=== Real Time Project on EDA ===

Bank Loan Default Risk Analysis

BUSINESS UNDERSTANDING

The loan providing companies find it hard to give loans to the people due to their insufficient or non-existent credit history. Because of that, some consumers use it as their advantage by becoming a defaulter. Suppose you work for a consumer finance company which specialises in lending various types of loans to urban customers. You have to use EDA to analyse the patterns present in the data. This will ensure that the applicants capable of repaying the loan are not rejected. When the company receives a loan application, the company has to decide for loan approval based on the applicant's profile. Two types of risks are associated with the bank's decision:

If the applicant is likely to repay the loan, then not approving the loan results in a loss of business to the company If the applicant is not likely to repay the loan, i.e. he/she is likely to default, then approving the loan may lead to a financial loss for the company.

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.

In [7]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [8]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from matplotlib import style as style
```

```
import itertools
%matplotlib inline
```

In [9]:
```
application=pd.read_csv(r"C:\Users\Jan Saida\OneDrive\Documents\application_data.csv")
```

In [10]:
```
application
```

Out[10]:

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

307511 rows × 122 columns

In [11]:
```
previous_DF=pd.read_csv(r"C:\Users\Jan Saida\OneDrive\Documents\previous_application.csv")
```

In [12]:
```
previous_DF
```

Out[12]:

	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	
...	...	...	...	...	...	...	...	
1048570	2230795	255000	Consumer loans	50978.475	210960.0	189864.0	21096.0	
1048571	1823303	158245	Cash loans	NaN	0.0	0.0	NaN	
1048572	1730537	429268	Consumer loans	5793.120	38070.0	36526.5	3807.0	
1048573	2100360	389043	Consumer loans	4775.355	35356.5	30109.5	7110.0	
1048574	1283481	250078	Consumer loans	6617.925	44986.5	23229.0	22500.0	

1048575 rows × 37 columns

In [13]:
```
application
```

Out[13]:

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

307511 rows × 122 columns

In [14]:
```
application.head()
```

Out[14]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TC
0	100002	1	Cash loans	M	N	Y	0	2025
1	100003	0	Cash loans	F	N	N	0	2700
2	100004	0	Revolving loans	M	Y	Y	0	675
3	100006	0	Cash loans	F	N	Y	0	1350
4	100007	0	Cash loans	M	N	Y	0	1215

5 rows × 122 columns

In [15]:
```
application.tail()
```

Out[15]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCO
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	

5 rows × 122 columns

In [16]:
```
application.shape
```

Out[16]:
```
(307511, 122)
```

In [17]:
```
application.describe()
```

Out[17]:

	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 [18]:
```
application.columns
```

Out[18]:
```
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 [19]:
```
application.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 [20]:
```
previous_DF
```

Out[20]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	A
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	
...	...	...	...	...	...	...	...	
1048570	2230795	255000	Consumer loans	50978.475	210960.0	189864.0	21096.0	
1048571	1823303	158245	Cash loans	NaN	0.0	0.0	NaN	
1048572	1730537	429268	Consumer loans	5793.120	38070.0	36526.5	3807.0	
1048573	2100360	389043	Consumer loans	4775.355	35356.5	30109.5	7110.0	
1048574	1283481	250078	Consumer loans	6617.925	44986.5	23229.0	22500.0	

1048575 rows × 37 columns

In [21]:
```
previous_DF.head()
```

Out[21]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GO
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	

5 rows × 37 columns

In [22]:
```
previous_DF.tail()
```

Out[22]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	A
1048570	2230795	255000	Consumer loans	50978.475	210960.0	189864.0	21096.0	
1048571	1823303	158245	Cash loans	NaN	0.0	0.0	NaN	
1048572	1730537	429268	Consumer loans	5793.120	38070.0	36526.5	3807.0	
1048573	2100360	389043	Consumer loans	4775.355	35356.5	30109.5	7110.0	
1048574	1283481	250078	Consumer loans	6617.925	44986.5	23229.0	22500.0	

5 rows × 37 columns

In [23]:
```
previous_DF.shape
```

Out[23]:
```
(1048575, 37)
```

In [24]:
```
previous_DF.describe()
```

Out[24]:

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	HOUR
count	1.048575e+06	1.048575e+06	815566.000000	1.048575e+06	1.048575e+06	4.891790e+05	8.076100e+05	
mean	1.922775e+06	2.784367e+05	15891.265151	1.742698e+05	1.950000e+05	6.700778e+03	2.262892e+05	
std	5.329366e+05	1.028569e+05	14745.557438	2.910789e+05	3.169407e+05	2.078570e+04	3.134490e+05	
min	1.000001e+06	1.000010e+05	0.000000	0.000000e+00	0.000000e+00	-9.000000e-01	0.000000e+00	
25%	1.460642e+06	1.893860e+05	6301.350000	1.890000e+04	2.427750e+04	0.000000e+00	5.058000e+04	
50%	1.923419e+06	2.788100e+05	11250.000000	7.081650e+04	8.025300e+04	1.624500e+03	1.115116e+05	
75%	2.384448e+06	3.677445e+05	20523.003750	1.800000e+05	2.152395e+05	7.749000e+03	2.295000e+05	
max	2.845382e+06	4.562550e+05	418058.145000	6.905160e+06	6.905160e+06	2.150100e+06	6.905160e+06	

8 rows × 21 columns

In [25]:
```
previous_DF.columns
```

Out[25]:
```
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 [26]:
```
previous_DF.info(verbose=True)
```

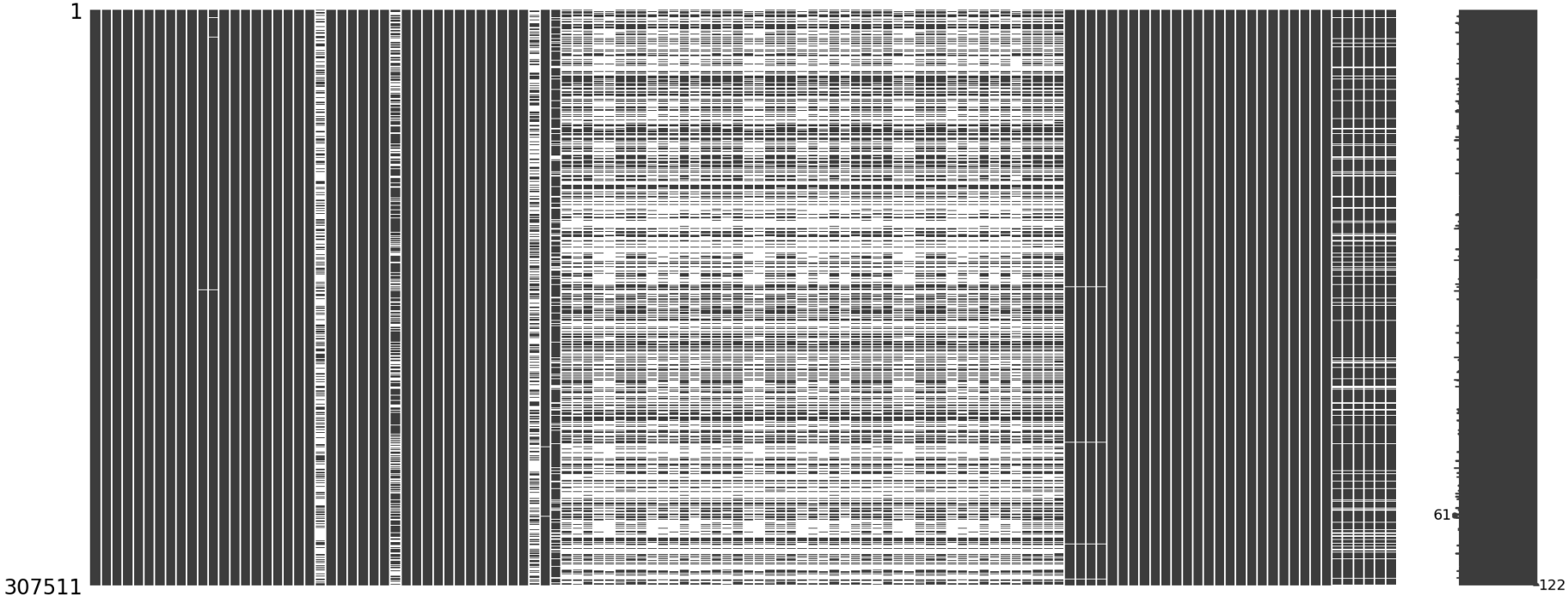
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   SK_ID_PREV                   1048575 non-null  int64
 1   SK_ID_CURR                   1048575 non-null  int64
 2   NAME_CONTRACT_TYPE           1048575 non-null  object
 3   AMT_ANNUITY                  815566 non-null   float64
 4   AMT_APPLICATION              1048575 non-null  float64
 5   AMT_CREDIT                   1048575 non-null  float64
 6   AMT_DOWN_PAYMENT             489179 non-null   float64
 7   AMT_GOODS_PRICE              807610 non-null   float64
 8   WEEKDAY_APPR_PROCESS_START   1048575 non-null  object
 9   HOUR_APPR_PROCESS_START      1048575 non-null  int64
 10  FLAG_LAST_APPL_PER_CONTRACT  1048575 non-null  object
 11  NFLAG_LAST_APPL_IN_DAY       1048575 non-null  int64
 12  RATE_DOWN_PAYMENT            489179 non-null   float64
 13  RATE_INTEREST_PRIMARY        3721 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     3721 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1048575 non-null  object
 16  NAME_CONTRACT_STATUS         1048575 non-null  object
 17  DAYS_DECISION                1048575 non-null  int64
 18  NAME_PAYMENT_TYPE            1048575 non-null  object
 19  CODE_REJECT_REASON           1048575 non-null  object
 20  NAME_TYPE_SUITE              533435 non-null   object
 21  NAME_CLIENT_TYPE             1048575 non-null  object
 22  NAME_GOODS_CATEGORY          1048575 non-null  object
 23  NAME_PORTFOLIO               1048575 non-null  object
 24  NAME_PRODUCT_TYPE            1048575 non-null  object
 25  CHANNEL_TYPE                 1048575 non-null  object
 26  SELLERPLACE_AREA             1048575 non-null  int64
 27  NAME_SELLER_INDUSTRY         1048575 non-null  object
 28  CNT_PAYMENT                  815569 non-null   float64
 29  NAME_YIELD_GROUP             1048575 non-null  object
 30  PRODUCT_COMBINATION          1048351 non-null  object
 31  DAYS_FIRST_DRAWING           627867 non-null   float64
 32  DAYS_FIRST_DUE               627867 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    627867 non-null   float64
 34  DAYS_LAST_DUE                627867 non-null   float64
 35  DAYS_TERMINATION             627867 non-null   float64
```

```
 36  NFLAG_INSURED_ON_APPROVAL    627867 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 296.0+ MB
```

DATA CLEANING

In [28]:
```
import missingno as mn
mn.matrix(application)
```

Out[28]: <Axes: >

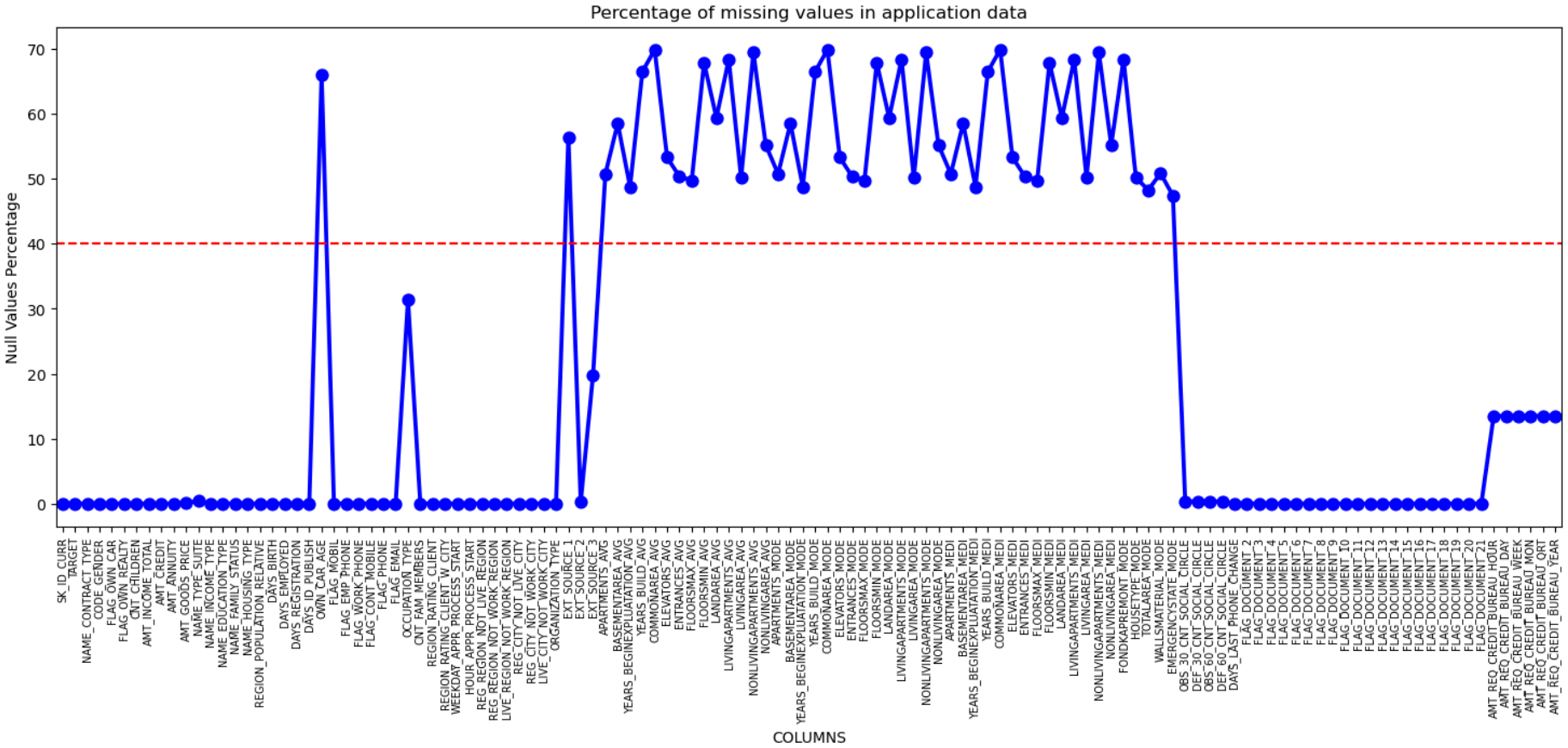


There are many columns in applicationDF dataframe where missing value is more than 40%. Let's plot the columns vs missing value % with 40% being the cut-off marks

In [30]:
```
null_application=pd.DataFrame((application.isnull().sum())*100/application.shape[0]).reset_index()
null_application.columns=['Column Name','Null Values Percentage']
fig=plt.figure(figsize=(18,6))
ax=sns.pointplot(x='Column Name',y='Null Values Percentage',data=null_application,color='blue')
```

```
plt.xticks(rotation=90,fontsize=7)
ax.axhline(40,ls='--',color='red')
plt.title('Percentage of missing values in application data')
plt.ylabel('Null Values Percentage')
plt.xlabel('COLUMNS')
```

Out[30]: Text(0.5, 0, 'COLUMNS')



From the plot we can see the columns in which percentage of null values more than 40% are marked above the red line and 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 [32]:
```
# more than or equal to 40% empty rows columns
```

```
nullcol_40_application=null_application[null_application['Null Values Percentage']>=40]
nullcol_40_application
```

Out[32]:

	Column Name	Null Values Percentage
21	OWN_CAR_AGE	65.990810
41	EXT_SOURCE_1	56.381073
44	APARTMENTS_AVG	50.749729
45	BASEMENTAREA_AVG	58.515956
46	YEARS_BEGINEXPLUATATION_AVG	48.781019
47	YEARS_BUILD_AVG	66.497784
48	COMMONAREA_AVG	69.872297
49	ELEVATORS_AVG	53.295980
50	ENTRANCES_AVG	50.348768
51	FLOORSMAX_AVG	49.760822
52	FLOORSMIN_AVG	67.848630
53	LANDAREA_AVG	59.376738
54	LIVINGAPARTMENTS_AVG	68.354953
55	LIVINGAREA_AVG	50.193326
56	NONLIVINGAPARTMENTS_AVG	69.432963
57	NONLIVINGAREA_AVG	55.179164
58	APARTMENTS_MODE	50.749729
59	BASEMENTAREA_MODE	58.515956
60	YEARS_BEGINEXPLUATATION_MODE	48.781019
61	YEARS_BUILD_MODE	66.497784
62	COMMONAREA_MODE	69.872297
63	ELEVATORS_MODE	53.295980

	Column Name	Null Values Percentage
64	ENTRANCES_MODE	50.348768
65	FLOORSMAX_MODE	49.760822
66	FLOORSMIN_MODE	67.848630
67	LANDAREA_MODE	59.376738
68	LIVINGAPARTMENTS_MODE	68.354953
69	LIVINGAREA_MODE	50.193326
70	NONLIVINGAPARTMENTS_MODE	69.432963
71	NONLIVINGAREA_MODE	55.179164
72	APARTMENTS_MEDI	50.749729
73	BASEMENTAREA_MEDI	58.515956
74	YEARS_BEGINEXPLUATATION_MEDI	48.781019
75	YEARS_BUILD_MEDI	66.497784
76	COMMONAREA_MEDI	69.872297
77	ELEVATORS_MEDI	53.295980
78	ENTRANCES_MEDI	50.348768
79	FLOORSMAX_MEDI	49.760822
80	FLOORSMIN_MEDI	67.848630
81	LANDAREA_MEDI	59.376738
82	LIVINGAPARTMENTS_MEDI	68.354953
83	LIVINGAREA_MEDI	50.193326
84	NONLIVINGAPARTMENTS_MEDI	69.432963
85	NONLIVINGAREA_MEDI	55.179164

	Column Name	Null Values Percentage
86	FONDKAPREMONT_MODE	68.386172
87	HOUSETYPE_MODE	50.176091
88	TOTALAREA_MODE	48.268517
89	WALLSMATERIAL_MODE	50.840783
90	EMERGENCYSTATE_MODE	47.398304

In [33]:
```
# How many columns have more than or euqal to 40% null values ?

len(nullcol_40_application)
```

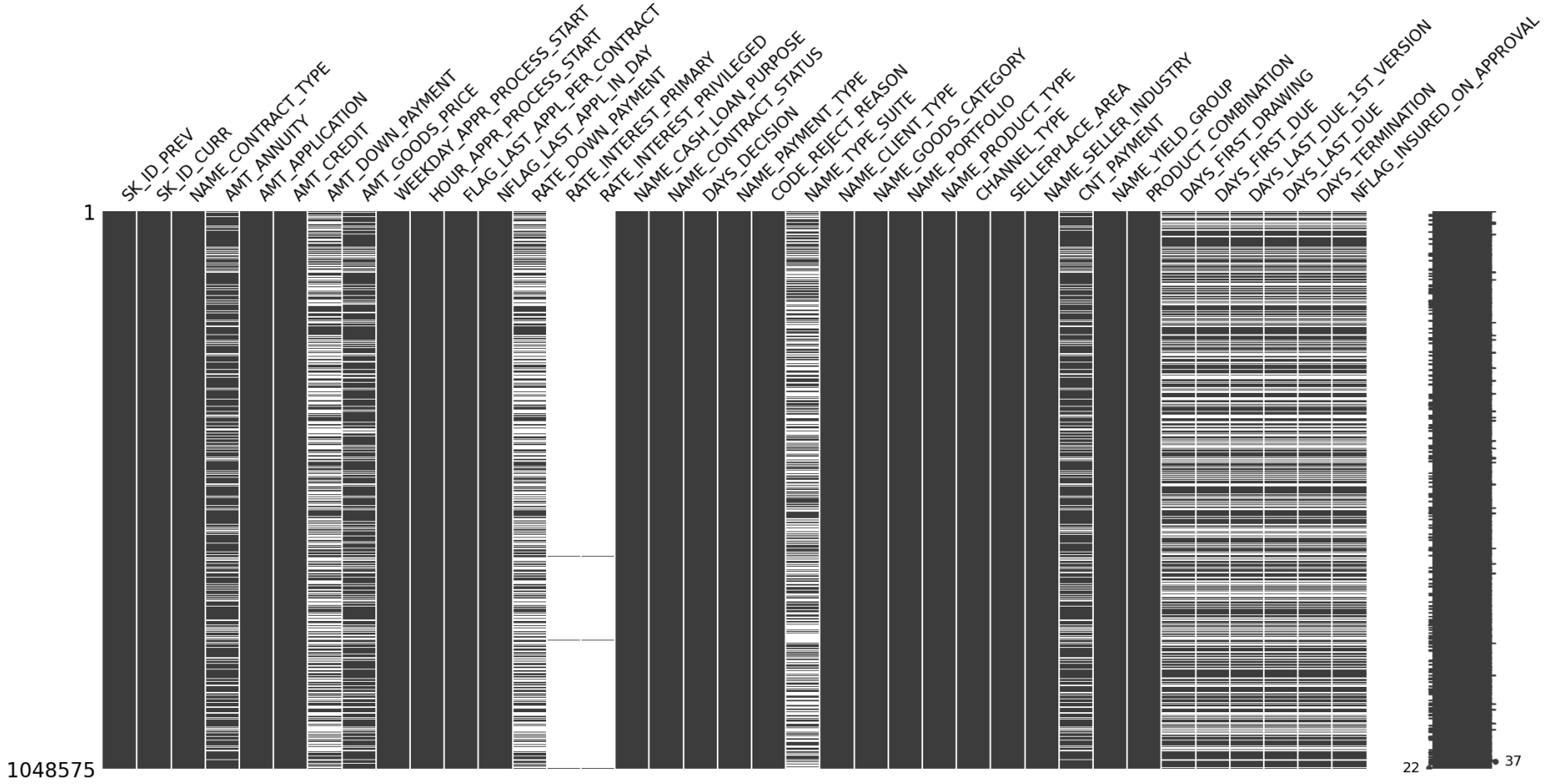
Out[33]: 49

Total of 49 columns are there which have more than 40% null values.Seems like most of the columns with high missing values are related to different area sizes on apartment owned/rented by the loan applicant

PreviousDF MISSING VALUES

In [36]:
```
mn.matrix(previous_DF)
```

Out[36]: <Axes: >



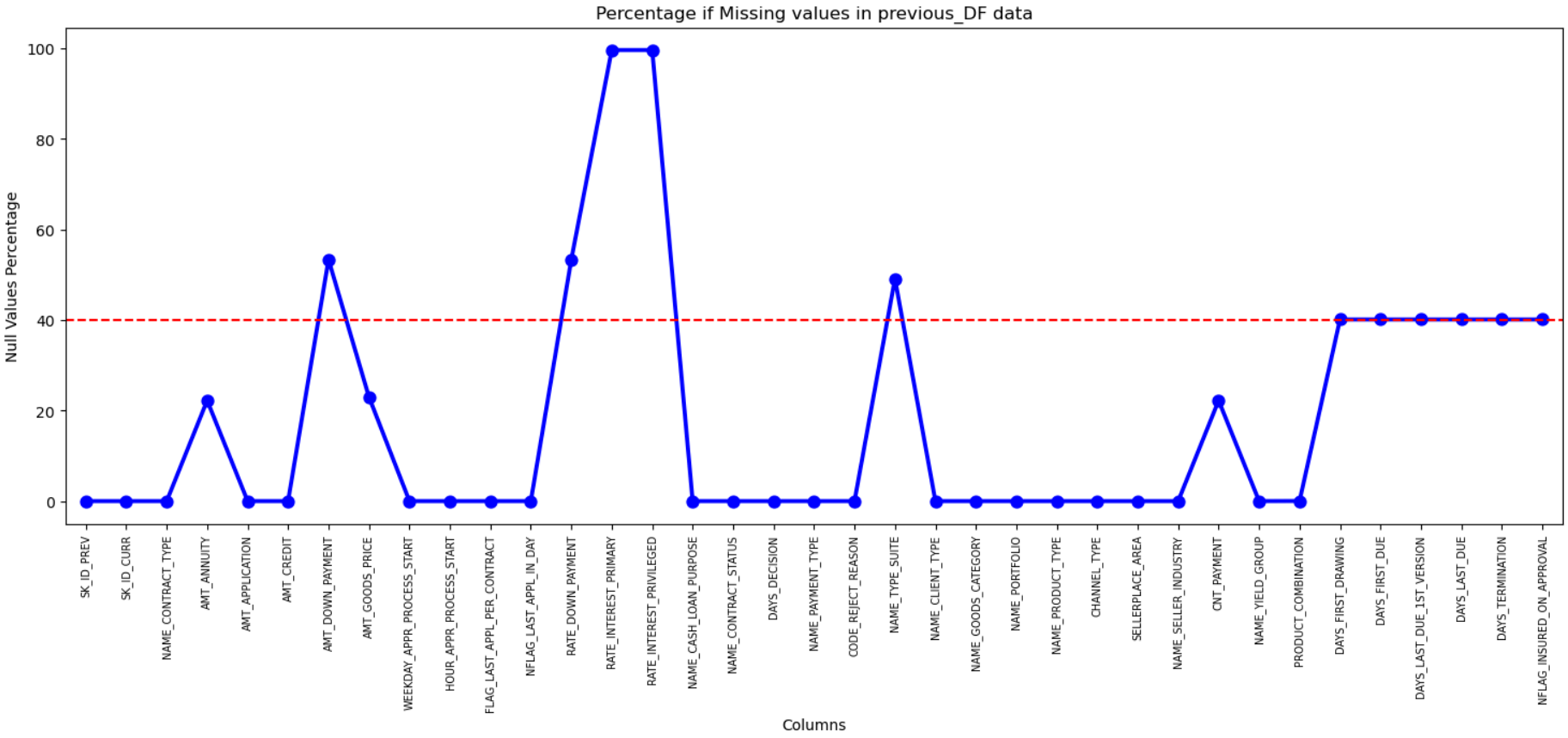
In [37]:
```
# checking the null value % of each column in previousDF dataframe

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

Out[37]:
```
SK_ID_PREV                    0.00
SK_ID_CURR                    0.00
NAME_CONTRACT_TYPE            0.00
AMT_ANNUITY                  22.22
AMT_APPLICATION               0.00
AMT_CREDIT                    0.00
AMT_DOWN_PAYMENT             53.35
AMT_GOODS_PRICE              22.98
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            53.35
RATE_INTEREST_PRIMARY        99.65
RATE_INTEREST_PRIVILEGED     99.65
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              49.13
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                  22.22
NAME_YIELD_GROUP              0.00
PRODUCT_COMBINATION           0.02
DAYS_FIRST_DRAWING           40.12
DAYS_FIRST_DUE               40.12
DAYS_LAST_DUE_1ST_VERSION    40.12
DAYS_LAST_DUE                40.12
DAYS_TERMINATION             40.12
NFLAG_INSURED_ON_APPROVAL    40.12
dtype: float64
```

There are many columns in previousDF dataframe where missing value is more than 40%. Let's plot the columns vs missing value % with 40% being the cut-off marks

In [39]:
```
null_previous_DF=pd.DataFrame((previous_DF.isnull().sum())*100/previous_DF.shape[0]).reset_index()
null_previous_DF.columns=['Column Name','Null Values Percentage']
fig=plt.figure(figsize=(18,6))
ax=sns.pointplot(x='Column Name',y='Null Values Percentage',data=null_previous_DF,color='blue')
plt.xticks(rotation=90,fontsize=7)
ax.axhline(40,ls='--',color='red')
plt.title('Percentage if Missing values in previous_DF data')
plt.ylabel('Null Values Percentage')
plt.xlabel('Columns')
```

Out[39]:
```
Text(0.5, 0, 'Columns')
```



From the plot we can see the columns in which percentage of null values more than 40% are marked above the red line and 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 [41]:
```
# more than or equal to 40% empty rows columns

nullcol_40_previous=null_previous_DF[null_previous_DF['Null Values Percentage']>=40]
nullcol_40_previous
```

Out[41]:

	Column Name	Null Values Percentage
6	AMT_DOWN_PAYMENT	53.348211
12	RATE_DOWN_PAYMENT	53.348211
13	RATE_INTEREST_PRIMARY	99.645137
14	RATE_INTEREST_PRIVILEGED	99.645137
20	NAME_TYPE_SUITE	49.127626
31	DAYS_FIRST_DRAWING	40.121880
32	DAYS_FIRST_DUE	40.121880
33	DAYS_LAST_DUE_1ST_VERSION	40.121880
34	DAYS_LAST_DUE	40.121880
35	DAYS_TERMINATION	40.121880
36	NFLAG_INSURED_ON_APPROVAL	40.121880

In [42]:
```
# How many columns have more than or euqal to 40% null values ?

len(nullcol_40_previous)
```

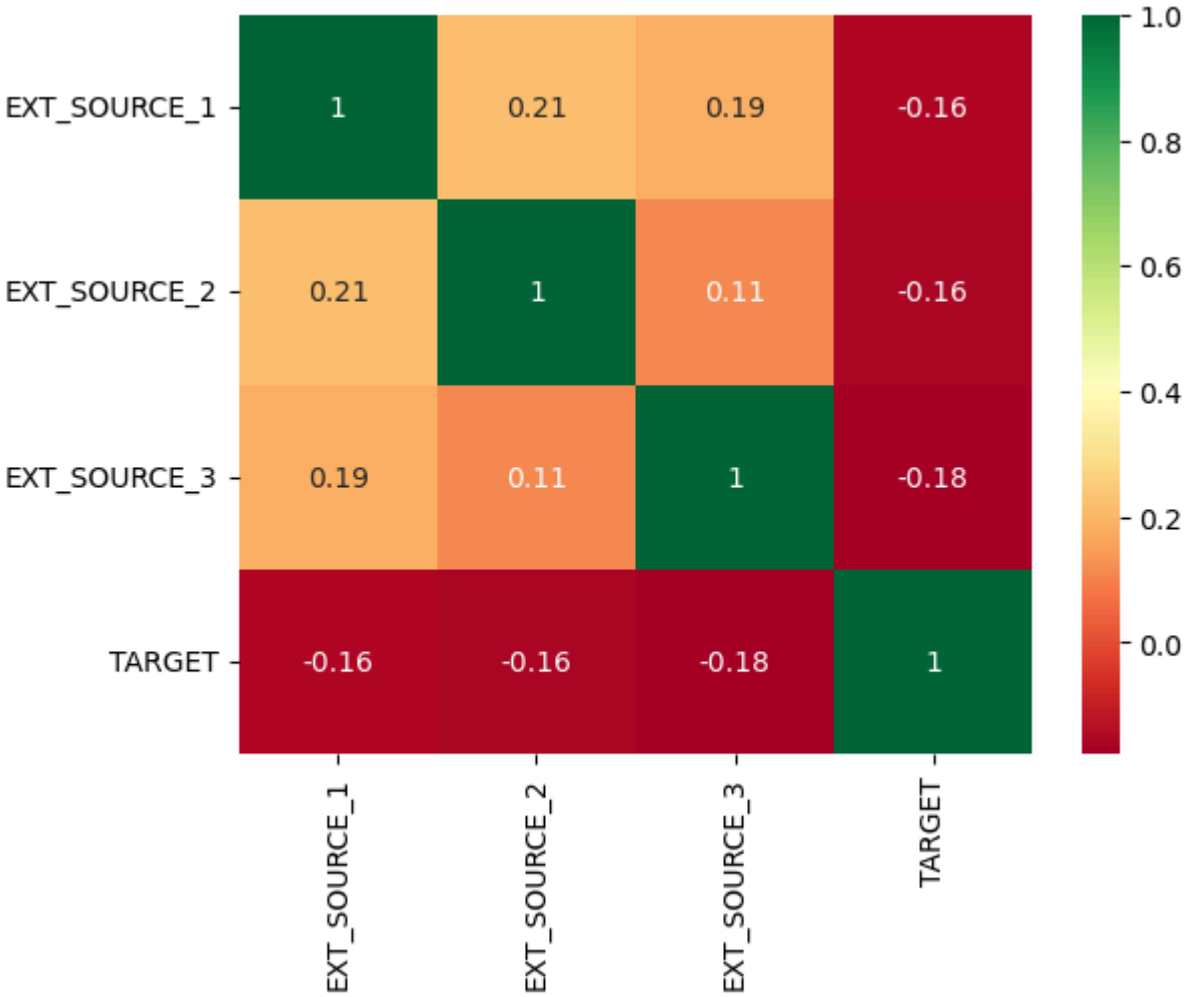
Out[42]: 11

Total of 11 columns are there which have more than 40% null values. These columns can be deleted. Before deleting these columns, let's review if there are more columns which can be dropped or not

Analyze & Delete Unnecessary Columns in applicationDF

In [45]:
```
# Checking correlation of EXT_SOURCE_X columns vs TARGET column

source=application[['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')
```



