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.

In [1]:
```
# Importing required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

pd.set_option('display.max_columns',200)
pd.set_option('display.max_rows',1000)

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

In [2]:
```
pre_app = pd.read_csv('previous_application.csv')       # Importing the dataset
```

In [3]:
```
pre_app.head()
```

Out[3]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRI
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	0.0	1714
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	NaN	60750
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	NaN	11250
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	NaN	45000
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	NaN	33750

In [4]:
```
# Calculating percentage of null-values
pd.DataFrame((pre_app.isnull().sum()*100)/pre_app.shape[0], columns=['% of null values'])
```

Out[4]:

	% of null values
SK_ID_PREV	0.000000
SK_ID_CURR	0.000000
NAME_CONTRACT_TYPE	0.000000
AMT_ANNUITY	22.286665
AMT_APPLICATION	0.000000
AMT_CREDIT	0.000060
AMT_DOWN_PAYMENT	53.636480
AMT_GOODS_PRICE	23.081773
WEEKDAY_APPR_PROCESS_START	0.000000
HOUR_APPR_PROCESS_START	0.000000
FLAG_LAST_APPL_PER_CONTRACT	0.000000
NFLAG_LAST_APPL_IN_DAY	0.000000
RATE_DOWN_PAYMENT	53.636480
RATE_INTEREST_PRIMARY	99.643698
RATE_INTEREST_PRIVILEGED	99.643698
NAME_CASH_LOAN_PURPOSE	0.000000
NAME_CONTRACT_STATUS	0.000000
DAYS_DECISION	0.000000
NAME_PAYMENT_TYPE	0.000000
CODE_REJECT_REASON	0.000000
NAME_TYPE_SUITE	49.119754
NAME_CLIENT_TYPE	0.000000
NAME_GOODS_CATEGORY	0.000000

	% of null values
NAME_PORTFOLIO	0.000000
NAME_PRODUCT_TYPE	0.000000
CHANNEL_TYPE	0.000000
SELLERPLACE_AREA	0.000000
NAME_SELLER_INDUSTRY	0.000000
CNT_PAYMENT	22.286366
NAME_YIELD_GROUP	0.000000
PRODUCT_COMBINATION	0.020716
DAYS_FIRST_DRAWING	40.298129
DAYS_FIRST_DUE	40.298129
DAYS_LAST_DUE_1ST_VERSION	40.298129
DAYS_LAST_DUE	40.298129
DAYS_TERMINATION	40.298129
NFLAG_INSURED_ON_APPROVAL	40.298129

In [5]:
```
# dropping those columns which have more than 5% of null values
pre_app.drop(['AMT_ANNUITY','AMT_DOWN_PAYMENT','AMT_GOODS_PRICE','RATE_DOWN_PAYMENT','RATE_INTEREST_PRIMARY','RATE_INTER
              'NAME_TYPE_SUITE','CNT_PAYMENT','DAYS_FIRST_DRAWING','DAYS_FIRST_DUE','DAYS_LAST_DUE_1ST_VERSION','DAYS_LA
              'DAYS_TERMINATION','NFLAG_INSURED_ON_APPROVAL'], axis=1, inplace=True)
```

In [6]:
```
# checking the dataset again after dropping the above columns
pre_app.head()
```

Out[6]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_APPLICATION	AMT_CREDIT	WEEKDAY_APPR_PROCESS_START	HOUR_APPR_PROCE
0	2030495	271877	Consumer loans	17145.0	17145.0	SATURDAY	
1	2802425	108129	Cash loans	607500.0	679671.0	THURSDAY	
2	2523466	122040	Cash loans	112500.0	136444.5	TUESDAY	
3	2819243	176158	Cash loans	450000.0	470790.0	MONDAY	
4	1784265	202054	Cash loans	337500.0	404055.0	THURSDAY	

In [7]:
```
pre_app.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 23 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   SK_ID_PREV                   1670214 non-null  int64  
 1   SK_ID_CURR                   1670214 non-null  int64  
 2   NAME_CONTRACT_TYPE           1670214 non-null  object 
 3   AMT_APPLICATION              1670214 non-null  float64
 4   AMT_CREDIT                   1670213 non-null  float64
 5   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object 
 6   HOUR_APPR_PROCESS_START      1670214 non-null  int64  
 7   FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object 
 8   NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64  
 9   NAME_CASH_LOAN_PURPOSE       1670214 non-null  object 
 10  NAME_CONTRACT_STATUS         1670214 non-null  object 
 11  DAYS_DECISION                1670214 non-null  int64  
 12  NAME_PAYMENT_TYPE            1670214 non-null  object 
 13  CODE_REJECT_REASON           1670214 non-null  object 
 14  NAME_CLIENT_TYPE             1670214 non-null  object 
 15  NAME_GOODS_CATEGORY          1670214 non-null  object 
 16  NAME_PORTFOLIO               1670214 non-null  object 
 17  NAME_PRODUCT_TYPE            1670214 non-null  object 
 18  CHANNEL_TYPE                 1670214 non-null  object 
 19  SELLERPLACE_AREA             1670214 non-null  int64  
 20  NAME_SELLER_INDUSTRY         1670214 non-null  object 
 21  NAME_YIELD_GROUP             1670214 non-null  object 
 22  PRODUCT_COMBINATION          1669868 non-null  object 
dtypes: float64(2), int64(6), object(15)
memory usage: 293.1+ MB
```

Imputing null-values in columns

AMT_CREDIT

In [8]:
```
pre_app.AMT_CREDIT.mean(), pre_app.AMT_CREDIT.median()
```

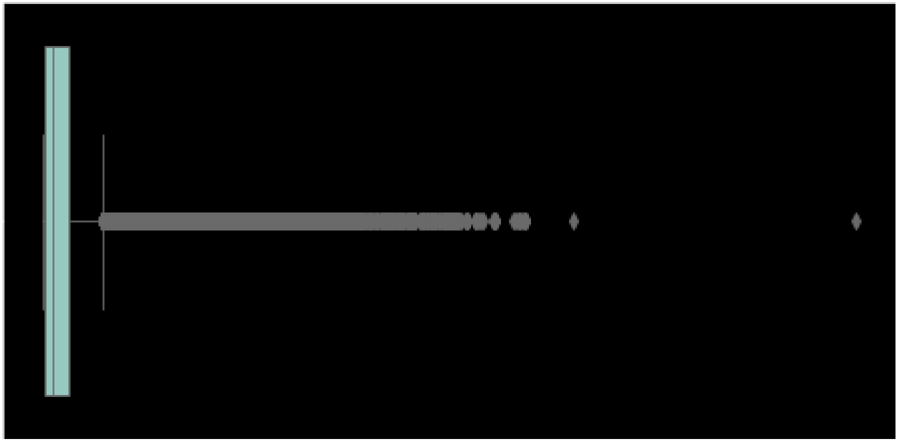
Out[8]:
```
(196114.0212179794, 80541.0)
```

In [9]:
```
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'g
rayscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark', 'seaborn-dark-palette', 'seaborn-darkgri
d', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-
talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']
```

In [10]:
```
# Creating a figure and declaring its style
fig = plt.figure(figsize=(8,4))
plt.style.use('dark_background')

# Plooting box-plot for detecting outliers
sns.boxplot(pre_app.AMT_CREDIT, color='red', palette='Set3', linewidth=0.8)
```

Out[10]:
```
<AxesSubplot:xlabel='AMT_CREDIT'>
```



In [11]:
```
# Since AMT_CREDIT above has outliers hence imputing its null-values with median
pre_app.AMT_CREDIT.fillna(pre_app.AMT_CREDIT.median(), inplace=True)
```

PRODUCT_COMBINATION

In [12]:
```
pre_app.PRODUCT_COMBINATION.value_counts()
```

Out[12]:
```
Cash                              285990
POS household with interest       263622
POS mobile with interest          220670
Cash X-Sell: middle               143883
Cash X-Sell: low                  130248
Card Street                       112582
POS industry with interest         98833
POS household without interest     82908
Card X-Sell                        80582
Cash Street: high                  59639
Cash X-Sell: high                  59301
Cash Street: middle                34658
Cash Street: low                   33834
POS mobile without interest        24082
POS other with interest            23879
POS industry without interest      12602
POS others without interest         2555
Name: PRODUCT_COMBINATION, dtype: int64
```

In [13]:
```
# Since Product_Combination is a categorical column that's why imputing its null-values with mode
pre_app.PRODUCT_COMBINATION.fillna(pre_app.PRODUCT_COMBINATION.mode()[0], inplace=True)
```

In [14]:
```
pre_app.info()                      # Checking Info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 23 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   SK_ID_PREV                   1670214 non-null  int64  
 1   SK_ID_CURR                   1670214 non-null  int64  
 2   NAME_CONTRACT_TYPE           1670214 non-null  object 
 3   AMT_APPLICATION              1670214 non-null  float64
 4   AMT_CREDIT                   1670214 non-null  float64
 5   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object 
 6   HOUR_APPR_PROCESS_START      1670214 non-null  int64  
 7   FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object 
 8   NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64  
 9   NAME_CASH_LOAN_PURPOSE       1670214 non-null  object 
 10  NAME_CONTRACT_STATUS         1670214 non-null  object 
 11  DAYS_DECISION                1670214 non-null  int64  
 12  NAME_PAYMENT_TYPE            1670214 non-null  object 
 13  CODE_REJECT_REASON           1670214 non-null  object 
 14  NAME_CLIENT_TYPE             1670214 non-null  object 
 15  NAME_GOODS_CATEGORY          1670214 non-null  object 
 16  NAME_PORTFOLIO               1670214 non-null  object 
 17  NAME_PRODUCT_TYPE            1670214 non-null  object 
 18  CHANNEL_TYPE                 1670214 non-null  object 
 19  SELLERPLACE_AREA             1670214 non-null  int64  
 20  NAME_SELLER_INDUSTRY         1670214 non-null  object 
 21  NAME_YIELD_GROUP             1670214 non-null  object 
 22  PRODUCT_COMBINATION          1670214 non-null  object 
dtypes: float64(2), int64(6), object(15)
memory usage: 293.1+ MB
```

NEW APPLICATION DATA ANALYSIS

In [15]:
```
# Importing the 2nd dataset: application.csv
app_data = pd.read_csv('application_data.csv')
```

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

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

In [17]:
```
app_data.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
```

In [18]:
```
# Checking the percentage of null-values in each column
pd.DataFrame((app_data.isnull().sum()*100)/app_data.shape[0], columns=['% of null values'])
```

FLAG_DOCUMENT_8	0.000000
FLAG_DOCUMENT_9	0.000000
FLAG_DOCUMENT_10	0.000000
FLAG_DOCUMENT_11	0.000000
FLAG_DOCUMENT_12	0.000000
FLAG_DOCUMENT_13	0.000000
FLAG_DOCUMENT_14	0.000000
FLAG_DOCUMENT_15	0.000000
FLAG_DOCUMENT_16	0.000000
FLAG_DOCUMENT_17	0.000000
FLAG_DOCUMENT_18	0.000000
FLAG_DOCUMENT_19	0.000000
FLAG_DOCUMENT_20	0.000000

In [19]:
```
those columns where null-values are more than 5%
op(['OWN_CAR_AGE','OCCUPATION_TYPE','EXT_SOURCE_1','EXT_SOURCE_3','APARTMENTS_AVG','BASEMENTAREA_AVG',
   'YEARS_BEGINEXPLUATATION_AVG','YEARS_BUILD_AVG','COMMONAREA_AVG','ELEVATORS_AVG','ENTRANCES_AVG','FLOORSMAX_AVG',
   'FLOORSMIN_AVG','LANDAREA_AVG','LIVINGAPARTMENTS_AVG','NONLIVINGAPARTMENTS_AVG','NONLIVINGAREA_AVG','APARTMENTS_MODE',
   'BASEMENTAREA_MODE','YEARS_BEGINEXPLUATATION_MODE','YEARS_BUILD_MODE','COMMONAREA_MODE','ELEVATORS_MODE','ENTRANCES_MO
   'FLOORSMAX_MODE','FLOORSMIN_MODE','LANDAREA_MODE','LIVINGAPARTMENTS_MODE','LIVINGAREA_MODE','NONLIVINGAPARTMENTS_MODE'
   'NONLIVINGAREA_MODE','APARTMENTS_MEDI','BASEMENTAREA_MEDI','YEARS_BEGINEXPLUATATION_MEDI','YEARS_BUILD_MEDI',
   'COMMONAREA_MEDI','ELEVATORS_MEDI','ENTRANCES_MEDI','FLOORSMAX_MEDI','FLOORSMIN_MEDI','LANDAREA_MEDI','LIVINGAPARTMENT
   'NONLIVINGAREA_MEDI','FONDKAPREMONT_MODE','HOUSETYPE_MODE','TOTALAREA_MODE','WALLSMATERIAL_MODE','EMERGENCYSTATE_MODE'
   'AMT_REQ_CREDIT_BUREAU_HOUR','AMT_REQ_CREDIT_BUREAU_DAY','AMT_REQ_CREDIT_BUREAU_WEEK','AMT_REQ_CREDIT_BUREAU_MON',
    'AMT_REQ_CREDIT_BUREAU_QRT','AMT_REQ_CREDIT_BUREAU_YEAR','LIVINGAREA_AVG','LIVINGAREA_MEDI','NONLIVINGAPARTMENTS_MEDI
   , axis=1, inplace=True)
```

In [20]:
```
app_data.shape
```

Out[20]:
```
(307511, 65)
```

In [21]:
```
app_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 65 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   SK_ID_CURR                   307511 non-null  int64  
 1   TARGET                       307511 non-null  int64  
 2   NAME_CONTRACT_TYPE           307511 non-null  object 
 3   CODE_GENDER                  307511 non-null  object 
 4   FLAG_OWN_CAR                 307511 non-null  object 
 5   FLAG_OWN_REALTY              307511 non-null  object 
 6   CNT_CHILDREN                 307511 non-null  int64  
 7   AMT_INCOME_TOTAL             307511 non-null  float64
 8   AMT_CREDIT                   307511 non-null  float64
 9   AMT_ANNUITY                  307499 non-null  float64
 10  AMT_GOODS_PRICE              307233 non-null  float64
 11  NAME_TYPE_SUITE              306219 non-null  object 
 12  NAME_INCOME_TYPE             307511 non-null  object 
 13  NAME_EDUCATION_TYPE          307511 non-null  object 
 14  NAME_FAMILY_STATUS           307511 non-null  object 
 15  NAME_HOUSING_TYPE            307511 non-null  object 
 16  REGION_POPULATION_RELATIVE   307511 non-null  float64
 17  DAYS_BIRTH                   307511 non-null  int64  
 18  DAYS_EMPLOYED                307511 non-null  int64  
 19  DAYS_REGISTRATION            307511 non-null  float64
 20  DAYS_ID_PUBLISH              307511 non-null  int64  
 21  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  CNT_FAM_MEMBERS              307509 non-null  float64
 28  REGION_RATING_CLIENT         307511 non-null  int64  
 29  REGION_RATING_CLIENT_W_CITY  307511 non-null  int64  
 30  WEEKDAY_APPR_PROCESS_START   307511 non-null  object 
 31  HOUR_APPR_PROCESS_START      307511 non-null  int64  
 32  REG_REGION_NOT_LIVE_REGION   307511 non-null  int64  
 33  REG_REGION_NOT_WORK_REGION   307511 non-null  int64  
```

