LOAN CASE STUDY SUBMISSION

Drive Link(For Loan_Casestudy.ipynb):
https://colab.research.google.com/drive/1wqmirQiOfb6PNA5_FT5AgNC7-hXBDg4n?usp=sharing

Drive Link(For IMDB final pdf submission):
https://drive.google.com/file/d/1kpcZwQ8Qz3oz3NKSrPFU99Xh6FJktbOO/view?usp=sharing

Project Description:

- This is a project about data analytics terms and research on the LOAN Dataset which is available in ".csv" file format.
- From simple dataframe reading/inspection to data cleaning, replacing NaN values with some value, Deleting Certain columns which are not of use to, analysis of the data by using certain graphs (like line/bar/box/PIE and etc.) and finding certain values or row/column like max age, education type and etc. These are the tasks or things that I have tried to find and analyze in this dataframe.
- Along with above all the points, I have tried to find some others insights in this dataset.

Approach:

Simply I have started this task by uploading/reading the dataframe and finding all the dataframe info i.e. dtypes, columns, and rows with other details along with importing some libraries like pandas, matplotlib, seaborn and etc. Furthermore, I have tried some different approaches and some other mine questions too, to find some extra insights available in the dataframe.

Tech-Stack Used:

I have used an online very much known "Google Collaboratory" which is a cloud version of the Jupyter notebook.

- Used platform: Google Collaboratory
- Used libraries: pandas, NumPy
- Used Language: Python
- Used System: Asus Vivobook 14 (Windows 11)

Insights:

- Maximum age of applicant is 70.

- Nearly 62% of loan application has approved. Whereas 18% has cancelled and 17% has refused by the bank.
- Female's application is more compare to Male Data.
- Business type 3 has maximum numbers and Industry type 8 has the lowest numbers of loan demand.
- Secondary educational applicant has applied maximum for the loan as compared to others.
- 68% of applicants own the property.
- Academic Degree holders have higher income.
- Previous contracts dataset shows, Refused applications were most as compare to approved.
- Degree holders had applied mostly for housing loan.

Results:

By doing this project I have achieved the following things:

- I found some insights.
- Practiced my data analysis knowledge.
- Developed self-confidence related to data analysis.
- Learned some different methods and attributes.

Drive link:
https://drive.google.com/file/d/10KuGzFik_OloRWWRsIpKdOTZTPH6abta/view?usp=sharing

final

March 17, 2022

0.1 Mounting Dataset From The Drive

[73]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

0.2 Importing all necessary libraries

[74]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

0.3 Reading Datset (application_data.csv)

[75]:
```
df = pd.read_csv('/content/drive/MyDrive/Final_dataset/application_data.csv')
```

Viewing first n = 10 rows from dataset

[76]:
```
df.head(10)
```

[76]:
```
   SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
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
```

	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	\
0	Y	0	202500.0	406597.5	24700.5	
1	N	0	270000.0	1293502.5	35698.5	
2	Y	0	67500.0	135000.0	6750.0	
3	Y	0	135000.0	312682.5	29686.5	
4	Y	0	121500.0	513000.0	21865.5	
5	Y	0	99000.0	490495.5	27517.5	
6	Y	1	171000.0	1560726.0	41301.0	
7	Y	0	360000.0	1530000.0	42075.0	
8	Y	0	112500.0	1019610.0	33826.5	
9	Y	0	135000.0	405000.0	20250.0	

	…	FLAG_DOCUMENT_18	FLAG_DOCUMENT_19	FLAG_DOCUMENT_20	FLAG_DOCUMENT_21	\
0	…	0	0	0	0	
1	…	0	0	0	0	
2	…	0	0	0	0	
3	…	0	0	0	0	
4	…	0	0	0	0	
5	…	0	0	0	0	
6	…	0	0	0	0	
7	…	0	0	0	0	
8	…	0	0	0	0	
9	…	0	0	0	0	

	AMT_REQ_CREDIT_BUREAU_HOUR	AMT_REQ_CREDIT_BUREAU_DAY	\
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	NaN	NaN	
4	0.0	0.0	
5	0.0	0.0	
6	0.0	0.0	
7	0.0	0.0	
8	0.0	0.0	
9	NaN	NaN	

	AMT_REQ_CREDIT_BUREAU_WEEK	AMT_REQ_CREDIT_BUREAU_MON	\
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	NaN	NaN	
4	0.0	0.0	
5	0.0	0.0	
6	0.0	1.0	
7	0.0	0.0	
8	0.0	0.0	

```
9                          NaN                          NaN

   AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
0                        0.0                         1.0
1                        0.0                         0.0
2                        0.0                         0.0
3                        NaN                         NaN
4                        0.0                         0.0
5                        1.0                         1.0
6                        1.0                         2.0
7                        0.0                         0.0
8                        0.0                         1.0
9                        NaN                         NaN

[10 rows x 122 columns]
```

0.4 Inspecting Dataframe

[77]:
```python
# inspecting columns
print('Inspecting Columns: ')
print(df.columns)

print('=================================================')

# inspecting shapes
print('Inspecting Shapes: ')
print(df.shape)

print('=================================================')

# inspecting datatypes
print('Inspecting Datatypes: ')
print(df.dtypes)
```

```
Inspecting Columns:
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)
=================================================
```

```
Inspecting Shapes:
(307511, 122)
=================================================
Inspecting Datatypes:
SK_ID_CURR                     int64
TARGET                         int64
NAME_CONTRACT_TYPE            object
CODE_GENDER                   object
FLAG_OWN_CAR                  object
                               …
AMT_REQ_CREDIT_BUREAU_DAY    float64
AMT_REQ_CREDIT_BUREAU_WEEK   float64
AMT_REQ_CREDIT_BUREAU_MON    float64
AMT_REQ_CREDIT_BUREAU_QRT    float64
AMT_REQ_CREDIT_BUREAU_YEAR   float64
Length: 122, dtype: object
```

0.5 Cleaning the Data

Finding number of null values in all the columns & rows.

[78]:
```python
# inspecting Null Values & finding Column-wise Null count Percentage

Clm_wise_null = df.isnull().sum(axis=0)

Missing_value = ((Clm_wise_null)/len(df)).sort_values(ascending=False)

round((Missing_value[:50]*100),2)
```

[78]:
```
COMMONAREA_MEDI                 69.87
COMMONAREA_AVG                  69.87
COMMONAREA_MODE                 69.87
NONLIVINGAPARTMENTS_MODE        69.43
NONLIVINGAPARTMENTS_AVG         69.43
NONLIVINGAPARTMENTS_MEDI        69.43
FONDKAPREMONT_MODE              68.39
LIVINGAPARTMENTS_MODE           68.35
LIVINGAPARTMENTS_AVG            68.35
LIVINGAPARTMENTS_MEDI           68.35
FLOORSMIN_AVG                   67.85
FLOORSMIN_MODE                  67.85
FLOORSMIN_MEDI                  67.85
YEARS_BUILD_MEDI                66.50
YEARS_BUILD_MODE                66.50
YEARS_BUILD_AVG                 66.50
OWN_CAR_AGE                     65.99
LANDAREA_MEDI                   59.38
```

```
LANDAREA_MODE                   59.38
LANDAREA_AVG                    59.38
BASEMENTAREA_MEDI               58.52
BASEMENTAREA_AVG                58.52
BASEMENTAREA_MODE               58.52
EXT_SOURCE_1                    56.38
NONLIVINGAREA_MODE              55.18
NONLIVINGAREA_AVG               55.18
NONLIVINGAREA_MEDI              55.18
ELEVATORS_MEDI                  53.30
ELEVATORS_AVG                   53.30
ELEVATORS_MODE                  53.30
WALLSMATERIAL_MODE              50.84
APARTMENTS_MEDI                 50.75
APARTMENTS_AVG                  50.75
APARTMENTS_MODE                 50.75
ENTRANCES_MEDI                  50.35
ENTRANCES_AVG                   50.35
ENTRANCES_MODE                  50.35
LIVINGAREA_AVG                  50.19
LIVINGAREA_MODE                 50.19
LIVINGAREA_MEDI                 50.19
HOUSETYPE_MODE                  50.18
FLOORSMAX_MODE                  49.76
FLOORSMAX_MEDI                  49.76
FLOORSMAX_AVG                   49.76
YEARS_BEGINEXPLUATATION_MODE    48.78
YEARS_BEGINEXPLUATATION_MEDI    48.78
YEARS_BEGINEXPLUATATION_AVG     48.78
TOTALAREA_MODE                  48.27
EMERGENCYSTATE_MODE             47.40
OCCUPATION_TYPE                 31.35
dtype: float64
```

[79]:
```python
# Listing columns having missing percentage greater than 30%

Nullcols = df.isnull().sum()
Nullcols = Nullcols[Nullcols.values>(0.3*len(Nullcols))]
print("Count of columns having null percentage greater than 30% are:",␣
 ↪len(Nullcols))
```

```
Count of columns having null percentage greater than 30% are: 64
```

Dropping above 64 columns

[81]:
```python
df.drop(Nullcols[Nullcols.values > 0.3].index, axis=1, inplace=True)

df.isnull().sum()/len(df)*100
```

```
[81]: SK_ID_CURR                     0.000000
      TARGET                         0.000000
      NAME_CONTRACT_TYPE             0.000000
      CODE_GENDER                    0.000000
      FLAG_OWN_CAR                   0.000000
      FLAG_OWN_REALTY                0.000000
      CNT_CHILDREN                   0.000000
      AMT_INCOME_TOTAL               0.000000
      AMT_CREDIT                     0.000000
      AMT_ANNUITY                    0.003902
      NAME_INCOME_TYPE               0.000000
      NAME_EDUCATION_TYPE            0.000000
      NAME_FAMILY_STATUS             0.000000
      NAME_HOUSING_TYPE              0.000000
      REGION_POPULATION_RELATIVE     0.000000
      DAYS_BIRTH                     0.000000
      DAYS_EMPLOYED                  0.000000
      DAYS_REGISTRATION              0.000000
      DAYS_ID_PUBLISH                0.000000
      FLAG_MOBIL                     0.000000
      FLAG_EMP_PHONE                 0.000000
      FLAG_WORK_PHONE                0.000000
      FLAG_CONT_MOBILE               0.000000
      FLAG_PHONE                     0.000000
      FLAG_EMAIL                     0.000000
      CNT_FAM_MEMBERS                0.000650
      REGION_RATING_CLIENT           0.000000
      REGION_RATING_CLIENT_W_CITY    0.000000
      WEEKDAY_APPR_PROCESS_START     0.000000
      HOUR_APPR_PROCESS_START        0.000000
      REG_REGION_NOT_LIVE_REGION     0.000000
      REG_REGION_NOT_WORK_REGION     0.000000
      LIVE_REGION_NOT_WORK_REGION    0.000000
      REG_CITY_NOT_LIVE_CITY         0.000000
      REG_CITY_NOT_WORK_CITY         0.000000
      LIVE_CITY_NOT_WORK_CITY        0.000000
      ORGANIZATION_TYPE              0.000000
      DAYS_LAST_PHONE_CHANGE         0.000325
      FLAG_DOCUMENT_2                0.000000
      FLAG_DOCUMENT_3                0.000000
      FLAG_DOCUMENT_4                0.000000
      FLAG_DOCUMENT_5                0.000000
      FLAG_DOCUMENT_6                0.000000
      FLAG_DOCUMENT_7                0.000000
      FLAG_DOCUMENT_8                0.000000
      FLAG_DOCUMENT_9                0.000000
      FLAG_DOCUMENT_10               0.000000
```

```
FLAG_DOCUMENT_11                0.000000
FLAG_DOCUMENT_12                0.000000
FLAG_DOCUMENT_13                0.000000
FLAG_DOCUMENT_14                0.000000
FLAG_DOCUMENT_15                0.000000
FLAG_DOCUMENT_16                0.000000
FLAG_DOCUMENT_17                0.000000
FLAG_DOCUMENT_18                0.000000
FLAG_DOCUMENT_19                0.000000
FLAG_DOCUMENT_20                0.000000
FLAG_DOCUMENT_21                0.000000
dtype: float64
```

Above, there are still some columns which are having some null values

like: AMT_ANNUITY, CNT_FAM_NUMBERS, DAYS_LAST_PHONE_CHANGE

0.6 Dealing with Null Values

[82]:
```
# Filling null values of AMT_ANNUITY with its median

med = df['AMT_ANNUITY'].median()
df.loc[df['AMT_ANNUITY'].isnull(),'AMT_ANNUITY'] = med
```

[83]:
```
df.isnull().sum().sort_values(ascending=True)
```

[83]:
```
SK_ID_CURR                     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
FLAG_DOCUMENT_2                0
FLAG_DOCUMENT_3                0
FLAG_DOCUMENT_4                0
FLAG_DOCUMENT_5                0
FLAG_DOCUMENT_6                0
FLAG_DOCUMENT_7                0
FLAG_DOCUMENT_8                0
FLAG_DOCUMENT_9                0
FLAG_DOCUMENT_10               0
FLAG_DOCUMENT_11               0
FLAG_DOCUMENT_12               0
FLAG_DOCUMENT_13               0
FLAG_DOCUMENT_14               0
```

```
FLAG_DOCUMENT_15               0
FLAG_DOCUMENT_16               0
FLAG_DOCUMENT_17               0
FLAG_DOCUMENT_18               0
FLAG_DOCUMENT_19               0
HOUR_APPR_PROCESS_START        0
FLAG_DOCUMENT_20               0
WEEKDAY_APPR_PROCESS_START     0
REGION_RATING_CLIENT           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                    0
NAME_INCOME_TYPE               0
NAME_EDUCATION_TYPE            0
NAME_FAMILY_STATUS             0
REGION_RATING_CLIENT_W_CITY    0
NAME_HOUSING_TYPE              0
DAYS_BIRTH                     0
DAYS_EMPLOYED                  0
DAYS_REGISTRATION              0
DAYS_ID_PUBLISH                0
FLAG_MOBIL                     0
FLAG_EMP_PHONE                 0
FLAG_WORK_PHONE                0
FLAG_CONT_MOBILE               0
FLAG_PHONE                     0
FLAG_EMAIL                     0
REGION_POPULATION_RELATIVE     0
FLAG_DOCUMENT_21               0
DAYS_LAST_PHONE_CHANGE         1
CNT_FAM_MEMBERS                2
dtype: int64
```

Now above remaining columns have negligible null values and we have successfully removed all null values from AMT_ANNUITY columns.

[84]:
```
# Now checking missing values respected to rows

df.isnull().sum(axis=1).sort_values(ascending=False)
```

[84]:
```
187348    1
15709     1
41982     1
0         0
205009    0
         ..
102504    0
102503    0
102502    0
102501    0
307510    0
Length: 307511, dtype: int64
```

By observing above output we can say almost every rows are having negligible null values.

Now, some insights related to column values

[85]:
```
# Now we will look in some columns for null/unique values and others insights␣
↪related to values.

print('CODE_GENDER Unique Values: ')
df.CODE_GENDER.value_counts()
```

```
CODE_GENDER Unique Values:
```

[85]:
```
F      202448
M      105059
XNA         4
Name: CODE_GENDER, dtype: int64
```

Column **CODE_GENDER** has an unknown value which has F = female, M = male, and the third value is **XNA**. Simply, we can keep it as it is or we can replace these values with **F = female** because they are the values that are most commonly occurring than men and this will not affect our analysis or we can also consider them as the third gender.

[86]:
```
# updating the column CODE_GENDER with "F" and eliminating "XNA" completely

df.loc[df['CODE_GENDER'] == 'XNA', 'CODE_GENDER'] = 'F'
df['CODE_GENDER'].value_counts()
```

[86]:
```
F    202452
M    105059
Name: CODE_GENDER, dtype: int64
```

[87]:
```
print(df['ORGANIZATION_TYPE'].describe())
print('==============================')
```

```python
print('ORGANIZATION_TYPE Unique Values: ')
df.ORGANIZATION_TYPE.value_counts()
```

```
count                      307511
unique                         58
top        Business Entity Type 3
freq                        67992
Name: ORGANIZATION_TYPE, dtype: object
==============================
ORGANIZATION_TYPE Unique Values:
```

[87]:

```
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              3492
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
```

Column **'ORGANIZATION_TYPE'** having total count of **307511** rows and out of which **55374** rows are having **'XNA'** values which means not having any type of information related to the same. To deal with this, we can keep it as it is and in the analysis/presentation part we can say that these are the values of the applicant which are unknown for the ORGANIZATION_TYPE column and consider these are the people which are not having any connection with any type of the 'ORGANIZATION'. Otherwise, we can drop it from the column as it is approx 18% of the column. Hence, it will not have any major impact on our analysis of the database.

Dropping "XNA" values from the rows of "ORGANIZATION_TYPE"

[88]:
```
df = df.drop(df.loc[df['ORGANIZATION_TYPE'] == 'XNA'].index)

df[df['ORGANIZATION_TYPE'] == 'XNA'].shape
```

[88]:
```
(0, 58)
```

Observing the below output related to column **NAME_FAMILY_STATUS**, we can say that there are only two values that are unknown simply we can modify it to any unique values of NAME_FAMILY_STATUS column as it is only 2 unknown values, it will not affect our analysis.

[89]:
```
print('NAME_FAMILY_STATUS Unique Values: ')

df.NAME_FAMILY_STATUS.value_counts()
```

```
NAME_FAMILY_STATUS Unique Values:
```

[89]:
```
Married                 163914
Single / not married     39316
Civil marriage           26197
Separated                16000
Widow                     6708
Unknown                      2
Name: NAME_FAMILY_STATUS, dtype: int64
```

[90]:
```
df.loc[df['NAME_FAMILY_STATUS']=='Unknown', 'NAME_FAMILY_STATUS'] = 'Single /␣
↪not married'

df['NAME_FAMILY_STATUS'].value_counts()
```

[90]:
```
Married                 163914
Single / not married     39318
Civil marriage           26197
Separated                16000
Widow                     6708
Name: NAME_FAMILY_STATUS, dtype: int64
```

Successfully we have modified ”Unknown” value to the ”Single / not married”.

