In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings ("ignore")
pd.set_option("display.max_rows",1000)
sns.set_theme(style="whitegrid")
```

1.Reading application_data.csv

In [2]:

```
app=pd.read_csv("application_data.csv")
```

In [3]:

```
app.head(10)
```

Out[3]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_(
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100004	0	Revolving loans	M	Y	
3	100006	0	Cash loans	F	N	
4	100007	0	Cash loans	M	N	
5	100008	0	Cash loans	M	N	
6	100009	0	Cash loans	F	Y	
7	100010	0	Cash loans	M	Y	
8	100011	0	Cash loans	F	N	
9	100012	0	Revolving loans	M	N	

10 rows × 122 columns

In [4]:

```
app.shape
```

Out[4]:

```
(307511, 122)
```

In [5]:

```
app.info()
```

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

In [6]:

```
list(app.columns)
```

Out[6]:

```
['SK_ID_CURR',
 'TARGET',
 'NAME_CONTRACT_TYPE',
 'CODE_GENDER',
 'FLAG_OWN_CAR',
 'FLAG_OWN_REALTY',
 'CNT_CHILDREN',
 'AMT_INCOME_TOTAL',
 'AMT_CREDIT',
 'AMT_ANNUITY',
 'AMT_GOODS_PRICE',
 'NAME_TYPE_SUITE',
 'NAME_INCOME_TYPE',
 'NAME_EDUCATION_TYPE',
 'NAME_FAMILY_STATUS',
 'NAME_HOUSING_TYPE',
 'REGION_POPULATION_RELATIVE',
 'DAYS_BIRTH',
```

In [7]:

```
app.dtypes
```

Out[7]:

```
SK_ID_CURR                       int64
TARGET                           int64
NAME_CONTRACT_TYPE              object
CODE_GENDER                     object
FLAG_OWN_CAR                    object
FLAG_OWN_REALTY                 object
CNT_CHILDREN                     int64
AMT_INCOME_TOTAL               float64
AMT_CREDIT                     float64
AMT_ANNUITY                    float64
AMT_GOODS_PRICE                float64
NAME_TYPE_SUITE                 object
NAME_INCOME_TYPE                object
NAME_EDUCATION_TYPE             object
NAME_FAMILY_STATUS              object
NAME_HOUSING_TYPE               object
REGION_POPULATION_RELATIVE     float64
DAYS_BIRTH                       int64
```

In [8]:

```
app.describe()
```

Out[8]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AM
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	307
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	27
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	14
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	24
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	258

8 rows × 106 columns

Comment on applicattion_data.csv

1.It has 122 columns, 307511 entries,

2. It has 65 coulmns with float datatype, 41 coulmns with integer data type, 16 coulmns with object datatype

2. Reading previous_application.csv

In [9]:

```
pre=pd.read_csv("previous_application.csv")
```

In [10]:

```
pre.head(10)
```

Out[10]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AI
0	2030495	271877	Consumer loans	1730.430	17145.0	
1	2802425	108129	Cash loans	25188.615	607500.0	
2	2523466	122040	Cash loans	15060.735	112500.0	
3	2819243	176158	Cash loans	47041.335	450000.0	
4	1784265	202054	Cash loans	31924.395	337500.0	
5	1383531	199383	Cash loans	23703.930	315000.0	
6	2315218	175704	Cash loans	NaN	0.0	
7	1656711	296299	Cash loans	NaN	0.0	
8	2367563	342292	Cash loans	NaN	0.0	
9	2579447	334349	Cash loans	NaN	0.0	

10 rows × 37 columns

In [11]:

```
pre.shape
```

Out[11]:

```
(1670214, 37)
```

In [12]:

```
list(pre.columns)
```

Out[12]:

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

In [13]:

```
pre.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
```

In [14]:

```
pre.describe()
```

Out[14]:

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DO
count	1.670214e+06	1.670214e+06	1.297979e+06	1.670214e+06	1.670213e+06	
mean	1.923089e+06	2.783572e+05	1.595512e+04	1.752339e+05	1.961140e+05	
std	5.325980e+05	1.028148e+05	1.478214e+04	2.927798e+05	3.185746e+05	
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00	0.000000e+00	
25%	1.461857e+06	1.893290e+05	6.321780e+03	1.872000e+04	2.416050e+04	
50%	1.923110e+06	2.787145e+05	1.125000e+04	7.104600e+04	8.054100e+04	
75%	2.384280e+06	3.675140e+05	2.065842e+04	1.803600e+05	2.164185e+05	
max	2.845382e+06	4.562550e+05	4.180581e+05	6.905160e+06	6.905160e+06	

8 rows × 21 columns

In [14]:

```
pre.dtypes
```

Out[14]:

```
SK_ID_PREV                      int64
SK_ID_CURR                      int64
NAME_CONTRACT_TYPE             object
AMT_ANNUITY                   float64
AMT_APPLICATION               float64
AMT_CREDIT                    float64
AMT_DOWN_PAYMENT              float64
AMT_GOODS_PRICE               float64
WEEKDAY_APPR_PROCESS_START     object
HOUR_APPR_PROCESS_START         int64
FLAG_LAST_APPL_PER_CONTRACT    object
NFLAG_LAST_APPL_IN_DAY          int64
RATE_DOWN_PAYMENT             float64
RATE_INTEREST_PRIMARY         float64
RATE_INTEREST_PRIVILEGED      float64
NAME_CASH_LOAN_PURPOSE         object
NAME_CONTRACT_STATUS           object
DAYS_DECISION                   int64
NAME_PAYMENT_TYPE              object
CODE_REJECT_REASON             object
NAME_TYPE_SUITE                object
NAME_CLIENT_TYPE               object
NAME_GOODS_CATEGORY            object
NAME_PORTFOLIO                 object
NAME_PRODUCT_TYPE              object
CHANNEL_TYPE                   object
SELLERPLACE_AREA                int64
NAME_SELLER_INDUSTRY           object
CNT_PAYMENT                   float64
NAME_YIELD_GROUP               object
PRODUCT_COMBINATION            object
DAYS_FIRST_DRAWING            float64
DAYS_FIRST_DUE                float64
DAYS_LAST_DUE_1ST_VERSION     float64
DAYS_LAST_DUE                 float64
DAYS_TERMINATION              float64
NFLAG_INSURED_ON_APPROVAL     float64
dtype: object
```

Comment on previous_application.csv

1. It is has 1670214 rows and 37 columns

2. It has 6 integer columns, 15 float columns, 16 string columns

3. Cleaning Data-Checking Missing Values(application_data.csv)

In [15]:

```
#checking missing values in application_data.csv
app.isnull().sum()
```

Out[15]:

```
SK_ID_CURR                     0
TARGET                         0
NAME_CONTRACT_TYPE             0
CODE_GENDER                    0
FLAG_OWN_CAR                   0
FLAG_OWN_REALTY                0
CNT_CHILDREN                   0
AMT_INCOME_TOTAL               0
AMT_CREDIT                     0
AMT_ANNUITY                   12
AMT_GOODS_PRICE              278
NAME_TYPE_SUITE             1292
NAME_INCOME_TYPE               0
NAME_EDUCATION_TYPE            0
NAME_FAMILY_STATUS             0
NAME_HOUSING_TYPE              0
REGION_POPULATION_RELATIVE     0
DAYS_BIRTH                     0
```

In [16]:

```
#show percentage of null values in each column in application_data.csv
app_null=(100*app.isnull().sum()/len(app)).round(2)
app_null
```

Out[16]:

```
SK_ID_CURR                    0.00
TARGET                        0.00
NAME_CONTRACT_TYPE            0.00
CODE_GENDER                   0.00
FLAG_OWN_CAR                  0.00
FLAG_OWN_REALTY               0.00
CNT_CHILDREN                  0.00
AMT_INCOME_TOTAL              0.00
AMT_CREDIT                    0.00
AMT_ANNUITY                   0.00
AMT_GOODS_PRICE               0.09
NAME_TYPE_SUITE               0.42
NAME_INCOME_TYPE              0.00
NAME_EDUCATION_TYPE           0.00
NAME_FAMILY_STATUS            0.00
NAME_HOUSING_TYPE             0.00
REGION_POPULATION_RELATIVE    0.00
DAYS_BIRTH                    0.00
```

In [17]:

```
#just show columns having null values is decreasing order
app_null=app_null[app_null.values>0].sort_values(ascending=False)
app_null
```

Out[17]:

```
COMMONAREA_AVG                 69.87
COMMONAREA_MODE                69.87
COMMONAREA_MEDI                69.87
NONLIVINGAPARTMENTS_MODE       69.43
NONLIVINGAPARTMENTS_AVG        69.43
NONLIVINGAPARTMENTS_MEDI       69.43
FONDKAPREMONT_MODE             68.39
LIVINGAPARTMENTS_MODE          68.35
LIVINGAPARTMENTS_MEDI          68.35
LIVINGAPARTMENTS_AVG           68.35
FLOORSMIN_MEDI                 67.85
FLOORSMIN_AVG                  67.85
FLOORSMIN_MODE                 67.85
YEARS_BUILD_AVG                66.50
YEARS_BUILD_MEDI               66.50
YEARS_BUILD_MODE               66.50
OWN_CAR_AGE                    65.99
LANDAREA_MEDI                  59.38
```

In [18]:

```
len(app_null)
```

Out[18]:

```
64
```

Dropping Missing Values >40%

In [19]:

```
# Dropping columns with missing value >40%
app_null=app_null[app_null.values>=40]
len(app_null)
```

Out[19]:

```
49
```

In [20]:

```
# Hence, the dataset (df_null) is accordingly modified to exclude these values
app_null=list(app_null.index)
app.drop(labels=app_null,axis=1,inplace=True)
app.shape
```

Out[20]:

```
(307511, 73)
```

In [21]:

```
#Checking the data again as to the columns >40% null values are dropped or not
app_null=(app.isnull().sum()*100/len(app)).round(2)
app_null[app_null.values>0].sort_values(ascending=False)
```

Out[21]:

```
OCCUPATION_TYPE               31.35
EXT_SOURCE_3                  19.83
AMT_REQ_CREDIT_BUREAU_HOUR    13.50
AMT_REQ_CREDIT_BUREAU_DAY     13.50
AMT_REQ_CREDIT_BUREAU_WEEK    13.50
AMT_REQ_CREDIT_BUREAU_MON     13.50
AMT_REQ_CREDIT_BUREAU_QRT     13.50
AMT_REQ_CREDIT_BUREAU_YEAR    13.50
NAME_TYPE_SUITE                0.42
OBS_30_CNT_SOCIAL_CIRCLE       0.33
DEF_30_CNT_SOCIAL_CIRCLE       0.33
OBS_60_CNT_SOCIAL_CIRCLE       0.33
DEF_60_CNT_SOCIAL_CIRCLE       0.33
EXT_SOURCE_2                   0.21
AMT_GOODS_PRICE                0.09
dtype: float64
```

Comments:

1. application_data.csv has missing values in 64 coulumns

2. Off which 49 coumns have more than 40 % missing values

Cleaning application_data.csv---Imputing missing values

In [22]:

```
#Checking for coulmns >13% null values
app_null=app_null[app_null.values>=13]
app_null
```

Out[22]:

```
OCCUPATION_TYPE               31.35
EXT_SOURCE_3                  19.83
AMT_REQ_CREDIT_BUREAU_HOUR    13.50
AMT_REQ_CREDIT_BUREAU_DAY     13.50
AMT_REQ_CREDIT_BUREAU_WEEK    13.50
AMT_REQ_CREDIT_BUREAU_MON     13.50
AMT_REQ_CREDIT_BUREAU_QRT     13.50
AMT_REQ_CREDIT_BUREAU_YEAR    13.50
dtype: float64
```

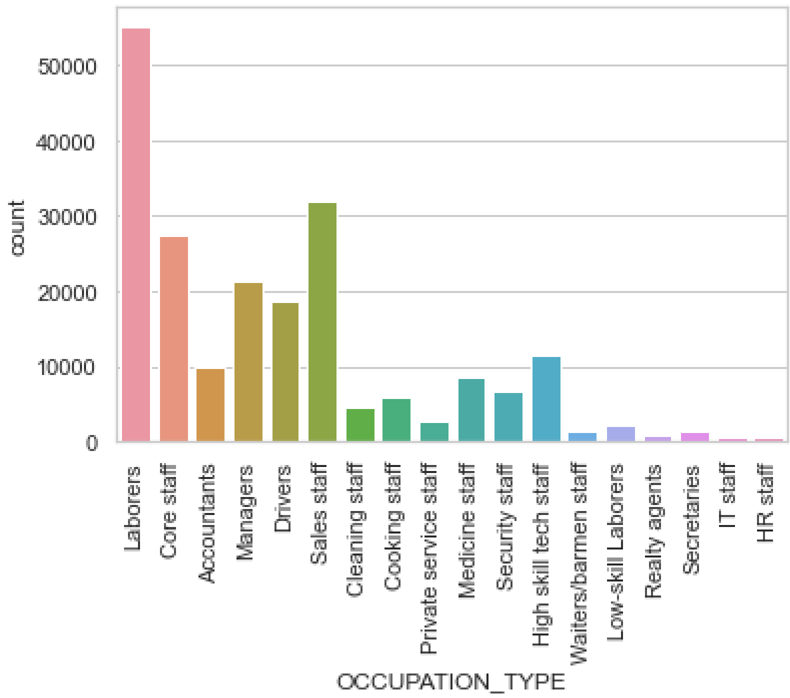
In [23]:

```
#most frequent occuring value in OCCUPATION_TYPE column
occu_mode=app.OCCUPATION_TYPE.mode()
print(f'Frequent value of OCCUPATION_TYPE column: ', occu_mode)
```

```
Frequent value of OCCUPATION_TYPE column:  0    Laborers
dtype: object
```

In [24]:

```
sns.countplot(x='OCCUPATION_TYPE',data=app)
plt.xticks(rotation=90)
plt.show()
```



In [25]:

```
#imputimg missing value with mode() function
app['OCCUPATION_TYPE']=app['OCCUPATION_TYPE'].fillna(occu_mode[0])
```

In [26]:

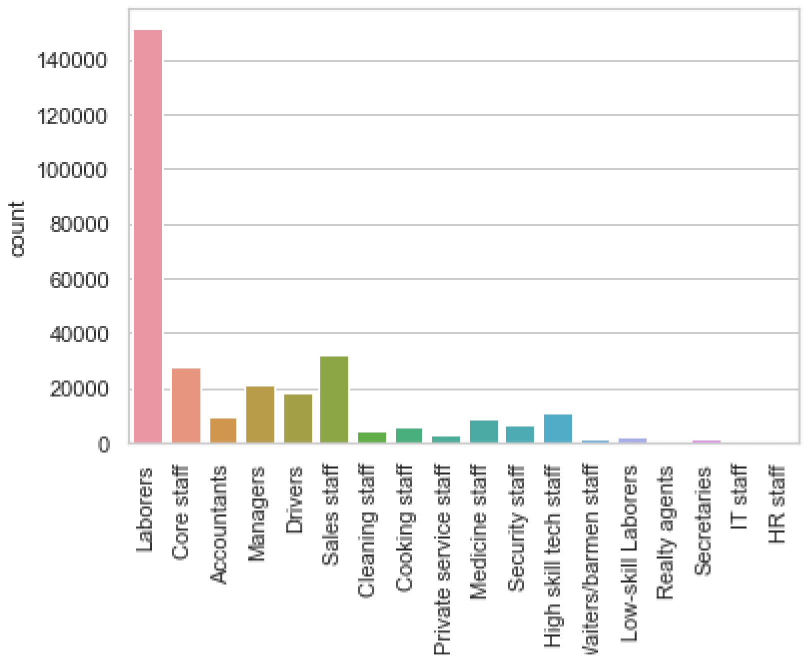
```
#checking if missing values are sucessfully removed
app.OCCUPATION_TYPE.isnull().sum()
```

Out[26]:

```
0
```

In [27]:

```
#replotting to see the change in Ocuupation typye after imputing
sns.countplot(x='OCCUPATION_TYPE',data=app)
plt.xticks(rotation=90)
plt.show()
```



In [28]:

```
#dropping coulmn with missing values`13.50 %
app.drop(['AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY','AMT_REQ_CREDIT_BUREAU_
app.shape
```

Out[28]:

```
(307511, 67)
```

In [29]:

```
#checking if coulmns are dropped
app_null=(100*app.isnull().sum()/len(app)).round(2)
```

In [30]:

```
app_null=app_null[app_null.values>0].sort_values(ascending=False)
app_null
```

Out[30]:

```
EXT_SOURCE_3              19.83
NAME_TYPE_SUITE            0.42
OBS_30_CNT_SOCIAL_CIRCLE   0.33
DEF_30_CNT_SOCIAL_CIRCLE   0.33
OBS_60_CNT_SOCIAL_CIRCLE   0.33
DEF_60_CNT_SOCIAL_CIRCLE   0.33
EXT_SOURCE_2               0.21
AMT_GOODS_PRICE            0.09
dtype: float64
```

In [31]:

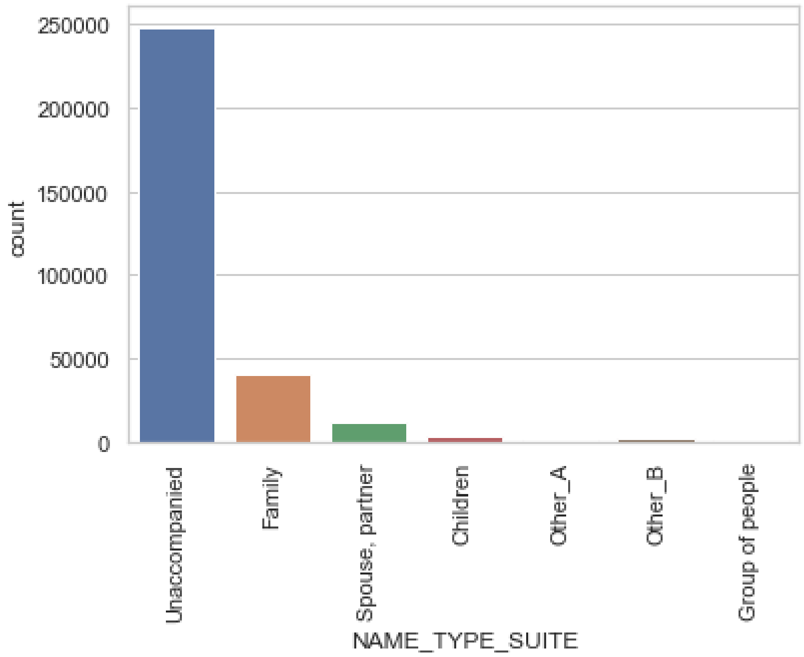
```
#finding most frequent value in NAME_TYPE_SUITE column
app['NAME_TYPE_SUITE'].describe()
```

Out[31]:

```
count            306219
unique                7
top       Unaccompanied
freq             248526
Name: NAME_TYPE_SUITE, dtype: object
```

In [32]:

```
sns.set_theme(style="whitegrid")
sns.countplot(x="NAME_TYPE_SUITE", data=app)
plt.xticks(rotation=90)
plt.show()
```



In [33]:

```
app['NAME_TYPE_SUITE'].isnull().sum()
```

Out[33]:

```
1292
```

In [34]:

```
app['NAME_TYPE_SUITE'].isnull().sum()/len(app['NAME_TYPE_SUITE'])*100
```

Out[34]:

```
0.42014757195677555
```

In [35]:

```
#imputing NAME_TYPE_SUITE by mode
nts=app.NAME_TYPE_SUITE.mode()
nts
```

Out[35]:

```
0    Unaccompanied
dtype: object
```

In [36]:

```
app['NAME_TYPE_SUITE']=app['NAME_TYPE_SUITE'].fillna(app['NAME_TYPE_SUITE'].mode()[0])
```

In [37]:

```
#checking if imputing worked or not
(app.NAME_TYPE_SUITE.isnull().sum()*100/len(app))
```

Out[37]:

```
0.0
```

In [38]:

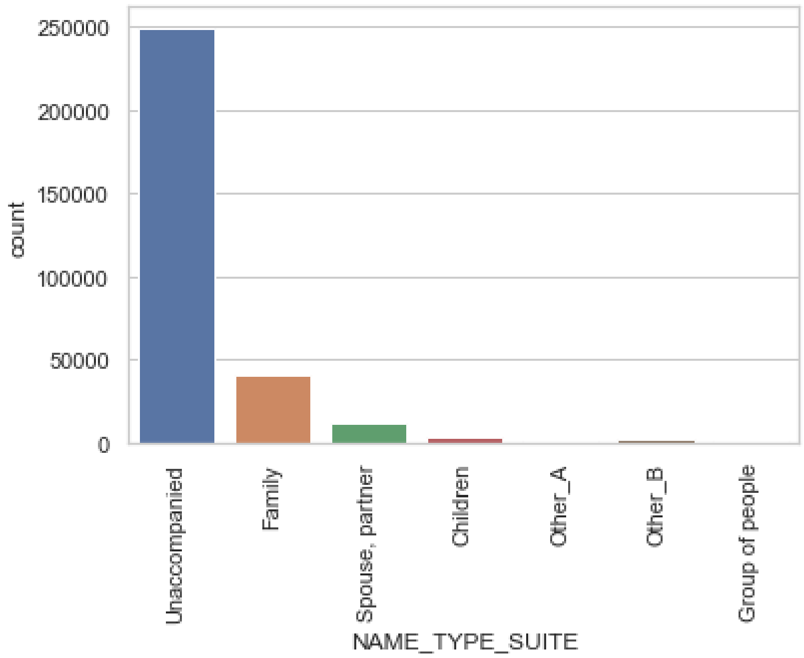
```
app['NAME_TYPE_SUITE'].describe()
```

Out[38]:

```
count            307511
unique                7
top       Unaccompanied
freq             249818
Name: NAME_TYPE_SUITE, dtype: object
```

In [39]:

```
sns.countplot(x="NAME_TYPE_SUITE", data=app)
plt.xticks(rotation=90)
plt.show()
```

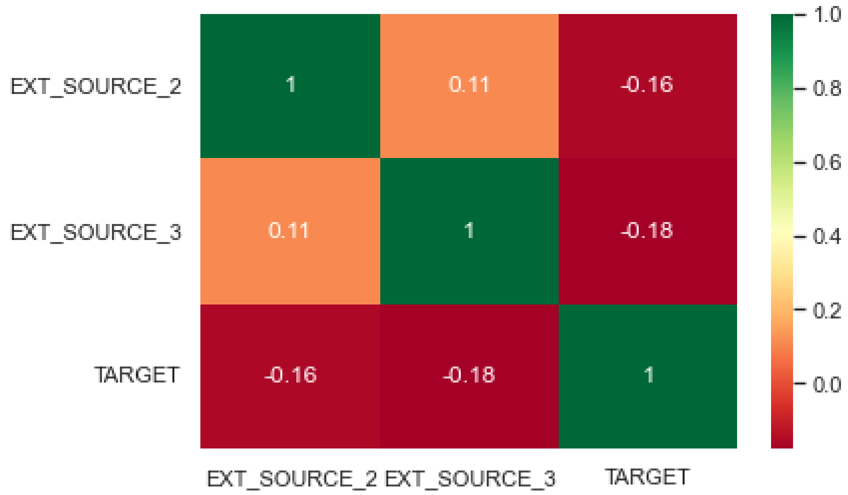


In [40]:

```
#Using multivariate analysis to check correlation between EXT_SOURCE_3, EXT_SOURCE_2 with T
sns.heatmap(app[["EXT_SOURCE_2","EXT_SOURCE_3","TARGET"]].corr(),annot=True,cmap="RdYlGn")
```

Out[40]:

```
<AxesSubplot:>
```



In [41]:

```
#since there is not much correlation between the TARGET column and EXT_SOURCE_2,EXT_SOURCE_
app.drop(['EXT_SOURCE_2','EXT_SOURCE_3'],axis=1,inplace=True)
app.shape
```

Out[41]:

```
(307511, 65)
```

In [42]:

```
#checking missing values after dropping columns
app_null=(100*app.isnull().sum()/len(app)).round(2)
app_null=app_null[app_null.values>0].sort_values(ascending=False)
app_null
```

Out[42]:

```
OBS_30_CNT_SOCIAL_CIRCLE    0.33
DEF_30_CNT_SOCIAL_CIRCLE    0.33
OBS_60_CNT_SOCIAL_CIRCLE    0.33
DEF_60_CNT_SOCIAL_CIRCLE    0.33
AMT_GOODS_PRICE             0.09
dtype: float64
```

Summary:

1.Imputed Null values to 0 for following columns: (AMT_REQ_CREDIT_BUREAU_HOUR, AMT_REQ_CREDIT_BUREAU_DAY,

***AMT_REQ_CREDIT_BUREAU_WEEK,AMT_REQ_CREDIT_BUREAU_MON,AMT_REQ_CREDIT_BUREAU_(
)***

2. Imputed OCCUPATION_TYPE and NAME_TYPE_SUITE columns with mode() as they are categorical columns

3. Dropped EXT_SOURCE_2,EXT_SOURCE_3 columns as they do not have much significance.

4. After treatment of missing values, 73 columns are left in application_data.csv dataset

4. Converting datatypes

In [43]:

```
# Checking the number of unique values is a column; if number of unique values <=40 --> Cat
#number of unique values> 50--> Continuous Column

app.nunique().sort_values()
```

Out[43]:

```
FLAG_DOCUMENT_21               2
FLAG_WORK_PHONE                2
FLAG_DOCUMENT_5                2
FLAG_PHONE                     2
FLAG_EMAIL                     2
FLAG_DOCUMENT_20               2
REG_REGION_NOT_LIVE_REGION     2
REG_REGION_NOT_WORK_REGION     2
LIVE_REGION_NOT_WORK_REGION    2
FLAG_EMP_PHONE                 2
REG_CITY_NOT_LIVE_CITY         2
LIVE_CITY_NOT_WORK_CITY        2
FLAG_DOCUMENT_10               2
FLAG_DOCUMENT_9                2
FLAG_DOCUMENT_8                2
FLAG_DOCUMENT_7                2
FLAG_DOCUMENT_2                2
FLAG_DOCUMENT_3                2
```

In [44]:

```
app.nunique().sort_values()<=40
```

Out[44]:

```
FLAG_DOCUMENT_21               True
FLAG_WORK_PHONE                True
FLAG_DOCUMENT_5                True
FLAG_PHONE                     True
FLAG_EMAIL                     True
FLAG_DOCUMENT_20               True
REG_REGION_NOT_LIVE_REGION     True
REG_REGION_NOT_WORK_REGION     True
LIVE_REGION_NOT_WORK_REGION    True
FLAG_EMP_PHONE                 True
REG_CITY_NOT_LIVE_CITY         True
LIVE_CITY_NOT_WORK_CITY        True
FLAG_DOCUMENT_10               True
FLAG_DOCUMENT_9                True
FLAG_DOCUMENT_8                True
FLAG_DOCUMENT_7                True
FLAG_DOCUMENT_2                True
FLAG_DOCUMENT_3                True
```

In [44]:

```
app.dtypes
```

Out[44]:

```
SK_ID_CURR                       int64
TARGET                           int64
NAME_CONTRACT_TYPE              object
CODE_GENDER                     object
FLAG_OWN_CAR                    object
FLAG_OWN_REALTY                 object
CNT_CHILDREN                     int64
AMT_INCOME_TOTAL               float64
AMT_CREDIT                     float64
AMT_ANNUITY                    float64
AMT_GOODS_PRICE                float64
NAME_TYPE_SUITE                 object
NAME_INCOME_TYPE                object
NAME_EDUCATION_TYPE             object
NAME_FAMILY_STATUS              object
NAME_HOUSING_TYPE               object
REGION_POPULATION_RELATIVE     float64
DAYS_BIRTH                       int64
DAYS_EMPLOYED                    int64
DAYS_REGISTRATION              float64
DAYS_ID_PUBLISH                  int64
FLAG_MOBIL                       int64
FLAG_EMP_PHONE                   int64
FLAG_WORK_PHONE                  int64
FLAG_CONT_MOBILE                 int64
FLAG_PHONE                       int64
FLAG_EMAIL                       int64
OCCUPATION_TYPE                 object
CNT_FAM_MEMBERS                float64
REGION_RATING_CLIENT             int64
REGION_RATING_CLIENT_W_CITY      int64
WEEKDAY_APPR_PROCESS_START      object
HOUR_APPR_PROCESS_START          int64
REG_REGION_NOT_LIVE_REGION       int64
REG_REGION_NOT_WORK_REGION       int64
LIVE_REGION_NOT_WORK_REGION      int64
REG_CITY_NOT_LIVE_CITY           int64
REG_CITY_NOT_WORK_CITY           int64
LIVE_CITY_NOT_WORK_CITY          int64
ORGANIZATION_TYPE               object
OBS_30_CNT_SOCIAL_CIRCLE       float64
DEF_30_CNT_SOCIAL_CIRCLE       float64
OBS_60_CNT_SOCIAL_CIRCLE       float64
DEF_60_CNT_SOCIAL_CIRCLE       float64
DAYS_LAST_PHONE_CHANGE         float64
FLAG_DOCUMENT_2                  int64
FLAG_DOCUMENT_3                  int64
FLAG_DOCUMENT_4                  int64
FLAG_DOCUMENT_5                  int64
FLAG_DOCUMENT_6                  int64
FLAG_DOCUMENT_7                  int64
FLAG_DOCUMENT_8                  int64
FLAG_DOCUMENT_9                  int64
FLAG_DOCUMENT_10                 int64
FLAG_DOCUMENT_11                 int64
```

```
FLAG_DOCUMENT_12                 int64
FLAG_DOCUMENT_13                 int64
FLAG_DOCUMENT_14                 int64
FLAG_DOCUMENT_15                 int64
FLAG_DOCUMENT_16                 int64
FLAG_DOCUMENT_17                 int64
FLAG_DOCUMENT_18                 int64
FLAG_DOCUMENT_19                 int64
FLAG_DOCUMENT_20                 int64
FLAG_DOCUMENT_21                 int64
dtype: object
```

In [45]:

```
#converting from object to bool
app[['REG_REGION_NOT_LIVE_REGION','REG_REGION_NOT_WORK_REGION','LIVE_REGION_NOT_WORK_REGION
```

In [46]:

```
#converting object to bool
app[['FLAG_OWN_CAR','FLAG_OWN_REALTY','FLAG_MOBIL','FLAG_EMP_PHONE']] = app[['FLAG_OWN_CAR'
```

In [47]:

```
app.dtypes
```

Out[47]:

```
SK_ID_CURR                       int64
TARGET                           int64
NAME_CONTRACT_TYPE              object
CODE_GENDER                     object
FLAG_OWN_CAR                      bool
FLAG_OWN_REALTY                   bool
CNT_CHILDREN                     int64
AMT_INCOME_TOTAL               float64
AMT_CREDIT                     float64
AMT_ANNUITY                    float64
AMT_GOODS_PRICE                float64
NAME_TYPE_SUITE                 object
NAME_INCOME_TYPE                object
NAME_EDUCATION_TYPE             object
NAME_FAMILY_STATUS              object
NAME_HOUSING_TYPE               object
REGION_POPULATION_RELATIVE     float64
DAYS_BIRTH                       int64
DAYS_EMPLOYED                    int64
DAYS_REGISTRATION              float64
DAYS_ID_PUBLISH                  int64
FLAG_MOBIL                        bool
FLAG_EMP_PHONE                    bool
FLAG_WORK_PHONE                  int64
FLAG_CONT_MOBILE                 int64
FLAG_PHONE                       int64
FLAG_EMAIL                       int64
OCCUPATION_TYPE                 object
CNT_FAM_MEMBERS                float64
REGION_RATING_CLIENT             int64
REGION_RATING_CLIENT_W_CITY      int64
WEEKDAY_APPR_PROCESS_START      object
HOUR_APPR_PROCESS_START          int64
REG_REGION_NOT_LIVE_REGION        bool
REG_REGION_NOT_WORK_REGION        bool
LIVE_REGION_NOT_WORK_REGION       bool
REG_CITY_NOT_LIVE_CITY            bool
REG_CITY_NOT_WORK_CITY            bool
LIVE_CITY_NOT_WORK_CITY           bool
ORGANIZATION_TYPE               object
OBS_30_CNT_SOCIAL_CIRCLE       float64
DEF_30_CNT_SOCIAL_CIRCLE       float64
OBS_60_CNT_SOCIAL_CIRCLE       float64
DEF_60_CNT_SOCIAL_CIRCLE       float64
DAYS_LAST_PHONE_CHANGE         float64
FLAG_DOCUMENT_2                  int64
FLAG_DOCUMENT_3                  int64
FLAG_DOCUMENT_4                  int64
FLAG_DOCUMENT_5                  int64
FLAG_DOCUMENT_6                  int64
FLAG_DOCUMENT_7                  int64
FLAG_DOCUMENT_8                  int64
FLAG_DOCUMENT_9                  int64
FLAG_DOCUMENT_10                 int64
FLAG_DOCUMENT_11                 int64
```

```
FLAG_DOCUMENT_12              int64
FLAG_DOCUMENT_13              int64
FLAG_DOCUMENT_14              int64
FLAG_DOCUMENT_15              int64
FLAG_DOCUMENT_16              int64
FLAG_DOCUMENT_17              int64
FLAG_DOCUMENT_18              int64
FLAG_DOCUMENT_19              int64
FLAG_DOCUMENT_20              int64
FLAG_DOCUMENT_21              int64
dtype: object
```

5. Converting columns with negative values to positive

In [48]:

```
app['DAYS_BIRTH']=app['DAYS_BIRTH'].abs()
app['DAYS_EMPLOYED']=app['DAYS_EMPLOYED'].abs()
app['DAYS_REGISTRATION']=app['DAYS_REGISTRATION'].abs()
app['DAYS_ID_PUBLISH']=app['DAYS_ID_PUBLISH'].abs()
app['DAYS_LAST_PHONE_CHANGE']=app['DAYS_LAST_PHONE_CHANGE'].abs()
app.head()
```

Out[48]:

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

5 rows × 65 columns

In [49]:

```
app['DAYS_EMPLOYED']=app['DAYS_EMPLOYED'].astype("int64")
```

In [50]:

```
app['DAYS_BIRTH'].describe()
```

Out[50]:

```
count    307511.000000
mean      16036.995067
std        4363.988632
min        7489.000000
25%       12413.000000
50%       15750.000000
75%       19682.000000
max       25229.000000
Name: DAYS_BIRTH, dtype: float64
```

5. Checking columns with FLAGS and their relation with TARGET column inorder to remove irrelevant ones

In [51]:

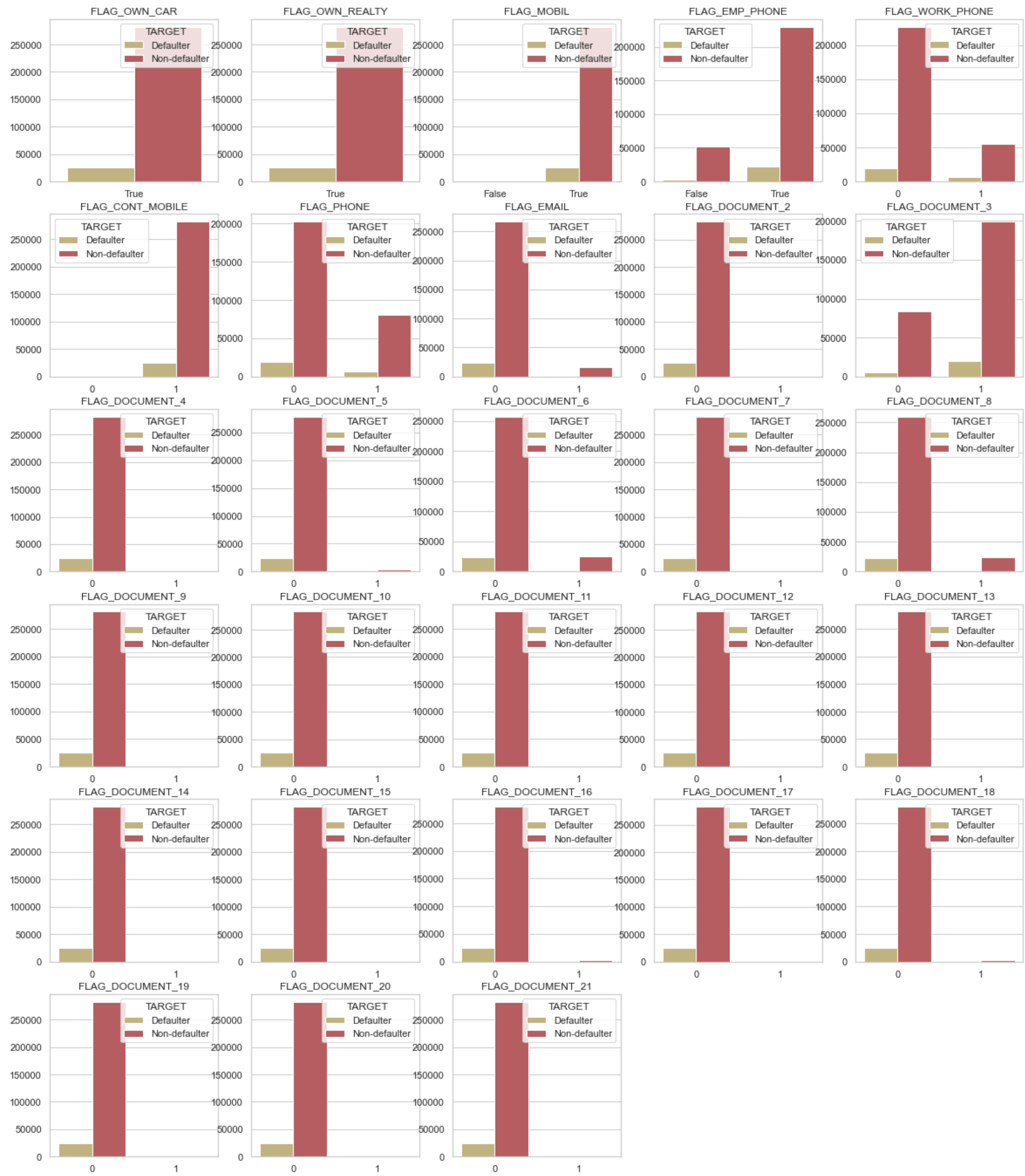
```
# adding all flags coloumns in variable "flag_columns"

flag_cols = [col for col in app.columns if "FLAG" in col]
flag_cols
```

Out[51]:

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

In [52]:

```
import itertools
# Plotting all the graphs to find the relation
app_flag = app[flag_cols+["TARGET"]]
app_flag["TARGET"] = app_flag["TARGET"].replace({1:"Defaulter",0:"Non-defaulter"})
plt.figure(figsize = [20,24])

for l,m in itertools.zip_longest(flag_cols,range(len(flag_cols))):
    plt.subplot(6,5,m+1)
    bx = sns.countplot(app_flag[l], hue = app_flag["TARGET"], palette = ["y","r"])
    plt.xlabel("")
    plt.ylabel("")
    plt.title(l)
```



Cumclusion:Columns (FLAG_OWN_REALTY, FLAG_MOBIL ,FLAG_CONT_MOBILE,FLAG_EMP_PHONE,, FLAG_DOCUMENT_3) have more nondefaulters than defaulter. Hence keeping these columns,alongwith FLAG_OWN_CAR and deleting rest of the fLAG columns for analysis.

In [53]:

```
#removing required columns from app_flag list
app_flag.drop(['FLAG_DOCUMENT_3','FLAG_OWN_REALTY', 'FLAG_MOBIL' ,'FLAG_EMP_PHONE', 'FLAG_C
```

In [54]:

```
#dropping the unwanted Flag columns
app.drop(app_flag.columns, axis=1, inplace= True)
```

In [55]:

```
#checking shape of application_data.csv after dropping Flag columns
app.shape
```

Out[55]:

```
(307511, 43)
```

In [56]:

```
app.FLAG_OWN_CAR.describe()
```

Out[56]:

```
count     307511
unique         1
top         True
freq      307511
Name: FLAG_OWN_CAR, dtype: object
```

In [57]:

```
app.FLAG_OWN_REALTY.describe()
```

Out[57]:

```
count     307511
unique         1
top         True
freq      307511
Name: FLAG_OWN_REALTY, dtype: object
```

Further dropping columns not important for analysis

In [58]:

```
app.drop(['DAYS_REGISTRATION','DAYS_ID_PUBLISH', 'CNT_CHILDREN','OBS_30_CNT_SOCIAL_CIRCLE',
app.shape
```

Out[58]:

```
(307511, 34)
```

In [59]:

```
app.drop([],axis=1,inplace=True)
app.shape
```

Out[59]:

```
(307511, 34)
```

In [60]:

```
app.drop(['REGION_RATING_CLIENT','REGION_RATING_CLIENT_W_CITY','HOUR_APPR_PROCESS_START','W
app.shape
```

Out[60]:

```
(307511, 24)
```

In [61]:

```
app.columns
```

Out[61]:

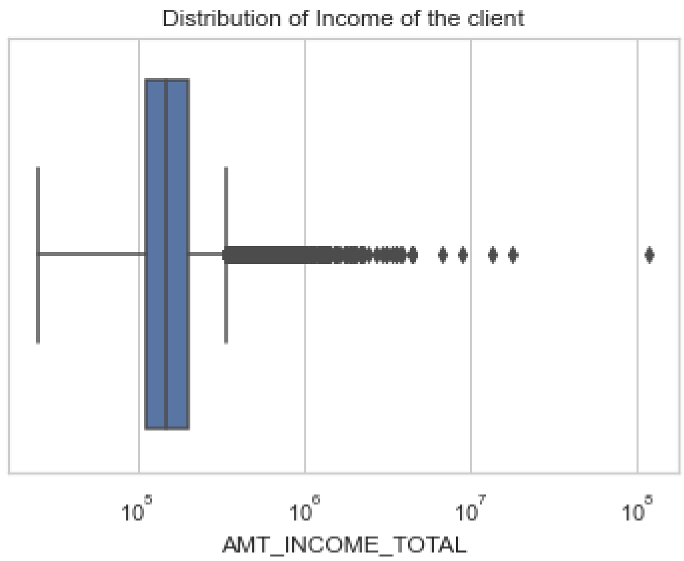
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'AMT_INCOME_TOTAL', 'AMT_CREDIT',
       'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYP
E',
       'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE',
       'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED',
       'FLAG_MOBIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'ORGANIZATION_TYP
E',
       'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_3'],
      dtype='object')
```

Conclusion: After deleting the unnecessary columns we are left with 24 Columns in application_data.csv dataset

6. Checking for outliers

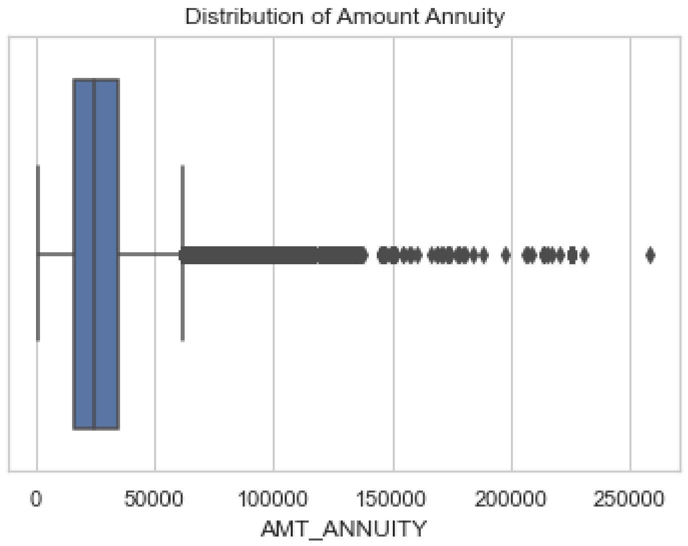
In [62]:

```
sns.boxplot(app.AMT_INCOME_TOTAL)
plt.title('Distribution of Income of the client')
plt.xscale('log')
plt.show()
```



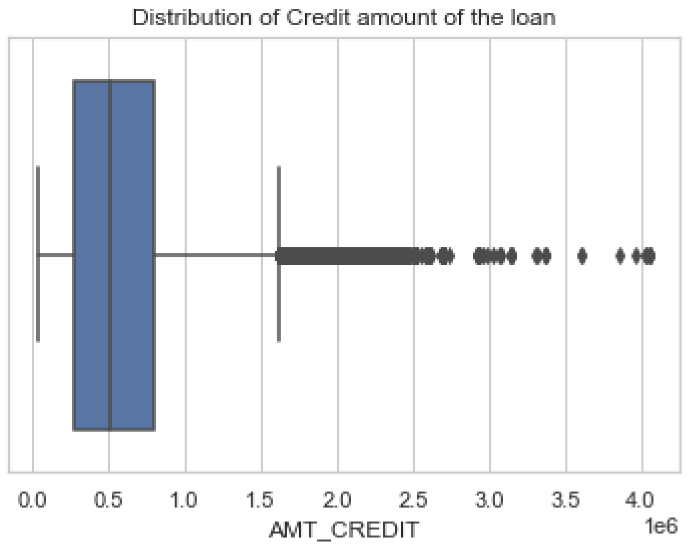
In [63]:

```
sns.boxplot(app.AMT_ANNUITY)
plt.title('Distribution of Amount Annuity')
plt.show()
```



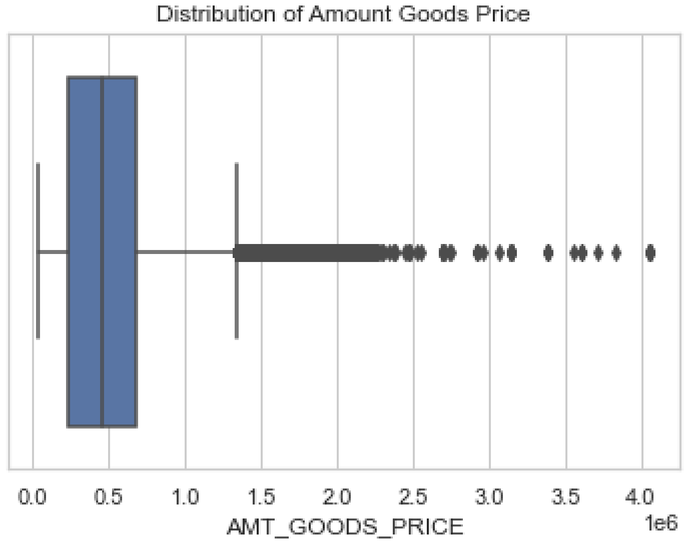
In [64]:

```
sns.boxplot(app.AMT_CREDIT)
plt.title('Distribution of Credit amount of the loan')
plt.show()
```



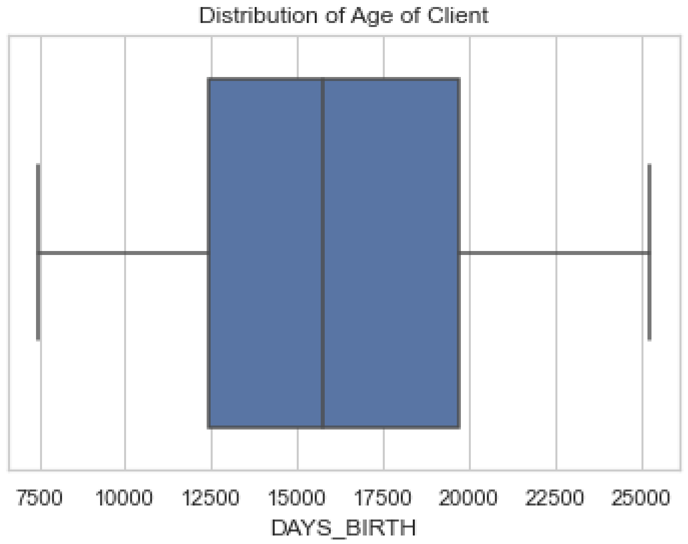
In [65]:

```
sns.boxplot(app.AMT_GOODS_PRICE)
plt.title('Distribution of Amount Goods Price')
plt.show()
```



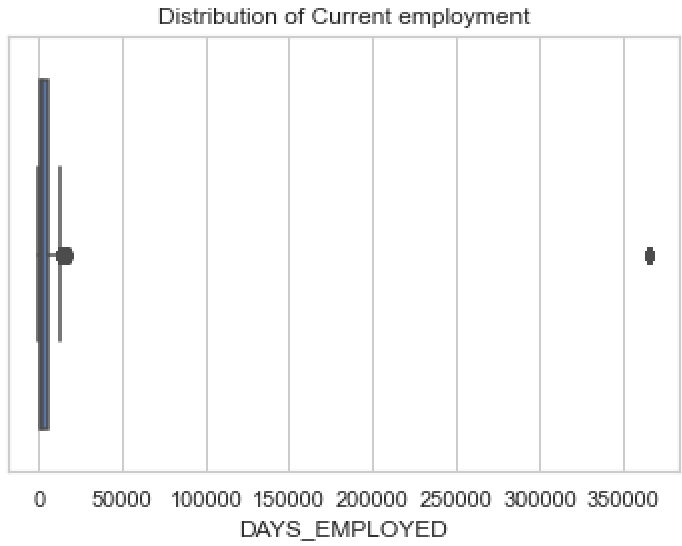
In [65]:

```
sns.boxplot(app.DAYS_BIRTH)
plt.title('Distribution of Age of Client')
plt.show()
```



In [66]:

```
sns.boxplot(app.DAYS_EMPLOYED)
plt.title('Distribution of Current employment')
plt.show()
```



Conclusion:

1. AMT_ANNUITY, AMT_CREDIT, AMT_GOODS_PRICE, and AMT_INCOME_TOTAL have high number of outliers which indicating few loan applicants have high income in comparison the others.
2. DAYS_BIRTH has no outliers meaning available data is reliable.
3. DAYS_EMPLOYED has outlier value above 350000(days) which is impossible and hence this must to be incorrect entry.

7.Binning

7.1 Binning age of clients

In [67]:

```
app["DAYS_BIRTH"].describe()
```

Out[67]:

```
count    307511.000000
mean      16036.995067
std        4363.988632
min        7489.000000
25%       12413.000000
50%       15750.000000
75%       19682.000000
max       25229.000000
Name: DAYS_BIRTH, dtype: float64
```

In [68]:

```
app["AGE_YEARS"] = app["DAYS_BIRTH"]//365
app["AGE_YEARS"]
```

Out[68]:

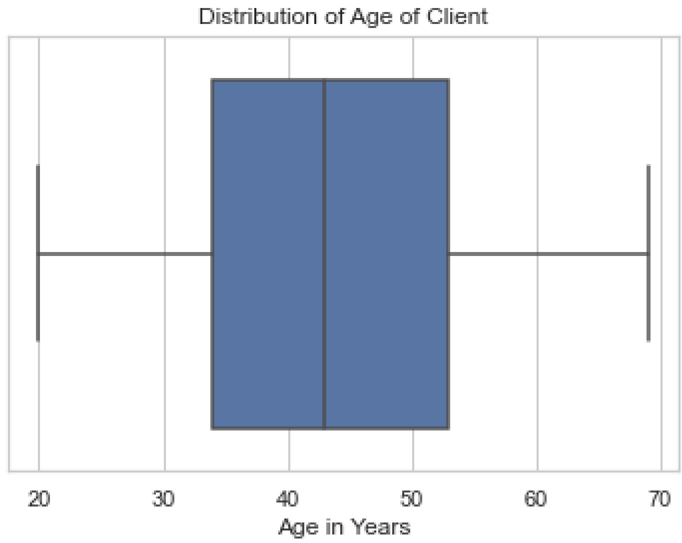
```
0         25
1         45
2         52
3         52
4         54
          ..
307506    25
307507    56
307508    41
307509    32
307510    46
Name: AGE_YEARS, Length: 307511, dtype: int64
```

In [69]:

```
bins = [0,20,25,30,35,40,45,50,55,60,100]
label = ["0-20","20-25","25-30","30-35","35-40","40-45","45-50","50-55","55-60","60 Above"]
app["AGE_GROUP"] = pd.cut(app["AGE_YEARS"], bins=bins, labels=label)
```

In [70]:

```
sns.boxplot(app['AGE_YEARS'])
plt.title('Distribution of Age of Client')
plt.xlabel("Age in Years")
plt.show()
```



In [71]:

```
round(app.AGE_GROUP.value_counts(normalize=True)*100,2)
```

Out[71]:

```
35-40        14.20
40-45        13.01
30-35        12.82
25-30        11.87
50-55        11.41
45-50        11.19
55-60        10.64
60 Above      9.55
20-25         5.31
0-20          0.00
Name: AGE_GROUP, dtype: float64
```

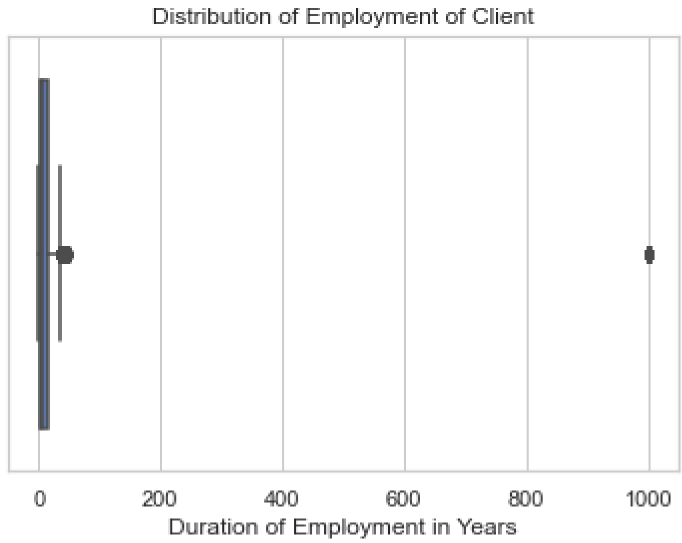
7.2 Binning Employement years

In [72]:

```
app["EMPLOYEMENT_YEARS"] = app["DAYS_EMPLOYED"]//365
bins = [0,5,10,15,20,25,30,50]
label = ["0-5","5-10","10-15","15-20","20-25","25-30","30 Above"]

app["EMPLOYEMENT_YEARS_GROUP"] = pd.cut(app["EMPLOYEMENT_YEARS"], bins=bins, labels=label)
```

In [73]:

```
sns.boxplot(app['EMPLOYEMENT_YEARS'])
plt.title('Distribution of Employment of Client')
plt.xlabel("Duration of Employment in Years")
plt.show()
```



In [74]:

```
round(app.EMPLOYEMENT_YEARS_GROUP.value_counts(normalize=True)*100,2)
```

Out[74]:

```
0-5         55.58
5-10        24.97
10-15       10.23
15-20        4.34
20-25        2.44
25-30        1.31
30 Above     1.14
Name: EMPLOYEMENT_YEARS_GROUP, dtype: float64
```

7.3 Binning Income

In [75]:

```
app["AMT_INCOME_TOTAL"].describe()
```

Out[75]:

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

In [76]:

```
bins = [0,20000,50000,90000,100000]
label = ['Low','Medium','High','Very_high']
app["INCOME_RANGE"] = pd.cut(app["AMT_INCOME_TOTAL"], bins=bins, labels=label)
```

In [77]:

```
round(app.INCOME_RANGE.value_counts(normalize=True)*100,2)
```

Out[77]:

```
High         83.84
Very_high     9.07
Medium        7.09
Low           0.00
Name: INCOME_RANGE, dtype: float64
```

7.4 Bining Credit amount

In [78]:

```
bins = [0,350000,700000,1000000000]
label= ['Low','Medium','High']
app['CREDIT_RANGE']=pd.cut(app['AMT_CREDIT'],bins=bins,labels=label)
```

In [79]:

```
round(app.CREDIT_RANGE.value_counts(normalize=True)*100,2)
```

Out[79]:

```
Low       34.85
Medium    32.67
High      32.49
Name: CREDIT_RANGE, dtype: float64
```

8 Calculating imbalance %

In [80]:

```
app.TARGET.value_counts()
```

Out[80]:

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

In [81]:

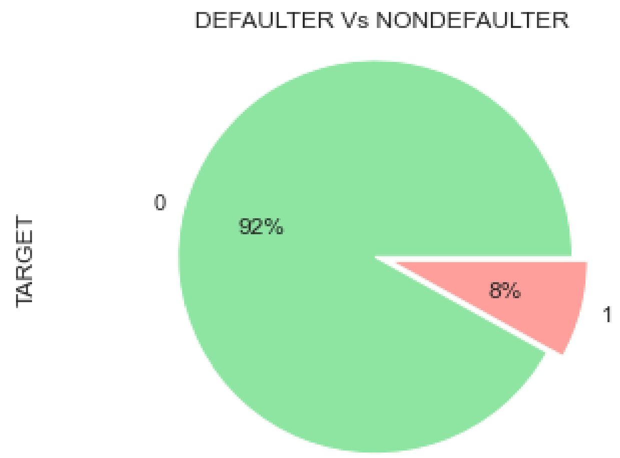
```
app.TARGET.value_counts(normalize=True)*100
```

Out[81]:

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

In [82]:

```
px=app.TARGET.value_counts(normalize=True).plot.pie(colors=sns.color_palette('pastel')[2:8]
px.axis('equal')
plt.title('DEFAULTER Vs NONDEFAULTER')
plt.show()
```



Dividing dataset based on imbalance

In [82]:

```
defaulters=app[app['TARGET']==1]
non_defaulters=app[app["TARGET"]==0]
```

In [88]:

```
defaulters.head()
```

Out[88]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_
0	100002	1	Cash loans	M	True	
26	100031	1	Cash loans	F	True	
40	100047	1	Cash loans	M	True	
42	100049	1	Cash loans	F	True	
81	100096	1	Cash loans	F	True	

5 rows × 30 columns

In [83]:

```
defaulters.shape
```

Out[83]:

```
(24825, 30)
```

In [84]:

```
non_defaulters.head()
```

Out[84]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_(
1	100003	0	Cash loans	F	True	
2	100004	0	Revolving loans	M	True	
3	100006	0	Cash loans	F	True	
4	100007	0	Cash loans	M	True	
5	100008	0	Cash loans	M	True	

5 rows × 30 columns

In [86]:

```
non_defaulters.shape
```

Out[86]:

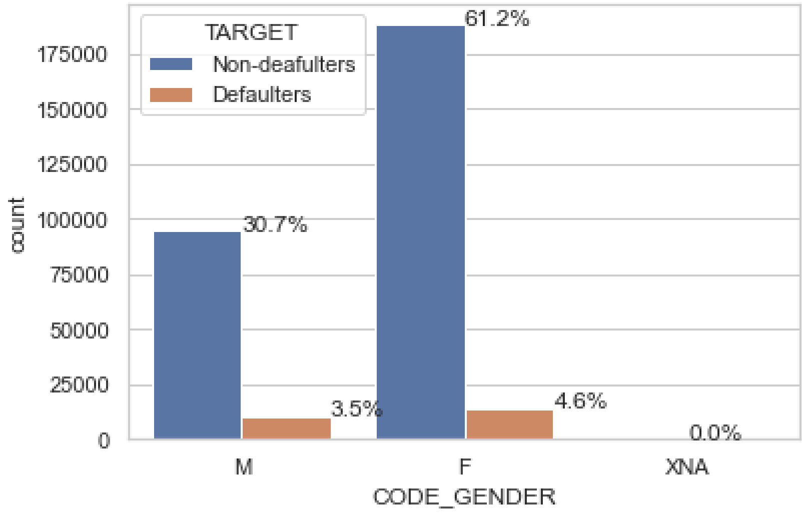
```
(282686, 30)
```

UNIVARIATE ANALYSIS

CATEGORICAL VARIABLES

In [85]:

```
ax=sns.countplot(app['CODE_GENDER'], hue=app["TARGET"])
plt.legend(labels=["Non-deafulters","Defaulters"], title = "TARGET")
total = float(len(app))
for p in ax.patches:
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate('{:.1f}%'.format(100 * p.get_height()/total),(x,y))
plt.show()
```



In [86]:

```
defaulters.CODE_GENDER.value_counts(normalize=True)*100
```

Out[86]:

```
F    57.079557
M    42.920443
Name: CODE_GENDER, dtype: float64
```

In [87]:

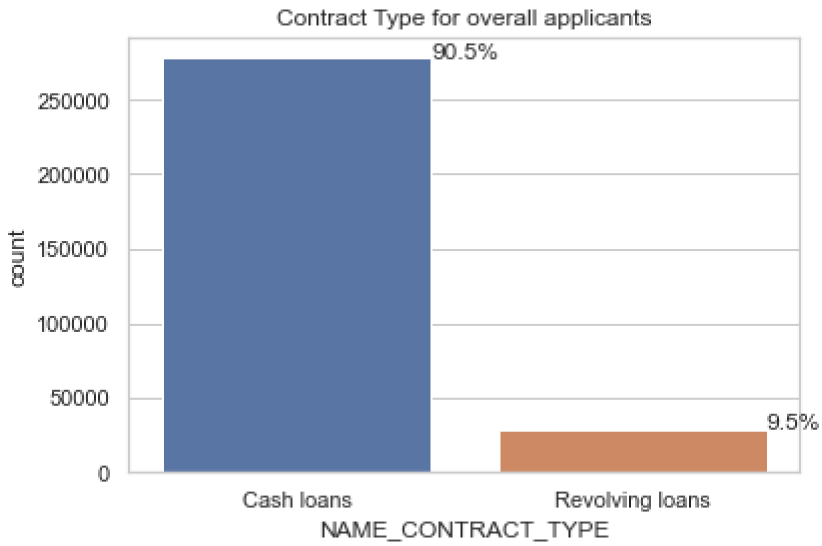
```
non_defaulters.CODE_GENDER.value_counts(normalize=True)*100
```

Out[87]:

```
F      66.603228
M      33.395357
XNA     0.001415
Name: CODE_GENDER, dtype: float64
```

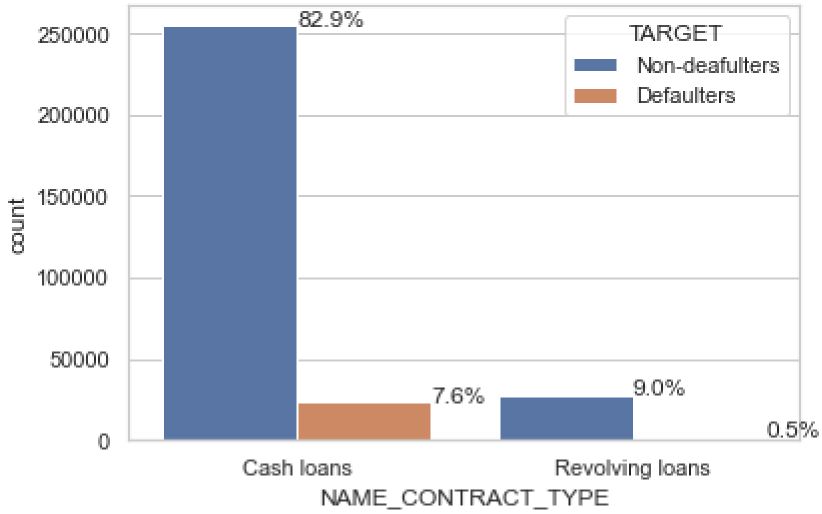
In [88]:

```
ax=sns.countplot("NAME_CONTRACT_TYPE",data=app)
total = float(len(app))
for p in ax.patches:
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate('{:.1f}%'.format(100 * p.get_height()/total),(x,y))
plt.title("Contract Type for overall applicants")
plt.show()
```



In [89]:

```
ax=sns.countplot(app['NAME_CONTRACT_TYPE'], hue=app["TARGET"])
plt.legend(labels=["Non-deafulters","Defaulters"], title = "TARGET")
total = float(len(app))
for p in ax.patches:
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate('{:.1f}%'.format(100 * p.get_height()/total),(x,y))
plt.show()
```



Comments:

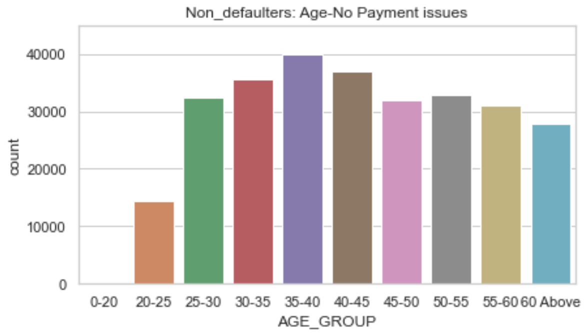
1.Bank offers two kinds of loans, viz., Cash loans and Revolving loans. 2. Cash loans have much higher % than Revolving loans

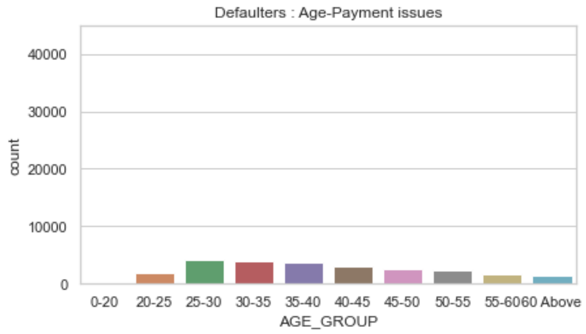
Comments:

1. More % of loans are provided to non-defaulters

In [90]:

```
plt.figure(figsize = (15, 8))
plt.subplot(2, 2, 1)
plt.ylim(0,45000)
plt.title('Non_defaulters: Age-No Payment issues')
sns.countplot(non_defaulters['AGE_GROUP'])

# subplot 2
plt.subplot(2, 2, 2)
plt.title('Defaulters : Age-Payment issues')
plt.ylim(0,45000)
sns.countplot(defaulters['AGE_GROUP'])
plt.show()
```

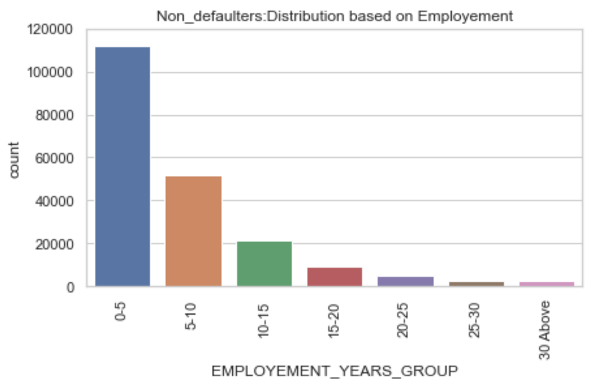


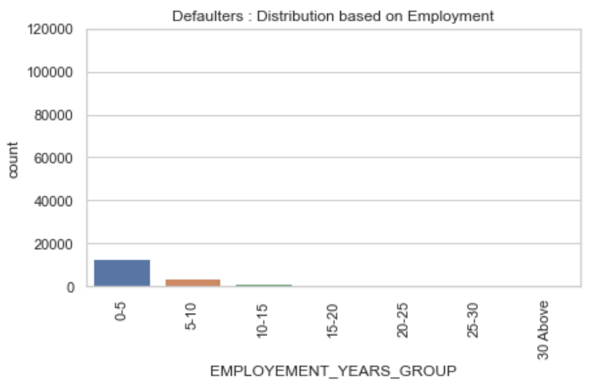


comments: Customers from age 35-45 most likely to make payment.

In [91]:

```
plt.figure(figsize = (15, 8))
plt.subplot(2, 2, 1)
plt.ylim(0,120000)
plt.title('Non_defaulters:Distribution based on Employement')
sns.countplot(non_defaulters['EMPLOYEMENT_YEARS_GROUP'])
plt.xticks(rotation=90)

# subplot 2
plt.subplot(2, 2, 2)
plt.title('Defaulters : Distribution based on Employment')
plt.ylim(0,120000)
sns.countplot(defaulters['EMPLOYEMENT_YEARS_GROUP'])
plt.xticks(rotation=90)
plt.show()
```



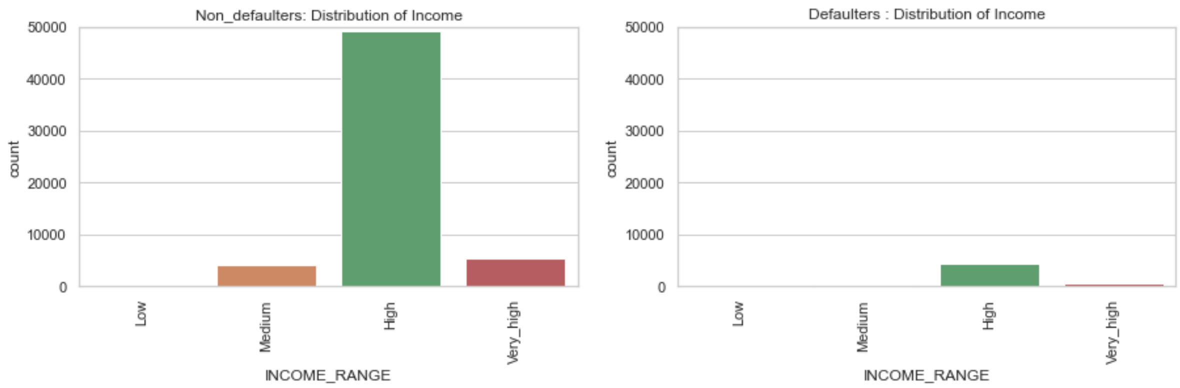


comments:

1. People employed from 0-5 years take more loans
2. People employed for 0-5 years are more likely to be defaulters

In [92]:

```
plt.figure(figsize = (15, 8))
plt.subplot(2, 2, 1)
plt.ylim(0,50000)
plt.title('Non_defaulters: Distribution of Income')
sns.countplot(non_defaulters['INCOME_RANGE'])
plt.xticks(rotation=90)

plt.subplot(2, 2, 2)
plt.title('Defaulters : Distribution of Income')
plt.ylim(0,50000)
sns.countplot(defaulters['INCOME_RANGE'])
plt.xticks(rotation=90)
plt.show()
```



Comment: Defaulters have significantly lesser income than non_defaulters

In [93]:

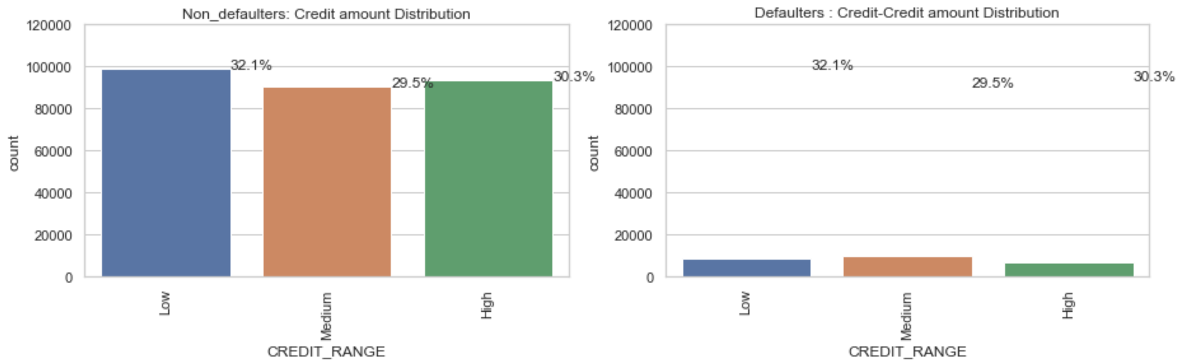
```
defaulters.INCOME_RANGE.value_counts(normalize=True)*100
```

Out[93]:

```
High         84.038278
Very_high     9.397129
Medium        6.564593
Low           0.000000
Name: INCOME_RANGE, dtype: float64
```

In [97]:

```
plt.figure(figsize = (15, 8))
plt.subplot(2, 2, 1)
plt.ylim(0,120000)
plt.title('Non_defaulters: Credit amount Distribution')
ax=sns.countplot(non_defaulters['CREDIT_RANGE'])
total = float(len(app))
for p in ax.patches:
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate('{:.1f}%'.format(100 * p.get_height()/total),(x,y))
plt.xticks(rotation=90)

# subplot 2
plt.subplot(2, 2, 2)
plt.title('Defaulters : Credit-Credit amount Distribution')
plt.ylim(0,120000)
ax1=sns.countplot(defaulters['CREDIT_RANGE'])
plt.xticks(rotation=90)
total = float(len(app))
for p in ax.patches:
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax1.annotate('{:.1f}%'.format(100 * p.get_height()/total),(x,y))
plt.show()
```

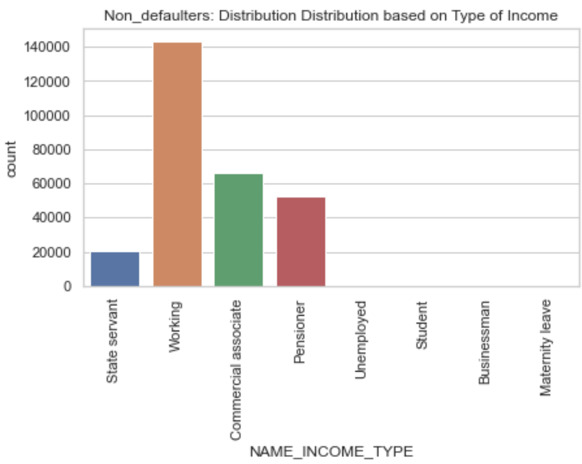


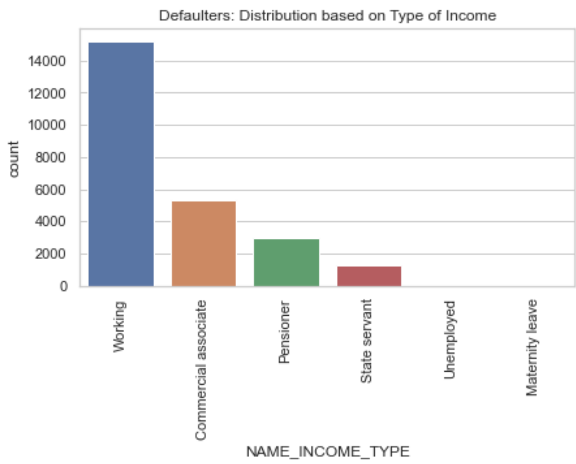
Comment:

1.Low credit amount have high distribution in non-defaulters, followed by High credit amount and then Medium.
2. While in defaulters with medium credit amount showed more susceptibility towards defaulting.

In [94]:

```
plt.figure(figsize = (15, 8))
plt.subplot(2, 2, 1)
plt.title('Non_defaulters: Distribution Distribution based on Type of Income')
sns.countplot(non_defaulters['NAME_INCOME_TYPE'])
plt.xticks(rotation=90)

plt.subplot(2, 2, 2)
plt.title('Defaulters: Distribution based on Type of Income')
sns.countplot(defaulters['NAME_INCOME_TYPE'])
plt.xticks(rotation=90)
plt.show()
```





In [95]:

```
defaulters['NAME_INCOME_TYPE'].value_counts(normalize=True)*100
```

Out[95]:

```
Working                 61.325277
Commercial associate    21.591138
Pensioner               12.012085
State servant            5.031219
Unemployed               0.032226
Maternity leave          0.008056
Name: NAME_INCOME_TYPE, dtype: float64
```

In [96]:

```
non_defaulters['NAME_INCOME_TYPE'].value_counts(normalize=True)*100
```

Out[96]:

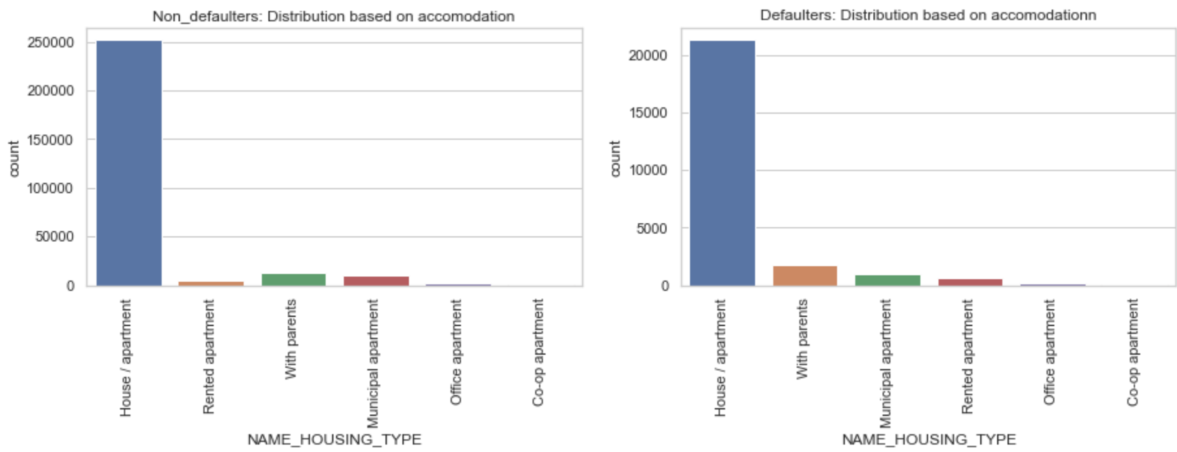
```
Working                 50.780725
Commercial associate    23.438373
Pensioner               18.529393
State servant            7.235590
Student                  0.006367
Unemployed               0.004952
Businessman              0.003537
Maternity leave          0.001061
Name: NAME_INCOME_TYPE, dtype: float64
```

Comments:

1. Loans are mostly given to working class, then by Commercial associate and Pensioner
2. Highest percentage of defaulters and non-defaulters are working classes.
3. Students are non-defaulters: as may be they are not required to pay when they are students
4. Businessman are non-defaulters.
5. State servants are not likely to be defaulters

In [97]:

```
plt.figure(figsize = (15, 8))
plt.subplot(2, 2, 1)
plt.title('Non_defaulters: Distribution based on accomodation')
sns.countplot(non_defaulters['NAME_HOUSING_TYPE'])
plt.xticks(rotation=90)

plt.subplot(2, 2, 2)
plt.title('Defaulters: Distribution based on accomodationn')
sns.countplot(defaulters['NAME_HOUSING_TYPE'])
plt.xticks(rotation=90)
plt.show()
```



In [98]:

```
defaulters.NAME_HOUSING_TYPE.value_counts(normalize=True)*100
```

Out[98]:

```
House / apartment      85.687815
With parents            6.992951
Municipal apartment     3.846928
Rented apartment        2.420947
Office apartment        0.692850
Co-op apartment         0.358510
Name: NAME_HOUSING_TYPE, dtype: float64
```

In [99]:

```
non_defaulters.NAME_HOUSING_TYPE.value_counts(normalize=True)*100
```

Out[99]:

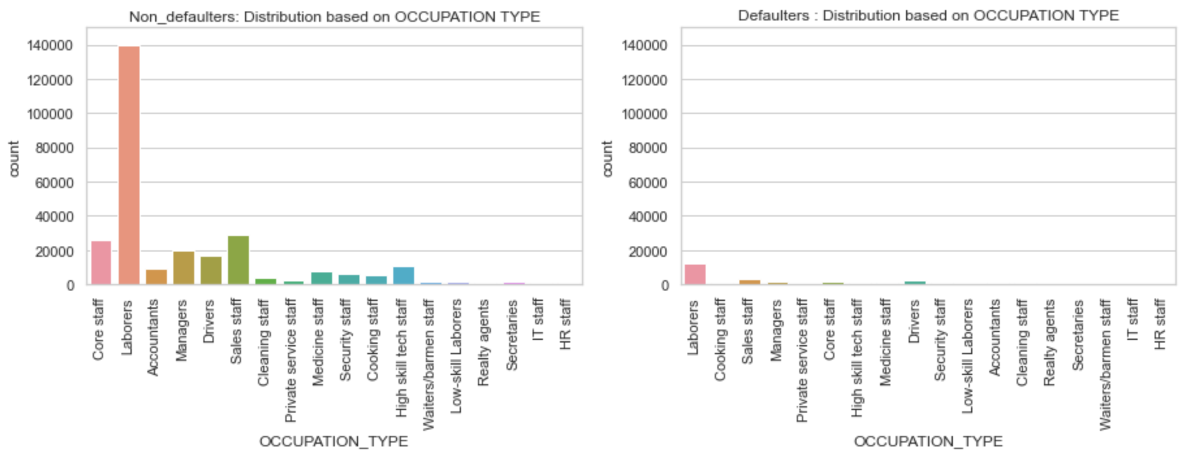
```
House / apartment      89.001931
With parents            4.635532
Municipal apartment     3.618149
Rented apartment        1.514047
Office apartment        0.864917
Co-op apartment         0.365423
Name: NAME_HOUSING_TYPE, dtype: float64
```

COMMENTS:

1.People with own apartments are given max loan, making them more prone to defaulter status 2.People in rented apartment are not likely to be defaulters 3.People living with parents have more chances of being defaulters

In [100]:

```
plt.figure(figsize = (15, 8))
plt.subplot(2, 2, 1)
plt.ylim(0,150000)
plt.title('Non_defaulters: Distribution based on OCCUPATION TYPE')
sns.countplot(non_defaulters['OCCUPATION_TYPE'])
plt.xticks(rotation=90)

# subplot 2
plt.subplot(2, 2, 2)
plt.title('Defaulters : Distribution based on OCCUPATION TYPE')
plt.ylim(0,150000)
sns.countplot(defaulters['OCCUPATION_TYPE'])
plt.xticks(rotation=90)
plt.show()
```



Comments:

1. Loans are mostly taken by Laborers, then by Sales staff and Core staff.
2. IT staff, HR staff and Realty staff are take less loan.
3. Category with highest percent of defaulters are laborer's followed by Drivers and Sales staff, Managers and Core staff.

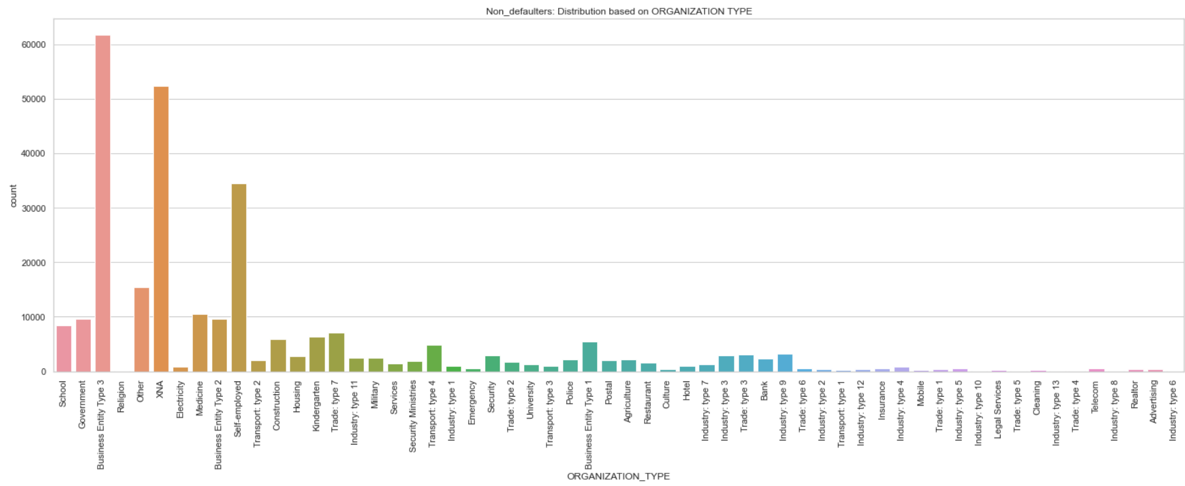
In [101]:

```
plt.figure(figsize = (25, 8))
plt.title('Non_defaulters: Distribution based on ORGANIZATION TYPE')
sns.countplot(non_defaulters['ORGANIZATION_TYPE'])
plt.xticks(rotation=90)
```

Out[101]:

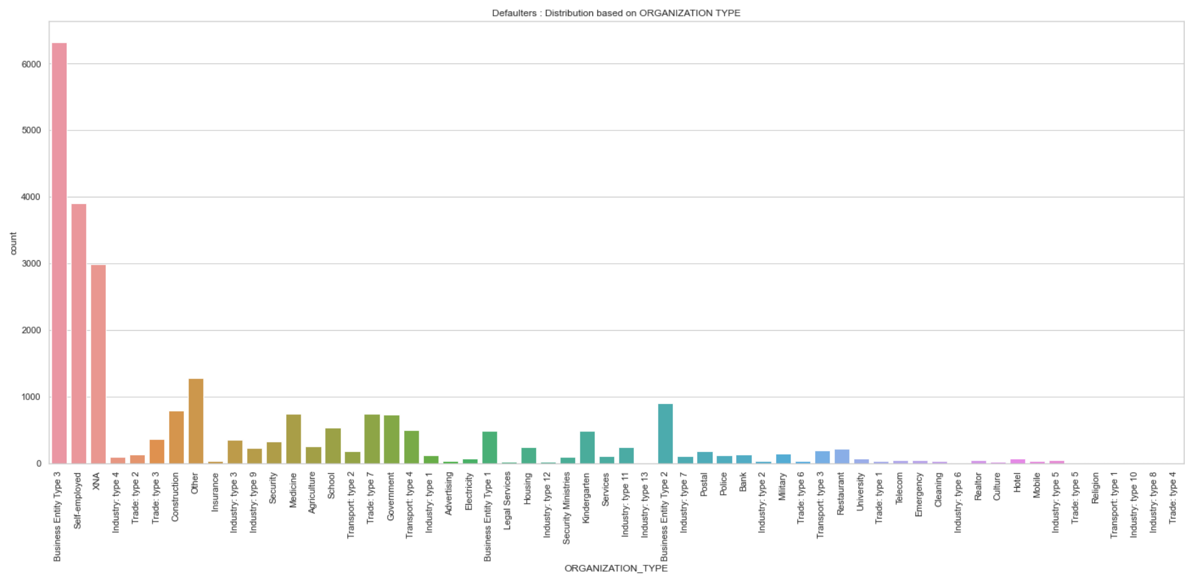
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57]),
 [Text(0, 0, 'School'),
  Text(1, 0, 'Government'),
  Text(2, 0, 'Business Entity Type 3'),
  Text(3, 0, 'Religion'),
  Text(4, 0, 'Other'),
  Text(5, 0, 'XNA'),
  Text(6, 0, 'Electricity'),
  Text(7, 0, 'Medicine'),
  Text(8, 0, 'Business Entity Type 2'),
  Text(9, 0, 'Self-employed'),
  Text(10, 0, 'Transport: type 2'),
  Text(11, 0, 'Construction'),
  Text(12, 0, 'Housing'),
  Text(13, 0, 'Kindergarten'),
  Text(14, 0, 'Trade: type 7'),
  Text(15, 0, 'Industry: type 11'),
  Text(16, 0, 'Military'),
  Text(17, 0, 'Services'),
  Text(18, 0, 'Security Ministries'),
  Text(19, 0, 'Transport: type 4'),
  Text(20, 0, 'Industry: type 1'),
  Text(21, 0, 'Emergency'),
  Text(22, 0, 'Security'),
  Text(23, 0, 'Trade: type 2'),
  Text(24, 0, 'University'),
  Text(25, 0, 'Transport: type 3'),
  Text(26, 0, 'Police'),
  Text(27, 0, 'Business Entity Type 1'),
  Text(28, 0, 'Postal'),
  Text(29, 0, 'Agriculture'),
  Text(30, 0, 'Restaurant'),
  Text(31, 0, 'Culture'),
  Text(32, 0, 'Hotel'),
  Text(33, 0, 'Industry: type 7'),
  Text(34, 0, 'Industry: type 3'),
  Text(35, 0, 'Trade: type 3'),
  Text(36, 0, 'Bank'),
  Text(37, 0, 'Industry: type 9'),
  Text(38, 0, 'Trade: type 6'),
  Text(39, 0, 'Industry: type 2'),
  Text(40, 0, 'Transport: type 1'),
  Text(41, 0, 'Industry: type 12'),
  Text(42, 0, 'Insurance'),
  Text(43, 0, 'Industry: type 4'),
  Text(44, 0, 'Mobile'),
  Text(45, 0, 'Trade: type 1'),
  Text(46, 0, 'Industry: type 5'),
  Text(47, 0, 'Industry: type 10'),
```

```
 Text(48, 0, 'Legal Services'),
 Text(49, 0, 'Trade: type 5'),
 Text(50, 0, 'Cleaning'),
 Text(51, 0, 'Industry: type 13'),
 Text(52, 0, 'Trade: type 4'),
 Text(53, 0, 'Telecom'),
 Text(54, 0, 'Industry: type 8'),
 Text(55, 0, 'Realtor'),
 Text(56, 0, 'Advertising'),
 Text(57, 0, 'Industry: type 6')])
```



In [102]:

```python
plt.figure(figsize = (25, 10))
plt.title('Defaulters : Distribution based on ORGANIZATION TYPE')
sns.countplot(defaulters['ORGANIZATION_TYPE'])
plt.xticks(rotation=90)
plt.show()
```



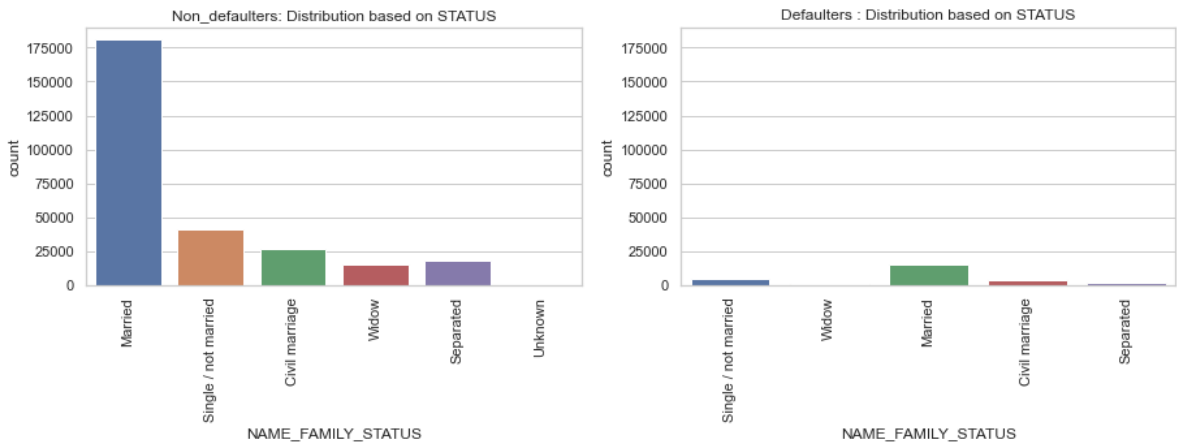
Comments:

1. People from Bussiness Entity Type 3 are most likely to take loans, follwed by Business Entity Type 2 and then Self employed, increasing the risk of having more number of defaulters

2. Least number of defaulters being in Trade-Type 4 and 5 follwed by Industry 8

In [103]:

```
plt.figure(figsize = (15, 8))
plt.subplot(2, 2, 1)
plt.ylim(0,190000)
plt.title('Non_defaulters: Distribution based on STATUS')
sns.countplot(non_defaulters['NAME_FAMILY_STATUS'])
plt.xticks(rotation=90)

plt.subplot(2, 2, 2)
plt.title('Defaulters : Distribution based on STATUS')
plt.ylim(0,190000)
sns.countplot(defaulters['NAME_FAMILY_STATUS'])
plt.xticks(rotation=90)
plt.show()
```

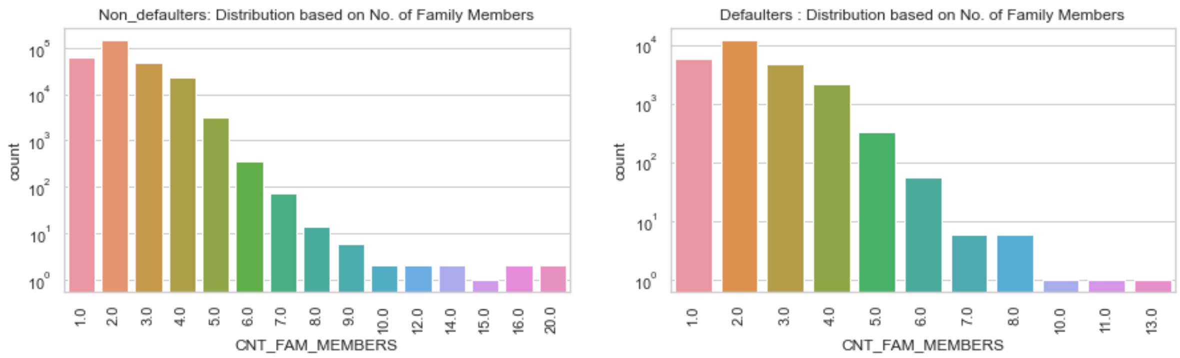


Comments:

1. Married people are most likely to take loans
2. Married people are more likely to be defaulters followed by Single, Civil Marriage and separated
3. Windows are most reliable to give loans

In [104]:

```
plt.figure(figsize = (15, 8))
plt.subplot(2, 2, 1)
plt.yscale("log")
plt.title('Non_defaulters: Distribution based on No. of Family Members')
sns.countplot(non_defaulters['CNT_FAM_MEMBERS'])
plt.xticks(rotation=90)

plt.subplot(2, 2, 2)
plt.title('Defaulters : Distribution based on No. of Family Members')
plt.yscale("log")
sns.countplot(defaulters['CNT_FAM_MEMBERS'])
plt.xticks(rotation=90)
plt.show()
```



In [105]:

```
non_defaulters.CNT_FAM_MEMBERS.value_counts(normalize=True)*100
```

Out[105]:

```
2.0     51.770882
1.0     21.993463
3.0     16.977615
4.0      7.980996
5.0      1.114672
6.0      0.124874
7.0      0.026531
8.0      0.004953
9.0      0.002123
10.0     0.000708
14.0     0.000708
12.0     0.000708
20.0     0.000708
16.0     0.000708
15.0     0.000354
Name: CNT_FAM_MEMBERS, dtype: float64
```

In [106]:

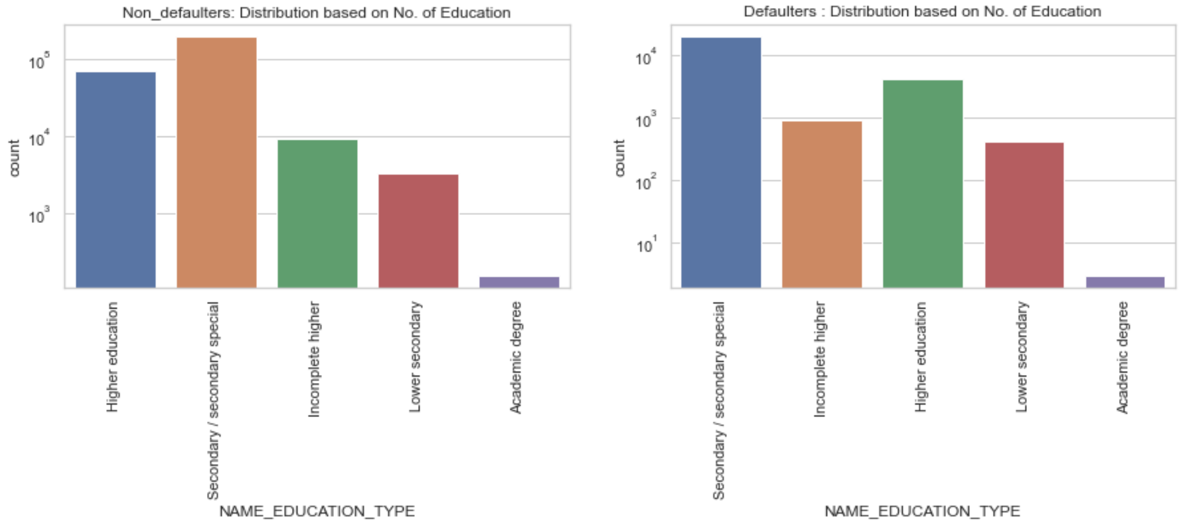
```
defaulters.CNT_FAM_MEMBERS.value_counts(normalize=True)*100
```

Out[106]:

```
2.0     48.374622
1.0     22.860020
3.0     18.561934
4.0      8.604230
5.0      1.317221
6.0      0.221551
7.0      0.024169
8.0      0.024169
10.0     0.004028
13.0     0.004028
11.0     0.004028
Name: CNT_FAM_MEMBERS, dtype: float64
```

In [107]:

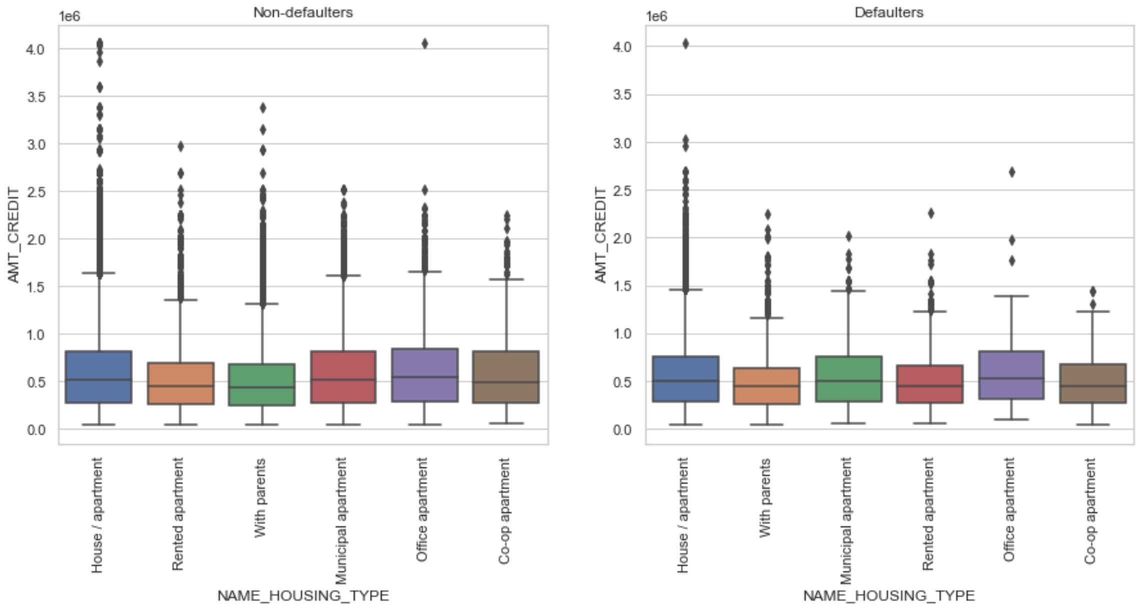
```
plt.figure(figsize = (15, 8))
plt.subplot(2, 2, 1)
plt.yscale("log")
plt.title('Non_defaulters: Distribution based on No. of Education')
sns.countplot(non_defaulters['NAME_EDUCATION_TYPE'])
plt.xticks(rotation=90)

plt.subplot(2, 2, 2)
plt.title('Defaulters : Distribution based on No. of Education')
plt.yscale("log")
sns.countplot(defaulters['NAME_EDUCATION_TYPE'])
plt.xticks(rotation=90)
plt.show()
```



BIVARIATE ANALYSIS

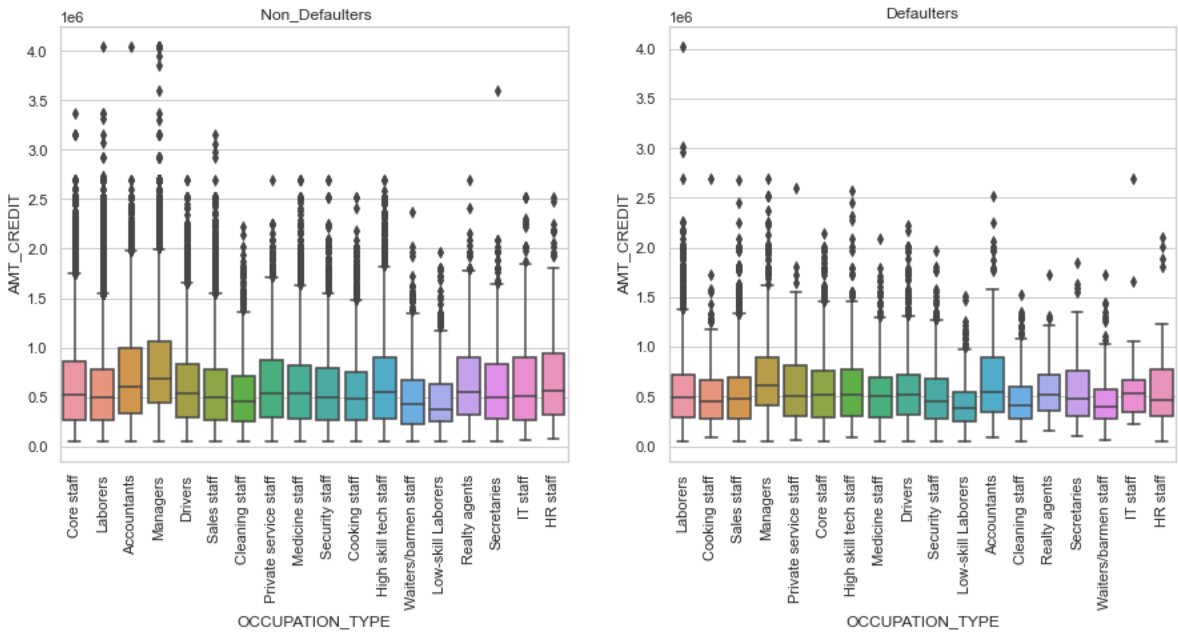
In [108]:

```
plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
plt.title('Non-defaulters')
sns.boxplot(x='NAME_HOUSING_TYPE',y='AMT_CREDIT',data=non_defaulters)
plt.xticks(rotation=90)
plt.subplot(1,2,2)
plt.title('Defaulters')
sns.boxplot(x='NAME_HOUSING_TYPE',y='AMT_CREDIT',data=defaulters)
plt.xticks(rotation=90)
plt.show()
```



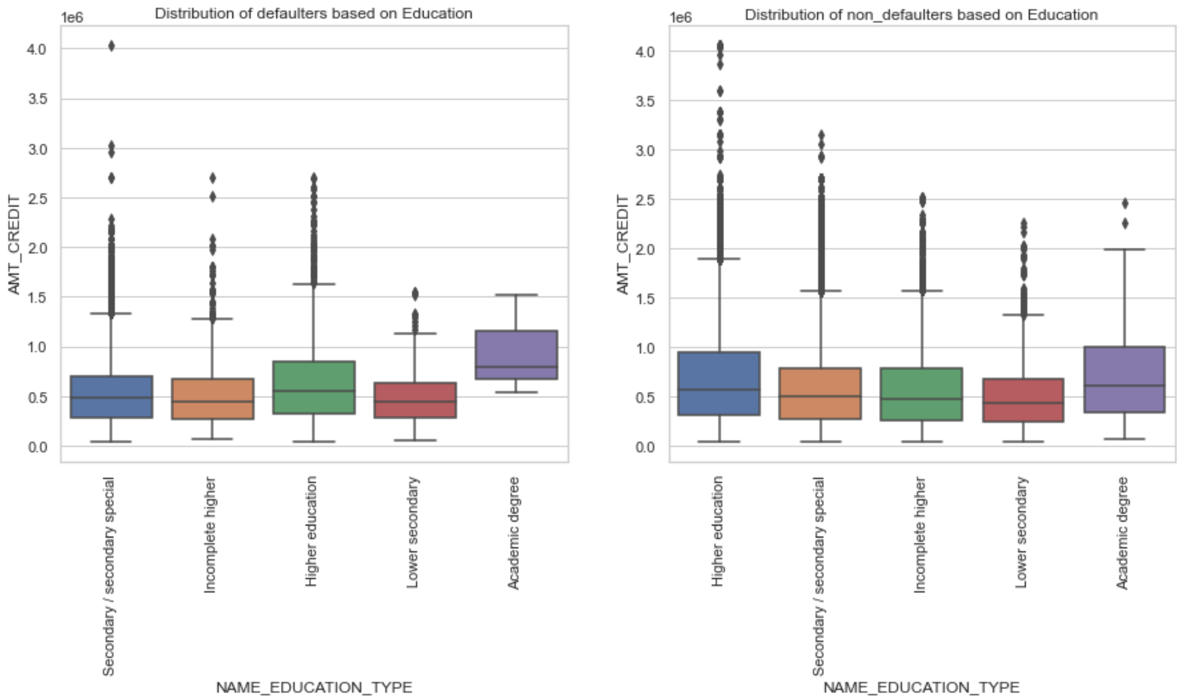
In [109]:

```
plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
plt.title(' Non_Defaulters')
sns.boxplot(x='OCCUPATION_TYPE',y='AMT_CREDIT',data=non_defaulters)
plt.xticks(rotation=90)
plt.subplot(1,2,2)
sns.boxplot(x='OCCUPATION_TYPE',y='AMT_CREDIT',data=defaulters)
plt.title('Defaulters')
plt.xticks(rotation=90)
plt.show()
```



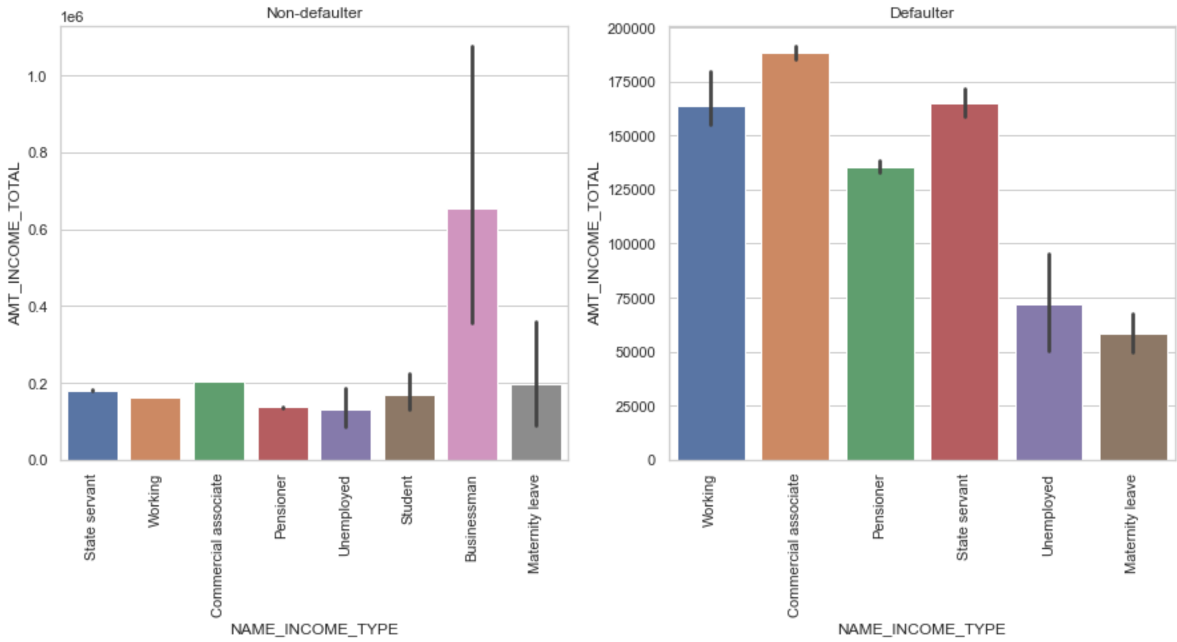
In [110]:

```
plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
plt.title('Distribution of defaulters based on Education')
sns.boxplot(x='NAME_EDUCATION_TYPE',y='AMT_CREDIT',data=defaulters)
plt.xticks(rotation=90)
plt.subplot(1,2,2)
sns.boxplot(x='NAME_EDUCATION_TYPE',y='AMT_CREDIT',data=non_defaulters)
plt.title('Distribution of non_defaulters based on Education')
plt.xticks(rotation=90)
plt.show()
```



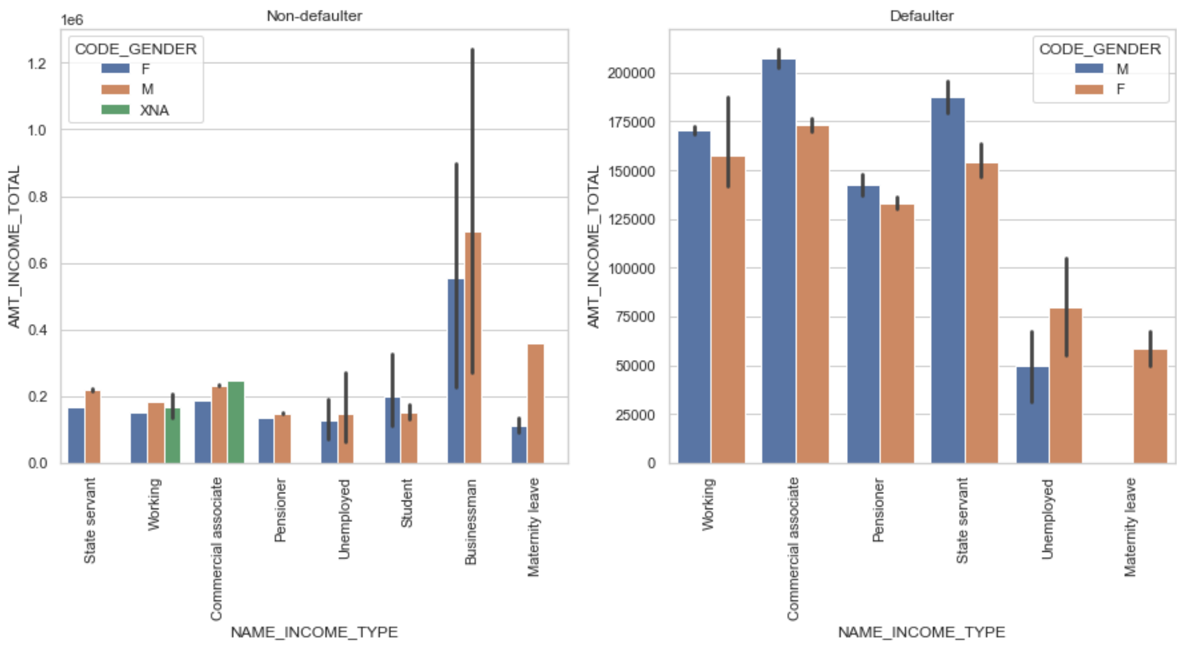
In [111]:

```
plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
plt.title('Non-defaulter')
sns.barplot(x='NAME_INCOME_TYPE',y='AMT_INCOME_TOTAL',data=non_defaulters)
plt.xticks(rotation=90)

plt.subplot(1,2,2)
sns.barplot(x='NAME_INCOME_TYPE',y='AMT_INCOME_TOTAL',data=defaulters)
plt.title('Defaulter')
plt.xticks(rotation=90)
plt.show()
```



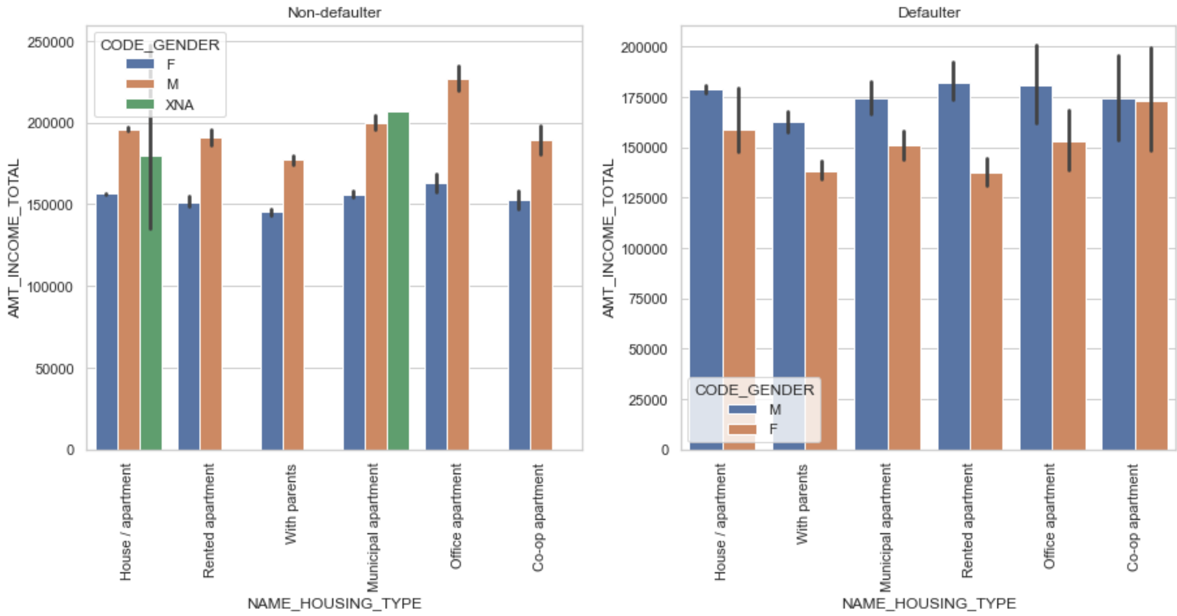
In [112]:

```
plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
plt.title('Non-defaulter')
sns.barplot(x='NAME_INCOME_TYPE',y='AMT_INCOME_TOTAL',hue="CODE_GENDER",data=non_defaulters
plt.xticks(rotation=90)

plt.subplot(1,2,2)
sns.barplot(x='NAME_INCOME_TYPE',y='AMT_INCOME_TOTAL',hue="CODE_GENDER",data=defaulters)
plt.title('Defaulter')
plt.xticks(rotation=90)
plt.show()
```



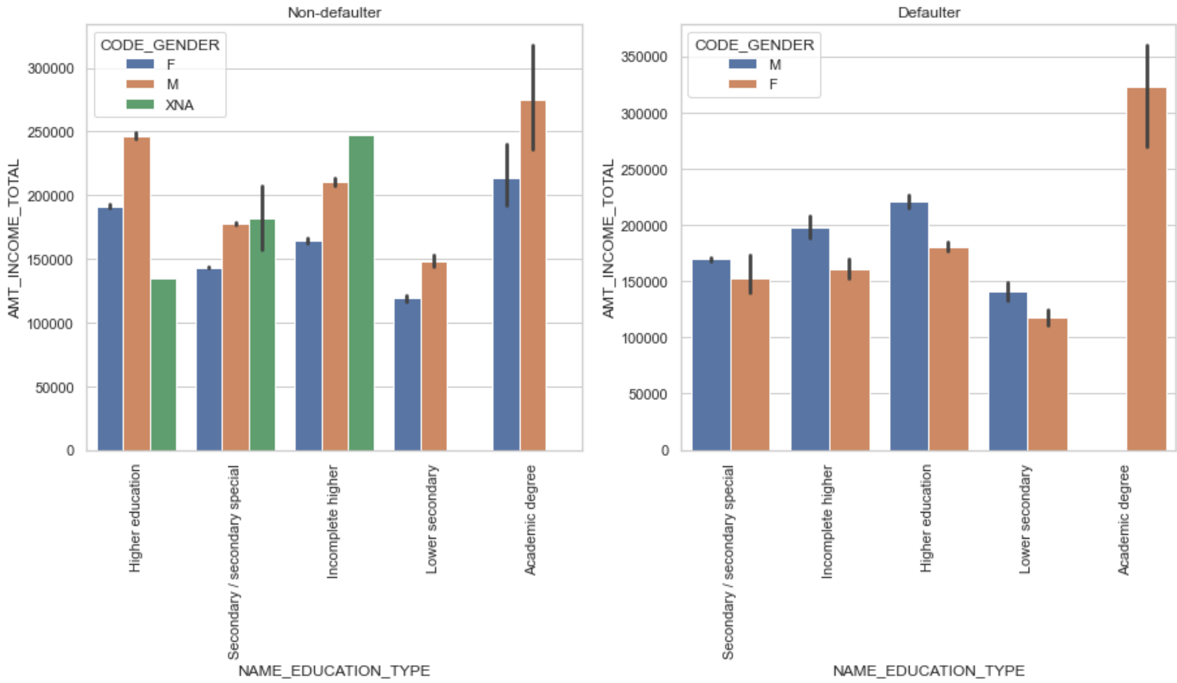
In [121]:

```
plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
plt.title('Non-defaulter')
sns.barplot(x='NAME_HOUSING_TYPE',y='AMT_INCOME_TOTAL',hue="CODE_GENDER",data=non_defaulter
plt.xticks(rotation=90)

plt.subplot(1,2,2)
sns.barplot(x='NAME_HOUSING_TYPE',y='AMT_INCOME_TOTAL',hue="CODE_GENDER",data=defaulters)
plt.title('Defaulter')
plt.xticks(rotation=90)
plt.show()
```



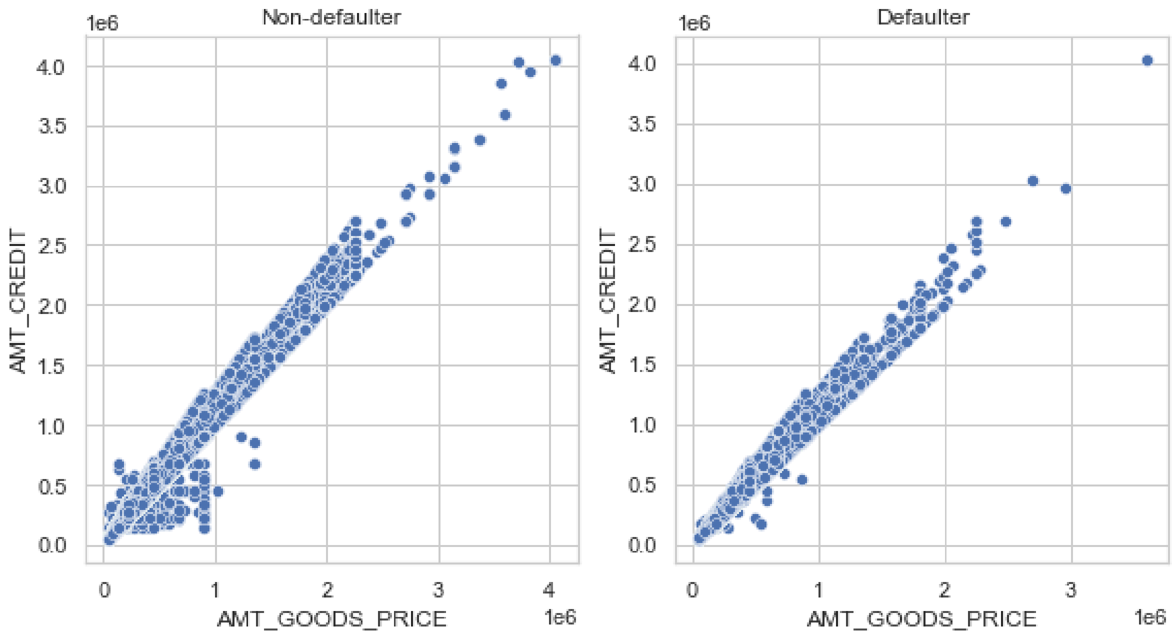
In [113]:

```
plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
plt.title('Non-defaulter')
sns.barplot(x='NAME_EDUCATION_TYPE',y='AMT_INCOME_TOTAL',hue="CODE_GENDER",data=non_default
plt.xticks(rotation=90)

plt.subplot(1,2,2)
sns.barplot(x='NAME_EDUCATION_TYPE',y='AMT_INCOME_TOTAL',hue="CODE_GENDER",data=defaulters)
plt.title('Defaulter')
plt.xticks(rotation=90)
plt.show()
```



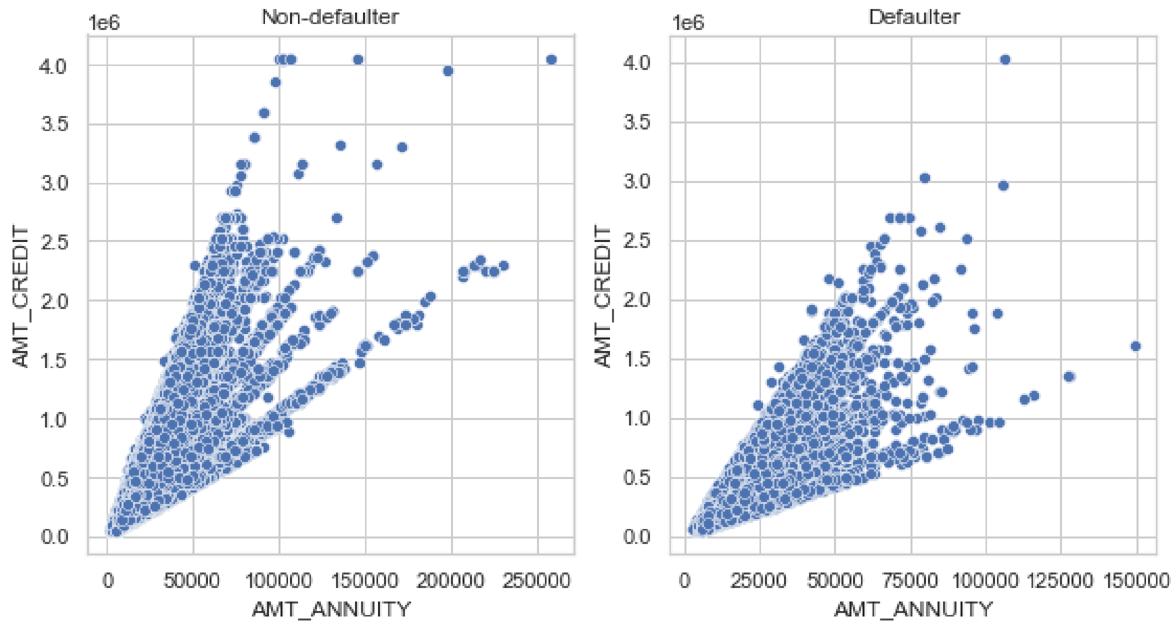
In [114]:

```
fig = plt.figure(figsize=(10,5))

ax0 = fig.add_subplot(1, 2, 1, title="Non-defaulter")
ax1 = fig.add_subplot(1, 2, 2, title="Defaulter")

sns.scatterplot(app[app["TARGET"] == 0]['AMT_GOODS_PRICE'], app[app["TARGET"] == 0]['AMT_CR
sns.scatterplot(app[app["TARGET"] == 1]['AMT_GOODS_PRICE'], app[app["TARGET"] == 1]['AMT_CR
plt.show()
```



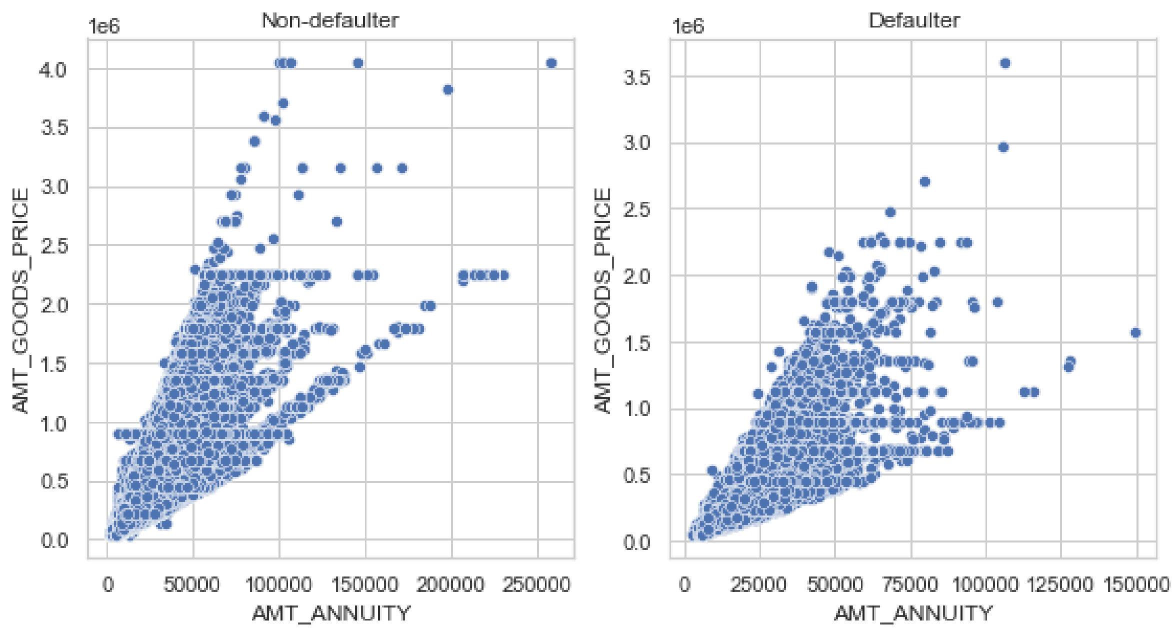
In [115]:

```
fig = plt.figure(figsize=(10,5))

ax0 = fig.add_subplot(1, 2, 1, title="Non-defaulter")
ax1 = fig.add_subplot(1, 2, 2, title="Defaulter")

sns.scatterplot(app[app["TARGET"] == 0]['AMT_ANNUITY'], app[app["TARGET"] == 0]['AMT_CREDIT
sns.scatterplot(app[app["TARGET"] == 1]['AMT_ANNUITY'], app[app["TARGET"] == 1]['AMT_CREDIT
plt.show()
```



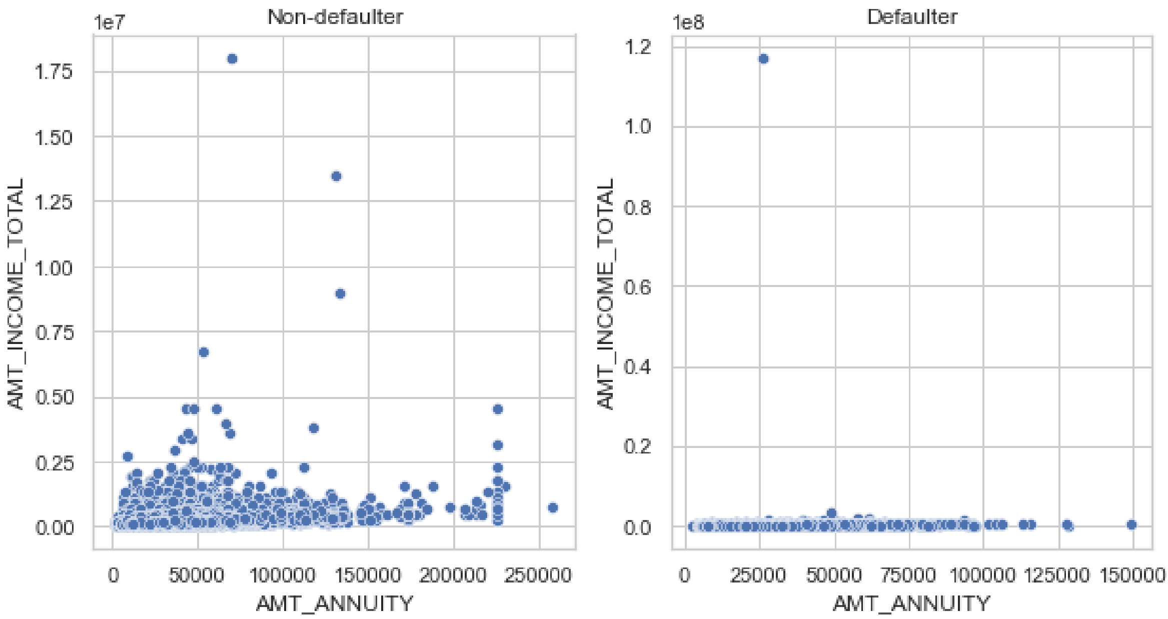
In [116]:

```
fig = plt.figure(figsize=(10,5))

ax0 = fig.add_subplot(1, 2, 1, title="Non-defaulter")
ax1 = fig.add_subplot(1, 2, 2, title="Defaulter")

sns.scatterplot(app[app["TARGET"] == 0]['AMT_ANNUITY'], app[app["TARGET"] == 0]['AMT_GOODS_
sns.scatterplot(app[app["TARGET"] == 1]['AMT_ANNUITY'], app[app["TARGET"] == 1]['AMT_GOODS_
plt.show()
```



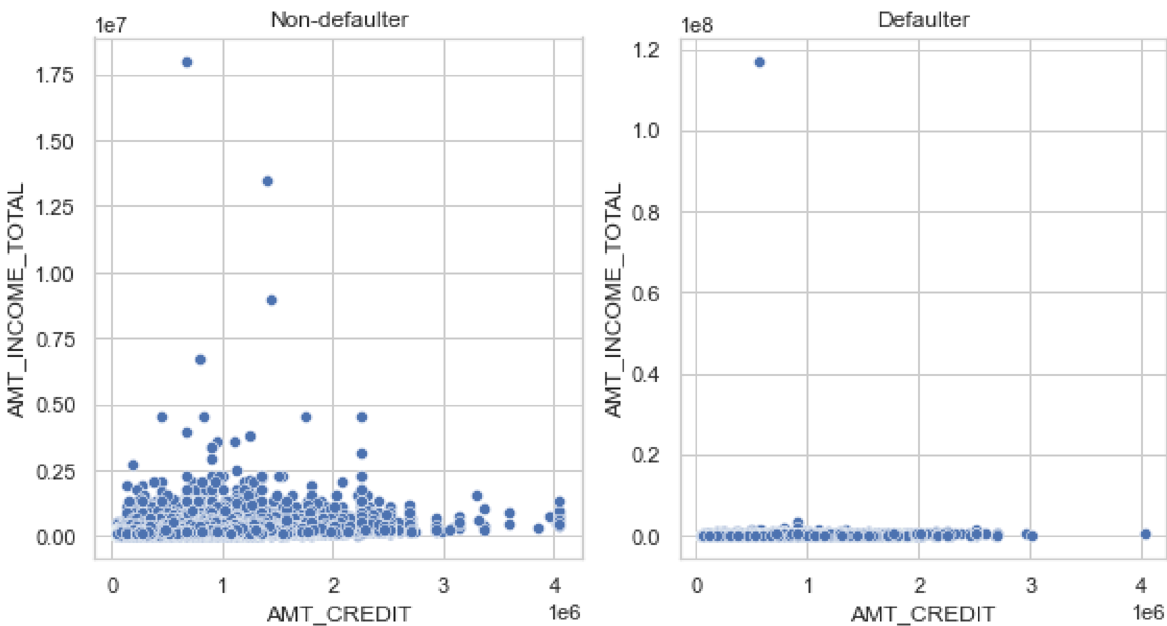
In [117]:

```
fig = plt.figure(figsize=(10,5))

ax0 = fig.add_subplot(1, 2, 1, title="Non-defaulter")
ax1 = fig.add_subplot(1, 2, 2, title="Defaulter")

sns.scatterplot(app[app["TARGET"] == 0]['AMT_ANNUITY'], app[app["TARGET"] == 0]['AMT_INCOME
sns.scatterplot(app[app["TARGET"] == 1]['AMT_ANNUITY'], app[app["TARGET"] == 1]['AMT_INCOME
plt.show()
```



In [118]:

```
fig = plt.figure(figsize=(10,5))

ax0 = fig.add_subplot(1, 2, 1, title="Non-defaulter")
ax1 = fig.add_subplot(1, 2, 2, title="Defaulter")

sns.scatterplot(app[app["TARGET"] == 0]['AMT_CREDIT'], app[app["TARGET"] == 0]['AMT_INCOME_
sns.scatterplot(app[app["TARGET"] == 1]['AMT_CREDIT'], app[app["TARGET"] == 1]['AMT_INCOME_
plt.show()
```



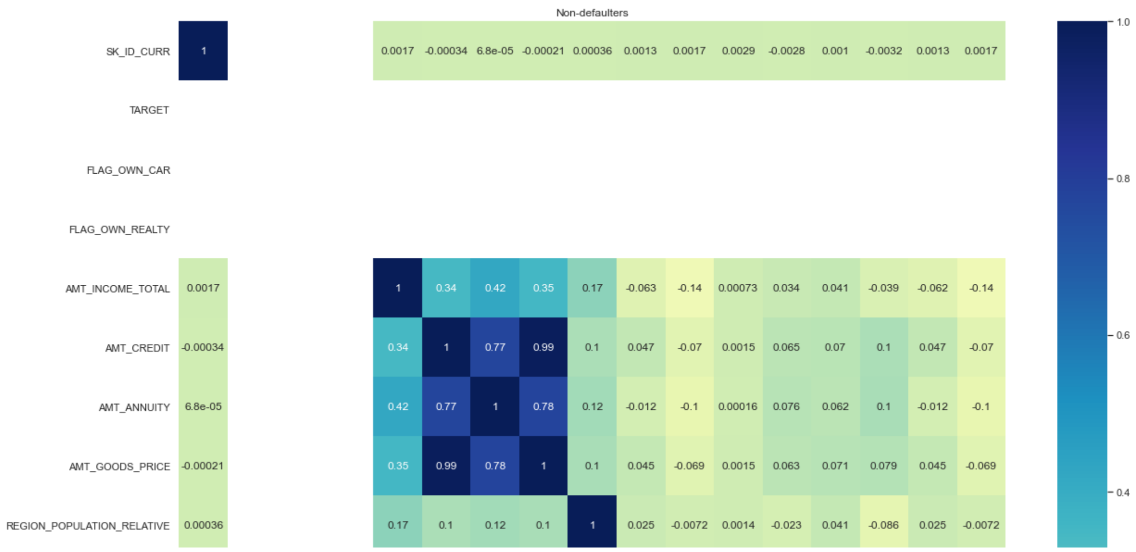
NUMERICAL ANALYSIS

In [119]:

```
##Getting correlation bewteen numerical data for Non_defaulters
plt.figure(figsize=(20,20))
plt.title("Non-defaulters")
corr_nd= non_defaulters.corr()
sns.heatmap(corr_nd, annot=True,cmap="YlGnBu")
```

Out[119]:

```
<AxesSubplot:title={'center':'Non-defaulters'}>
```



Inferences: Important Correlating factors amongst non-defaulters

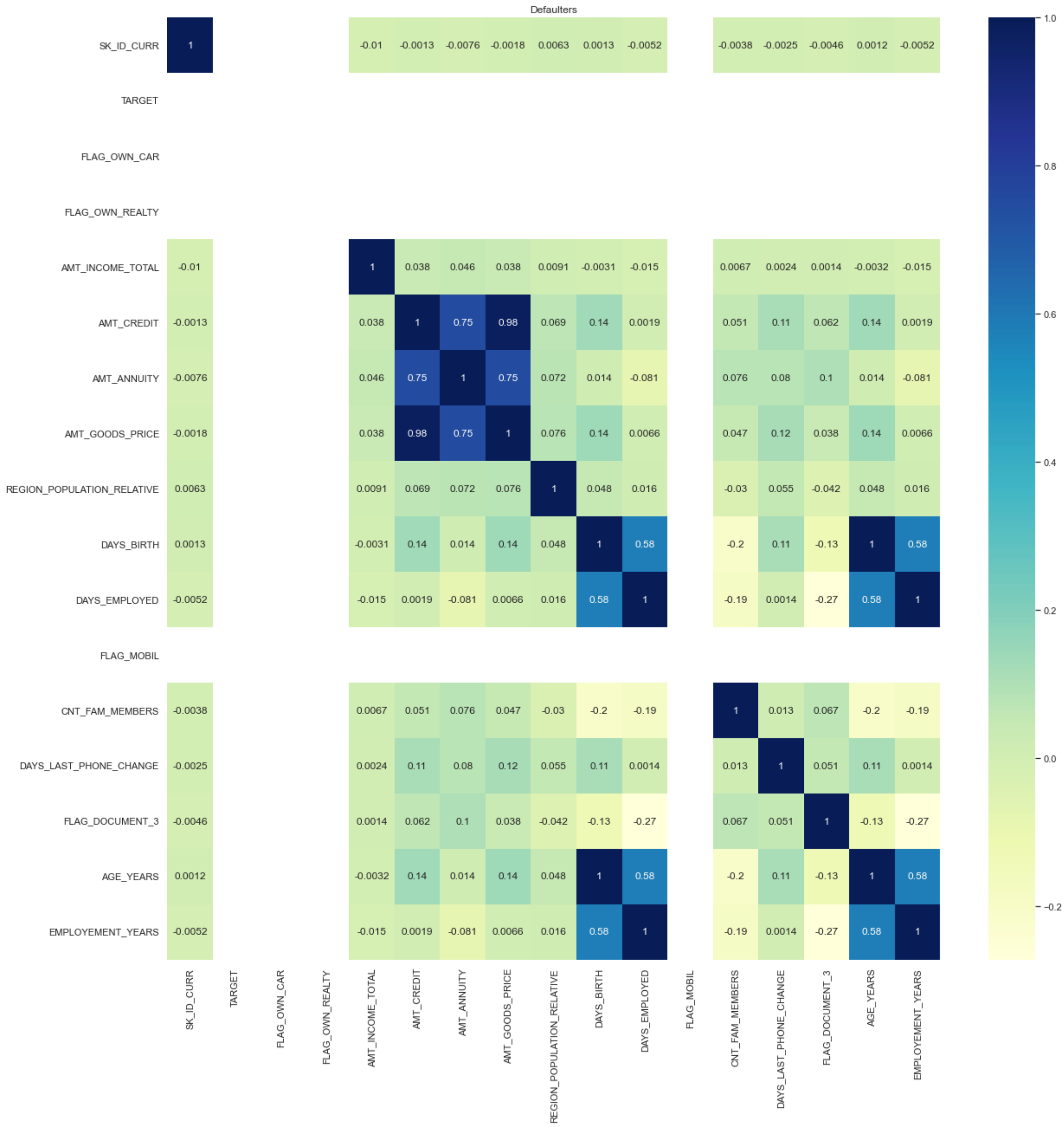
1.Credit amount is highly correlated with:Goods Price Amount(0.98), Loan Annuity(0.75), Income (0.34)
2.Income has high correlation with: Credit amount(0.34), Annuity amount(0.42) and Goods Price Amount(0.35)

In [120]:

```
##Getting correlation bewteen numerical data for Non_defaulters
plt.figure(figsize=(20,20))
plt.title("Defaulters")
corr_d= defaulters.corr()
sns.heatmap(corr_d, annot=True,cmap="YlGnBu")
```

Out[120]:

```
<AxesSubplot:title={'center':'Defaulters'}>
```



Inferences: Important Correlating factors amongst defaulters

1. Credit amount shows highly correlation with: Goods Price Amount(0.98) & Loan Annuity(0.75)

In [ ]:

```

```

Cleaning previous_application.csv

In [121]:

```
## Checking missing data in previous_application.csv
pre_null= (100*pre.isnull().sum()/len(pre)).round(2)
pre_null
```

Out[121]:

```
SK_ID_PREV                      0.00
SK_ID_CURR                      0.00
NAME_CONTRACT_TYPE              0.00
AMT_ANNUITY                    22.29
AMT_APPLICATION                 0.00
AMT_CREDIT                      0.00
AMT_DOWN_PAYMENT               53.64
AMT_GOODS_PRICE                23.08
WEEKDAY_APPR_PROCESS_START      0.00
HOUR_APPR_PROCESS_START         0.00
FLAG_LAST_APPL_PER_CONTRACT     0.00
NFLAG_LAST_APPL_IN_DAY          0.00
RATE_DOWN_PAYMENT              53.64
RATE_INTEREST_PRIMARY          99.64
RATE_INTEREST_PRIVILEGED       99.64
NAME_CASH_LOAN_PURPOSE          0.00
NAME_CONTRACT_STATUS            0.00
DAYS_DECISION                   0.00
NAME_PAYMENT_TYPE               0.00
CODE_REJECT_REASON              0.00
NAME_TYPE_SUITE                49.12
NAME_CLIENT_TYPE                0.00
NAME_GOODS_CATEGORY             0.00
NAME_PORTFOLIO                  0.00
NAME_PRODUCT_TYPE               0.00
CHANNEL_TYPE                    0.00
SELLERPLACE_AREA                0.00
NAME_SELLER_INDUSTRY            0.00
CNT_PAYMENT                    22.29
NAME_YIELD_GROUP                0.00
PRODUCT_COMBINATION             0.02
DAYS_FIRST_DRAWING             40.30
DAYS_FIRST_DUE                 40.30
DAYS_LAST_DUE_1ST_VERSION      40.30
DAYS_LAST_DUE                  40.30
DAYS_TERMINATION               40.30
NFLAG_INSURED_ON_APPROVAL      40.30
dtype: float64
```

In [131]:

```
# Displaying missing columns with value >0 in descending order
pre_null=pre_null[pre_null.values>0].sort_values(ascending=False)
len(pre_null)
```

Out[131]:

```
15
```

Data Cleaning-Dropping & Imputing Missing Values

In [122]:

```
# Dropping columns with missing values >45 %
pre_null=pre_null[pre_null.values>45]
pre_null=list(pre_null.index)
pre.drop(labels=pre_null,axis=1,inplace=True)
pre.shape
```

Out[122]:

```
(1670214, 32)
```

In [123]:

```
pre_null=(100*pre.isnull().sum()/len(pre)).round(2)
pre_null[pre_null.values>0].sort_values(ascending=False)
```

Out[123]:

```
DAYS_FIRST_DRAWING           40.30
DAYS_FIRST_DUE               40.30
DAYS_LAST_DUE_1ST_VERSION    40.30
DAYS_LAST_DUE                40.30
DAYS_TERMINATION             40.30
NFLAG_INSURED_ON_APPROVAL    40.30
AMT_GOODS_PRICE              23.08
AMT_ANNUITY                  22.29
CNT_PAYMENT                  22.29
PRODUCT_COMBINATION           0.02
dtype: float64
```

In [124]:

```
# converting Negative columns to positive columns
pre.drop(['DAYS_DECISION','SELLERPLACE_AREA','DAYS_FIRST_DUE','DAYS_LAST_DUE_1ST_VERSION','
pre.columns
```

Out[124]:

```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_ANNUITY',
       'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_GOODS_PRICE',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       'FLAG_LAST_APPL_PER_CONTRACT', 'NFLAG_LAST_APPL_IN_DAY',
       'NAME_CASH_LOAN_PURPOSE', 'NAME_CONTRACT_STATUS', 'NAME_PAYMENT_TYP
E',
       'CODE_REJECT_REASON', 'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY',
       'NAME_PORTFOLIO', 'NAME_PRODUCT_TYPE', 'CNT_PAYMENT',
       'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION'],
      dtype='object')
```

In [125]:

```
pre.shape
```

Out[125]:

```
(1670214, 22)
```

In [126]:

```
#Missing values in AMT_GOODS_PRICE could be imputed by mean value for this var since this i
agp_mean=pre['AMT_GOODS_PRICE'].mean()
pre['AMT_GOODS_PRICE'].fillna(agp_mean, inplace=True)
```

In [127]:

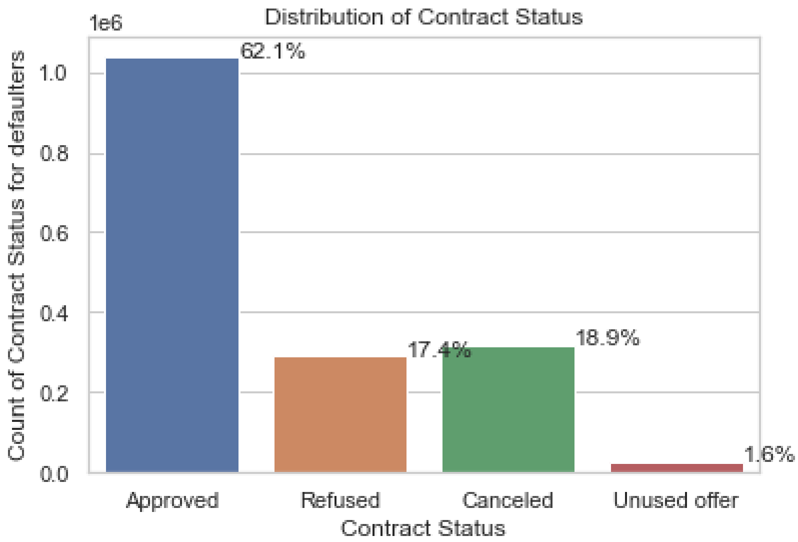
```
pre['AMT_GOODS_PRICE'].isnull().sum()
```

Out[127]:

0

In [128]:

```
ax=sns.countplot(pre.NAME_CONTRACT_STATUS)
plt.xlabel("Contract Status")
plt.ylabel("Count of Contract Status for defaulters")
plt.title("Distribution of Contract Status")
total = float(len(pre))
for p in ax.patches:
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate('{:.1f}%'.format(100 * p.get_height()/total),(x,y))
plt.show()
```

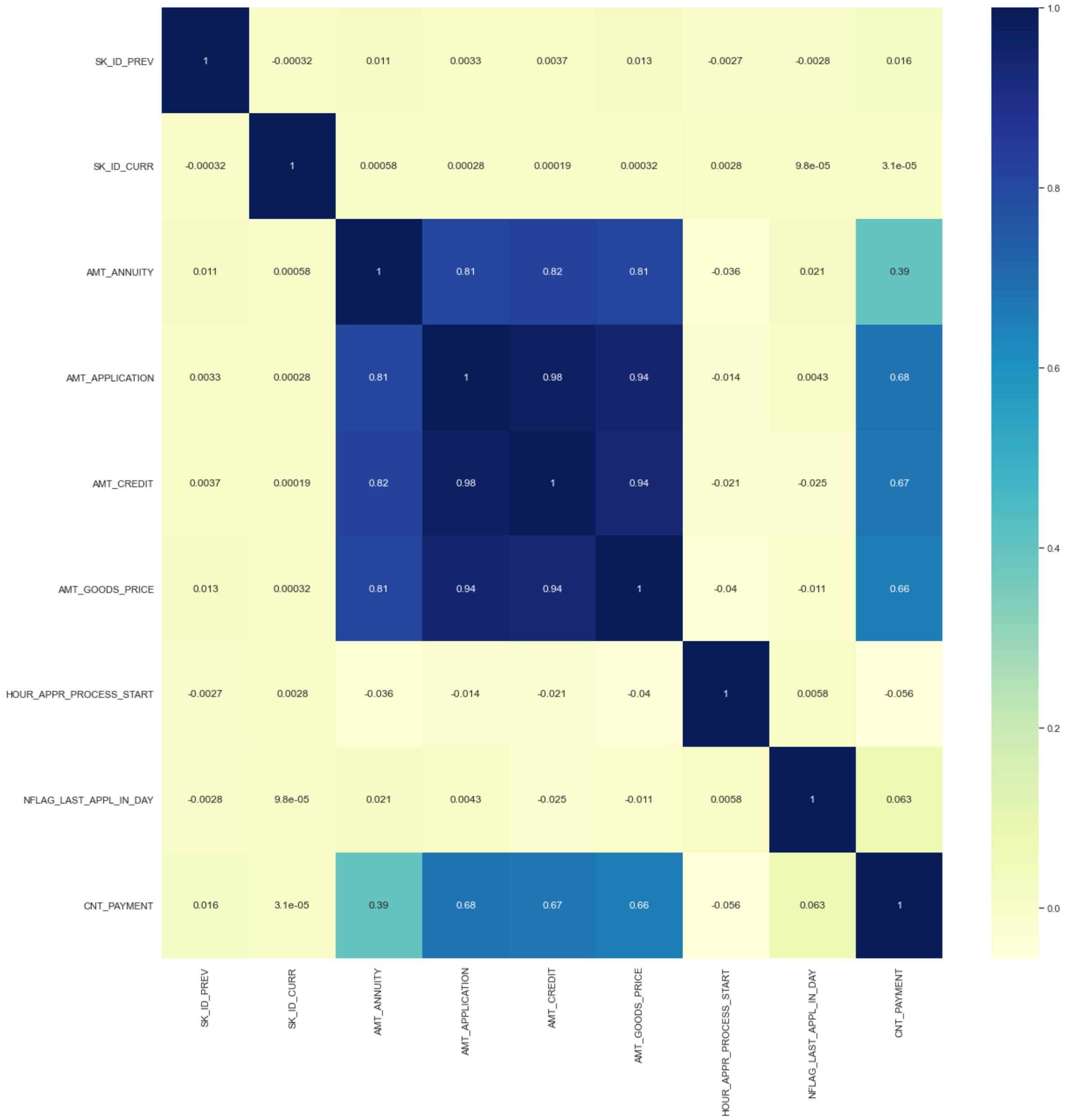


In [139]:

```
plt.figure(figsize=(20,20))
sns.heatmap(pre.corr(), annot=True,cmap="YlGnBu")
```

Out[139]:

<AxesSubplot:>



MERGING DATASETS TO CHECK DEFAULTER STATUS

In [129]:

```
#Merge the previous_application.csv with the defaulter data file
combined= pd.merge(non_defaulters, pre, how='left', on='SK_ID_CURR',suffixes=('_Cur', '_Pre
combined.head()
```

Out[129]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_Cur	CODE_GENDER	FLAG_OWN_CAR	FL
0	100003	0	Cash loans	F	True	
1	100003	0	Cash loans	F	True	
2	100003	0	Cash loans	F	True	
3	100004	0	Revolving loans	M	True	
4	100006	0	Cash loans	F	True	

5 rows × 51 columns

In [141]:

```
combined.describe()
```

Out[141]:

	SK_ID_CURR	TARGET	AMT_INCOME_TOTAL	AMT_CREDIT_Cur	AMT_ANNUITY_Cur	A
count	1.306815e+06	1306815.0	1.306815e+06	1.306815e+06	1.306722e+06	
mean	2.785087e+05	0.0	1.737656e+05	5.922285e+05	2.707761e+04	
std	1.028074e+05	0.0	1.049183e+05	3.911268e+05	1.418544e+04	
min	1.000030e+05	0.0	2.565000e+04	4.500000e+04	1.615500e+03	
25%	1.893400e+05	0.0	1.125000e+05	2.700000e+05	1.666800e+04	
50%	2.790130e+05	0.0	1.575000e+05	5.094000e+05	2.488050e+04	
75%	3.676080e+05	0.0	2.115000e+05	8.086500e+05	3.459600e+04	
max	4.562550e+05	0.0	1.800009e+07	4.050000e+06	2.580255e+05	

8 rows × 22 columns

In [130]:

```
combined.columns
```

Out[130]:

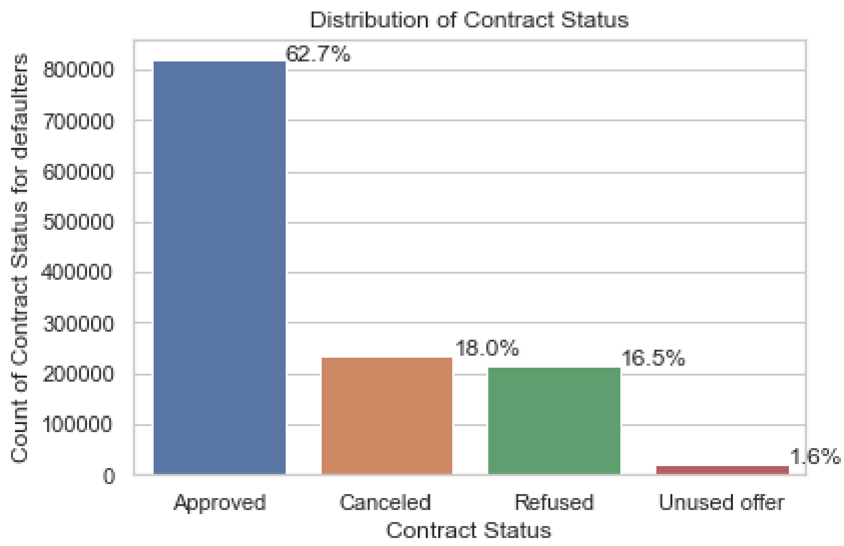
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE_Cur', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'AMT_INCOME_TOTAL', 'AMT_CREDIT_Cu
r',
       'AMT_ANNUITY_Cur', 'AMT_GOODS_PRICE_Cur', 'NAME_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'FLAG_MOBIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS',
       'ORGANIZATION_TYPE', 'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_3',
       'AGE_YEARS', 'AGE_GROUP', 'EMPLOYEMENT_YEARS',
       'EMPLOYEMENT_YEARS_GROUP', 'INCOME_RANGE', 'CREDIT_RANGE', 'SK_ID_PRE
V',
       'NAME_CONTRACT_TYPE_Pre', 'AMT_ANNUITY_Pre', 'AMT_APPLICATION',
       'AMT_CREDIT_Pre', 'AMT_GOODS_PRICE_Pre', 'WEEKDAY_APPR_PROCESS_STAR
T',
       'HOUR_APPR_PROCESS_START', 'FLAG_LAST_APPL_PER_CONTRACT',
       'NFLAG_LAST_APPL_IN_DAY', 'NAME_CASH_LOAN_PURPOSE',
       'NAME_CONTRACT_STATUS', 'NAME_PAYMENT_TYPE', 'CODE_REJECT_REASON',
       'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO',
       'NAME_PRODUCT_TYPE', 'CNT_PAYMENT', 'NAME_YIELD_GROUP',
       'PRODUCT_COMBINATION'],
      dtype='object')
```

In [131]:

```
ax=sns.countplot(combined.NAME_CONTRACT_STATUS)
total = float(len(combined))
for p in ax.patches:
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate('{:.1f}%'.format(100 * p.get_height()/total),(x,y))
plt.xlabel("Contract Status")
plt.ylabel("Count of Contract Status for defaulters")
plt.title("Distribution of Contract Status")
```

Out[131]:

```
Text(0.5, 1.0, 'Distribution of Contract Status')
```



In [132]:

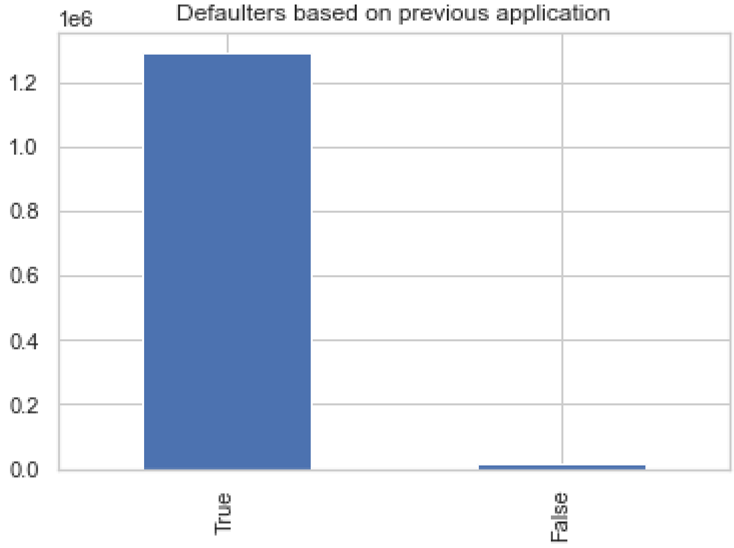
```
combined['SK_ID_PREV'].notna().value_counts()
```

Out[132]:

```
True     1291341
False      15474
Name: SK_ID_PREV, dtype: int64
```

In [133]:

```
combined['SK_ID_PREV'].notna().value_counts().plot(kind='bar')
plt.title("Defaulters based on previous application")
plt.show()
```



Comments: 1291341 number were defaulters and 15474