```
 34  LIVE_REGION_NOT_WORK_REGION  307511 non-null  int64
 35  REG_CITY_NOT_LIVE_CITY       307511 non-null  int64
 36  REG_CITY_NOT_WORK_CITY       307511 non-null  int64
 37  LIVE_CITY_NOT_WORK_CITY      307511 non-null  int64
 38  ORGANIZATION_TYPE            307511 non-null  object
 39  EXT_SOURCE_2                 306851 non-null  float64
 40  OBS_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 41  DEF_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 42  OBS_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 43  DEF_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 44  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64
 45  FLAG_DOCUMENT_2              307511 non-null  int64
 46  FLAG_DOCUMENT_3              307511 non-null  int64
 47  FLAG_DOCUMENT_4              307511 non-null  int64
 48  FLAG_DOCUMENT_5              307511 non-null  int64
 49  FLAG_DOCUMENT_6              307511 non-null  int64
 50  FLAG_DOCUMENT_7              307511 non-null  int64
 51  FLAG_DOCUMENT_8              307511 non-null  int64
 52  FLAG_DOCUMENT_9              307511 non-null  int64
 53  FLAG_DOCUMENT_10             307511 non-null  int64
 54  FLAG_DOCUMENT_11             307511 non-null  int64
 55  FLAG_DOCUMENT_12             307511 non-null  int64
 56  FLAG_DOCUMENT_13             307511 non-null  int64
 57  FLAG_DOCUMENT_14             307511 non-null  int64
 58  FLAG_DOCUMENT_15             307511 non-null  int64
 59  FLAG_DOCUMENT_16             307511 non-null  int64
 60  FLAG_DOCUMENT_17             307511 non-null  int64
 61  FLAG_DOCUMENT_18             307511 non-null  int64
 62  FLAG_DOCUMENT_19             307511 non-null  int64
 63  FLAG_DOCUMENT_20             307511 non-null  int64
 64  FLAG_DOCUMENT_21             307511 non-null  int64
dtypes: float64(13), int64(41), object(11)
memory usage: 152.5+ MB
```

In [22]:
```
app_data.head()
```

Out[22]:

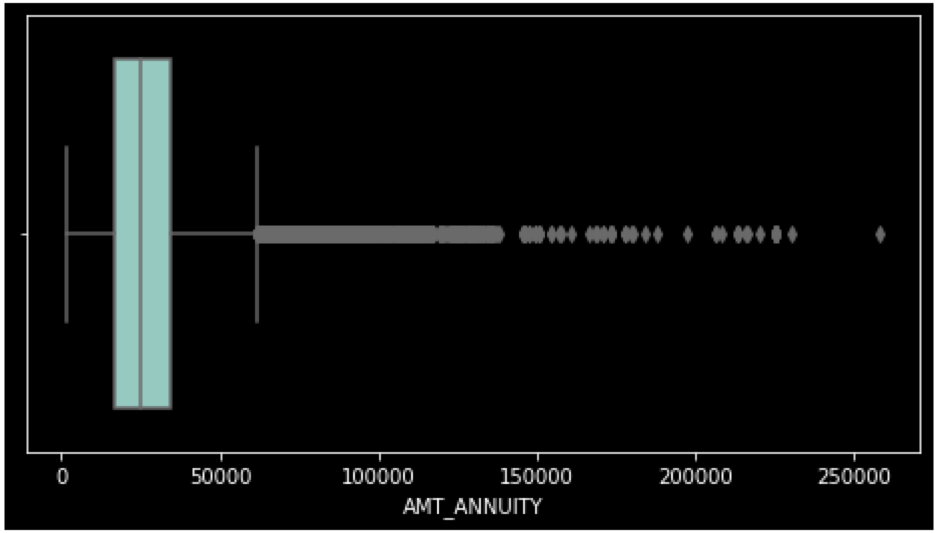
	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL
0	100002	1	Cash loans	M	N	Y	0	202500.0
1	100003	0	Cash loans	F	N	N	0	270000.0
2	100004	0	Revolving loans	M	Y	Y	0	67500.0
3	100006	0	Cash loans	F	N	Y	0	135000.0
4	100007	0	Cash loans	M	N	Y	0	121500.0

In [23]:
```
# columns to impute missing values
list(app_data.columns[(app_data.isnull().mean()<=5) & (app_data.isnull().mean()>0)])
```

Out[23]:
```
['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']
```

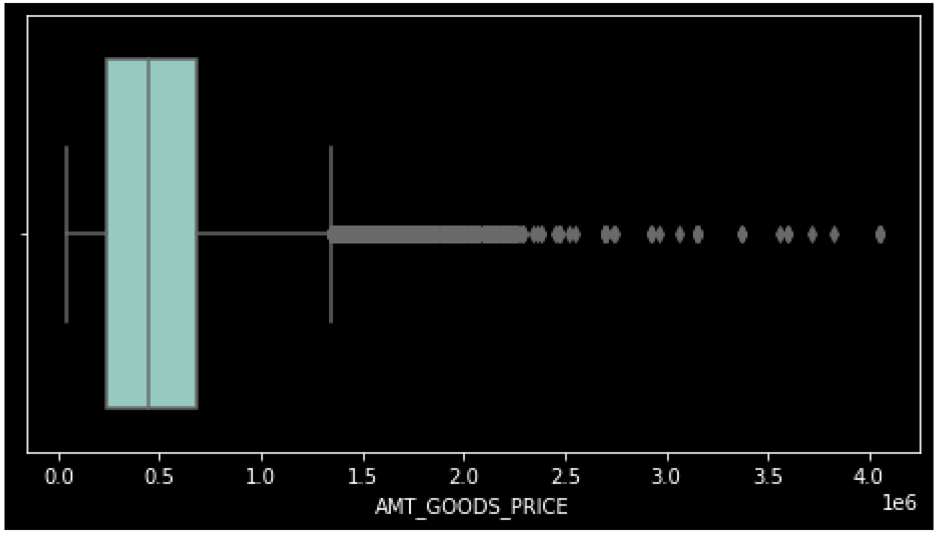
In [24]:
```
# Defining a function to plot the columns
def plot(var):
    fig= plt.figure(figsize=(8,4))
    plt.style.use('dark_background')
    sns.boxplot(app_data[var])
```

In [25]:
```
plot('AMT_ANNUITY')
```



In [26]:
```
# Since outliers are present that's why imputing missing values with median
app_data.AMT_ANNUITY.fillna(app_data.AMT_ANNUITY.median(), inplace=True)
```

In [27]:
```
plot('AMT_GOODS_PRICE')
```



In [28]:
```
# Since outliers are present that's why imputing missing values with median
app_data.AMT_GOODS_PRICE.fillna(app_data.AMT_GOODS_PRICE.median(), inplace=True)
```

In [29]:
```
app_data.NAME_TYPE_SUITE.value_counts()
```

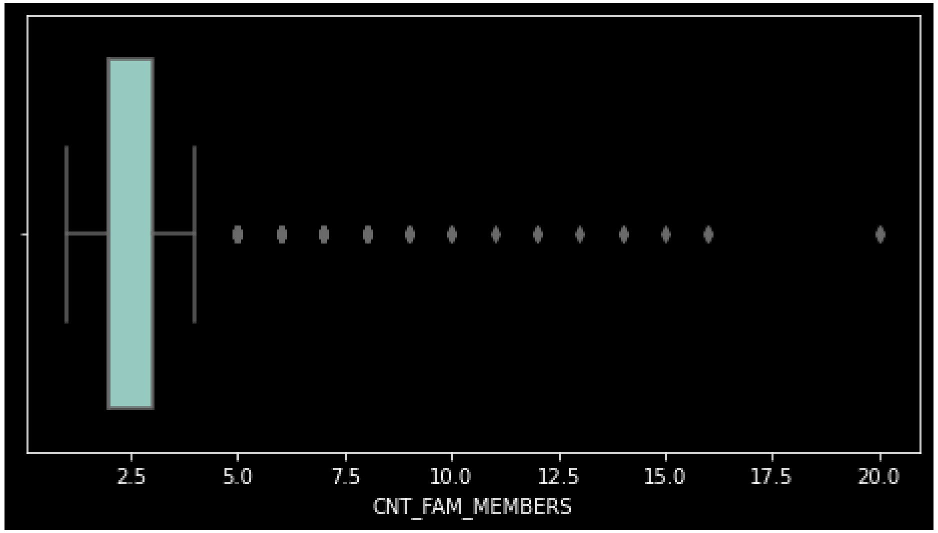
Out[29]:
```
Unaccompanied      248526
Family              40149
Spouse, partner     11370
Children             3267
Other_B              1770
Other_A               866
Group of people       271
Name: NAME_TYPE_SUITE, dtype: int64
```

In [30]:
```
# Since NAME_TYPE_SUITE is a categorical column that's why imputing missing values with mode
app_data.NAME_TYPE_SUITE.fillna(app_data.NAME_TYPE_SUITE.mode()[0], inplace=True)
```

In [31]:
```
app_data['CNT_FAM_MEMBERS'].value_counts()
```

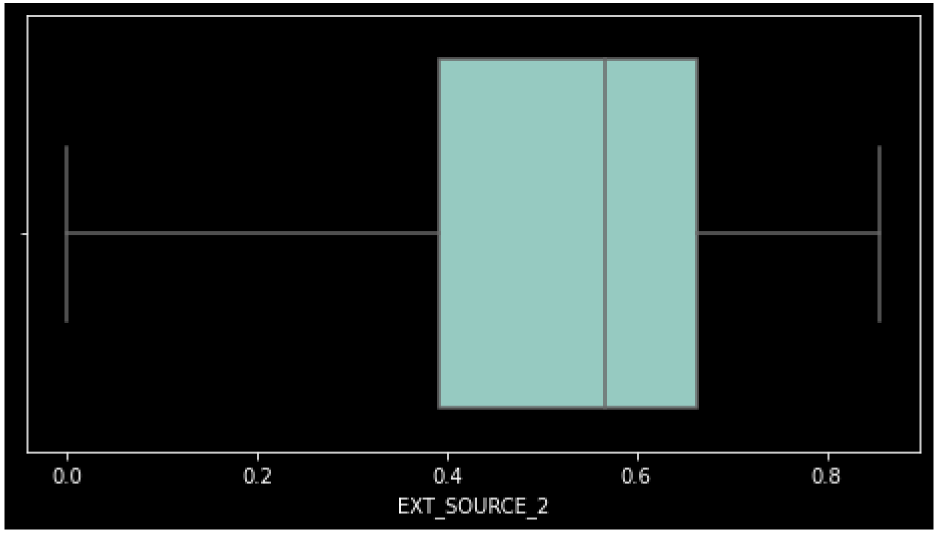
Out[31]:
```
2.0     158357
1.0      67847
3.0      52601
4.0      24697
5.0       3478
6.0        408
7.0         81
8.0         20
9.0          6
10.0         3
14.0         2
16.0         2
12.0         2
20.0         2
11.0         1
13.0         1
15.0         1
Name: CNT_FAM_MEMBERS, dtype: int64
```

In [32]:
```
plot('CNT_FAM_MEMBERS')
```



In [33]:
```
# Since outliers are present that's why imputing missing values with median (imputing mean value will not make sense here
# as it will give some decimal value which can't be a possible value for count_of_family_members)
app_data.CNT_FAM_MEMBERS.fillna(app_data.CNT_FAM_MEMBERS.median(), inplace=True)
```

In [34]:
```
plot('EXT_SOURCE_2')
```

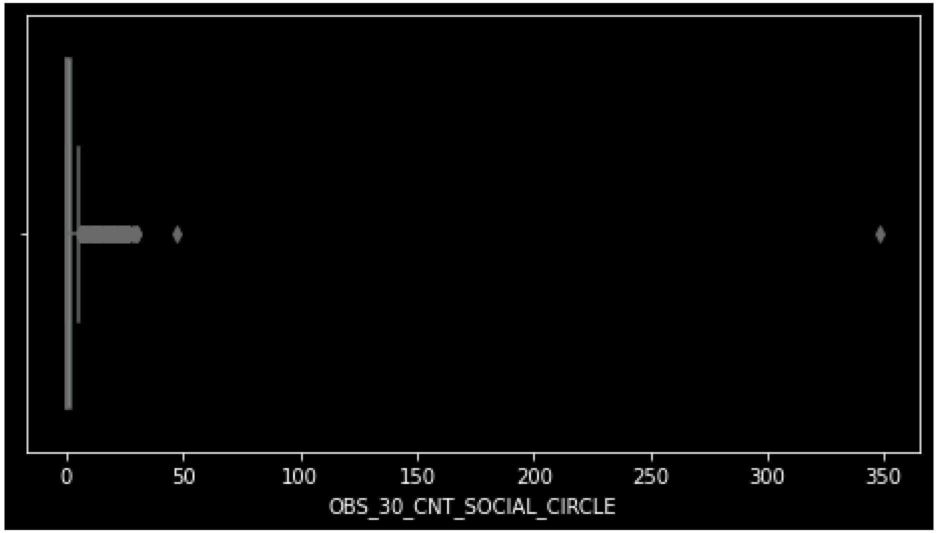


In [35]:
```
app_data.EXT_SOURCE_2.value_counts()
```

Out[35]:
```
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
```

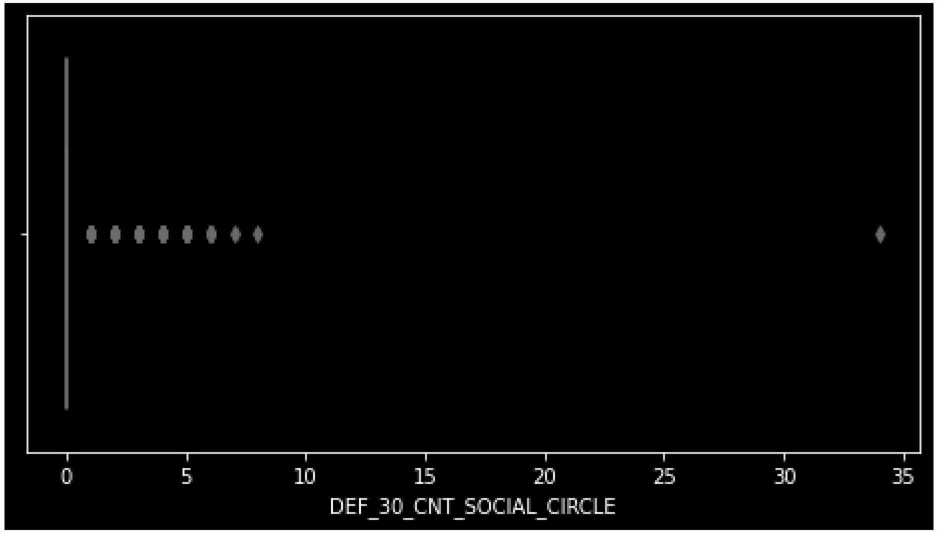
In [36]:
```
# Since outliers are not present that's why imputing missing values with mean
app_data.EXT_SOURCE_2 = app_data.EXT_SOURCE_2.fillna(app_data.EXT_SOURCE_2.mean(), inplace=True)
```

In [37]:
```
plot('OBS_30_CNT_SOCIAL_CIRCLE')
```



In [38]:
```
# Since outliers are present that's why imputing missing values with median
app_data.OBS_30_CNT_SOCIAL_CIRCLE = app_data.OBS_30_CNT_SOCIAL_CIRCLE.fillna(app_data.OBS_30_CNT_SOCIAL_CIRCLE.median(),
                                                                              inplace=True)
```

In [39]:
```
plot('DEF_30_CNT_SOCIAL_CIRCLE')
```

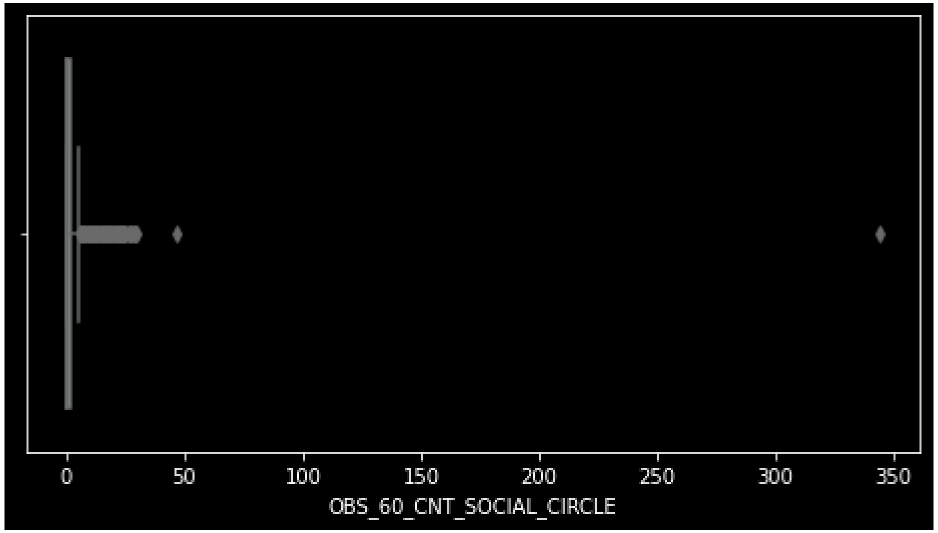


In [40]:
```
app_data.DEF_30_CNT_SOCIAL_CIRCLE.value_counts()
```

Out[40]:
```
0.0     271324
1.0      28328
2.0       5323
3.0       1192
4.0        253
5.0         56
6.0         11
7.0          1
8.0          1
34.0         1
Name: DEF_30_CNT_SOCIAL_CIRCLE, dtype: int64
```

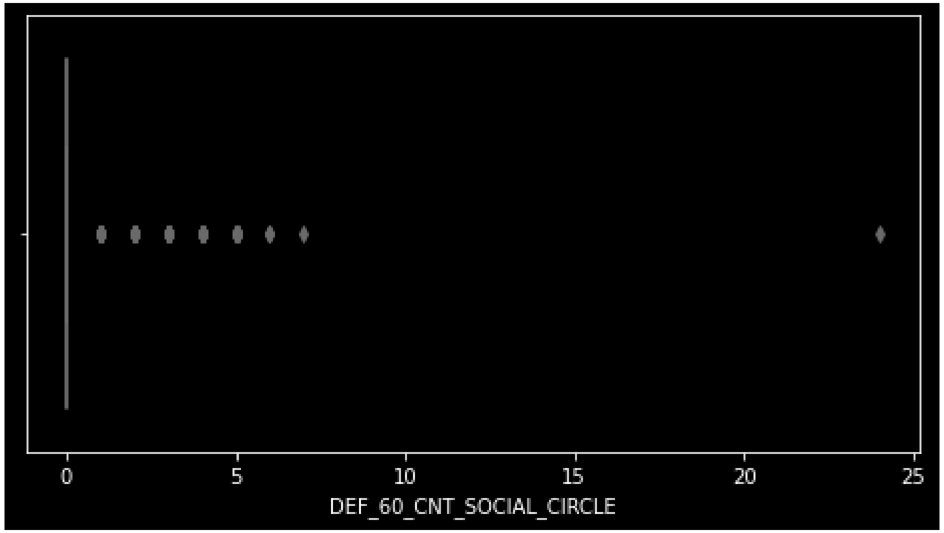
In [41]:
```
# Since outliers are present that's why imputing missing values with median
app_data.DEF_30_CNT_SOCIAL_CIRCLE = app_data.DEF_30_CNT_SOCIAL_CIRCLE.fillna(app_data.DEF_30_CNT_SOCIAL_CIRCLE.median(),
                                                                             inplace=True)
```

In [42]:
```
plot('OBS_60_CNT_SOCIAL_CIRCLE')
```



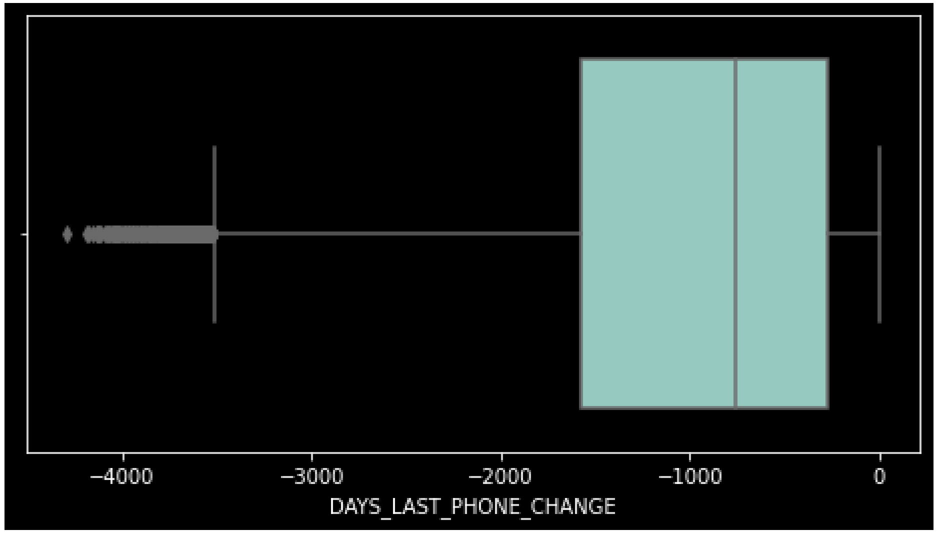
In [43]:
```
# Since outliers are present that's why imputing missing values with median
app_data.OBS_60_CNT_SOCIAL_CIRCLE = app_data.OBS_60_CNT_SOCIAL_CIRCLE.fillna(app_data.OBS_60_CNT_SOCIAL_CIRCLE.median(),
                                                                              inplace=True)
```

In [44]:
```
plot('DEF_60_CNT_SOCIAL_CIRCLE')
```



In [45]:
```
# Since outliers are present that's why imputing missing values with median
app_data.DEF_60_CNT_SOCIAL_CIRCLE = app_data.DEF_60_CNT_SOCIAL_CIRCLE.fillna(app_data.DEF_60_CNT_SOCIAL_CIRCLE.median(),
                                                                              inplace=True)
```

In [46]:
```
plot('DAYS_LAST_PHONE_CHANGE')
```



In [47]:
```
# Since outliers are present that's why imputing missing values with median
app_data.DAYS_LAST_PHONE_CHANGE = app_data.DAYS_LAST_PHONE_CHANGE.fillna(app_data.DAYS_LAST_PHONE_CHANGE.median(), inpla
```

In [48]:
```
# Checking CODE_GENDER column
app_data['CODE_GENDER'].value_counts()
```

Out[48]:
```
F      202448
M      105059
XNA         4
Name: CODE_GENDER, dtype: int64
```

In [49]:
```
# Droppping rows with CODE_GENDER = XNA since the rows are very less
new_app_data = app_data[app_data['CODE_GENDER']!='XNA']
```

In [50]:
```
# Making Gender more readable
new_app_data['CODE_GENDER'].replace({'F':'Female','M':'Male'}, inplace=True)
```

In [51]:
```
# Checking the dataset
new_app_data.head()
```

Out[51]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL
0	100002	1	Cash loans	Male	N	Y	0	202500.0
1	100003	0	Cash loans	Female	N	N	0	270000.0
2	100004	0	Revolving loans	Male	Y	Y	0	67500.0
3	100006	0	Cash loans	Female	N	Y	0	135000.0
4	100007	0	Cash loans	Male	N	Y	0	121500.0

Binning Numerical Variables For Analysis

In [52]:
```
# Calculating quantiles for numerical variable, here AMT_INCOME_TOTAL
new_app_data['AMT_INCOME_TOTAL'].quantile([0,0.1,0.3,0.5,0.6,0.8,1.0])
```

Out[52]:
```
0.0        25650.0
0.1        81000.0
0.3       112500.0
0.5       147150.0
0.6       162000.0
0.8       225000.0
1.0    117000000.0
Name: AMT_INCOME_TOTAL, dtype: float64
```

In [53]:
```
# Creating a new categorical variable based on above numerical column for analysis
new_app_data['INCOME_GROUP'] = pd.qcut(new_app_data['AMT_INCOME_TOTAL'], q =[0,0.1,0.3,0.6,0.8,1],
                                        labels = ['Very Low','Low','Medium','High','Very High'])
```

In [54]:
```
new_app_data['INCOME_GROUP'] = new_app_data['INCOME_GROUP'].astype('object')   #Converting into categorical column type
```

In [55]:
```
# Binning Days Birth
abs(new_app_data['DAYS_BIRTH']).quantile([0,0.1,0.3,0.6,0.8,1])
```

Out[55]:
```
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 [56]:
```
# Creating a column age using days_birth
new_app_data['AGE'] = abs(new_app_data['DAYS_BIRTH'])//365.25
```