Column **NAME_FAMILY_STATUS** having value **Civil marriage** which is same as **Married** and **Single / not married** can be converted to **Single**

[91]:
```
# Converting values related to NAME_FAMILY_STATUS column

df['NAME_FAMILY_STATUS'].replace({'Civil marriage':'Married', 'Single / not␣
↪married':'Single'}, inplace=True)
```

[92]:
```
df.NAME_FAMILY_STATUS.value_counts()
```

[92]:
```
Married      190111
Single        39318
Separated     16000
Widow          6708
Name: NAME_FAMILY_STATUS, dtype: int64
```

0.7 Handling Outliers

[94]:
```
# describe() method gives count, mean, standard deviation, min, max and other␣
↪values of available float & integer columns.

df.describe()
```

[94]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	\
count	252137.000000	252137.000000	252137.000000	2.521370e+05	
mean	278114.643103	0.086600	0.498515	1.759141e+05	
std	102815.635309	0.281248	0.763161	2.588516e+05	
min	100002.000000	0.000000	0.000000	2.565000e+04	
25%	189035.000000	0.000000	0.000000	1.125000e+05	
50%	278064.000000	0.000000	0.000000	1.575000e+05	
75%	367165.000000	0.000000	1.000000	2.115000e+05	
max	456255.000000	1.000000	19.000000	1.170000e+08	

	AMT_CREDIT	AMT_ANNUITY	REGION_POPULATION_RELATIVE	DAYS_BIRTH	\
count	2.521370e+05	252137.000000	252137.000000	252137.000000	
mean	6.113985e+05	27812.186704	0.020894	-14769.133174	
std	4.065272e+05	14647.424282	0.013874	3662.573769	
min	4.500000e+04	1980.000000	0.000290	-25200.000000	
25%	2.779695e+05	17073.000000	0.010006	-17563.000000	
50%	5.212800e+05	25834.500000	0.018850	-14573.000000	
75%	8.292240e+05	35617.500000	0.028663	-11775.000000	
max	4.050000e+06	258025.500000	0.072508	-7489.000000	

	DAYS_EMPLOYED	DAYS_REGISTRATION	…	FLAG_DOCUMENT_12	\
count	252137.000000	252137.000000	…	252137.000000	
mean	-2384.169325	-4635.430849	…	0.000008	
std	2338.360162	3252.169156	…	0.002816	
min	-17912.000000	-22928.000000	…	0.000000	
25%	-3175.000000	-6952.000000	…	0.000000	
50%	-1648.000000	-4265.000000	…	0.000000	
75%	-767.000000	-1845.000000	…	0.000000	
max	0.000000	0.000000	…	1.000000	

	FLAG_DOCUMENT_13	FLAG_DOCUMENT_14	FLAG_DOCUMENT_15	FLAG_DOCUMENT_16	\
count	252137.000000	252137.000000	252137.000000	252137.000000	
mean	0.004244	0.003534	0.001444	0.011926	
std	0.065006	0.059341	0.037968	0.108554	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	

	FLAG_DOCUMENT_17	FLAG_DOCUMENT_18	FLAG_DOCUMENT_19	FLAG_DOCUMENT_20	\
count	252137.000000	252137.000000	252137.000000	252137.000000	
mean	0.000321	0.009836	0.000710	0.000615	
std	0.017921	0.098687	0.026635	0.024786	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	

```
75%            0.000000          0.000000          0.000000          0.000000
max            1.000000          1.000000          1.000000          1.000000

       FLAG_DOCUMENT_21
count     252137.000000
mean           0.000409
std            0.020207
min            0.000000
25%            0.000000
50%            0.000000
75%            0.000000
max            1.000000

[8 rows x 48 columns]
```

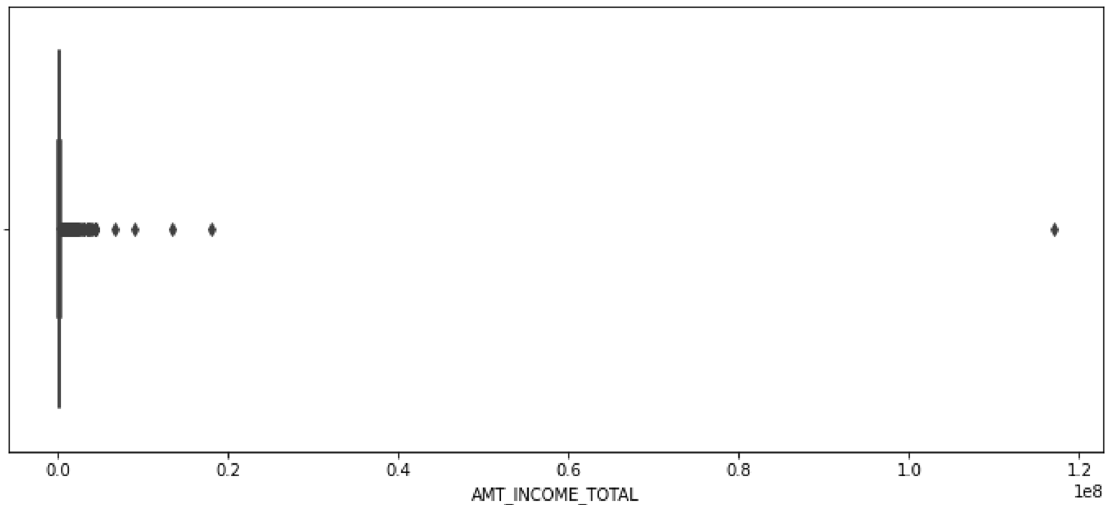
Analyzing AMT_INCOME_TOTAL column for outliers

[95]:
```
df.AMT_INCOME_TOTAL.describe()
```

[95]:
```
count    2.521370e+05
mean     1.759141e+05
std      2.588516e+05
min      2.565000e+04
25%      1.125000e+05
50%      1.575000e+05
75%      2.115000e+05
max      1.170000e+08
Name: AMT_INCOME_TOTAL, dtype: float64
```

[96]:
```
plt.figure(figsize=[12,5])
sns.boxplot(df.AMT_INCOME_TOTAL)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



By observing above graph we can say that, there is only single entry which is way higher than others.

[98]:
```
temp = df['AMT_INCOME_TOTAL'].sort_values(ascending=False)

temp[:20]
```

[98]:
```
12840     117000000.0
203693     18000090.0
246858     13500000.0
77768       9000000.0
131127      6750000.0
103006      4500000.0
204564      4500000.0
187833      4500000.0
287463      4500000.0
181698      3950059.5
20216       3825000.0
49645       3600000.0
284311      3600000.0
86026       3375000.0
82846       3375000.0
101007      3150000.0
248159      3150000.0
107926      2930026.5
258773      2700000.0
298082      2475000.0
Name: AMT_INCOME_TOTAL, dtype: float64
```

By observing above output we can say that, that most outlier entry's value is "117000000.0". Others are continuous and we can retain them as income is normally spread.

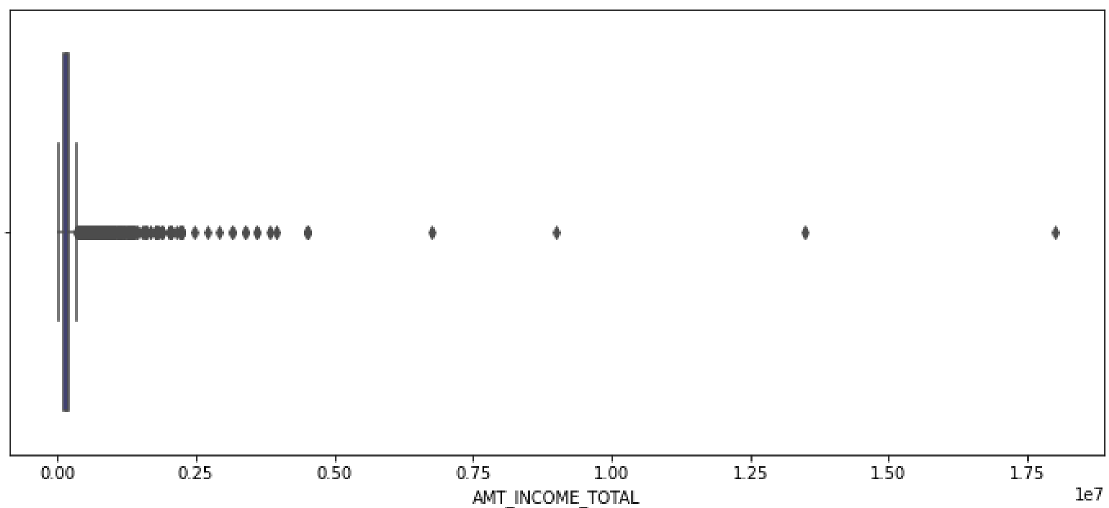
[99]:
```
df = df[~(df.AMT_INCOME_TOTAL > 0.2*10**(8))]
df.shape
```

[99]: (252136, 58)

[100]:
```
# Checking AMT_INCOME_TOTAL again

plt.figure(figsize=[12,5])
sns.boxplot(df.AMT_INCOME_TOTAL, color='blue')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



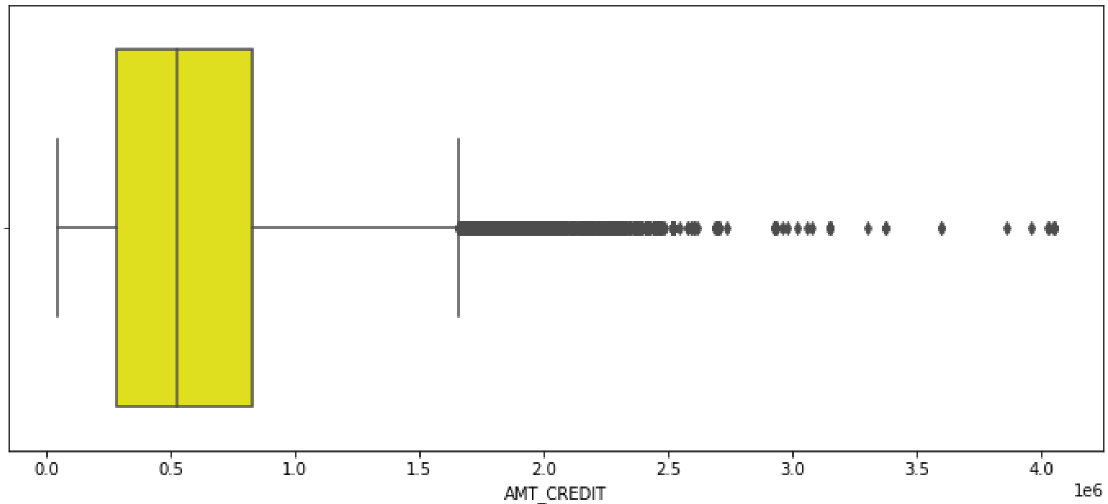
There are still many outliers but these all are spreaded in meaningful manner and we can keep this as it is.

Analyzinge AMT_CREDIT column for outliers

[101]:
```
plt.figure(figsize=[12,5])
sns.boxplot(df.AMT_CREDIT, color = 'yellow')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
```

```
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



We can categorize amt_credit in various groups of quantiles/ranges to understand more effeciently this AMT_CREDIT column but by observing the above graph we can say that there is no such outliers which will affect our analysis. So, we will keep this as it is.

Analyzing DAYS_BIRTH column for outliers

[102]:
```
df.DAYS_BIRTH.describe()
```

[102]:
```
count    252136.000000
mean     -14769.141717
std        3662.578520
min      -25200.000000
25%      -17563.000000
50%      -14573.000000
75%      -11775.000000
max       -7489.000000
Name: DAYS_BIRTH, dtype: float64
```

By observing above describe method of DAYS_BIRTH column we can say that present all values of this columns are less than **Zero** as **max = -7489**

Adding a new column **"Updated_DAYS_BIRTH"** in years to transforms the present values to positive and analyse them.

[103]:
```
# using numpy.ceil() method, gives us rounded value of each element
# and it is always greater than equal to given value.

df['Updated_DAYS_BIRTH'] = np.ceil(df.DAYS_BIRTH/-365)
```

```python
df.Updated_DAYS_BIRTH.describe()
```

[103]:
```
count    252136.000000
mean         40.960283
std          10.033044
min          21.000000
25%          33.000000
50%          40.000000
75%          49.000000
max          70.000000
Name: Updated_DAYS_BIRTH, dtype: float64
```

Now we have all positive values and maximum age is 70 which is okay.

As DAYS_BIRTH columns had negative values of time/age similarly, after observing the dataframe others columns related to Dates are having negative values too. These columns are: **DAYS_EMPLOYED, DAYS_REGISTRATION and DAYS_ID_PUBLISH**

Analyzing DAYS_EMPLOYED

[104]:
```python
df['Updated_DAYS_EMPLOYED'] = np.ceil(df.DAYS_EMPLOYED/-365)

print(df.Updated_DAYS_EMPLOYED.describe())
```

```
count    252136.000000
mean          7.034612
std           6.415832
min          -0.000000
25%           3.000000
50%           5.000000
75%           9.000000
max          50.000000
Name: Updated_DAYS_EMPLOYED, dtype: float64
```

Analyzing DAYS_REGISTRATION and modifying to +ve time in the years

[105]:
```python
df['Updated_DAYS_REGISTRATION'] = np.ceil(df.DAYS_REGISTRATION/-365)

print(df.Updated_DAYS_REGISTRATION.describe())
```

```
count    252136.000000
mean         13.205334
std           8.905717
min          -0.000000
25%           6.000000
50%          12.000000
75%          20.000000
max          63.000000
Name: Updated_DAYS_REGISTRATION, dtype: float64
```

Analyzing DAYS_ID_PUBLISH and modifying to +ve time in the years

[106]:
```
df['Updated_DAYS_ID_PUBLISH'] = np.ceil(df.DAYS_ID_PUBLISH/-365)

print(df.Updated_DAYS_ID_PUBLISH.describe())
```

```
count    252136.000000
mean          8.181196
std           4.149690
min          -0.000000
25%           5.000000
50%           8.000000
75%          12.000000
max          20.000000
Name: Updated_DAYS_ID_PUBLISH, dtype: float64
```

Successfully we'he modified all the columns to +ve value in terms of year related to date. Thsese columns are: DAYS_BIRTH,DAYS_EMPLOYED, DAYS_REGISTRATION and DAYS_ID_PUBLISH

[112]:
```
df.shape
```

[112]: (252136, 62)

[113]:
```
# Dropping all old columns

df.drop(['DAYS_BIRTH','DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH'],␣
 ↪axis=1, inplace=True)

df.shape
```

[113]: (252136, 58)

We've successfully dropped above 4 columns.

==

0.8 Analyzing Previous application file

==

previous_application can be used to find the old details of applicant and it will help to determine whether applicant is new or not. previous_application dataframe can played an important role in failure/acceptance of loan.

[114]:
```
# Reading previous_application file from the drive location

previous_df = pd.read_csv('/content/drive/MyDrive/Final_dataset/
 ↪previous_application.csv')
```

```python
previous_df.head(10)
```

[114]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	\
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	

	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	WEEKDAY_APPR_PROCESS_START	\
0	17145.0	0.0	17145.0	SATURDAY	
1	679671.0	NaN	607500.0	THURSDAY	
2	136444.5	NaN	112500.0	TUESDAY	
3	470790.0	NaN	450000.0	MONDAY	
4	404055.0	NaN	337500.0	THURSDAY	
5	340573.5	NaN	315000.0	SATURDAY	
6	0.0	NaN	NaN	TUESDAY	
7	0.0	NaN	NaN	MONDAY	
8	0.0	NaN	NaN	MONDAY	
9	0.0	NaN	NaN	SATURDAY	

	HOUR_APPR_PROCESS_START	…	NAME_SELLER_INDUSTRY	CNT_PAYMENT	\
0	15	…	Connectivity	12.0	
1	11	…	XNA	36.0	
2	11	…	XNA	12.0	
3	7	…	XNA	12.0	
4	9	…	XNA	24.0	
5	8	…	XNA	18.0	
6	11	…	XNA	NaN	
7	7	…	XNA	NaN	
8	15	…	XNA	NaN	
9	15	…	XNA	NaN	

	NAME_YIELD_GROUP	PRODUCT_COMBINATION	DAYS_FIRST_DRAWING	\
0	middle	POS mobile with interest	365243.0	
1	low_action	Cash X-Sell: low	365243.0	
2	high	Cash X-Sell: high	365243.0	
3	middle	Cash X-Sell: middle	365243.0	
4	high	Cash Street: high	NaN	
5	low_normal	Cash X-Sell: low	365243.0	
6	XNA	Cash	NaN	

```
7              XNA                       Cash                 NaN
8              XNA                       Cash                 NaN
9              XNA                       Cash                 NaN

  DAYS_FIRST_DUE DAYS_LAST_DUE_1ST_VERSION  DAYS_LAST_DUE DAYS_TERMINATION  \
0          -42.0                     300.0          -42.0            -37.0
1         -134.0                     916.0       365243.0         365243.0
2         -271.0                      59.0       365243.0         365243.0
3         -482.0                    -152.0         -182.0           -177.0
4            NaN                       NaN            NaN              NaN
5         -654.0                    -144.0         -144.0           -137.0
6            NaN                       NaN            NaN              NaN
7            NaN                       NaN            NaN              NaN
8            NaN                       NaN            NaN              NaN
9            NaN                       NaN            NaN              NaN

  NFLAG_INSURED_ON_APPROVAL
0                       0.0
1                       1.0
2                       1.0
3                       1.0
4                       NaN
5                       1.0
6                       NaN
7                       NaN
8                       NaN
9                       NaN

[10 rows x 37 columns]
```

Finding shape, columns and datatypes of dataframe by using info() method.

[115]:
```python
previous_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   SK_ID_PREV                   1670214 non-null  int64
 1   SK_ID_CURR                   1670214 non-null  int64
 2   NAME_CONTRACT_TYPE           1670214 non-null  object
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_DOWN_PAYMENT             774370 non-null   float64
 7   AMT_GOODS_PRICE              1284699 non-null  float64
 8   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object
```

