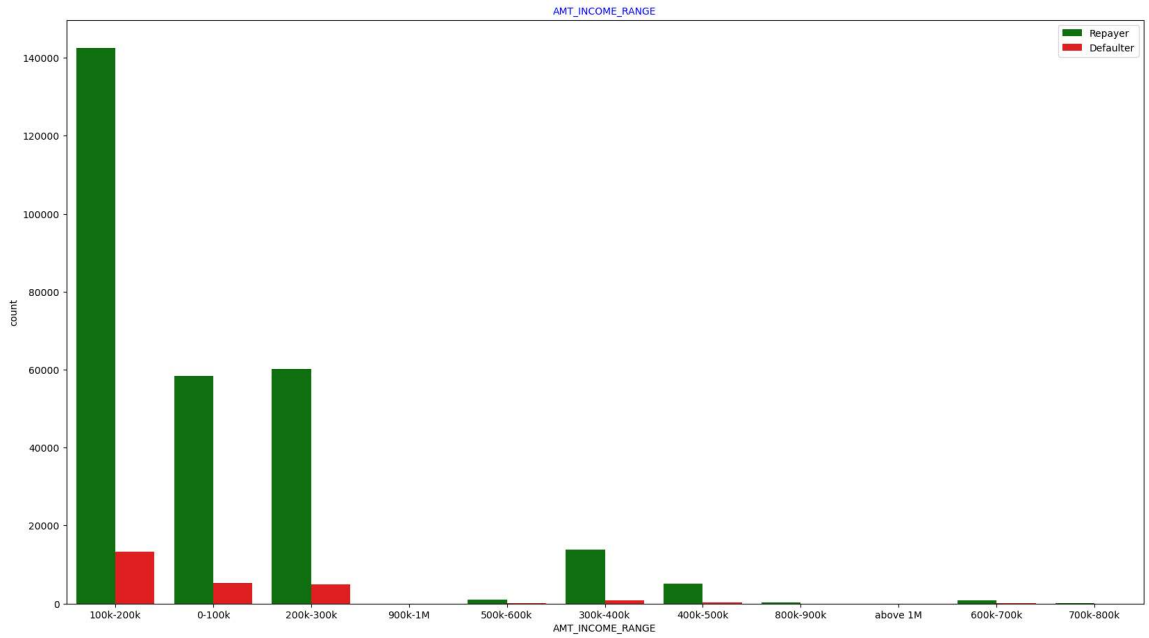
In [108]:
```
# Analyzing Amount_Credit based on loan repayment status
univariate_categorical("AMT_CREDIT_RANGE",False,False,False)
```

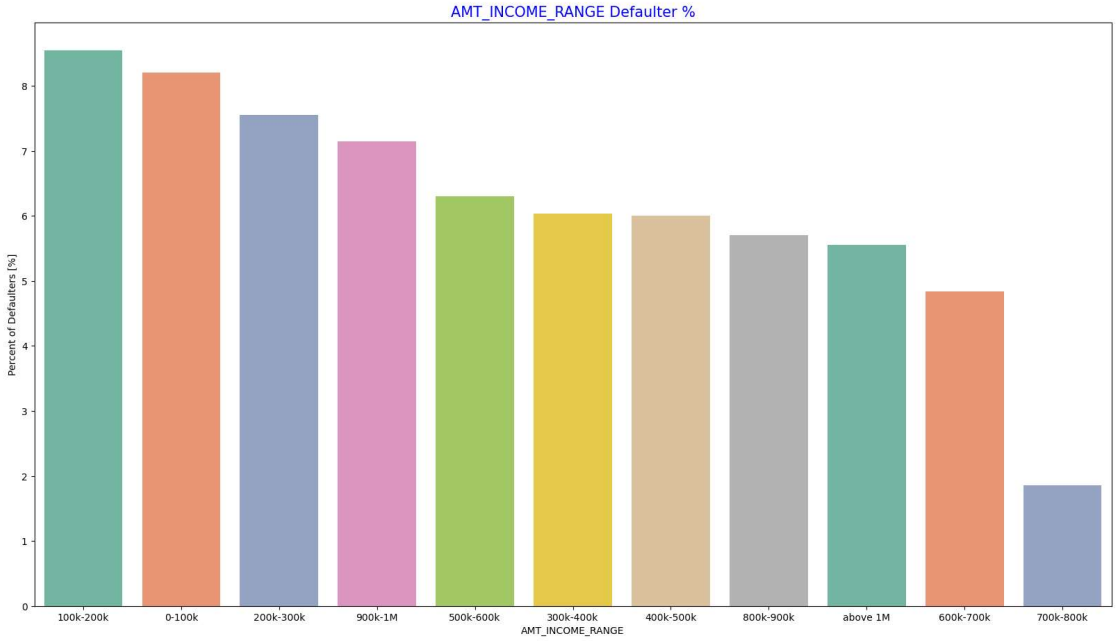




More than 80% of the loan provided are for amount less than 900,000 People who get loan for 300-600k tend to default more than others.