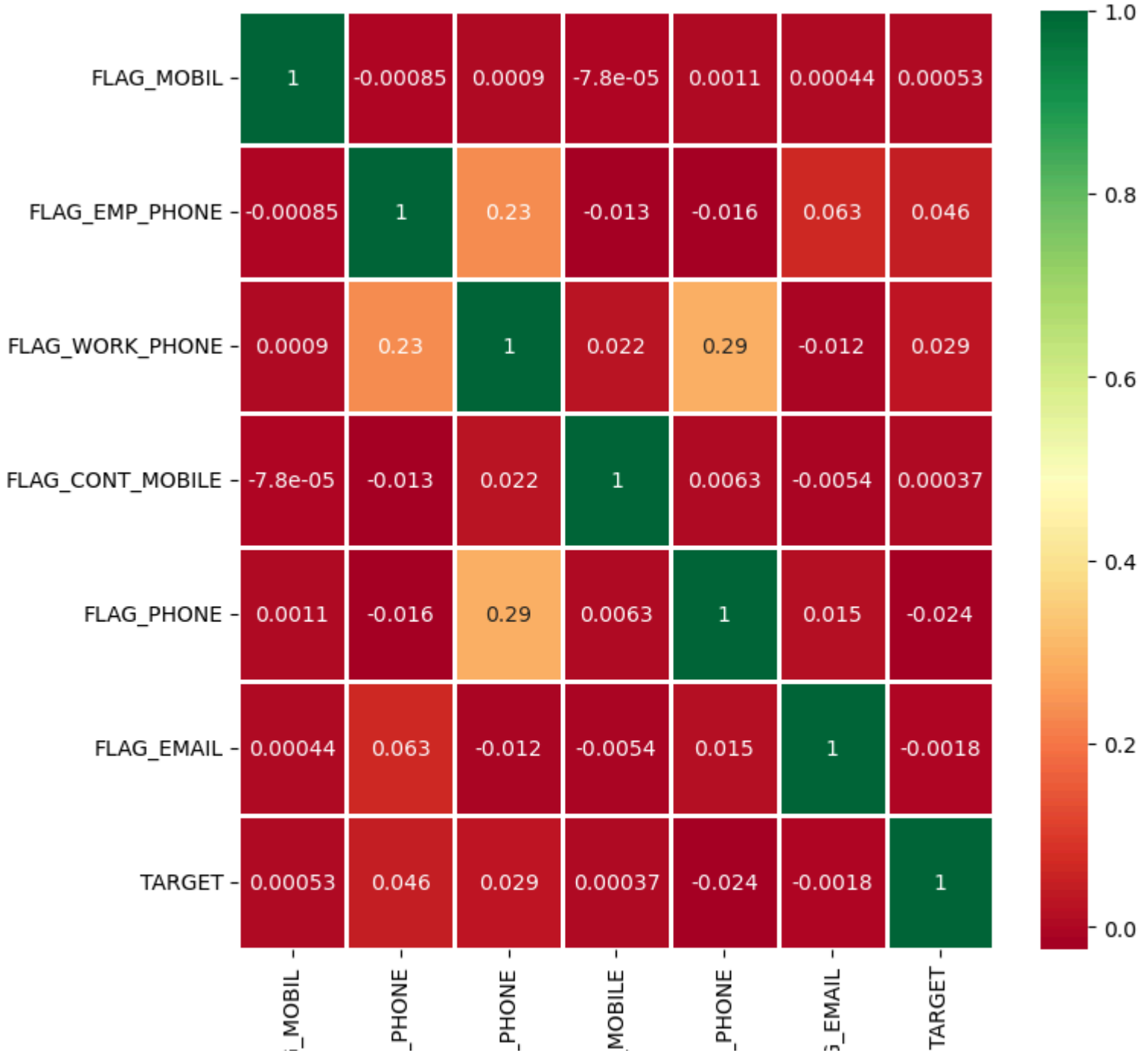
Based on the above Heatmap, we can see there is almost no correlation between EXT_SOURCE_X columns and target column, thus we can drop these columns. EXT_SOURCE_1 has 56% null values, where as EXT_SOURCE_3 has close to 20% null values

In [47]:
```
# create a list of columns that needs to be dropped including the columns with >40% null values

Unwanted_application=nullcol_40_application['Column Name'].tolist()+['EXT_SOURCE_2','EXT_SOURCE_3']

#  as EXT_SOURCE_1 column is already included in nullcol_40_application

len(Unwanted_application)
```

Out[47]: 51

In [48]:
```
#  Checking the relevance of Flag_Document and whether it has any relation with loan repayment status

col_Doc=['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']
df_flag=application[col_Doc+['TARGET']]

length=len(col_Doc)

df_flag['TARGET']=df_flag['TARGET'].replace({1:'Defaulter',0:'Repayer'})
```

In [49]:
```
# checking is there is any correlation between mobile phone, work phone etc, email, Family members and Region rating

contact_col=['FLAG_MOBIL','FLAG_EMP_PHONE','FLAG_WORK_PHONE','FLAG_CONT_MOBILE',
             'FLAG_PHONE','FLAG_EMAIL','TARGET']
Contact_corr=application[contact_col].corr()
fig=plt.figure(figsize=(8,8))
ax=sns.heatmap(Contact_corr,
               xticklabels=Contact_corr.columns,
               yticklabels=Contact_corr.columns,
               annot=True,
               cmap='RdYlGn',
               linewidth=1)
```


FLAG_MOBIL
FLAG_EMP_PHONE
FLAG_WORK_PHONE
FLAG_CONT_MOBILE
FLAG_PHONE
FLAG_EMAIL
TARGET
1 -0.00085 0.0009 -7.8e-05 0.0011 0.00044 0.00053
-0.00085 1 0.23 -0.013 -0.016 0.063 0.046
0.0009 0.23 1 0.022 0.29 -0.012 0.029
-7.8e-05 -0.013 0.022 1 0.0063 -0.0054 0.00037
0.0011 -0.016 0.29 0.0063 1 0.015 -0.024
0.00044 0.063 -0.012 -0.0054 0.015 1 -0.0018
0.00053 0.046 0.029 0.00037 -0.024 -0.0018 1
1.0
0.8
0.6
0.4
0.2
0.0

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

In [51]:
```
# including the 6 FLAG columns to be deleted

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

Out[51]: 57

In [52]:
```
# Dropping the unnecessary columns from applicationDF

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

In [53]:
```
application.shape
```

Out[53]: (307511, 65)

In [54]:
```
application.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 65 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_2              307511 non-null  int64
 40  FLAG_DOCUMENT_3              307511 non-null  int64
 41  FLAG_DOCUMENT_4              307511 non-null  int64
 42  FLAG_DOCUMENT_5              307511 non-null  int64
 43  FLAG_DOCUMENT_6              307511 non-null  int64
 44  FLAG_DOCUMENT_7              307511 non-null  int64
 45  FLAG_DOCUMENT_8              307511 non-null  int64
 46  FLAG_DOCUMENT_9              307511 non-null  int64
 47  FLAG_DOCUMENT_10             307511 non-null  int64
 48  FLAG_DOCUMENT_11             307511 non-null  int64
 49  FLAG_DOCUMENT_12             307511 non-null  int64
 50  FLAG_DOCUMENT_13             307511 non-null  int64
 51  FLAG_DOCUMENT_14             307511 non-null  int64
 52  FLAG_DOCUMENT_15             307511 non-null  int64
 53  FLAG_DOCUMENT_16             307511 non-null  int64
 54  FLAG_DOCUMENT_17             307511 non-null  int64
 55  FLAG_DOCUMENT_18             307511 non-null  int64
 56  FLAG_DOCUMENT_19             307511 non-null  int64
 57  FLAG_DOCUMENT_20             307511 non-null  int64
 58  FLAG_DOCUMENT_21             307511 non-null  int64
 59  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64
 60  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64
 61  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64
 62  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64
 63  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64
 64  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64
dtypes: float64(18), int64(35), object(12)
memory usage: 152.5+ MB
```

In [55]:
```
unwanted_previous=nullcol_40_previous['Column Name'].tolist()
unwanted_previous
```

Out[55]:
```
['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 [56]:
```
# Listing down columns which are not needed

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

In [57]:
```
len(unneccessary_previous)
```

Out[57]:
```
4
```

In [58]:
```
Unwanted_previous=unwanted_previous+unneccessary_previous
len(Unwanted_previous)
```

Out[58]:
```
15
```

In [59]:
```
# Dropping the unnecessary columns from previous

previous_DF.drop(labels=Unwanted_previous,axis=1,inplace=True)

# Inspecting the dataframe after removal of unnecessary columns

previous_DF.shape
```

Out[59]:
```
(1048575, 22)
```

In [60]:
```
# Inspecting the column types after after removal of unnecessary columns

previous_DF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 22 columns):
 #   Column                  Non-Null Count    Dtype
---  ------                  --------------    -----
 0   SK_ID_PREV              1048575 non-null  int64
 1   SK_ID_CURR              1048575 non-null  int64
 2   NAME_CONTRACT_TYPE      1048575 non-null  object
 3   AMT_ANNUITY             815566 non-null   float64
 4   AMT_APPLICATION         1048575 non-null  float64
 5   AMT_CREDIT              1048575 non-null  float64
 6   AMT_GOODS_PRICE         807610 non-null   float64
 7   NAME_CASH_LOAN_PURPOSE  1048575 non-null  object
 8   NAME_CONTRACT_STATUS    1048575 non-null  object
 9   DAYS_DECISION           1048575 non-null  int64
 10  NAME_PAYMENT_TYPE       1048575 non-null  object
 11  CODE_REJECT_REASON      1048575 non-null  object
 12  NAME_CLIENT_TYPE        1048575 non-null  object
 13  NAME_GOODS_CATEGORY     1048575 non-null  object
 14  NAME_PORTFOLIO          1048575 non-null  object
 15  NAME_PRODUCT_TYPE       1048575 non-null  object
 16  CHANNEL_TYPE            1048575 non-null  object
 17  SELLERPLACE_AREA        1048575 non-null  int64
 18  NAME_SELLER_INDUSTRY    1048575 non-null  object
 19  CNT_PAYMENT             815569 non-null   float64
 20  NAME_YIELD_GROUP        1048575 non-null  object
 21  PRODUCT_COMBINATION     1048351 non-null  object
dtypes: float64(5), int64(4), object(13)
memory usage: 176.0+ MB
```

STANDRDIZE VALUE

In [62]:

```
# CONVERTING NEGATIVE DAYS TO POSITIVE DAYS

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

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

In [63]:
```
# Binning Numerical Columns to create a categorical column

# Creating bins for income amount

application['AMT_INCOME_TOTAL']=application['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', '

application['AMT_INCOME_RANGE']=pd.cut(application['AMT_INCOME_TOTAL'],bins,labels=slot)
```

In [64]:
```
application['AMT_INCOME_RANGE'].value_counts(normalize=True)*100
```

Out[64]:
```
AMT_INCOME_RANGE
100K-200K    50.735000
200k-300k    21.210691
0-100K       20.729695
300k-400k     4.776116
400k-500k     1.744669
500k-600k     0.356354
600k-700k     0.282805
800k-900k     0.096980
700k-800k     0.052721
900k-1M       0.009112
1M Above      0.005858
Name: proportion, dtype: float64
```

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

In [66]:
```
# Creating bins for Credit amount

application['AMT_CREDIT']=application['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', '1M Above']

application['AMT_CREDIT_RANGE']=pd.cut(application['AMT_CREDIT'],bins=bins,labels=slots)
```

In [67]:
```
# checking the binning of data and % of data in each category

application['AMT_CREDIT_RANGE'].value_counts(normalize=True)*100
```

Out[67]:
```
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
```

MORE THAN 16% LOAN APPLICATIONS HAVE TAKEN LOAN WHICH AMOUNTS TO MORE THAN 1M

In [69]:
```
# Creating bins for Age

application['AGE'] = application['DAYS_BIRTH'] // 365
bins = [0,20,30,40,50,100]
slots = ['0-20','20-30','30-40','40-50','50 above']

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

In [70]:
```
# checking the binning of data and % of data in each category

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

Out[70]:
```
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
```

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

In [72]:
```
# Creating bins for Employement Time

application['YEARS_EMPLOYED'] = application['DAYS_EMPLOYED'] // 365
bins = [0,5,10,20,30,40,50,60,150]
slots = ['0-5','5-10','10-20','20-30','30-40','40-50','50-60','60 above']

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

In [73]:
```
# checking the binning of data and % of data in each category

application['EMPLOYMENT_YEAR'].value_counts(normalize=True)*100
```

Out[73]:
```
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
```

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 [75]:
```
# Checking the number of unique values each column possess to identify categorical columns

application.nunique().sort_values()
```

Out[75]:
```
FLAG_DOCUMENT_14                 2
FLAG_DOCUMENT_6                  2
FLAG_DOCUMENT_13                 2
REG_REGION_NOT_LIVE_REGION       2
REG_REGION_NOT_WORK_REGION       2
                              ...
DAYS_EMPLOYED                12574
AMT_ANNUITY                  13672
DAYS_REGISTRATION            15688
DAYS_BIRTH                   17460
SK_ID_CURR                  307511
Length: 71, dtype: int64
```

In [76]:
```
# inspecting the column types if they are in correct data type using the above result.

application.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 71 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_2              307511 non-null  int64
 40  FLAG_DOCUMENT_3              307511 non-null  int64
 41  FLAG_DOCUMENT_4              307511 non-null  int64
 42  FLAG_DOCUMENT_5              307511 non-null  int64
 43  FLAG_DOCUMENT_6              307511 non-null  int64
 44  FLAG_DOCUMENT_7              307511 non-null  int64
 45  FLAG_DOCUMENT_8              307511 non-null  int64
 46  FLAG_DOCUMENT_9              307511 non-null  int64
 47  FLAG_DOCUMENT_10             307511 non-null  int64
 48  FLAG_DOCUMENT_11             307511 non-null  int64
 49  FLAG_DOCUMENT_12             307511 non-null  int64
 50  FLAG_DOCUMENT_13             307511 non-null  int64
 51  FLAG_DOCUMENT_14             307511 non-null  int64
 52  FLAG_DOCUMENT_15             307511 non-null  int64
 53  FLAG_DOCUMENT_16             307511 non-null  int64
 54  FLAG_DOCUMENT_17             307511 non-null  int64
 55  FLAG_DOCUMENT_18             307511 non-null  int64
 56  FLAG_DOCUMENT_19             307511 non-null  int64
 57  FLAG_DOCUMENT_20             307511 non-null  int64
 58  FLAG_DOCUMENT_21             307511 non-null  int64
 59  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64
 60  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64
 61  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64
 62  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64
 63  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64
 64  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64
 65  AMT_INCOME_RANGE             307279 non-null  category
 66  AMT_CREDIT_RANGE             307511 non-null  category
 67  AGE                          307511 non-null  int64
 68  AGE_GROUP                    307511 non-null  category
 69  YEARS_EMPLOYED               307511 non-null  int64
 70  EMPLOYMENT_YEAR              224233 non-null  category
dtypes: category(4), float64(18), int64(37), object(12)
memory usage: 158.4+ MB
```

Numeric columns are already in int64 and float64 format. Hence proceeding with other columns.

In [78]:
```
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:
    application[col] =pd.Categorical(application[col])
```

In [79]:
```
application.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 71 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_2              307511 non-null  int64
 40  FLAG_DOCUMENT_3              307511 non-null  int64
 41  FLAG_DOCUMENT_4              307511 non-null  int64
 42  FLAG_DOCUMENT_5              307511 non-null  int64
 43  FLAG_DOCUMENT_6              307511 non-null  int64
 44  FLAG_DOCUMENT_7              307511 non-null  int64
 45  FLAG_DOCUMENT_8              307511 non-null  int64
 46  FLAG_DOCUMENT_9              307511 non-null  int64
 47  FLAG_DOCUMENT_10             307511 non-null  int64
 48  FLAG_DOCUMENT_11             307511 non-null  int64
 49  FLAG_DOCUMENT_12             307511 non-null  int64
 50  FLAG_DOCUMENT_13             307511 non-null  int64
 51  FLAG_DOCUMENT_14             307511 non-null  int64
 52  FLAG_DOCUMENT_15             307511 non-null  int64
 53  FLAG_DOCUMENT_16             307511 non-null  int64
 54  FLAG_DOCUMENT_17             307511 non-null  int64
 55  FLAG_DOCUMENT_18             307511 non-null  int64
 56  FLAG_DOCUMENT_19             307511 non-null  int64
 57  FLAG_DOCUMENT_20             307511 non-null  int64
 58  FLAG_DOCUMENT_21             307511 non-null  int64
 59  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64
 60  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64
 61  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64
 62  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64
 63  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64
 64  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64
 65  AMT_INCOME_RANGE             307279 non-null  category
 66  AMT_CREDIT_RANGE             307511 non-null  category
 67  AGE                          307511 non-null  int64
 68  AGE_GROUP                    307511 non-null  category
 69  YEARS_EMPLOYED               307511 non-null  int64
 70  EMPLOYMENT_YEAR              224233 non-null  category
dtypes: category(23), float64(18), int64(30)
memory usage: 119.4 MB
```

In [80]:
```
previous_DF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 22 columns):
 #   Column                  Non-Null Count    Dtype
---  ------                  --------------    -----
 0   SK_ID_PREV              1048575 non-null  int64
 1   SK_ID_CURR              1048575 non-null  int64
 2   NAME_CONTRACT_TYPE      1048575 non-null  object
 3   AMT_ANNUITY             815566 non-null   float64
 4   AMT_APPLICATION         1048575 non-null  float64
 5   AMT_CREDIT              1048575 non-null  float64
 6   AMT_GOODS_PRICE         807610 non-null   float64
 7   NAME_CASH_LOAN_PURPOSE  1048575 non-null  object
 8   NAME_CONTRACT_STATUS    1048575 non-null  object
 9   DAYS_DECISION           1048575 non-null  int64
 10  NAME_PAYMENT_TYPE       1048575 non-null  object
 11  CODE_REJECT_REASON      1048575 non-null  object
 12  NAME_CLIENT_TYPE        1048575 non-null  object
 13  NAME_GOODS_CATEGORY     1048575 non-null  object
 14  NAME_PORTFOLIO          1048575 non-null  object
 15  NAME_PRODUCT_TYPE       1048575 non-null  object
 16  CHANNEL_TYPE            1048575 non-null  object
 17  SELLERPLACE_AREA        1048575 non-null  int64
 18  NAME_SELLER_INDUSTRY    1048575 non-null  object
 19  CNT_PAYMENT             815569 non-null   float64
 20  NAME_YIELD_GROUP        1048575 non-null  object
 21  PRODUCT_COMBINATION     1048351 non-null  object
dtypes: float64(5), int64(4), object(13)
memory usage: 176.0+ MB
```

In [81]:
```
previous_DF.nunique().sort_values()
```

Out[81]:
```
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                  48
SELLERPLACE_AREA           2023
DAYS_DECISION              2921
AMT_CREDIT                74637
AMT_GOODS_PRICE           75635
AMT_APPLICATION           75635
AMT_ANNUITY              282291
SK_ID_CURR               305828
SK_ID_PREV              1048575
dtype: int64
```

In [82]:
```
application.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 71 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_2              307511 non-null  int64
 40  FLAG_DOCUMENT_3              307511 non-null  int64
 41  FLAG_DOCUMENT_4              307511 non-null  int64
 42  FLAG_DOCUMENT_5              307511 non-null  int64
 43  FLAG_DOCUMENT_6              307511 non-null  int64
 44  FLAG_DOCUMENT_7              307511 non-null  int64
 45  FLAG_DOCUMENT_8              307511 non-null  int64
 46  FLAG_DOCUMENT_9              307511 non-null  int64
 47  FLAG_DOCUMENT_10             307511 non-null  int64
 48  FLAG_DOCUMENT_11             307511 non-null  int64
 49  FLAG_DOCUMENT_12             307511 non-null  int64
 50  FLAG_DOCUMENT_13             307511 non-null  int64
 51  FLAG_DOCUMENT_14             307511 non-null  int64
 52  FLAG_DOCUMENT_15             307511 non-null  int64
 53  FLAG_DOCUMENT_16             307511 non-null  int64
 54  FLAG_DOCUMENT_17             307511 non-null  int64
 55  FLAG_DOCUMENT_18             307511 non-null  int64
 56  FLAG_DOCUMENT_19             307511 non-null  int64
 57  FLAG_DOCUMENT_20             307511 non-null  int64
 58  FLAG_DOCUMENT_21             307511 non-null  int64
 59  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64
 60  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64
 61  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64
 62  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64
 63  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64
 64  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64
 65  AMT_INCOME_RANGE             307279 non-null  category
 66  AMT_CREDIT_RANGE             307511 non-null  category
 67  AGE                          307511 non-null  int64
 68  AGE_GROUP                    307511 non-null  category
 69  YEARS_EMPLOYED               307511 non-null  int64
 70  EMPLOYMENT_YEAR              224233 non-null  category
dtypes: category(23), float64(18), int64(30)
memory usage: 119.4 MB
```

In [83]:
```
#Converting negative days to positive days
```

```
previous_DF['DAYS_DECISION']=abs(previous_DF['DAYS_DECISION'])
```

In [84]:
```
previous_DF['DAYS_DECISION_GROUP'] = (previous_DF['DAYS_DECISION']-(previous_DF['DAYS_DECISION'] % 400)).astype(str)+'-'+ ((pr
```

In [85]:
```
previous_DF['DAYS_DECISION_GROUP'].value_counts(normalize=True)*100
```

Out[85]:
```
DAYS_DECISION_GROUP
0-400        37.392747
400-800      22.969840
800-1200     12.444508
1200-1600     7.932766
2400-2800     6.323820
1600-2000     5.829435
2000-2400     5.672556
2800-3200     1.434328
Name: proportion, dtype: float64
```

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

In [87]:
```
#Converting Categorical columns from Object to categorical

Catgorical_col_p = ['NAME_CASH_LOAN_PURPOSE','NAME_CONTRACT_STATUS','NAME_PAYMENT_TYPE',
                    'CODE_REJECT_REASON','NAME_CLIENT_TYPE','NAME_GOODS_CATEGORY','NAME_PORTFOLIO',
                   'NAME_PRODUCT_TYPE','CHANNEL_TYPE','NAME_SELLER_INDUSTRY','NAME_YIELD_GROUP','PRODUCT_COMBINATION',
                    'NAME_CONTRACT_TYPE','DAYS_DECISION_GROUP']

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

In [88]:
```
previous_DF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 23 columns):
 #   Column                  Non-Null Count    Dtype
---  ------                  --------------    -----
 0   SK_ID_PREV              1048575 non-null  int64
 1   SK_ID_CURR              1048575 non-null  int64
 2   NAME_CONTRACT_TYPE      1048575 non-null  category
 3   AMT_ANNUITY             815566 non-null   float64
 4   AMT_APPLICATION         1048575 non-null  float64
 5   AMT_CREDIT              1048575 non-null  float64
 6   AMT_GOODS_PRICE         807610 non-null   float64
 7   NAME_CASH_LOAN_PURPOSE  1048575 non-null  category
 8   NAME_CONTRACT_STATUS    1048575 non-null  category
 9   DAYS_DECISION           1048575 non-null  int64
 10  NAME_PAYMENT_TYPE       1048575 non-null  category
 11  CODE_REJECT_REASON      1048575 non-null  category
 12  NAME_CLIENT_TYPE        1048575 non-null  category
 13  NAME_GOODS_CATEGORY     1048575 non-null  category
 14  NAME_PORTFOLIO          1048575 non-null  category
 15  NAME_PRODUCT_TYPE       1048575 non-null  category
 16  CHANNEL_TYPE            1048575 non-null  category
 17  SELLERPLACE_AREA        1048575 non-null  int64
 18  NAME_SELLER_INDUSTRY    1048575 non-null  category
 19  CNT_PAYMENT             815569 non-null   float64
 20  NAME_YIELD_GROUP        1048575 non-null  category
 21  PRODUCT_COMBINATION     1048351 non-null  category
 22  DAYS_DECISION_GROUP     1048575 non-null  category
dtypes: category(14), float64(5), int64(4)
memory usage: 86.0 MB
```

In [89]:
```
# checking the null value % of each column in applicationDF dataframe

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

Out[89]:
```
SK_ID_CURR             0.00
TARGET                 0.00
NAME_CONTRACT_TYPE     0.00
CODE_GENDER            0.00
FLAG_OWN_CAR           0.00
                       ...
AMT_CREDIT_RANGE       0.00
AGE                    0.00
AGE_GROUP              0.00
YEARS_EMPLOYED         0.00
EMPLOYMENT_YEAR       27.08
Length: 71, dtype: float64
```

In [90]:
```
application['NAME_TYPE_SUITE'].describe()
```

Out[90]:
```
count            306219
unique                7
top       Unaccompanied
freq             248526
Name: NAME_TYPE_SUITE, dtype: object
```

In [91]:
```
application['NAME_TYPE_SUITE'].fillna((application['NAME_TYPE_SUITE'].mode()[0]),inplace = True)
```

In [92]:
```
application['OCCUPATION_TYPE'] = application['OCCUPATION_TYPE'].cat.add_categories('Unknown')
application['OCCUPATION_TYPE'].fillna('Unknown', inplace =True)
```

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

In [94]:
```
application[['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[94]:

	AMT_REQ_CREDIT_BUREAU_HOUR	AMT_REQ_CREDIT_BUREAU_DAY	AMT_REQ_CREDIT_BUREAU_WEEK	AMT_REQ_CREDIT_BUREAU_MO
count	265992.000000	265992.000000	265992.000000	265992.00000
mean	0.006402	0.007000	0.034362	0.26739
std	0.083849	0.110757	0.204685	0.91600
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

Impute with median as mean has decimals and this is number of requests

In [96]:
```
amount = ['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']

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

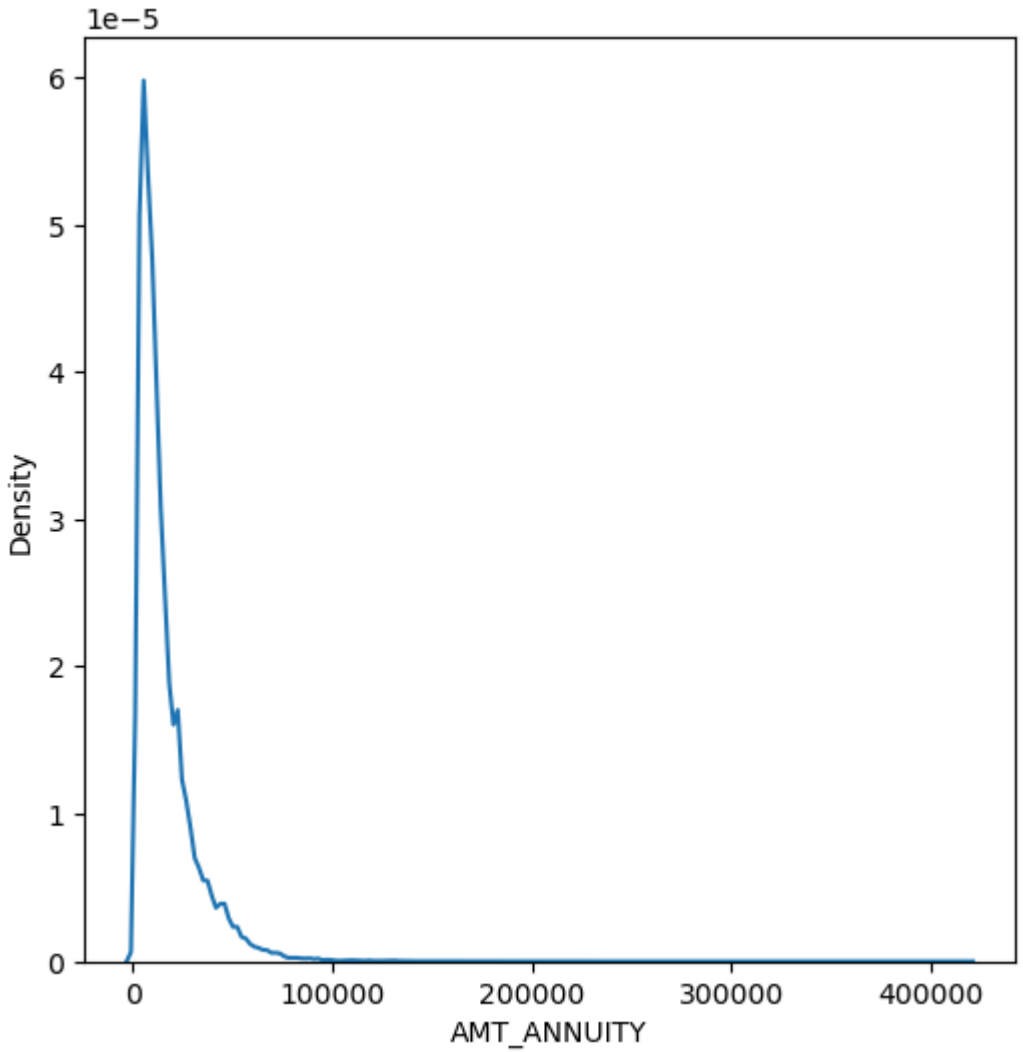
In [97]:
```
# checking the null value % of each column in previousDF dataframe

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

Out[97]:
```
SK_ID_CURR            0.00
TARGET                0.00
NAME_CONTRACT_TYPE    0.00
CODE_GENDER           0.00
FLAG_OWN_CAR          0.00
                      ...
AMT_CREDIT_RANGE      0.00
AGE                   0.00
AGE_GROUP             0.00
YEARS_EMPLOYED        0.00
EMPLOYMENT_YEAR       7.94
Length: 71, dtype: float64
```

In [98]:
```
# checking the null value % of each column in previousDF dataframe

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

Out[98]:
```
SK_ID_PREV                0.00
SK_ID_CURR                0.00
NAME_CONTRACT_TYPE        0.00
AMT_ANNUITY              22.22
AMT_APPLICATION           0.00
AMT_CREDIT                0.00
AMT_GOODS_PRICE          22.98
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              22.22
NAME_YIELD_GROUP          0.00
PRODUCT_COMBINATION       0.02
DAYS_DECISION_GROUP       0.00
dtype: float64
```

In [99]:
```
plt.figure(figsize=(6,6))
sns.kdeplot(previous_DF['AMT_ANNUITY'])
plt.show()
```

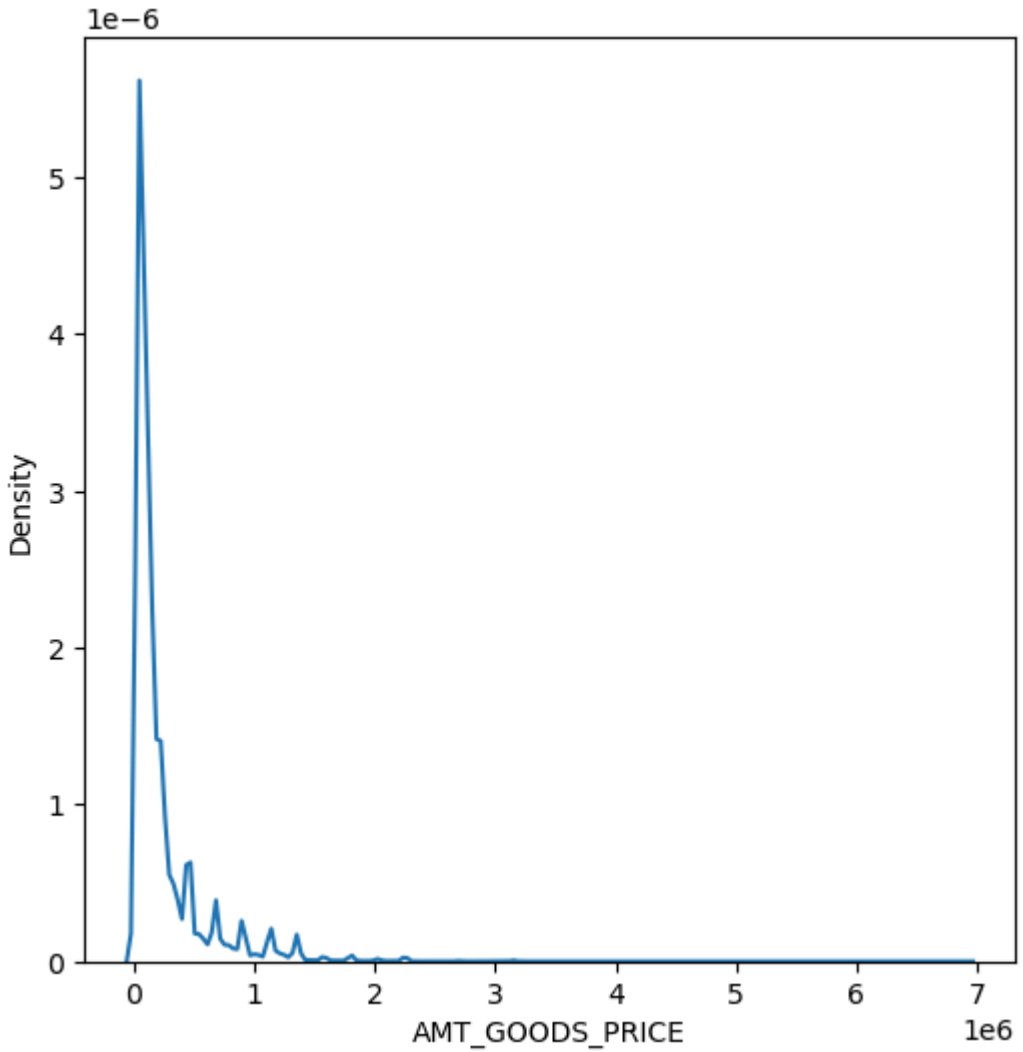


There is a single peak at the left side of the distribution and it indicates the presence of outliers and hence imputing with mean would not be the right approach and hence imputing with median.