```
 9   HOUR_APPR_PROCESS_START      1670214 non-null  int64
 10  FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object
 16  NAME_CONTRACT_STATUS         1670214 non-null  object
 17  DAYS_DECISION                1670214 non-null  int64
 18  NAME_PAYMENT_TYPE            1670214 non-null  object
 19  CODE_REJECT_REASON           1670214 non-null  object
 20  NAME_TYPE_SUITE              849809 non-null   object
 21  NAME_CLIENT_TYPE             1670214 non-null  object
 22  NAME_GOODS_CATEGORY          1670214 non-null  object
 23  NAME_PORTFOLIO               1670214 non-null  object
 24  NAME_PRODUCT_TYPE            1670214 non-null  object
 25  CHANNEL_TYPE                 1670214 non-null  object
 26  SELLERPLACE_AREA             1670214 non-null  int64
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object
 30  PRODUCT_COMBINATION          1669868 non-null  object
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
```

[116]:

```
(previous_df.isna().mean() * 100).sort_values(ascending=True)
```

[116]:

```
SK_ID_PREV                    0.000000
NAME_YIELD_GROUP              0.000000
NAME_SELLER_INDUSTRY          0.000000
SELLERPLACE_AREA              0.000000
CHANNEL_TYPE                  0.000000
NAME_PRODUCT_TYPE             0.000000
NAME_PORTFOLIO                0.000000
NAME_GOODS_CATEGORY           0.000000
NAME_CLIENT_TYPE              0.000000
CODE_REJECT_REASON            0.000000
DAYS_DECISION                 0.000000
NAME_CONTRACT_STATUS          0.000000
NAME_CASH_LOAN_PURPOSE        0.000000
NAME_PAYMENT_TYPE             0.000000
```

```
AMT_APPLICATION                 0.000000
NFLAG_LAST_APPL_IN_DAY          0.000000
FLAG_LAST_APPL_PER_CONTRACT     0.000000
HOUR_APPR_PROCESS_START         0.000000
WEEKDAY_APPR_PROCESS_START      0.000000
NAME_CONTRACT_TYPE              0.000000
SK_ID_CURR                      0.000000
AMT_CREDIT                      0.000060
PRODUCT_COMBINATION             0.020716
CNT_PAYMENT                    22.286366
AMT_ANNUITY                    22.286665
AMT_GOODS_PRICE                23.081773
DAYS_LAST_DUE                  40.298129
DAYS_LAST_DUE_1ST_VERSION      40.298129
DAYS_FIRST_DUE                 40.298129
DAYS_FIRST_DRAWING             40.298129
NFLAG_INSURED_ON_APPROVAL      40.298129
DAYS_TERMINATION               40.298129
NAME_TYPE_SUITE                49.119754
AMT_DOWN_PAYMENT               53.636480
RATE_DOWN_PAYMENT              53.636480
RATE_INTEREST_PRIMARY          99.643698
RATE_INTEREST_PRIVILEGED       99.643698
dtype: float64
```

isna() method returns boolean value where 1 represents previous one was successful otherwise not or failure and why. If successful, is the loan over or not, was there any due or not.

Normalize NAME_CONTRACT_STATUS Column

[117]:
```
previous_df.NAME_CONTRACT_STATUS.value_counts(normalize=True)*100
```

[117]:
```
Approved        62.074740
Canceled        18.938831
Refused         17.403638
Unused offer     1.582791
Name: NAME_CONTRACT_STATUS, dtype: float64
```

Normalize FLAG_LAST_APPL_PER_CONTRACT Column

[118]:
```
previous_df.FLAG_LAST_APPL_PER_CONTRACT.value_counts(normalize=True)
```

[118]:
```
Y    0.994926
N    0.005074
Name: FLAG_LAST_APPL_PER_CONTRACT, dtype: float64
```

Let us drop all N values and keep only previous application

[119]:
```
previous_df = previous_df[previous_df.FLAG_LAST_APPL_PER_CONTRACT == 'Y']
previous_df.FLAG_LAST_APPL_PER_CONTRACT.value_counts(normalize=True)
```

[119]:
```
Y    1.0
Name: FLAG_LAST_APPL_PER_CONTRACT, dtype: float64
```

[120]:
```
previous_df.NFLAG_LAST_APPL_IN_DAY.value_counts(normalize=True)
```

[120]:
```
1    0.999527
0    0.000473
Name: NFLAG_LAST_APPL_IN_DAY, dtype: float64
```

Dropping all duplicates & sorting based on application id

[121]:
```
previous_df = previous_df.sort_values('SK_ID_PREV', ascending=False).
 ↪drop_duplicates('SK_ID_CURR')

previous_df.shape
```

[121]:
```
(338857, 37)
```

The columns of interest from this dataset are SK_ID_CURR, AMT_CREDIT, NAME_CONTRACT_STATUS, CODE_REJECT_REASON, NAME_YIELD_GROUP and DAYS_TERMINATION. Remaining can be dropped for the time being

[122]:
```
previous_updated_df = previous_df[['SK_ID_CURR', 'AMT_CREDIT',␣
 ↪'NAME_CONTRACT_STATUS', 'CODE_REJECT_REASON', 'NAME_YIELD_GROUP',␣
 ↪'DAYS_TERMINATION']]
previous_updated_df.head()
```

[122]:
```
        SK_ID_CURR  AMT_CREDIT NAME_CONTRACT_STATUS CODE_REJECT_REASON  \
205485      406596    30912.75         Unused offer             CLIENT
717142      140761    41499.00         Unused offer             CLIENT
886179      237546    60673.50              Refused              LIMIT
359118      100125    59503.50              Refused                SCO
70058       250234   108180.00              Refused                SCO

       NAME_YIELD_GROUP  DAYS_TERMINATION
205485              XNA               NaN
717142              XNA               NaN
886179           middle               NaN
359118           middle               NaN
70058        low_action               NaN
```

As this previous_updated_df dataframe is the taken from previous_df, we can modify each column of it with prefix ”prev”. It will help us to recognize in large dataset and comparision with the columns of application dataframe.

[123]:
```
new_names = {'AMT_CREDIT': 'PREV_AMT_CREDIT', 'NAME_CONTRACT_STATUS':␣
↪'PREV_CONTRACT_STATUS',
             'DAYS_TERMINATION':'PREV_DAYS_TERMINATION', 'CODE_REJECT_REASON':
↪'PREV_REJECT_REASON',
            'NAME_YIELD_GROUP':'PREV_YIELD_GROUP'}

previous_updated_df = previous_updated_df.rename(columns=new_names)
previous_updated_df.head()
```

[123]:
```
        SK_ID_CURR  PREV_AMT_CREDIT PREV_CONTRACT_STATUS PREV_REJECT_REASON  \
205485      406596         30912.75         Unused offer             CLIENT
717142      140761         41499.00         Unused offer             CLIENT
886179      237546         60673.50              Refused              LIMIT
359118      100125         59503.50              Refused                SCO
70058       250234        108180.00              Refused                SCO

       PREV_YIELD_GROUP  PREV_DAYS_TERMINATION
205485              XNA                    NaN
717142              XNA                    NaN
886179           middle                    NaN
359118           middle                    NaN
70058        low_action                    NaN
```

Analyzing PREV_DAYS_TERMINATION column

[124]:
```
previous_updated_df.PREV_DAYS_TERMINATION.value_counts(normalize=True).
↪sort_values(ascending=True)
```

[124]:
```
-2790.0       0.000004
-2743.0       0.000004
-2796.0       0.000004
-2804.0       0.000004
-2758.0       0.000004
                …
-144.0        0.000902
-17.0         0.000902
-15.0         0.000906
-9.0          0.000911
 365243.0     0.232947
Name: PREV_DAYS_TERMINATION, Length: 2785, dtype: float64
```

by observing above output we can clearly say that value **365243.0** is something impossible value because its related to termination days and the value 365243 is nearly 1000 years. Hence, this value is impossible and can be replaced by NaN because its having large percentage of total PREV_DAYS_TERMINATION columns, so dropping can be risky.

replacing "365243" with NaN

[125]:
```
previous_updated_df.PREV_DAYS_TERMINATION.replace({365243.0 : 'NaN'},inplace =␣
↪True)
```

Checking replacement of "365243.0" with "NaN"

[126]:
```
previous_updated_df.PREV_DAYS_TERMINATION.value_counts(normalize=True).
↪sort_values(ascending=True)
```

[126]:
```
-2790.0    0.000004
-2743.0    0.000004
-2796.0    0.000004
-2804.0    0.000004
-2758.0    0.000004
             …
-144.0     0.000902
-17.0      0.000902
-15.0      0.000906
-9.0       0.000911
NaN        0.232947
Name: PREV_DAYS_TERMINATION, Length: 2785, dtype: float64
```

===

0.9 Merging df & previous_updated_df

Here,

df = application_data.csv

previous_updated_df = previous_application.csv

===

[127]:
```
df = pd.merge(left=df,right=previous_updated_df, how='left',␣
↪left_on='SK_ID_CURR', right_on='SK_ID_CURR')

df.head(10)
```

[127]:
```
   SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
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      100012       0    Revolving loans           M            N
9      100014       0         Cash loans           F            N
```

```
  FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  \
0               Y             0          202500.0    406597.5      24700.5
1               N             0          270000.0   1293502.5      35698.5
2               Y             0           67500.0    135000.0       6750.0
3               Y             0          135000.0    312682.5      29686.5
4               Y             0          121500.0    513000.0      21865.5
5               Y             0           99000.0    490495.5      27517.5
6               Y             1          171000.0   1560726.0      41301.0
7               Y             0          360000.0   1530000.0      42075.0
8               Y             0          135000.0    405000.0      20250.0
9               Y             1          112500.0    652500.0      21177.0

   … FLAG_DOCUMENT_21 Updated_DAYS_BIRTH Updated_DAYS_EMPLOYED  \
0  …                0               26.0                   2.0
1  …                0               46.0                   4.0
2  …                0               53.0                   1.0
3  …                0               53.0                   9.0
4  …                0               55.0                   9.0
5  …                0               47.0                   5.0
6  …                0               38.0                   9.0
7  …                0               52.0                   2.0
8  …                0               40.0                   6.0
9  …                0               28.0                   2.0

  Updated_DAYS_REGISTRATION  Updated_DAYS_ID_PUBLISH  PREV_AMT_CREDIT  \
0                      10.0                      6.0         179055.0
1                       4.0                      1.0         348637.5
2                      12.0                      7.0          20106.0
3                      27.0                      7.0              0.0
4                      12.0                     10.0         284400.0
5                      14.0                      2.0         501975.0
6                       4.0                      2.0          88632.0
7                      13.0                      7.0         260811.0
8                      40.0                     11.0         114273.0
9                      13.0                      3.0          73800.0

   PREV_CONTRACT_STATUS  PREV_REJECT_REASON  PREV_YIELD_GROUP  \
0              Approved                 XAP        low_normal
1              Approved                 XAP            middle
2              Approved                 XAP            middle
3              Canceled                 XAP               XNA
4              Approved                 XAP            middle
5              Approved                 XAP            middle
6              Approved                 XAP            middle
7              Approved                 XAP        low_action
8              Approved                 XAP              high
```

```
9               Approved                    XAP             middle

   PREV_DAYS_TERMINATION
0                  -17.0
1                 -639.0
2                 -714.0
3                    NaN
4                    NaN
5                 -388.0
6                    NaN
7                 -762.0
8                 -144.0
9                    NaN

[10 rows x 63 columns]
```

Checking null values of PREV_CONTRACT_STATUS column

[128]:
```
df.PREV_CONTRACT_STATUS.isna().mean()
```

[128]:
```
0.05421280578735286
```

There are still some null values present in the PREV_CONTRACT_STATUS column. It can be filled as the **"First_time_application"** because they don't have any present record in previous database.

[129]:
```
# Replacing na of PREV_CONTRACT_STATUS with First_time_application

df.PREV_CONTRACT_STATUS.fillna('First_time_application', inplace=True)

df.PREV_CONTRACT_STATUS.isna().mean()
```

[129]:
```
0.0
```

Successfully, we've replaced all the NaN values with "First_time_application".

==

0.10 DATA ANALYSIS

==

Available columns after the merge operation

[130]:
```
df.columns.sort_values(ascending=True)
```

[130]:
```
Index(['AMT_ANNUITY', 'AMT_CREDIT', 'AMT_INCOME_TOTAL', 'CNT_CHILDREN',
       'CNT_FAM_MEMBERS', 'CODE_GENDER', 'DAYS_LAST_PHONE_CHANGE',
       'FLAG_CONT_MOBILE', '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_2',
       'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21', 'FLAG_DOCUMENT_3',
       'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6',
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9', 'FLAG_EMAIL',
       'FLAG_EMP_PHONE', 'FLAG_MOBIL', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY',
       'FLAG_PHONE', 'FLAG_WORK_PHONE', 'HOUR_APPR_PROCESS_START',
       'LIVE_CITY_NOT_WORK_CITY', 'LIVE_REGION_NOT_WORK_REGION',
       'NAME_CONTRACT_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'NAME_INCOME_TYPE', 'ORGANIZATION_TYPE',
       'PREV_AMT_CREDIT', 'PREV_CONTRACT_STATUS', 'PREV_DAYS_TERMINATION',
       'PREV_REJECT_REASON', 'PREV_YIELD_GROUP', 'REGION_POPULATION_RELATIVE',
       'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY',
       'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY',
       'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'SK_ID_CURR', 'TARGET', 'Updated_DAYS_BIRTH', 'Updated_DAYS_EMPLOYED',
       'Updated_DAYS_ID_PUBLISH', 'Updated_DAYS_REGISTRATION',
       'WEEKDAY_APPR_PROCESS_START'],
      dtype='object')
```

0.11 UNIVARIATE ANALYSIS

Analyzing Gender distribution in the data

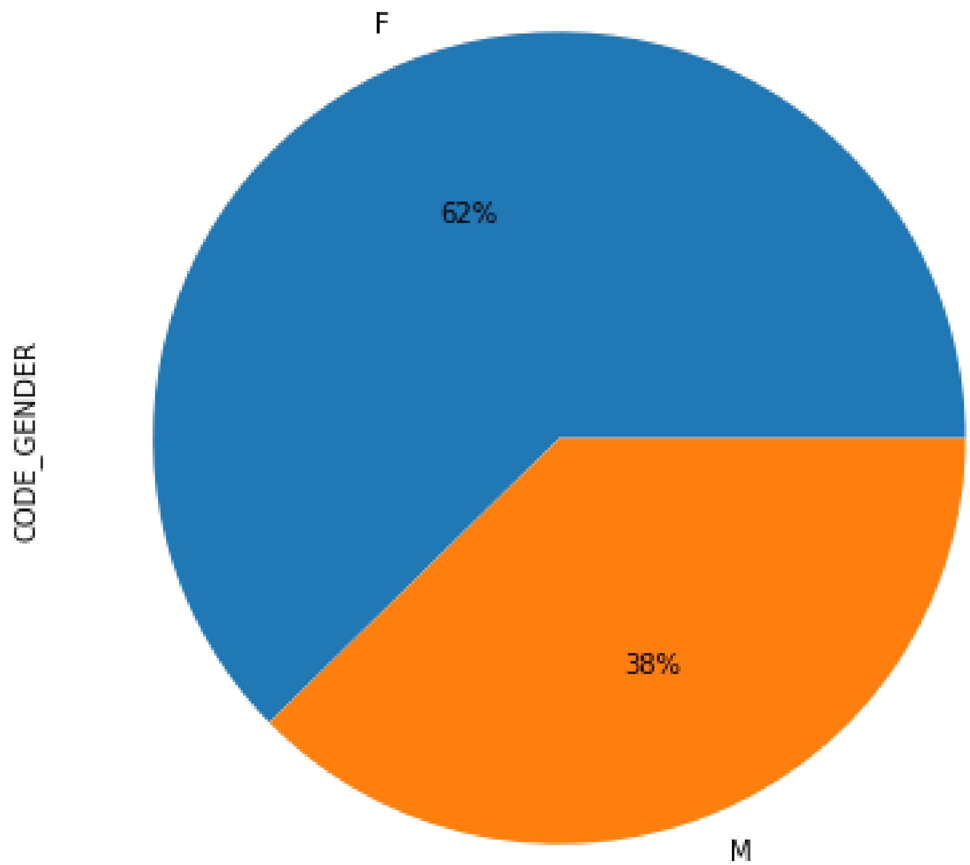
[131]:
```
df.CODE_GENDER.value_counts(normalize=True)*100
```

[131]:
```
F    62.339372
M    37.660628
Name: CODE_GENDER, dtype: float64
```

Pie Chart for Gender Distribution

[204]:
```
plt.figure(figsize=(7,7))
df.CODE_GENDER.value_counts(normalize=True).plot.pie(autopct='%1.0f%%')
```

[204]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2029391b90>
```