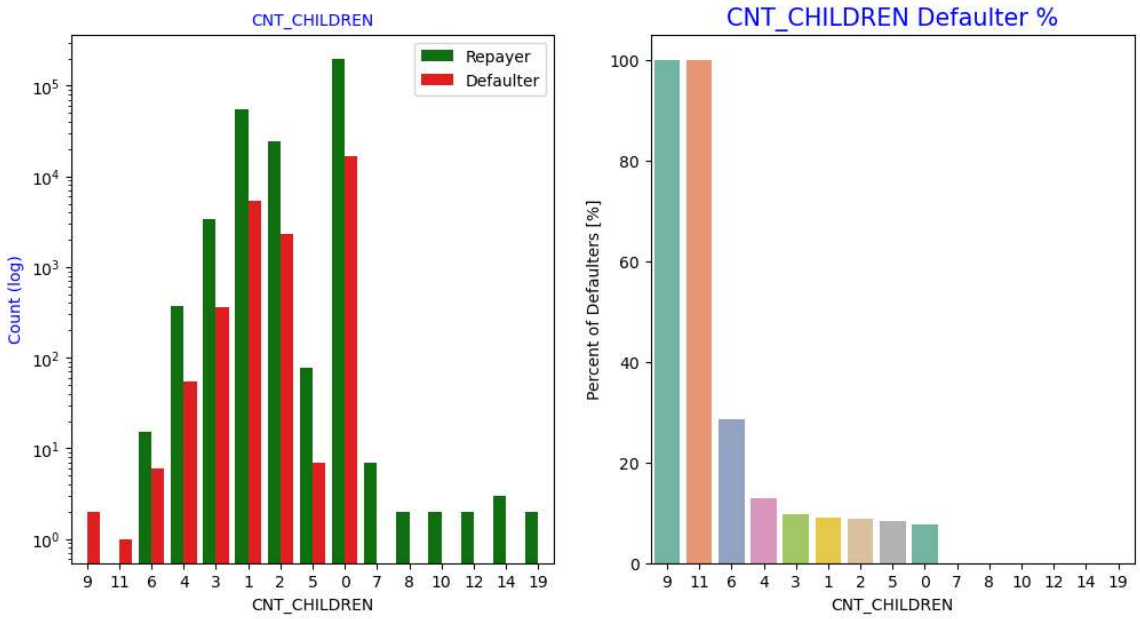
In [109]:
```
# Analyzing Amount_Income Range based on loan repayment status
univariate_categorical("AMT_INCOME_RANGE",False,False,False)
```





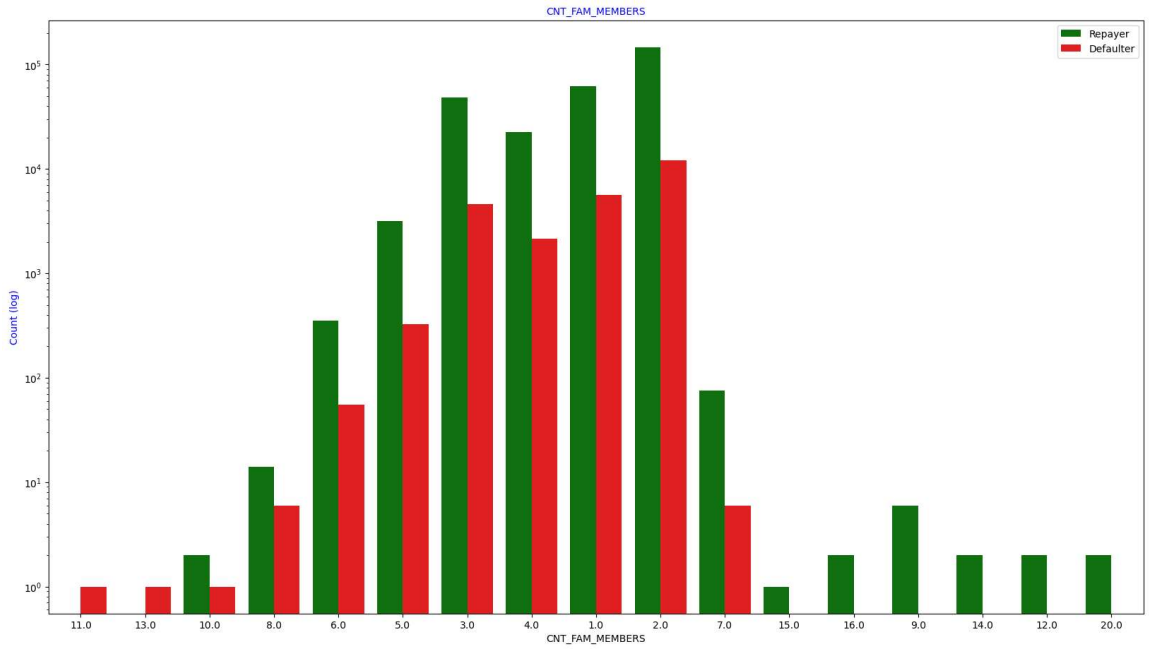
90% of the applications have Income total less than 300,000 Application with Income less than 300,000 has high probability of defaulting Applicant with Income more than 700,000 are less likely to default

