In [1]:
```
#import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#load the dataset
#application data
data = pd.read_csv(r"C:\Users\mani ganesh\Desktop\loan data set\application_data.csv")

#previous application data
pdata = pd.read_csv(r"C:\Users\mani ganesh\Desktop\loan data set\previous_application.csv")

#columns description
descritption = pd.read_csv(r"C:\Users\mani ganesh\Desktop\loan data set\columns_description.csv
```

INSPECING APPLICATION DATA

In [2]:
```
data.shape
```

Out[2]: (307511, 122)

There 122 columns in this dataset its a lot so we start with checking the null values of each column

In [3]:
```
missing = pd.DataFrame(data.isna().sum().sort_values(ascending = False))
missing
```

Out[3]:

	0
COMMONAREA_MEDI	214865
COMMONAREA_AVG	214865
COMMONAREA_MODE	214865
NONLIVINGAPARTMENTS_MODE	213514
NONLIVINGAPARTMENTS_AVG	213514
...	...
NAME_HOUSING_TYPE	0
NAME_FAMILY_STATUS	0
NAME_EDUCATION_TYPE	0
NAME_INCOME_TYPE	0
SK_ID_CURR	0

122 rows × 1 columns

We can reset the index and rename the column name and add new column as percent that have percentage of null values in each column

In [4]:
```
missing.reset_index(inplace=True)
missing.head()
```

Out[4]:

	index	0
0	COMMONAREA_MEDI	214865
1	COMMONAREA_AVG	214865
2	COMMONAREA_MODE	214865
3	NONLIVINGAPARTMENTS_MODE	213514
4	NONLIVINGAPARTMENTS_AVG	213514

In [5]:
```
missing.rename(columns={'index':'column',0:'null_count'},inplace=True)
missing['percent'] = missing['null_count']/data.shape[0]*100
missing.head()
```

Out[5]:

	column	null_count	percent
0	COMMONAREA_MEDI	214865	69.872297
1	COMMONAREA_AVG	214865	69.872297
2	COMMONAREA_MODE	214865	69.872297
3	NONLIVINGAPARTMENTS_MODE	213514	69.432963
4	NONLIVINGAPARTMENTS_AVG	213514	69.432963

Calculate the number of columns that has missing percentage more than 40

In [6]:
```
missing[missing.percent>40]
```

Out[6]:

	column	null_count	percent
0	COMMONAREA_MEDI	214865	69.872297
1	COMMONAREA_AVG	214865	69.872297
2	COMMONAREA_MODE	214865	69.872297
3	NONLIVINGAPARTMENTS_MODE	213514	69.432963
4	NONLIVINGAPARTMENTS_AVG	213514	69.432963
5	NONLIVINGAPARTMENTS_MEDI	213514	69.432963
6	FONDKAPREMONT_MODE	210295	68.386172
7	LIVINGAPARTMENTS_MODE	210199	68.354953
8	LIVINGAPARTMENTS_AVG	210199	68.354953
9	LIVINGAPARTMENTS_MEDI	210199	68.354953
10	FLOORSMIN_AVG	208642	67.848630
11	FLOORSMIN_MODE	208642	67.848630
12	FLOORSMIN_MEDI	208642	67.848630
13	YEARS_BUILD_MEDI	204488	66.497784
14	YEARS_BUILD_MODE	204488	66.497784
15	YEARS_BUILD_AVG	204488	66.497784
16	OWN_CAR_AGE	202929	65.990810
17	LANDAREA_MEDI	182590	59.376738
18	LANDAREA_MODE	182590	59.376738
19	LANDAREA_AVG	182590	59.376738
20	BASEMENTAREA_MEDI	179943	58.515956
21	BASEMENTAREA_AVG	179943	58.515956
22	BASEMENTAREA_MODE	179943	58.515956
23	EXT_SOURCE_1	173378	56.381073
24	NONLIVINGAREA_MODE	169682	55.179164
25	NONLIVINGAREA_AVG	169682	55.179164
26	NONLIVINGAREA_MEDI	169682	55.179164
27	ELEVATORS_MEDI	163891	53.295980
28	ELEVATORS_AVG	163891	53.295980
29	ELEVATORS_MODE	163891	53.295980
30	WALLSMATERIAL_MODE	156341	50.840783
31	APARTMENTS_MEDI	156061	50.749729
32	APARTMENTS_AVG	156061	50.749729
33	APARTMENTS_MODE	156061	50.749729
34	ENTRANCES_MEDI	154828	50.348768
35	ENTRANCES_AVG	154828	50.348768
36	ENTRANCES_MODE	154828	50.348768
37	LIVINGAREA_AVG	154350	50.193326
38	LIVINGAREA_MODE	154350	50.193326
39	LIVINGAREA_MEDI	154350	50.193326
40	HOUSETYPE_MODE	154297	50.176091
41	FLOORSMAX_MODE	153020	49.760822
42	FLOORSMAX_MEDI	153020	49.760822
43	FLOORSMAX_AVG	153020	49.760822
44	YEARS_BEGINEXPLUATATION_MODE	150007	48.781019
45	YEARS_BEGINEXPLUATATION_MEDI	150007	48.781019

	column	null_count	percent
46	YEARS_BEGINEXPLUATATION_AVG	150007	48.781019
47	TOTALAREA_MODE	148431	48.268517
48	EMERGENCYSTATE_MODE	145755	47.398304

There are 49 columns that has missing values more than 40%. I believe this kind of data will not make much sense even when imputation, so we can remove those columns

In [7]:
```
data.columns
```

Out[7]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY',
       ...
       'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20',
       'FLAG_DOCUMENT_21', 'AMT_REQ_CREDIT_BUREAU_HOUR',
       'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
       'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT',
       'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object', length=122)
```

In [8]:
```
data.drop(missing[missing.percent>40]['column'].values,axis=1,inplace=True)
```

In [9]:
```
data.shape
```

Out[9]:
```
(307511, 73)
```

the columns has been reduced from 122 to 73 columns we removed 49 columns

INVESTIGATING THE COLUMNS

In [10]:
```
data.columns
```

Out[10]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'FLAG_MOBIL',
       'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE',
       'FLAG_EMAIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS',
       'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY',
       'ORGANIZATION_TYPE', 'EXT_SOURCE_2', 'EXT_SOURCE_3',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3',
       'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6',
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9',
       'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12',
       'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15',
       'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18',
       'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')
```

we can see there are lots of column starting with the FLAG, so we can investigate them

In [11]:
```
cols_with_flag = data.columns[data.columns.str.startswith('FLAG')]
cols_with_flag
```

Out[11]:
```
Index(['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'],
      dtype='object')
```

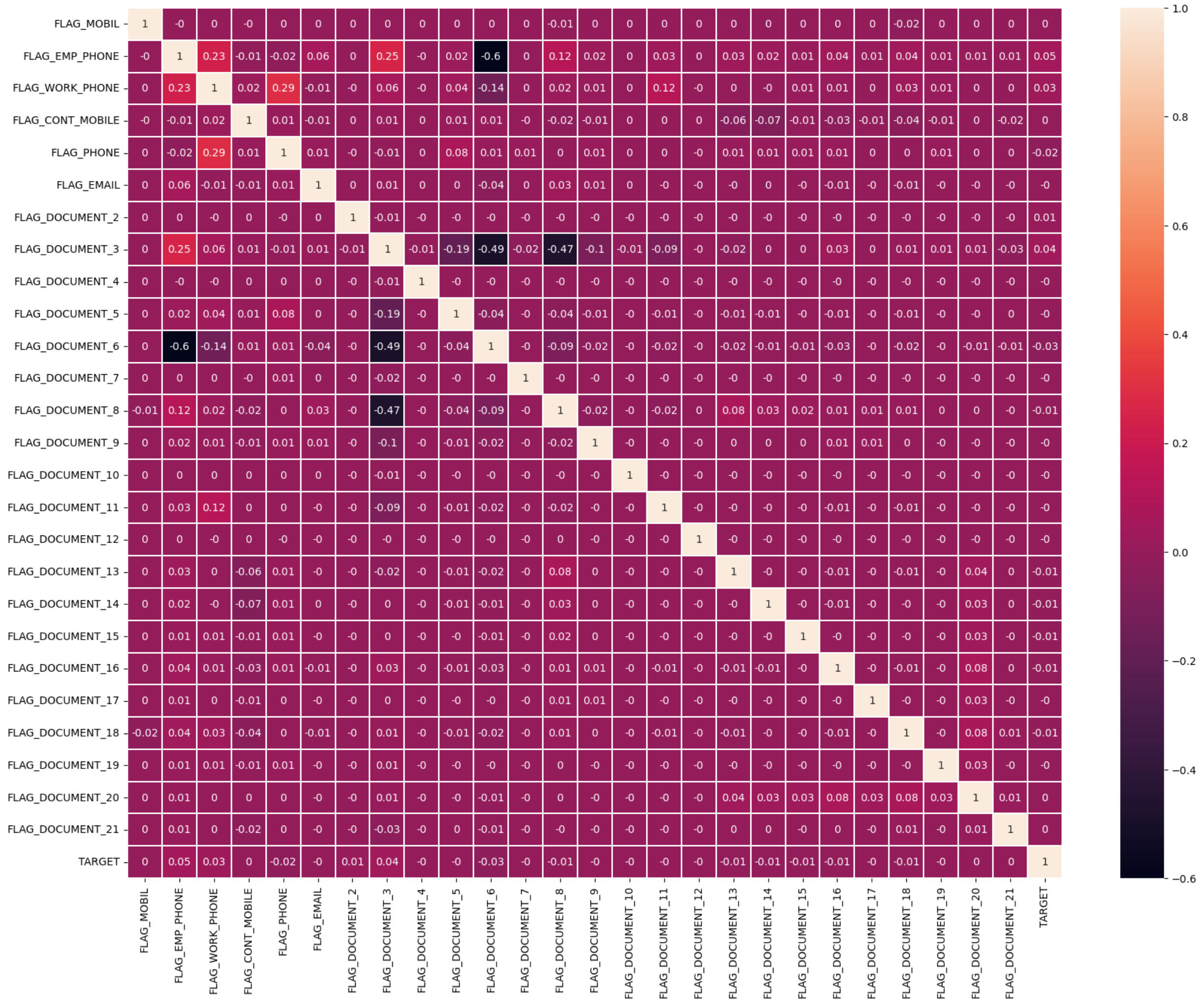
In [12]:
```
#concatenating the cols_with_flag and TARGET column from data
flags_cols_data = data[np.concatenate([cols_with_flag,np.array(['TARGET'])])]
flags_cols_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 29 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   FLAG_OWN_CAR      307511 non-null  object
 1   FLAG_OWN_REALTY   307511 non-null  object
 2   FLAG_MOBIL        307511 non-null  int64 
 3   FLAG_EMP_PHONE    307511 non-null  int64 
 4   FLAG_WORK_PHONE   307511 non-null  int64 
 5   FLAG_CONT_MOBILE  307511 non-null  int64 
 6   FLAG_PHONE        307511 non-null  int64 
 7   FLAG_EMAIL        307511 non-null  int64 
 8   FLAG_DOCUMENT_2   307511 non-null  int64 
 9   FLAG_DOCUMENT_3   307511 non-null  int64 
 10  FLAG_DOCUMENT_4   307511 non-null  int64 
 11  FLAG_DOCUMENT_5   307511 non-null  int64 
 12  FLAG_DOCUMENT_6   307511 non-null  int64 
 13  FLAG_DOCUMENT_7   307511 non-null  int64 
 14  FLAG_DOCUMENT_8   307511 non-null  int64 
 15  FLAG_DOCUMENT_9   307511 non-null  int64 
 16  FLAG_DOCUMENT_10  307511 non-null  int64 
 17  FLAG_DOCUMENT_11  307511 non-null  int64 
 18  FLAG_DOCUMENT_12  307511 non-null  int64 
 19  FLAG_DOCUMENT_13  307511 non-null  int64 
 20  FLAG_DOCUMENT_14  307511 non-null  int64 
 21  FLAG_DOCUMENT_15  307511 non-null  int64 
 22  FLAG_DOCUMENT_16  307511 non-null  int64 
 23  FLAG_DOCUMENT_17  307511 non-null  int64 
 24  FLAG_DOCUMENT_18  307511 non-null  int64 
 25  FLAG_DOCUMENT_19  307511 non-null  int64 
 26  FLAG_DOCUMENT_20  307511 non-null  int64 
 27  FLAG_DOCUMENT_21  307511 non-null  int64 
 28  TARGET            307511 non-null  int64 
dtypes: int64(27), object(2)
memory usage: 68.0+ MB
```