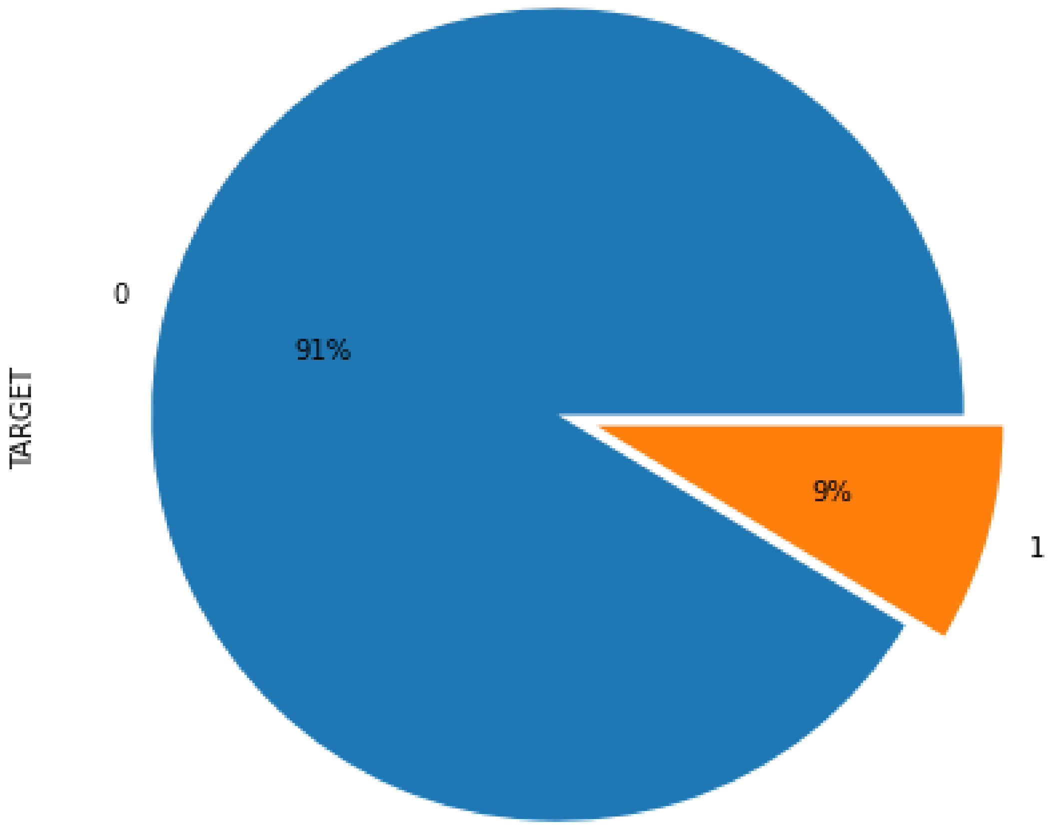
Analyze Target Column

[133]:
```
df.TARGET.value_counts(normalize=True)*100
```

[133]:
```
0    91.340388
1     8.659612
Name: TARGET, dtype: float64
```

Pie Chart for "TARGET"

[134]:
```
# pie chart of target_cat
my_explode=(0,0.1)
plt.figure(figsize=(7,7))
df.TARGET.value_counts(normalize=True).plot.pie(autopct='%1.0f%%',␣
↪explode=my_explode)
plt.show()
```



Analyzing Occupation Type

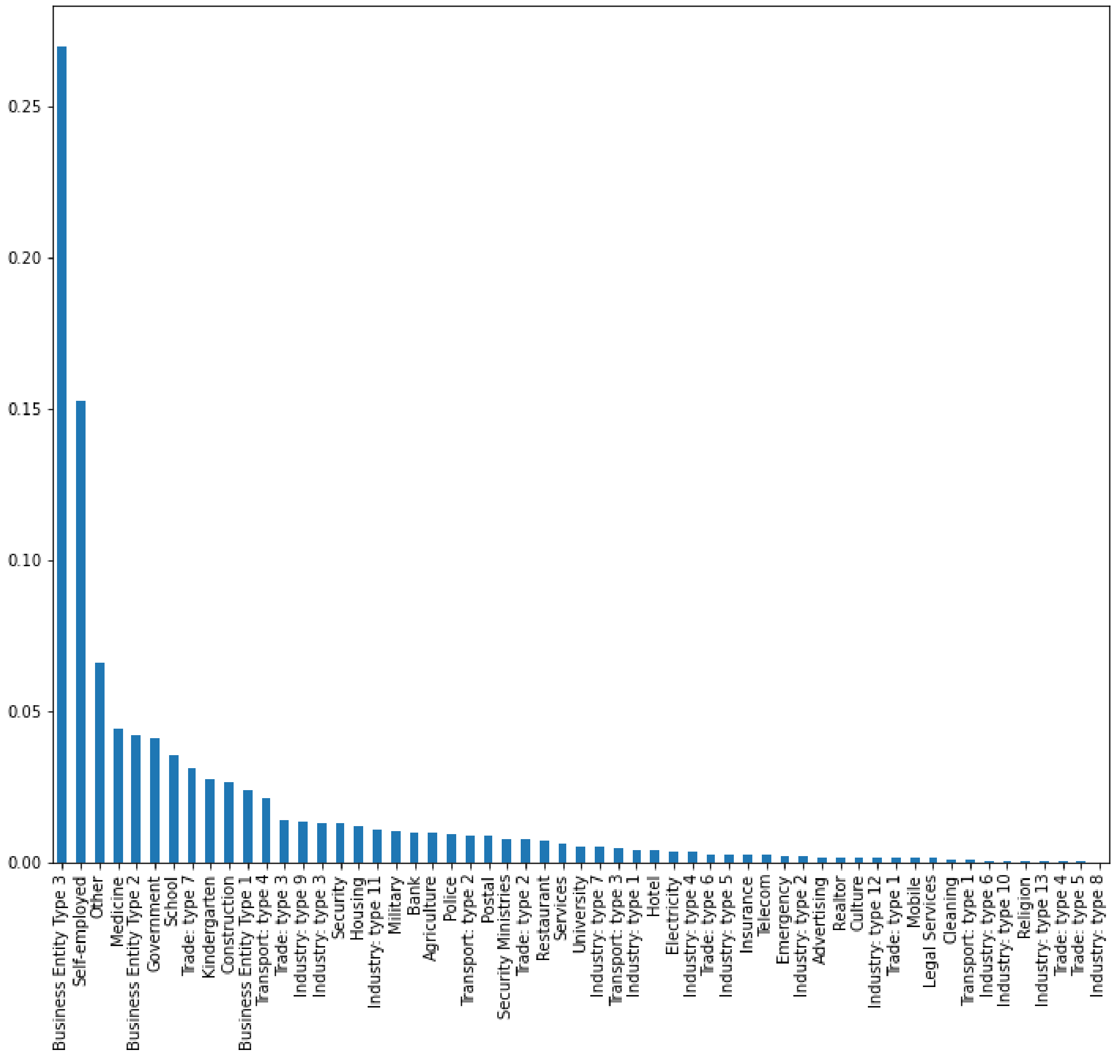
[135]:
```
df.ORGANIZATION_TYPE.value_counts(normalize=True)*100
```

[135]:
```
Business Entity Type 3    26.966002
Self-employed             15.234635
Other                      6.616667
Medicine                   4.439271
Business Entity Type 2     4.185440
Government                 4.126345
School                     3.527065
Trade: type 7              3.105864
Kindergarten               2.728686
Construction               2.665625
Business Entity Type 1     2.373322
```

```
Transport: type 4         2.140908
Trade: type 3             1.384967
Industry: type 9          1.335787
Industry: type 3          1.300092
Security                  1.287797
Housing                   1.173176
Industry: type 11         1.072437
Military                  1.044674
Bank                      0.994305
Agriculture               0.973284
Police                    0.928467
Transport: type 2         0.874131
Postal                    0.855491
Security Ministries       0.782911
Trade: type 2             0.753562
Restaurant                0.718263
Services                  0.624663
University                0.526303
Industry: type 7          0.518371
Transport: type 3         0.470778
Industry: type 1          0.412079
Hotel                     0.383127
Electricity               0.376781
Industry: type 4          0.347828
Trade: type 6             0.250262
Industry: type 5          0.237570
Insurance                 0.236777
Telecom                   0.228845
Emergency                 0.222102
Industry: type 2          0.181648
Advertising               0.170146
Realtor                   0.157058
Culture                   0.150316
Industry: type 12         0.146350
Trade: type 1             0.138021
Mobile                    0.125726
Legal Services            0.120966
Cleaning                  0.103119
Transport: type 1         0.079719
Industry: type 6          0.044420
Industry: type 10         0.043231
Religion                  0.033712
Industry: type 13         0.026573
Trade: type 4             0.025383
Trade: type 5             0.019434
Industry: type 8          0.009519
Name: ORGANIZATION_TYPE, dtype: float64
```

Bar chart for ORGANIZATION_TYPE

[136]:
```
# bar chart for ORGANIZATION_TYPE excluding others
plt.figure(figsize=(12,10))
df[~(df.ORGANIZATION_TYPE == 'Others')].ORGANIZATION_TYPE.
 ↪value_counts(normalize=True).plot.bar()
plt.show()
```



Analyzing Education Type

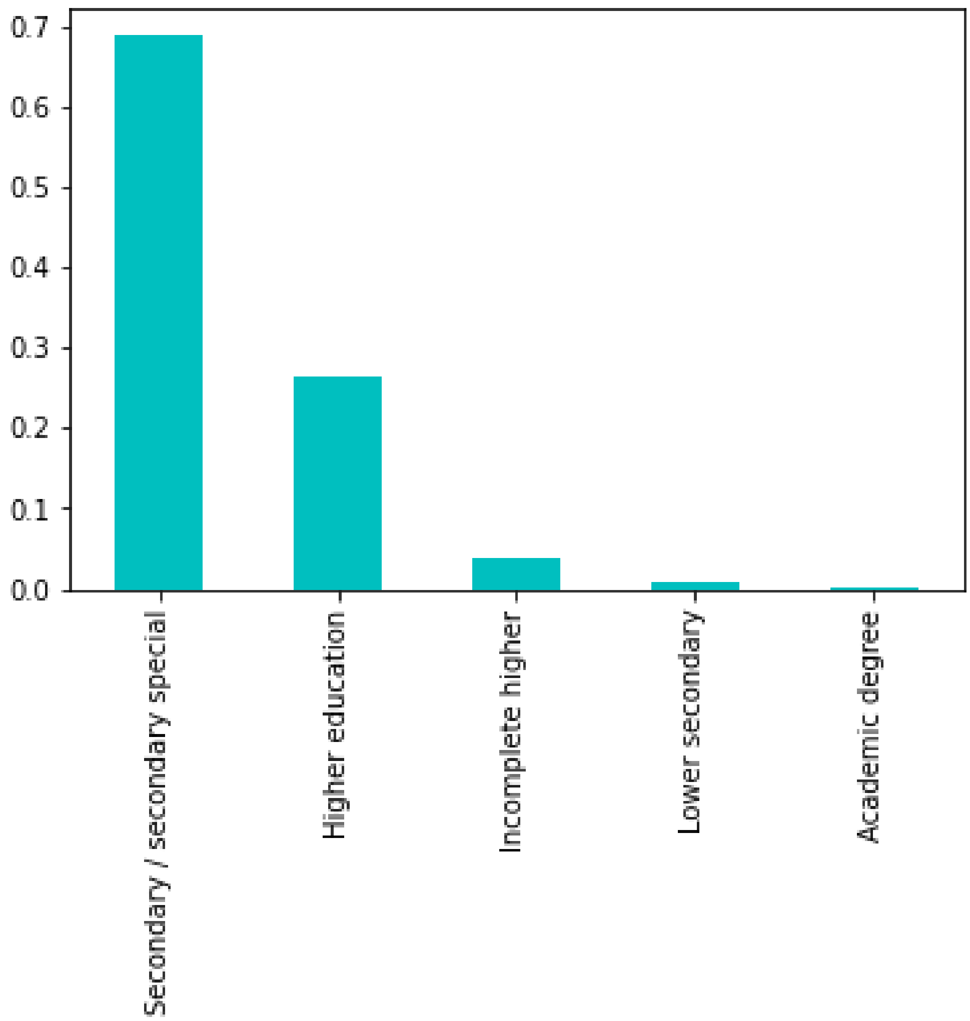
[137]:
```
df.NAME_EDUCATION_TYPE.value_counts()
```

[137]:
```
Secondary / secondary special    173285
Higher education                  66669
Incomplete higher                  9757
```

```
Lower secondary                    2287
Academic degree                     138
Name: NAME_EDUCATION_TYPE, dtype: int64
```

bar chart for education type

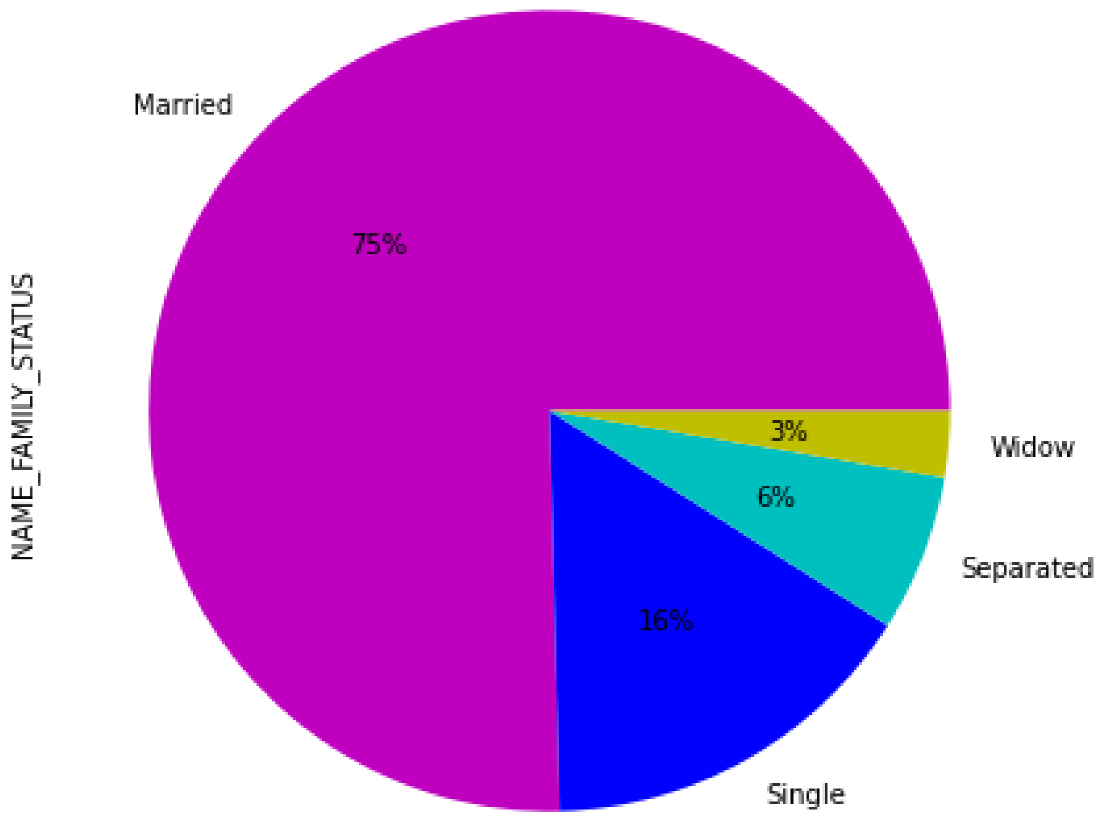
[138]:
```
df.NAME_EDUCATION_TYPE.value_counts(normalize=True).plot.bar(color="c")
plt.show()
```



Analyzing Family Status and drawing Pie Chart

[139]:
```
my_colors=['m','b','c','y']
val = df.NAME_FAMILY_STATUS.value_counts(normalize=True)
plt.figure(figsize=(7,7))
df.NAME_FAMILY_STATUS.value_counts(normalize=True).plot.pie(autopct='%1.0f%%',␣
↪colors = my_colors)
```

```
plt.show()
```



1 Previous Application Status

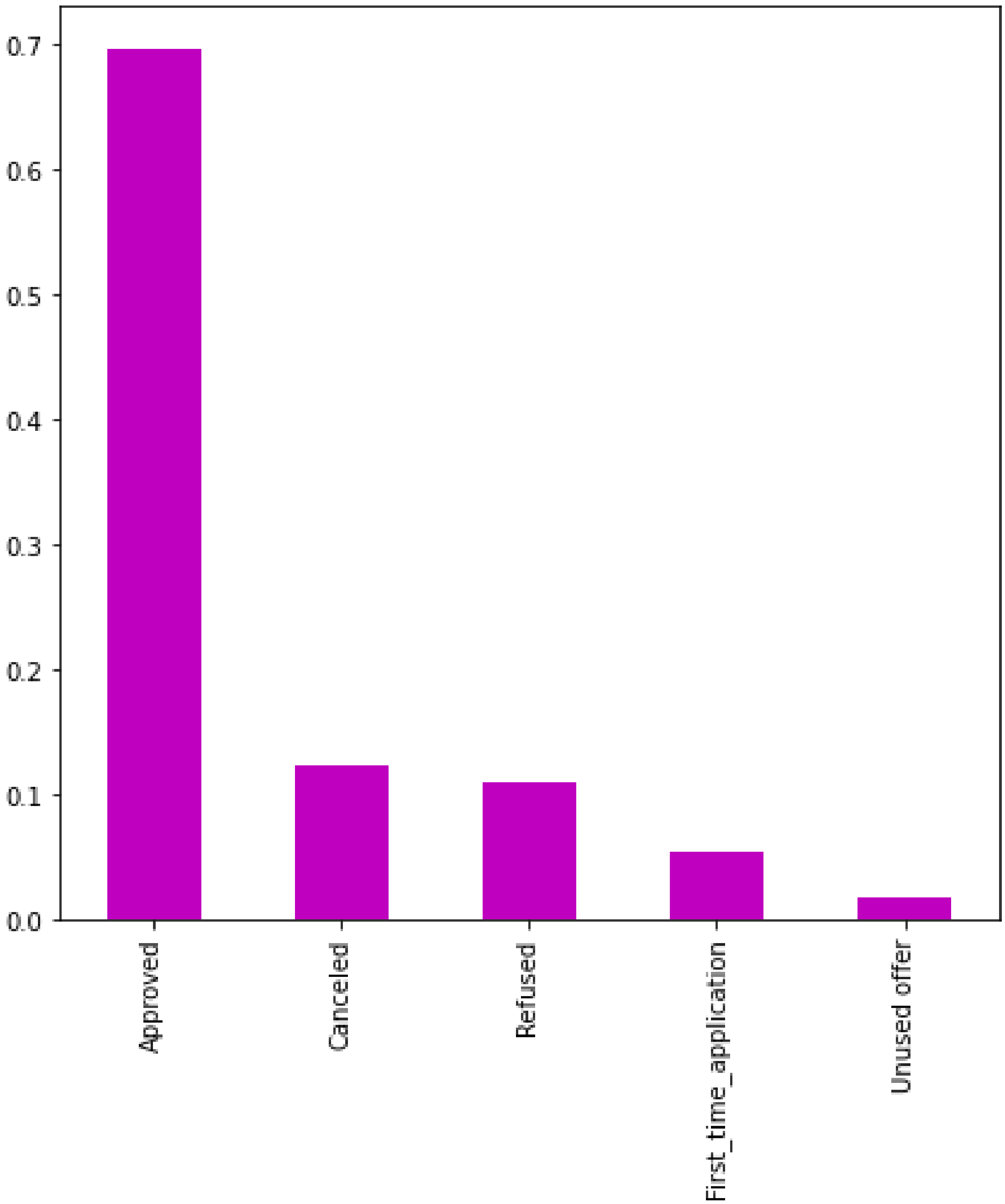
[140]:
```
df.PREV_CONTRACT_STATUS.value_counts(normalize=True)*100
```