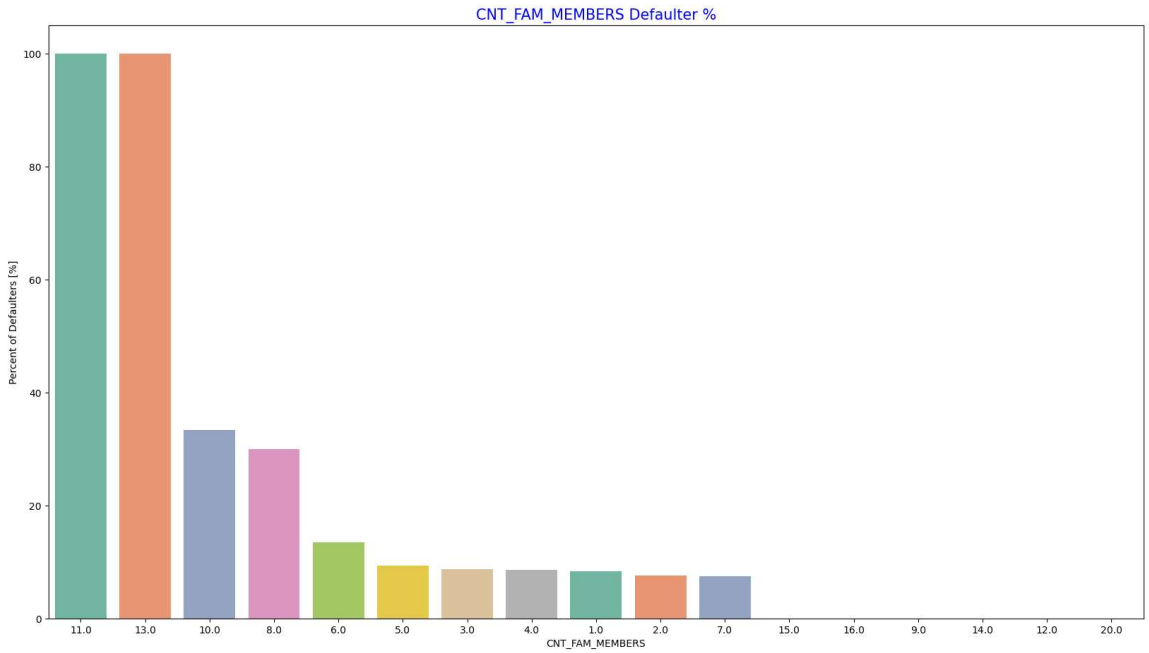
In [110]:
```
# Analyzing Number of children based on loan repayment status
univariate_categorical("CNT_CHILDREN",True)
```



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 [111]:
```
# Analyzing Number of family members based on loan repayment status
univariate_categorical("CNT_FAM_MEMBERS",True, False, False)
```





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

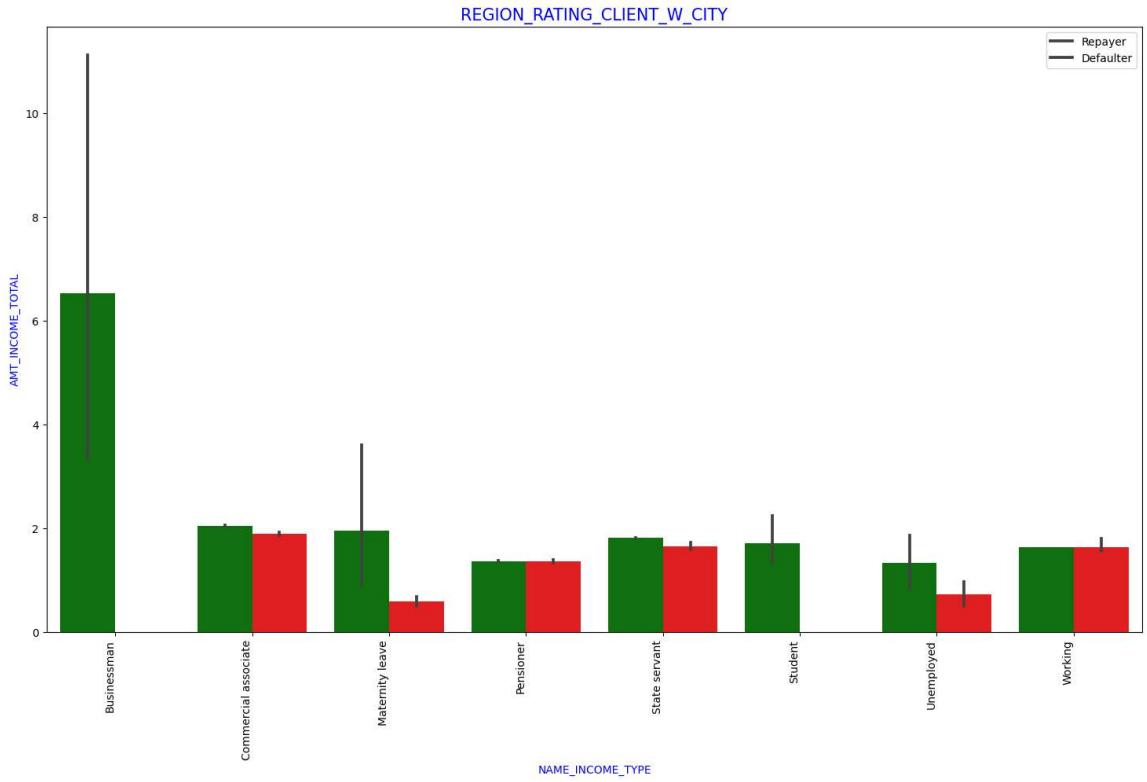
Categorical Bi/Multivariate Analysis

In [112]:
```
app_data.groupby('NAME_INCOME_TYPE')['AMT_INCOME_TOTAL'].describe()
```

