EDA : Bank Loan Default Risk Analysis

Business Objective:

This case study aims to identify patterns which indicate if a client has difficulty paying their installments which may be used for taking actions such as denying the loan, reducing the amount of loan, lending (to risky applicants) at a higher interest rate, etc. This will ensure that the consumers capable of repaying the loan are not rejected. Identification of such applicants using EDA is the aim of this case study.

Getting Jupyter Ready:

Importing the Libraries for EDA and Plotting the Graph

Remove Warnings

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.style as style
import seaborn as sns
%matplotlib inline


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

In [3]:
```
#pd.options.display.max_columns = 125
```

In [4]:
```
_ApplicationDF = pd.read_csv(r'D:\Python_Code\EDAProject_BankLoanRiskAnalysis\application_data.csv')
_ApplicationDF.head(20)
```

Out[4]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_T
0	100002	1	Cash loans	M	N	Y	0	20250
1	100003	0	Cash loans	F	N	N	0	27000
2	100004	0	Revolving loans	M	Y	Y	0	6750
3	100006	0	Cash loans	F	N	Y	0	13500
4	100007	0	Cash loans	M	N	Y	0	12150
5	100008	0	Cash loans	M	N	Y	0	9900
6	100009	0	Cash loans	F	Y	Y	1	17100
7	100010	0	Cash loans	M	Y	Y	0	36000
8	100011	0	Cash loans	F	N	Y	0	11250
9	100012	0	Revolving loans	M	N	Y	0	13500
10	100014	0	Cash loans	F	N	Y	1	11250
11	100015	0	Cash loans	F	N	Y	0	3841
12	100016	0	Cash loans	F	N	Y	0	6750
13	100017	0	Cash loans	M	Y	N	1	22500
14	100018	0	Cash loans	F	N	Y	0	18900
15	100019	0	Cash loans	M	Y	Y	0	15750
16	100020	0	Cash loans	M	N	N	0	10800
17	100021	0	Revolving loans	F	N	Y	1	8100
18	100022	0	Revolving loans	F	N	Y	0	11250
19	100023	0	Cash loans	F	N	Y	1	9000

20 rows × 122 columns

In [5]:
```
_ApplicationDF.tail(10)
```

Out[5]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCO
307501	456245	0	Cash loans	F	N	Y	3	
307502	456246	0	Cash loans	F	N	Y	1	
307503	456247	0	Cash loans	F	N	Y	0	
307504	456248	0	Cash loans	F	N	Y	0	
307505	456249	0	Cash loans	F	N	Y	0	
307506	456251	0	Cash loans	M	N	N	0	
307507	456252	0	Cash loans	F	N	Y	0	
307508	456253	0	Cash loans	F	N	Y	0	
307509	456254	1	Cash loans	F	N	Y	0	
307510	456255	0	Cash loans	F	N	N	0	

10 rows × 122 columns

In [6]:
```
_PrevApplicationDF = pd.read_csv(r'D:\Python_Code\EDAProject_BankLoanRiskAnalysis\\previous_application.csv')
_PrevApplicationDF
```

Out[6]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AI
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	0.0	
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	NaN	
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	NaN	
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	NaN	
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	NaN	
...	...	...	...	...	...	...	...	
1670209	2300464	352015	Consumer loans	14704.290	267295.5	311400.0	0.0	
1670210	2357031	334635	Consumer loans	6622.020	87750.0	64291.5	29250.0	
1670211	2659632	249544	Consumer loans	11520.855	105237.0	102523.5	10525.5	
1670212	2785582	400317	Cash loans	18821.520	180000.0	191880.0	NaN	
1670213	2418762	261212	Cash loans	16431.300	360000.0	360000.0	NaN	

1670214 rows × 37 columns

In [7]:
```
ra,ca =_ApplicationDF.shape
rp,cp = _PrevApplicationDF.shape
```

Defining the Dimensions,Rows & Columns and Size of the Data Frame (ApplicationDF and PreviousDF)

In [9]:
```
print('Data Dimension - Application DF  :',_ApplicationDF.shape)
print('Rows and Columns - Application DF:', ra,ca)
print('Data Size - Application DF       :',_ApplicationDF.size)

print('Data Dimension - Previous DF     :',_PrevApplicationDF.shape)
print('Rows and Columns - Previous DF   :', rp,cp)
print('Data Size - Application DF       :',_PrevApplicationDF.size)
```

```
Data Dimension - Application DF  : (307511, 122)
Rows and Columns - Application DF: 307511 122
Data Size - Application DF       : 37516342
Data Dimension - Previous DF     : (1670214, 37)
Rows and Columns - Previous DF   : 1670214 37
Data Size - Application DF       : 61797918
```

In [10]:
```
_ApplicationDF.columns
```

Out[10]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY',
       ...
       'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20',
       'FLAG_DOCUMENT_21', 'AMT_REQ_CREDIT_BUREAU_HOUR',
       'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
       'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT',
       'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object', length=122)
```

In [11]:
```
_PrevApplicationDF.columns
```

Out[11]:
```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_ANNUITY',
       'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_DOWN_PAYMENT', 'AMT_GOODS_PRICE',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       'FLAG_LAST_APPL_PER_CONTRACT', 'NFLAG_LAST_APPL_IN_DAY',
       'RATE_DOWN_PAYMENT', 'RATE_INTEREST_PRIMARY',
       'RATE_INTEREST_PRIVILEGED', 'NAME_CASH_LOAN_PURPOSE',
       'NAME_CONTRACT_STATUS', 'DAYS_DECISION', 'NAME_PAYMENT_TYPE',
       'CODE_REJECT_REASON', 'NAME_TYPE_SUITE', 'NAME_CLIENT_TYPE',
       'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO', 'NAME_PRODUCT_TYPE',
       'CHANNEL_TYPE', 'SELLERPLACE_AREA', 'NAME_SELLER_INDUSTRY',
       'CNT_PAYMENT', 'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION',
       'DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION',
       'DAYS_LAST_DUE', 'DAYS_TERMINATION', 'NFLAG_INSURED_ON_APPROVAL'],
      dtype='object')
```

In [12]:
```
_ApplicationDF.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
 14   NAME_FAMILY_STATUS            object
 15   NAME_HOUSING_TYPE             object
 16   REGION_POPULATION_RELATIVE    float64
 17   DAYS_BIRTH                    int64
 18   DAYS_EMPLOYED                 int64
 19   DAYS_REGISTRATION             float64
 20   DAYS_ID_PUBLISH               int64
 21   OWN_CAR_AGE                   float64
 22   FLAG_MOBIL                    int64
 23   FLAG_EMP_PHONE                int64
 24   FLAG_WORK_PHONE               int64
 25   FLAG_CONT_MOBILE              int64
 26   FLAG_PHONE                    int64
 27   FLAG_EMAIL                    int64
 28   OCCUPATION_TYPE               object
 29   CNT_FAM_MEMBERS               float64
 30   REGION_RATING_CLIENT          int64
 31   REGION_RATING_CLIENT_W_CITY   int64
 32   WEEKDAY_APPR_PROCESS_START    object
 33   HOUR_APPR_PROCESS_START       int64
 34   REG_REGION_NOT_LIVE_REGION    int64
 35   REG_REGION_NOT_WORK_REGION    int64
```

36	LIVE_REGION_NOT_WORK_REGION	int64
37	REG_CITY_NOT_LIVE_CITY	int64
38	REG_CITY_NOT_WORK_CITY	int64
39	LIVE_CITY_NOT_WORK_CITY	int64
40	ORGANIZATION_TYPE	object
41	EXT_SOURCE_1	float64
42	EXT_SOURCE_2	float64
43	EXT_SOURCE_3	float64
44	APARTMENTS_AVG	float64
45	BASEMENTAREA_AVG	float64
46	YEARS_BEGINEXPLUATATION_AVG	float64
47	YEARS_BUILD_AVG	float64
48	COMMONAREA_AVG	float64
49	ELEVATORS_AVG	float64
50	ENTRANCES_AVG	float64
51	FLOORSMAX_AVG	float64
52	FLOORSMIN_AVG	float64
53	LANDAREA_AVG	float64
54	LIVINGAPARTMENTS_AVG	float64
55	LIVINGAREA_AVG	float64
56	NONLIVINGAPARTMENTS_AVG	float64
57	NONLIVINGAREA_AVG	float64
58	APARTMENTS_MODE	float64
59	BASEMENTAREA_MODE	float64
60	YEARS_BEGINEXPLUATATION_MODE	float64
61	YEARS_BUILD_MODE	float64
62	COMMONAREA_MODE	float64
63	ELEVATORS_MODE	float64
64	ENTRANCES_MODE	float64
65	FLOORSMAX_MODE	float64
66	FLOORSMIN_MODE	float64
67	LANDAREA_MODE	float64
68	LIVINGAPARTMENTS_MODE	float64
69	LIVINGAREA_MODE	float64
70	NONLIVINGAPARTMENTS_MODE	float64
71	NONLIVINGAREA_MODE	float64
72	APARTMENTS_MEDI	float64
73	BASEMENTAREA_MEDI	float64
74	YEARS_BEGINEXPLUATATION_MEDI	float64
75	YEARS_BUILD_MEDI	float64
76	COMMONAREA_MEDI	float64

77	ELEVATORS_MEDI	float64
78	ENTRANCES_MEDI	float64
79	FLOORSMAX_MEDI	float64
80	FLOORSMIN_MEDI	float64
81	LANDAREA_MEDI	float64
82	LIVINGAPARTMENTS_MEDI	float64
83	LIVINGAREA_MEDI	float64
84	NONLIVINGAPARTMENTS_MEDI	float64
85	NONLIVINGAREA_MEDI	float64
86	FONDKAPREMONT_MODE	object
87	HOUSETYPE_MODE	object
88	TOTALAREA_MODE	float64
89	WALLSMATERIAL_MODE	object
90	EMERGENCYSTATE_MODE	object
91	OBS_30_CNT_SOCIAL_CIRCLE	float64
92	DEF_30_CNT_SOCIAL_CIRCLE	float64
93	OBS_60_CNT_SOCIAL_CIRCLE	float64
94	DEF_60_CNT_SOCIAL_CIRCLE	float64
95	DAYS_LAST_PHONE_CHANGE	float64
96	FLAG_DOCUMENT_2	int64
97	FLAG_DOCUMENT_3	int64
98	FLAG_DOCUMENT_4	int64
99	FLAG_DOCUMENT_5	int64
100	FLAG_DOCUMENT_6	int64
101	FLAG_DOCUMENT_7	int64
102	FLAG_DOCUMENT_8	int64
103	FLAG_DOCUMENT_9	int64
104	FLAG_DOCUMENT_10	int64
105	FLAG_DOCUMENT_11	int64
106	FLAG_DOCUMENT_12	int64
107	FLAG_DOCUMENT_13	int64
108	FLAG_DOCUMENT_14	int64
109	FLAG_DOCUMENT_15	int64
110	FLAG_DOCUMENT_16	int64
111	FLAG_DOCUMENT_17	int64
112	FLAG_DOCUMENT_18	int64
113	FLAG_DOCUMENT_19	int64
114	FLAG_DOCUMENT_20	int64
115	FLAG_DOCUMENT_21	int64
116	AMT_REQ_CREDIT_BUREAU_HOUR	float64
117	AMT_REQ_CREDIT_BUREAU_DAY	float64

```
 118  AMT_REQ_CREDIT_BUREAU_WEEK    float64
 119  AMT_REQ_CREDIT_BUREAU_MON     float64
 120  AMT_REQ_CREDIT_BUREAU_QRT     float64
 121  AMT_REQ_CREDIT_BUREAU_YEAR    float64
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

In [13]:
```
_PrevApplicationDF.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
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object 
 16  NAME_CONTRACT_STATUS         1670214 non-null  object 
 17  DAYS_DECISION                1670214 non-null  int64  
 18  NAME_PAYMENT_TYPE            1670214 non-null  object 
 19  CODE_REJECT_REASON           1670214 non-null  object 
 20  NAME_TYPE_SUITE              849809 non-null   object 
 21  NAME_CLIENT_TYPE             1670214 non-null  object 
 22  NAME_GOODS_CATEGORY          1670214 non-null  object 
 23  NAME_PORTFOLIO               1670214 non-null  object 
 24  NAME_PRODUCT_TYPE            1670214 non-null  object 
 25  CHANNEL_TYPE                 1670214 non-null  object 
 26  SELLERPLACE_AREA             1670214 non-null  int64  
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object 
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object 
 30  PRODUCT_COMBINATION          1669868 non-null  object 
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
```

```
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
```

In [14]:
```
_ApplicationDF.describe()
```

Out[14]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_P
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	307499.000000	3.072330e+05	
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	27108.573909	5.383962e+05	
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	14493.737315	3.694465e+05	
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1615.500000	4.050000e+04	
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16524.000000	2.385000e+05	
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	24903.000000	4.500000e+05	
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.000000	6.795000e+05	
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	258025.500000	4.050000e+06	

8 rows × 106 columns

In [15]:
```
_PrevApplicationDF.describe()
```

Out[15]:

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	HOUR
count	1.670214e+06	1.670214e+06	1.297979e+06	1.670214e+06	1.670213e+06	7.743700e+05	1.284699e+06	
mean	1.923089e+06	2.783572e+05	1.595512e+04	1.752339e+05	1.961140e+05	6.697402e+03	2.278473e+05	
std	5.325980e+05	1.028148e+05	1.478214e+04	2.927798e+05	3.185746e+05	2.092150e+04	3.153966e+05	
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00	0.000000e+00	-9.000000e-01	0.000000e+00	
25%	1.461857e+06	1.893290e+05	6.321780e+03	1.872000e+04	2.416050e+04	0.000000e+00	5.084100e+04	
50%	1.923110e+06	2.787145e+05	1.125000e+04	7.104600e+04	8.054100e+04	1.638000e+03	1.123200e+05	
75%	2.384280e+06	3.675140e+05	2.065842e+04	1.803600e+05	2.164185e+05	7.740000e+03	2.340000e+05	
max	2.845382e+06	4.562550e+05	4.180581e+05	6.905160e+06	6.905160e+06	3.060045e+06	6.905160e+06	

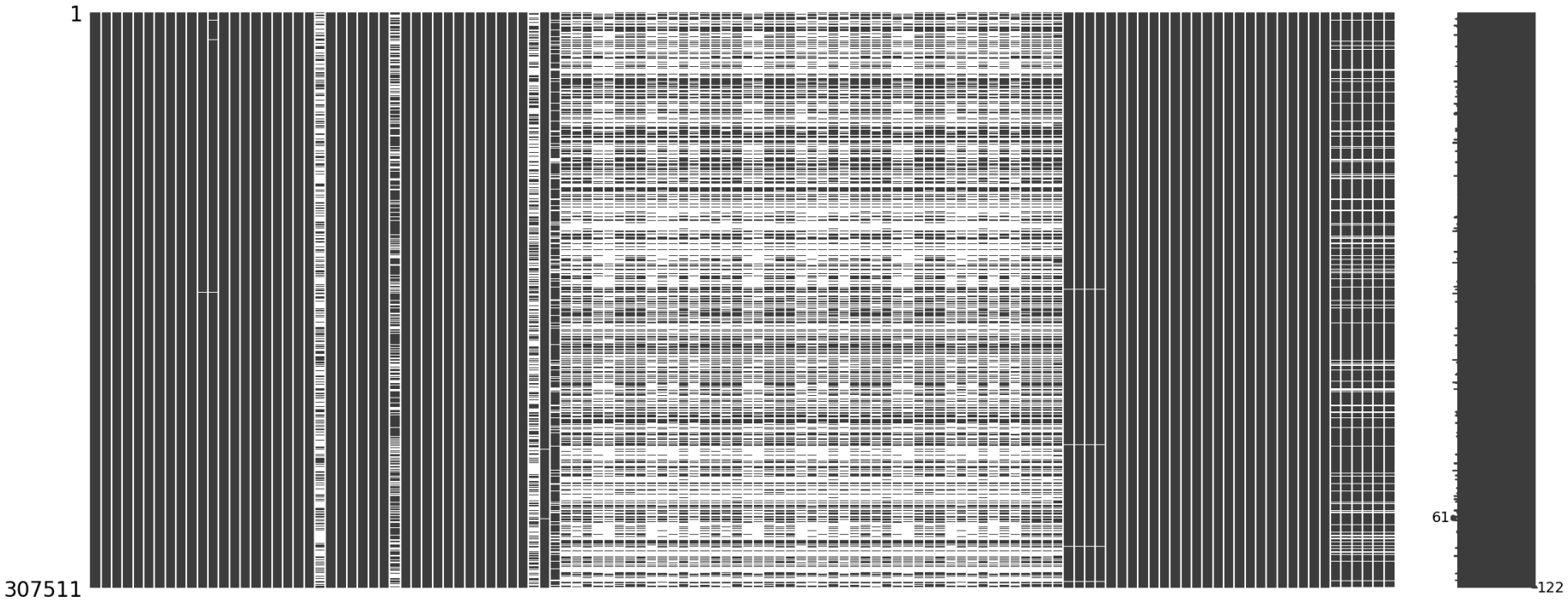
8 rows × 21 columns

Plotting Missing value graph in ApplicationDF

In [17]:
```
import missingno as mn
```

In [18]:
```
#pip install missingno
```

In [19]:
```
#import missingno as mn
mn.matrix(_ApplicationDF)
plt.show()
```



In [20]:
```
#to display all the columns in the Dataframe.
pd.options.display.max_rows = 125
```

In [21]:
```
_ApplicationDF.isnull().sum()
```

Out[21]:

SK_ID_CURR	0
TARGET	0
NAME_CONTRACT_TYPE	0
CODE_GENDER	0
FLAG_OWN_CAR	0
FLAG_OWN_REALTY	0
CNT_CHILDREN	0
AMT_INCOME_TOTAL	0
AMT_CREDIT	0
AMT_ANNUITY	12
AMT_GOODS_PRICE	278
NAME_TYPE_SUITE	1292
NAME_INCOME_TYPE	0
NAME_EDUCATION_TYPE	0
NAME_FAMILY_STATUS	0
NAME_HOUSING_TYPE	0
REGION_POPULATION_RELATIVE	0
DAYS_BIRTH	0
DAYS_EMPLOYED	0
DAYS_REGISTRATION	0
DAYS_ID_PUBLISH	0
OWN_CAR_AGE	202929
FLAG_MOBIL	0
FLAG_EMP_PHONE	0
FLAG_WORK_PHONE	0
FLAG_CONT_MOBILE	0
FLAG_PHONE	0
FLAG_EMAIL	0
OCCUPATION_TYPE	96391
CNT_FAM_MEMBERS	2
REGION_RATING_CLIENT	0
REGION_RATING_CLIENT_W_CITY	0
WEEKDAY_APPR_PROCESS_START	0
HOUR_APPR_PROCESS_START	0
REG_REGION_NOT_LIVE_REGION	0
REG_REGION_NOT_WORK_REGION	0
LIVE_REGION_NOT_WORK_REGION	0
REG_CITY_NOT_LIVE_CITY	0
REG_CITY_NOT_WORK_CITY	0
LIVE_CITY_NOT_WORK_CITY	0

```
ORGANIZATION_TYPE                 0
EXT_SOURCE_1                 173378
EXT_SOURCE_2                    660
EXT_SOURCE_3                  60965
APARTMENTS_AVG               156061
BASEMENTAREA_AVG             179943
YEARS_BEGINEXPLUATATION_AVG  150007
YEARS_BUILD_AVG              204488
COMMONAREA_AVG               214865
ELEVATORS_AVG                163891
ENTRANCES_AVG                154828
FLOORSMAX_AVG                153020
FLOORSMIN_AVG                208642
LANDAREA_AVG                 182590
LIVINGAPARTMENTS_AVG         210199
LIVINGAREA_AVG               154350
NONLIVINGAPARTMENTS_AVG      213514
NONLIVINGAREA_AVG            169682
APARTMENTS_MODE              156061
BASEMENTAREA_MODE            179943
YEARS_BEGINEXPLUATATION_MODE 150007
YEARS_BUILD_MODE             204488
COMMONAREA_MODE              214865
ELEVATORS_MODE               163891
ENTRANCES_MODE               154828
FLOORSMAX_MODE               153020
FLOORSMIN_MODE               208642
LANDAREA_MODE                182590
LIVINGAPARTMENTS_MODE        210199
LIVINGAREA_MODE              154350
NONLIVINGAPARTMENTS_MODE     213514
NONLIVINGAREA_MODE           169682
APARTMENTS_MEDI              156061
BASEMENTAREA_MEDI            179943
YEARS_BEGINEXPLUATATION_MEDI 150007
YEARS_BUILD_MEDI             204488
COMMONAREA_MEDI              214865
ELEVATORS_MEDI               163891
ENTRANCES_MEDI               154828
FLOORSMAX_MEDI               153020
FLOORSMIN_MEDI               208642
```

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

```
AMT_REQ_CREDIT_BUREAU_YEAR       41519
dtype: int64
```

In [22]:
```
np.round((_ApplicationDF.isnull().sum()/_ApplicationDF.shape[0]) * 100,2)
```

Out[22]:

SK_ID_CURR	0.00
TARGET	0.00
NAME_CONTRACT_TYPE	0.00
CODE_GENDER	0.00
FLAG_OWN_CAR	0.00
FLAG_OWN_REALTY	0.00
CNT_CHILDREN	0.00
AMT_INCOME_TOTAL	0.00
AMT_CREDIT	0.00
AMT_ANNUITY	0.00
AMT_GOODS_PRICE	0.09
NAME_TYPE_SUITE	0.42
NAME_INCOME_TYPE	0.00
NAME_EDUCATION_TYPE	0.00
NAME_FAMILY_STATUS	0.00
NAME_HOUSING_TYPE	0.00
REGION_POPULATION_RELATIVE	0.00
DAYS_BIRTH	0.00
DAYS_EMPLOYED	0.00
DAYS_REGISTRATION	0.00
DAYS_ID_PUBLISH	0.00
OWN_CAR_AGE	65.99
FLAG_MOBIL	0.00
FLAG_EMP_PHONE	0.00
FLAG_WORK_PHONE	0.00
FLAG_CONT_MOBILE	0.00
FLAG_PHONE	0.00
FLAG_EMAIL	0.00
OCCUPATION_TYPE	31.35
CNT_FAM_MEMBERS	0.00
REGION_RATING_CLIENT	0.00
REGION_RATING_CLIENT_W_CITY	0.00
WEEKDAY_APPR_PROCESS_START	0.00
HOUR_APPR_PROCESS_START	0.00
REG_REGION_NOT_LIVE_REGION	0.00
REG_REGION_NOT_WORK_REGION	0.00
LIVE_REGION_NOT_WORK_REGION	0.00
REG_CITY_NOT_LIVE_CITY	0.00
REG_CITY_NOT_WORK_CITY	0.00
LIVE_CITY_NOT_WORK_CITY	0.00

ORGANIZATION_TYPE	0.00
EXT_SOURCE_1	56.38
EXT_SOURCE_2	0.21
EXT_SOURCE_3	19.83
APARTMENTS_AVG	50.75
BASEMENTAREA_AVG	58.52
YEARS_BEGINEXPLUATATION_AVG	48.78
YEARS_BUILD_AVG	66.50
COMMONAREA_AVG	69.87
ELEVATORS_AVG	53.30
ENTRANCES_AVG	50.35
FLOORSMAX_AVG	49.76
FLOORSMIN_AVG	67.85
LANDAREA_AVG	59.38
LIVINGAPARTMENTS_AVG	68.35
LIVINGAREA_AVG	50.19
NONLIVINGAPARTMENTS_AVG	69.43
NONLIVINGAREA_AVG	55.18
APARTMENTS_MODE	50.75
BASEMENTAREA_MODE	58.52
YEARS_BEGINEXPLUATATION_MODE	48.78
YEARS_BUILD_MODE	66.50
COMMONAREA_MODE	69.87
ELEVATORS_MODE	53.30
ENTRANCES_MODE	50.35
FLOORSMAX_MODE	49.76
FLOORSMIN_MODE	67.85
LANDAREA_MODE	59.38
LIVINGAPARTMENTS_MODE	68.35
LIVINGAREA_MODE	50.19
NONLIVINGAPARTMENTS_MODE	69.43
NONLIVINGAREA_MODE	55.18
APARTMENTS_MEDI	50.75
BASEMENTAREA_MEDI	58.52
YEARS_BEGINEXPLUATATION_MEDI	48.78
YEARS_BUILD_MEDI	66.50
COMMONAREA_MEDI	69.87
ELEVATORS_MEDI	53.30
ENTRANCES_MEDI	50.35
FLOORSMAX_MEDI	49.76
FLOORSMIN_MEDI	67.85

```
LANDAREA_MEDI                 59.38
LIVINGAPARTMENTS_MEDI         68.35
LIVINGAREA_MEDI               50.19
NONLIVINGAPARTMENTS_MEDI      69.43
NONLIVINGAREA_MEDI            55.18
FONDKAPREMONT_MODE            68.39
HOUSETYPE_MODE                50.18
TOTALAREA_MODE                48.27
WALLSMATERIAL_MODE            50.84
EMERGENCYSTATE_MODE           47.40
OBS_30_CNT_SOCIAL_CIRCLE       0.33
DEF_30_CNT_SOCIAL_CIRCLE       0.33
OBS_60_CNT_SOCIAL_CIRCLE       0.33
DEF_60_CNT_SOCIAL_CIRCLE       0.33
DAYS_LAST_PHONE_CHANGE         0.00
FLAG_DOCUMENT_2                0.00
FLAG_DOCUMENT_3                0.00
FLAG_DOCUMENT_4                0.00
FLAG_DOCUMENT_5                0.00
FLAG_DOCUMENT_6                0.00
FLAG_DOCUMENT_7                0.00
FLAG_DOCUMENT_8                0.00
FLAG_DOCUMENT_9                0.00
FLAG_DOCUMENT_10               0.00
FLAG_DOCUMENT_11               0.00
FLAG_DOCUMENT_12               0.00
FLAG_DOCUMENT_13               0.00
FLAG_DOCUMENT_14               0.00
FLAG_DOCUMENT_15               0.00
FLAG_DOCUMENT_16               0.00
FLAG_DOCUMENT_17               0.00
FLAG_DOCUMENT_18               0.00
FLAG_DOCUMENT_19               0.00
FLAG_DOCUMENT_20               0.00
FLAG_DOCUMENT_21               0.00
AMT_REQ_CREDIT_BUREAU_HOUR    13.50
AMT_REQ_CREDIT_BUREAU_DAY     13.50
AMT_REQ_CREDIT_BUREAU_WEEK    13.50
AMT_REQ_CREDIT_BUREAU_MON     13.50
AMT_REQ_CREDIT_BUREAU_QRT     13.50
```

```
AMT_REQ_CREDIT_BUREAU_YEAR      13.50
dtype: float64
```

INSIGHTS FROM THE DATASET

1. There are many columns in applicationDF dataframe where missing value is more than 40%.

2.Let's plot the columns vs missing value % with 40% being the cut-off marks.

In [24]:
```
Null_Values_Percent_ApplicationDF = pd.DataFrame(round((_ApplicationDF.isnull().sum()/_ApplicationDF.shape[0])*100,2)).reset_i
Null_Values_Percent_ApplicationDF
```

Out[24]:

	index	0
0	SK_ID_CURR	0.00
1	TARGET	0.00
2	NAME_CONTRACT_TYPE	0.00
3	CODE_GENDER	0.00
4	FLAG_OWN_CAR	0.00
5	FLAG_OWN_REALTY	0.00
6	CNT_CHILDREN	0.00
7	AMT_INCOME_TOTAL	0.00
8	AMT_CREDIT	0.00
9	AMT_ANNUITY	0.00
10	AMT_GOODS_PRICE	0.09
11	NAME_TYPE_SUITE	0.42
12	NAME_INCOME_TYPE	0.00
13	NAME_EDUCATION_TYPE	0.00
14	NAME_FAMILY_STATUS	0.00
15	NAME_HOUSING_TYPE	0.00
16	REGION_POPULATION_RELATIVE	0.00
17	DAYS_BIRTH	0.00
18	DAYS_EMPLOYED	0.00
19	DAYS_REGISTRATION	0.00
20	DAYS_ID_PUBLISH	0.00
21	OWN_CAR_AGE	65.99

	index	0
22	FLAG_MOBIL	0.00
23	FLAG_EMP_PHONE	0.00
24	FLAG_WORK_PHONE	0.00
25	FLAG_CONT_MOBILE	0.00
26	FLAG_PHONE	0.00
27	FLAG_EMAIL	0.00
28	OCCUPATION_TYPE	31.35
29	CNT_FAM_MEMBERS	0.00
30	REGION_RATING_CLIENT	0.00
31	REGION_RATING_CLIENT_W_CITY	0.00
32	WEEKDAY_APPR_PROCESS_START	0.00
33	HOUR_APPR_PROCESS_START	0.00
34	REG_REGION_NOT_LIVE_REGION	0.00
35	REG_REGION_NOT_WORK_REGION	0.00
36	LIVE_REGION_NOT_WORK_REGION	0.00
37	REG_CITY_NOT_LIVE_CITY	0.00
38	REG_CITY_NOT_WORK_CITY	0.00
39	LIVE_CITY_NOT_WORK_CITY	0.00
40	ORGANIZATION_TYPE	0.00
41	EXT_SOURCE_1	56.38
42	EXT_SOURCE_2	0.21
43	EXT_SOURCE_3	19.83

	index	0
44	APARTMENTS_AVG	50.75
45	BASEMENTAREA_AVG	58.52
46	YEARS_BEGINEXPLUATATION_AVG	48.78
47	YEARS_BUILD_AVG	66.50
48	COMMONAREA_AVG	69.87
49	ELEVATORS_AVG	53.30
50	ENTRANCES_AVG	50.35
51	FLOORSMAX_AVG	49.76
52	FLOORSMIN_AVG	67.85
53	LANDAREA_AVG	59.38
54	LIVINGAPARTMENTS_AVG	68.35
55	LIVINGAREA_AVG	50.19
56	NONLIVINGAPARTMENTS_AVG	69.43
57	NONLIVINGAREA_AVG	55.18
58	APARTMENTS_MODE	50.75
59	BASEMENTAREA_MODE	58.52
60	YEARS_BEGINEXPLUATATION_MODE	48.78
61	YEARS_BUILD_MODE	66.50
62	COMMONAREA_MODE	69.87
63	ELEVATORS_MODE	53.30
64	ENTRANCES_MODE	50.35
65	FLOORSMAX_MODE	49.76

	index	0
66	FLOORSMIN_MODE	67.85
67	LANDAREA_MODE	59.38
68	LIVINGAPARTMENTS_MODE	68.35
69	LIVINGAREA_MODE	50.19
70	NONLIVINGAPARTMENTS_MODE	69.43
71	NONLIVINGAREA_MODE	55.18
72	APARTMENTS_MEDI	50.75
73	BASEMENTAREA_MEDI	58.52
74	YEARS_BEGINEXPLUATATION_MEDI	48.78
75	YEARS_BUILD_MEDI	66.50
76	COMMONAREA_MEDI	69.87
77	ELEVATORS_MEDI	53.30
78	ENTRANCES_MEDI	50.35
79	FLOORSMAX_MEDI	49.76
80	FLOORSMIN_MEDI	67.85
81	LANDAREA_MEDI	59.38
82	LIVINGAPARTMENTS_MEDI	68.35
83	LIVINGAREA_MEDI	50.19
84	NONLIVINGAPARTMENTS_MEDI	69.43
85	NONLIVINGAREA_MEDI	55.18
86	FONDKAPREMONT_MODE	68.39
87	HOUSETYPE_MODE	50.18

	index	0
88	TOTALAREA_MODE	48.27
89	WALLSMATERIAL_MODE	50.84
90	EMERGENCYSTATE_MODE	47.40
91	OBS_30_CNT_SOCIAL_CIRCLE	0.33
92	DEF_30_CNT_SOCIAL_CIRCLE	0.33
93	OBS_60_CNT_SOCIAL_CIRCLE	0.33
94	DEF_60_CNT_SOCIAL_CIRCLE	0.33
95	DAYS_LAST_PHONE_CHANGE	0.00
96	FLAG_DOCUMENT_2	0.00
97	FLAG_DOCUMENT_3	0.00
98	FLAG_DOCUMENT_4	0.00
99	FLAG_DOCUMENT_5	0.00
100	FLAG_DOCUMENT_6	0.00
101	FLAG_DOCUMENT_7	0.00
102	FLAG_DOCUMENT_8	0.00
103	FLAG_DOCUMENT_9	0.00
104	FLAG_DOCUMENT_10	0.00
105	FLAG_DOCUMENT_11	0.00
106	FLAG_DOCUMENT_12	0.00
107	FLAG_DOCUMENT_13	0.00
108	FLAG_DOCUMENT_14	0.00
109	FLAG_DOCUMENT_15	0.00

	index	0
110	FLAG_DOCUMENT_16	0.00
111	FLAG_DOCUMENT_17	0.00
112	FLAG_DOCUMENT_18	0.00
113	FLAG_DOCUMENT_19	0.00
114	FLAG_DOCUMENT_20	0.00
115	FLAG_DOCUMENT_21	0.00
116	AMT_REQ_CREDIT_BUREAU_HOUR	13.50
117	AMT_REQ_CREDIT_BUREAU_DAY	13.50
118	AMT_REQ_CREDIT_BUREAU_WEEK	13.50
119	AMT_REQ_CREDIT_BUREAU_MON	13.50
120	AMT_REQ_CREDIT_BUREAU_QRT	13.50
121	AMT_REQ_CREDIT_BUREAU_YEAR	13.50

In [25]:
```
Null_Values_Percent_ApplicationDF.columns = ['Column_Names','Null_Value_Percentage']
Null_Values_Percent_ApplicationDF
```

Out[25]:

	Column_Names	Null_Value_Percentage
0	SK_ID_CURR	0.00
1	TARGET	0.00
2	NAME_CONTRACT_TYPE	0.00
3	CODE_GENDER	0.00
4	FLAG_OWN_CAR	0.00
5	FLAG_OWN_REALTY	0.00
6	CNT_CHILDREN	0.00
7	AMT_INCOME_TOTAL	0.00
8	AMT_CREDIT	0.00
9	AMT_ANNUITY	0.00
10	AMT_GOODS_PRICE	0.09
11	NAME_TYPE_SUITE	0.42
12	NAME_INCOME_TYPE	0.00
13	NAME_EDUCATION_TYPE	0.00
14	NAME_FAMILY_STATUS	0.00
15	NAME_HOUSING_TYPE	0.00
16	REGION_POPULATION_RELATIVE	0.00
17	DAYS_BIRTH	0.00
18	DAYS_EMPLOYED	0.00
19	DAYS_REGISTRATION	0.00
20	DAYS_ID_PUBLISH	0.00
21	OWN_CAR_AGE	65.99

	Column_Names	Null_Value_Percentage
22	FLAG_MOBIL	0.00
23	FLAG_EMP_PHONE	0.00
24	FLAG_WORK_PHONE	0.00
25	FLAG_CONT_MOBILE	0.00
26	FLAG_PHONE	0.00
27	FLAG_EMAIL	0.00
28	OCCUPATION_TYPE	31.35
29	CNT_FAM_MEMBERS	0.00
30	REGION_RATING_CLIENT	0.00
31	REGION_RATING_CLIENT_W_CITY	0.00
32	WEEKDAY_APPR_PROCESS_START	0.00
33	HOUR_APPR_PROCESS_START	0.00
34	REG_REGION_NOT_LIVE_REGION	0.00
35	REG_REGION_NOT_WORK_REGION	0.00
36	LIVE_REGION_NOT_WORK_REGION	0.00
37	REG_CITY_NOT_LIVE_CITY	0.00
38	REG_CITY_NOT_WORK_CITY	0.00
39	LIVE_CITY_NOT_WORK_CITY	0.00
40	ORGANIZATION_TYPE	0.00
41	EXT_SOURCE_1	56.38
42	EXT_SOURCE_2	0.21
43	EXT_SOURCE_3	19.83

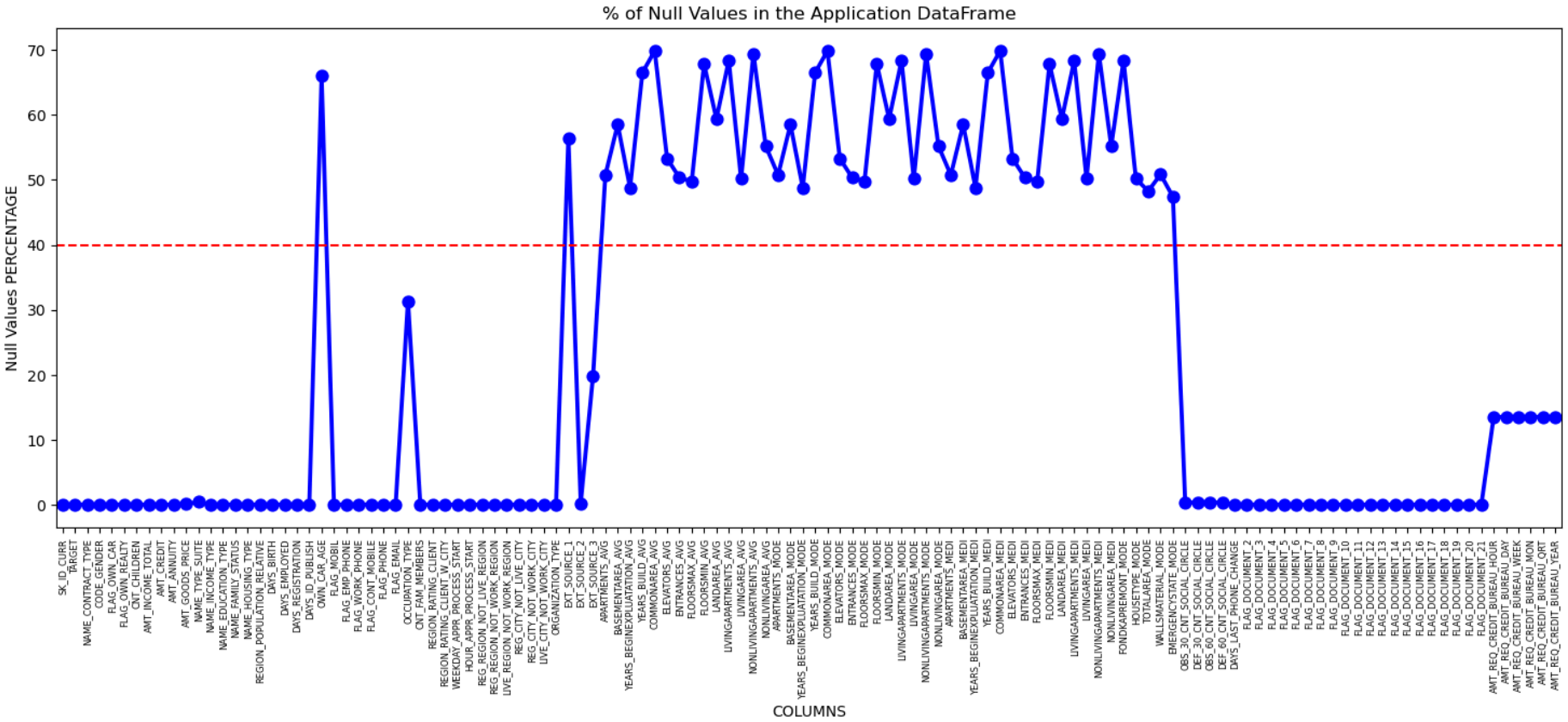
	Column_Names	Null_Value_Percentage
44	APARTMENTS_AVG	50.75
45	BASEMENTAREA_AVG	58.52
46	YEARS_BEGINEXPLUATATION_AVG	48.78
47	YEARS_BUILD_AVG	66.50
48	COMMONAREA_AVG	69.87
49	ELEVATORS_AVG	53.30
50	ENTRANCES_AVG	50.35
51	FLOORSMAX_AVG	49.76
52	FLOORSMIN_AVG	67.85
53	LANDAREA_AVG	59.38
54	LIVINGAPARTMENTS_AVG	68.35
55	LIVINGAREA_AVG	50.19
56	NONLIVINGAPARTMENTS_AVG	69.43
57	NONLIVINGAREA_AVG	55.18
58	APARTMENTS_MODE	50.75
59	BASEMENTAREA_MODE	58.52
60	YEARS_BEGINEXPLUATATION_MODE	48.78
61	YEARS_BUILD_MODE	66.50
62	COMMONAREA_MODE	69.87
63	ELEVATORS_MODE	53.30
64	ENTRANCES_MODE	50.35
65	FLOORSMAX_MODE	49.76

	Column_Names	Null_Value_Percentage
66	FLOORSMIN_MODE	67.85
67	LANDAREA_MODE	59.38
68	LIVINGAPARTMENTS_MODE	68.35
69	LIVINGAREA_MODE	50.19
70	NONLIVINGAPARTMENTS_MODE	69.43
71	NONLIVINGAREA_MODE	55.18
72	APARTMENTS_MEDI	50.75
73	BASEMENTAREA_MEDI	58.52
74	YEARS_BEGINEXPLUATATION_MEDI	48.78
75	YEARS_BUILD_MEDI	66.50
76	COMMONAREA_MEDI	69.87
77	ELEVATORS_MEDI	53.30
78	ENTRANCES_MEDI	50.35
79	FLOORSMAX_MEDI	49.76
80	FLOORSMIN_MEDI	67.85
81	LANDAREA_MEDI	59.38
82	LIVINGAPARTMENTS_MEDI	68.35
83	LIVINGAREA_MEDI	50.19
84	NONLIVINGAPARTMENTS_MEDI	69.43
85	NONLIVINGAREA_MEDI	55.18
86	FONDKAPREMONT_MODE	68.39
87	HOUSETYPE_MODE	50.18

	Column_Names	Null_Value_Percentage
88	TOTALAREA_MODE	48.27
89	WALLSMATERIAL_MODE	50.84
90	EMERGENCYSTATE_MODE	47.40
91	OBS_30_CNT_SOCIAL_CIRCLE	0.33
92	DEF_30_CNT_SOCIAL_CIRCLE	0.33
93	OBS_60_CNT_SOCIAL_CIRCLE	0.33
94	DEF_60_CNT_SOCIAL_CIRCLE	0.33
95	DAYS_LAST_PHONE_CHANGE	0.00
96	FLAG_DOCUMENT_2	0.00
97	FLAG_DOCUMENT_3	0.00
98	FLAG_DOCUMENT_4	0.00
99	FLAG_DOCUMENT_5	0.00
100	FLAG_DOCUMENT_6	0.00
101	FLAG_DOCUMENT_7	0.00
102	FLAG_DOCUMENT_8	0.00
103	FLAG_DOCUMENT_9	0.00
104	FLAG_DOCUMENT_10	0.00
105	FLAG_DOCUMENT_11	0.00
106	FLAG_DOCUMENT_12	0.00
107	FLAG_DOCUMENT_13	0.00
108	FLAG_DOCUMENT_14	0.00
109	FLAG_DOCUMENT_15	0.00

	Column_Names	Null_Value_Percentage
110	FLAG_DOCUMENT_16	0.00
111	FLAG_DOCUMENT_17	0.00
112	FLAG_DOCUMENT_18	0.00
113	FLAG_DOCUMENT_19	0.00
114	FLAG_DOCUMENT_20	0.00
115	FLAG_DOCUMENT_21	0.00
116	AMT_REQ_CREDIT_BUREAU_HOUR	13.50
117	AMT_REQ_CREDIT_BUREAU_DAY	13.50
118	AMT_REQ_CREDIT_BUREAU_WEEK	13.50
119	AMT_REQ_CREDIT_BUREAU_MON	13.50
120	AMT_REQ_CREDIT_BUREAU_QRT	13.50
121	AMT_REQ_CREDIT_BUREAU_YEAR	13.50

In [26]:
```
ax = sns.pointplot(x="Column_Names",y="Null_Value_Percentage",data=Null_Values_Percent_ApplicationDF,color='blue')
#fig = plt.figure(figsize=(10,6))
#plt.figure(figsize=(8.27,11.7))
plt.gcf().set_size_inches(18,6)
plt.xticks(rotation =90,fontsize =6)
ax.axhline(40, ls='--',color='red')
plt.title('% of Null Values in the Application DataFrame')
plt.ylabel("Null Values PERCENTAGE")
plt.xlabel("COLUMNS")
plt.show()
```



Insight:

From the plot we can see:

1)The columns in which percentage of null values more than 40% are marked above the red line .

2) The columns which have less than 40 % null values below the red line.

Let's check the columns which has more than 40% missing values

In [28]:
```
NullCol_Gr8Than_40Percent_ApplicationDF = Null_Values_Percent_ApplicationDF[Null_Values_Percent_ApplicationDF['Null_Value_Perc
NullCol_Gr8Than_40Percent_ApplicationDF
```

Out[28]:

	Column_Names	Null_Value_Percentage
21	OWN_CAR_AGE	65.99
41	EXT_SOURCE_1	56.38
44	APARTMENTS_AVG	50.75
45	BASEMENTAREA_AVG	58.52
46	YEARS_BEGINEXPLUATATION_AVG	48.78
47	YEARS_BUILD_AVG	66.50
48	COMMONAREA_AVG	69.87
49	ELEVATORS_AVG	53.30
50	ENTRANCES_AVG	50.35
51	FLOORSMAX_AVG	49.76
52	FLOORSMIN_AVG	67.85
53	LANDAREA_AVG	59.38
54	LIVINGAPARTMENTS_AVG	68.35
55	LIVINGAREA_AVG	50.19
56	NONLIVINGAPARTMENTS_AVG	69.43
57	NONLIVINGAREA_AVG	55.18
58	APARTMENTS_MODE	50.75
59	BASEMENTAREA_MODE	58.52
60	YEARS_BEGINEXPLUATATION_MODE	48.78
61	YEARS_BUILD_MODE	66.50
62	COMMONAREA_MODE	69.87
63	ELEVATORS_MODE	53.30

	Column_Names	Null_Value_Percentage
64	ENTRANCES_MODE	50.35
65	FLOORSMAX_MODE	49.76
66	FLOORSMIN_MODE	67.85
67	LANDAREA_MODE	59.38
68	LIVINGAPARTMENTS_MODE	68.35
69	LIVINGAREA_MODE	50.19
70	NONLIVINGAPARTMENTS_MODE	69.43
71	NONLIVINGAREA_MODE	55.18
72	APARTMENTS_MEDI	50.75
73	BASEMENTAREA_MEDI	58.52
74	YEARS_BEGINEXPLUATATION_MEDI	48.78
75	YEARS_BUILD_MEDI	66.50
76	COMMONAREA_MEDI	69.87
77	ELEVATORS_MEDI	53.30
78	ENTRANCES_MEDI	50.35
79	FLOORSMAX_MEDI	49.76
80	FLOORSMIN_MEDI	67.85
81	LANDAREA_MEDI	59.38
82	LIVINGAPARTMENTS_MEDI	68.35
83	LIVINGAREA_MEDI	50.19
84	NONLIVINGAPARTMENTS_MEDI	69.43
85	NONLIVINGAREA_MEDI	55.18

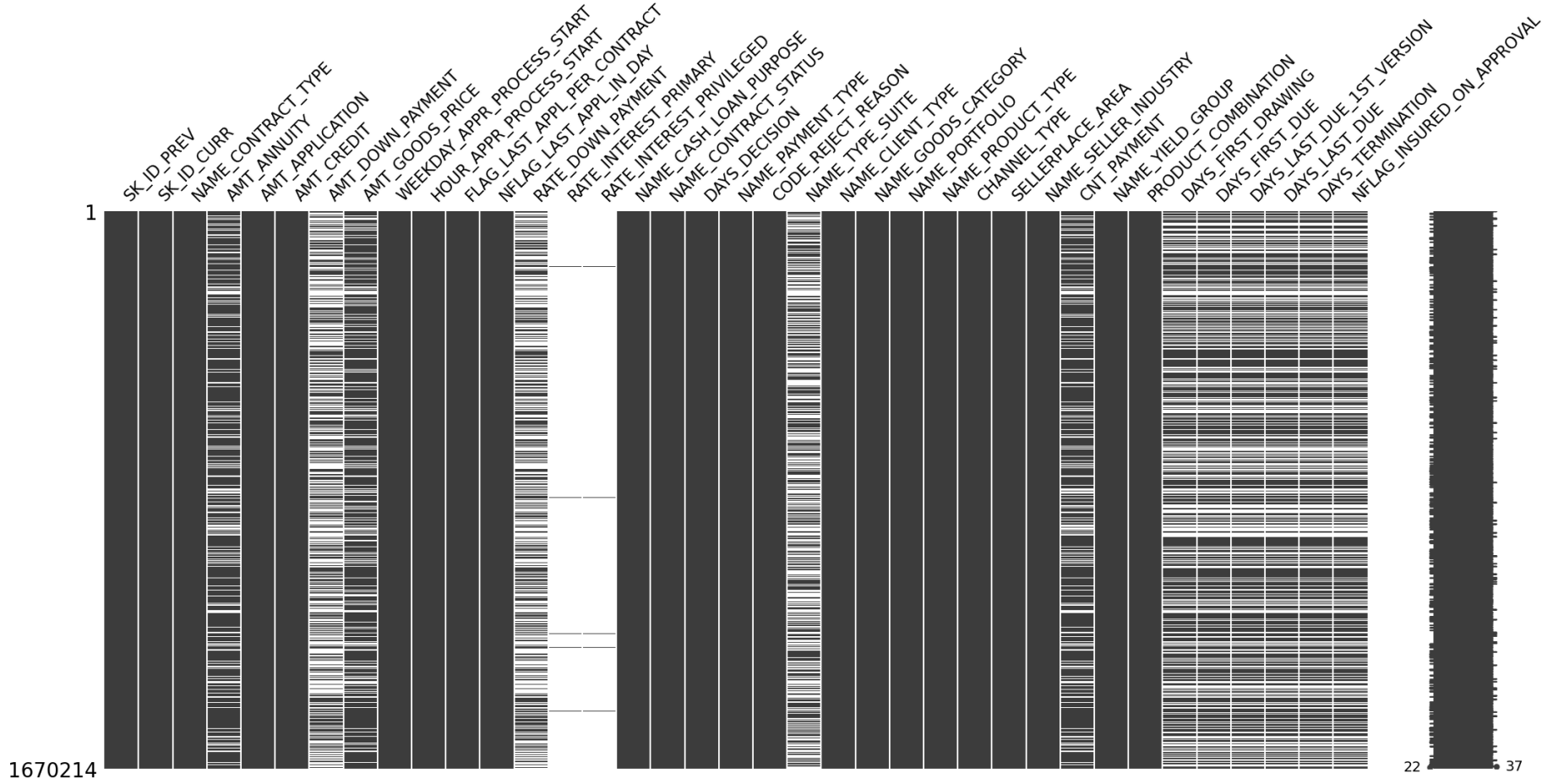
	Column_Names	Null_Value_Percentage
86	FONDKAPREMONT_MODE	68.39
87	HOUSETYPE_MODE	50.18
88	TOTALAREA_MODE	48.27
89	WALLSMATERIAL_MODE	50.84
90	EMERGENCYSTATE_MODE	47.40

In [29]:
```
len(NullCol_Gr8Than_40Percent_ApplicationDF)
```

Out[29]: 49

Plotting Missing value graph in Previous DF

In [31]:
```
mn.matrix(_PrevApplicationDF)
plt.show()
```



In [32]:
```
_PrevApplicationDF.isnull().sum()
```

Out[32]:
```
SK_ID_PREV                      0
SK_ID_CURR                      0
NAME_CONTRACT_TYPE              0
AMT_ANNUITY                372235
AMT_APPLICATION                 0
AMT_CREDIT                      1
AMT_DOWN_PAYMENT           895844
AMT_GOODS_PRICE            385515
WEEKDAY_APPR_PROCESS_START      0
HOUR_APPR_PROCESS_START         0
FLAG_LAST_APPL_PER_CONTRACT     0
NFLAG_LAST_APPL_IN_DAY          0
RATE_DOWN_PAYMENT          895844
RATE_INTEREST_PRIMARY     1664263
RATE_INTEREST_PRIVILEGED  1664263
NAME_CASH_LOAN_PURPOSE          0
NAME_CONTRACT_STATUS            0
DAYS_DECISION                   0
NAME_PAYMENT_TYPE               0
CODE_REJECT_REASON              0
NAME_TYPE_SUITE            820405
NAME_CLIENT_TYPE                0
NAME_GOODS_CATEGORY             0
NAME_PORTFOLIO                  0
NAME_PRODUCT_TYPE               0
CHANNEL_TYPE                    0
SELLERPLACE_AREA                0
NAME_SELLER_INDUSTRY            0
CNT_PAYMENT                372230
NAME_YIELD_GROUP                0
PRODUCT_COMBINATION           346
DAYS_FIRST_DRAWING         673065
DAYS_FIRST_DUE             673065
DAYS_LAST_DUE_1ST_VERSION  673065
DAYS_LAST_DUE              673065
DAYS_TERMINATION           673065
NFLAG_INSURED_ON_APPROVAL  673065
dtype: int64
```

In [33]:
```
np.round((_PrevApplicationDF.isnull().sum()/_PrevApplicationDF.shape[0]) * 100,2)
```

Out[33]:
```
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 [34]:
```
Null_Values_Percent_PreviousApplicationDF = pd.DataFrame(np.round((_PrevApplicationDF.isnull().sum()/_PrevApplicationDF.shape[
Null_Values_Percent_PreviousApplicationDF
```

Out[34]:

	index	0
0	SK_ID_PREV	0.00
1	SK_ID_CURR	0.00
2	NAME_CONTRACT_TYPE	0.00
3	AMT_ANNUITY	22.29
4	AMT_APPLICATION	0.00
5	AMT_CREDIT	0.00
6	AMT_DOWN_PAYMENT	53.64
7	AMT_GOODS_PRICE	23.08
8	WEEKDAY_APPR_PROCESS_START	0.00
9	HOUR_APPR_PROCESS_START	0.00
10	FLAG_LAST_APPL_PER_CONTRACT	0.00
11	NFLAG_LAST_APPL_IN_DAY	0.00
12	RATE_DOWN_PAYMENT	53.64
13	RATE_INTEREST_PRIMARY	99.64
14	RATE_INTEREST_PRIVILEGED	99.64
15	NAME_CASH_LOAN_PURPOSE	0.00
16	NAME_CONTRACT_STATUS	0.00
17	DAYS_DECISION	0.00
18	NAME_PAYMENT_TYPE	0.00
19	CODE_REJECT_REASON	0.00
20	NAME_TYPE_SUITE	49.12
21	NAME_CLIENT_TYPE	0.00

	index	0
22	NAME_GOODS_CATEGORY	0.00
23	NAME_PORTFOLIO	0.00
24	NAME_PRODUCT_TYPE	0.00
25	CHANNEL_TYPE	0.00
26	SELLERPLACE_AREA	0.00
27	NAME_SELLER_INDUSTRY	0.00
28	CNT_PAYMENT	22.29
29	NAME_YIELD_GROUP	0.00
30	PRODUCT_COMBINATION	0.02
31	DAYS_FIRST_DRAWING	40.30
32	DAYS_FIRST_DUE	40.30
33	DAYS_LAST_DUE_1ST_VERSION	40.30
34	DAYS_LAST_DUE	40.30
35	DAYS_TERMINATION	40.30
36	NFLAG_INSURED_ON_APPROVAL	40.30

In [35]:
```
Null_Values_Percent_PreviousApplicationDF.columns = ['Column_Names','Null_Value_Percentage']
Null_Values_Percent_PreviousApplicationDF
```

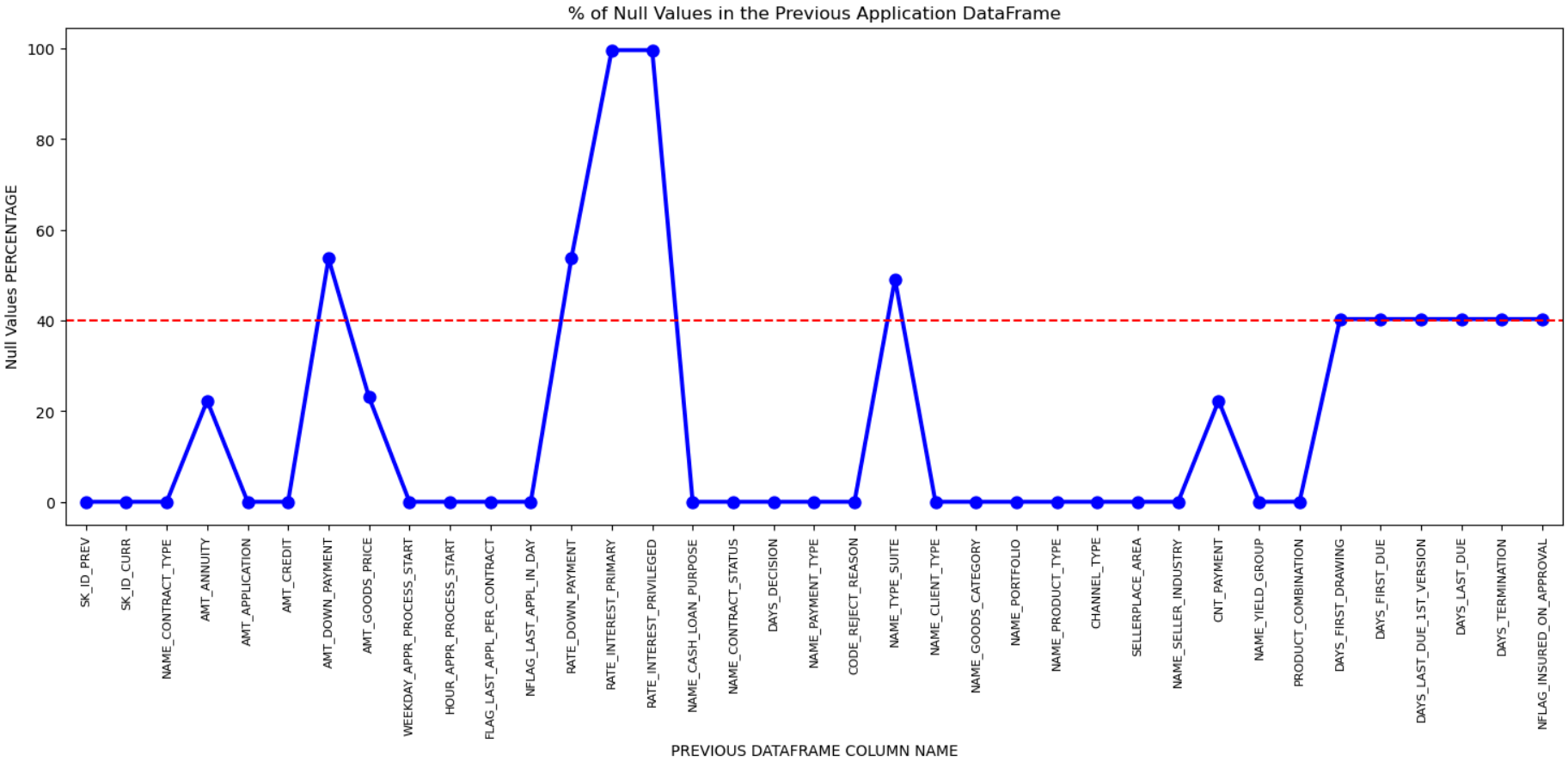
Out[35]:

	Column_Names	Null_Value_Percentage
0	SK_ID_PREV	0.00
1	SK_ID_CURR	0.00
2	NAME_CONTRACT_TYPE	0.00
3	AMT_ANNUITY	22.29
4	AMT_APPLICATION	0.00
5	AMT_CREDIT	0.00
6	AMT_DOWN_PAYMENT	53.64
7	AMT_GOODS_PRICE	23.08
8	WEEKDAY_APPR_PROCESS_START	0.00
9	HOUR_APPR_PROCESS_START	0.00
10	FLAG_LAST_APPL_PER_CONTRACT	0.00
11	NFLAG_LAST_APPL_IN_DAY	0.00
12	RATE_DOWN_PAYMENT	53.64
13	RATE_INTEREST_PRIMARY	99.64
14	RATE_INTEREST_PRIVILEGED	99.64
15	NAME_CASH_LOAN_PURPOSE	0.00
16	NAME_CONTRACT_STATUS	0.00
17	DAYS_DECISION	0.00
18	NAME_PAYMENT_TYPE	0.00
19	CODE_REJECT_REASON	0.00
20	NAME_TYPE_SUITE	49.12
21	NAME_CLIENT_TYPE	0.00

	Column_Names	Null_Value_Percentage
22	NAME_GOODS_CATEGORY	0.00
23	NAME_PORTFOLIO	0.00
24	NAME_PRODUCT_TYPE	0.00
25	CHANNEL_TYPE	0.00
26	SELLERPLACE_AREA	0.00
27	NAME_SELLER_INDUSTRY	0.00
28	CNT_PAYMENT	22.29
29	NAME_YIELD_GROUP	0.00
30	PRODUCT_COMBINATION	0.02
31	DAYS_FIRST_DRAWING	40.30
32	DAYS_FIRST_DUE	40.30
33	DAYS_LAST_DUE_1ST_VERSION	40.30
34	DAYS_LAST_DUE	40.30
35	DAYS_TERMINATION	40.30
36	NFLAG_INSURED_ON_APPROVAL	40.30

In [311…

```
ax = sns.pointplot(x="Column_Names",y="Null_Value_Percentage",data=Null_Values_Percent_PreviousApplicationDF,color='blue')
#fig = plt.figure(figsize=(18,6))
plt.gcf().set_size_inches(18,6)
plt.xticks(rotation =90,fontsize =8)
ax.axhline(40, ls='--',color='red')
plt.title('% of Null Values in the Previous Application DataFrame')
plt.ylabel("Null Values PERCENTAGE")
plt.xlabel("PREVIOUS DATAFRAME COLUMN NAME")
plt.show()
```



Insight (Previous DataFrame):

From the plot we can see:

1)The columns in which percentage of null values more than 40% are marked above the red line .

2) The columns which have less than 40 % null values below the red line.

Let's check the columns which has more than 40% missing values

In [313…
```
NullColumn_Values_Gr8Than_40Percent_PreviousApplicationDF = Null_Values_Percent_PreviousApplicationDF[Null_Values_Percent_Prev
NullColumn_Values_Gr8Than_40Percent_PreviousApplicationDF
```

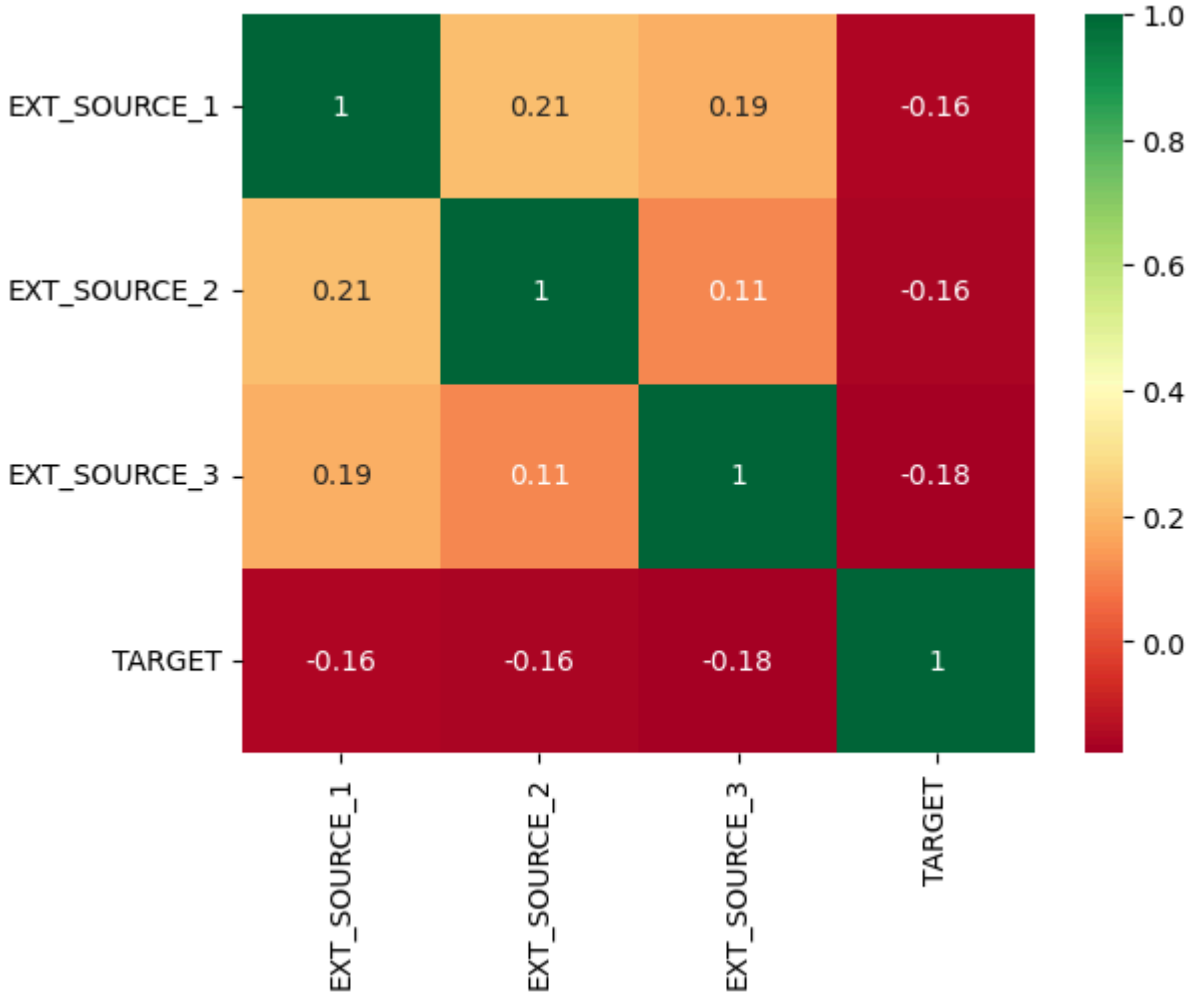
Out[313…

	Column_Names	Null_Value_Percentage
6	AMT_DOWN_PAYMENT	53.64
12	RATE_DOWN_PAYMENT	53.64
13	RATE_INTEREST_PRIMARY	99.64
14	RATE_INTEREST_PRIVILEGED	99.64
20	NAME_TYPE_SUITE	49.12
31	DAYS_FIRST_DRAWING	40.30
32	DAYS_FIRST_DUE	40.30
33	DAYS_LAST_DUE_1ST_VERSION	40.30
34	DAYS_LAST_DUE	40.30
35	DAYS_TERMINATION	40.30
36	NFLAG_INSURED_ON_APPROVAL	40.30

In [39]:
```
len(NullColumn_Values_Gr8Than_40Percent_PreviousApplicationDF)
```

Out[39]: 11

In [40]:
```
# Checking correlation of EXT_SOURCE_X columns vs TARGET column
Source = _ApplicationDF[["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")
plt.show()
```



In [41]:
```
Unwanted_ApplicationDF = NullCol_Gr8Than_40Percent_ApplicationDF["Column_Names"].tolist()+ ['EXT_SOURCE_2','EXT_SOURCE_3']
Unwanted_ApplicationDF
```

Out[41]:
```
['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']
```

In [42]:
```
len(Unwanted_ApplicationDF)
```

Out[42]: 51

Checking the Relevance of Flag_Doc columns against Target Column (Loan_Payment_Status)

99	application_data	FLAG_DOCUMENT_2	Did client provide document 2
100	application_data	FLAG_DOCUMENT_3	Did client provide document 3
101	application_data	FLAG_DOCUMENT_4	Did client provide document 4
102	application_data	FLAG_DOCUMENT_5	Did client provide document 5
103	application_data	FLAG_DOCUMENT_6	Did client provide document 6
104	application_data	FLAG_DOCUMENT_7	Did client provide document 7
105	application_data	FLAG_DOCUMENT_8	Did client provide document 8
106	application_data	FLAG_DOCUMENT_9	Did client provide document 9
107	application_data	FLAG_DOCUMENT_10	Did client provide document 10
108	application_data	FLAG_DOCUMENT_11	Did client provide document 11
109	application_data	FLAG_DOCUMENT_12	Did client provide document 12
110	application_data	FLAG_DOCUMENT_13	Did client provide document 13
111	application_data	FLAG_DOCUMENT_14	Did client provide document 14
112	application_data	FLAG_DOCUMENT_15	Did client provide document 15
113	application_data	FLAG_DOCUMENT_16	Did client provide document 16
114	application_data	FLAG_DOCUMENT_17	Did client provide document 17
115	application_data	FLAG_DOCUMENT_18	Did client provide document 18
116	application_data	FLAG_DOCUMENT_19	Did client provide document 19
117	application_data	FLAG_DOCUMENT_20	Did client provide document 20
118	application_data	FLAG_DOCUMENT_21	Did client provide document 21

In [86]:
```
_ApplicationDF.columns
```

Out[86]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY',
       ...
       'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20',
       'FLAG_DOCUMENT_21', 'AMT_REQ_CREDIT_BUREAU_HOUR',
       'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
       'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT',
       'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object', length=122)
```

In [88]:
```
_ApplicationDFFlagCols = _ApplicationDF.filter(regex='FLAG_DOC')
_ApplicationDFFlagCols.head(10)
```

Out[88]:

	FLAG_DOCUMENT_2	FLAG_DOCUMENT_3	FLAG_DOCUMENT_4	FLAG_DOCUMENT_5	FLAG_DOCUMENT_6	FLAG_DOCUMENT_7	FLAG_DC
0	0	1	0	0	0	0	
1	0	1	0	0	0	0	
2	0	0	0	0	0	0	
3	0	1	0	0	0	0	
4	0	0	0	0	0	0	
5	0	1	0	0	0	0	
6	0	0	0	0	0	0	
7	0	1	0	0	0	0	
8	0	1	0	0	0	0	
9	0	0	0	0	0	0	

In [46]:
```
_ApplicationDFFlagCols.columns
```

Out[46]:
```
Index(['FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3', 'FLAG_DOCUMENT_4',
       'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7',
       'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9', 'FLAG_DOCUMENT_10',
       'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12', 'FLAG_DOCUMENT_13',
       'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15', 'FLAG_DOCUMENT_16',
       'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19',
       'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21'],
      dtype='object')
```

In [47]:
```
Col_FlagDoc_ApplicationDF = [ 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3','FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6','FL
           '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']
len(Col_FlagDoc_ApplicationDF)
```

Out[47]:
```
20
```

In [48]:
```
Col_FlagDoc_ApplicationDF
```

Out[48]:
```
['FLAG_DOCUMENT_2',
 'FLAG_DOCUMENT_3',
 'FLAG_DOCUMENT_4',
 'FLAG_DOCUMENT_5',
 'FLAG_DOCUMENT_6',
 'FLAG_DOCUMENT_7',
 'FLAG_DOCUMENT_8',
 'FLAG_DOCUMENT_9',
 'FLAG_DOCUMENT_10',
 'FLAG_DOCUMENT_11',
 'FLAG_DOCUMENT_12',
 'FLAG_DOCUMENT_13',
 'FLAG_DOCUMENT_14',
 'FLAG_DOCUMENT_15',
 'FLAG_DOCUMENT_16',
 'FLAG_DOCUMENT_17',
 'FLAG_DOCUMENT_18',
 'FLAG_DOCUMENT_19',
 'FLAG_DOCUMENT_20',
 'FLAG_DOCUMENT_21']
```

In [49]:
```
len(Unwanted_ApplicationDF)
```

Out[49]:
```
51
```

In [50]:
```
_ApplicationDFFlagDoc = _ApplicationDF[Col_FlagDoc_ApplicationDF  + ['TARGET']]
_ApplicationDFFlagDoc
```

Out[50]:

	FLAG_DOCUMENT_2	FLAG_DOCUMENT_3	FLAG_DOCUMENT_4	FLAG_DOCUMENT_5	FLAG_DOCUMENT_6	FLAG_DOCUMENT_7	FL
0	0	1	0	0	0	0	
1	0	1	0	0	0	0	
2	0	0	0	0	0	0	
3	0	1	0	0	0	0	
4	0	0	0	0	0	0	
...	...	...	...	...	...	...	
307506	0	0	0	0	0	0	
307507	0	1	0	0	0	0	
307508	0	1	0	0	0	0	
307509	0	1	0	0	0	0	
307510	0	1	0	0	0	0	

307511 rows × 21 columns

In [51]:
```
_ApplicationDFFlagDoc.tail(10)
```

Out[51]:

	FLAG_DOCUMENT_2	FLAG_DOCUMENT_3	FLAG_DOCUMENT_4	FLAG_DOCUMENT_5	FLAG_DOCUMENT_6	FLAG_DOCUMENT_7	FL
307501	0	1	0	0	0	0	
307502	0	1	0	0	0	0	
307503	0	1	0	0	0	0	
307504	0	1	0	0	0	0	
307505	0	0	0	0	1	0	
307506	0	0	0	0	0	0	
307507	0	1	0	0	0	0	
307508	0	1	0	0	0	0	
307509	0	1	0	0	0	0	
307510	0	1	0	0	0	0	

10 rows × 21 columns

By Seeing the description we can conclude that for Column Target (1 - Means who have defaulted, 0 Means who have not)

	Table	Row	Description
2	application_data	TARGET	Target variable (1 - client with payment difficulties: he/she had late payment more than X days on at least one of the first Y installments of the loan in our sample, 0 - all other cases)

In [53]:
```
_ApplicationDFFlagDoc['TARGET'] = _ApplicationDFFlagDoc['TARGET'].replace({0:'Repayer',1:'Deafulter'})
_ApplicationDFFlagDoc['TARGET']
```

Out[53]:
```
0         Deafulter
1           Repayer
2           Repayer
3           Repayer
4           Repayer
            ...
307506      Repayer
307507      Repayer
307508      Repayer
307509    Deafulter
307510      Repayer
Name: TARGET, Length: 307511, dtype: object
```

In [54]:
```
_ApplicationDFFlagDoc.head(5)
```

Out[54]:

	FLAG_DOCUMENT_2	FLAG_DOCUMENT_3	FLAG_DOCUMENT_4	FLAG_DOCUMENT_5	FLAG_DOCUMENT_6	FLAG_DOCUMENT_7	FLAG_DC
0	0	1	0	0	0	0	
1	0	1	0	0	0	0	
2	0	0	0	0	0	0	
3	0	1	0	0	0	0	
4	0	0	0	0	0	0	

5 rows × 21 columns

In [55]:
```
_ApplicationDFFlagDoc['TARGET'].value_counts()
```

Out[55]:
```
TARGET
Repayer      282686
Deafulter     24825
Name: count, dtype: int64
```

In [56]:
```
_ApplicationDFFlagDoc
```

Out[56]:

	FLAG_DOCUMENT_2	FLAG_DOCUMENT_3	FLAG_DOCUMENT_4	FLAG_DOCUMENT_5	FLAG_DOCUMENT_6	FLAG_DOCUMENT_7	FL
0	0	1	0	0	0	0	
1	0	1	0	0	0	0	
2	0	0	0	0	0	0	
3	0	1	0	0	0	0	
4	0	0	0	0	0	0	
...	...	...	...	...	...	...	
307506	0	0	0	0	0	0	
307507	0	1	0	0	0	0	
307508	0	1	0	0	0	0	
307509	0	1	0	0	0	0	
307510	0	1	0	0	0	0	

307511 rows × 21 columns

In [57]:
```
import itertools
```

In [58]:
```
_length = len(Col_FlagDoc_ApplicationDF)
_length
```

Out[58]: 20

In [59]:
```
fig = plt.figure(figsize=(21,24))
for i,j in itertools.zip_longest(Col_FlagDoc_ApplicationDF,range(_length)):
    plt.subplot(5,4,j+1)
    ax = sns.countplot(_ApplicationDFFlagDoc[i],hue=_ApplicationDFFlagDoc['TARGET'],palette=["r","g"])
    plt.yticks(fontsize=8)
    plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[59], line 4
      2 for i,j in itertools.zip_longest(Col_FlagDoc_ApplicationDF,range(_length)):
      3     plt.subplot(5,4,j+1)
----> 4     ax = sns.countplot(_ApplicationDFFlagDoc[i],hue=_ApplicationDFFlagDoc['TARGET'],palette=["r","g"])
      5     plt.yticks(fontsize=8)
      6     plt.show()

File ~\anaconda3\Lib\site-packages\seaborn\categorical.py:2631, in countplot(data, x, y, hue, order, hue_order, orient, color,
palette, saturation, fill, hue_norm, stat, width, dodge, gap, log_scale, native_scale, formatter, legend, ax, **kwargs)
   2628 elif x is not None and y is not None:
   2629     raise TypeError("Cannot pass values for both `x` and `y`.")
-> 2631 p = _CategoricalAggPlotter(
   2632     data=data,
   2633     variables=dict(x=x, y=y, hue=hue),
   2634     order=order,
   2635     orient=orient,
   2636     color=color,
   2637     legend=legend,
   2638 )
   2640 if ax is None:
   2641     ax = plt.gca()

File ~\anaconda3\Lib\site-packages\seaborn\categorical.py:67, in _CategoricalPlotter.__init__(self, data, variables, order, ori
ent, require_numeric, color, legend)
     56 def __init__(
     57     self,
     58     data=None,
   (...)
     64     legend="auto",
     65 ):
---> 67     super().__init__(data=data, variables=variables)
     69     # This method takes care of some bookkeeping that is necessary because the
     70     # original categorical plots (prior to the 2021 refactor) had some rules that
     71     # don't fit exactly into VectorPlotter logic. It may be wise to have a second
   (...)
     76     # default VectorPlotter rules. If we do decide to make orient part of the
     77     # _base variable assignment, we'll want to figure out how to express that.
     78     if self.input_format == "wide" and orient in ["h", "y"]:
```

```
File ~\anaconda3\Lib\site-packages\seaborn\_base.py:634, in VectorPlotter.__init__(self, data, variables)
    629 # var_ordered is relevant only for categorical axis variables, and may
    630 # be better handled by an internal axis information object that tracks
    631 # such information and is set up by the scale_* methods. The analogous
    632 # information for numeric axes would be information about log scales.
    633 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 634 self.assign_variables(data, variables)
    636 # TODO Lots of tests assume that these are called to initialize the
    637 # mappings to default values on class initialization. I'd prefer to
    638 # move away from that and only have a mapping when explicitly called.
    639 for var in ["hue", "size", "style"]:

File ~\anaconda3\Lib\site-packages\seaborn\_base.py:673, in VectorPlotter.assign_variables(self, data, variables)
    671 if x is None and y is None:
    672     self.input_format = "wide"
--> 673     frame, names = self._assign_variables_wideform(data, **variables)
    674 else:
    675     # When dealing with long-form input, use the newer PlotData
    676     # object (internal but introduced for the objects interface)
    677     # to centralize / standardize data consumption logic.
    678     self.input_format = "long"

File ~\anaconda3\Lib\site-packages\seaborn\_base.py:723, in VectorPlotter._assign_variables_wideform(self, data, **kwargs)
    721     err = f"The following variable{s} cannot be assigned with wide-form data: "
    722     err += ", ".join(f"`{v}`" for v in assigned)
--> 723     raise ValueError(err)
    725 # Determine if the data object actually has any data in it
    726 empty = data is None or not len(data)

ValueError: The following variable cannot be assigned with wide-form data: `hue`
```

In [90]:
```
len(Unwanted_ApplicationDF)
```

Out[90]: 51

In [92]:
```
len(Col_FlagDoc_ApplicationDF)
```

Out[92]: 20

In [94]:
```
Col_FlagDoc_ApplicationDF.remove('FLAG_DOCUMENT_3')
Unwanted_ApplicationDF = Unwanted_ApplicationDF + Col_FlagDoc_ApplicationDF
```

In [96]:
```
len(Unwanted_ApplicationDF)
```

Out[96]: 70

Analysing the APPLICATIONDF Flag Contact Parameters

25	application_data	FLAG_MOBIL	Did client provide mobile phone (1=YES, 0=NO)
26	application_data	FLAG_EMP_PHONE	Did client provide work phone (1=YES, 0=NO)
27	application_data	FLAG_WORK_PHONE	Did client provide home phone (1=YES, 0=NO)
28	application_data	FLAG_CONT_MOBILE	Was mobile phone reachable (1=YES, 0=NO)
29	application_data	FLAG_PHONE	Did client provide home phone (1=YES, 0=NO)
30	application_data	FLAG_EMAIL	Did client provide email (1=YES, 0=NO)

In [98]:
```
_ApplicationDFContactFlagCols = _ApplicationDF.filter(regex='FLAG_')
_ApplicationDFContactFlagCols
```

Out[98]:

	FLAG_OWN_CAR	FLAG_OWN_REALTY	FLAG_MOBIL	FLAG_EMP_PHONE	FLAG_WORK_PHONE	FLAG_CONT_MOBILE	FLAG_PHONE
0	N	Y	1	1	0	1	1
1	N	N	1	1	0	1	1
2	Y	Y	1	1	1	1	1
3	N	Y	1	1	0	1	0
4	N	Y	1	1	0	1	0
...	...	...	...	...	...	...	...
307506	N	N	1	1	0	1	0
307507	N	Y	1	0	0	1	1
307508	N	Y	1	1	0	1	0
307509	N	Y	1	1	0	1	0
307510	N	N	1	1	1	1	1

307511 rows × 28 columns

In [100…

```
_ApplicationDFContactFlagCols.columns
```

Out[100…

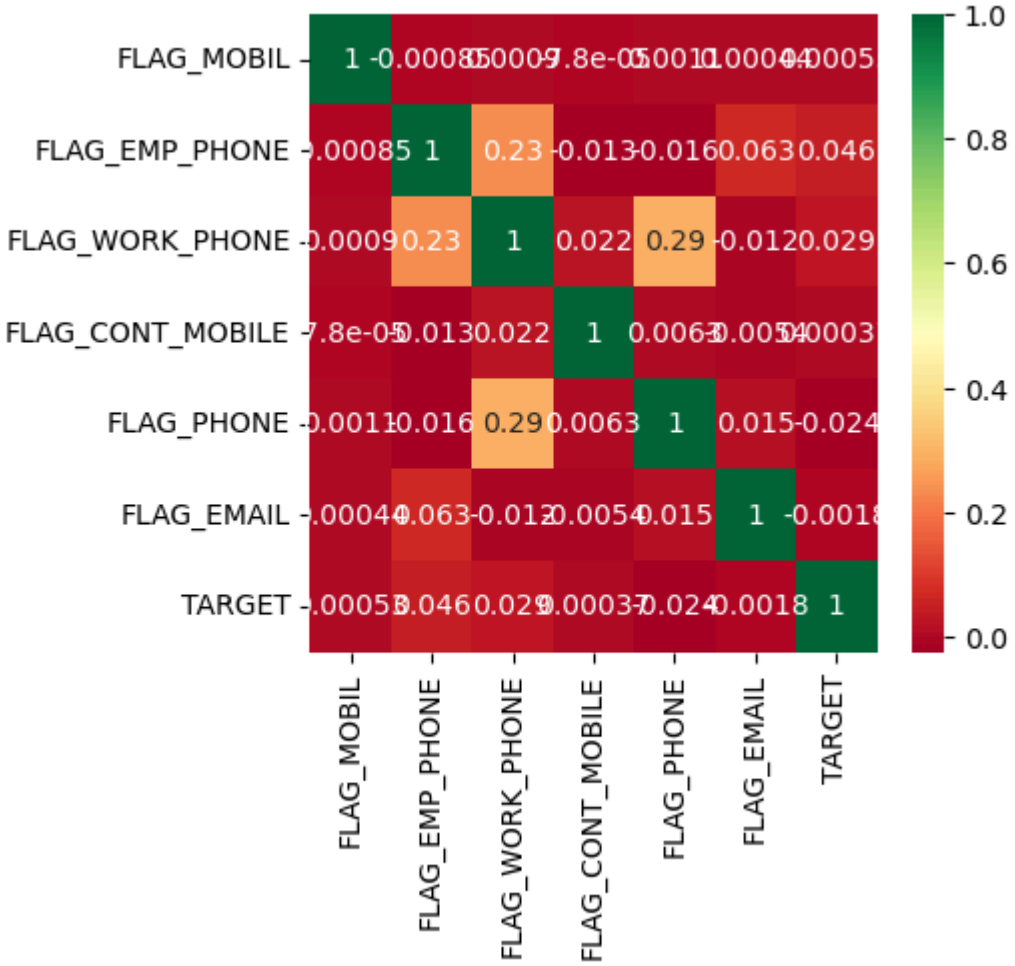
```
Index(['FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'FLAG_MOBIL', 'FLAG_EMP_PHONE',
       'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL',
       'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3', 'FLAG_DOCUMENT_4',
       'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7',
       'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9', 'FLAG_DOCUMENT_10',
       'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12', 'FLAG_DOCUMENT_13',
       'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15', 'FLAG_DOCUMENT_16',
       'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19',
       'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21'],
      dtype='object')
```

In [102…
```
_ApplicationDFContact_Cols = ['FLAG_MOBIL', 'FLAG_EMP_PHONE','FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL'
_ApplicationDFContact_corr = _ApplicationDF[_ApplicationDFContact_Cols].corr()
_ApplicationDFContact_corr
```

Out[102…

	FLAG_MOBIL	FLAG_EMP_PHONE	FLAG_WORK_PHONE	FLAG_CONT_MOBILE	FLAG_PHONE	FLAG_EMAIL	TARGET
FLAG_MOBIL	1.000000	-0.000845	0.000900	-0.000078	0.001128	0.000442	0.000534
FLAG_EMP_PHONE	-0.000845	1.000000	0.233801	-0.012819	-0.016131	0.062542	0.045982
FLAG_WORK_PHONE	0.000900	0.233801	1.000000	0.021580	0.293105	-0.011520	0.028524
FLAG_CONT_MOBILE	-0.000078	-0.012819	0.021580	1.000000	0.006257	-0.005356	0.000370
FLAG_PHONE	0.001128	-0.016131	0.293105	0.006257	1.000000	0.014657	-0.023806
FLAG_EMAIL	0.000442	0.062542	-0.011520	-0.005356	0.014657	1.000000	-0.001758
TARGET	0.000534	0.045982	0.028524	0.000370	-0.023806	-0.001758	1.000000

In [104…
```
ax = sns.heatmap(_ApplicationDFContact_corr,
            xticklabels=_ApplicationDFContact_corr.columns,
            yticklabels=_ApplicationDFContact_corr.columns,
            annot = True,
            cmap ="RdYlGn")
plt.show()
```



Insight:

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

In [106…

```
_ApplicationDFContact_Cols.remove('TARGET')
_ApplicationDFContact_Cols
```

Out[106…
```
['FLAG_MOBIL',
 'FLAG_EMP_PHONE',
 'FLAG_WORK_PHONE',
 'FLAG_CONT_MOBILE',
 'FLAG_PHONE',
 'FLAG_EMAIL']
```

Conclusion: We Can Include the FlagContacts columns in the Unwanted Columns list and Delete the columns from the Actual ApplicationDF

In [108…
```
_ApplicationDFContact_Cols
```

Out[108…
```
['FLAG_MOBIL',
 'FLAG_EMP_PHONE',
 'FLAG_WORK_PHONE',
 'FLAG_CONT_MOBILE',
 'FLAG_PHONE',
 'FLAG_EMAIL']
```

In [110…
```
Unwanted_ApplicationDF = Unwanted_ApplicationDF + _ApplicationDFContact_Cols
Unwanted_ApplicationDF
```

Out[110…

```
['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',
 'FLAG_MOBIL',
 'FLAG_EMP_PHONE',
 'FLAG_WORK_PHONE',
 'FLAG_CONT_MOBILE',
 'FLAG_PHONE',
 'FLAG_EMAIL']
```

In [ ]:
```
len(Unwanted_ApplicationDF)
```

Conclusion: We delete the 76 columns from the Actual ApplicationDF

In [112…
```
print('Before deleting there were',_ApplicationDF.shape[0],'rows and', _ApplicationDF.shape[1],'columns in Application DF.')
```

```
Before deleting there were 307511 rows and 122 columns in Application DF.
```

In [114…
```
_ApplicationDF.drop(labels = Unwanted_ApplicationDF,inplace=True,axis=1)
```

In [116…
```
print('After deleting there were',_ApplicationDF.shape[0],'rows and', _ApplicationDF.shape[1],'columns in Application DF.')
```

```
After deleting there were 307511 rows and 46 columns in Application DF.
```

In [118…
```
_ApplicationDF.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
```

Insight:

After deleting unnecessary columns, there are 46 columns remaining in applicationDF.

Analysing the Unwanted Columns in PreviousApplicationDF :

In [122…

```
NullColumn_Values_Gr8Than_40Percent_PreviousApplicationDF
```

Out[122...

	Column_Names	Null_Value_Percentage
6	AMT_DOWN_PAYMENT	53.64
12	RATE_DOWN_PAYMENT	53.64
13	RATE_INTEREST_PRIMARY	99.64
14	RATE_INTEREST_PRIVILEGED	99.64
20	NAME_TYPE_SUITE	49.12
31	DAYS_FIRST_DRAWING	40.30
32	DAYS_FIRST_DUE	40.30
33	DAYS_LAST_DUE_1ST_VERSION	40.30
34	DAYS_LAST_DUE	40.30
35	DAYS_TERMINATION	40.30
36	NFLAG_INSURED_ON_APPROVAL	40.30

In [124...
```
len(NullColumn_Values_Gr8Than_40Percent_PreviousApplicationDF)
```

Out[124... 11

In [126...
```
Unwanted_PreviousApplicationDF = NullColumn_Values_Gr8Than_40Percent_PreviousApplicationDF["Column_Names"].tolist()
Unwanted_PreviousApplicationDF
```

Out[126…
```
['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 [128…
```
Unnecessary_PreviousApplicationDF = ['WEEKDAY_APPR_PROCESS_START','HOUR_APPR_PROCESS_START','FLAG_LAST_APPL_PER_CONTRACT','NFL
```

In [130…
```
Unwanted_PreviousApplicationDF = Unwanted_PreviousApplicationDF + Unnecessary_PreviousApplicationDF
Unwanted_PreviousApplicationDF
```

Out[130…
```
['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',
 'WEEKDAY_APPR_PROCESS_START',
 'HOUR_APPR_PROCESS_START',
 'FLAG_LAST_APPL_PER_CONTRACT',
 'NFLAG_LAST_APPL_IN_DAY']
```

In [132…
```
len(Unwanted_PreviousApplicationDF)
```

Out[132…
```
15
```

Conclusion: We delete the 15 columns from the Previous ApplicationDF

In [134…
```
print('Before deleting there were',_PrevApplicationDF.shape[0],'rows and', _PrevApplicationDF.shape[1],'columns in Previous Ap
```

```
Before deleting there were 1670214 rows and 37 columns in Previous Application DF.
```

In [136…
```
_PrevApplicationDF.drop(labels = Unwanted_PreviousApplicationDF,inplace=True,axis=1)
```

In [138…
```
print('After deleting there were',_PrevApplicationDF.shape[0],'rows and', _PrevApplicationDF.shape[1],'columns in Previous App
```

```
After deleting there were 1670214 rows and 22 columns in Previous Application DF.
```

In [140…
```
_PrevApplicationDF.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
```

Insight:

After deleting unnecessary columns, there are 22 columns remaining in previous applicationDF.

In [142…
```
date_cols = ['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH']
_ApplicationDF[date_cols].head(10)
```

Out[142…

	DAYS_BIRTH	DAYS_EMPLOYED	DAYS_REGISTRATION	DAYS_ID_PUBLISH
0	-9461	-637	-3648.0	-2120
1	-16765	-1188	-1186.0	-291
2	-19046	-225	-4260.0	-2531
3	-19005	-3039	-9833.0	-2437
4	-19932	-3038	-4311.0	-3458
5	-16941	-1588	-4970.0	-477
6	-13778	-3130	-1213.0	-619
7	-18850	-449	-4597.0	-2379
8	-20099	365243	-7427.0	-3514
9	-14469	-2019	-14437.0	-3992

Converting Negative days to positive days

In [322…
```
for col in date_cols:
    _ApplicationDF[col] = abs(_ApplicationDF[col])
_ApplicationDF[date_cols].head(10)
```

Out[322...

	DAYS_BIRTH	DAYS_EMPLOYED	DAYS_REGISTRATION	DAYS_ID_PUBLISH
0	9461	637	3648.0	2120
1	16765	1188	1186.0	291
2	19046	225	4260.0	2531
3	19005	3039	9833.0	2437
4	19932	3038	4311.0	3458
5	16941	1588	4970.0	477
6	13778	3130	1213.0	619
7	18850	449	4597.0	2379
8	20099	365243	7427.0	3514
9	14469	2019	14437.0	3992

Creating bins for

1. AMT_INCOME_TOTAL
2. AGE_GROUP
3. AMT_CREDIT

In [332...

```
_ApplicationDF['AMT_INCOME_TOTAL']=_ApplicationDF['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','700k-800k','800k-900k','900k-1M', '
_ApplicationDF['AMT_INCOME_RANGE']=pd.cut(_ApplicationDF['AMT_INCOME_TOTAL'],bins,labels=slot)
```

In [334...

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

Out[334…
```
AMT_INCOME_RANGE
0-100K       100.0
100K-200K      0.0
200k-300k      0.0
300k-400k      0.0
400k-500k      0.0
500k-600k      0.0
600k-700k      0.0
700k-800k      0.0
800k-900k      0.0
900k-1M        0.0
1M Above       0.0
Name: proportion, dtype: float64
```

Insight:

More than 50% loan applicants have income amount in the range of 100K-200K. Almost 92% loan applicants have income less than 300K

In [150…
```
_ApplicationDF['AMT_CREDIT']=_ApplicationDF['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-700k','700k-800k','800k-900k','900k-1M',
_ApplicationDF['AMT_CREDIT_RANGE']=pd.cut(_ApplicationDF['AMT_CREDIT'],bins=bins,labels=slots)
```

In [152…
```
_ApplicationDF['AMT_CREDIT_RANGE'].head(10)
```

Out[152…
```
0    400k-500k
1     1M Above
2    100K-200K
3    300k-400k
4    500k-600k
5    400k-500k
6     1M Above
7     1M Above
8     1M Above
9    400k-500k
Name: AMT_CREDIT_RANGE, dtype: category
Categories (11, object): ['0-100K' < '100K-200K' < '200k-300k' < '300k-400k' ... '700k-800k' < '800k-900k' < '900k-1M' < '1M
Above']
```

In [336…
```
_ApplicationDF['AMT_CREDIT_RANGE'].value_counts(normalize=True)*100
```

Out[336…
```
AMT_CREDIT_RANGE
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: proportion, dtype: float64
```

Insight:

close to 16.3% loan applicants have taken loan which amounts to more than 1M.

In [338…
```
_ApplicationDF['DAYS_BIRTH'].head(10)
```

Out[338…
```
0     9461
1    16765
2    19046
3    19005
4    19932
5    16941
6    13778
7    18850
8    20099
9    14469
Name: DAYS_BIRTH, dtype: int64
```

In [340…
```
# Creating bins for Age
_ApplicationDF['AGE'] = _ApplicationDF['DAYS_BIRTH'] // 365
bins = [0,20,30,40,50,100]
slots = ['0-20','20-30','30-40','40-50','50 above']
```

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

In [342…
```
_ApplicationDF['AGE_GROUP'].value_counts(normalize=True)*100
```

Out[342…
```
AGE_GROUP
50 above    31.604398
30-40       27.028952
40-50       24.194582
20-30       17.171743
0-20         0.000325
Name: proportion, dtype: float64
```

Insight:

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

In [350…
```
#Creating bins for Employement Time
_ApplicationDF['YEARS_EMPLOYED'] = _ApplicationDF['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']

_ApplicationDF['EMPLOYMENT_YEAR']=pd.cut(_ApplicationDF['YEARS_EMPLOYED'],bins=bins,labels=slots)
```

In [352…
```
_ApplicationDF['EMPLOYMENT_YEAR'].value_counts(normalize=True)*100
```

Out[352…
```
EMPLOYMENT_YEAR
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: proportion, dtype: float64
```

Insight:

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 [164…

```
_ApplicationDF.nunique().sort_values()
```

Out[164…

AGE_GROUP	0
AMT_INCOME_RANGE	1
TARGET	2
NAME_CONTRACT_TYPE	2
FLAG_OWN_CAR	2
FLAG_OWN_REALTY	2
FLAG_DOCUMENT_3	2
LIVE_CITY_NOT_WORK_CITY	2
REG_CITY_NOT_WORK_CITY	2
REG_CITY_NOT_LIVE_CITY	2
LIVE_REGION_NOT_WORK_REGION	2
REG_REGION_NOT_WORK_REGION	2
REG_REGION_NOT_LIVE_REGION	2
REGION_RATING_CLIENT	3
REGION_RATING_CLIENT_W_CITY	3
CODE_GENDER	3
AMT_REQ_CREDIT_BUREAU_HOUR	5
NAME_EDUCATION_TYPE	5
NAME_FAMILY_STATUS	6
NAME_HOUSING_TYPE	6
WEEKDAY_APPR_PROCESS_START	7
NAME_TYPE_SUITE	7
NAME_INCOME_TYPE	8
AMT_REQ_CREDIT_BUREAU_DAY	9
DEF_60_CNT_SOCIAL_CIRCLE	9
AMT_REQ_CREDIT_BUREAU_WEEK	9
DEF_30_CNT_SOCIAL_CIRCLE	10
AMT_REQ_CREDIT_BUREAU_QRT	11
AMT_CREDIT_RANGE	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_30_CNT_SOCIAL_CIRCLE	33
OBS_60_CNT_SOCIAL_CIRCLE	33
AGE	50
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 [166…

```
_ApplicationDF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 50 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             307511 non-null  category
 47  AMT_CREDIT_RANGE             307511 non-null  category
 48  AGE                          307511 non-null  int64
 49  AGE_GROUP                    0 non-null       category
dtypes: category(3), float64(18), int64(17), object(12)
memory usage: 111.1+ MB
```

In [168…

