Link -
https://drive.google.com/file/d/1ugm7H2Qz9nkDoN0tj_BNbqSr-IaIx076/view?usp=sharing

WEEK 9 : Task 14 – Loan Case Study

Project Description - We are given Loan case study, which consists of 3 datasets i) Application data – contains all data of applicants who have applied for loan ii) Previous application data – contains data of previously applied applicants iii) Columns description – contains information about columns in application data and previous application data. With help of this we need to identify patterns and insights which will help loan providing companies to approve loans with less risks.

Approach – We will apply Exploratory Data Analysis (EDA) to identify accurate insights and patterns. For that we need to clean and modify data i.e Identifying and handling missing data, identifying outliers, checking datatypes, categorizing columns etc.

Tech-Stack Used - Jupiter Notebook 6.4.5 which allows us to create and share documents which contains codes, plots, visualizations and project documentations.

Insights – PPT attached (below)

Result – Discover how EDA is applied to real-life business scenarios and how it can reduce the risk of losing money in finance. Gain insight into risk analytics in banking and financial services so that you can minimize the risk of losing money when lending.

Loan Case Study

In [1]:

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

In [2]:

```
#Import libraries and load dataset
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
pd.set_option('display.width', 1000)
pd.set_option('display.max_columns', 200)
pd.set_option('display.max_rows', 500)
```

Understanding Data

In [3]:

```
# Reading application data and previous_application data
app_data=pd.read_csv("application_data.csv")
app_data.head()
```

Out[3]:

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

In [4]:

```
pre_data=pd.read_csv("previous_application.csv")
pre_data.head()
```

Out[4]:

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

In [5]:

```
# Inspecting dataframes
app_data.shape
```

Out[5]:

```
(307511, 122)
```

In [6]:

```
app_data.info()
```

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

In [7]:

```
app_data.info(verbose=True) #Understanding datatype of each column of application_data
```

```
<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 [8]:

```
round(app_data.describe(),2)
```

Out[8]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANN
count	307511.00	307511.00	307511.00	3.075110e+05	307511.00	3074
mean	278180.52	0.08	0.42	1.687979e+05	599026.00	271
std	102790.18	0.27	0.72	2.371231e+05	402490.78	144
min	100002.00	0.00	0.00	2.565000e+04	45000.00	16
25%	189145.50	0.00	0.00	1.125000e+05	270000.00	165
50%	278202.00	0.00	0.00	1.471500e+05	513531.00	249
75%	367142.50	0.00	1.00	2.025000e+05	808650.00	345
max	456255.00	1.00	19.00	1.170000e+08	4050000.00	2580

In [9]:

```
pre_data.shape
```

Out[9]:

```
(1670214, 37)
```

In [10]:

```
pre_data.info(verbose=True) #Understanding datatype of each column of previous_application
```

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

In [11]:

```
round(pre_data.describe(),2)
```

Out[11]:

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOV
count	1670214.00	1670214.00	1297979.00	1670214.00	1670213.00	
mean	1923089.14	278357.17	15955.12	175233.86	196114.02	
std	532597.96	102814.82	14782.14	292779.76	318574.62	
min	1000001.00	100001.00	0.00	0.00	0.00	
25%	1461857.25	189329.00	6321.78	18720.00	24160.50	
50%	1923110.50	278714.50	11250.00	71046.00	80541.00	
75%	2384279.75	367514.00	20658.42	180360.00	216418.50	
max	2845382.00	456255.00	418058.14	6905160.00	6905160.00	

Data Cleaning and Manipulation

Application Data

Checking missing data and Outliers

In [12]:

```
# First, we will check if any duplicate rows are present
app_data.SK_ID_CURR.duplicated().sum()
```

Out[12]:

```
0
```

In [13]:

```
#Checking missing values
app_data.isnull().sum()
```

Out[13]:

```
SK_ID_CURR                          0
TARGET                              0
NAME_CONTRACT_TYPE                  0
CODE_GENDER                         0
FLAG_OWN_CAR                        0
FLAG_OWN_REALTY                     0
CNT_CHILDREN                        0
AMT_INCOME_TOTAL                    0
AMT_CREDIT                          0
AMT_ANNUITY                        12
AMT_GOODS_PRICE                   278
NAME_TYPE_SUITE                  1292
NAME_INCOME_TYPE                    0
NAME_EDUCATION_TYPE                 0
NAME_FAMILY_STATUS                  0
NAME_HOUSING_TYPE                   0
REGION_POPULATION_RELATIVE          0
DAYS_BIRTH                          0
DAYS_EMPLOYED                       0
DAYS_REGISTRATION                   0
DAYS_ID_PUBLISH                     0
OWN_CAR_AGE                    202929
FLAG_MOBIL                          0
FLAG_EMP_PHONE                      0
FLAG_WORK_PHONE                     0
FLAG_CONT_MOBILE                    0
FLAG_PHONE                          0
FLAG_EMAIL                          0
OCCUPATION_TYPE                 96391
CNT_FAM_MEMBERS                     2
REGION_RATING_CLIENT                0
REGION_RATING_CLIENT_W_CITY         0
WEEKDAY_APPR_PROCESS_START          0
HOUR_APPR_PROCESS_START             0
REG_REGION_NOT_LIVE_REGION          0
REG_REGION_NOT_WORK_REGION          0
LIVE_REGION_NOT_WORK_REGION         0
REG_CITY_NOT_LIVE_CITY              0
REG_CITY_NOT_WORK_CITY              0
LIVE_CITY_NOT_WORK_CITY             0
ORGANIZATION_TYPE                   0
EXT_SOURCE_1                   173378
EXT_SOURCE_2                      660
EXT_SOURCE_3                    60965
APARTMENTS_AVG                 156061
BASEMENTAREA_AVG               179943
YEARS_BEGINEXPLUATATION_AVG    150007
YEARS_BUILD_AVG                204488
COMMONAREA_AVG                 214865
ELEVATORS_AVG                  163891
ENTRANCES_AVG                  154828
FLOORSMAX_AVG                  153020
FLOORSMIN_AVG                  208642
LANDAREA_AVG                   182590
```

```
LIVINGAPARTMENTS_AVG            210199
LIVINGAREA_AVG                  154350
NONLIVINGAPARTMENTS_AVG         213514
NONLIVINGAREA_AVG               169682
APARTMENTS_MODE                 156061
BASEMENTAREA_MODE               179943
YEARS_BEGINEXPLUATATION_MODE    150007
YEARS_BUILD_MODE                204488
COMMONAREA_MODE                 214865
ELEVATORS_MODE                  163891
ENTRANCES_MODE                  154828
FLOORSMAX_MODE                  153020
FLOORSMIN_MODE                  208642
LANDAREA_MODE                   182590
LIVINGAPARTMENTS_MODE           210199
LIVINGAREA_MODE                 154350
NONLIVINGAPARTMENTS_MODE        213514
NONLIVINGAREA_MODE              169682
APARTMENTS_MEDI                 156061
BASEMENTAREA_MEDI               179943
YEARS_BEGINEXPLUATATION_MEDI    150007
YEARS_BUILD_MEDI                204488
COMMONAREA_MEDI                 214865
ELEVATORS_MEDI                  163891
ENTRANCES_MEDI                  154828
FLOORSMAX_MEDI                  153020
FLOORSMIN_MEDI                  208642
LANDAREA_MEDI                   182590
LIVINGAPARTMENTS_MEDI           210199
LIVINGAREA_MEDI                 154350
NONLIVINGAPARTMENTS_MEDI        213514
NONLIVINGAREA_MEDI              169682
FONDKAPREMONT_MODE              210295
HOUSETYPE_MODE                  154297
TOTALAREA_MODE                  148431
WALLSMATERIAL_MODE              156341
EMERGENCYSTATE_MODE             145755
OBS_30_CNT_SOCIAL_CIRCLE          1021
DEF_30_CNT_SOCIAL_CIRCLE          1021
OBS_60_CNT_SOCIAL_CIRCLE          1021
DEF_60_CNT_SOCIAL_CIRCLE          1021
DAYS_LAST_PHONE_CHANGE               1
FLAG_DOCUMENT_2                      0
FLAG_DOCUMENT_3                      0
FLAG_DOCUMENT_4                      0
FLAG_DOCUMENT_5                      0
FLAG_DOCUMENT_6                      0
FLAG_DOCUMENT_7                      0
FLAG_DOCUMENT_8                      0
FLAG_DOCUMENT_9                      0
FLAG_DOCUMENT_10                     0
FLAG_DOCUMENT_11                     0
FLAG_DOCUMENT_12                     0
FLAG_DOCUMENT_13                     0
FLAG_DOCUMENT_14                     0
FLAG_DOCUMENT_15                     0
FLAG_DOCUMENT_16                     0
FLAG_DOCUMENT_17                     0
FLAG_DOCUMENT_18                     0
FLAG_DOCUMENT_19                     0
FLAG_DOCUMENT_20                     0
```

```
FLAG_DOCUMENT_21                  0
AMT_REQ_CREDIT_BUREAU_HOUR    41519
AMT_REQ_CREDIT_BUREAU_DAY     41519
AMT_REQ_CREDIT_BUREAU_WEEK    41519
AMT_REQ_CREDIT_BUREAU_MON     41519
AMT_REQ_CREDIT_BUREAU_QRT     41519
AMT_REQ_CREDIT_BUREAU_YEAR    41519
dtype: int64
```

- Clearly, we can see that dataset has many missing values. So, let's check column wise percentage of missing values

In [14]:

```
round(app_data.isnull().sum() / len(app_data) * 100,2)
```

Out[14]:

```
SK_ID_CURR                       0.00
TARGET                           0.00
NAME_CONTRACT_TYPE               0.00
CODE_GENDER                      0.00
FLAG_OWN_CAR                     0.00
FLAG_OWN_REALTY                  0.00
CNT_CHILDREN                     0.00
AMT_INCOME_TOTAL                 0.00
AMT_CREDIT                       0.00
AMT_ANNUITY                      0.00
AMT_GOODS_PRICE                  0.09
NAME_TYPE_SUITE                  0.42
NAME_INCOME_TYPE                 0.00
NAME_EDUCATION_TYPE              0.00
NAME_FAMILY_STATUS               0.00
NAME_HOUSING_TYPE                0.00
REGION_POPULATION_RELATIVE       0.00
DAYS_BIRTH                       0.00
DAYS_EMPLOYED                    0.00
DAYS_REGISTRATION                0.00
DAYS_ID_PUBLISH                  0.00
OWN_CAR_AGE                     65.99
FLAG_MOBIL                       0.00
FLAG_EMP_PHONE                   0.00
FLAG_WORK_PHONE                  0.00
FLAG_CONT_MOBILE                 0.00
FLAG_PHONE                       0.00
FLAG_EMAIL                       0.00
OCCUPATION_TYPE                 31.35
CNT_FAM_MEMBERS                  0.00
REGION_RATING_CLIENT             0.00
REGION_RATING_CLIENT_W_CITY      0.00
WEEKDAY_APPR_PROCESS_START       0.00
HOUR_APPR_PROCESS_START          0.00
REG_REGION_NOT_LIVE_REGION       0.00
REG_REGION_NOT_WORK_REGION       0.00
LIVE_REGION_NOT_WORK_REGION      0.00
REG_CITY_NOT_LIVE_CITY           0.00
REG_CITY_NOT_WORK_CITY           0.00
LIVE_CITY_NOT_WORK_CITY          0.00
ORGANIZATION_TYPE                0.00
EXT_SOURCE_1                    56.38
EXT_SOURCE_2                     0.21
EXT_SOURCE_3                    19.83
APARTMENTS_AVG                  50.75
BASEMENTAREA_AVG                58.52
YEARS_BEGINEXPLUATATION_AVG     48.78
YEARS_BUILD_AVG                 66.50
COMMONAREA_AVG                  69.87
ELEVATORS_AVG                   53.30
ENTRANCES_AVG                   50.35
FLOORSMAX_AVG                   49.76
FLOORSMIN_AVG                   67.85
LANDAREA_AVG                    59.38
```

```
LIVINGAPARTMENTS_AVG            68.35
LIVINGAREA_AVG                  50.19
NONLIVINGAPARTMENTS_AVG         69.43
NONLIVINGAREA_AVG               55.18
APARTMENTS_MODE                 50.75
BASEMENTAREA_MODE               58.52
YEARS_BEGINEXPLUATATION_MODE    48.78
YEARS_BUILD_MODE                66.50
COMMONAREA_MODE                 69.87
ELEVATORS_MODE                  53.30
ENTRANCES_MODE                  50.35
FLOORSMAX_MODE                  49.76
FLOORSMIN_MODE                  67.85
LANDAREA_MODE                   59.38
LIVINGAPARTMENTS_MODE           68.35
LIVINGAREA_MODE                 50.19
NONLIVINGAPARTMENTS_MODE        69.43
NONLIVINGAREA_MODE              55.18
APARTMENTS_MEDI                 50.75
BASEMENTAREA_MEDI               58.52
YEARS_BEGINEXPLUATATION_MEDI    48.78
YEARS_BUILD_MEDI                66.50
COMMONAREA_MEDI                 69.87
ELEVATORS_MEDI                  53.30
ENTRANCES_MEDI                  50.35
FLOORSMAX_MEDI                  49.76
FLOORSMIN_MEDI                  67.85
LANDAREA_MEDI                   59.38
LIVINGAPARTMENTS_MEDI           68.35
LIVINGAREA_MEDI                 50.19
NONLIVINGAPARTMENTS_MEDI        69.43
NONLIVINGAREA_MEDI              55.18
FONDKAPREMONT_MODE              68.39
HOUSETYPE_MODE                  50.18
TOTALAREA_MODE                  48.27
WALLSMATERIAL_MODE              50.84
EMERGENCYSTATE_MODE             47.40
OBS_30_CNT_SOCIAL_CIRCLE         0.33
DEF_30_CNT_SOCIAL_CIRCLE         0.33
OBS_60_CNT_SOCIAL_CIRCLE         0.33
DEF_60_CNT_SOCIAL_CIRCLE         0.33
DAYS_LAST_PHONE_CHANGE           0.00
FLAG_DOCUMENT_2                  0.00
FLAG_DOCUMENT_3                  0.00
FLAG_DOCUMENT_4                  0.00
FLAG_DOCUMENT_5                  0.00
FLAG_DOCUMENT_6                  0.00
FLAG_DOCUMENT_7                  0.00
FLAG_DOCUMENT_8                  0.00
FLAG_DOCUMENT_9                  0.00
FLAG_DOCUMENT_10                 0.00
FLAG_DOCUMENT_11                 0.00
FLAG_DOCUMENT_12                 0.00
FLAG_DOCUMENT_13                 0.00
FLAG_DOCUMENT_14                 0.00
FLAG_DOCUMENT_15                 0.00
FLAG_DOCUMENT_16                 0.00
FLAG_DOCUMENT_17                 0.00
FLAG_DOCUMENT_18                 0.00
FLAG_DOCUMENT_19                 0.00
FLAG_DOCUMENT_20                 0.00
```

```
FLAG_DOCUMENT_21              0.00
AMT_REQ_CREDIT_BUREAU_HOUR   13.50
AMT_REQ_CREDIT_BUREAU_DAY    13.50
AMT_REQ_CREDIT_BUREAU_WEEK   13.50
AMT_REQ_CREDIT_BUREAU_MON    13.50
AMT_REQ_CREDIT_BUREAU_QRT    13.50
AMT_REQ_CREDIT_BUREAU_YEAR   13.50
dtype: float64
```

In [15]:

```
#Now will check columns which have most missing values >=40%
null_app = pd.DataFrame((app_data.isnull().sum())*100/len(app_data)).reset_index()
null_app.columns = ['Column Name', 'Null Percentage']
null_40_app = round(null_app[null_app["Null Percentage"]>=40],2)
null_40_app
```

Out[15]:

	Column Name	Null Percentage
21	OWN_CAR_AGE	65.99
41	EXT_SOURCE_1	56.38
44	APARTMENTS_AVG	50.75
45	BASEMENTAREA_AVG	58.52
46	YEARS_BEGINEXPLUATATION_AVG	48.78
47	YEARS_BUILD_AVG	66.50
48	COMMONAREA_AVG	69.87
49	ELEVATORS_AVG	53.30
50	ENTRANCES_AVG	50.35
51	FLOORSMAX_AVG	49.76
52	FLOORSMIN_AVG	67.85
53	LANDAREA_AVG	59.38
54	LIVINGAPARTMENTS_AVG	68.35
55	LIVINGAREA_AVG	50.19
56	NONLIVINGAPARTMENTS_AVG	69.43
57	NONLIVINGAREA_AVG	55.18
58	APARTMENTS_MODE	50.75
59	BASEMENTAREA_MODE	58.52
60	YEARS_BEGINEXPLUATATION_MODE	48.78
61	YEARS_BUILD_MODE	66.50
62	COMMONAREA_MODE	69.87
63	ELEVATORS_MODE	53.30
64	ENTRANCES_MODE	50.35
65	FLOORSMAX_MODE	49.76
66	FLOORSMIN_MODE	67.85
67	LANDAREA_MODE	59.38
68	LIVINGAPARTMENTS_MODE	68.35
69	LIVINGAREA_MODE	50.19
70	NONLIVINGAPARTMENTS_MODE	69.43
71	NONLIVINGAREA_MODE	55.18
72	APARTMENTS_MEDI	50.75
73	BASEMENTAREA_MEDI	58.52

	Column Name	Null Percentage
74	YEARS_BEGINEXPLUATATION_MEDI	48.78
75	YEARS_BUILD_MEDI	66.50
76	COMMONAREA_MEDI	69.87
77	ELEVATORS_MEDI	53.30
78	ENTRANCES_MEDI	50.35
79	FLOORSMAX_MEDI	49.76
80	FLOORSMIN_MEDI	67.85
81	LANDAREA_MEDI	59.38
82	LIVINGAPARTMENTS_MEDI	68.35
83	LIVINGAREA_MEDI	50.19
84	NONLIVINGAPARTMENTS_MEDI	69.43
85	NONLIVINGAREA_MEDI	55.18
86	FONDKAPREMONT_MODE	68.39
87	HOUSETYPE_MODE	50.18
88	TOTALAREA_MODE	48.27
89	WALLSMATERIAL_MODE	50.84
90	EMERGENCYSTATE_MODE	47.40

In [16]:

```
# Removing columns with NULL values >=40% as this columns have to many missing values
col_to_del = null_40_app["Column Name"].tolist()
app_data.drop(col_to_del,axis=1,inplace=True)
```

- Other than above columns we will also try to analyze other columns which are <40% and are unnecessary with respect to our objective.

In [17]:

```
app_data.shape
```

Out[17]:

```
(307511, 73)
```

In [18]:

```
#Inspecting columns with NULL values <40%
null_under40_app= null_app[null_app["Null Percentage"] <40]
null_under40_app.sort_values(by = 'Null Percentage', ascending = False)
```

Out[18]:

	Column Name	Null Percentage
28	OCCUPATION_TYPE	31.345545
43	EXT_SOURCE_3	19.825307
121	AMT_REQ_CREDIT_BUREAU_YEAR	13.501631
120	AMT_REQ_CREDIT_BUREAU_QRT	13.501631
119	AMT_REQ_CREDIT_BUREAU_MON	13.501631
118	AMT_REQ_CREDIT_BUREAU_WEEK	13.501631
117	AMT_REQ_CREDIT_BUREAU_DAY	13.501631
116	AMT_REQ_CREDIT_BUREAU_HOUR	13.501631
11	NAME_TYPE_SUITE	0.420148
91	OBS_30_CNT_SOCIAL_CIRCLE	0.332021
92	DEF_30_CNT_SOCIAL_CIRCLE	0.332021
93	OBS_60_CNT_SOCIAL_CIRCLE	0.332021
94	DEF_60_CNT_SOCIAL_CIRCLE	0.332021
42	EXT_SOURCE_2	0.214626
10	AMT_GOODS_PRICE	0.090403
9	AMT_ANNUITY	0.003902
29	CNT_FAM_MEMBERS	0.000650
95	DAYS_LAST_PHONE_CHANGE	0.000325
111	FLAG_DOCUMENT_17	0.000000
112	FLAG_DOCUMENT_18	0.000000
115	FLAG_DOCUMENT_21	0.000000
114	FLAG_DOCUMENT_20	0.000000
113	FLAG_DOCUMENT_19	0.000000
96	FLAG_DOCUMENT_2	0.000000
97	FLAG_DOCUMENT_3	0.000000
98	FLAG_DOCUMENT_4	0.000000
99	FLAG_DOCUMENT_5	0.000000
110	FLAG_DOCUMENT_16	0.000000
100	FLAG_DOCUMENT_6	0.000000
101	FLAG_DOCUMENT_7	0.000000
102	FLAG_DOCUMENT_8	0.000000
103	FLAG_DOCUMENT_9	0.000000

	Column Name	Null Percentage
104	FLAG_DOCUMENT_10	0.000000
105	FLAG_DOCUMENT_11	0.000000
40	ORGANIZATION_TYPE	0.000000
107	FLAG_DOCUMENT_13	0.000000
108	FLAG_DOCUMENT_14	0.000000
109	FLAG_DOCUMENT_15	0.000000
106	FLAG_DOCUMENT_12	0.000000
0	SK_ID_CURR	0.000000
39	LIVE_CITY_NOT_WORK_CITY	0.000000
19	DAYS_REGISTRATION	0.000000
2	NAME_CONTRACT_TYPE	0.000000
3	CODE_GENDER	0.000000
4	FLAG_OWN_CAR	0.000000
5	FLAG_OWN_REALTY	0.000000
6	CNT_CHILDREN	0.000000
7	AMT_INCOME_TOTAL	0.000000
8	AMT_CREDIT	0.000000
12	NAME_INCOME_TYPE	0.000000
13	NAME_EDUCATION_TYPE	0.000000
14	NAME_FAMILY_STATUS	0.000000
15	NAME_HOUSING_TYPE	0.000000
16	REGION_POPULATION_RELATIVE	0.000000
17	DAYS_BIRTH	0.000000
18	DAYS_EMPLOYED	0.000000
20	DAYS_ID_PUBLISH	0.000000
38	REG_CITY_NOT_WORK_CITY	0.000000
22	FLAG_MOBIL	0.000000
23	FLAG_EMP_PHONE	0.000000
24	FLAG_WORK_PHONE	0.000000
25	FLAG_CONT_MOBILE	0.000000
26	FLAG_PHONE	0.000000
27	FLAG_EMAIL	0.000000
30	REGION_RATING_CLIENT	0.000000
31	REGION_RATING_CLIENT_W_CITY	0.000000
32	WEEKDAY_APPR_PROCESS_START	0.000000
33	HOUR_APPR_PROCESS_START	0.000000
34	REG_REGION_NOT_LIVE_REGION	0.000000
35	REG_REGION_NOT_WORK_REGION	0.000000
36	LIVE_REGION_NOT_WORK_REGION	0.000000

	Column Name	Null Percentage
1	TARGET	0.000000
37	REG_CITY_NOT_LIVE_CITY	0.000000

- Here we can observe that OCCUPATION_TYPE, EXT_SOURCE_3, AMT_REQ_CREDIT_BUREAU_YEAR, AMT_REQ_CREDIT_BUREAU_QRT, AMT_REQ_CREDIT_BUREAU_MON, AMT_REQ_CREDIT_BUREAU_WEEK, AMT_REQ_CREDIT_BUREAU_DAY, AMT_REQ_CREDIT_BUREAU_HOUR are columns who seems to have highest missing values percentage and along with AMT_GOODS_PRICE, AMT_ANNUITY. So, we need to address them.

In [19]:

```
# OCCUPATION_TYPE
app_data['OCCUPATION_TYPE'].value_counts()
```

Out[19]:

```
Laborers                 55186
Sales staff              32102
Core staff               27570
Managers                 21371
Drivers                  18603
High skill tech staff    11380
Accountants               9813
Medicine staff            8537
Security staff            6721
Cooking staff             5946
Cleaning staff            4653
Private service staff     2652
Low-skill Laborers        2093
Waiters/barmen staff      1348
Secretaries               1305
Realty agents              751
HR staff                   563
IT staff                   526
Name: OCCUPATION_TYPE, dtype: int64
```

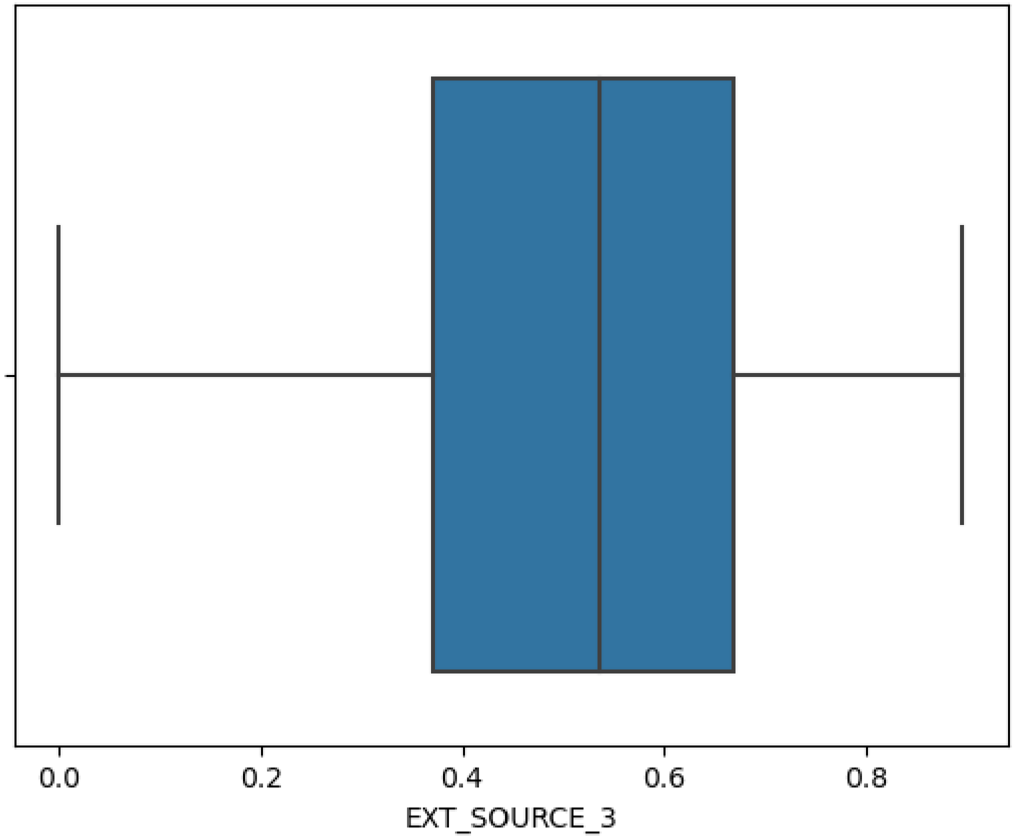
In [20]:

```
plt.rcdefaults()
plt.rcParams.update({'axes.facecolor':'white'})
```

- OCCUPATION_TYPE is one of the important column, we can't impute it with any values. We will leave it unchange.

In [21]:

```
# EXT_SOURCE_3
# With help of boxplot we will check for outliers
sns.boxplot(app_data['EXT_SOURCE_3'])
plt.show()
```



In [22]:

```
app_data["EXT_SOURCE_3"].fillna(app_data.EXT_SOURCE_3.mean(), inplace = True)
app_data["EXT_SOURCE_3"].isnull().sum()
```

Out[22]:

```
0
```

- We with help of boxplot we can observe that EXT_SOURCE_3 has no outliers. So, we can impute all NULL values with mean of the column

In [23]:

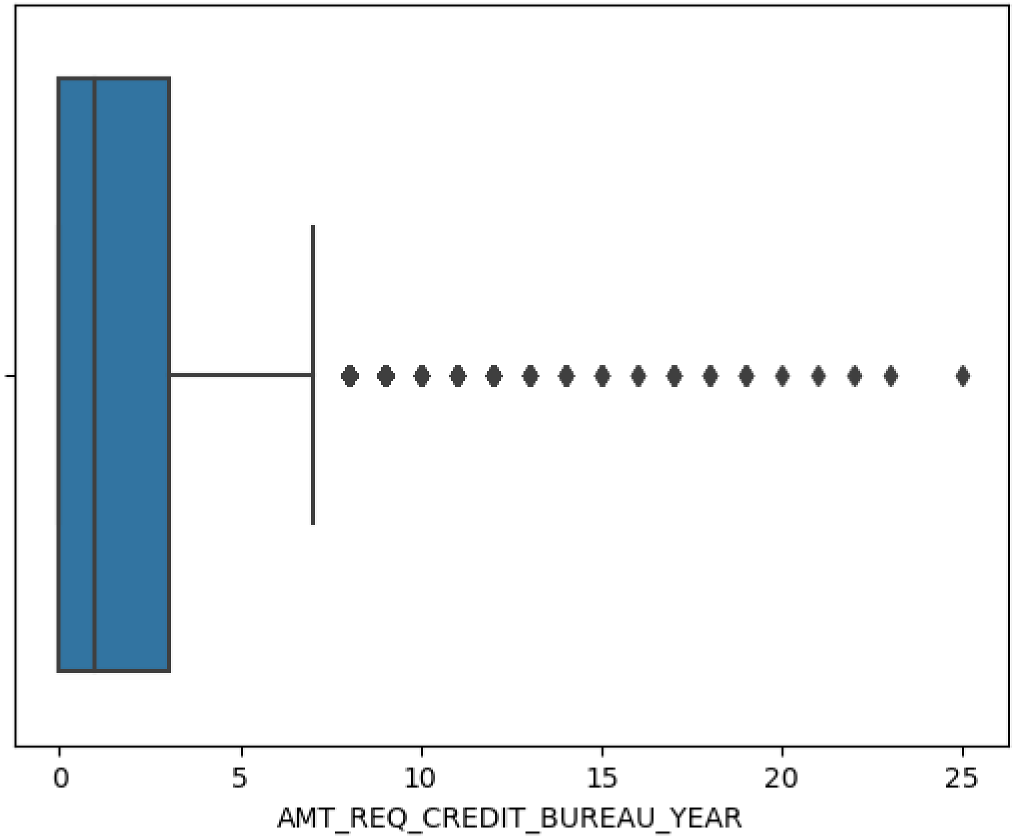
```
# AMT_REQ_CREDIT_BUREAU_YEAR
app_data.AMT_REQ_CREDIT_BUREAU_YEAR.value_counts()
```

Out[23]:

```
0.0     71801
1.0     63405
2.0     50192
3.0     33628
4.0     20714
5.0     12052
6.0      6967
7.0      3869
8.0      2127
9.0      1096
11.0       31
12.0       30
10.0       22
13.0       19
14.0       10
17.0        7
15.0        6
19.0        4
18.0        4
16.0        3
25.0        1
23.0        1
22.0        1
21.0        1
20.0        1
Name: AMT_REQ_CREDIT_BUREAU_YEAR, dtype: int64
```

In [24]:

```
sns.boxplot(app_data["AMT_REQ_CREDIT_BUREAU_YEAR"])
plt.show()
```



In [25]:

```
app_data["AMT_REQ_CREDIT_BUREAU_YEAR"].fillna(0, inplace = True)
app_data["AMT_REQ_CREDIT_BUREAU_YEAR"].isnull().sum()
```

Out[25]:

```
0
```

- We can see that 0 appears highest number of times, 0 means no credit. When applicant have no credit then they tend not to answer to queries. So, we will replace all NULL values with 0. Similarly we can do this with AMT_REQ_CREDIT_BUREAU_QRT, AMT_REQ_CREDIT_BUREAU_MON, AMT_REQ_CREDIT_BUREAU_WEEK, AMT_REQ_CREDIT_BUREAU_DAY, AMT_REQ_CREDIT_BUREAU_HOUR and all has outliers.

In [26]:

```
app_data["AMT_REQ_CREDIT_BUREAU_QRT"].fillna(0, inplace = True)
app_data["AMT_REQ_CREDIT_BUREAU_MON"].fillna(0, inplace = True)
app_data["AMT_REQ_CREDIT_BUREAU_WEEK"].fillna(0, inplace = True)
app_data["AMT_REQ_CREDIT_BUREAU_DAY"].fillna(0, inplace = True)
app_data["AMT_REQ_CREDIT_BUREAU_HOUR"].fillna(0, inplace = True)
```

In [27]:

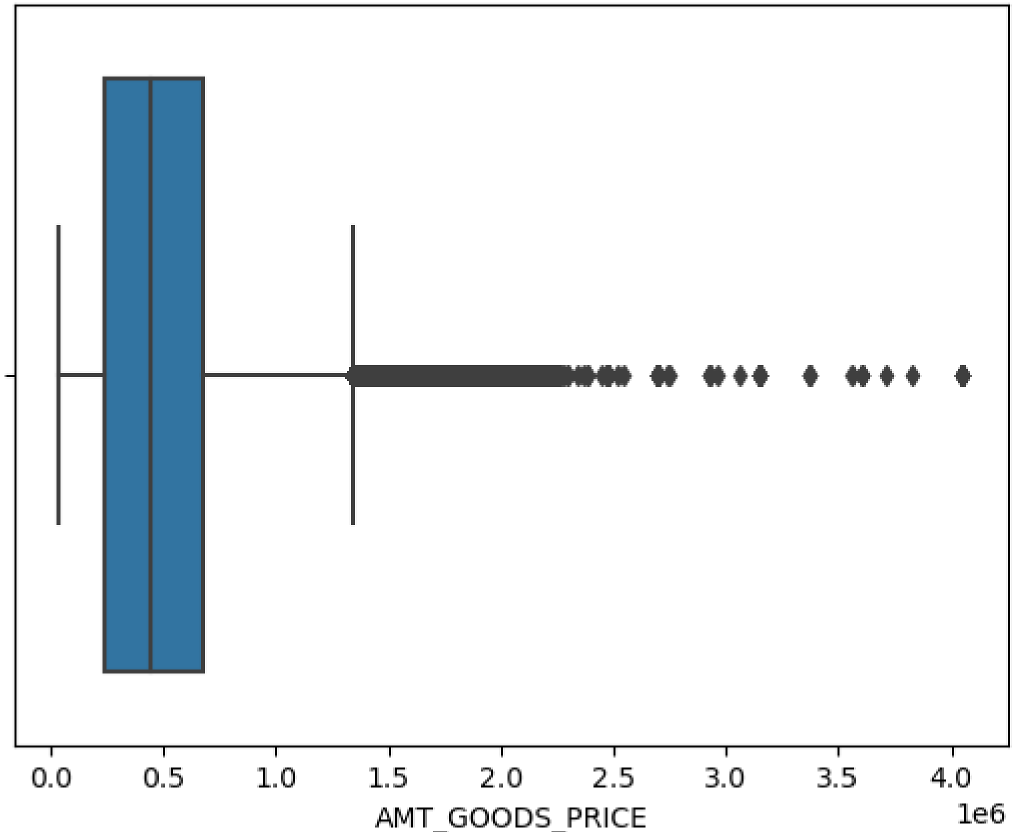
```
# AMT_GOODS_PRICE
app_data.AMT_GOODS_PRICE.value_counts()
```

Out[27]:

```
450000.0     26022
225000.0     25282
675000.0     24962
900000.0     15416
270000.0     11428
             ...
1265751.0        1
503266.5         1
810778.5         1
666090.0         1
743863.5         1
Name: AMT_GOODS_PRICE, Length: 1002, dtype: int64
```

In [28]:

```
sns.boxplot(app_data.AMT_GOODS_PRICE)
plt.show()
```



In [29]:

```
# AMT_GOODS_PRICE:For consumer loans it is the price of the goods for which the loan is giv
app_data["AMT_GOODS_PRICE"].fillna(app_data.AMT_GOODS_PRICE.median(), inplace = True)
app_data["AMT_GOODS_PRICE"].isnull().sum()
# We replace NULL values with median because it is given to highest number of times given
```

Out[29]:

```
0
```

In [30]:

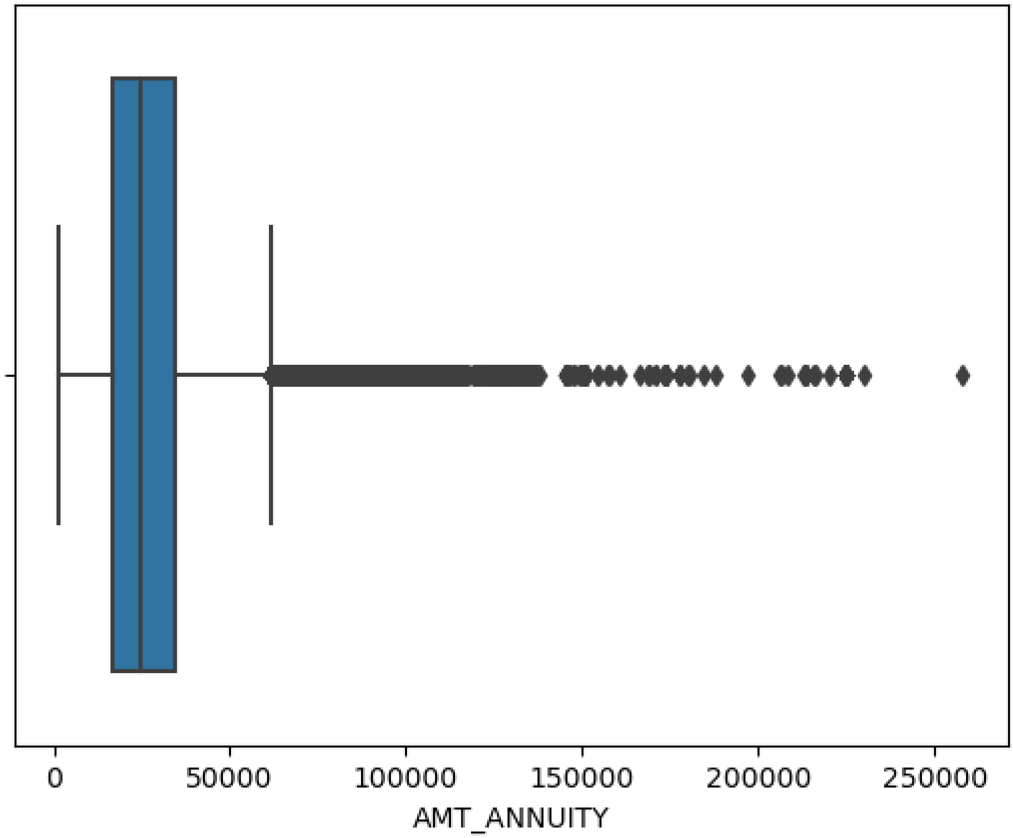
```
# AMT_ANNUITY
app_data.AMT_ANNUITY.value_counts()
```

Out[30]:

```
9000.0      6385
13500.0     5514
6750.0      2279
10125.0     2035
37800.0     1602
            ...
79902.0        1
106969.5       1
60885.0        1
59661.0        1
77809.5        1
Name: AMT_ANNUITY, Length: 13672, dtype: int64
```

In [31]:

```
sns.boxplot(app_data.AMT_ANNUITY)
plt.show()
```



In [32]:

```
app_data["AMT_ANNUITY"].fillna(app_data.AMT_GOODS_PRICE.median(), inplace = True)
app_data["AMT_ANNUITY"].isnull().sum()
# AMT_ANNUITY has outliers . So replacing NULL values with median
```

Out[32]:

```
0
```

Checking Data types and dealing with invalid data

In [33]:

```
# Converting Numerical and Object columns to Category
category_col = ['NAME_CONTRACT_TYPE','CODE_GENDER','NAME_TYPE_SUITE','NAME_INCOME_TYPE','NA
                'NAME_FAMILY_STATUS','NAME_HOUSING_TYPE','OCCUPATION_TYPE','WEEKDAY_
                'ORGANIZATION_TYPE','FLAG_OWN_CAR','FLAG_OWN_REALTY','LIVE_CITY_NOT_
                'REG_CITY_NOT_LIVE_CITY','REG_CITY_NOT_WORK_CITY','REG_REGION_NOT_WO
                'LIVE_REGION_NOT_WORK_REGION','REGION_RATING_CLIENT','WEEKDAY_APPR_P
                'REGION_RATING_CLIENT_W_CITY'
               ]
for col in category_col:
    app_data[col] = pd.Categorical(app_data[col])
```

In [34]:

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 73 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                  307511 non-null  float64 
 10  AMT_GOODS_PRICE              307511 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  FLAG_MOBIL                   307511 non-null  int64   
 22  FLAG_EMP_PHONE               307511 non-null  int64   
 23  FLAG_WORK_PHONE              307511 non-null  int64   
 24  FLAG_CONT_MOBILE             307511 non-null  int64   
 25  FLAG_PHONE                   307511 non-null  int64   
 26  FLAG_EMAIL                   307511 non-null  int64   
 27  OCCUPATION_TYPE              211120 non-null  category
 28  CNT_FAM_MEMBERS              307509 non-null  float64 
 29  REGION_RATING_CLIENT         307511 non-null  category
 30  REGION_RATING_CLIENT_W_CITY  307511 non-null  category
 31  WEEKDAY_APPR_PROCESS_START   307511 non-null  category
 32  HOUR_APPR_PROCESS_START      307511 non-null  int64   
 33  REG_REGION_NOT_LIVE_REGION   307511 non-null  int64   
 34  REG_REGION_NOT_WORK_REGION   307511 non-null  category
 35  LIVE_REGION_NOT_WORK_REGION  307511 non-null  category
 36  REG_CITY_NOT_LIVE_CITY       307511 non-null  category
 37  REG_CITY_NOT_WORK_CITY       307511 non-null  category
 38  LIVE_CITY_NOT_WORK_CITY      307511 non-null  category
 39  ORGANIZATION_TYPE            307511 non-null  category
 40  EXT_SOURCE_2                 306851 non-null  float64 
 41  EXT_SOURCE_3                 307511 non-null  float64 
 42  OBS_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64 
 43  DEF_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64 
 44  OBS_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64 
 45  DEF_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64 
 46  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64 
 47  FLAG_DOCUMENT_2              307511 non-null  int64   
 48  FLAG_DOCUMENT_3              307511 non-null  int64   
 49  FLAG_DOCUMENT_4              307511 non-null  int64   
 50  FLAG_DOCUMENT_5              307511 non-null  int64   
 51  FLAG_DOCUMENT_6              307511 non-null  int64   
```

```
 52  FLAG_DOCUMENT_7             307511 non-null  int64
 53  FLAG_DOCUMENT_8             307511 non-null  int64
 54  FLAG_DOCUMENT_9             307511 non-null  int64
 55  FLAG_DOCUMENT_10            307511 non-null  int64
 56  FLAG_DOCUMENT_11            307511 non-null  int64
 57  FLAG_DOCUMENT_12            307511 non-null  int64
 58  FLAG_DOCUMENT_13            307511 non-null  int64
 59  FLAG_DOCUMENT_14            307511 non-null  int64
 60  FLAG_DOCUMENT_15            307511 non-null  int64
 61  FLAG_DOCUMENT_16            307511 non-null  int64
 62  FLAG_DOCUMENT_17            307511 non-null  int64
 63  FLAG_DOCUMENT_18            307511 non-null  int64
 64  FLAG_DOCUMENT_19            307511 non-null  int64
 65  FLAG_DOCUMENT_20            307511 non-null  int64
 66  FLAG_DOCUMENT_21            307511 non-null  int64
 67  AMT_REQ_CREDIT_BUREAU_HOUR  307511 non-null  float64
 68  AMT_REQ_CREDIT_BUREAU_DAY   307511 non-null  float64
 69  AMT_REQ_CREDIT_BUREAU_WEEK  307511 non-null  float64
 70  AMT_REQ_CREDIT_BUREAU_MON   307511 non-null  float64
 71  AMT_REQ_CREDIT_BUREAU_QRT   307511 non-null  float64
 72  AMT_REQ_CREDIT_BUREAU_YEAR  307511 non-null  float64
dtypes: category(19), float64(20), int64(34)
memory usage: 132.3 MB
```

In [35]:

```
# Checking values of TARGET
app_data.TARGET.value_counts()
```

Out[35]:

```
0    282686
1     24825
Name: TARGET, dtype: int64
```

- Looks Good

In [36]:

```
# Checking values of NAME_CONTRACT_TYPE
app_data.NAME_CONTRACT_TYPE.value_counts()
```

Out[36]:

```
Cash loans         278232
Revolving loans     29279
Name: NAME_CONTRACT_TYPE, dtype: int64
```

- No strange values found

In [37]:

```
# Checking values of CODE_GENDER
app_data.CODE_GENDER.value_counts()
```

Out[37]:

```
F      202448
M      105059
XNA         4
Name: CODE_GENDER, dtype: int64
```

In [38]:

```
app_data.CODE_GENDER.mode()
```

Out[38]:

```
0    F
Name: CODE_GENDER, dtype: category
Categories (3, object): ['F', 'M', 'XNA']
```

In [39]:

```
# Replacing XNA with F as it is most frequently occured. This will not affect our data.
app_data["CODE_GENDER"].replace({"XNA": "F"}, inplace=True)
app_data.CODE_GENDER.value_counts()
```

Out[39]:

```
F    202452
M    105059
Name: CODE_GENDER, dtype: int64
```

In [40]:

```
# Checking day time columns 'DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLI
app_data.DAYS_BIRTH[app_data.DAYS_BIRTH<0]
```

Out[40]:

```
0          -9461
1         -16765
2         -19046
3         -19005
4         -19932
           ...
307506     -9327
307507    -20775
307508    -14966
307509    -11961
307510    -16856
Name: DAYS_BIRTH, Length: 307511, dtype: int64
```

- We can observe that applicants age is negative. So, we need to change it to positive. We can also observe that age is in days, we will convert it into years. Similarily we can do with rest if any.