[140]:
```
Approved                  69.711981
Canceled                  12.233081
Refused                   10.971063
First_time_application     5.421281
Unused offer               1.662595
Name: PREV_CONTRACT_STATUS, dtype: float64
```

prev application status bar graph

[141]:

```
plt.figure(figsize=(7,7))
df.PREV_CONTRACT_STATUS.value_counts(normalize=True).plot.bar(color='m')
plt.show()
```



Analyzing Owns Property

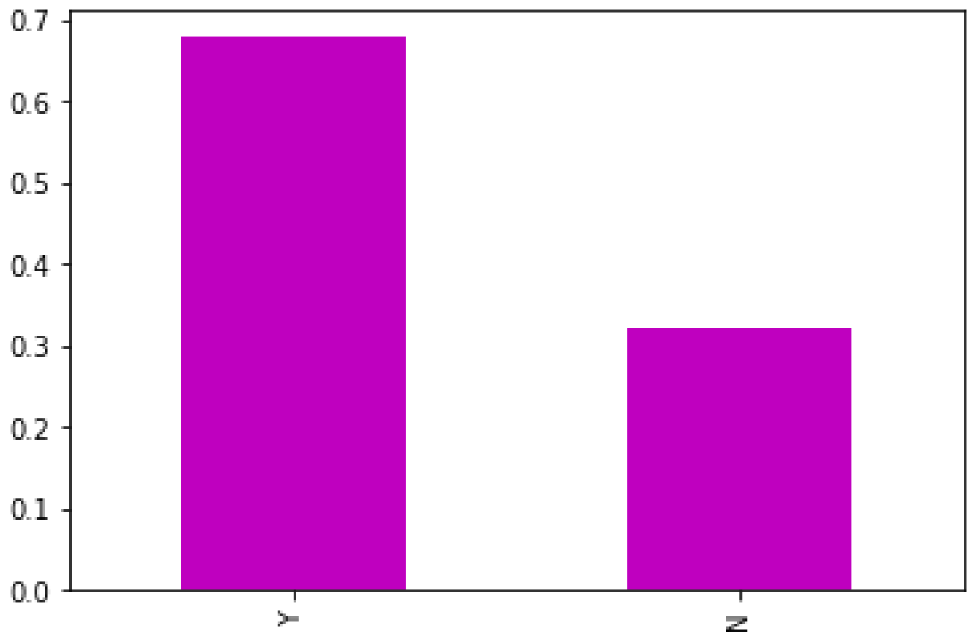
[142]:

```
df.FLAG_OWN_REALTY.value_counts(normalize=True)*100
```

[142]: Y   67.852667
N   32.147333
Name: FLAG_OWN_REALTY, dtype: float64

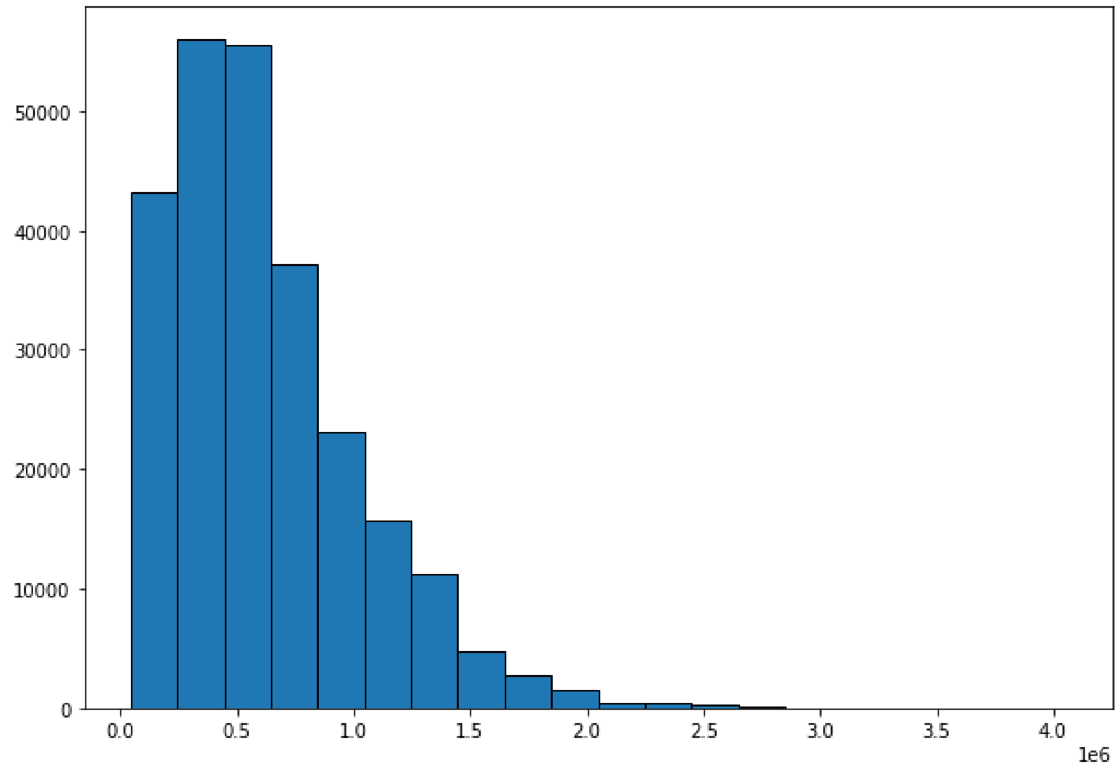
plot bar chart for FLAG_OWN_REALTY

[143]:
```
df.FLAG_OWN_REALTY.value_counts(normalize=True).plot.bar(color='m')
plt.show()
```



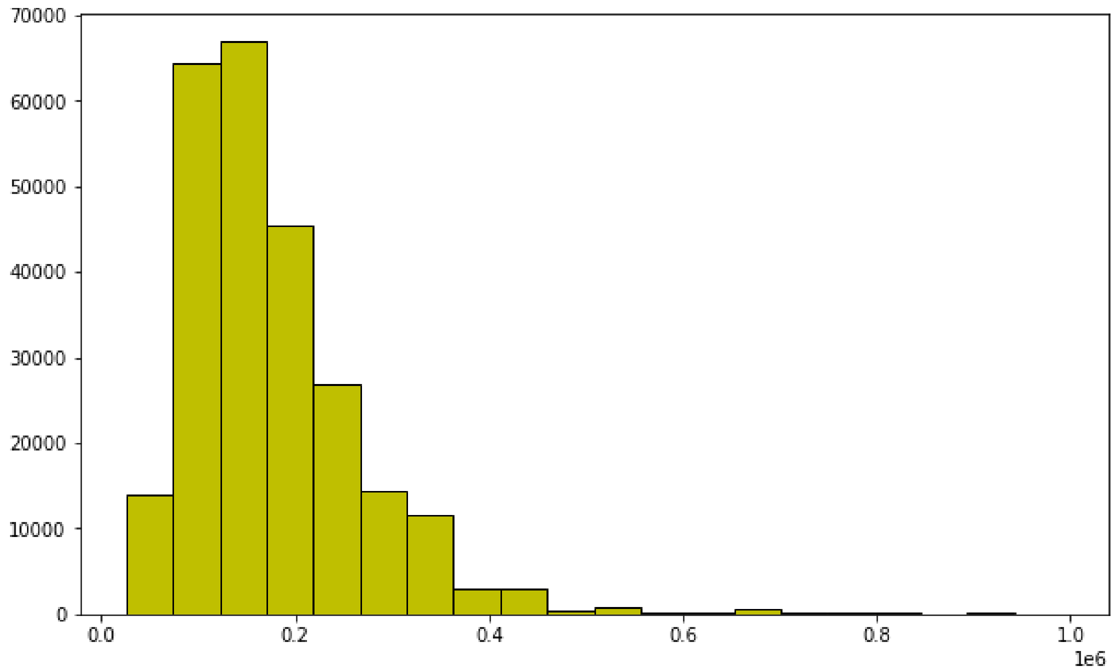
Analyzing AMT_CREDIT

[144]:
```
plt.figure(figsize=(10,7))
plt.hist(df.AMT_CREDIT, bins=20, edgecolor='black')
plt.show()
```



Analyzing AMT_INCOME_TOTAL

[145]:
```
plt.figure(figsize=(10,6))
plt.hist(df[df.AMT_INCOME_TOTAL < 10**6].AMT_INCOME_TOTAL, bins=20,␣
↪edgecolor='black',color='y')
plt.show()
```



1.1 Organization type vs Total Income

[146]:
```
df.groupby('ORGANIZATION_TYPE').AMT_INCOME_TOTAL.aggregate(['mean', 'median'])
```

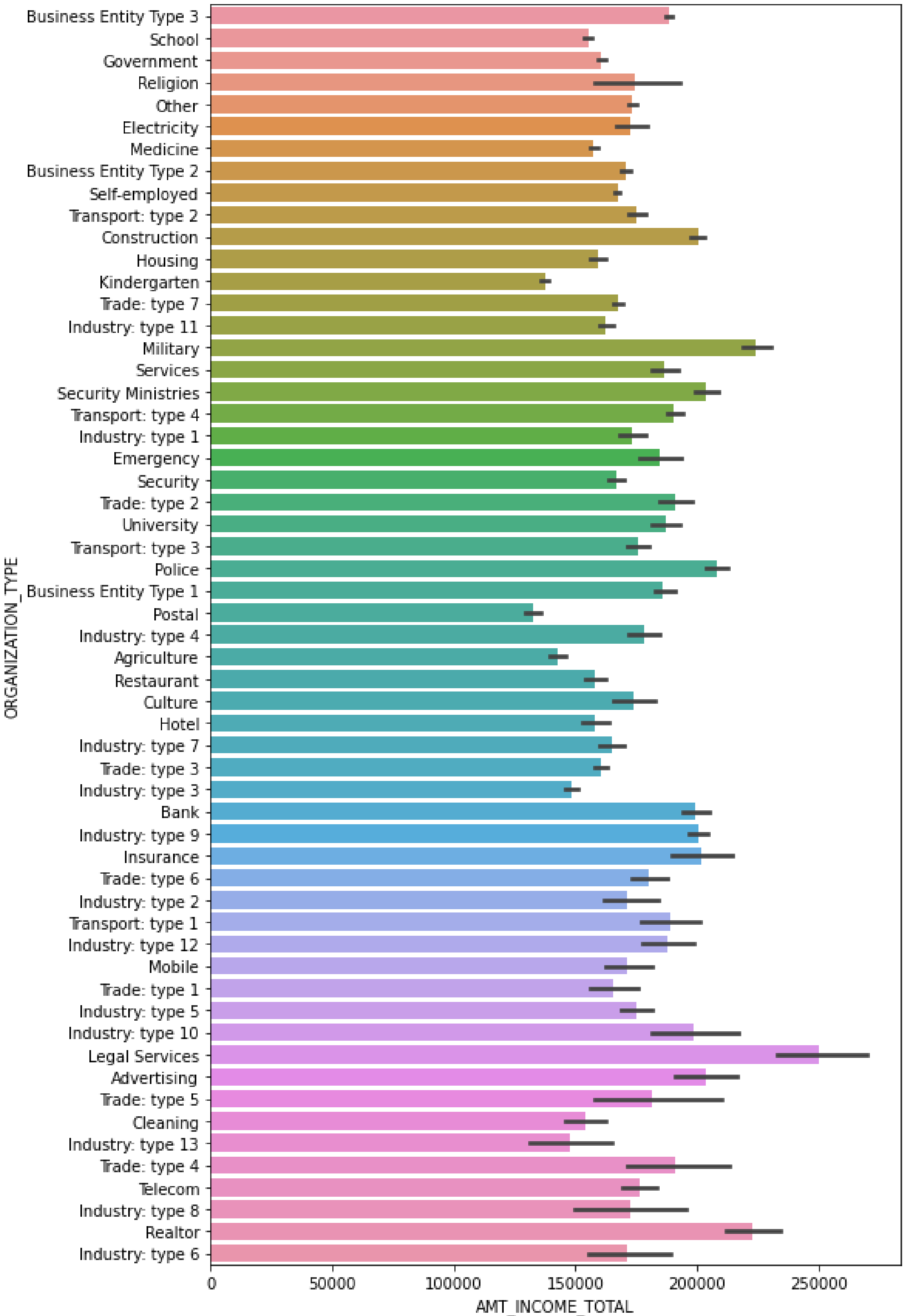
[146]:
```
                                 mean    median
ORGANIZATION_TYPE
Advertising             203919.230769  165600.0
Agriculture             143024.492054  126000.0
Bank                    199308.171719  157500.0
Business Entity Type 1  186195.135862  157500.0
Business Entity Type 2  170854.997664  157500.0
Business Entity Type 3  188339.639157  157500.0
Cleaning                154220.192308  135000.0
Construction            200227.861553  180000.0
Culture                 174009.300792  157500.0
Electricity             172648.781053  157500.0
Emergency               184712.938393  162000.0
Government              160790.739935  135000.0
Hotel                   158304.484472  135000.0
Housing                 159420.249493  135000.0
Industry: type 1        173310.119827  157500.0
Industry: type 10       198454.128440  180000.0
Industry: type 11       162680.525148  137227.5
Industry: type 12       187659.560976  157500.0
```

```
Industry: type 13       147915.671642  135000.0
Industry: type 2        171449.253275  157500.0
Industry: type 3        148759.136714  135000.0
Industry: type 4        178382.216648  157500.0
Industry: type 5        174979.224541  157500.0
Industry: type 6        171212.946429  139050.0
Industry: type 7        164829.622035  144000.0
Industry: type 8        172537.500000  162450.0
Industry: type 9        200557.401574  180000.0
Insurance               201483.934673  157500.0
Kindergarten            137594.210756  126000.0
Legal Services          249875.409836  225000.0
Medicine                157507.972663  135000.0
Military                224256.462010  202500.0
Mobile                  171468.525237  157500.0
Other                   173527.222712  157500.0
Police                  208047.569897  189000.0
Postal                  132651.582267  112500.0
Realtor                 222954.545455  202500.0
Religion                174705.882353  162000.0
Restaurant              158203.898399  135000.0
School                  155171.148898  135000.0
Security                166784.133728  153000.0
Security Ministries     203772.592705  180000.0
Self-employed           167442.393791  148500.0
Services                186420.062857  157500.0
Telecom                 176405.459272  157500.0
Trade: type 1           165380.360690  135000.0
Trade: type 2           190712.744526  157500.0
Trade: type 3           160481.651933  135000.0
Trade: type 4           191141.015625  176625.0
Trade: type 5           181694.387755  157500.0
Trade: type 6           179940.266244  157500.0
Trade: type 7           167565.157068  144000.0
Transport: type 1       189268.656716  157500.0
Transport: type 2       175439.750696  157500.0
Transport: type 3       175571.651222  157500.0
Transport: type 4       190658.856058  180000.0
University              187161.131123  162000.0
```