In [57]:
```
new_app_data.head()
```

Out[57]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL
0	100002	1	Cash loans	Male	N	Y	0	202500.0
1	100003	0	Cash loans	Female	N	N	0	270000.0
2	100004	0	Revolving loans	Male	Y	Y	0	67500.0
3	100006	0	Cash loans	Female	N	Y	0	135000.0
4	100007	0	Cash loans	Male	N	Y	0	121500.0

In [58]:
```
new_app_data.AGE.describe()
```

Out[58]:
```
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 [59]:
```
# Now converting this age into a categorical column via binning for analysis
## Since the AGE varies from 20 to 69, we can create bins of 5 years starting from 20 to 70
new_app_data['AGE_GROUP'] = pd.cut(new_app_data['AGE'],bins=np.arange(20,71,5))  # Here we didn't use .qcut instead we h
new_app_data['AGE_GROUP'] = new_app_data['AGE_GROUP'].astype('object')
```

In [60]:
```
new_app_data.head()
```

Out[60]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL
0	100002	1	Cash loans	Male	N	Y	0	202500.0
1	100003	0	Cash loans	Female	N	N	0	270000.0
2	100004	0	Revolving loans	Male	Y	Y	0	67500.0
3	100006	0	Cash loans	Female	N	Y	0	135000.0
4	100007	0	Cash loans	Male	N	Y	0	121500.0

In [61]:
```
#app0_data = app_data[app_data.TARGET==0]
#app1_data = app_data[app_data.TARGET==1]
```

In [62]:
```
#app0_data.shape, app1_data.shape
```

In [63]:
```
# Adding one more column
new_app_data['CREDIT_INCOME_RATIO'] = round((new_app_data['AMT_CREDIT']/new_app_data['AMT_INCOME_TOTAL']))
```

In [64]:
```
new_app_data
```

Out[64]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_T
0	100002	1	Cash loans	Male	N	Y	0	202
1	100003	0	Cash loans	Female	N	N	0	270
2	100004	0	Revolving loans	Male	Y	Y	0	67
3	100006	0	Cash loans	Female	N	Y	0	135
4	100007	0	Cash loans	Male	N	Y	0	121
...	...	...	...	...	...	...	...	
307506	456251	0	Cash loans	Male	N	N	0	157
307507	456252	0	Cash loans	Female	N	Y	0	72
307508	456253	0	Cash loans	Female	N	Y	0	153
307509	456254	1	Cash loans	Female	N	Y	0	171
307510	456255	0	Cash loans	Female	N	N	0	157

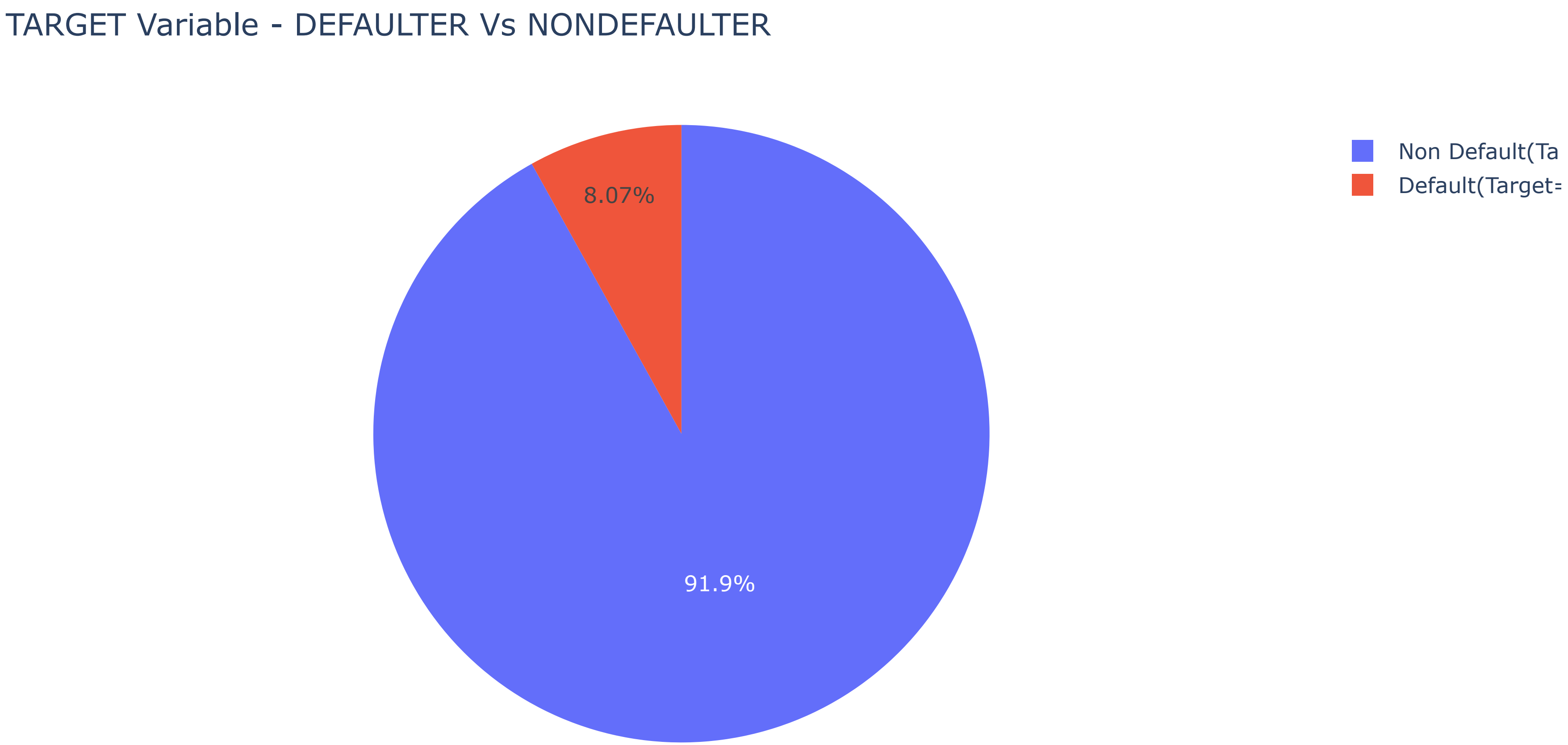
307507 rows × 69 columns

In [65]:
```
Getting the percentage of social circles who defaulted
ew_app_data['SOCIAL_CIRCLE_30_DAYS_DEF_PERC'] = new_app_data['DEF_30_CNT_SOCIAL_CIRCLE']/new_app_data['OBS_30_CNT_SOCIAL_(
ew_app_data['SOCIAL_CIRCLE_60_DAYS_DEF_PERC'] = new_app_data['DEF_60_CNT_SOCIAL_CIRCLE']/new_app_data['OBS_60_CNT_SOCIAL_(
```

In [66]:
```
new_app_data.TARGET.value_counts(normalize=True)*100
```

Out[66]:
```
0    91.927013
1     8.072987
Name: TARGET, dtype: float64
```

In [67]:
```
# Now that the feature engineering is done, lets move onto our target variable and analyze it
px.pie(new_app_data.TARGET.value_counts(normalize=True)*100, values='TARGET', names = ['Non Default(Target=0)', 'Default
      title = 'TARGET Variable - DEFAULTER Vs NONDEFAULTER')
```



In [68]:
```
# From the remaining columns about 30 are selected based on their description and relevance with problem statement
# for further analysis
FinalColumns = ['SK_ID_CURR','TARGET','CODE_GENDER','FLAG_OWN_CAR','FLAG_OWN_REALTY','INCOME_GROUP','AGE_GROUP','AMT_CREI
'CREDIT_INCOME_RATIO','NAME_INCOME_TYPE','NAME_EDUCATION_TYPE','NAME_FAMILY_STATUS','NAME_HOUSING_TYPE','DAYS_EMPLOYED',
'DAYS_REGISTRATION','FLAG_EMAIL', 'CNT_FAM_MEMBERS','REGION_RATING_CLIENT_W_CITY','ORGANIZATION_TYPE','SOCIAL_CIRCLE_30_I
'SOCIAL_CIRCLE_60_DAYS_DEF_PERC','NAME_CONTRACT_TYPE','AMT_ANNUITY','REGION_RATING_CLIENT','AMT_GOODS_PRICE']
```

In [69]:
```
new_app_data = new_app_data[FinalColumns]
```

In [70]:
```
new_app_data.shape
```

Out[70]:
```
(307507, 26)
```

In [71]:
```
# Splitting the df into two different dataframes
newapp0 = new_app_data[new_app_data.TARGET==0]          # Dataframe with all data related to non-defaulters
newapp1 = new_app_data[new_app_data.TARGET==1]          # Dataframe with all data related to defaulters
```

In [72]:
```
newapp0.shape, newapp1.shape
```

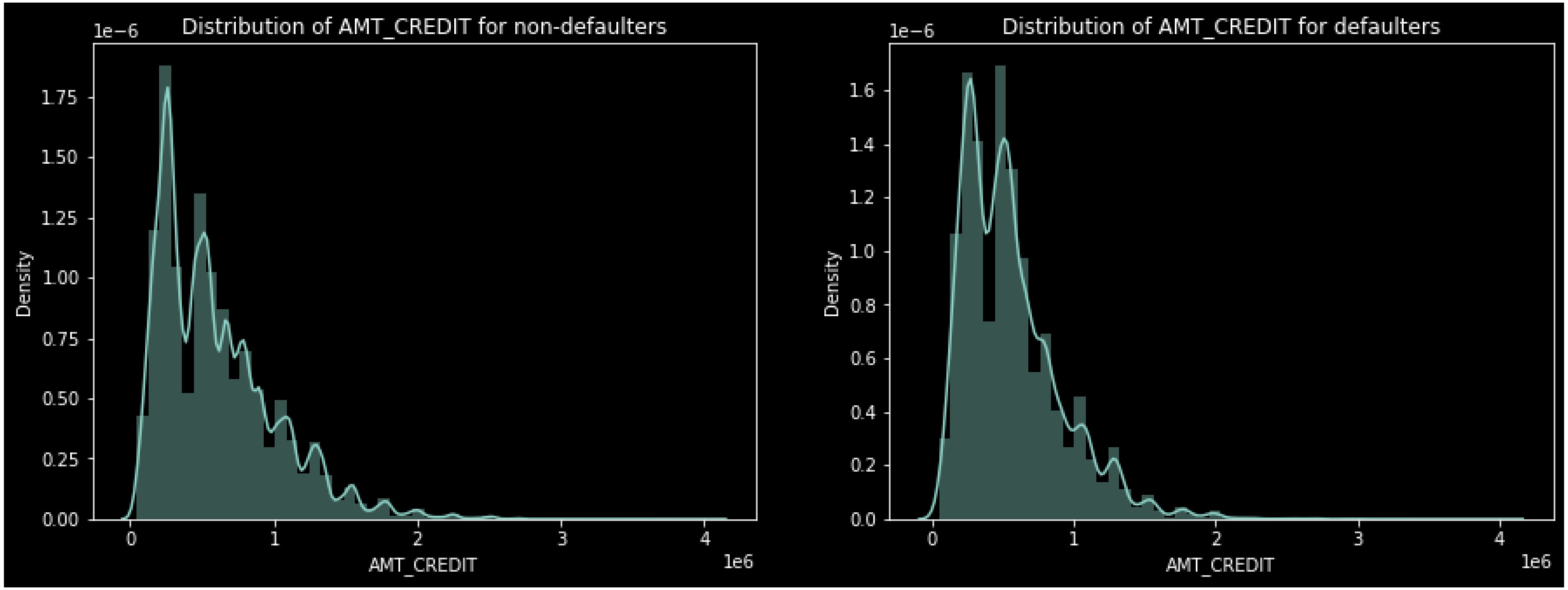
Out[72]:
```
((282682, 26), (24825, 26))
```

Univariate analysis for each of these datasets

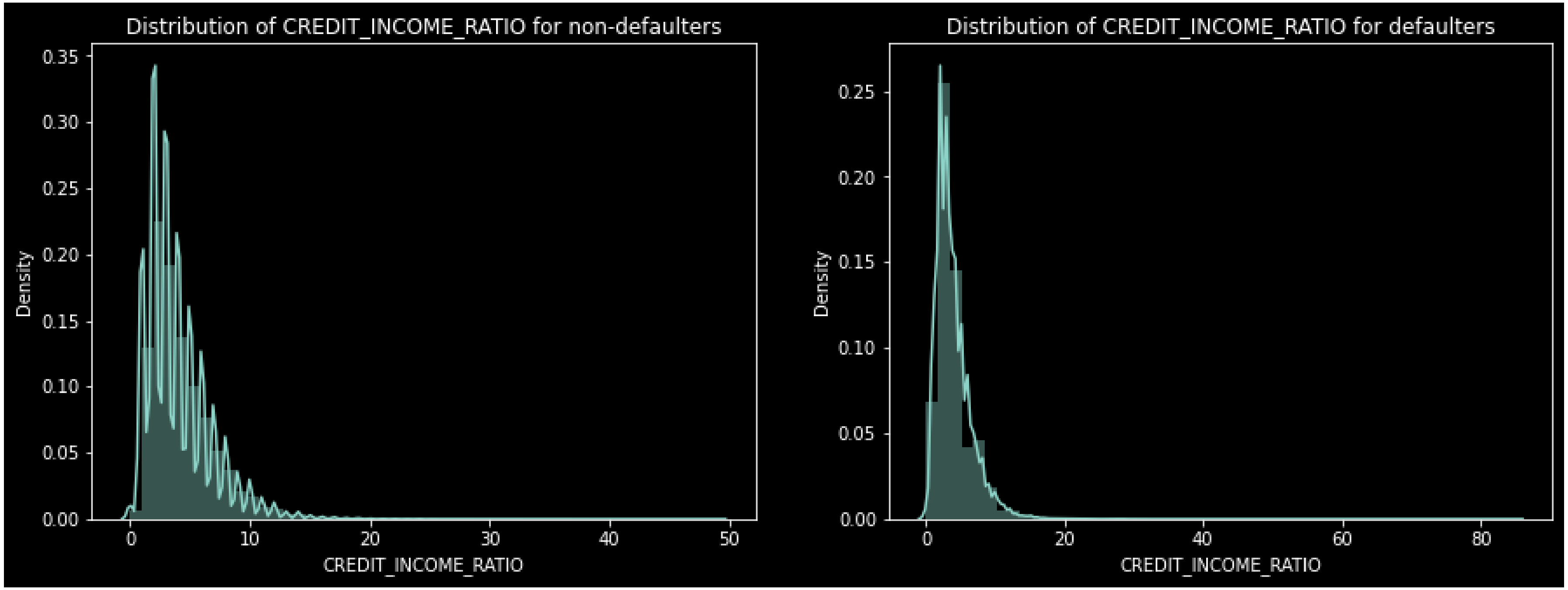
Function to plot univariate numerical variables

In [73]:
```
def unicomp(var):
    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')
    plt.xlabel(var)

    sns.distplot(a=newapp1[var], ax=ax2)
    ax2.set_title(f'Distribution of {var} for defaulters')
    plt.xlabel(var)
    plt.show()
```

In [74]:
```
unicomp('AMT_CREDIT')
```

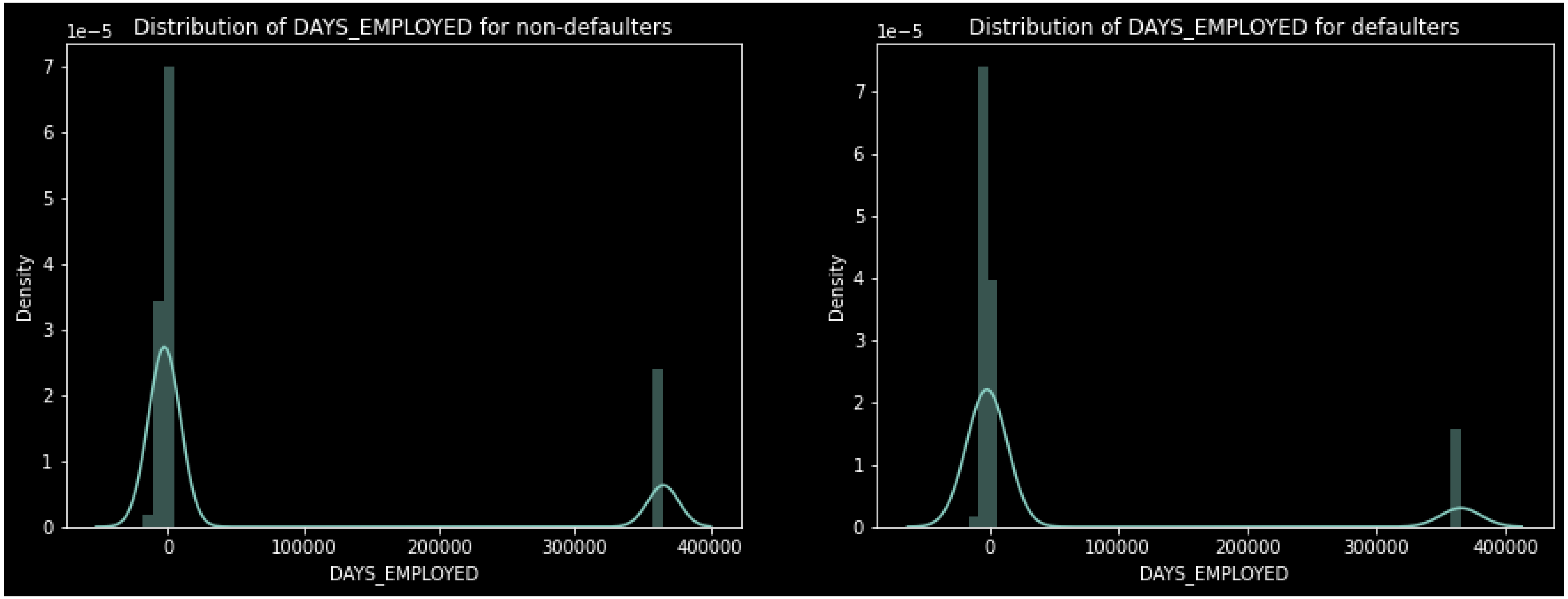


In [75]:
```
unicomp('CREDIT_INCOME_RATIO')
```



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

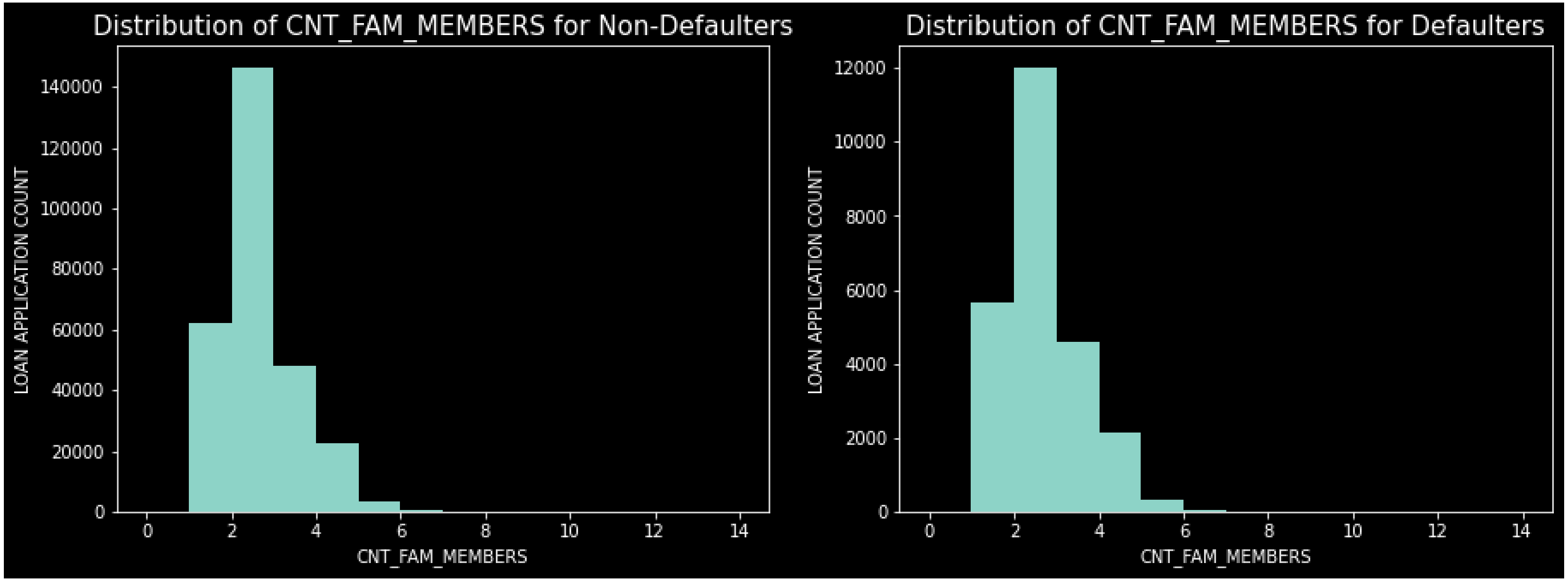
In [76]:
```
unicomp('DAYS_EMPLOYED')
```



In [77]:
```
new_app_data.CNT_FAM_MEMBERS.value_counts()
```

Out[77]:
```
2.0     158357
1.0      67847
3.0      52600
4.0      24696
5.0       3478
6.0        408
7.0         81
8.0         20
9.0          6
10.0         3
14.0         2
16.0         2
12.0         2
20.0         2
11.0         1
13.0         1
15.0         1
Name: CNT_FAM_MEMBERS, dtype: int64
```

In [78]:
```
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 apply for loan more often than other size families

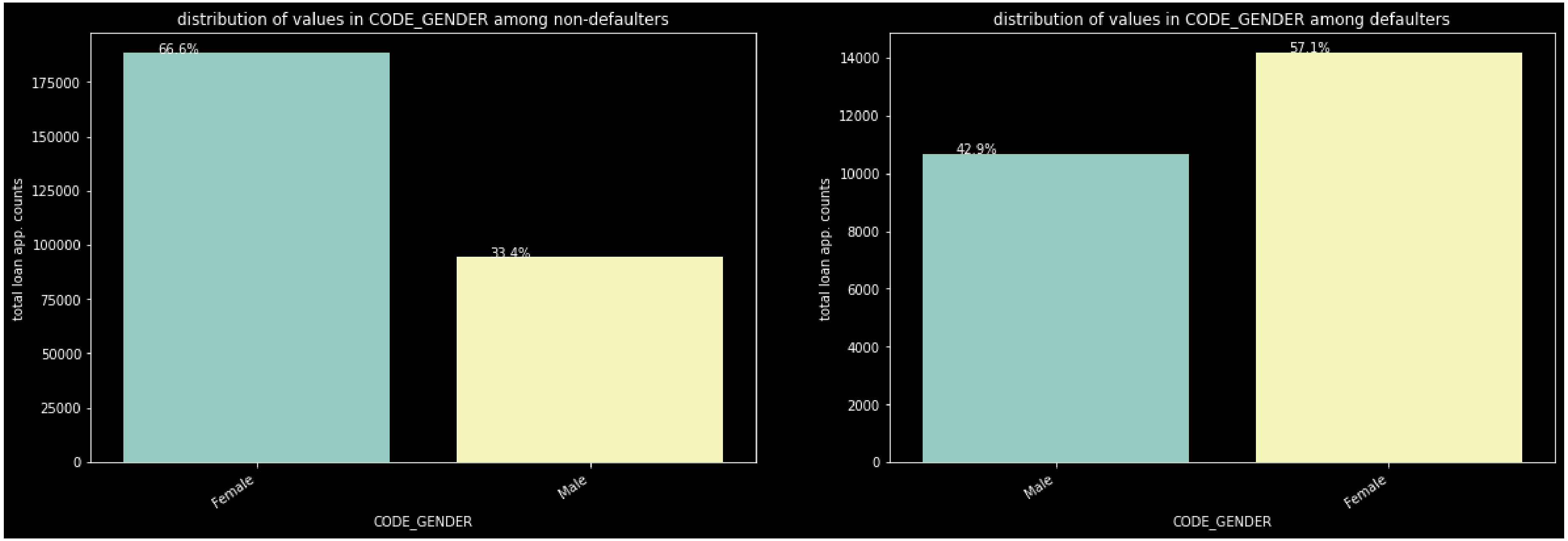
Functions to plot univariate categorical variables

In [79]:
```
def unicat(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_title(f'distribution of values in {var} among non-defaulters')
    ax1.set_ylabel('total loan app. counts')
    ax1.set_xticklabels(ax1.get_xticklabels(), rotation=35, ha='right')     # modifying x tick labels

    # Getting annotations for ax1 plot for easier comparison between defaulters and non-defaulters
    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_title(f'distribution of values in {var} among defaulters')
    ax2.set_ylabel('total loan app. counts')
    ax2.set_xticklabels(ax2.get_xticklabels(), rotation=35, ha='right')      # modifying x tick labels

    # Getting annotations for ax1 plot for easier comparison between defaulters and non-defaulters
    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()
```

In [80]:
```
newapp0.select_dtypes(include='object').columns         # Checking categorical columns in newapp0 (non-defaulters)
```

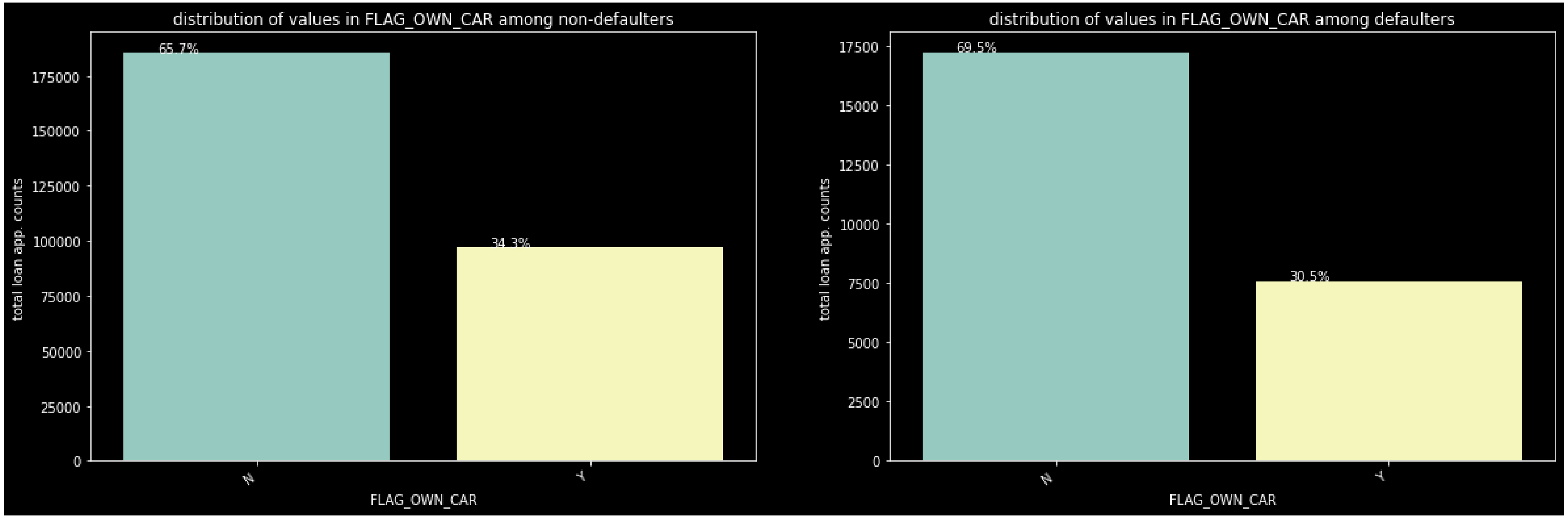
Out[80]:
```
Index(['CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'INCOME_GROUP',
       'AGE_GROUP', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'ORGANIZATION_TYPE',
       'SOCIAL_CIRCLE_30_DAYS_DEF_PERC', 'SOCIAL_CIRCLE_60_DAYS_DEF_PERC',
       'NAME_CONTRACT_TYPE'],
      dtype='object')
```

In [81]:
```
unicat('CODE_GENDER')
```



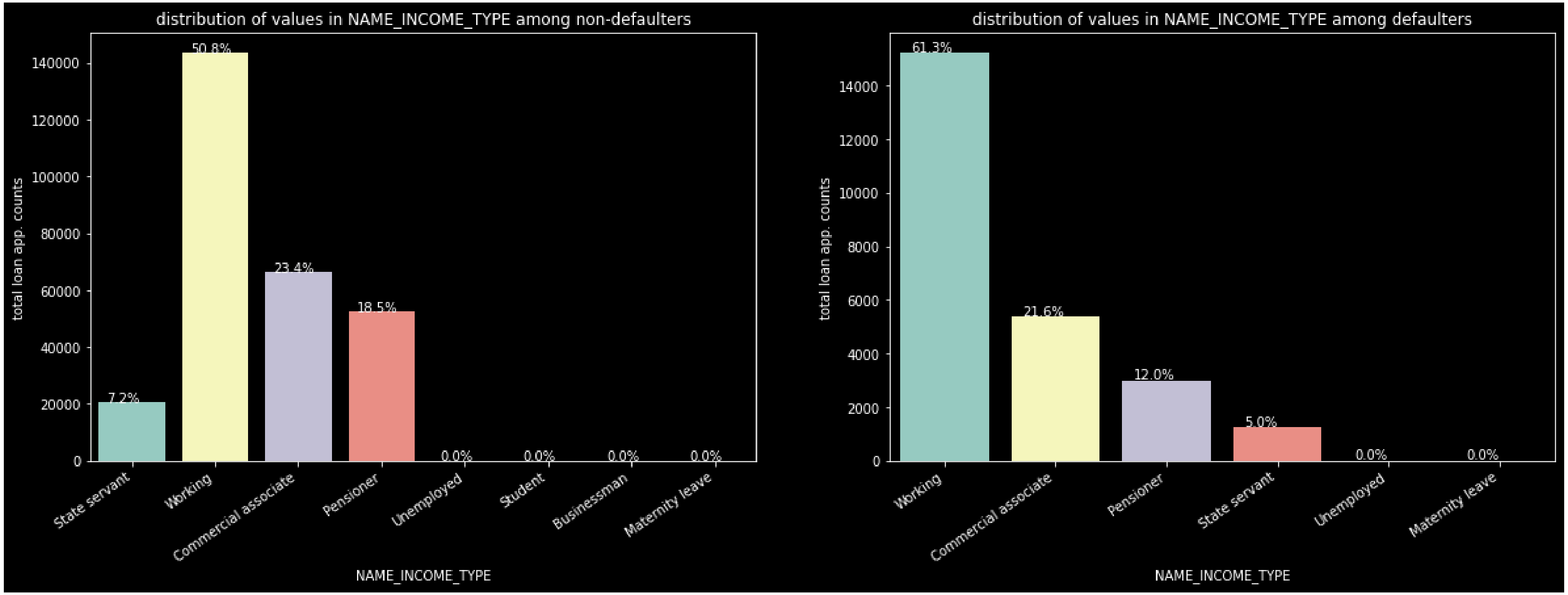
We can see that Female contribute 67% to the non-defaulters while 57% to the defaulters. We can conclude that we see more female applying for loans than males and hence the more number of female defaulters as well.
But the rate of defaulting of FEMALE is much lower compared to their MALE counterparts.

In [82]:
```
unicat('FLAG_OWN_CAR')
```



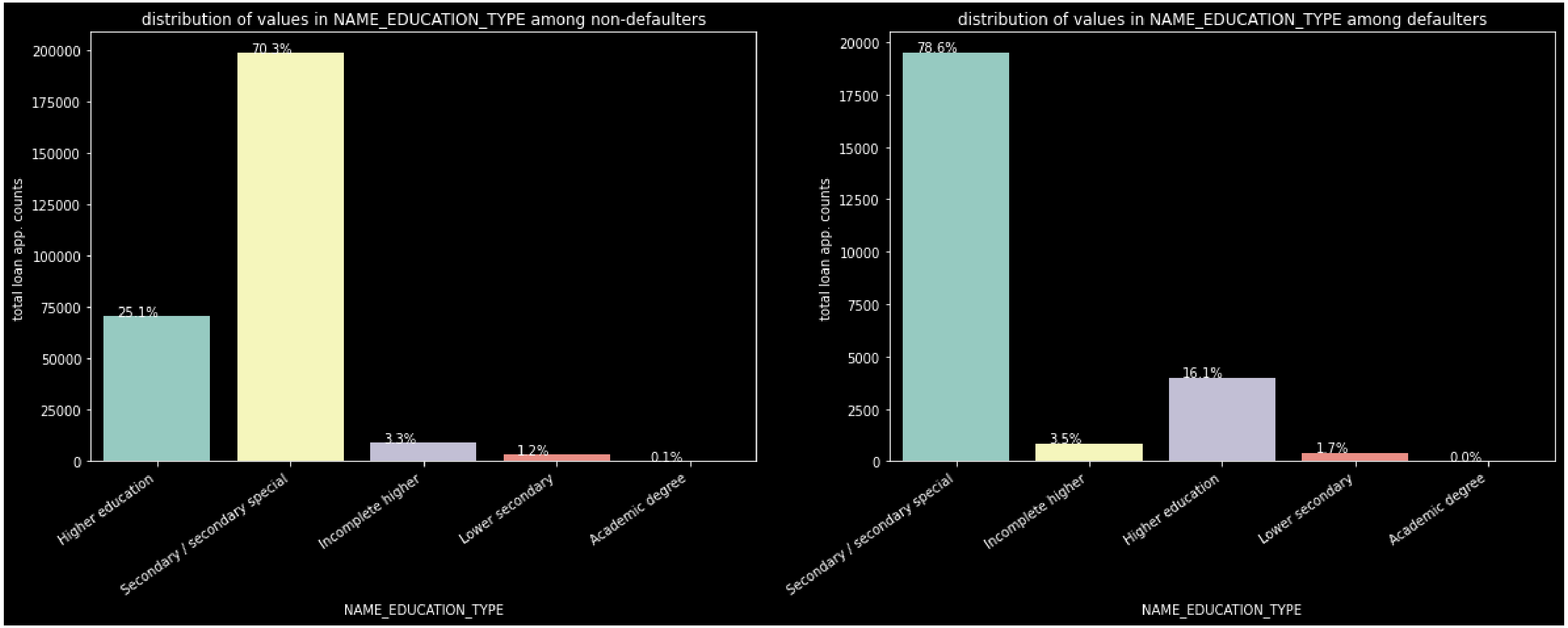
We can see that people with cars contribute 65.7% to the non-defaulters while 69.5% to the defaulters. We can conclude that While people who have no car default more often, the reason could be there are simply more people without cars
Looking at the percentages in both the charts, we can conclude that the rate of default of people having car is low compared to people who don't.

In [83]:
```
unicat('NAME_INCOME_TYPE')
```



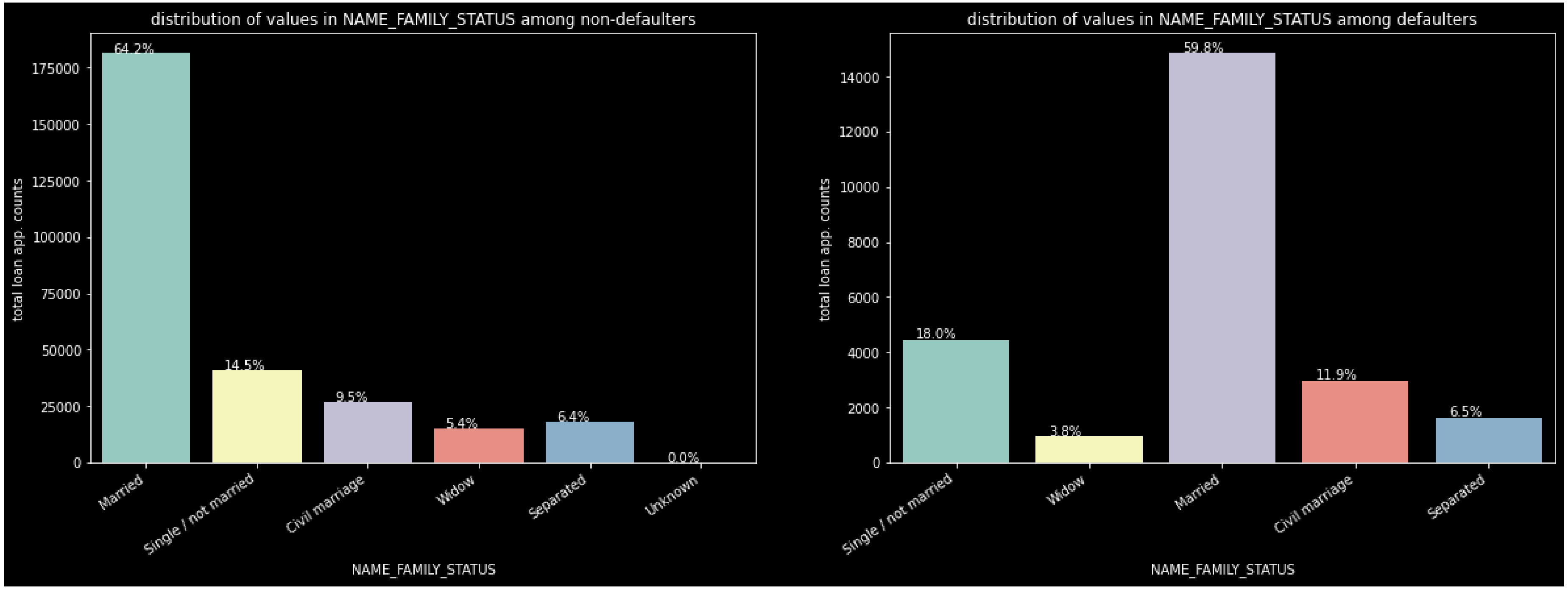
We can notice that the students don't default. The reason could be they are not required to pay during the time they are students.
We can also see that the BusinessMen never default.
Most of the loans are distributed to working class people
We also see that working class people contribute 51% to non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

In [84]:
```
unicat('NAME_EDUCATION_TYPE')
```



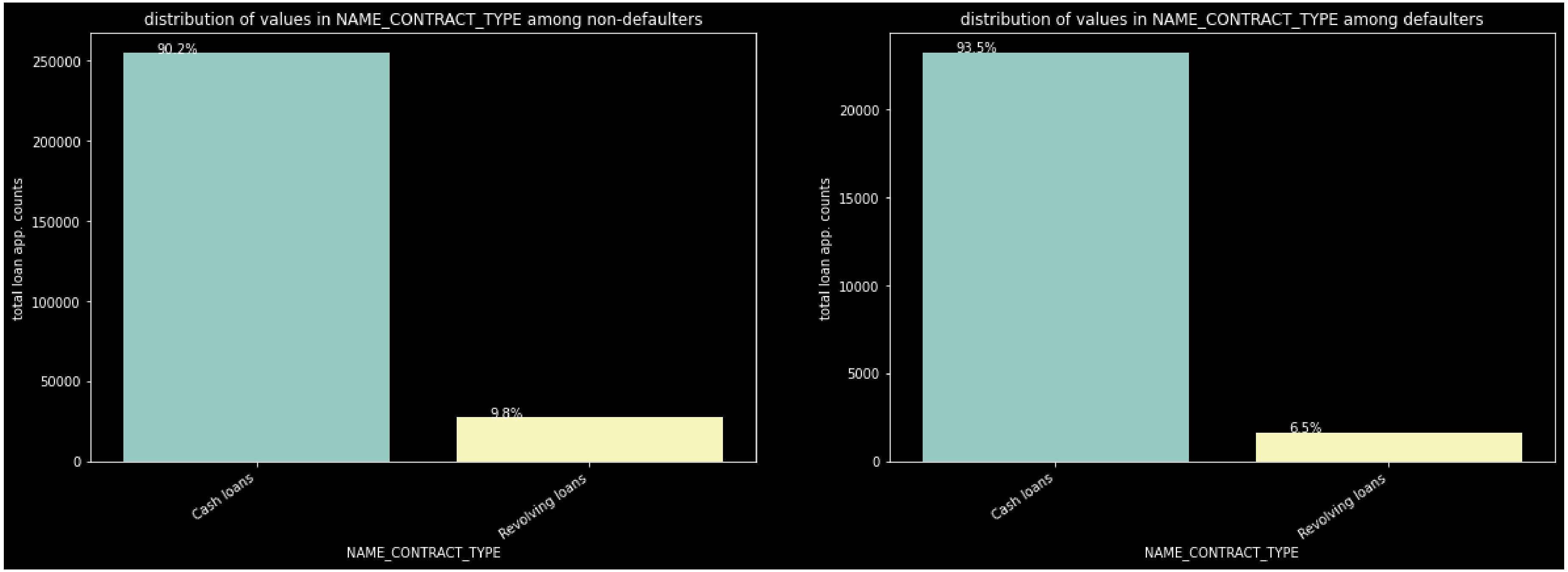
Almost all of the Education categories are equally likely to default except for the higher educated ones who are less likely to default and secondary educated people are more likely to default

In [85]:
```
unicat('NAME_FAMILY_STATUS')
```

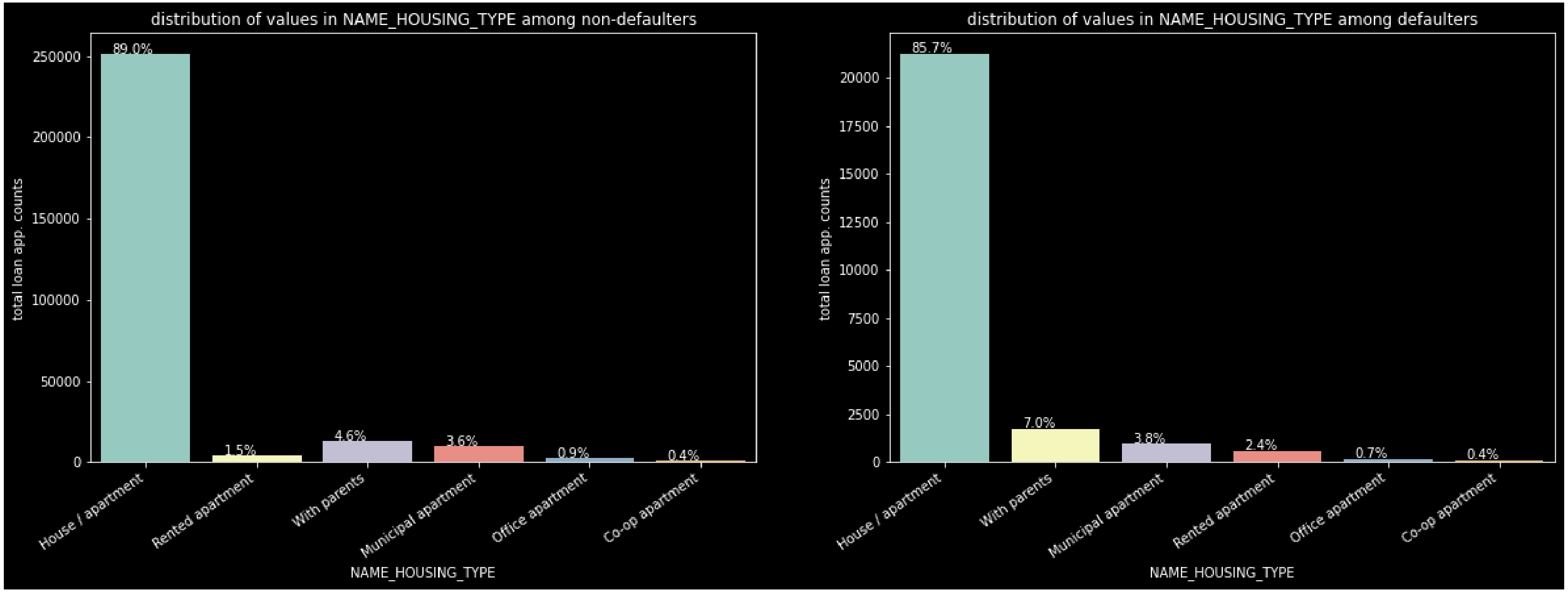


Married people tend to apply for more loans comparatively.
But from the graph we see that Single/non Married people contribute 14.5% to Non Defaulters and 18% to the defaulters. So there is more risk associated with them.

In [86]:
```
unicat('NAME_CONTRACT_TYPE')
```



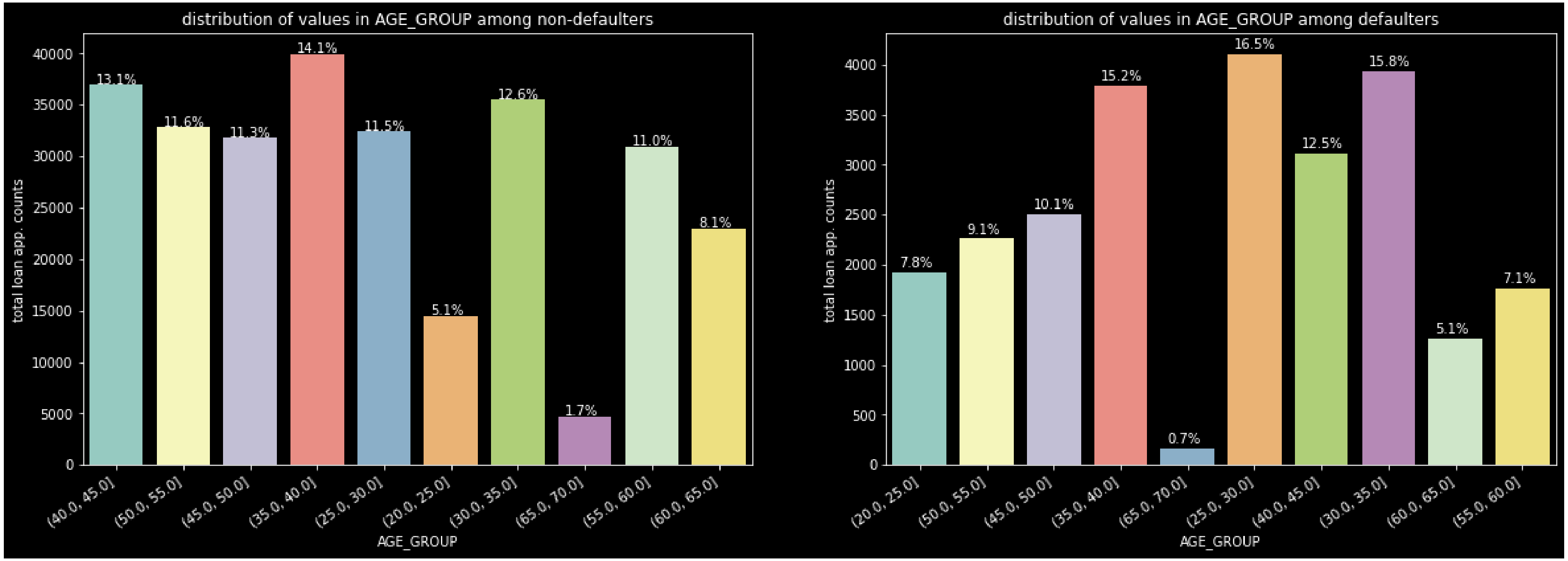
In [87]:
```
unicat('NAME_HOUSING_TYPE')
```



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

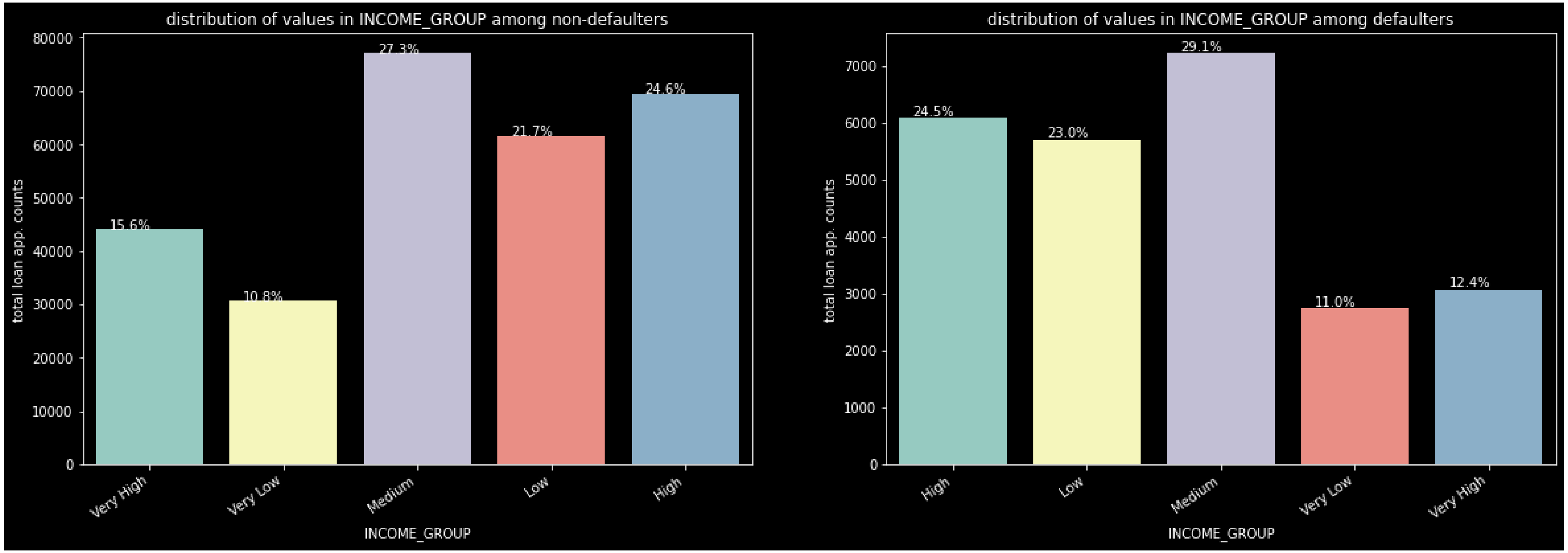
due to their parents living with them.

In [88]:
```
unicat('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 [89]:
```
unicat('INCOME_GROUP')
```

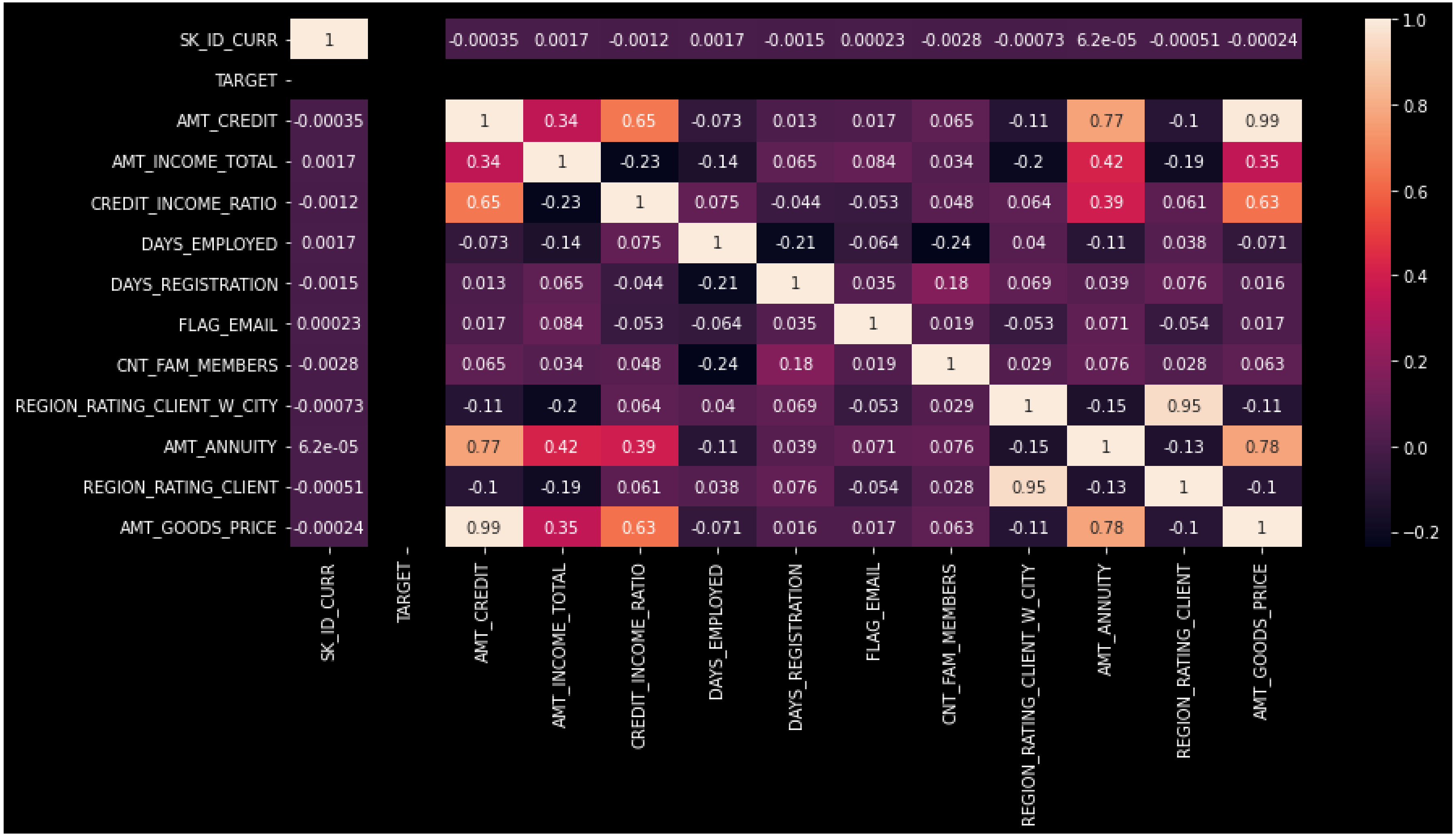


The Very High income group tend to default less often. They contribute 12.4% to the total number of defaulters, while they contribute 15.6% to the Non-Defaulters.

Getting the top 10 correlations of the selected columns in both the datasets

In [90]:
```
# newapp0
corr_map0 = newapp0.corr()
fig = plt.figure(figsize=(14,6))
sns.heatmap(corr_map0, annot=True)
```

Out[90]: <AxesSubplot:>



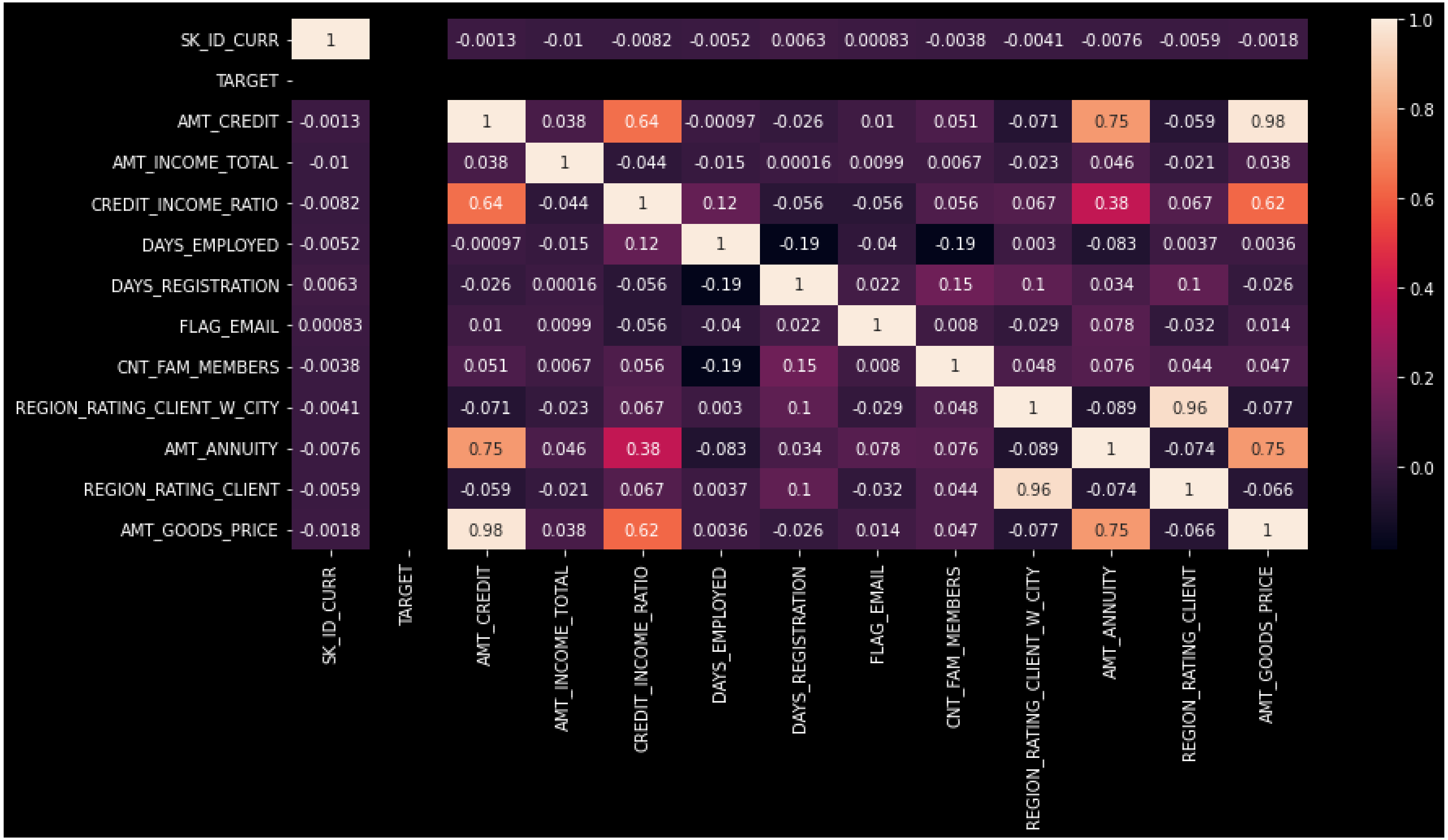
In [91]:
```
corr_df = corr_map0.where(np.triu(np.ones(corr_map0.shape), k=1).astype(np.bool)).unstack().reset_index()
corr_df.columns = ['Column1', 'Column2', 'Correlation']
corr_df.dropna(subset=['Correlation'], inplace=True)
corr_df['Abs Correlation'] = abs(corr_df['Correlation'])
corr_df = corr_df.sort_values(by='Abs Correlation', ascending=False)
corr_df.head(10)                    # Getting top 10 correlation in newapp0 (non-defaulters)
```

Out[91]:

	Column1	Column2	Correlation	Abs Correlation
158	AMT_GOODS_PRICE	AMT_CREDIT	0.987024	0.987024
152	REGION_RATING_CLIENT	REGION_RATING_CLIENT_W_CITY	0.950148	0.950148
166	AMT_GOODS_PRICE	AMT_ANNUITY	0.776421	0.776421
132	AMT_ANNUITY	AMT_CREDIT	0.771296	0.771296
54	CREDIT_INCOME_RATIO	AMT_CREDIT	0.648589	0.648589
160	AMT_GOODS_PRICE	CREDIT_INCOME_RATIO	0.628732	0.628732
133	AMT_ANNUITY	AMT_INCOME_TOTAL	0.418949	0.418949
134	AMT_ANNUITY	CREDIT_INCOME_RATIO	0.391498	0.391498
159	AMT_GOODS_PRICE	AMT_INCOME_TOTAL	0.349425	0.349425
41	AMT_INCOME_TOTAL	AMT_CREDIT	0.342801	0.342801

In [92]:
```
# newapp1
corr_map1 = newapp1.corr()
fig = plt.figure(figsize=(14,6))
sns.heatmap(corr_map1, annot=True)
```

Out[92]: <AxesSubplot:>



In [93]:
```
corr_df2 = corr_map1.where(np.triu(np.ones(corr_map1.shape), k=1).astype(np.bool)).unstack().reset_index()
corr_df2.columns = ['Column1', 'Column2', 'Correlation']
corr_df2.dropna(subset=['Correlation'], inplace=True)
corr_df2['Abs Correlation'] = abs(corr_df2['Correlation'])
corr_df2 = corr_df2.sort_values(by='Abs Correlation', ascending=False)
corr_df2.head(10)     # Getting top 10 correlations for defaulters
```

Out[93]:

	Column1	Column2	Correlation	Abs Correlation
158	AMT_GOODS_PRICE	AMT_CREDIT	0.982783	0.982783
152	REGION_RATING_CLIENT	REGION_RATING_CLIENT_W_CITY	0.956637	0.956637
166	AMT_GOODS_PRICE	AMT_ANNUITY	0.752295	0.752295
132	AMT_ANNUITY	AMT_CREDIT	0.752195	0.752195
54	CREDIT_INCOME_RATIO	AMT_CREDIT	0.639744	0.639744
160	AMT_GOODS_PRICE	CREDIT_INCOME_RATIO	0.623100	0.623100
134	AMT_ANNUITY	CREDIT_INCOME_RATIO	0.381298	0.381298
83	DAYS_REGISTRATION	DAYS_EMPLOYED	-0.188929	0.188929
109	CNT_FAM_MEMBERS	DAYS_EMPLOYED	-0.186561	0.186561
110	CNT_FAM_MEMBERS	DAYS_REGISTRATION	0.145828	0.145828

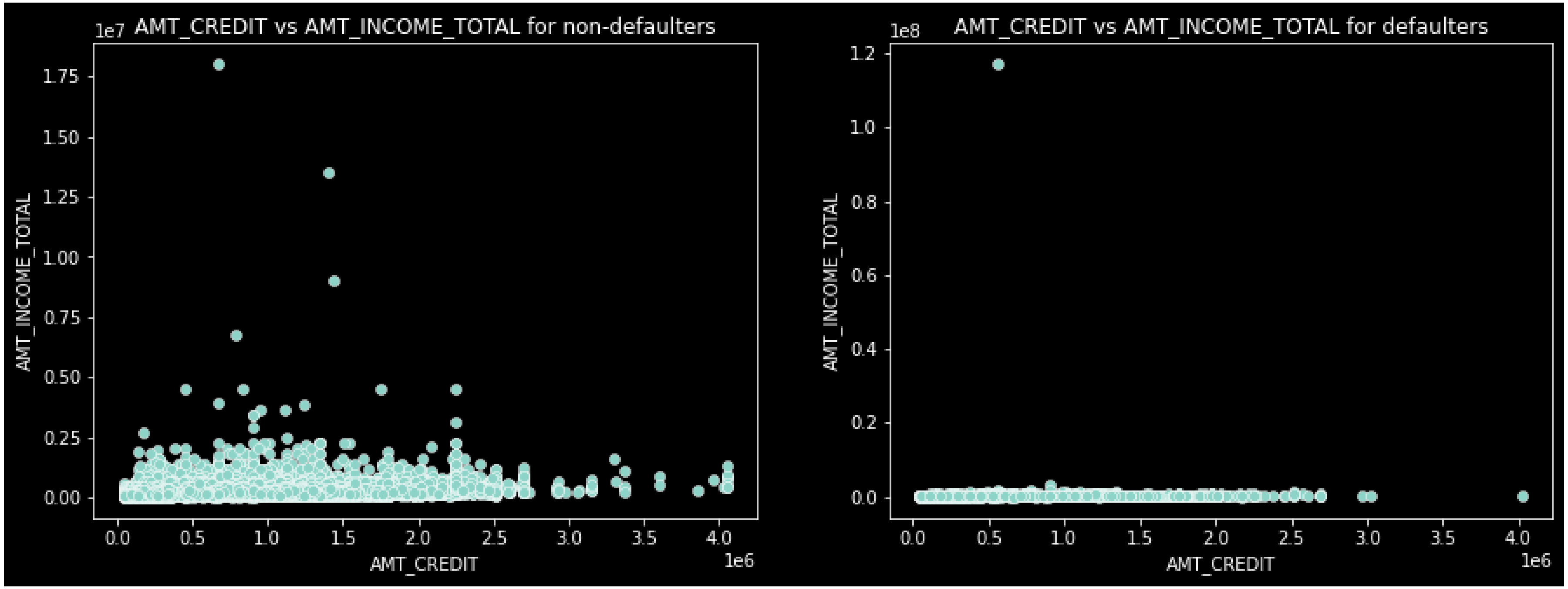
Bivariate Numerical Variable analysis

In [94]:
```
newapp0.select_dtypes(['int', 'float']).columns        # Getting numerical columns
```

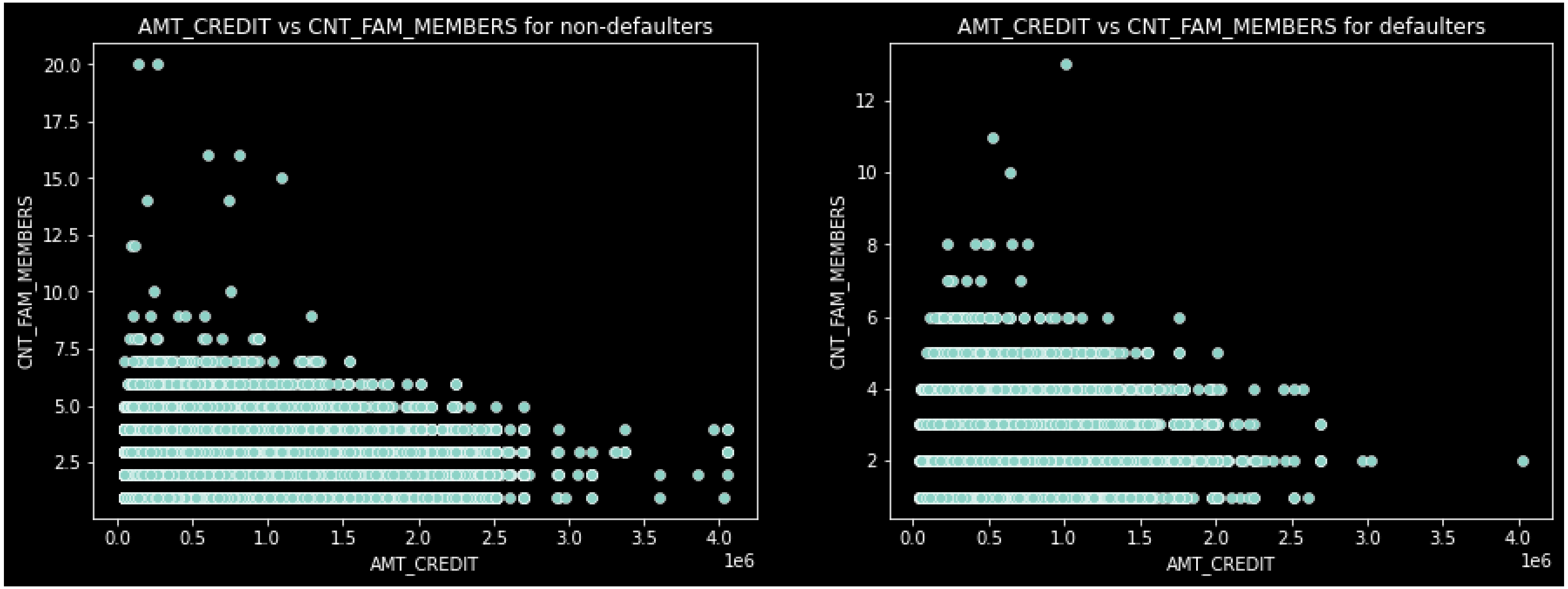
Out[94]:
```
Index(['AMT_CREDIT', 'AMT_INCOME_TOTAL', 'CREDIT_INCOME_RATIO',
       'DAYS_REGISTRATION', 'CNT_FAM_MEMBERS', 'AMT_ANNUITY',
       'AMT_GOODS_PRICE'],
      dtype='object')
```

In [95]:
```
def binumvar(var1, var2):
    plt.style.use('dark_background')
    fig, (ax1, ax2) = plt.subplots(1,2, figsize=(15,5))

    sns.scatterplot(x=var1, y=var2, data=newapp0, ax=ax1)
    ax1.set_title(f'{var1} vs {var2} for non-defaulters')

    sns.scatterplot(x=var1, y=var2, data=newapp1, ax=ax2)
    ax2.set_title(f'{var1} vs {var2} for defaulters')