```
categorical_columns = ['NAME_CONTRACT_TYPE','CODE_GENDER','NAME_TYPE_SUITE','NAME_INCOME_TYPE','NAME_EDUCATION_TYPE',
                       'NAME_FAMILY_STATUS','NAME_HOUSING_TYPE','OCCUPATION_TYPE','WEEKDAY_APPR_PROCESS_START',
                       'ORGANIZATION_TYPE','FLAG_OWN_CAR','FLAG_OWN_REALTY','LIVE_CITY_NOT_WORK_CITY',
                       'REG_CITY_NOT_LIVE_CITY','REG_CITY_NOT_WORK_CITY','REG_REGION_NOT_WORK_REGION',
                       'LIVE_REGION_NOT_WORK_REGION','REGION_RATING_CLIENT','WEEKDAY_APPR_PROCESS_START',
                       'REGION_RATING_CLIENT_W_CITY'
                      ]
for col in categorical_columns:
    _ApplicationDF[col] =pd.Categorical(_ApplicationDF[col])
```

In [170…

```
_ApplicationDF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 50 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             307511 non-null  category
 47  AMT_CREDIT_RANGE             307511 non-null  category
 48  AGE                          307511 non-null  int64
 49  AGE_GROUP                    0 non-null       category
dtypes: category(22), float64(18), int64(10)
memory usage: 72.2 MB
```

In [172…

```
_PrevApplicationDF.nunique().sort_values()
```

Out[172…

```
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 [174…

```
# inspecting the column types if the above conversion is reflected
_PrevApplicationDF.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 [176…

```
_PrevApplicationDF['DAYS_DECISION'].head(10)
```

Out[176…
```
0    -73
1   -164
2   -301
3   -512
4   -781
5   -684
6    -14
7    -21
8   -386
9    -57
Name: DAYS_DECISION, dtype: int64
```

In [178…
```
_PrevApplicationDF['DAYS_DECISION'] = abs(_PrevApplicationDF['DAYS_DECISION'])
_PrevApplicationDF['DAYS_DECISION'].head(10)
```

Out[178…
```
0     73
1    164
2    301
3    512
4    781
5    684
6     14
7     21
8    386
9     57
Name: DAYS_DECISION, dtype: int64
```

In [354…
```
#age group calculation e.g. 388 will be grouped as 300-_PrevApplicationDF
_PrevApplicationDF['DAYS_DECISION_GROUP'] = (_PrevApplicationDF['DAYS_DECISION']-(_PrevApplicationDF['DAYS_DECISION'] % 400)).
       + (400 - (_PrevApplicationDF['DAYS_DECISION'] % 400))).astype(str)
```

In [356…
```
_PrevApplicationDF['DAYS_DECISION_GROUP'].value_counts(normalize=True)*100
```

Out[356…
```
DAYS_DECISION_GROUP
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: proportion, dtype: float64
```

In [358…
```
_PrevApplicationDF.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             1670214 non-null  float64
 4   AMT_APPLICATION         1670214 non-null  float64
 5   AMT_CREDIT              1670213 non-null  float64
 6   AMT_GOODS_PRICE         1670214 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             1670214 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  object
dtypes: category(13), float64(5), int64(4), object(1)
memory usage: 148.1+ MB
```

In [360…
```
PrevApplicationDF_CatColumns = _PrevApplicationDF.select_dtypes(include=['object'])
PrevApplicationDF_CatColumns.columns
```

Out[360…
```
Index(['DAYS_DECISION_GROUP'], dtype='object')
```

In [186…
```
for col in PrevApplicationDF_CatColumns:
    _PrevApplicationDF[col] =pd.Categorical(_PrevApplicationDF[col])
```

In [188…
```
_PrevApplicationDF.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  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
dtypes: category(13), float64(5), int64(4)
memory usage: 135.4 MB
```

Imputing Null Values in applicationDF

In [190…

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

Out[190…

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                 0.00
AMT_CREDIT_RANGE                 0.00
AGE                              0.00
AGE_GROUP                      100.00
dtype: float64
```

In [362…
```
_ApplicationDF['NAME_TYPE_SUITE'].describe()
```

Out[362…
```
count            307511
unique                7
top       Unaccompanied
freq             249818
Name: NAME_TYPE_SUITE, dtype: object
```

In [364…
```
_ApplicationDF['NAME_TYPE_SUITE'].fillna(_ApplicationDF['NAME_TYPE_SUITE'].mode()[0],inplace=True)
```

In [366…
```
_ApplicationDF['NAME_TYPE_SUITE'].isnull().sum()
```

Out[366…
```
0
```

In [374…
```
_ApplicationDF['OCCUPATION_TYPE'].isnull().sum()
```

Out[374…
```
0
```

In [376…
```
_ApplicationDF['OCCUPATION_TYPE'] = _ApplicationDF['OCCUPATION_TYPE'].cat.add_categories('Unknown')
_ApplicationDF['OCCUPATION_TYPE'].fillna('Unknown', inplace =True)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[376], line 1
----> 1 _ApplicationDF['OCCUPATION_TYPE'] = _ApplicationDF['OCCUPATION_TYPE'].cat.add_categories('Unknown')
      2 _ApplicationDF['OCCUPATION_TYPE'].fillna('Unknown', inplace =True)

File ~\anaconda3\Lib\site-packages\pandas\core\accessor.py:112, in PandasDelegate._add_delegate_accessors.<locals>._create_delegator_method.<locals>.f(self, *args, **kwargs)
    111 def f(self, *args, **kwargs):
--> 112     return self._delegate_method(name, *args, **kwargs)

File ~\anaconda3\Lib\site-packages\pandas\core\arrays\categorical.py:2941, in CategoricalAccessor._delegate_method(self, name, *args, **kwargs)
   2938 from pandas import Series
   2940 method = getattr(self._parent, name)
-> 2941 res = method(*args, **kwargs)
   2942 if res is not None:
   2943     return Series(res, index=self._index, name=self._name)

File ~\anaconda3\Lib\site-packages\pandas\core\arrays\categorical.py:1330, in Categorical.add_categories(self, new_categories)
   1328 already_included = set(new_categories) & set(self.dtype.categories)
   1329 if len(already_included) != 0:
-> 1330     raise ValueError(
   1331         f"new categories must not include old categories: {already_included}"
   1332     )
   1334 if hasattr(new_categories, "dtype"):
   1335     from pandas import Series

ValueError: new categories must not include old categories: {'Unknown'}
```

In [378…
```
_ApplicationDF['OCCUPATION_TYPE'].isnull().sum()
```

Out[378…
```
0
```

In [380…
```
_ApplicationDF[['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[380...

	AMT_REQ_CREDIT_BUREAU_HOUR	AMT_REQ_CREDIT_BUREAU_DAY	AMT_REQ_CREDIT_BUREAU_WEEK	AMT_REQ_CREDIT_BUREAU_MO
count	307511.000000	307511.000000	307511.000000	307511.00000
mean	0.005538	0.006055	0.029723	0.23129
std	0.078014	0.103037	0.190728	0.85681
min	0.000000	0.000000	0.000000	0.00000
25%	0.000000	0.000000	0.000000	0.00000
50%	0.000000	0.000000	0.000000	0.00000
75%	0.000000	0.000000	0.000000	0.00000
max	4.000000	9.000000	8.000000	27.00000

In [204...

```
Amount_Cols = ['AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY','AMT_REQ_CREDIT_BUREAU_WEEK','AMT_REQ_CREDIT_BUREAU_M
        'AMT_REQ_CREDIT_BUREAU_QRT','AMT_REQ_CREDIT_BUREAU_YEAR']
```

In [384...

```
_ApplicationDF[Amount_Cols].isnull().sum()
```

Out[384...

```
AMT_REQ_CREDIT_BUREAU_HOUR    0
AMT_REQ_CREDIT_BUREAU_DAY     0
AMT_REQ_CREDIT_BUREAU_WEEK    0
AMT_REQ_CREDIT_BUREAU_MON     0
AMT_REQ_CREDIT_BUREAU_QRT     0
AMT_REQ_CREDIT_BUREAU_YEAR    0
dtype: int64
```

In [386...

```
for col in Amount_Cols:
    _ApplicationDF[col].fillna(_ApplicationDF[col].median(),inplace = True)
```

In [210...

```
_ApplicationDF[Amount_Cols].isnull().sum()
```

Out[210…

```
AMT_REQ_CREDIT_BUREAU_HOUR    0
AMT_REQ_CREDIT_BUREAU_DAY     0
AMT_REQ_CREDIT_BUREAU_WEEK    0
AMT_REQ_CREDIT_BUREAU_MON     0
AMT_REQ_CREDIT_BUREAU_QRT     0
AMT_REQ_CREDIT_BUREAU_YEAR    0
dtype: int64
```

In [388…

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

Out[388…

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	0.02
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	0.06
DEF_30_CNT_SOCIAL_CIRCLE	0.06
OBS_60_CNT_SOCIAL_CIRCLE	0.06
DEF_60_CNT_SOCIAL_CIRCLE	0.06
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               0.00
AMT_CREDIT_RANGE               0.00
AGE                            0.00
AGE_GROUP                      0.00
YEARS_EMPLOYED                 0.00
EMPLOYMENT_YEAR                4.99
dtype: float64
```

4.6.2 Imputing Null Values in previousDF

In [214…

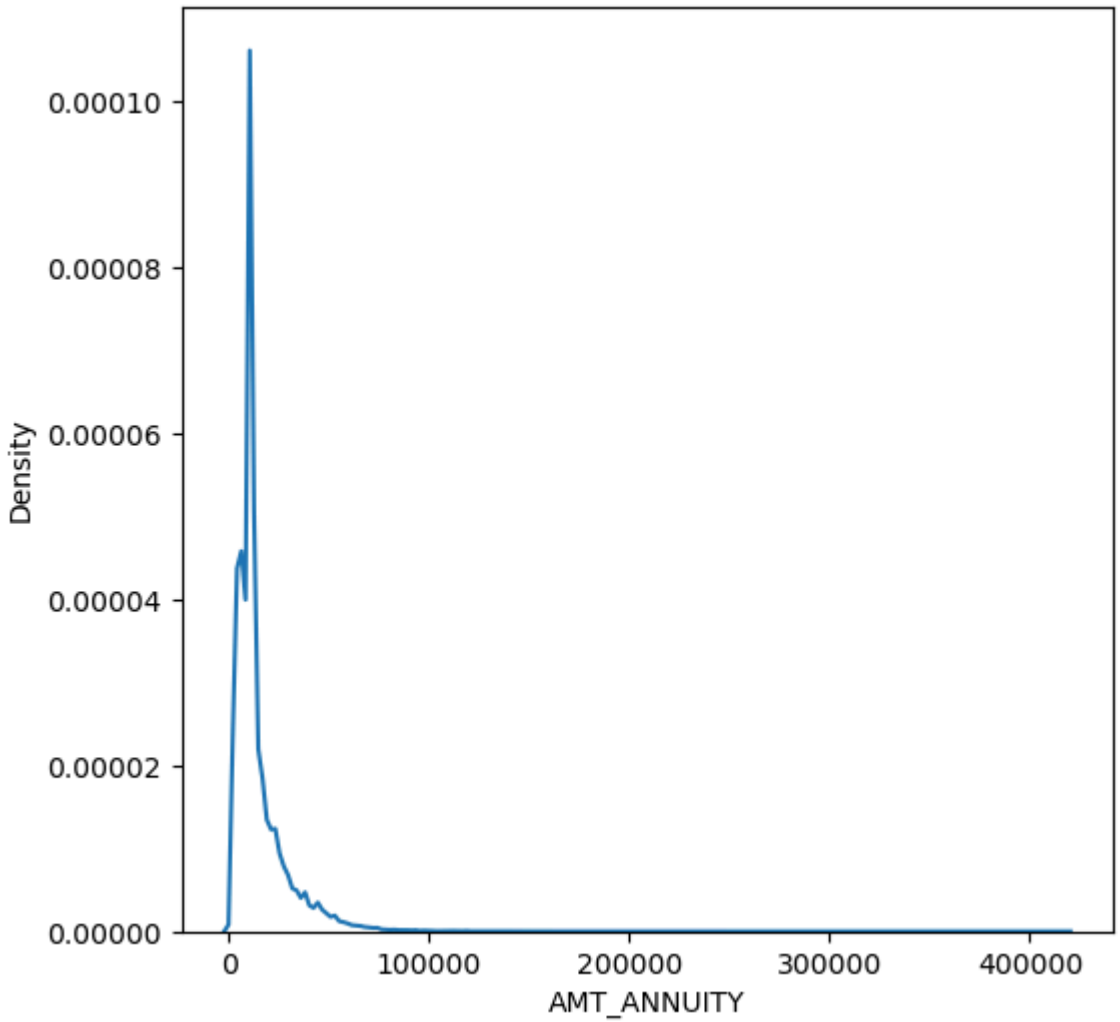
```
# checking the null value % of each column in previousDF dataframe
round(_PrevApplicationDF.isnull().sum() / _PrevApplicationDF.shape[0] * 100.00,2)
```

Out[214…

```
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_GOODS_PRICE          23.08
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
NAME_SELLER_INDUSTRY      0.00
CNT_PAYMENT              22.29
NAME_YIELD_GROUP          0.00
PRODUCT_COMBINATION       0.02
dtype: float64
```

In [390…

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



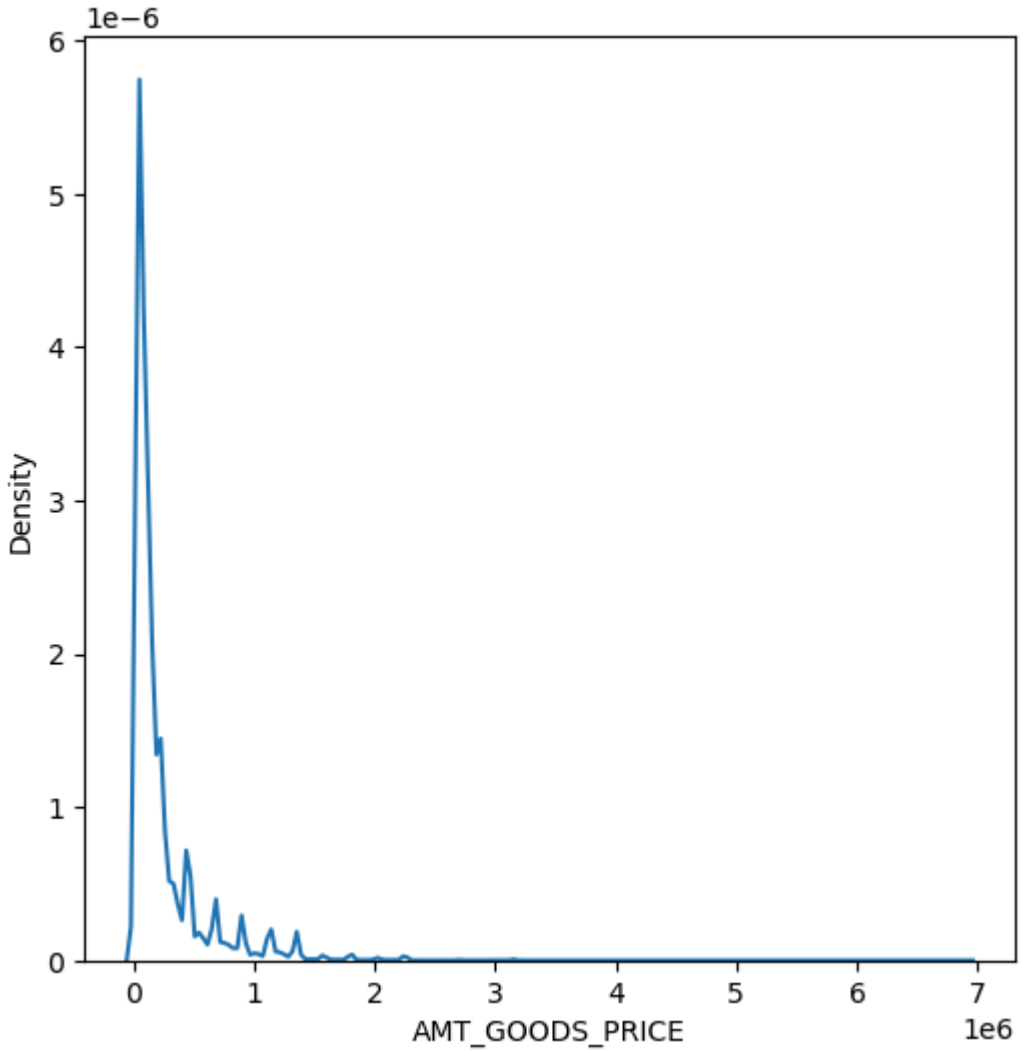
In [219…
```
_PrevApplicationDF['AMT_ANNUITY'].fillna(_PrevApplicationDF['AMT_ANNUITY'].median(),inplace = True)
```

In [221…
```
_PrevApplicationDF['AMT_ANNUITY'].isnull().sum()
```

Out[221…
```
0
```

In [223…
```
plt.figure(figsize=(6,6))
sns.kdeplot(_PrevApplicationDF['AMT_GOODS_PRICE'][pd.notnull(_PrevApplicationDF['AMT_GOODS_PRICE'])])
```

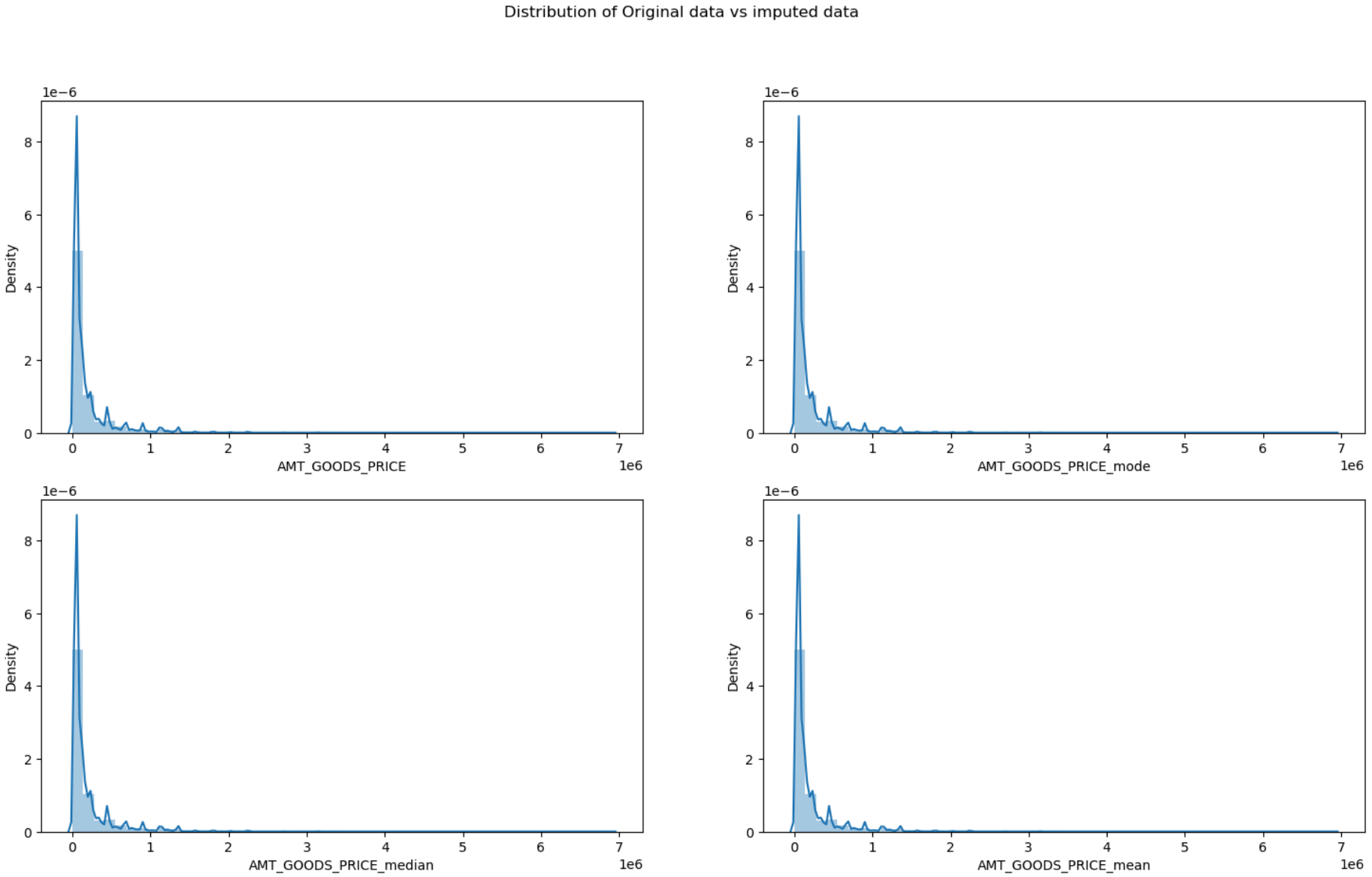
```
plt.show()
```



In [392…

```
statsDF = pd.DataFrame() # new dataframe with columns imputed with mode, median and mean
statsDF['AMT_GOODS_PRICE_mode'] = _PrevApplicationDF['AMT_GOODS_PRICE'].fillna(_PrevApplicationDF['AMT_GOODS_PRICE'].mode()[0]
statsDF['AMT_GOODS_PRICE_median'] = _PrevApplicationDF['AMT_GOODS_PRICE'].fillna(_PrevApplicationDF['AMT_GOODS_PRICE'].median(
statsDF['AMT_GOODS_PRICE_mean'] = _PrevApplicationDF['AMT_GOODS_PRICE'].fillna(_PrevApplicationDF['AMT_GOODS_PRICE'].mean())
cols = ['AMT_GOODS_PRICE_mode', 'AMT_GOODS_PRICE_median','AMT_GOODS_PRICE_mean']
```

```
plt.figure(figsize=(18,10))
plt.suptitle('Distribution of Original data vs imputed data')
plt.subplot(221)
sns.distplot(_PrevApplicationDF['AMT_GOODS_PRICE'][pd.notnull(_PrevApplicationDF['AMT_GOODS_PRICE'])]);
for i in enumerate(cols):
    plt.subplot(2,2,i[0]+2)
    sns.distplot(statsDF[i[1]])
plt.show()
```



In [227…
```
_PrevApplicationDF['AMT_GOODS_PRICE'].isnull().sum()
```

Out[227…
```
385515
```

In [229…
```
_PrevApplicationDF['AMT_GOODS_PRICE'].fillna(_PrevApplicationDF['AMT_GOODS_PRICE'].mode()[0], inplace=True)
```

In [231…
```
_PrevApplicationDF['AMT_GOODS_PRICE'].isnull().sum()
```

Out[231…
```
0
```

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

In [233…
```
_PrevApplicationDF.loc[_PrevApplicationDF['CNT_PAYMENT'].isnull(),'NAME_CONTRACT_STATUS'].value_counts()
```

Out[233…
```
NAME_CONTRACT_STATUS
Canceled        305805
Refused          40897
Unused offer     25524
Approved             4
Name: count, dtype: int64
```

In [235…
```
_PrevApplicationDF['CNT_PAYMENT'].fillna(0,inplace = True)
```

In [237…
```
_PrevApplicationDF['CNT_PAYMENT'].isnull().sum()
```

Out[237…
```
0
```

Checking the null % in each column on PreviousApplicationDF:

In [239…
```
np.round(_PrevApplicationDF.isnull().sum()/_PrevApplicationDF.shape[0] * 100,2)
```

Out[239…

```
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
NAME_SELLER_INDUSTRY      0.00
CNT_PAYMENT               0.00
NAME_YIELD_GROUP          0.00
PRODUCT_COMBINATION       0.02
dtype: float64
```

Insight:

If we check the Null% of the columns in PreviousApplicationDF except for PRODUCT_COMBINATION rest all are 0 and even in PRODUCT_COMBINATION its 0.02 which is very minimal and hence can be ignored.

Identifying the outliers in ApplicationDF

In [241…

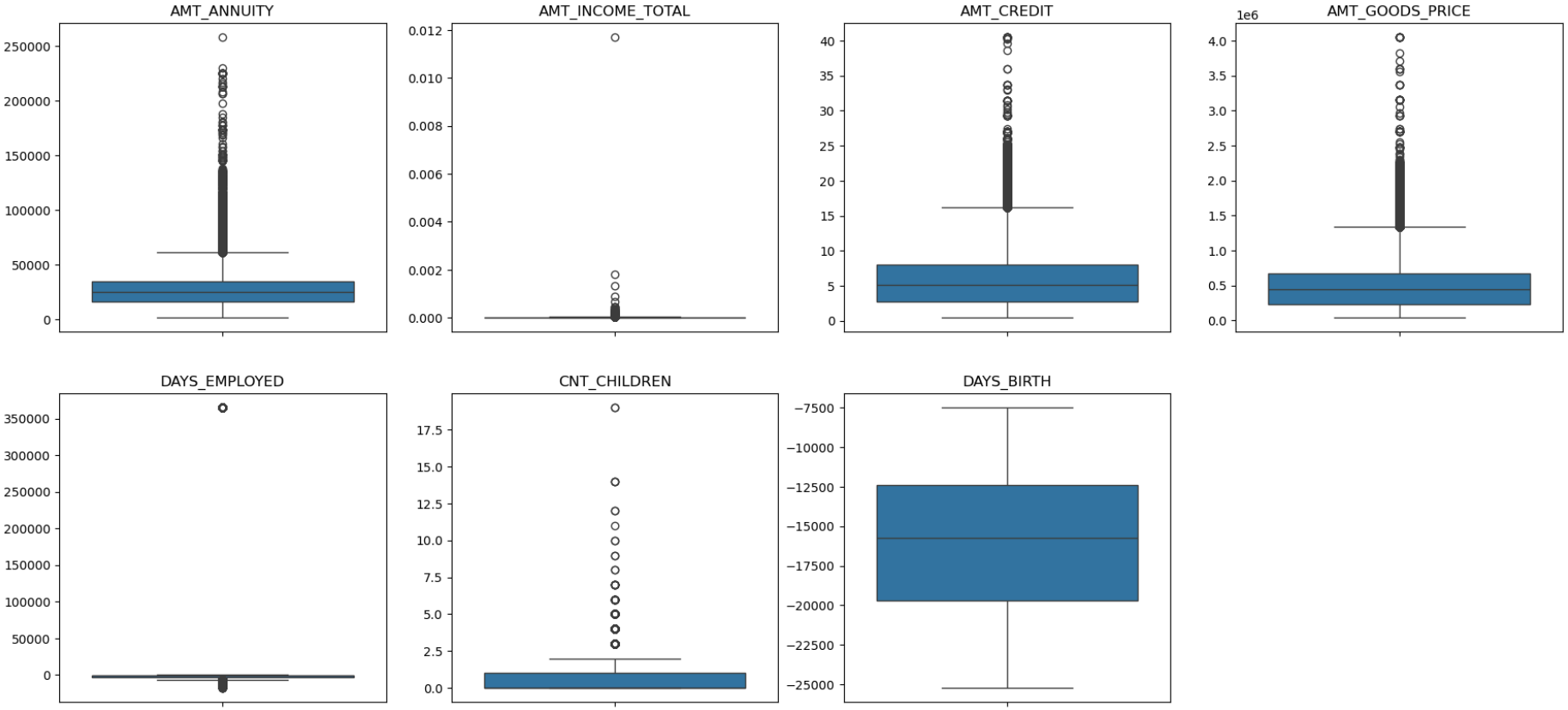
```
plt.figure(figsize=(22,10))

app_outlier_col_1 = ['AMT_ANNUITY','AMT_INCOME_TOTAL','AMT_CREDIT','AMT_GOODS_PRICE','DAYS_EMPLOYED']
app_outlier_col_2 = ['CNT_CHILDREN','DAYS_BIRTH']
for i in enumerate(app_outlier_col_1):
    plt.subplot(2,4,i[0]+1)
    sns.boxplot(y=_ApplicationDF[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=_ApplicationDF[i[1]])
    plt.title(i[1])
    plt.ylabel("")

plt.show()
```



Insight:

It can be seen that in current application data:

1.AMT_ANNUITY, AMT_CREDIT, AMT_GOODS_PRICE,CNT_CHILDREN have some number of outliers.

2.AMT_INCOME_TOTAL has huge number of outliers which indicate that few of the loan applicants have high income when compared to the others.

3.DAYS_BIRTH has no outliers which means the data available is reliable.

4.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 [243…
```
_ApplicationDF[['AMT_ANNUITY', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_GOODS_PRICE', 'DAYS_BIRTH','CNT_CHILDREN','DAYS_EMPLOYED
```

Out[243…

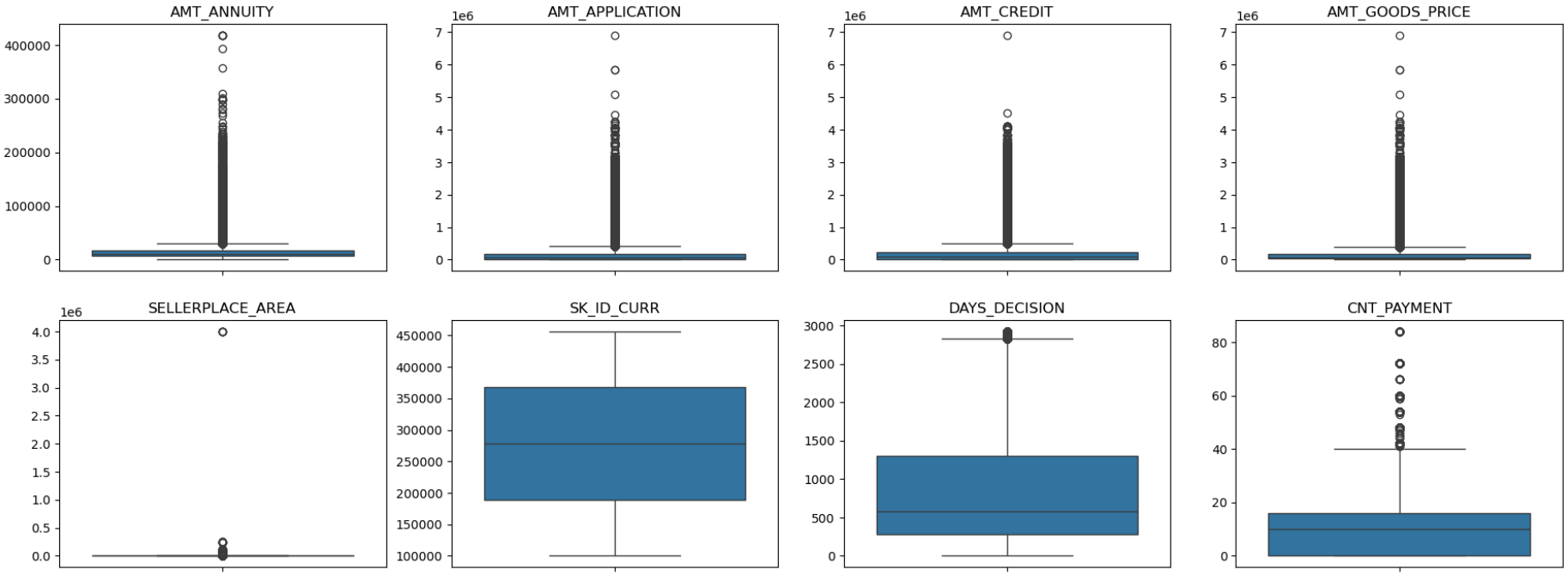
	AMT_ANNUITY	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_GOODS_PRICE	DAYS_BIRTH	CNT_CHILDREN	DAYS_EMPLOYED
count	307499.000000	307511.000000	307511.000000	3.072330e+05	307511.000000	307511.000000	307511.000000
mean	27108.573909	0.000017	5.990260	5.383962e+05	-16036.995067	0.417052	63815.045904
std	14493.737315	0.000024	4.024908	3.694465e+05	4363.988632	0.722121	141275.766519
min	1615.500000	0.000003	0.450000	4.050000e+04	-25229.000000	0.000000	-17912.000000
25%	16524.000000	0.000011	2.700000	2.385000e+05	-19682.000000	0.000000	-2760.000000
50%	24903.000000	0.000015	5.135310	4.500000e+05	-15750.000000	0.000000	-1213.000000
75%	34596.000000	0.000020	8.086500	6.795000e+05	-12413.000000	1.000000	-289.000000
max	258025.500000	0.011700	40.500000	4.050000e+06	-7489.000000	19.000000	365243.000000

Identifying the outliers in PreviousApplicationDF

In [245…
```
plt.figure(figsize=(22,8))

prev_outlier_col_1 = ['AMT_ANNUITY','AMT_APPLICATION','AMT_CREDIT','AMT_GOODS_PRICE','SELLERPLACE_AREA']
prev_outlier_col_2 = ['SK_ID_CURR','DAYS_DECISION','CNT_PAYMENT']
for i in enumerate(prev_outlier_col_1):
    plt.subplot(2,4,i[0]+1)
    sns.boxplot(y=_PrevApplicationDF[i[1]])
    plt.title(i[1])
    plt.ylabel("")
```

```
for i in enumerate(prev_outlier_col_2):
    plt.subplot(2,4,i[0]+6)
    sns.boxplot(y=_PrevApplicationDF[i[1]])
    plt.title(i[1])
    plt.ylabel("")
plt.show()
```



In [247…

```
_PrevApplicationDF[['AMT_ANNUITY', 'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_GOODS_PRICE', 'SELLERPLACE_AREA','CNT_PAYMENT','DAYS_
```

Out[247...