In [101…
```
previous_DF['AMT_ANNUITY'].fillna(previous_DF['AMT_ANNUITY'].median(),inplace = True)
```

In [102…
```
plt.figure(figsize=(6,6))
sns.kdeplot(previous_DF['AMT_GOODS_PRICE'][pd.notnull(previous_DF['AMT_GOODS_PRICE'])])
```

```
plt.show()
```

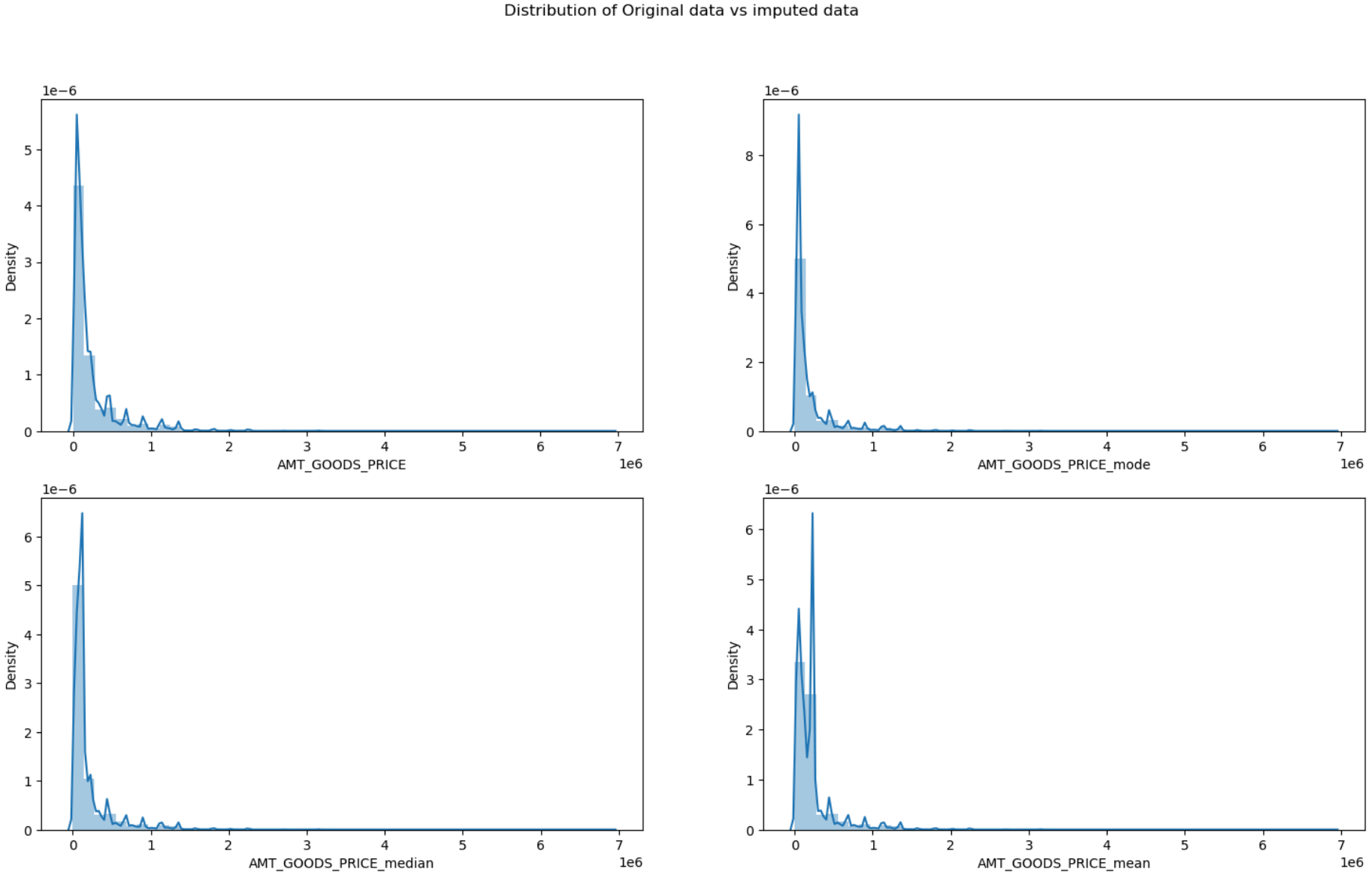


There are several peaks along the distribution. Let's impute using the mode, mean and median and see if the distribution is still about the same.

In [104…

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

```
statsDF['AMT_GOODS_PRICE_median'] = previous_DF['AMT_GOODS_PRICE'].fillna(previous_DF['AMT_GOODS_PRICE'].median())
statsDF['AMT_GOODS_PRICE_mean'] = previous_DF['AMT_GOODS_PRICE'].fillna(previous_DF['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(previous_DF['AMT_GOODS_PRICE'][pd.notnull(previous_DF['AMT_GOODS_PRICE'])]);
for i in enumerate(cols):
    plt.subplot(2,2,i[0]+2)
    sns.distplot(statsDF[i[1]])
```



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

In [106…

```
previous_DF['AMT_GOODS_PRICE'].fillna(previous_DF['AMT_GOODS_PRICE'].mode()[0], inplace=True)
```

In [107…
```
previous_DF.loc[previous_DF['CNT_PAYMENT'].isnull(),'NAME_CONTRACT_STATUS'].value_counts()
```

Out[107…
```
NAME_CONTRACT_STATUS
Canceled        191156
Refused          25655
Unused offer     16192
Approved             3
Name: count, dtype: int64
```

In [108…
```
previous_DF['CNT_PAYMENT'].fillna(0,inplace = True)
```

In [109…
```
# checking the null value % of each column in previousDF dataframe

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

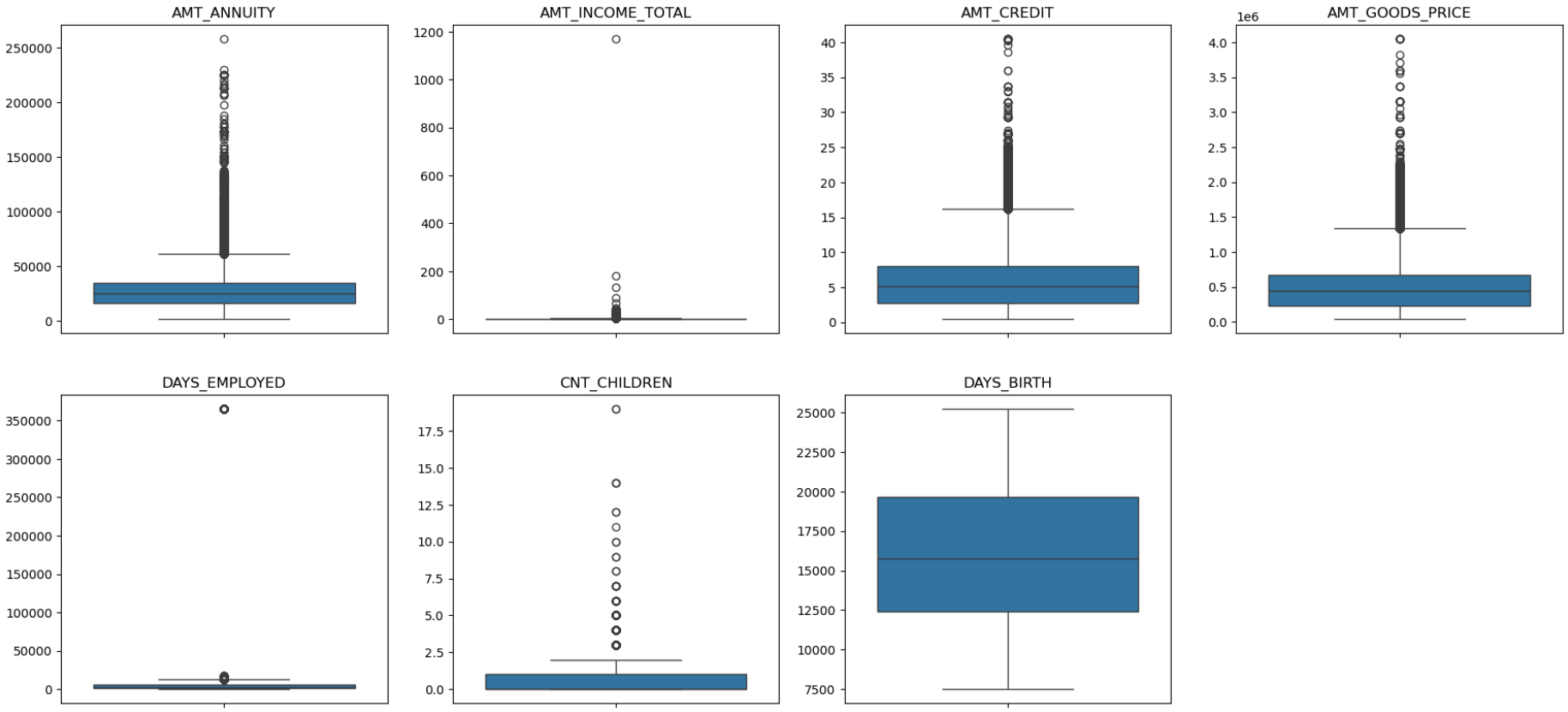
Out[109…
```
SK_ID_PREV                0.00
SK_ID_CURR                0.00
NAME_CONTRACT_TYPE        0.00
AMT_ANNUITY               0.00
AMT_APPLICATION           0.00
AMT_CREDIT                0.00
AMT_GOODS_PRICE           0.00
NAME_CASH_LOAN_PURPOSE    0.00
NAME_CONTRACT_STATUS      0.00
DAYS_DECISION             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
DAYS_DECISION_GROUP       0.00
dtype: float64
```

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

Identifying The Outliers

In [112…

```
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=application[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=application[i[1]])
    plt.title(i[1])
    plt.ylabel("")
```



It can be seen that in current application data

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

In [114…

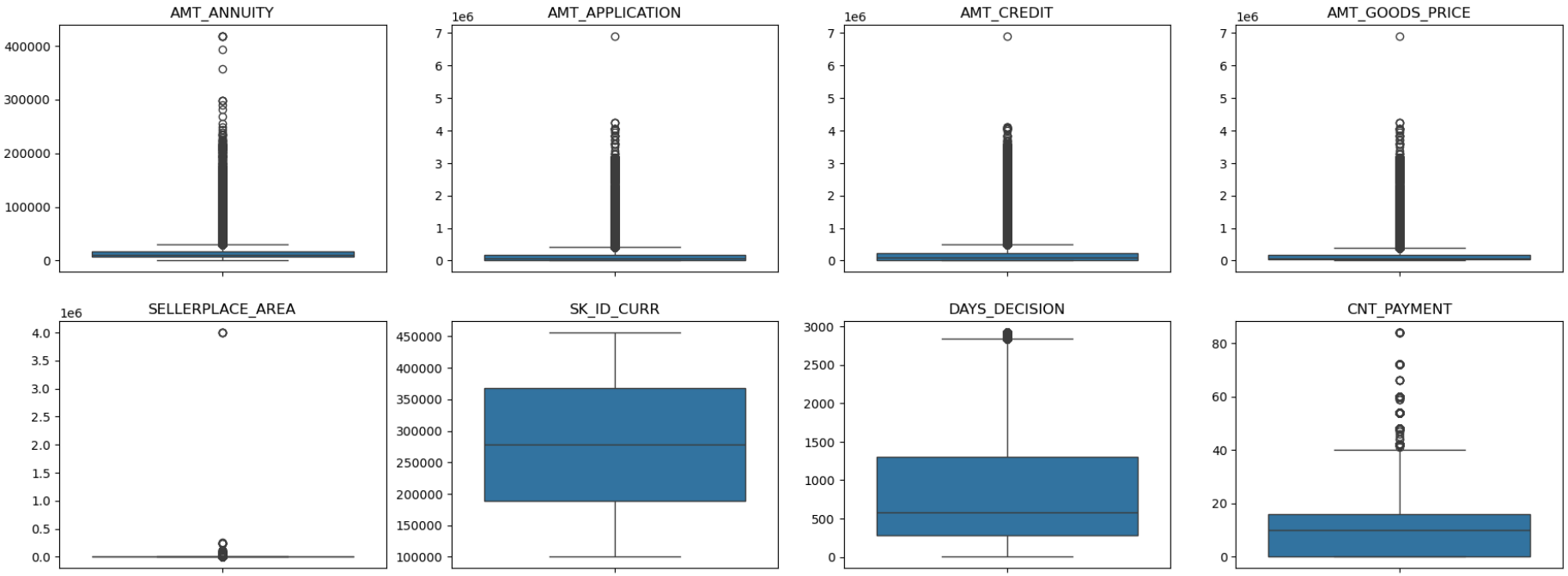
```
application[['AMT_ANNUITY', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_GOODS_PRICE', 'DAYS_BIRTH','CNT_CHILDREN','DAYS_EMPLOYED']]
```

Out[114...

	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	1.687979	5.990260	5.383962e+05	16036.995067	0.417052	67724.742149
std	14493.737315	2.371231	4.024908	3.694465e+05	4363.988632	0.722121	139443.751806
min	1615.500000	0.256500	0.450000	4.050000e+04	7489.000000	0.000000	0.000000
25%	16524.000000	1.125000	2.700000	2.385000e+05	12413.000000	0.000000	933.000000
50%	24903.000000	1.471500	5.135310	4.500000e+05	15750.000000	0.000000	2219.000000
75%	34596.000000	2.025000	8.086500	6.795000e+05	19682.000000	1.000000	5707.000000
max	258025.500000	1170.000000	40.500000	4.050000e+06	25229.000000	19.000000	365243.000000

In [115...

```
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=previous_DF[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=previous_DF[i[1]])
    plt.title(i[1])
    plt.ylabel("")
```



It can be seen that in previous application data

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

In [117…

```
previous_DF[['AMT_ANNUITY', 'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_GOODS_PRICE', 'SELLERPLACE_AREA','CNT_PAYMENT','DAYS_DECISIO
```

Out[117...

	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_GOODS_PRICE	SELLERPLACE_AREA	CNT_PAYMENT	DAYS_DECISION
count	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06
mean	1.485991e+04	1.742698e+05	1.950000e+05	1.846285e+05	3.183904e+02	1.244121e+01	8.820381e+02
std	1.314679e+04	2.910789e+05	3.169407e+05	2.854630e+05	7.996734e+03	1.441992e+01	7.792649e+02
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-1.000000e+00	0.000000e+00	2.000000e+00
25%	7.506765e+03	1.890000e+04	2.427750e+04	4.500000e+04	-1.000000e+00	0.000000e+00	2.810000e+02
50%	1.125000e+04	7.081650e+04	8.025300e+04	7.082100e+04	4.000000e+00	1.000000e+01	5.830000e+02
75%	1.673721e+04	1.800000e+05	2.152395e+05	1.800000e+05	8.500000e+01	1.600000e+01	1.303000e+03
max	4.180581e+05	6.905160e+06	6.905160e+06	6.905160e+06	4.000000e+06	8.400000e+01	2.922000e+03

In [118...

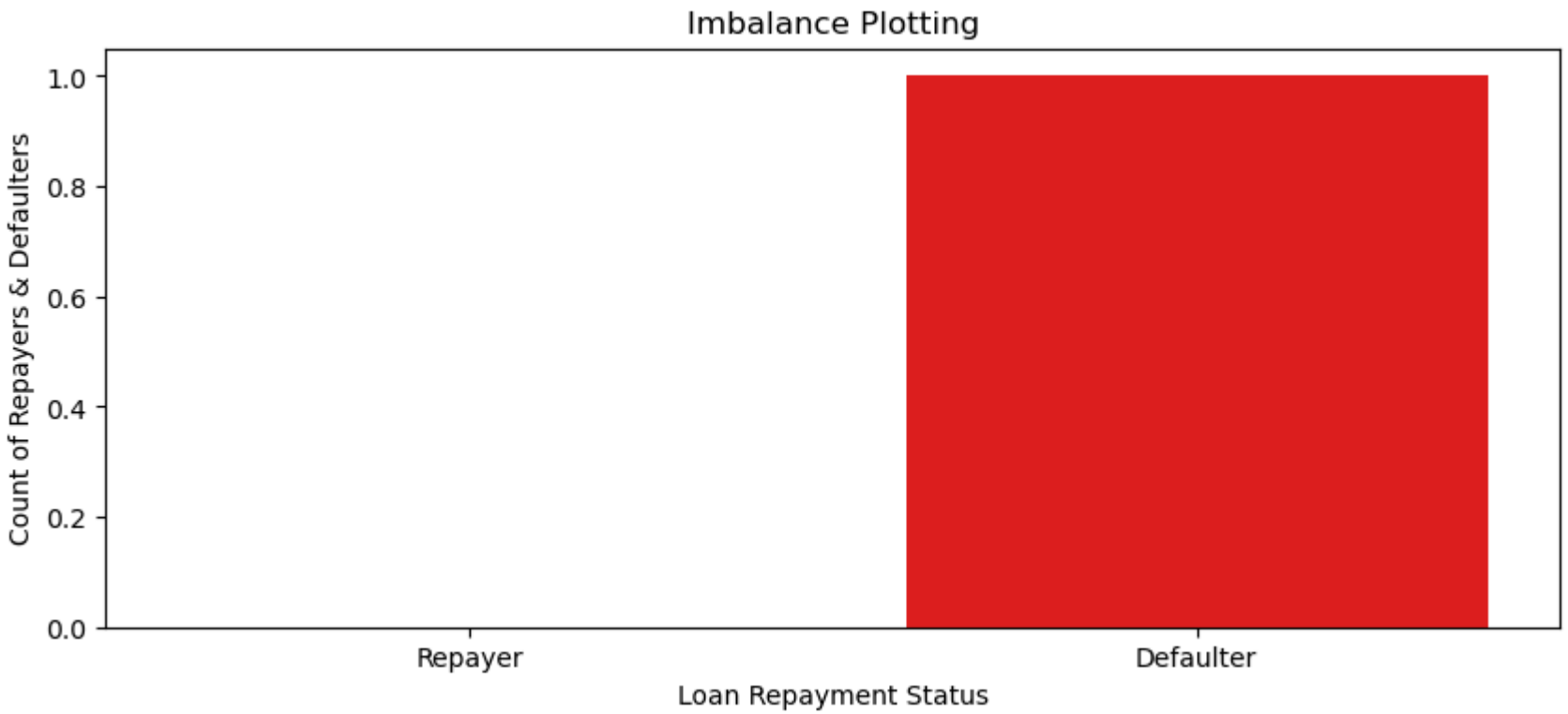
```
application["TARGET"].value_counts().reset_index()
```

Out[118...

	TARGET	count
0	0	282686
1	1	24825

In [119...

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

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



In [120…

```
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 [121…

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

def univariate_categorical(feature,ylog=False,label_rotation=False,horizontal_layout=True):
```

```
    temp = application[feature].value_counts()
    df1 = pd.DataFrame({feature: temp.index,'Number of contracts': temp.values})

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

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

    # 1. Subplot 1: Count plot of categorical column
    # sns.set_palette("Set2")
    s = sns.countplot(ax=ax1,
                      x = feature,
                      data=application,
                      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 [122…

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

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

    plt.figure(figsize=figsize)
    sns.barplot(x=x,
                y=y,
                data=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 [123…

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

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 [124…

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

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()
```

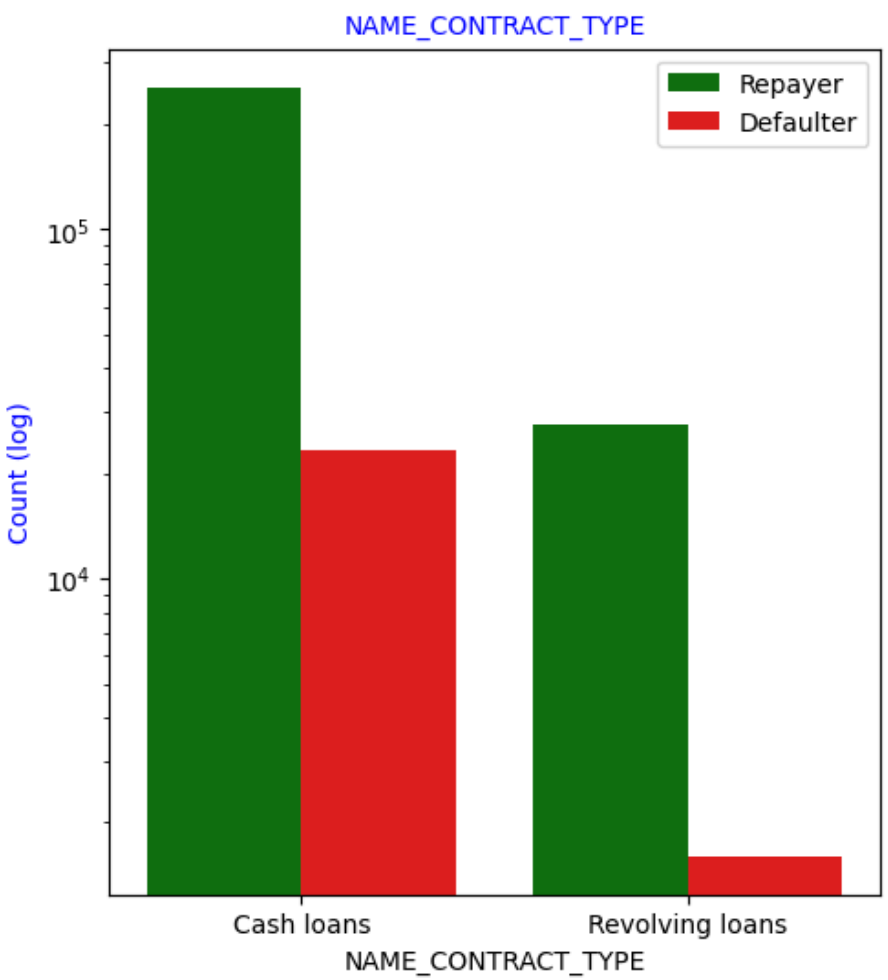
In [125…

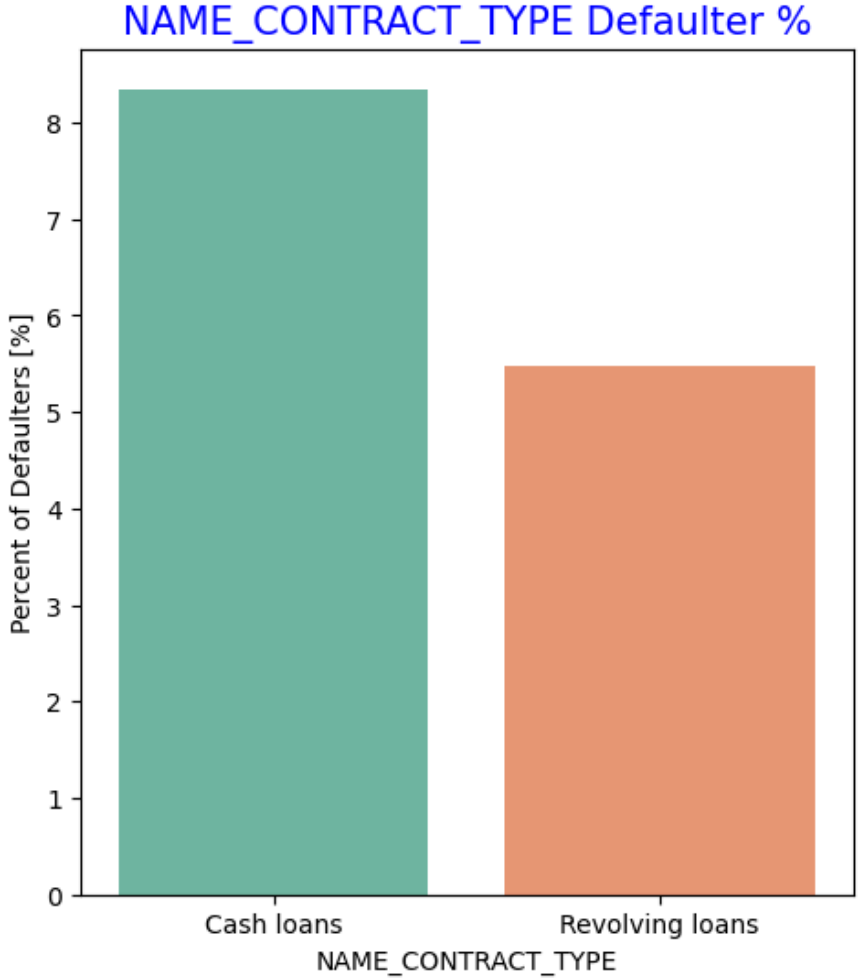
```
# Function to plot point plots on merged dataframe

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'])
```

Segmented Univariate Analysis

In [127…

```
# Checking the contract type based on loan repayment status

univariate_categorical('NAME_CONTRACT_TYPE',True)
```

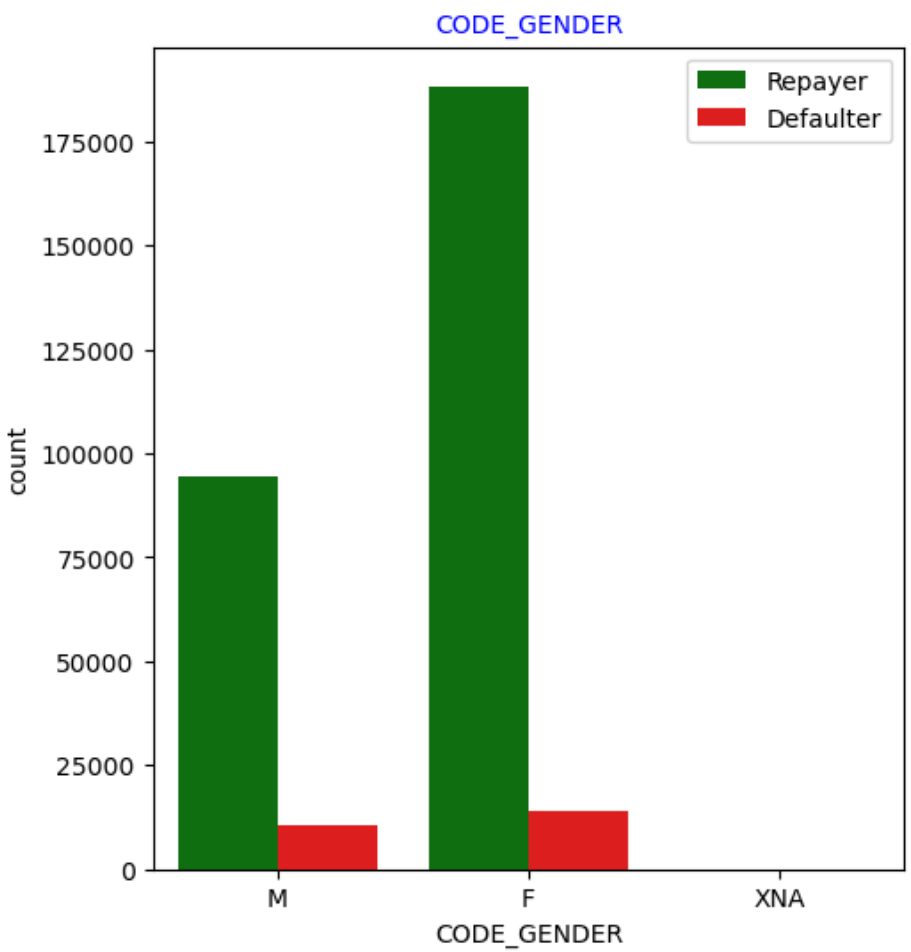


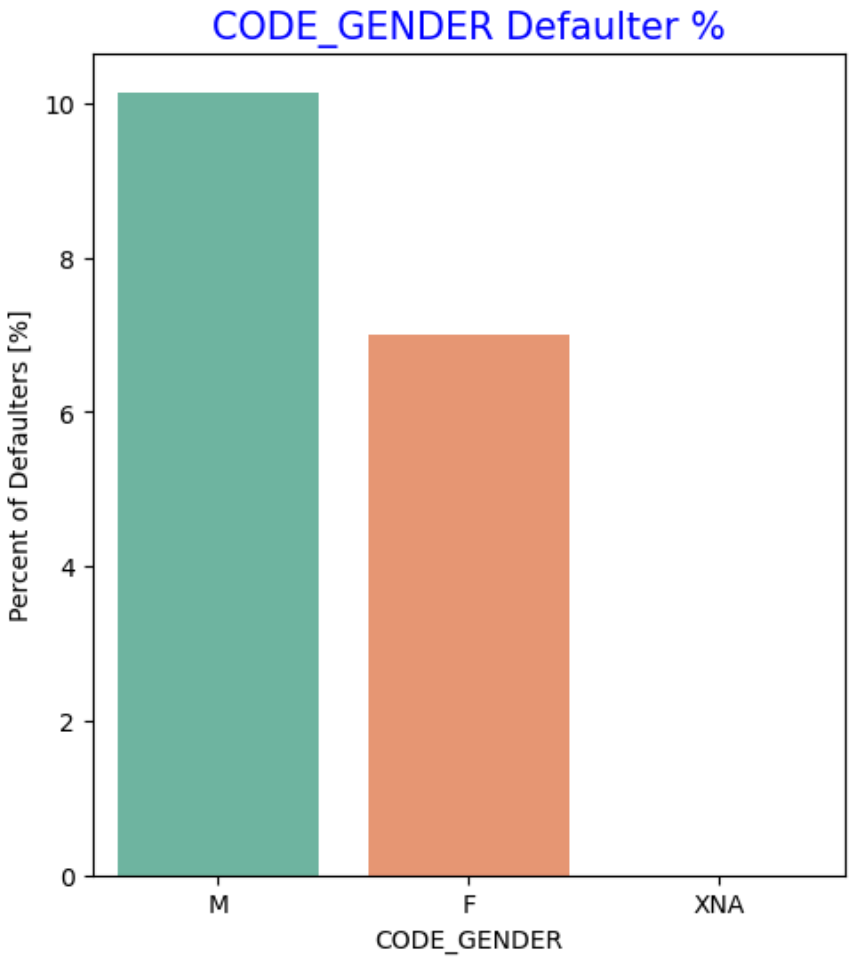


Contract type: Revolving loans are just a small fraction (10%) from the total number of loans; in the same time, a larger amount of Revolving loans, comparing with their frequency, are not repaid.

In [129…