In [13]:
```
#removing the columns with object datatypes so we can plot the corr_matrix
object_columns = flags_cols_data.select_dtypes(include = ['object']).columns
flags_cols_data = flags_cols_data.drop(object_columns, axis = 1 )
```

```
In [14]: #Observing the correlation
plt.figure(figsize = (20,15))
corr_matrix = round(flags_cols_data.corr(),2)
sns.heatmap(corr_matrix,linewidth = 0.2, annot = True)
```

```
Out[14]: <Axes: >
```



As we see they are very less correlation with the target, which is totaly insignificant, so we can remove them

```
In [15]: data.drop(cols_with_flag, axis = 1, inplace = True)
```

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

```
Out[16]: (307511, 45)
```

As we see the column has been reduced from 73 to 45

Check if any other columns can be removed

In [17]:
```
data.head()
```

Out[17]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CRI
0	100002	1	Cash loans	M	0	202500.0	4065
1	100003	0	Cash loans	F	0	270000.0	12935
2	100004	0	Revolving loans	M	0	67500.0	1350
3	100006	0	Cash loans	F	0	135000.0	3126
4	100007	0	Cash loans	M	0	121500.0	5130

5 rows × 45 columns

In [18]:
```
data.columns
```

Out[18]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY',
       'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE',
       'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE',
       'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED',
       'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'OCCUPATION_TYPE',
       'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START',
       'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION',
       'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION',
       'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY',
       'LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_TYPE', 'EXT_SOURCE_2',
       'EXT_SOURCE_3', 'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', 'AMT_REQ_CREDIT_BUREAU_HOUR',
       'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
       'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT',
       'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')
```

In [19]:
```
missing = pd.DataFrame(data.isna().sum().sort_values(ascending = False))
missing.reset_index(inplace=True)
missing.rename(columns = {'index':'column',0:'null_count'},inplace = True)
missing['percent'] = missing['null_count']/data.shape[0]*100
```

In [20]:
```
missing
```

Out[20]:

	column	null_count	percent
0	OCCUPATION_TYPE	96391	31.345545
1	EXT_SOURCE_3	60965	19.825307
2	AMT_REQ_CREDIT_BUREAU_YEAR	41519	13.501631
3	AMT_REQ_CREDIT_BUREAU_QRT	41519	13.501631
4	AMT_REQ_CREDIT_BUREAU_MON	41519	13.501631
5	AMT_REQ_CREDIT_BUREAU_WEEK	41519	13.501631
6	AMT_REQ_CREDIT_BUREAU_DAY	41519	13.501631
7	AMT_REQ_CREDIT_BUREAU_HOUR	41519	13.501631
8	NAME_TYPE_SUITE	1292	0.420148
9	DEF_30_CNT_SOCIAL_CIRCLE	1021	0.332021
10	OBS_30_CNT_SOCIAL_CIRCLE	1021	0.332021
11	OBS_60_CNT_SOCIAL_CIRCLE	1021	0.332021
12	DEF_60_CNT_SOCIAL_CIRCLE	1021	0.332021
13	EXT_SOURCE_2	660	0.214626
14	AMT_GOODS_PRICE	278	0.090403
15	AMT_ANNUITY	12	0.003902
16	CNT_FAM_MEMBERS	2	0.000650
17	DAYS_LAST_PHONE_CHANGE	1	0.000325
18	REG_REGION_NOT_LIVE_REGION	0	0.000000
19	ORGANIZATION_TYPE	0	0.000000
20	LIVE_CITY_NOT_WORK_CITY	0	0.000000
21	REG_CITY_NOT_WORK_CITY	0	0.000000
22	REG_CITY_NOT_LIVE_CITY	0	0.000000
23	LIVE_REGION_NOT_WORK_REGION	0	0.000000
24	REG_REGION_NOT_WORK_REGION	0	0.000000
25	SK_ID_CURR	0	0.000000
26	HOUR_APPR_PROCESS_START	0	0.000000
27	NAME_FAMILY_STATUS	0	0.000000
28	NAME_CONTRACT_TYPE	0	0.000000
29	CODE_GENDER	0	0.000000
30	CNT_CHILDREN	0	0.000000
31	AMT_INCOME_TOTAL	0	0.000000
32	AMT_CREDIT	0	0.000000
33	NAME_INCOME_TYPE	0	0.000000
34	NAME_EDUCATION_TYPE	0	0.000000
35	NAME_HOUSING_TYPE	0	0.000000
36	WEEKDAY_APPR_PROCESS_START	0	0.000000
37	REGION_POPULATION_RELATIVE	0	0.000000
38	DAYS_BIRTH	0	0.000000
39	DAYS_EMPLOYED	0	0.000000
40	DAYS_REGISTRATION	0	0.000000
41	DAYS_ID_PUBLISH	0	0.000000
42	REGION_RATING_CLIENT	0	0.000000
43	TARGET	0	0.000000
44	REGION_RATING_CLIENT_W_CITY	0	0.000000

As we see there is no column that has null values more 40%, we can check them...

In [21]:
```
missing[missing.percent>40]
```

Out[21]:

column	null_count	percent

DEALING MISSING VALUES OF NUMERIC VARIABLES

The mean is used for normal number distributions, which have a low amount of outliers. If there are more outliers in the data, then median is generally used as it returns the central tendency for skewed number distributions The mode is typically used when dealing with categorical or discrete data, rather than numeric variables.

We can deal column wise for the rest of missing values

DAYS_LAST_PHONE_CHANGE

In [22]:
```
data['DAYS_LAST_PHONE_CHANGE'].isna().sum()
```

Out[22]: 1

In [23]:
```
data.dropna(subset=['DAYS_LAST_PHONE_CHANGE'],inplace = True)
```

As there is only one row with null value, decided to remove it.

CNT_FAM_MEMBERS

In [24]:
```
data['CNT_FAM_MEMBERS'] = data['CNT_FAM_MEMBERS'].fillna(data['CNT_FAM_MEMBERS'].mode()[0])
data['CNT_FAM_MEMBERS'].isnull().sum()
```

Out[24]: 0

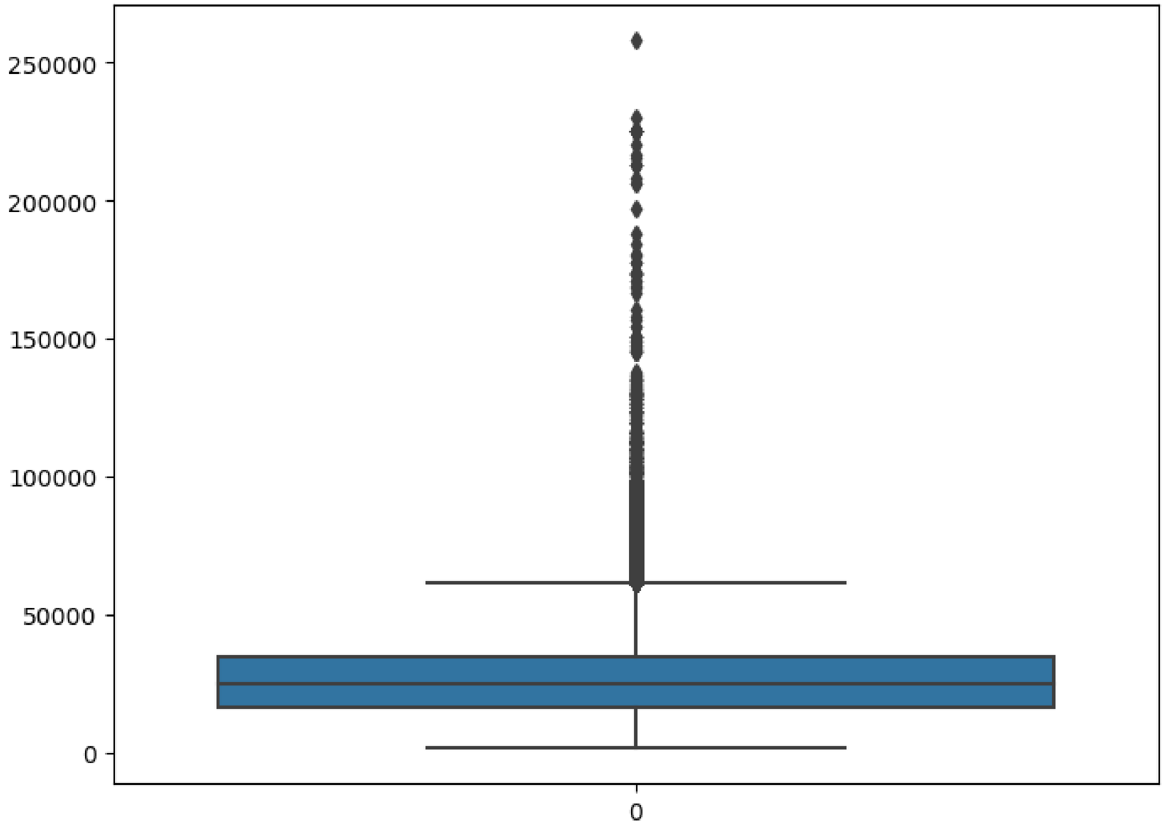
AMT_ANNUITY

In [25]:
```
data['AMT_ANNUITY'].isna().sum()
```

Out[25]: 12

In [26]:
```
plt.figure(figsize =(8,6))
sns.boxplot(data['AMT_ANNUITY'])
```

Out[26]: <Axes: >



Observing that it has significant amount of outliers, decided to impute with median

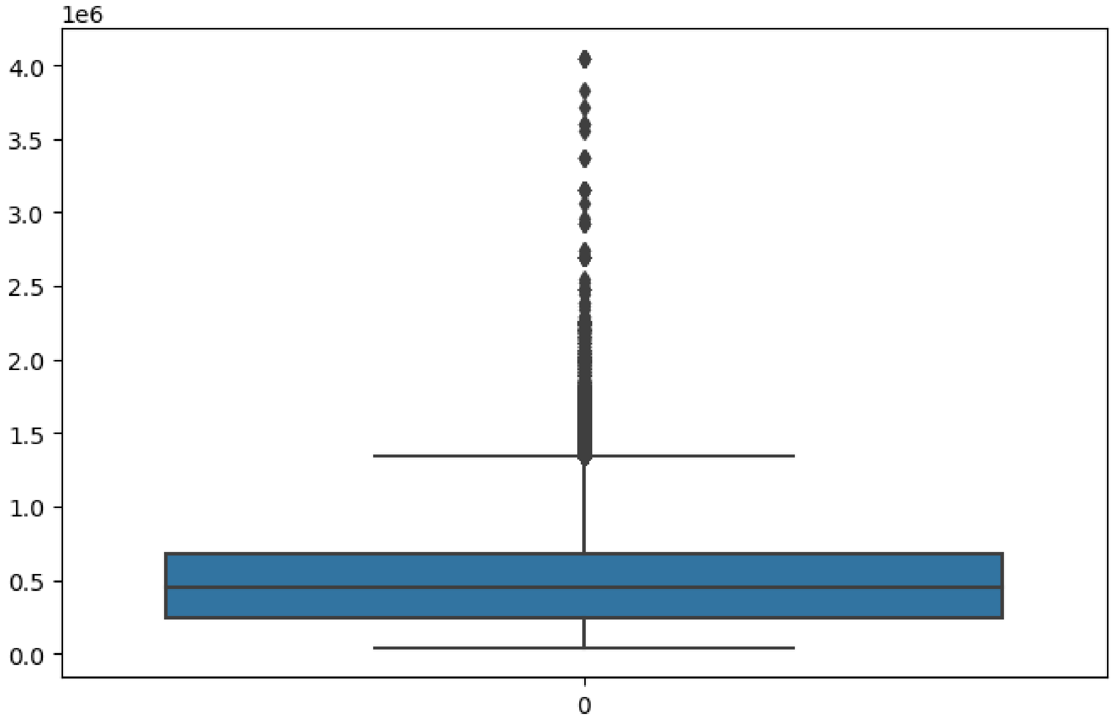
In [27]:
```
data['AMT_ANNUITY'] = data['AMT_ANNUITY'].fillna(data['AMT_ANNUITY'].median())
data['AMT_ANNUITY'].isna().sum()
```

Out[27]: 0

AMT_GOODS_PRICE

In [28]:
```
plt.figure(figsize=(8,5))
sns.boxplot(data['AMT_GOODS_PRICE'])
```

Out[28]: <Axes: >



Observing that it has significant amount of outliers, decided to impute with median

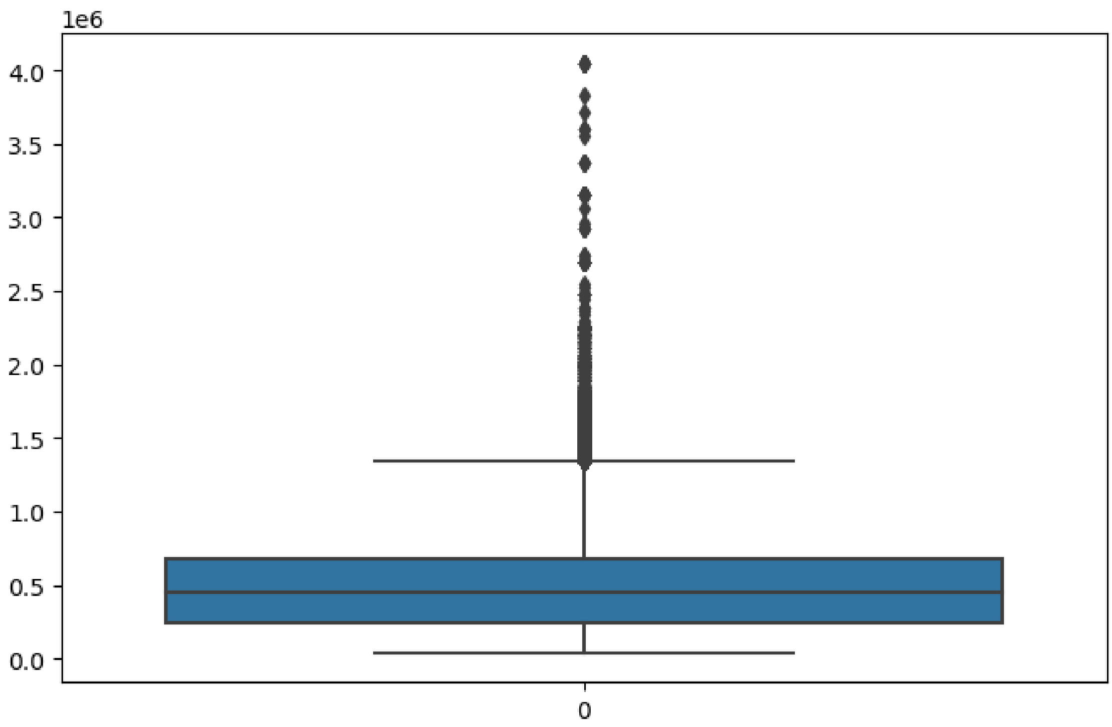
In [29]:
```
data['AMT_GOODS_PRICE'] = data['AMT_GOODS_PRICE'].fillna(data['AMT_GOODS_PRICE'].median())
data['AMT_GOODS_PRICE'].isna().sum()
```

Out[29]: 0

AMT_GOODS_PRICE

In [30]:
```
plt.figure(figsize=(8,5))
sns.boxplot(data['AMT_GOODS_PRICE'])
```

Out[30]: <Axes: >



Observing that it has significant amount of outliers, decided to impute with median

In [31]:
```
data['AMT_GOODS_PRICE'] = data['AMT_GOODS_PRICE'].fillna(data['AMT_GOODS_PRICE'].median())
data['AMT_GOODS_PRICE'].isna().sum()
```

Out[31]: 0

DEALING MISSING VALUES OF CATEGORICAL VARIABLES

In [32]:
```
data['OCCUPATION_TYPE'].unique()
```

Out[32]:
```
array(['Laborers', 'Core staff', 'Accountants', 'Managers', nan,
       'Drivers', 'Sales staff', 'Cleaning staff', 'Cooking staff',
       'Private service staff', 'Medicine staff', 'Security staff',
       'High skill tech staff', 'Waiters/barmen staff',
       'Low-skill Laborers', 'Realty agents', 'Secretaries', 'IT staff',
       'HR staff'], dtype=object)
```

In [33]:
```
#fill missing values with a new class 'unknown'
data['OCCUPATION_TYPE'] = data['OCCUPATION_TYPE'].fillna('Unknown')
```

In [34]:
```
# Fill the missing values with mode
data['DEF_60_CNT_SOCIAL_CIRCLE'] = data['DEF_60_CNT_SOCIAL_CIRCLE'].fillna((data['DEF_60_CNT_SO
data['OBS_30_CNT_SOCIAL_CIRCLE'] = data['OBS_30_CNT_SOCIAL_CIRCLE'].fillna((data['OBS_30_CNT_SO
data['DEF_30_CNT_SOCIAL_CIRCLE'] = data['DEF_30_CNT_SOCIAL_CIRCLE'].fillna((data['DEF_30_CNT_SO
data['OBS_60_CNT_SOCIAL_CIRCLE'] = data['OBS_60_CNT_SOCIAL_CIRCLE'].fillna((data['OBS_60_CNT_SO
```

In [35]:
```python
# NAME_TYPE_SUITE
print(data['NAME_TYPE_SUITE'].value_counts())
# 'Unaccompanied' class is purely dominating the distribution. So, we use it to fill the missing
data['NAME_TYPE_SUITE'] = data['NAME_TYPE_SUITE'].fillna((data['NAME_TYPE_SUITE'].mode()[0]))
```

```
Unaccompanied      248525
Family              40149
Spouse, partner     11370
Children             3267
Other_B              1770
Other_A               866
Group of people       271
Name: NAME_TYPE_SUITE, dtype: int64
```

In [36]:
```python
data.isna().sum().sort_values(ascending = False).head(10)
```

Out[36]:
```
EXT_SOURCE_3                  60964
AMT_REQ_CREDIT_BUREAU_YEAR    41518
AMT_REQ_CREDIT_BUREAU_QRT     41518
AMT_REQ_CREDIT_BUREAU_MON     41518
AMT_REQ_CREDIT_BUREAU_WEEK    41518
AMT_REQ_CREDIT_BUREAU_DAY     41518
AMT_REQ_CREDIT_BUREAU_HOUR    41518
EXT_SOURCE_2                    659
LIVE_CITY_NOT_WORK_CITY           0
REG_REGION_NOT_LIVE_REGION        0
dtype: int64
```

DEALING WITH COLUMNS RELATED TO DATE

In [37]:
```python
data[data['AMT_REQ_CREDIT_BUREAU_DAY'].isna()].head()
```

Out[37]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CI
3	100006	0	Cash loans	F	0	135000.0	312
9	100012	0	Revolving loans	M	0	135000.0	405
14	100018	0	Cash loans	F	0	189000.0	773
17	100021	0	Revolving loans	F	1	81000.0	270
20	100024	0	Revolving loans	M	0	135000.0	427

5 rows × 45 columns

In [38]:
```python
# Fetching the columns
amt_req = []

for k in data.columns:
    if k.startswith('AMT_REQ_CREDIT_BUREAU_'):
        amt_req.append(k) #add the features to the list

amt_req
```

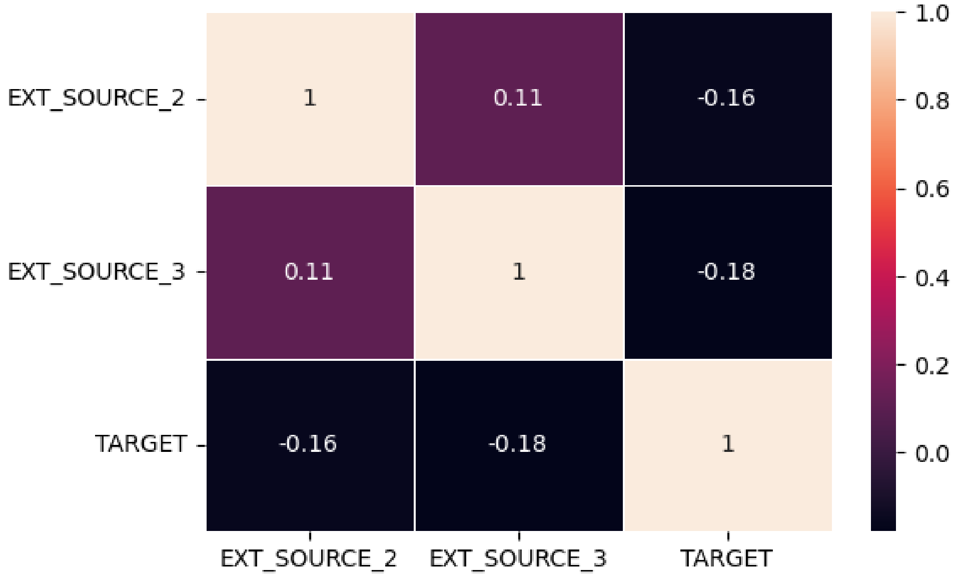
Out[38]:
```
['AMT_REQ_CREDIT_BUREAU_HOUR',
 'AMT_REQ_CREDIT_BUREAU_DAY',
 'AMT_REQ_CREDIT_BUREAU_WEEK',
 'AMT_REQ_CREDIT_BUREAU_MON',
 'AMT_REQ_CREDIT_BUREAU_QRT',
 'AMT_REQ_CREDIT_BUREAU_YEAR']
```

In [39]:
```python
#Impute missing values with median
for col in amt_req:
    data[col] = data[col].fillna(data[col].median())
```

In [40]:
```python
data.isna().sum().sort_values(ascending=False).head(20)
```

Out[40]:
```
EXT_SOURCE_3                 60964
EXT_SOURCE_2                   659
SK_ID_CURR                       0
WEEKDAY_APPR_PROCESS_START       0
REG_REGION_NOT_LIVE_REGION       0
REG_REGION_NOT_WORK_REGION       0
LIVE_REGION_NOT_WORK_REGION      0
REG_CITY_NOT_LIVE_CITY           0
REG_CITY_NOT_WORK_CITY           0
LIVE_CITY_NOT_WORK_CITY          0
ORGANIZATION_TYPE                0
OBS_30_CNT_SOCIAL_CIRCLE         0
DEF_30_CNT_SOCIAL_CIRCLE         0
OBS_60_CNT_SOCIAL_CIRCLE         0
DEF_60_CNT_SOCIAL_CIRCLE         0
DAYS_LAST_PHONE_CHANGE           0
AMT_REQ_CREDIT_BUREAU_HOUR       0
AMT_REQ_CREDIT_BUREAU_DAY        0
AMT_REQ_CREDIT_BUREAU_WEEK       0
AMT_REQ_CREDIT_BUREAU_MON        0
dtype: int64
```

In [41]:
```python
# Correlation matrix
plt.figure(figsize=(6,4))
sns.heatmap(round(data[['EXT_SOURCE_2', 'EXT_SOURCE_3', 'TARGET']].corr(),2),linewidths=0.5, an
plt.show()
```



In [42]:
```python
# Drop features
data = data.drop(columns=['EXT_SOURCE_2','EXT_SOURCE_3'])
```

NUMERICAL VARIABLES BINNING FOR DATA VISUALIZATION

In [43]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 307510 entries, 0 to 307510
Data columns (total 43 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   SK_ID_CURR                   307510 non-null  int64  
 1   TARGET                       307510 non-null  int64  
 2   NAME_CONTRACT_TYPE           307510 non-null  object 
 3   CODE_GENDER                  307510 non-null  object 
 4   CNT_CHILDREN                 307510 non-null  int64  
 5   AMT_INCOME_TOTAL             307510 non-null  float64
 6   AMT_CREDIT                   307510 non-null  float64
 7   AMT_ANNUITY                  307510 non-null  float64
 8   AMT_GOODS_PRICE              307510 non-null  float64
 9   NAME_TYPE_SUITE              307510 non-null  object 
 10  NAME_INCOME_TYPE             307510 non-null  object 
 11  NAME_EDUCATION_TYPE          307510 non-null  object 
 12  NAME_FAMILY_STATUS           307510 non-null  object 
 13  NAME_HOUSING_TYPE            307510 non-null  object 
 14  REGION_POPULATION_RELATIVE   307510 non-null  float64
 15  DAYS_BIRTH                   307510 non-null  int64  
 16  DAYS_EMPLOYED                307510 non-null  int64  
 17  DAYS_REGISTRATION            307510 non-null  float64
 18  DAYS_ID_PUBLISH              307510 non-null  int64  
 19  OCCUPATION_TYPE              307510 non-null  object 
 20  CNT_FAM_MEMBERS              307510 non-null  float64
 21  REGION_RATING_CLIENT         307510 non-null  int64  
 22  REGION_RATING_CLIENT_W_CITY  307510 non-null  int64  
 23  WEEKDAY_APPR_PROCESS_START   307510 non-null  object 
 24  HOUR_APPR_PROCESS_START      307510 non-null  int64  
 25  REG_REGION_NOT_LIVE_REGION   307510 non-null  int64  
 26  REG_REGION_NOT_WORK_REGION   307510 non-null  int64  
 27  LIVE_REGION_NOT_WORK_REGION  307510 non-null  int64  
 28  REG_CITY_NOT_LIVE_CITY       307510 non-null  int64  
 29  REG_CITY_NOT_WORK_CITY       307510 non-null  int64  
 30  LIVE_CITY_NOT_WORK_CITY      307510 non-null  int64  
 31  ORGANIZATION_TYPE            307510 non-null  object 
 32  OBS_30_CNT_SOCIAL_CIRCLE     307510 non-null  float64
 33  DEF_30_CNT_SOCIAL_CIRCLE     307510 non-null  float64
 34  OBS_60_CNT_SOCIAL_CIRCLE     307510 non-null  float64
 35  DEF_60_CNT_SOCIAL_CIRCLE     307510 non-null  float64
 36  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64
 37  AMT_REQ_CREDIT_BUREAU_HOUR   307510 non-null  float64
 38  AMT_REQ_CREDIT_BUREAU_DAY    307510 non-null  float64
 39  AMT_REQ_CREDIT_BUREAU_WEEK   307510 non-null  float64
 40  AMT_REQ_CREDIT_BUREAU_MON    307510 non-null  float64
 41  AMT_REQ_CREDIT_BUREAU_QRT    307510 non-null  float64
 42  AMT_REQ_CREDIT_BUREAU_YEAR   307510 non-null  float64
dtypes: float64(18), int64(15), object(10)
memory usage: 111.3+ MB
```

In [44]:
```
data.select_dtypes(include = 'float')
```

Out[44]:

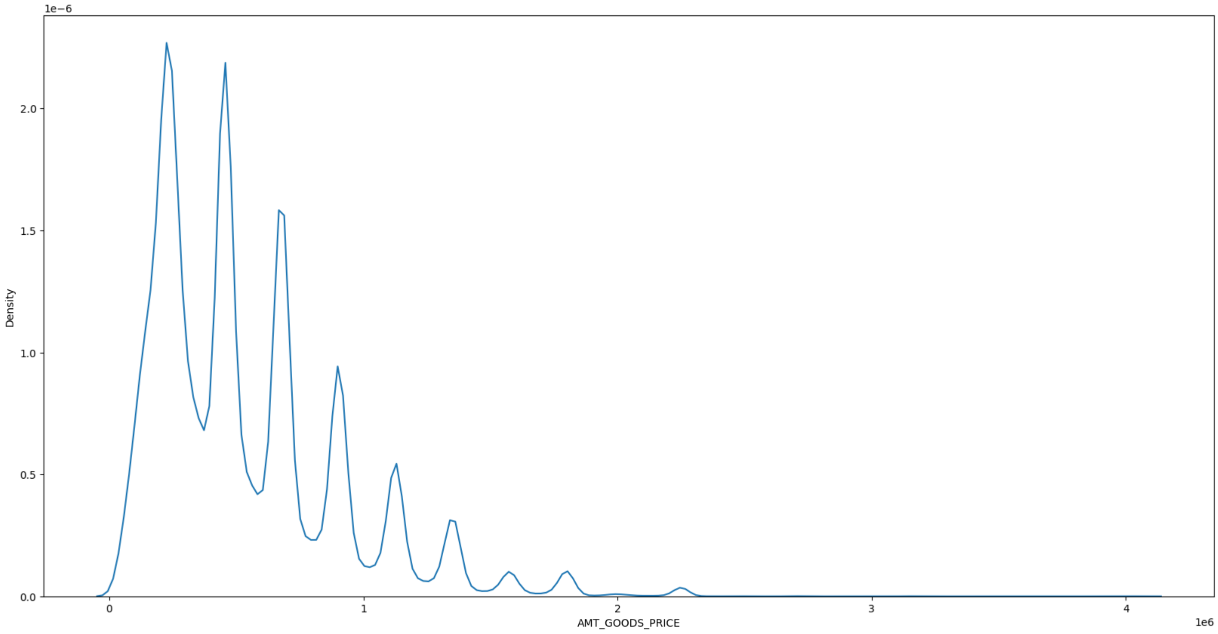
	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_POPULATION_RELATIVE	D
0	202500.0	406597.5	24700.5	351000.0	0.018801	
1	270000.0	1293502.5	35698.5	1129500.0	0.003541	
2	67500.0	135000.0	6750.0	135000.0	0.010032	
3	135000.0	312682.5	29686.5	297000.0	0.008019	
4	121500.0	513000.0	21865.5	513000.0	0.028663	
...	...	...	...	...	...	
307506	157500.0	254700.0	27558.0	225000.0	0.032561	
307507	72000.0	269550.0	12001.5	225000.0	0.025164	
307508	153000.0	677664.0	29979.0	585000.0	0.005002	
307509	171000.0	370107.0	20205.0	319500.0	0.005313	
307510	157500.0	675000.0	49117.5	675000.0	0.046220	

307510 rows × 18 columns

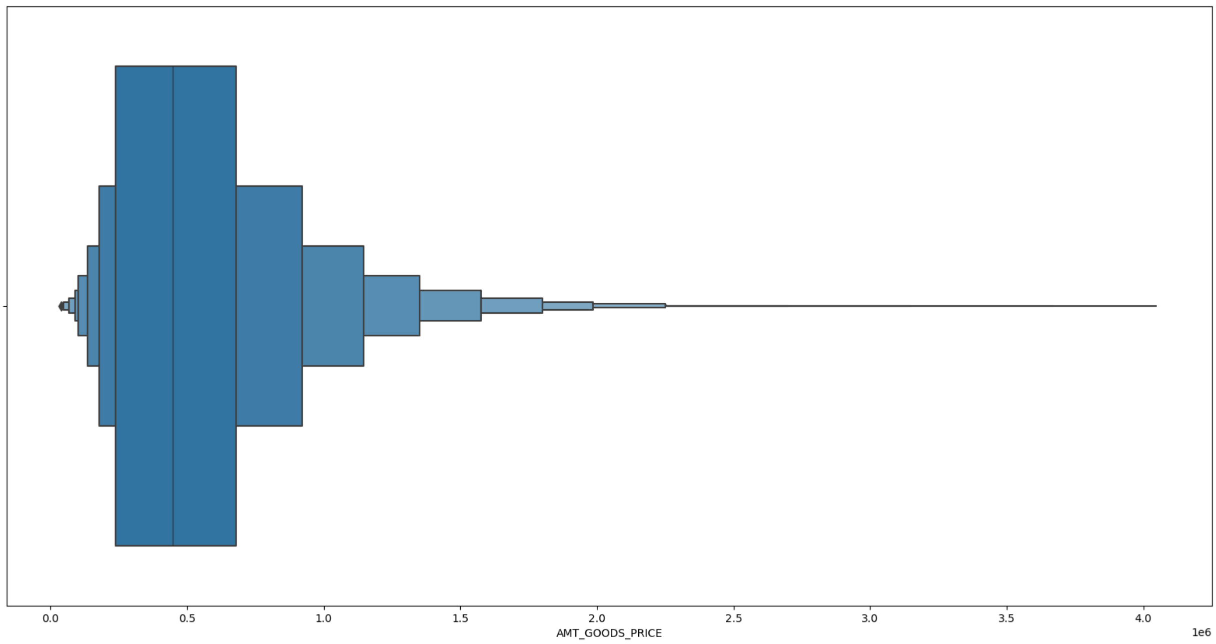
In [45]:
```python
data.nunique().sort_values(ascending = False)
```

Out[45]:
```
SK_ID_CURR                     307510
DAYS_BIRTH                      17460
DAYS_REGISTRATION               15688
AMT_ANNUITY                     13672
DAYS_EMPLOYED                   12574
DAYS_ID_PUBLISH                  6168
AMT_CREDIT                       5603
DAYS_LAST_PHONE_CHANGE           3773
AMT_INCOME_TOTAL                 2548
AMT_GOODS_PRICE                  1002
REGION_POPULATION_RELATIVE         81
ORGANIZATION_TYPE                  58
OBS_60_CNT_SOCIAL_CIRCLE           33
OBS_30_CNT_SOCIAL_CIRCLE           33
AMT_REQ_CREDIT_BUREAU_YEAR         25
HOUR_APPR_PROCESS_START            24
AMT_REQ_CREDIT_BUREAU_MON          24
OCCUPATION_TYPE                    19
CNT_FAM_MEMBERS                    17
CNT_CHILDREN                       15
AMT_REQ_CREDIT_BUREAU_QRT          11
DEF_30_CNT_SOCIAL_CIRCLE           10
AMT_REQ_CREDIT_BUREAU_WEEK          9
AMT_REQ_CREDIT_BUREAU_DAY           9
DEF_60_CNT_SOCIAL_CIRCLE            9
NAME_INCOME_TYPE                    8
WEEKDAY_APPR_PROCESS_START          7
NAME_TYPE_SUITE                     7
NAME_FAMILY_STATUS                  6
NAME_HOUSING_TYPE                   6
NAME_EDUCATION_TYPE                 5
AMT_REQ_CREDIT_BUREAU_HOUR          5
CODE_GENDER                         3
REGION_RATING_CLIENT                3
REGION_RATING_CLIENT_W_CITY         3
LIVE_CITY_NOT_WORK_CITY             2
REG_CITY_NOT_LIVE_CITY              2
NAME_CONTRACT_TYPE                  2
LIVE_REGION_NOT_WORK_REGION         2
REG_REGION_NOT_WORK_REGION          2
REG_REGION_NOT_LIVE_REGION          2
TARGET                              2
REG_CITY_NOT_WORK_CITY              2
dtype: int64
```

In [46]:
```
plt.figure(figsize = (20,10))
sns.kdeplot(data = data, x = 'AMT_GOODS_PRICE')
plt.show()
```



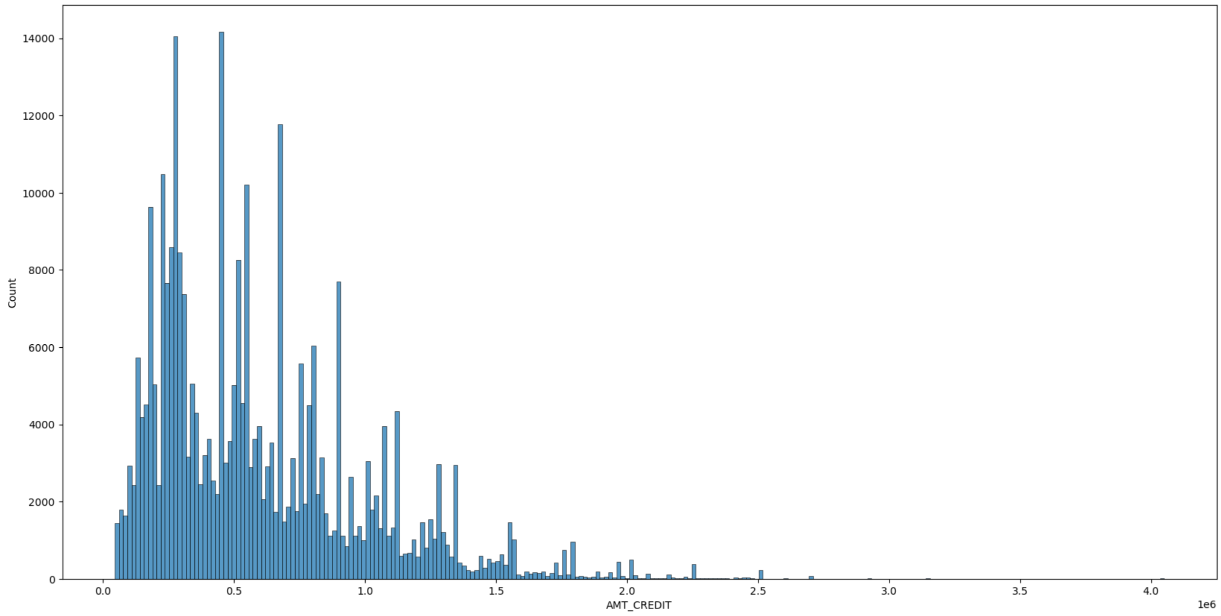
In [47]:
```
plt.figure(figsize = (20,10))
sns.boxenplot(data = data, x = 'AMT_GOODS_PRICE')
plt.show()
```



In [48]:
```
#AMT_GOODS_PRICE
data['AMT_GOODS_PRICE'].quantile([0.1,0.25,0.5,0.75,0.9])
```

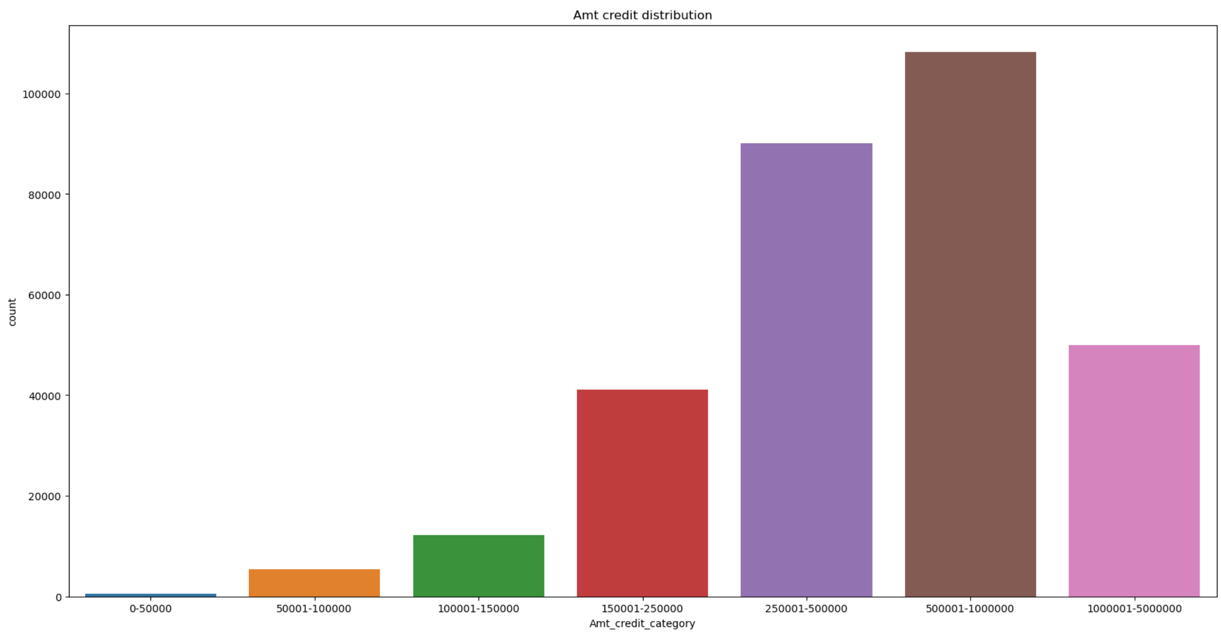
Out[48]:
```
0.10     180000.0
0.25     238500.0
0.50     450000.0
0.75     679500.0
0.90    1093500.0
Name: AMT_GOODS_PRICE, dtype: float64
```

In [49]:
```
plt.figure(figsize = (20,10))
sns.histplot(data.AMT_CREDIT)
```
Out[49]: <Axes: xlabel='AMT_CREDIT', ylabel='Count'>



In [50]:
```
data.AMT_CREDIT.describe().loc[['min','max']]
```
Out[50]:
```
min      45000.0
max    4050000.0
Name: AMT_CREDIT, dtype: float64
```

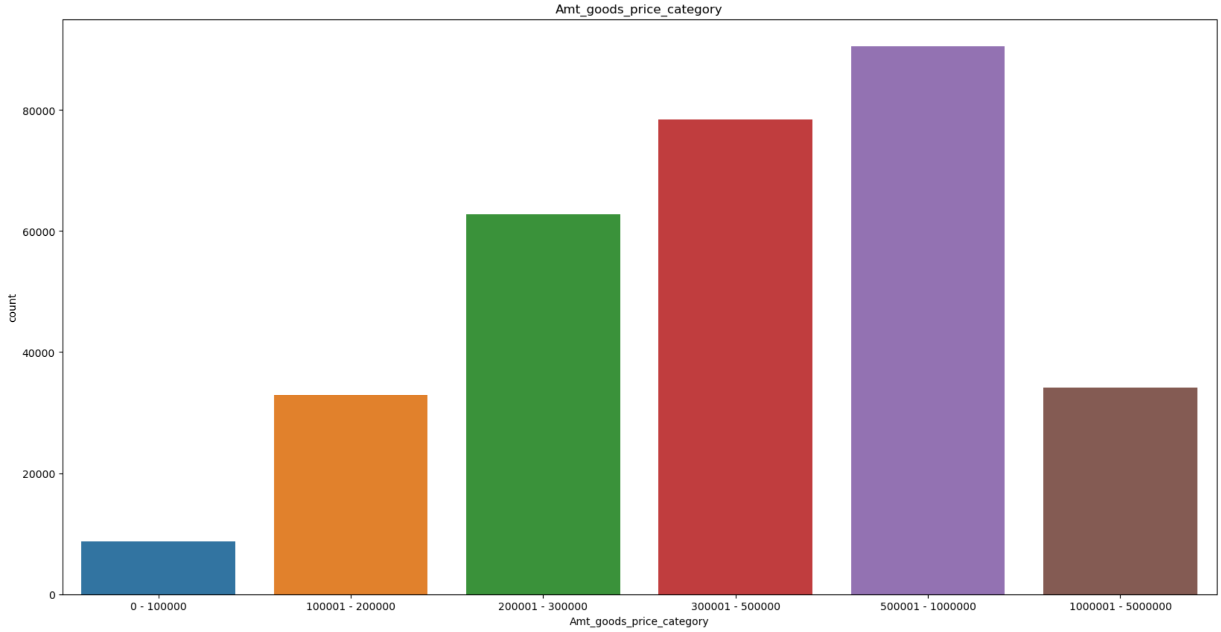
In [51]:
```
labels = ['0-50000', '50001-100000', '100001-150000', '150001-250000', '250001-500000', '500001
data['Amt_credit_category'] = pd.cut(data['AMT_CREDIT'], bins=[0, 50000, 100000, 150000, 250000

plt.figure(figsize=(20, 10))
sns.countplot(x=data['Amt_credit_category'])
plt.title('Amt credit distribution')
plt.show()
```



In [52]:
```python
data.AMT_GOODS_PRICE.describe().loc[['min','max']]
```

Out[52]:
```
min      40500.0
max    4050000.0
Name: AMT_GOODS_PRICE, dtype: float64
```

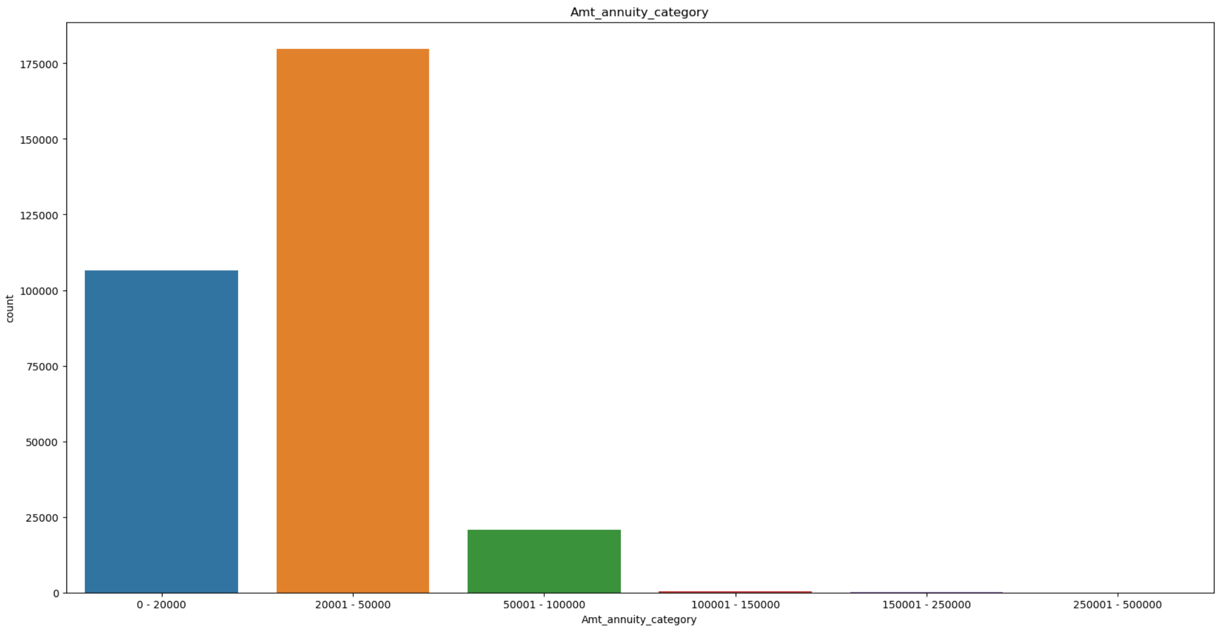
In [53]:
```python
# AMT_GOODS_PRICE
labels= ['0 - 100000','100001 - 200000','200001 - 300000','300001 - 500000','500001 - 1000000',
data['Amt_goods_price_category'] = pd.cut(data['AMT_GOODS_PRICE'], bins=[0,100000,200000,300000

plt.figure(figsize=(20,10))
sns.countplot(x=data['Amt_goods_price_category'])
plt.title('Amt_goods_price_category')
plt.show()
```



In [54]:
```python
data.AMT_ANNUITY.describe().loc[['min','max']]
```

Out[54]:
```
min      1615.5
max    258025.5
Name: AMT_ANNUITY, dtype: float64
```

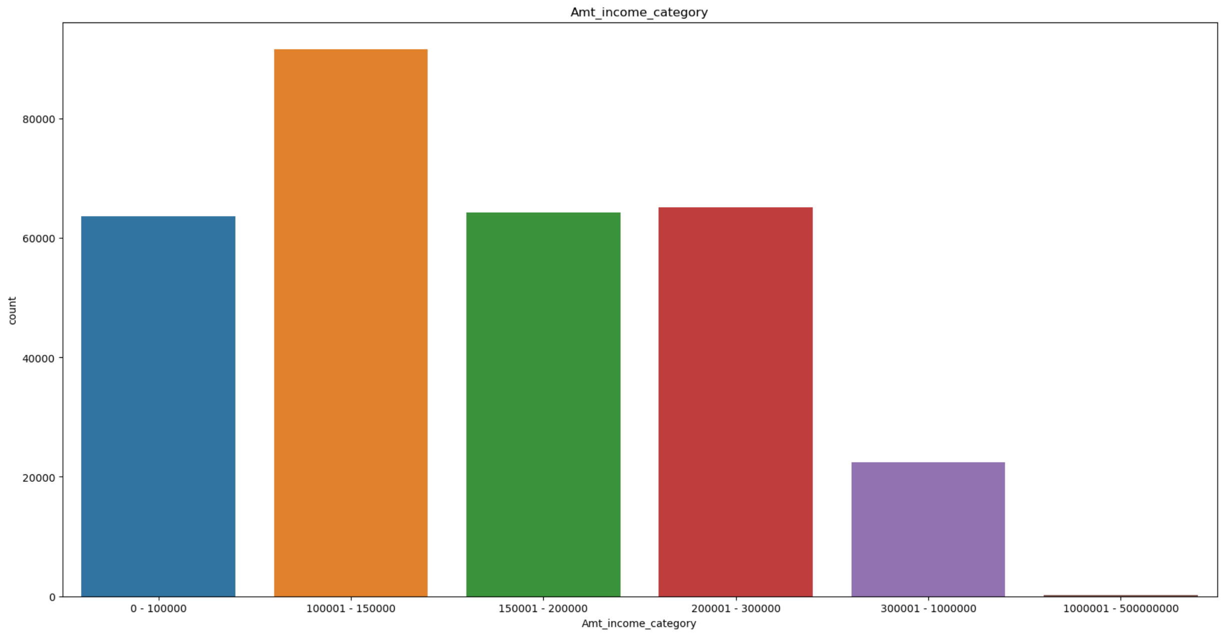
In [55]:
```
# AMT_ANNUITY
labels= ['0 - 20000','20001 - 50000','50001 - 100000','100001 - 150000','150001 - 250000','2500
data['Amt_annuity_category'] = pd.cut(data['AMT_ANNUITY'], bins=[0,20000,50000,100000,150000,25

plt.figure(figsize=(20,10))
sns.countplot(x=data['Amt_annuity_category'])
plt.title('Amt_annuity_category')
plt.show()
```



In [56]:
```
data.AMT_INCOME_TOTAL.describe().loc[['min','max']]
```

Out[56]:
```
min        25650.0
max    117000000.0
Name: AMT_INCOME_TOTAL, dtype: float64
```

In [57]:
```
# AMT_INCOME_TOTAL
labels= ['0 - 100000','100001 - 150000','150001 - 200000','200001 - 300000','300001 - 1000000',
data['Amt_income_category'] = pd.cut(data['AMT_INCOME_TOTAL'], bins=[0,100000,150000,200000,300

plt.figure(figsize=(20,10))
sns.countplot(x=data['Amt_income_category'])
plt.title('Amt_income_category')
plt.show()
```

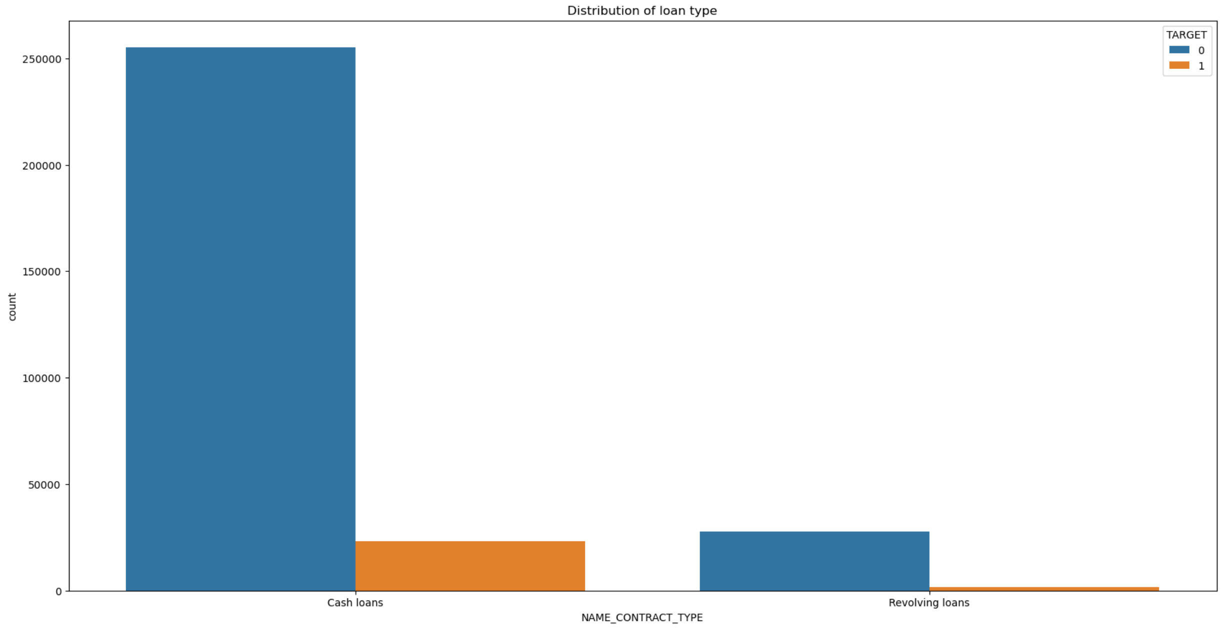


CATEGORICAL VARIABLES DATA VISUALIZATION

In [58]:
```
# NAME_CONTRACT_TYPE
data.NAME_CONTRACT_TYPE.value_counts()
```

Out[58]:
```
Cash loans         278231
Revolving loans     29279
Name: NAME_CONTRACT_TYPE, dtype: int64
```

In [59]:
```
#countplot
plt.figure(figsize = (20,10))
sns.countplot(x='NAME_CONTRACT_TYPE', data=data, hue = "TARGET")
plt.title('Distribution of loan type')
plt.show()
```



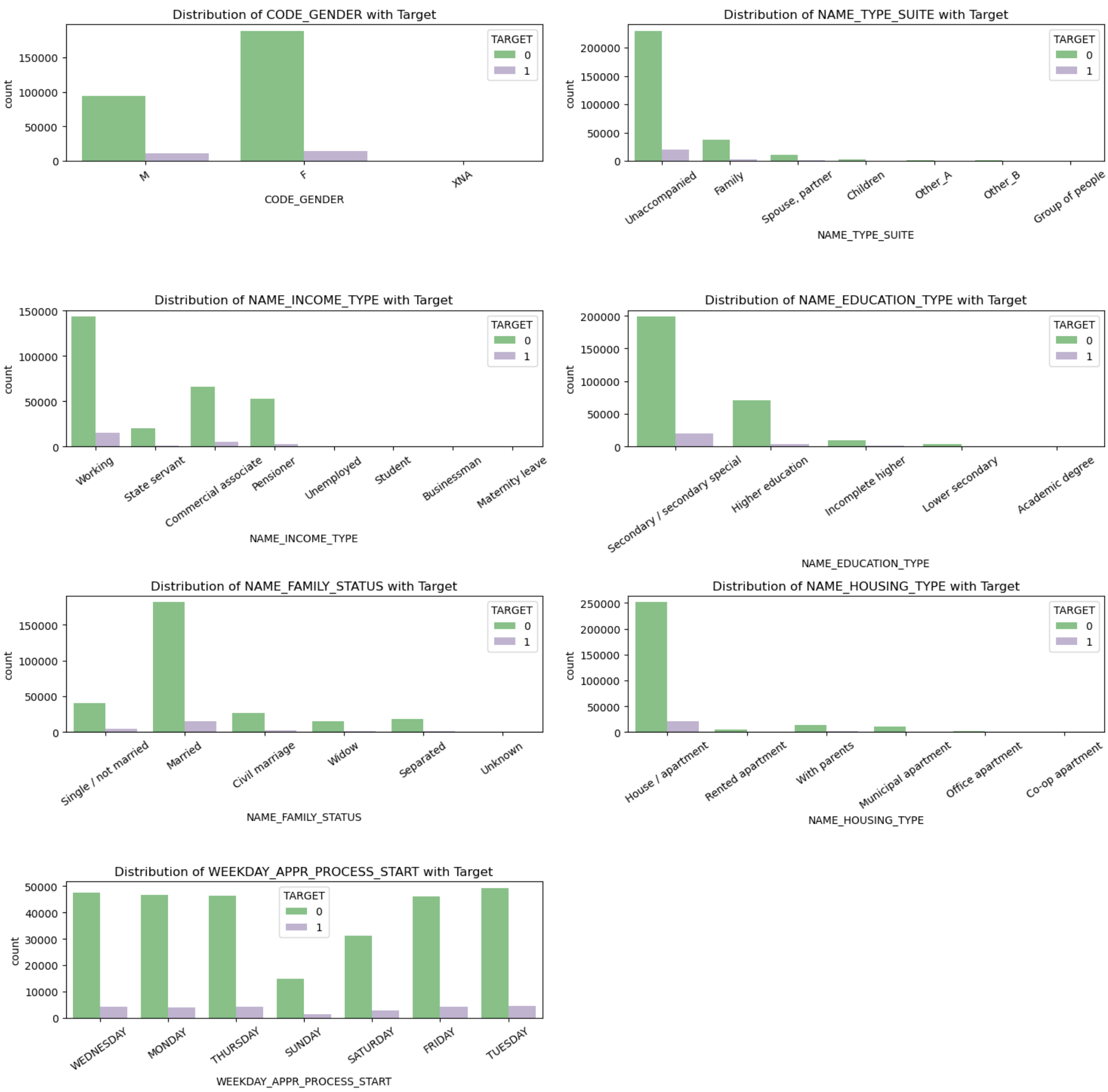
Based on the observations, there is a strong indication or pattern that individuals who have taken a cash loan (presumably a type of loan where the borrower receives cash directly) are more likely to default on their loans compared to individuals who have not taken a cash loan.

In [60]:
```
# Dataframe for loan type with target
loan_with_target = data.groupby(['NAME_CONTRACT_TYPE', 'TARGET']).size().reset_index(name='coun

loan_with_target['percentage'] = round((loan_with_target['count']/len(data['NAME_CONTRACT_TYPE'
loan_with_target
```

Out[60]:

	NAME_CONTRACT_TYPE	TARGET	count	percentage
0	Cash loans	0	255010	82.93
1	Cash loans	1	23221	7.55
2	Revolving loans	0	27675	9.00
3	Revolving loans	1	1604	0.52

In [61]:
```python
cols = ['CODE_GENDER', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMI

# Countplot
plt.figure(figsize=(15, 25))
index = 0  # Initialize an index variable
for c in cols:
    plt.subplot(len(cols), 2, index + 1)
    sns.countplot(x=c, data=data, hue='TARGET', palette='Accent')
    plt.title(f"Distribution of {c} with Target")
    plt.xticks(rotation=35)
    index += 1  # Increment the index

plt.tight_layout()
```



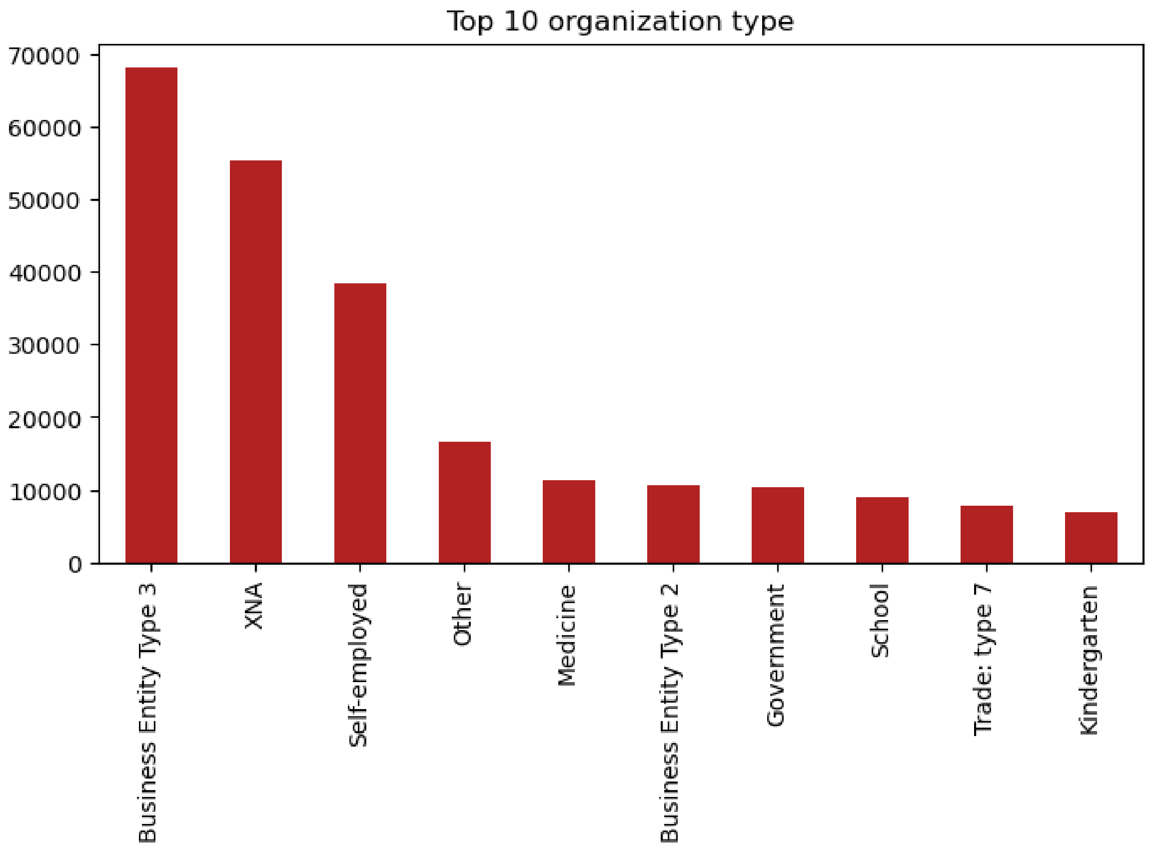
By Close observation of each bar chart, we can come to following conclusions:

1. Females are less likely to default the loan than male.
2. Working client, Commercial associate and Pensioner have taken more loans.
3. Unaccompanied has taken most number of loans.
4. Married client has received more number of credits.
5. Most of the clients have their house apartment.
6. All days have equal number of application received, except sunday.

In [62]:
```python
#organization type
data['ORGANIZATION_TYPE'].value_counts().sort_values(ascending = False)
```

Out[62]:
```
Business Entity Type 3    67992
XNA                       55374
Self-employed             38412
Other                     16683
Medicine                  11193
Business Entity Type 2    10553
Government                10404
School                     8893
Trade: type 7              7831
Kindergarten               6880
Construction               6721
Business Entity Type 1     5984
Transport: type 4          5398
Trade: type 3              3491
Industry: type 9           3368
Industry: type 3           3278
Security                   3247
Housing                    2958
Industry: type 11          2704
Military                   2634
Bank                       2507
Agriculture                2454
Police                     2341
Transport: type 2          2204
Postal                     2157
Security Ministries        1974
Trade: type 2              1900
Restaurant                 1811
Services                   1575
University                 1327
Industry: type 7           1307
Transport: type 3          1187
Industry: type 1           1039
Hotel                       966
Electricity                 950
Industry: type 4            877
Trade: type 6               631
Industry: type 5            599
Insurance                   597
Telecom                     577
Emergency                   560
Industry: type 2            458
Advertising                 429
Realtor                     396
Culture                     379
Industry: type 12           369
Trade: type 1               348
Mobile                      317
Legal Services              305
Cleaning                    260
Transport: type 1           201
Industry: type 6            112
Industry: type 10           109
Religion                     85
Industry: type 13            67
Trade: type 4                64
Trade: type 5                49
Industry: type 8             24
Name: ORGANIZATION_TYPE, dtype: int64
```

In [63]:
```
# Organization type
plt.figure(figsize = (8,4))
data['ORGANIZATION_TYPE'].value_counts().sort_values(ascending = False)[:10].plot(kind ='bar',
plt.title('Top 10 organization type')
plt.show()
```



In [64]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 307510 entries, 0 to 307510
Data columns (total 47 columns):
 #   Column                       Non-Null Count   Dtype   
---  ------                       --------------   -----   
 0   SK_ID_CURR                   307510 non-null  int64   
 1   TARGET                       307510 non-null  int64   
 2   NAME_CONTRACT_TYPE           307510 non-null  object  
 3   CODE_GENDER                  307510 non-null  object  
 4   CNT_CHILDREN                 307510 non-null  int64   
 5   AMT_INCOME_TOTAL             307510 non-null  float64 
 6   AMT_CREDIT                   307510 non-null  float64 
 7   AMT_ANNUITY                  307510 non-null  float64 
 8   AMT_GOODS_PRICE              307510 non-null  float64 
 9   NAME_TYPE_SUITE              307510 non-null  object  
 10  NAME_INCOME_TYPE             307510 non-null  object  
 11  NAME_EDUCATION_TYPE          307510 non-null  object  
 12  NAME_FAMILY_STATUS           307510 non-null  object  
 13  NAME_HOUSING_TYPE            307510 non-null  object  
 14  REGION_POPULATION_RELATIVE   307510 non-null  float64 
 15  DAYS_BIRTH                   307510 non-null  int64   
 16  DAYS_EMPLOYED                307510 non-null  int64   
 17  DAYS_REGISTRATION            307510 non-null  float64 
 18  DAYS_ID_PUBLISH              307510 non-null  int64   
 19  OCCUPATION_TYPE              307510 non-null  object  
 20  CNT_FAM_MEMBERS              307510 non-null  float64 
 21  REGION_RATING_CLIENT         307510 non-null  int64   
 22  REGION_RATING_CLIENT_W_CITY  307510 non-null  int64   
 23  WEEKDAY_APPR_PROCESS_START   307510 non-null  object  
 24  HOUR_APPR_PROCESS_START      307510 non-null  int64   
 25  REG_REGION_NOT_LIVE_REGION   307510 non-null  int64   
 26  REG_REGION_NOT_WORK_REGION   307510 non-null  int64   
 27  LIVE_REGION_NOT_WORK_REGION  307510 non-null  int64   
 28  REG_CITY_NOT_LIVE_CITY       307510 non-null  int64   
 29  REG_CITY_NOT_WORK_CITY       307510 non-null  int64   
 30  LIVE_CITY_NOT_WORK_CITY      307510 non-null  int64   
 31  ORGANIZATION_TYPE            307510 non-null  object  
 32  OBS_30_CNT_SOCIAL_CIRCLE     307510 non-null  float64 
 33  DEF_30_CNT_SOCIAL_CIRCLE     307510 non-null  float64 
 34  OBS_60_CNT_SOCIAL_CIRCLE     307510 non-null  float64 
 35  DEF_60_CNT_SOCIAL_CIRCLE     307510 non-null  float64 
 36  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64 
 37  AMT_REQ_CREDIT_BUREAU_HOUR   307510 non-null  float64 
 38  AMT_REQ_CREDIT_BUREAU_DAY    307510 non-null  float64 
 39  AMT_REQ_CREDIT_BUREAU_WEEK   307510 non-null  float64 
 40  AMT_REQ_CREDIT_BUREAU_MON    307510 non-null  float64 
 41  AMT_REQ_CREDIT_BUREAU_QRT    307510 non-null  float64 
 42  AMT_REQ_CREDIT_BUREAU_YEAR   307510 non-null  float64 
 43  Amt_credit_category          307510 non-null  category
 44  Amt_goods_price_category     307510 non-null  category
 45  Amt_annuity_category         307510 non-null  category
 46  Amt_income_category          307510 non-null  category
dtypes: category(4), float64(18), int64(15), object(10)
memory usage: 112.5+ MB
```

In [65]:
```
# numeric features and categorical features
num_features = data.select_dtypes(include = ['int','float']).columns
num_cat_features = data.select_dtypes(include = ['int','float','category']).columns
```

In [66]:
```
num_features
```

Out[66]:
```
Index(['SK_ID_CURR', 'TARGET', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE',
       'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED',
       'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'CNT_FAM_MEMBERS',
       'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY',
       'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION',
       'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION',
       'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY',
       'LIVE_CITY_NOT_WORK_CITY', 'OBS_30_CNT_SOCIAL_CIRCLE',
       'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE',
       'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')
```

In [67]:
```
# Create the num_data DataFrame
num_data = data[num_features]

