Loan Case Study

AIM:

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.

1. Importing the libraries and files

In [26]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
plt.style.use('dark_background')

pd.set_option('display.max_columns', 300) # to display all the columns
pd.set_option('display.max_rows', 300) # to display all the rows
pd.set_option('display.width', 1000)

warnings.filterwarnings('ignore') #To ignore the warnings
```

In [4]:

```
NewApplication = pd.read_csv('application_data.csv')
PreviousApplication = pd.read_csv('previous_application.csv')
```

2. NewApplication Data Routine Check

In [6]:

```
NewApplication.head()
```

Out[6]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN
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

In [7]:

```
NewApplication.shape
```

Out[7]:

(307511, 122)

```
(307511, 122)
```

In [8]:

```
NewApplication.info(verbose=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 122 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  OWN_CAR_AGE                   104582 non-null  float64
 22  FLAG_MOBIL                    307511 non-null  int64  
 23  FLAG_EMP_PHONE                307511 non-null  int64  
 24  FLAG_WORK_PHONE               307511 non-null  int64  
 25  FLAG_CONT_MOBILE              307511 non-null  int64  
 26  FLAG_PHONE                    307511 non-null  int64  
 27  FLAG_EMAIL                    307511 non-null  int64  
 28  OCCUPATION_TYPE               211120 non-null  object 
 29  CNT_FAM_MEMBERS               307509 non-null  float64
 30  REGION_RATING_CLIENT          307511 non-null  int64  
 31  REGION_RATING_CLIENT_W_CITY   307511 non-null  int64  
 32  WEEKDAY_APPR_PROCESS_START    307511 non-null  object 
 33  HOUR_APPR_PROCESS_START       307511 non-null  int64  
 34  REG_REGION_NOT_LIVE_REGION    307511 non-null  int64  
 35  REG_REGION_NOT_WORK_REGION    307511 non-null  int64  
 36  LIVE_REGION_NOT_WORK_REGION   307511 non-null  int64  
 37  REG_CITY_NOT_LIVE_CITY        307511 non-null  int64  
 38  REG_CITY_NOT_WORK_CITY        307511 non-null  int64  
 39  LIVE_CITY_NOT_WORK_CITY       307511 non-null  int64  
 40  ORGANIZATION_TYPE             307511 non-null  object 
 41  EXT_SOURCE_1                  134133 non-null  float64
 42  EXT_SOURCE_2                  306851 non-null  float64
 43  EXT_SOURCE_3                  246546 non-null  float64
 44  APARTMENTS_AVG                151450 non-null  float64
 45  BASEMENTAREA_AVG              127568 non-null  float64
 46  YEARS_BEGINEXPLUATATION_AVG   157504 non-null  float64
 47  YEARS_BUILD_AVG               103023 non-null  float64
 48  COMMONAREA_AVG                92646 non-null   float64
 49  ELEVATORS_AVG                 143620 non-null  float64
 50  ENTRANCES_AVG                 152683 non-null  float64
 51  FLOORSMAX_AVG                 154491 non-null  float64
 52  FLOORSMIN_AVG                 98869 non-null   float64
 53  LANDAREA_AVG                  124921 non-null  float64
 54  LIVINGAPARTMENTS_AVG          97312 non-null   float64
 55  LIVINGAREA_AVG                153161 non-null  float64
 56  NONLIVINGAPARTMENTS_AVG       93997 non-null   float64
 57  NONLIVINGAREA_AVG             137829 non-null  float64
 58  APARTMENTS_MODE               151450 non-null  float64
 59  BASEMENTAREA_MODE             127568 non-null  float64
 60  YEARS_BEGINEXPLUATATION_MODE  157504 non-null  float64
```

```
 61  YEARS_BUILD_MODE              103023 non-null  float64
 62  COMMONAREA_MODE               92646 non-null   float64
 63  ELEVATORS_MODE                143620 non-null  float64
 64  ENTRANCES_MODE                152683 non-null  float64
 65  FLOORSMAX_MODE                154491 non-null  float64
 66  FLOORSMIN_MODE                98869 non-null   float64
 67  LANDAREA_MODE                 124921 non-null  float64
 68  LIVINGAPARTMENTS_MODE         97312 non-null   float64
 69  LIVINGAREA_MODE               153161 non-null  float64
 70  NONLIVINGAPARTMENTS_MODE      93997 non-null   float64
 71  NONLIVINGAREA_MODE            137829 non-null  float64
 72  APARTMENTS_MEDI               151450 non-null  float64
 73  BASEMENTAREA_MEDI             127568 non-null  float64
 74  YEARS_BEGINEXPLUATATION_MEDI  157504 non-null  float64
 75  YEARS_BUILD_MEDI              103023 non-null  float64
 76  COMMONAREA_MEDI               92646 non-null   float64
 77  ELEVATORS_MEDI                143620 non-null  float64
 78  ENTRANCES_MEDI                152683 non-null  float64
 79  FLOORSMAX_MEDI                154491 non-null  float64
 80  FLOORSMIN_MEDI                98869 non-null   float64
 81  LANDAREA_MEDI                 124921 non-null  float64
 82  LIVINGAPARTMENTS_MEDI         97312 non-null   float64
 83  LIVINGAREA_MEDI               153161 non-null  float64
 84  NONLIVINGAPARTMENTS_MEDI      93997 non-null   float64
 85  NONLIVINGAREA_MEDI            137829 non-null  float64
 86  FONDKAPREMONT_MODE            97216 non-null   object
 87  HOUSETYPE_MODE                153214 non-null  object
 88  TOTALAREA_MODE                159080 non-null  float64
 89  WALLSMATERIAL_MODE            151170 non-null  object
 90  EMERGENCYSTATE_MODE           161756 non-null  object
 91  OBS_30_CNT_SOCIAL_CIRCLE      306490 non-null  float64
 92  DEF_30_CNT_SOCIAL_CIRCLE      306490 non-null  float64
 93  OBS_60_CNT_SOCIAL_CIRCLE      306490 non-null  float64
 94  DEF_60_CNT_SOCIAL_CIRCLE      306490 non-null  float64
 95  DAYS_LAST_PHONE_CHANGE        307510 non-null  float64
 96  FLAG_DOCUMENT_2               307511 non-null  int64
 97  FLAG_DOCUMENT_3               307511 non-null  int64
 98  FLAG_DOCUMENT_4               307511 non-null  int64
 99  FLAG_DOCUMENT_5               307511 non-null  int64
 100 FLAG_DOCUMENT_6               307511 non-null  int64
 101 FLAG_DOCUMENT_7               307511 non-null  int64
 102 FLAG_DOCUMENT_8               307511 non-null  int64
 103 FLAG_DOCUMENT_9               307511 non-null  int64
 104 FLAG_DOCUMENT_10              307511 non-null  int64
 105 FLAG_DOCUMENT_11              307511 non-null  int64
 106 FLAG_DOCUMENT_12              307511 non-null  int64
 107 FLAG_DOCUMENT_13              307511 non-null  int64
 108 FLAG_DOCUMENT_14              307511 non-null  int64
 109 FLAG_DOCUMENT_15              307511 non-null  int64
 110 FLAG_DOCUMENT_16              307511 non-null  int64
 111 FLAG_DOCUMENT_17              307511 non-null  int64
 112 FLAG_DOCUMENT_18              307511 non-null  int64
 113 FLAG_DOCUMENT_19              307511 non-null  int64
 114 FLAG_DOCUMENT_20              307511 non-null  int64
 115 FLAG_DOCUMENT_21              307511 non-null  int64
 116 AMT_REQ_CREDIT_BUREAU_HOUR    265992 non-null  float64
 117 AMT_REQ_CREDIT_BUREAU_DAY     265992 non-null  float64
 118 AMT_REQ_CREDIT_BUREAU_WEEK    265992 non-null  float64
 119 AMT_REQ_CREDIT_BUREAU_MON     265992 non-null  float64
 120 AMT_REQ_CREDIT_BUREAU_QRT     265992 non-null  float64
 121 AMT_REQ_CREDIT_BUREAU_YEAR    265992 non-null  float64
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

In [9]:

```
NewApplication.describe()
```

Out[9]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRI

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

3. PreviousApplication data check

In [10]:

```
PreviousApplication.head()
```

Out[10]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYM
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	

In [11]:

```
PreviousApplication.shape
```

Out[11]:

```
(1670214, 37)
```

In [12]:

```
PreviousApplication.info()
```

```
<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
```

```
 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 [10]:

```
PreviousApplication.describe()
```

Out[10]:

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_I
count	1.670214e+06	1.670214e+06	1.297979e+06	1.670214e+06	1.670213e+06	7.743700e+05	1.28469
mean	1.923089e+06	2.783572e+05	1.595512e+04	1.752339e+05	1.961140e+05	6.697402e+03	2.27847
std	5.325980e+05	1.028148e+05	1.478214e+04	2.927798e+05	3.185746e+05	2.092150e+04	3.15396
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00	0.000000e+00	-9.000000e-01	0.00000
25%	1.461857e+06	1.893290e+05	6.321780e+03	1.872000e+04	2.416050e+04	0.000000e+00	5.08410
50%	1.923110e+06	2.787145e+05	1.125000e+04	7.104600e+04	8.054100e+04	1.638000e+03	1.12320
75%	2.384280e+06	3.675140e+05	2.065842e+04	1.803600e+05	2.164185e+05	7.740000e+03	2.34000
max	2.845382e+06	4.562550e+05	4.180581e+05	6.905160e+06	6.905160e+06	3.060045e+06	6.90516

4. Data Analysis For NewApplication Data

4.1 Checking the NewApplication dataset

In [13]:

```
# Finding the percentage of missing values in all columns
round(NewApplication.isnull().mean()*100,2).sort_values(ascending = False)
```

Out[13]:

```
COMMONAREA_MEDI                 69.87
COMMONAREA_AVG                  69.87
COMMONAREA_MODE                 69.87
NONLIVINGAPARTMENTS_MODE        69.43
NONLIVINGAPARTMENTS_MEDI        69.43
NONLIVINGAPARTMENTS_AVG         69.43
FONDKAPREMONT_MODE              68.39
LIVINGAPARTMENTS_MEDI           68.35
LIVINGAPARTMENTS_MODE           68.35
LIVINGAPARTMENTS_AVG            68.35
FLOORSMIN_MEDI                  67.85
FLOORSMIN_MODE                  67.85
```

```
FLOORSMIN_AVG                    67.85
YEARS_BUILD_MEDI                 66.50
YEARS_BUILD_AVG                  66.50
YEARS_BUILD_MODE                 66.50
OWN_CAR_AGE                      65.99
LANDAREA_MODE                    59.38
LANDAREA_AVG                     59.38
LANDAREA_MEDI                    59.38
BASEMENTAREA_MEDI                58.52
BASEMENTAREA_AVG                 58.52
BASEMENTAREA_MODE                58.52
EXT_SOURCE_1                     56.38
NONLIVINGAREA_MEDI               55.18
NONLIVINGAREA_AVG                55.18
NONLIVINGAREA_MODE               55.18
ELEVATORS_MODE                   53.30
ELEVATORS_AVG                    53.30
ELEVATORS_MEDI                   53.30
WALLSMATERIAL_MODE               50.84
APARTMENTS_MODE                  50.75
APARTMENTS_AVG                   50.75
APARTMENTS_MEDI                  50.75
ENTRANCES_MEDI                   50.35
ENTRANCES_MODE                   50.35
ENTRANCES_AVG                    50.35
LIVINGAREA_MEDI                  50.19
LIVINGAREA_MODE                  50.19
LIVINGAREA_AVG                   50.19
HOUSETYPE_MODE                   50.18
FLOORSMAX_MODE                   49.76
FLOORSMAX_MEDI                   49.76
FLOORSMAX_AVG                    49.76
YEARS_BEGINEXPLUATATION_MEDI     48.78
YEARS_BEGINEXPLUATATION_AVG      48.78
YEARS_BEGINEXPLUATATION_MODE     48.78
TOTALAREA_MODE                   48.27
EMERGENCYSTATE_MODE              47.40
OCCUPATION_TYPE                  31.35
EXT_SOURCE_3                     19.83
AMT_REQ_CREDIT_BUREAU_QRT        13.50
AMT_REQ_CREDIT_BUREAU_YEAR       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_HOUR       13.50
NAME_TYPE_SUITE                   0.42
OBS_30_CNT_SOCIAL_CIRCLE          0.33
OBS_60_CNT_SOCIAL_CIRCLE          0.33
DEF_60_CNT_SOCIAL_CIRCLE          0.33
DEF_30_CNT_SOCIAL_CIRCLE          0.33
EXT_SOURCE_2                      0.21
AMT_GOODS_PRICE                   0.09
DAYS_ID_PUBLISH                   0.00
FLAG_EMP_PHONE                    0.00
FLAG_MOBIL                        0.00
DAYS_EMPLOYED                     0.00
FLAG_WORK_PHONE                   0.00
FLAG_CONT_MOBILE                  0.00
FLAG_PHONE                        0.00
FLAG_EMAIL                        0.00
DAYS_REGISTRATION                 0.00
NAME_HOUSING_TYPE                 0.00
DAYS_BIRTH                        0.00
REGION_POPULATION_RELATIVE        0.00
REGION_RATING_CLIENT              0.00
NAME_FAMILY_STATUS                0.00
NAME_EDUCATION_TYPE               0.00
NAME_INCOME_TYPE                  0.00
AMT_ANNUITY                       0.00
AMT_CREDIT                        0.00
AMT_INCOME_TOTAL                  0.00
CNT_CHILDREN                      0.00
```

```
FLAG_OWN_REALTY                 0.00
FLAG_OWN_CAR                    0.00
CODE_GENDER                     0.00
NAME_CONTRACT_TYPE              0.00
TARGET                          0.00
CNT_FAM_MEMBERS                 0.00
REG_REGION_NOT_LIVE_REGION      0.00
REGION_RATING_CLIENT_W_CITY     0.00
FLAG_DOCUMENT_14                0.00
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_15                0.00
WEEKDAY_APPR_PROCESS_START      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
ORGANIZATION_TYPE               0.00
LIVE_CITY_NOT_WORK_CITY         0.00
REG_CITY_NOT_WORK_CITY          0.00
REG_CITY_NOT_LIVE_CITY          0.00
LIVE_REGION_NOT_WORK_REGION     0.00
REG_REGION_NOT_WORK_REGION      0.00
HOUR_APPR_PROCESS_START         0.00
SK_ID_CURR                      0.00
dtype: float64
```

In [14]:

```
# Removing all the columns with more than 50% nulls values/Keeping all of them with <= 50
% null values
NewApplication = NewApplication.loc[:,NewApplication.isnull().mean()<=0.5]
NewApplication.shape
```

Out[14]:

```
(307511, 81)
```

In [15]:

```
#Selecting columns with less or equal to than 13% null vallues
list(NewApplication.columns[(NewApplication.isnull().mean()<=0.13) & (NewApplication.isn
ull().mean()>0)])