[147]:
```
plt.figure(figsize=(8,15))
sns.barplot(df.AMT_INCOME_TOTAL, df.ORGANIZATION_TYPE)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
```

```
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



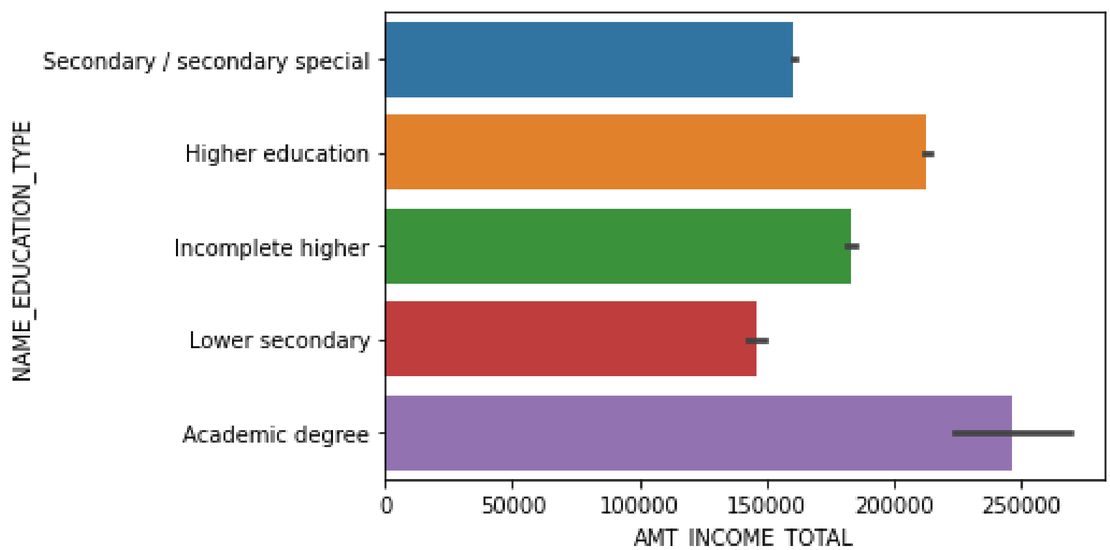
Analyzing Education level vs Income

[148]:
```
df.groupby('NAME_EDUCATION_TYPE').AMT_INCOME_TOTAL.aggregate(['mean', 'median'])
```

[148]:
```
                                          mean    median
NAME_EDUCATION_TYPE
Academic degree                  246808.695652  225000.0
Higher education                 213222.190573  180000.0
Incomplete higher                183275.966332  157500.0
Lower secondary                  145864.995190  135000.0
Secondary / secondary special    160811.722869  135000.0
```

[149]:
```
# Education level of the applicant vs Income
sns.barplot(df.AMT_INCOME_TOTAL, df.NAME_EDUCATION_TYPE)
plt.show()
```

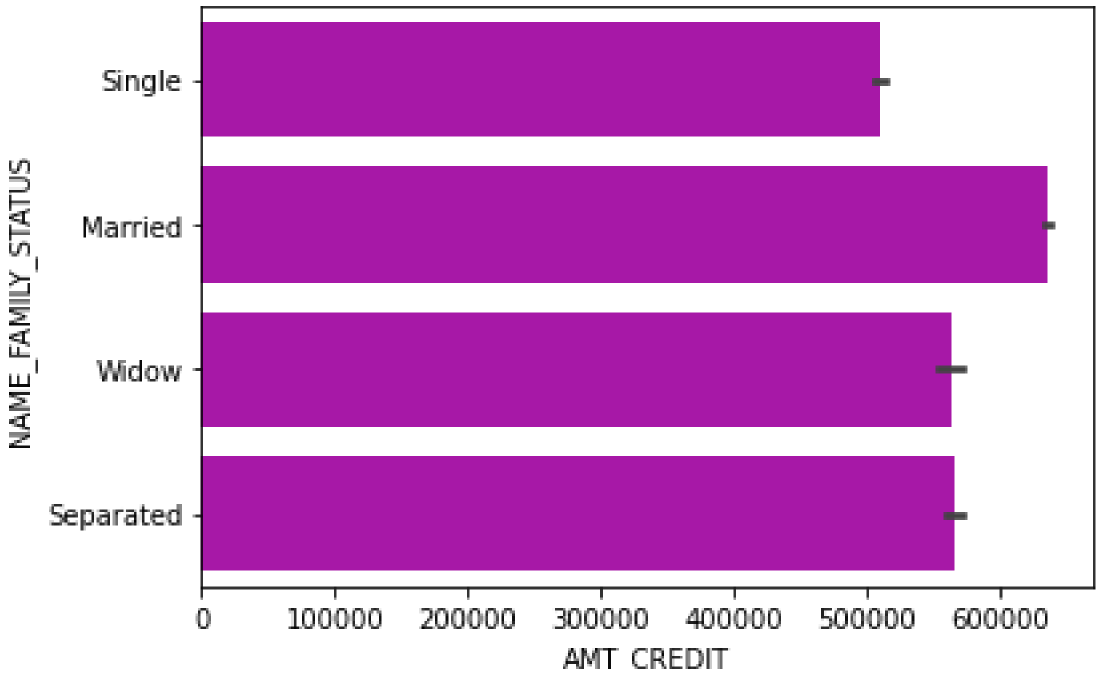
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



Marital status vs Amount requested for loan

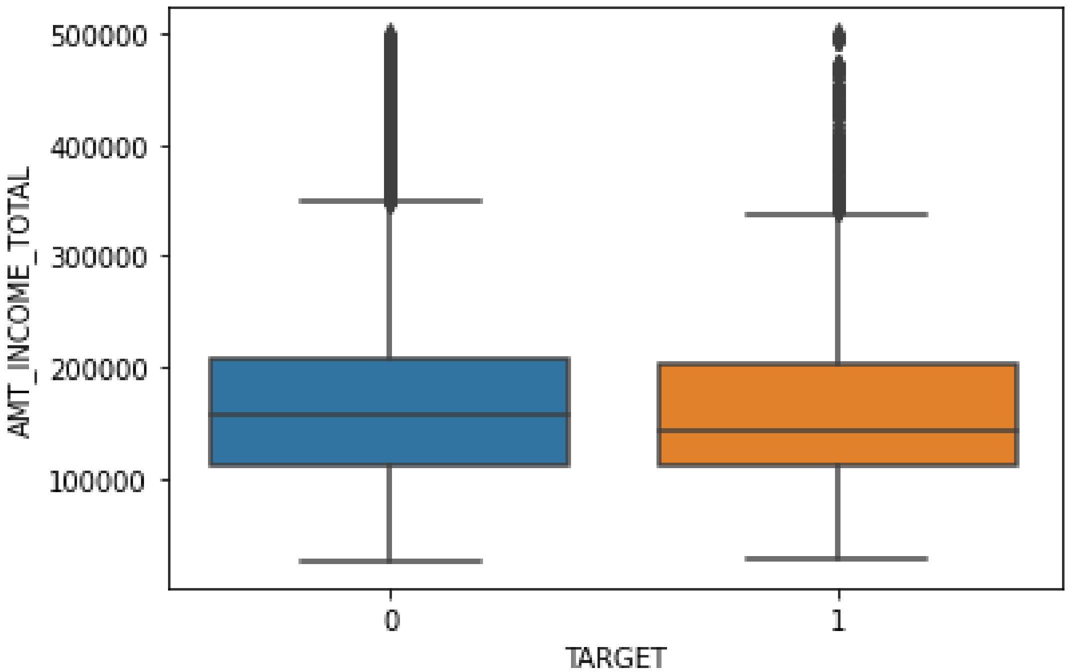
[150]:
```
# Marital Status of the applicant vs Amount requested for Loan
sns.barplot(df.AMT_CREDIT, df.NAME_FAMILY_STATUS, color = 'm')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



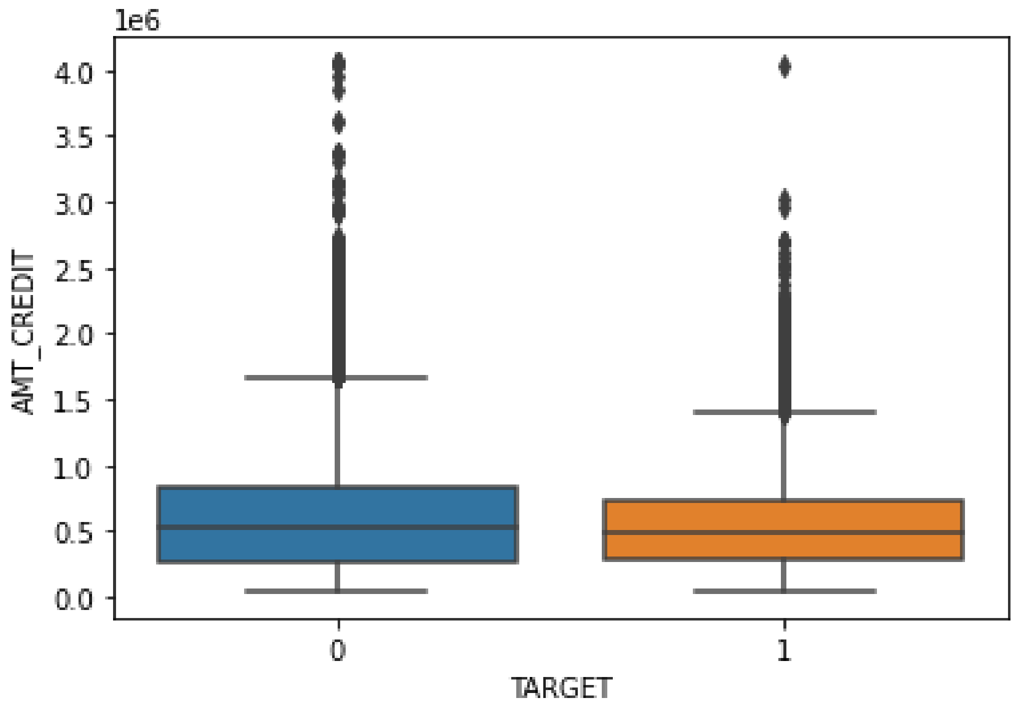
1.2 Income Amount vs target

[151]:
```
sns.boxplot(x=df.TARGET, y=df[df.AMT_INCOME_TOTAL < 0.5*10**6].AMT_INCOME_TOTAL)
plt.show()
```



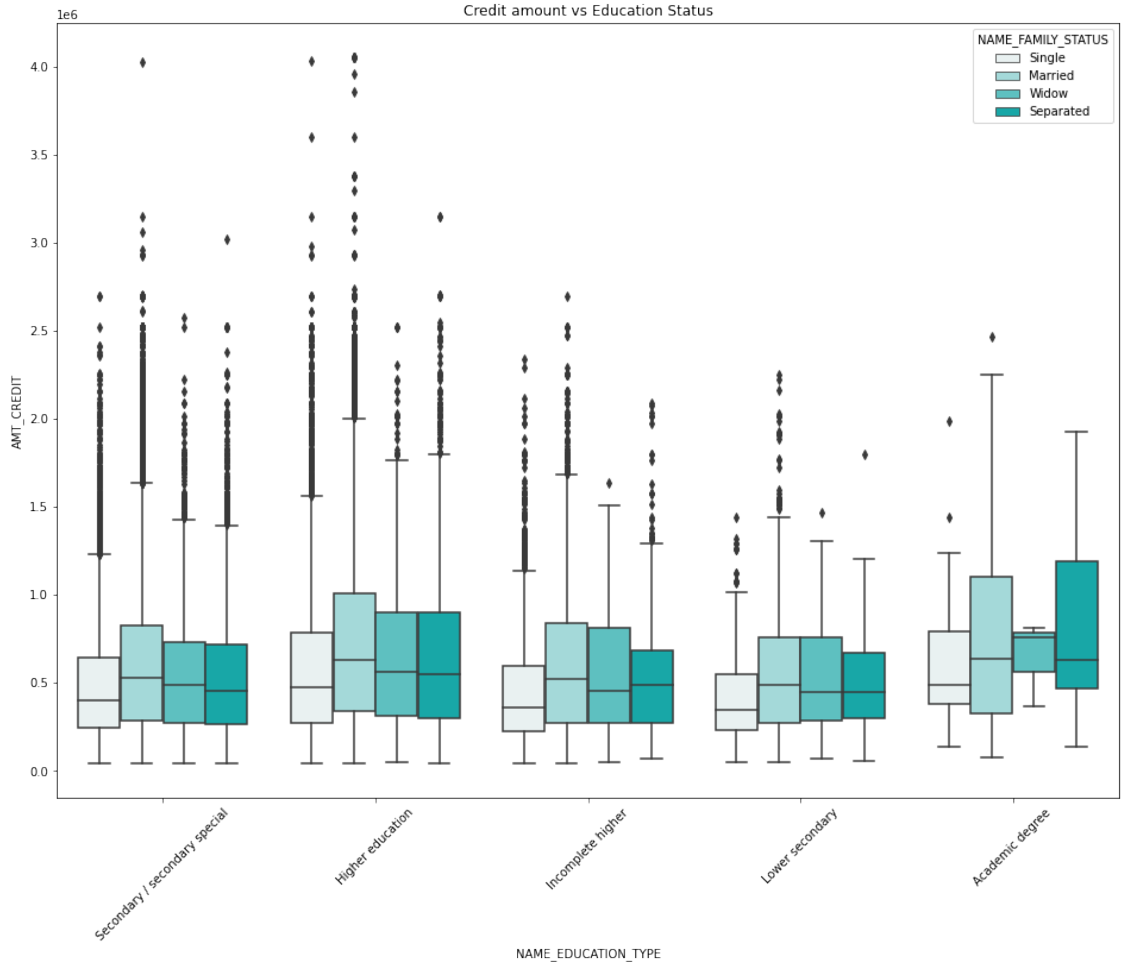
1.3 Amount of loan vs target

[152]:
```
sns.boxplot(x=df.TARGET, y=df.AMT_CREDIT)
plt.show()
```



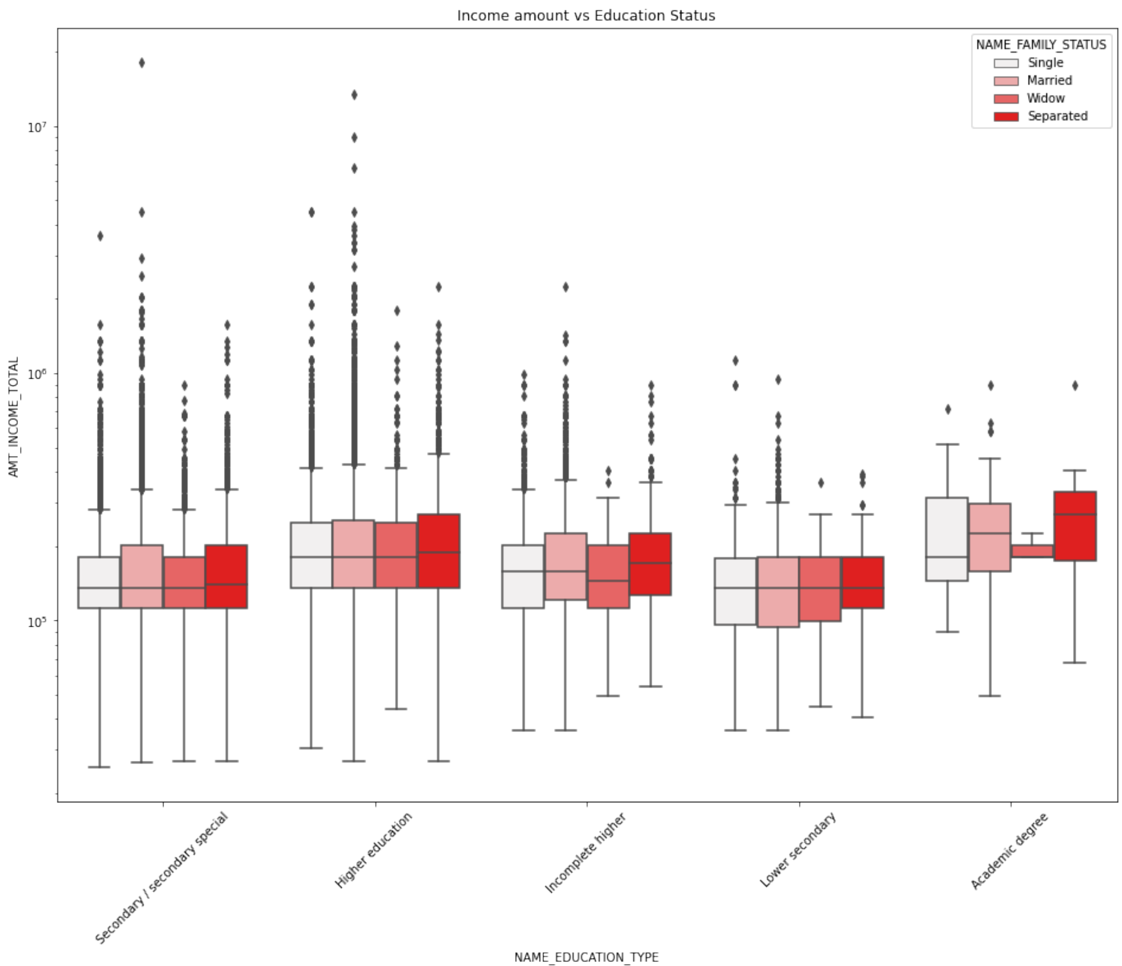
Credit amount vs Education Status

[206]:
```
plt.figure(figsize=(16,12))
plt.xticks(rotation=45)
sns.boxplot(data =df, x='NAME_EDUCATION_TYPE',y='AMT_CREDIT', hue␣
 ↪='NAME_FAMILY_STATUS',orient='v',color = 'c')
plt.title('Credit amount vs Education Status')
plt.show()
```



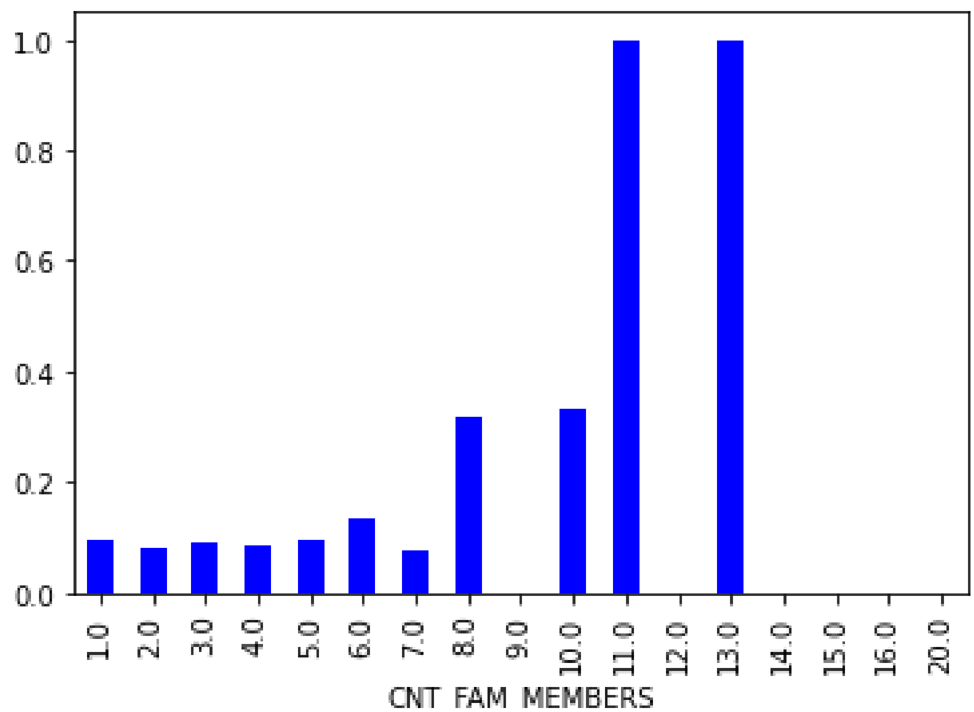
Income Amount vs Education Status

[211]:
```
plt.figure(figsize=(16,12))
plt.xticks(rotation=45)
plt.yscale('log')
sns.boxplot(data =df, x='NAME_EDUCATION_TYPE',y='AMT_INCOME_TOTAL', hue␣
↪='NAME_FAMILY_STATUS',orient='v', color='r')
plt.title('Income amount vs Education Status')
plt.show()
```



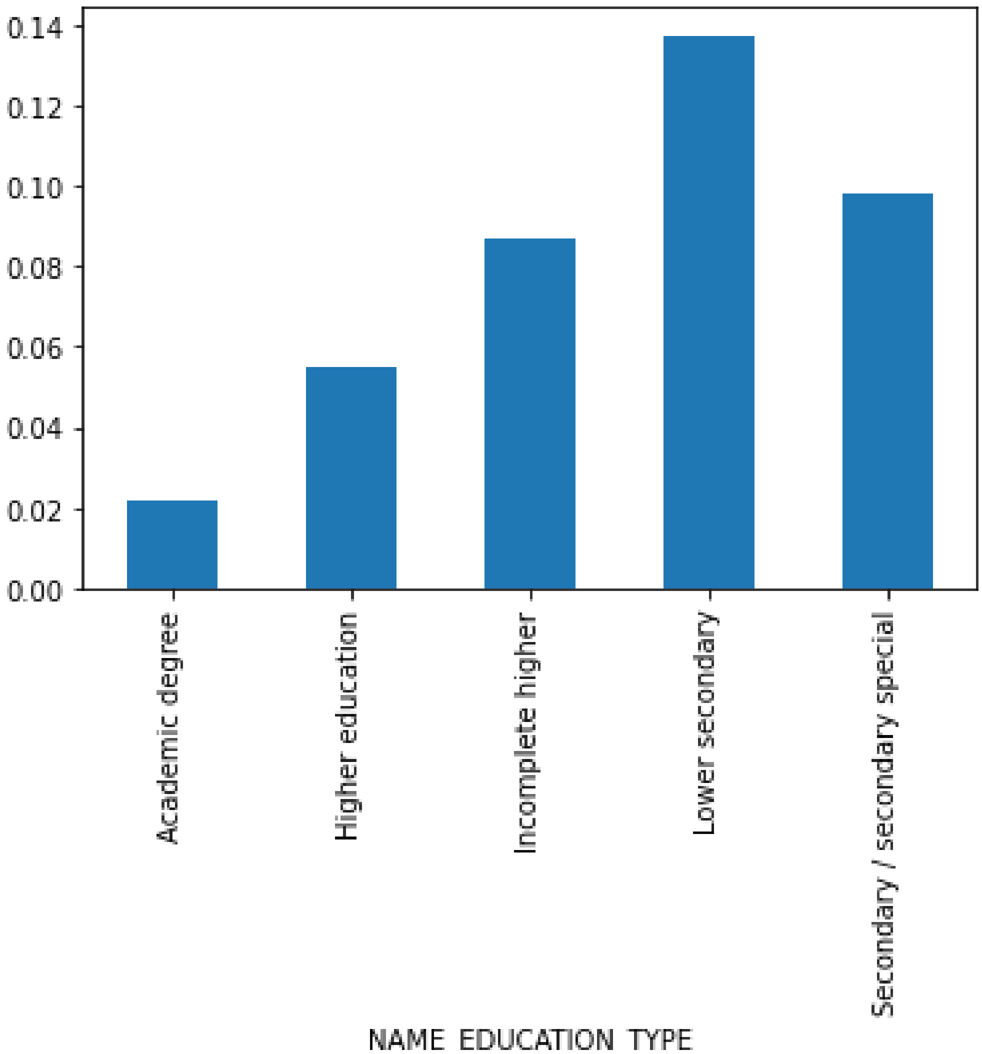
1.4 Family member count vs target

[153]:
```
df.groupby('CNT_FAM_MEMBERS').TARGET.mean().plot.bar(color='b')
plt.show()
```



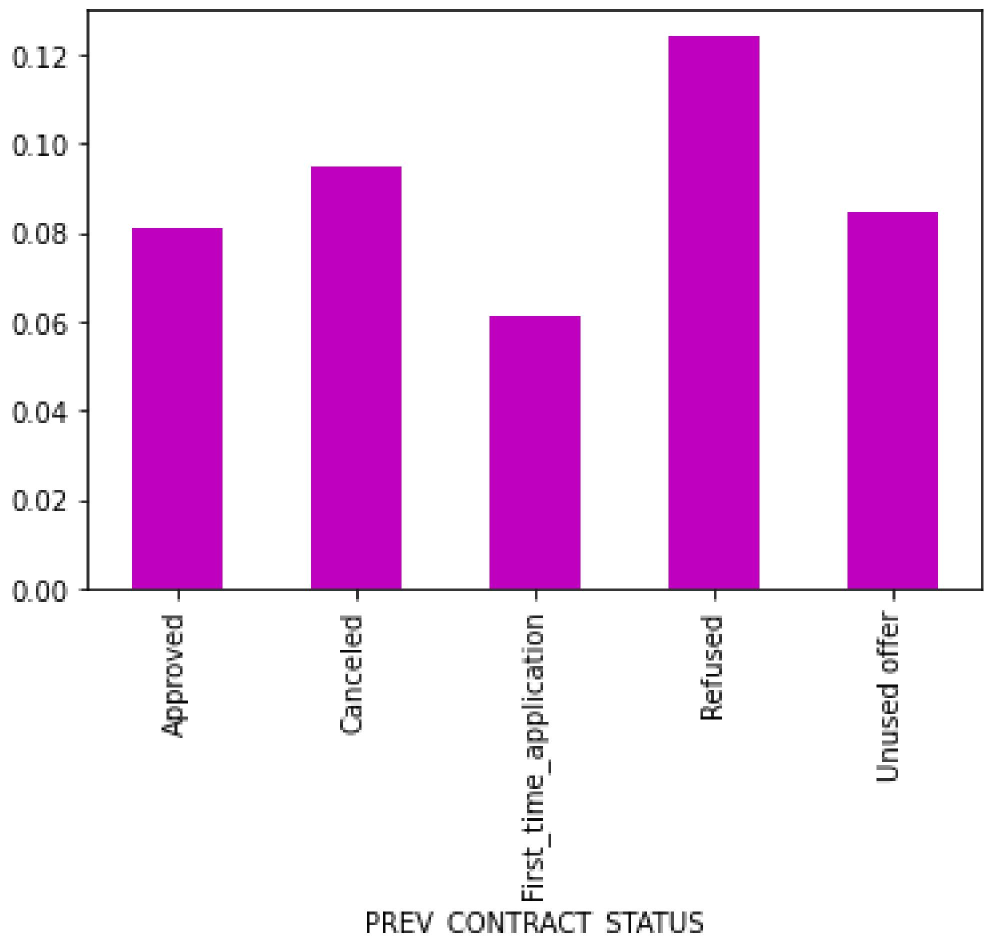
NAME_EDUCATION_TYPE vs target

[154]:
```
df.groupby('NAME_EDUCATION_TYPE').TARGET.mean().plot.bar()
plt.show()
```



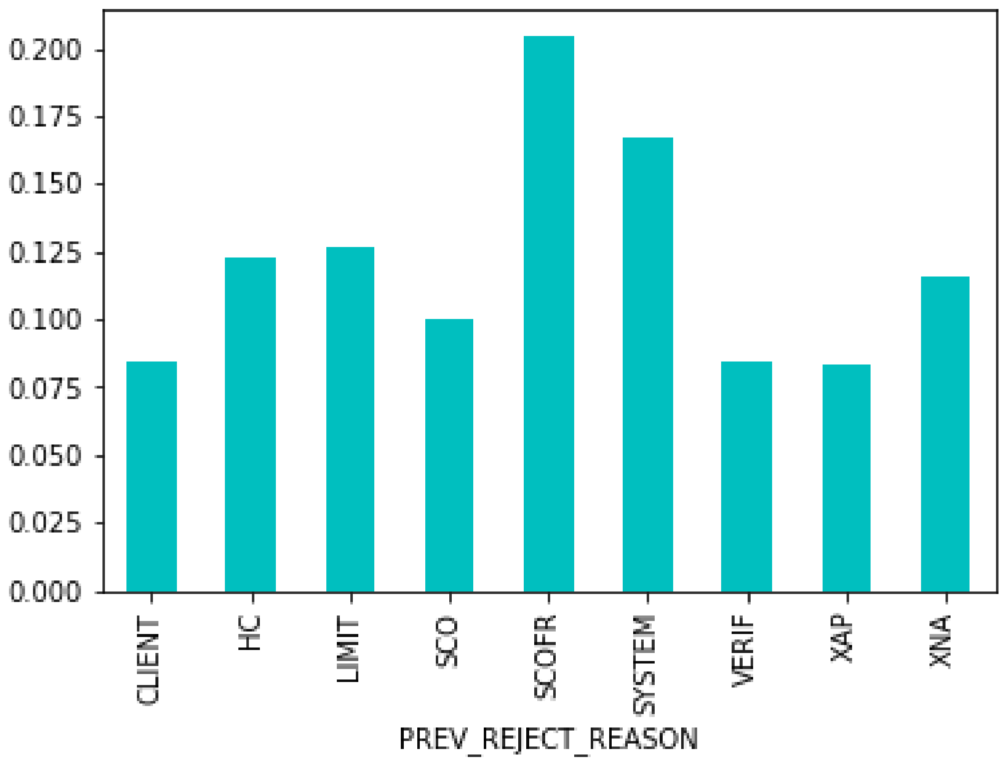
1.5 PREV_CONTRACT_STATUS vs target

[155]:
```
df.groupby('PREV_CONTRACT_STATUS').TARGET.mean().plot.bar(color='m')
plt.show()
```



1.6 PREV_REJECT_REASON vs target

[156]:
```
df.groupby('PREV_REJECT_REASON').TARGET.mean().plot.bar(color='c')
plt.show()
```

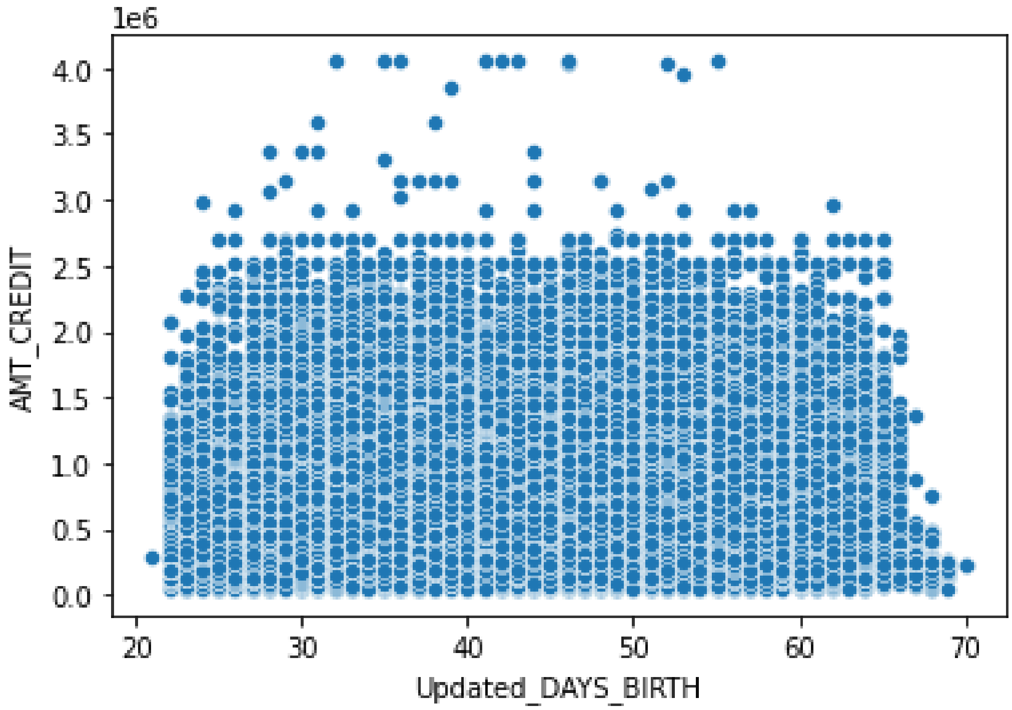


2 Numeric Analysis

Scatterplot for Loan amount vs Age

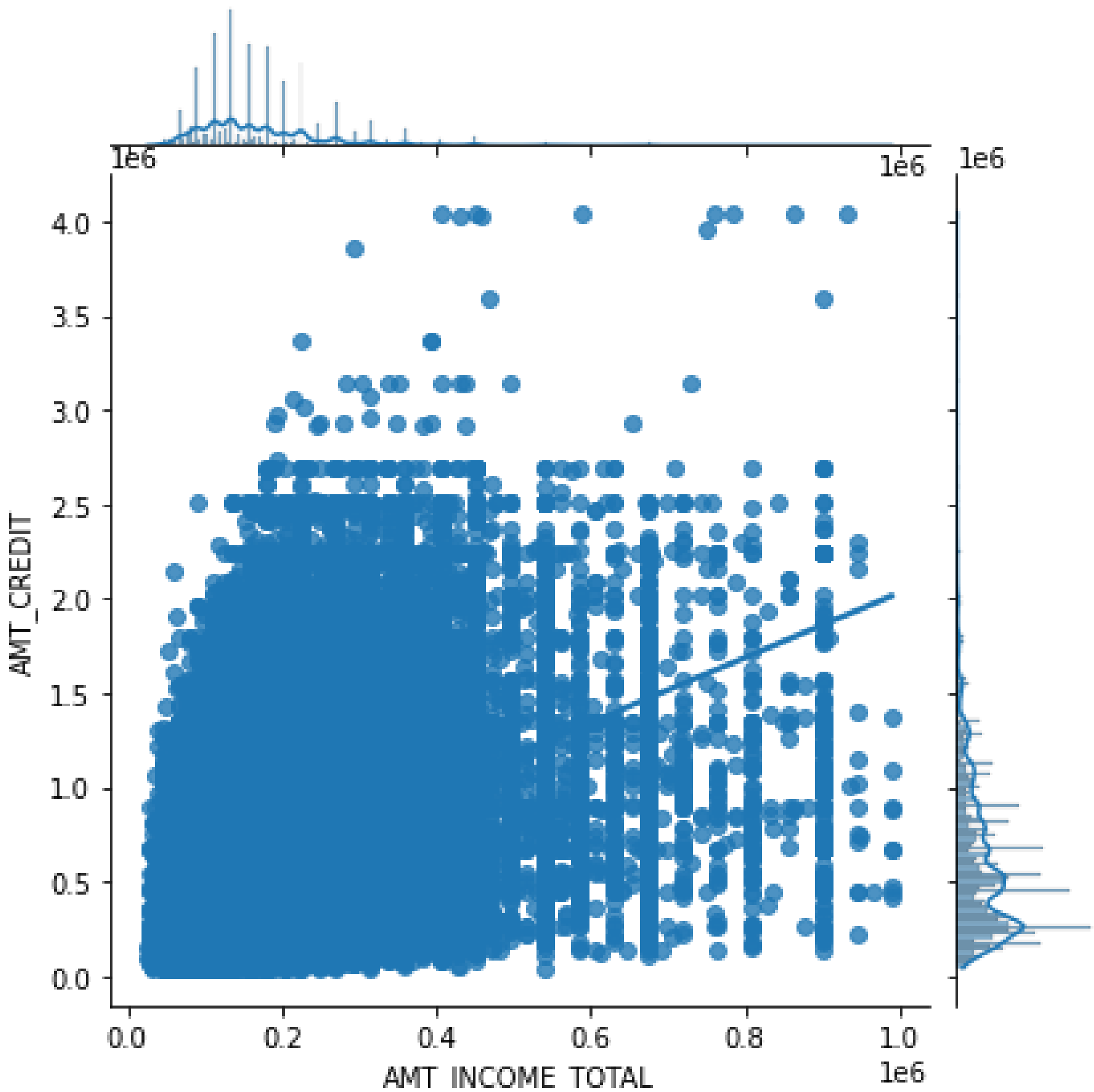
[176]:
```
#Age vs Requested Loan Amount
sns.scatterplot(df.Updated_DAYS_BIRTH, df.AMT_CREDIT)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



Jointplot of "Total Income" vs "Amount requested for the loan"

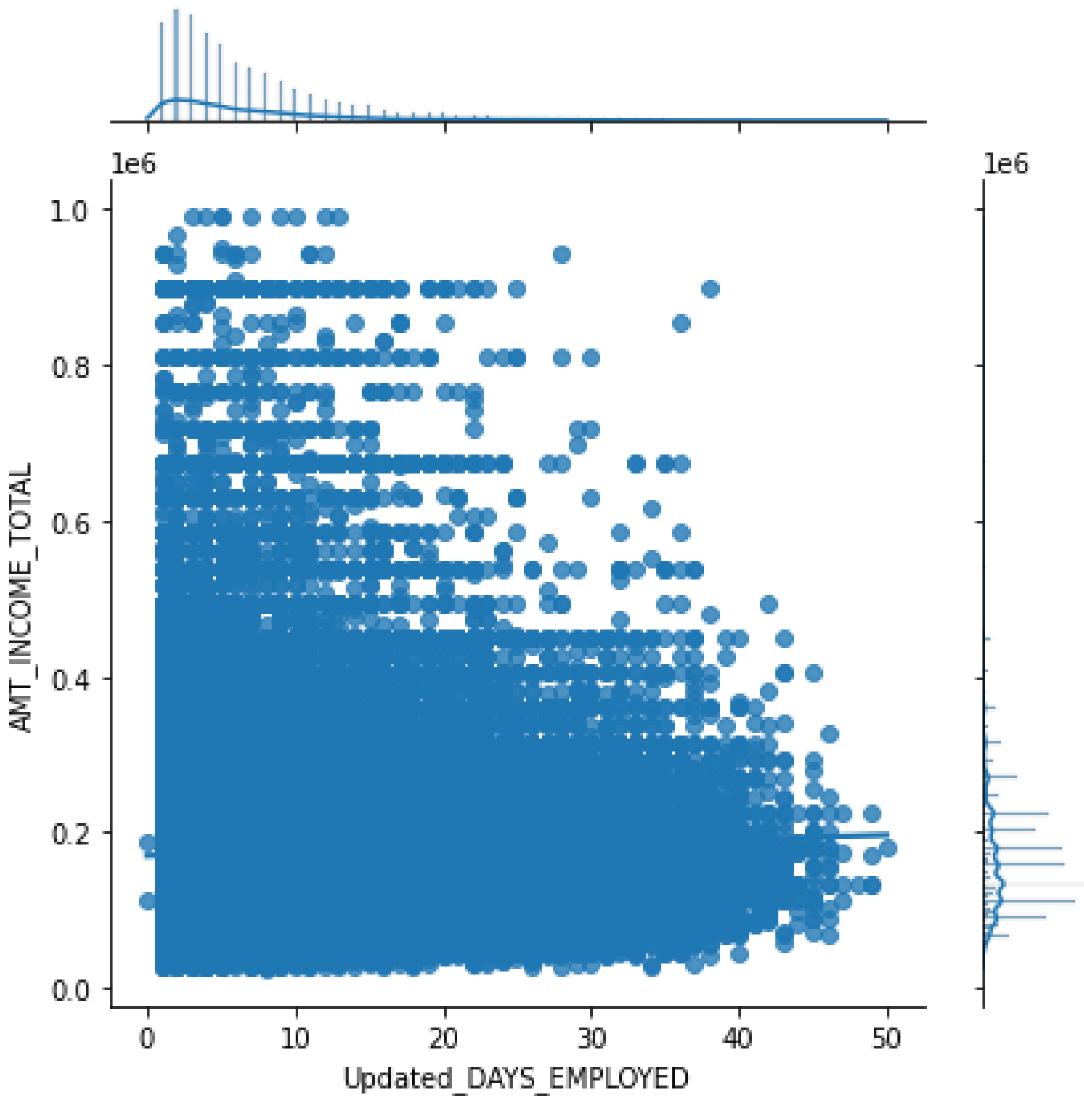
[170]:
```
sns.jointplot(data = df[df.AMT_INCOME_TOTAL < 10**6], x = 'AMT_INCOME_TOTAL', y␣
↪= 'AMT_CREDIT', kind='reg')
plt.show()
```



Jointplot of "Total Income" vs "Experience in Years"

[174]:
```
sns.jointplot(data = df[df.AMT_INCOME_TOTAL < 10**6], x =␣
↪'Updated_DAYS_EMPLOYED', y = 'AMT_INCOME_TOTAL', space=0.5, kind='reg')

plt.show()
```



2.1 MULTIVARIATE ANALYSIS

NAME_FAMILY_STATUS vs NAME_EDUCATION_TYPE vs TARGET

[184]:
```
chart1 = pd.pivot_table(data=df, index='NAME_FAMILY_STATUS',␣
↪columns='NAME_EDUCATION_TYPE', values='TARGET')
chart1
```

[184]:
```
NAME_EDUCATION_TYPE  Academic degree  Higher education  Incomplete higher  \
NAME_FAMILY_STATUS
Married                     0.021739          0.052766           0.083670
Separated                   0.000000          0.058437           0.083658
```

```
Single                      0.034483         0.064023           0.095870
Widow                       0.000000         0.048536           0.067416

NAME_EDUCATION_TYPE  Lower secondary  Secondary / secondary special
NAME_FAMILY_STATUS
Married                     0.134524                      0.094710
Separated                   0.171429                      0.097737
Single                      0.149746                      0.124455
Widow                       0.082192                      0.061892
```

[187]:

```
plt.figure(figsize=(10,8))
sns.heatmap(chart1, annot=True, cmap='YlOrRd', center=0.081)
plt.show()
```

NAME_EDUCATION_TYPE vs ORGANIZATION_TYPE vs TARGET

[191]:
```
chart2 = pd.pivot_table(data=df, index='ORGANIZATION_TYPE',␣
↪columns='NAME_EDUCATION_TYPE', values='TARGET')
chart2
```

[191]:
```
NAME_EDUCATION_TYPE     Academic degree  Higher education  Incomplete higher  \
ORGANIZATION_TYPE
Advertising                     0.00000          0.057416           0.086957
Agriculture                     0.00000          0.069444           0.025000
```

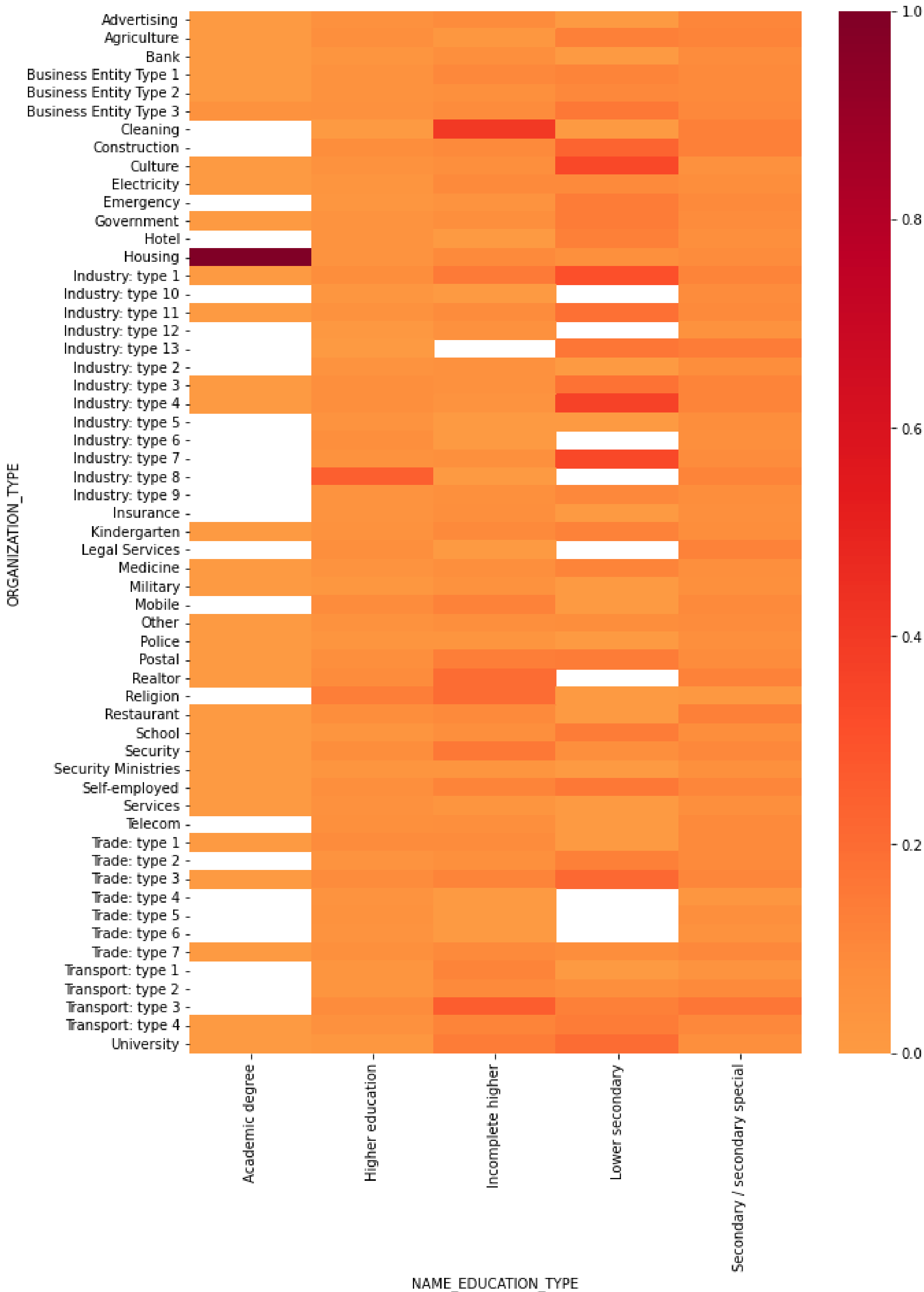
Bank	0.00000	0.038484	0.073684
Business Entity Type 1	0.00000	0.047470	0.099099
Business Entity Type 2	0.00000	0.048110	0.063584
Business Entity Type 3	0.04878	0.062423	0.086114
Cleaning	NaN	0.000000	0.400000
Construction	NaN	0.078125	0.092437
Culture	0.00000	0.047059	0.071429
Electricity	0.00000	0.034602	0.093750
Emergency	NaN	0.030488	0.040000
Government	0.00000	0.041784	0.070946
Hotel	NaN	0.040404	0.000000
Housing	1.00000	0.043831	0.090909
Industry: type 1	0.00000	0.077720	0.150000
Industry: type 10	NaN	0.028571	0.000000
Industry: type 11	0.00000	0.049505	0.089744
Industry: type 12	NaN	0.016667	0.058824
Industry: type 13	NaN	0.000000	NaN
Industry: type 2	NaN	0.045455	0.058824
Industry: type 3	0.00000	0.067100	0.059524
Industry: type 4	0.00000	0.073892	0.046512
Industry: type 5	NaN	0.043478	0.000000
Industry: type 6	NaN	0.066667	0.000000
Industry: type 7	NaN	0.050909	0.064516
Industry: type 8	NaN	0.250000	0.000000
Industry: type 9	NaN	0.039871	0.064286
Insurance	NaN	0.046729	0.071429
Kindergarten	0.00000	0.052384	0.097166
Legal Services	NaN	0.073930	0.000000
Medicine	0.00000	0.043118	0.073913
Military	0.00000	0.027203	0.052174
Mobile	NaN	0.082759	0.119048
Other	0.00000	0.051862	0.066440
Police	0.00000	0.037447	0.038462
Postal	0.00000	0.068259	0.140351
Realtor	0.00000	0.084158	0.200000
Religion	NaN	0.136364	0.200000
Restaurant	0.00000	0.074906	0.091954
School	0.00000	0.036695	0.072650
Security	0.00000	0.079491	0.157303
Security Ministries	0.00000	0.038462	0.037975
Self-employed	0.00000	0.069564	0.113869
Services	0.00000	0.059432	0.035294
Telecom	NaN	0.062500	0.071429
Trade: type 1	0.00000	0.084337	0.083333
Trade: type 2	NaN	0.045226	0.064639
Trade: type 3	0.00000	0.084011	0.115385
Trade: type 4	NaN	0.041667	0.000000

Trade: type 5	NaN	0.052632	0.000000
Trade: type 6	NaN	0.044444	0.000000
Trade: type 7	0.00000	0.063580	0.089928
Transport: type 1	NaN	0.035088	0.111111
Transport: type 2	NaN	0.035225	0.090909
Transport: type 3	NaN	0.086957	0.256410
Transport: type 4	0.00000	0.056572	0.113043
University	0.00000	0.025424	0.142857

NAME_EDUCATION_TYPE	Lower secondary	Secondary / secondary special
ORGANIZATION_TYPE		
Advertising	0.000000	0.107692
Agriculture	0.132353	0.110355
Bank	0.000000	0.082391
Business Entity Type 1	0.114754	0.093330
Business Entity Type 2	0.100000	0.097194
Business Entity Type 3	0.157980	0.105104
Cleaning	0.000000	0.132353
Construction	0.232877	0.129794
Culture	0.333333	0.057592
Electricity	0.090909	0.079545
Emergency	0.142857	0.090659
Government	0.142857	0.084339
Hotel	0.125000	0.072109
Housing	0.064516	0.088681
Industry: type 1	0.307692	0.114532
Industry: type 10	NaN	0.089552
Industry: type 11	0.187500	0.093870
Industry: type 12	NaN	0.047414
Industry: type 13	0.166667	0.145455
Industry: type 2	0.000000	0.080229
Industry: type 3	0.177778	0.113179
Industry: type 4	0.363636	0.109855
Industry: type 5	0.000000	0.077088
Industry: type 6	NaN	0.074074
Industry: type 7	0.333333	0.088176
Industry: type 8	NaN	0.111111
Industry: type 9	0.100000	0.077632
Insurance	0.000000	0.069388
Kindergarten	0.117647	0.074764
Legal Services	NaN	0.119048
Medicine	0.115789	0.071187
Military	0.000000	0.065286
Mobile	0.000000	0.095238
Other	0.080925	0.088251
Police	0.000000	0.066733
Postal	0.142857	0.084821

```
Realtor                         NaN                 0.117284
Religion                   0.000000                 0.017544
Restaurant                 0.000000                 0.127867
School                     0.142857                 0.079002
Security                   0.066667                 0.103324
Security Ministries        0.000000                 0.062871
Self-employed              0.157767                 0.107982
Services                   0.000000                 0.071233
Telecom                    0.000000                 0.089965
Trade: type 1              0.000000                 0.092050
Trade: type 2              0.125000                 0.094838
Trade: type 3              0.217391                 0.107268
Trade: type 4                   NaN                 0.027778
Trade: type 5                   NaN                 0.074074
Trade: type 6                   NaN                 0.050265
Trade: type 7              0.078125                 0.103495
Transport: type 1          0.000000                 0.045113
Transport: type 2          0.066667                 0.091193
Transport: type 3          0.125000                 0.167722
Transport: type 4          0.145833                 0.102171
University                 0.200000                 0.070909
```

[194]:
```
plt.figure(figsize=(10,14))
sns.heatmap(chart2, annot=False, cmap='YlOrRd', center=0.081)
plt.show()
```



Correlation between target and some numeric variables

[198]:

```
corre = df[['TARGET', 'Updated_DAYS_BIRTH', 'AMT_INCOME_TOTAL','AMT_CREDIT',␣
↪'CNT_FAM_MEMBERS', 'Updated_DAYS_EMPLOYED', 'AMT_INCOME_TOTAL']].corr()
corre
```

[198]:

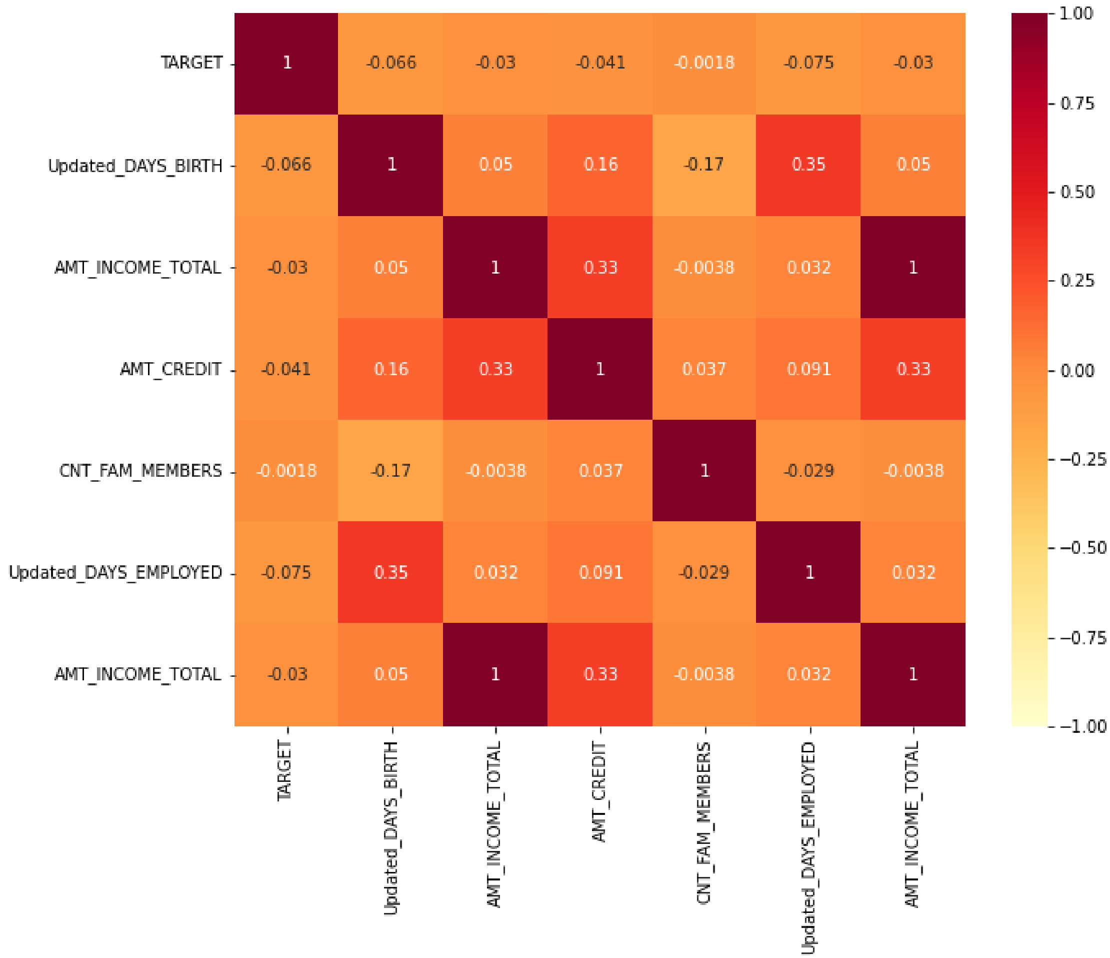
```
                        TARGET  Updated_DAYS_BIRTH  AMT_INCOME_TOTAL  \
TARGET                1.000000           -0.065780         -0.029988
Updated_DAYS_BIRTH   -0.065780            1.000000          0.050481
AMT_INCOME_TOTAL     -0.029988            0.050481          1.000000
AMT_CREDIT           -0.040658            0.157312          0.326937
CNT_FAM_MEMBERS      -0.001834           -0.171677         -0.003800
Updated_DAYS_EMPLOYED -0.074739           0.351604          0.032017
AMT_INCOME_TOTAL     -0.029988            0.050481          1.000000

                      AMT_CREDIT  CNT_FAM_MEMBERS  Updated_DAYS_EMPLOYED  \
TARGET                 -0.040658        -0.001834              -0.074739
Updated_DAYS_BIRTH      0.157312        -0.171677               0.351604
AMT_INCOME_TOTAL        0.326937        -0.003800               0.032017
AMT_CREDIT              1.000000         0.037407               0.091184
CNT_FAM_MEMBERS         0.037407         1.000000              -0.028820
Updated_DAYS_EMPLOYED   0.091184        -0.028820               1.000000
AMT_INCOME_TOTAL        0.326937        -0.003800               0.032017

                      AMT_INCOME_TOTAL
TARGET                       -0.029988
Updated_DAYS_BIRTH            0.050481
AMT_INCOME_TOTAL              1.000000
AMT_CREDIT                    0.326937
CNT_FAM_MEMBERS              -0.003800
Updated_DAYS_EMPLOYED         0.032017
AMT_INCOME_TOTAL              1.000000
```

[202]:

```
plt.figure(figsize=(10,8))
sns.heatmap(corre, annot=True, cmap='YlOrRd', vmin=-1,vmax=1)
plt.show()
```



[ ]:
```
%%capture
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('final.ipynb')
```