```
# Checking the type of Gender on loan repayment status

univariate_categorical('CODE_GENDER')
```

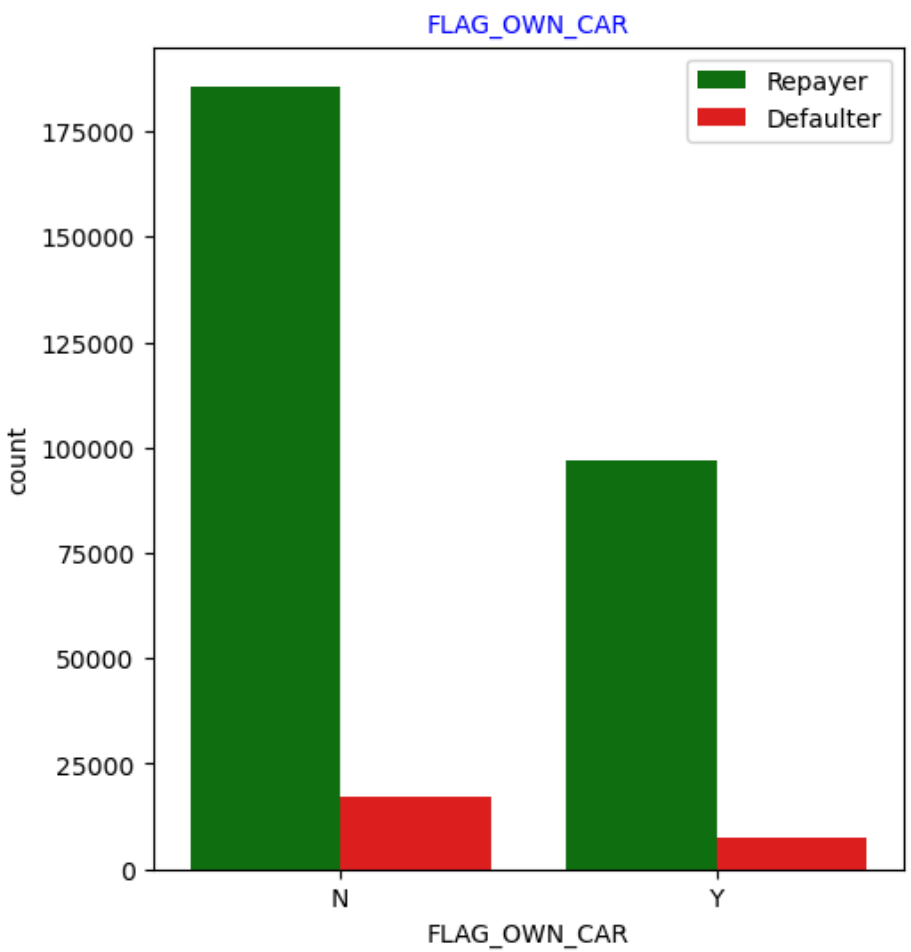


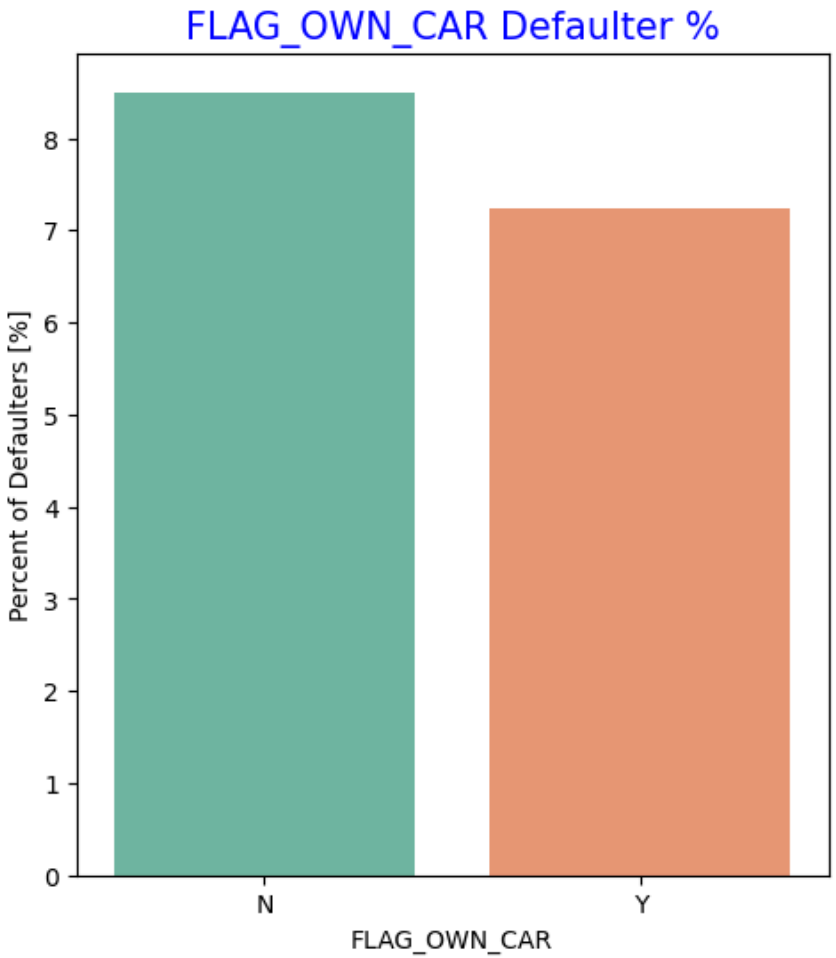


The number of female clients is almost double the number of male clients. Based on the percentage of defaulted credits, males have a higher chance of not returning their loans (10%), comparing with women (7%)

In [131…

```
# Checking if owning a car is related to loan repayment status

univariate_categorical('FLAG_OWN_CAR')
```



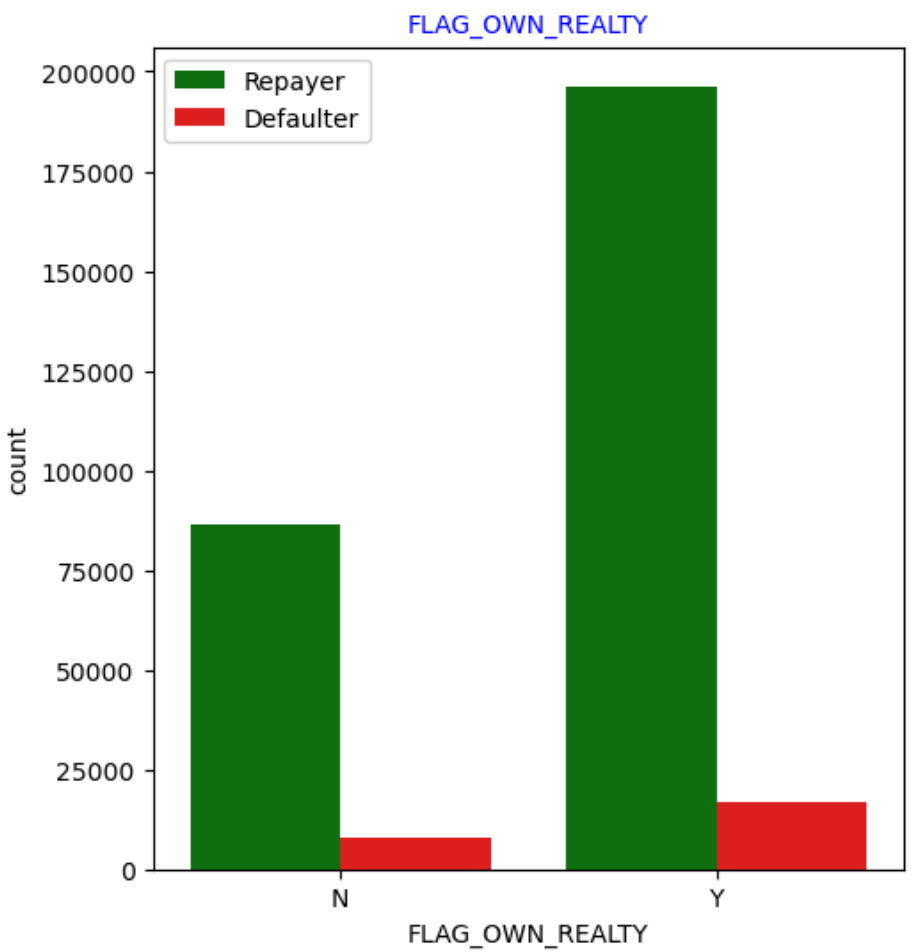


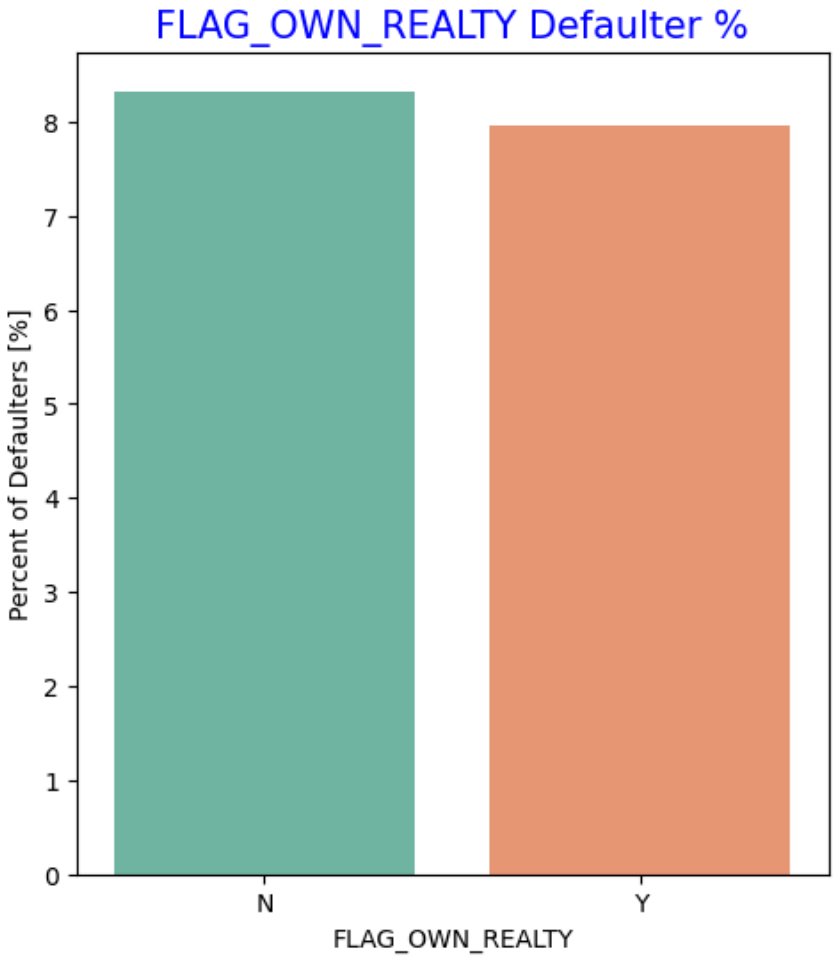
Clients who own a car are half in number of the clients who dont own a car. But based on the percentage of deault, there is no correlation between owning a car and loan repayment as in both cases the default percentage is almost same.

In [133…

```
# Checking if owning a realty is related to loan repayment status
```

```
univariate_categorical('FLAG_OWN_REALTY')
```

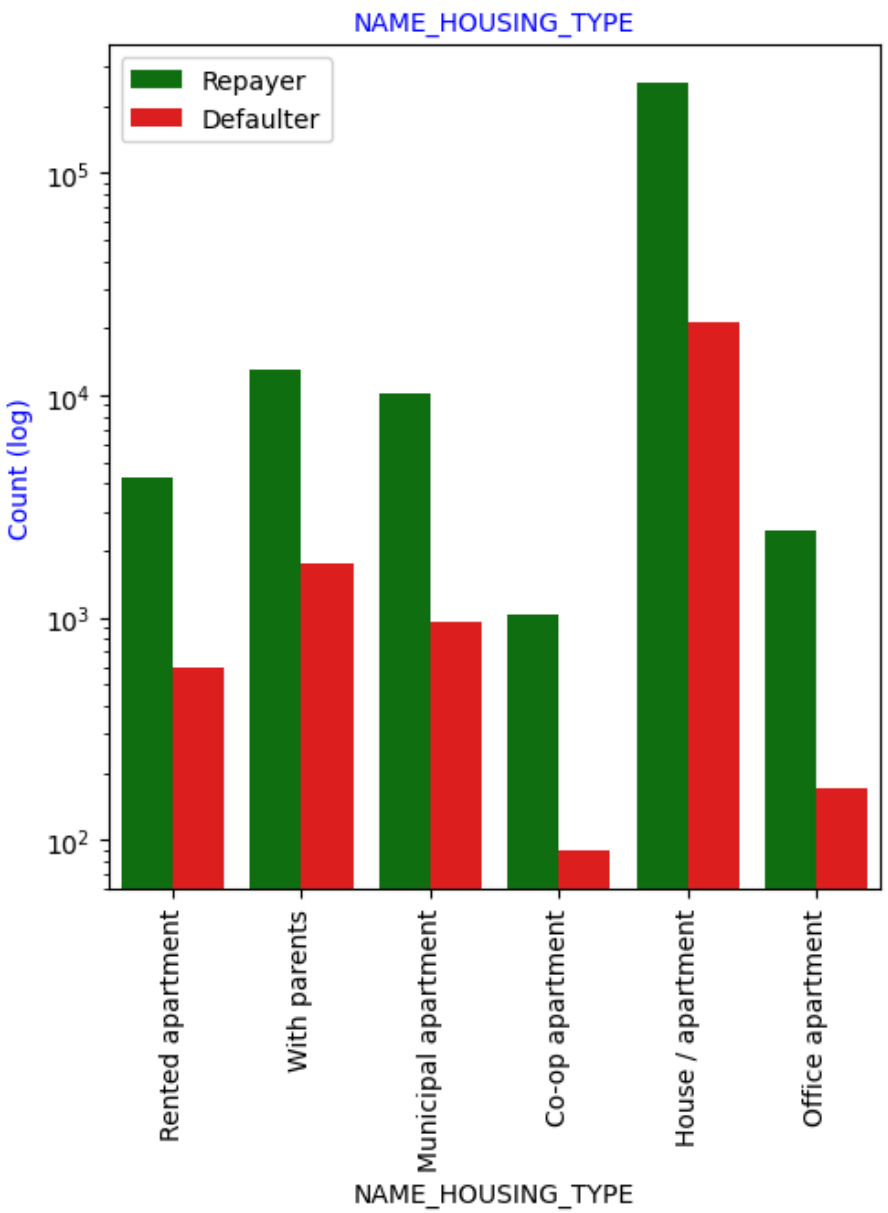


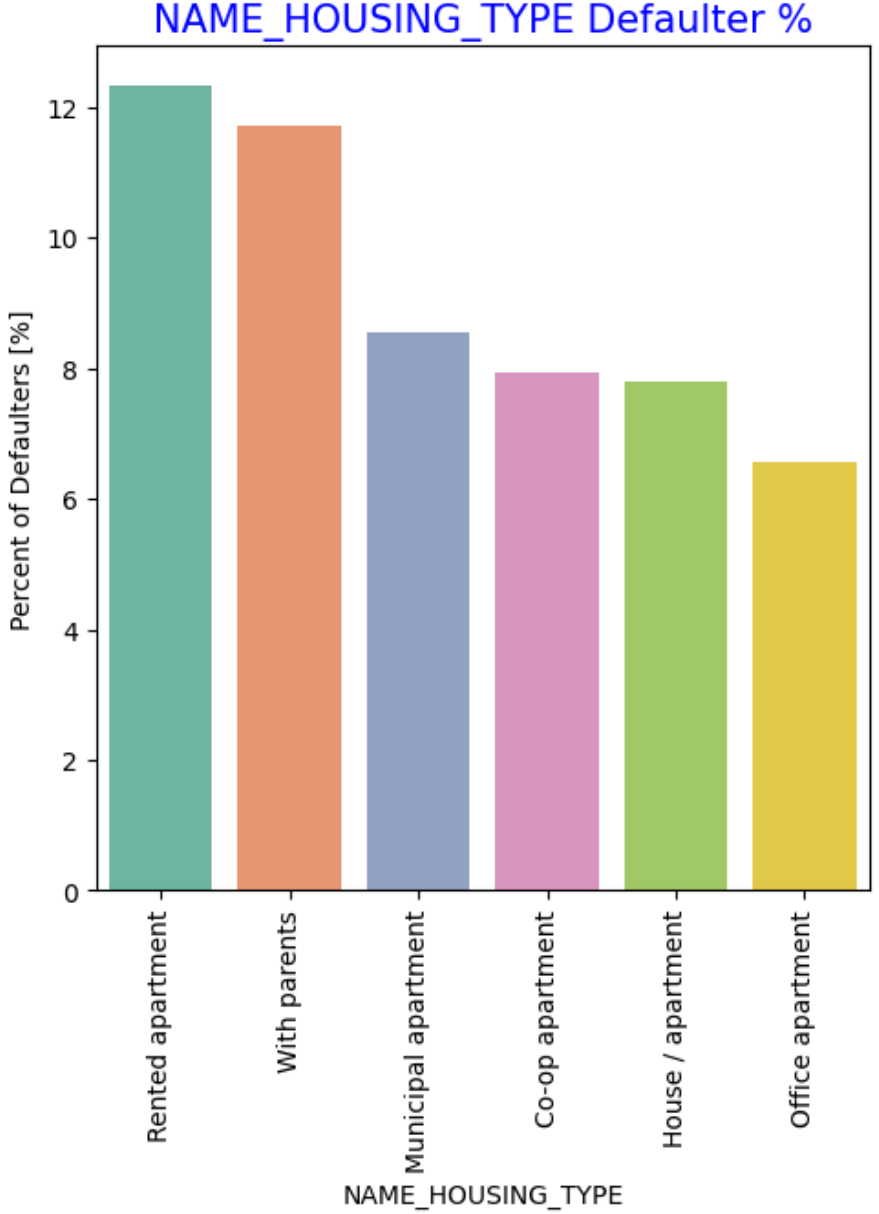


The clients who own real estate are more than double of the ones that don't own. But the defaulting rate of both categories are around the same (8%). Thus there is no correlation between owning a reality and defaulting the loan.

In [135…

```
# Analyzing Housing Type based on loan repayment status

univariate_categorical("NAME_HOUSING_TYPE",True,True,True)
```





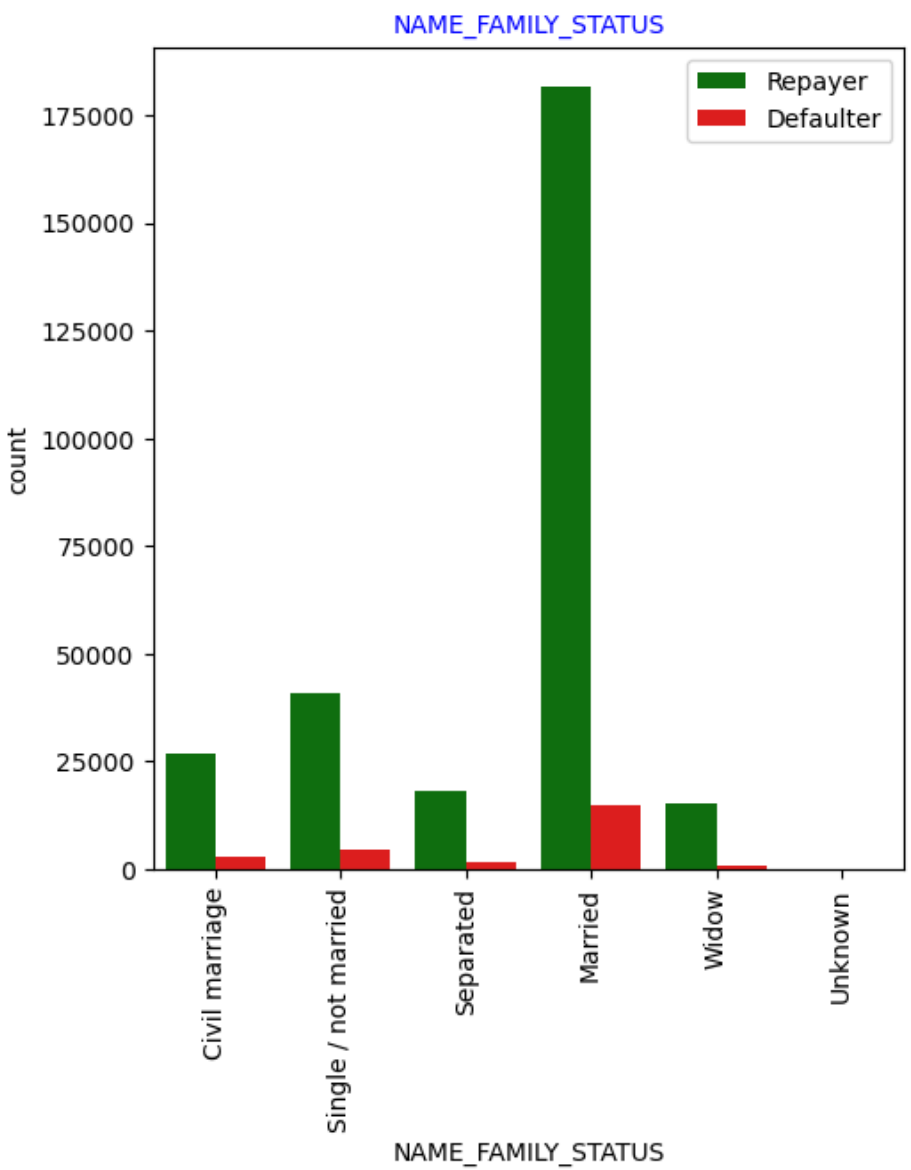
Majority of people live in House/apartment

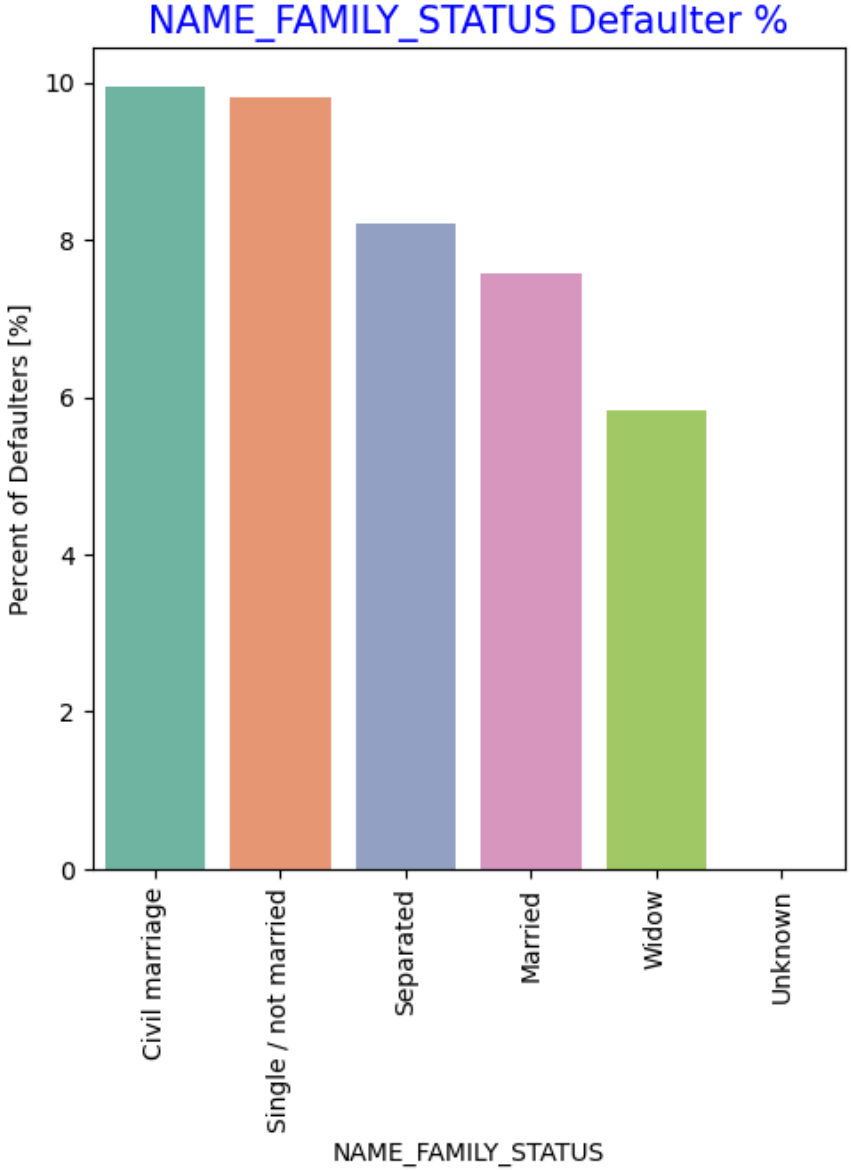
People living in office apartments have lowest default rate

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

In [137…

```
# Analyzing Family status based on loan repayment status

univariate_categorical("NAME_FAMILY_STATUS",False,True,True)
```

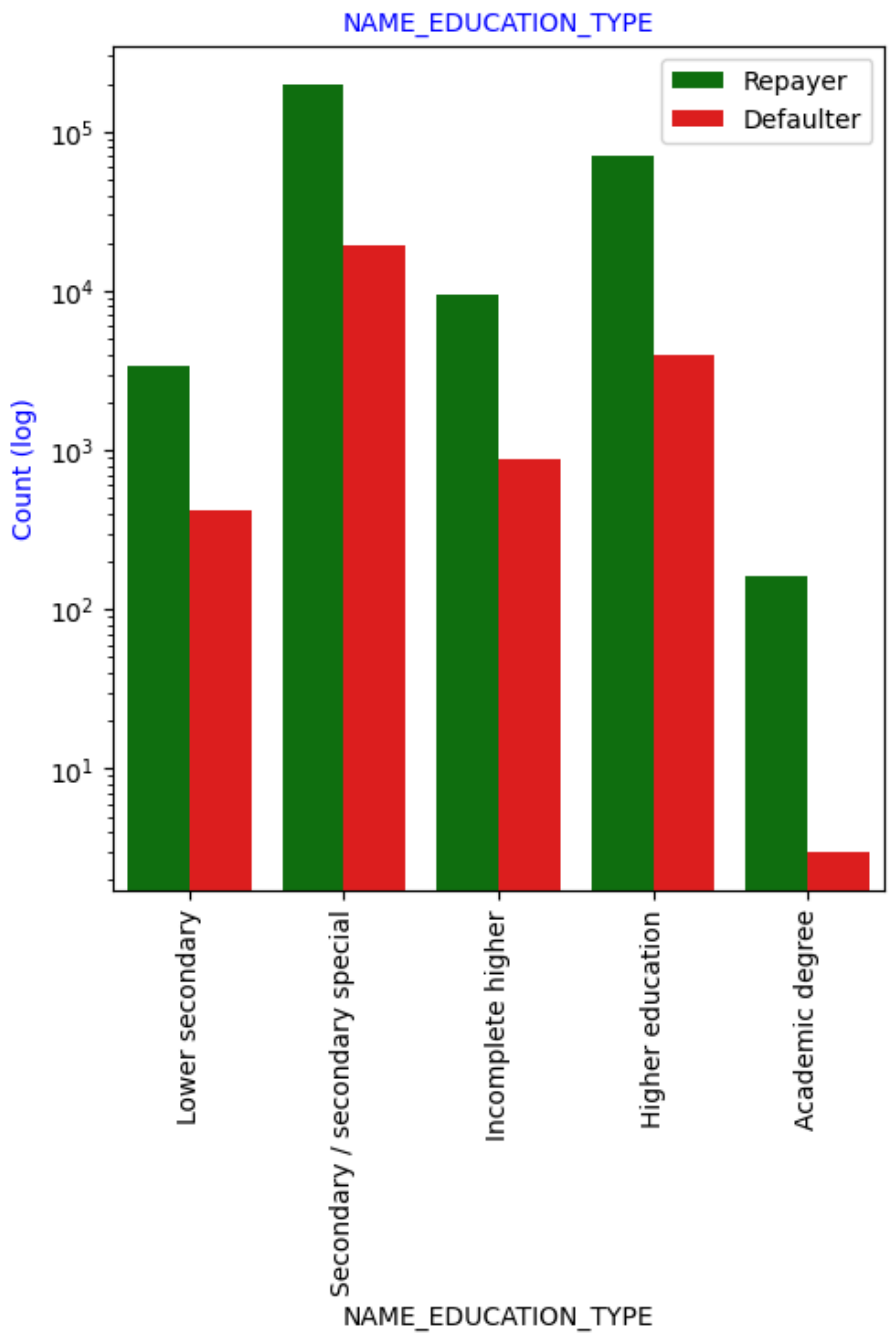


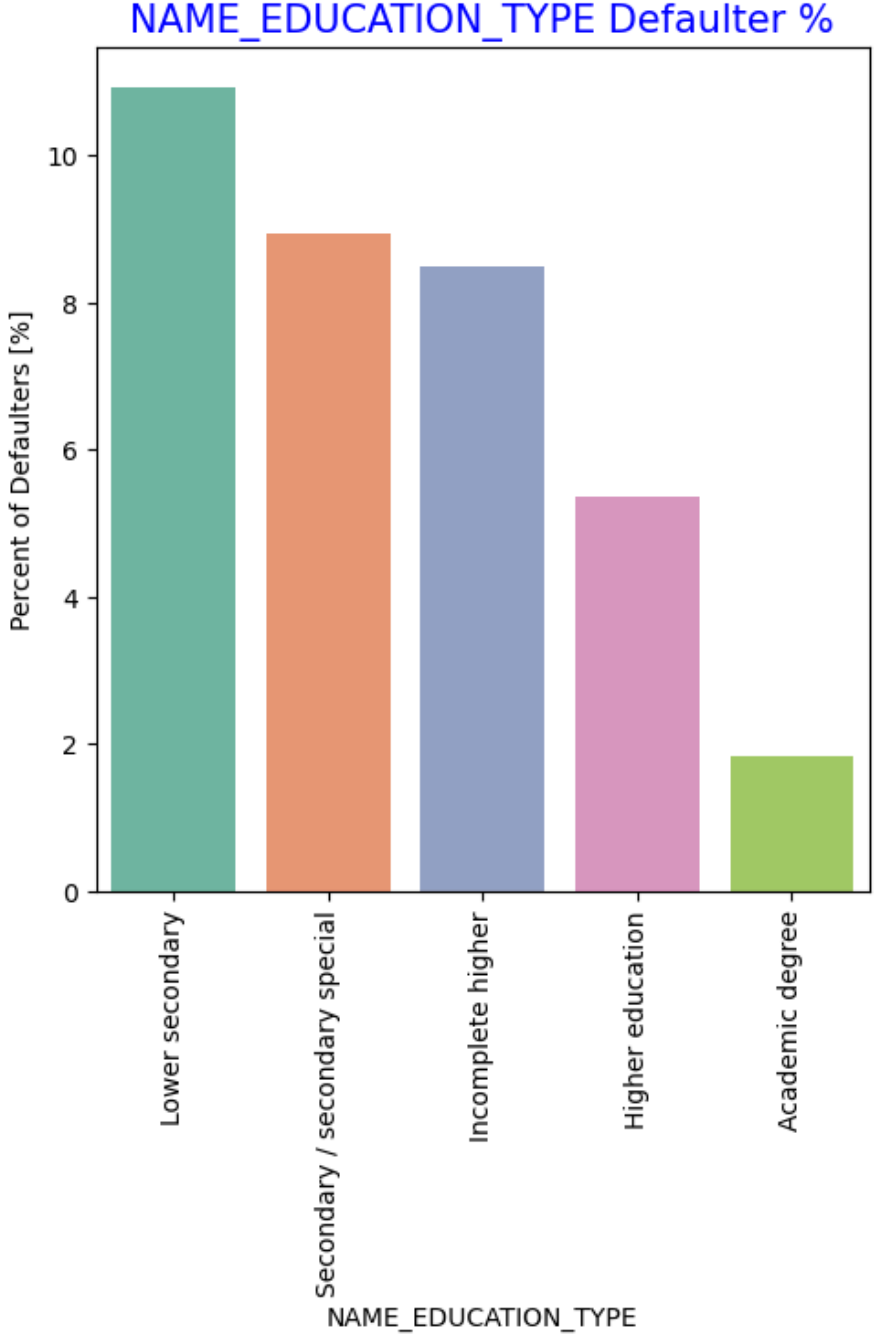


Most of the people who have taken loan are married, followed by Single/not married and civil marriage In terms of percentage of not repayment of loan, Civil marriage has the highest percent of not repayment (10%), with Widow the lowest (exception being Unknown).

In [139…