Out[112]:

NAME_INCOME_TYPE	count	mean	std	min	25%	50%	75%	max
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 [113]:
```
# Income type vs Income Amount Range
bivariate_bar("NAME_INCOME_TYPE","AMT_INCOME_TOTAL",app_data,"TARGET",(
```



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

Numeric Variables Analysis

In [116]:
```
 # Bifurcating the app-data dataframe based on Target value 0 and 1 for
app_data.columns

```

Out[116]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_
OWN_CAR',
       'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDI
T', 'AMT_ANNUITY',
       'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDU
CATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELAT
IVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'OCCUPATIO
N_TYPE',
       'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_
CITY',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START', 'REG_REGIO
N_NOT_LIVE_REGION',
       'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION', 'REG_C
ITY_NOT_LIVE_CITY',
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_
TYPE',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT
_SOCIAL_CIRCLE',
       'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMEN
T_3',
       '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',
       'AMT_INCOME_RANGE', 'AMT_CREDIT_RANGE', 'AGE', 'AGE_GROUP', 'YEARS_
EMPLOYED',
       'EMPLOYMENT_YEAR'],
      dtype='object')
```

In [118]:
```
# Bifurcating the app_data dataframe based on Target value 0 and 1 for
cols_for_correlation = ['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_
                        'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT
                        'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_ED
                        'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIV
                        'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'OCCUPA
                        'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PR
                        'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_W
                        'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CI
                        'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_
                        'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_B
                        'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BU


Repayer_df = app_data.loc[app_data['TARGET']==0, cols_for_correlation]
Defaulter_df = app_data.loc[app_data['TARGET']==1, cols_for_correlation
```

Correlation between numeric variable

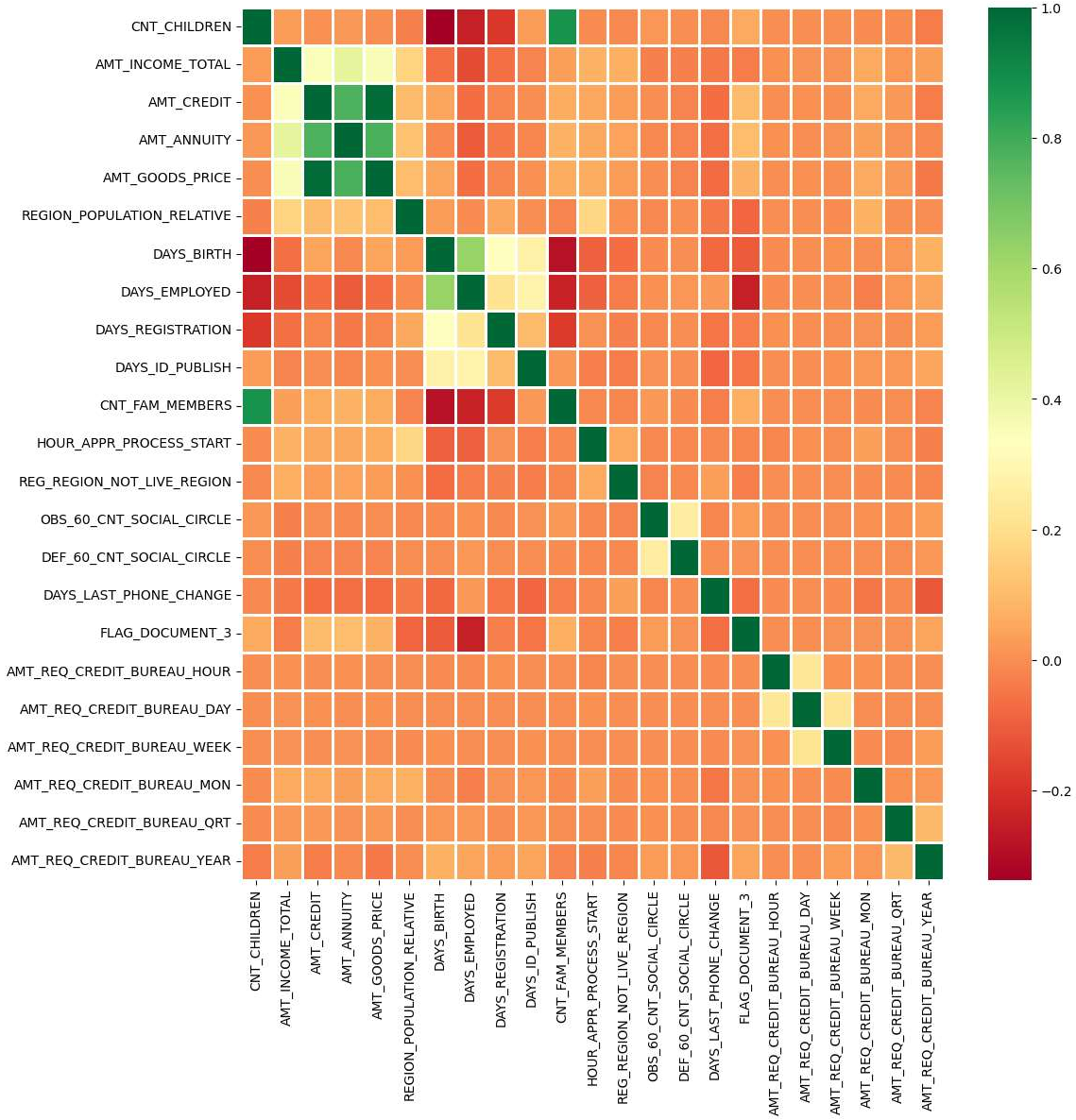
In [120]:

```
# Getting the top 10 correlation for the Repayers data
corr_repayer = Repayer_df.corr()
corr_repayer = corr_repayer.where(np.triu(np.ones(corr_repayer.shape),
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=
corr_df_repayer.head(10)
```

Out[120]:

	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 [121]:
```
fig = plt.figure(figsize=(12,12))
ax = sns.heatmap(Repayer_df.corr(), cmap="RdYlGn",annot=False,linewidth
```



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.

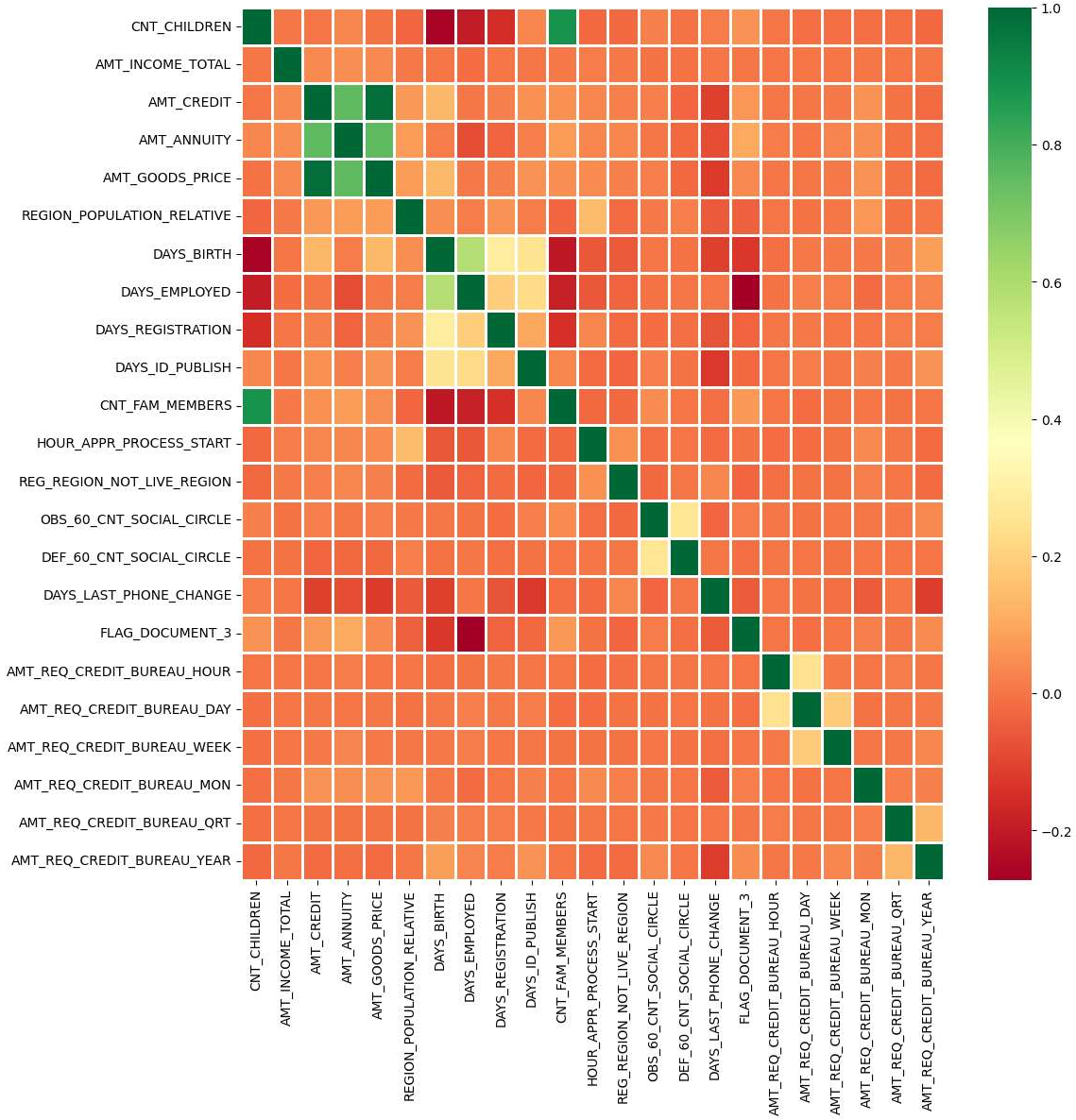
In [123]:

```
# Getting the top 10 correlation for the Defaulter data
corr_Defaulter = Defaulter_df.corr()
corr_Defaulter = corr_Defaulter.where(np.triu(np.ones(corr_Defaulter.sh
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, inplac
corr_df_Defaulter.head(10)
```

Out[123]:

	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 [124]:
```
fig = plt.figure(figsize=(12,12))
ax = sns.heatmap(Defaulter_df.corr(), cmap="RdYlGn",annot=False,linewic
```



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)

Numerical Univariate Analysis

In [125]:

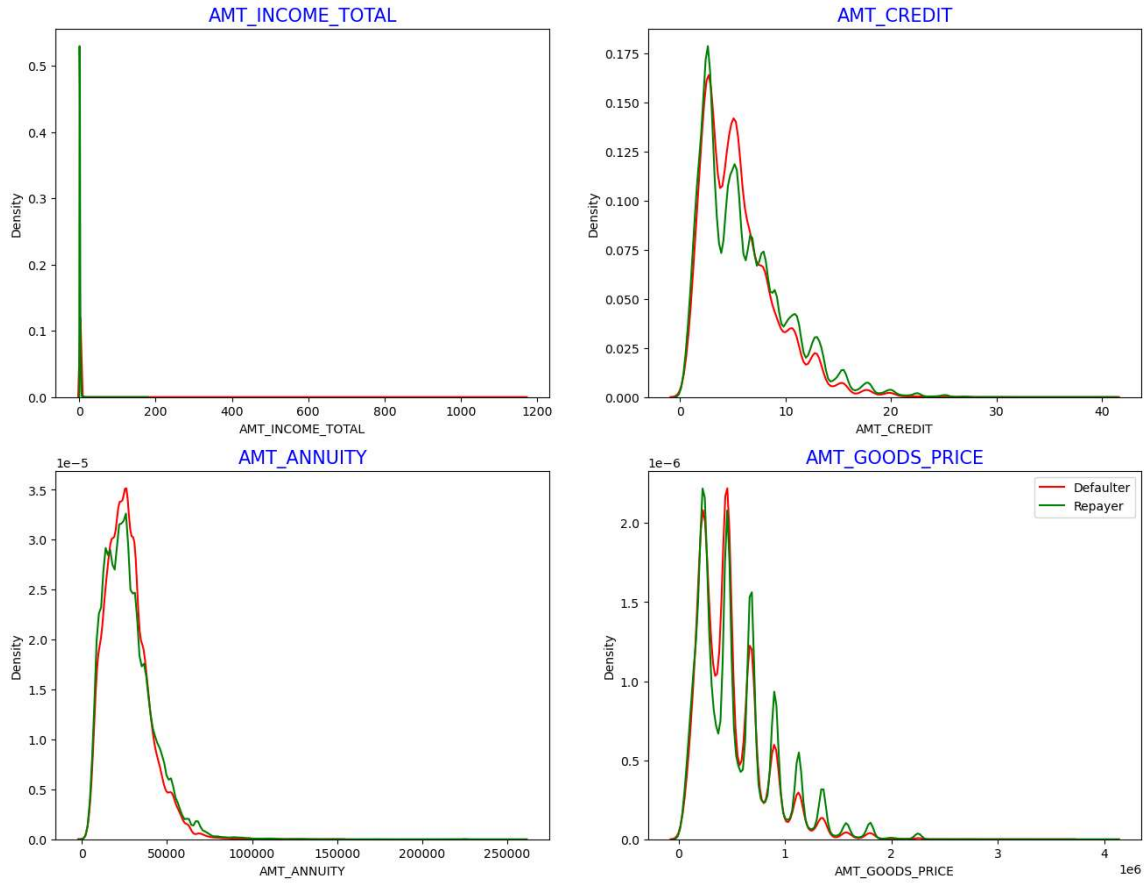
```
# Plotting the numerical columns related to amount as distribution plot
amount = app_data[[ 'AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY', 'AMT

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

for i in enumerate(amount):
    plt.subplot(2,2,i[0]+1)
    sns.distplot(Defaulter_df[i[1]], hist=False, color='r',label ="Defa
    sns.distplot(Repayer_df[i[1]], hist=False, color='g', label ="Repay
    plt.title(i[1], fontdict={'fontsize' : 15, 'fontweight' : 5, 'color

plt.legend()