# Print the head of the num_data DataFrame
num_data.head()
```

Out[67]:

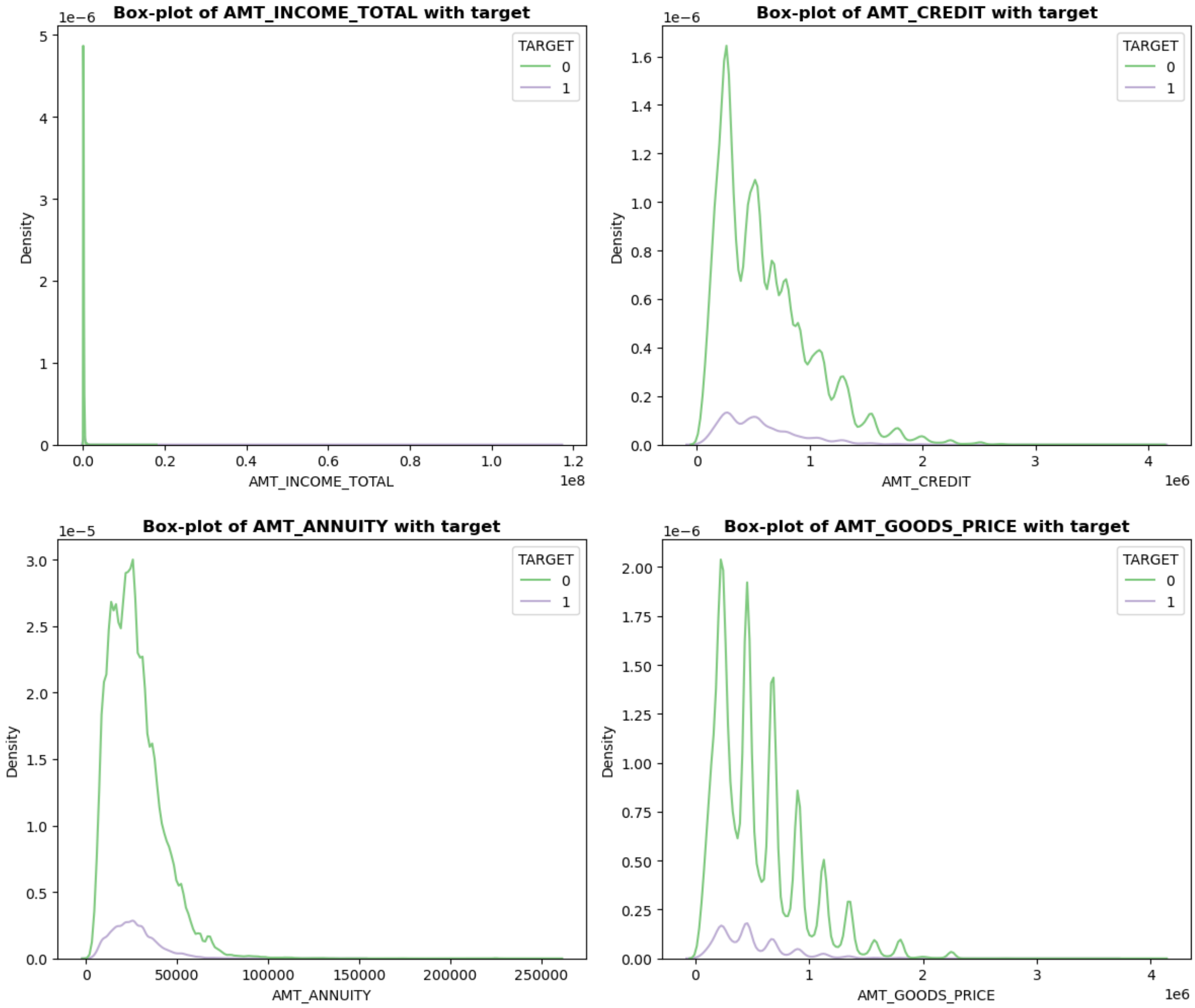
	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE
0	100002	1	0	202500.0	406597.5	24700.5	351000.0
1	100003	0	0	270000.0	1293502.5	35698.5	1129500.0
2	100004	0	0	67500.0	135000.0	6750.0	135000.0
3	100006	0	0	135000.0	312682.5	29686.5	297000.0
4	100007	0	0	121500.0	513000.0	21865.5	513000.0

5 rows × 33 columns

In [68]:
```
data['TARGET'].unique()
```

Out[68]:
```
array([1, 0], dtype=int64)
```

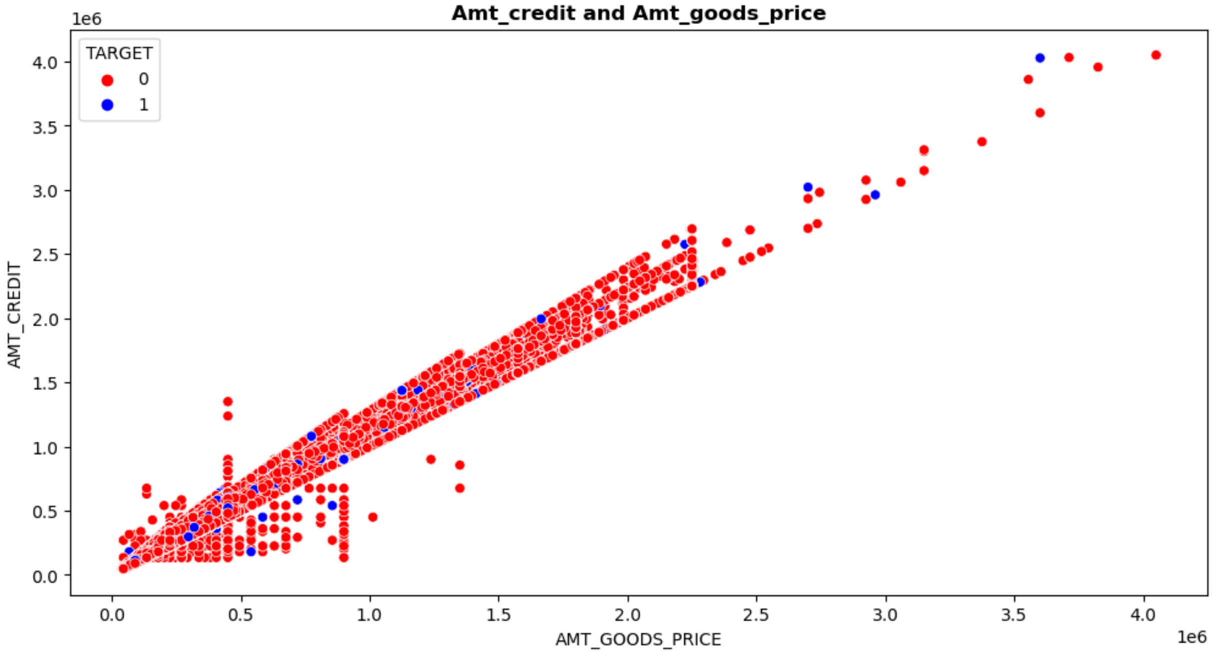
In [69]:
```
amt_var = ['AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','AMT_GOODS_PRICE']

plt.figure(figsize=(12,20))
for index, k in enumerate(amt_var):
 plt.subplot(len(amt_var),2, index+1)
 sns.kdeplot(x=k, data=num_data, hue='TARGET', palette='Accent')
 plt.title(f"Box-plot of {k} with target", fontweight='bold')

plt.tight_layout()
```



Observations:

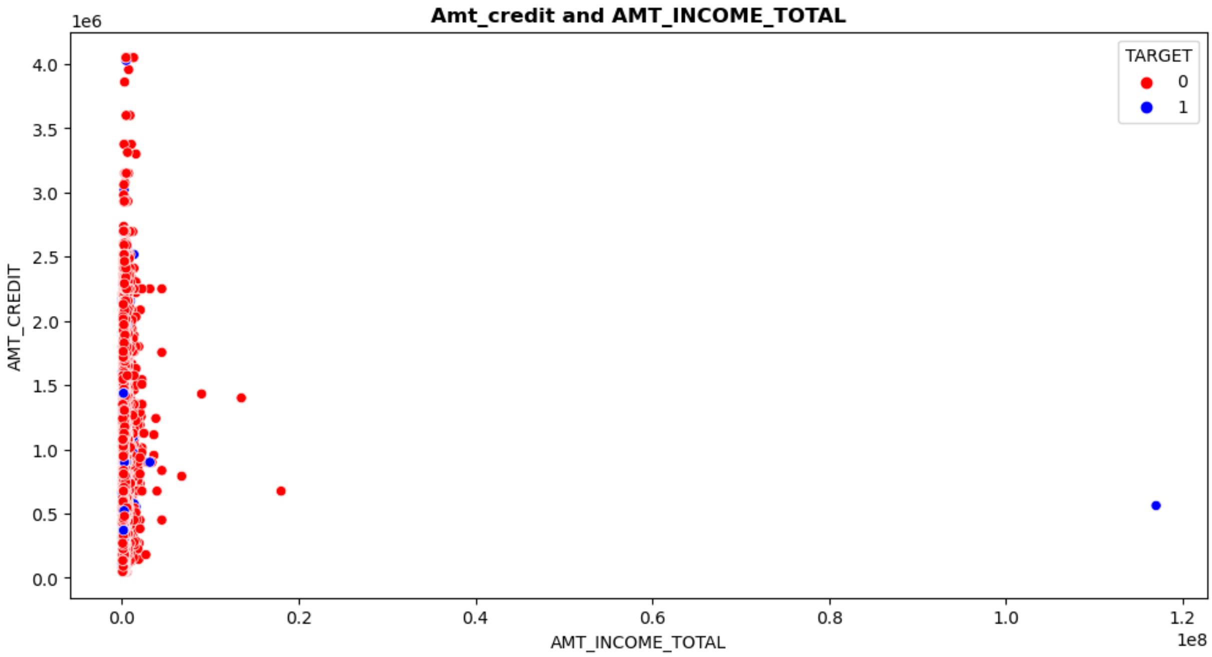
1. Most of the defaulters are from high-income groups.
2. Most defaulters fall under the category of amt_credit between 0 to 1 million.
3. Annuity payment of 0 to 50000 have more number of defaults.
4. Amount goods price between o to 1 million have more number of defaults.

In [70]:
```
# Scatter plot
plt.figure(figsize=(12,6))
sns.scatterplot(data=num_data, x='AMT_GOODS_PRICE', y='AMT_CREDIT', palette=['red', 'blue'], hu
plt.title("Amt_credit and Amt_goods_price", fontweight='bold')
plt.show()
```



Here we can observe that **Amt_goods_price and Amt_credit have linear relation.** And, most of the defaulters are under 1 million level.

In [71]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(data=num_data, x='AMT_INCOME_TOTAL', y='AMT_CREDIT', palette=['red', 'blue'], h
plt.title("Amt_credit and AMT_INCOME_TOTAL", fontweight='bold')
plt.show()
```



People with income less than 1 million is taking more number of loans. And, people who got credit/loans less than 150,000 are more likely to default.

Final Observations:

1. Female loan has less default rate. So, the bank should give a little bit priority to females.

2. Those clients who do not have any accompany should be focused.
3. Safest segementation of employment are workers, commercial associates and pensioners.
4. Client who have the higher education should be given more loans.
5. Married clients are safer than unmarried.
6. People having house/apartment are safer to provide loans.
7. Low-skill laborers and drivers should be given less priority as they have high probability of making defaults.
8. People having income less than 1 million and taking loans near to 1 million have higher chance of defaults. So, should not be given focus.
9. Married couples having children less than five are safe for providng loans.
10. Client having annuity less than 100K are safer side for the bank.