	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_GOODS_PRICE	SELLERPLACE_AREA	CNT_PAYMENT	DAYS_DECISION
count	1.670214e+06	1.670214e+06	1.670213e+06	1.670214e+06	1.670214e+06	1.670214e+06	1.670214e+06
mean	1.490651e+04	1.752339e+05	1.961140e+05	1.856429e+05	3.139511e+02	1.247621e+01	8.806797e+02
std	1.317751e+04	2.927798e+05	3.185746e+05	2.871413e+05	7.127443e+03	1.447588e+01	7.790997e+02
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-1.000000e+00	0.000000e+00	1.000000e+00
25%	7.547096e+03	1.872000e+04	2.416050e+04	4.500000e+04	-1.000000e+00	0.000000e+00	2.800000e+02
50%	1.125000e+04	7.104600e+04	8.054100e+04	7.105050e+04	3.000000e+00	1.000000e+01	5.810000e+02
75%	1.682403e+04	1.803600e+05	2.164185e+05	1.804050e+05	8.200000e+01	1.600000e+01	1.300000e+03
max	4.180581e+05	6.905160e+06	6.905160e+06	6.905160e+06	4.000000e+06	8.400000e+01	2.922000e+03

Data Analysis

Strategy:

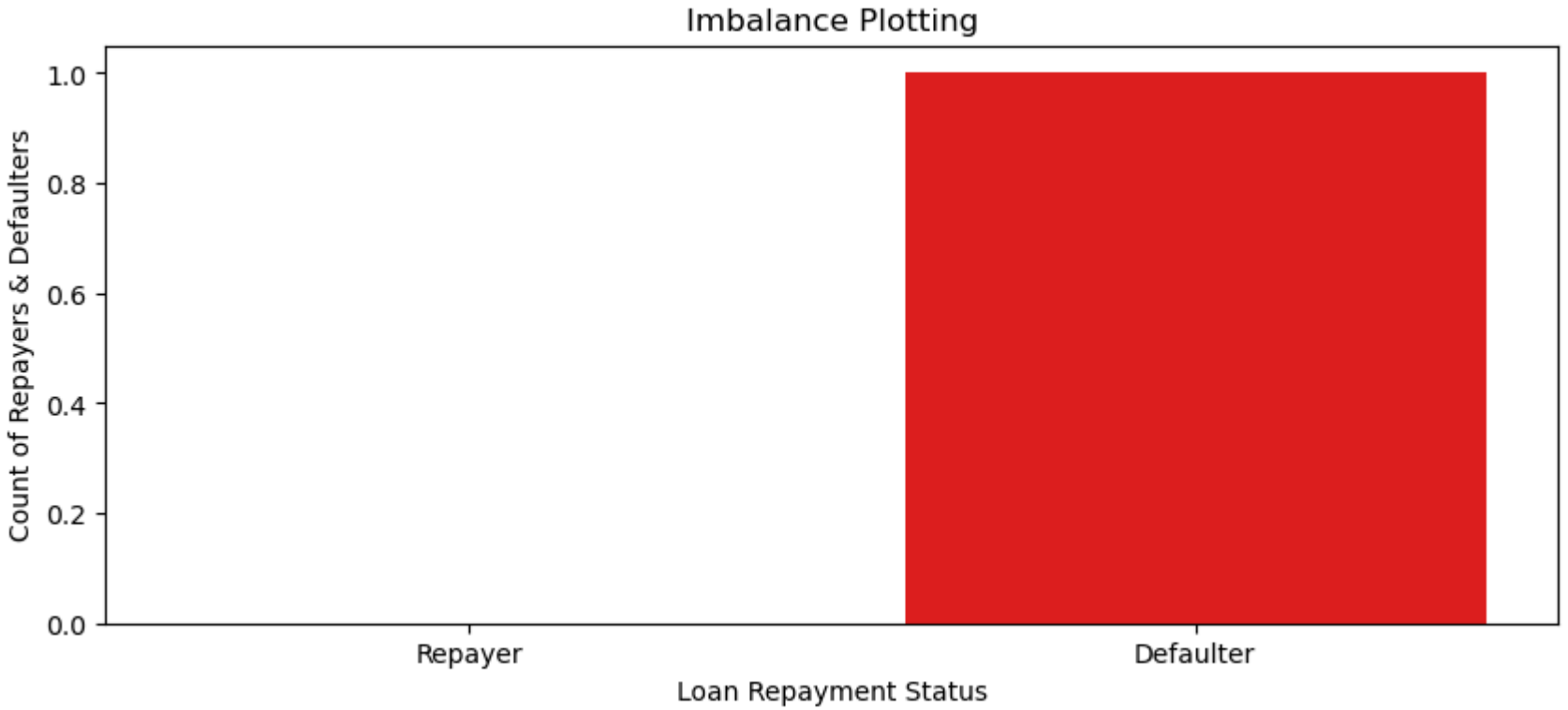
The data analysis flow has been planned in following way :

1.Imbalance in Data
2.Categorical Data Analysis
    Categorical segmented Univariate Analysis
    Categorical Bi/Multivariate analysis
3.Numeric Data Analysis
    Bi-furcation of databased based on TARGET data
    Correlation Matrix
    Numerical segmented Univariate Analysis
    Numerical Bi/Multivariate analysis

In [249...

```
Imbalance = _ApplicationDF["TARGET"].value_counts().reset_index()
plt.figure(figsize=(10,4))
x= ['Repayer','Defaulter']
sns.barplot(x=x,y="TARGET",data = Imbalance,palette= ['g','r'])
```

```
plt.xlabel("Loan Repayment Status")
plt.ylabel("Count of Repayers & Defaulters")
plt.title("Imbalance Plotting")
plt.show()
```



In [251…

```
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: %.2f and %.2f'%(count_0_perc,count_1
print('Ratios of imbalance in relative with respect to Repayer and Defaulter datas is %.2f : 1 (approx)'%(count_0/count_1))
```

```
Ratios of imbalance in percentage with respect to Repayer and Defaulter datas are: 0.00 and 100.00
Ratios of imbalance in relative with respect to Repayer and Defaulter datas is 0.00 : 1 (approx)
```

In [253…

```
def univariate_categorical(feature,ylog=False,label_rotation=False,horizontal_layout=True):
    temp = _ApplicationDF[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 = _ApplicationDF[[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=_ApplicationDF,
                      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' : 'Blue'})


    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, 'color' : 'Blue'})

    plt.show();
```

In [255…

```
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 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 [257…

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

    plt.figure(figsize=figsize)
    sns.relplot(x=x,
                y=y,
                data=_ApplicationDF,
                hue="TARGET",
                kind=kind,
                palette = ['g','r'],
                legend = False)
    plt.legend(['Repayer','Defaulter'])
```

```
    plt.xticks(rotation=90, ha='right')
    plt.show()
```

In [259…

```
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' : 3, 'color' : 'Blue'})
    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()
```

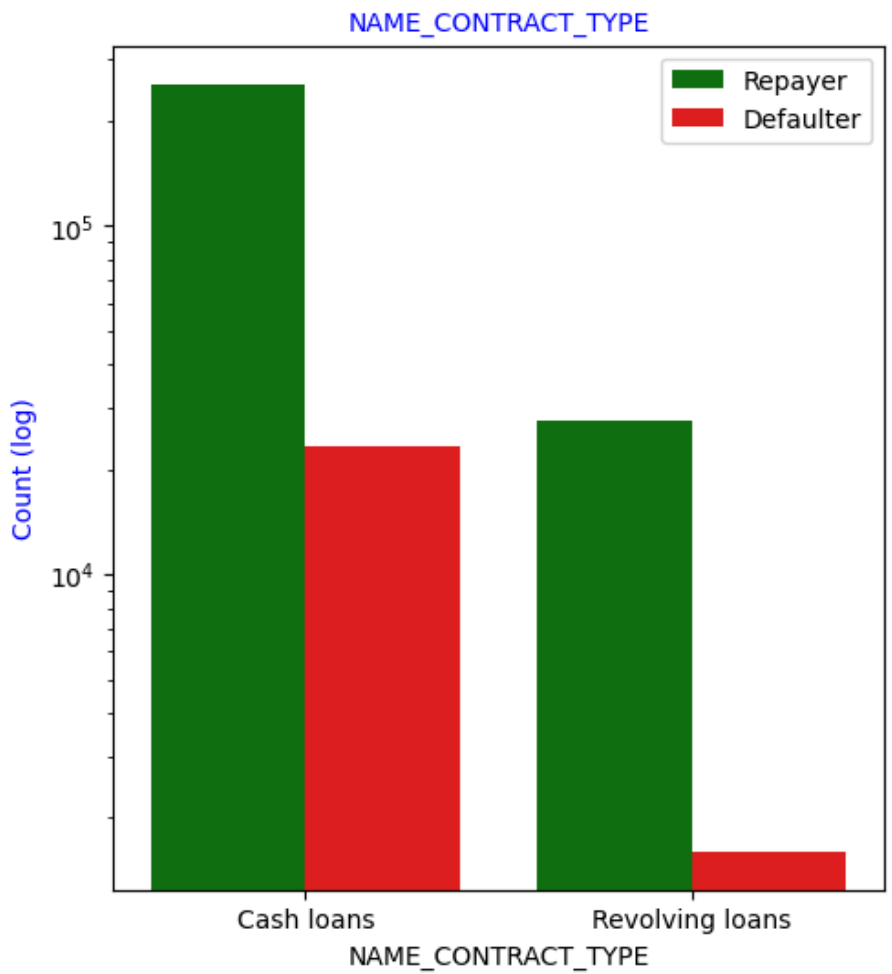
In [261…

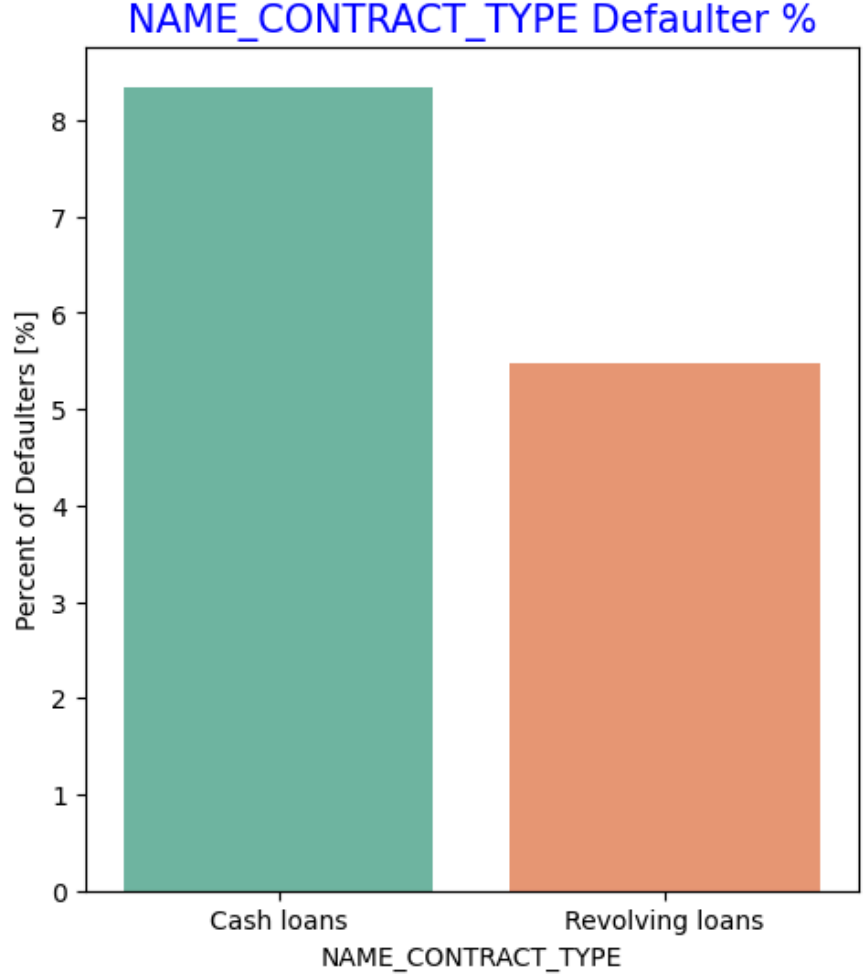
```
# Function to plot point plots on merged dataframe

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'])
```

In [263…

```
univariate_categorical('NAME_CONTRACT_TYPE',True)
```



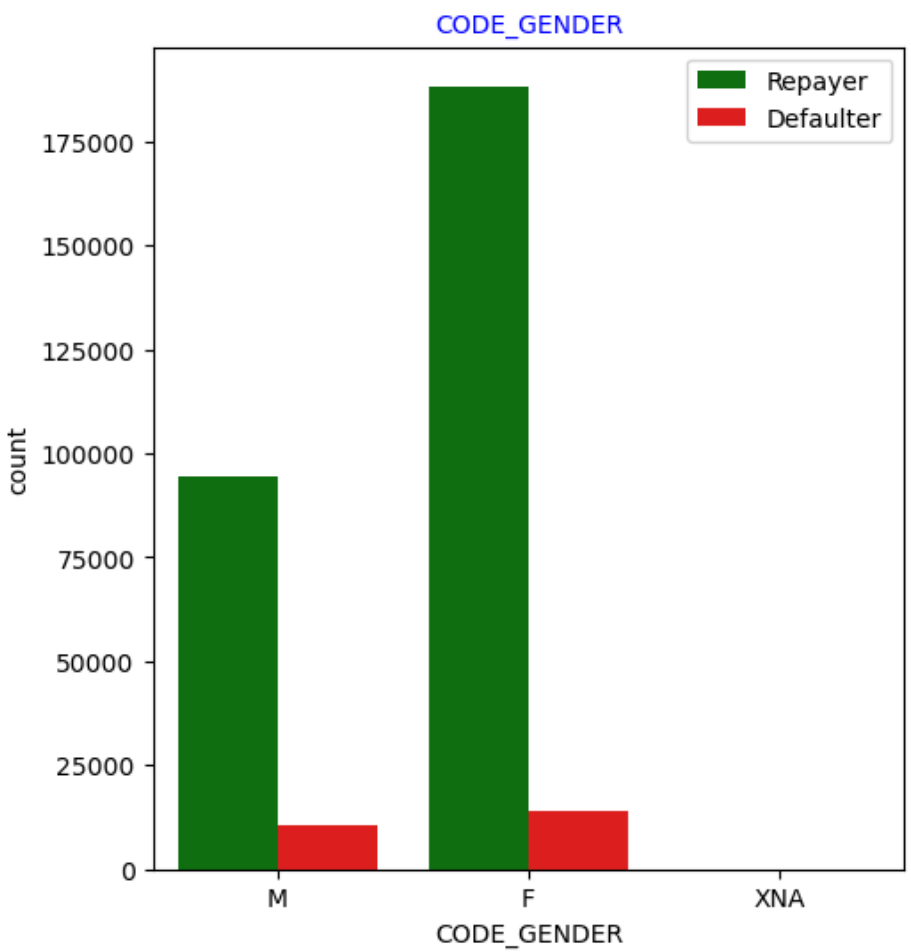


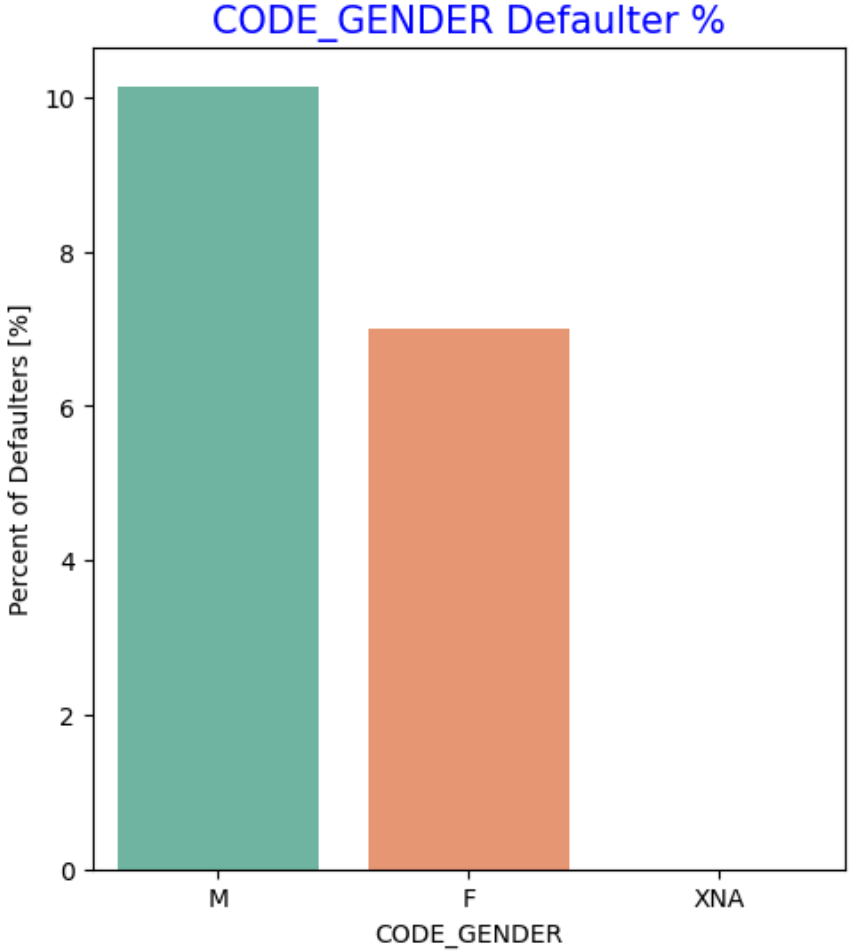
Inferences:

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 [265…

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



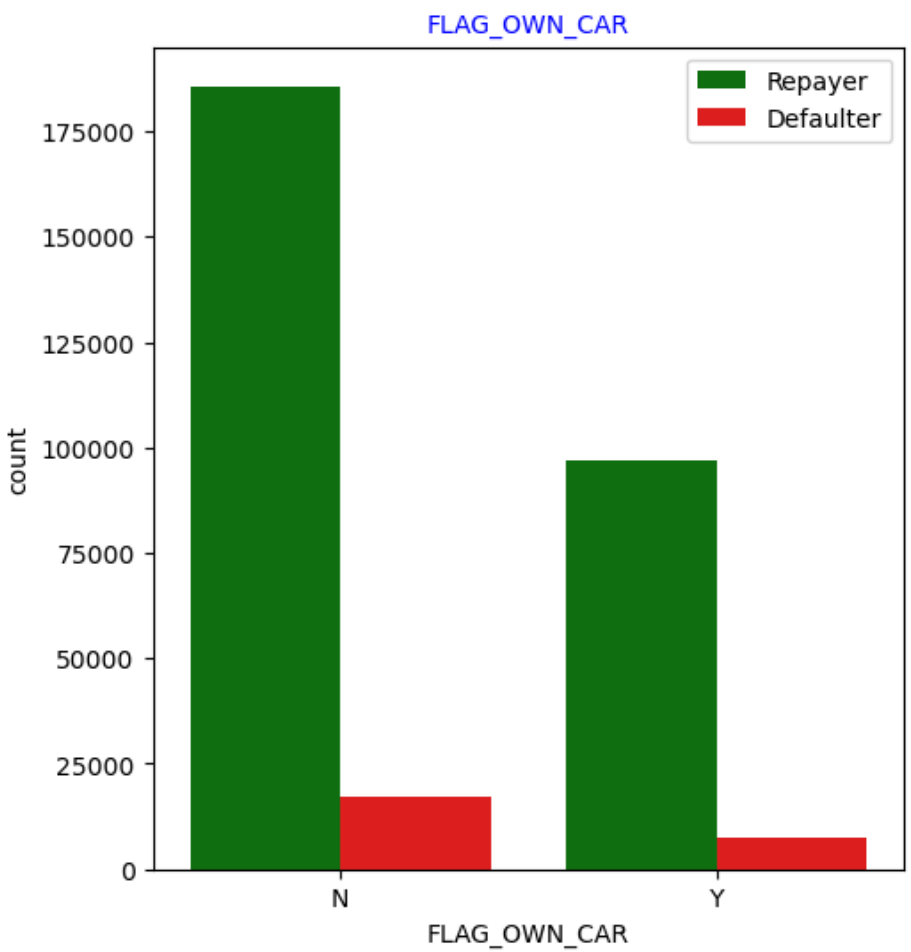


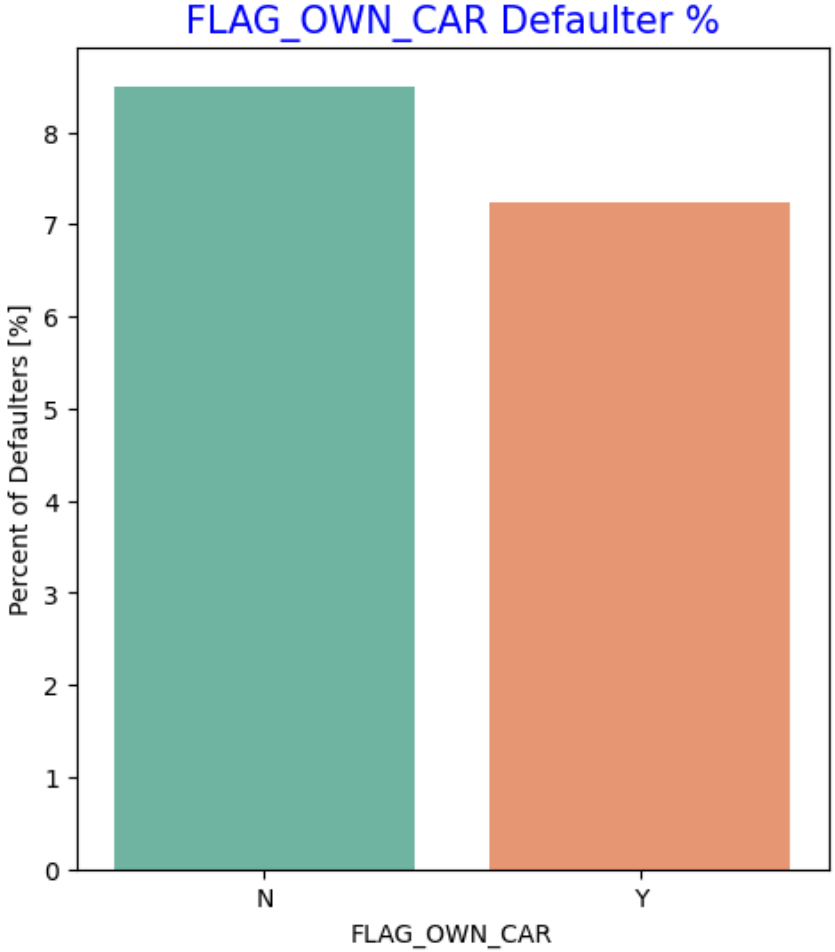
Inferences:

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 [267…

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



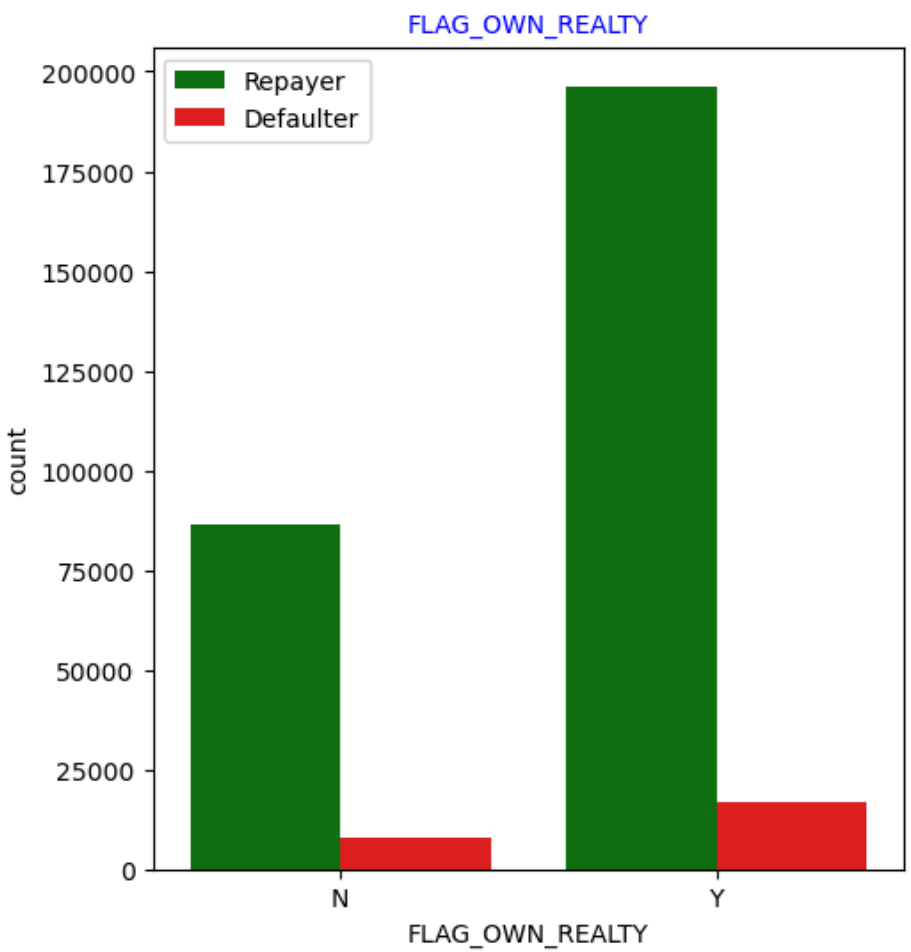


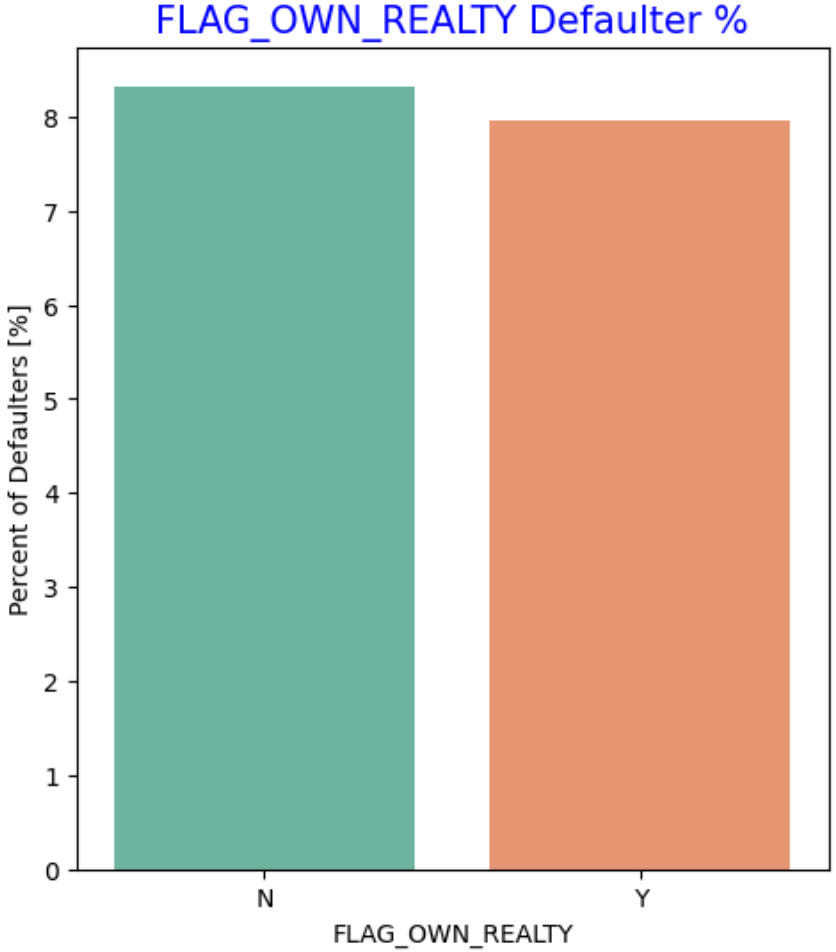
Inferences:

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 [269…

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



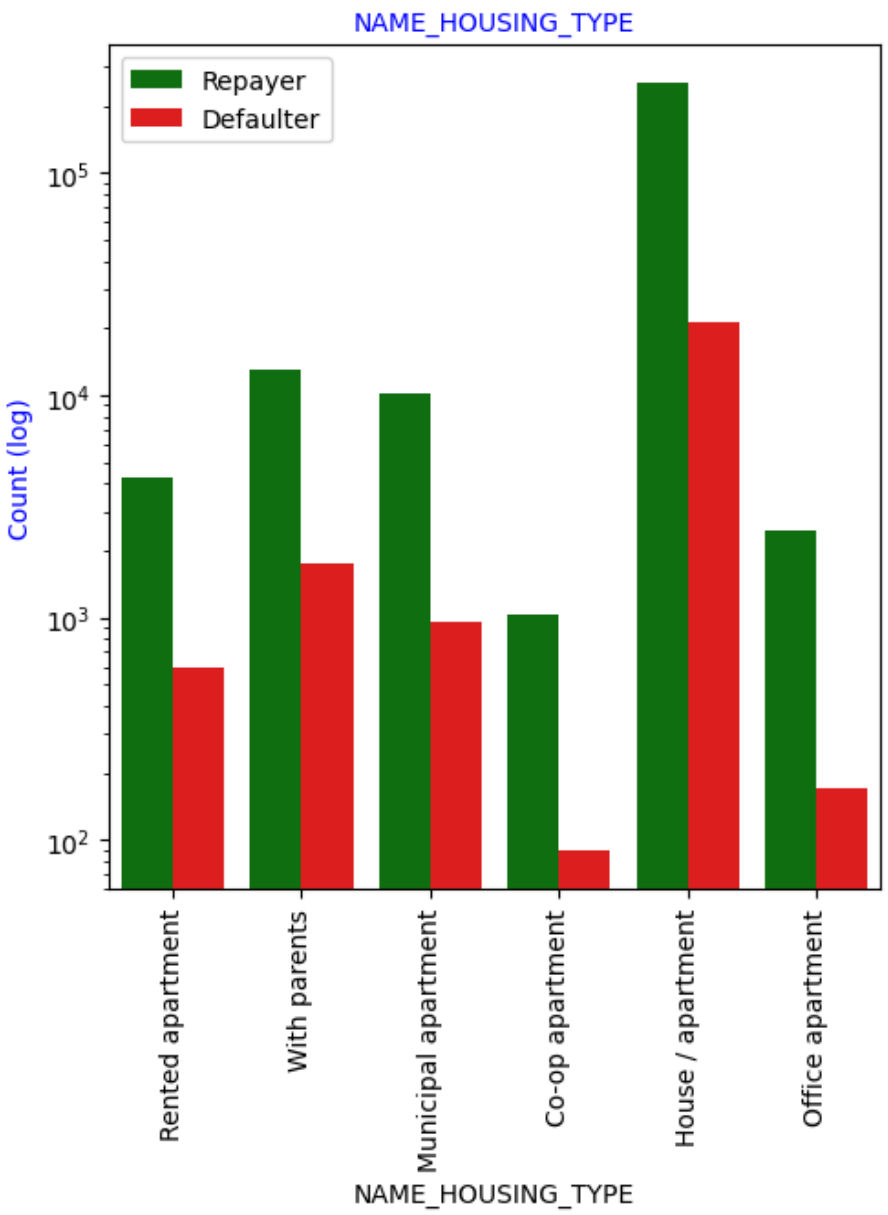


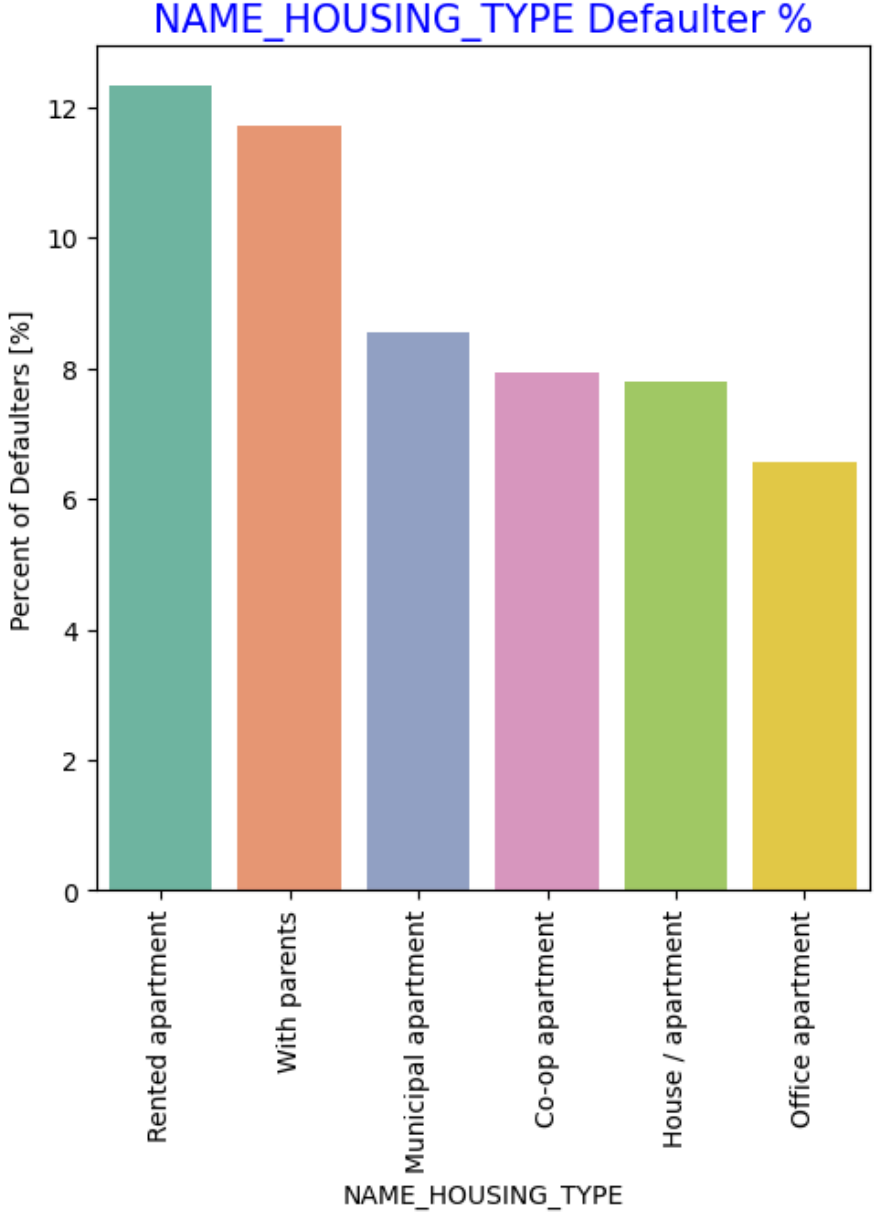
Inferences:

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 [271…

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





Inferences:

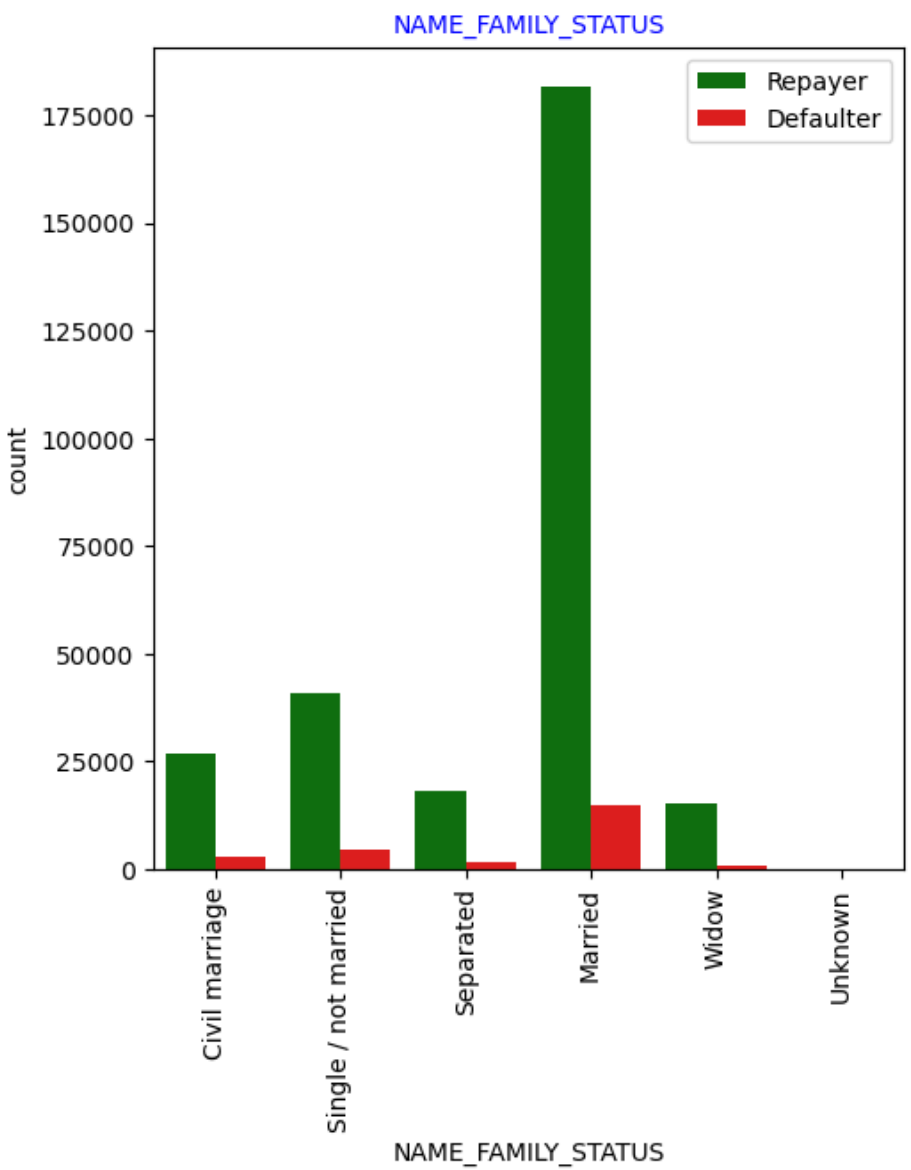
1.Majority of people live in House/apartment

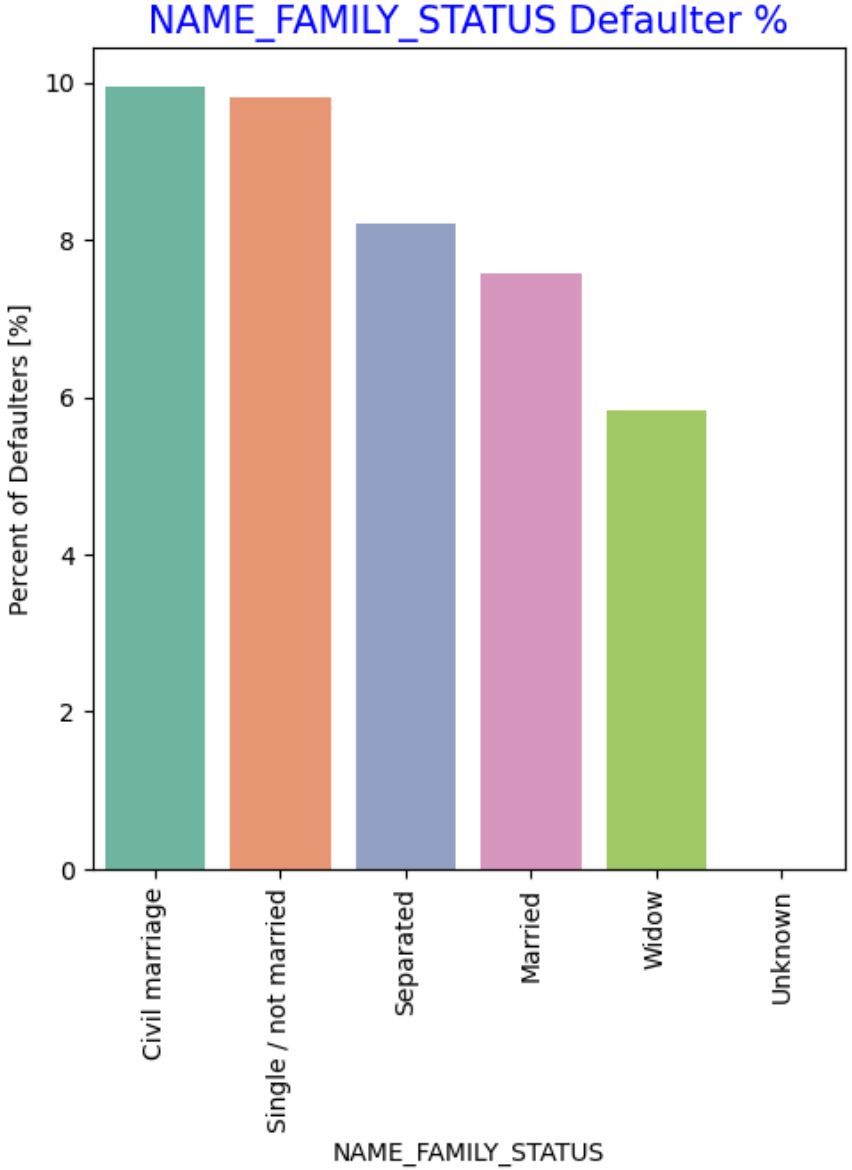
2.People living in office apartments have lowest default rate

3.People living with parents (~11.5%) and living in rented apartments(>12%) have higher probability of defaulting.

In [273…

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





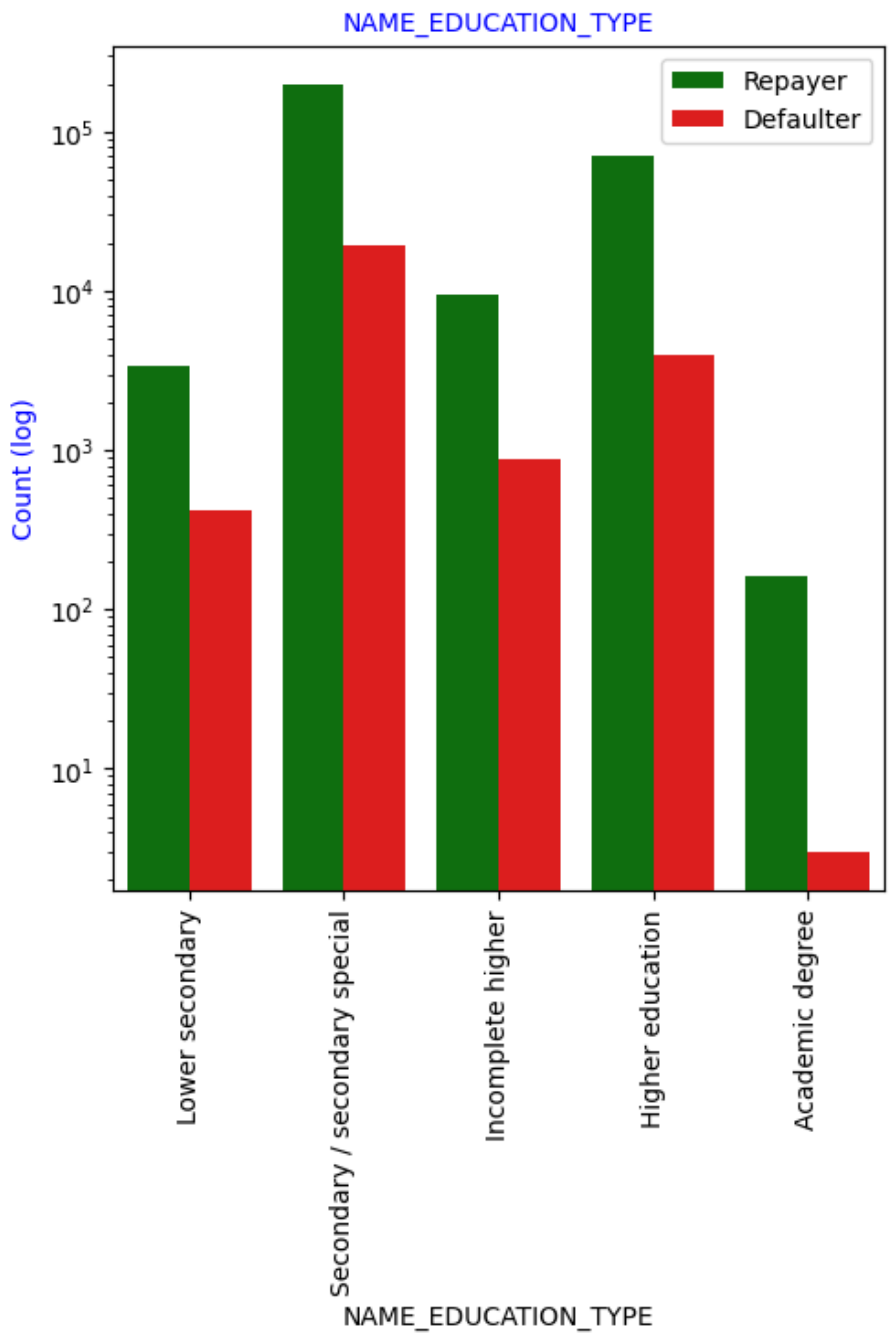
Inferences:

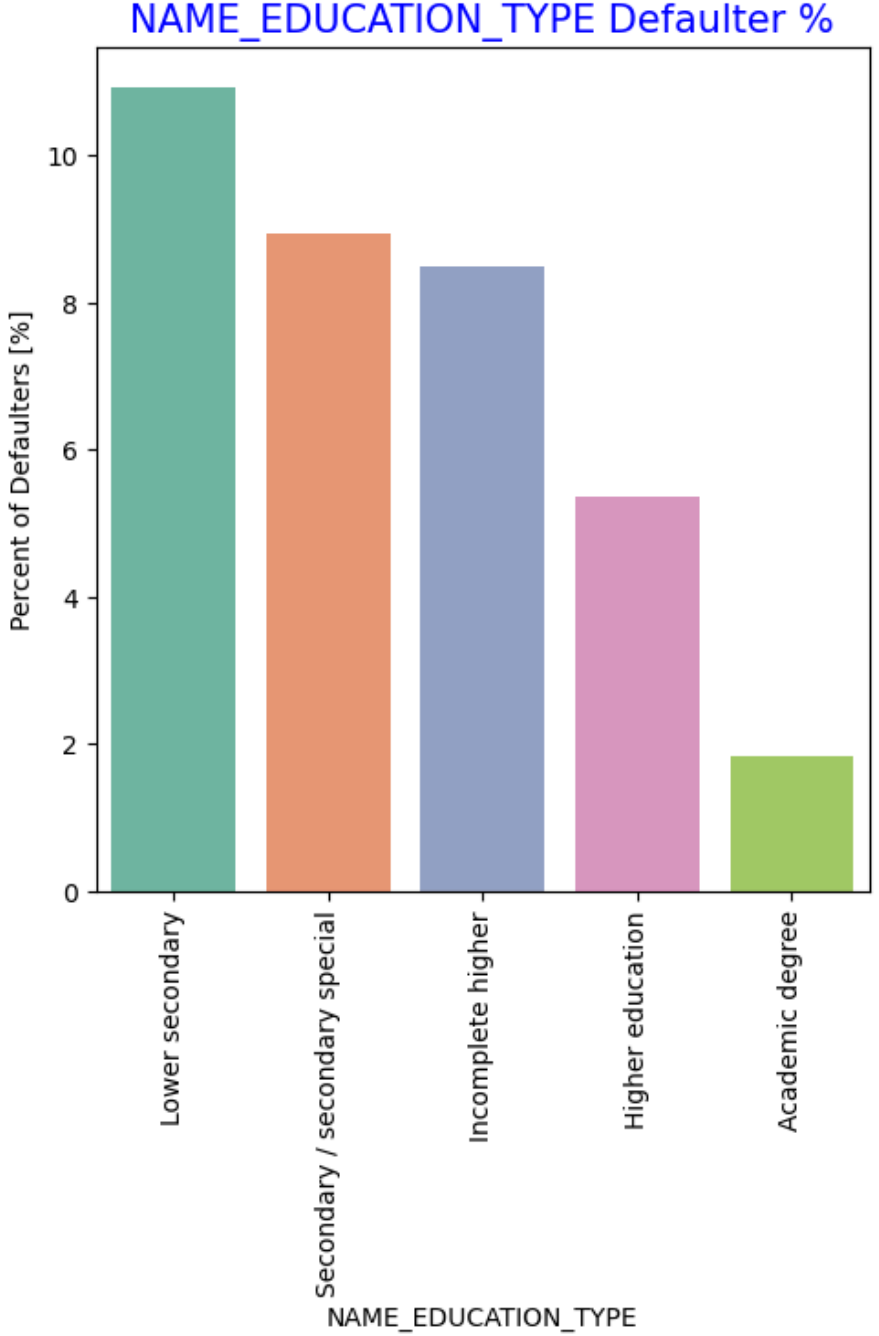
1.Most of the people who have taken loan are married, followed by Single/not married and civil marriage.

2.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 [275…

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


NAME_EDUCATION_TYPE
Repayer
Defaulter
Count (log)
10^5
10^4
10^3
10^2
10^1
Lower secondary
Secondary / secondary special
Incomplete higher
Higher education
Academic degree
NAME_EDUCATION_TYPE


NAME_EDUCATION_TYPE Defaulter %
Percent of Defaulters [%]
10
8
6
4
2
0
Lower secondary
Secondary / secondary special
Incomplete higher
Higher education
Academic degree
NAME_EDUCATION_TYPE

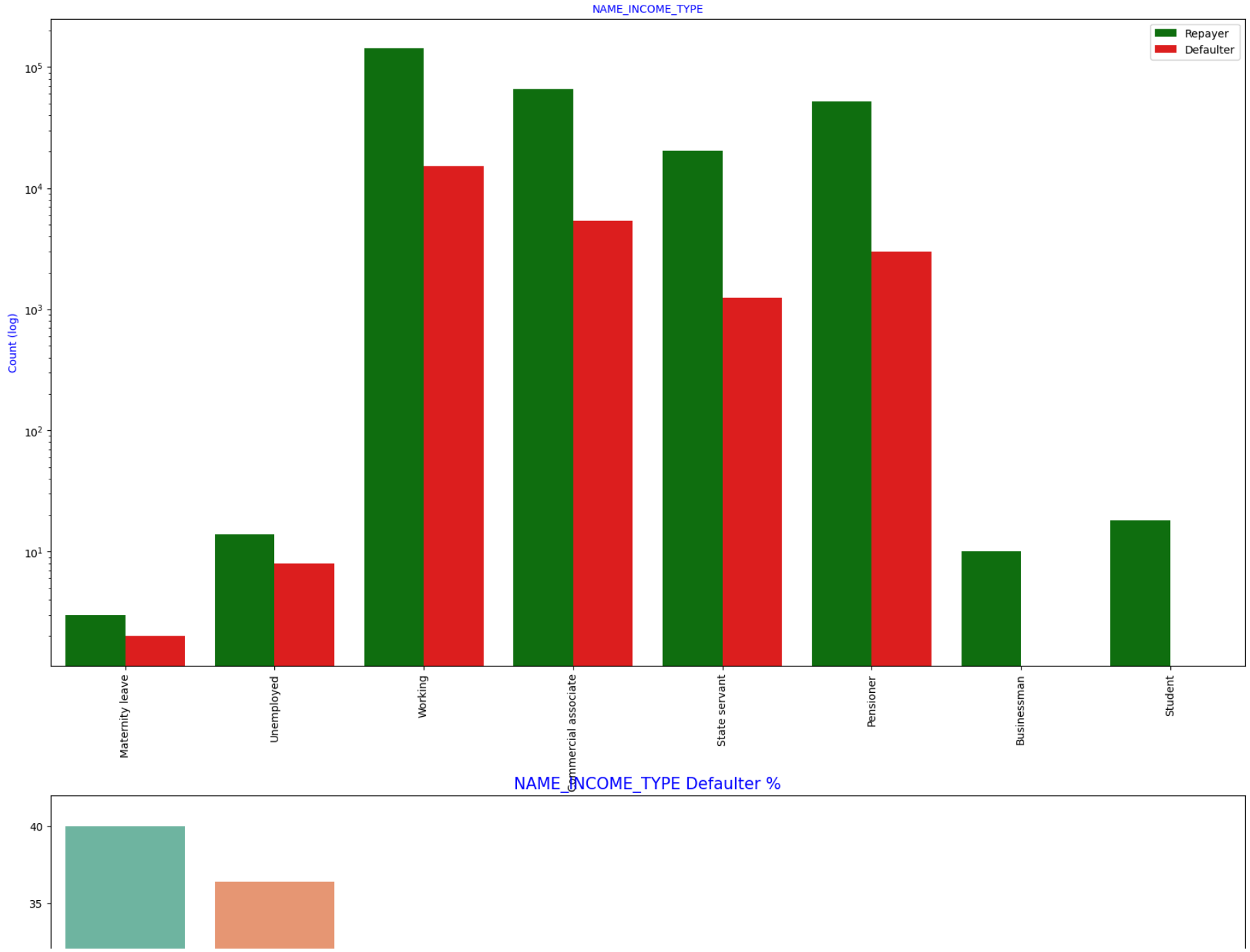
Inferences:

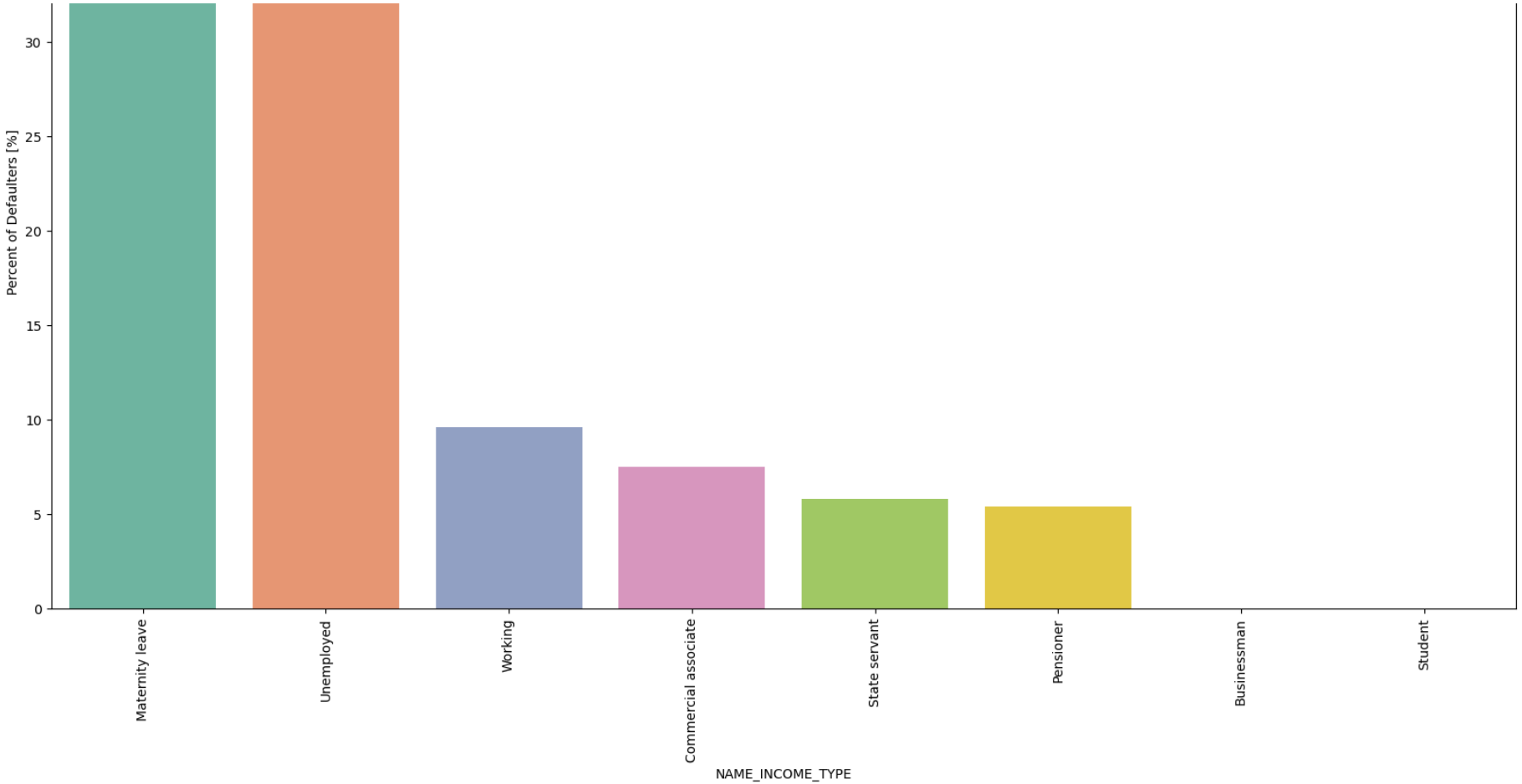
1.Majority of the clients have Secondary / secondary special education, followed by clients with Higher education. Only a very small number having an academic degree.

2.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 [277…

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


NAME_INCOME_TYPE
Repayer
Defaulter
Count (log)
10^5
10^4
10^3
10^2
10^1
Maternity leave
Unemployed
Working
Commercial associate
State servant
Pensioner
Businessman
Student
NAME_INCOME_TYPE Defaulter %
40
35



Inferences:

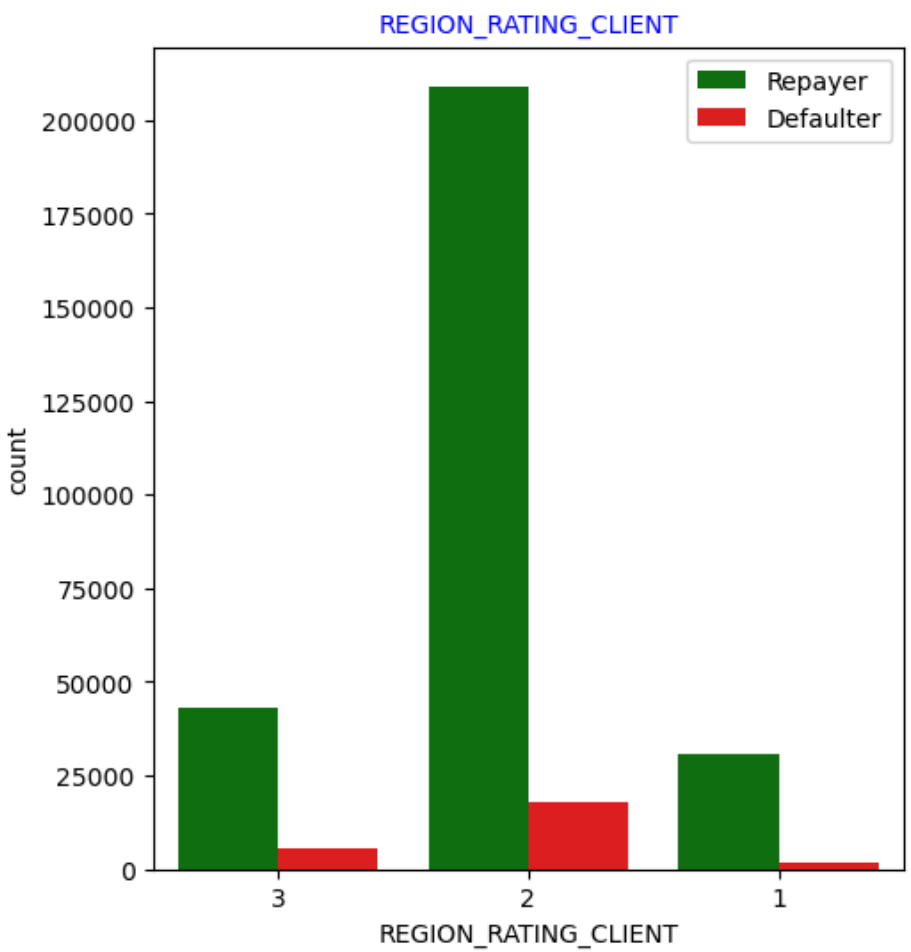
1.Most of applicants for loans have income type as Working, followed by Commercial associate, Pensioner and State servant.

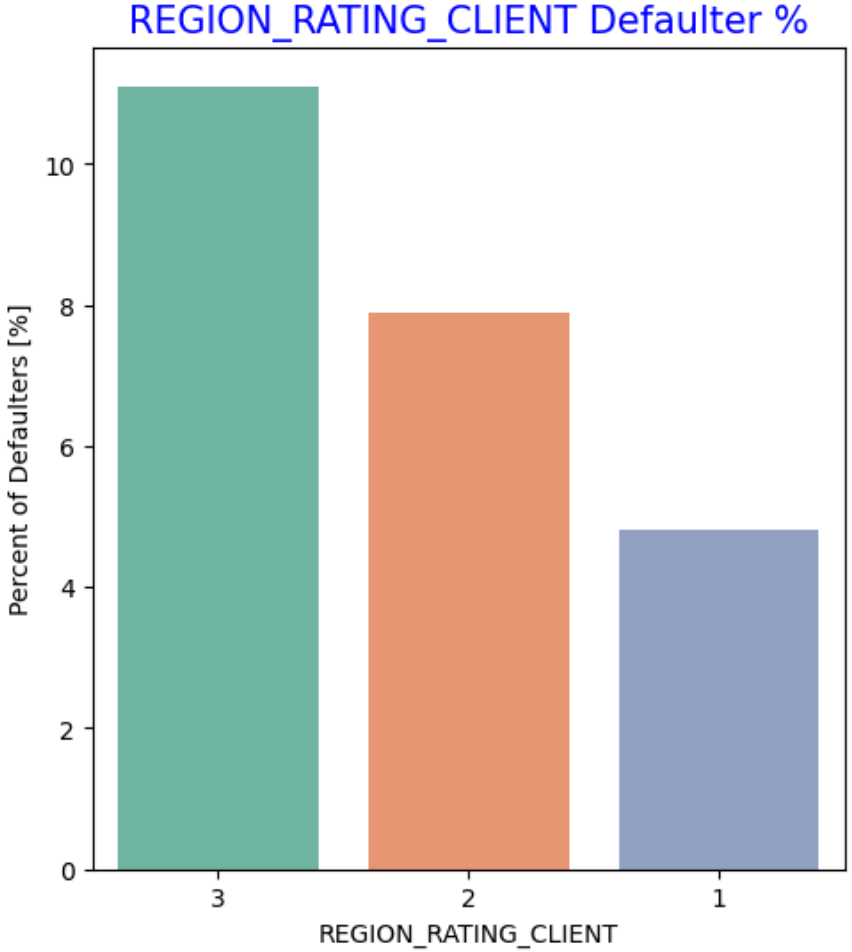
2.The applicants with the type of income Maternity leave have almost 40% ratio of not returning loans, followed by Unemployed (37%). The rest of types of incomes are under the average of 10% for not returning loans.

3.Student and Businessmen, though less in numbers do not have any default record. Thus these two category are safest for providing loan.

In [279…

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





Inferences:

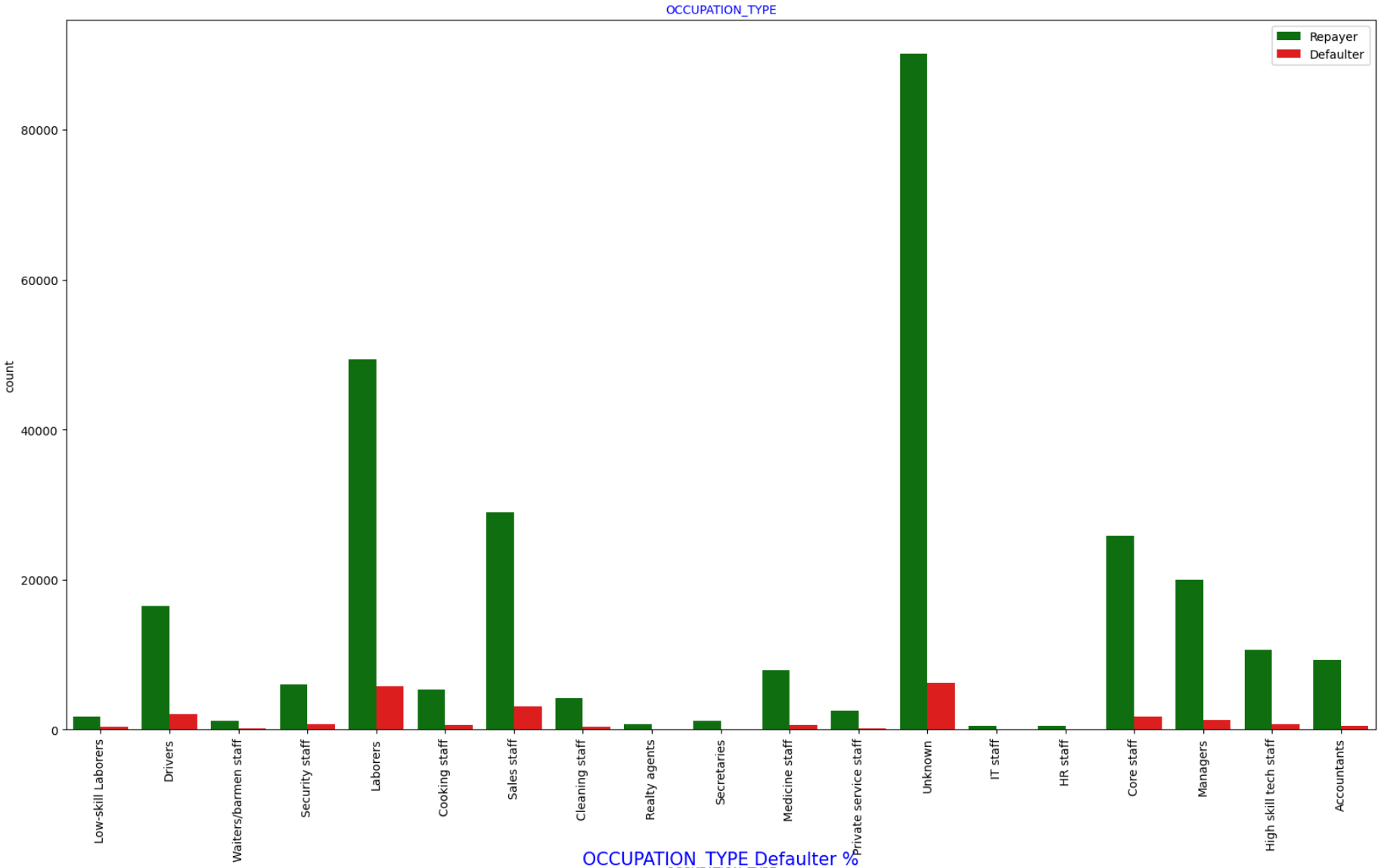
1.Most of the applicants are living in Region_Rating 2 place.

2.Region Rating 3 has the highest default rate (11%).

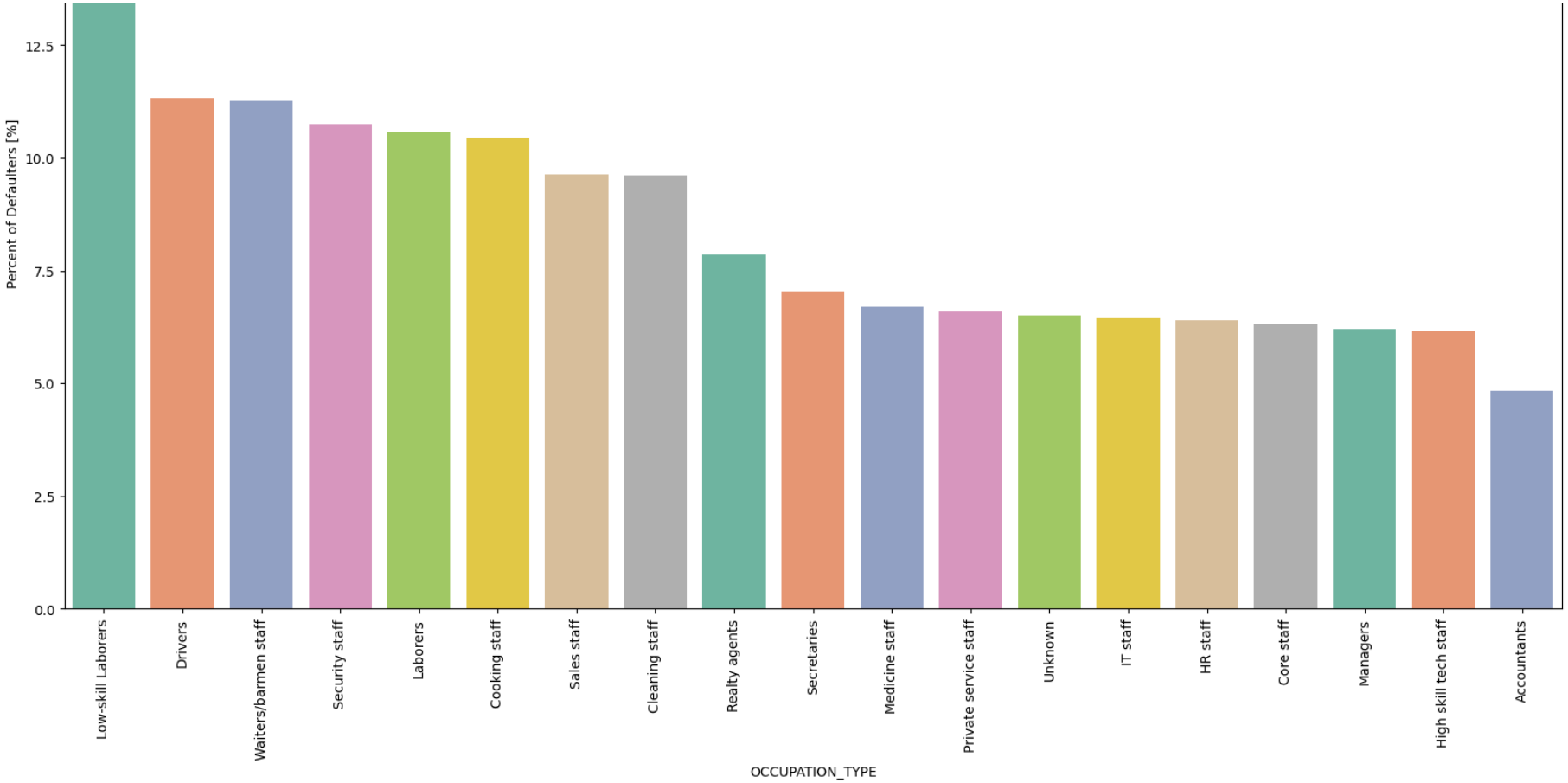
3.Applicant living in Region_Rating 1 has the lowest probability of defaulting, thus safer for approving loans.

In [281…

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


OCCUPATION_TYPE
Repayer
Defaulter
count
80000
60000
40000
20000
0
Low-skill Laborers
Drivers
Waiters/barmen staff
Security staff
Laborers
Cooking staff
Sales staff
Cleaning staff
Realty agents
Secretaries
Medicine staff
Private service staff
Unknown
IT staff
HR staff
Core staff
Managers
High skill tech staff
Accountants


OCCUPATION_TYPE Defaulter %
17.5
15.0



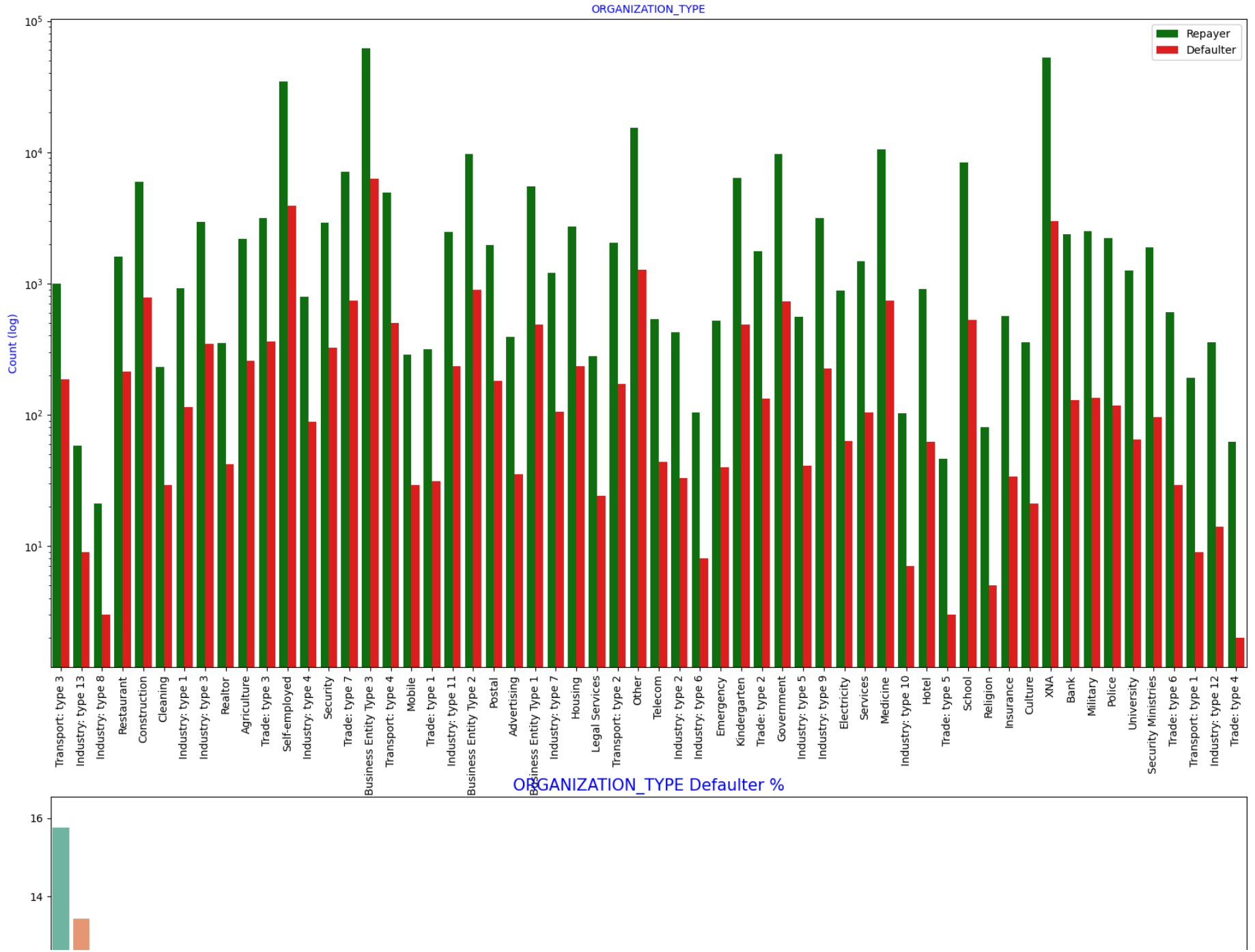
Inferences:

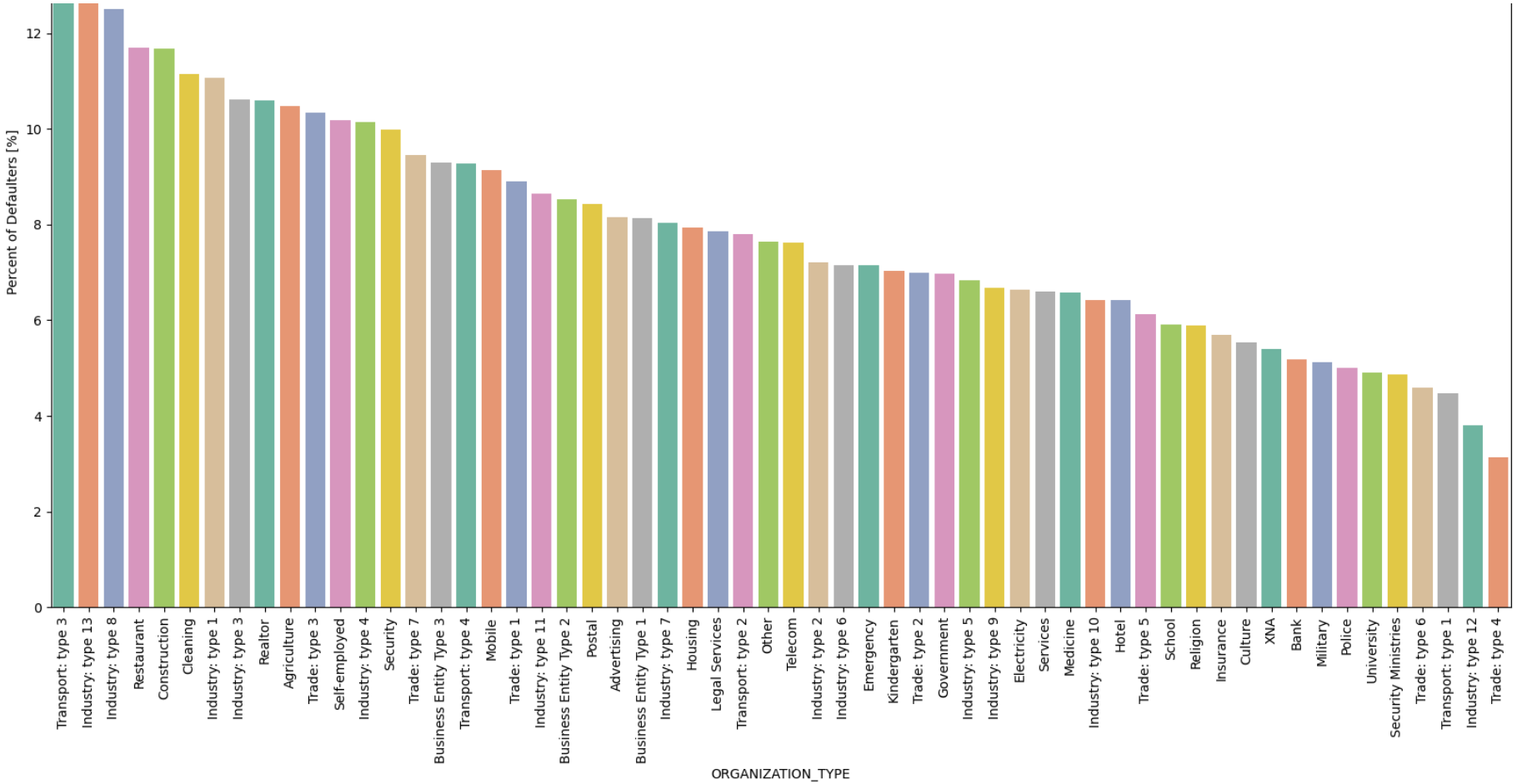
1.Most of the loans are taken by Laborers, followed by Sales staff. IT staff take the lowest amount of loans.

2.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 [283…

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


ORGANIZATION_TYPE
Repayer
Defaulter
Count (log)
Transport: type 3
Industry: type 13
Industry: type 8
Restaurant
Construction
Cleaning
Industry: type 1
Industry: type 3
Realtor
Agriculture
Trade: type 3
Self-employed
Industry: type 4
Security
Trade: type 7
Business Entity Type 3
Transport: type 4
Mobile
Trade: type 1
Industry: type 11
Business Entity Type 2
Postal
Advertising
Business Entity Type 1
Industry: type 7
Housing
Legal Services
Transport: type 2
Other
Telecom
Industry: type 2
Industry: type 6
Emergency
Kindergarten
Trade: type 2
Government
Industry: type 5
Industry: type 9
Electricity
Services
Medicine
Industry: type 10
Hotel
Trade: type 5
School
Religion
Insurance
Culture
XNA
Bank
Military
Police
University
Security Ministries
Trade: type 6
Transport: type 1
Industry: type 12
Trade: type 4
ORGANIZATION_TYPE Defaulter %



Inferences:

1.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.

2.Most of the people application for loan are from Business Entity Type 3.

3.For a very high number of applications, Organization type information is unavailable(XNA).

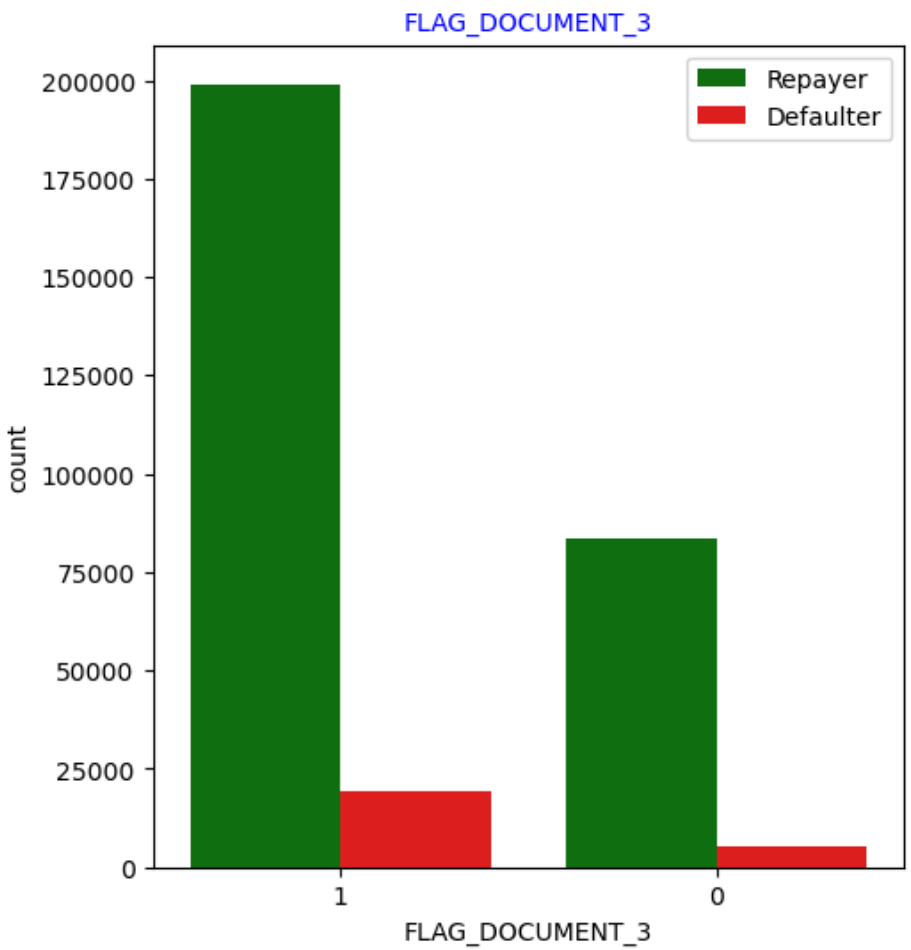
It can be seen that following category of organization type has lesser defaulters thus safer for providing loans:

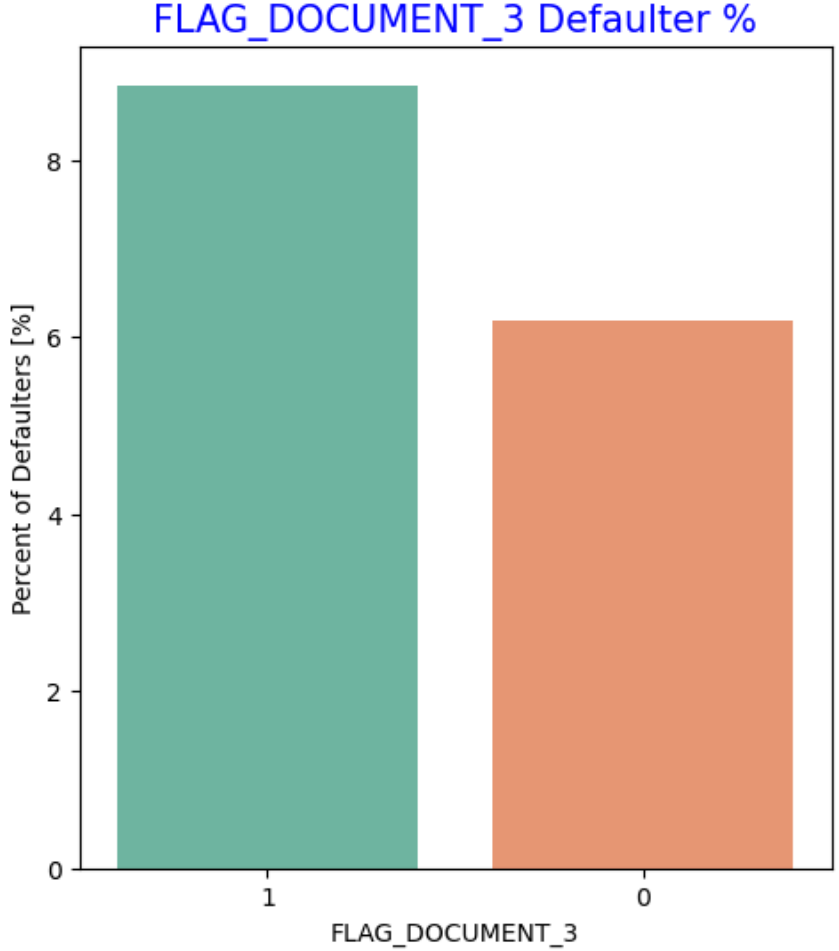
1.Trade Type 4 and 5

2.Industry type 8

In [285…

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

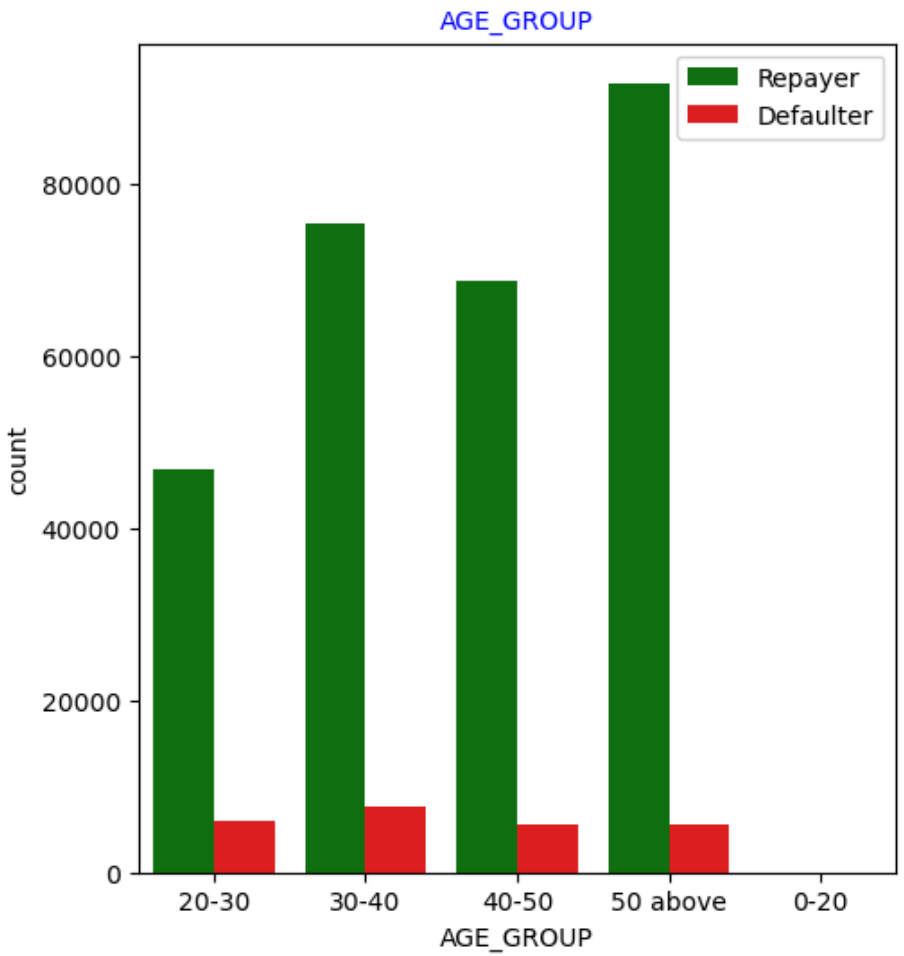


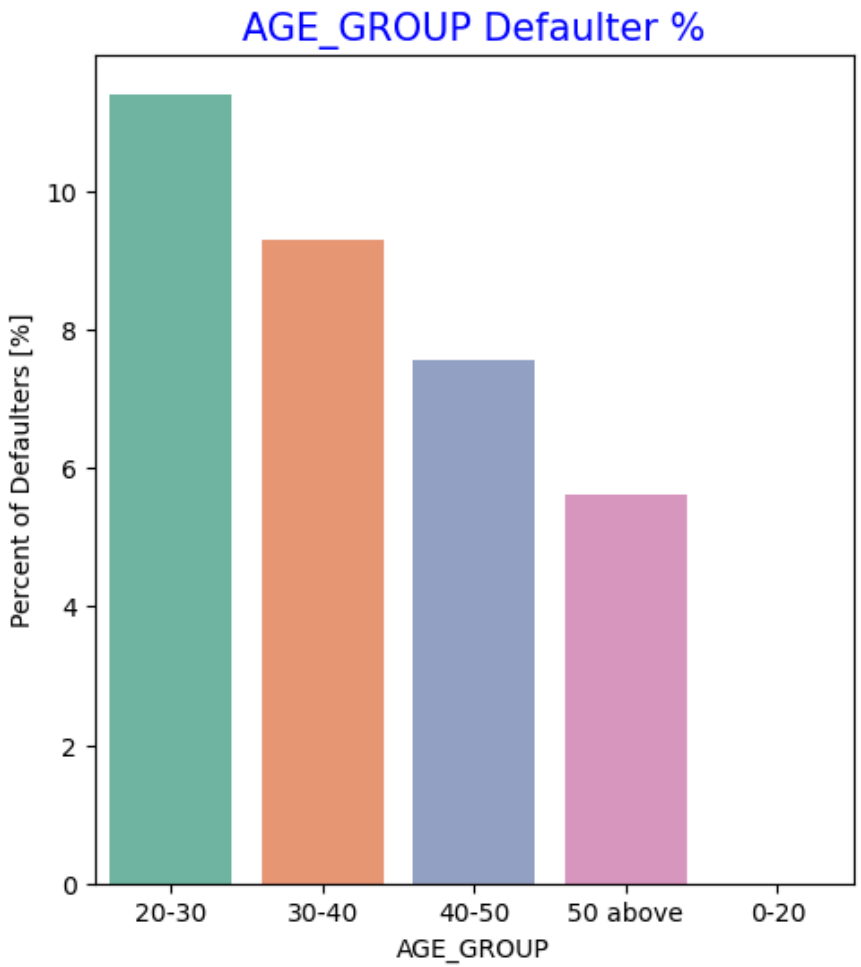


Inferences:

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 [399…
```
# Analyzing Age Group based on loan repayment status
univariate_categorical("AGE_GROUP",False,False,True)
```





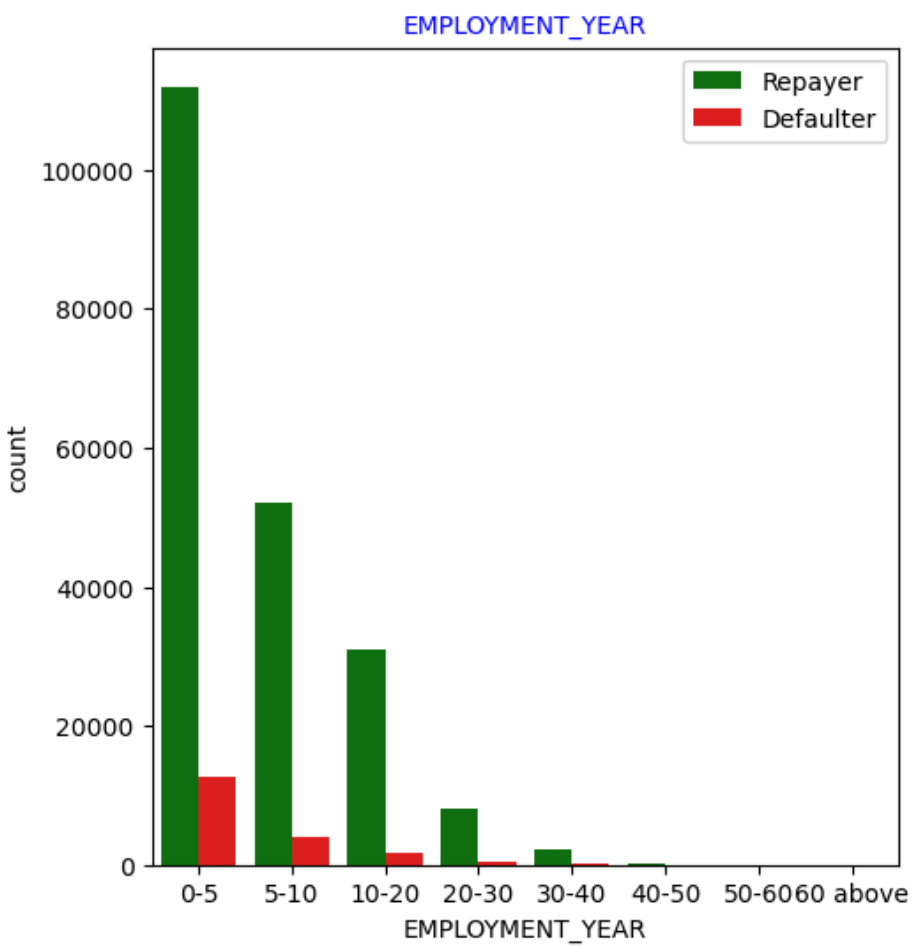
Inferences:

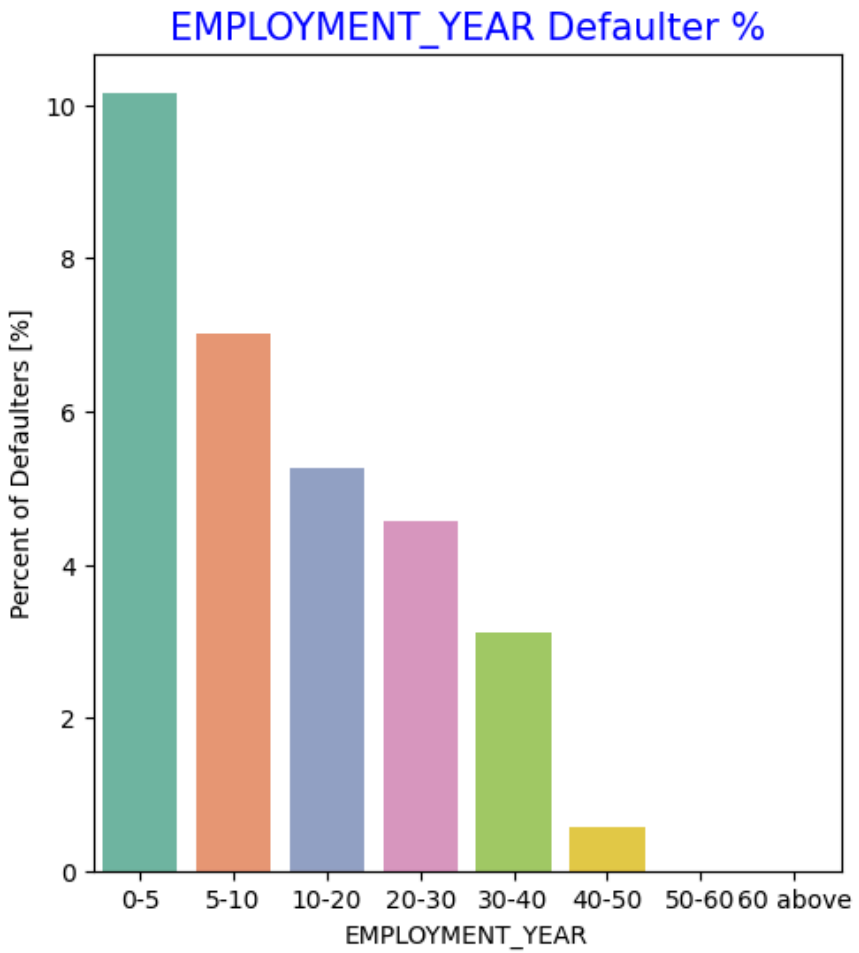
1.People in the age group range 20-40 have higher probability of defaulting

2.People above age of 50 have low probability of defailting.

In [401…

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





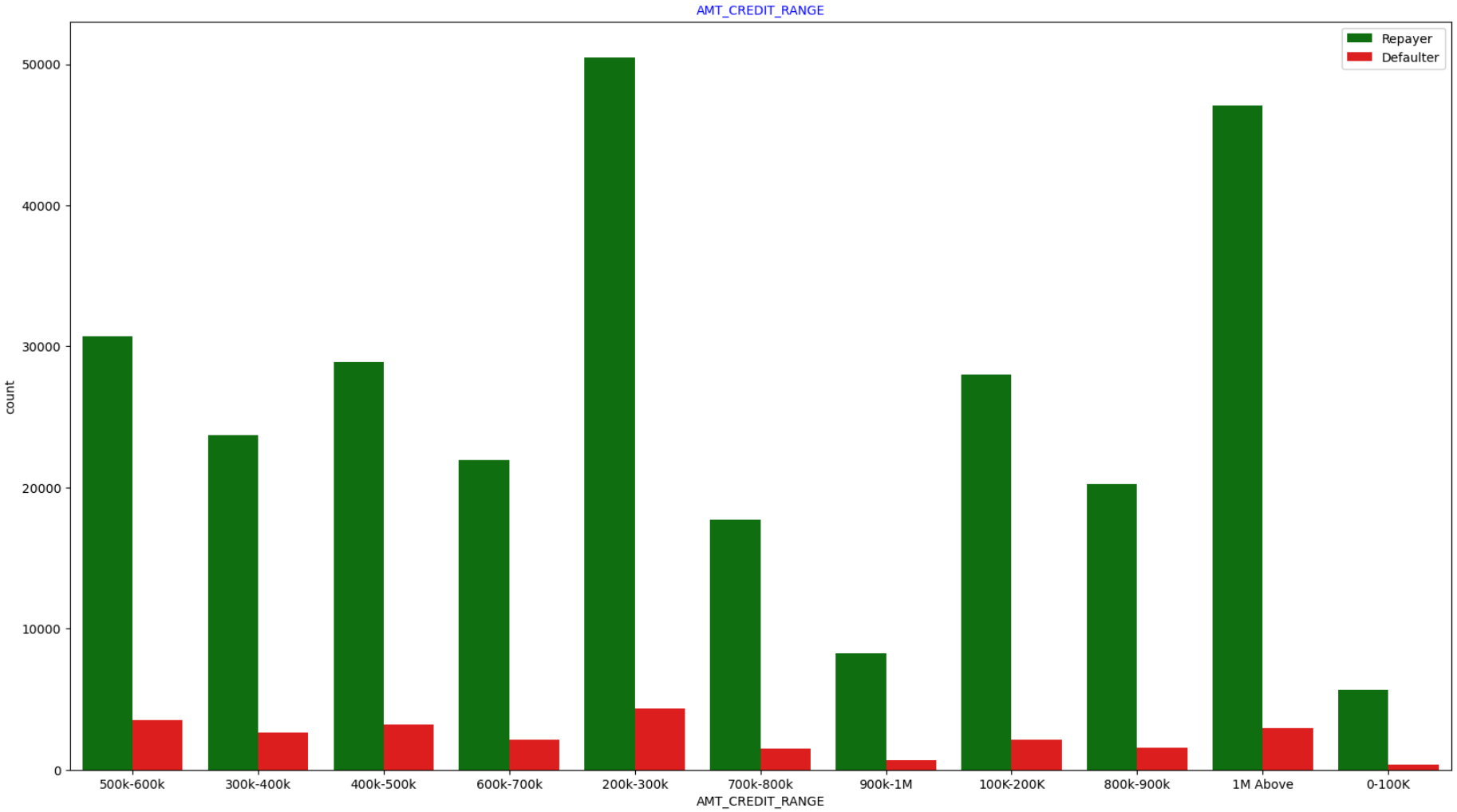
Inferences:

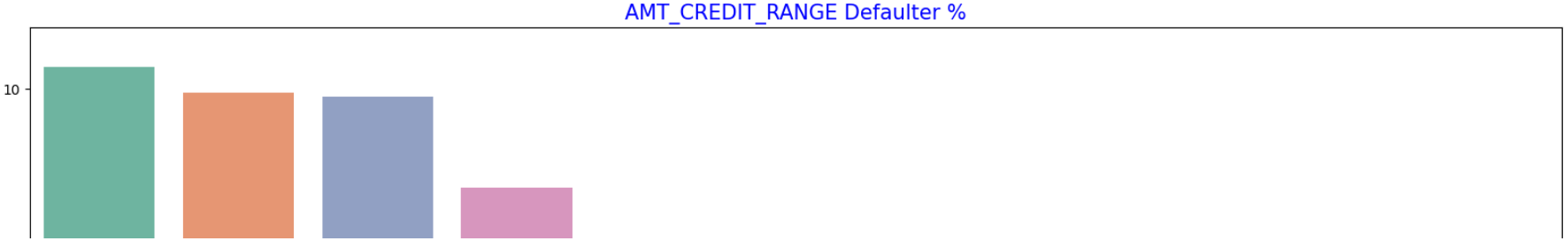
1.Majority of the applicants have been employeed in between 0-5 years. The defaulting rating of this group is also the highest which is 10%.

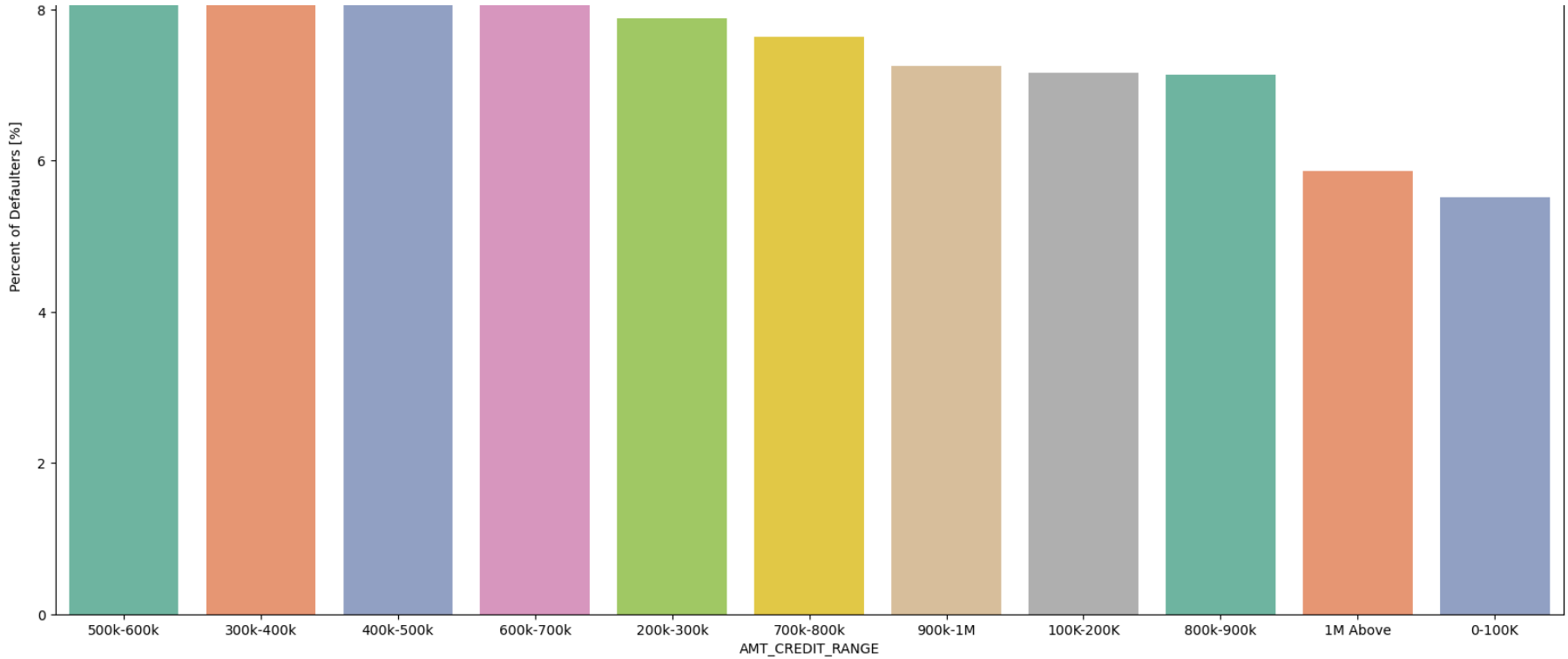
2.With increase of employment year, defaulting rate is gradually decreasing with people having 40+ year experience having less than 1% default rate.

In [293…

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


AMT_CREDIT_RANGE
Repayer
Defaulter
count
50000
40000
30000
20000
10000
0
500k-600k
300k-400k
400k-500k
600k-700k
200k-300k
700k-800k
900k-1M
100K-200K
800k-900k
1M Above
0-100K
AMT_CREDIT_RANGE


AMT_CREDIT_RANGE Defaulter %
10



Inferences:

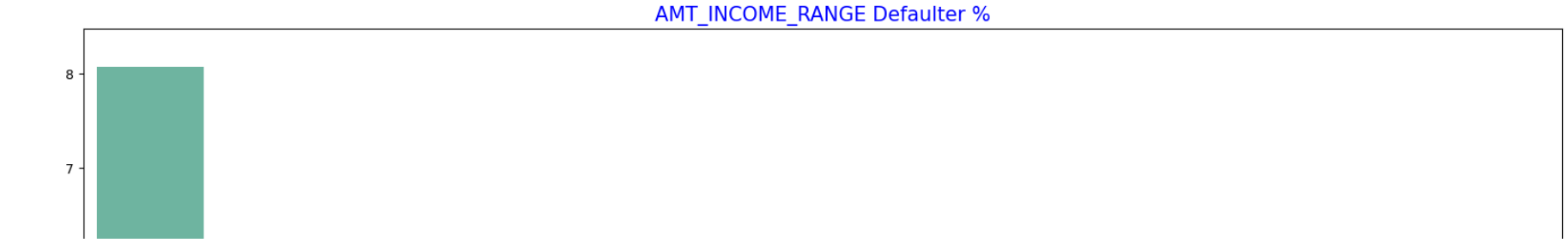
1.More than 80% of the loan provided are for amount less than 900,000.

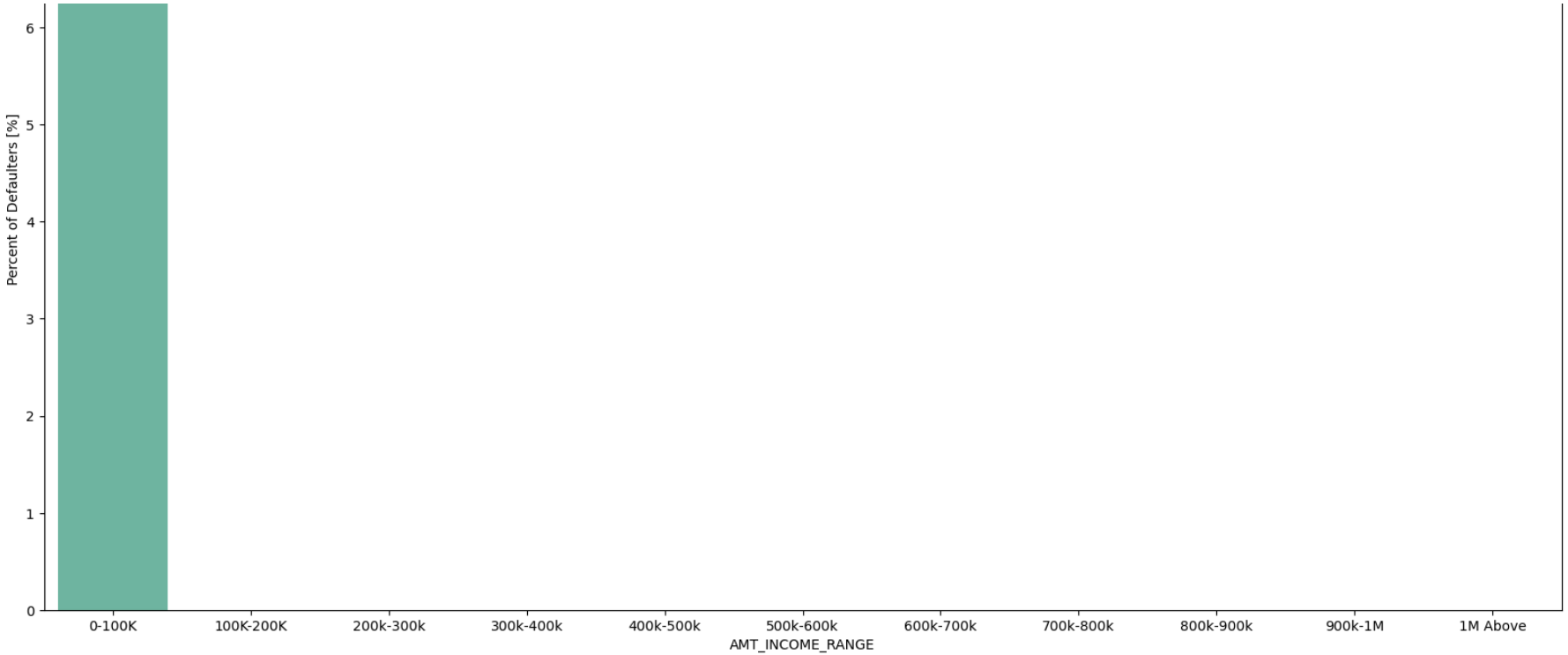
2.People who get loan for 300-600k tend to default more than others.

In [403…

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


AMT_INCOME_RANGE
Repayer
Defaulter
count
250000
200000
150000
100000
50000
0
0-100K
100K-200K
200k-300k
300k-400k
400k-500k
500k-600k
600k-700k
700k-800k
800k-900k
900k-1M
1M Above
AMT_INCOME_RANGE


AMT_INCOME_RANGE Defaulter %
8
7



Inferences:

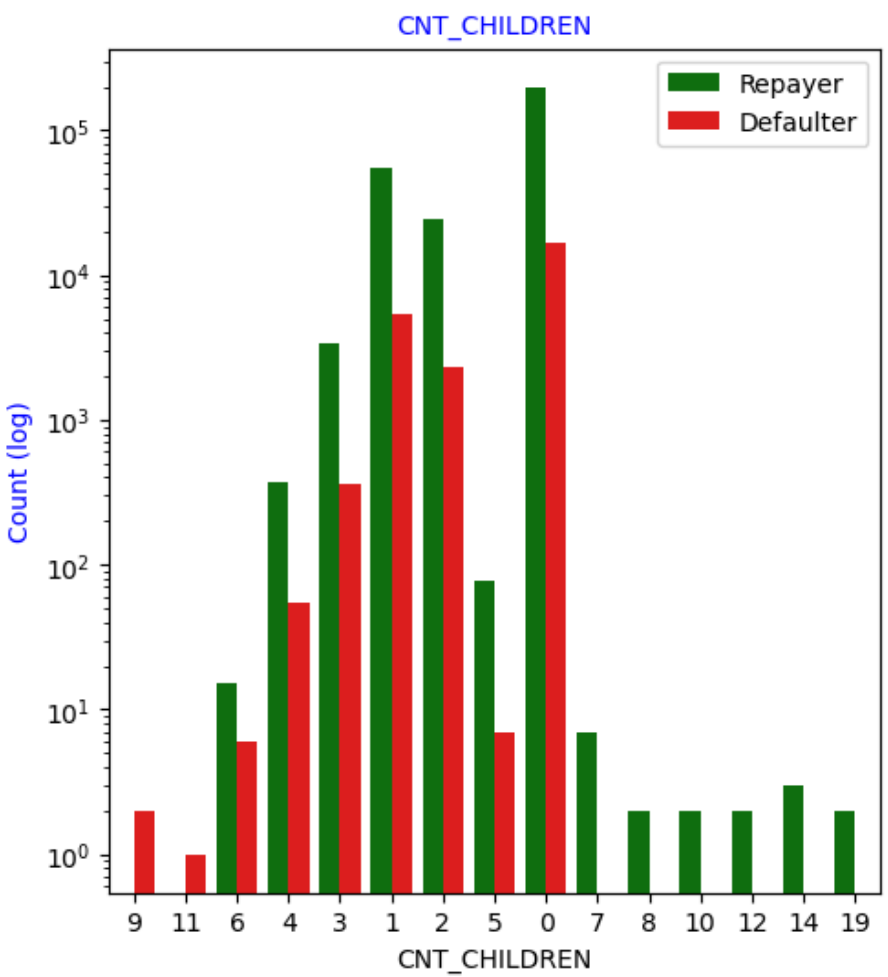
1.90% of the applications have Income total less than 300,000.

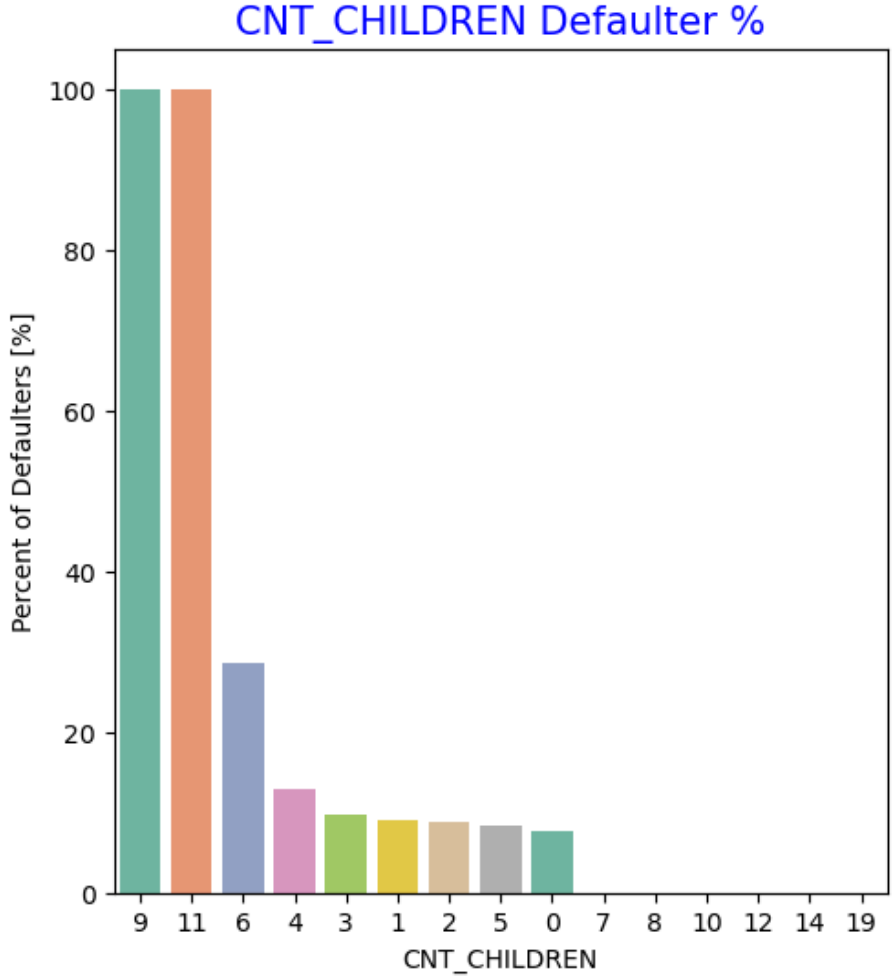
2.Application with Income less than 300,000 has high probability of defaulting.

3.Applicant with Income more than 700,000 are less likely to default.

In [407…

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





Inferences:

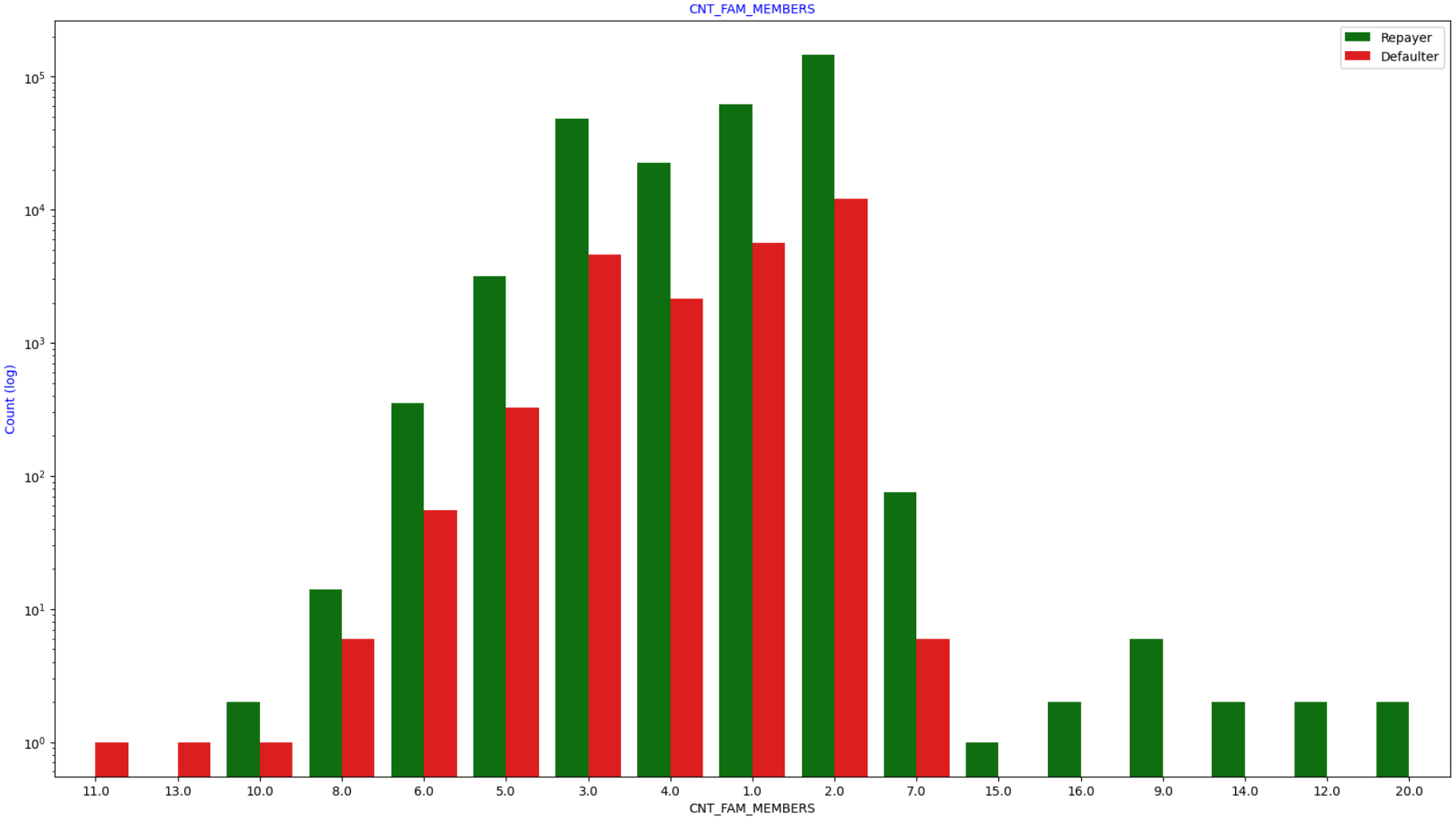
1.Most of the applicants do not have children.

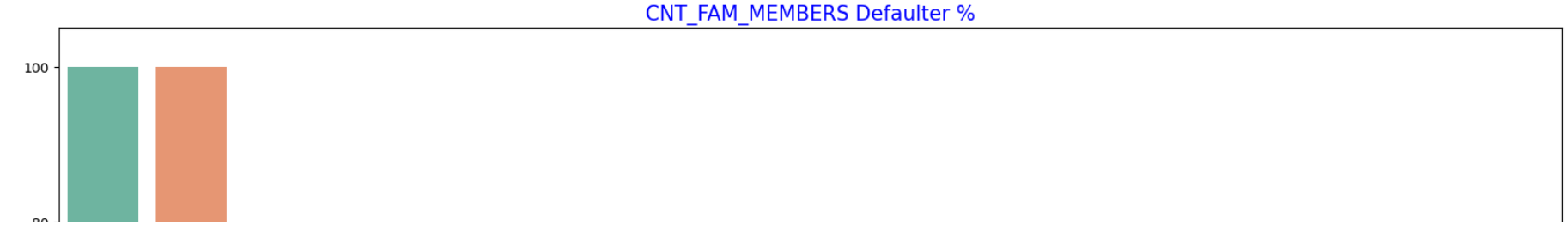
2.Very few clients have more than 3 children.

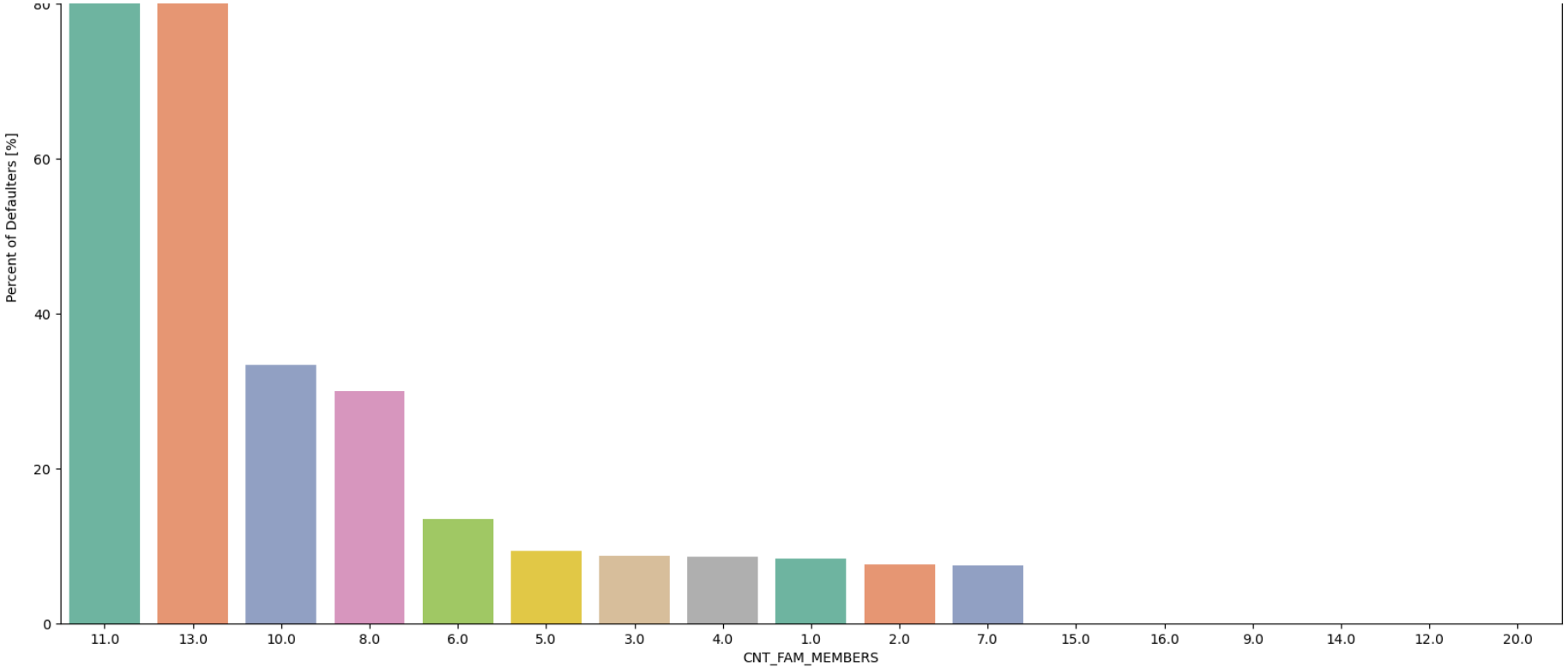
3.Client who have more than 4 children has a very high default rate with child count 9 and 11 showing 100% default rate.

In [301…

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


CNT_FAM_MEMBERS
Repayer
Defaulter
Count (log)
10^5
10^4
10^3
10^2
10^1
10^0
11.0
13.0
10.0
8.0
6.0
5.0
3.0
4.0
1.0
2.0
7.0
15.0
16.0
9.0
14.0
12.0
20.0
CNT_FAM_MEMBERS


CNT_FAM_MEMBERS Defaulter %
100



Inferences:

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

Categorical Bi/Multivariate Analysis

In [303…

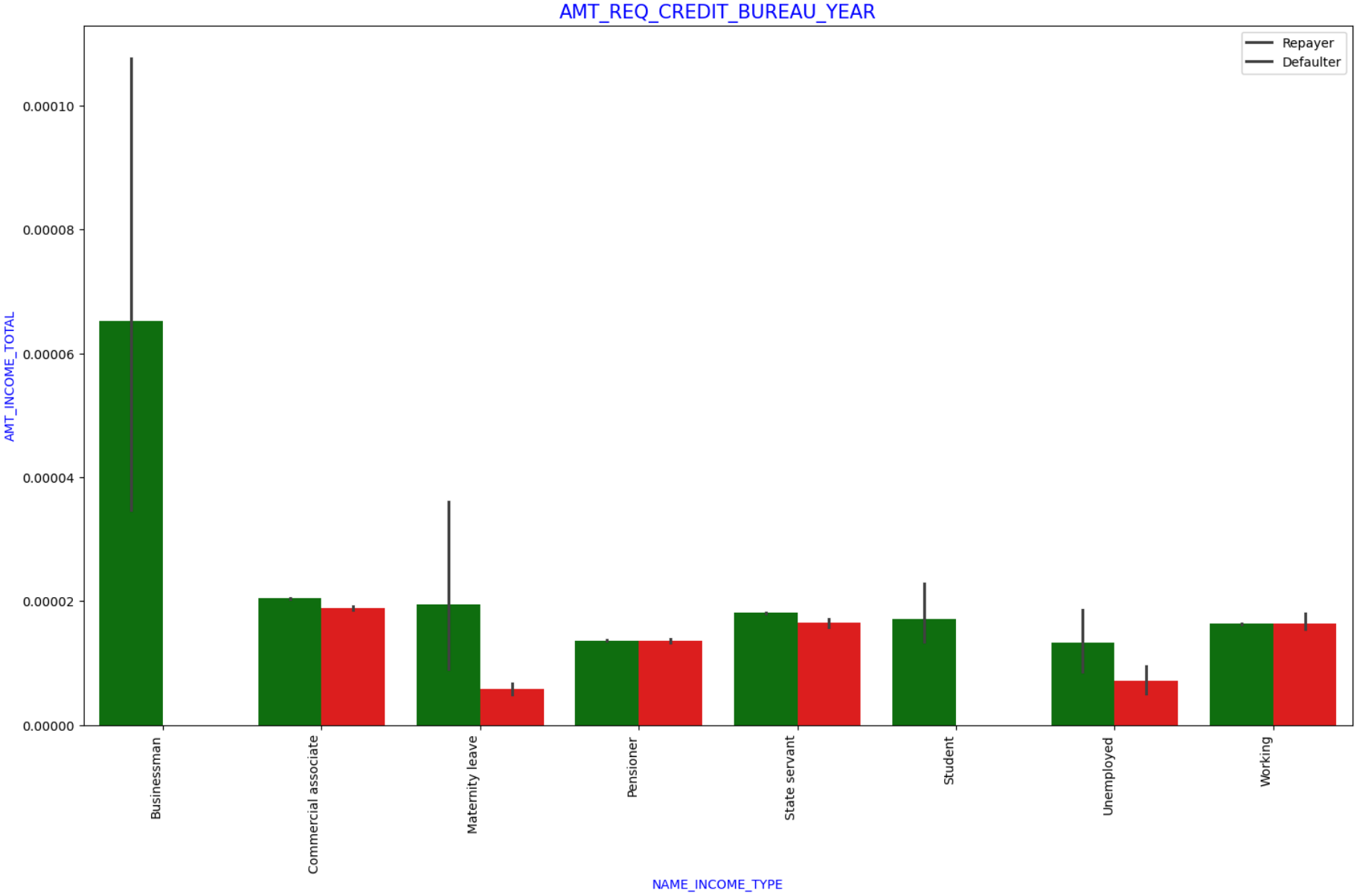
```
_ApplicationDF.groupby('NAME_INCOME_TYPE')['AMT_INCOME_TOTAL'].describe()
```

Out[303…

NAME_INCOME_TYPE	count	mean	std	min	25%	50%	75%	max
Businessman	10.0	0.000065	0.000063	0.000018	0.000023	0.000050	0.000084	0.000225
Commercial associate	71617.0	0.000020	0.000015	0.000003	0.000014	0.000018	0.000023	0.001800
Maternity leave	5.0	0.000014	0.000013	0.000005	0.000007	0.000009	0.000014	0.000036
Pensioner	55362.0	0.000014	0.000008	0.000003	0.000009	0.000012	0.000017	0.000225
State servant	21703.0	0.000018	0.000010	0.000003	0.000011	0.000016	0.000023	0.000315
Student	18.0	0.000017	0.000011	0.000008	0.000011	0.000016	0.000018	0.000056
Unemployed	22.0	0.000011	0.000009	0.000003	0.000005	0.000008	0.000014	0.000034
Working	158774.0	0.000016	0.000031	0.000003	0.000011	0.000014	0.000020	0.011700

In [305…

```
# Income type vs Income Amount Range
bivariate_bar("NAME_INCOME_TYPE","AMT_INCOME_TOTAL",_ApplicationDF,"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 [307…
```
_ApplicationDF.columns
```

Out[307…
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH',
       'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START',
       'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION',
       'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION',
       'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY',
       'LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_TYPE',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_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'],
      dtype='object')
```

In [309…
```
# Bifurcating the applicationDF dataframe based on Target value 0 and 1 for correlation and other analysis
cols_for_correlation = ['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY',
                        'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE',
                        'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
                        'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED',
                        'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
                        'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
                        'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION',
                        'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_TYPE',
                        'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_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']


Repayer_df = _ApplicationDF.loc[_ApplicationDF['TARGET']==0, cols_for_correlation] # Repayers
Defaulter_df = _ApplicationDF.loc[_ApplicationDF['TARGET']==1, cols_for_correlation] # Defaulters
```

In [ ]:
```
# 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),k=1).astype(np.bool))
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)
```

In [ ]:
```
fig = plt.figure(figsize=(12,12))
ax = sns.heatmap(Repayer_df.corr(), cmap="RdYlGn",annot=False,linewidth =1)
```

In [ ]:
```

```