In [41]:

```
day_time_col = ['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH','DAYS_LA
for col in day_time_col:
    app_data[col] = abs(app_data[col])
```

In [42]:

```
# Converting DAYS_BIRTH into years and adding YEARS_BIRTH
app_data["Age"] = (app_data.DAYS_BIRTH/365).astype(int)
app_data["Age"].value_counts()
```

Out[42]:

```
38    8873
37    8799
39    8770
40    8624
36    8614
27    8476
41    8449
31    8377
43    8308
42    8216
28    7975
32    7911
44    7819
30    7806
35    7804
33    7714
29    7670
34    7631
54    7551
53    7457
46    7293
45    7205
47    7018
48    6984
56    6828
57    6768
52    6763
51    6689
55    6637
59    6631
49    6627
50    6482
58    6268
60    6227
62    5514
61    5418
63    5197
64    5117
26    4561
25    4168
23    4057
24    3905
65    3113
22    2933
66    2085
67    2042
21    1254
68     866
69      16
20       1
Name: Age, dtype: int64
```

In [43]:

```
# Categorizing AMT_INCOME_TOTAL column
app_data.AMT_INCOME_TOTAL.value_counts()
```

Out[43]:

```
135000.0    35750
112500.0    31019
157500.0    26556
180000.0    24719
90000.0     22483
            ...
117324.0        1
64584.0         1
142897.5        1
109170.0        1
113062.5        1
Name: AMT_INCOME_TOTAL, Length: 2548, dtype: int64
```

In [44]:

```
app_data.AMT_INCOME_TOTAL.describe()
```

Out[44]:

```
count    3.075110e+05
mean     1.687979e+05
std      2.371231e+05
min      2.565000e+04
25%      1.125000e+05
50%      1.471500e+05
75%      2.025000e+05
max      1.170000e+08
Name: AMT_INCOME_TOTAL, dtype: float64
```

In [45]:

```
#Binning RANGE_AMT_INCOME based on quantiles.
income_labels = ['Very Low', 'Low', "Medium", 'High', 'Very high']
app_data["RANGE_AMT_INCOME"] = pd.qcut(app_data.AMT_INCOME_TOTAL, q=[0,0.1,0.3,0.6,0.8,1],
app_data["RANGE_AMT_INCOME"].value_counts()
```

Out[45]:

```
Medium       84302
High         75513
Low          67187
Very high    47118
Very Low     33391
Name: RANGE_AMT_INCOME, dtype: int64
```

In [46]:

```
#Doing same with AMT_CREDIT
app_data.AMT_CREDIT.value_counts()
```

Out[46]:

```
450000.0     9709
675000.0     8877
225000.0     8162
180000.0     7342
270000.0     7241
             ...
487318.5        1
630400.5        1
1875276.0       1
1395895.5       1
1391130.0       1
Name: AMT_CREDIT, Length: 5603, dtype: int64
```

In [47]:

```
#Binning employment time
app_data["EMPLOYMENT_TIME"] = app_data["DAYS_EMPLOYED"] // 365
bins = [0,5,10,20,30,40,50,60,150]
slots = ['0-5','5-10','10-20','20-30','30-40','40-50','50-60','60 above']
app_data["EMPLOYMENT_TIME"]=pd.cut(app_data["EMPLOYMENT_TIME"],bins=bins,labels=slots)
app_data["EMPLOYMENT_TIME"].value_counts()
```

Out[47]:

```
0-5         124634
5-10         55983
10-20        32658
20-30         8409
30-40         2374
40-50          175
50-60            0
60 above         0
Name: EMPLOYMENT_TIME, dtype: int64
```

In [48]:

```
app_data.AMT_CREDIT.describe()
```

Out[48]:

```
count    3.075110e+05
mean     5.990260e+05
std      4.024908e+05
min      4.500000e+04
25%      2.700000e+05
50%      5.135310e+05
75%      8.086500e+05
max      4.050000e+06
Name: AMT_CREDIT, dtype: float64
```

In [49]:

```
# Binning AMT_CREDIT based on quantiles.
app_data["RANGE_AMT_CREDIT"] = pd.qcut(app_data.AMT_CREDIT, q=[0, .2, .4, .6, .8, 1], label
app_data["RANGE_AMT_CREDIT"].value_counts()
```

Out[49]:

```
Very Low     64925
High         64024
Medium       61552
Very high    58912
Low          58098
Name: RANGE_AMT_CREDIT, dtype: int64
```

In [50]:

```
# Categorizing Age column
app_data.Age.describe()
```

Out[50]:

```
count    307511.000000
mean         43.435968
std          11.954593
min          20.000000
25%          34.000000
50%          43.000000
75%          53.000000
max          69.000000
Name: Age, dtype: float64
```

In [51]:

```
# Binning Age to Groups
app_data["RANGE_YEARS_BIRTH"]=pd.cut(app_data['Age'], bins=[20,30,40,60,70],
                                        labels=['Young Adult', 'Adult', 'Middle Age', 'Seni
app_data["RANGE_YEARS_BIRTH"].value_counts()
```

Out[51]:

```
Middle Age        142220
Adult              83117
Young Adult        52805
Senior_citizen     29368
Name: RANGE_YEARS_BIRTH, dtype: int64
```

Identifying Imbalance in Data

In [52]:

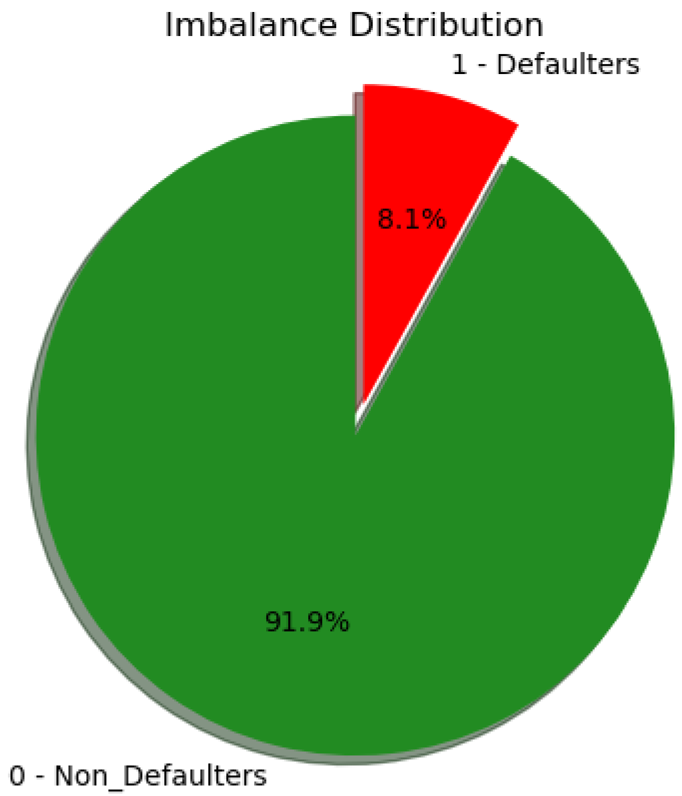
```
# Checking the Target attribute
app_data['TARGET'].value_counts(normalize=True)*100
```

Out[52]:

```
0    91.927118
1     8.072882
Name: TARGET, dtype: float64
```

In [53]:

```
labels = '0 - Non_Defaulters', '1 - Defaulters'
fig1, ax1 = plt.subplots()
colors = ('#228B22','#ff0000')
ax1.pie(app_data['TARGET'].value_counts(),  colors = colors, explode=(0, 0.1), labels=label
        shadow=True, startangle=90)
ax1.axis('equal')
plt.title('Imbalance Distribution')
plt.show()
```



- CLearly we can observe that the data imbalance ratio is 92:8(approx)

Dividing dataframe into 2 sets

In [54]:

```
# Splitting dataframe into 2 sets based on TARGET variable i.e Defaulter and Non Defaulter
defaulter = app_data[app_data["TARGET"]==1]
non_defaulter = app_data[app_data["TARGET"]==0]
```

Univariate Analysis

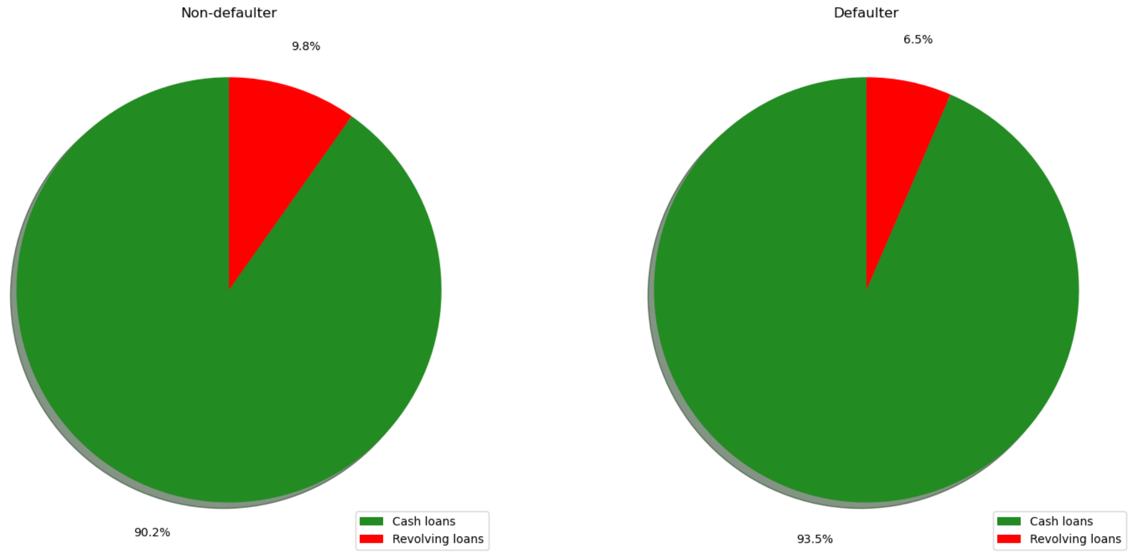
Categorical

In [55]:

```
def plot_univariate_pie(variable):
    fig, (ax1, ax2) = plt.subplots(1,2,figsize=(18,18))
    new_0 = non_defaulter[variable].value_counts()
    labels = new_0.index
    colors = ('#228B22','#ff0000','#c203fc','#03fcc2','#fc03b6','#fcfc03','#fc9003','#03dbf
    ax1.pie(new_0, autopct='%1.1f%%',colors =colors,pctdistance=1.2,
        labeldistance=1.2,shadow=True, startangle=90)
    ax1.set_title('Non-defaulter')
    ax1.legend(labels, loc="lower right")

    new_1 = defaulter[variable].value_counts()
    labels = new_1.index
    colors = ('#228B22','#ff0000','#c203fc','#03fcc2','#fc03b6','#fcfc03','#fc9003','#03dbf
    ax2.pie(new_1, autopct='%1.1f%%',colors = colors,pctdistance=1.2,
        labeldistance=1.2,shadow=True, startangle=90)
    ax2.set_title('Defaulter')
    ax2.legend(labels, loc="lower right")
    plt.show()
```

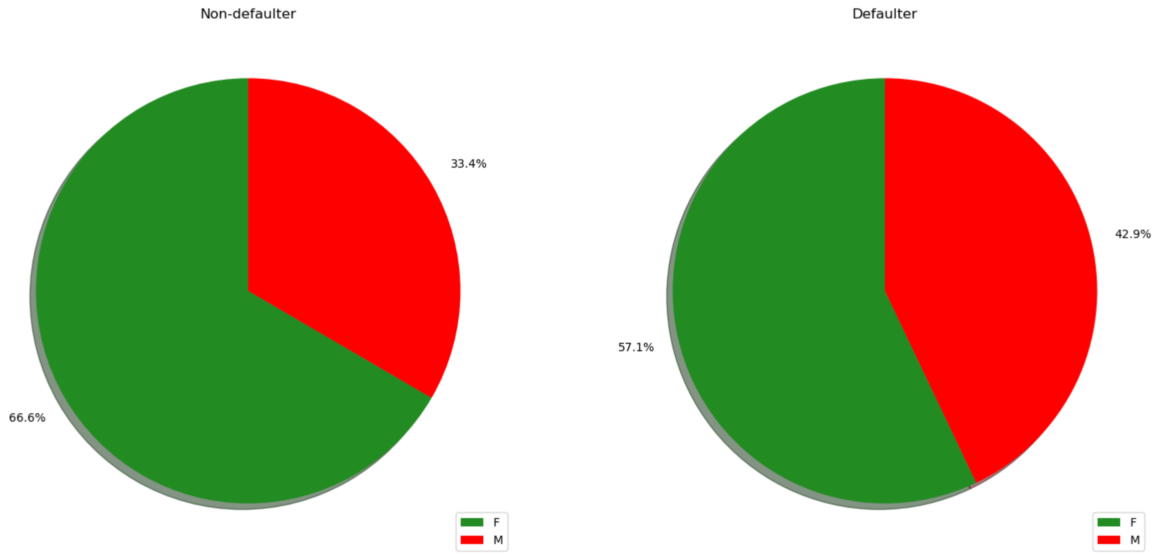
In [56]:

```
# Plotting whether Loan is cash or revolving
plot_univariate_pie("NAME_CONTRACT_TYPE")
```



In [57]:

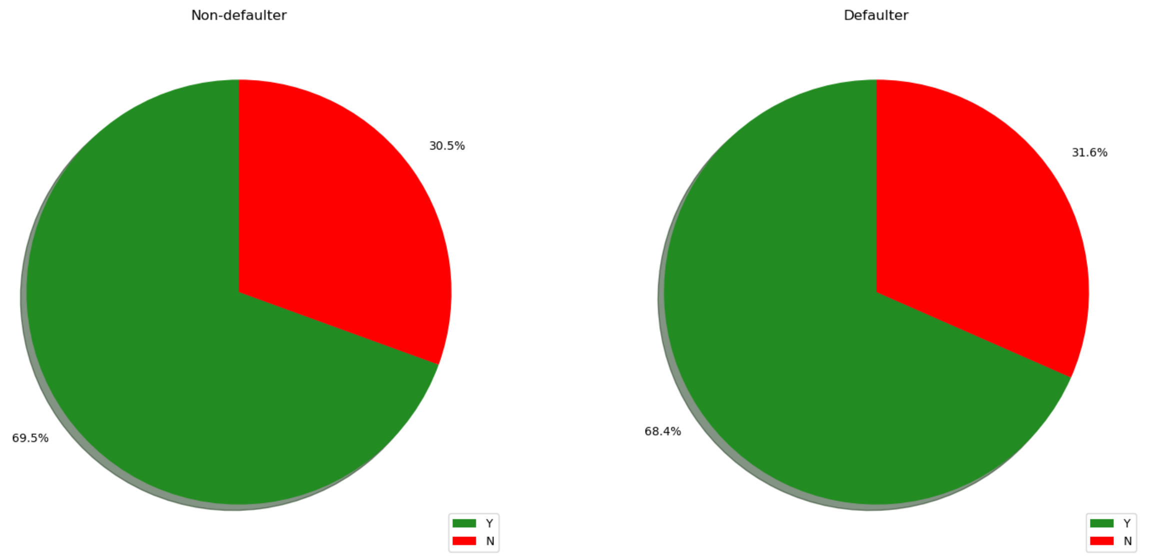
```
# Plotting applicants gender
plot_univariate_pie("CODE_GENDER")
```



- The number of female applicants is almost twice the number of male clients. Based on the percentage of defaulter, males have a higher chance of not returning their loans, comparing with women

In [58]:

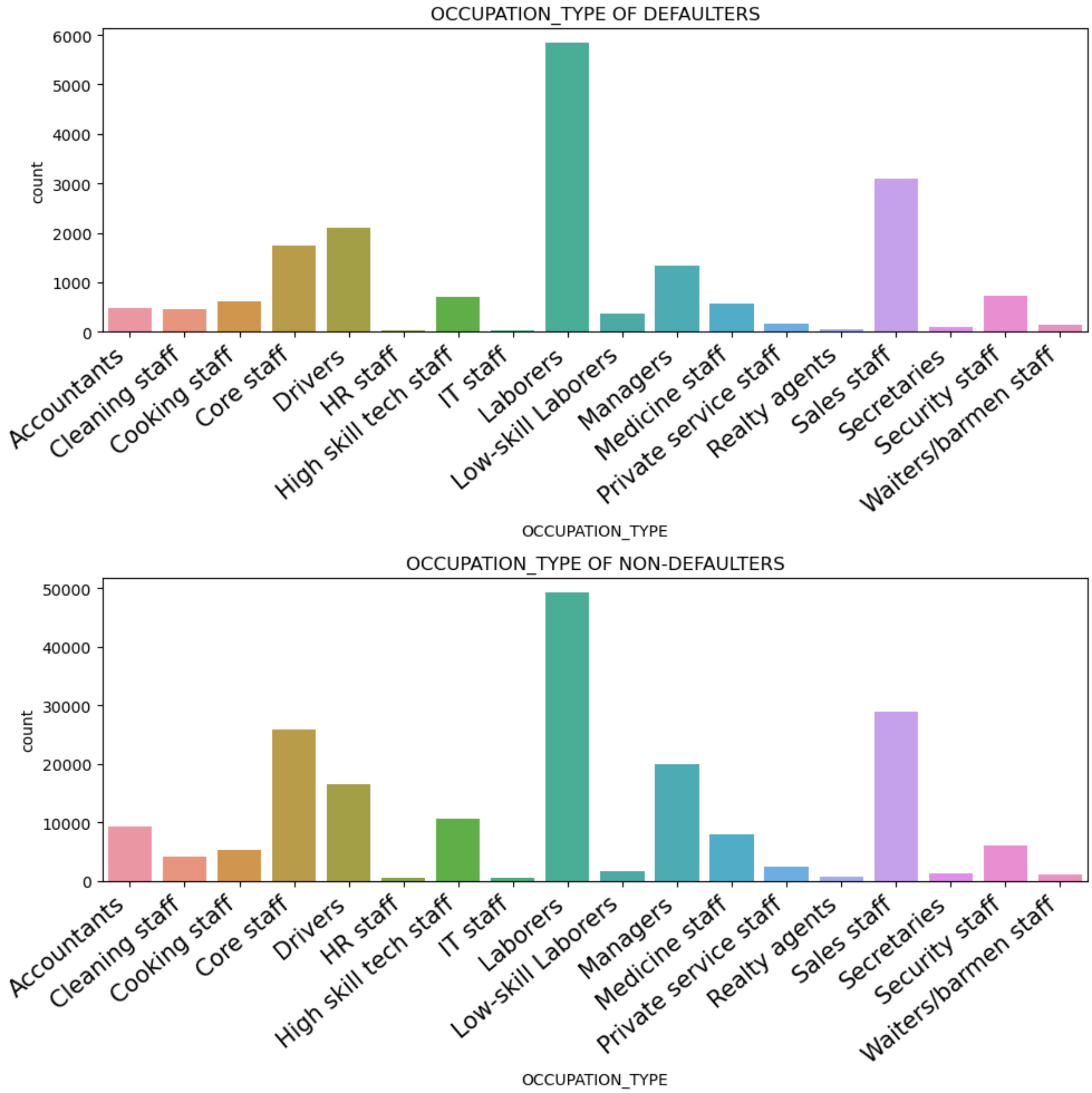
```
# Plotting if applicant owns a house or flat
plot_univariate_pie("FLAG_OWN_REALTY")
```



- Applicants who own house or flat is twice than that of who don't own.
- And defaulting percentage of both are almost same. So, we found that there is no correlation owning house or flat that of
deafaulting the loan

In [59]:

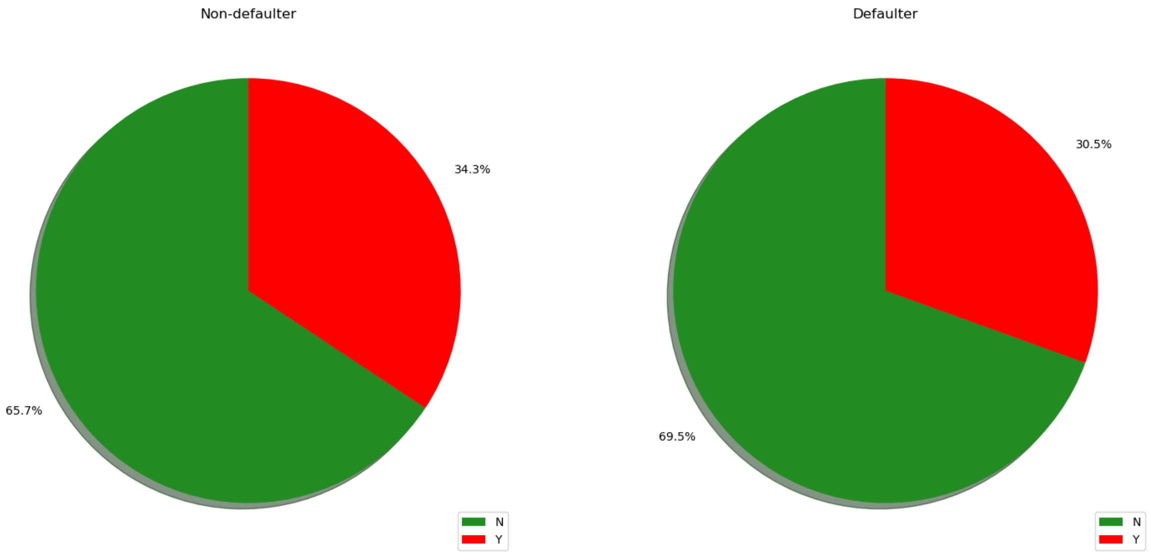
```
# OCCUPATION_TYPE
plt.figure(figsize=(10,10))
plt.subplot(2,1,1)
plt.title('OCCUPATION_TYPE OF DEFAULTERS')
ax = sns.countplot(x='OCCUPATION_TYPE',data=defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()

plt.subplot(2,1,2)
plt.title('OCCUPATION_TYPE OF NON-DEFAULTERS')
ax = sns.countplot(x='OCCUPATION_TYPE',data=non_defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()
plt.show()
```



- Most loans are taken by Laborers
- Group of applicants with highest deafaulters are also Laborers followed by sales staff

In [60]:

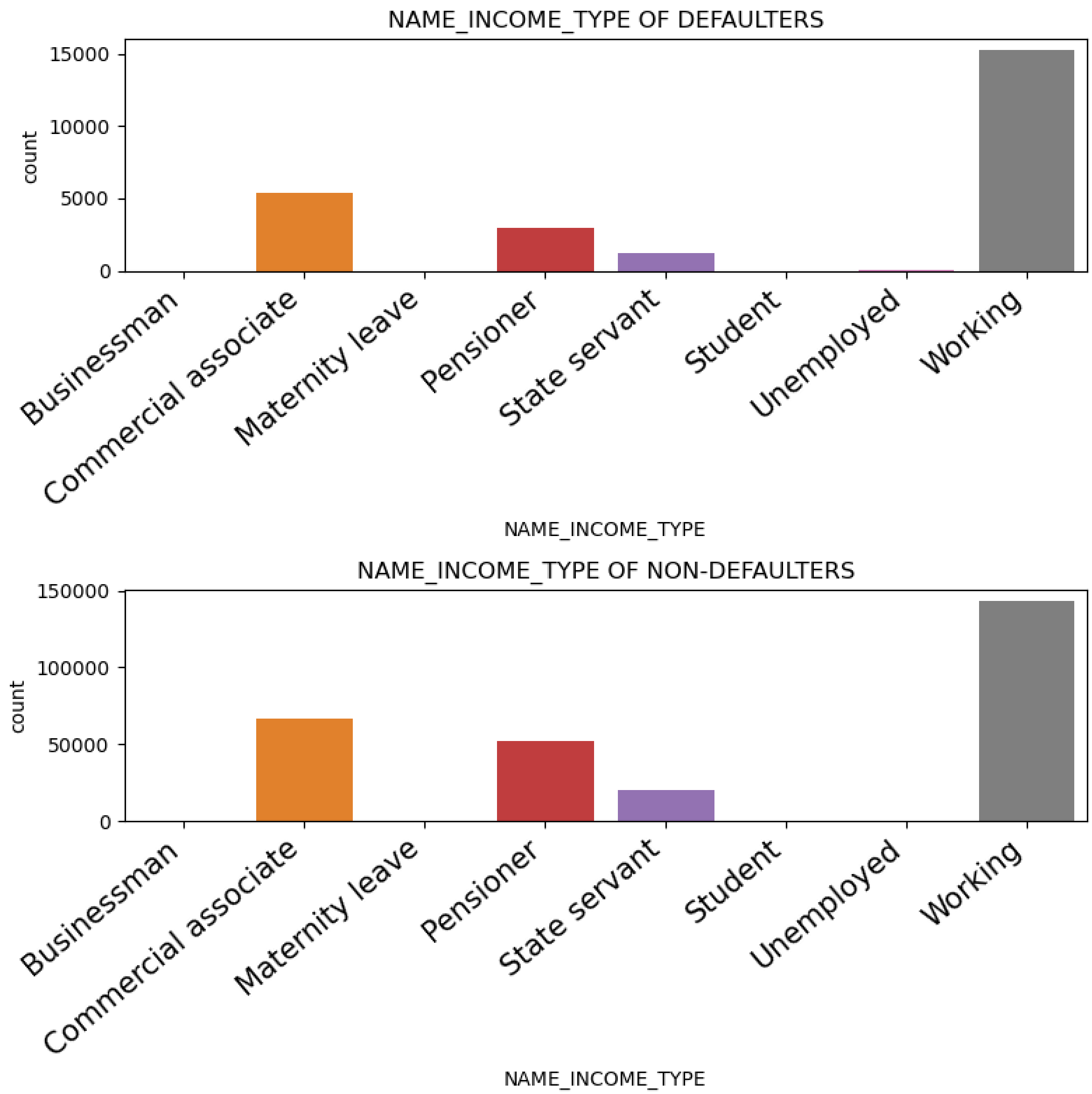
```
# Plotting if applicant owns car
plot_univariate_pie("FLAG_OWN_CAR")
```



- Concluding that applicants who owns car are less likely to be deafulters.

In [61]:

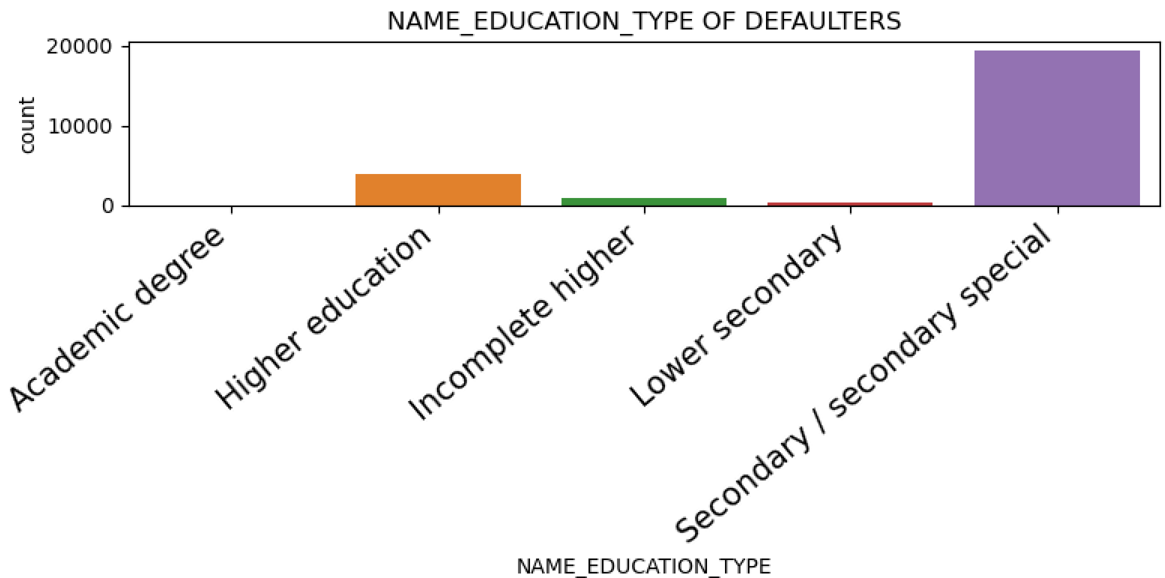
```
# Plotting applicants income type
plt.figure(figsize=(8,8))
plt.subplot(2,1,1)
plt.title('NAME_INCOME_TYPE OF DEFAULTERS')
ax = sns.countplot(x='NAME_INCOME_TYPE',data=defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()

plt.subplot(2,1,2)
plt.title('NAME_INCOME_TYPE OF NON-DEFAULTERS')
ax = sns.countplot(x='NAME_INCOME_TYPE',data=non_defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()
plt.show()
```

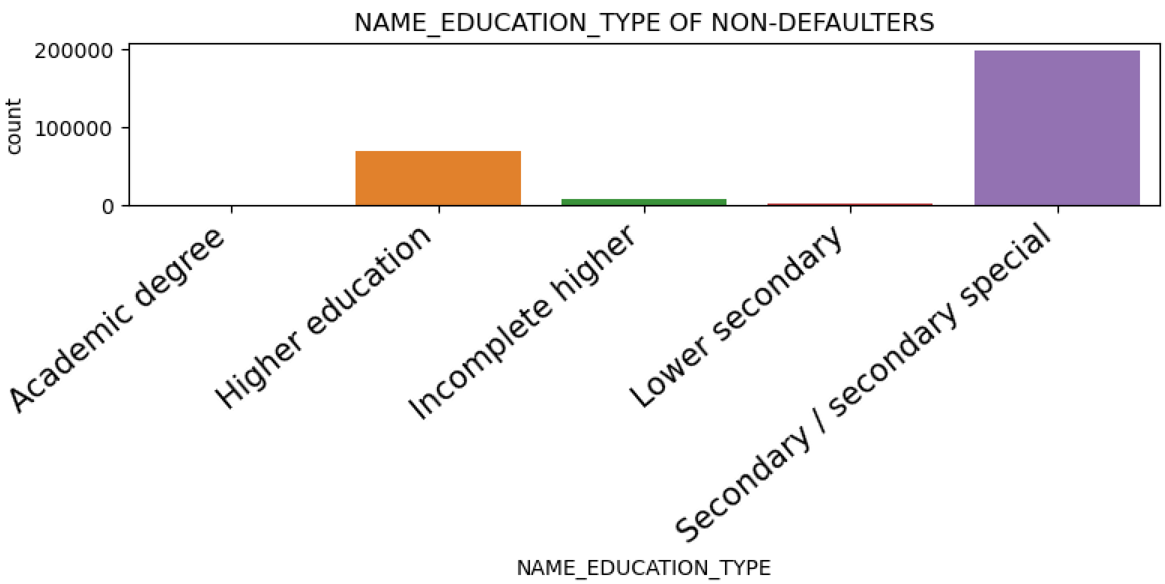


- Most loans taken by Working followed by Commercial associate applicants.
- Applicants with income type Maternity leave are highest defaulters followed by Unemployed.
- Students and Businessman are safest to give loans

In [62]:

```
# Plotting Level of highest education the applicants achieved
plt.figure(figsize=(8,8))
plt.subplot(2,1,1)
plt.title('NAME_EDUCATION_TYPE OF DEFAULTERS')
ax = sns.countplot(x='NAME_EDUCATION_TYPE',data=defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()

plt.subplot(2,1,2)
plt.title('NAME_EDUCATION_TYPE OF NON-DEFAULTERS')
ax = sns.countplot(x='NAME_EDUCATION_TYPE',data=non_defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()
plt.show()
```

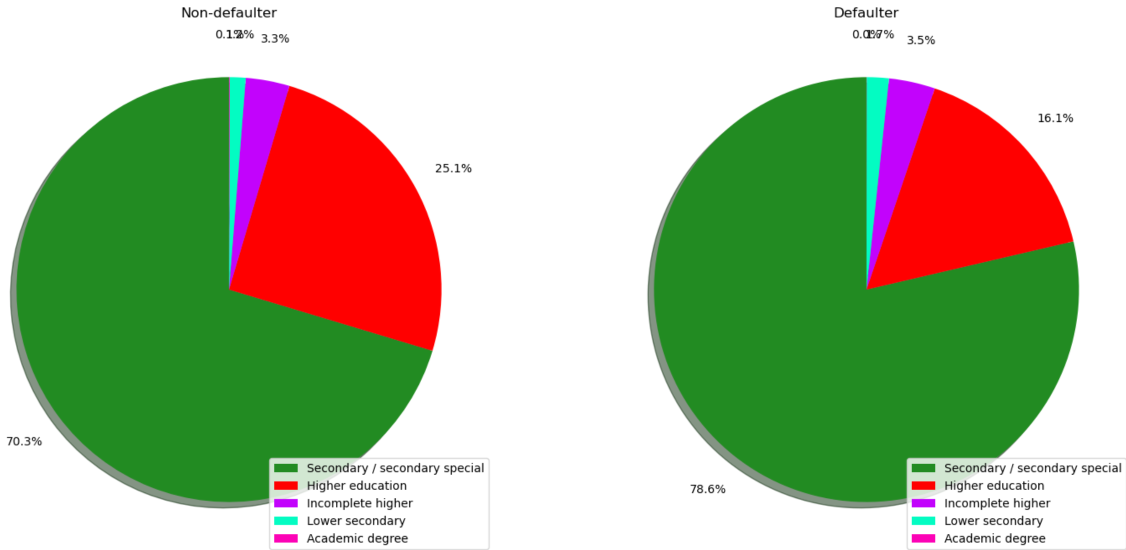




- Applicants with secondary/secondary special education are major defaulters and non_defaulters too, followed by Higher
education

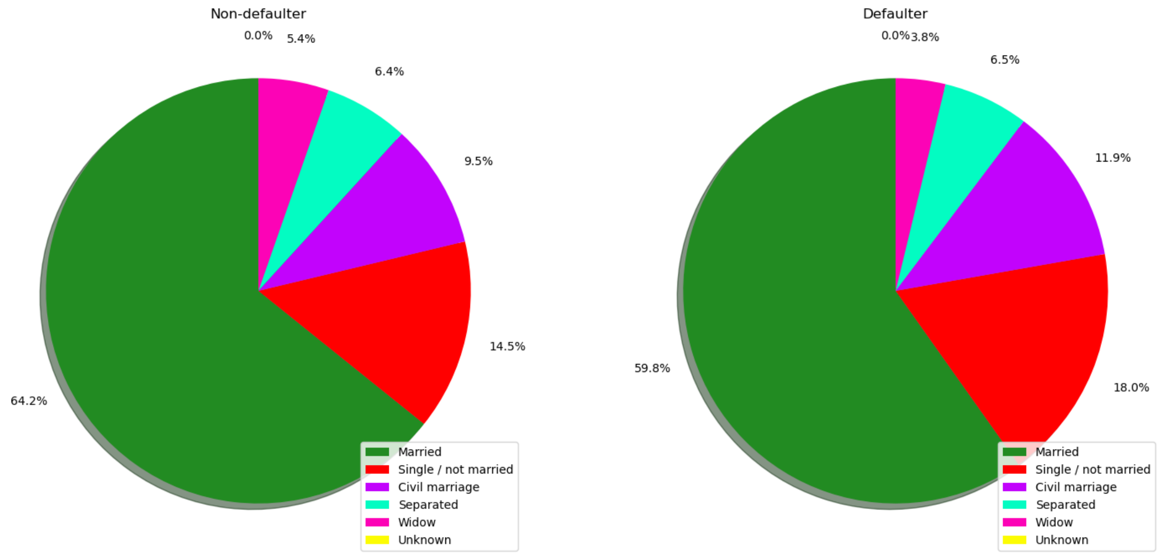
In [63]:

```
# Family status of applicants
plot_univariate_pie("NAME_EDUCATION_TYPE")
```



In [64]:

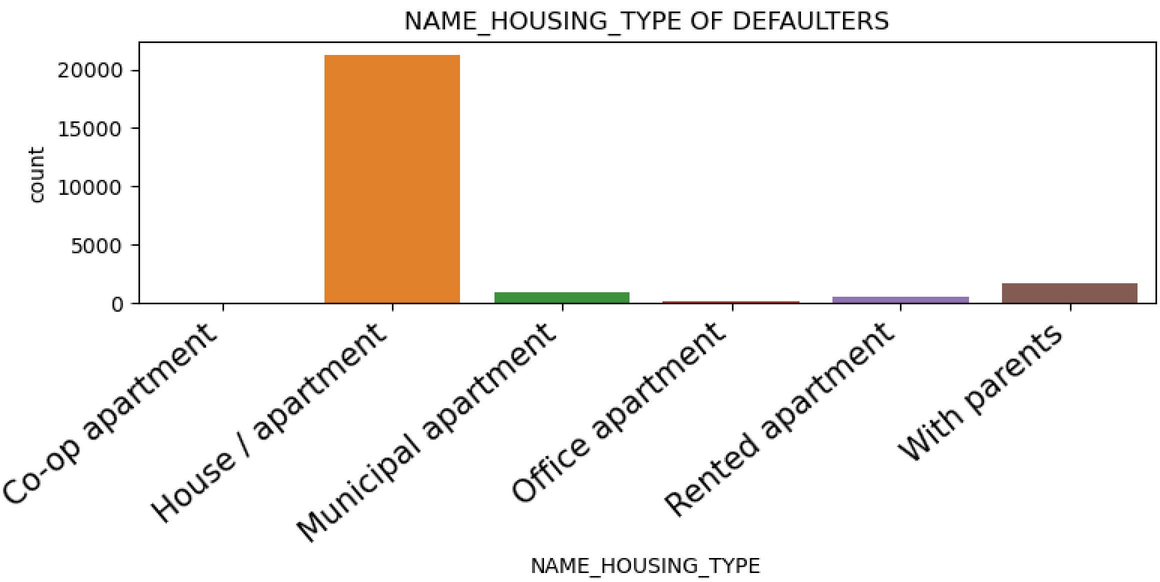
```
# Plotting Family status of applicants
plot_univariate_pie("NAME_FAMILY_STATUS")
```

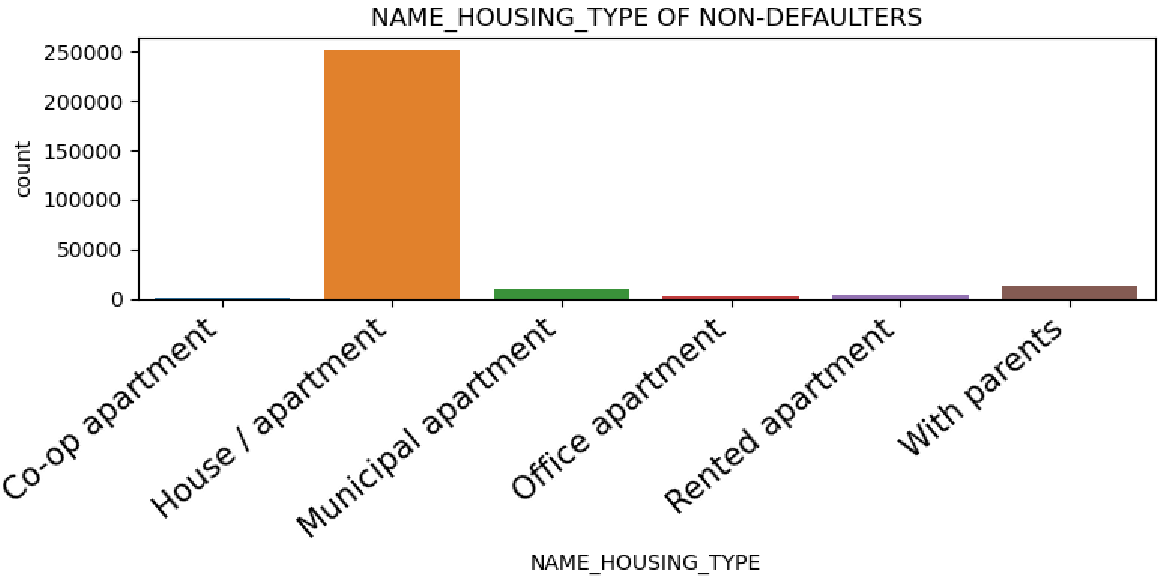


- Most of the applicants who have taken loan are married.
- From pie chart we can observe that risk is more when loan given to Single/not married applicants.

In [65]:

```
# Plotting current housing situation of applicants
plt.figure(figsize=(8,8))
plt.subplot(2,1,1)
plt.title('NAME_HOUSING_TYPE OF DEFAULTERS')
ax = sns.countplot(x='NAME_HOUSING_TYPE',data=defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()

plt.subplot(2,1,2)
plt.title('NAME_HOUSING_TYPE OF NON-DEFAULTERS')
ax = sns.countplot(x='NAME_HOUSING_TYPE',data=non_defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()
plt.show()
```

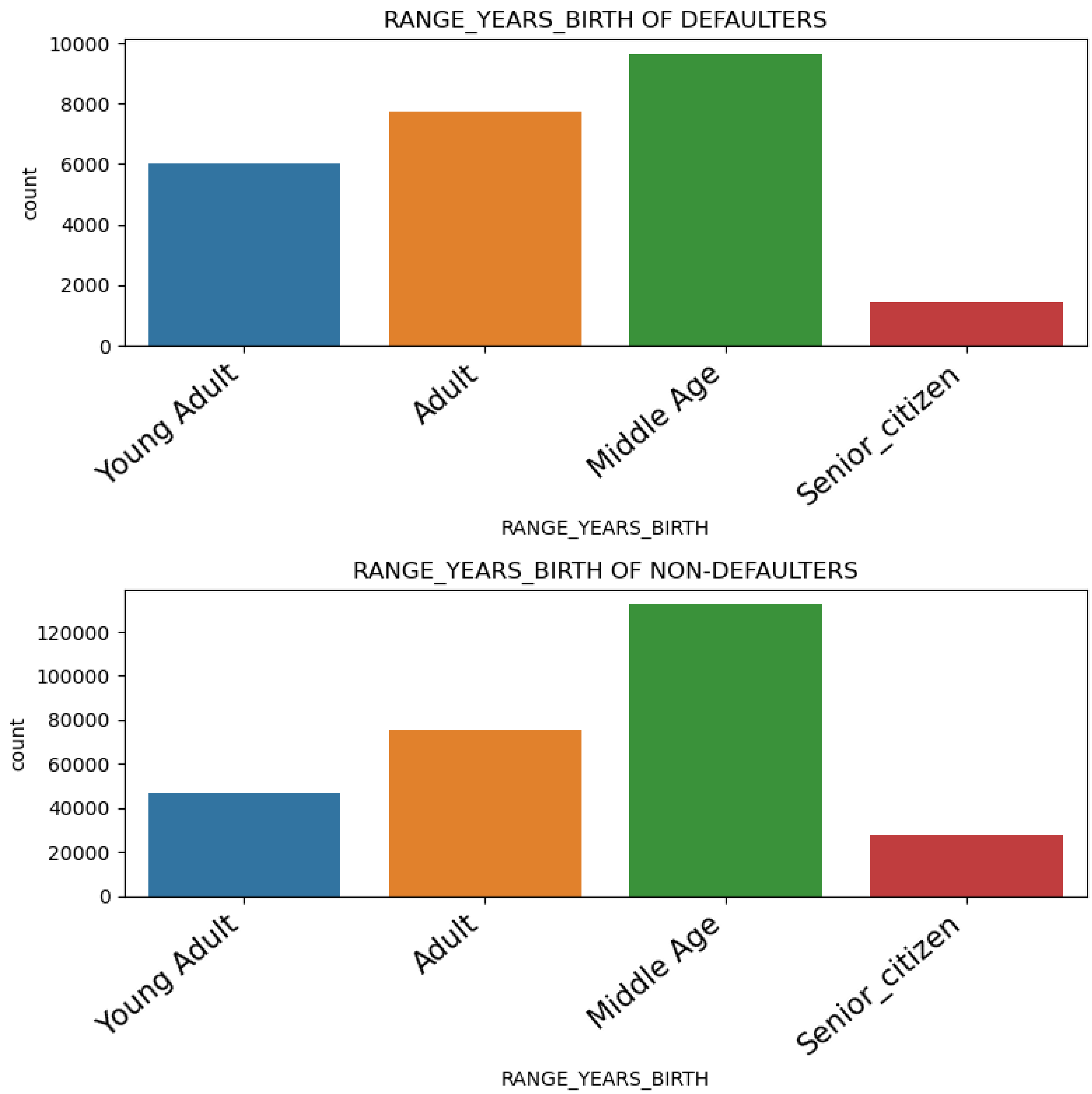




- Applicants mostlty live in House/apartment.
- Applicants living in office apartment have lowest default rate.
- Applicants living in rented apartment and with parents have high probablity of defaulting.

In [66]:

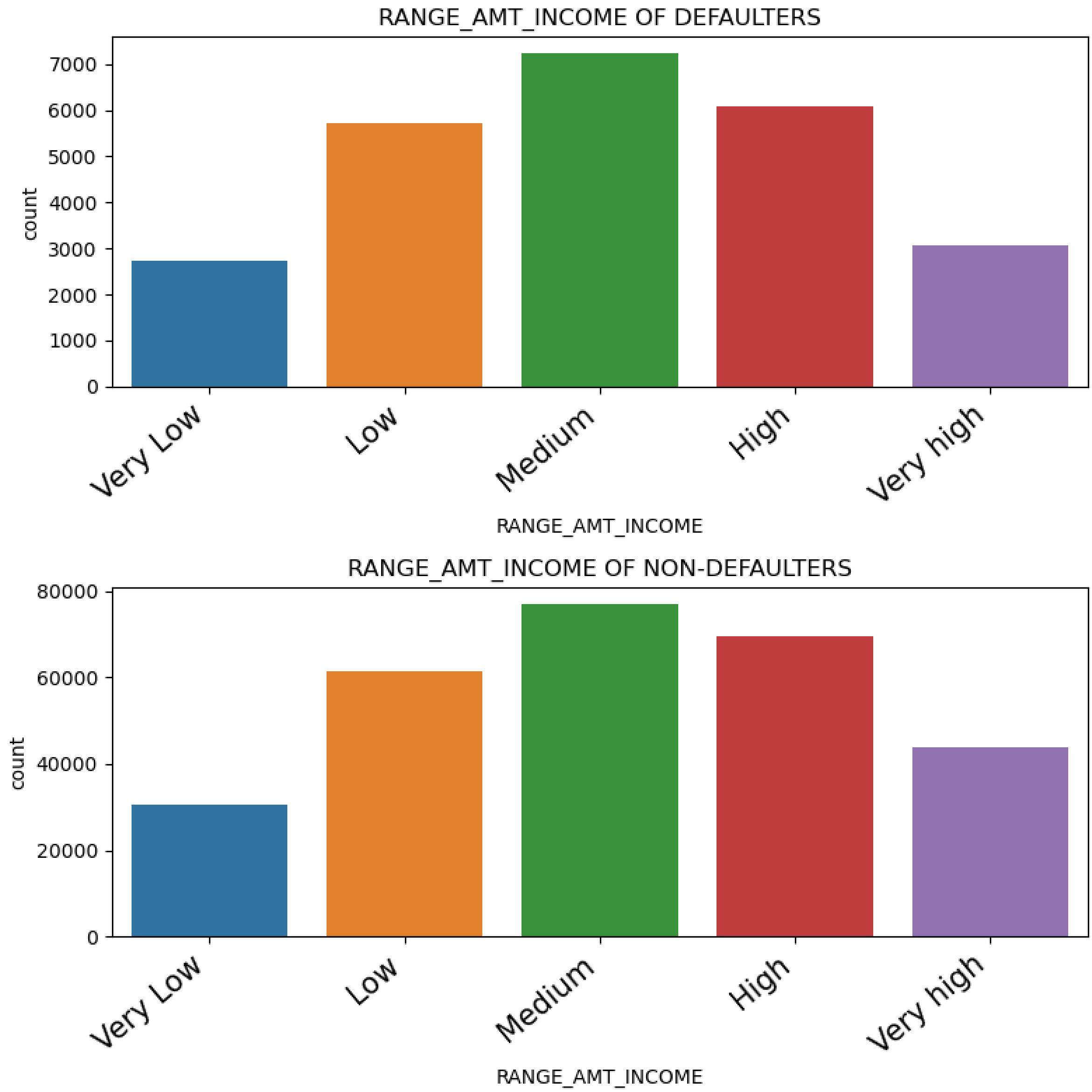
```
# Plotting age groups
plt.figure(figsize=(8,8))
plt.subplot(2,1,1)
plt.title('RANGE_YEARS_BIRTH OF DEFAULTERS')
ax = sns.countplot(x='RANGE_YEARS_BIRTH',data=defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()

plt.subplot(2,1,2)
plt.title('RANGE_YEARS_BIRTH OF NON-DEFAULTERS')
ax = sns.countplot(x='RANGE_YEARS_BIRTH',data=non_defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()
plt.show()
```



- Risk of giving loan to Young adult and Adult are higher
- Applicants with Senior_citizen and Middle Age are safer to give loan.

In [67]:

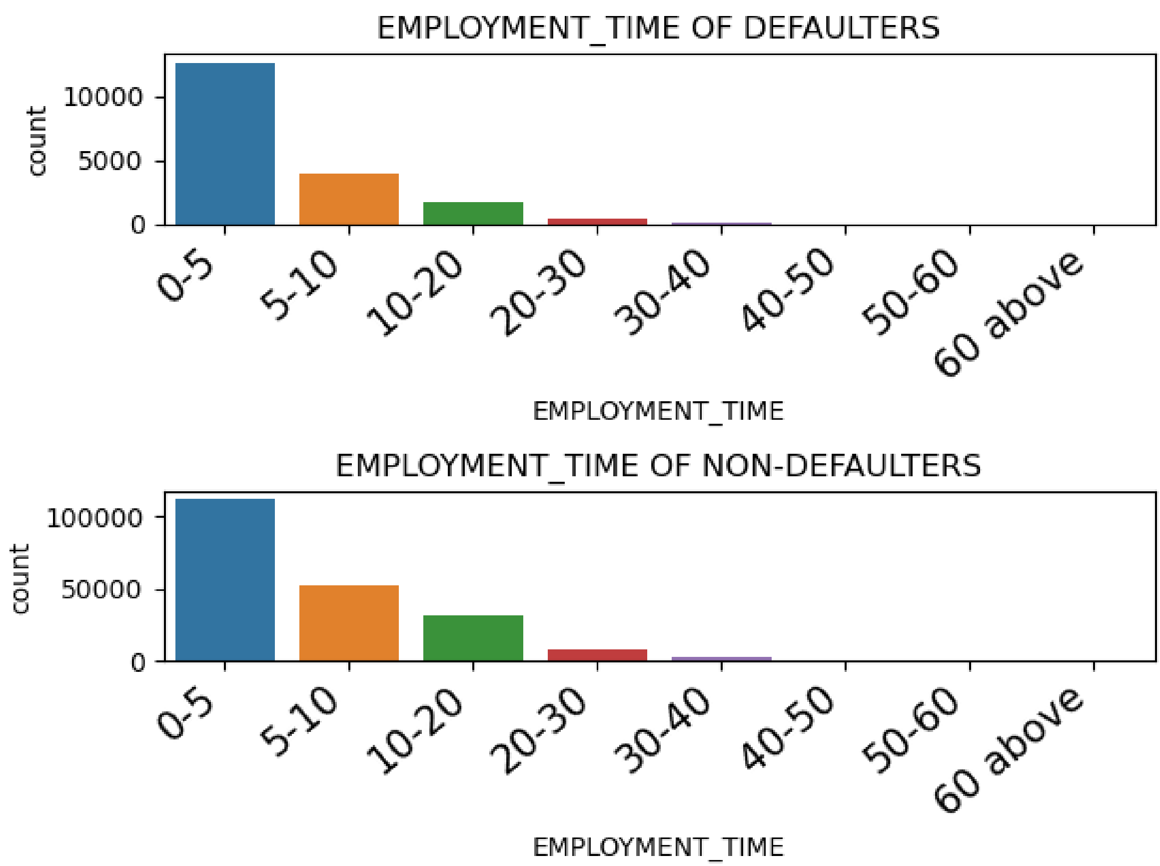
```
plt.figure(figsize=(8,8))
plt.subplot(2,1,1)
plt.title('RANGE_AMT_INCOME OF DEFAULTERS')
ax = sns.countplot(x='RANGE_AMT_INCOME',data=defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()

plt.subplot(2,1,2)
plt.title('RANGE_AMT_INCOME OF NON-DEFAULTERS')
ax = sns.countplot(x='RANGE_AMT_INCOME',data=non_defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()
plt.show()
```



- Applicants with very high income less likely to default

In [68]:

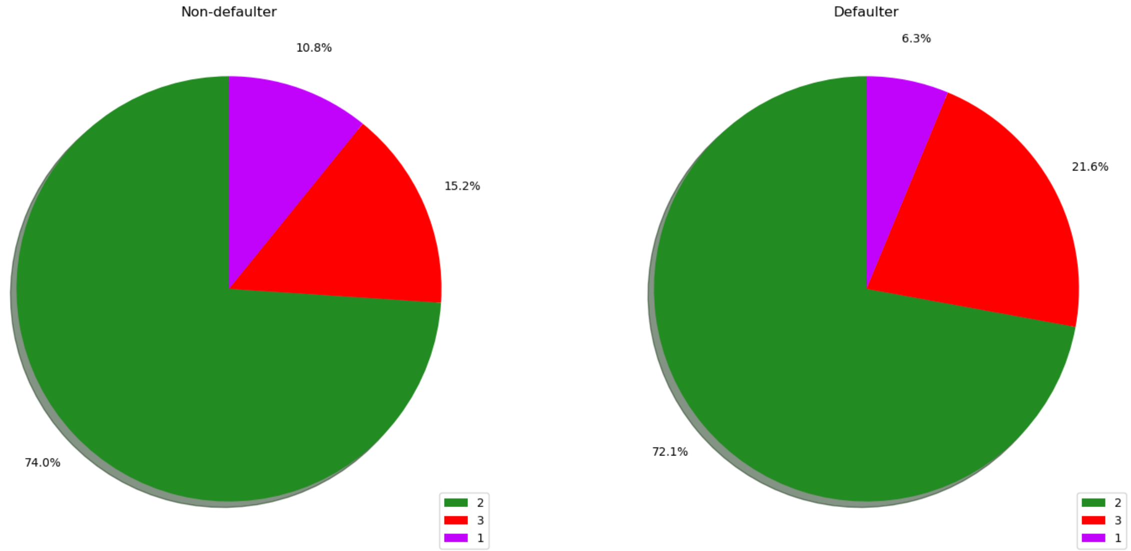
```
plt.subplot(2,1,1)
plt.title('EMPLOYMENT_TIME OF DEFAULTERS')
ax = sns.countplot(x='EMPLOYMENT_TIME',data=defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()

plt.subplot(2,1,2)
plt.title('EMPLOYMENT_TIME OF NON-DEFAULTERS')
ax = sns.countplot(x='EMPLOYMENT_TIME',data=non_defaulter)

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right", fontsize=15)
plt.tight_layout()
plt.show()
```



- Majority of applicants are employed in between 0-5 years and which are also most likely to be defaulters.
- Applicants with 40+ years of experience are non_defaulters.

In [69]:

```
plot_univariate_pie("REGION_RATING_CLIENT")
```



- Applicants with Region rating 3 have highest defaulters percentage
- Applicants wiith Region rating 1 are safer for approving loans.

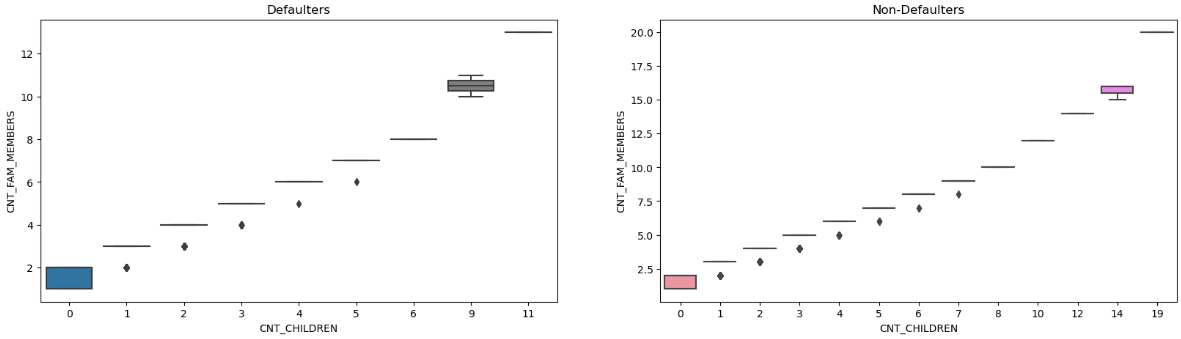
Bivariate Analysis

In [70]:

```
def boxplot_bivariate(variable_1,variable_2):

    plt.figure(figsize=(20,5))
    plt.subplot(1,2,1)
    plt.title('Defaulters')
    sns.boxplot(x=variable_1,y=variable_2,data=defaulter)

    plt.subplot(1,2,2)
    plt.title('Non-Defaulters')
    sns.boxplot(x=variable_1,y=variable_2,data=non_defaulter)

    plt.show()
```

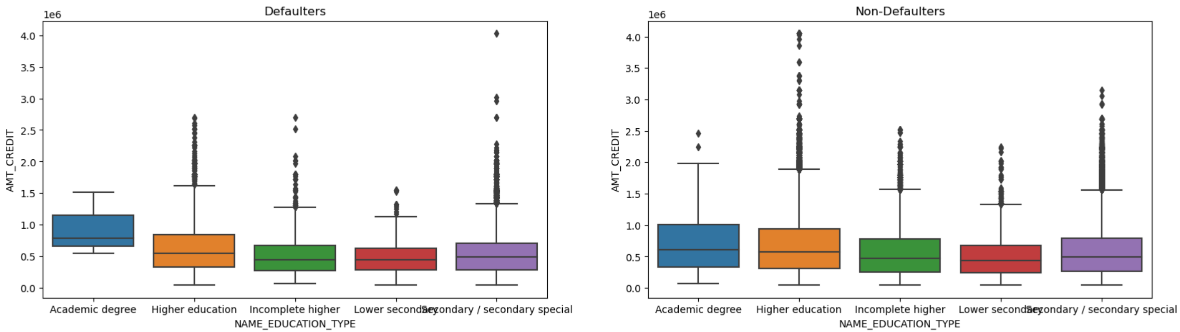
In [71]:

```
boxplot_bivariate('CNT_CHILDREN','CNT_FAM_MEMBERS')
```



In [72]:

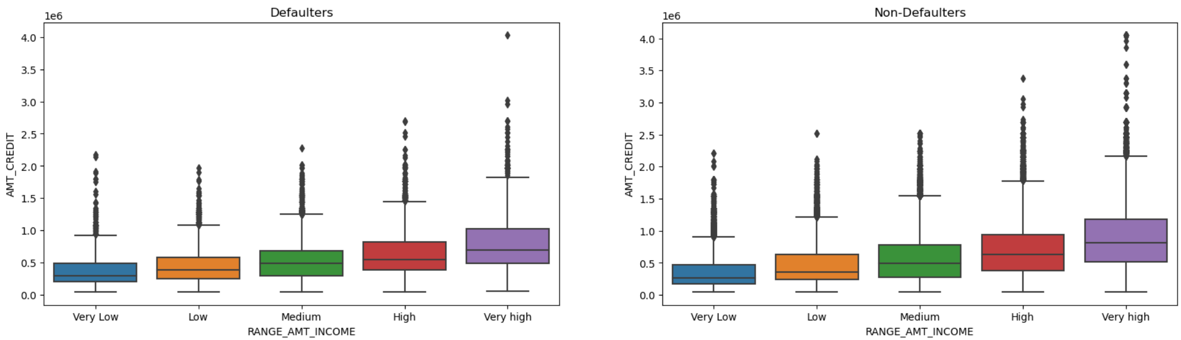
```
# Plotting Amount of credit and Education
boxplot_bivariate('NAME_EDUCATION_TYPE','AMT_CREDIT')
```



- Applicants with High education and Academic Degree have more credit

In [73]:

```
# Plotting Amount of credit and Range of Income
boxplot_bivariate('RANGE_AMT_INCOME','AMT_CREDIT')
```



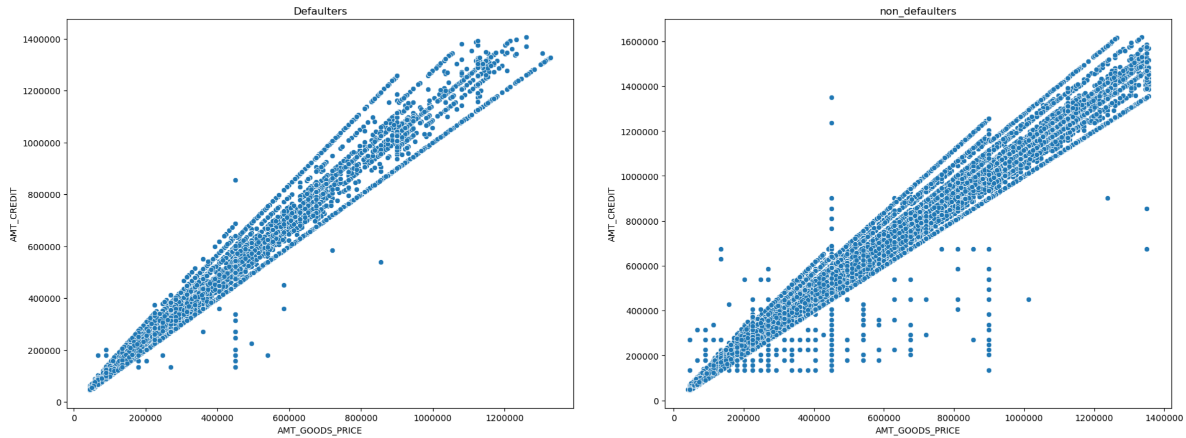
- Applicants with high income have high credit

In [74]:

```
def outlier_range(dataset,column):
    Q1 = dataset[column].quantile(0.25)
    Q3 = dataset[column].quantile(0.75)
    IQR = Q3 - Q1
    Min_value = (Q1 - 1.5 * IQR)
    Max_value = (Q3 + 1.5 * IQR)
    return Max_value
```

In [75]:

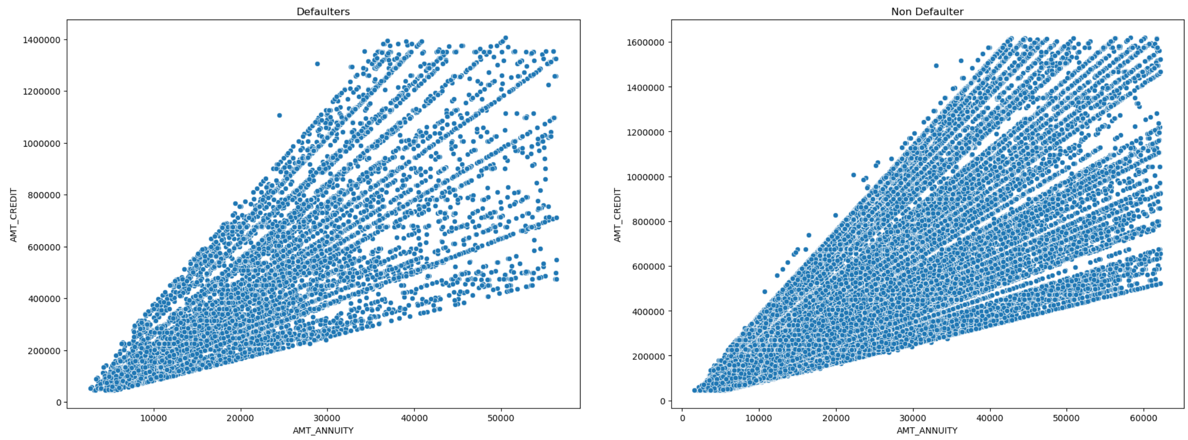
```
# Plotting scatterplot AMT_GOODS_PRICE vs AMT_CREDIT
max_1_AMT_GOODS_PRICE = outlier_range(defaulter,'AMT_GOODS_PRICE')
max_1_AMT_CREDIT = outlier_range(defaulter,'AMT_CREDIT')
max_0_AMT_GOODS_PRICE = outlier_range(non_defaulter,'AMT_GOODS_PRICE')
max_0_AMT_CREDIT = outlier_range(non_defaulter,'AMT_CREDIT')

plt.figure(figsize = [20,8])
plt.subplot(1,2,1)
plt.title('Defaulters')
sns.scatterplot(x = defaulter[defaulter['AMT_GOODS_PRICE'] < max_1_AMT_GOODS_PRICE].AMT_GOO
                y = defaulter[defaulter['AMT_CREDIT'] < max_1_AMT_CREDIT].AMT_CREDIT, data
plt.ticklabel_format(style='plain', axis='x')
plt.ticklabel_format(style='plain', axis='y')

plt.subplot(1,2,2)
plt.title('non_defaulters')
sns.scatterplot(x = non_defaulter[non_defaulter['AMT_GOODS_PRICE'] < max_0_AMT_GOODS_PRICE]
                y = non_defaulter[non_defaulter['AMT_CREDIT'] < max_0_AMT_CREDIT].AMT_CREDI
plt.ticklabel_format(style='plain', axis='x')
plt.ticklabel_format(style='plain', axis='y')

plt.tight_layout(pad = 4)
plt.show()
```



- Both columns holds strong correlation, which means as Goods price increases Credit amount also increases.

In [76]:

```
# Plotting scatterplot AMT_ANNUITY vs AMT_CREDIT
max_1_AMT_ANNUITY = outlier_range(defaulter,'AMT_ANNUITY')
max_1_AMT_CREDIT = outlier_range(defaulter,'AMT_CREDIT')
max_0_AMT_ANNUITY = outlier_range(non_defaulter,'AMT_ANNUITY')
max_0_AMT_CREDIT = outlier_range(non_defaulter,'AMT_CREDIT')

plt.figure(figsize = [20,8])
plt.subplot(1,2,1)
plt.title('Defaulters')
sns.scatterplot(x = defaulter[defaulter['AMT_ANNUITY'] < max_1_AMT_ANNUITY].AMT_ANNUITY,
                y = defaulter[defaulter['AMT_CREDIT'] < max_1_AMT_CREDIT].AMT_CREDIT, data
plt.ticklabel_format(style='plain', axis='x')
plt.ticklabel_format(style='plain', axis='y')

plt.subplot(1,2,2)
plt.title('Non Defaulter')
sns.scatterplot(x = non_defaulter[non_defaulter['AMT_ANNUITY'] < max_0_AMT_ANNUITY].AMT_ANN
                y = non_defaulter[non_defaulter['AMT_CREDIT'] < max_0_AMT_CREDIT].AMT_CREDI
plt.ticklabel_format(style='plain', axis='x')
plt.ticklabel_format(style='plain', axis='y')

plt.tight_layout(pad = 4)
plt.show()
```



- Above scatterplot means as Annuity amount increases, Credit amount also increases.

Top 10 correlation of the selected columns

In [77]:

```
# Top 10 correlation of Defaulters
defaulter_cor = defaulter.corr()
defaulter_cor = defaulter_cor.where(np.triu(np.ones(defaulter_cor.shape),k=1).astype(np.boo
defaulter_cor_app = defaulter_cor.unstack().reset_index()
defaulter_cor_app.columns =['Column_1','Column_2','Correlation']
defaulter_cor_app.dropna(subset = ["Correlation"])
defaulter_cor_app["Correlation"]=defaulter_cor_app["Correlation"].abs()
defaulter_cor_app.sort_values(by='Correlation', ascending=False, inplace=True)
defaulter_cor_app.head(10)
```

...

Previous Application Data

In [78]:

```
pre_data.shape
```

Out[78]:

(1670214, 37)

In [79]:

```
# Removing columns with missing values >=50%
null_pre_app=pd.DataFrame((pre_data.isnull().sum())*100/len(pre_data)).reset_index()
null_pre_app.columns = ['Column Name', 'Null Percentage']
null_50predata= round(null_pre_app[null_pre_app["Null Percentage"]>=50],2)
null_50predata
```

Out[79]:

	Column Name	Null Percentage
6	AMT_DOWN_PAYMENT	53.64
12	RATE_DOWN_PAYMENT	53.64
13	RATE_INTEREST_PRIMARY	99.64
14	RATE_INTEREST_PRIVILEGED	99.64

In [80]:

```
col_pre_del= null_50predata["Column Name"].tolist()
pre_data.drop(col_pre_del,axis=1,inplace=True)
```

In [81]:

```
pre_data.shape
```

Out[81]:

(1670214, 33)

In [82]:

```
pre_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 33 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   SK_ID_PREV                   1670214 non-null  int64  
 1   SK_ID_CURR                   1670214 non-null  int64  
 2   NAME_CONTRACT_TYPE           1670214 non-null  object 
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_GOODS_PRICE              1284699 non-null  float64
 7   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object 
 8   HOUR_APPR_PROCESS_START      1670214 non-null  int64  
 9   FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object 
 10  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64  
 11  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object 
 12  NAME_CONTRACT_STATUS         1670214 non-null  object 
 13  DAYS_DECISION                1670214 non-null  int64  
 14  NAME_PAYMENT_TYPE            1670214 non-null  object 
 15  CODE_REJECT_REASON           1670214 non-null  object 
 16  NAME_TYPE_SUITE              849809 non-null   object 
 17  NAME_CLIENT_TYPE             1670214 non-null  object 
 18  NAME_GOODS_CATEGORY          1670214 non-null  object 
 19  NAME_PORTFOLIO               1670214 non-null  object 
 20  NAME_PRODUCT_TYPE            1670214 non-null  object 
 21  CHANNEL_TYPE                 1670214 non-null  object 
 22  SELLERPLACE_AREA             1670214 non-null  int64  
 23  NAME_SELLER_INDUSTRY         1670214 non-null  object 
 24  CNT_PAYMENT                  1297984 non-null  float64
 25  NAME_YIELD_GROUP             1670214 non-null  object 
 26  PRODUCT_COMBINATION          1669868 non-null  object 
 27  DAYS_FIRST_DRAWING           997149 non-null   float64
 28  DAYS_FIRST_DUE               997149 non-null   float64
 29  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 30  DAYS_LAST_DUE                997149 non-null   float64
 31  DAYS_TERMINATION             997149 non-null   float64
 32  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(11), int64(6), object(16)
memory usage: 420.5+ MB
```

In [83]:

```
Cat_col_p = ['NAME_CASH_LOAN_PURPOSE','NAME_CONTRACT_STATUS','NAME_PAYMENT_TYPE',
                    'CODE_REJECT_REASON','NAME_CLIENT_TYPE','NAME_GOODS_CATEGORY','NAME_POR
                   'NAME_PRODUCT_TYPE','CHANNEL_TYPE','NAME_SELLER_INDUSTRY','NAME_YIELD_GR
                    'NAME_CONTRACT_TYPE']

for col in Cat_col_p:
    pre_data[col] =pd.Categorical(pre_data[col])
```

Univariate analysis

In [84]:

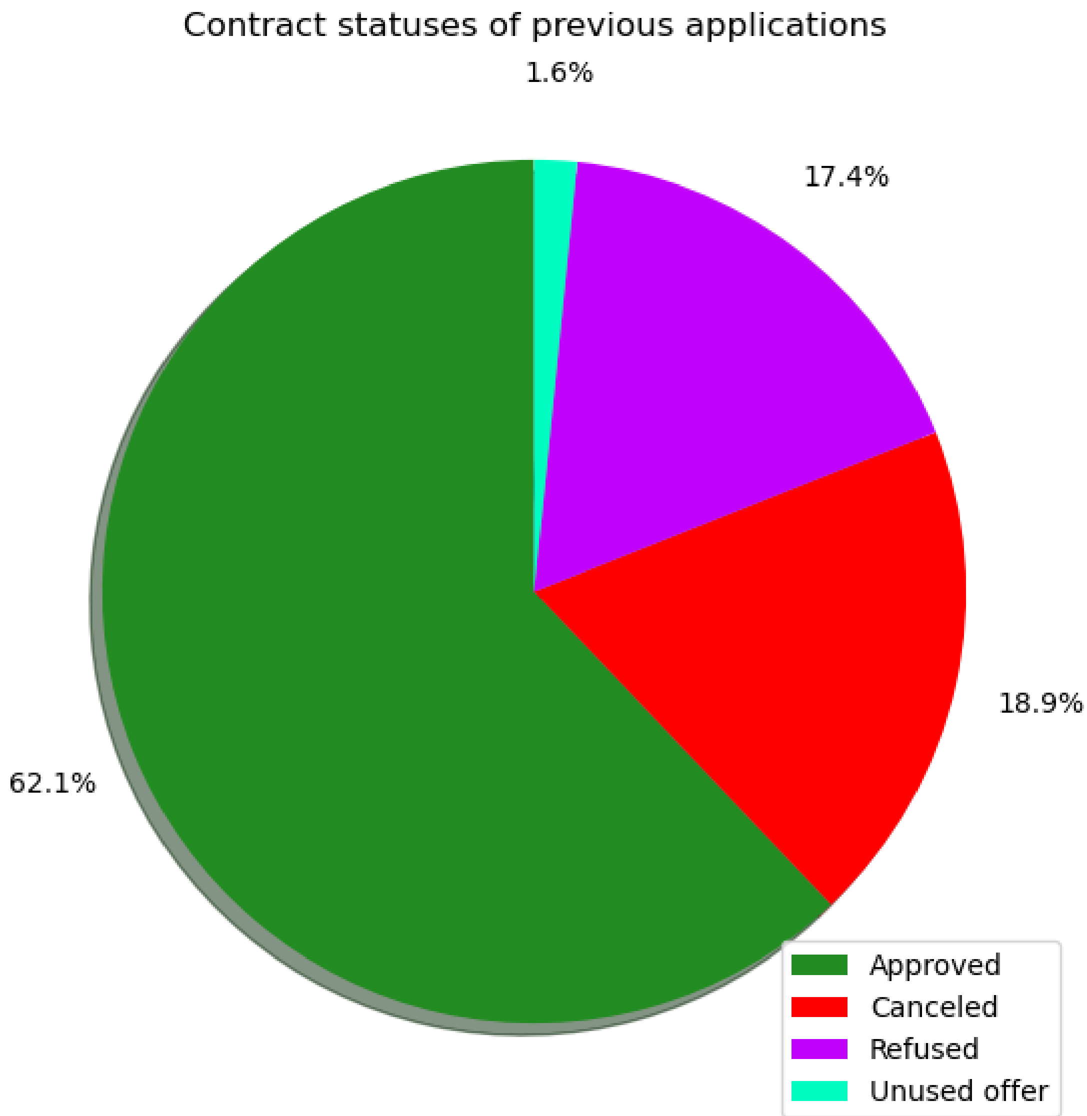
```
pre_data['NAME_CONTRACT_STATUS'].value_counts()
```

Out[84]:

```
Approved        1036781
Canceled         316319
Refused          290678
Unused offer      26436
Name: NAME_CONTRACT_STATUS, dtype: int64
```

In [85]:

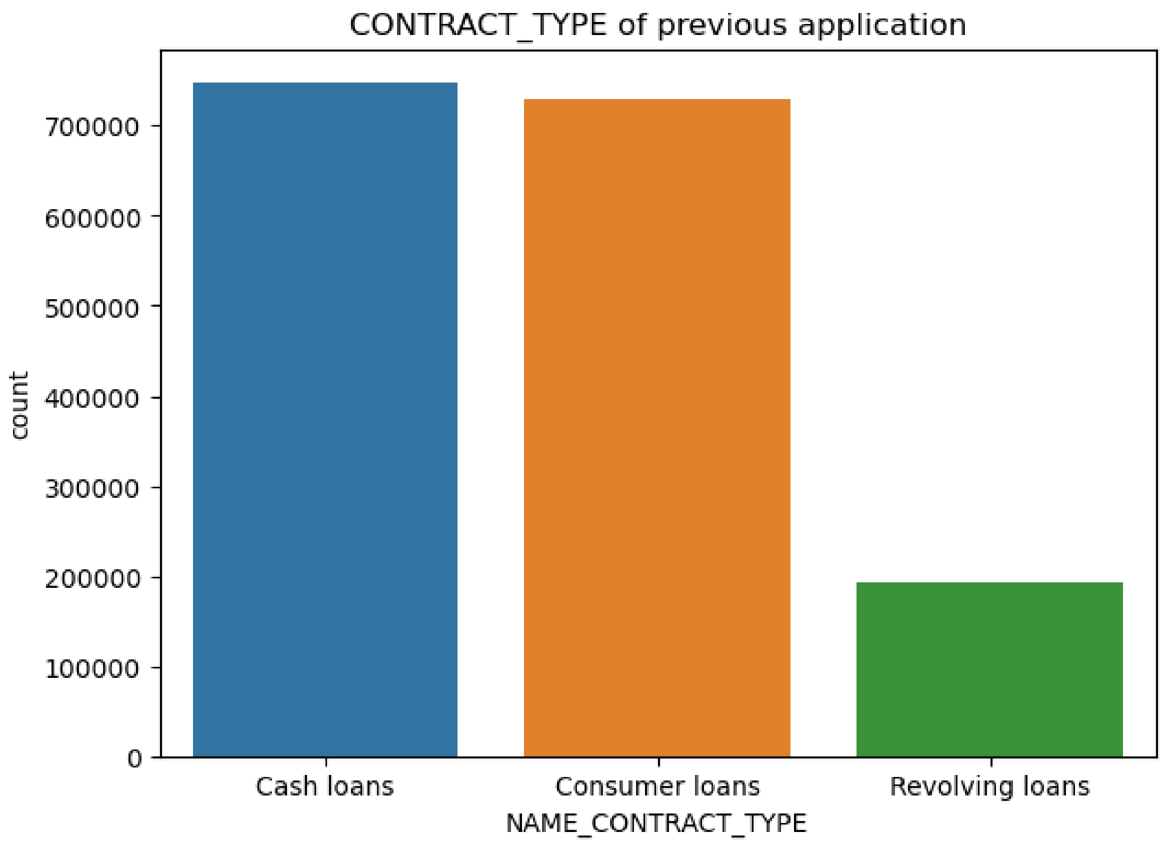
```
#Plotting pie chart for Contract Status
plt.figure(figsize=(10,7))
colors = ('#228B22','#ff0000','#c203fc','#03fcc2','#fc03b6','#fcfc03','#fc9003','#03dbfc')
data = pre_data["NAME_CONTRACT_STATUS"].value_counts()
labels = data.index
plt.pie(data, autopct='%1.1f%%',colors=colors,shadow=True,pctdistance=1.2,
        labeldistance=1.2,startangle=90)
plt.title('Contract statuses of previous applications')
plt.legend(labels, loc="lower right")
plt.show()
```



- Most of the applications are approved and very less percentage of applcations are unused.

In [86]:

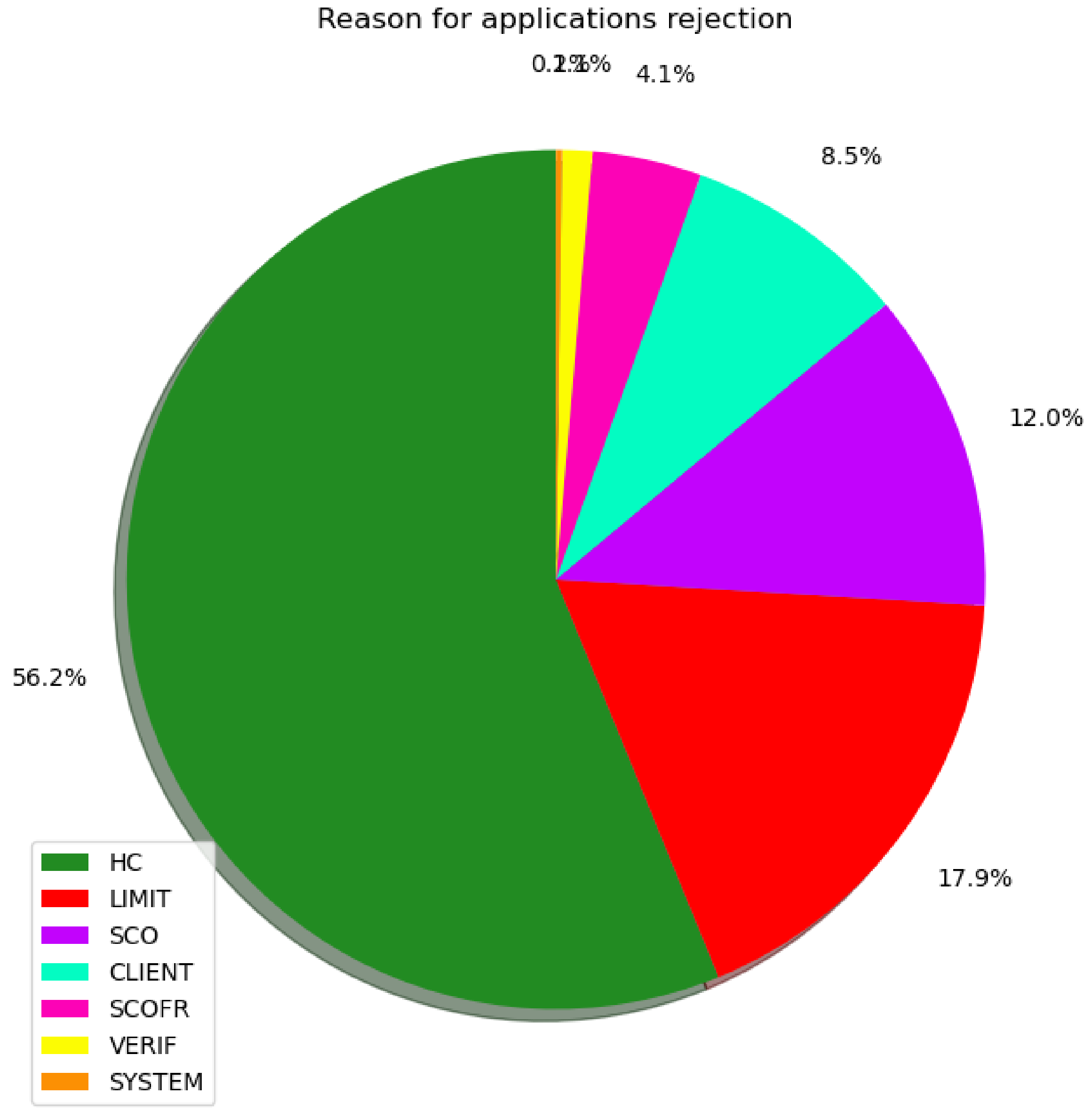
```
# Plotting name contract type
pre_data=pre_data.replace('XNA', np.NaN)
pre_data=pre_data.replace('XAP', np.NaN)
plt.figure(figsize=(7,5))
plt.title('CONTRACT_TYPE of previous application')
sns.countplot(x='NAME_CONTRACT_TYPE',data=pre_data)
plt.show()
```



- We can observe that there are more applications for consumer loans and cash loans than revolving loans.

In [87]:

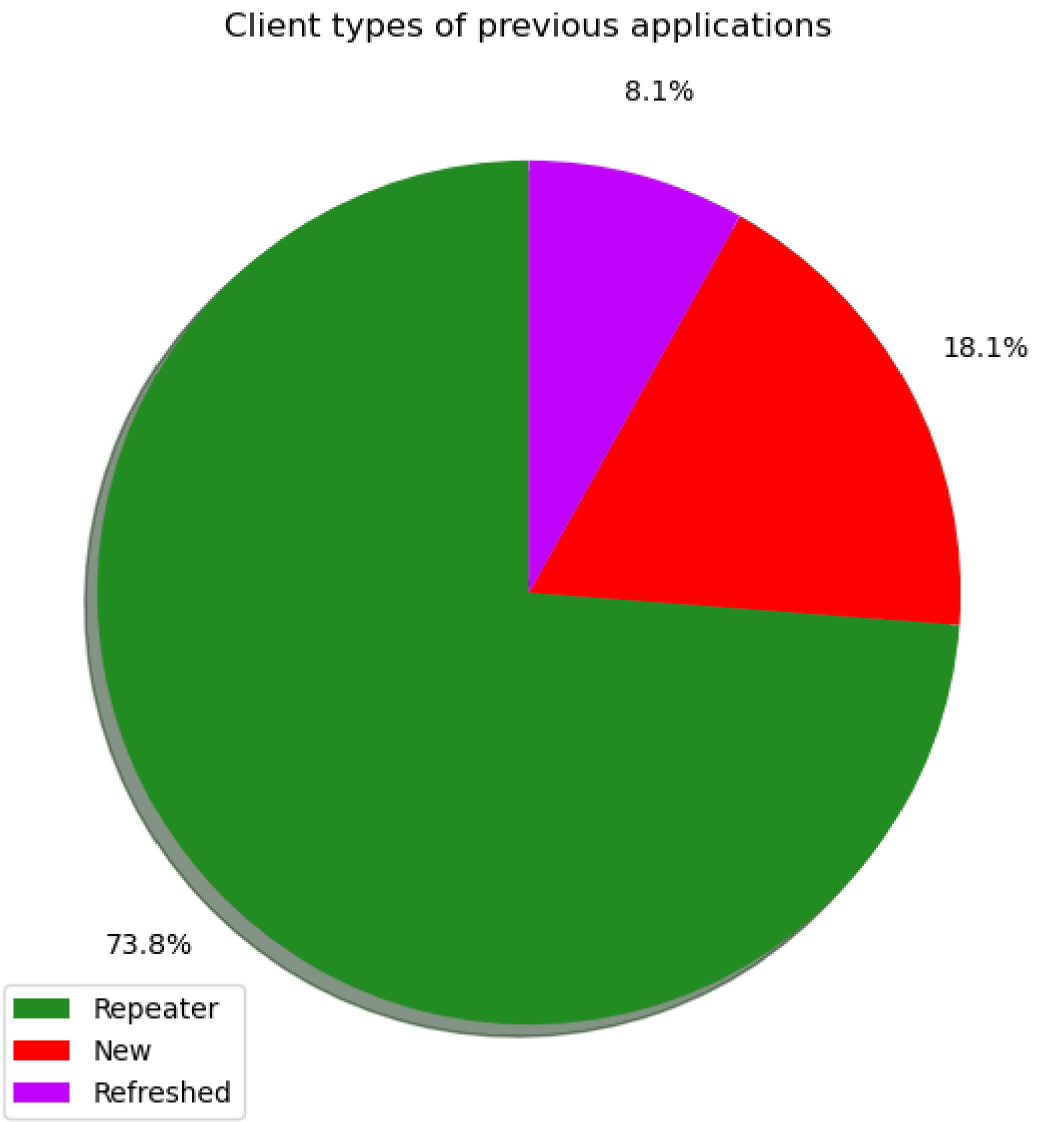
```
# Plotting reason for applications rejection
plt.figure(figsize=(8,8))
colors = ('#228B22','#ff0000','#c203fc','#03fcc2','#fc03b6','#fcfc03','#fc9003','#03dbfc')
data = pre_data["CODE_REJECT_REASON"].value_counts()
labels = data.index
plt.pie(data, autopct='%1.1f%%',colors=colors,shadow=True,pctdistance=1.2,
        labeldistance=1.2,startangle=90)
plt.title('Reason for applications rejection')
plt.legend(labels, loc="lower left")
plt.show()
```



- HC was reason why applications were rejected

In [88]:

```
#Plotting client type
plt.figure(figsize=(7,7))
colors = ('#228B22','#ff0000','#c203fc','#03fcc2','#fc03b6','#fcfc03','#fc9003','#03dbfc')
data = pre_data["NAME_CLIENT_TYPE"].value_counts()
labels = data.index
plt.pie(data, autopct='%1.1f%%',colors=colors,shadow=True,pctdistance=1.2,
        labeldistance=1.2, startangle=90)
plt.title('Client types of previous applications')
plt.legend(labels, loc="lower left")
plt.show()
```



- 73.8% applications are of repeaters.

Top 10 correlation of Previous application data

In [89]:

```
defaulter_cor = pre_data.corr()
defaulter_cor = defaulter_cor.where(np.triu(np.ones(defaulter_cor.shape),k=1).astype(np.boo
defaulter_cor_app = defaulter_cor.unstack().reset_index()
defaulter_cor_app.columns =['Column_1','Column_2','Correlation']
defaulter_cor_app.dropna(subset = ["Correlation"])
defaulter_cor_app["Correlation"]=defaulter_cor_app["Correlation"].abs()
defaulter_cor_app.sort_values(by='Correlation', ascending=False, inplace=True)
defaulter_cor_app.head(10)
```

Out[89]:

	Column_1	Column_2	Correlation
88	AMT_GOODS_PRICE	AMT_APPLICATION	0.999884
89	AMT_GOODS_PRICE	AMT_CREDIT	0.993087
71	AMT_CREDIT	AMT_APPLICATION	0.975824
269	DAYS_TERMINATION	DAYS_LAST_DUE	0.927990
87	AMT_GOODS_PRICE	AMT_ANNUITY	0.820895
70	AMT_CREDIT	AMT_ANNUITY	0.816429
53	AMT_APPLICATION	AMT_ANNUITY	0.808872
232	DAYS_LAST_DUE_1ST_VERSION	DAYS_FIRST_DRAWING	0.803494
173	CNT_PAYMENT	AMT_APPLICATION	0.680630
174	CNT_PAYMENT	AMT_CREDIT	0.674278

Analyzing Application data and Previous application data together

In [90]:

```
combined_data = app_data.merge(pre_data,on='SK_ID_CURR',how='inner')
combined_data.head()
```

Out[90]:

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

In [91]:

```
combined_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1413701 entries, 0 to 1413700
Columns: 110 entries, SK_ID_CURR to NFLAG_INSURED_ON_APPROVAL
dtypes: category(36), float64(31), int32(1), int64(39), object(3)
memory usage: 852.1+ MB
```

In [92]:

```
combined_data.describe()
```

Out[92]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT_x	AM
count	1.413701e+06	1.413701e+06	1.413701e+06	1.413701e+06	1.413701e+06	
mean	2.784813e+05	8.655296e-02	4.048933e-01	1.733160e+05	5.875537e+05	
std	1.028118e+05	2.811789e-01	7.173454e-01	1.985734e+05	3.849173e+05	
min	1.000020e+05	0.000000e+00	0.000000e+00	2.565000e+04	4.500000e+04	
25%	1.893640e+05	0.000000e+00	0.000000e+00	1.125000e+05	2.700000e+05	
50%	2.789920e+05	0.000000e+00	0.000000e+00	1.575000e+05	5.084955e+05	
75%	3.675560e+05	0.000000e+00	1.000000e+00	2.070000e+05	8.079840e+05	
max	4.562550e+05	1.000000e+00	1.900000e+01	1.170000e+08	4.050000e+06	

In [93]:

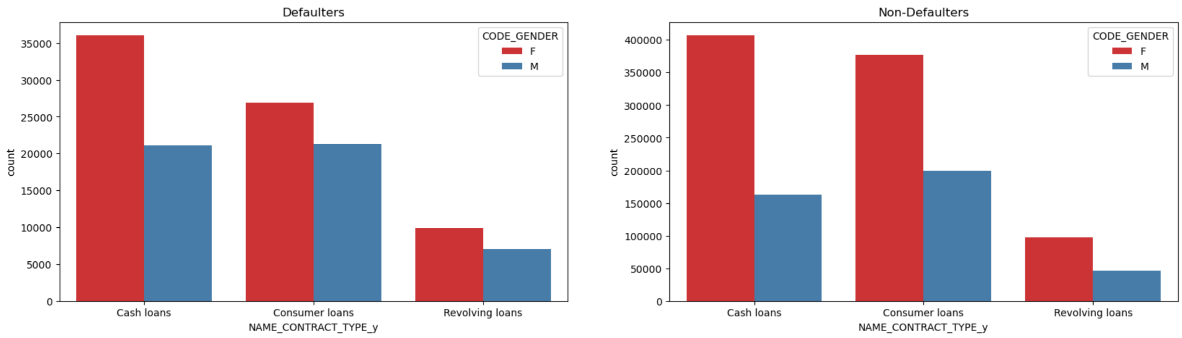
```
def countplot_bivariate(variable_1,variable_2):

    com_0 = combined_data[combined_data['TARGET']==0]
    com_1 = combined_data[combined_data['TARGET']==1]

    plt.figure(figsize=(20,5))
    plt.subplot(1,2,1)
    plt.title('Defaulters')
    sns.countplot(x=variable_1,hue=variable_2,data=com_1,palette='Set1')
    plt.subplot(1,2,2)
    plt.title('Non-Defaulters')
    sns.countplot(x=variable_1,hue=variable_2,data=com_0,palette='Set1')

    plt.show()
```

In [94]:

```
#Plotting Gender and Previous contract status
countplot_bivariate('NAME_CONTRACT_STATUS','CODE_GENDER')
```

In [95]:

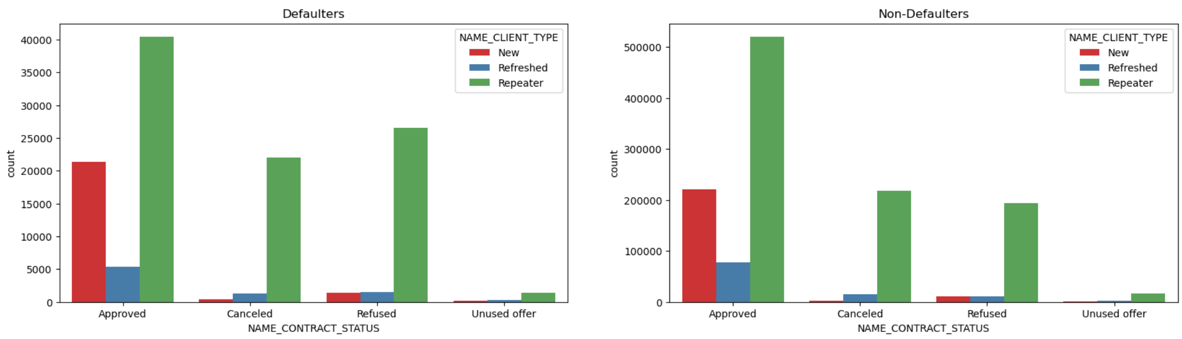
```
countplot_bivariate('NAME_CONTRACT_TYPE_y','CODE_GENDER')
```



- Female applicants have all types of previous contracts.

In [96]:

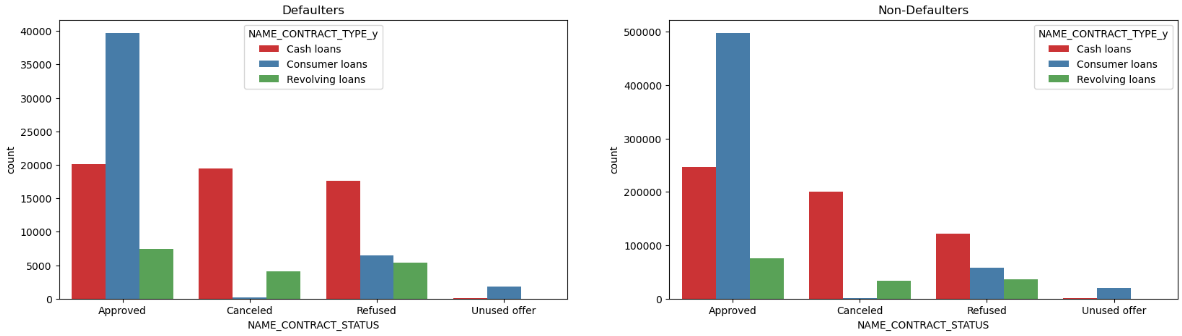
```
countplot_bivariate('NAME_CONTRACT_STATUS','NAME_CLIENT_TYPE')
```



- More 'Approved' previous applications are less likely to be defaulters whereas 'Refused' previous applications are more likely to be defaulters

In [97]:

```
countplot_bivariate('NAME_CONTRACT_STATUS','NAME_CONTRACT_TYPE_y')
```



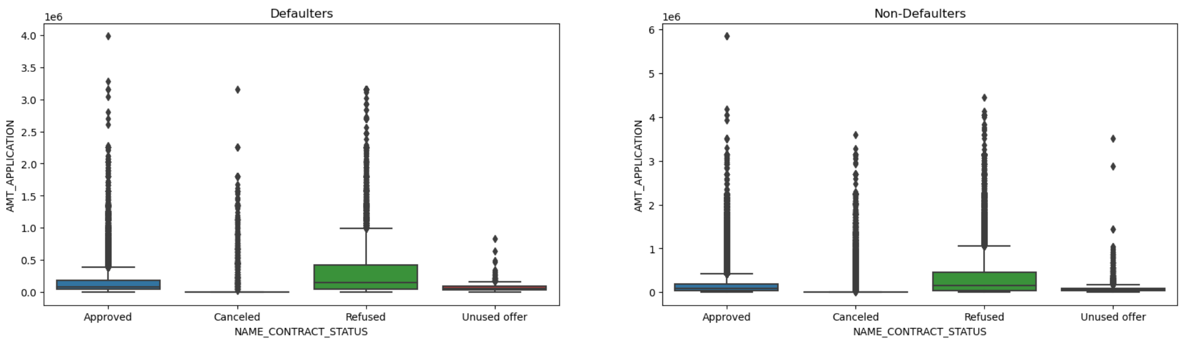
- More 'Consumer loans' previous applications are less likely to be deafulters whereas 'Revolving loans' previous applications are more likely to be defaulters

In [98]:

```
def boxplot_bivariate(variable_1,variable_2):

    com_0 = combined_data[combined_data['TARGET']==0]
    com_1 = combined_data[combined_data['TARGET']==1]

    plt.figure(figsize=(20,5))
    plt.subplot(1,2,1)
    plt.title('Defaulters')
    sns.boxplot(x=variable_1,y=variable_2,data=com_1)

    plt.subplot(1,2,2)
    plt.title('Non-Defaulters')
    sns.boxplot(x=variable_1,y=variable_2,data=com_0)

    plt.show()
```

In [99]:

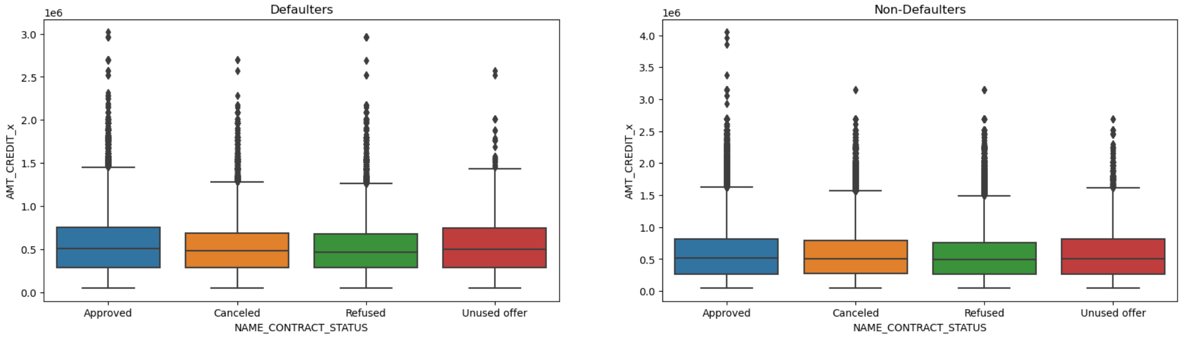
```
boxplot_bivariate('NAME_CONTRACT_STATUS','AMT_APPLICATION')
```



- Similar for both defaulters and non-defaulters, applications being refused had higher credits.

In [100]:

```
boxplot_bivariate('NAME_CONTRACT_STATUS','AMT_CREDIT_x')
```



- As we can see applications are rejected when had higher credits

In [ ]:

Loan Case Study

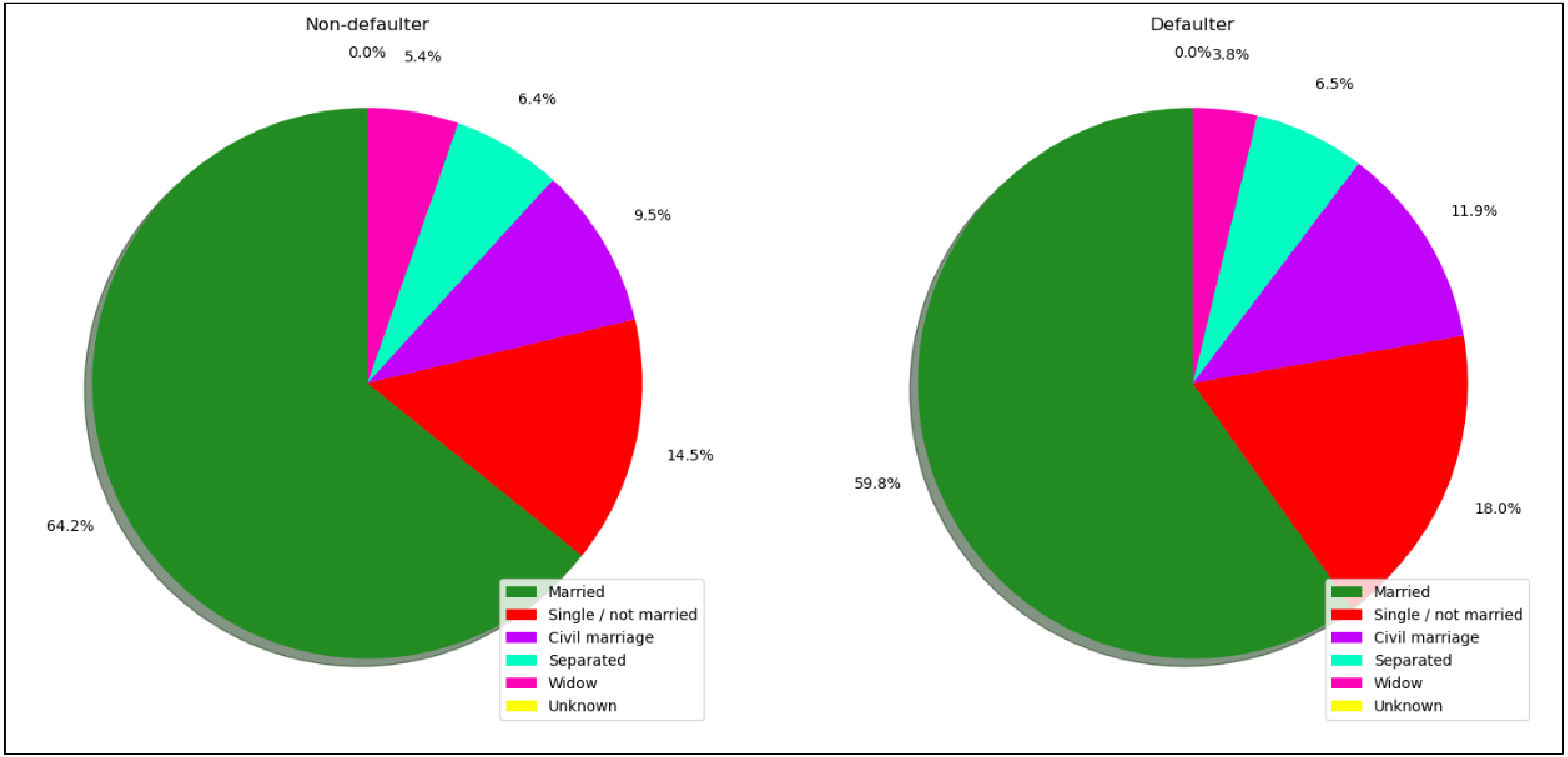
By - Chetan Mahajan

Problem Statement

- To develop a basic understanding of risk analytics in banking and financial services and understand how data is used to minimize the risk of losing money while lending to customers.
- Identification of such applicants who are capable of replaying the loan.
- Identification of applicants who can likely to be defaulters.