```
# Analyzing Education Type based on loan repayment status

univariate_categorical("NAME_EDUCATION_TYPE",True,True,True)
```

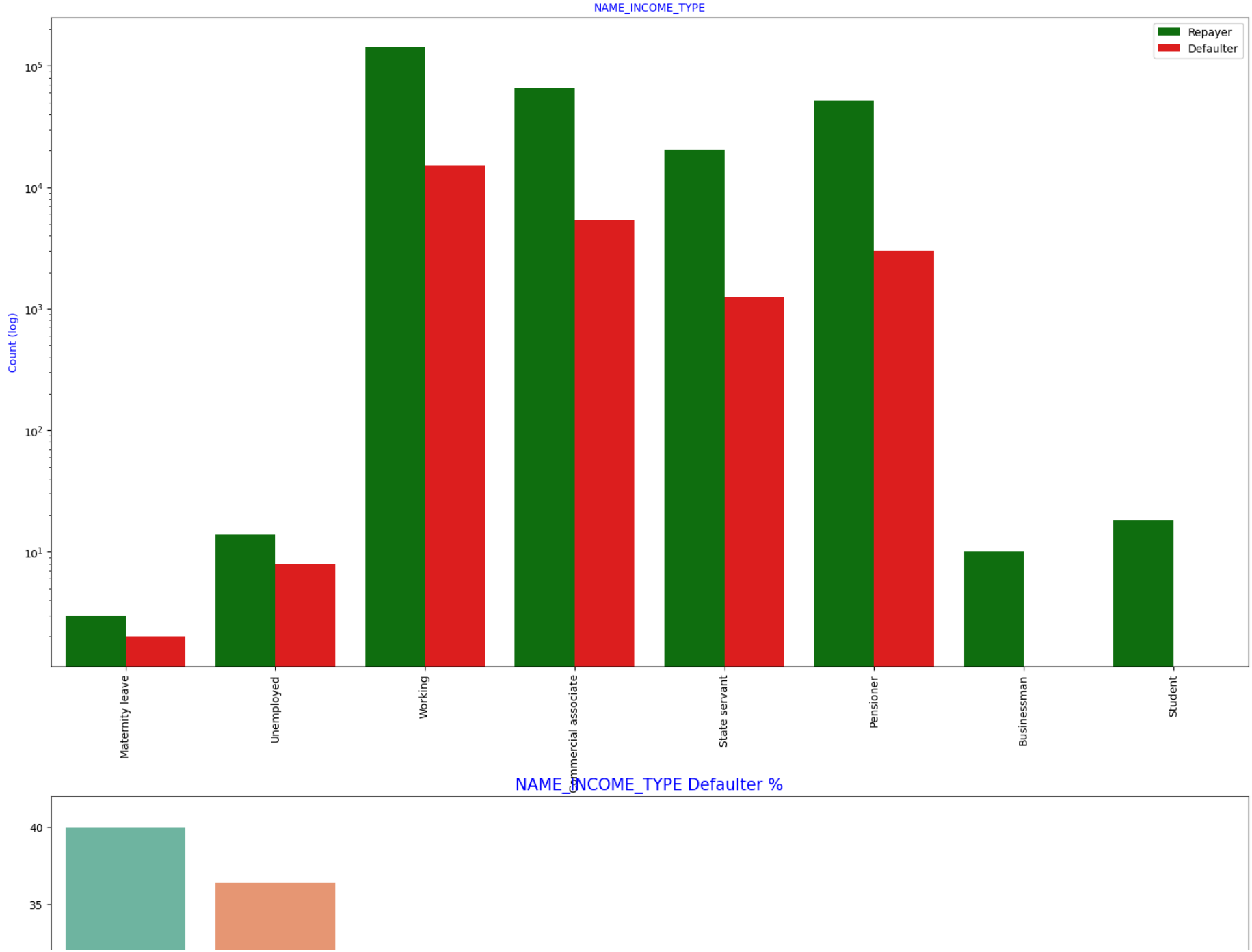

NAME_EDUCATION_TYPE
Repayer
Defaulter
Count (log)
10^5
10^4
10^3
10^2
10^1
Lower secondary
Secondary / secondary special
Incomplete higher
Higher education
Academic degree
NAME_EDUCATION_TYPE

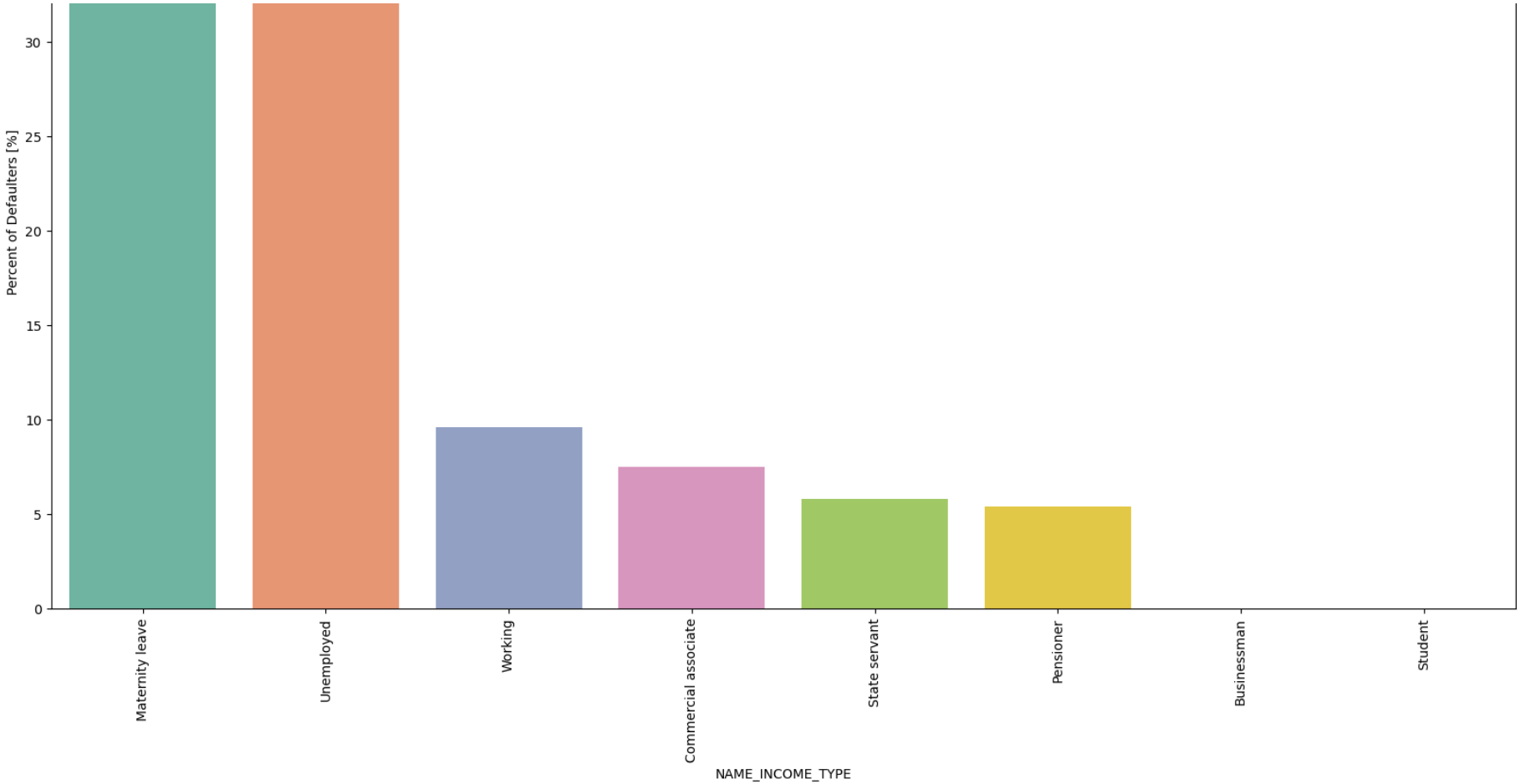

NAME_EDUCATION_TYPE Defaulter %
Percent of Defaulters [%]
10
8
6
4
2
0
Lower secondary
Secondary / secondary special
Incomplete higher
Higher education
Academic degree
NAME_EDUCATION_TYPE

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

In [141…

```
# 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



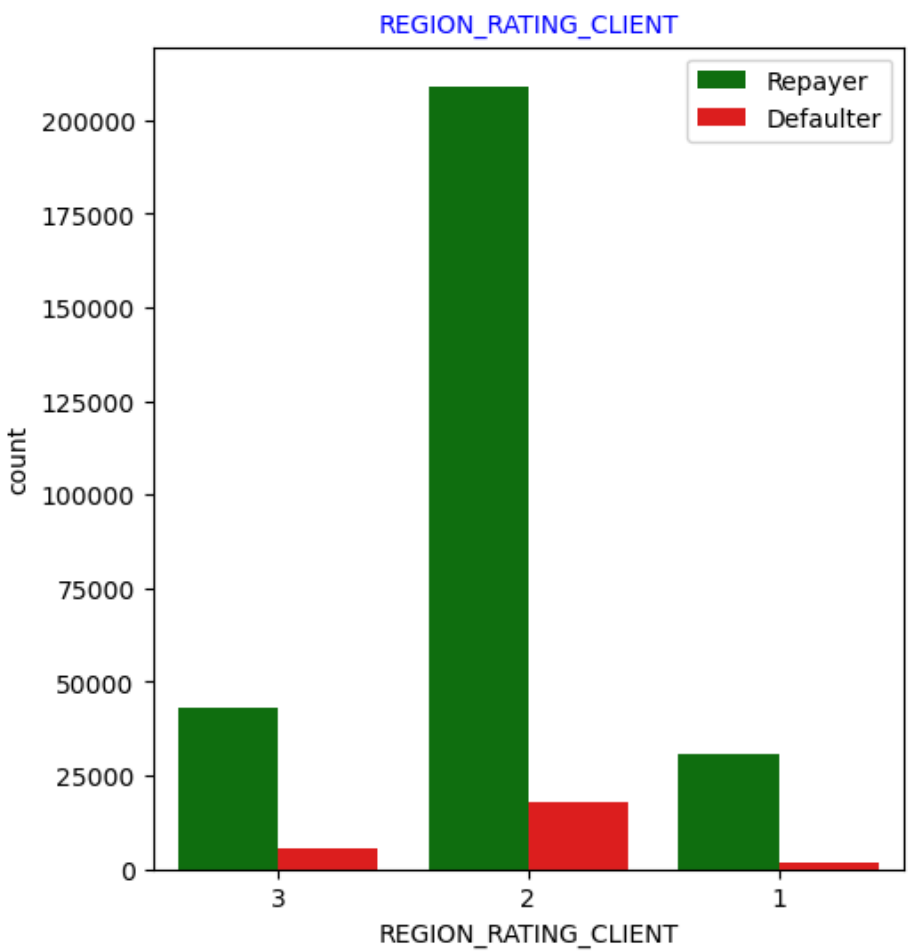
Most of applicants for loans have income type as Working, followed by Commercial associate, Pensioner and State servant.

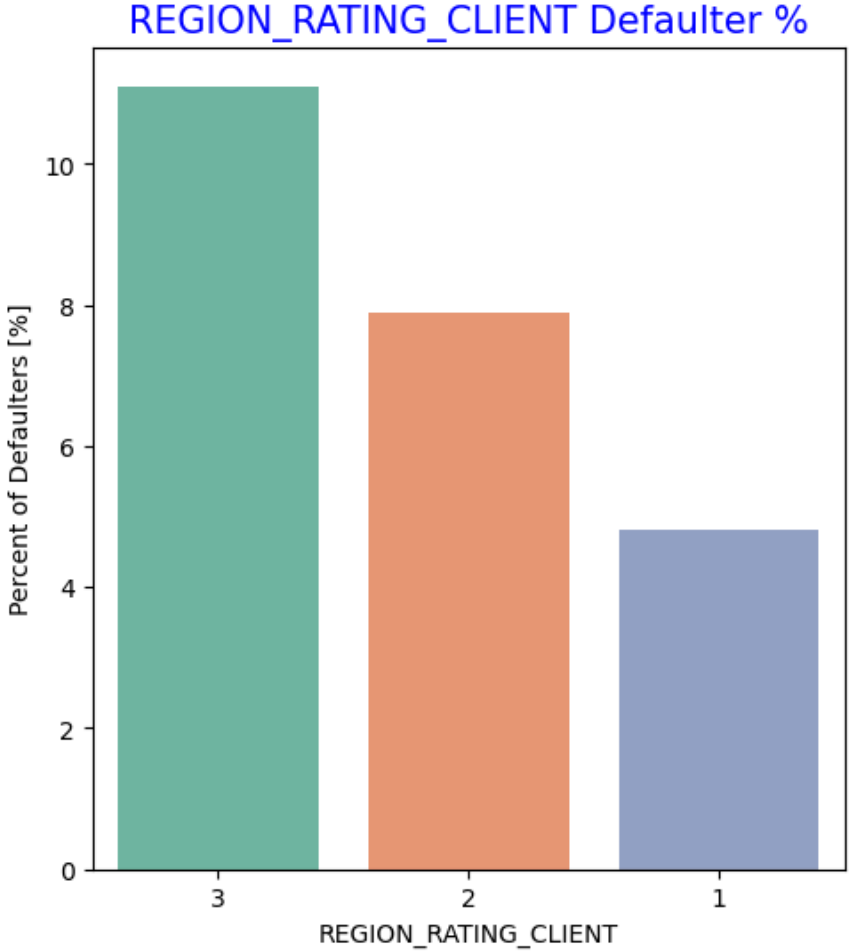
The applicants with the type of income Maternity leave have almost 40% ratio of not returning loans, followed by Unemployed (37%). The rest of types of incomes are under the average of 10% for not returning loans.

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

In [143…

```
# Analyzing Region rating where applicant lives based on loan repayment status

univariate_categorical("REGION_RATING_CLIENT",False,False,True)
```





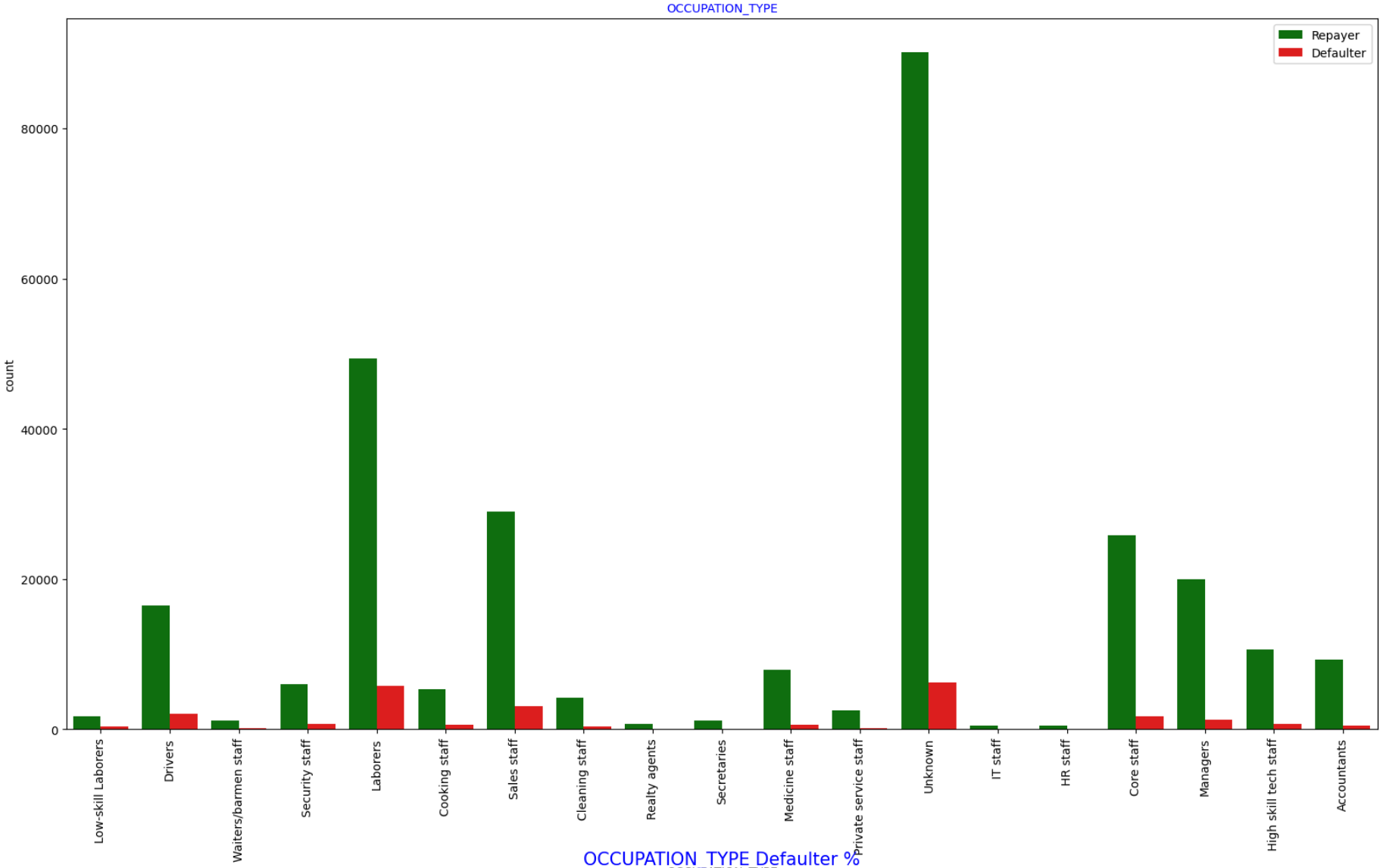
Most of the applicants are living in Region_Rating 2 place.

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

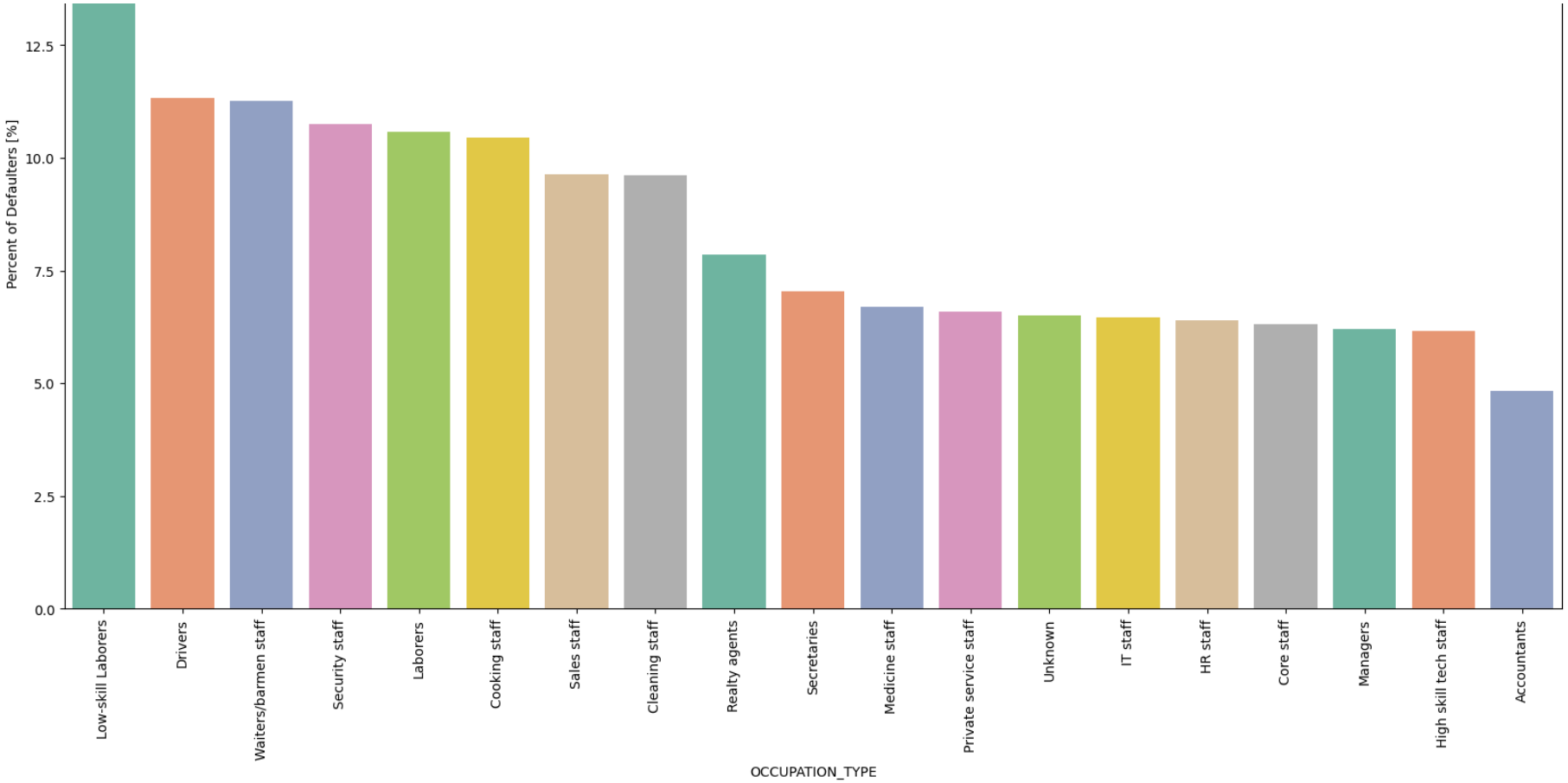
Applicant living in Region_Rating 1 has the lowest probability of defaulting, thus safer for approving loans

In [145…

```
# Analyzing Occupation Type where applicant lives based on loan repayment status

univariate_categorical("OCCUPATION_TYPE",False,True,False)
```


OCCUPATION_TYPE
Repayer
Defaulter
count
0
20000
40000
60000
80000
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

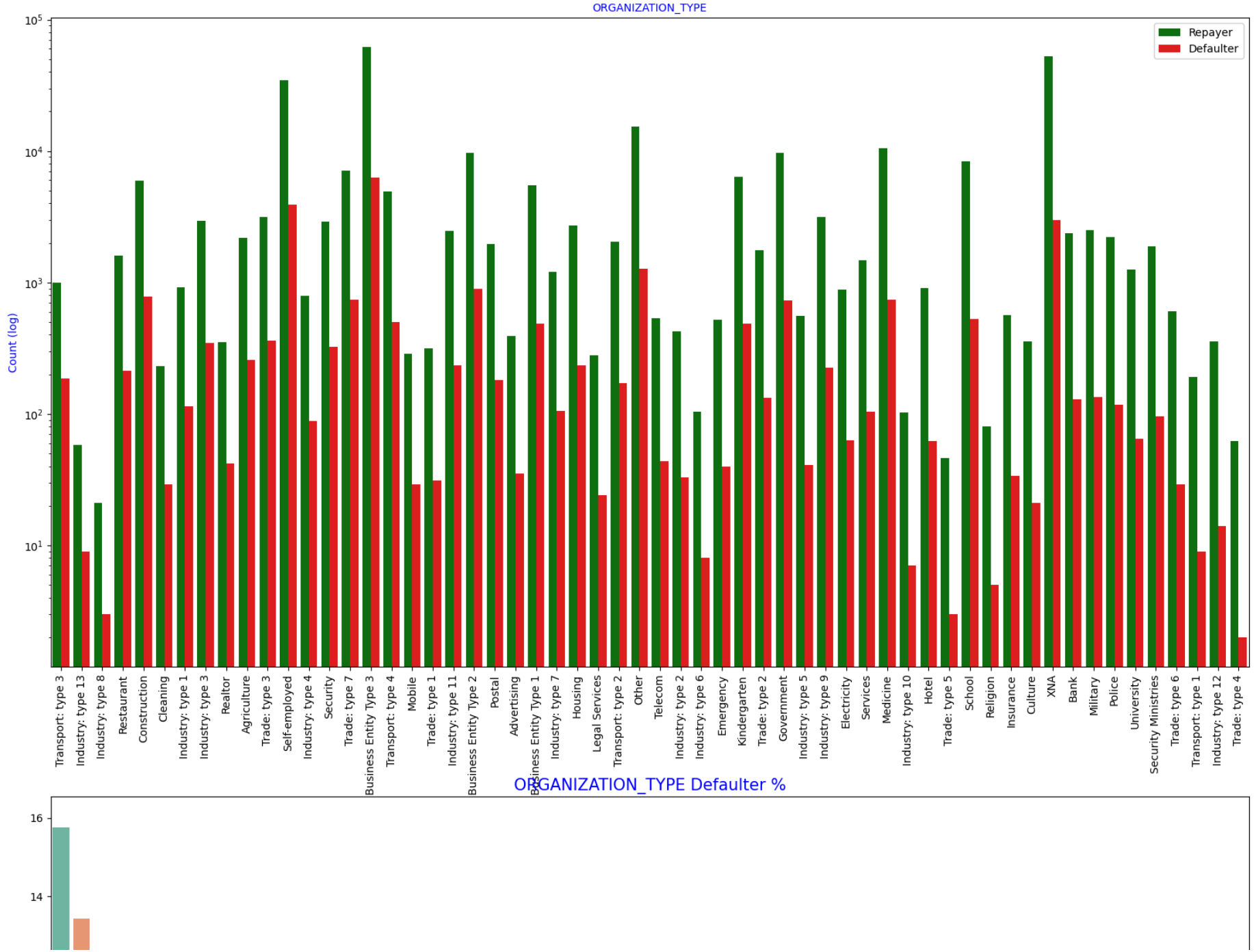


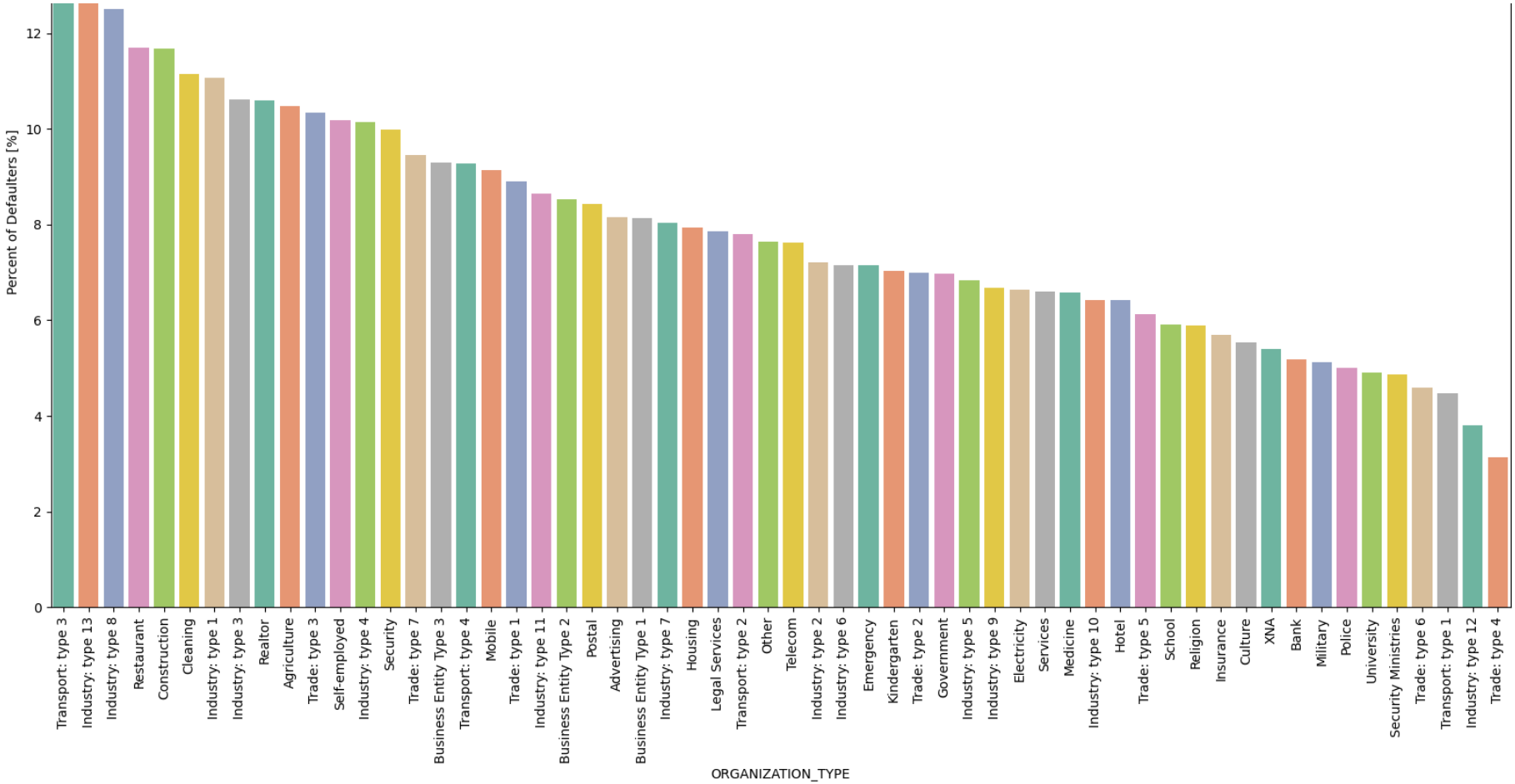
Most of the loans are taken by Laborers, followed by Sales staff. IT staff take the lowest amount of loans.

The category with highest percent of not repaid loans are Low-skill Laborers (above 17%), followed by Drivers and Waiters/barmen staff, Security staff, Laborers and Cooking staff.

In [147…

```
# Checking Loan repayment status based on Organization type

univariate_categorical("ORGANIZATION_TYPE",True,True,False)
```


ORGANIZATION_TYPE
Repayer
Defaulter
Count (log)
10^5
10^4
10^3
10^2
10^1
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 %
16
14



Organizations with highest percent of loans not repaid are Transport: type 3 (16%), Industry: type 13 (13.5%), Industry: type 8 (12.5%) and Restaurant (less than 12%). Self employed people have relative high defaulting rate, and thus should be avoided to be approved for loan or provide loan with higher interest rate to mitigate the risk of defaulting.

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

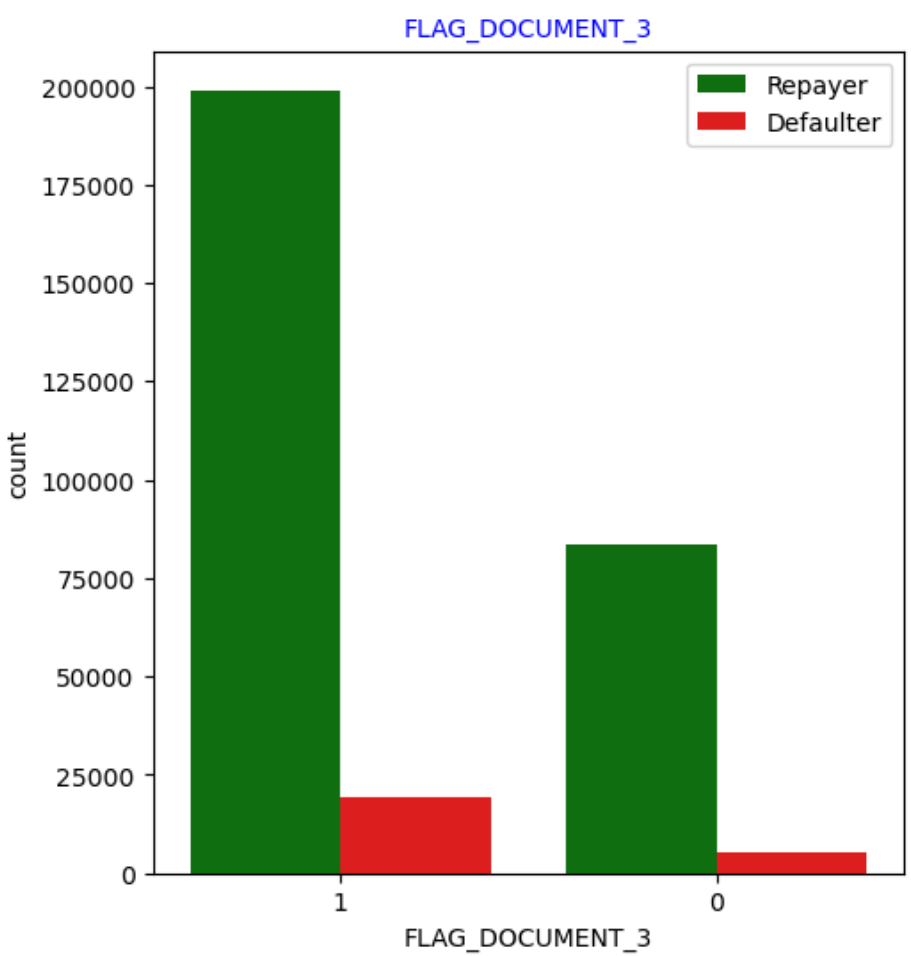
For a very high number of applications, Organization type information is unavailable(XNA) It can be seen that following category of organization type has lesser defaulters thus safer for providing loans:

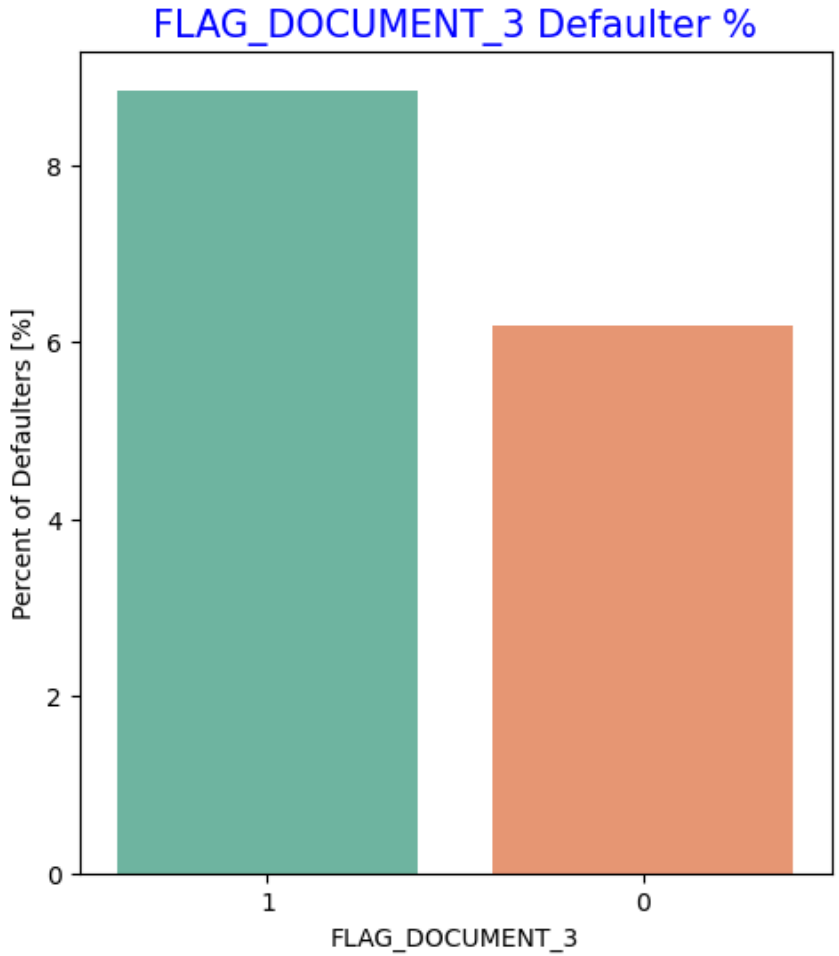
Trade Type 4 and 5

Industry type 8

In [149…

```
# Analyzing Flag_Doc_3 submission status based on loan repayment status

univariate_categorical("FLAG_DOCUMENT_3",False,False,True)
```