    plt.show()
```

In [96]:
```
binumvar('AMT_CREDIT','AMT_INCOME_TOTAL')
```

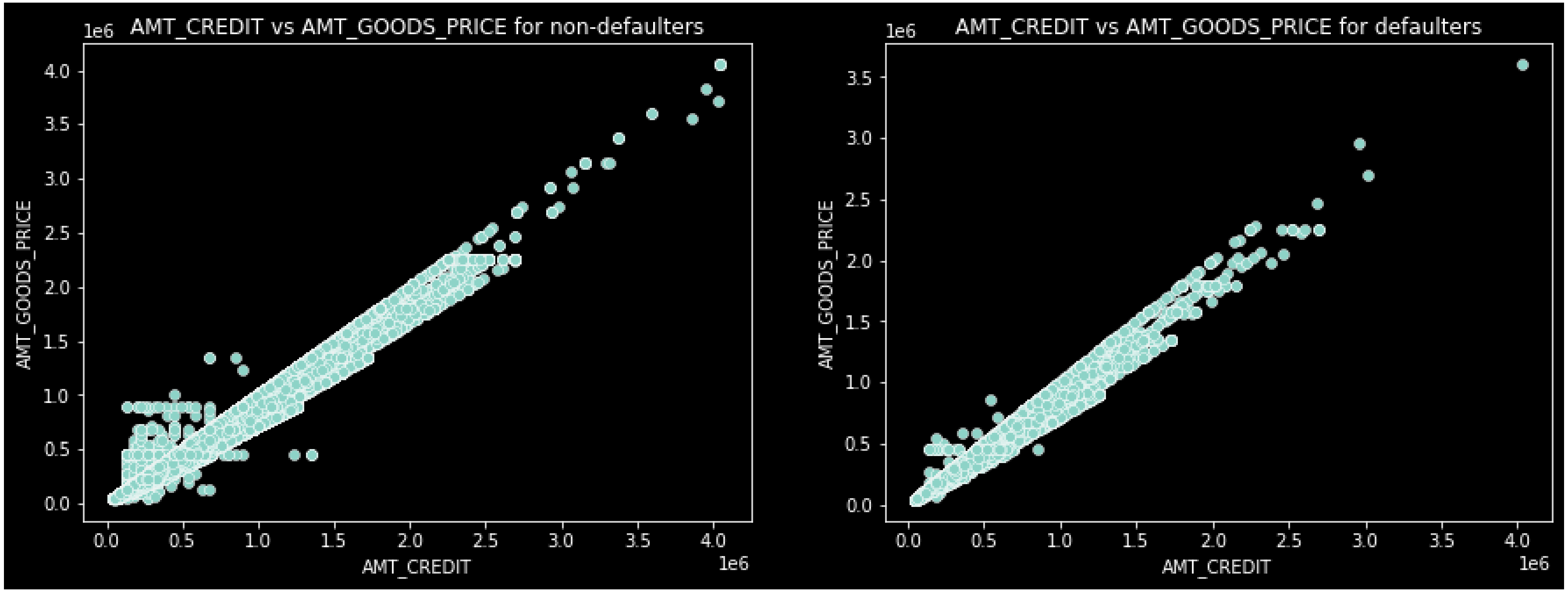


In [97]:
```
binumvar('AMT_CREDIT', 'CNT_FAM_MEMBERS')
```



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 [98]:
```
binumvar('AMT_CREDIT', 'AMT_GOODS_PRICE')
```



Data Analysis on previous application data

In [99]:
```
pre_app.head()
```

Out[99]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_APPLICATION	AMT_CREDIT	WEEKDAY_APPR_PROCESS_START	HOUR_APPR_PROCE
0	2030495	271877	Consumer loans	17145.0	17145.0	SATURDAY	
1	2802425	108129	Cash loans	607500.0	679671.0	THURSDAY	
2	2523466	122040	Cash loans	112500.0	136444.5	TUESDAY	
3	2819243	176158	Cash loans	450000.0	470790.0	MONDAY	
4	1784265	202054	Cash loans	337500.0	404055.0	THURSDAY	

In [100]:
```
# Deleting all the columns with null-value > 5% using loc function
pre_app = pre_app.loc[:, pre_app.isnull().mean()<=0.05]
pre_app.shape
```

Out[100]: (1670214, 23)

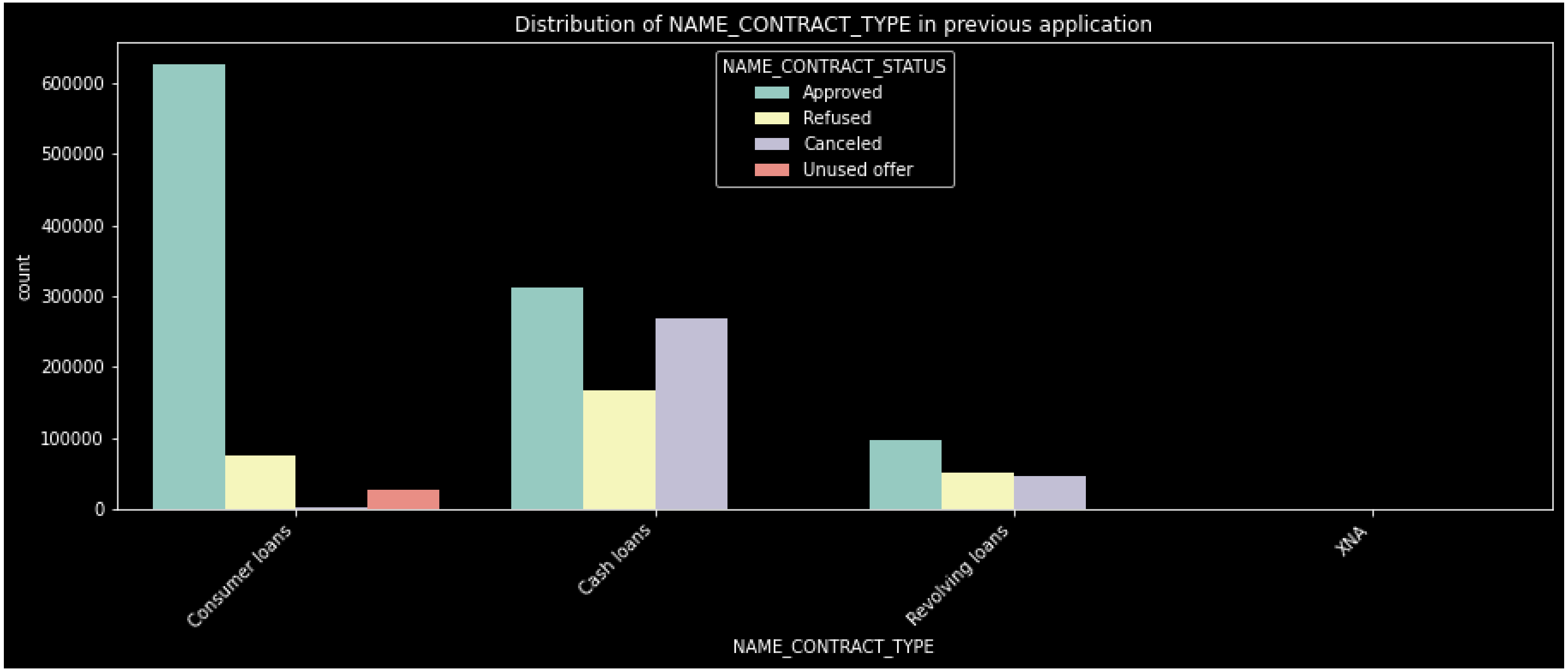
Univariate Categorical Analysis

In [101]:
```
def univar2(var):
    plt.style.use('dark_background')
    plt.figure(figsize=(15,5))
    sns.countplot(x=var, data=pre_app, hue='NAME_CONTRACT_STATUS')
    plt.title(f'Distribution of {var} in previous application')
    plt.xticks(rotation=45, ha='right')
    plt.show()
```

In [102]:
```
pre_app.select_dtypes(['object']).columns                # Selecting categorical columns
```

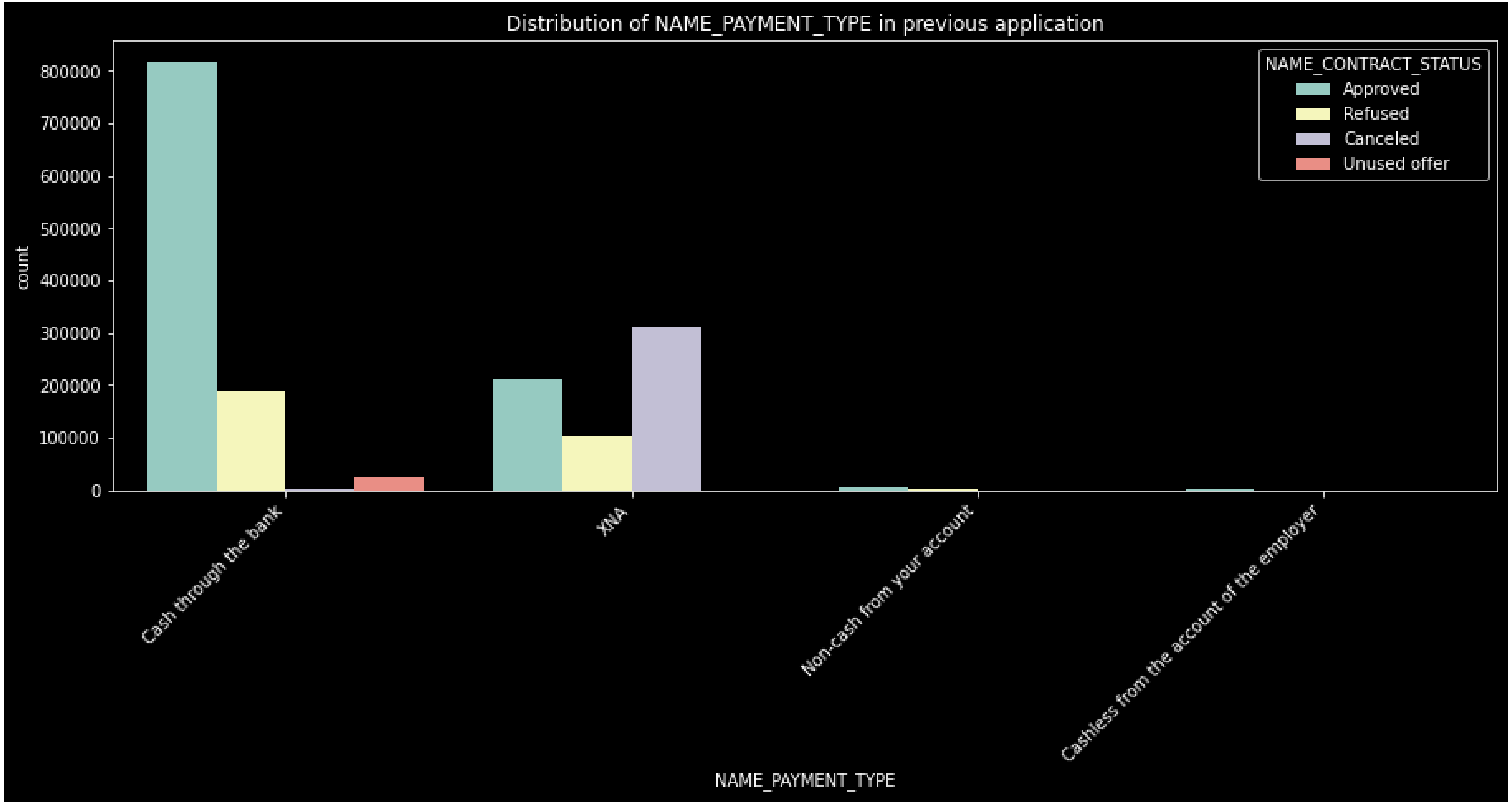
Out[102]:
```
Index(['NAME_CONTRACT_TYPE', 'WEEKDAY_APPR_PROCESS_START',
       'FLAG_LAST_APPL_PER_CONTRACT', 'NAME_CASH_LOAN_PURPOSE',
       'NAME_CONTRACT_STATUS', 'NAME_PAYMENT_TYPE', 'CODE_REJECT_REASON',
       'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO',
       'NAME_PRODUCT_TYPE', 'CHANNEL_TYPE', 'NAME_SELLER_INDUSTRY',
       'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION'],
      dtype='object')
```

In [103]:
```
univar2('NAME_CONTRACT_TYPE')
```



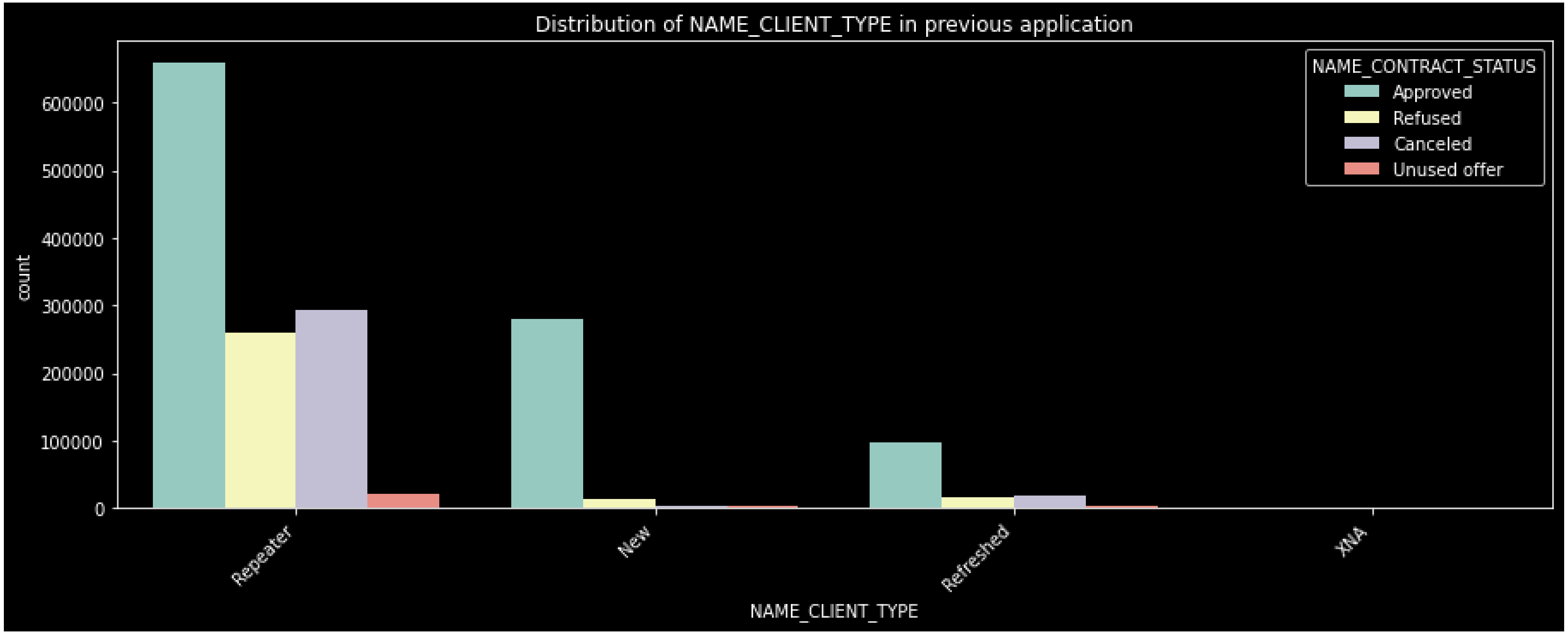
From the above chart, we can infer that, most of the applications are for 'Cash loan' and 'Consumer loan'. Although the cash loans are refused more often than others.

In [104]:
```
univar2('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.

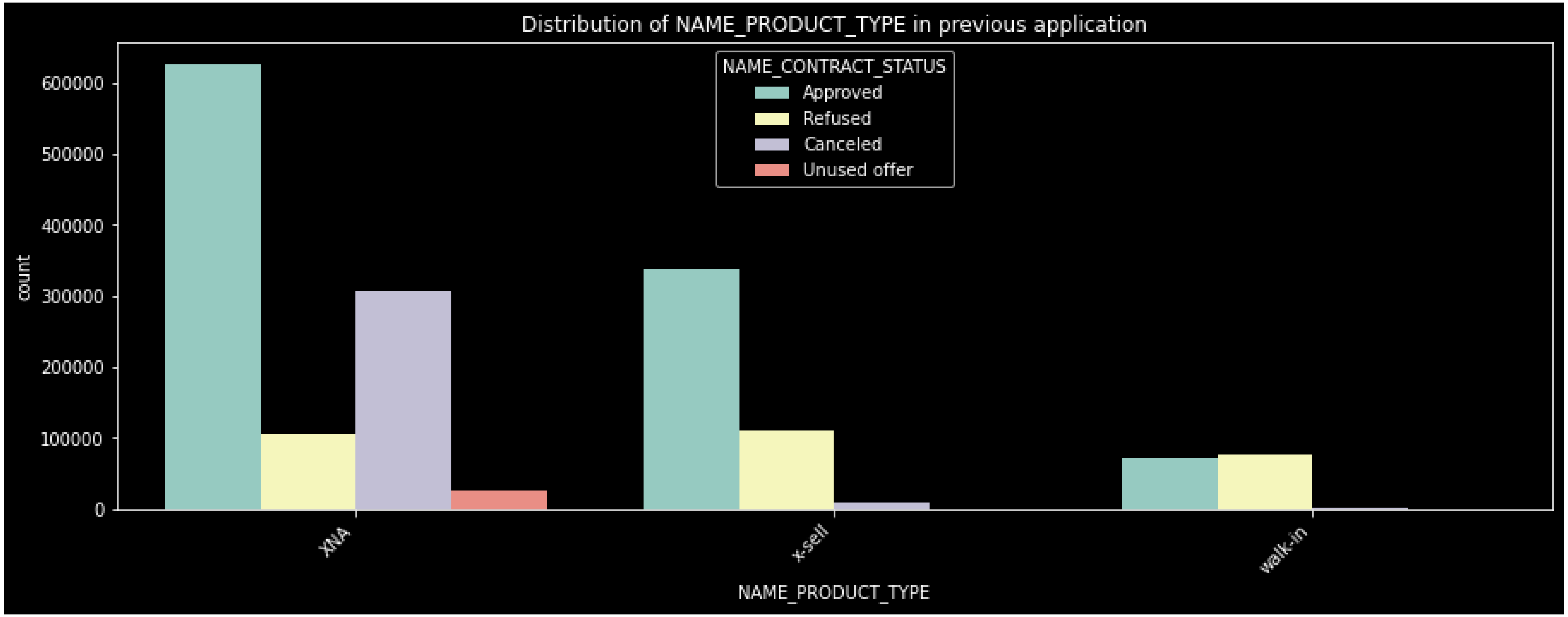
In [105]:
```
univar2('NAME_CLIENT_TYPE')
```



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.

In [106]:
```
#univar2('NAME_GOODS_CATEGORY')
```

In [107]:
```
univar2('NAME_PRODUCT_TYPE')
```



Checking the correlation in the Previous application dataset

In [108]:
```
pre_corr = pre_app.corr()
plt.figure(figsize=(14,6))
sns.heatmap(pre_corr, annot=True)
```

Out[108]: <AxesSubplot:>

In [109]:
```
# Getting top 10 correlation on previous_application dataset
corr_df2 = pre_corr.where(np.triu(np.ones(pre_corr.shape),k=1).astype(np.bool)).unstack().reset_index()
corr_df2.columns=['Column1','Column2','Correlation']
corr_df2.dropna(subset=['Correlation'],inplace=True)
corr_df2['Abs_Correlation']=corr_df2['Correlation'].abs()
corr_df2 = corr_df2.sort_values(by=['Abs_Correlation'], ascending=False)
corr_df2.head(10)
```

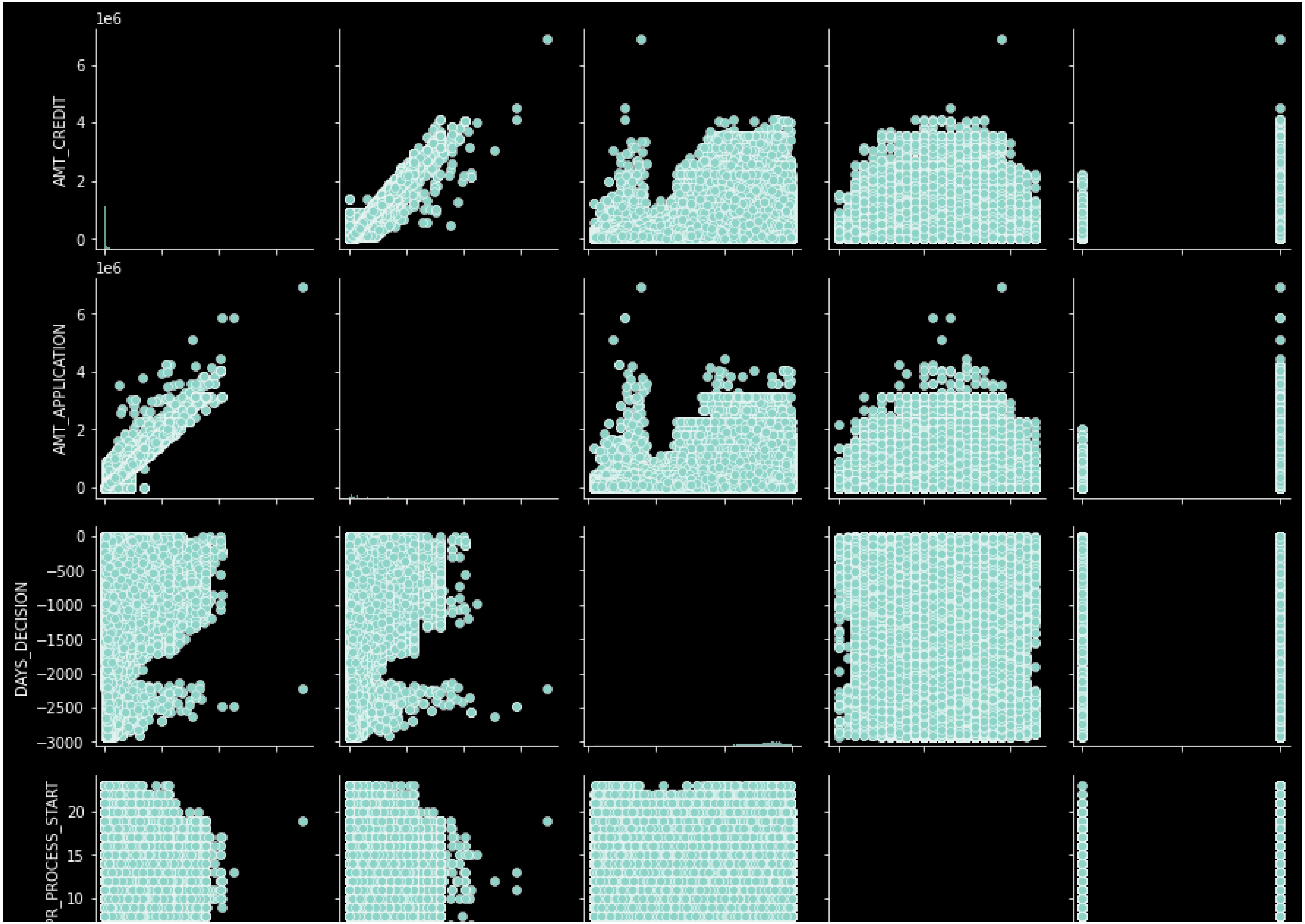
Out[109]:

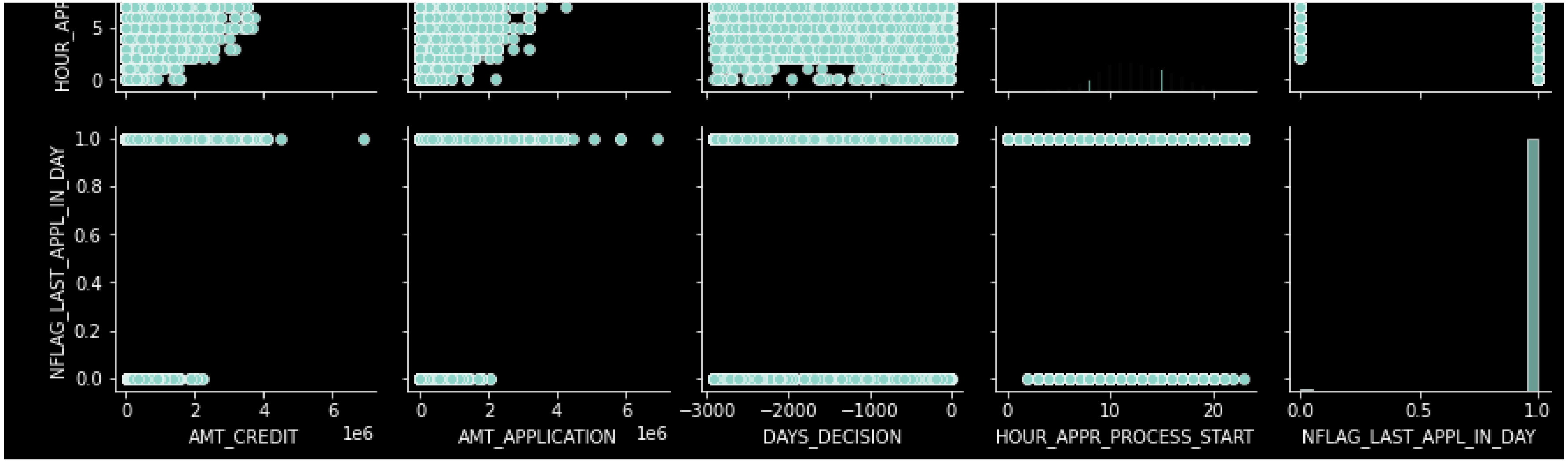
	Column1	Column2	Correlation	Abs_Correlation
26	AMT_CREDIT	AMT_APPLICATION	0.975824	0.975824
51	DAYS_DECISION	AMT_CREDIT	0.133763	0.133763
50	DAYS_DECISION	AMT_APPLICATION	0.133660	0.133660
52	DAYS_DECISION	HOUR_APPR_PROCESS_START	-0.039962	0.039962
43	NFLAG_LAST_APPL_IN_DAY	AMT_CREDIT	-0.025179	0.025179
35	HOUR_APPR_PROCESS_START	AMT_CREDIT	-0.021039	0.021039
48	DAYS_DECISION	SK_ID_PREV	0.019100	0.019100
62	SELLERPLACE_AREA	DAYS_DECISION	-0.018382	0.018382
53	DAYS_DECISION	NFLAG_LAST_APPL_IN_DAY	0.016555	0.016555
60	SELLERPLACE_AREA	HOUR_APPR_PROCESS_START	0.015671	0.015671

Univariate numerical variable analysis in previous_application data

In [110]:
```
# Plotting the relationships between highly correlated numerical columns  # A great thought process indeed
sns.pairplot(pre_app[['AMT_CREDIT','AMT_APPLICATION','DAYS_DECISION','HOUR_APPR_PROCESS_START','NFLAG_LAST_APPL_IN_DAY']
```

Out[110]: <seaborn.axisgrid.PairGrid at 0x2159b7d32e0>





AMT credited on the previous appliaction is heavily influened by the AMT the client requested in his previous application.