plt.show()
```

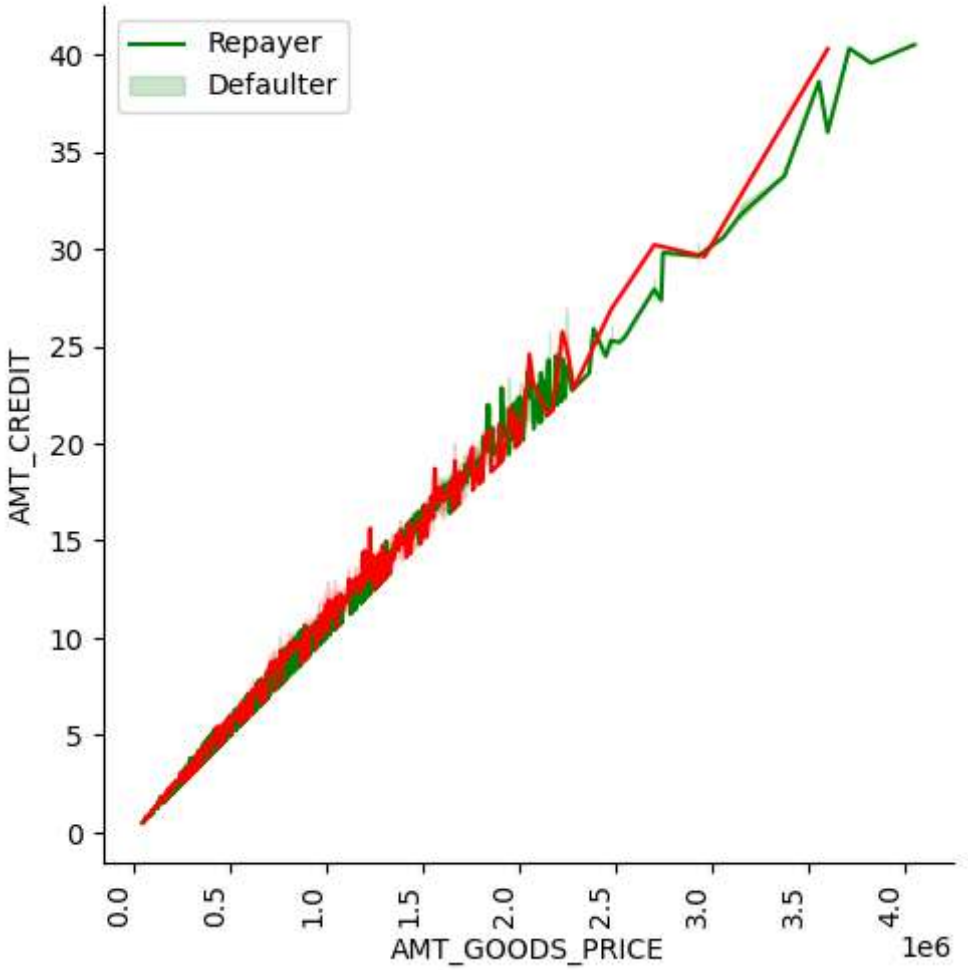


Most no of loans are given for goods price below 10 lakhs Most people pay annuity below 50000 for the credit loan Credit amount of the loan is mostly less then 10 lakhs The repayers and defaulters distribution overlap in all the plots and hence we cannot use any of these variables in isolation to make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

Numerical Bivariate Analysis

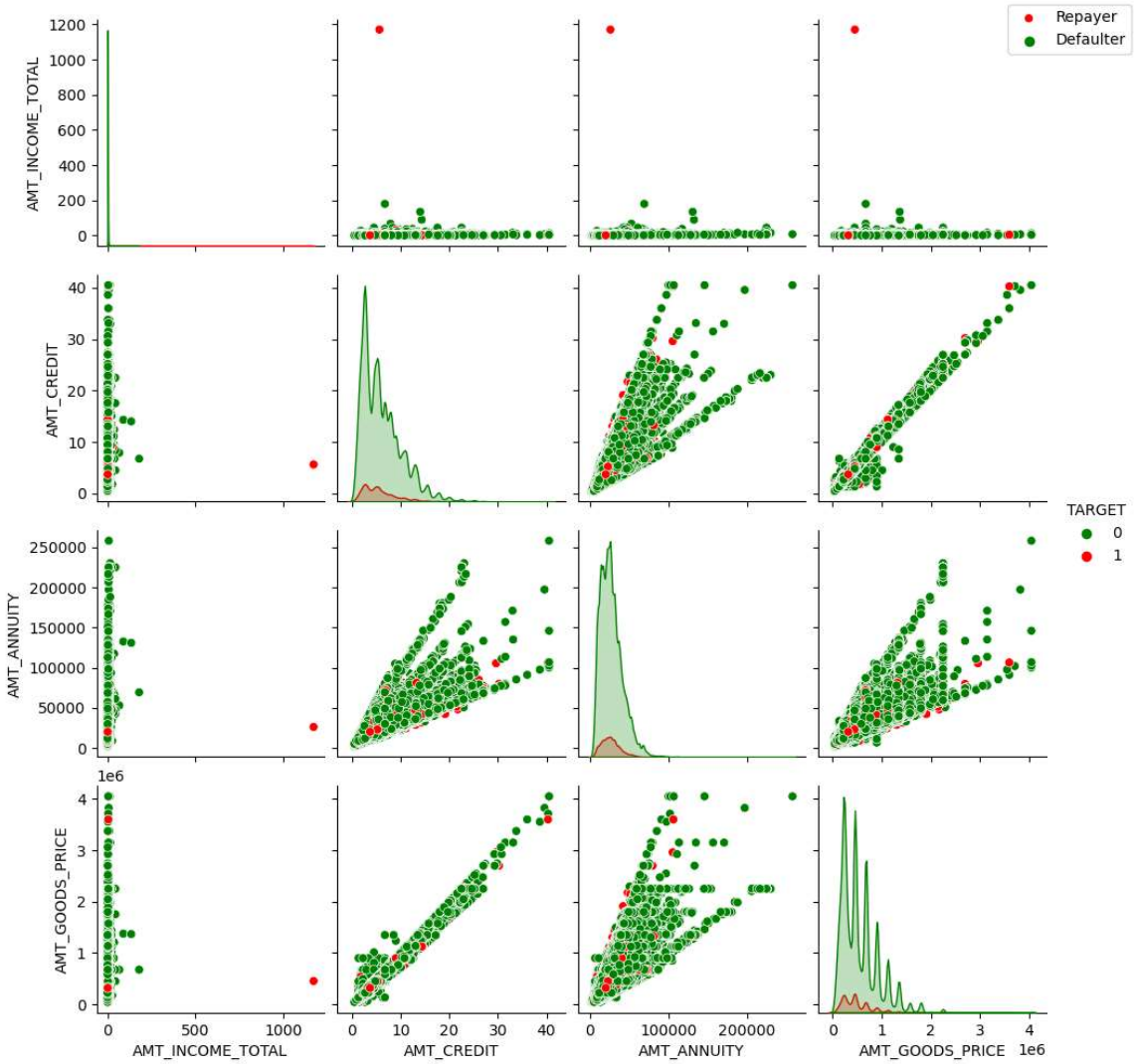
In [126]:
```
# Checking the relationship between Goods price and credit and comparin
bivariate_rel('AMT_GOODS_PRICE','AMT_CREDIT',app_data,"TARGET", "line",
```