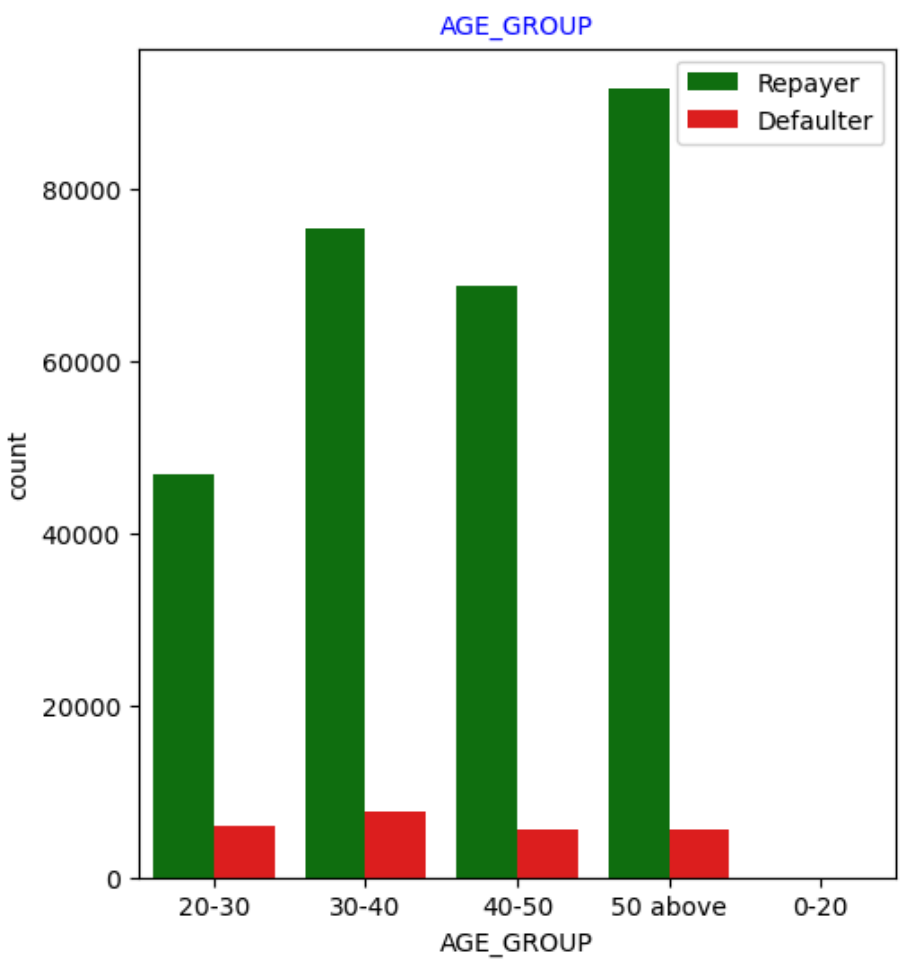


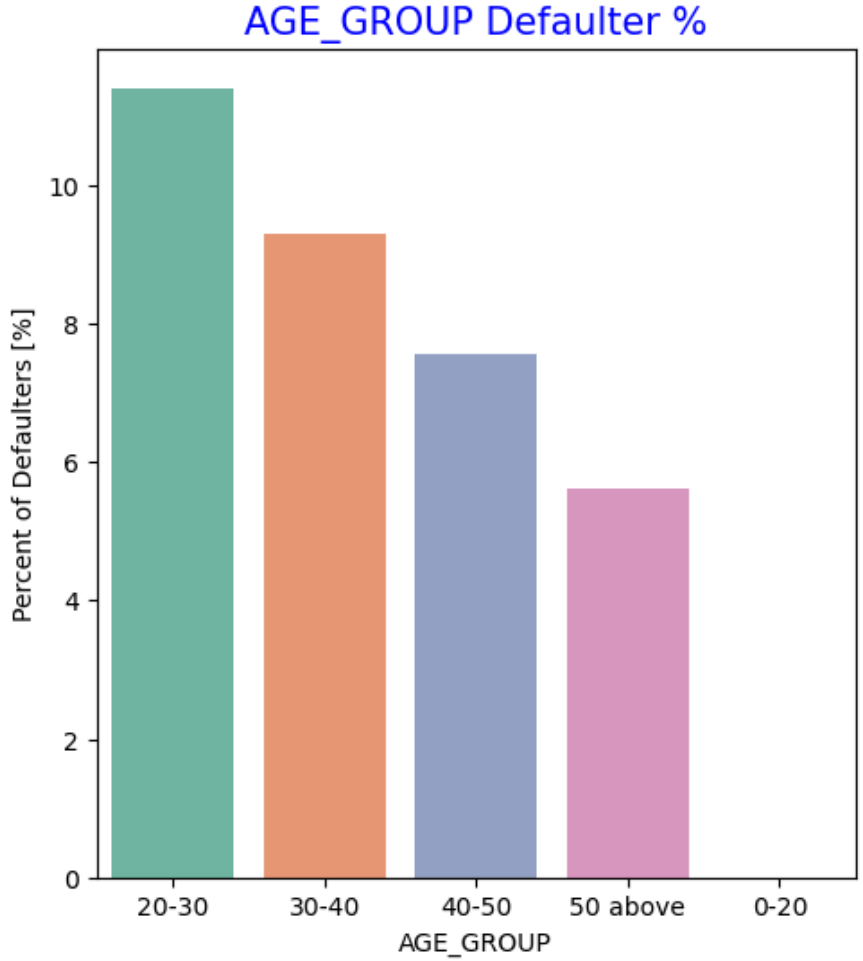


There is no significant correlation between repayers and defaulters in terms of submitting document 3 as we see even if applicants have submitted the document, they have defaulted a slightly more (9%) than who have not submitted the document (6%)

In [151…

```
# Analyzing Age Group based on loan repayment status

univariate_categorical("AGE_GROUP",False,False,True)
```



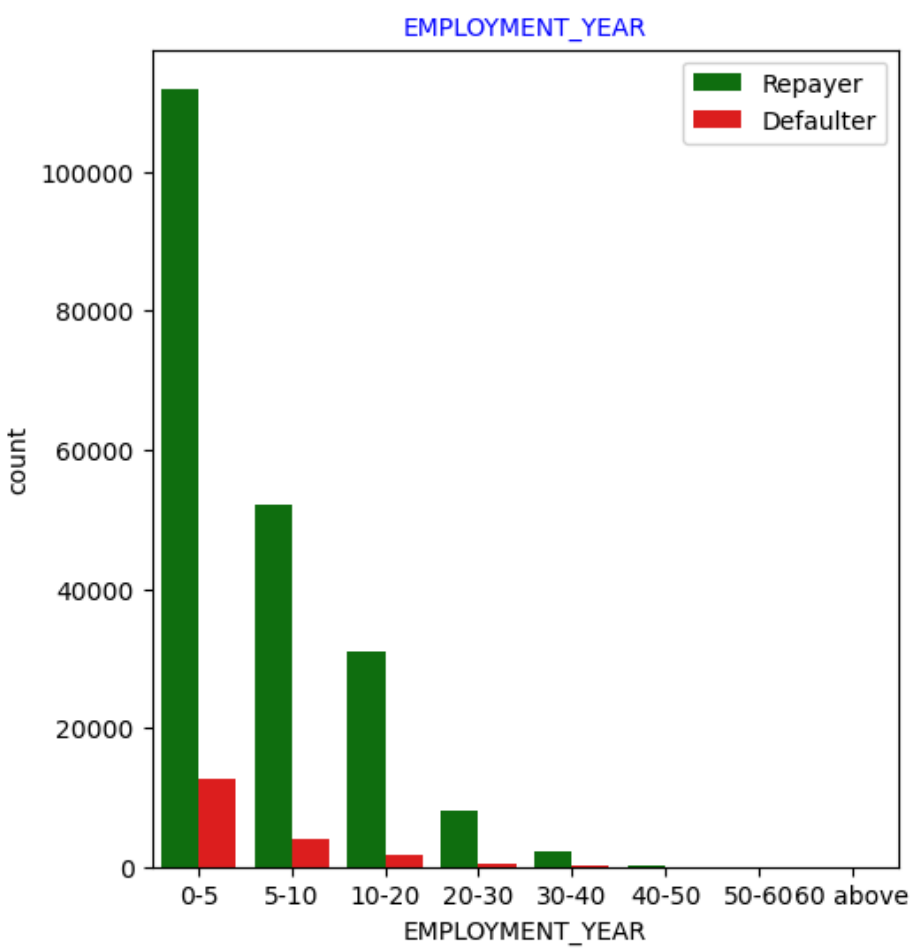


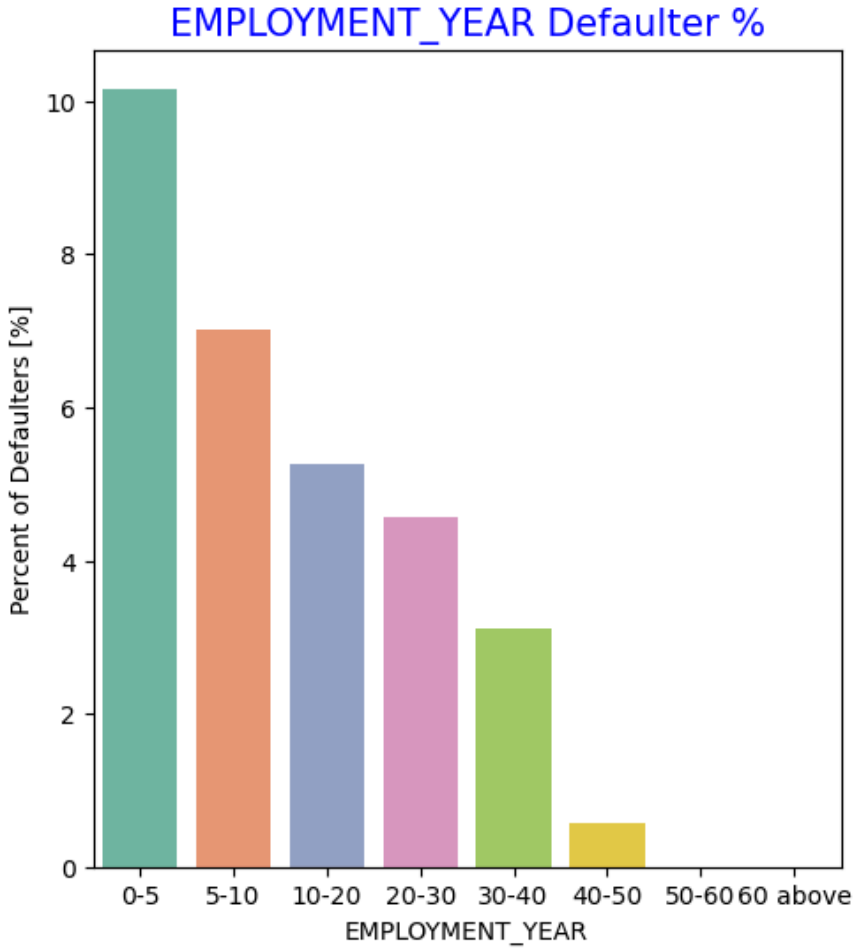
People in the age group range 20-40 have higher probability of defaulting

People above age of 50 have low probability of defailting

In [153…

```
# Analyzing Employment_Year based on loan repayment status

univariate_categorical("EMPLOYMENT_YEAR",False,False,True)
```



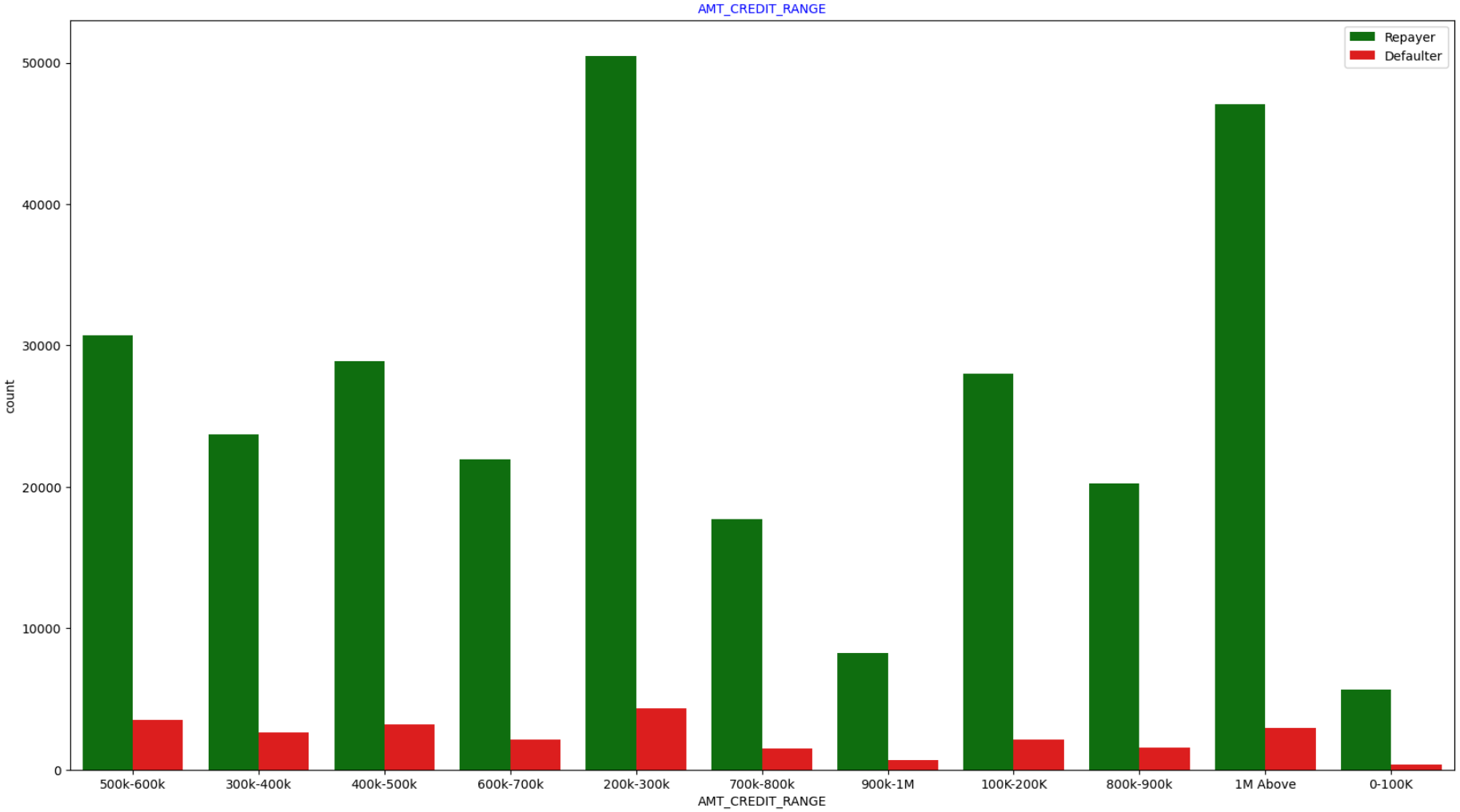


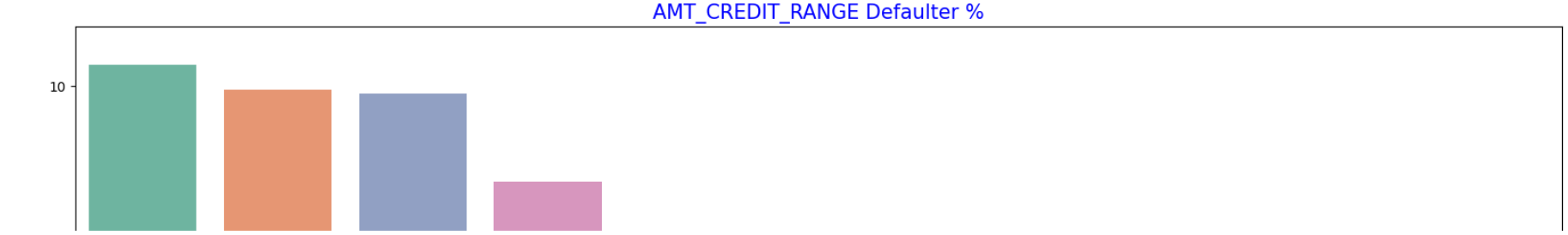
Majority of the applicants have been employeed in between 0-5 years. The defaulting rating of this group is also the highest which is 10%

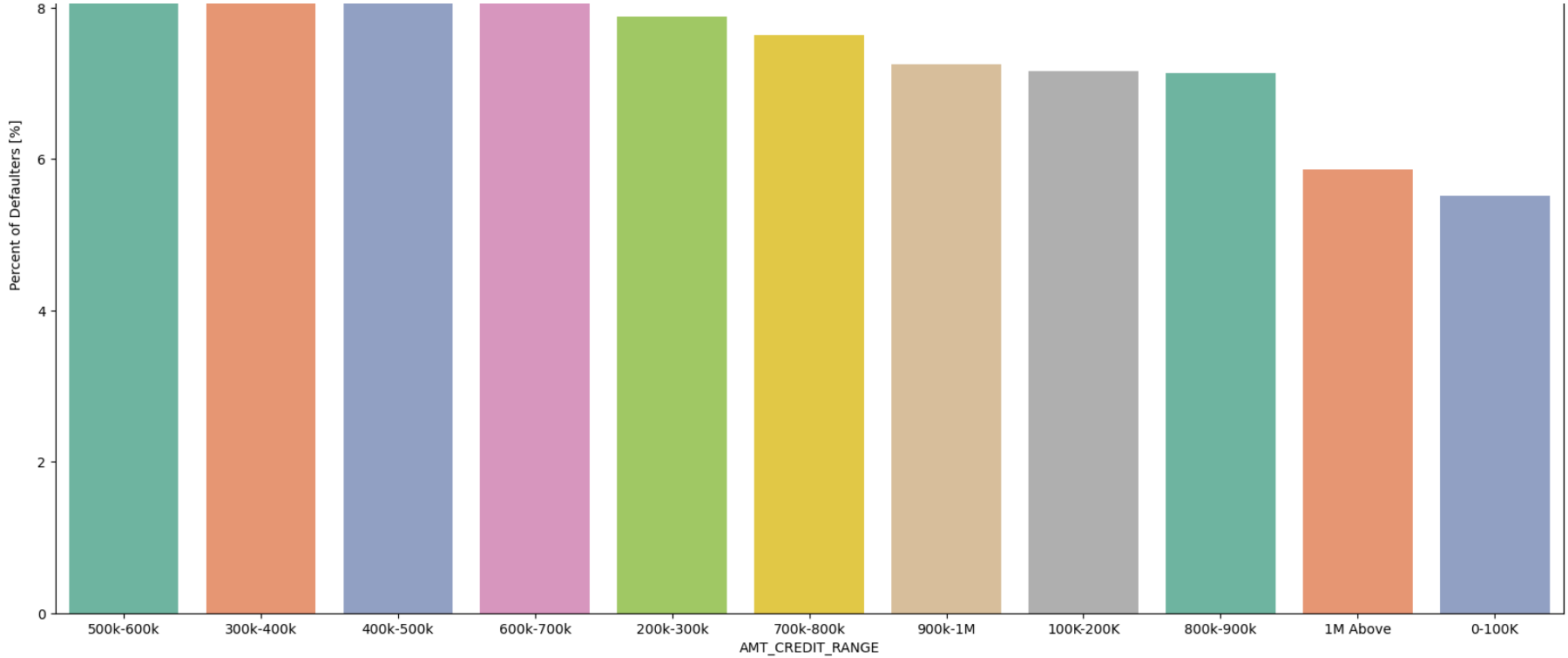
With increase of employment year, defaulting rate is gradually decreasing with people having 40+ year experience having less than 1% default rate

In [155…

```
# 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

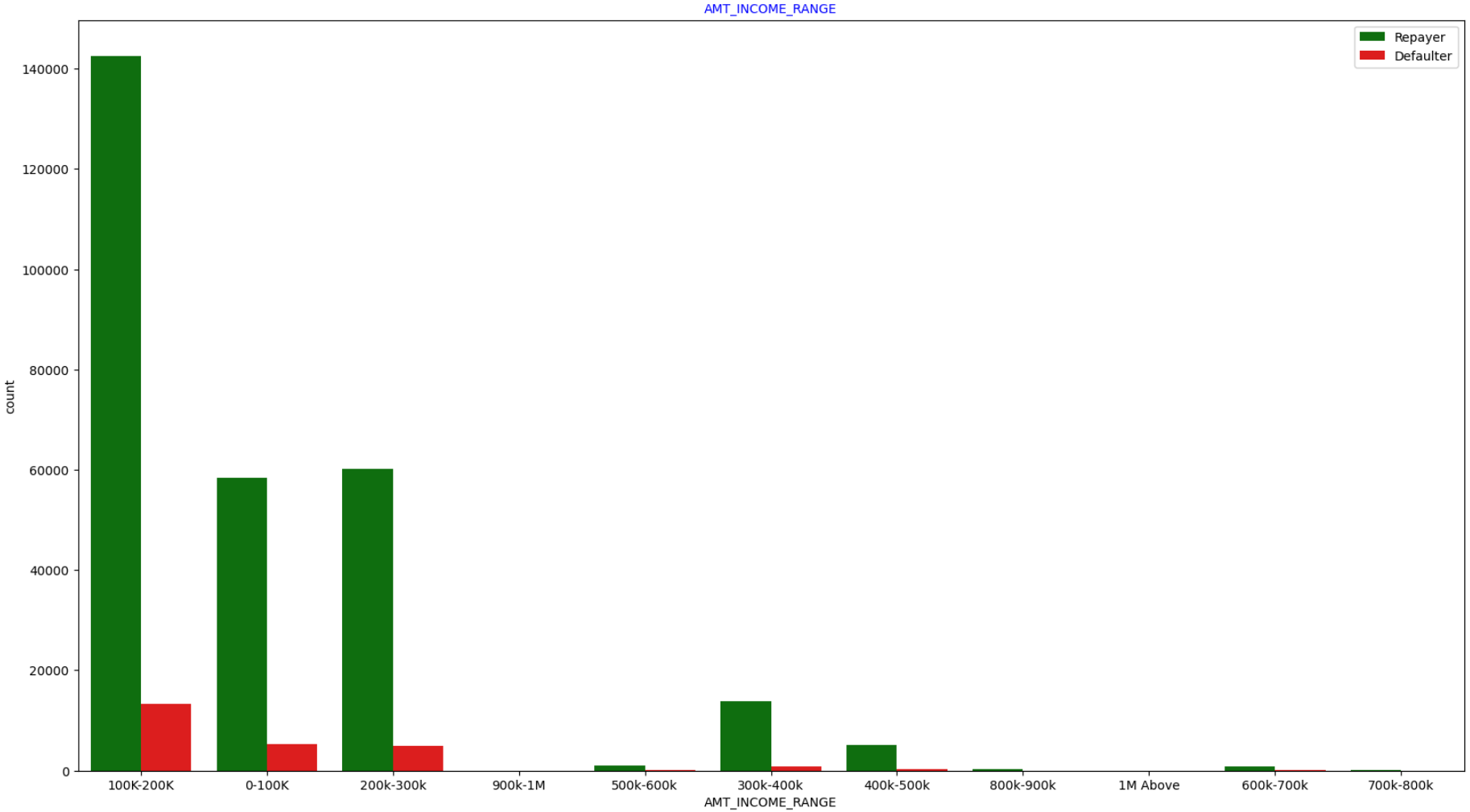

AMT_CREDIT_RANGE Defaulter %
10

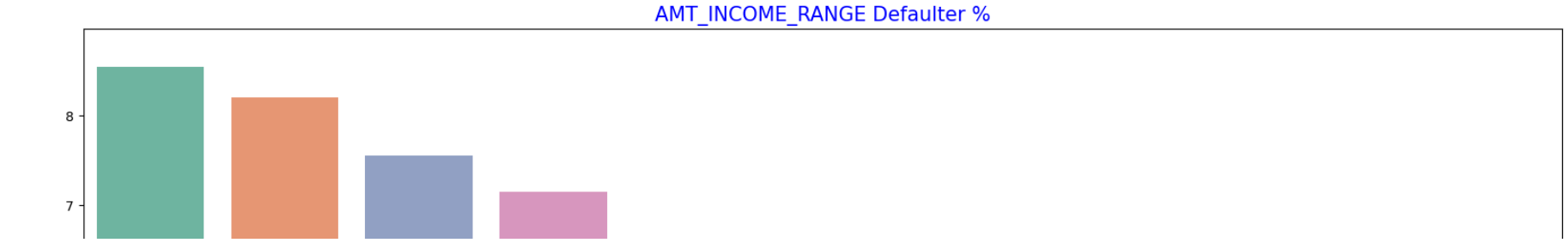


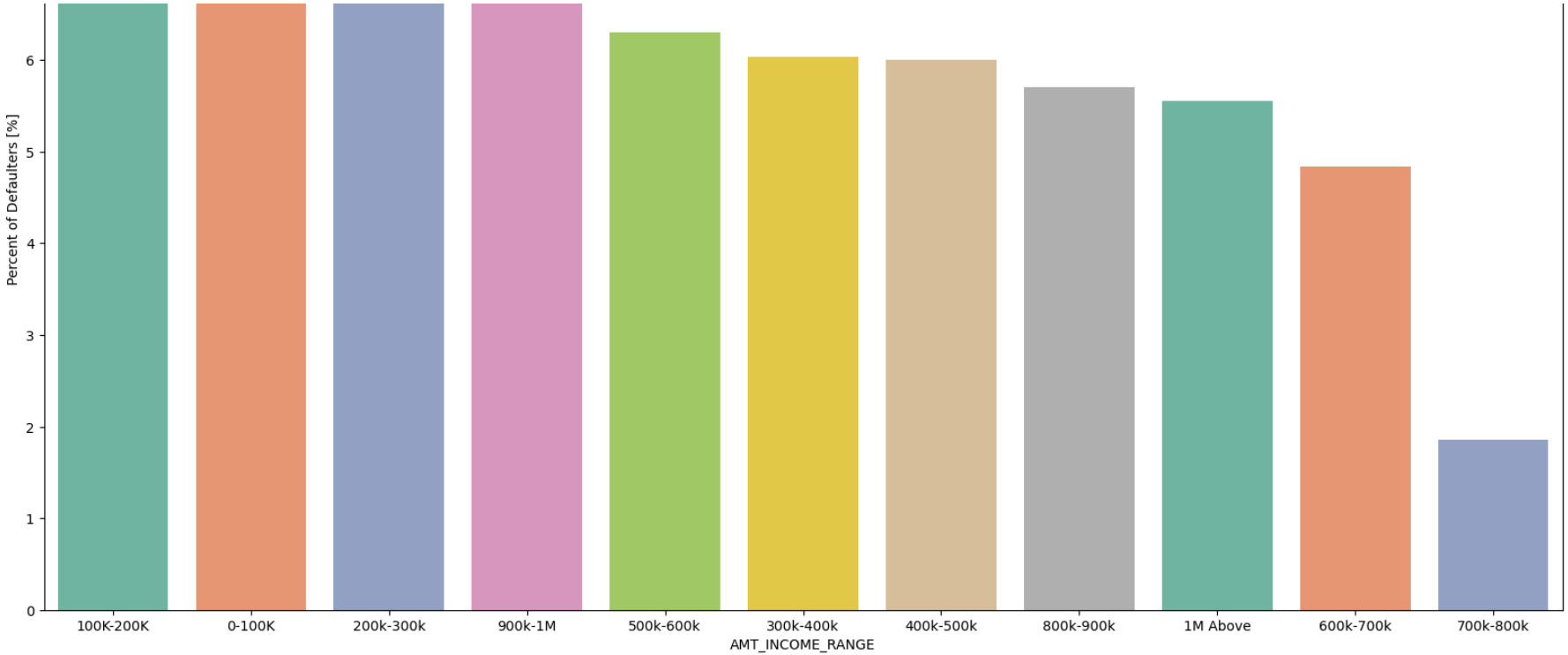
More than 80% of the loan provided are for amount less than 900,000 People who get loan for 300-600k tend to default more than others.

In [157…

```
# Analyzing Amount_Income Range based on loan repayment status

univariate_categorical("AMT_INCOME_RANGE",False,False,False)
```


AMT_INCOME_RANGE
Repayer
Defaulter
count
140000
120000
100000
80000
60000
40000
20000
0
100K-200K
0-100K
200k-300k
900k-1M
500k-600k
300k-400k
400k-500k
800k-900k
1M Above
600k-700k
700k-800k
AMT_INCOME_RANGE


AMT_INCOME_RANGE Defaulter %
8
7



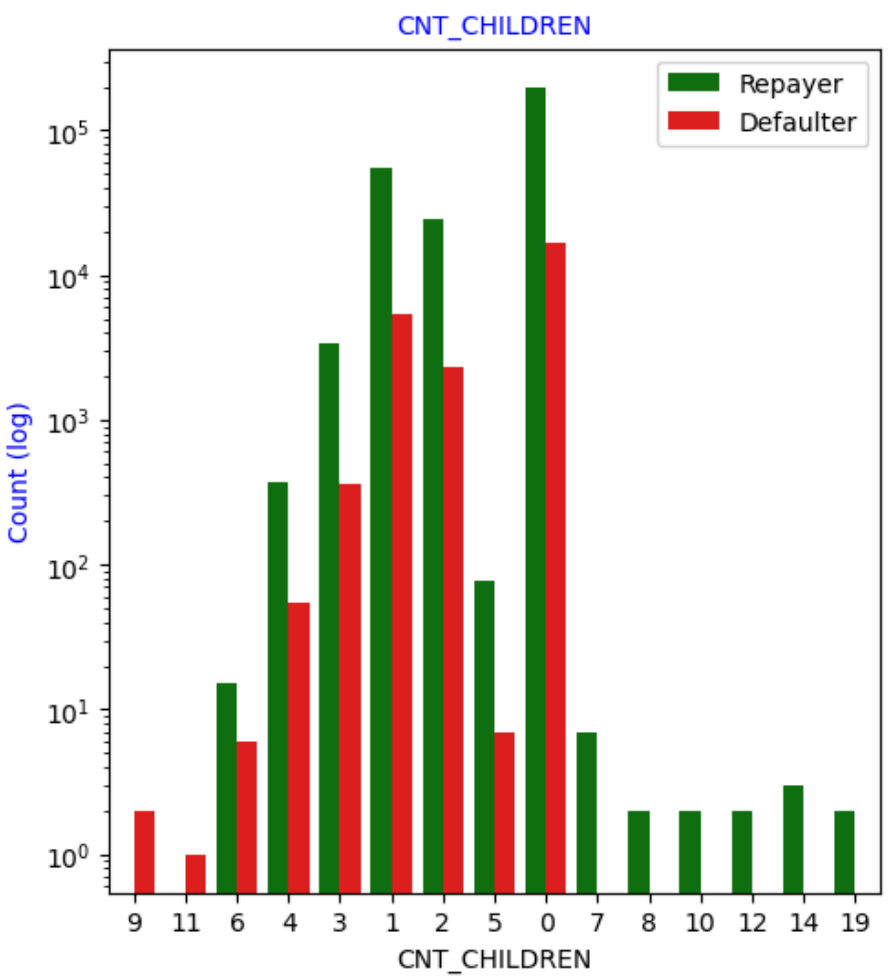
90% of the applications have Income total less than 300,000

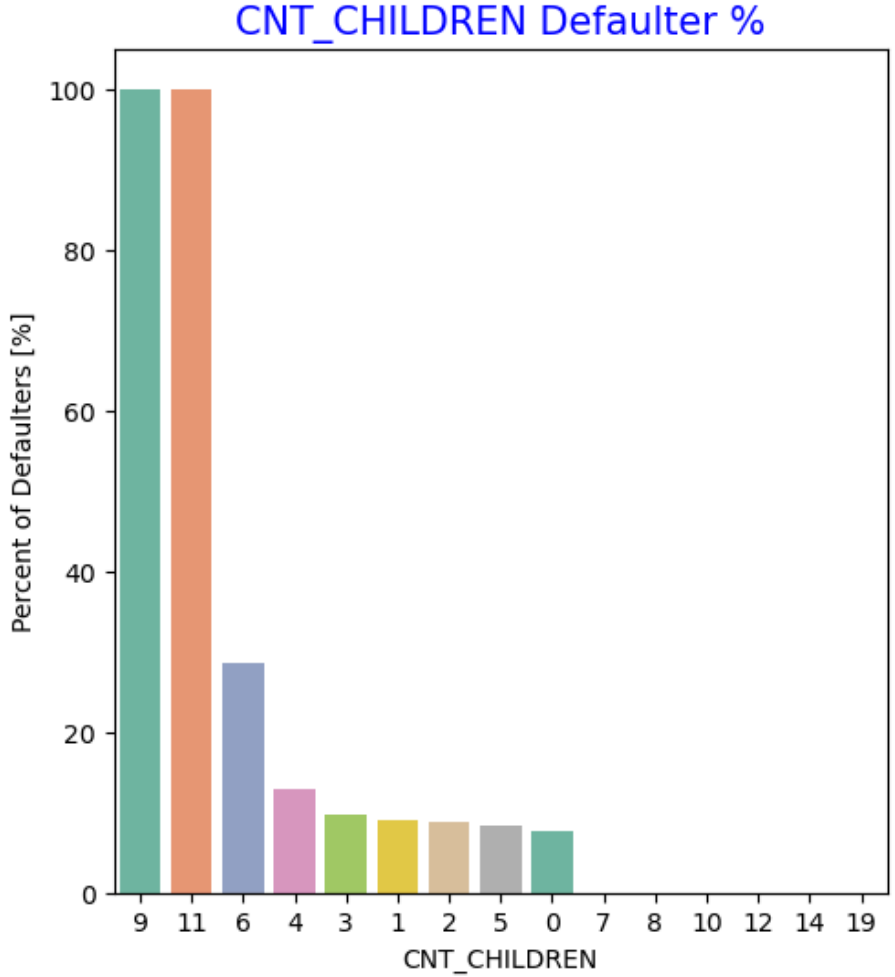
Application with Income less than 300,000 has high probability of defaulting

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

In [159…