Likely Non-Defaulters vs Defaulters

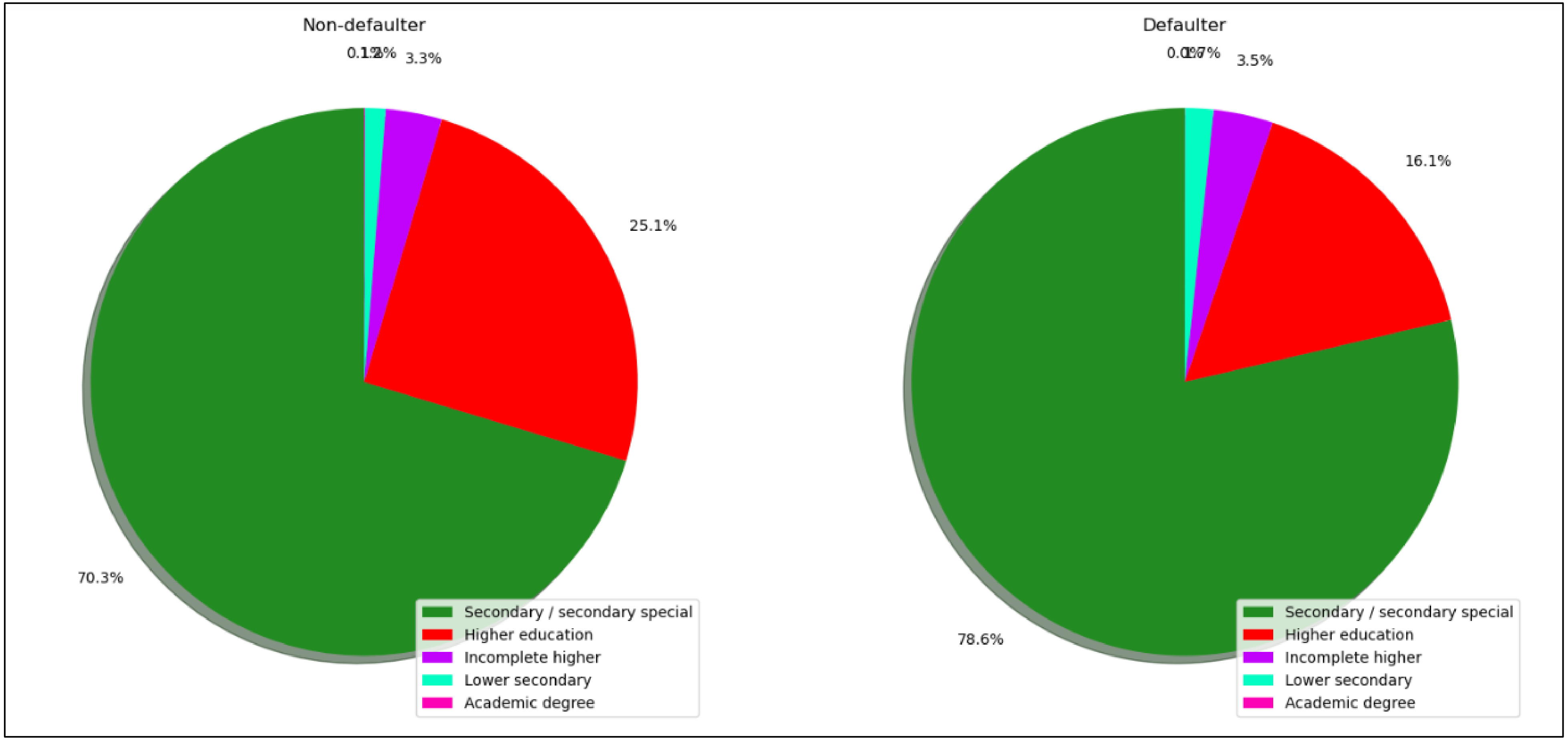
- Based on Relationship status



Preferred choice to approve loans – Married applicants

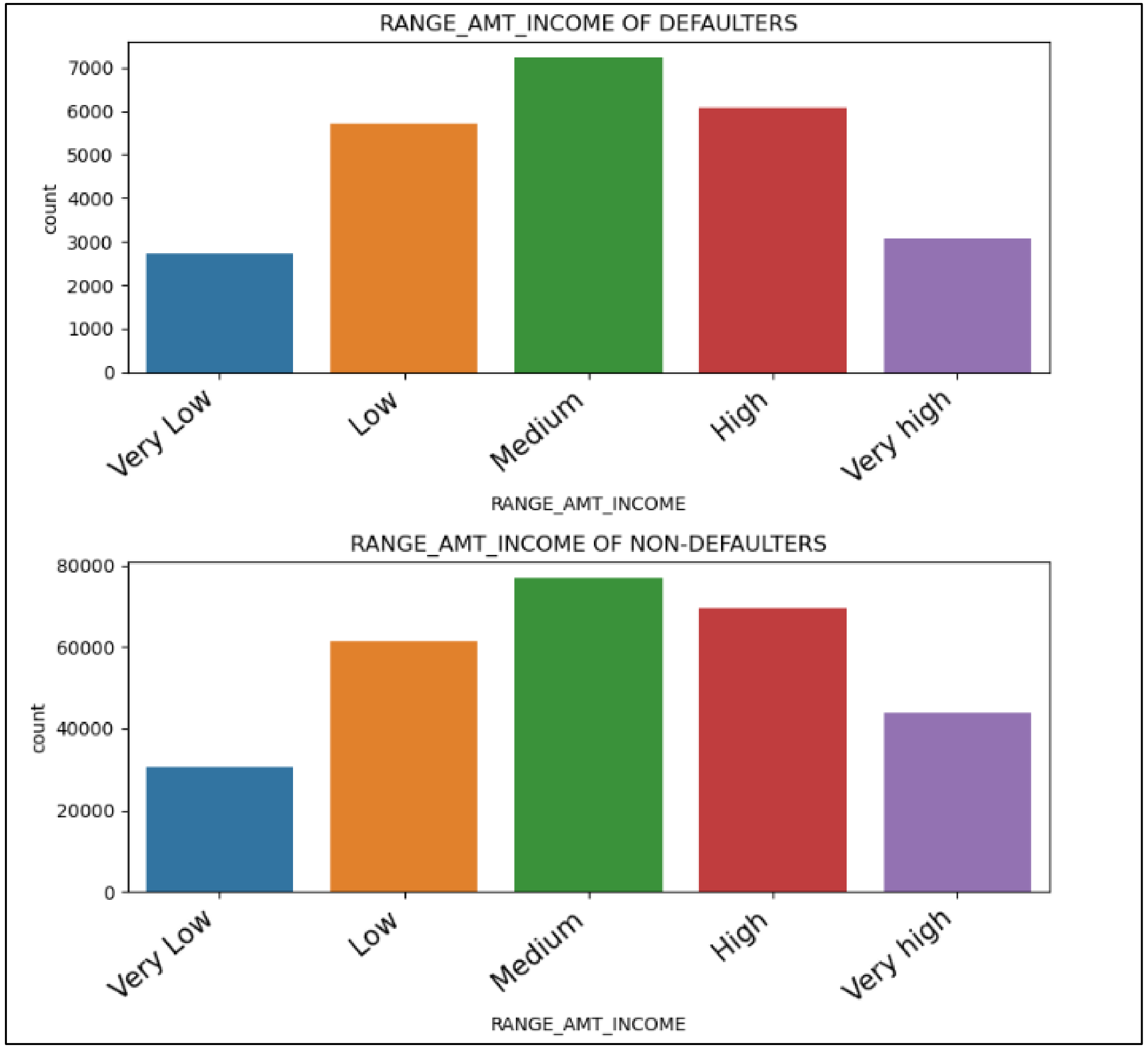
Preferred choice not to approve loans – Single applicants

- Based on Applicants Education



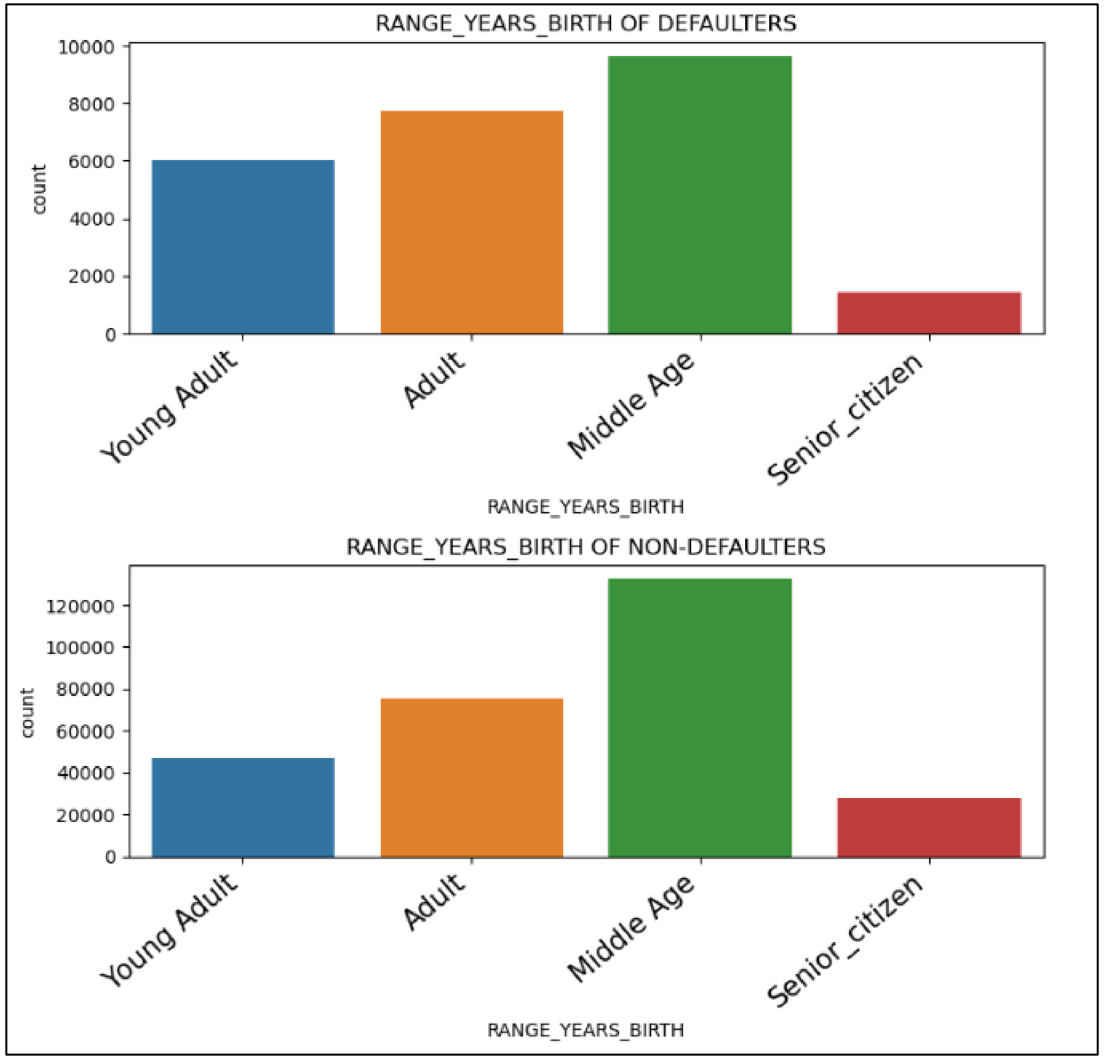
Preferred choice to approve loans – Higher Education
Preferred choice not to approve loans – Secondary Education

- Based on Income of applicants



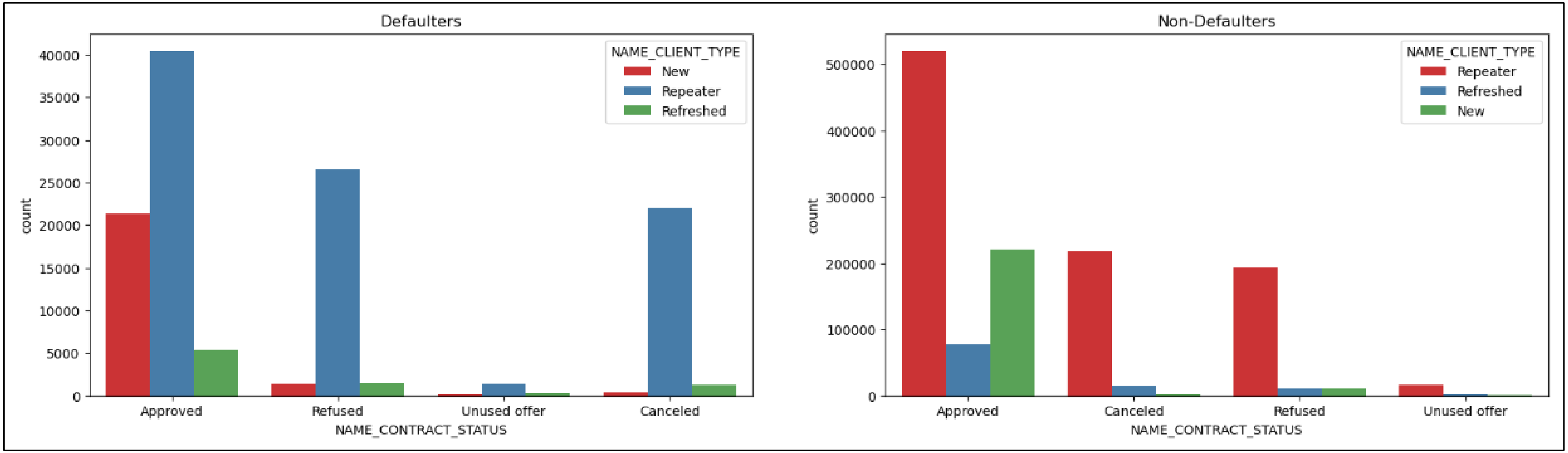
Preferred choice to approve loans – High and Very High Income
Preferred choice not to approve loans – Low and Very Low Income

- Based on Age of applicants



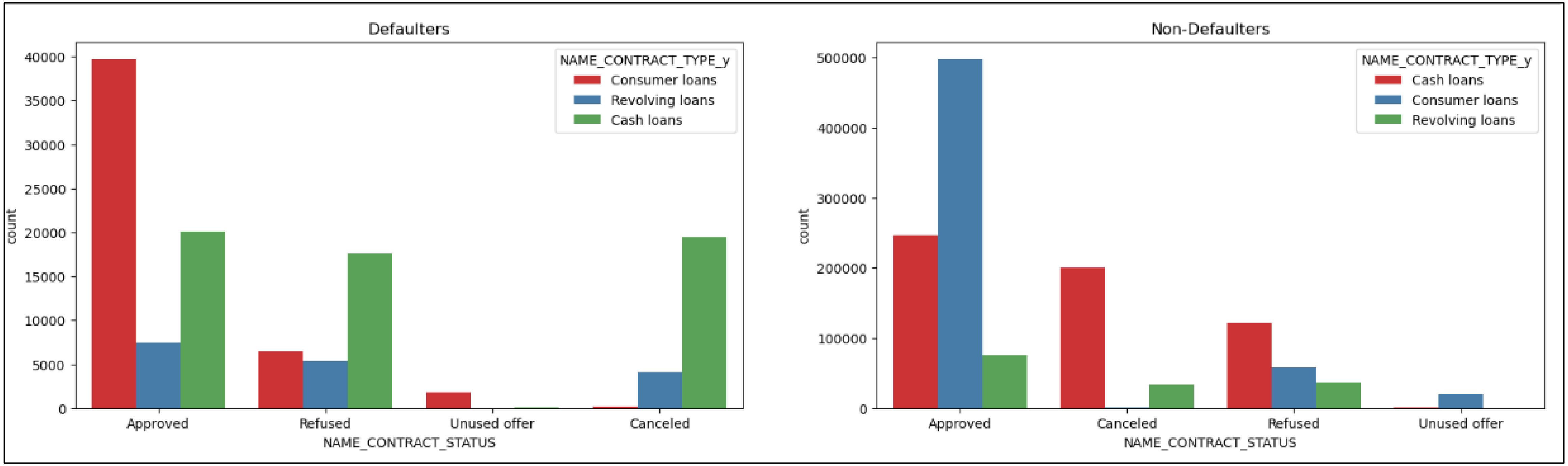
Preferred choice to approve loans – Middle Age and Senior_citizen
Preferred choice not to approve loans – Young Adult and Adult

- Based on Contract Status and Client Type



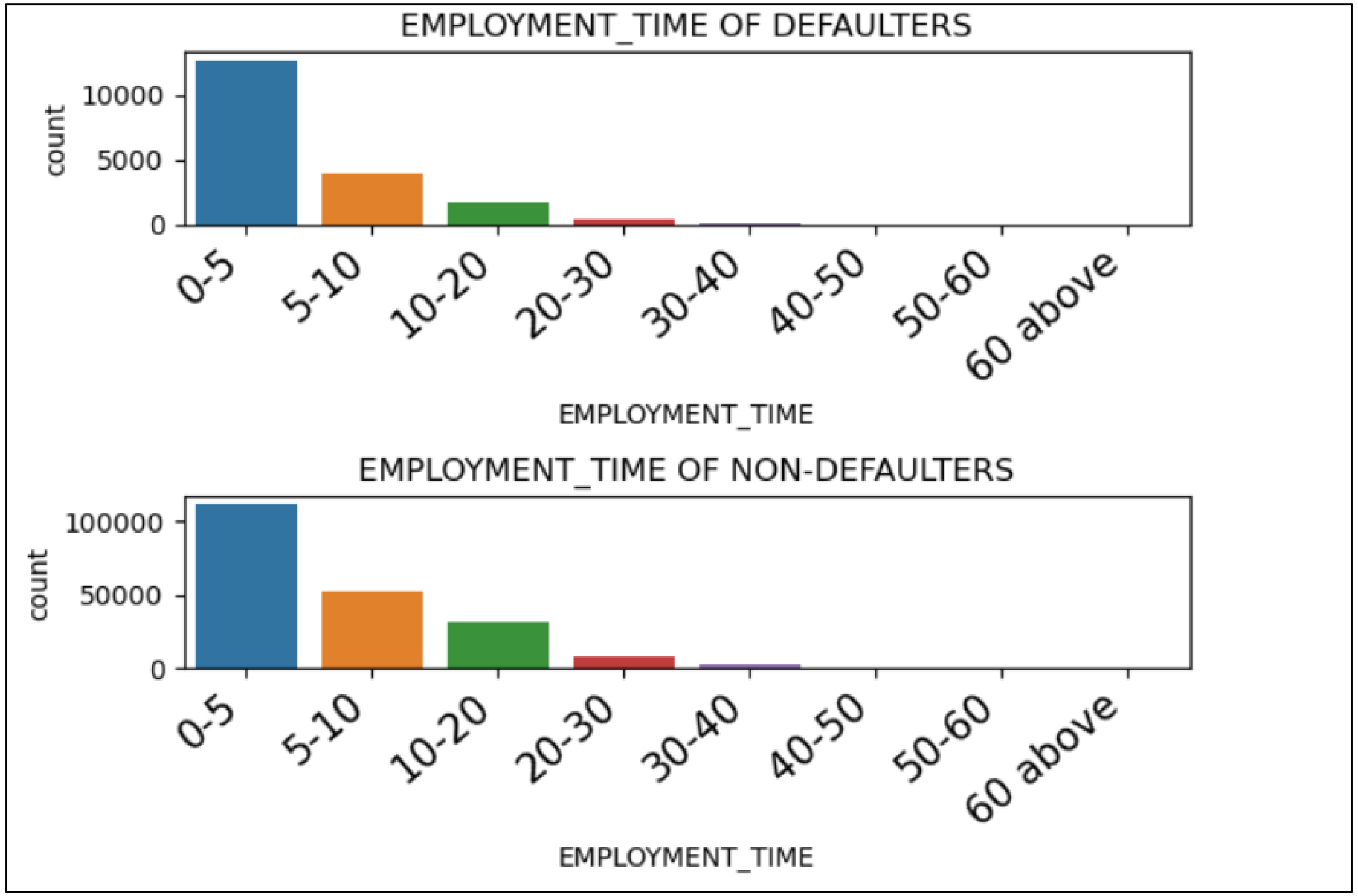
Preferred choice to approve loans – Approved previous applications
Preferred choice not to approve loans – Refused previous applications

- Based on Contract Status and Contract Type



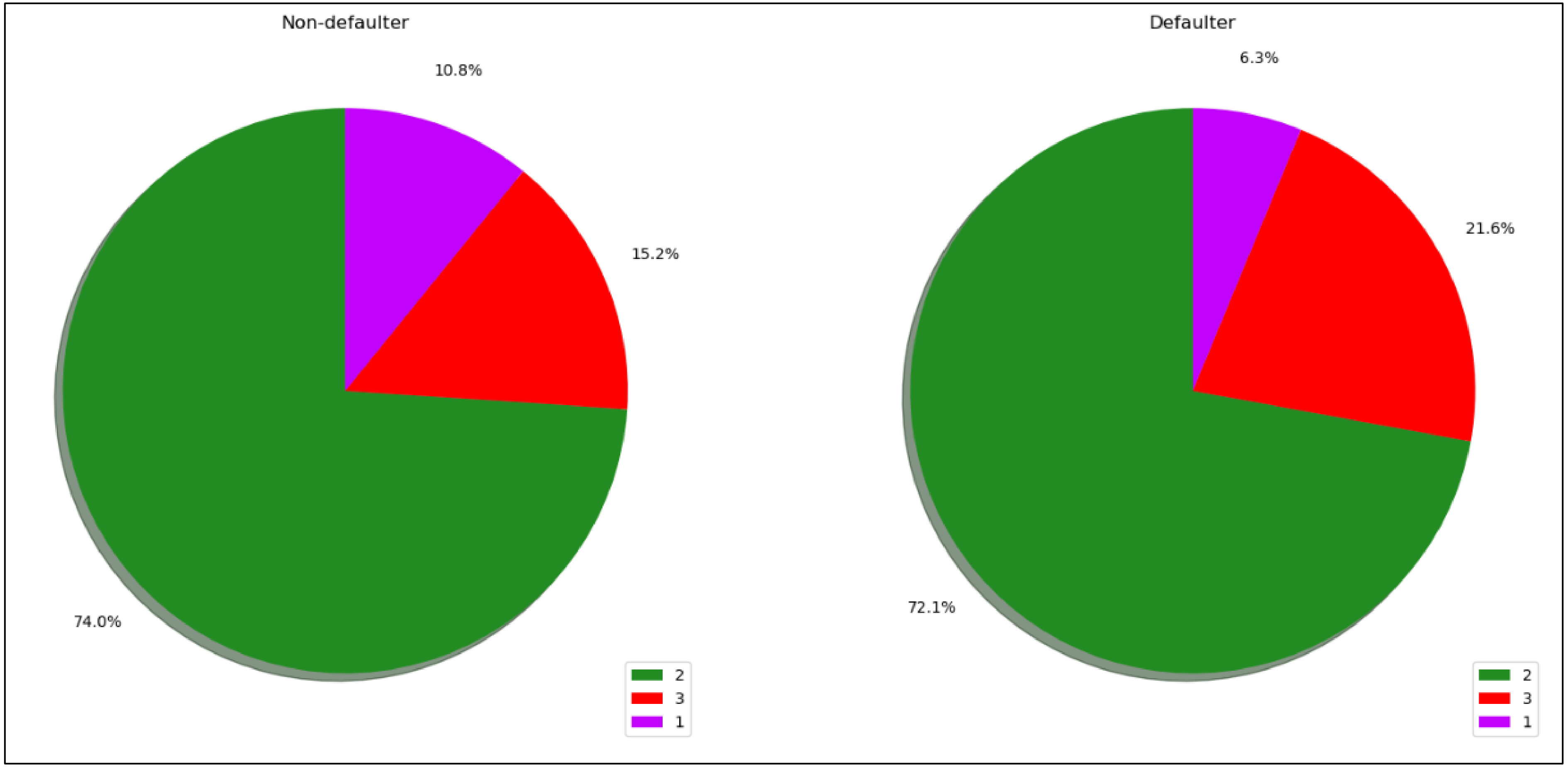
Preferred choice to approve loans – Consumer loans of previous applications
Preferred choice not to approve loans – Revolving loans of previous applications

- Based on Employment time



Preferred choice to approve loans – 40+ years of experience
Preferred choice not to approve loans – 0-5 years of experience

- Based on Region rating



Preferred choice to approve loans – Region rating 1
Preferred choice not to approve loans – Region rating 3