```
<Figure size 1500x600 with 0 Axes>
```



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

In [127]:
```
# Plotting pairplot between amount variable to draw reference against l
amount = app_data[[ 'AMT_INCOME_TOTAL','AMT_CREDIT',
                           'AMT_ANNUITY', 'AMT_GOODS_PRICE','TARGET']]
amount = amount[(amount["AMT_GOODS_PRICE"].notnull()) & (amount["AMT_AN
ax= sns.pairplot(amount,hue="TARGET",palette=["g","r"])
ax.fig.legend(labels=['Repayer','Defaulter'])
plt.show()
```



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

Merged Dataframes Analysis

```
Conclusions

Decisive Factor whether an applicant will be Repayer:
NAME_EDUCATION_TYPE: Academic degree has less defaults.
NAME_INCOME_TYPE: Student and Businessmen have no defaults.
REGION_RATING_CLIENT: RATING 1 is safer.
ORGANIZATION_TYPE: Clients with Trade Type 4 and 5 and Industry type
8 have defaulted less than 3%
DAYS_BIRTH: People above age of 50 have low probability of defaulting
```

```
DAYS_EMPLOYED: Clients with 40+ year experience having less than 1% default rate
AMT_INCOME_TOTAL:Applicant with Income more than 700,000 are less likely to default
NAME_CASH_LOAN_PURPOSE: Loans bought for Hobby, Buying garage are being repayed mostly.
CNT_CHILDREN: People with zero to two children tend to repay the loans.


Decisive Factor whether an applicant will be Defaulter:
CODE_GENDER: Men are at relatively higher default rate
NAME_FAMILY_STATUS : People who have civil marriage or who are single default a lot.
NAME_EDUCATION_TYPE: People with Lower Secondary & Secondary education
NAME_INCOME_TYPE: Clients who are either at Maternity leave OR Unemployed default a lot.
REGION_RATING_CLIENT: People who live in Rating 3 has highest defaults.
OCCUPATION_TYPE: Avoid Low-skill Laborers, Drivers and Waiters/barmen staff, Security staff, Laborers
and Cooking staff as the default rate is huge.
ORGANIZATION_TYPE: 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.
DAYS_BIRTH: Avoid young people who are in age group of 20-40 as they have higher probability of
defaulting
DAYS_EMPLOYED: People who have less than 5 years of employment have high default rate.
CNT_CHILDREN & CNT_FAM_MEMBERS: Client who have children equal to or more than 9 default 100% and hence
their applications are to be rejected.
AMT_GOODS_PRICE: When the credit amount goes beyond 3M, there is an increase in defaulters.


NAME_HOUSING_TYPE: High number of loan applications are from the category of people who live in
Rented apartments & living with parents and hence offering the loan would mitigate the loss if any of
those default.
AMT_CREDIT: People who get loan for 300-600k tend to default more than others and hence having higher
interest specifically for this credit range would be ideal.
AMT_INCOME: Since 90% of the applications have Income total less than 300,000 and they have high
probability of defaulting, they could be offered loan with higher interest compared to other income
category.
CNT_CHILDREN & CNT_FAM_MEMBERS: Clients who have 4 to 8 children has a very high default rate and
hence higher interest should be imposed on their loans.
NAME_CASH_LOAN_PURPOSE: Loan taken for the purpose of Repairs seems to have highest default rate.
high loan interest rate which is not feasible by the clients, thus they refuse the loan.
```

```
45 The same approach could be followed in future as well.
```

In [ ]:

```
1
```