```
# Analyzing Number of children based on loan repayment status

univariate_categorical("CNT_CHILDREN",True)
```





Most of the applicants do not have children

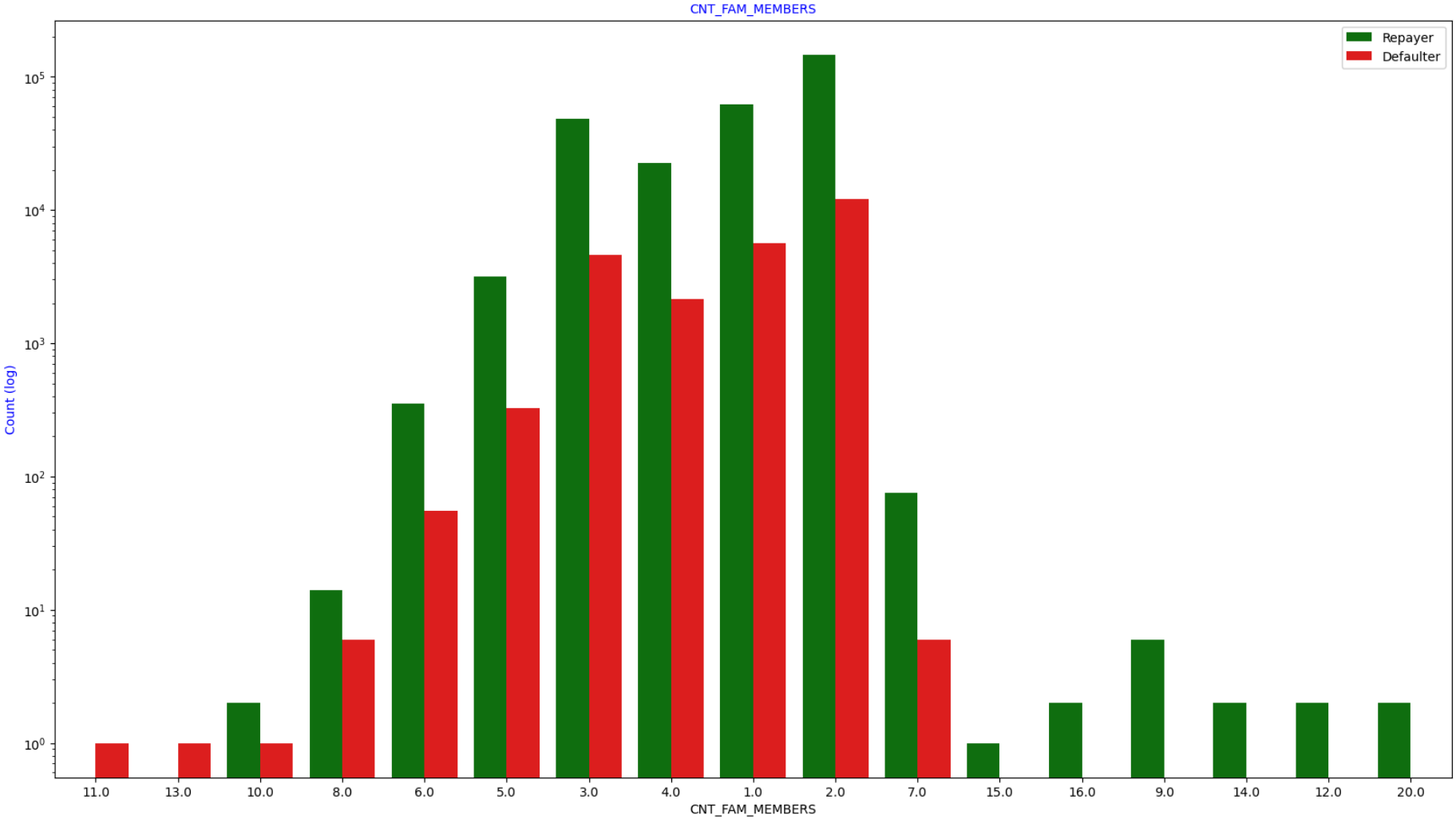
Very few clients have more than 3 children.

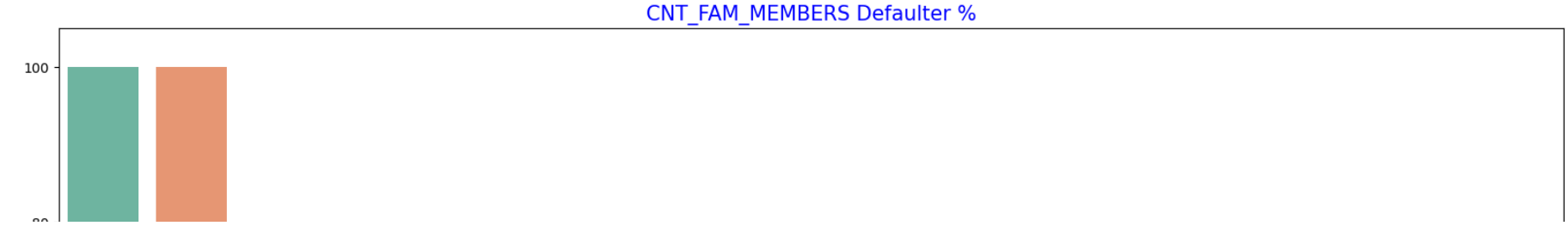
Client who have more than 4 children has a very high default rate with child count 9 and 11 showing 100% default rate

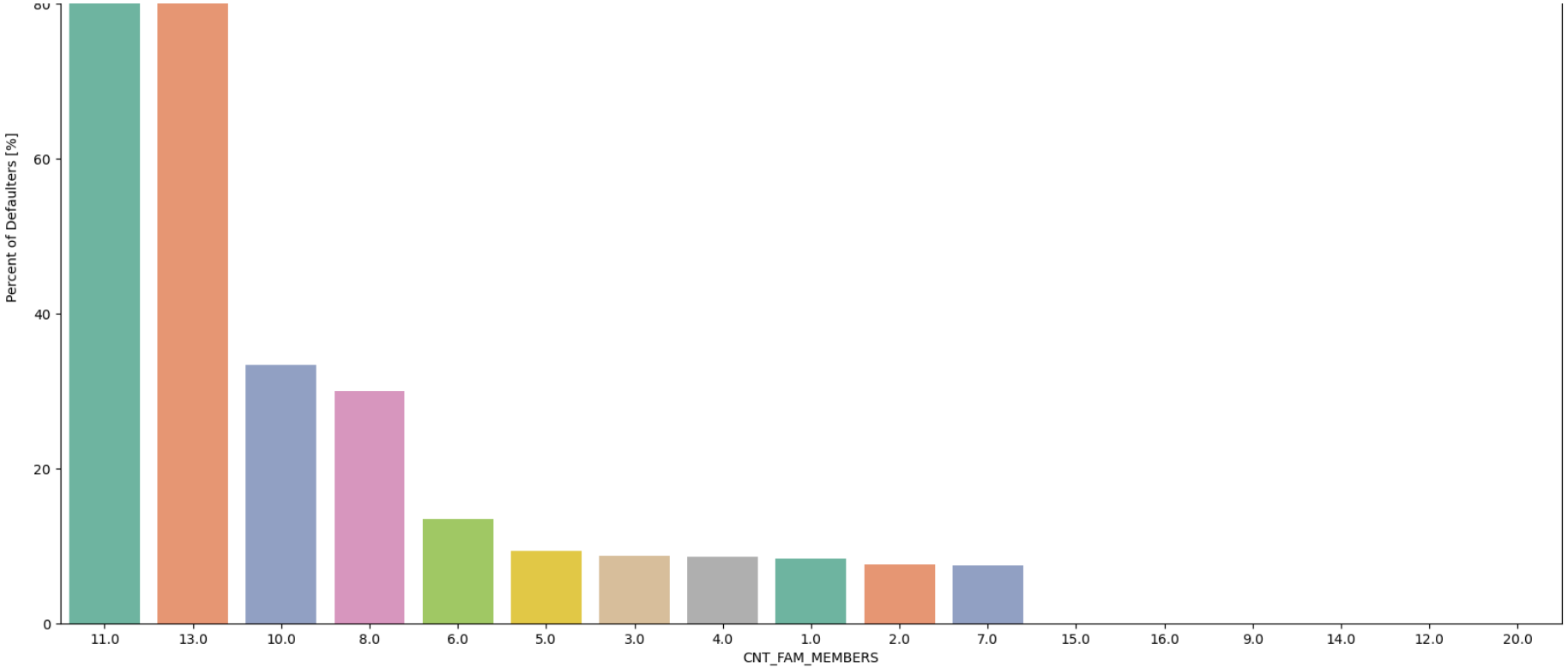
In [161…

```
# 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



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

Multivariate Analysis

In [164…

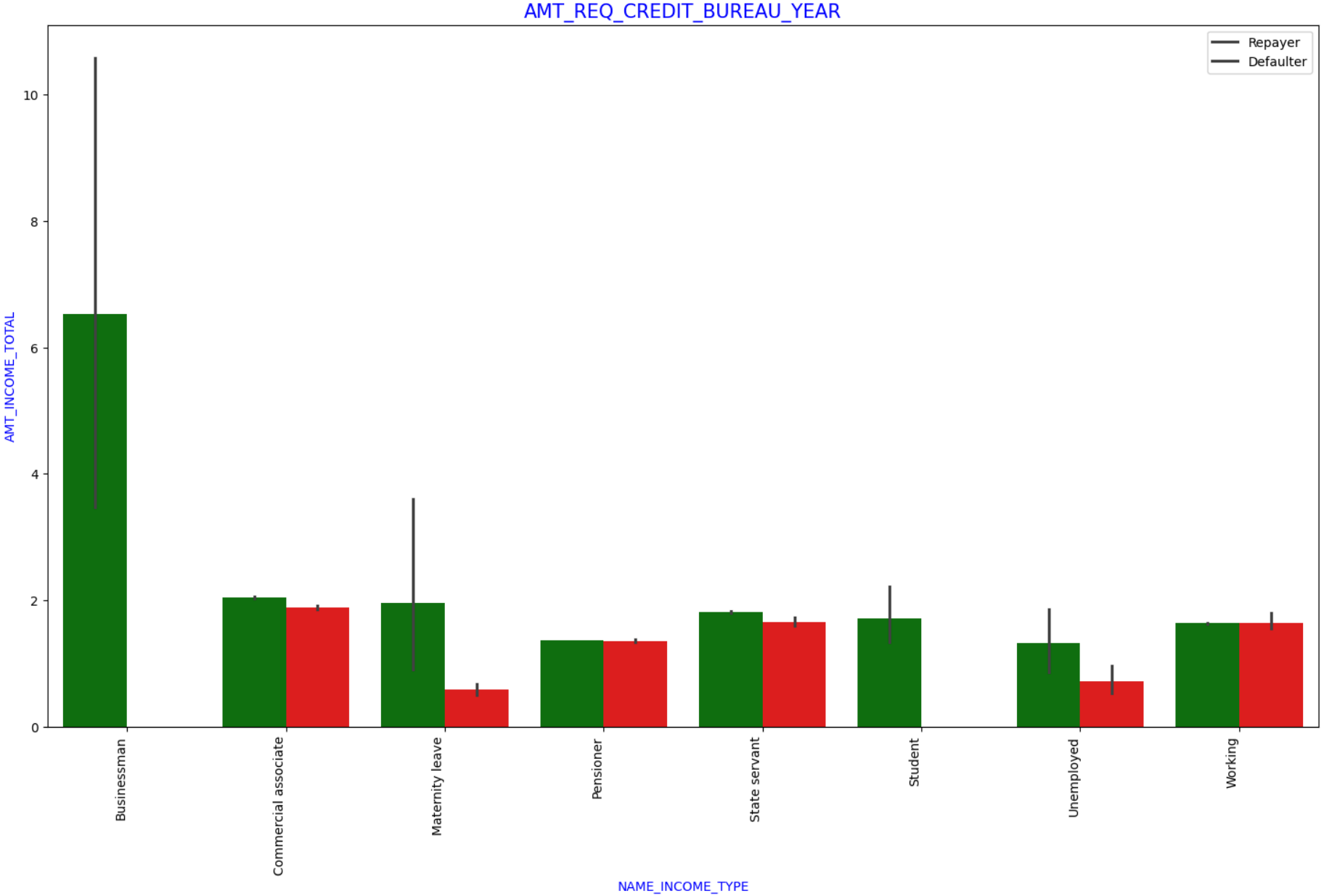
```
application.groupby('NAME_INCOME_TYPE')['AMT_INCOME_TOTAL'].describe()
```

Out[164...

NAME_INCOME_TYPE	count	mean	std	min	25%	50%	75%	max
Businessman	10.0	6.525000	6.272260	1.8000	2.250	4.9500	8.43750	22.5000
Commercial associate	71617.0	2.029553	1.479742	0.2655	1.350	1.8000	2.25000	180.0009
Maternity leave	5.0	1.404000	1.268569	0.4950	0.675	0.9000	1.35000	3.6000
Pensioner	55362.0	1.364013	0.766503	0.2565	0.900	1.1700	1.66500	22.5000
State servant	21703.0	1.797380	1.008806	0.2700	1.125	1.5750	2.25000	31.5000
Student	18.0	1.705000	1.066447	0.8100	1.125	1.5750	1.78875	5.6250
Unemployed	22.0	1.105364	0.880551	0.2655	0.540	0.7875	1.35000	3.3750
Working	158774.0	1.631699	3.075777	0.2565	1.125	1.3500	2.02500	1170.0000

In [165...

```
#Income type vs Income Amount Range

bivariate_bar("NAME_INCOME_TYPE","AMT_INCOME_TOTAL",application,"TARGET",(18,10))
```


AMT_REQ_CREDIT_BUREAU_YEAR
Repayer
Defaulter
AMT_INCOME_TOTAL
0
2
4
6
8
10
Businessman
Commercial associate
Maternity leave
Pensioner
State servant
Student
Unemployed
Working
NAME_INCOME_TYPE

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

Numerical Variable Analysis

In [168…
```
application.columns
```

Out[168…
```
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_2', 'FLAG_DOCUMENT_3',
       'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6',
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9',
       'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12',
       'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15',
       'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18',
       'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR',
       'AMT_INCOME_RANGE', 'AMT_CREDIT_RANGE', 'AGE', 'AGE_GROUP',
       'YEARS_EMPLOYED', 'EMPLOYMENT_YEAR'],
      dtype='object')
```

In [169…
```
# 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 = application.loc[application['TARGET']==0, cols_for_correlation] # Repayers
Defaulter_df = application.loc[application['TARGET']==1, cols_for_correlation] # Defaulters
```

In [170…

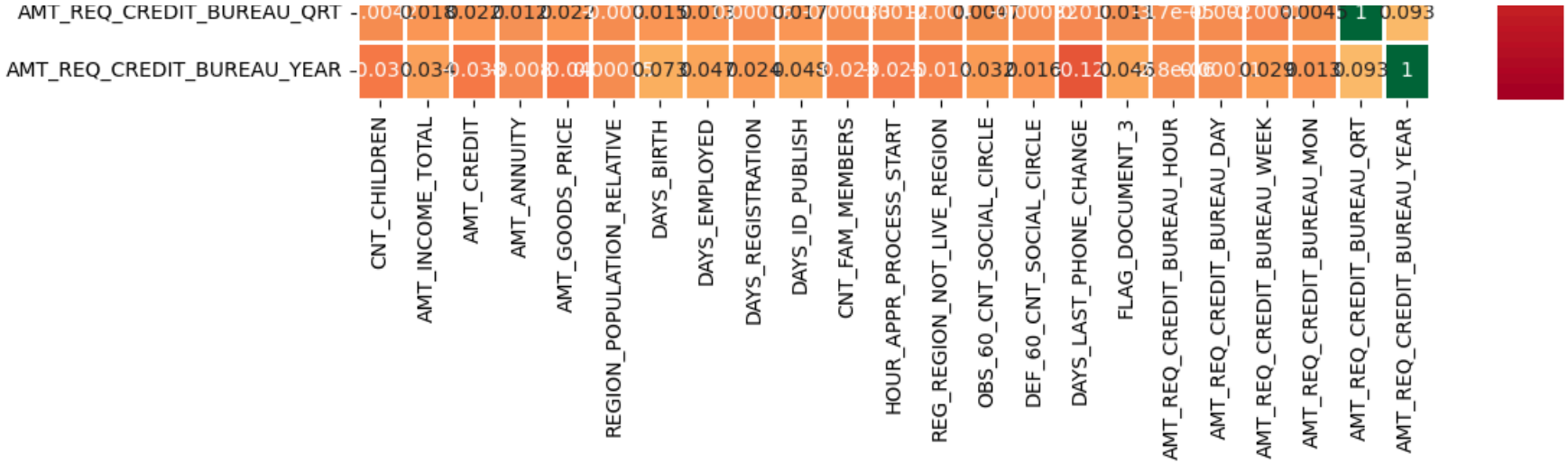
```
# Select only numeric columns
numeric_columns = Repayer_df.select_dtypes(include=['number'])

# Calculate the correlation matrix
corr_repayer = numeric_columns.corr()

# Upper triangle of the correlation matrix to avoid redundancy
corr_repayer = corr_repayer.where(np.triu(np.ones(corr_repayer.shape), k=1).astype(bool))  # Changed np.bool to bool

# Unstack the correlation matrix and reset index
corr_df_repayer = corr_repayer.unstack().reset_index()

# Rename columns
corr_df_repayer.columns = ['VAR1', 'VAR2', 'Correlation']

# Drop NaN values
corr_df_repayer.dropna(subset=["Correlation"], inplace=True)

# Take the absolute value of correlation
corr_df_repayer["Correlation"] = corr_df_repayer["Correlation"].abs()

# Sort values by correlation in descending order
corr_df_repayer.sort_values(by='Correlation', ascending=False, inplace=True)

# Show the top 10 correlations
top_10_corr_repayer = corr_df_repayer.head(10)
print(top_10_corr_repayer)
```

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

In [177…

```
# Plot the heatmap

corr_matrix = numeric_columns.corr()
fig = plt.figure(figsize=(12, 12))
ax = sns.heatmap(corr_matrix, cmap="RdYlGn", annot=True, linewidth=1)
plt.show()
```


CNT_CHILDREN
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
REGION_POPULATION_RELATIVE
DAYS_BIRTH
DAYS_EMPLOYED
DAYS_REGISTRATION
DAYS_ID_PUBLISH
CNT_FAM_MEMBERS
HOUR_APPR_PROCESS_START
REG_REGION_NOT_LIVE_REGION
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
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



Credit amount is highly correlated with amount of goods price which is same as repayers.

But the loan annuity correlation with credit amount has slightly reduced in defaulters(0.75) when compared to repayers(0.77)

We can also see that repayers have high correlation in number of days employed(0.62) when compared to defaulters(0.58).

There is a severe drop in the correlation between total income of the client and the credit amount(0.038) amongst defaulters whereas it is 0.342 among repayers.

Days_birth and number of children correlation has reduced to 0.259 in defaulters when compared to 0.337 in repayers.

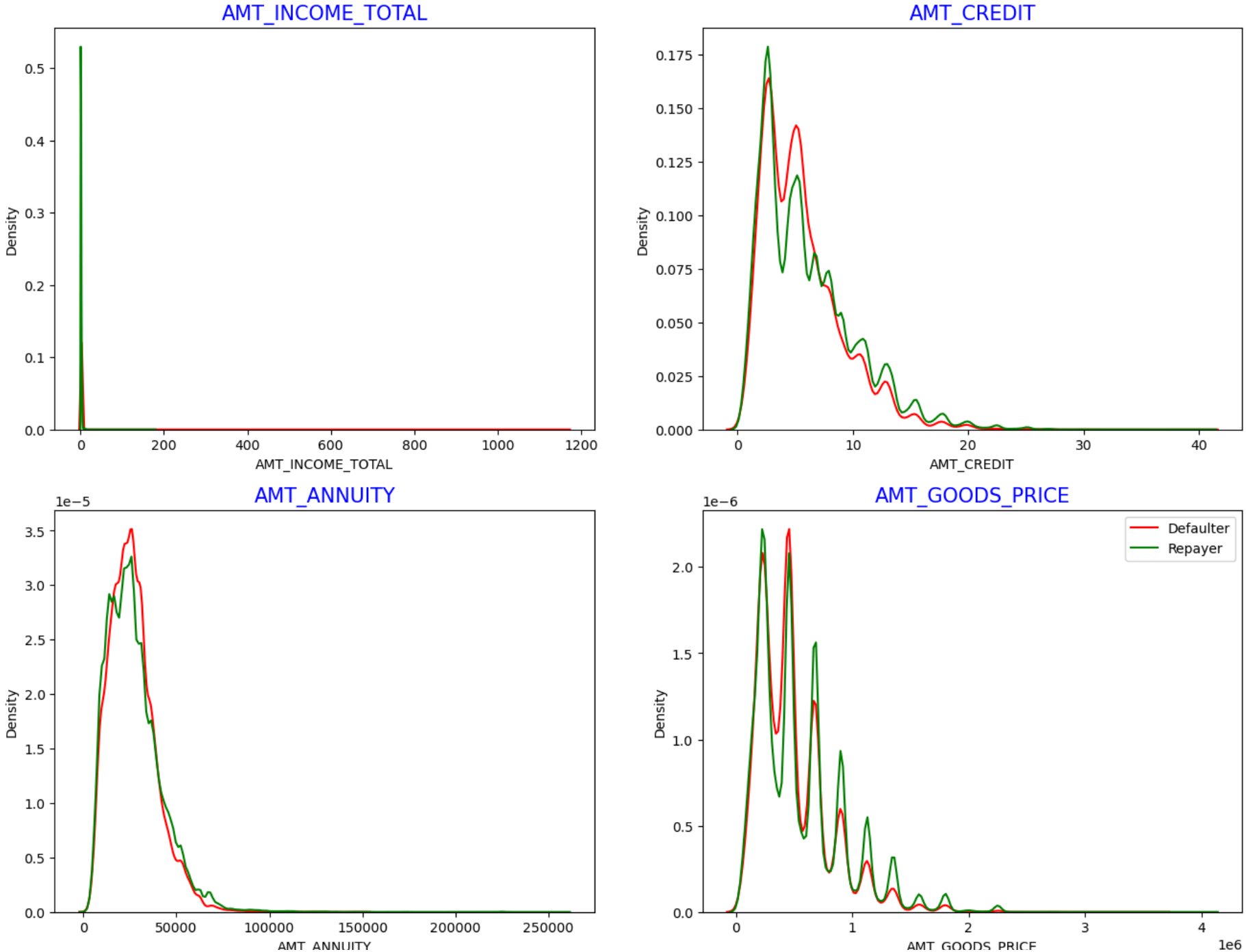
There is a slight increase in defaulted to observed count in social circle among defaulters(0.264) when compared to repayers(0.254)

Numerical UniVariate Analysis

In [181…

```
# Plotting the numerical columns related to amount as distribution plot to see density

amount = application[[ 'AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY', 'AMT_GOODS_PRICE']]

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

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

plt.legend()

plt.show()
```


AMT_INCOME_TOTAL
Density
AMT_INCOME_TOTAL
AMT_CREDIT
Density
AMT_CREDIT
AMT_ANNUITY
1e−5
Density
AMT_ANNUITY
AMT_GOODS_PRICE
1e−6
Defaulter
Repayer
Density
AMT_GOODS_PRICE
1e6

Most no of loans are given for goods price below 10 lakhs

Most people pay annuity below 50000 for the credit loan

Credit amount of the loan is mostly less then 10 lakhs

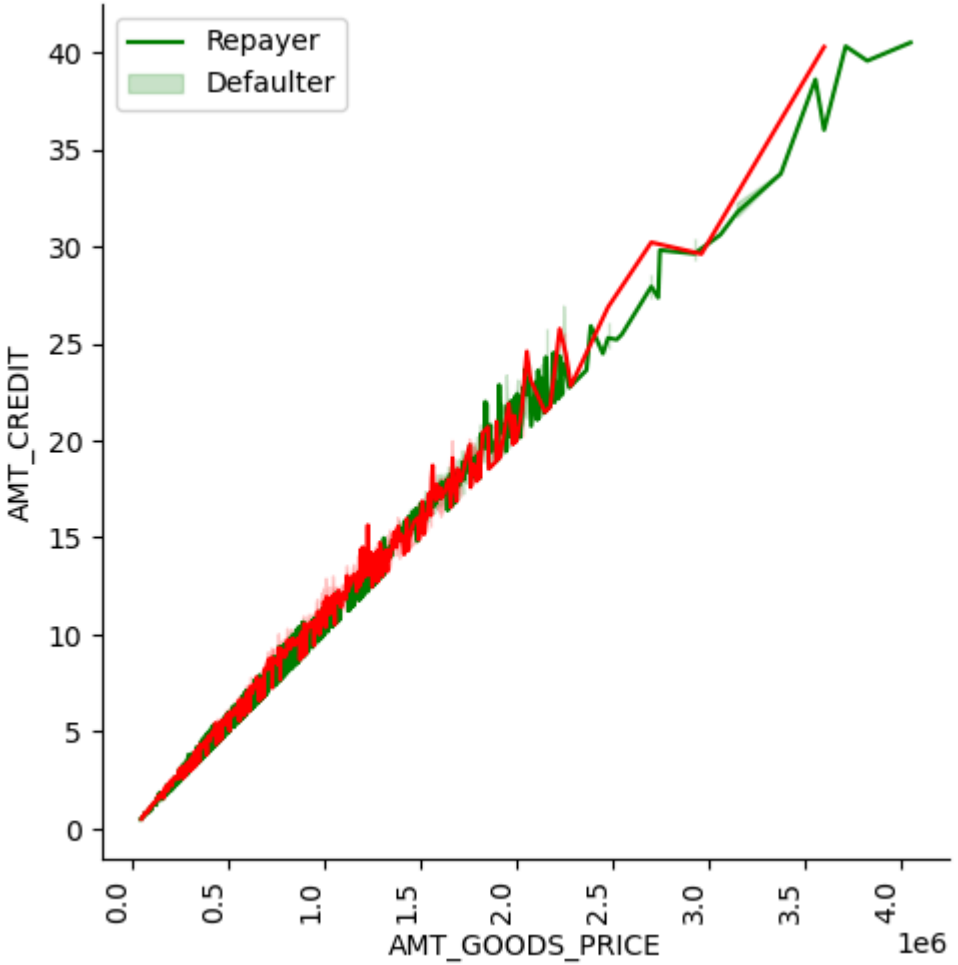
The repayers and defaulters distribution overlap in all the plots and hence we cannot use any of these variables in isolation to make a decision

Numerical BiVariate Analysis

In [203…

```
# Checking the relationship between Goods price and credit and comparing with loan repayment status

bivariate_rel('AMT_GOODS_PRICE','AMT_CREDIT',application,'TARGET','line',['g','r'], False,(15,6))
```

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

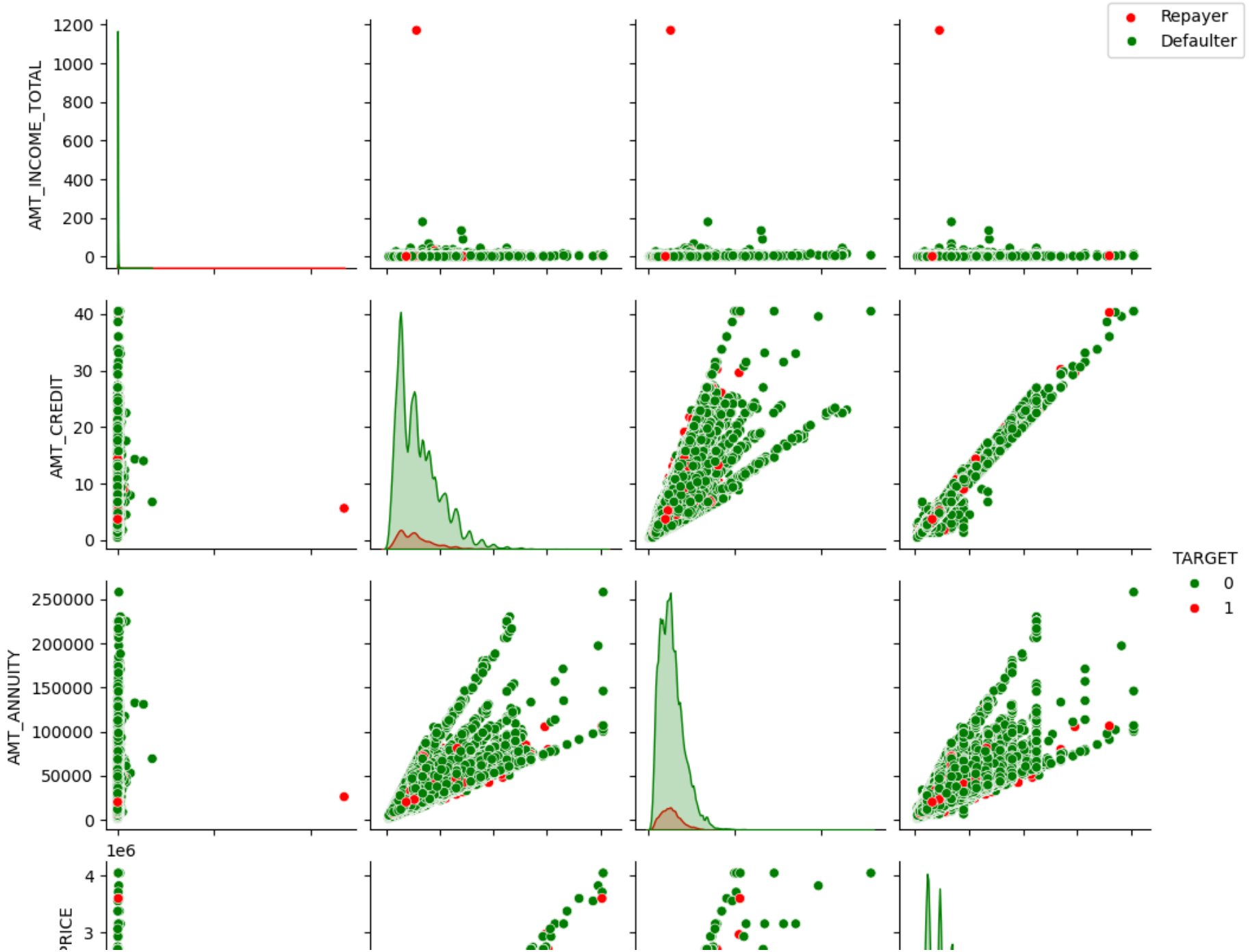


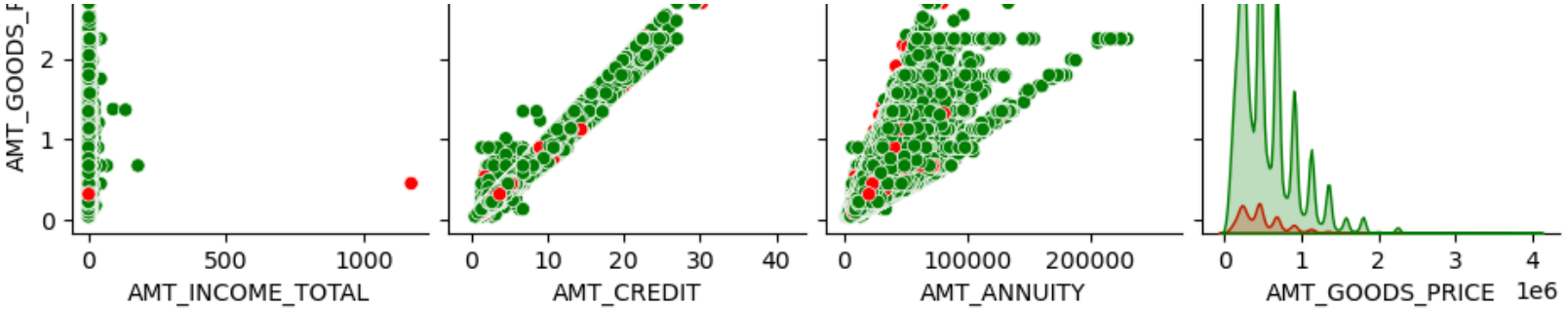
When the credit amount goes beyond 3M, there is an increase in defaulters.