#We will check those columns for possible imputation
```

Out[15]:

```
['AMT_ANNUITY',
 'AMT_GOODS_PRICE',
 'NAME_TYPE_SUITE',
 'CNT_FAM_MEMBERS',
 'EXT_SOURCE_2',
 'OBS_30_CNT_SOCIAL_CIRCLE',
 'DEF_30_CNT_SOCIAL_CIRCLE',
 'OBS_60_CNT_SOCIAL_CIRCLE',
 'DEF_60_CNT_SOCIAL_CIRCLE',
 'DAYS_LAST_PHONE_CHANGE']
```

4.2 Checking for values to impute in columns

4.2.1. EXT_SOURCE_2 imputation

In [16]:

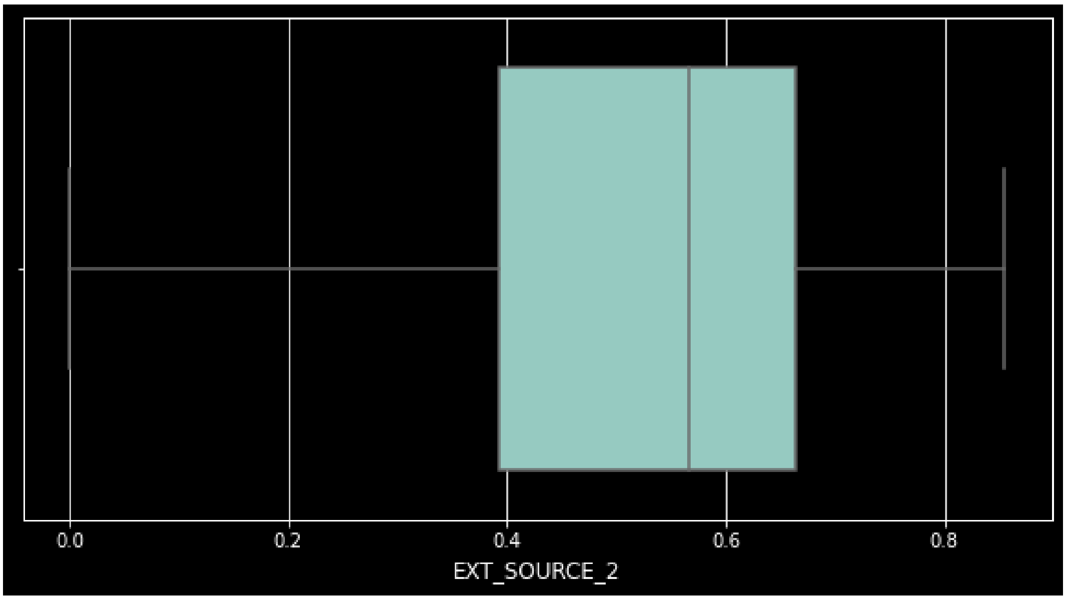
```
NewApplication['EXT_SOURCE_2'].value_counts()
```

Out[16]:

```
0.285898    721
0.262258    417
0.265256    343
0.159679    322
0.265312    306
           ...
0.169134      1
0.213753      1
0.057994      1
0.229146      1
0.336367      1
Name: EXT_SOURCE_2, Length: 119831, dtype: int64
```

In [25]:

```
# EXT_SOURCE_2 is a continuous variable. So checking for outliers
plt.style.use('dark_background')
plt.figure(figsize=[10,5])
sns.boxplot(NewApplication['EXT_SOURCE_2'])
plt.show()
```



In [27]:

```
# Since EXT_SOURCE_2 has no outlier, we can choose mean to impute the column
imputVAL = round(NewApplication['EXT_SOURCE_2'].mean(),2)
print(f'Since EXT_SOURCE_2 has no outlier, the column can be imputed using the mean of th
e coumn i.e. {imputVAL}')
```

```
Since EXT_SOURCE_2 has no outlier, the column can be imputed using the mean of the coumn
i.e. 0.51
```

4.2.2. OCCUPATION_TYPE imputation

In [28]:

```
NewApplication['AMT_ANNUITY'].value_counts()
```

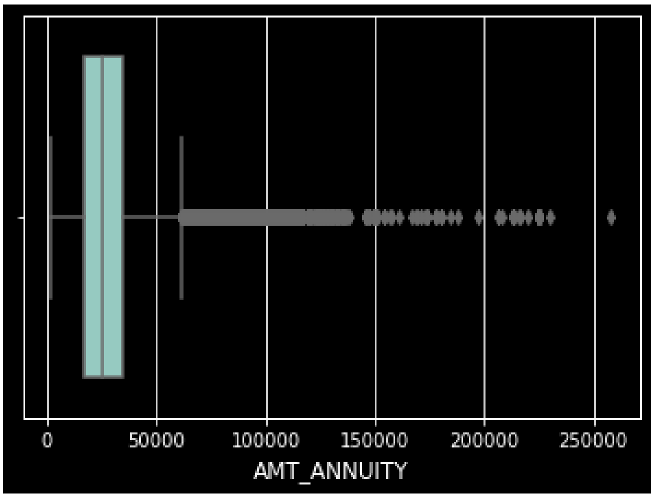
Out[28]:

```
9000.0      6385
```

```
9000.0       6385
13500.0      5514
6750.0       2279
10125.0      2035
37800.0      1602
             ...
15210.0         1
50265.0         1
73012.5         1
40558.5         1
4437.0          1
Name: AMT_ANNUITY, Length: 13672, dtype: int64
```

In [29]:

```
# Since AMT_ANNUITY is a continuous variable. So checking for outliers
sns.boxplot(NewApplication['AMT_ANNUITY'])
plt.show()
```



In [30]:

```
imputVAL = round(NewApplication['AMT_ANNUITY'].median(),2)
print(f'Since AMT_ANNUITY has outliers, the column can be imputed using the median of the coumn i.e. {imputVAL}')
```

```
Since AMT_ANNUITY has outliers, the column can be imputed using the median of the coumn i.e. 24903.0
```

4.2.3. NAME_TYPE_SUITE imputation

In [31]:

```
NewApplication['NAME_TYPE_SUITE'].value_counts()
```

Out[31]:

```
Unaccompanied      248526
Family              40149
Spouse, partner     11370
Children             3267
Other_B              1770
Other_A               866
Group of people       271
Name: NAME_TYPE_SUITE, dtype: int64
```

In [32]:

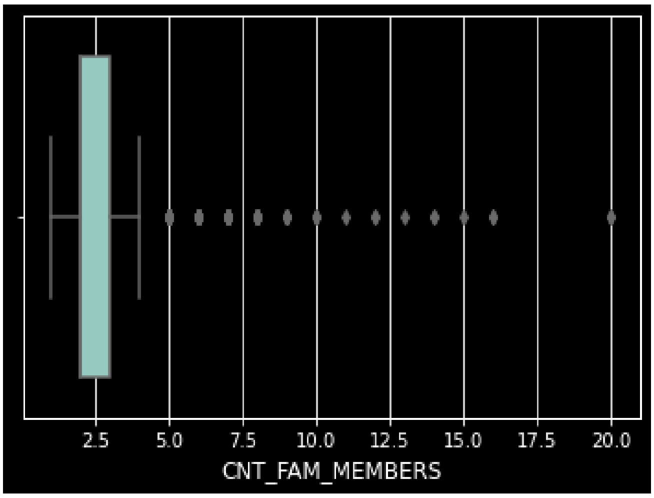
```
imputVAL = NewApplication['NAME_TYPE_SUITE'].mode()
print(f'Clearly the column NAME_TYPE_SUITE is a categorical column. So this column can be imputed using the mode of the column i.e {imputVAL[0]}')
```

```
Clearly the column NAME_TYPE_SUITE is a categorical column. So this column can be imputed using the mode of the column i.e Unaccompanied
```

4.2.4. CNT_FAM_MEMBERS imputation

In [33]:

```
# Since this is count of family members, this is a continuous variable and we can impute
the mean/median
sns.boxplot(NewApplication['CNT_FAM_MEMBERS'])
plt.show()
```



In [34]:

```
imputVAL = round(NewApplication['CNT_FAM_MEMBERS'].median(),2)
print(f'Since CNT_FAM_MEMBERS has outliers, the column can be imputed using the median of
the coumn i.e. {imputVAL}')
```

```
Since CNT_FAM_MEMBERS has outliers, the column can be imputed using the median of the cou
mn i.e. 2.0
```

4.2.5. AMT_GOODS_PRICE imputation

In [35]:

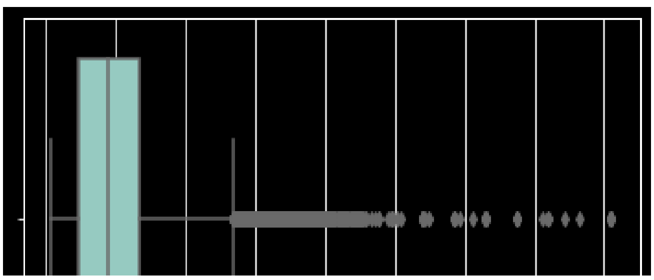
```
NewApplication['AMT_GOODS_PRICE'].value_counts()
```

Out[35]:

```
450000.0    26022
225000.0    25282
675000.0    24962
900000.0    15416
270000.0    11428
            ...
705892.5        1
442062.0        1
353641.5        1
353749.5        1
738945.0        1
Name: AMT_GOODS_PRICE, Length: 1002, dtype: int64
```

In [36]:

```
# AMT_GOODS_PRICE is a continuous variable. So checking for outliers
sns.boxplot(NewApplication['AMT_GOODS_PRICE'])
plt.show()
```



In [37]:

```
# Since this is a continuous variable with outliers we can impute column using median val
ue
imputVAL = round(NewApplication['AMT_GOODS_PRICE'].median(),2)
print(f'Since AMT_GOODS_PRICE has outliers, the column can be imputed using the median of
the coumn i.e. {imputVAL}')
```

```
Since AMT_GOODS_PRICE has outliers, the column can be imputed using the median of the cou
mn i.e. 450000.0
```

4.3 Check datatypes of columns and modify them appropriately

In [38]:

```
#Checking the float type columns
NewApplication.select_dtypes(include='float64').columns
```

Out[38]:

```
Index(['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'REGION_POPULA
TION_RELATIVE', 'DAYS_REGISTRATION', 'CNT_FAM_MEMBERS', 'EXT_SOURCE_2', 'EXT_SOURCE_3', '
YEARS_BEGINEXPLUATATION_AVG', 'FLOORSMAX_AVG', 'YEARS_BEGINEXPLUATATION_MODE', 'FLOORSMAX
_MODE', 'YEARS_BEGINEXPLUATATION_MEDI', 'FLOORSMAX_MEDI', 'TOTALAREA_MODE', 'OBS_30_CNT_S
OCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL
_CIRCLE', 'DAYS_LAST_PHONE_CHANGE', 'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_
DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_Q
RT', 'AMT_REQ_CREDIT_BUREAU_YEAR'], dtype='object')
```

In [39]:

```
#Converting these count columns to int64
ColumnToConvert = ['OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SO
CIAL_CIRCLE',
                   'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE', 'AMT_REQ_CREDIT
_BUREAU_HOUR',
                   'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_C
REDIT_BUREAU_MON',
                   'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR']
NewApplication.loc[:,ColumnToConvert]=NewApplication.loc[:,ColumnToConvert].apply(lambda
col: col.astype('int',errors='ignore'))
```

In [40]:

```
#Checking the object type columns
ColumnToConvert = list(NewApplication.select_dtypes(include='object').columns)
```

In [41]:

```
NewApplication.loc[:,ColumnToConvert]=NewApplication.loc[:,ColumnToConvert].apply(lambda
col: col.astype('str',errors='ignore'))
```

In [42]:

```
NewApplication.head()
```

Out[42]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN
0	100002	1	Cash loans	M	N	Y	0

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN
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

In [43]:

```
#Making Gender more readable
NewApplication['CODE_GENDER'].value_counts()
```

Out[43]:

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

In [44]:

```
# Dropping the Gender = XNA from the data set as there is not enough data regarding that
NewApplication = NewApplication[NewApplication['CODE_GENDER']!='XNA']
NewApplication['CODE_GENDER'].replace(['M','F'],['Male','Female'],inplace=True)
```

4.4 Binning variables for analysis

In [45]:

```
NewApplication['AMT_INCOME_TOTAL'].quantile([0,0.1,0.3,0.6,0.8,1])
```

Out[45]:

```
0.0        25650.0
0.1        81000.0
0.3       112500.0
0.6       162000.0
0.8       225000.0
1.0    117000000.0
Name: AMT_INCOME_TOTAL, dtype: float64
```

In [46]:

```
#Creating A new categorical variable based on income total
NewApplication['INCOME_GROUP']=pd.qcut(NewApplication['AMT_INCOME_TOTAL'],
                                       q=[0,0.1,0.3,0.6,0.8,1],
                                       labels=['VeryLow','Low','Medium','High','VeryHigh
'])
```

In [47]:

```
#Binning DAYS_BIRTH
abs(NewApplication['DAYS_BIRTH']).quantile([0,0.1,0.3,0.6,0.8,1])
```

Out[47]:

```
0.0     7489.0
0.1    10284.6
0.3    13140.0
0.6    17220.0
0.8    20474.0
1.0    25229.0
Name: DAYS_BIRTH, dtype: float64
```

In [48]:

```
#Creating a column AGE using DAYS_BIRTH
```

```
NewApplication['AGE']=abs(NewApplication['DAYS_BIRTH'])//365.25
```

In [49]:

```
NewApplication['AGE'].describe()
```

Out[49]:

```
count    307507.000000
mean         43.405223
std          11.945763
min          20.000000
25%          33.000000
50%          43.000000
75%          53.000000
max          69.000000
Name: AGE, dtype: float64
```

In [50]:

```
## Since the AGE varies from 20 to 69, we can create bins of 5 years starting from 20 to 70
NewApplication['AGE_GROUP'] = pd.cut(NewApplication['AGE'],bins=np.arange(20,71,5))
```

In [51]:

```
## Adding one more column that will be used for analysis later
NewApplication['CREDIT_INCOME_RATIO']=round((NewApplication['AMT_CREDIT']/NewApplication['AMT_INCOME_TOTAL']))
```

In [52]:

```
### Getting the percentage of social circle who defaulted
NewApplication['SOCIAL_CIRCLE_30_DAYS_DEF_PERC']=NewApplication['DEF_30_CNT_SOCIAL_CIRCLE']/NewApplication['OBS_30_CNT_SOCIAL_CIRCLE']
NewApplication['SOCIAL_CIRCLE_60_DAYS_DEF_PERC']=NewApplication['DEF_60_CNT_SOCIAL_CIRCLE']/NewApplication['OBS_60_CNT_SOCIAL_CIRCLE']
```

4.5 - Checking for imbalance in Target

In [53]:

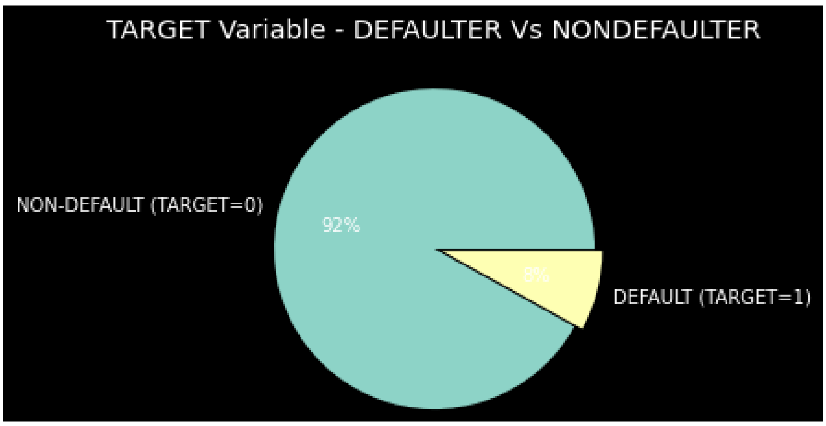
```
NewApplication['TARGET'].value_counts(normalize=True)*100
```

Out[53]:

```
0    91.927013
1     8.072987
Name: TARGET, dtype: float64
```

In [54]:

```
plt.pie(NewApplication['TARGET'].value_counts(normalize=True)*100,labels=['NON-DEFAULT (TARGET=0)','DEFAULT (TARGET=1)'],explode=(0,0.05),autopct='%1.f%%')
plt.title('TARGET Variable - DEFAULTER Vs NONDEFAULTER')
plt.show()
```



[redacted]

> **Its clear that there is an imbalance between people who defaulted and who didn't default. More than 92% of people didn't default as opposed to 8% who defaulted.**

In [55]:

```
# From the remaining columns about 30 are selected based on their description and relevan
ce with problem statement
#for further analysis
FinalColumns = ['SK_ID_CURR','TARGET','CODE_GENDER','FLAG_OWN_CAR','FLAG_OWN_REALTY','INC
OME_GROUP','AGE_GROUP','AMT_CREDIT','AMT_INCOME_TOTAL',
'CREDIT_INCOME_RATIO','NAME_INCOME_TYPE','NAME_EDUCATION_TYPE','NAME_FAMILY_STATUS','NAME
_HOUSING_TYPE','DAYS_EMPLOYED',
'DAYS_REGISTRATION','FLAG_EMAIL','OCCUPATION_TYPE',
'CNT_FAM_MEMBERS','REGION_RATING_CLIENT_W_CITY','ORGANIZATION_TYPE','SOCIAL_CIRCLE_30_DAY
S_DEF_PERC',
'SOCIAL_CIRCLE_60_DAYS_DEF_PERC','AMT_REQ_CREDIT_BUREAU_DAY',
'AMT_REQ_CREDIT_BUREAU_MON','AMT_REQ_CREDIT_BUREAU_QRT','NAME_CONTRACT_TYPE','AMT_ANNUITY
','REGION_RATING_CLIENT','AMT_GOODS_PRICE']
```

In [56]:

```
NewApplication_Final=NewApplication[FinalColumns]
```

In [57]:

```
NewApplication_Final.shape
```

Out[57]:

```
(307507, 30)
```

4.6 - Splitting the dataframe into two separate dfs

In [58]:

```
NEWAPP0=NewApplication_Final[NewApplication_Final.TARGET==0]    # Dataframe with all the
data related to non-defaulters
NEWAPP1=NewApplication_Final[NewApplication_Final.TARGET==1]    # Dataframe with all the
data related to defaulters
```

4.7 Univariate Analysis

Function to plot the univariate categorical variables

In [66]:

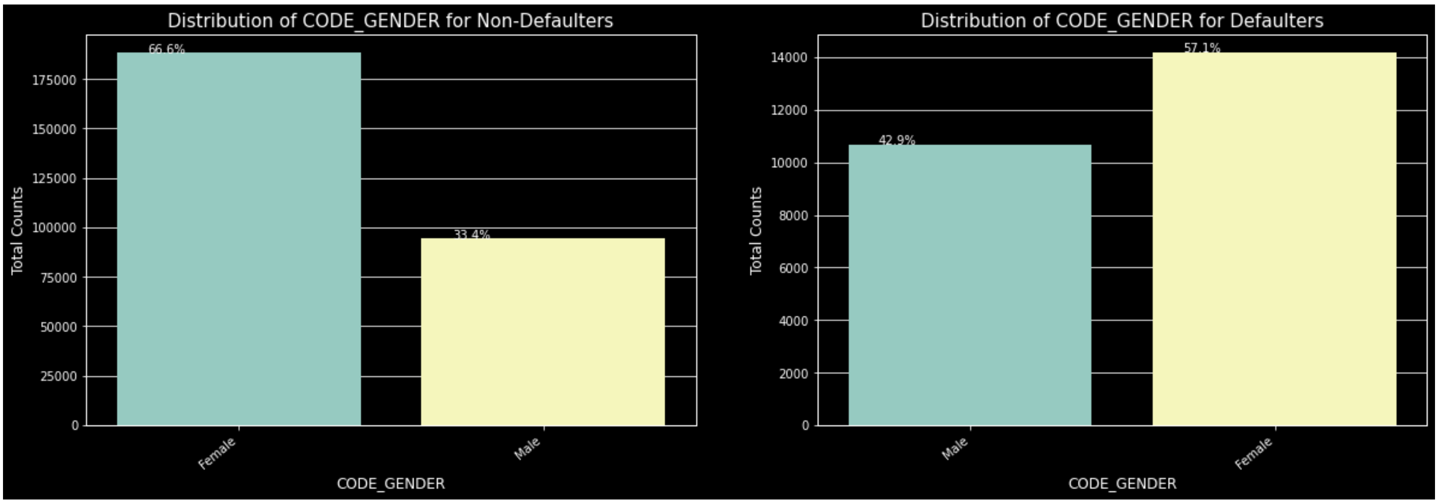
```
# function to count plot for categorical variables
def plotuninewapp(var):
    plt.style.use('dark_background')
    sns.despine
    fig,(ax1,ax2) = plt.subplots(1,2,figsize=(20,6))

    sns.countplot(x=var, data=NEWAPP0,ax=ax1)
    ax1.set_ylabel('Total Counts')
    ax1.set_title(f'Distribution of {var} for Non-Defaulters',fontsize=15)
    ax1.set_xticklabels(ax1.get_xticklabels(), rotation=40, ha="right")

    # Adding the normalized percentage for easier comparision between defaulter and non-d
efaulter
    for p in ax1.patches:
        ax1.annotate('{:.1f}%'.format((p.get_height()/len(NEWAPP0))*100), (p.get_x()+0.1
, p.get_height()+50))
```

```
    sns.countplot(x=var, data=NEWAPP1,ax=ax2)
    ax2.set_ylabel('Total Counts')
    ax2.set_title(f'Distribution of {var} for Defaulters',fontsize=15)
    ax2.set_xticklabels(ax2.get_xticklabels(), rotation=40, ha="right")

    # Adding the normalized percentage for easier comparision between defaulter and non-d
efaulter
    for p in ax2.patches:
        ax2.annotate('{:.1f}%'.format((p.get_height()/len(NEWAPP1))*100), (p.get_x()+0.1
, p.get_height()+50))

    plt.show()
```

4.7.1 Univariate Categorical Ordered Analysis

In [67]:

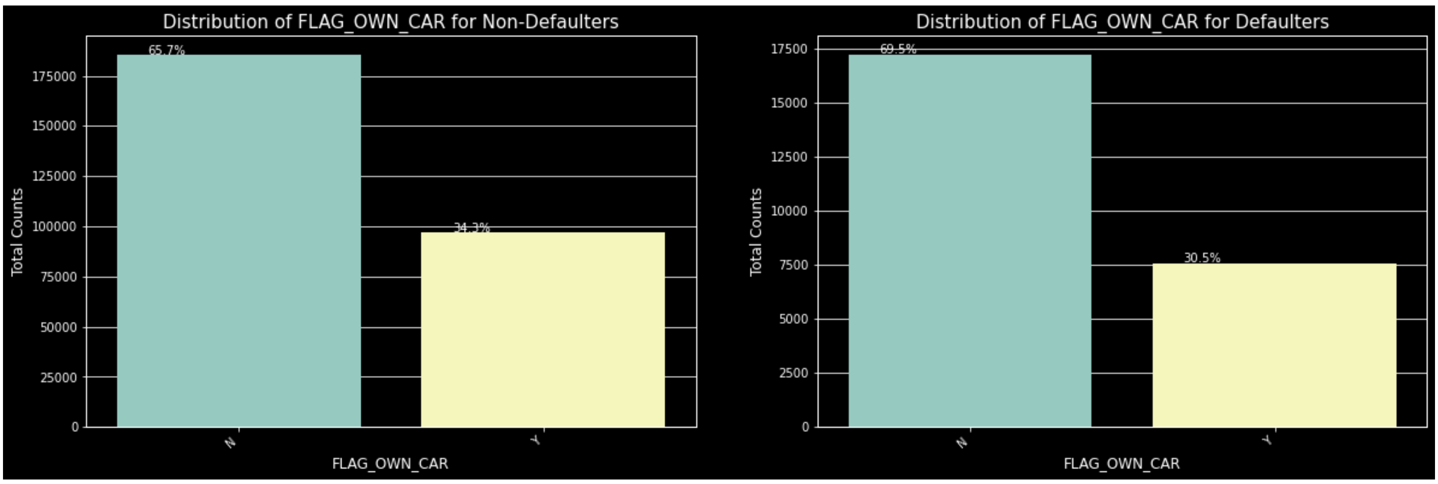
```
plotuninewapp('CODE_GENDER')
```



> **We can see that Female contribute 67% to the non-defaulters while 57% to the defaulters. We can conclude that**
> **We see more female applying for loans than males and hence the more number of female defaulters as well.**
> **But the rate of defaulting of FEMALE is much lower compared to their MALE counterparts.**

In [68]:

```
plotuninewapp('FLAG_OWN_CAR')
```

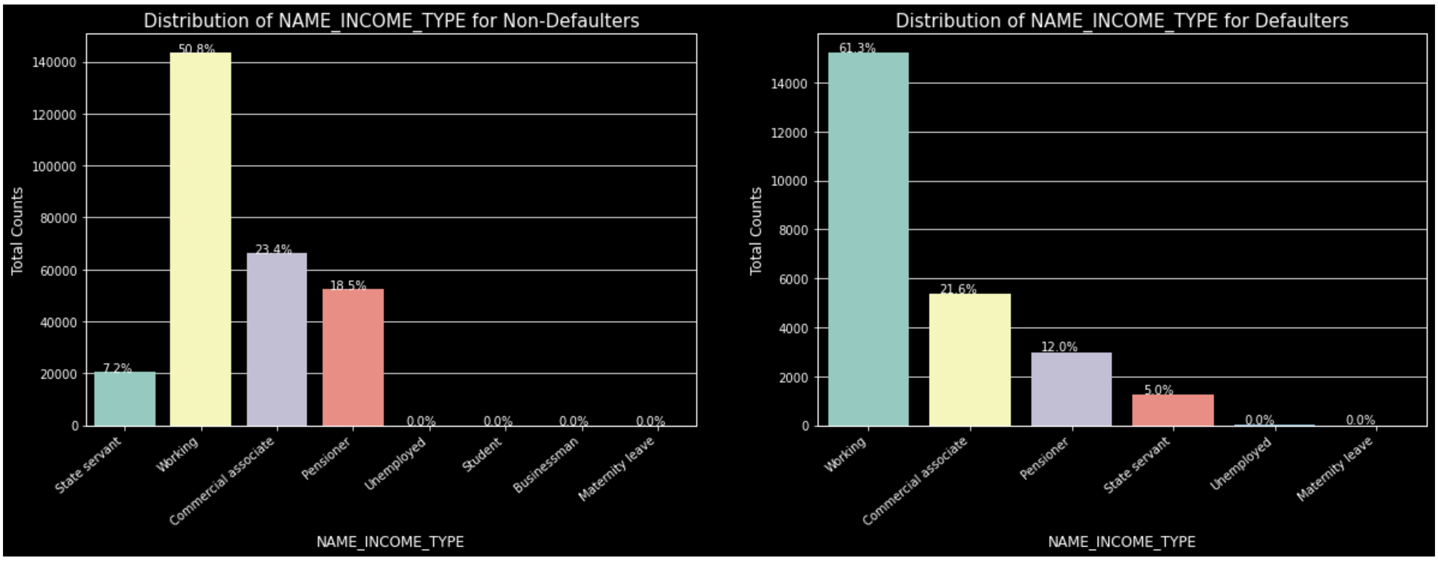


> **We can see that people with cars contribute 65.7% to the non-defaulters while 69.5% to the defaulters. We can conclude that**

defaulters. We can conclude that
While people who have car default more often, the reason could be there are simply more people without cars
Looking at the percentages in both the charts, we can conclude that the rate of default of people having car is low compared to people who don't.

In [69]:

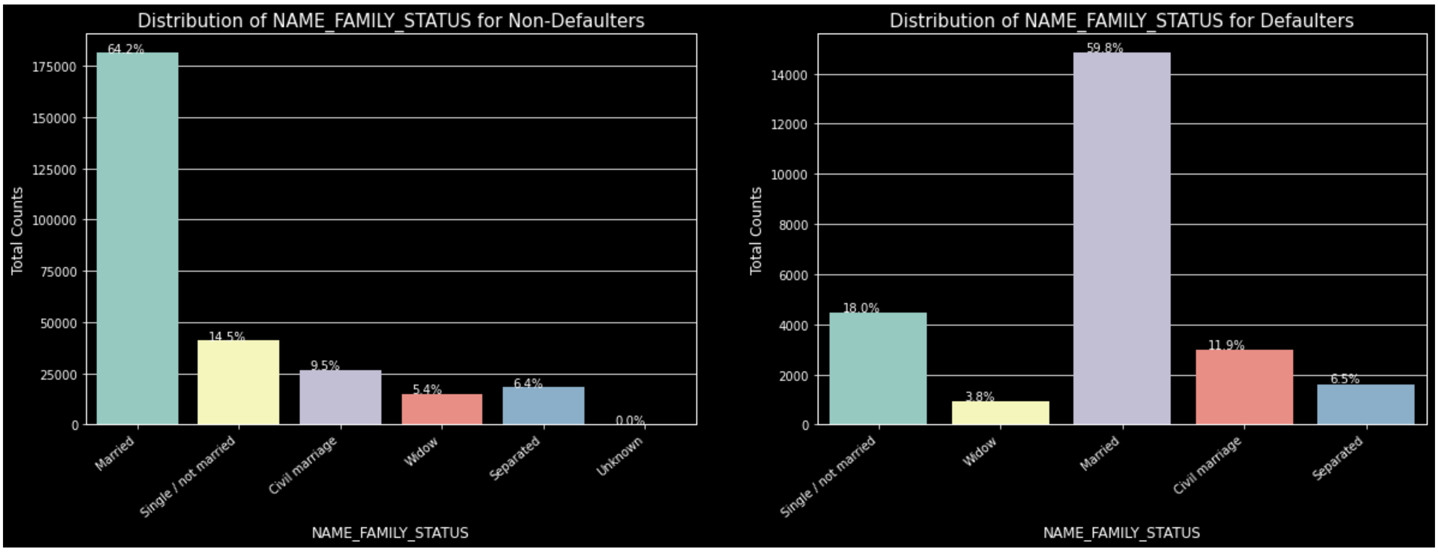
```
plotuninewapp('NAME_INCOME_TYPE')
```



We can notice that the students don't default. The reason could be they are not required to pay during the time they are students.
We can also see that the BusinessMen never default.
Most of the loans are distributed to working class people
We also see that working class people contribute 51% to non defaulters while they contribute to 61% of the defaulters. Clearly, the chances of defaulting are more in their case.

In [70]:

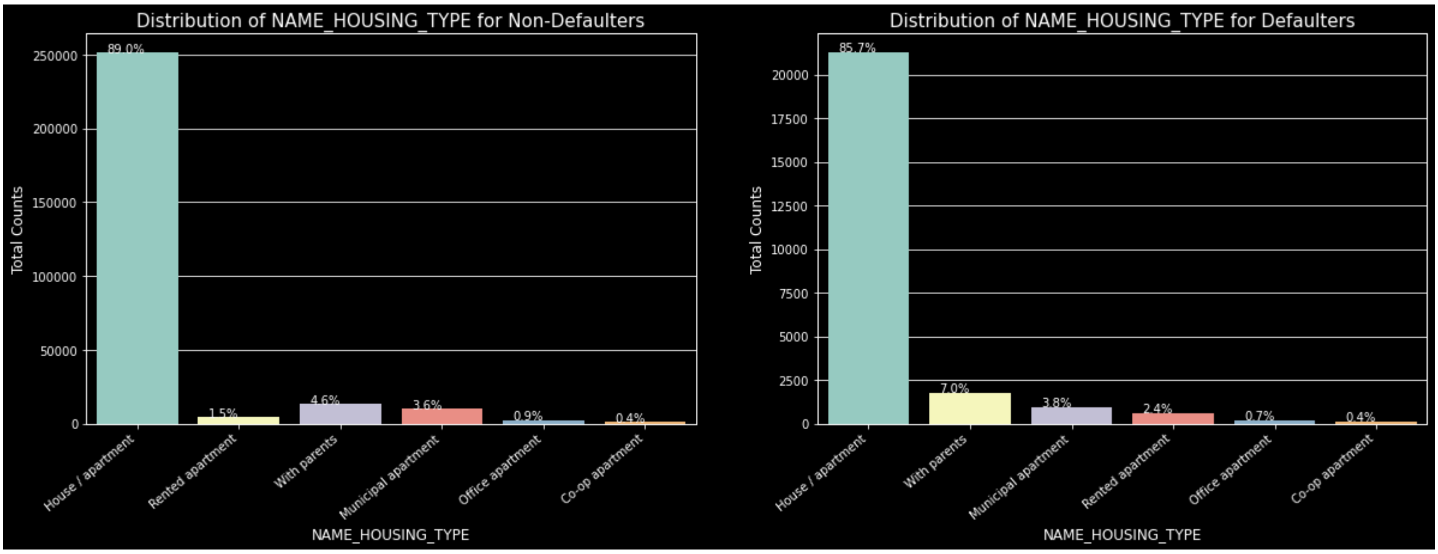
```
plotuninewapp('NAME_FAMILY_STATUS')
```



Married people tend to apply for more loans comparatively.
But from the graph we see that Single/non Married people contribute 14.5% to Non Defaulters and 18% to the defaulters. So there is more risk associated with them.

In [71]:

```
plotuninewapp('NAME_HOUSING_TYPE')
```

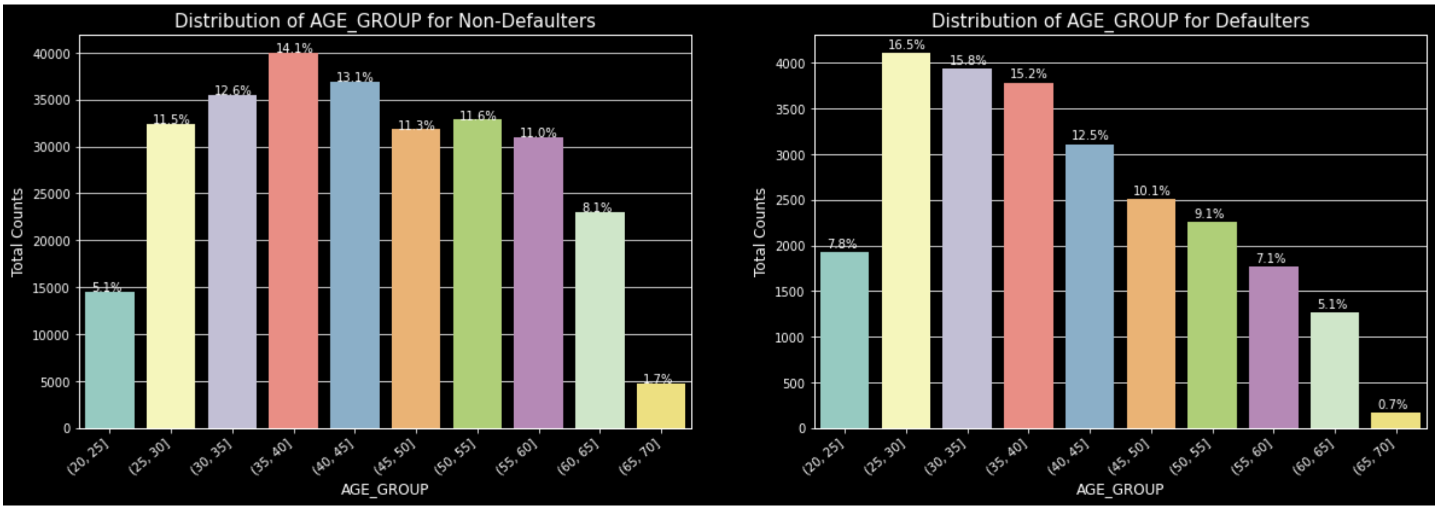


> **It is clear from the graph that people who have House/Appartment, tend to apply for more loans. People living with parents tend to default more often when compared with others.The reason could be their living expenses are more due to their parents living with them.**

4.7.2 Univariate Categorical Ordered Analysis

In [72]:

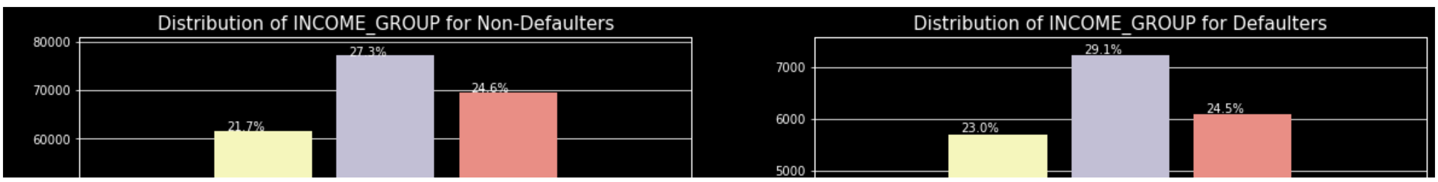
```
plotuninewapp('AGE_GROUP')
```

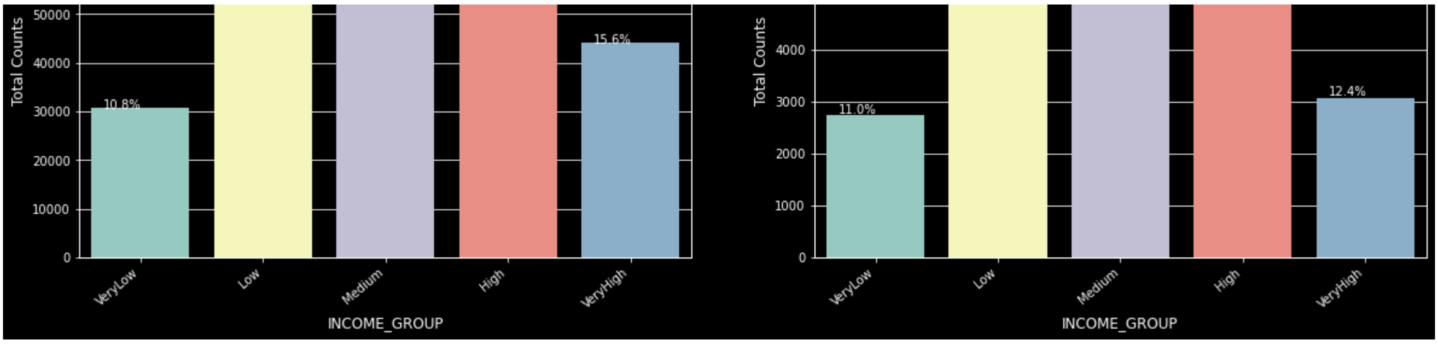


> **We see that (25,30] age group tend to default more often. So they are the riskiest people to loan to. With increasing age group, people tend to default less starting from the age 25. One of the reasons could be they get employed around that age and with increasing age, their salary also increases.**

In [73]:

```
plotuninewapp('INCOME_GROUP')
```

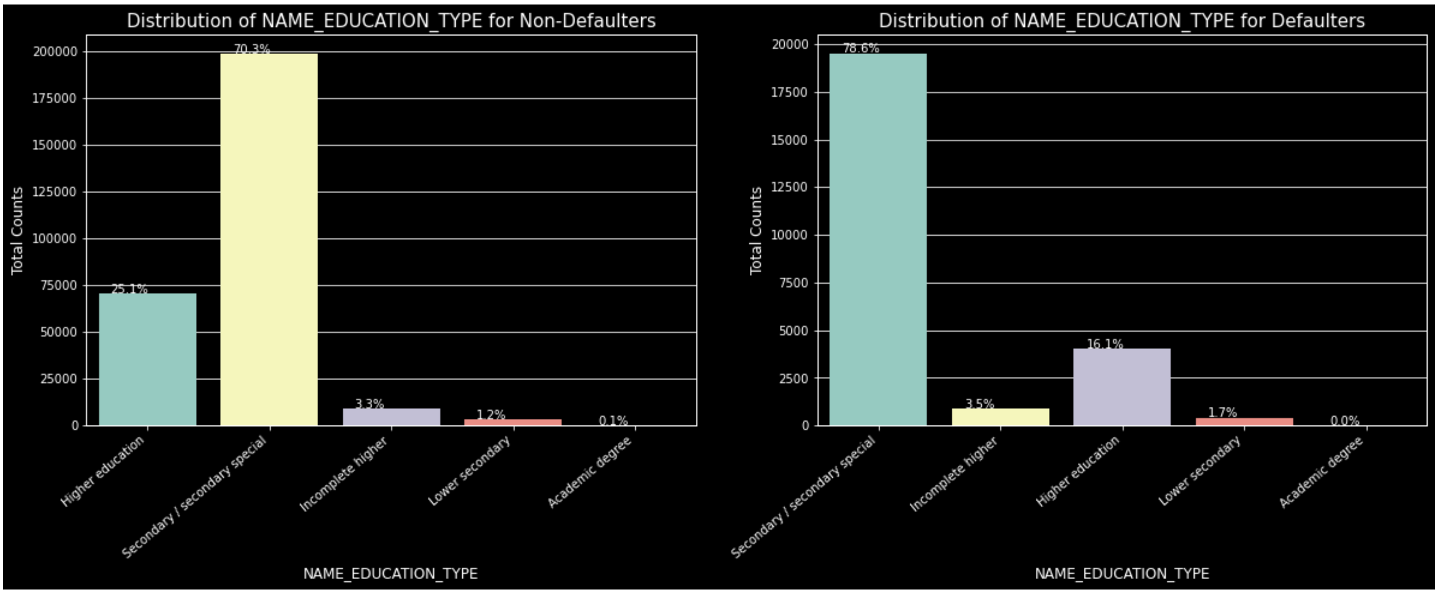




> **The Very High income group tend to default less often. They contribute 12.4% to the total number of defaulters, while they contribute 15.6% to the Non-Defaulters.**

In [74]:

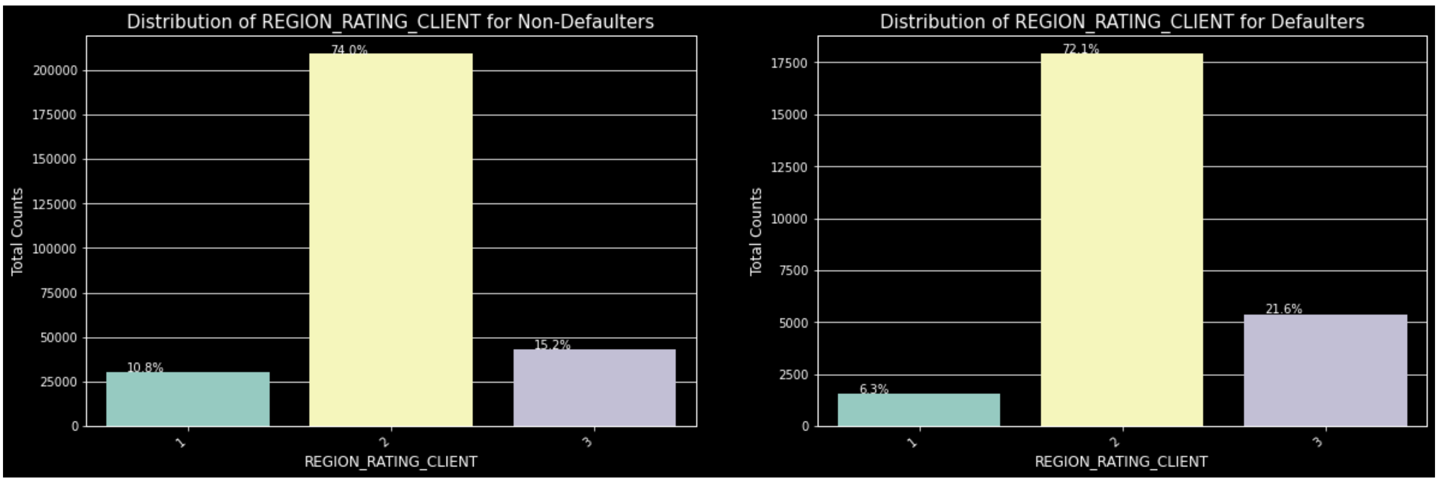
```
plotuninewapp('NAME_EDUCATION_TYPE')
```



> **Almost all of the Education categories are equally likely to default except for the higher educated ones who are less likely to default and secondary educated people are more likely to default**

In [75]:

```
plotuninewapp('REGION_RATING_CLIENT')
```



> **More people from second tier regions tend to apply for loans.**

> **We can infer that people living in better areas(Rating 3) tend contribute more to the defaulters by their weightage.**
> **People living in 1 rated areas**

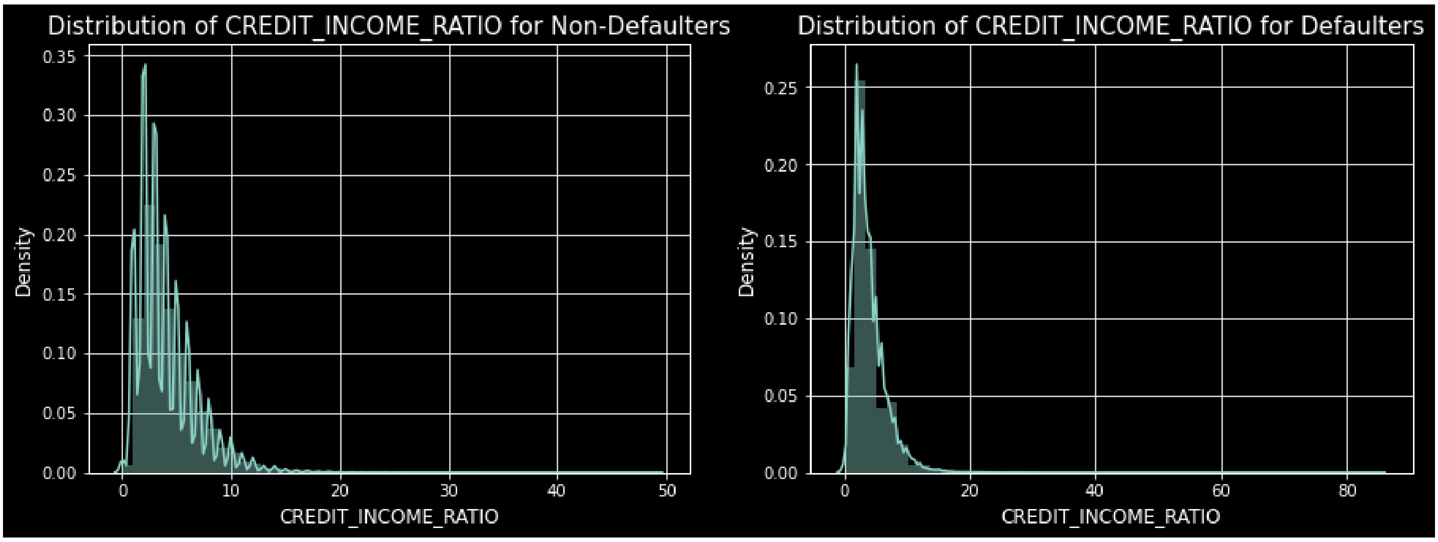
4.7.3 Univariate continuous variable analysis

In [78]:

```
# function to dist plot for continuous variables
def plotunidist(var):

    plt.style.use('dark_background')
    sns.despine
    fig,(ax1,ax2) = plt.subplots(1,2,figsize=(15,5))

    sns.distplot(a=NEWAPP0[var],ax=ax1)

    ax1.set_title(f'Distribution of {var} for Non-Defaulters',fontsize=15)

    sns.distplot(a=NEWAPP1[var],ax=ax2)
    ax2.set_title(f'Distribution of {var} for Defaulters',fontsize=15)

    plt.show()
```

In [79]:

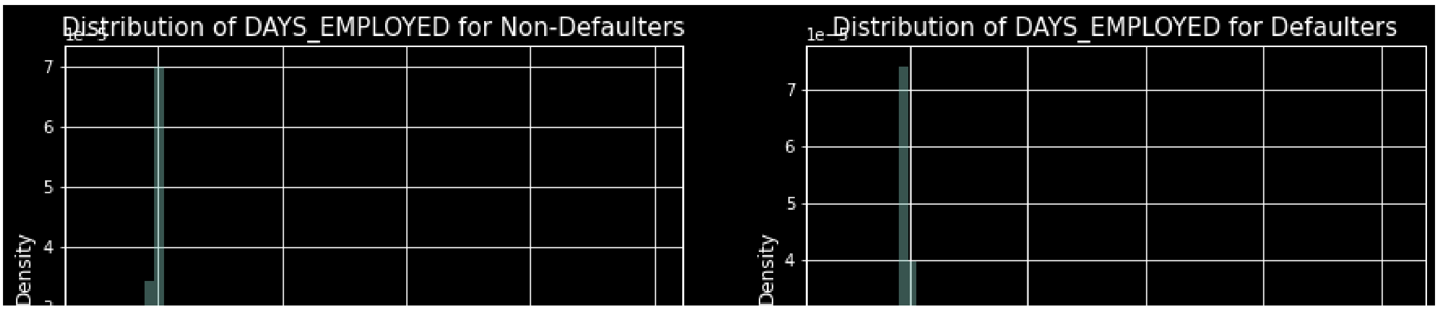
```
plotunidist('CREDIT_INCOME_RATIO')
```

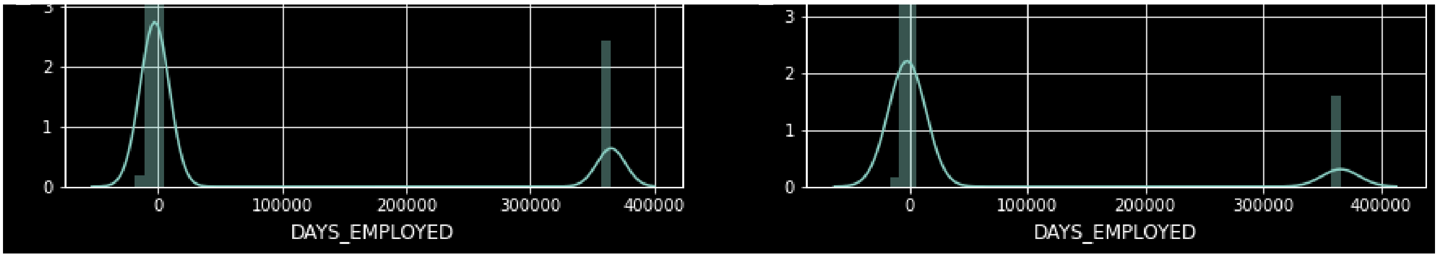


> **Credit income ratio the ratio of AMT_CREDIT/AMT_INCOME_TOTAL.**
> **Although there doesn't seem to be a clear distiguish between the group which defaulted vs the group which didn't when compared using the ratio, we can see that when the CREDIT_INCOME_RATIO is more than 50, people default.**

In [80]:

```
plotunidist('DAYS_EMPLOYED')
```





In [81]:

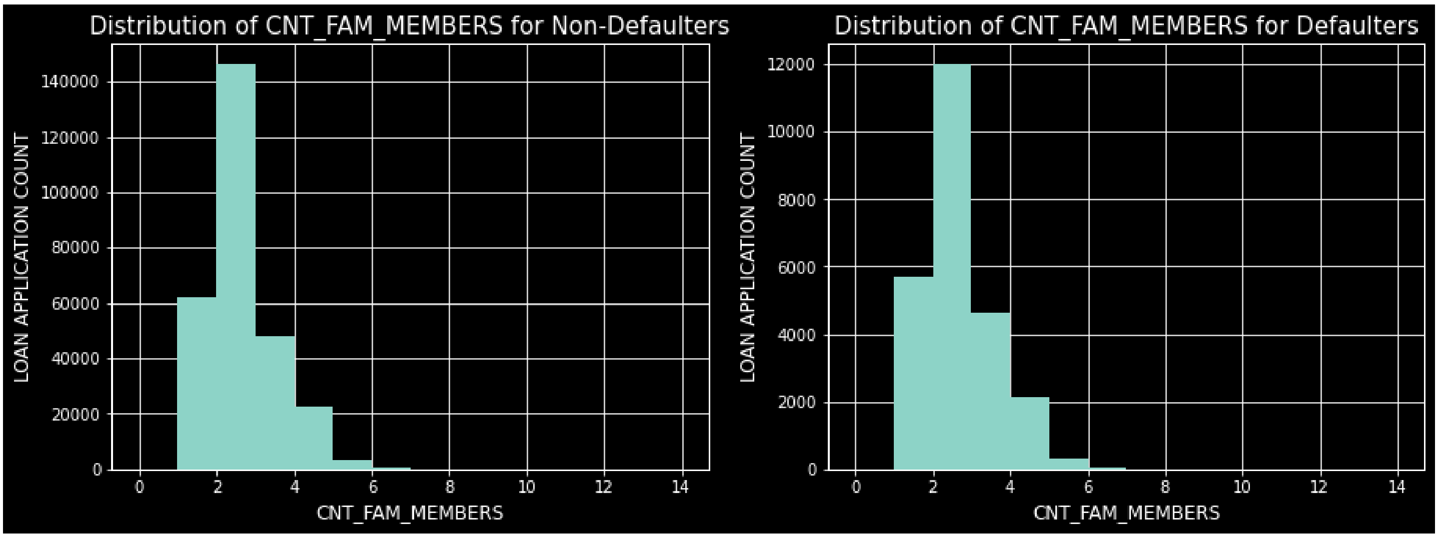
```
NEWAPP1['CNT_FAM_MEMBERS'].value_counts()
```

Out[81]:

```
2.0      12009
1.0       5675
3.0       4608
4.0       2136
5.0        327
6.0         55
7.0          6
8.0          6
11.0         1
10.0         1
13.0         1
Name: CNT_FAM_MEMBERS, dtype: int64
```

In [82]:

```
plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)
NEWAPP0['CNT_FAM_MEMBERS'].plot.hist(bins=range(15))
plt.title('Distribution of CNT_FAM_MEMBERS for Non-Defaulters',fontsize=15)
plt.xlabel('CNT_FAM_MEMBERS')
plt.ylabel('LOAN APPLICATION COUNT')

plt.subplot(1, 2, 2)
NEWAPP1['CNT_FAM_MEMBERS'].plot.hist(bins=range(15))
plt.title(f'Distribution of CNT_FAM_MEMBERS for Defaulters',fontsize=15)
plt.xlabel('CNT_FAM_MEMBERS')
plt.ylabel('LOAN APPLICATION COUNT')

plt.show()
```



> **We can see that a family of 3 applies loan more often than the other families**

4.8 Getting the top 10 correlation of the selected columns

In [83]:

```
#Getting the top 10 correlation in NEWAPP0
corr=NEWAPP0.corr()
corr_df = corr.where(np.triu(np.ones(corr.shape),k=1).astype(np.bool)).unstack().reset_i
ndex()
corr_df.columns=['Column1','Column2','Correlation']
corr_df.dropna(subset=['Correlation'],inplace=True)
corr_df['Abs_Correlation']=corr_df['Correlation'].abs()
corr_df = corr_df.sort_values(by=['Abs_Correlation'], ascending=False)
corr_df.head(10)
```

Out[83]:

	Column1	Column2	Correlation	Abs_Correlation
308	AMT_GOODS_PRICE	AMT_CREDIT	0.987253	0.987253
297	REGION_RATING_CLIENT	REGION_RATING_CLIENT_W_CITY	0.950148	0.950148
208	SOCIAL_CIRCLE_60_DAYS_DEF_PERC	SOCIAL_CIRCLE_30_DAYS_DEF_PERC	0.873003	0.873003
321	AMT_GOODS_PRICE	AMT_ANNUITY	0.776686	0.776686
272	AMT_ANNUITY	AMT_CREDIT	0.771308	0.771308
74	CREDIT_INCOME_RATIO	AMT_CREDIT	0.648589	0.648589
310	AMT_GOODS_PRICE	CREDIT_INCOME_RATIO	0.628749	0.628749
273	AMT_ANNUITY	AMT_INCOME_TOTAL	0.418954	0.418954
274	AMT_ANNUITY	CREDIT_INCOME_RATIO	0.391499	0.391499
309	AMT_GOODS_PRICE	AMT_INCOME_TOTAL	0.349461	0.349461

In [84]:

```
#Getting the top 10 correlation NEWAPP1
corr=NEWAPP1.corr()
corr_df = corr.where(np.triu(np.ones(corr.shape),k=1).astype(np.bool)).unstack().reset_i
ndex()
corr_df.columns=['Column1','Column2','Correlation']
corr_df.dropna(subset=['Correlation'],inplace=True)
corr_df['Abs_Correlation']=corr_df['Correlation'].abs()
corr_df = corr_df.sort_values(by=['Abs_Correlation'], ascending=False)
corr_df.head(10)
```

Out[84]:

	Column1	Column2	Correlation	Abs_Correlation
308	AMT_GOODS_PRICE	AMT_CREDIT	0.983103	0.983103
297	REGION_RATING_CLIENT	REGION_RATING_CLIENT_W_CITY	0.956637	0.956637
208	SOCIAL_CIRCLE_60_DAYS_DEF_PERC	SOCIAL_CIRCLE_30_DAYS_DEF_PERC	0.874562	0.874562
321	AMT_GOODS_PRICE	AMT_ANNUITY	0.752699	0.752699
272	AMT_ANNUITY	AMT_CREDIT	0.752195	0.752195
74	CREDIT_INCOME_RATIO	AMT_CREDIT	0.639744	0.639744
310	AMT_GOODS_PRICE	CREDIT_INCOME_RATIO	0.623163	0.623163
274	AMT_ANNUITY	CREDIT_INCOME_RATIO	0.381298	0.381298
113	DAYS_REGISTRATION	DAYS_EMPLOYED	-0.188929	0.188929
149	CNT_FAM_MEMBERS	DAYS_EMPLOYED	-0.186561	0.186561

4.9 Bivariate Analysis of numerical variables

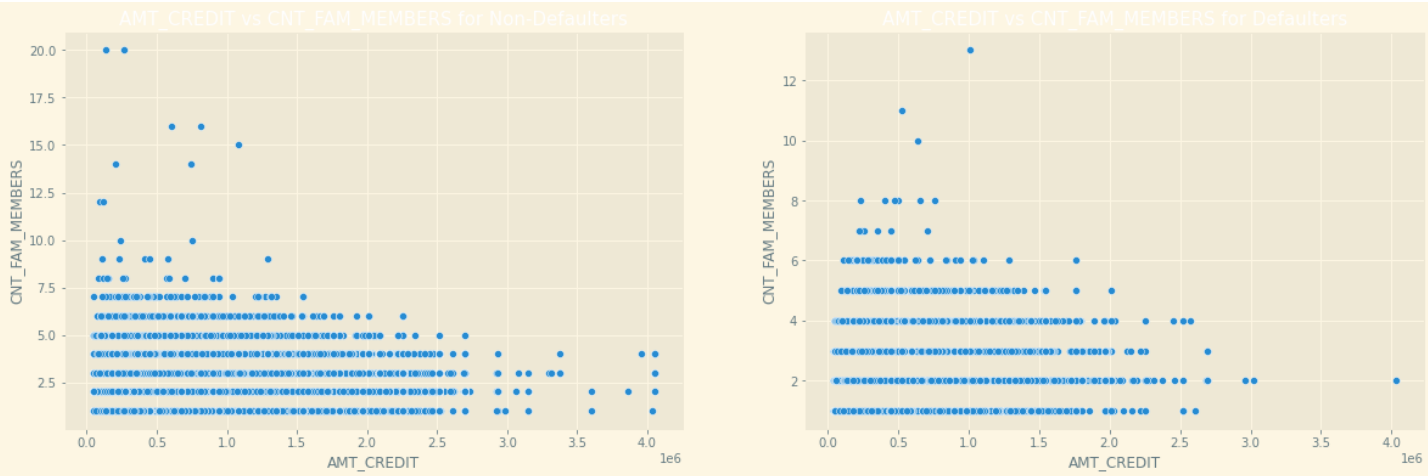
In [86]:

```
# function for scatter plot for continuous variables
```

```python
# function for scatter plot for continuous variables
def plotbivar(var1,var2):

    plt.style.use('Solarize_Light2')
    sns.despine
    fig,(ax1,ax2) = plt.subplots(1,2,figsize=(20,6))

    sns.scatterplot(x=var1, y=var2,data=NEWAPP0,ax=ax1)
    ax1.set_xlabel(var1)
    ax1.set_ylabel(var2)
    ax1.set_title(f'{var1} vs {var2} for Non-Defaulters',fontsize=15)

    sns.scatterplot(x=var1, y=var2,data=NEWAPP1,ax=ax2)
    ax2.set_xlabel(var1)
    ax2.set_ylabel(var2)
    ax2.set_title(f'{var1} vs {var2} for Defaulters',fontsize=15)

    plt.show()
```

In [87]:

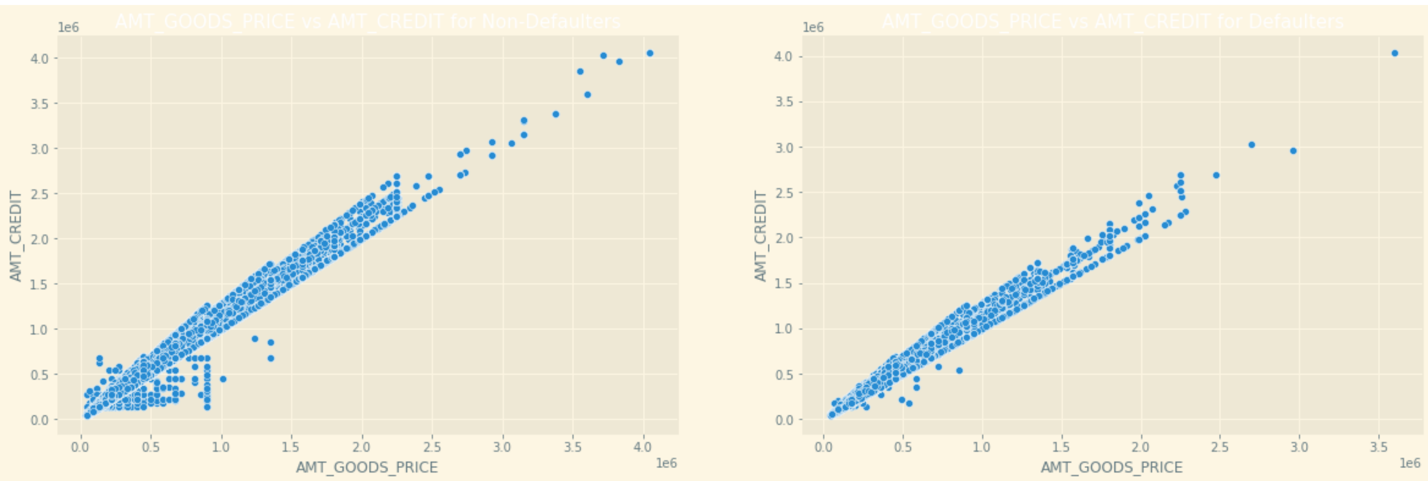
```python
plotbivar('AMT_CREDIT','CNT_FAM_MEMBERS')
```



> **We can see that the density in the lower left corner is similar in both the case, so the people are equally likely to default if the family is small and the AMT_CREDIT is low. We can observe that larger families and people with larger AMT_CREDIT default less often**

In [88]:

```python
plotbivar('AMT_GOODS_PRICE','AMT_CREDIT')
```



5. Data Analysis For Previous Application Data

5.1 Doing some more routine check

In [89]:

```
PreviousApplication.head(3)
```

Out[89]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYI
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	

In [90]:

```
# Removing all the columns with more than 50% of null values
PreviousApplication = PreviousApplication.loc[:,PreviousApplication.isnull().mean()<=0.5]
PreviousApplication.shape
```

Out[90]:

```
(1670214, 33)
```

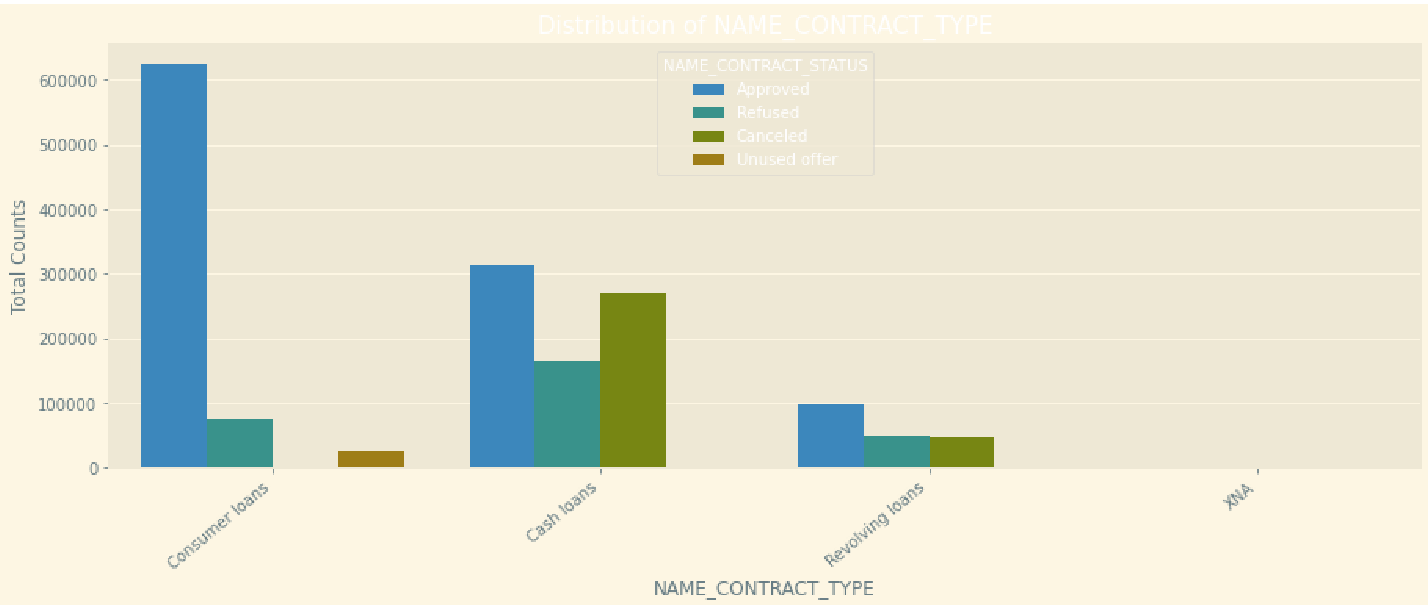
5.2 Univariate analysis

In [91]:

```
# function to count plot for categorical variables
def plot_uni(var):

    plt.style.use('Solarize_Light2')
    sns.despine
    fig,ax = plt.subplots(1,1,figsize=(15,5))

    sns.countplot(x=var, data=PreviousApplication,ax=ax,hue='NAME_CONTRACT_STATUS')
    ax.set_ylabel('Total Counts')
    ax.set_title(f'Distribution of {var}',fontsize=15)
    ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")

    plt.show()
```

In [92]:

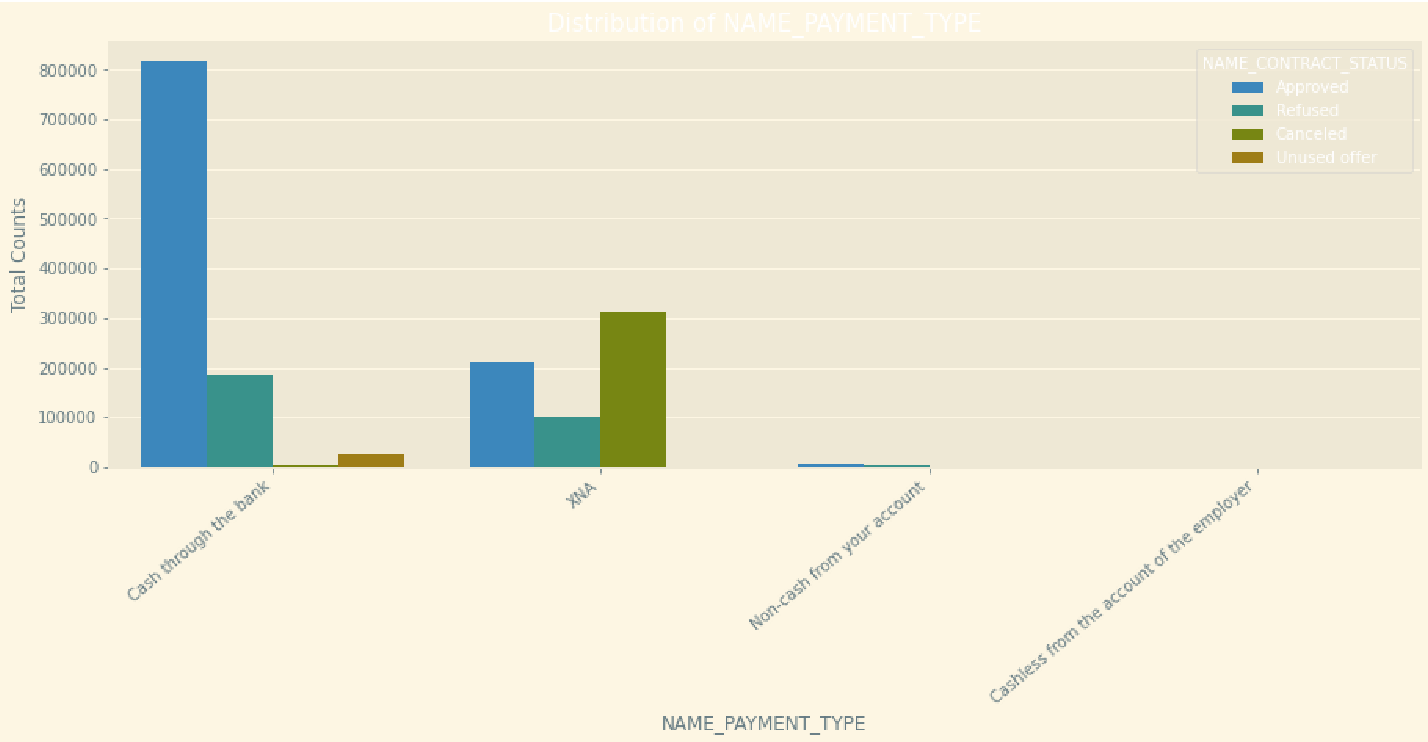
```
plot_uni('NAME_CONTRACT_TYPE')
```



From the above chart, we can infer that, most of the applications are for 'Cash loan' and 'Consumer loan'. Although the cash loans are refused more often than others.

In [93]:

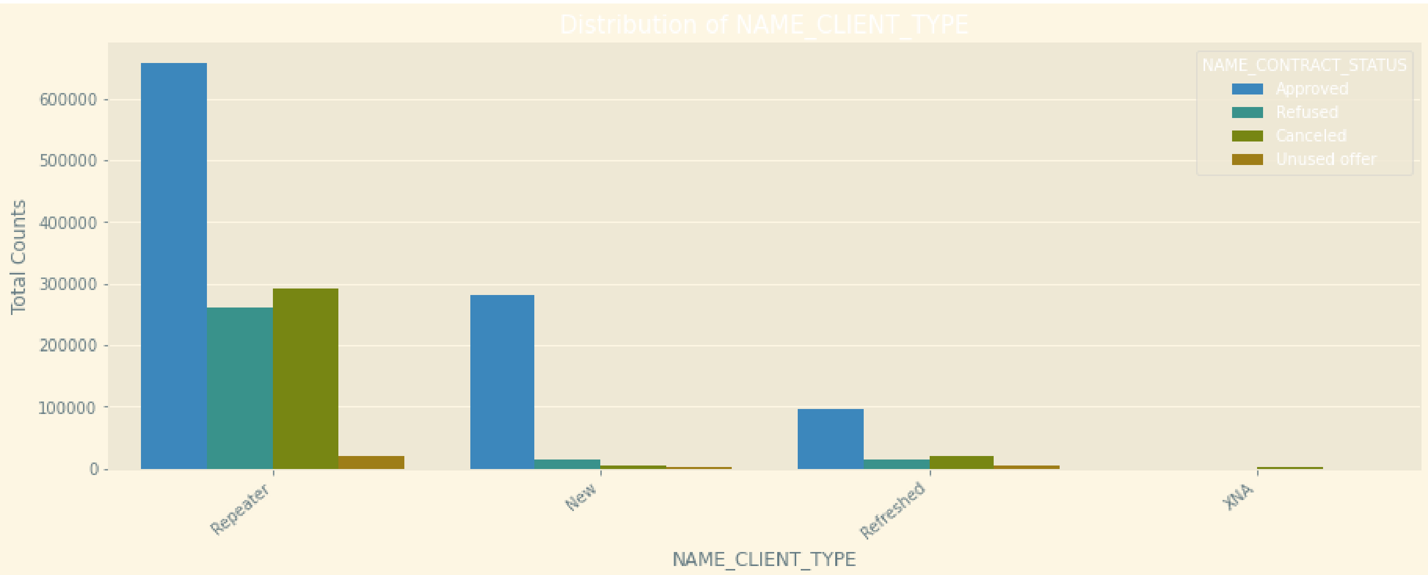
```
plot_uni('NAME_PAYMENT_TYPE')
```



From the above chart, we can infer that most of the clients chose to repay the loan using the 'Cash through the bank' option
We can also see that 'Non-Cash from your account' & 'Cashless from the account of the employee' options are not at all popular in terms of loan repayment amongst the customers.

In [94]:

```
plot_uni('NAME_CLIENT_TYPE')
```



Most of the loan applications are from repeat customers, out of the total applications 70% of customers are repeaters. They also get refused most often.

5.3 Checking the correlation in the PreviousApplication dataset

In [95]:

```
#Getting the top 10 correlation PreviousApplication
corr=PreviousApplication.corr()
corr_df = corr.where(np.triu(np.ones(corr.shape),k=1).astype(np.bool)).unstack().reset_i
ndex()
corr_df.columns=['Column1','Column2','Correlation']
corr_df.dropna(subset=['Correlation'],inplace=True)
corr_df['Abs_Correlation']=corr_df['Correlation'].abs()
corr_df = corr_df.sort_values(by=['Abs_Correlation'], ascending=False)
corr_df.head(10)
```

Out[95]:

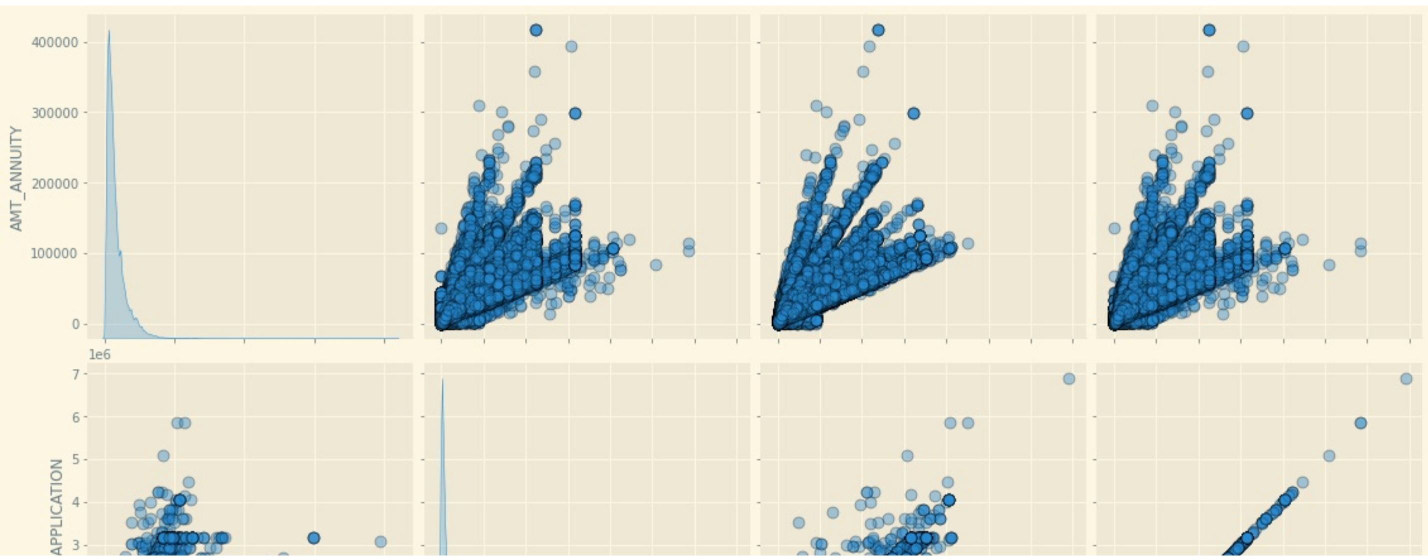
	Column1	Column2	Correlation	Abs_Correlation
88	AMT_GOODS_PRICE	AMT_APPLICATION	0.999884	0.999884
89	AMT_GOODS_PRICE	AMT_CREDIT	0.993087	0.993087
71	AMT_CREDIT	AMT_APPLICATION	0.975824	0.975824
269	DAYS_TERMINATION	DAYS_LAST_DUE	0.927990	0.927990
87	AMT_GOODS_PRICE	AMT_ANNUITY	0.820895	0.820895
70	AMT_CREDIT	AMT_ANNUITY	0.816429	0.816429
53	AMT_APPLICATION	AMT_ANNUITY	0.808872	0.808872
232	DAYS_LAST_DUE_1ST_VERSION	DAYS_FIRST_DRAWING	-0.803494	0.803494
173	CNT_PAYMENT	AMT_APPLICATION	0.680630	0.680630
174	CNT_PAYMENT	AMT_CREDIT	0.674278	0.674278

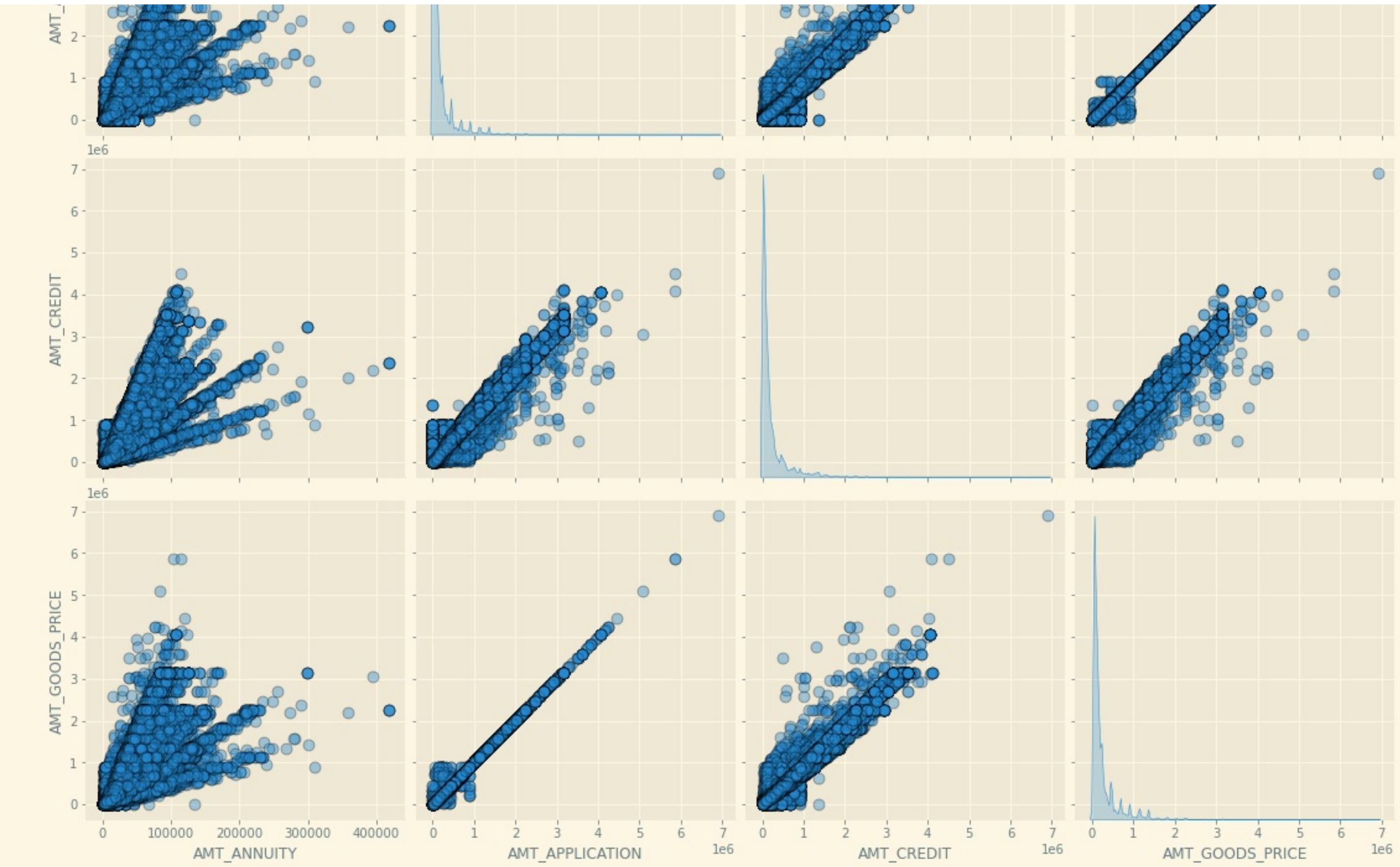
5.4 Using pairplot to perform bivariate analysis on numerical columns

In [96]:

```
#plotting the relation between correlated highly corelated numeric vriables
plt.figure(figsize=[20,8])
sns.pairplot(PreviousApplication[['AMT_ANNUITY','AMT_APPLICATION','AMT_CREDIT','AMT_GOODS
_PRICE','NAME_CONTRACT_STATUS']],
             diag_kind = 'kde',
             plot_kws = {'alpha': 0.4, 's': 80, 'edgecolor': 'k'},
             size = 4)
plt.show()
```

<Figure size 1440x576 with 0 Axes>





> 1. **Annuity of previous application has a very high and positive influence over: (Increase of annuity increases below factors)**
> **(1) How much credit did client asked on the previous application**
> **(2)Final credit amount on the previous application that was approved by the bank**
> **(3) Goods price of good that client asked for on the previous application.**
>
> 2. **For how much credit did client ask on the previous application is highly influenced by the Goods price of good that client has asked for on the previous application**
>
> 3. **Final credit amount disbursed to the customer previously, after approval is highly influence by the application amount and also the goods price of good that client asked for on the previous application.**

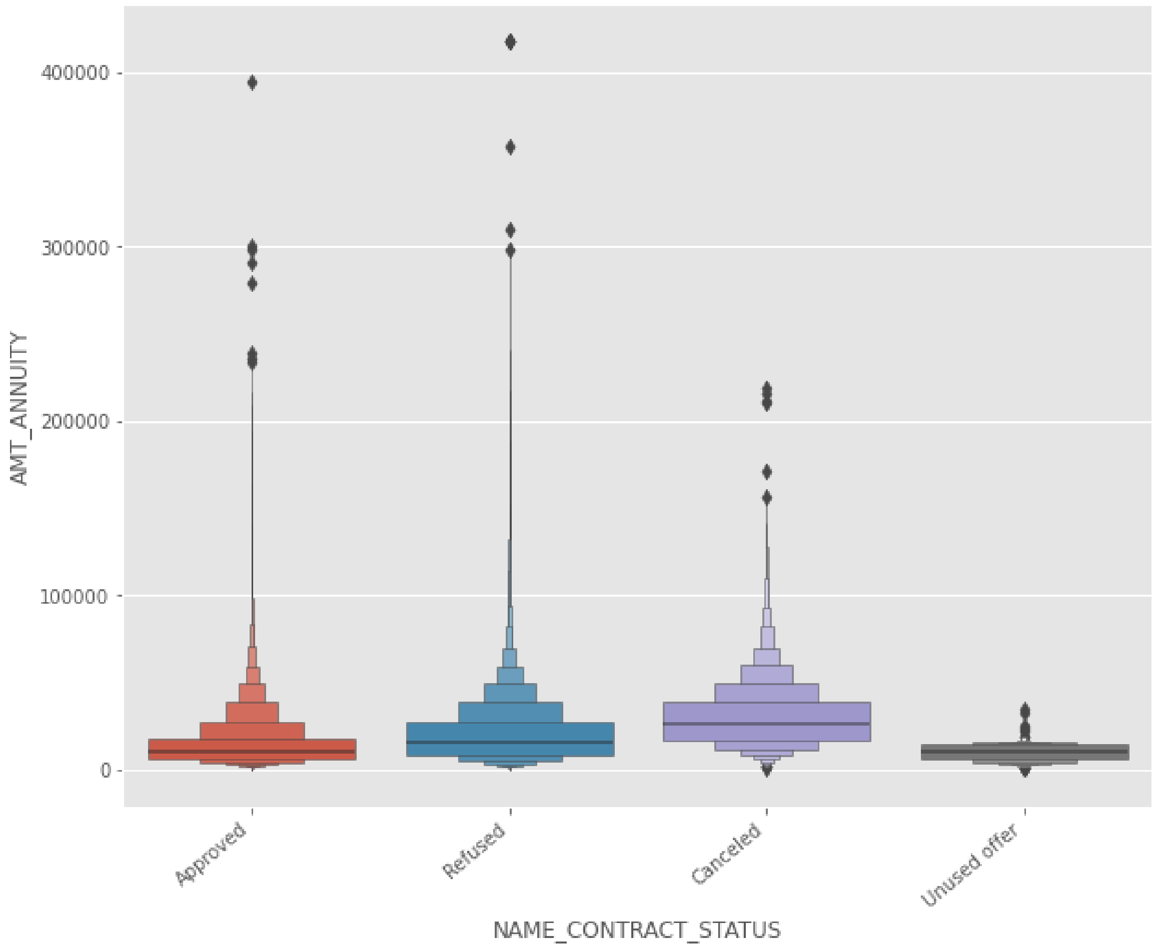
5.5 Using box plot to do some more bivariate analysis on categorical vs numeric columns

In [97]:

```
#by variant analysis function
def plot_by_cat_num(cat, num):

    plt.style.use('ggplot')
    sns.despine
    fig,ax = plt.subplots(1,1,figsize=(10,8))

    sns.boxenplot(x=cat,y = num, data=PreviousApplication)
    ax.set_ylabel(f'{num}')
    ax.set_xlabel(f'{cat}')

    ax.set_title(f'{cat} Vs {num}',fontsize=15)
    ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")

    plt.show()
```

In [98]:

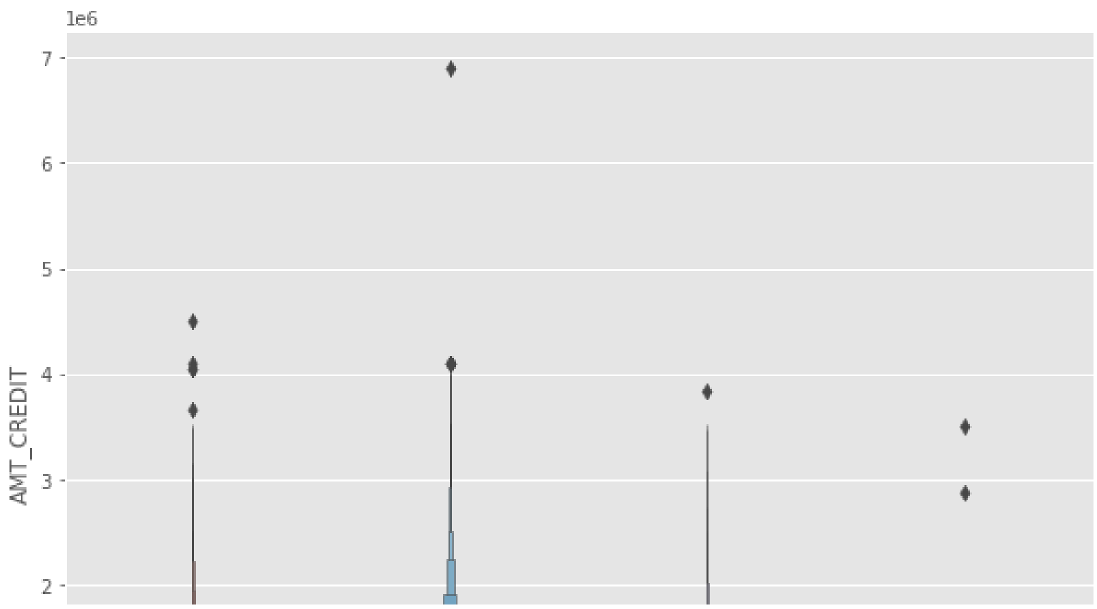
```
#by-varient analysis of Contract status and Annuity of previous appliction
plot_by_cat_num('NAME_CONTRACT_STATUS', 'AMT_ANNUITY')
```

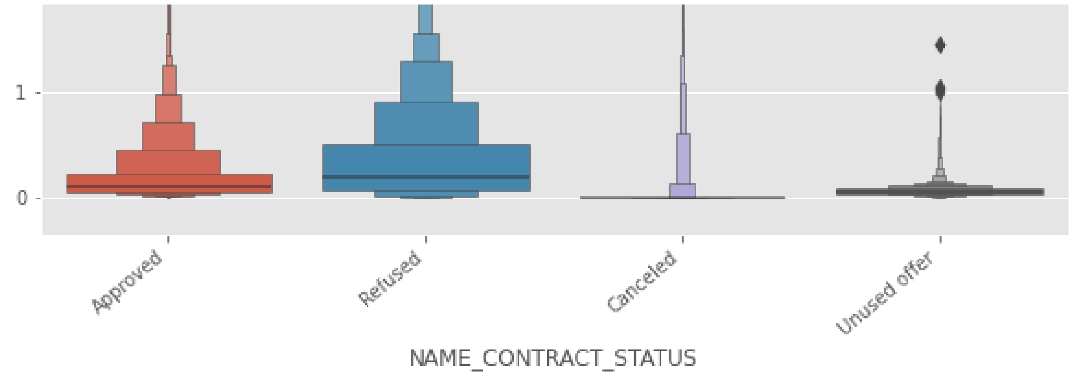


> **From the above plot we can see that loan application for people with lower AMT_ANNUITY gets canceled or Unused most of the time.**
> **We also see that applications with too high AMT ANNUITY also got refused more often than others.**

In [99]:

```
#by-varient analysis of Contract status and Final credit amount disbursed to the customer previously, after approval
plot_by_cat_num('NAME_CONTRACT_STATUS', 'AMT_CREDIT')
```





> **We can infer that when the AMT_CREDIT is too low, it get's cancelled/unused most of the time.**

6. Merging the files and analyzing the data

In [100]:

```
## Merging the two files to do some analysis
NewLeftPrev = pd.merge(NewApplication_Final, PreviousApplication, how='left', on=['SK_ID
_CURR'])
```

6.1 Basic checks on NewLeftPrev

In [101]:

```
NewLeftPrev.shape
```

Out[101]:

```
(1430100, 62)
```

In [102]:

```
NewLeftPrev.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1430100 entries, 0 to 1430099
Data columns (total 62 columns):
 #   Column                         Non-Null Count    Dtype
---  ------                         --------------    -----
 0   SK_ID_CURR                     1430100 non-null  int64
 1   TARGET                         1430100 non-null  int64
 2   CODE_GENDER                    1430100 non-null  object
 3   FLAG_OWN_CAR                   1430100 non-null  object
 4   FLAG_OWN_REALTY                1430100 non-null  object
 5   INCOME_GROUP                   1430100 non-null  category
 6   AGE_GROUP                      1430096 non-null  category
 7   AMT_CREDIT_x                   1430100 non-null  float64
 8   AMT_INCOME_TOTAL               1430100 non-null  float64
 9   CREDIT_INCOME_RATIO            1430100 non-null  float64
 10  NAME_INCOME_TYPE               1430100 non-null  object
 11  NAME_EDUCATION_TYPE            1430100 non-null  object
 12  NAME_FAMILY_STATUS             1430100 non-null  object
 13  NAME_HOUSING_TYPE              1430100 non-null  object
 14  DAYS_EMPLOYED                  1430100 non-null  int64
 15  DAYS_REGISTRATION              1430100 non-null  float64
 16  FLAG_EMAIL                     1430100 non-null  int64
 17  OCCUPATION_TYPE                1430100 non-null  object
 18  CNT_FAM_MEMBERS                1430098 non-null  float64
 19  REGION_RATING_CLIENT_W_CITY    1430100 non-null  int64
 20  ORGANIZATION_TYPE              1430100 non-null  object
 21  SOCIAL_CIRCLE_30_DAYS_DEF_PERC 684767 non-null   float64
 22  SOCIAL_CIRCLE_60_DAYS_DEF_PERC 681441 non-null   float64
```

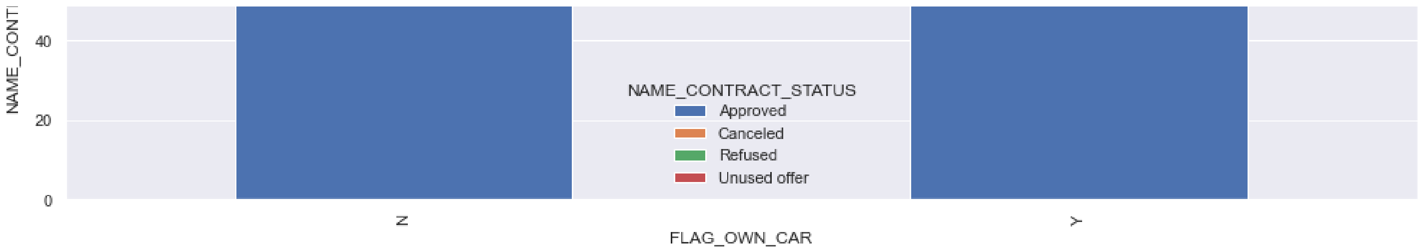
```
 23  AMT_REQ_CREDIT_BUREAU_DAY       1264288 non-null  float64
 24  AMT_REQ_CREDIT_BUREAU_MON       1264288 non-null  float64
 25  AMT_REQ_CREDIT_BUREAU_QRT       1264288 non-null  float64
 26  NAME_CONTRACT_TYPE_x            1430100 non-null  object
 27  AMT_ANNUITY_x                   1430007 non-null  float64
 28  REGION_RATING_CLIENT            1430100 non-null  int64
 29  AMT_GOODS_PRICE_x               1428881 non-null  float64
 30  SK_ID_PREV                      1413646 non-null  float64
 31  NAME_CONTRACT_TYPE_y            1413646 non-null  object
 32  AMT_ANNUITY_y                   1106438 non-null  float64
 33  AMT_APPLICATION                 1413646 non-null  float64
 34  AMT_CREDIT_y                    1413645 non-null  float64
 35  AMT_GOODS_PRICE_y               1094130 non-null  float64
 36  WEEKDAY_APPR_PROCESS_START      1413646 non-null  object
 37  HOUR_APPR_PROCESS_START         1413646 non-null  float64
 38  FLAG_LAST_APPL_PER_CONTRACT     1413646 non-null  object
 39  NFLAG_LAST_APPL_IN_DAY          1413646 non-null  float64
 40  NAME_CASH_LOAN_PURPOSE          1413646 non-null  object
 41  NAME_CONTRACT_STATUS            1413646 non-null  object
 42  DAYS_DECISION                   1413646 non-null  float64
 43  NAME_PAYMENT_TYPE               1413646 non-null  object
 44  CODE_REJECT_REASON              1413646 non-null  object
 45  NAME_TYPE_SUITE                 718992 non-null   object
 46  NAME_CLIENT_TYPE                1413646 non-null  object
 47  NAME_GOODS_CATEGORY             1413646 non-null  object
 48  NAME_PORTFOLIO                  1413646 non-null  object
 49  NAME_PRODUCT_TYPE               1413646 non-null  object
 50  CHANNEL_TYPE                    1413646 non-null  object
 51  SELLERPLACE_AREA                1413646 non-null  float64
 52  NAME_SELLER_INDUSTRY            1413646 non-null  object
 53  CNT_PAYMENT                     1106443 non-null  float64
 54  NAME_YIELD_GROUP                1413646 non-null  object
 55  PRODUCT_COMBINATION             1413333 non-null  object
 56  DAYS_FIRST_DRAWING              852573 non-null   float64
 57  DAYS_FIRST_DUE                  852573 non-null   float64
 58  DAYS_LAST_DUE_1ST_VERSION       852573 non-null   float64
 59  DAYS_LAST_DUE                   852573 non-null   float64
 60  DAYS_TERMINATION                852573 non-null   float64
 61  NFLAG_INSURED_ON_APPROVAL       852573 non-null   float64
dtypes: category(2), float64(28), int64(6), object(26)
memory usage: 668.3+ MB
```

In [111]:

```
def plotuni_combined(Varx,Vary):
    # 100% bar chart
    plt.style.use('seaborn-darkgrid')
    sns.despine
    NewDat = NewLeftPrev.pivot_table(values='SK_ID_CURR',
                      index=Varx,
                      columns=Vary,
                      aggfunc='count')
    NewDat=NewDat.div(NewDat.sum(axis=1),axis='rows')*100
    sns.set()
    NewDat.plot(kind='bar',stacked=True,figsize=(15,5))
    plt.title(f'Effect Of {Varx} on Loan Approval')
    plt.xlabel(f'{Varx}')
    plt.ylabel(f'{Vary}%')
    plt.show()
```

In [112]:

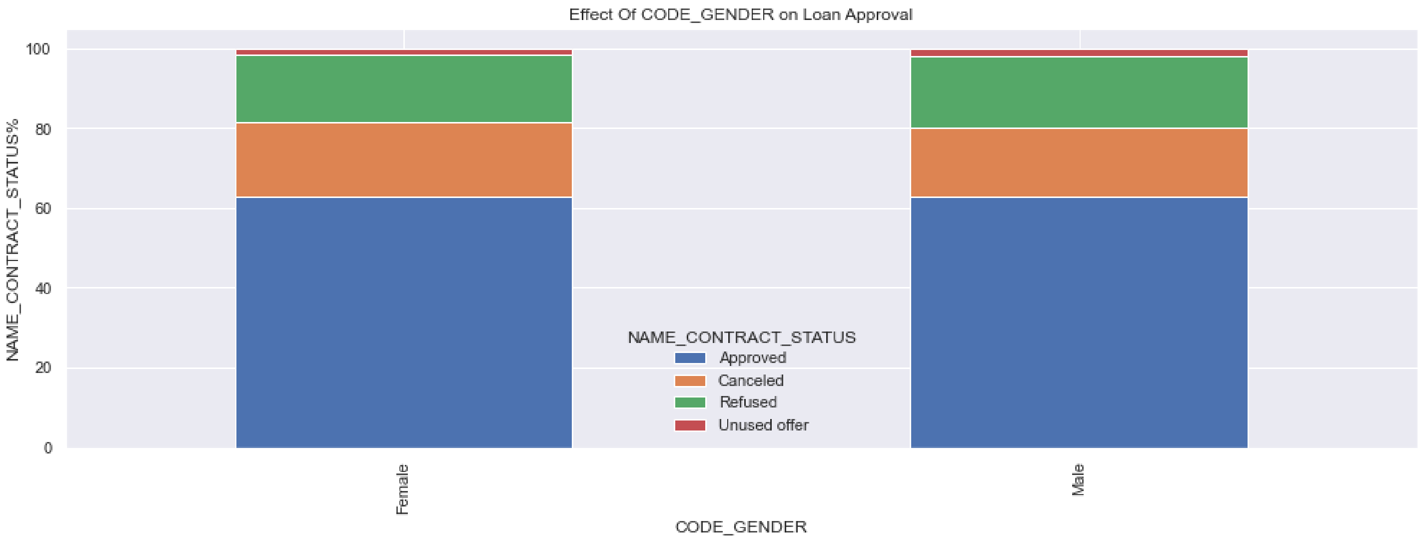
```
plotuni_combined('FLAG_OWN_CAR','NAME_CONTRACT_STATUS')
```



> **We see that car ownership doesn't have any effect on application approval or rejection. But we saw earlier that the people who has a car has lesser chances of default. The bank can add more weightage to car ownership while approving a loan amount**

In [113]:

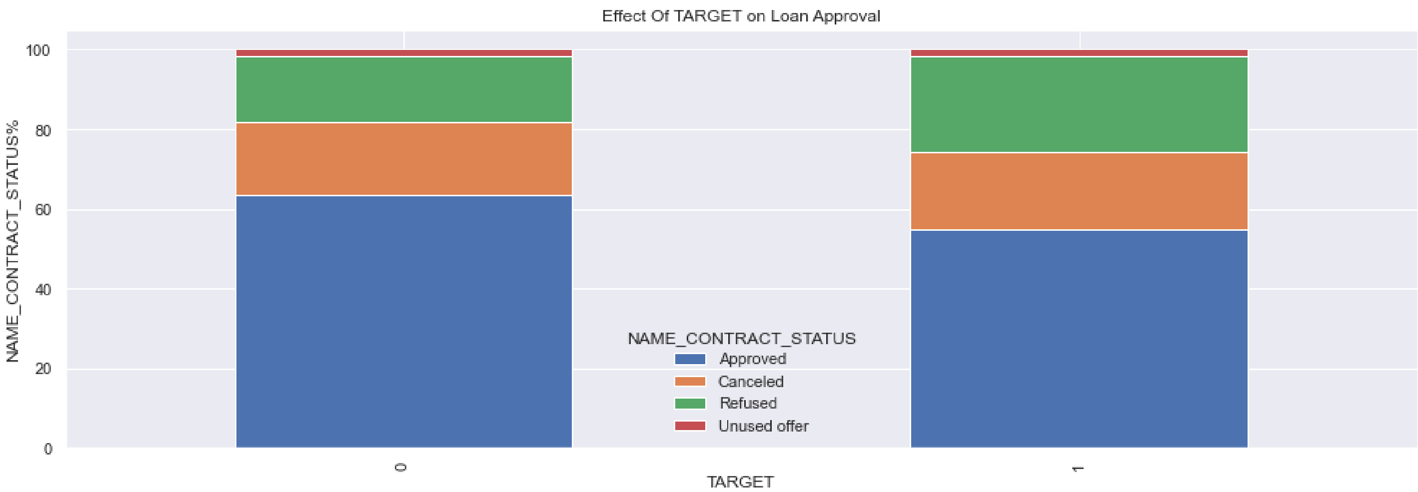
```
plotuni_combined('CODE_GENDER','NAME_CONTRACT_STATUS')
```



> **We see that code gender doesn't have any effect on application approval or rejection.**
> **But we saw earlier that female have lesser chances of default compared to males. The bank can add more weightage to female while approving a loan amount.**

In [114]:

```
plotuni_combined('TARGET','NAME_CONTRACT_STATUS')
```



Target variable (0 - Non Defaulter 1 - Defaulter)

We can see that the people who were approved for a loan earlier, defaulted less often where as people who

We can see that the people who were approved for a loan earlier, defaulted less often where as people who were refused a loan earlier have higher chances of defaulting.