Categorical vs Numerical variables analysis on previous application dataset

In [111]:
```
def catnum(cat, num):
    plt.style.use('ggplot')
    sns.despine
    fig, ax = plt.subplots(1,1, figsize=(10,8))
    sns.boxenplot(x=cat, y=num, data=pre_app, ax=ax)

    ax.set_title(f'{cat} vs {num}')
    ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha='right')
    plt.show()
```

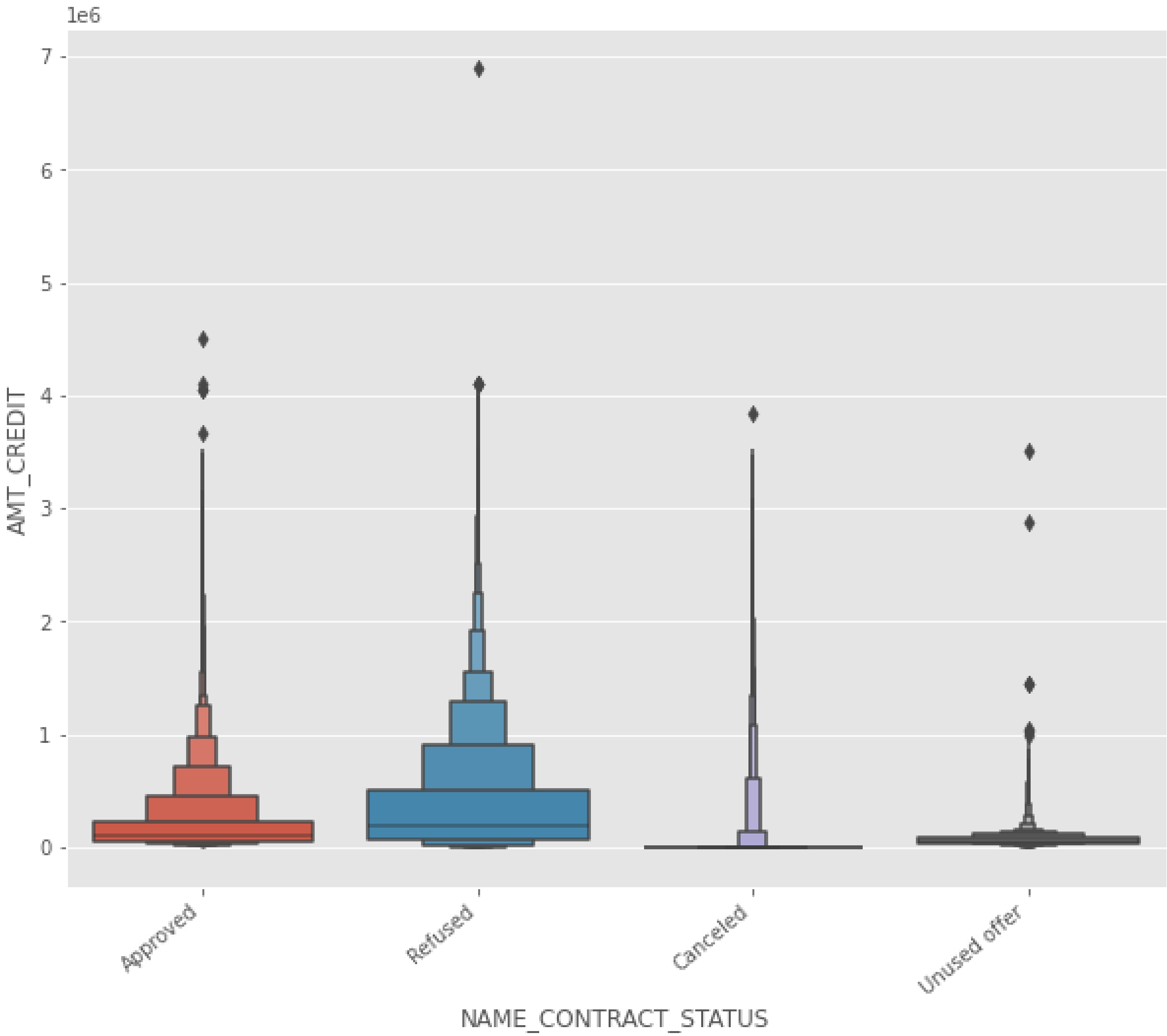
In [112]:
```
pre_app.select_dtypes(['int64', 'float64']).columns
```

Out[112]:
```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'AMT_APPLICATION', 'AMT_CREDIT',
       'HOUR_APPR_PROCESS_START', 'NFLAG_LAST_APPL_IN_DAY', 'DAYS_DECISION',
       'SELLERPLACE_AREA'],
      dtype='object')
```

In [113]:
```
pre_app.select_dtypes(['object']).columns
```

Out[113]:
```
Index(['NAME_CONTRACT_TYPE', 'WEEKDAY_APPR_PROCESS_START',
       'FLAG_LAST_APPL_PER_CONTRACT', 'NAME_CASH_LOAN_PURPOSE',
       'NAME_CONTRACT_STATUS', 'NAME_PAYMENT_TYPE', 'CODE_REJECT_REASON',
       'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO',
       'NAME_PRODUCT_TYPE', 'CHANNEL_TYPE', 'NAME_SELLER_INDUSTRY',
       'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION'],
      dtype='object')
```

In [114]:
```
#by-varient analysis of Contract status and Final credit amount disbursed to the customer previously, after approval
catnum('NAME_CONTRACT_STATUS', 'AMT_CREDIT')
```



Now merging the files and analyzing the data

In [115]:
```
# Merging (not concatenating) the files to do some analysis using pd.merge() function
merged_data = pd.merge(left=new_app_data, right=pre_app, how='left', on=['SK_ID_CURR'])
```

In [116]:
```
merged_data.shape
```

Out[116]:
```
(1430100, 48)
```

In [117]:
```
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1430100 entries, 0 to 1430099
Data columns (total 48 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  object 
 6   AGE_GROUP                      1430096 non-null  object 
 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  CNT_FAM_MEMBERS                1430100 non-null  float64
 18  REGION_RATING_CLIENT_W_CITY    1430100 non-null  int64  
 19  ORGANIZATION_TYPE              1430100 non-null  object 
 20  SOCIAL_CIRCLE_30_DAYS_DEF_PERC  0 non-null       object 
 21  SOCIAL_CIRCLE_60_DAYS_DEF_PERC  0 non-null       object 
 22  NAME_CONTRACT_TYPE_x           1430100 non-null  object 
 23  AMT_ANNUITY                    1430100 non-null  float64
 24  REGION_RATING_CLIENT           1430100 non-null  int64  
 25  AMT_GOODS_PRICE                1430100 non-null  float64
 26  SK_ID_PREV                     1413646 non-null  float64
 27  NAME_CONTRACT_TYPE_y           1413646 non-null  object 
 28  AMT_APPLICATION                1413646 non-null  float64
 29  AMT_CREDIT_y                   1413646 non-null  float64
 30  WEEKDAY_APPR_PROCESS_START     1413646 non-null  object 
 31  HOUR_APPR_PROCESS_START        1413646 non-null  float64
 32  FLAG_LAST_APPL_PER_CONTRACT    1413646 non-null  object 
 33  NFLAG_LAST_APPL_IN_DAY         1413646 non-null  float64
```

```
 34  NAME_CASH_LOAN_PURPOSE          1413646 non-null  object
 35  NAME_CONTRACT_STATUS            1413646 non-null  object
 36  DAYS_DECISION                   1413646 non-null  float64
 37  NAME_PAYMENT_TYPE               1413646 non-null  object
 38  CODE_REJECT_REASON              1413646 non-null  object
 39  NAME_CLIENT_TYPE                1413646 non-null  object
 40  NAME_GOODS_CATEGORY             1413646 non-null  object
 41  NAME_PORTFOLIO                  1413646 non-null  object
 42  NAME_PRODUCT_TYPE               1413646 non-null  object
 43  CHANNEL_TYPE                    1413646 non-null  object
 44  SELLERPLACE_AREA                1413646 non-null  float64
 45  NAME_SELLER_INDUSTRY            1413646 non-null  object
 46  NAME_YIELD_GROUP                1413646 non-null  object
 47  PRODUCT_COMBINATION             1413646 non-null  object
dtypes: float64(14), int64(6), object(28)
memory usage: 534.6+ MB
```

In [118]:
```
a = merged_data.pivot_table(values='SK_ID_CURR',
                      index='FLAG_OWN_CAR',
                      columns='NAME_CONTRACT_STATUS',
                      aggfunc='count')
```

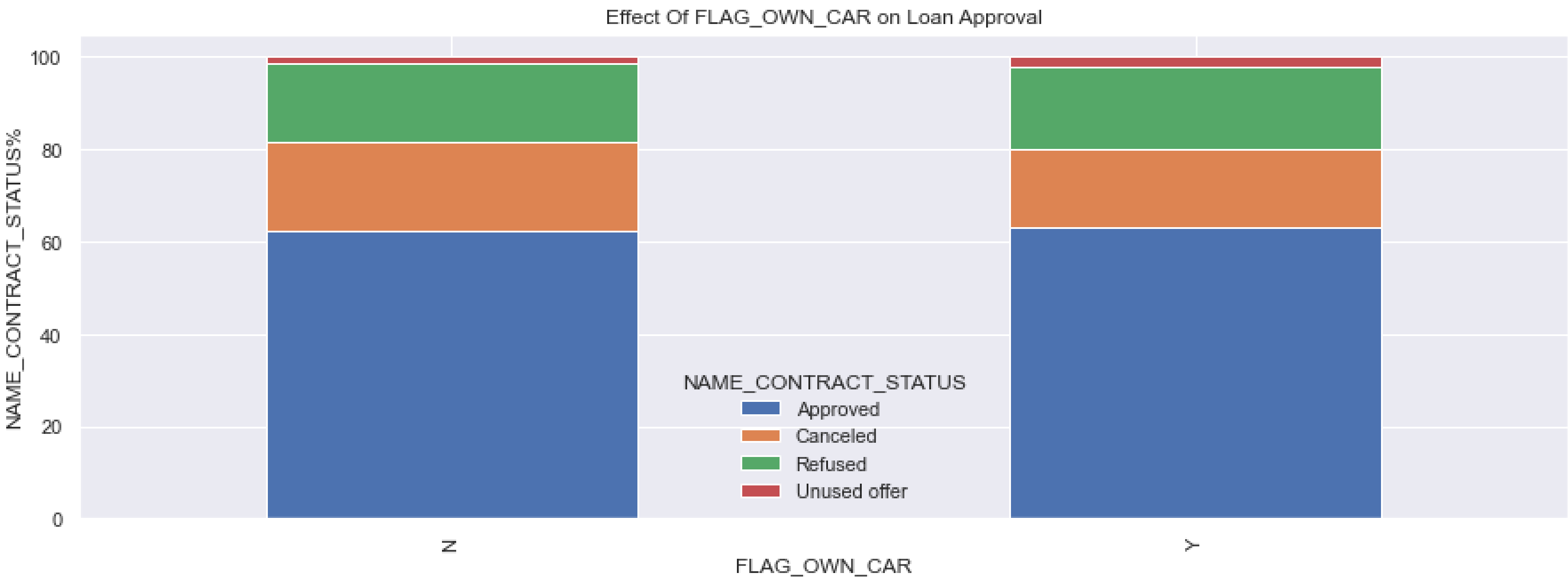
In [119]:
```
a.div(a.sum(axis=1),axis='rows')*100
```

Out[119]:

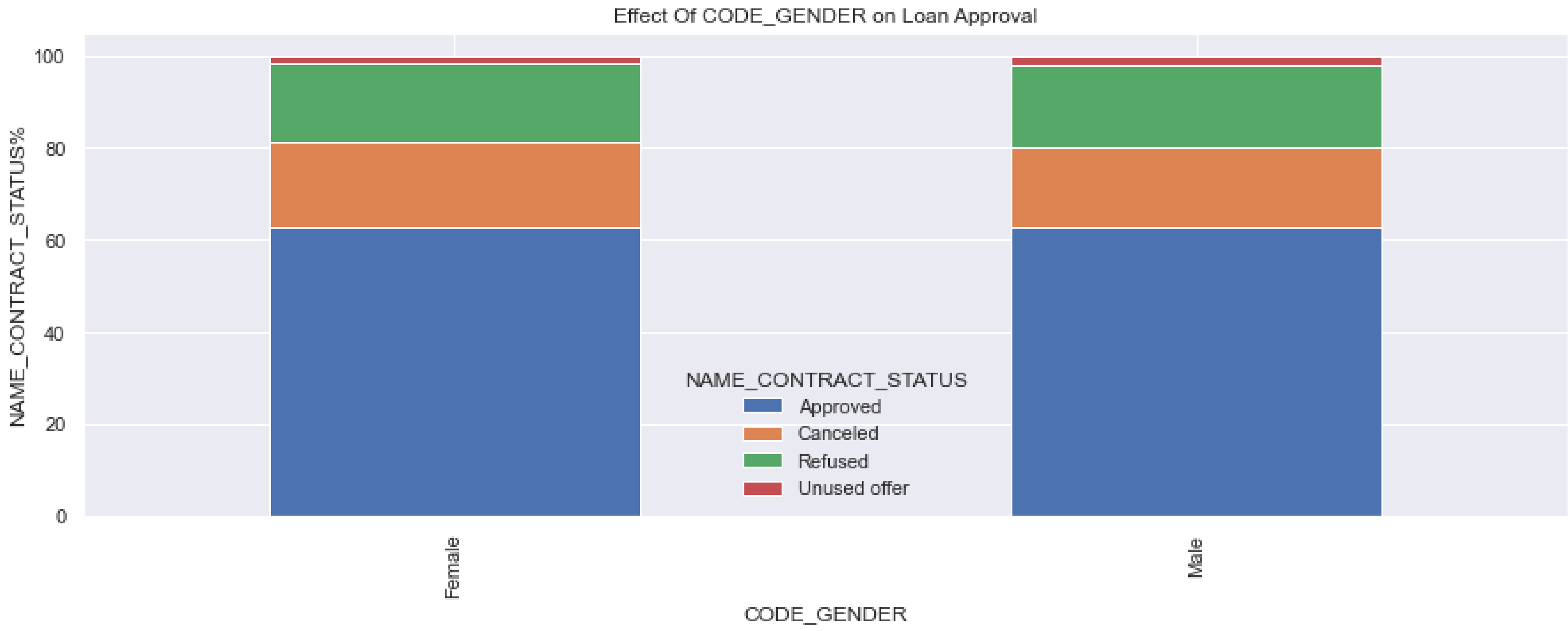
NAME_CONTRACT_STATUS	Approved	Canceled	Refused	Unused offer
FLAG_OWN_CAR				
N	62.412194	19.158254	17.103507	1.326046
Y	63.207268	16.766667	17.855232	2.170834

In [120]:
```
def merge_plot(Varx, Vary):
    # 100% bar chart
    plt.style.use('seaborn-darkgrid')
    sns.despine
    NewDat = merged_data.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()
```

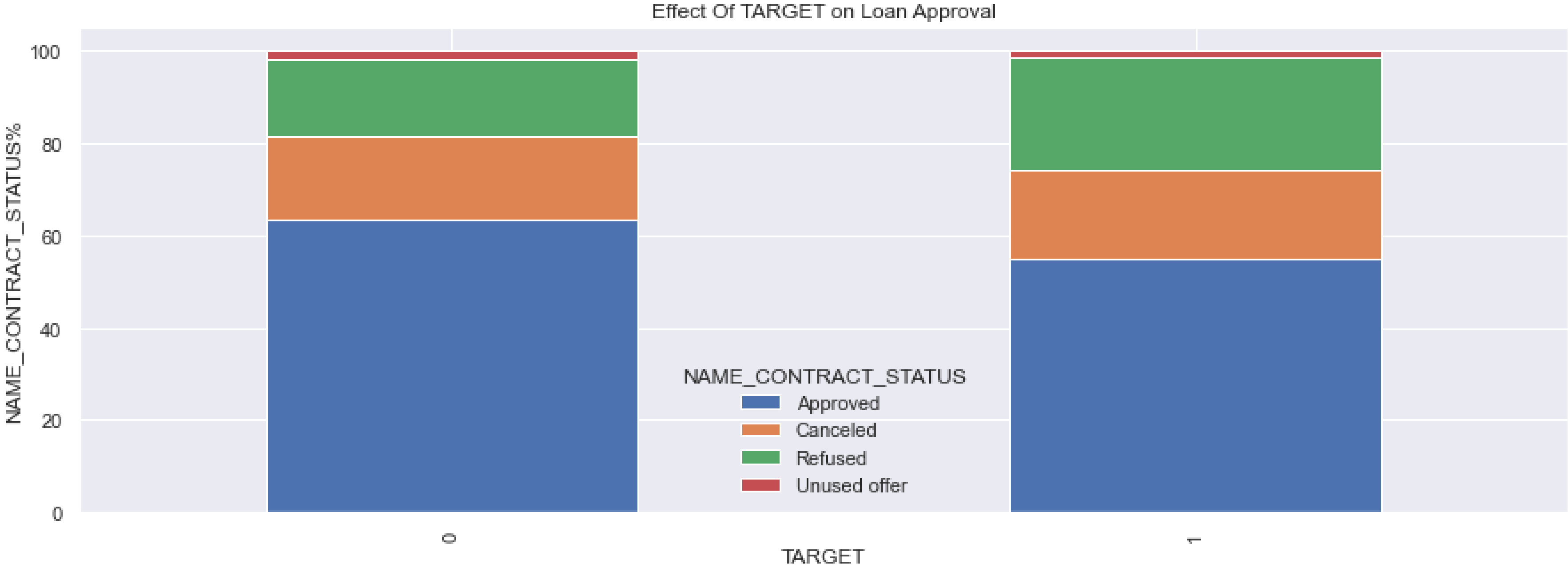
In [121]:
```
merge_plot('FLAG_OWN_CAR', 'NAME_CONTRACT_STATUS')
```



In [122]:
```
merge_plot('CODE_GENDER', 'NAME_CONTRACT_STATUS')
```



In [123]:
```
merge_plot('TARGET', 'NAME_CONTRACT_STATUS')
```



In above 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.