In [208…

```
# Plotting pairplot between amount variable to draw reference against loan repayment status

amount = application[[ 'AMT_INCOME_TOTAL','AMT_CREDIT',
                       'AMT_ANNUITY', 'AMT_GOODS_PRICE','TARGET']]
amount = amount[(amount["AMT_GOODS_PRICE"].notnull()) & (amount["AMT_ANNUITY"].notnull())]
ax= sns.pairplot(amount,hue="TARGET",palette=["g","r"])
```

```
ax.fig.legend(labels=['Repayer','Defaulter'])
plt.show()
```


Repayer
Defaulter
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
1e6
TARGET
0
1



When amt_annuity >15000 amt_goods_price> 3M, there is a lesser chance of defaulters

AMT_CREDIT and AMT_GOODS_PRICE are highly correlated as based on the scatterplot where most of the data are consolidated in form of a line

There are very less defaulters for AMT_CREDIT >3M

Inferences related to distribution plot has been already mentioned in previous distplot graphs inferences section

Merged DataFrame Analysis

In [211…

```
#merge both the dataframe on SK_ID_CURR with Inner Joins

loan_process_df = pd.merge(application, previous_DF, how='inner', on='SK_ID_CURR')
loan_process_df.head()
```

Out[211…

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_
0	100002	1	Cash loans	M	N	Y	0	
1	100003	0	Cash loans	F	N	N	0	
2	100003	0	Cash loans	F	N	N	0	
3	100004	0	Revolving loans	M	Y	Y	0	
4	100006	0	Cash loans	F	N	Y	0	

5 rows × 93 columns

In [212…

```
#Checking the details of the merged dataframe

loan_process_df.shape
```

Out[212…

```
(887347, 93)
```

In [213…

```
# Checking the element count of the dataframe

loan_process_df.size
```

Out[213…

```
82523271
```

In [214…

```
# checking the columns and column types of the dataframe

loan_process_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 887347 entries, 0 to 887346
Data columns (total 93 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   SK_ID_CURR                   887347 non-null  int64
 1   TARGET                       887347 non-null  int64
 2   NAME_CONTRACT_TYPE_x         887347 non-null  category
 3   CODE_GENDER                  887347 non-null  category
 4   FLAG_OWN_CAR                 887347 non-null  category
 5   FLAG_OWN_REALTY              887347 non-null  category
 6   CNT_CHILDREN                 887347 non-null  int64
 7   AMT_INCOME_TOTAL             887347 non-null  float64
 8   AMT_CREDIT_x                 887347 non-null  float64
 9   AMT_ANNUITY_x                887287 non-null  float64
 10  AMT_GOODS_PRICE_x            886608 non-null  float64
 11  NAME_TYPE_SUITE              887347 non-null  category
 12  NAME_INCOME_TYPE             887347 non-null  category
 13  NAME_EDUCATION_TYPE          887347 non-null  category
 14  NAME_FAMILY_STATUS           887347 non-null  category
 15  NAME_HOUSING_TYPE            887347 non-null  category
 16  REGION_POPULATION_RELATIVE   887347 non-null  float64
 17  DAYS_BIRTH                   887347 non-null  int64
 18  DAYS_EMPLOYED                887347 non-null  int64
 19  DAYS_REGISTRATION            887347 non-null  float64
 20  DAYS_ID_PUBLISH              887347 non-null  int64
 21  OCCUPATION_TYPE              887347 non-null  category
 22  CNT_FAM_MEMBERS              887347 non-null  float64
 23  REGION_RATING_CLIENT         887347 non-null  category
 24  REGION_RATING_CLIENT_W_CITY  887347 non-null  category
 25  WEEKDAY_APPR_PROCESS_START   887347 non-null  category
 26  HOUR_APPR_PROCESS_START      887347 non-null  int64
 27  REG_REGION_NOT_LIVE_REGION   887347 non-null  int64
 28  REG_REGION_NOT_WORK_REGION   887347 non-null  category
 29  LIVE_REGION_NOT_WORK_REGION  887347 non-null  category
 30  REG_CITY_NOT_LIVE_CITY       887347 non-null  category
 31  REG_CITY_NOT_WORK_CITY       887347 non-null  category
 32  LIVE_CITY_NOT_WORK_CITY      887347 non-null  category
 33  ORGANIZATION_TYPE            887347 non-null  category
 34  OBS_30_CNT_SOCIAL_CIRCLE     885364 non-null  float64
 35  DEF_30_CNT_SOCIAL_CIRCLE     885364 non-null  float64
```

```
36  OBS_60_CNT_SOCIAL_CIRCLE     885364 non-null  float64
37  DEF_60_CNT_SOCIAL_CIRCLE     885364 non-null  float64
38  DAYS_LAST_PHONE_CHANGE       887347 non-null  float64
39  FLAG_DOCUMENT_2              887347 non-null  int64
40  FLAG_DOCUMENT_3              887347 non-null  int64
41  FLAG_DOCUMENT_4              887347 non-null  int64
42  FLAG_DOCUMENT_5              887347 non-null  int64
43  FLAG_DOCUMENT_6              887347 non-null  int64
44  FLAG_DOCUMENT_7              887347 non-null  int64
45  FLAG_DOCUMENT_8              887347 non-null  int64
46  FLAG_DOCUMENT_9              887347 non-null  int64
47  FLAG_DOCUMENT_10             887347 non-null  int64
48  FLAG_DOCUMENT_11             887347 non-null  int64
49  FLAG_DOCUMENT_12             887347 non-null  int64
50  FLAG_DOCUMENT_13             887347 non-null  int64
51  FLAG_DOCUMENT_14             887347 non-null  int64
52  FLAG_DOCUMENT_15             887347 non-null  int64
53  FLAG_DOCUMENT_16             887347 non-null  int64
54  FLAG_DOCUMENT_17             887347 non-null  int64
55  FLAG_DOCUMENT_18             887347 non-null  int64
56  FLAG_DOCUMENT_19             887347 non-null  int64
57  FLAG_DOCUMENT_20             887347 non-null  int64
58  FLAG_DOCUMENT_21             887347 non-null  int64
59  AMT_REQ_CREDIT_BUREAU_HOUR   887347 non-null  float64
60  AMT_REQ_CREDIT_BUREAU_DAY    887347 non-null  float64
61  AMT_REQ_CREDIT_BUREAU_WEEK   887347 non-null  float64
62  AMT_REQ_CREDIT_BUREAU_MON    887347 non-null  float64
63  AMT_REQ_CREDIT_BUREAU_QRT    887347 non-null  float64
64  AMT_REQ_CREDIT_BUREAU_YEAR   887347 non-null  float64
65  AMT_INCOME_RANGE             886908 non-null  category
66  AMT_CREDIT_RANGE             887347 non-null  category
67  AGE                          887347 non-null  int64
68  AGE_GROUP                    887347 non-null  category
69  YEARS_EMPLOYED               887347 non-null  int64
70  EMPLOYMENT_YEAR              647974 non-null  category
71  SK_ID_PREV                   887347 non-null  int64
72  NAME_CONTRACT_TYPE_y         887347 non-null  category
73  AMT_ANNUITY_y                887347 non-null  float64
74  AMT_APPLICATION              887347 non-null  float64
75  AMT_CREDIT_y                 887347 non-null  float64
76  AMT_GOODS_PRICE_y            887347 non-null  float64
```

```
 77  NAME_CASH_LOAN_PURPOSE       887347 non-null  category
 78  NAME_CONTRACT_STATUS         887347 non-null  category
 79  DAYS_DECISION                887347 non-null  int64
 80  NAME_PAYMENT_TYPE            887347 non-null  category
 81  CODE_REJECT_REASON           887347 non-null  category
 82  NAME_CLIENT_TYPE             887347 non-null  category
 83  NAME_GOODS_CATEGORY          887347 non-null  category
 84  NAME_PORTFOLIO               887347 non-null  category
 85  NAME_PRODUCT_TYPE            887347 non-null  category
 86  CHANNEL_TYPE                 887347 non-null  category
 87  SELLERPLACE_AREA             887347 non-null  int64
 88  NAME_SELLER_INDUSTRY         887347 non-null  category
 89  CNT_PAYMENT                  887347 non-null  float64
 90  NAME_YIELD_GROUP             887347 non-null  category
 91  PRODUCT_COMBINATION          887143 non-null  category
 92  DAYS_DECISION_GROUP          887347 non-null  category
dtypes: category(37), float64(23), int64(33)
memory usage: 410.4 MB
```

In [215…

```
# Checking merged dataframe numerical columns statistics

loan_process_df.describe()
```

Out[215...

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT_x	AMT_ANNUITY_x	AMT_GOODS_PRICE_x	REGI
count	887347.000000	887347.000000	887347.000000	887347.000000	887347.000000	887287.000000	8.866080e+05	
mean	278573.355245	0.086533	0.404863	1.735124	5.875341	27014.933342	5.276874e+05	
std	102862.966629	0.281150	0.716925	2.388415	3.851740	13958.001832	3.533979e+05	
min	100002.000000	0.000000	0.000000	0.256500	0.450000	1615.500000	4.050000e+04	
25%	189427.000000	0.000000	0.000000	1.125000	2.700000	16807.500000	2.385000e+05	
50%	279118.000000	0.000000	0.000000	1.575000	5.084955	24907.500000	4.500000e+05	
75%	367786.000000	0.000000	1.000000	2.070000	8.086500	34537.500000	6.795000e+05	
max	456255.000000	1.000000	19.000000	1170.000000	39.562740	220297.500000	3.825000e+06	

8 rows × 56 columns

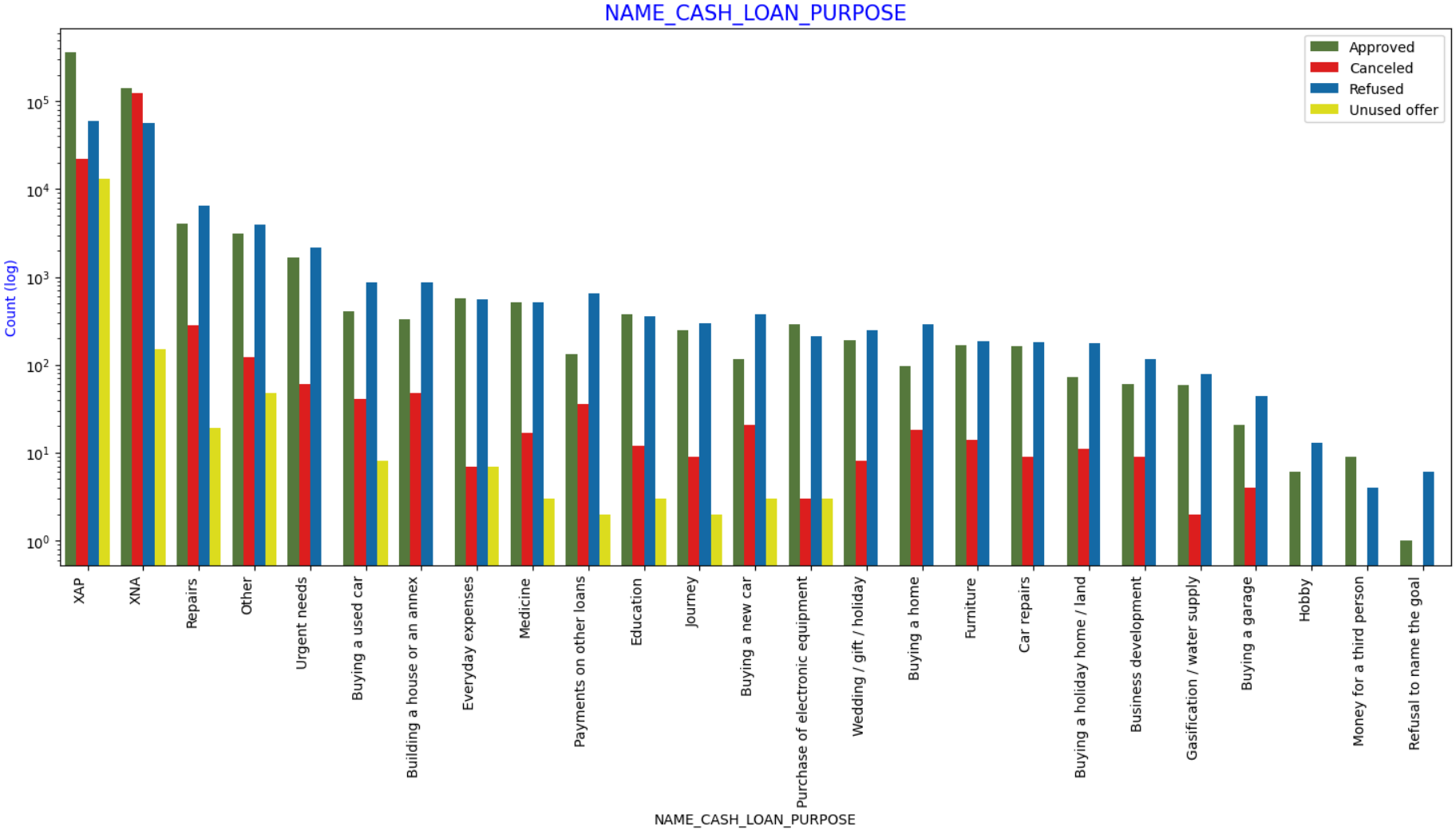
In [243...

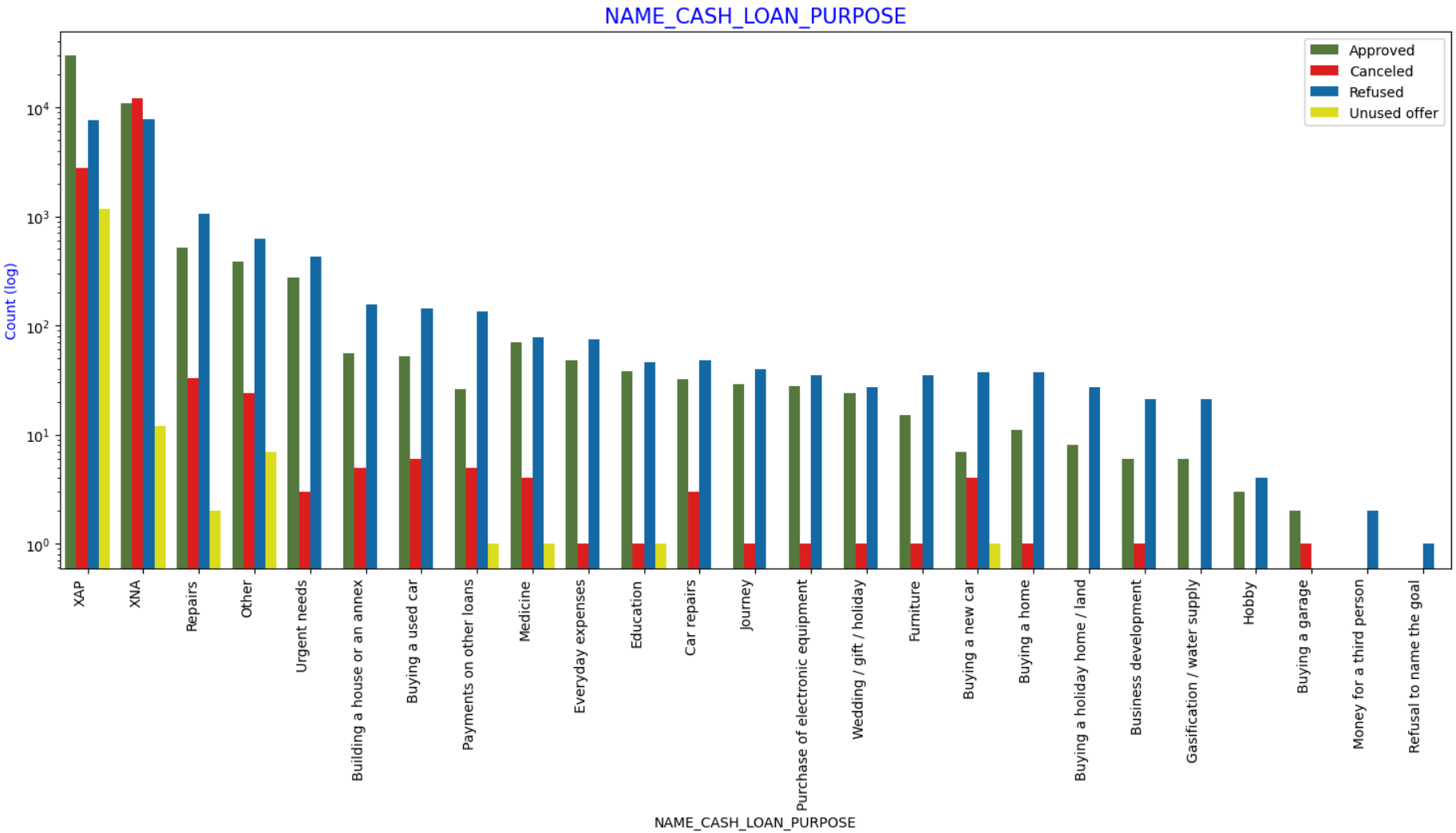
```
# Bifurcating the application dataframe based on Target value 0 and 1 for correlation and other analysis

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

Plotting Contract Status vs purpose of the loan:

In [246...

```
univariate_merged("NAME_CASH_LOAN_PURPOSE",L0,"NAME_CONTRACT_STATUS",["#548235","#FF0000","#0070C0","#FFFF00"],True,(18,7))

univariate_merged("NAME_CASH_LOAN_PURPOSE",L1,"NAME_CONTRACT_STATUS",["#548235","#FF0000","#0070C0","#FFFF00"],True,(18,7))
```


NAME_CASH_LOAN_PURPOSE
Approved
Canceled
Refused
Unused offer
Count (log)
10^5
10^4
10^3
10^2
10^1
10^0
XAP
XNA
Repairs
Other
Urgent needs
Buying a used car
Building a house or an annex
Everyday expenses
Medicine
Payments on other loans
Education
Journey
Buying a new car
Purchase of electronic equipment
Wedding / gift / holiday
Buying a home
Furniture
Car repairs
Buying a holiday home / land
Business development
Gasification / water supply
Buying a garage
Hobby
Money for a third person
Refusal to name the goal
NAME_CASH_LOAN_PURPOSE



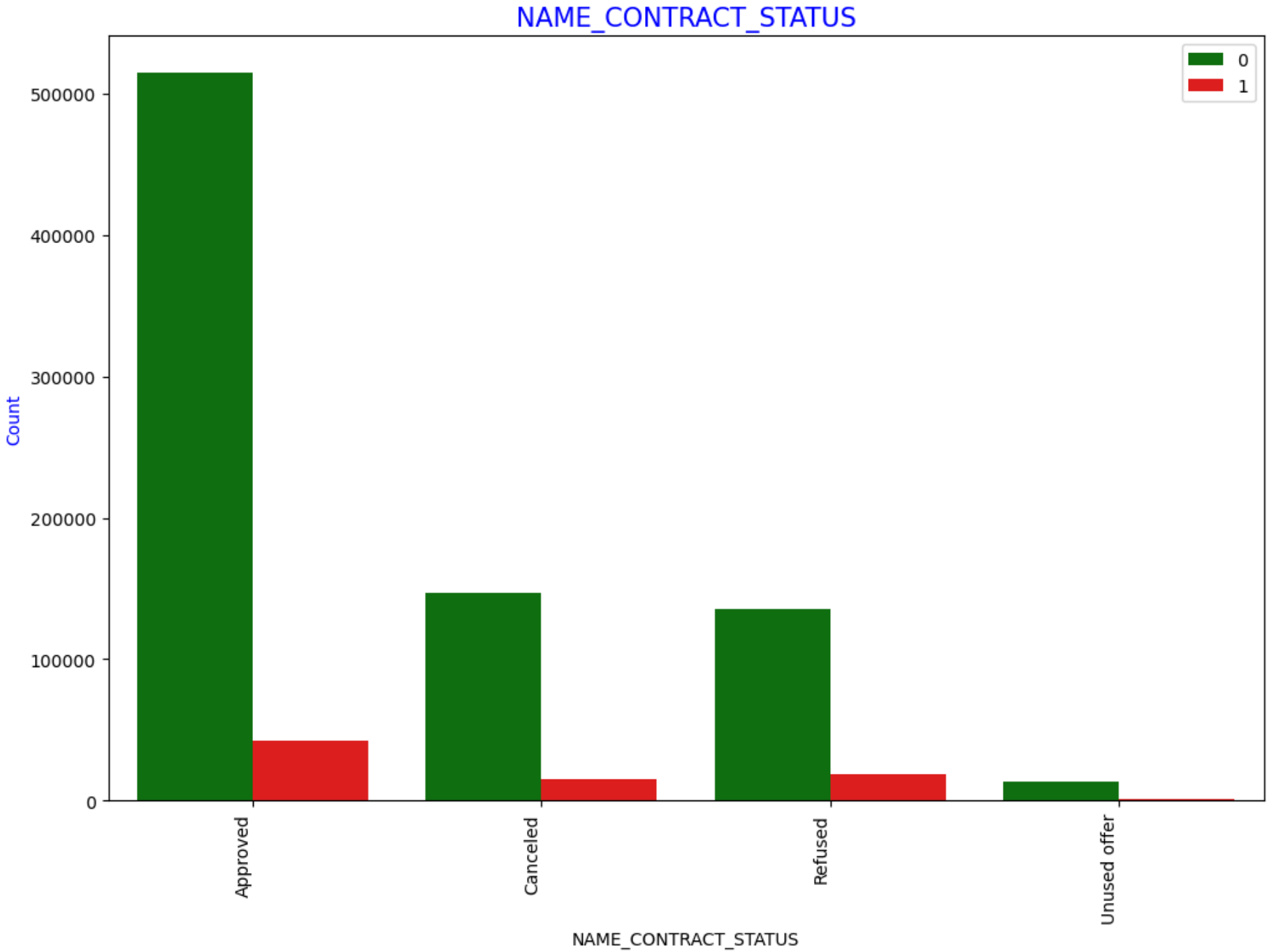
Loan purpose has high number of unknown values (XAP, XNA)

Loan taken for the purpose of Repairs seems to have highest default rate

A very high number application have been rejected by bank or refused by client which has purpose as repair or other. This shows that purpose repair is taken as high risk by bank and either they are rejected or bank offers very high loan interest rate which is not feasible by the clients, thus they refuse the loan.

In [248…

```
# Checking the Contract Status based on loan repayment status and whether there is any business loss or financial loss

univariate_merged("NAME_CONTRACT_STATUS",loan_process_df,"TARGET",['g','r'],False,(12,8))
g = loan_process_df.groupby("NAME_CONTRACT_STATUS")["TARGET"]
df1 = pd.concat([g.value_counts(),round(g.value_counts(normalize=True).mul(100),2)],axis=1, keys=('Counts','Percentage'))
df1['Percentage'] = df1['Percentage'].astype(str) +"%" # adding percentage symbol in the results for understanding
print (df1)
```


NAME_CONTRACT_STATUS
0
1
500000
400000
300000
200000
100000
0
Count
Approved
Canceled
Refused
Unused offer
NAME_CONTRACT_STATUS

```
                            Counts Percentage
NAME_CONTRACT_STATUS TARGET
Approved             0      515220     92.41%
                     1       42313      7.59%
Canceled             0      146729     90.82%
                     1       14823      9.18%
Refused              0      135396      88.0%
                     1       18463      12.0%
Unused offer         0       13217     91.77%
                     1        1186      8.23%
```

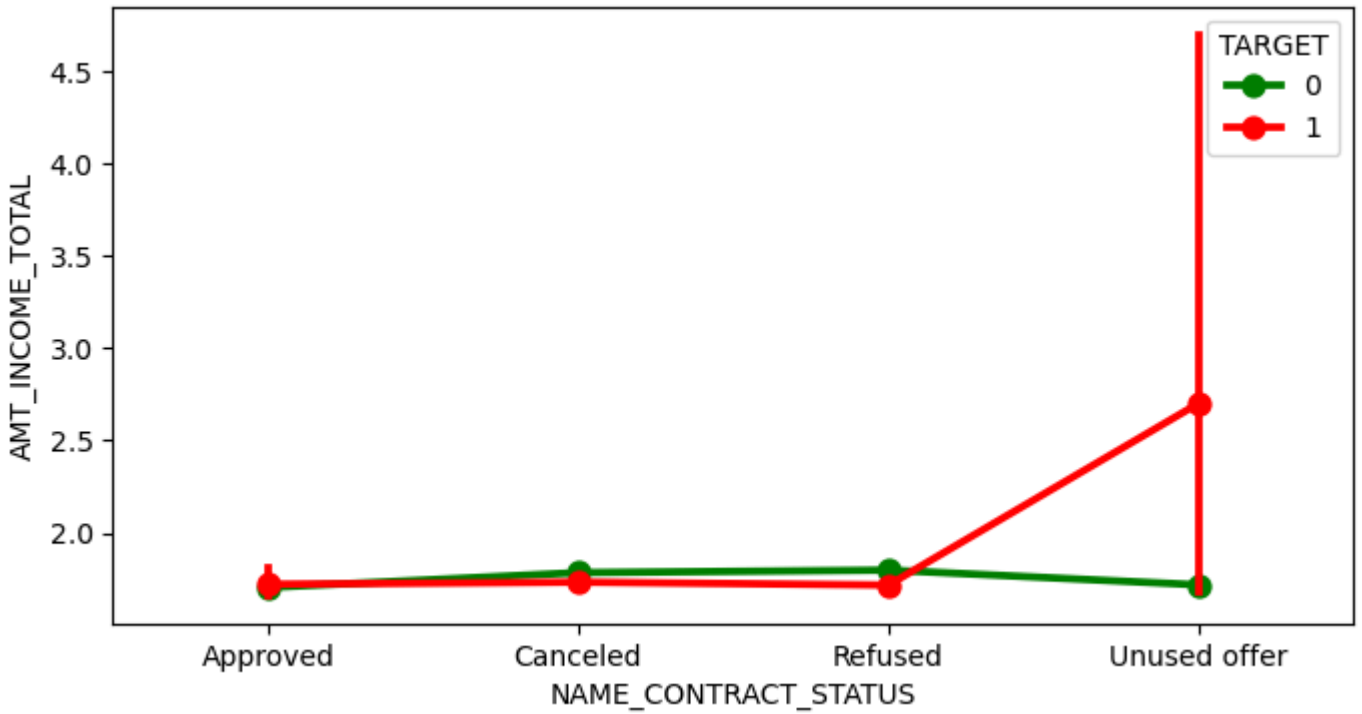
90% of the previously cancelled client have actually repayed the loan. Revisiting the interest rates would increase business opoortunity for these clients

88% of the clients who have been previously refused a loan has payed back the loan in current case.

Refual reason should be recorded for further analysis as these clients would turn into potential repaying customer.

In [250…

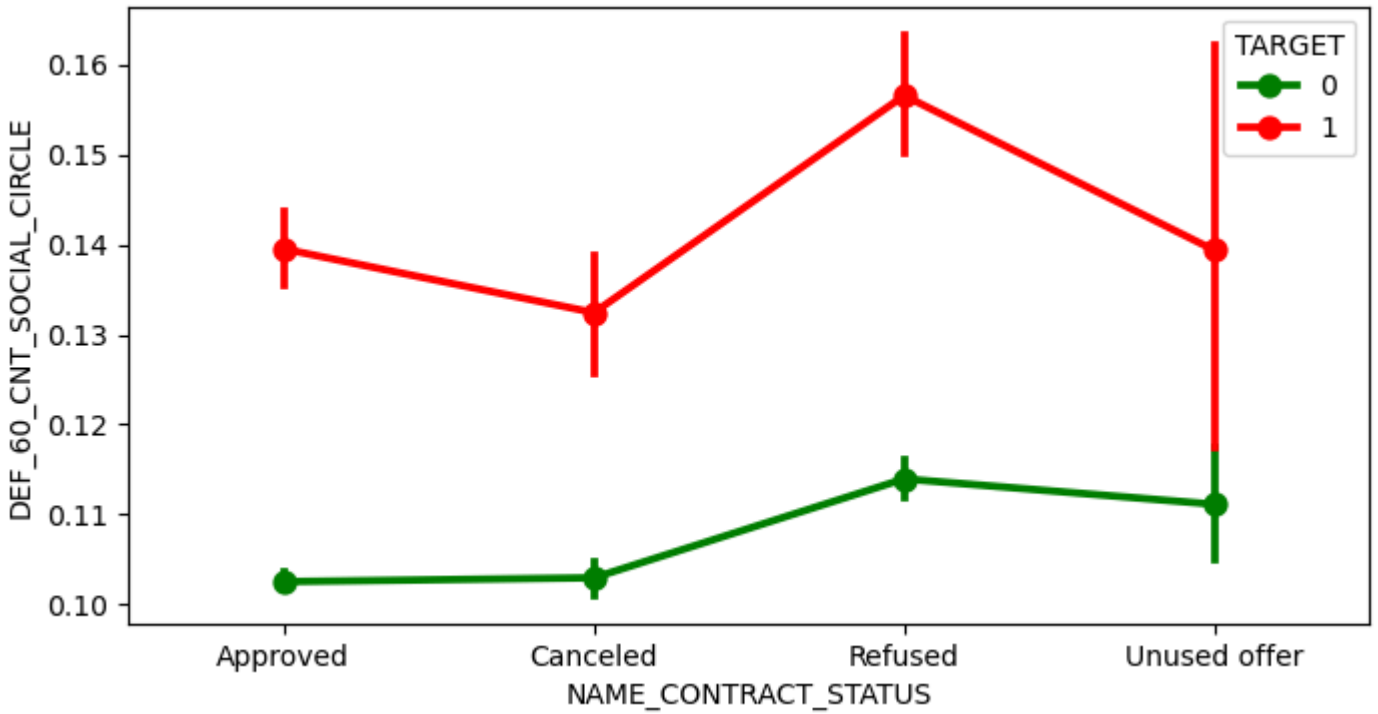
```
# plotting the relationship between income total and contact status

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



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

In [252…

```
# plotting the relationship between people who defaulted in last 60 days being in client's social circle and contact status

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



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

Conclusion for the project

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

Decisive Factor whether an applicant will be Repayer:

NAME_EDUCATION_TYPE: Academic degree has less defaults.

NAME_INCOME_TYPE: Student and Businessmen have no defaults.

REGION_RATING_CLIENT: RATING 1 is safer.

ORGANIZATION_TYPE: Clients with Trade Type 4 and 5 and Industry type 8 have defaulted less than 3%

DAYS_BIRTH: People above age of 50 have low probability of defaulting

DAYS_EMPLOYED: Clients with 40+ year experience having less than 1% default rate

AMT_INCOME_TOTAL:Applicant with Income more than 700,000 are less likely to default

NAME_CASH_LOAN_PURPOSE: Loans bought for Hobby, Buying garage are being repayed mostly.

CNT_CHILDREN: People with zero to two children tend to repay the loans

Decisive Factor whether an applicant will be Defaulter:

CODE_GENDER: Men are at relatively higher default rate

NAME_FAMILY_STATUS : People who have civil marriage or who are single default a lot.

NAME_EDUCATION_TYPE: People with Lower Secondary & Secondary education

NAME_INCOME_TYPE: Clients who are either at Maternity leave OR Unemployed default a lot.

REGION_RATING_CLIENT: People who live in Rating 3 has highest defaults.

OCCUPATION_TYPE: Avoid Low-skill Laborers, Drivers and Waiters/barmen staff, Security staff, Laborers and Cooking staff as the default rate is huge.

ORGANIZATION_TYPE: Organizations with highest percent of loans not repaid are Transport: type 3 (16%), Industry: type 13 (13.5%), Industry: type 8 (12.5%) and Restaurant (less than 12%). Self-employed people have relative high defaulting rate, and thus should be avoided to be approved for loan or provide loan with higher interest rate to mitigate the risk of defaulting.

DAYS_BIRTH: Avoid young people who are in age group of 20-40 as they have higher probability of defaulting

DAYS_EMPLOYED: People who have less than 5 years of employment have high default rate.

CNT_CHILDREN & CNT_FAM_MEMBERS: Client who have children equal to or more than 9 default 100% and hence their applications are to be rejected.

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

The following attributes indicate that people from these category tend to default but then due to the number of people and the amount of loan, the bank could provide loan with higher interest to mitigate any default risk thus preventing business loss:

NAME_HOUSING_TYPE: High number of loan applications are from the category of people who live in Rented apartments & living with parents and hence offering the loan would mitigate the loss if any of those default.

AMT_CREDIT: People who get loan for 300-600k tend to default more than others and hence having higher interest specifically for this credit range would be ideal.

AMT_INCOME: Since 90% of the applications have Income total less than 300,000 and they have high probability of defaulting, they could be offered loan with higher interest compared to other income category.

CNT_CHILDREN & CNT_FAM_MEMBERS: Clients who have 4 to 8 children has a very high default rate and hence higher interest should be imposed on their loans.

NAME_CASH_LOAN_PURPOSE: Loan taken for the purpose of Repairs seems to have highest default rate. A very high number applications have been rejected by bank or refused by client in previous applications as well which has purpose as repair or other. This shows that purpose repair is taken as high risk by bank and either they are rejected, or bank offers very high loan interest rate which is not feasible by the clients, thus they refuse the loan. The same approach could be followed in future as well.

Other suggestions:

90% of the previously cancelled client have actually repayed the loan. Record the reason for cancellation which might help the bank to determine and negotiate terms with these repaying customers in future for increase business opportunity.

88% of the clients who were refused by bank for loan earlier have now turned into a repaying client. Hence documenting the reason for rejection could mitigate the business loss and these clients could be contacted for further loans.

In [ ]: