Table of Contents

Please click on the link to jump to specific section

EDA Credit Case study Overivew

Risk analytics in banking and financial services and understand how data is used to minimise the risk of losing money while lending to customers

Two types of risks are associated with the bank's decision:

- If the applicant is likely to repay the loan, then not approving the loan result in a loss of business to the company
- If the applicant is not likely to repay the loan, i.e., he/she is likely to default, then approving the loan may lead to a financial loss for the company

Below information can be taken from loan application

1. The client with payment difficulties
2. All other cases

Four types of decisions that could be taken by client/company

1. Approved
2. Cancelled
3. Refused
4. Unused offer

Business Objectives

```
1. Identify the variable which are strong indicator of default
2. These variable will be utilized in portfolio and risk assessment
```

Understanding Data

1. **'application_data.csv'** contains all the information of the client at the time of application. The data is about whether a client has payment difficulties.
2. **'previous_application.csv'** contains information about the client’s previous loan data. It contains the data whether the previous application had been Approved, Cancelled, Refused or Unused offer.
3. **'columns_description.csv'** is data dictionary which describes the meaning of the variables

Identification of variables and data types

In [1845]:

```
# Filtering out the warnings
import warnings
warnings.filterwarnings('ignore')
```

In [1846]:

```
# All library imports
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

In [1847]:

```
# Reading csv and assign to variable
# For simplicity, let's call dataframes as 'application_df' and 'prev_applicatio

application_df = pd.read_csv('application_data.csv')
prev_application_df = pd.read_csv('previous_application.csv')
```

In [1848]:

```
application_df.head()
```

Out[1848]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OV
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 rows × 122 columns

In [1849]:

```
1 prev_application_df.head()
```

Out[1849]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AM
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 rows × 37 columns

In [1850]:

```
1 application_df.shape
```

Out[1850]:

```
(307511, 122)
```

In [1851]:

```
1 prev_application_df.shape
```

Out[1851]:

```
(1670214, 37)
```

In [1852]:

```
1 application_df.info()
```

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

In [1853]:

```
prev_application_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
```

In [1854]:

```
# Numerical values
application_df.describe(include=[np.number])
```

Out[1854]:

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

8 rows × 106 columns

In [1855]:

```
# Numerical and categorical values
application_df.describe(include='all')
```

Out[1855]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CA
count	307511.000000	307511.000000	307511	307511	3075
unique	NaN	NaN	2	3	
top	NaN	NaN	Cash loans	F	
freq	NaN	NaN	278232	202448	2029
mean	278180.518577	0.080729	NaN	NaN	Na
std	102790.175348	0.272419	NaN	NaN	Na
min	100002.000000	0.000000	NaN	NaN	Na
25%	189145.500000	0.000000	NaN	NaN	Na
50%	278202.000000	0.000000	NaN	NaN	Na
75%	367142.500000	0.000000	NaN	NaN	Na
max	456255.000000	1.000000	NaN	NaN	Na

11 rows × 122 columns

In [1856]:

```
# Numerical values
prev_application_df.describe(include=[np.number])
```

Out[1856]:

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

8 rows × 21 columns

In [1857]:

```
# Numerical and Categorical values
prev_application_df.describe(include='all')
```

Out[1857]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATIOI
count	1.670214e+06	1.670214e+06	1670214	1.297979e+06	1.670214e+0
unique	NaN	NaN	4	NaN	Nal
top	NaN	NaN	Cash loans	NaN	Nal
freq	NaN	NaN	747553	NaN	Nal
mean	1.923089e+06	2.783572e+05	NaN	1.595512e+04	1.752339e+0
std	5.325980e+05	1.028148e+05	NaN	1.478214e+04	2.927798e+0
min	1.000001e+06	1.000010e+05	NaN	0.000000e+00	0.000000e+0
25%	1.461857e+06	1.893290e+05	NaN	6.321780e+03	1.872000e+0
50%	1.923110e+06	2.787145e+05	NaN	1.125000e+04	7.104600e+0
75%	2.384280e+06	3.675140e+05	NaN	2.065842e+04	1.803600e+0
max	2.845382e+06	4.562550e+05	NaN	4.180581e+05	6.905160e+0

11 rows × 37 columns

In [1858]:

```
application_df.dtypes
```

Out[1858]:

```
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
```

In [1859]:

```
# since we are unable to see all columns
for column in application_df:
    print(column,'\t', application_df[column].dtypes)
```

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

In [1860]:

```
# since we are unable to see all columns
for column in prev_application_df:
    print(column,'\t', prev_application_df[column].dtypes)
```

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

Fixing the rows and columns

From the observation, **application_df** has 124 columns, of which there are many irrelevant columns that may not be required for analysis and are off the objective of the analysis. In this section we will remove such columns first and then we will remove the rows which are insufficient for analysis

Example: EXT_SOURCE_1, EXT_SOURCE_2, EXT_SOURCE_3 and so on

In [1861]:

```
# marking irrelevant columns (mostly 0 or no information) and removing it
irrelevant_columns = [i for i in range (41,122,1)]
application_df.drop(application_df.columns[irrelevant_columns], axis=1, inplace=
```

In [1862]:

```
# verification
print(application_df.shape)
application_df.head()
```

```
(307511, 41)
```

Out[1862]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OV
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 rows × 41 columns

In [1863]:

```
prev_application_df.head()
```

Out[1863]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AM
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 rows × 37 columns

In [1864]:

```
application_df['AMT_GOODS_PRICE']= application_df['AMT_GOODS_PRICE'].apply(lambd
```

In [1865]:

```
application_df['AMT_GOODS_PRICE']
```

Out[1865]:

```
0          351000.0
1         1129500.0
2          135000.0
3          297000.0
4          513000.0
             ...
307506     225000.0
307507     225000.0
307508     585000.0
307509     319500.0
307510     675000.0
Name: AMT_GOODS_PRICE, Length: 307511, dtype: float64
```

Missing value treatment

In [1866]:

```
# finding null values through out the dataframe
application_df.isnull().sum()
```

Out[1866]:

```
SK_ID_CURR                          0
TARGET                              0
NAME_CONTRACT_TYPE                  0
CODE_GENDER                         0
FLAG_OWN_CAR                        0
FLAG_OWN_REALTY                     0
CNT_CHILDREN                        0
AMT_INCOME_TOTAL                    0
AMT_CREDIT                          0
AMT_ANNUITY                        12
AMT_GOODS_PRICE                   278
NAME_TYPE_SUITE                  1292
NAME_INCOME_TYPE                    0
NAME_EDUCATION_TYPE                 0
NAME_FAMILY_STATUS                  0
NAME_HOUSING_TYPE                   0
REGION_POPULATION_RELATIVE          0
DAYS_BIRTH                          0
DAYS_EMPLOYED                       0
DAYS_REGISTRATION                   0
DAYS_ID_PUBLISH                     0
OWN_CAR_AGE                    202929
FLAG_MOBIL                          0
FLAG_EMP_PHONE                      0
FLAG_WORK_PHONE                     0
FLAG_CONT_MOBILE                    0
FLAG_PHONE                          0
FLAG_EMAIL                          0
OCCUPATION_TYPE                 96391
CNT_FAM_MEMBERS                     2
REGION_RATING_CLIENT                0
REGION_RATING_CLIENT_W_CITY         0
WEEKDAY_APPR_PROCESS_START          0
HOUR_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
dtype: int64
```

In [1867]:

```
application_df.shape
```

Out[1867]:

```
(307511, 41)
```

In [1868]:

```
# from the above 2 cells, we can see that column 'OCCUPATION_TYPE' and 'OWN_CAR_
#significantly amount of null value

# let's inspect 'OWN_CAR_AGE': Age of client's car
application_df['OWN_CAR_AGE'].isna().sum()
```

Out[1868]:

```
202929
```

In [1869]:

```
# seems like nan is for the clients who never own car, we may create new column
# useful to derive more insights
application_df['HAS_OWN_CAR'] = np.where(application_df['OWN_CAR_AGE'].isnull(),
```

In [1870]:

```
# verification
application_df[application_df['HAS_OWN_CAR']==False]
```

Out[1870]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FL/
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
3	100006	0	Cash loans	F	N	
4	100007	0	Cash loans	M	N	
5	100008	0	Cash loans	M	N	
...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	
307507	456252	0	Cash loans	F	N	
307508	456253	0	Cash loans	F	N	
307509	456254	1	Cash loans	F	N	
307510	456255	0	Cash loans	F	N	

202929 rows × 42 columns

In [1871]:

```
# let's inspect 'OCCUPATION_TYPE'
application_df['OCCUPATION_TYPE'].isna().sum()
```

Out[1871]:

```
96391
```

In [1872]:

```
application_df['OCCUPATION_TYPE'].unique()
```

Out[1872]:

```
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 staf
f',
       'HR staff'], dtype=object)
```

In [1873]:

```
# from the above observation, null values are significantly high,
# seems like the client refrain to tell information, low chances of human error
# let's call all those null values as 'Others'
application_df['OCCUPATION_TYPE'].fillna('Others', inplace=True)
```

In [1874]:

```
# verification
application_df['OCCUPATION_TYPE'].isna().sum()
```

Out[1874]:

```
0
```

In [1875]:

```
application_df['OCCUPATION_TYPE'].unique()
```

Out[1875]:

```
array(['Laborers', 'Core staff', 'Accountants', 'Managers', 'Others',
       '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 staf
f',
       'HR staff'], dtype=object)
```

In [1876]:

```
# let's inspect column 'NAME_TYPE_SUITE': Who was accompanying client when he wa
application_df['NAME_TYPE_SUITE'].unique()
```

Out[1876]:

```
array(['Unaccompanied', 'Family', 'Spouse, partner', 'Children',
       'Other_A', nan, 'Other_B', 'Group of people'], dtype=object)
```

In [1877]:

```
application_df['NAME_TYPE_SUITE'].isna().sum()
```

Out[1877]:

```
1292
```

In [1878]:

```
# value is insignificantly low; let's see what are the most used values because
application_df['NAME_TYPE_SUITE'].value_counts()
```

Out[1878]:

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

In [1879]:

```
# most of the applications has 'NAME_TYPE_SUITE' as 'Unaccompanied', so let's fi
application_df['NAME_TYPE_SUITE'].fillna('Unaccompanied', inplace=True)
```

In [1880]:

```
# verification
application_df['NAME_TYPE_SUITE'].unique()
```

Out[1880]:

```
array(['Unaccompanied', 'Family', 'Spouse, partner', 'Children',
       'Other_A', 'Other_B', 'Group of people'], dtype=object)
```

In [1881]:

```
application_df['NAME_TYPE_SUITE'].isna().sum()
```

Out[1881]:

```
0
```

In [1882]:

```
# let inspect Column 'AMT_GOODS_PRICE': For consumer loans it is the price of th
application_df['AMT_GOODS_PRICE'].isna().sum()
```

Out[1882]:

```
278
```

In [1883]:

```
application_df[application_df['AMT_GOODS_PRICE'].isna() & application_df['TARGET
```

Out[1883]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLA
7880	109190	1	Revolving loans	F	N	
41099	147593	1	Revolving loans	F	N	
50540	158525	1	Revolving loans	F	N	
56002	164897	1	Revolving loans	F	N	
69461	180561	1	Revolving loans	F	N	
78786	191335	1	Revolving loans	F	N	
86000	199789	1	Revolving loans	F	N	
86005	199794	1	Revolving loans	F	N	
124770	244697	1	Revolving loans	F	N	
152898	277210	1	Revolving loans	F	N	
153801	278254	1	Revolving loans	F	N	
186634	316367	1	Revolving loans	M	N	
190113	320433	1	Revolving loans	F	N	
210718	344187	1	Revolving loans	F	N	
214803	348904	1	Revolving loans	F	N	
226725	362616	1	Revolving loans	F	N	
229877	366256	1	Revolving loans	M	N	
249616	388813	1	Revolving loans	F	N	
253126	392897	1	Revolving loans	M	N	
260704	401702	1	Revolving loans	F	N	
270616	413674	1	Revolving loans	F	N	

21 rows × 42 columns

In [1884]:

```
# only 21 clients are having difficulty in paying loans and all of the loans are
# that is why there no value for AMT_GOODS_PRICE
```

In [1885]:

```
# let's inspect AMT_ANNUITY
application_df[application_df['AMT_ANNUITY'].isnull()]
```

Out[1885]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLA
47531	155054	0	Cash loans	M	N	
50035	157917	0	Cash loans	F	N	
51594	159744	0	Cash loans	F	N	
55025	163757	0	Cash loans	F	N	
59934	169487	0	Cash loans	M	Y	
75873	187985	0	Cash loans	M	Y	
89343	203726	0	Cash loans	F	Y	
123872	243648	0	Cash loans	F	N	
207186	340147	0	Cash loans	M	N	
227939	364022	0	Cash loans	F	N	
239329	377174	0	Cash loans	F	N	
241835	379997	0	Cash loans	F	N	

12 rows × 42 columns

In [1886]:

```
#since only insignificant amount of null values, so dropping off those
application_df = application_df[~application_df['AMT_ANNUITY'].isnull()]
```

In [1887]:

```
# verification
application_df['AMT_ANNUITY'].isnull().sum()
```

Out[1887]:

0

In [1888]:

```
# inspecting prev_application_df
prev_application_df.shape
```

Out[1888]:

(1670214, 37)

In [1889]:

```
prev_application_df.isnull().sum()
```

Out[1889]:

```
SK_ID_PREV                          0
SK_ID_CURR                          0
NAME_CONTRACT_TYPE                  0
AMT_ANNUITY                    372235
AMT_APPLICATION                     0
AMT_CREDIT                          1
AMT_DOWN_PAYMENT               895844
AMT_GOODS_PRICE                385515
WEEKDAY_APPR_PROCESS_START          0
HOUR_APPR_PROCESS_START             0
FLAG_LAST_APPL_PER_CONTRACT         0
NFLAG_LAST_APPL_IN_DAY              0
RATE_DOWN_PAYMENT              895844
RATE_INTEREST_PRIMARY         1664263
RATE_INTEREST_PRIVILEGED      1664263
NAME_CASH_LOAN_PURPOSE              0
NAME_CONTRACT_STATUS                0
DAYS_DECISION                       0
NAME_PAYMENT_TYPE                   0
CODE_REJECT_REASON                  0
NAME_TYPE_SUITE                820405
NAME_CLIENT_TYPE                    0
NAME_GOODS_CATEGORY                 0
NAME_PORTFOLIO                      0
NAME_PRODUCT_TYPE                   0
CHANNEL_TYPE                        0
SELLERPLACE_AREA                    0
NAME_SELLER_INDUSTRY                0
CNT_PAYMENT                    372230
NAME_YIELD_GROUP                    0
PRODUCT_COMBINATION               346
DAYS_FIRST_DRAWING             673065
DAYS_FIRST_DUE                 673065
DAYS_LAST_DUE_1ST_VERSION      673065
DAYS_LAST_DUE                  673065
DAYS_TERMINATION               673065
NFLAG_INSURED_ON_APPROVAL      673065
dtype: int64
```

In [1890]:

```
# value is insignificantly low; let's see what are the most used values because
prev_application_df['NAME_TYPE_SUITE'].value_counts()
```

Out[1890]:

```
Unaccompanied      508970
Family             213263
Spouse, partner     67069
Children            31566
Other_B             17624
Other_A              9077
Group of people      2240
Name: NAME_TYPE_SUITE, dtype: int64
```

In [1891]:

```
# most of the applications has 'NAME_TYPE_SUITE' as 'Unaccompanied', so let's fi
prev_application_df['NAME_TYPE_SUITE'].fillna('Unaccompanied', inplace=True)
```

In [1892]:

```
# verification
prev_application_df['NAME_TYPE_SUITE'].unique()
```

Out[1892]:

```
array(['Unaccompanied', 'Spouse, partner', 'Family', 'Children',
       'Other_B', 'Other_A', 'Group of people'], dtype=object)
```

In [1893]:

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

Out[1893]:

0

In [1894]:

```
# since most of the data missing from columns RATE_INTEREST_PRIMARY and RATE_INT
# hence droping those columns
prev_application_df.drop(columns=['RATE_INTEREST_PRIMARY','RATE_INTEREST_PRIVILE
```

In [1895]:

```
prev_application_df.head()
```

Out[1895]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AM
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 rows × 35 columns

In [1896]:

```
# since most of the data missing from columns
#DAYS_FIRST_DRAWING
#DAYS_FIRST_DUE
#DAYS_LAST_DUE_1ST_VERSION
#DAYS_LAST_DUE
#DAYS_TERMINATION
#NFLAG_INSURED_ON_APPROVAL
# hence droping these columns

prev_application_df.drop(columns = ['DAYS_FIRST_DRAWING','DAYS_FIRST_DUE','DAYS_
                                    'DAYS_LAST_DUE', 'DAYS_TERMINATION', 'NFLAG_INS
                         inplace = True)
```

In [1897]:

```
prev_application_df.head()
```

Out[1897]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AM
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 rows × 29 columns

In [1898]:

```
# AMT_DOWN_PAYMENT and RATE_DOWN_PAYMENT are missing significantly and cannot be
# hence dropping those columns
prev_application_df.drop(columns = ['AMT_DOWN_PAYMENT', 'RATE_DOWN_PAYMENT'],
                         inplace = True)
```

In [1899]:

```
prev_application_df.head()
```

Out[1899]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AM
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 rows × 27 columns

In [1900]:

```
prev_application_df.isnull().sum()
```

Out[1900]:

```
SK_ID_PREV                          0
SK_ID_CURR                          0
NAME_CONTRACT_TYPE                  0
AMT_ANNUITY                    372235
AMT_APPLICATION                     0
AMT_CREDIT                          1
AMT_GOODS_PRICE                385515
WEEKDAY_APPR_PROCESS_START          0
HOUR_APPR_PROCESS_START             0
FLAG_LAST_APPL_PER_CONTRACT         0
NFLAG_LAST_APPL_IN_DAY              0
NAME_CASH_LOAN_PURPOSE              0
NAME_CONTRACT_STATUS                0
DAYS_DECISION                       0
NAME_PAYMENT_TYPE                   0
CODE_REJECT_REASON                  0
NAME_TYPE_SUITE                     0
NAME_CLIENT_TYPE                    0
NAME_GOODS_CATEGORY                 0
NAME_PORTFOLIO                      0
NAME_PRODUCT_TYPE                   0
CHANNEL_TYPE                        0
SELLERPLACE_AREA                    0
NAME_SELLER_INDUSTRY                0
CNT_PAYMENT                    372230
NAME_YIELD_GROUP                    0
PRODUCT_COMBINATION               346
dtype: int64
```

In [1901]:

```
prev_application_df.shape
```

Out[1901]:

```
(1670214, 27)
```

In [1902]:

```
# Columns AMT_ANNUITY, AMT_GOODS_PRICE, CNT_PAYMENT are almost 20% null
# removing those rows

prev_application_df = prev_application_df[~prev_application_df['AMT_ANNUITY'].is
prev_application_df = prev_application_df[~prev_application_df['AMT_GOODS_PRICE'
prev_application_df = prev_application_df[~prev_application_df['CNT_PAYMENT'].is
```

In [1903]:

```
# verification
prev_application_df.isnull().sum()
```

Out[1903]:

```
SK_ID_PREV                     0
SK_ID_CURR                     0
NAME_CONTRACT_TYPE             0
AMT_ANNUITY                    0
AMT_APPLICATION                0
AMT_CREDIT                     0
AMT_GOODS_PRICE                0
WEEKDAY_APPR_PROCESS_START     0
HOUR_APPR_PROCESS_START        0
FLAG_LAST_APPL_PER_CONTRACT    0
NFLAG_LAST_APPL_IN_DAY         0
NAME_CASH_LOAN_PURPOSE         0
NAME_CONTRACT_STATUS           0
DAYS_DECISION                  0
NAME_PAYMENT_TYPE              0
CODE_REJECT_REASON             0
NAME_TYPE_SUITE                0
NAME_CLIENT_TYPE               0
NAME_GOODS_CATEGORY            0
NAME_PORTFOLIO                 0
NAME_PRODUCT_TYPE              0
CHANNEL_TYPE                   0
SELLERPLACE_AREA               0
NAME_SELLER_INDUSTRY           0
CNT_PAYMENT                    0
NAME_YIELD_GROUP               0
PRODUCT_COMBINATION            0
dtype: int64
```

In [1904]:

```
prev_application_df.shape
```

Out[1904]:

```
(1246320, 27)
```

In [1905]:

```
application_df.shape
```

Out[1905]:

```
(307499, 42)
```

Outlier treatment

Let's try to find outliers for below columns

from application_df

- AMT_INCOME_TOTAL
- AMT_CREDIT
- AMT_ANNUITY
- AMT_GOODS_PRICE

from prev_application_df

- AMT_ANNUITY
- AMT_CREDIT
- AMT_GOODS_PRICE

In [1906]:

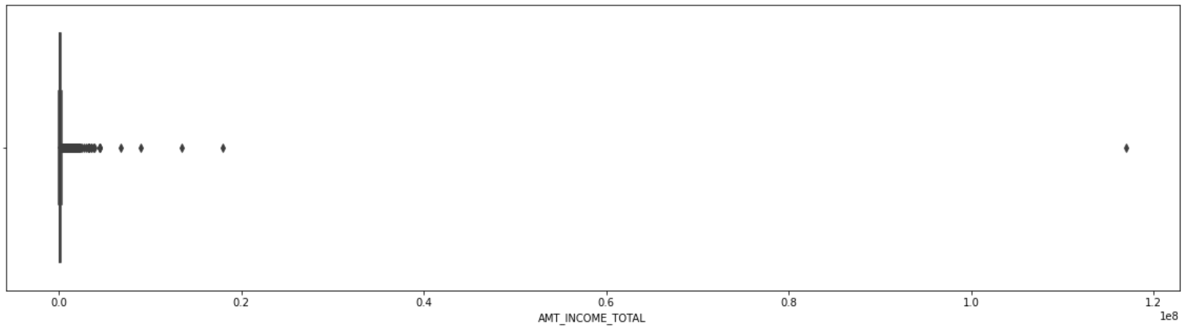
```
# let's inspect column AMT_INCOME_TOTAL from application_df
application_df.AMT_INCOME_TOTAL.describe().apply("{0:.2f}".format)
```

Out[1906]:

```
count        307499.00
mean         168797.23
std          237127.37
min           25650.00
25%          112500.00
50%          146997.00
75%          202500.00
max       117000000.00
Name: AMT_INCOME_TOTAL, dtype: object
```

In [1907]:

```
plt.figure(figsize=[20,5])
sns.boxplot(application_df.AMT_INCOME_TOTAL)
plt.show()
```



In [1908]:

```
application_df.AMT_INCOME_TOTAL.quantile([0.05,0.10,0.5,0.7,0.85, 0.9,0.95, 0.9
```

Out[1908]:

```
0.05      67500.0
0.10      81000.0
0.50     146997.0
0.70     180000.0
0.85     234000.0
0.90     270000.0
0.95     337500.0
0.99     472500.0
Name: AMT_INCOME_TOTAL, dtype: float64
```

In [1909]:

```
application_df[application_df.AMT_INCOME_TOTAL>337500].describe()
```

Out[1909]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_
count	14035.000000	14035.000000	14035.000000	1.403500e+04	1.403500e+04	140
mean	278022.099394	0.058140	0.477948	4.727078e+05	1.002075e+06	449
std	102949.543339	0.234017	0.759432	1.024482e+06	5.366317e+05	224
min	100010.000000	0.000000	0.000000	3.375450e+05	4.500000e+04	35
25%	189469.500000	0.000000	0.000000	3.600000e+05	5.925600e+05	308
50%	277411.000000	0.000000	0.000000	4.050000e+05	9.000000e+05	421
75%	368321.000000	0.000000	1.000000	4.500000e+05	1.306008e+06	548
max	456240.000000	1.000000	5.000000	1.170000e+08	4.050000e+06	2580

8 rows × 29 columns

In [1910]:

```
application_df[(application_df.AMT_INCOME_TOTAL>3375000.0) & (application_df.TAR
```

Out[1910]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLA(
12840	114967	1	Cash loans	F	N	

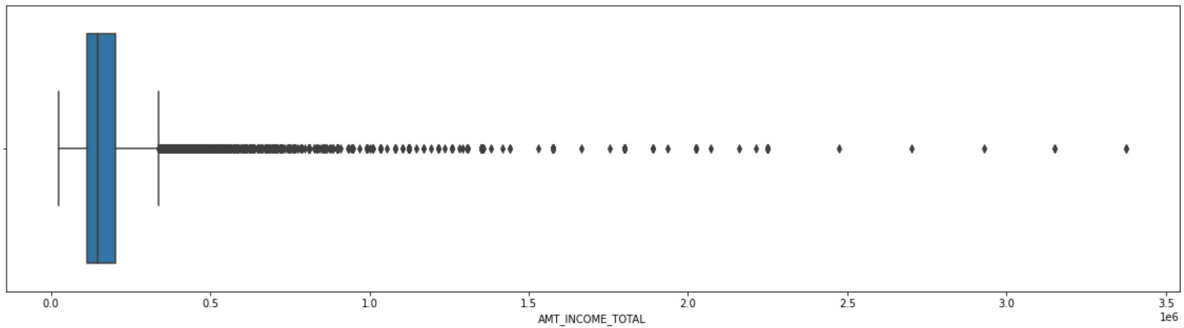
1 rows × 42 columns

In [1911]:

```
# from the above, we can remove clients above 95% as they are high earner and or
# difficulty paying the Installment
application_df = application_df[~(application_df.AMT_INCOME_TOTAL>3375000)]
```

In [1912]:

```
#verification
plt.figure(figsize=[20,5])
sns.boxplot(application_df.AMT_INCOME_TOTAL)
plt.show()
```



In [1913]:

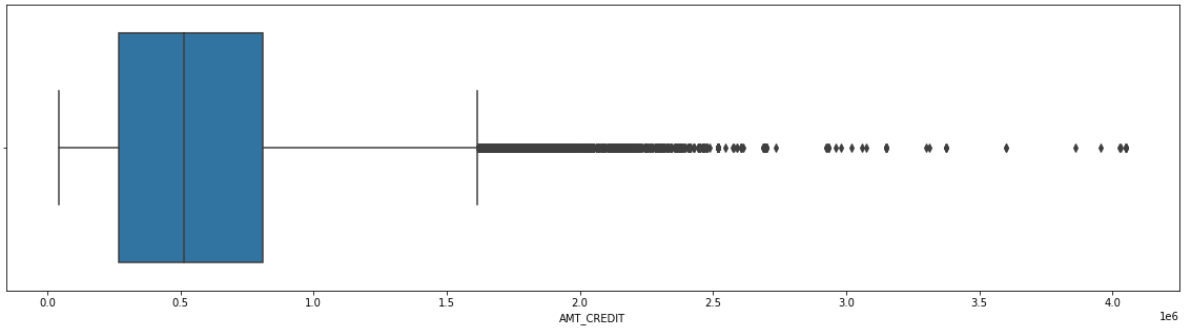
```
# let's inspect column AMT_CREDIT from application_df
application_df.AMT_CREDIT.describe().apply("{0:.2f}".format)
```

Out[1913]:

```
count     307486.00
mean      599006.65
std       402475.60
min        45000.00
25%       270000.00
50%       513531.00
75%       808650.00
max      4050000.00
Name: AMT_CREDIT, dtype: object
```

In [1914]:

```
plt.figure(figsize=[20,5])
sns.boxplot(application_df.AMT_CREDIT)
plt.show()
```



In [1915]:

```
application_df.AMT_CREDIT.quantile([0.05,0.10,0.5,0.7,0.85, 0.9,0.95, 0.99])
```

Out[1915]:

```
0.05     135000.0
0.10     180000.0
0.50     513531.0
0.70     755190.0
0.85    1024740.0
0.90    1133748.0
0.95    1350000.0
0.99    1854000.0
Name: AMT_CREDIT, dtype: float64
```

In [1916]:

```
application_df[(application_df.AMT_CREDIT>1854000.0) & (application_df.TARGET ==
```

Out[1916]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FL/
678	100784	1	Cash loans	F	N	
5069	105925	1	Cash loans	F	Y	
8940	110403	1	Cash loans	M	Y	
10149	111813	1	Cash loans	M	N	
11474	113359	1	Cash loans	F	N	
...	...	...	...	...	...	
295674	442560	1	Cash loans	M	Y	
297164	444280	1	Cash loans	M	Y	
299225	446646	1	Cash loans	M	Y	
301841	449691	1	Cash loans	F	Y	
305180	453583	1	Cash loans	M	Y	

124 rows × 42 columns

In [1917]:

```
application_df[(application_df.AMT_CREDIT>1854000) & (application_df.TARGET ==0)
```

Out[1917]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FL/
189	100219	0	Cash loans	M	N	
337	100389	0	Cash loans	M	Y	
341	100393	0	Cash loans	M	Y	
441	100508	0	Cash loans	F	Y	
485	100559	0	Cash loans	F	Y	
...	...	...	...	...	...	
307055	455739	0	Cash loans	F	N	
307095	455785	0	Cash loans	F	Y	
307165	455868	0	Cash loans	F	Y	
307214	455922	0	Cash loans	M	Y	
307422	456155	0	Cash loans	F	N	

2950 rows × 42 columns

In [1918]:

```
# Although only 124 clients with extremly high credit,are having difficulty payi
# these are the credit amount, shouldn't be imputed
```

In [1919]:

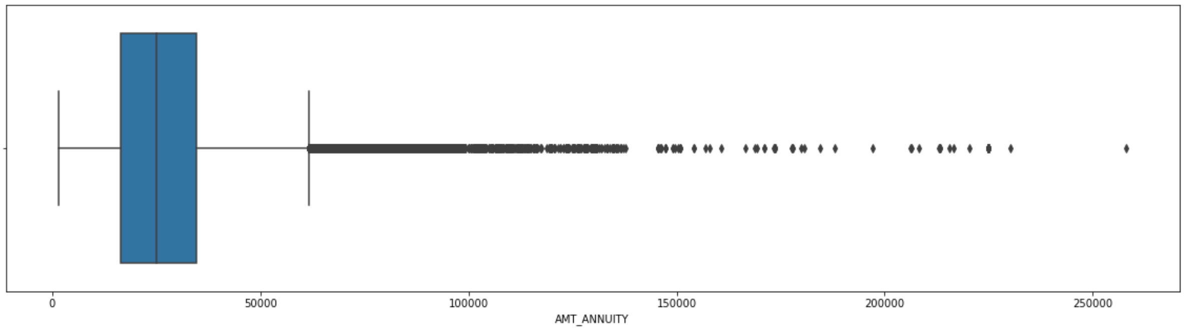
```
# let's inspect column AMT_ANNUITY from application_df
application_df.AMT_ANNUITY.describe().apply("{0:.2f}".format)
```

Out[1919]:

```
count    307486.00
mean      27106.19
std       14485.40
min        1615.50
25%       16524.00
50%       24903.00
75%       34596.00
max      258025.50
Name: AMT_ANNUITY, dtype: object
```

In [1920]:

```
plt.figure(figsize=[20,5])
sns.boxplot(application_df.AMT_ANNUITY)
plt.show()
```



In [1921]:

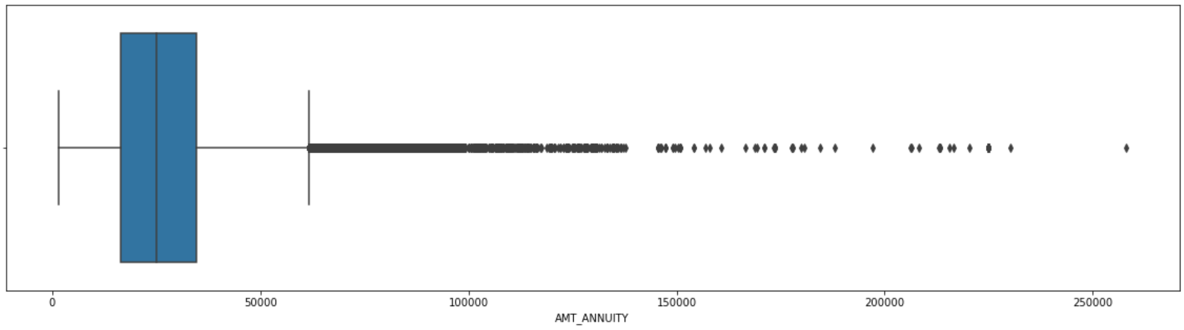
```
# let's inspect column AMT_GOODS_PRICE from application_df
application_df.AMT_GOODS_PRICE.describe().apply("{0:.2f}".format)
```

Out[1921]:

```
count     307208.00
mean      538376.64
std       369427.12
min        40500.00
25%       238500.00
50%       450000.00
75%       679500.00
max      4050000.00
Name: AMT_GOODS_PRICE, dtype: object
```

In [1922]:

```
plt.figure(figsize=[20,5])
sns.boxplot(application_df.AMT_ANNUITY)
plt.show()
```



In [1923]:

```
# AMT_GOODS_PRICE and AMT_ANNUITY are the factor that may affect the credit with
# further analysis will be done in multivariate analysis section
# hence keeping all the records even outliers at this point of time
```

Standardising values

In [1924]:

```
# Rounding off all the numerical values to 2 decimal  and transforming FLAGS to
```

In [1925]:

```
application_df['AMT_INCOME_TOTAL'] = application_df['AMT_INCOME_TOTAL'].apply(la
```

In [1926]:

```
application_df['AMT_CREDIT'] = application_df['AMT_CREDIT'].apply(lambda x:round
```

In [1927]:

```
application_df['AMT_ANNUITY'] = application_df['AMT_ANNUITY'].apply(lambda x:rou
```

In [1928]:

```
application_df['AMT_GOODS_PRICE'] = application_df['AMT_GOODS_PRICE'].apply(lamb
```

In [1929]:

```
prev_application_df['AMT_ANNUITY'] = prev_application_df['AMT_ANNUITY'].apply(la
```

In [1930]:

```
prev_application_df['AMT_CREDIT'] = prev_application_df['AMT_CREDIT'].apply(lamb
```

In [1931]:

```
prev_application_df['AMT_GOODS_PRICE'] = prev_application_df['AMT_GOODS_PRICE'].
```

In [1932]:

```
application_df['FLAG_OWN_CAR'].unique()
```

Out[1932]:

```
array(['N', 'Y'], dtype=object)
```

In [1933]:

```
application_df['FLAG_OWN_CAR'] = application_df['FLAG_OWN_CAR'].apply(lambda x :
```

In [1934]:

```
application_df['FLAG_OWN_CAR'].unique()
```

Out[1934]:

```
array([False,  True])
```

In [1935]:

```
application_df['FLAG_OWN_CAR'].dtype
```

Out[1935]:

```
dtype('bool')
```

In [1936]:

```
application_df['FLAG_OWN_REALTY'].unique()
```

Out[1936]:

```
array(['Y', 'N'], dtype=object)
```

In [1937]:

```
application_df['FLAG_OWN_REALTY']=application_df['FLAG_OWN_REALTY'].apply(lambda
```

In [1938]:

```
application_df['FLAG_OWN_REALTY'].unique()
```

Out[1938]:

```
array([ True, False])
```

In [1939]:

```
application_df['FLAG_OWN_CAR'].dtype
```

Out[1939]:

```
dtype('bool')
```

In [1940]:

```
application_df['FLAG_OWN_REALTY'].unique()
```

Out[1940]:

```
array([ True, False])
```

In [1941]:

```
application_df['AGE'] = application_df['DAYS_BIRTH'].apply(lambda x: round(abs(x
```

In [1942]:

```
application_df['AGE']
```

Out[1942]:

```
0         26
1         46
2         52
3         52
4         55
          ..
307506    26
307507    57
307508    41
307509    33
307510    46
Name: AGE, Length: 307486, dtype: int64
```

Categorical Unordered Univariate Analysis

Unordered variable in application_df

- TARGET
- CODE_GENDER
- FLAG_OWN_CAR
- FLAG_OWN_REALTY
- NAME_FAMILY_STATUS

In [1943]:

```
application_df['TARGET'].value_counts()
```

Out[1943]:

```
0    282662
1     24824
Name: TARGET, dtype: int64
```

In [1944]:

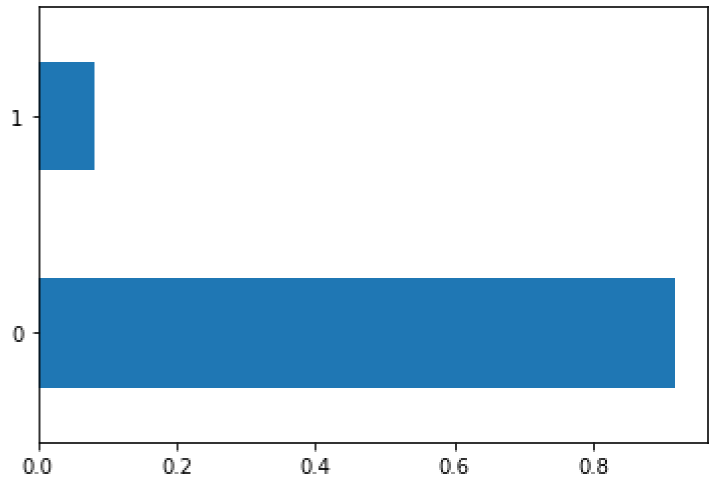
```
application_df['TARGET'].value_counts(normalize=True)
```

Out[1944]:

```
0    0.919268
1    0.080732
Name: TARGET, dtype: float64
```

In [1945]:

```
application_df['TARGET'].value_counts(normalize=True).plot.barh()
plt.show()
```



In [1946]:

```
application_df['CODE_GENDER'].value_counts()
```

Out[1946]:

```
F      202437
M      105045
XNA         4
Name: CODE_GENDER, dtype: int64
```

In [1947]:

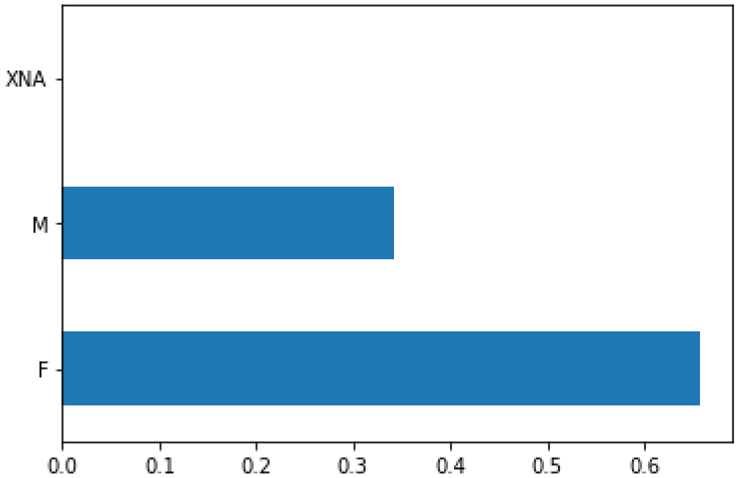
```
application_df['CODE_GENDER'].value_counts(normalize=True)
```

Out[1947]:

```
F      0.658362
M      0.341625
XNA    0.000013
Name: CODE_GENDER, dtype: float64
```

In [1948]:

```
application_df['CODE_GENDER'].value_counts(normalize=True).plot.barh()
plt.show()
```



In [1949]:

```
application_df['FLAG_OWN_CAR'].value_counts()
```

Out[1949]:

```
False    202912
True     104574
Name: FLAG_OWN_CAR, dtype: int64
```

In [1950]:

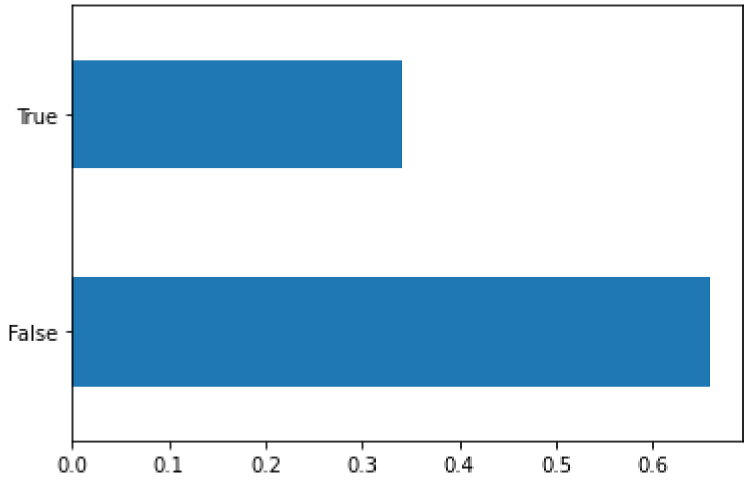
```
application_df['FLAG_OWN_CAR'].value_counts(normalize=True)
```

Out[1950]:

```
False    0.659906
True     0.340094
Name: FLAG_OWN_CAR, dtype: float64
```

In [1951]:

```
application_df['FLAG_OWN_CAR'].value_counts(normalize=True).plot.barh()
plt.show()
```



In [1952]:

```
application_df['FLAG_OWN_REALTY'].value_counts()
```

Out[1952]:

```
True     213303
False     94183
Name: FLAG_OWN_REALTY, dtype: int64
```

In [1953]:

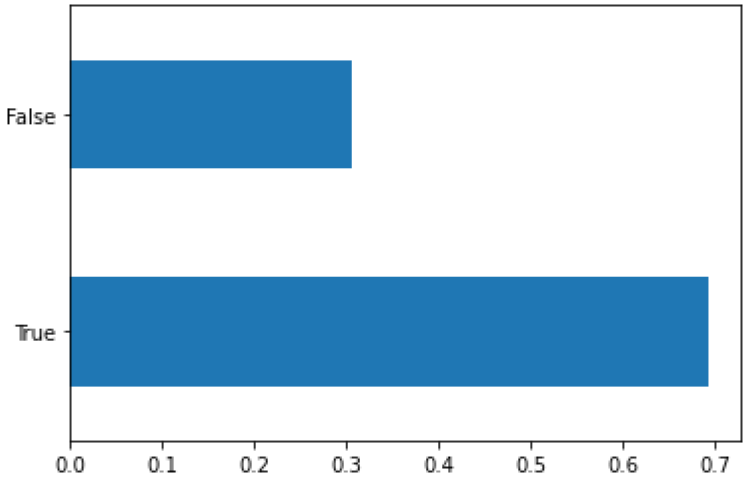
```
application_df['FLAG_OWN_REALTY'].value_counts(normalize=True)
```

Out[1953]:

```
True     0.6937
False    0.3063
Name: FLAG_OWN_REALTY, dtype: float64
```

In [1954]:

```
application_df['FLAG_OWN_REALTY'].value_counts(normalize=True).plot.barh()
plt.show()
```



In [1955]:

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

Out[1955]:

```
Married                 196417
Single / not married     45438
Civil marriage           29771
Separated                19770
Widow                    16088
Unknown                      2
Name: NAME_FAMILY_STATUS, dtype: int64
```

In [1956]:

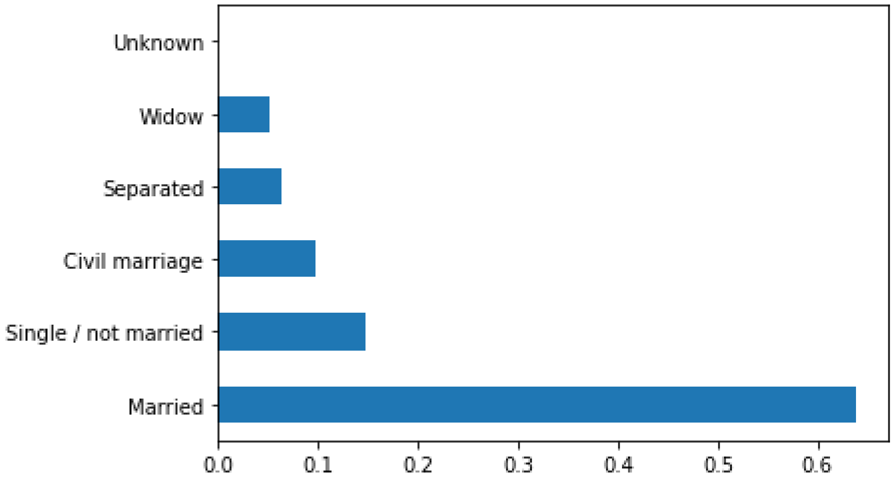
```
application_df['NAME_FAMILY_STATUS'].value_counts(normalize=True)
```

Out[1956]:

```
Married                 0.638784
Single / not married    0.147773
Civil marriage          0.096821
Separated               0.064296
Widow                   0.052321
Unknown                 0.000007
Name: NAME_FAMILY_STATUS, dtype: float64
```

In [1957]:

```
application_df['NAME_FAMILY_STATUS'].value_counts(normalize=True).plot.barh()
plt.show()
```



Categorical Ordered Univariate Analysis

Ordered variable in application_df

- NAME_EDUCATION_TYPE

In [1958]:

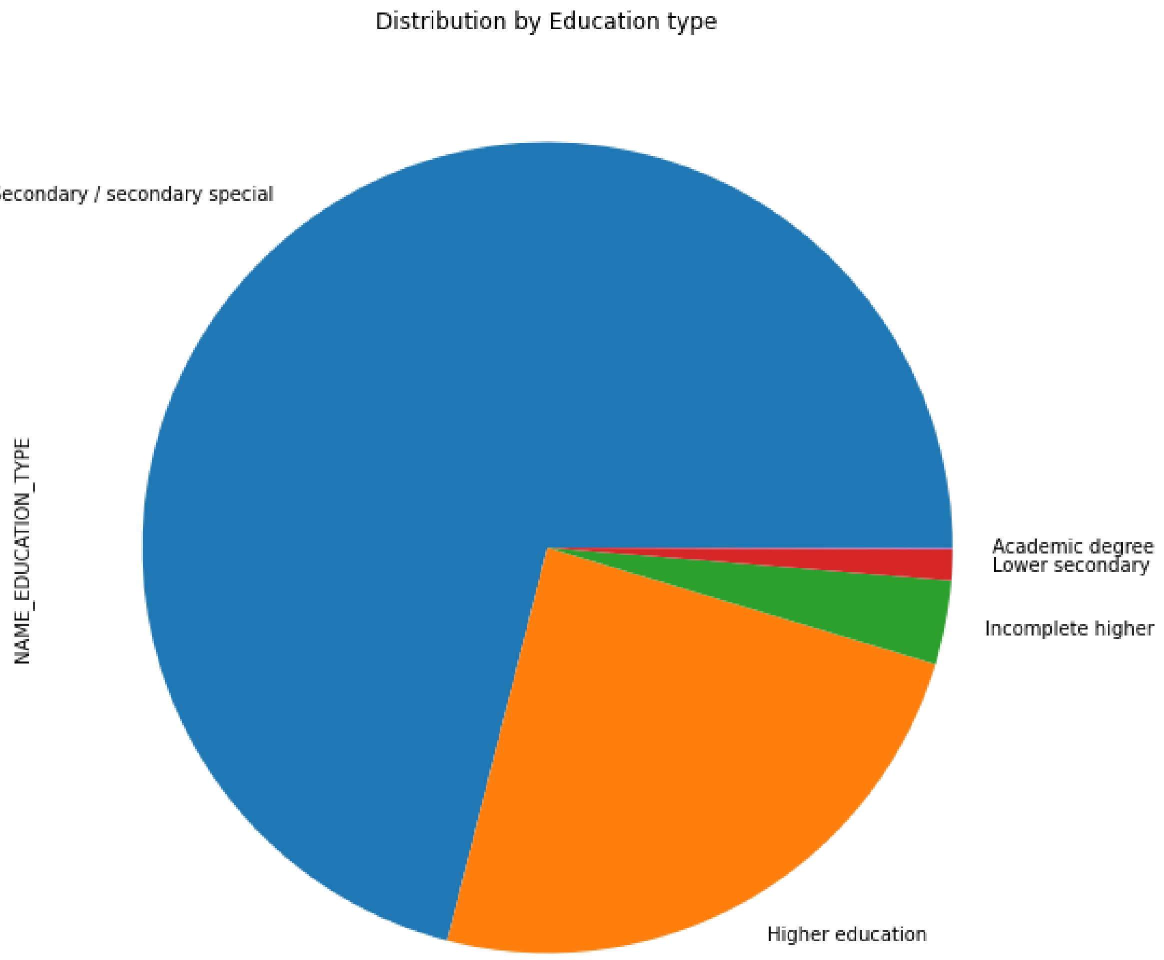
```
application_df['NAME_EDUCATION_TYPE'].value_counts()
```

Out[1958]:

```
Secondary / secondary special    218382
Higher education                  74849
Incomplete higher                 10276
Lower secondary                    3815
Academic degree                     164
Name: NAME_EDUCATION_TYPE, dtype: int64
```

In [1959]:

```
plt.figure(figsize=[10,10])
application_df['NAME_EDUCATION_TYPE'].value_counts(normalize=True).plot.pie(titl
plt.show()
```



Numerical Bivariate and Multivariate Analysis

In [1960]:

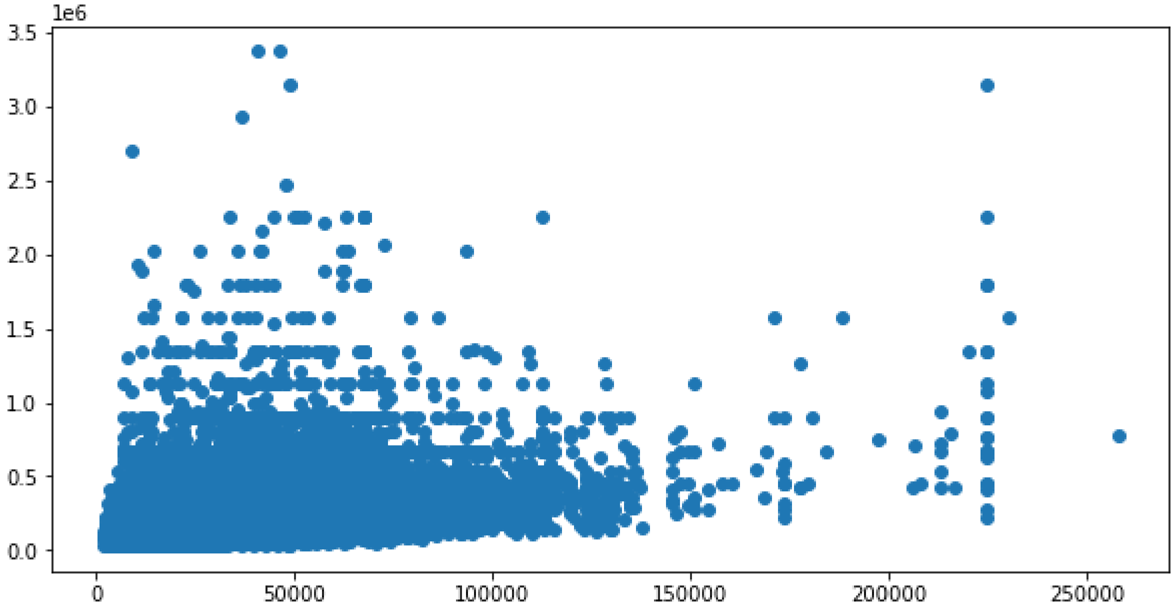
```
# AMT_INCOME_TOTAL              float64
# AMT_CREDIT                    float64
# AMT_ANNUITY                   float64
# AMT_GOODS_PRICE
application_df.dtypes
```

Out[1960]:

```
SK_ID_CURR                      int64
TARGET                          int64
NAME_CONTRACT_TYPE             object
CODE_GENDER                    object
FLAG_OWN_CAR                     bool
FLAG_OWN_REALTY                  bool
CNT_CHILDREN                    int64
AMT_INCOME_TOTAL              float64
AMT_CREDIT                    float64
AMT_ANNUITY                   float64
AMT_GOODS_PRICE               float64
NAME_TYPE_SUITE                object
NAME_INCOME_TYPE               object
NAME_EDUCATION_TYPE            object
NAME_FAMILY_STATUS             object
NAME_HOUSING_TYPE              object
REGION_POPULATION_RELATIVE    float64
DAYS_BIRTH                      int64
DAYS_EMPLOYED                   int64
DAYS_REGISTRATION             float64
DAYS_ID_PUBLISH                 int64
OWN_CAR_AGE                   float64
FLAG_MOBIL                      int64
FLAG_EMP_PHONE                  int64
FLAG_WORK_PHONE                 int64
FLAG_CONT_MOBILE                int64
FLAG_PHONE                      int64
FLAG_EMAIL                      int64
OCCUPATION_TYPE                object
CNT_FAM_MEMBERS               float64
REGION_RATING_CLIENT            int64
REGION_RATING_CLIENT_W_CITY     int64
WEEKDAY_APPR_PROCESS_START     object
HOUR_APPR_PROCESS_START         int64
REG_REGION_NOT_LIVE_REGION      int64
REG_REGION_NOT_WORK_REGION      int64
LIVE_REGION_NOT_WORK_REGION     int64
REG_CITY_NOT_LIVE_CITY          int64
REG_CITY_NOT_WORK_CITY          int64
LIVE_CITY_NOT_WORK_CITY         int64
ORGANIZATION_TYPE              object
HAS_OWN_CAR                      bool
AGE                             int64
dtype: object
```

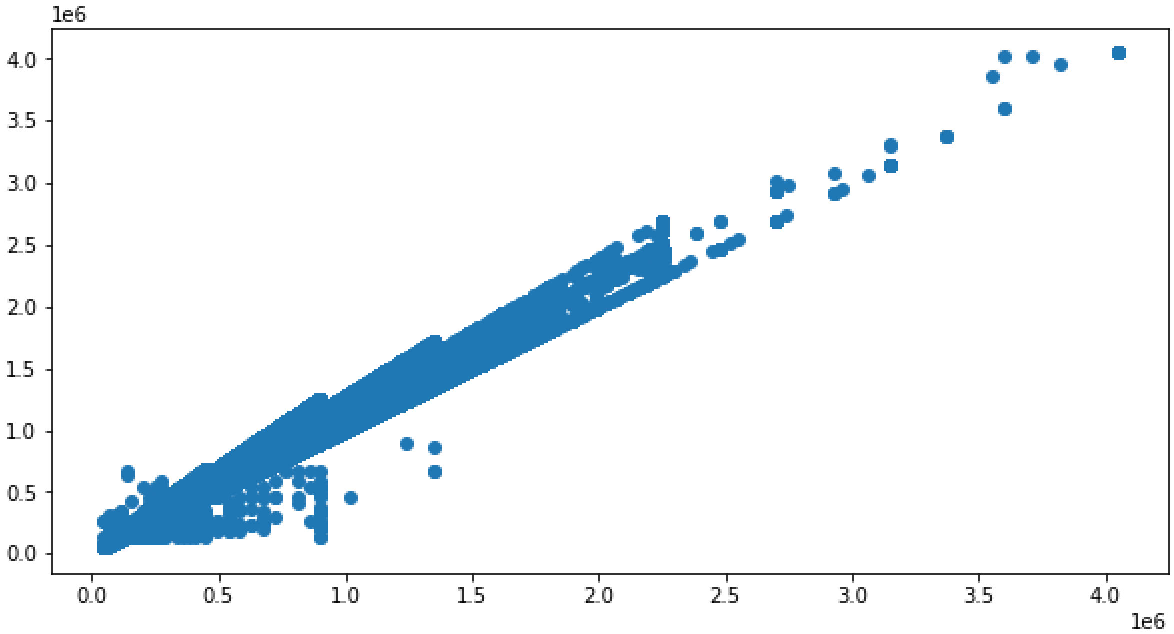
In [1961]:

```
plt.figure(figsize=[10, 5])
plt.scatter(application_df['AMT_ANNUITY'], application_df['AMT_INCOME_TOTAL'])
plt.show()
```



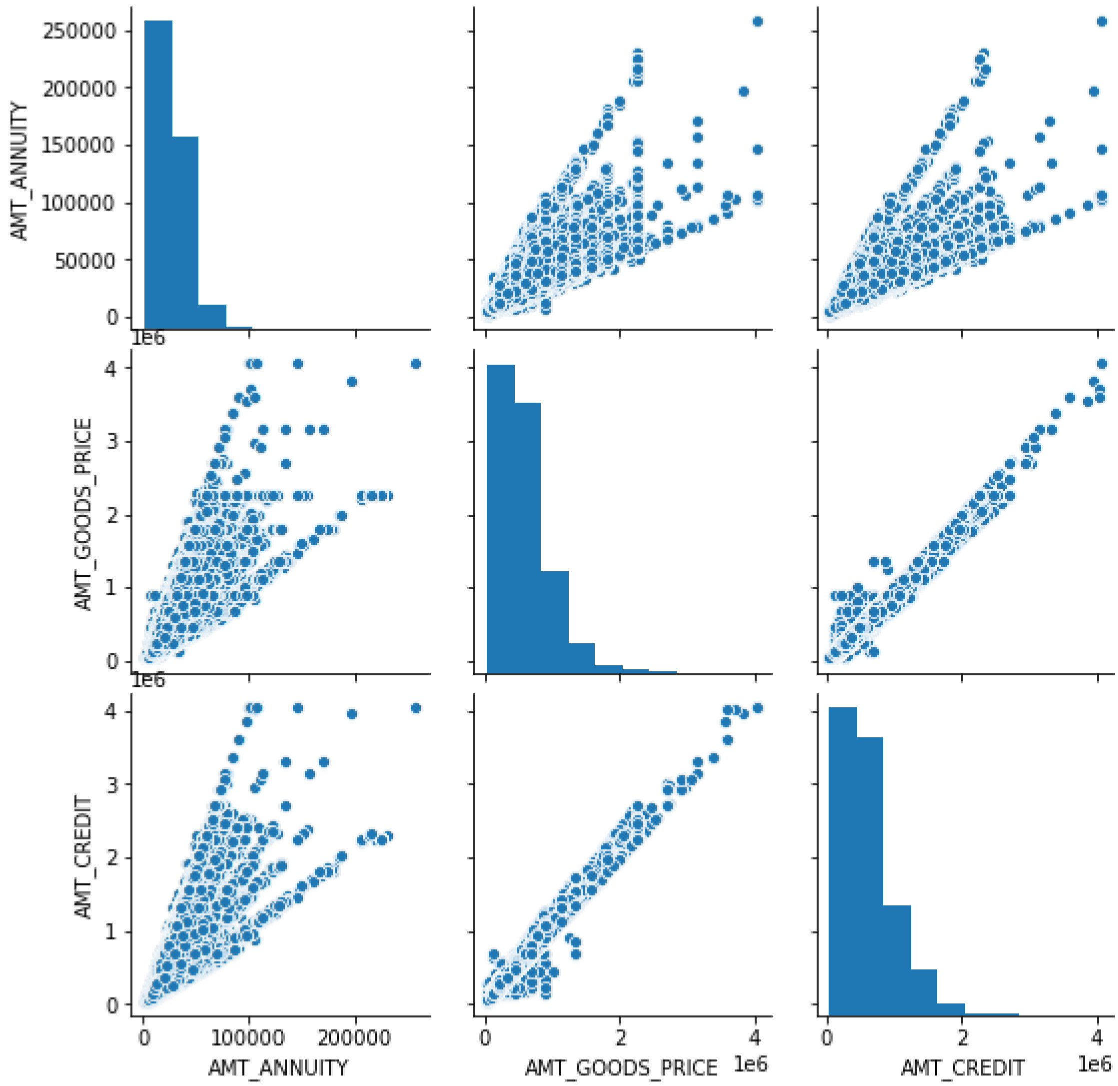
In [1962]:

```
plt.figure(figsize=[10, 5])
plt.scatter(application_df['AMT_GOODS_PRICE'], application_df['AMT_CREDIT'])
plt.show()
```



In [1963]:

```
sns.pairplot(data=application_df, vars=['AMT_ANNUITY','AMT_GOODS_PRICE','AMT_CRE
plt.show()
```



Correlation Analysis

In [1964]:

```
application_df[['AMT_ANNUITY','AMT_GOODS_PRICE','AMT_CREDIT']].corr()
```

Out[1964]:

	AMT_ANNUITY	AMT_GOODS_PRICE	AMT_CREDIT
AMT_ANNUITY	1.000000	0.775237	0.770295
AMT_GOODS_PRICE	0.775237	1.000000	0.986968
AMT_CREDIT	0.770295	0.986968	1.000000

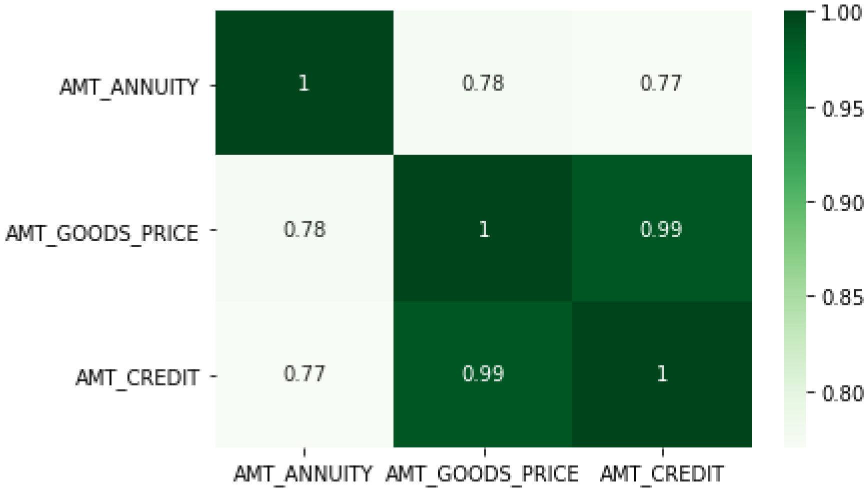
Correlation heatmap

In [1965]:

```
sns.heatmap(application_df[['AMT_ANNUITY','AMT_GOODS_PRICE','AMT_CREDIT']].corr(
```

Out[1965]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f99f1115190>

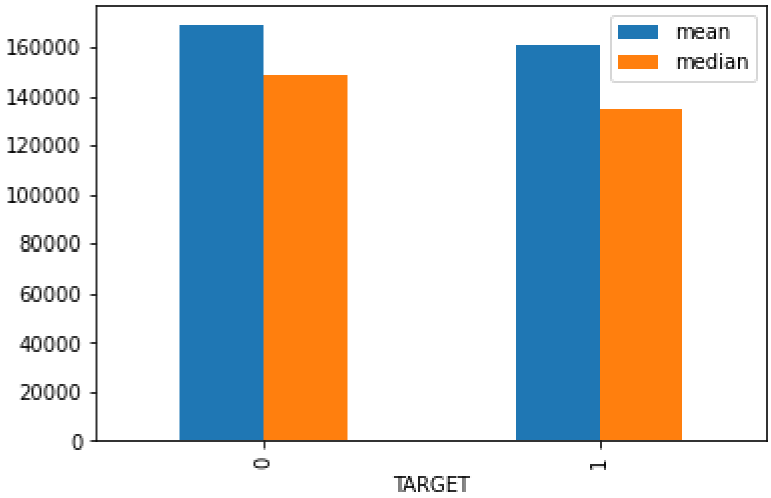


In [1966]:

```
# let's analyse TARGET vs AMT_INCOME_TOTAL
application_df.groupby('TARGET')['AMT_INCOME_TOTAL'].aggregate(["mean", "median'
```

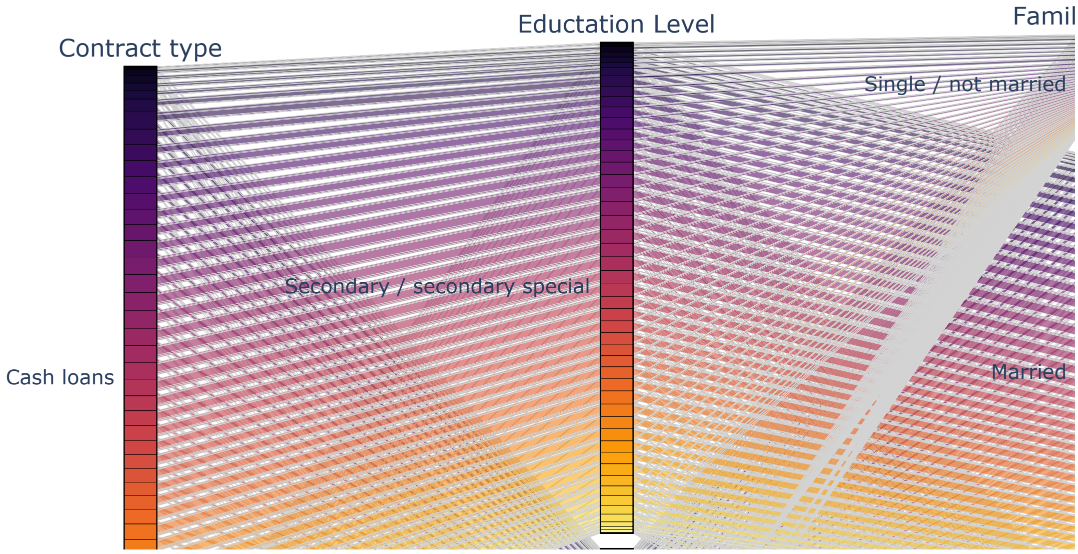
Out[1966]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f99f058ba60>



In [1967]:

```
# parallel graph
fig = px.parallel_categories(application_df, dimensions=['NAME_CONTRACT_TYPE','N
                color="AGE", color_continuous_scale=px.colors.sequential.Infernc
                labels={'NAME_CONTRACT_TYPE':'Contract type', 'NAME_EDUCATION_TY
fig.show()
```



Conclusion from above

- Most of the contract types are Cash loans
- Most of the application are from Married and Single/not married categaries
- Single/ not married with lower education are having difficulties
- Older applicants are having less difficulties and tends to apply for cash loans