Part 1 with Current Applicants

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import missingno as msno
        import os
        import warnings
        warnings.simplefilter(action='ignore')
        %matplotlib inline
```

```
In [2]: os.chdir("D:\Lohith\DA\Handson\Risk_Analytics")
```

```
In [3]: Client_Data = pd.read_csv("application_data.csv")    #Reading the Client data
        Description = pd.read_csv(r"columns_description.csv",encoding='cp1252')
```

By Default it will show only 20 columns,Since we have more then 20 column, this function is used

```
In [4]: pd.set_option("display.max_columns",None,"display.max_rows",None)
```

```
In [5]: Description.drop(columns="Unnamed: 0", axis =1, inplace = True)   # Description of our Columns
        Description[:122]
```

Out[5]:

	Table	Row	Description	Special
0	application_data	SK_ID_CURR	ID of loan in our sample	NaN
1	application_data	TARGET	Target variable (1 - client with payment diffi...	NaN
2	application_data	NAME_CONTRACT_TYPE	Identification if loan is cash or revolving	NaN
3	application_data	CODE_GENDER	Gender of the client	NaN
4	application_data	FLAG_OWN_CAR	Flag if the client owns a car	NaN
5	application_data	FLAG_OWN_REALTY	Flag if client owns a house or flat	NaN
6	application_data	CNT_CHILDREN	Number of children the client has	NaN
7	application_data	AMT_INCOME_TOTAL	Income of the client	NaN
8	application_data	AMT_CREDIT	Credit amount of the loan	NaN
9	application_data	AMT_ANNUITY	Loan annuity	NaN
10	application_data	AMT_GOODS_PRICE	For consumer loans it is the price of the good...	NaN
11	application_data	NAME_TYPE_SUITE	Who was accompanying client when he was applyi...	NaN
12	application_data	NAME_INCOME_TYPE	Clients income type (businessman, working, mat...	NaN
13	application_data	NAME_EDUCATION_TYPE	Level of highest education the client achieved	NaN
14	application_data	NAME_FAMILY_STATUS	Family status of the client	NaN
15	application_data	NAME_HOUSING_TYPE	What is the housing situation of the client (r...	NaN
16	application_data	REGION_POPULATION_RELATIVE	Normalized population of region where client l...	normalized
17	application_data	DAYS_BIRTH	Client's age in days at the time of application	time only relative to the application
18	application_data	DAYS_EMPLOYED	How many days before the application the perso...	time only relative to the application
19	application_data	DAYS_REGISTRATION	How many days before the application did clien...	time only relative to the application
20	application_data	DAYS_ID_PUBLISH	How many days before the application did clien...	time only relative to the application
21	application_data	OWN_CAR_AGE	Age of client's car	NaN
22	application_data	FLAG_MOBIL	Did client provide mobile phone (1=YES, 0=NO)	NaN
23	application_data	FLAG_EMP_PHONE	Did client provide work phone (1=YES, 0=NO)	NaN
24	application_data	FLAG_WORK_PHONE	Did client provide home phone (1=YES, 0=NO)	NaN
25	application_data	FLAG_CONT_MOBILE	Was mobile phone reachable (1=YES, 0=NO)	NaN
26	application_data	FLAG_PHONE	Did client provide home phone (1=YES, 0=NO)	NaN
27	application_data	FLAG_EMAIL	Did client provide email (1=YES, 0=NO)	NaN
28	application_data	OCCUPATION_TYPE	What kind of occupation does the client have	NaN
29	application_data	CNT_FAM_MEMBERS	How many family members does client have	NaN
30	application_data	REGION_RATING_CLIENT	Our rating of the region where client lives (1...	NaN
31	application_data	REGION_RATING_CLIENT_W_CITY	Our rating of the region where client lives wi...	NaN
32	application_data	WEEKDAY_APPR_PROCESS_START	On which day of the week did the client apply ...	NaN
33	application_data	HOUR_APPR_PROCESS_START	Approximately at what hour did the client appl...	rounded
34	application_data	REG_REGION_NOT_LIVE_REGION	Flag if client's permanent address does not ma...	NaN
35	application_data	REG_REGION_NOT_WORK_REGION	Flag if client's permanent address does not ma...	NaN
36	application_data	LIVE_REGION_NOT_WORK_REGION	Flag if client's contact address does not matc...	NaN
37	application_data	REG_CITY_NOT_LIVE_CITY	Flag if client's permanent address does not ma...	NaN
38	application_data	REG_CITY_NOT_WORK_CITY	Flag if client's permanent address does not ma...	NaN
39	application_data	LIVE_CITY_NOT_WORK_CITY	Flag if client's contact address does not matc...	NaN
40	application_data	ORGANIZATION_TYPE	Type of organization where client works	NaN
41	application_data	EXT_SOURCE_1	Normalized score from external data source	normalized
42	application_data	EXT_SOURCE_2	Normalized score from external data source	normalized
43	application_data	EXT_SOURCE_3	Normalized score from external data source	normalized
44	application_data	APARTMENTS_AVG	Normalized information about building where th...	normalized
45	application_data	BASEMENTAREA_AVG	Normalized information about building where th...	normalized
46	application_data	YEARS_BEGINEXPLUATATION_AVG	Normalized information about building where th...	normalized
47	application_data	YEARS_BUILD_AVG	Normalized information about building where th...	normalized
48	application_data	COMMONAREA_AVG	Normalized information about building where th...	normalized
49	application_data	ELEVATORS_AVG	Normalized information about building where th...	normalized
50	application_data	ENTRANCES_AVG	Normalized information about building where th...	normalized
51	application_data	FLOORSMAX_AVG	Normalized information about building where th...	normalized
52	application_data	FLOORSMIN_AVG	Normalized information about building where th...	normalized
53	application_data	LANDAREA_AVG	Normalized information about building where th...	normalized
54	application_data	LIVINGAPARTMENTS_AVG	Normalized information about building where th...	normalized
55	application_data	LIVINGAREA_AVG	Normalized information about building where th...	normalized
56	application_data	NONLIVINGAPARTMENTS_AVG	Normalized information about building where th...	normalized
57	application_data	NONLIVINGAREA_AVG	Normalized information about building where th...	normalized
58	application_data	APARTMENTS_MODE	Normalized information about building where th...	normalized
59	application_data	BASEMENTAREA_MODE	Normalized information about building where th...	normalized
60	application_data	YEARS_BEGINEXPLUATATION_MODE	Normalized information about building where th...	normalized
61	application_data	YEARS_BUILD_MODE	Normalized information about building where th...	normalized
62	application_data	COMMONAREA_MODE	Normalized information about building where th...	normalized
63	application_data	ELEVATORS_MODE	Normalized information about building where th...	normalized
64	application_data	ENTRANCES_MODE	Normalized information about building where th...	normalized
65	application_data	FLOORSMAX_MODE	Normalized information about building where th...	normalized
66	application_data	FLOORSMIN_MODE	Normalized information about building where th...	normalized
67	application_data	LANDAREA_MODE	Normalized information about building where th...	normalized
68	application_data	LIVINGAPARTMENTS_MODE	Normalized information about building where th...	normalized
69	application_data	LIVINGAREA_MODE	Normalized information about building where th...	normalized
70	application_data	NONLIVINGAPARTMENTS_MODE	Normalized information about building where th...	normalized
71	application_data	NONLIVINGAREA_MODE	Normalized information about building where th...	normalized
72	application_data	APARTMENTS_MEDI	Normalized information about building where th...	normalized
73	application_data	BASEMENTAREA_MEDI	Normalized information about building where th...	normalized
74	application_data	YEARS_BEGINEXPLUATATION_MEDI	Normalized information about building where th...	normalized
75	application_data	YEARS_BUILD_MEDI	Normalized information about building where th...	normalized
76	application_data	COMMONAREA_MEDI	Normalized information about building where th...	normalized
77	application_data	ELEVATORS_MEDI	Normalized information about building where th...	normalized
78	application_data	ENTRANCES_MEDI	Normalized information about building where th...	normalized
79	application_data	FLOORSMAX_MEDI	Normalized information about building where th...	normalized
80	application_data	FLOORSMIN_MEDI	Normalized information about building where th...	normalized
81	application_data	LANDAREA_MEDI	Normalized information about building where th...	normalized
82	application_data	LIVINGAPARTMENTS_MEDI	Normalized information about building where th...	normalized
83	application_data	LIVINGAREA_MEDI	Normalized information about building where th...	normalized
84	application_data	NONLIVINGAPARTMENTS_MEDI	Normalized information about building where th...	normalized
85	application_data	NONLIVINGAREA_MEDI	Normalized information about building where th...	normalized
86	application_data	FONDKAPREMONT_MODE	Normalized information about building where th...	normalized
87	application_data	HOUSETYPE_MODE	Normalized information about building where th...	normalized
88	application_data	TOTALAREA_MODE	Normalized information about building where th...	normalized
89	application_data	WALLSMATERIAL_MODE	Normalized information about building where th...	normalized
90	application_data	EMERGENCYSTATE_MODE	Normalized information about building where th...	normalized
91	application_data	OBS_30_CNT_SOCIAL_CIRCLE	How many observation of client's social surrou...	NaN
92	application_data	DEF_30_CNT_SOCIAL_CIRCLE	How many observation of client's social surrou...	NaN
93	application_data	OBS_60_CNT_SOCIAL_CIRCLE	How many observation of client's social surrou...	NaN
94	application_data	DEF_60_CNT_SOCIAL_CIRCLE	How many observation of client's social surrou...	NaN
95	application_data	DAYS_LAST_PHONE_CHANGE	How many days before application did client ch...	NaN
96	application_data	FLAG_DOCUMENT_2	Did client provide document 2	NaN
97	application_data	FLAG_DOCUMENT_3	Did client provide document 3	NaN
98	application_data	FLAG_DOCUMENT_4	Did client provide document 4	NaN
99	application_data	FLAG_DOCUMENT_5	Did client provide document 5	NaN
100	application_data	FLAG_DOCUMENT_6	Did client provide document 6	NaN
101	application_data	FLAG_DOCUMENT_7	Did client provide document 7	NaN
102	application_data	FLAG_DOCUMENT_8	Did client provide document 8	NaN
103	application_data	FLAG_DOCUMENT_9	Did client provide document 9	NaN
104	application_data	FLAG_DOCUMENT_10	Did client provide document 10	NaN
105	application_data	FLAG_DOCUMENT_11	Did client provide document 11	NaN
106	application_data	FLAG_DOCUMENT_12	Did client provide document 12	NaN
107	application_data	FLAG_DOCUMENT_13	Did client provide document 13	NaN
108	application_data	FLAG_DOCUMENT_14	Did client provide document 14	NaN
109	application_data	FLAG_DOCUMENT_15	Did client provide document 15	NaN
110	application_data	FLAG_DOCUMENT_16	Did client provide document 16	NaN
111	application_data	FLAG_DOCUMENT_17	Did client provide document 17	NaN
112	application_data	FLAG_DOCUMENT_18	Did client provide document 18	NaN
113	application_data	FLAG_DOCUMENT_19	Did client provide document 19	NaN
114	application_data	FLAG_DOCUMENT_20	Did client provide document 20	NaN
115	application_data	FLAG_DOCUMENT_21	Did client provide document 21	NaN
116	application_data	AMT_REQ_CREDIT_BUREAU_HOUR	Number of enquiries to Credit Bureau about the...	NaN
117	application_data	AMT_REQ_CREDIT_BUREAU_DAY	Number of enquiries to Credit Bureau about the...	NaN
118	application_data	AMT_REQ_CREDIT_BUREAU_WEEK	Number of enquiries to Credit Bureau about the...	NaN
119	application_data	AMT_REQ_CREDIT_BUREAU_MON	Number of enquiries to Credit Bureau about the...	NaN
120	application_data	AMT_REQ_CREDIT_BUREAU_QRT	Number of enquiries to Credit Bureau about the...	NaN
121	application_data	AMT_REQ_CREDIT_BUREAU_YEAR	Number of enquiries to Credit Bureau about the...	NaN

```
In [6]: Client_Data.head(5) # Checking the top 5 records
```

Out[6]:

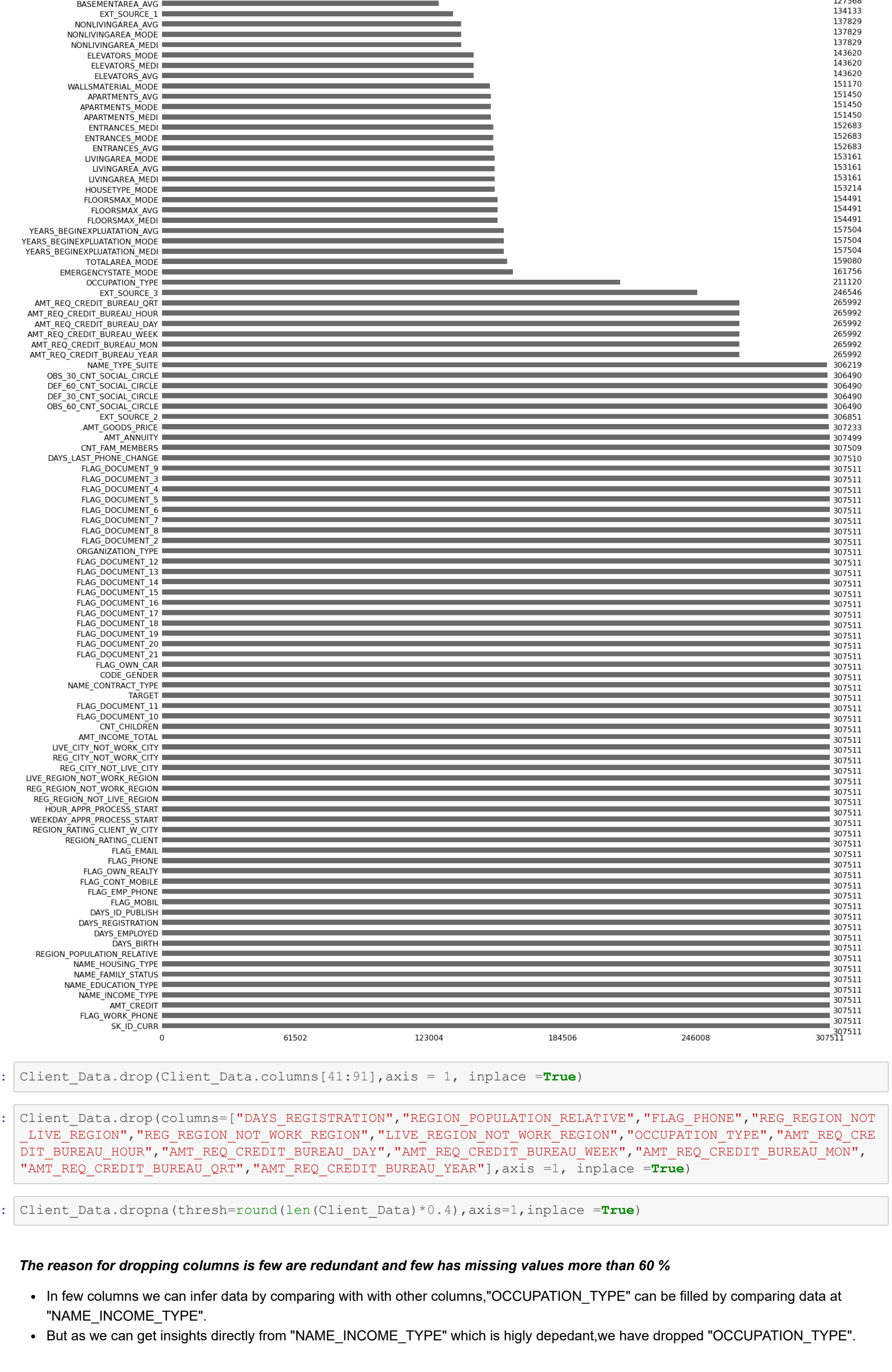
	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCO
0	100002	1	Cash loans	M	N	Y	0	
1	100003	0	Cash loans	F	N	N	0	
2	100004	0	Revolving loans	M	Y	Y	0	
3	100006	0	Cash loans	F	N	Y	0	
4	100007	0	Cash loans	M	N	Y	0	

```
In [7]: print("Client_Data: ",Client_Data.shape)
```

```
Client_Data:  (307511, 122)
```

```
In [8]: msno.bar(Client_Data,sort = "descending")
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x2808006aa30>



```
In [9]: Client_Data.drop(Client_Data.columns[41:91],axis = 1, inplace =True)
```

```
In [10]: Client_Data.drop(columns=["DAYS_REGISTRATION","REGION_POPULATION_RELATIVE","FLAG_PHONE","REG_REGION_NOT_LIVE_REGION","REG_REGION_NOT_WORK_REGION","LIVE_REGION_NOT_WORK_REGION","OCCUPATION_TYPE","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"],axis =1, inplace =True)
```

```
In [11]: Client_Data.dropna(thresh=round(len(Client_Data)*0.4),axis=1,inplace =True)
```

The reason for dropping columns is few are redundant and few has missing values more than 60 %

- In few columns we can infer data by comparing with with other columns,"OCCUPATION_TYPE" can be filled by comparing data at "NAME_INCOME_TYPE".
- But as we can get insights directly from "NAME_INCOME_TYPE" which is higly dependant,we have dropped "OCCUPATION_TYPE".

```
In [12]: print("Client_Data: ",Client_Data.shape)
```

```
Client_Data:  (307511, 58)
```

```
In [13]: Client_Data.isnull().sum()
```

Out[13]:

```
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
DAYS_BIRTH                        0
DAYS_EMPLOYED                     0
DAYS_ID_PUBLISH                   0
FLAG_MOBIL                        0
FLAG_EMP_PHONE                    0
FLAG_WORK_PHONE                   0
FLAG_CONT_MOBILE                  0
FLAG_EMAIL                        0
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_CITY_NOT_LIVE_CITY            0
REG_CITY_NOT_WORK_CITY            0
LIVE_CITY_NOT_WORK_CITY           0
ORGANIZATION_TYPE                 0
OBS_30_CNT_SOCIAL_CIRCLE       1021
DEF_30_CNT_SOCIAL_CIRCLE       1021
OBS_60_CNT_SOCIAL_CIRCLE       1021
DEF_60_CNT_SOCIAL_CIRCLE       1021
DAYS_LAST_PHONE_CHANGE            1
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
FLAG_DOCUMENT_20                  0
FLAG_DOCUMENT_21                  0
dtype: int64
```

```
In [14]: Client_Data.dropna(how ="any",inplace =True)
```

```
In [15]: Client_Data.shape
```

Out[15]: (305185, 58)

```
In [16]: Client_Data.dtypes
```

Out[16]:

```
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
DAYS_BIRTH                       int64
DAYS_EMPLOYED                    int64
DAYS_ID_PUBLISH                  int64
FLAG_MOBIL                       int64
FLAG_EMP_PHONE                   int64
FLAG_WORK_PHONE                  int64
FLAG_CONT_MOBILE                 int64
FLAG_EMAIL                       int64
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_CITY_NOT_LIVE_CITY           int64
REG_CITY_NOT_WORK_CITY           int64
LIVE_CITY_NOT_WORK_CITY          int64
ORGANIZATION_TYPE               object
OBS_30_CNT_SOCIAL_CIRCLE       float64
DEF_30_CNT_SOCIAL_CIRCLE       float64
OBS_60_CNT_SOCIAL_CIRCLE       float64
DEF_60_CNT_SOCIAL_CIRCLE       float64
DAYS_LAST_PHONE_CHANGE         float64
FLAG_DOCUMENT_2                  int64
FLAG_DOCUMENT_3                  int64
FLAG_DOCUMENT_4                  int64
FLAG_DOCUMENT_5                  int64
FLAG_DOCUMENT_6                  int64
FLAG_DOCUMENT_7                  int64
FLAG_DOCUMENT_8                  int64
FLAG_DOCUMENT_9                  int64
FLAG_DOCUMENT_10                 int64
FLAG_DOCUMENT_11                 int64
FLAG_DOCUMENT_12                 int64
FLAG_DOCUMENT_13                 int64
FLAG_DOCUMENT_14                 int64
FLAG_DOCUMENT_15                 int64
FLAG_DOCUMENT_16                 int64
FLAG_DOCUMENT_17                 int64
FLAG_DOCUMENT_18                 int64
FLAG_DOCUMENT_19                 int64
FLAG_DOCUMENT_20                 int64
FLAG_DOCUMENT_21                 int64
dtype: object
```

Converting the below columns to Absolute values

```
In [17]: Client_Data["CODE_GENDER"]=Client_Data["CODE_GENDER"].replace("XNA","Others")
         Client_Data["DAYS_LAST_PHONE_CHANGE"] = Client_Data["DAYS_LAST_PHONE_CHANGE"].abs()
         Client_Data["DAYS_BIRTH"] = Client_Data["DAYS_BIRTH"].abs()
         Client_Data["DAYS_EMPLOYED"] = Client_Data["DAYS_EMPLOYED"].abs()
         Client_Data["DAYS_ID_PUBLISH"] = Client_Data["DAYS_ID_PUBLISH"].abs()
         Client_Data["AMT_INCOME_TOTAL"] = Client_Data["AMT_INCOME_TOTAL"].abs()
         Client_Data["AMT_CREDIT"] = Client_Data["AMT_CREDIT"].abs()
         Client_Data["AMT_ANNUITY"] = Client_Data["AMT_ANNUITY"].abs()
         Client_Data["AMT_GOODS_PRICE"] = Client_Data["AMT_GOODS_PRICE"].abs()
```

```
In [18]: Client_Data[["CODE_GENDER","DAYS_LAST_PHONE_CHANGE","DAYS_BIRTH","DAYS_EMPLOYED","DAYS_ID_PUBLISH","AMT_INCOME_TOTAL"]].head(5)
```

Out[18]:

	CODE_GENDER	DAYS_LAST_PHONE_CHANGE	DAYS_BIRTH	DAYS_EMPLOYED	DAYS_ID_PUBLISH	AMT_INCOME_TOTAL
0	M	1134.0	9461	637	2120	202500.0
1	F	828.0	16765	1188	291	270000.0
2	M	815.0	19046	225	2531	67500.0
3	F	617.0	19005	3039	2437	135000.0
4	M	1106.0	19932	3038	3458	121500.0

Converting number of Days to Years

```
In [19]: Client_Data.insert(loc=12,column="Age",value=(Client_Data[["DAYS_BIRTH"]]/365.25).astype(int))
         Client_Data["DAYS_EMPLOYED"] = round(Client_Data[["DAYS_EMPLOYED"]]/365.25,1)
         Client_Data["DAYS_ID_PUBLISH"] = round(Client_Data[["DAYS_ID_PUBLISH"]]/365.25,1)
         Client_Data.drop("DAYS_BIRTH",axis =1,inplace= True)
```

```
In [20]: Client_Data[["Age","DAYS_EMPLOYED","DAYS_ID_PUBLISH"]].describe()
```

Out[20]:

	Age	DAYS_EMPLOYED	DAYS_ID_PUBLISH
count	305185.000000	305185.000000	305185.000000
mean	43.417825	185.592190	8.198860
std	11.941714	381.919921	4.132549
min	20.000000	0.000000	0.000000
25%	34.000000	2.600000	4.700000
50%	43.000000	6.100000	8.900000
75%	53.000000	15.700000	11.800000
max	69.000000	1000.000000	19.700000

- If we take 70 years as a standard bar

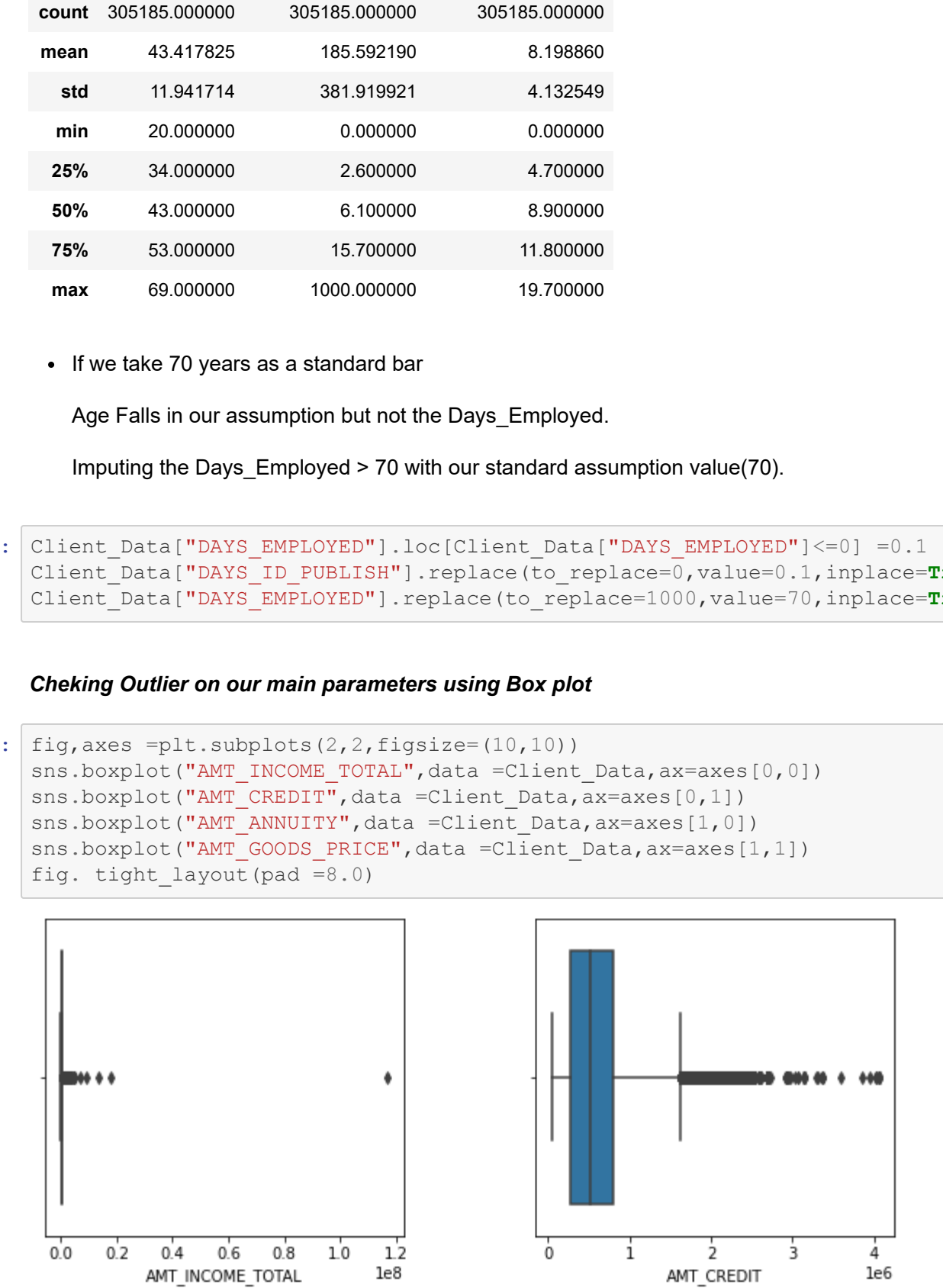
  Age Falls in our assumption but not the Days_Employed.

  Imputing the Days_Employed > 70 with our standard assumption value(70).

```
In [21]: Client_Data["DAYS_EMPLOYED"].loc[Client_Data["DAYS_EMPLOYED"]<=0] =0.1
         Client_Data["DAYS_ID_PUBLISH"].replace(to_replace=0,value=0.1,inplace=True)
         Client_Data["DAYS_EMPLOYED"].replace(to_replace=1000,value=70,inplace=True)
```

Cheking Outlier on our main parameters using Box plot

```
In [22]: fig,axes =plt.subplots(2,2,figsize=(10,10))
         sns.boxplot("AMT_INCOME_TOTAL",data =Client_Data,ax=axes[0,0])
         sns.boxplot("AMT_CREDIT",data =Client_Data,ax=axes[0,1])
         sns.boxplot("AMT_ANNUITY",data =Client_Data,ax=axes[1,0])
         sns.boxplot("AMT_GOODS_PRICE",data =Client_Data,ax=axes[1,1])
         fig. tight_layout(pad =8.0)
```




```
In [23]: Client_Data[["AMT_INCOME_TOTAL","AMT_CREDIT"]].describe()
```

Out[23]:

	AMT_INCOME_TOTAL	AMT_CREDIT
count	3.051850e+05	3.051850e+05
mean	1.686140e+05	5.992816e+05
std	2.376652e+05	4.020170e+05
min	2.565000e+04	4.500000e+04
25%	1.125000e+05	2.700000e+05
50%	1.458000e+05	5.147775e+05
75%	2.025000e+05	8.086500e+05
max	1.170000e+08	4.050000e+06

```
In [24]: Client_Data.loc[Client_Data["AMT_INCOME_TOTAL"]>6000000,["TARGET","AMT_INCOME_TOTAL","AMT_CREDIT"]].head(10)
```

Out[24]:

	TARGET	AMT_INCOME_TOTAL	AMT_CREDIT
12840	1	117000000.0	562491.0
77768	0	9000000.0	1431531.0
131127	0	6750000.0	790830.0
203693	0	18000090.0	675000.0
246858	0	13500000.0	1400503.5

- Income Data has an outlier between 6 and 120 Millions,But the "Amt_Credit"(Loan amount provided) has 1Million Maximum and "Amt_Annuity" has 2.5 Lakhs maximum.
- As our Anlaysis gets effeted with values between 6 and 120 millions, it is better to drop rather to impute because there may be chances of person earning such huge amount but probability getting ~5% of income as loan is very low.

```
In [25]: Client_Data.drop(Client_Data[Client_Data["AMT_INCOME_TOTAL"]>6000000].index,inplace=True)
```

```
In [26]: Client_Data.loc[Client_Data["AMT_INCOME_TOTAL"]>6000000,["TARGET","AMT_INCOME_TOTAL","AMT_CREDIT"]].head(10)
```

Out[26]:

TARGET	AMT_INCOME_TOTAL	AMT_CREDIT

```
In [27]: Client_Data.describe()
```

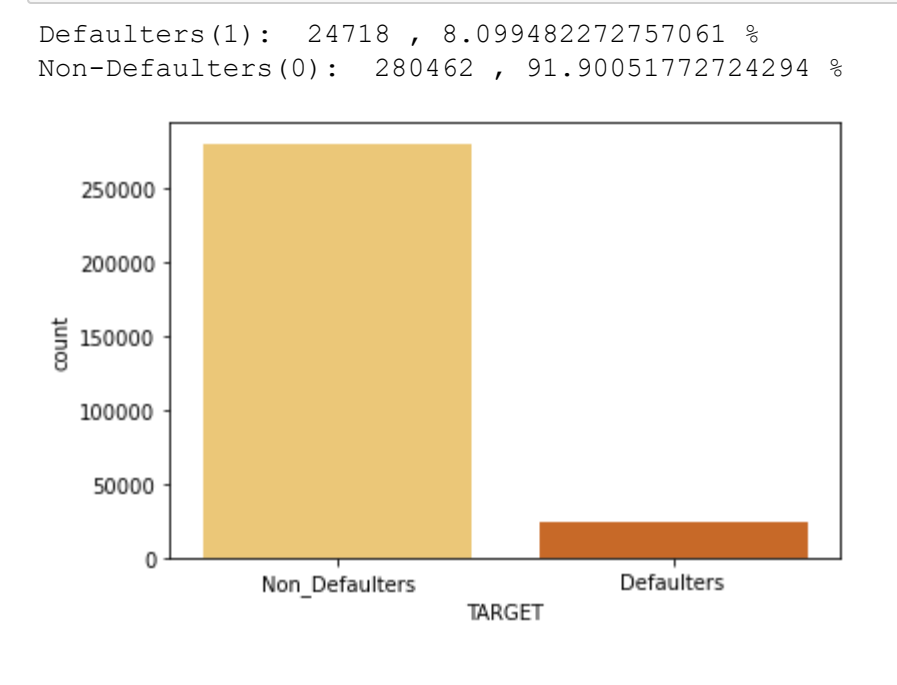
Out[27]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	
count	305180.000000	305180.000000	305180.000000	3.051800e+05	3.051800e+05	305180.000000	3.051800e+05	305180.0
mean	278164.564955	0.080995	0.417147	1.680786e+05	5.992755e+05	27135.854142	5.382210e+05	43.4
std	102784.567591	0.272828	0.722266	9.867711e+04	4.020147e+05	14473.432700	3.689252e+05	11.9
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1615.500000	4.050000e+04	20.0
25%	189128.750000	0.000000	0.000000	1.125000e+05	2.700000e+05	16573.500000	2.385000e+05	34.0
50%	278189.500000	0.000000	0.000000	1.455008e+05	5.147775e+05	24939.000000	4.500000e+05	43.0
75%	367120.250000	0.000000	1.000000	2.025000e+05	8.086500e+05	34614.000000	6.795000e+05	53.0
max	456255.000000	1.000000	19.000000	4.500000e+06	4.050000e+06	258025.500000	4.050000e+06	69.0

Univariate Analysis

```
In [28]: sns.countplot(Client_Data["TARGET"],palette="YlOrBr").set(xticklabels=["Non_Defaulters","Defaulters"])
         print("Defaulters(1): ",Client_Data["TARGET"].value_counts()[1],",",(Client_Data["TARGET"].value_counts()[1]/len(Client_Data["TARGET"]))*100,"%")
         print("Non-Defaulters(0): ",Client_Data["TARGET"].value_counts()[0],",",(Client_Data["TARGET"].value_counts()[0]/len(Client_Data["TARGET"]))*100,"%")
```

```
Defaulters(1):  24718 , 8.099482272757061 %
Non-Defaulters(0):  280462 , 91.90051772724294 %
```

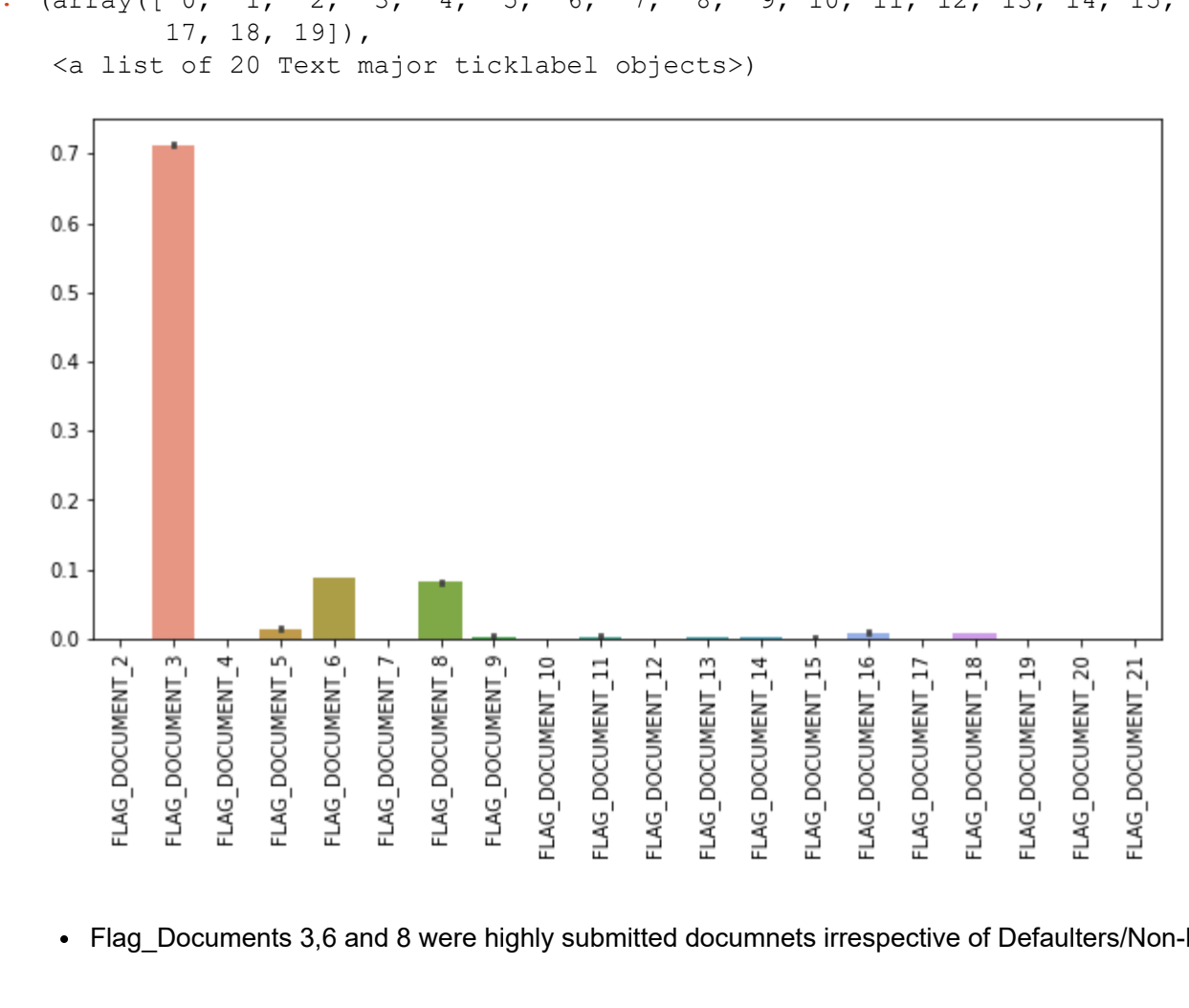


- Only 8% were Defaulters, we have to focus on this category

Documents Status

```
In [29]: Document_Status =Client_Data[["FLAG_DOCUMENT_2","FLAG_DOCUMENT_3","FLAG_DOCUMENT_4","FLAG_DOCUMENT_5","FLAG_DOCUMENT_6","FLAG_DOCUMENT_7","FLAG_DOCUMENT_8","FLAG_DOCUMENT_9","FLAG_DOCUMENT_10","FLAG_DOCUMENT_11","FLAG_DOCUMENT_12","FLAG_DOCUMENT_13","FLAG_DOCUMENT_14","FLAG_DOCUMENT_15","FLAG_DOCUMENT_16","FLAG_DOCUMENT_17","FLAG_DOCUMENT_18","FLAG_DOCUMENT_19","FLAG_DOCUMENT_20","FLAG_DOCUMENT_21"]]
         #Document_Status.head(10)
         plt.figure(figsize = (10,5))
         sns.barplot(data = Document_Status)
         plt.xticks(rotation =90)
```

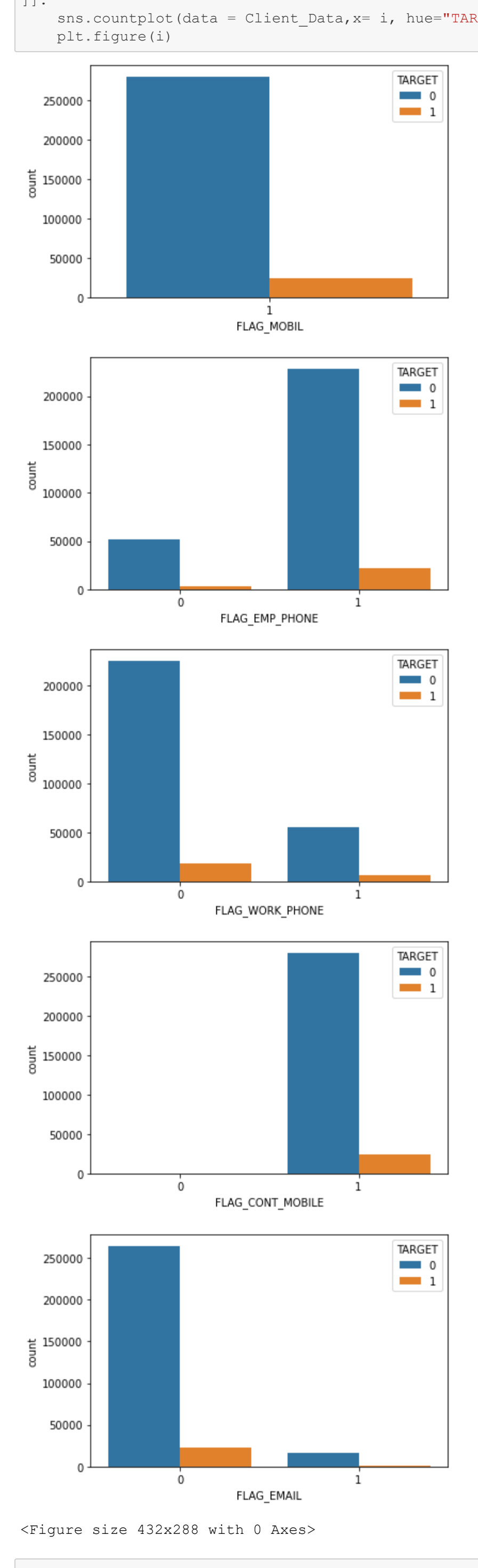
```
Out[29]: (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
                 17, 18, 19]),
          <a list of 20 Text major ticklabel objects>)
```



- Flag_Documents 3,6 and 8 were highly submitted documnets irrespective of Defaulters/Non-Defaulters

Contact Status

```
In [30]: for i in Client_Data[["FLAG_MOBIL","FLAG_EMP_PHONE","FLAG_WORK_PHONE","FLAG_CONT_MOBILE","FLAG_EMAIL"]]:
             sns.countplot(data = Client_Data,x= i, hue="TARGET")
             plt.figure(i)
```




```
<Figure size 432x288 with 0 Axes>
```

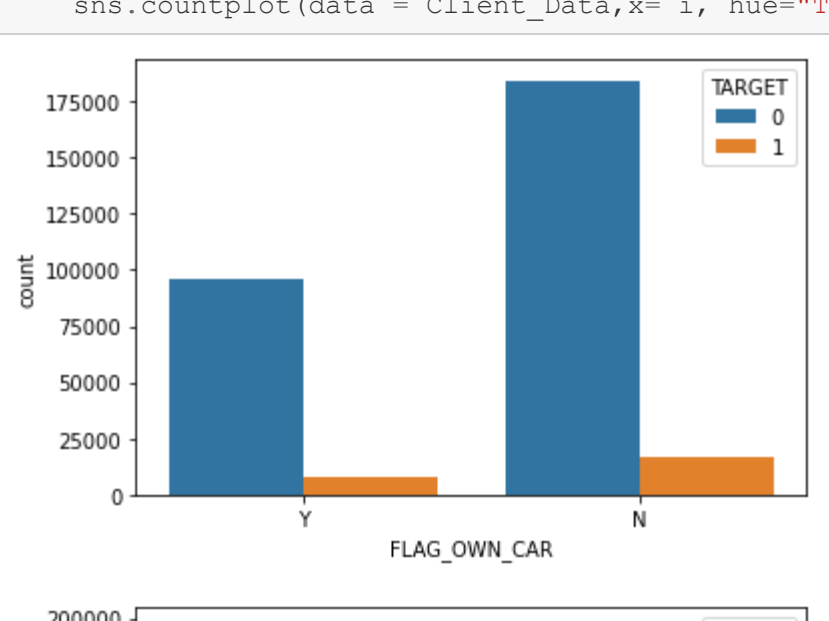
```
In [31]: for i in Client_Data[["FLAG_MOBIL","FLAG_EMP_PHONE","FLAG_WORK_PHONE","FLAG_CONT_MOBILE","FLAG_EMAIL"]]:
             defaulters = Client_Data[(Client_Data[i]==0) & (Client_Data["TARGET"] ==1)]
             Total = len(Client_Data[Client_Data["TARGET"] ==1])
             print(i,":   ",round((len(defaulters)/Total)*100,3))
```

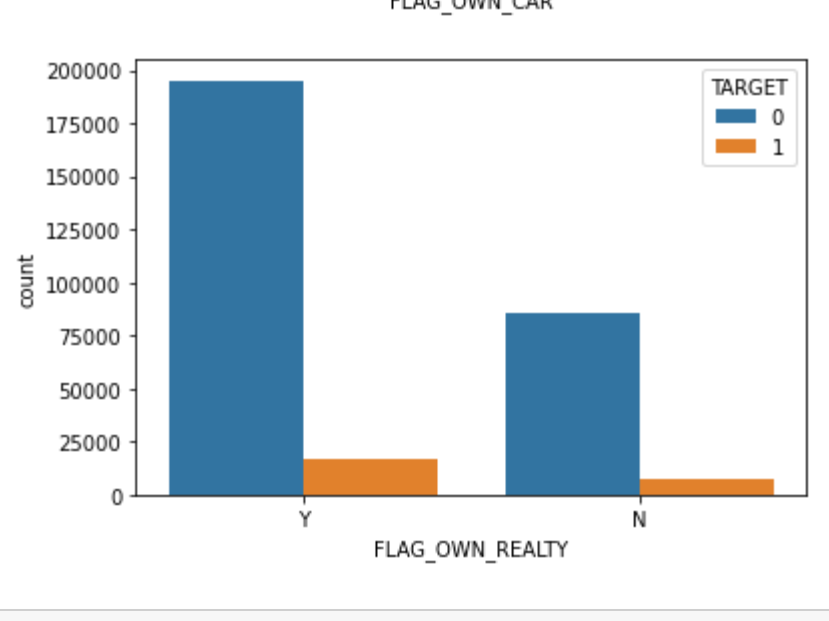
```
FLAG_MOBIL :   0.0
FLAG_EMP_PHONE :   12.036
FLAG_WORK_PHONE :   76.208
FLAG_CONT_MOBILE :   0.182
FLAG_EMAIL :   94.453
```

- 76% home phone data Missing with Defaulters.
- Email Category was higly neglected in all Applications.

Asset Status

```
In [32]: for i in Client_Data[["FLAG_OWN_CAR","FLAG_OWN_REALTY"]]:
    plt.figure(i)
    sns.countplot(data = Client_Data,x= i, hue="TARGET",order=("Y","N"))
```





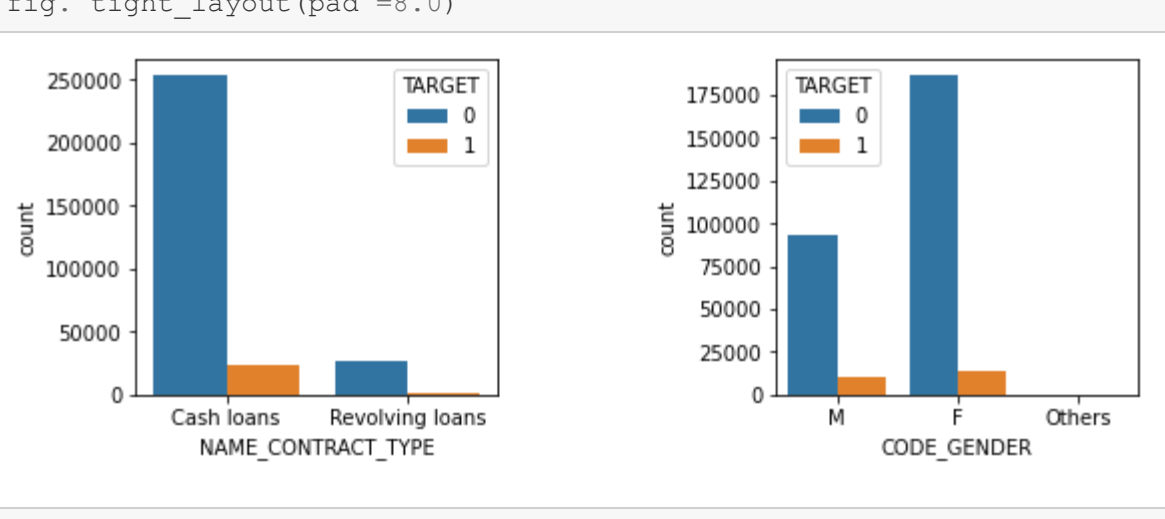

```
In [33]: for i in Client_Data[["FLAG_OWN_CAR","FLAG_OWN_REALTY"]]:
    defaulters1 = Client_Data[(Client_Data[i]=="Y") & (Client_Data["TARGET"] ==1)]
    defaulters2 = Client_Data[(Client_Data[i]=="N") & (Client_Data["TARGET"] ==1)]
    Total = len(Client_Data[Client_Data["TARGET"] ==1])
    print(i,"(H/C)",":  ",round((len(defaulters1)/Total)*100,3))
    print(i,"(NH/NC)",":  ",round((len(defaulters2)/Total)*100,3))

FLAG_OWN_CAR (H/C) :   30.532
FLAG_OWN_CAR (NH/NC) :   69.468
FLAG_OWN_REALTY (H/C) :   68.452
FLAG_OWN_REALTY (NH/NC) :   31.548
```

- 70% of Defaulters dont have own car but had Own house which is a Risk factor

Family Details

```
In [34]: Fam_Details = Client_Data[["NAME_CONTRACT_TYPE","CODE_GENDER","CNT_FAM_MEMBERS","CNT_CHILDREN","NAME_INCOME_TYPE","NAME_EDUCATION_TYPE","NAME_FAMILY_STATUS","NAME_TYPE_SUITE","NAME_HOUSING_TYPE"]]
```

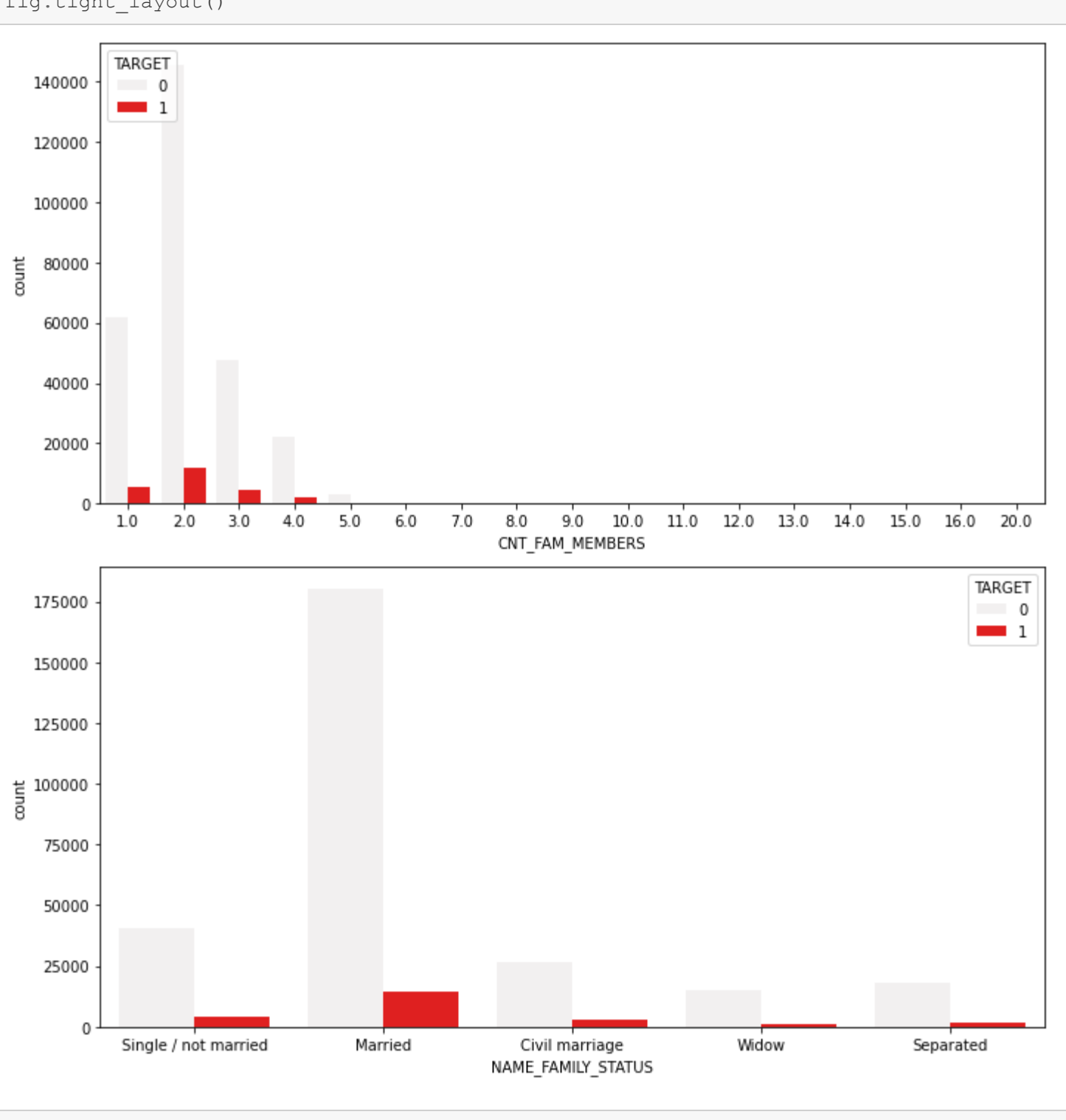
```
In [35]: fig,axes =plt.subplots(1,2,figsize=(10,5),)
sns.countplot(x="NAME_CONTRACT_TYPE",data =Client_Data,hue="TARGET",ax=axes[0])
sns.countplot(x="CODE_GENDER",hue="TARGET",data =Client_Data,ax=axes[1])
fig. tight_layout(pad =8.0)
```




```
In [36]: for i in Fam_Details["NAME_CONTRACT_TYPE"].unique():
    defaulters1 = Fam_Details[(Fam_Details["NAME_CONTRACT_TYPE"] == i) & (Client_Data["TARGET"] ==1)]
    Total = len(Client_Data[Client_Data["TARGET"] ==1])
    print(i,":  ",(len(defaulters1)/Total)*100)

Cash loans :   93.7454486608949
Revolving loans :   6.254551339105105
```

- Highest Applicants and Defaulters are Female and of Cash loans
- But when compared to total Male Applicants, the deafulters were quite high in Male

```
In [37]: fig,axes =plt.subplots(2,1,figsize=(10,10))
sns.countplot(data =Client_Data,x="CNT_FAM_MEMBERS",hue="TARGET",ax=axes[0],color="red")
sns.countplot(data =Client_Data,x="NAME_FAMILY_STATUS",hue="TARGET",ax=axes[1],color="red")
fig.tight_layout()
```

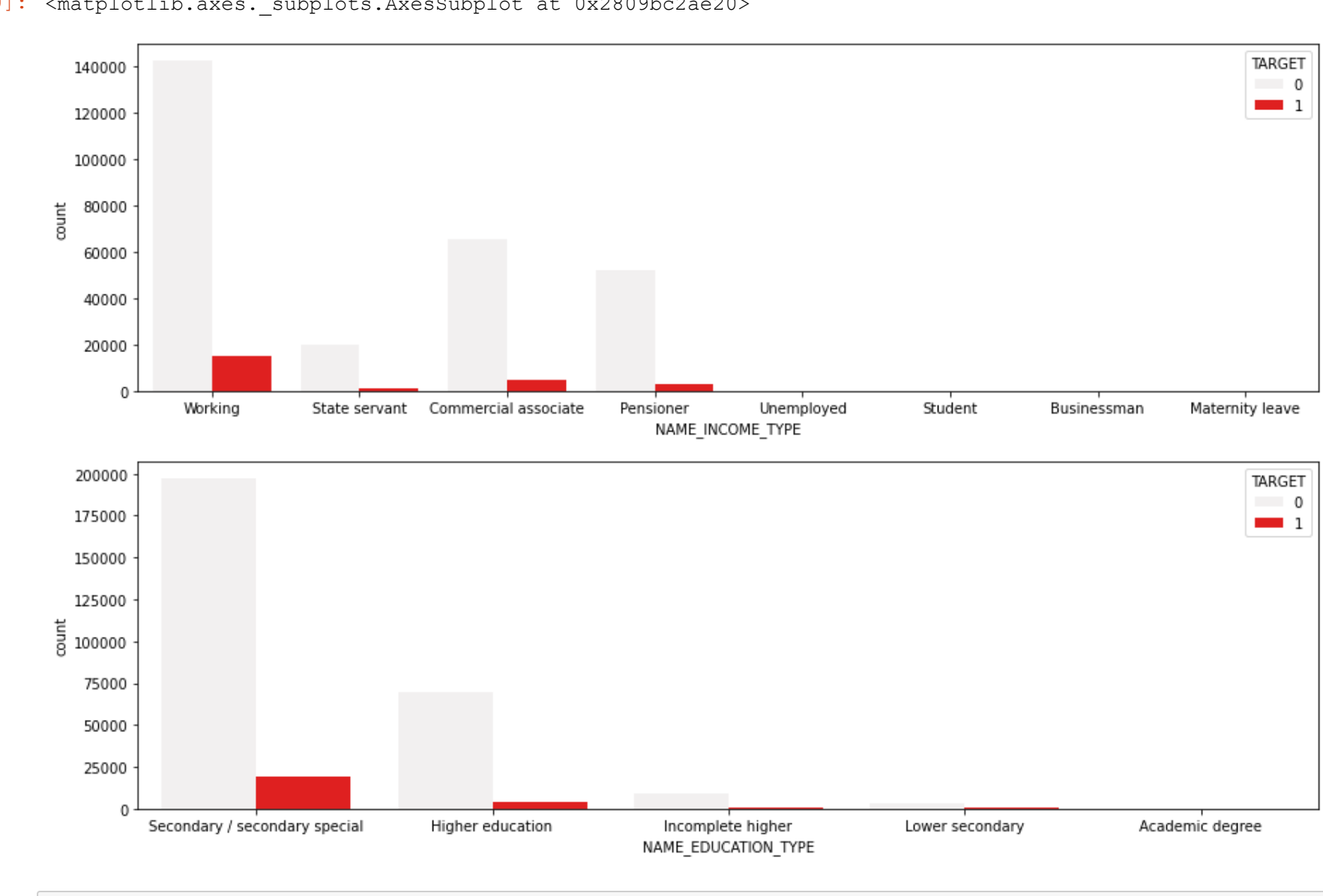



```
In [38]: for i in Fam_Details["NAME_FAMILY_STATUS"].unique():
    defaulters1 = Fam_Details[(Fam_Details["NAME_FAMILY_STATUS"] == i) & (Client_Data["TARGET"] ==1)]
    Total = len(Client_Data[Client_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))

Single / not married :   17.967
Married :   59.815
Civil marriage :   11.922
Widow :   3.779
Separated :   6.518
```

```
In [39]: for i in Fam_Details["CNT_FAM_MEMBERS"].unique():
    defaulters1 = Fam_Details[(Fam_Details["CNT_FAM_MEMBERS"] == i) & (Client_Data["TARGET"] ==1)]
    Total = len(Client_Data[Client_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))

1.0 :   22.87
2.0 :   48.357
3.0 :   18.557
4.0 :   8.609
5.0 :   1.323
6.0 :   0.223
9.0 :   0.0
7.0 :   0.024
8.0 :   0.024
10.0 :   0.004
13.0 :   0.004
14.0 :   0.0
12.0 :   0.0
20.0 :   0.0
15.0 :   0.0
16.0 :   0.0
11.0 :   0.004
```

- Eventhough highest Defaulters lie around Married, but risk factor is high with ~5 - ~10% Single,Civil marriages and Seperated
- Highest Defaulters with Family members less than 5

```
In [40]: fig,axes =plt.subplots(2,1,figsize=(15,10))
sns.countplot(data =Client_Data,x="NAME_INCOME_TYPE",hue="TARGET",ax=axes[0],color="red")
sns.countplot(data =Client_Data,x="NAME_EDUCATION_TYPE",hue="TARGET",ax=axes[1],color="red")

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x2809bc2ae20>
```

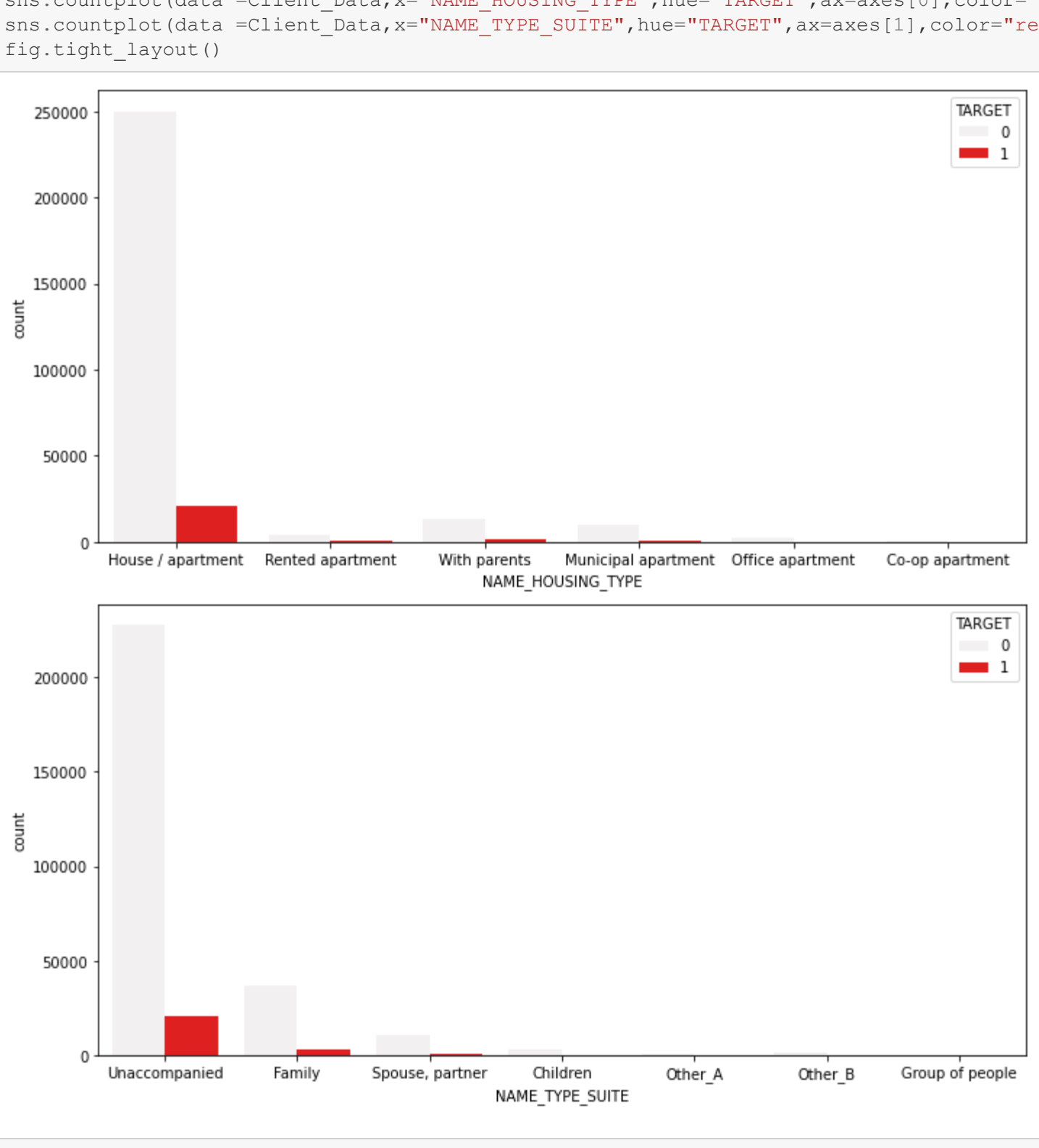



```
In [41]: for i in Fam_Details["NAME_EDUCATION_TYPE"].unique():
    defaulters1 = Fam_Details[(Fam_Details["NAME_EDUCATION_TYPE"] == i) & (Client_Data["TARGET"] ==1)]
    Total = len(Client_Data[Client_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))

Secondary / secondary special :   78.684
Higher education :   16.118
Incomplete higher :   3.508
Lower secondary :   1.679
Academic degree :   0.012
```

```
In [42]: for i in Fam_Details["NAME_INCOME_TYPE"].unique():
    defaulters1 = Fam_Details[(Fam_Details["NAME_INCOME_TYPE"] == i) & (Client_Data["TARGET"] ==1)]
    Total = len(Client_Data[Client_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))

Working :   61.372
State servant :   5.021
Commercial associate :   21.567
Pensioner :   11.999
Unemployed :   0.032
Student :   0.0
Businessman :   0.0
Maternity leave :   0.008
```

- Highest Defaulters were Working and Commercial associate, but the risk to be assesed on 11% Pensioner based on their asset value.

```
In [43]: fig,axes =plt.subplots(2,1,figsize=(10,10))
sns.countplot(data =Client_Data,x="NAME_HOUSING_TYPE",hue="TARGET",ax=axes[0],color="red")
sns.countplot(data =Client_Data,x="NAME_TYPE_SUITE",hue="TARGET",ax=axes[1],color="red")
fig.tight_layout()
```




```
In [44]: for i in Fam_Details["NAME_HOUSING_TYPE"].unique():
    defaulters1 = Fam_Details[(Fam_Details["NAME_HOUSING_TYPE"] == i) & (Client_Data["TARGET"] ==1)]
    Total = len(Client_Data[Client_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))

House / apartment :   85.691
Rented apartment :   2.415
With parents :   7.007
Municipal apartment :   3.843
Office apartment :   0.688
Co-op apartment :   0.356
```

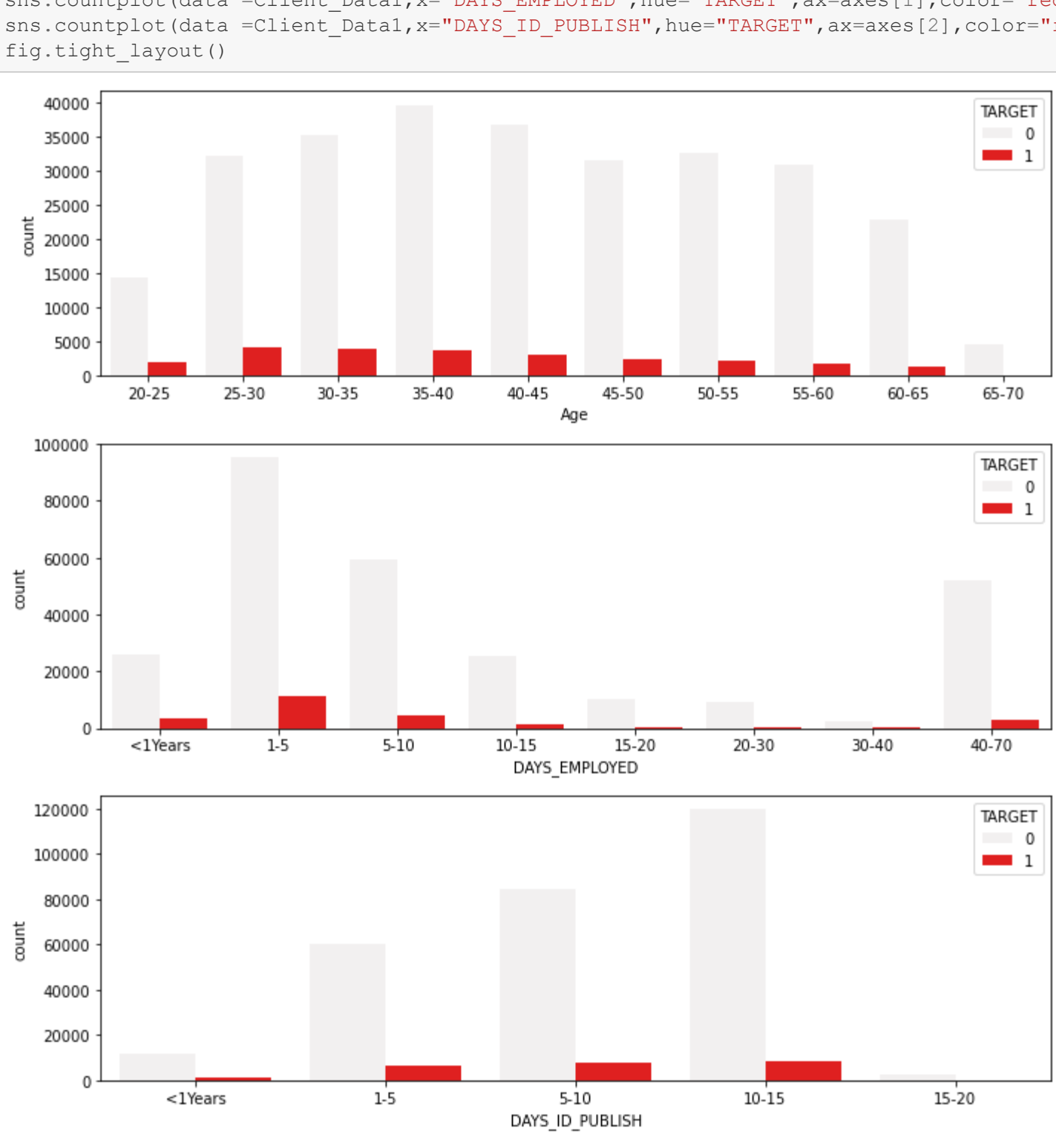
```
In [45]: for i in Fam_Details["NAME_TYPE_SUITE"].unique():
    defaulters1 = Fam_Details[(Fam_Details["NAME_TYPE_SUITE"] == i) & (Client_Data["TARGET"] ==1)]
    Total = len(Client_Data[Client_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))

Unaccompanied :   82.13
Family :   12.169
Spouse, partner :   3.621
Children :   0.975
Other_A :   0.307
Other_B :   0.704
Group of people :   0.093
```

- 82% Defaulters were Unaccompanied while submiting loan Application, but cannot be considered as primary risk factor
- 2% Defaulters live in Rented Apartments a concern factor.

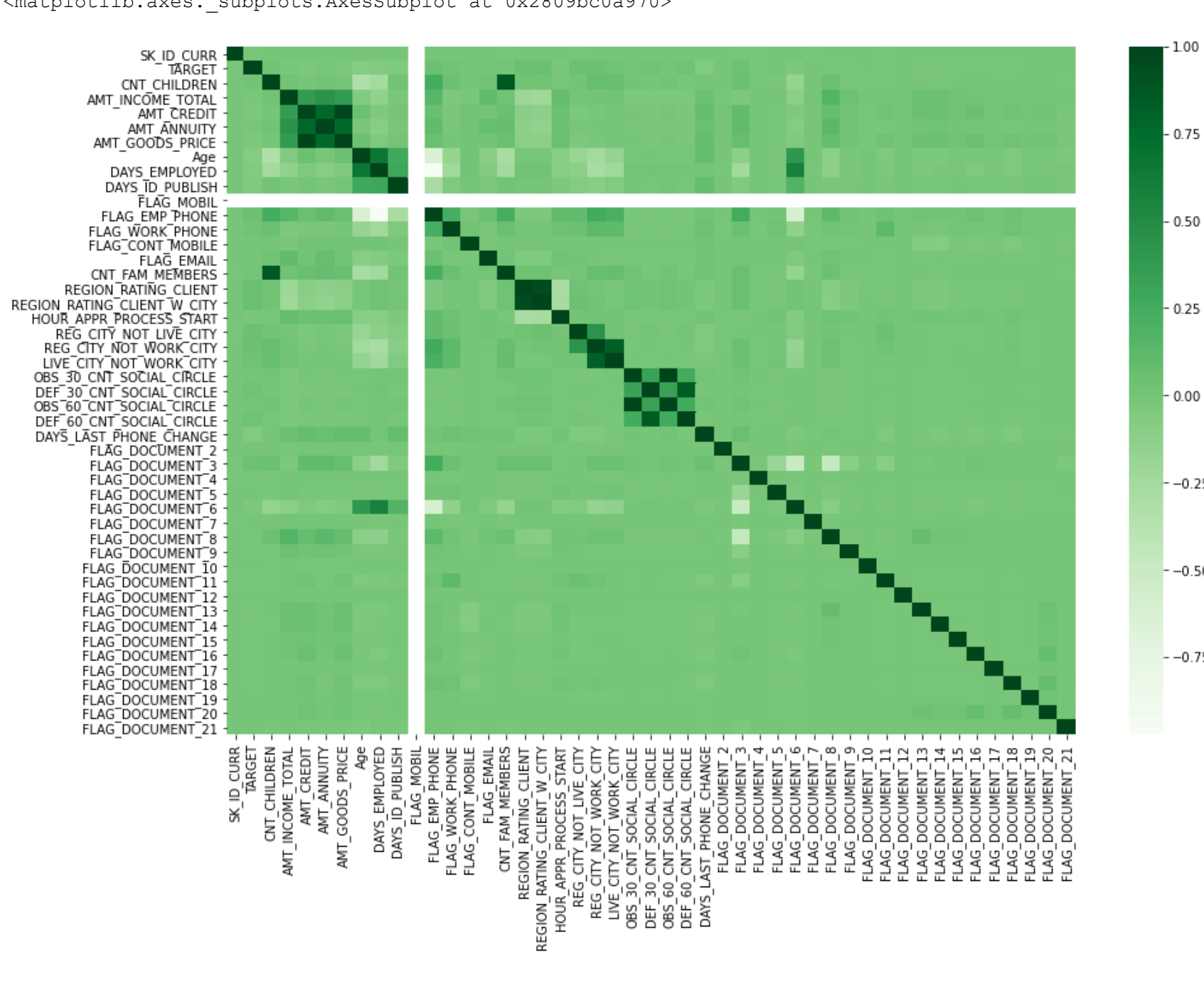
```
In [46]: Client_Data1=Client_Data.copy()
```

```
In [47]: Client_Data1["Age"] = pd.cut(x= Client_Data1["Age"],bins = [20,25,30,35,40,45,50,55,60,65,70],labels = ['20-25','25-30','30-35','35-40','40-45','45-50','50-55','55-60','60-65','65-70'])
Client_Data1["DAYS_EMPLOYED"] = pd.cut(x= Client_Data1["DAYS_EMPLOYED"],bins = [0.1,1.0,5.0,10.0,15.0,20.0,30.0,40.0,70.0],labels = ['<1Years','1-5','5-10','10-15','15-20','20-30','30-40','40-70'])
Client_Data1["DAYS_ID_PUBLISH"] = pd.cut(x= Client_Data1["DAYS_ID_PUBLISH"],bins = [0.1,1.0,5.0,10.0,15.0,20.0],labels = ['<1Years','1-5','5-10','10-15','15-20'])
```

```
In [48]: fig,axes =plt.subplots(3,1,figsize=(10,10))
sns.countplot(data =Client_Data1,x="Age",hue="TARGET",ax=axes[0],color="red")
sns.countplot(data =Client_Data1,x="DAYS_EMPLOYED",hue="TARGET",ax=axes[1],color="red")
sns.countplot(data =Client_Data1,x="DAYS_ID_PUBLISH",hue="TARGET",ax=axes[2],color="red")
fig.tight_layout()
```



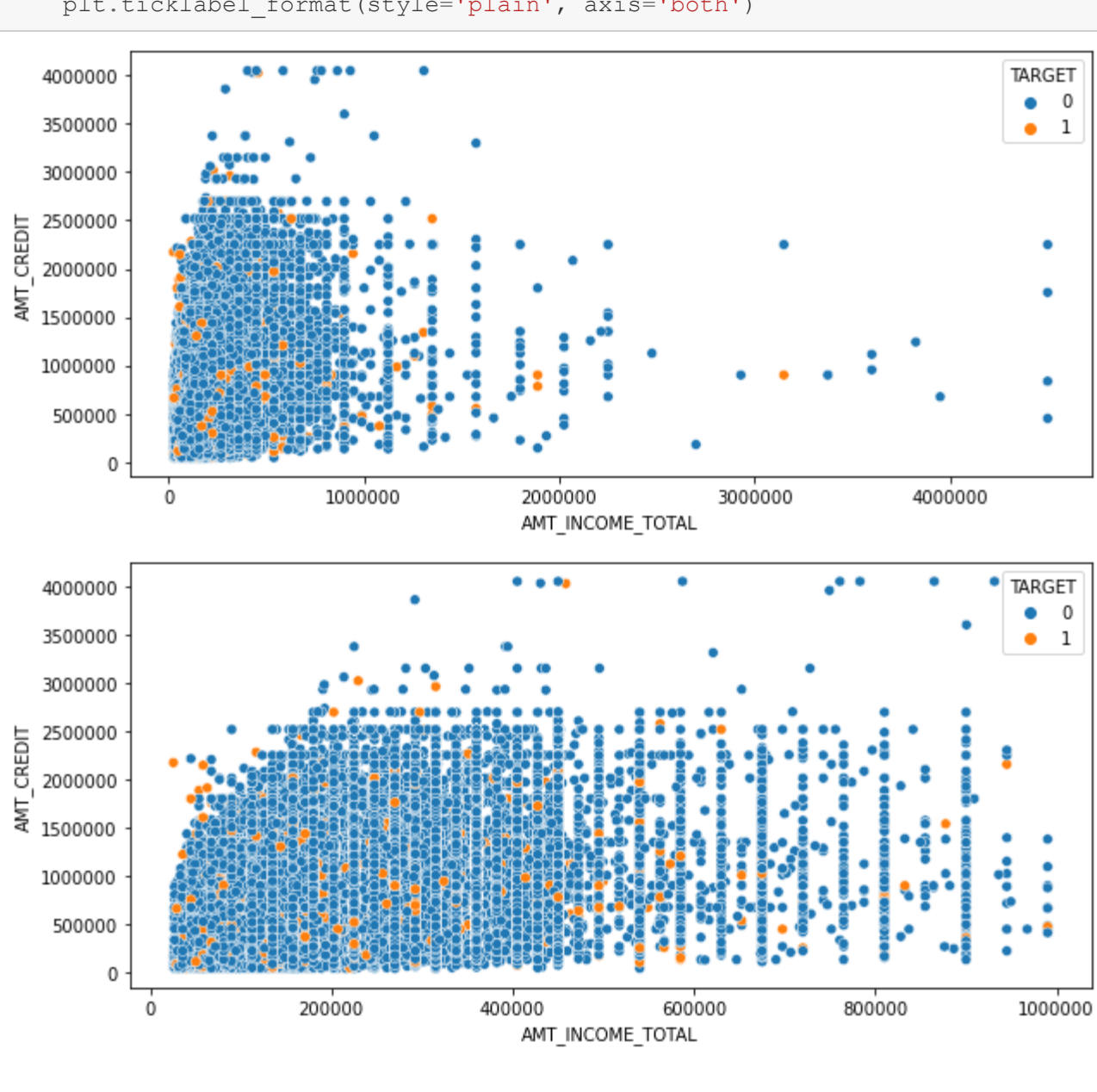

```
In [49]: plt.figure(figsize=(15,10))
sns.heatmap(Client_Data.corr(), cmap="Greens")

Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x2809bc0a970>
```



- Amt_Goods Price is highly correlated with Amt_Credit.
- From our Document Status plot, we didnt got much insights, but we could see correlation exists between Age,Days_Employed with Flag document6.

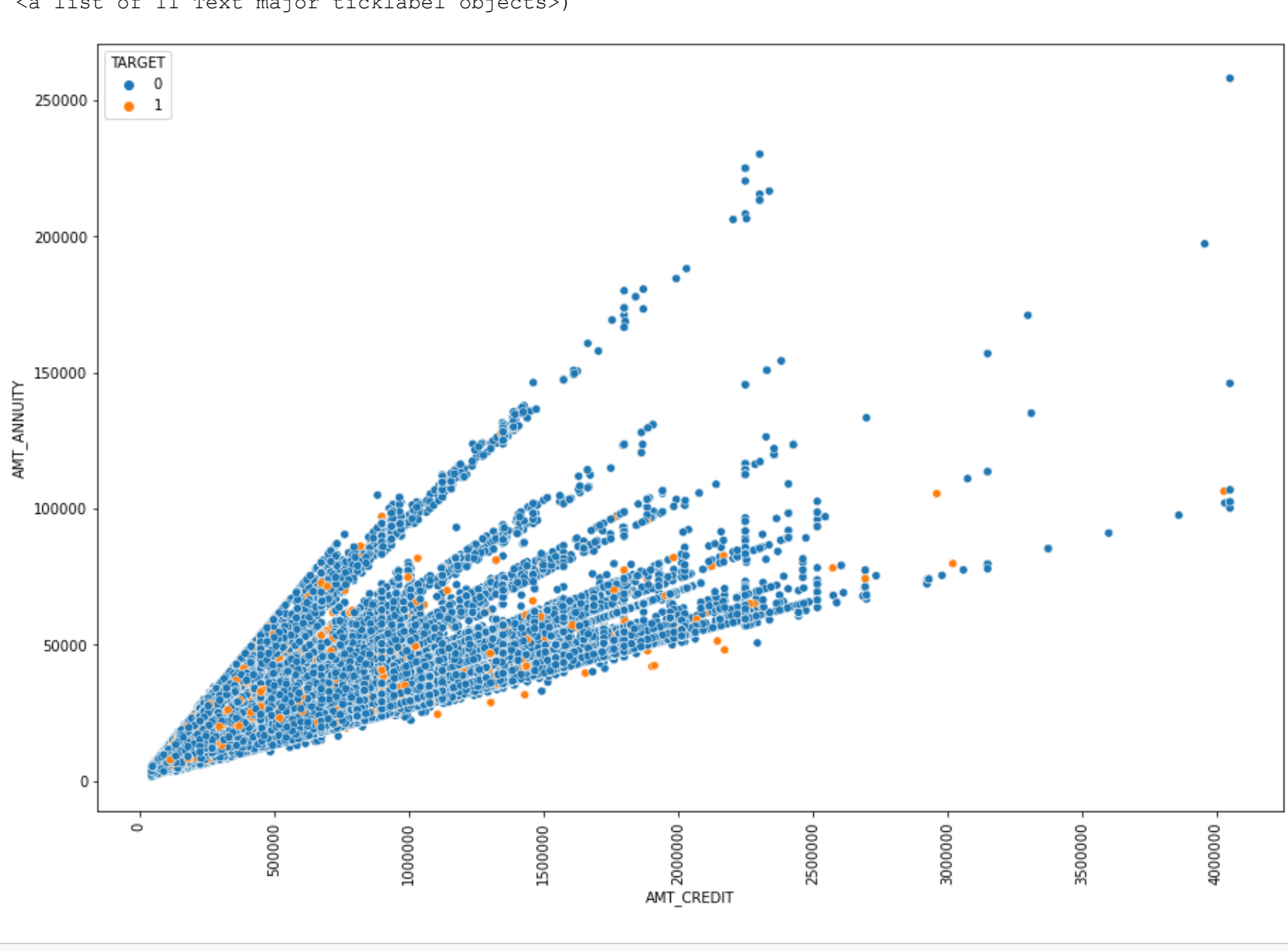
Bi-Variate Analysis

```
In [50]: fig,ax =plt.subplots(2,1,figsize=(10,10))
sns.scatterplot(data=Client_Data, x="AMT_INCOME_TOTAL", y="AMT_CREDIT",hue ="TARGET",ax=ax[0])
Inc =Client_Data.loc[Client_Data["AMT_INCOME_TOTAL"]<1000000,["AMT_INCOME_TOTAL","TARGET","AMT_CREDIT"]]
sns.scatterplot(data=Inc, x="AMT_INCOME_TOTAL", y="AMT_CREDIT",hue ="TARGET",ax=ax[1])

for ax in fig.axes:
    plt.sca(ax)
    plt.ticklabel_format(style='plain', axis='both')
```

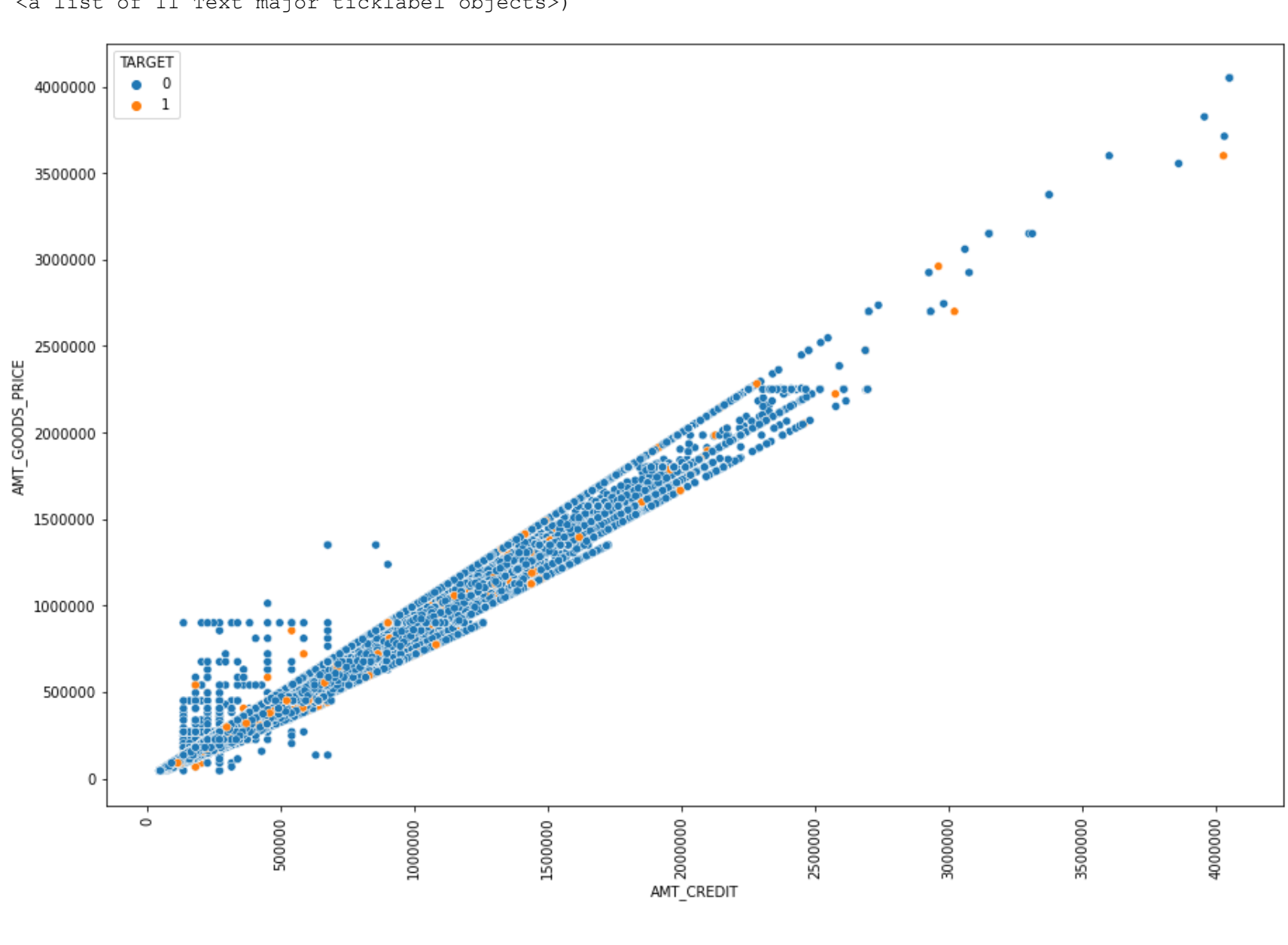


- Majority of Applicants has Income <1Millions and Credited loan amount is < 2.5 Millions
- More Defaulters fall under Income range of 0.4 Millions

```
In [51]: plt.figure(figsize=(15,10),)
sns.scatterplot(data=Client_Data, x="AMT_CREDIT", y="AMT_ANNUITY",hue ="TARGET")
plt.ticklabel_format(style='plain', axis='both')
plt.xticks(rotation =90)

Out[51]: (array([-500000.,       0.,  500000., 1000000., 1500000., 2000000.,
        2500000., 3000000., 3500000., 4000000., 4500000.]),
 <a list of 11 Text major ticklabel objects>)
```




```
In [52]: plt.figure(figsize=(15,10),)
sns.scatterplot(data=Client_Data, x="AMT_CREDIT", y="AMT_GOODS_PRICE",hue ="TARGET")
plt.ticklabel_format(style='plain', axis='both')
plt.xticks(rotation =90)

Out[52]: (array([-500000.,       0.,  500000., 1000000., 1500000., 2000000.,
        2500000., 3000000., 3500000., 4000000., 4500000.]),
 <a list of 11 Text major ticklabel objects>)
```



Conclusion:

8% were Defaulters out of 3Lakh Applicants.

Data_Discrepancies:

- AMT_INCOME column has income ranging from 6-120 Millions, but the credit amount 5% which is concern factor.
- Few Records in Days Employeed has crossed max 70 years, they were imputed with 70.

Findings:

- Maximum loans sanctioned were Cash loans.
- Highest Applicants are female, but deafuters were high in Male.
- Risk is less with applicants who are having either car or Own House.
- Applicants in Rented houses and Pensioners are of high risk.
- Risk is high with Single parent applicants
- Maximum Defaulters lie in Age group of 25-30.
- Maximum Defaulters has employee experince < 5years, but the risk lies with <1 year Experience Employees.
- Most Applicants and defaulters fall under Income range of 40K and 2.5 Lakhs.

```
In [53]: Client_Data.to_csv("Client_data.csv",index=False)

In [ ]:
```

Part 2 with Previous Applicants

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import os
import warnings
warnings.simplefilter(action='ignore')
%matplotlib inline
```

In [2]:
```python
os.chdir("D:\Lohith\DA\Handson\Risk_Analytics")
```

In [3]:
```python
Client_History = pd.read_csv("previous_application.csv")   #Reading the Client History
Description = pd.read_csv(r"columns_description.csv",encoding='cp1252')
Client_Data = pd.read_csv("Client_data.csv") #Reading the Present Client Data
```

By Default it will show only 20 columns,Since we have more then 20 column, this function is used

In [4]:
```python
pd.set_option("display.max_columns",None,"display.max_rows",None)
```

In [5]:
```python
Description.drop(columns=["Unnamed: 0","Special"], axis =1, inplace = True)  # Description of our Colum
ns
Description[122:]
```

Out[5]:

	Table	Row	Description
122	previous_application.csv	SK_ID_PREV	ID of previous credit in Home credit related t...
123	previous_application.csv	SK_ID_CURR	ID of loan in our sample
124	previous_application.csv	NAME_CONTRACT_TYPE	Contract product type (Cash loan, consumer loa...
125	previous_application.csv	AMT_ANNUITY	Annuity of previous application
126	previous_application.csv	AMT_APPLICATION	For how much credit did client ask on the prev...
127	previous_application.csv	AMT_CREDIT	Final credit amount on the previous applicatio...
128	previous_application.csv	AMT_DOWN_PAYMENT	Down payment on the previous application
129	previous_application.csv	AMT_GOODS_PRICE	Goods price of good that client asked for (if ...
130	previous_application.csv	WEEKDAY_APPR_PROCESS_START	On which day of the week did the client apply ...
131	previous_application.csv	HOUR_APPR_PROCESS_START	Approximately at what day hour did the client ...
132	previous_application.csv	FLAG_LAST_APPL_PER_CONTRACT	Flag if it was last application for the previo...
133	previous_application.csv	NFLAG_LAST_APPL_IN_DAY	Flag if the application was the last applicati...
134	previous_application.csv	NFLAG_MICRO_CASH	Flag Micro finance loan
135	previous_application.csv	RATE_DOWN_PAYMENT	Down payment rate normalized on previous credit
136	previous_application.csv	RATE_INTEREST_PRIMARY	Interest rate normalized on previous credit
137	previous_application.csv	RATE_INTEREST_PRIVILEGED	Interest rate normalized on previous credit
138	previous_application.csv	NAME_CASH_LOAN_PURPOSE	Purpose of the cash loan
139	previous_application.csv	NAME_CONTRACT_STATUS	Contract status (approved, cancelled, ...) of ...
140	previous_application.csv	DAYS_DECISION	Relative to current application when was the d...
141	previous_application.csv	NAME_PAYMENT_TYPE	Payment method that client chose to pay for th...
142	previous_application.csv	CODE_REJECT_REASON	Why was the previous application rejected
143	previous_application.csv	NAME_TYPE_SUITE	Who accompanied client when applying for the p...
144	previous_application.csv	NAME_CLIENT_TYPE	Was the client old or new client when applying...
145	previous_application.csv	NAME_GOODS_CATEGORY	What kind of goods did the client apply for in...
146	previous_application.csv	NAME_PORTFOLIO	Was the previous application for CASH, POS, CA...
147	previous_application.csv	NAME_PRODUCT_TYPE	Was the previous application x-sell o walk-in
148	previous_application.csv	CHANNEL_TYPE	Through which channel we acquired the client o...
149	previous_application.csv	SELLERPLACE_AREA	Selling area of seller place of the previous a...
150	previous_application.csv	NAME_SELLER_INDUSTRY	The industry of the seller
151	previous_application.csv	CNT_PAYMENT	Term of previous credit at application of the ...
152	previous_application.csv	NAME_YIELD_GROUP	Grouped interest rate into small medium and hi...
153	previous_application.csv	PRODUCT_COMBINATION	Detailed product combination of the previous a...
154	previous_application.csv	DAYS_FIRST_DRAWING	Relative to application date of current applic...
155	previous_application.csv	DAYS_FIRST_DUE	Relative to application date of current applic...
156	previous_application.csv	DAYS_LAST_DUE_1ST_VERSION	Relative to application date of current applic...
157	previous_application.csv	DAYS_LAST_DUE	Relative to application date of current applic...
158	previous_application.csv	DAYS_TERMINATION	Relative to application date of current applic...
159	previous_application.csv	NFLAG_INSURED_ON_APPROVAL	Did the client requested insurance during the ...

In [6]:
```python
Client_History.head(5) # Checking the top 5 records
```

Out[6]:

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

In [7]:
```python
Client_Data.head(5) # Checking the top 5 records
```

Out[7]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOI
0	100002	1	Cash loans	M	N	Y	0	
1	100003	0	Cash loans	F	N	N	0	
2	100004	0	Revolving loans	M	Y	Y	0	
3	100006	0	Cash loans	F	N	Y	0	
4	100007	0	Cash loans	M	N	Y	0	

In [8]:
```python
print("Client_History: ",Client_History.shape)
```

```
Client_History:  (1670214, 37)
```

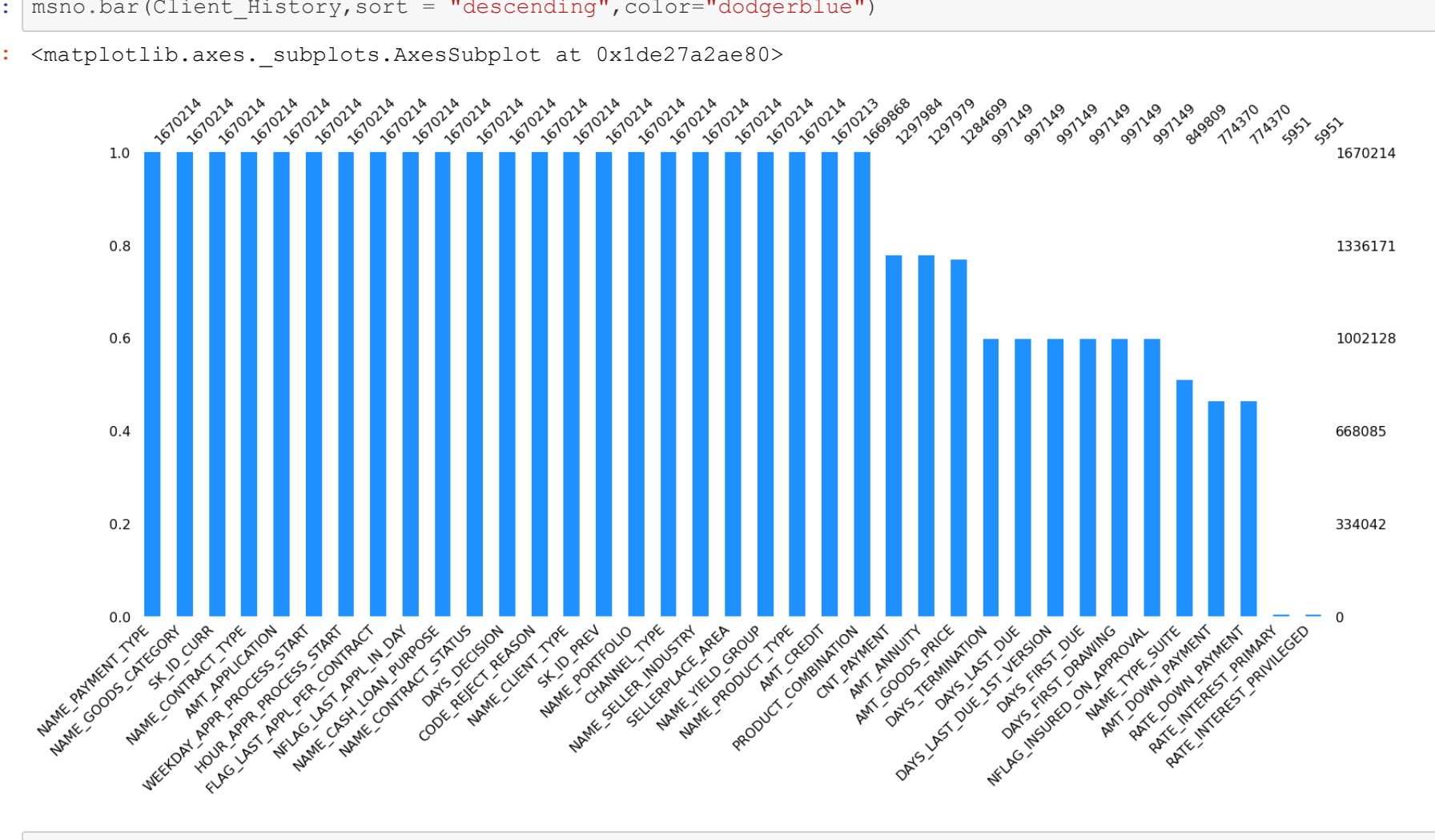
In [9]:
```python
print("Client_Data: ",Client_Data.shape)
```

```
Client_Data:  (305180, 58)
```

Drop the columns with missing vales more than 50%

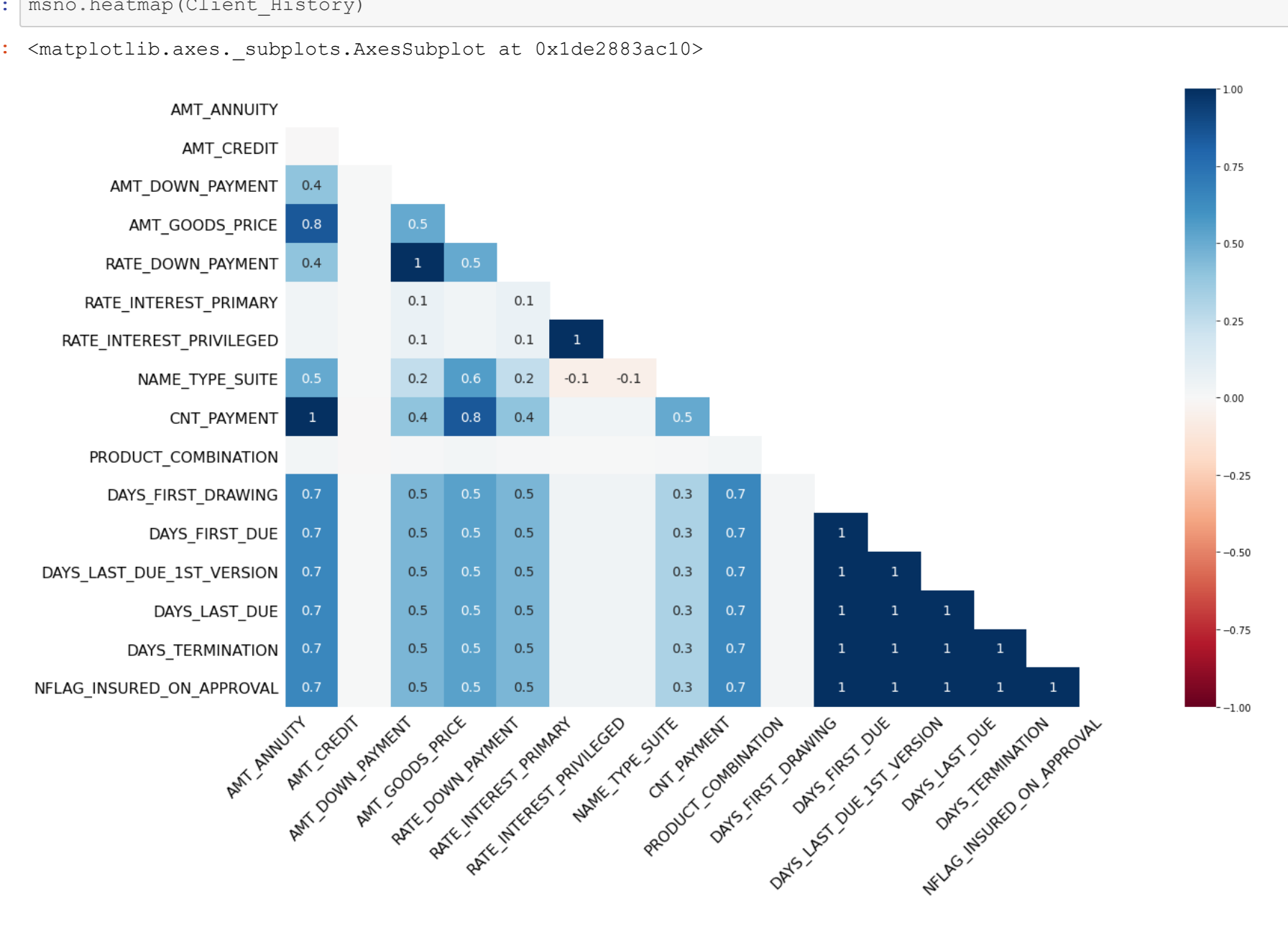
In [10]:
```python
msno.bar(Client_History,sort = "descending",color="dodgerblue")
```

Out[10]: `<matplotlib.axes._subplots.AxesSubplot at 0x1de27a2ae80>`



In [11]:
```python
msno.heatmap(Client_History)
```

Out[11]: `<matplotlib.axes._subplots.AxesSubplot at 0x1de2883ac10>`



- The reason for high Correlation between Days** columns is due to similar values in most of the corresponding columns.
- We have two option, (a)Drop the rows with null values. (b)Drop the Columns as there were 40% missing values. (C)To check why these are null and whether there is any relation between contract Status.

In [12]:
```python
Client_History["NAME_CONTRACT_STATUS"].unique()
```

Out[12]:
```
array(['Approved', 'Refused', 'Canceled', 'Unused offer'], dtype=object)
```

In [13]:
```python
Client_History.loc[(Client_History["NAME_CONTRACT_STATUS"]=='Refused') | (Client_History["NAME_CONTRACT
_STATUS"]=='Canceled') | (Client_History["NAME_CONTRACT_STATUS"]=='Unused offer'),["NAME_CONTRACT_STATU
S","AMT_ANNUITY","AMT_APPLICATION","AMT_CREDIT","AMT_DOWN_PAYMENT","AMT_GOODS_PRICE","DAYS_FIRST_DRAWIN
G","DAYS_FIRST_DUE","DAYS_LAST_DUE_1ST_VERSION","DAYS_LAST_DUE","DAYS_TERMINATION","NFLAG_INSURED_ON_AP
PROVAL"]].head(15)
```

Out[13]:

	NAME_CONTRACT_STATUS	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	DAYS_FIR
4	Refused	31924.395	337500.0	404055.0	NaN	337500.0	
6	Canceled	NaN	0.0	0.0	NaN	NaN	
7	Canceled	NaN	0.0	0.0	NaN	NaN	
8	Canceled	NaN	0.0	0.0	NaN	NaN	
9	Canceled	NaN	0.0	0.0	NaN	NaN	
19	Refused	4666.500	45000.0	49455.0	NaN	45000.0	
20	Refused	25454.025	450000.0	491580.0	NaN	450000.0	
22	Refused	NaN	0.0	0.0	NaN	NaN	
23	Refused	39475.305	1129500.0	1277104.5	NaN	1129500.0	
24	Canceled	NaN	0.0	0.0	NaN	NaN	
29	Refused	NaN	0.0	0.0	NaN	NaN	
32	Canceled	NaN	0.0	0.0	NaN	NaN	
33	Canceled	NaN	0.0	0.0	NaN	NaN	
36	Refused	7511.085	32490.0	38380.5	0.0	32490.0	
41	Unused offer	NaN	17950.5	17950.5	0.0	17950.5	

In [14]:
```python
Client_History.loc[((Client_History["NAME_CONTRACT_STATUS"]=='Approved') & ((Client_History["AMT_APPLIC
ATION"]==0) | (Client_History["AMT_CREDIT"]==0))),["NAME_CONTRACT_STATUS"]].count()
```

Out[14]:
```
NAME_CONTRACT_STATUS    45304
dtype: int64
```

In [15]:
```python
Client_History.loc[(Client_History["NAME_CONTRACT_STATUS"]=='Refused'),["NAME_CONTRACT_STATUS"]].count
()
```

Out[15]:
```
NAME_CONTRACT_STATUS    290678
dtype: int64
```

In [16]:
```python
Client_History.loc[Client_History["NAME_CONTRACT_STATUS"]=='Canceled',["NAME_CONTRACT_STATUS"]].count()
```

Out[16]:
```
NAME_CONTRACT_STATUS    316319
dtype: int64
```

In [17]:
```python
Client_History.loc[(Client_History["NAME_CONTRACT_STATUS"]=='Unused offer'),["NAME_CONTRACT_STATUS"]].c
ount()
```

Out[17]:
```
NAME_CONTRACT_STATUS    26436
dtype: int64
```

In [18]:
```python
Client_History.loc[(Client_History["NAME_CONTRACT_STATUS"]=='Refused') | (Client_History["NAME_CONTRACT
_STATUS"]=='Canceled') | (Client_History["NAME_CONTRACT_STATUS"]=='Unused offer'),["NAME_CONTRACT_STATU
S"]].count()
```

Out[18]:
```
NAME_CONTRACT_STATUS    633433
dtype: int64
```

In [19]:
```python
Client_History["AMT_ANNUITY"].isnull().sum()
```

Out[19]: `372235`

In [20]:
```python
Client_History["DAYS_FIRST_DRAWING"].isnull().sum()
```

Out[20]: `673065`

In [21]:
```python
#Client_History1 = Client_History.copy()
```

In [22]:
```python
Approved_Missings = Client_History.loc[(Client_History["NAME_CONTRACT_STATUS"]=='Approved') & ((Client_
History["DAYS_FIRST_DRAWING"].isna() ==True) | (Client_History["AMT_ANNUITY"].isna() ==True) | (Client_
History["AMT_GOODS_PRICE"].isna() ==True) | (Client_History["CNT_PAYMENT"].isna() ==True) | (Client_His
tory["AMT_APPLICATION"]==0) | (Client_History["AMT_CREDIT"]==0) | (Client_History["AMT_APPLICATION"].is
na()==True) | (Client_History["AMT_CREDIT"].isna()==True))].index
```

In [23]:
```python
len(Approved_Missings)
```

Out[23]: `84027`

In [24]:
```python
Client_History.drop(Approved_Missings,inplace=True)
```

In [25]:
```python
Client_History.drop(columns = ["RATE_INTEREST_PRIVILEGED","RATE_INTEREST_PRIMARY","RATE_DOWN_PAYMENT",
"AMT_DOWN_PAYMENT","DAYS_LAST_DUE_1ST_VERSION","SELLERPLACE_AREA"],axis=1,inplace =True)
```

In [26]:
```python
Client_History.isnull().sum()
```

Out[26]:
```
SK_ID_PREV                          0
SK_ID_CURR                          0
NAME_CONTRACT_TYPE                  0
AMT_ANNUITY                    372227
AMT_APPLICATION                     0
AMT_CREDIT                          0
AMT_GOODS_PRICE                342680
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                763394
NAME_CLIENT_TYPE                    0
NAME_GOODS_CATEGORY                 0
NAME_PORTFOLIO                      0
NAME_PRODUCT_TYPE                   0
CHANNEL_TYPE                        0
NAME_SELLER_INDUSTRY                0
CNT_PAYMENT                    372226
NAME_YIELD_GROUP                    0
PRODUCT_COMBINATION               346
DAYS_FIRST_DRAWING             633433
DAYS_FIRST_DUE                 633433
DAYS_LAST_DUE                  633433
DAYS_TERMINATION               633433
NFLAG_INSURED_ON_APPROVAL      633433
dtype: int64
```

In [27]:
```python
Client_History.shape
```

Out[27]: `(1586187, 31)`

- Imputing null values with mode in below columns

In [28]:
```python
Client_History["NAME_TYPE_SUITE"].fillna(value="Unaccompanied",inplace=True)
```

In [29]:
```python
#Client_History["NFLAG_INSURED_ON_APPROVAL"].fillna(Client_History["NFLAG_INSURED_ON_APPROVAL"].mode(),
inplace=True)
```

In [30]:
```python
Client_History.fillna({"DAYS_FIRST_DRAWING":0,"DAYS_FIRST_DUE":0,"DAYS_LAST_DUE":0,"DAYS_TERMINATION":0
,"AMT_ANNUITY":0,"AMT_GOODS_PRICE":0,"NFLAG_INSURED_ON_APPROVAL":0,"CNT_PAYMENT":0},inplace=True)
```

In [31]:
```python
Client_History.isnull().sum()
```

Out[31]:
```
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
NAME_SELLER_INDUSTRY             0
CNT_PAYMENT                      0
NAME_YIELD_GROUP                 0
PRODUCT_COMBINATION            346
DAYS_FIRST_DRAWING               0
DAYS_FIRST_DUE                   0
DAYS_LAST_DUE                    0
DAYS_TERMINATION                 0
NFLAG_INSURED_ON_APPROVAL        0
dtype: int64
```

In [32]:
```python
Client_History.shape
```

Out[32]: `(1586187, 31)`

In [33]:
```python
Client_History.dropna(how ="any",axis =0,inplace =True)
```

In [34]:
```python
Client_History.shape
```

Out[34]: `(1585841, 31)`

In [35]:
```python
Client_History.isnull().sum()
```

Out[35]:
```
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
NAME_SELLER_INDUSTRY           0
CNT_PAYMENT                    0
NAME_YIELD_GROUP               0
PRODUCT_COMBINATION            0
DAYS_FIRST_DRAWING             0
DAYS_FIRST_DUE                 0
DAYS_LAST_DUE                  0
DAYS_TERMINATION               0
NFLAG_INSURED_ON_APPROVAL      0
dtype: int64
```

In [36]:
```python
Final_Data = pd.merge(Client_Data,Client_History,left_on="SK_ID_CURR",right_on="SK_ID_CURR",how = "inne
r",)
```

In [37]:
```python
print("Final_Data: ",Final_Data.shape)
```

```
Final_Data:  (1334636, 88)
```

In [38]:
```python
Final_Data.head(5)
```

Out[38]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INC
0	100002	1	Cash loans	M	N	Y	0	
1	100003	0	Cash loans	F	N	N	0	
2	100003	0	Cash loans	F	N	N	0	
3	100003	0	Cash loans	F	N	N	0	
4	100004	0	Revolving loans	M	Y	Y	0	

In [39]:
```python
Current_Duplicates = Final_Data[Final_Data.duplicated(['SK_ID_CURR'])]
print("Number of Duplicates: ", len(Current_Duplicates))
```

```
Number of Duplicates:  1045939
```

In [40]:
```python
Previous_Duplicates = Final_Data[Final_Data.duplicated(['SK_ID_PREV'])]
print("Number of Duplicates: ", len(Previous_Duplicates))
```

```
Number of Duplicates:  0
```

In [41]:
```python
Current_Duplicates[['SK_ID_CURR']].nunique()
```

Out[41]:
```
SK_ID_CURR    233211
dtype: int64
```

In [42]:
```python
Final_Data[["SK_ID_CURR"]].nunique()
```

Out[42]:
```
SK_ID_CURR    288697
dtype: int64
```

We could see Current Id's were duplicated, but the previous Id,s and loan details were different. So it is good to go with Merged data instead of removing, which we can further use it for our Analysis.

- 288697 Applicants requested loan in past

In [43]:
```python
Final_Data.dtypes
```

Out[43]:
```
SK_ID_CURR                       int64
TARGET                           int64
NAME_CONTRACT_TYPE_x            object
CODE_GENDER                     object
FLAG_OWN_CAR                    object
FLAG_OWN_REALTY                 object
CNT_CHILDREN                     int64
AMT_INCOME_TOTAL               float64
AMT_CREDIT_x                   float64
AMT_ANNUITY_x                  float64
AMT_GOODS_PRICE_x              float64
NAME_TYPE_SUITE_x               object
Age                              int64
NAME_INCOME_TYPE                object
NAME_EDUCATION_TYPE             object
NAME_FAMILY_STATUS              object
NAME_HOUSING_TYPE               object
DAYS_EMPLOYED                  float64
DAYS_ID_PUBLISH                float64
FLAG_MOBIL                       int64
FLAG_EMP_PHONE                   int64
FLAG_WORK_PHONE                  int64
FLAG_CONT_MOBILE                 int64
FLAG_EMAIL                       int64
CNT_FAM_MEMBERS                float64
REGION_RATING_CLIENT             int64
REGION_RATING_CLIENT_W_CITY      int64
WEEKDAY_APPR_PROCESS_START_x    object
HOUR_APPR_PROCESS_START_x        int64
REG_CITY_NOT_LIVE_CITY           int64
REG_CITY_NOT_WORK_CITY           int64
LIVE_CITY_NOT_WORK_CITY          int64
ORGANIZATION_TYPE               object
OBS_30_CNT_SOCIAL_CIRCLE       float64
DEF_30_CNT_SOCIAL_CIRCLE       float64
OBS_60_CNT_SOCIAL_CIRCLE       float64
DEF_60_CNT_SOCIAL_CIRCLE       float64
DAYS_LAST_PHONE_CHANGE         float64
FLAG_DOCUMENT_2                  int64
FLAG_DOCUMENT_3                  int64
FLAG_DOCUMENT_4                  int64
FLAG_DOCUMENT_5                  int64
FLAG_DOCUMENT_6                  int64
FLAG_DOCUMENT_7                  int64
FLAG_DOCUMENT_8                  int64
FLAG_DOCUMENT_9                  int64
FLAG_DOCUMENT_10                 int64
FLAG_DOCUMENT_11                 int64
FLAG_DOCUMENT_12                 int64
FLAG_DOCUMENT_13                 int64
FLAG_DOCUMENT_14                 int64
FLAG_DOCUMENT_15                 int64
FLAG_DOCUMENT_16                 int64
FLAG_DOCUMENT_17                 int64
FLAG_DOCUMENT_18                 int64
FLAG_DOCUMENT_19                 int64
FLAG_DOCUMENT_20                 int64
FLAG_DOCUMENT_21                 int64
SK_ID_PREV                       int64
NAME_CONTRACT_TYPE_y            object
AMT_ANNUITY_y                  float64
AMT_APPLICATION                float64
AMT_CREDIT_y                   float64
AMT_GOODS_PRICE_y              float64
WEEKDAY_APPR_PROCESS_START_y    object
HOUR_APPR_PROCESS_START_y        int64
FLAG_LAST_APPL_PER_CONTRACT     object
NFLAG_LAST_APPL_IN_DAY           int64
NAME_CASH_LOAN_PURPOSE          object
NAME_CONTRACT_STATUS            object
DAYS_DECISION                    int64
NAME_PAYMENT_TYPE               object
CODE_REJECT_REASON              object
NAME_TYPE_SUITE_y               object
NAME_CLIENT_TYPE                object
NAME_GOODS_CATEGORY             object
NAME_PORTFOLIO                  object
NAME_PRODUCT_TYPE               object
CHANNEL_TYPE                    object
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                  float64
DAYS_TERMINATION               float64
NFLAG_INSURED_ON_APPROVAL      float64
dtype: object
```

Converting the below columns to Absolute values

In [44]:
```python
Final_Data["DAYS_TERMINATION"]=Final_Data["DAYS_TERMINATION"].abs()
Final_Data["DAYS_LAST_DUE"] = Final_Data["DAYS_TERMINATION"].abs()
Final_Data["DAYS_FIRST_DUE"] = Final_Data["DAYS_FIRST_DUE"].abs()
Final_Data["DAYS_DECISION"] = Final_Data["DAYS_DECISION"].abs()
Final_Data["DAYS_FIRST_DRAWING"] = Final_Data["DAYS_FIRST_DRAWING"].abs()
Final_Data["AMT_ANNUITY_y"] = Final_Data["AMT_ANNUITY_y"].abs()
Final_Data["AMT_APPLICATION"] = Final_Data["AMT_APPLICATION"].abs()
Final_Data["AMT_CREDIT_y"] = Final_Data["AMT_CREDIT_y"].abs()
Final_Data["AMT_GOODS_PRICE_y"] = Final_Data["AMT_GOODS_PRICE_y"].abs()
```

In [45]:
```python
Final_Data[["AMT_ANNUITY_y"]].head(5)
```

Out[45]:

	AMT_ANNUITY_y
0	9251.775
1	98356.995
2	64567.665
3	6737.310
4	5357.250

Converting number of Days to Months/Years

In [46]:
```python
Final_Data["DAYS_DECISION"] = round(Final_Data[["DAYS_DECISION"]]/365.25,1)
Final_Data["DAYS_FIRST_DRAWING"] = round(Final_Data[["DAYS_FIRST_DRAWING"]]/365.25,1)
Final_Data["DAYS_FIRST_DUE"] = round(Final_Data[["DAYS_FIRST_DUE"]]/365.25,1)
Final_Data["DAYS_LAST_DUE"] = round(Final_Data[["DAYS_LAST_DUE"]]/365.25,1)
Final_Data["DAYS_TERMINATION"] = round(Final_Data[["DAYS_TERMINATION"]]/365.25,1)
```

In [47]:
```python
Final_Data.rename(columns={"DAYS_DECISION":"DECISION_YEARS","DAYS_FIRST_DUE":"FIRST_DUE_YEARS","DAYS_LA
ST_DUE":"LAST_DUE_YEARS","DAYS_TERMINATION":"TERMINATION_YEARS","DAYS_FIRST_DRAWING":"FIRST_DRAWING_YEA
RS"},inplace =True)
```

In [48]:
```python
for i in Final_Data[["DECISION_YEARS","FIRST_DUE_YEARS","LAST_DUE_YEARS","TERMINATION_YEARS","FIRST_DRA
WING_YEARS"]]:
    print(i,": ","\n",Final_Data[i].sort_values().unique())
```

```
DECISION_YEARS :
 [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1 3.2 3.3 3.4 3.5
 3.6 3.7 3.8 3.9 4.  4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.  5.1 5.2 5.3
 5.4 5.5 5.6 5.7 5.8 5.9 6.  6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.  7.1
 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9 8. ]
FIRST_DUE_YEARS :
 [0.0e+00 1.0e-01 2.0e-01 3.0e-01 4.0e-01 5.0e-01 6.0e-01 7.0e-01 8.0e-01
 9.0e-01 1.0e+00 1.1e+00 1.2e+00 1.3e+00 1.4e+00 1.5e+00 1.6e+00 1.7e+00
 1.8e+00 1.9e+00 2.0e+00 2.1e+00 2.2e+00 2.3e+00 2.4e+00 2.5e+00 2.6e+00
 2.7e+00 2.8e+00 2.9e+00 3.0e+00 3.1e+00 3.2e+00 3.3e+00 3.4e+00 3.5e+00
 3.6e+00 3.7e+00 3.8e+00 3.9e+00 4.0e+00 4.1e+00 4.2e+00 4.3e+00 4.4e+00
 4.5e+00 4.6e+00 4.7e+00 4.8e+00 4.9e+00 5.0e+00 5.1e+00 5.2e+00 5.3e+00
 5.4e+00 5.5e+00 5.6e+00 5.7e+00 5.8e+00 5.9e+00 6.0e+00 6.1e+00 6.2e+00
 6.3e+00 6.4e+00 6.5e+00 6.6e+00 6.7e+00 6.8e+00 6.9e+00 7.0e+00 7.1e+00
 7.2e+00 7.3e+00 7.4e+00 7.5e+00 7.6e+00 7.7e+00 7.8e+00 7.9e+00 1.0e+03]
LAST_DUE_YEARS :
 [0.0e+00 1.0e-01 2.0e-01 3.0e-01 4.0e-01 5.0e-01 6.0e-01 7.0e-01 8.0e-01
 9.0e-01 1.0e+00 1.1e+00 1.2e+00 1.3e+00 1.4e+00 1.5e+00 1.6e+00 1.7e+00
 1.8e+00 1.9e+00 2.0e+00 2.1e+00 2.2e+00 2.3e+00 2.4e+00 2.5e+00 2.6e+00
 2.7e+00 2.8e+00 2.9e+00 3.0e+00 3.1e+00 3.2e+00 3.3e+00 3.4e+00 3.5e+00
 3.6e+00 3.7e+00 3.8e+00 3.9e+00 4.0e+00 4.1e+00 4.2e+00 4.3e+00 4.4e+00
 4.5e+00 4.6e+00 4.7e+00 4.8e+00 4.9e+00 5.0e+00 5.1e+00 5.2e+00 5.3e+00
 5.4e+00 5.5e+00 5.6e+00 5.7e+00 5.8e+00 5.9e+00 6.0e+00 6.1e+00 6.2e+00
 6.3e+00 6.4e+00 6.5e+00 6.6e+00 6.7e+00 6.8e+00 6.9e+00 7.0e+00 7.1e+00
 7.2e+00 7.3e+00 7.4e+00 7.5e+00 7.6e+00 7.7e+00 7.8e+00 7.9e+00 1.0e+03]
TERMINATION_YEARS :
 [0.0e+00 1.0e-01 2.0e-01 3.0e-01 4.0e-01 5.0e-01 6.0e-01 7.0e-01 8.0e-01
 9.0e-01 1.0e+00 1.1e+00 1.2e+00 1.3e+00 1.4e+00 1.5e+00 1.6e+00 1.7e+00
 1.8e+00 1.9e+00 2.0e+00 2.1e+00 2.2e+00 2.3e+00 2.4e+00 2.5e+00 2.6e+00
 2.7e+00 2.8e+00 2.9e+00 3.0e+00 3.1e+00 3.2e+00 3.3e+00 3.4e+00 3.5e+00
 3.6e+00 3.7e+00 3.8e+00 3.9e+00 4.0e+00 4.1e+00 4.2e+00 4.3e+00 4.4e+00
 4.5e+00 4.6e+00 4.7e+00 4.8e+00 4.9e+00 5.0e+00 5.1e+00 5.2e+00 5.3e+00
 5.4e+00 5.5e+00 5.6e+00 5.7e+00 5.8e+00 5.9e+00 6.0e+00 6.1e+00 6.2e+00
 6.3e+00 6.4e+00 6.5e+00 6.6e+00 6.7e+00 6.8e+00 6.9e+00 7.0e+00 7.1e+00
 7.2e+00 7.3e+00 7.4e+00 7.5e+00 7.6e+00 7.7e+00 7.8e+00 7.9e+00 1.0e+03]
FIRST_DRAWING_YEARS :
 [0.0e+00 1.0e-01 2.0e-01 3.0e-01 4.0e-01 5.0e-01 6.0e-01 7.0e-01 8.0e-01
 9.0e-01 1.0e+00 1.1e+00 1.2e+00 1.3e+00 1.4e+00 1.5e+00 1.6e+00 1.7e+00
 1.8e+00 1.9e+00 2.0e+00 2.1e+00 2.2e+00 2.3e+00 2.4e+00 2.5e+00 2.6e+00
 2.7e+00 2.8e+00 2.9e+00 3.0e+00 3.1e+00 5.9e+00 6.0e+00 6.1e+00 6.2e+00
 6.3e+00 6.4e+00 6.5e+00 6.6e+00 6.7e+00 6.8e+00 6.9e+00 7.0e+00 7.1e+00
 7.2e+00 7.3e+00 7.4e+00 7.5e+00 7.6e+00 1.0e+03]
```

In [49]:
```python
Final_Data["FIRST_DUE_YEARS"].sort_values().unique()[-2]
```

Out[49]: `7.9`

In [50]:
```python
print("FIRST_DUE_YEARS",Final_Data["FIRST_DUE_YEARS"].max())
print("LAST_DUE_YEARS",Final_Data["LAST_DUE_YEARS"].max())
print("TERMINATION_YEARS",Final_Data["TERMINATION_YEARS"].max())
print("FIRST_DRAWING_YEARS",Final_Data["FIRST_DRAWING_YEARS"].max())
```

```
FIRST_DUE_YEARS 1000.0
LAST_DUE_YEARS 1000.0
TERMINATION_YEARS 1000.0
FIRST_DRAWING_YEARS 1000.0
```

In [51]:
```python
for i in Final_Data[["FIRST_DUE_YEARS","LAST_DUE_YEARS","TERMINATION_YEARS","FIRST_DRAWING_YEARS"]]:
    Final_Data.replace({i:{Final_Data[i].max():Final_Data[i].sort_values().unique()[-2]}},inplace=True)
```

In [52]:
```python
for i in Final_Data[["DECISION_YEARS","FIRST_DUE_YEARS","LAST_DUE_YEARS","TERMINATION_YEARS","FIRST_DRA
WING_YEARS"]]:
    print(i,": ","\n",Final_Data[i].sort_values().unique())
```

```
DECISION_YEARS :
 [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1 3.2 3.3 3.4 3.5
 3.6 3.7 3.8 3.9 4.  4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.  5.1 5.2 5.3
 5.4 5.5 5.6 5.7 5.8 5.9 6.  6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.  7.1
 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9 8. ]
FIRST_DUE_YEARS :
 [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1 3.2 3.3 3.4 3.5
 3.6 3.7 3.8 3.9 4.  4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.  5.1 5.2 5.3
 5.4 5.5 5.6 5.7 5.8 5.9 6.  6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.  7.1
 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9]
LAST_DUE_YEARS :
 [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1 3.2 3.3 3.4 3.5
 3.6 3.7 3.8 3.9 4.  4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.  5.1 5.2 5.3
 5.4 5.5 5.6 5.7 5.8 5.9 6.  6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.  7.1
 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9]
TERMINATION_YEARS :
 [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1 3.2 3.3 3.4 3.5
 3.6 3.7 3.8 3.9 4.  4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.  5.1 5.2 5.3
 5.4 5.5 5.6 5.7 5.8 5.9 6.  6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.  7.1
 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9]
FIRST_DRAWING_YEARS :
 [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1 5.9 6.  6.1 6.2
 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.  7.1 7.2 7.3 7.4 7.5 7.6]
```

In [53]:
```python
Final_Data.head(5)
```

Out[53]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INC
0	100002	1	Cash loans	M	N	Y	0	
1	100003	0	Cash loans	F	N	N	0	
2	100003	0	Cash loans	F	N	N	0	
3	100003	0	Cash loans	F	N	N	0	
4	100004	0	Revolving loans	M	Y	Y	0	

In [54]:
```python
Final_Data["NAME_CASH_LOAN_PURPOSE"].unique()
```

Out[54]:
```
array(['XAP', 'XNA', 'Other', 'Payments on other loans',
       'Buying a used car', 'Repairs', 'Buying a new car',
       'Everyday expenses', 'Medicine', 'Car repairs', 'Urgent needs',
       'Buying a holiday home / land', 'Building a house or an annex',
       'Furniture', 'Journey', 'Purchase of electronic equipment',
       'Wedding / gift / holiday', 'Education', 'Buying a home',
       'Business development', 'Gasification / water supply',
       'Buying a garage', 'Hobby', 'Money for a third person',
       'Refusal to name the goal'], dtype=object)
```

- In the above result, we could see "XAP","XNA" these type of codes were present in other columns too, since we are not sure of its exact code whether it is some bank code or unavailable data, so leaving it with default values and proceeding with our analysis

Cheking Outlier on our main parameters using Box plot

In [55]:
```python
fig,axes =plt.subplots(3,2,figsize=(10,10))
sns.boxplot("AMT_INCOME_TOTAL",data =Final_Data,ax=axes[0,0])
sns.boxplot("AMT_APPLICATION",data =Final_Data,ax=axes[0,1])
sns.boxplot("AMT_CREDIT_y",data =Final_Data,ax=axes[1,0])
sns.boxplot("AMT_ANNUITY_y",data =Final_Data,ax=axes[1,1])
sns.boxplot("AMT_GOODS_PRICE_y",data =Final_Data,ax=axes[2,0])
fig. tight_layout(pad =3.0)
```

In [56]:
```python
Final_Data[["AMT_APPLICATION","AMT_CREDIT_y","AMT_ANNUITY_y","AMT_GOODS_PRICE_y"]].head()
```

Out[56]:

	AMT_APPLICATION	AMT_CREDIT_y	AMT_ANNUITY_y	AMT_GOODS_PRICE_y
0	179055.0	179055.0	9251.775	179055.0
1	900000.0	1035882.0	98356.995	900000.0
2	337500.0	348637.5	64567.665	337500.0
3	68809.5	68053.5	6737.310	68809.5
4	24282.0	20106.0	5357.250	24282.0

In [57]:
```python
Final_Data[["AMT_INCOME_TOTAL","AMT_APPLICATION","AMT_CREDIT_y","AMT_ANNUITY_y","AMT_GOODS_PRICE_y"]].d
escribe()
```

Out[57]:

	AMT_INCOME_TOTAL	AMT_APPLICATION	AMT_CREDIT_y	AMT_ANNUITY_y	AMT_GOODS_PRICE_y
count	1.334636e+06	1.334636e+06	1.334636e+06	1.334636e+06	1.334636e+06
mean	1.726969e+05	1.756473e+05	1.894188e+05	1.215177e+04	1.756741e+05
std	9.350291e+04	2.913092e+05	3.164248e+05	1.451061e+04	2.913324e+05
min	2.565000e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.125000e+05	2.289600e+04	2.286000e+04	2.250000e+03	2.289983e+04
50%	1.575000e+05	7.320600e+04	7.486650e+04	8.007660e+03	7.321050e+04
75%	2.070000e+05	1.800000e+05	1.988786e+05	1.634117e+04	1.800000e+05
max	4.500000e+06	5.850000e+06	4.509688e+06	4.180581e+05	5.850000e+06

In [58]:
```python
Amount = Final_Data[["AMT_INCOME_TOTAL","AMT_APPLICATION","AMT_CREDIT_y","AMT_ANNUITY_y","AMT_GOODS_PRI
CE_y"]]
```

Calculating 3Standard devaitaion, to check values fall outside 3SD

In [59]:
```
Max = Amount.mean() + (3 * Amount.std())
Max
```

Out[59]:
```
AMT_INCOME_TOTAL      4.532056e+05
AMT_APPLICATION       1.049575e+06
AMT_CREDIT_y          1.138693e+06
AMT_ANNUITY_y         5.568360e+04
AMT_GOODS_PRICE_y     1.049671e+06
dtype: float64
```

In [60]:
```
Min = Amount.mean() - (3 * Amount.std())
Min
```

Out[60]:
```
AMT_INCOME_TOTAL     -107811.858544
AMT_APPLICATION      -698280.332027
AMT_CREDIT_y         -759855.489177
AMT_ANNUITY_y         -31380.066738
AMT_GOODS_PRICE_y    -698322.998914
dtype: float64
```

In [61]:
```
def Outlier_values(Amount,col):
    Outlier = []
    for x in Amount[col]:
        if x < Min[col] or x > Max[col]:
            Outlier.append(x)
    return Outlier
```

In [62]:
```
Income_outlier = Outlier_values(Amount,"AMT_INCOME_TOTAL")
App_Outlier = Outlier_values(Amount,"AMT_APPLICATION")
Credit_Outlier = Outlier_values(Amount,"AMT_CREDIT_y")
Annuity_Outlier = Outlier_values(Amount,"AMT_ANNUITY_y")
Goods_Outlier = Outlier_values(Amount,"AMT_GOODS_PRICE_y")
```

In [63]:
```
print("App_Outlier: ",len(App_Outlier))
print("Credit_Outlier: ",len(Credit_Outlier))
print("Annuity_Outlier: ",len(Annuity_Outlier))
print("Goods_Outlier: ",len(Goods_Outlier))
```

```
App_Outlier:  39938
Credit_Outlier:  37585
Annuity_Outlier:  24555
Goods_Outlier:  39938
```

In [64]:
```
y =[x for x in Income_outlier if x>=4500000]
print("AMT_INCOME_TOTAL",": ",y)
y =[x for x in App_Outlier if x>=5000000]
print("App_Outlier",": ",y)
y =[x for x in Credit_Outlier if x>=4500000]
print("Credit_Outlier",": ",y)
y =[x for x in Annuity_Outlier if x>=400000]
print("Annuity_Outlier",": ",y)
y =[x for x in Goods_Outlier if x>=5000000]
print("Goods_Outlier",": ",y)
```

```
AMT_INCOME_TOTAL :  [4500000.0, 4500000.0, 4500000.0, 4500000.0, 4500000.0, 4500000.0, 4500000.0, 450
0000.0, 4500000.0, 4500000.0]
App_Outlier :  [5850000.0]
Credit_Outlier :  [4509688.5]
Annuity_Outlier :  [418058.145, 417927.645, 418058.145]
Goods_Outlier :  [5850000.0]
```

In [65]:
```
Final_Data.loc[Final_Data["AMT_APPLICATION"]>=5850000,["SK_ID_CURR","NAME_CONTRACT_STATUS","AMT_INCOME_
TOTAL","AMT_APPLICATION","AMT_CREDIT_y","AMT_ANNUITY_y","AMT_GOODS_PRICE_y"]].head()
```

Out[65]:

	SK_ID_CURR	NAME_CONTRACT_STATUS	AMT_INCOME_TOTAL	AMT_APPLICATION	AMT_CREDIT_y	AMT_ANNUITY_y	AMT_GOC
920658	346243	Approved	540000.0	5850000.0	4509688.5	113979.69	

In [66]:
```
Final_Data.loc[Final_Data["AMT_INCOME_TOTAL"]>=4500000,["SK_ID_CURR","NAME_CONTRACT_STATUS","AMT_INCOME
_TOTAL","AMT_APPLICATION","AMT_CREDIT_y","AMT_ANNUITY_y","AMT_GOODS_PRICE_y"]].head()
```

Out[66]:

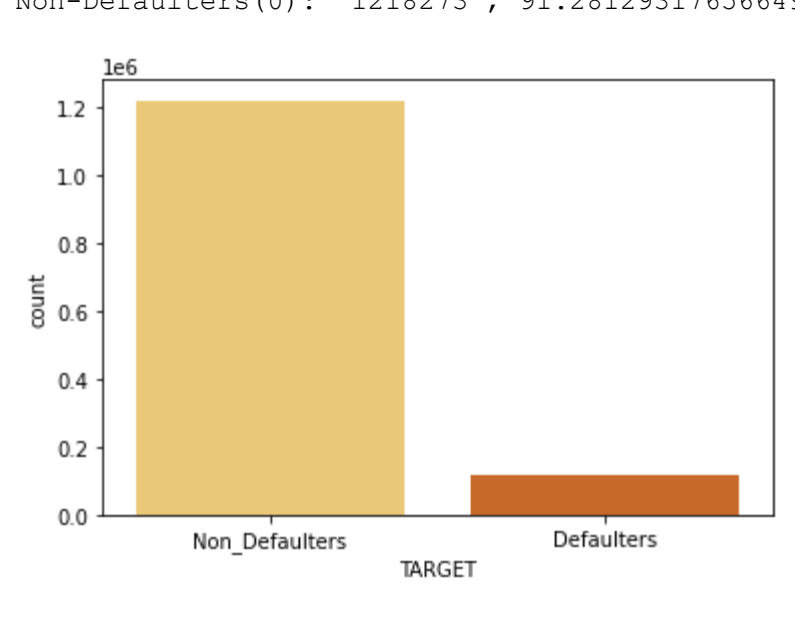
	SK_ID_CURR	NAME_CONTRACT_STATUS	AMT_INCOME_TOTAL	AMT_APPLICATION	AMT_CREDIT_y	AMT_ANNUITY_y	AMT_GOC
812898	317748	Unused offer	4500000.0	91881.0	91881.0	0.000	
812899	317748	Approved	4500000.0	76320.0	84379.5	8176.365	
812900	317748	Approved	4500000.0	450000.0	681354.0	37937.790	
885466	337151	Canceled	4500000.0	0.0	0.0	0.000	
885467	337151	Canceled	4500000.0	0.0	0.0	0.000	

- Even though values fall outside 3 standard deviation, but most of the values are continuos and if we correlate AMT_APPLICATION, Credit, Good price we could see data is feasible.If we remove these values analysis would lean towards applicants of low income range, So we are proceeding with existing values

Univariate Analysis

In [67]:
```
sns.countplot(Final_Data["TARGET"],palette="YlOrBr").set(xticklabels=["Non_Defaulters","Defaulters"])
print("Defaulters(1): ",Final_Data["TARGET"].value_counts()[1],",",(Final_Data["TARGET"].value_counts()
[1]/len(Final_Data["TARGET"]))*100,"%")
print("Non-Defaulters(0): ",Final_Data["TARGET"].value_counts()[0],",",(Final_Data["TARGET"].value_coun
ts()[0]/len(Final_Data["TARGET"]))*100,"%")
```

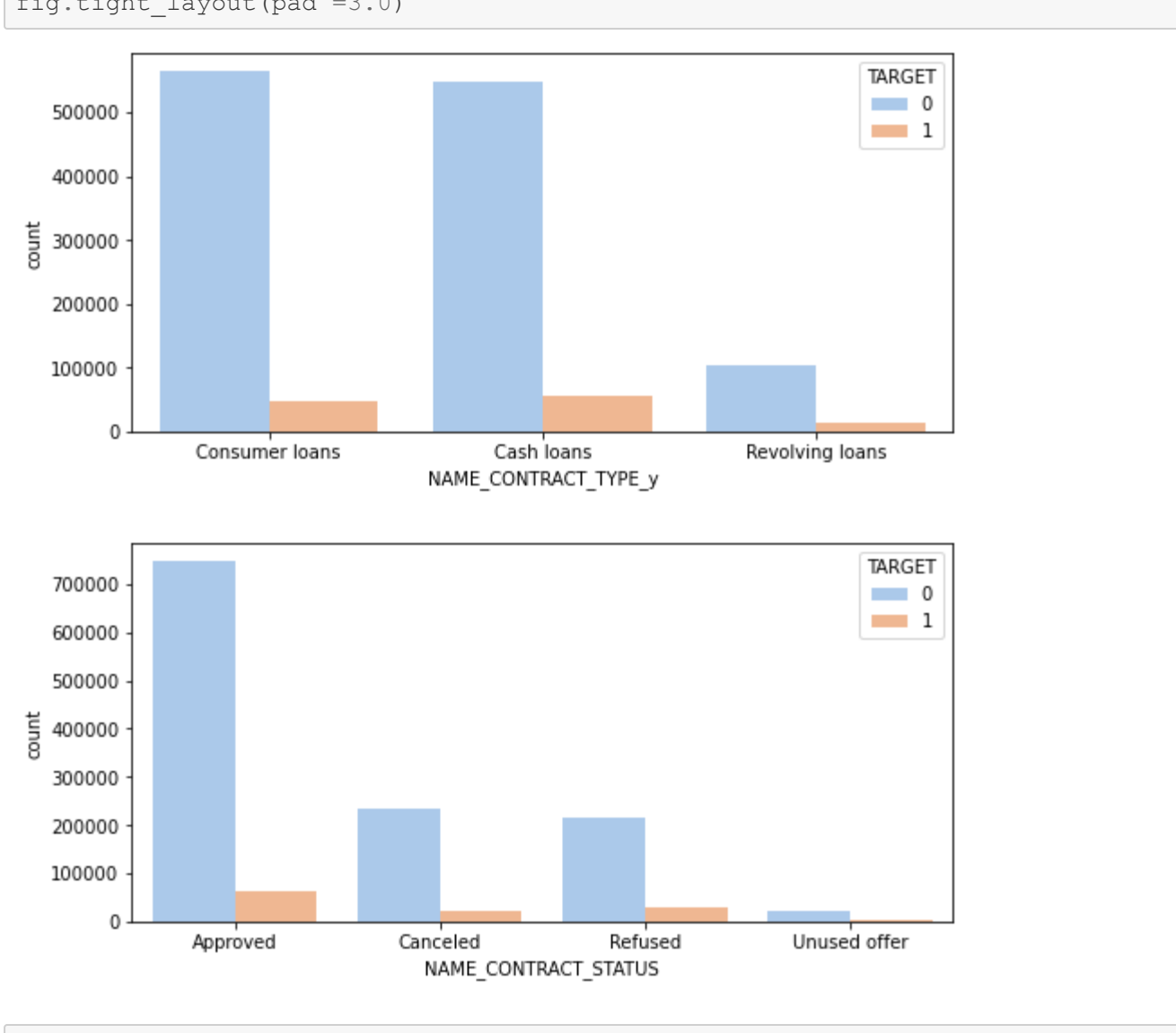
```
Defaulters(1):  116363 , 8.718706823433505 %
Non-Defaulters(0):  1218273 , 91.28129317656649 %
```



- Only 9% were Defaulters, we have to focus on this category

Contract Details

In [68]:
```
fig,axes =plt.subplots(2,1,figsize=(8,8))
sns.countplot(data =Final_Data,x="NAME_CONTRACT_TYPE_y",hue="TARGET",ax=axes[0],palette="pastel")
sns.countplot(data =Final_Data,x="NAME_CONTRACT_STATUS",hue="TARGET",ax=axes[1],palette="pastel")
fig.tight_layout(pad =3.0)
```



In [69]:
```
for i in Final_Data["NAME_CONTRACT_TYPE_y"].unique():
    defaulters1 = Final_Data[(Final_Data["NAME_CONTRACT_TYPE_y"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",(len(defaulters1)/Total)*100)
```

```
Consumer loans :   40.64780041765854
Cash loans :   47.70588589156347
Revolving loans :   11.646313690777998
```

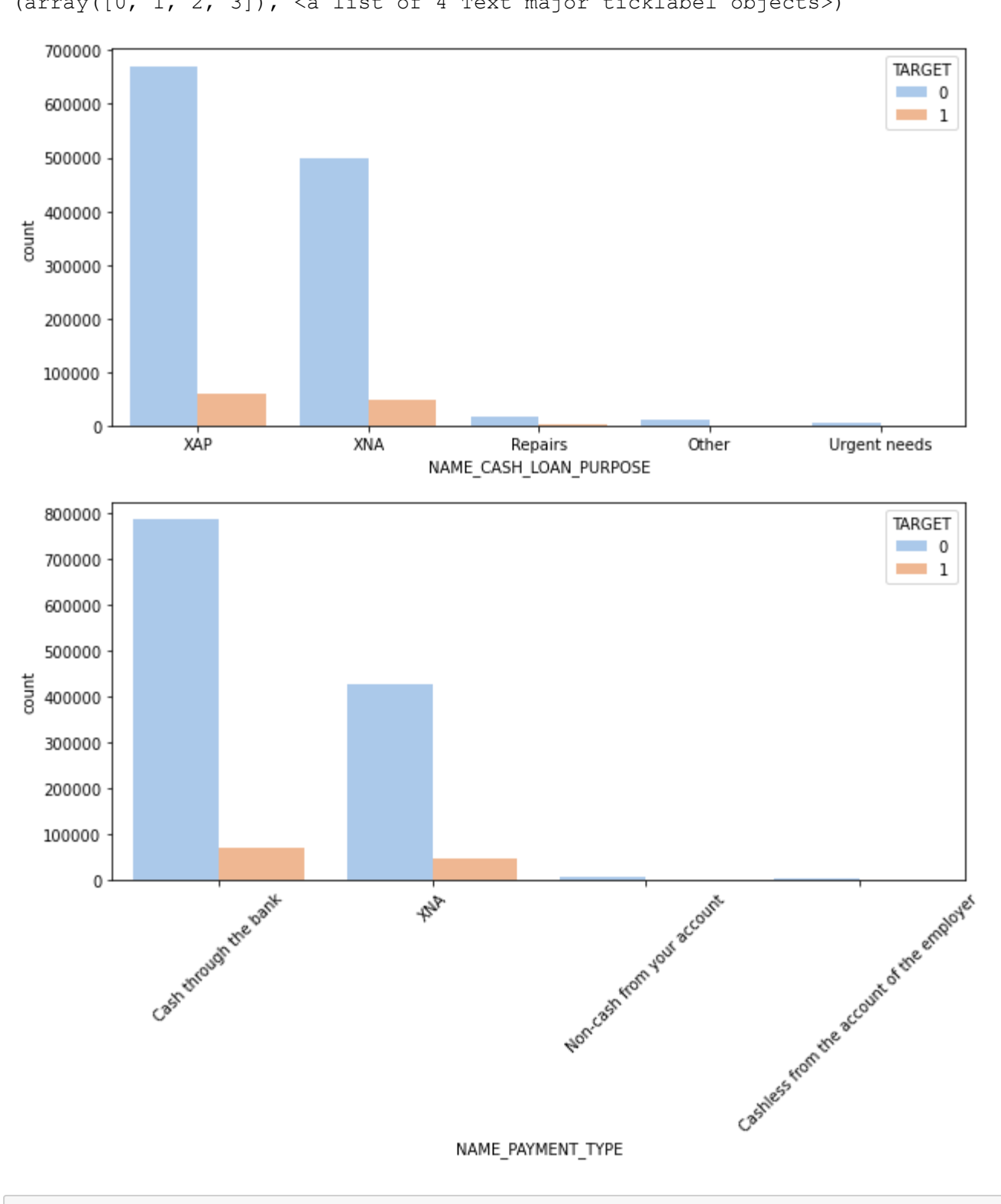
In [70]:
```
for i in Final_Data["NAME_CONTRACT_STATUS"].unique():
    defaulters1 = Final_Data[(Final_Data["NAME_CONTRACT_STATUS"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",(len(defaulters1)/Total)*100)
```

```
Approved :   52.88708610125211
Canceled :   20.340658112974054
Refused :   25.16349698787415
Unused offer :   1.608758797899676
```

- Highest Applicants were from Consumer loans & Cash loans but Defaulters were quite high from Cash loan applicants.
- Highest risk with applicants whose loan application were Canceled & Refused in past.

In [71]:
```
fig,axes =plt.subplots(2,1,figsize=(10,10))
sns.countplot(data= Final_Data,x="NAME_CASH_LOAN_PURPOSE",hue="TARGET",order=Final_Data["NAME_CASH_LOAN
_PURPOSE"].value_counts().sort_values(ascending = False).iloc[:5].index,ax=axes[0],palette="pastel")
sns.countplot(data =Final_Data,x="NAME_PAYMENT_TYPE",hue="TARGET",order=Final_Data["NAME_PAYMENT_TYPE"]
.value_counts().sort_values(ascending = False).iloc[:5].index,ax=axes[1],palette="pastel")
plt.xticks(rotation=45)
```

Out[71]: (array([0, 1, 2, 3]), <a list of 4 Text major ticklabel objects>)



In [72]:
```
for i in Final_Data["NAME_CASH_LOAN_PURPOSE"].unique():
    defaulters1 = Final_Data[(Final_Data["NAME_CASH_LOAN_PURPOSE"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))
```

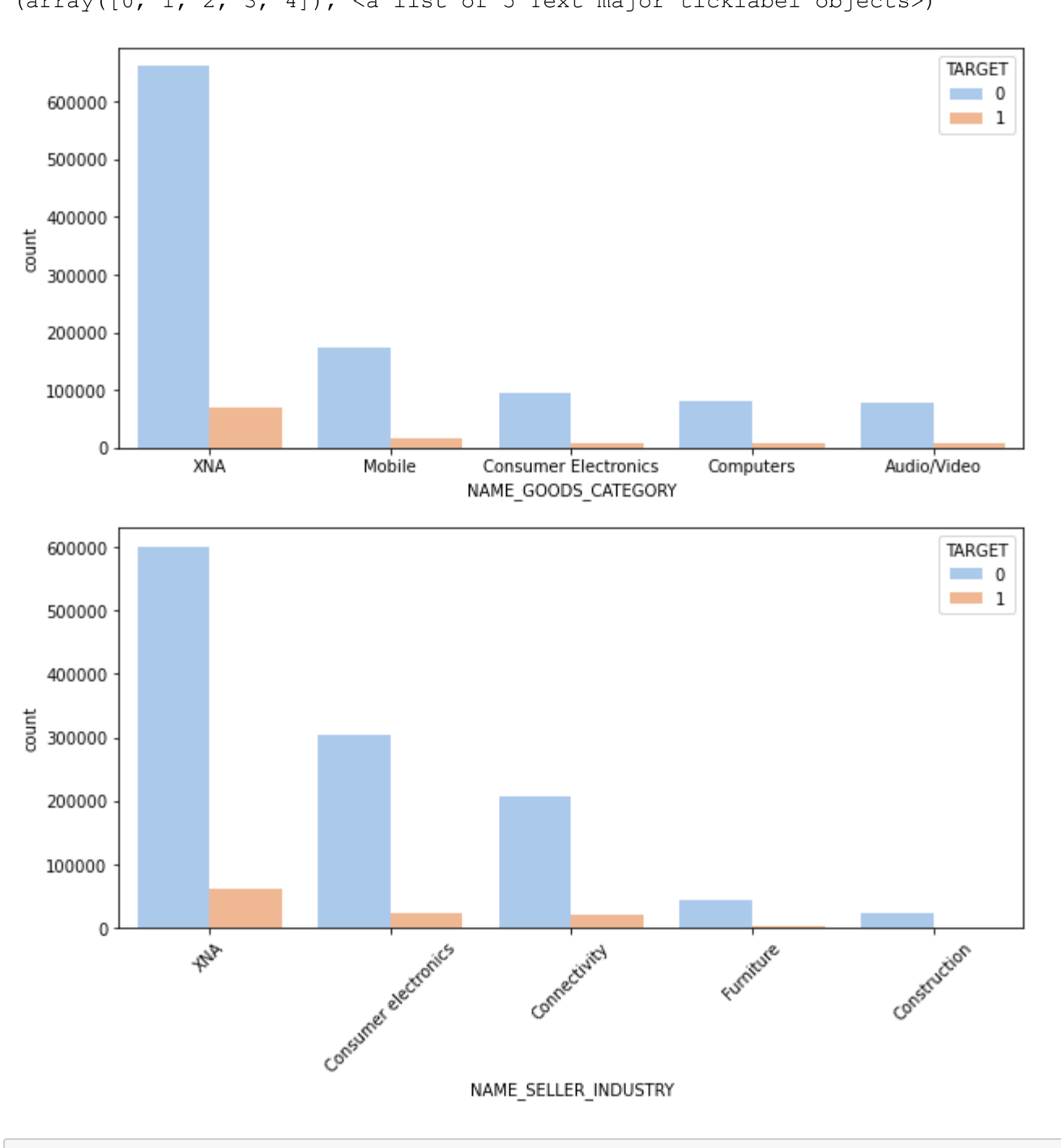
```
XAP :   52.294
XNA :   41.258
Other :   1.373
Payments on other loans :   0.21
Buying a used car :   0.258
Repairs :   2.194
Buying a new car :   0.066
Everyday expenses :   0.179
Medicine :   0.212
Car repairs :   0.102
Urgent needs :   0.914
Buying a holiday home / land :   0.046
Building a house or an annex :   0.272
Furniture :   0.071
Journey :   0.092
Purchase of electronic equipment :   0.096
Wedding / gift / holiday :   0.079
Education :   0.115
Buying a home :   0.071
Business development :   0.039
Gasification / water supply :   0.037
Buying a garage :   0.006
Hobby :   0.008
Money for a third person :   0.003
Refusal to name the goal :   0.003
```

In [73]:
```
for i in Final_Data["NAME_PAYMENT_TYPE"].unique():
    defaulters1 = Final_Data[(Final_Data["NAME_PAYMENT_TYPE"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))
```

```
XNA :   39.818
Cash through the bank :   59.625
Non-cash from your account :   0.494
Cashless from the account of the employer :   0.063
```

In [74]:
```
fig,axes =plt.subplots(2,1,figsize=(10,10))
sns.countplot(data =Final_Data,x="NAME_GOODS_CATEGORY",hue="TARGET",order=Final_Data["NAME_GOODS_CATEGO
RY"].value_counts().sort_values(ascending = False).iloc[:5].index,ax=axes[0],palette="pastel")
sns.countplot(data= Final_Data,x="NAME_SELLER_INDUSTRY",hue="TARGET",order=Final_Data["NAME_SELLER_INDU
STRY"].value_counts().sort_values(ascending = False).iloc[:5].index,ax=axes[1],palette="pastel")
plt.xticks(rotation=45)
```

Out[74]: (array([0, 1, 2, 3, 4]), <a list of 5 Text major ticklabel objects>)



In [75]:
```
for i in Final_Data["NAME_GOODS_CATEGORY"].unique():
    defaulters1 = Final_Data[(Final_Data["NAME_GOODS_CATEGORY"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))
```

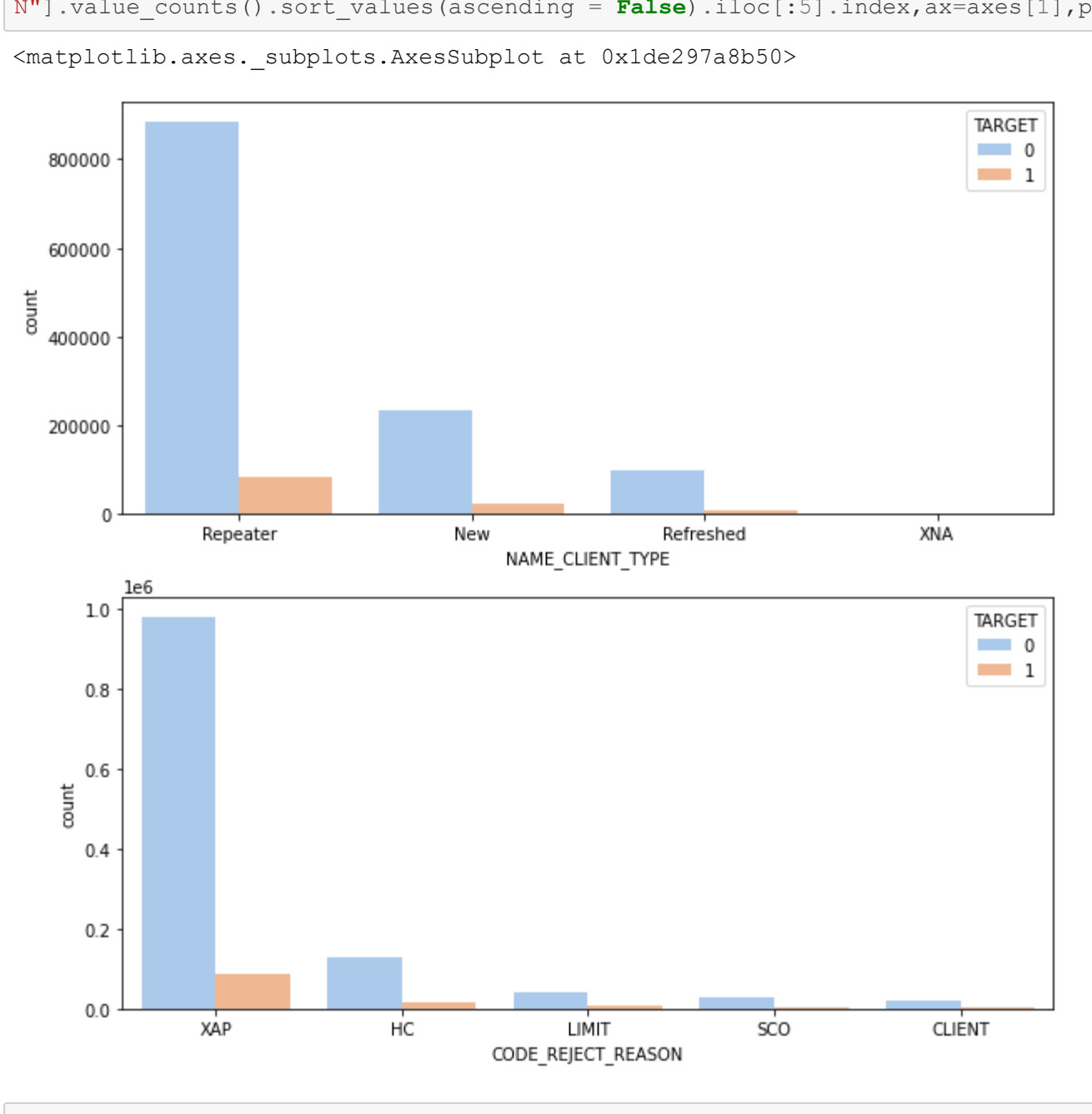
```
Vehicles :   0.244
XNA :   59.918
Furniture :   2.24
Consumer Electronics :   6.243
Mobile :   14.108
Audio/Video :   5.549
Construction Materials :   1.263
Gardening :   0.133
Photo / Cinema Equipment :   1.408
Computers :   6.127
Clothing and Accessories :   0.939
Homewares :   0.239
Medical Supplies :   0.15
Other :   0.114
Jewelry :   0.393
Office Appliances :   0.139
Tourism :   0.048
Auto Accessories :   0.478
Sport and Leisure :   0.155
Medicine :   0.065
Weapon :   0.004
Direct Sales :   0.021
Fitness :   0.006
Insurance :   0.005
Additional Service :   0.006
Education :   0.004
Animals :   0.0
```

In [76]:
```
for i in Final_Data["NAME_SELLER_INDUSTRY"].unique():
    defaulters1 = Final_Data[(Final_Data["NAME_SELLER_INDUSTRY"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))
```

```
Auto technology :   0.357
XNA :   54.08
Furniture :   2.507
Consumer electronics :   21.396
Connectivity :   18.069
Construction :   1.408
Clothing :   0.96
Industry :   1.005
Tourism :   0.014
Jewelry :   0.157
MLM partners :   0.047
```

In [77]:
```
fig,axes =plt.subplots(2,1,figsize=(10,10))
sns.countplot(data =Final_Data,x="NAME_CLIENT_TYPE",hue="TARGET",order=Final_Data["NAME_CLIENT_TYPE"].v
alue_counts().sort_values(ascending = False).iloc[:5].index,ax=axes[0],palette="pastel")
sns.countplot(data= Final_Data,x="CODE_REJECT_REASON",hue="TARGET",order=Final_Data["CODE_REJECT_REASO
N"].value_counts().sort_values(ascending = False).iloc[:5].index,ax=axes[1],palette="pastel")
```

Out[77]: <matplotlib.axes._subplots.AxesSubplot at 0x1de297a8b50>



In [78]:
```
for i in Final_Data["NAME_CLIENT_TYPE"].unique():
    defaulters1 = Final_Data[(Final_Data["NAME_CLIENT_TYPE"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))
```

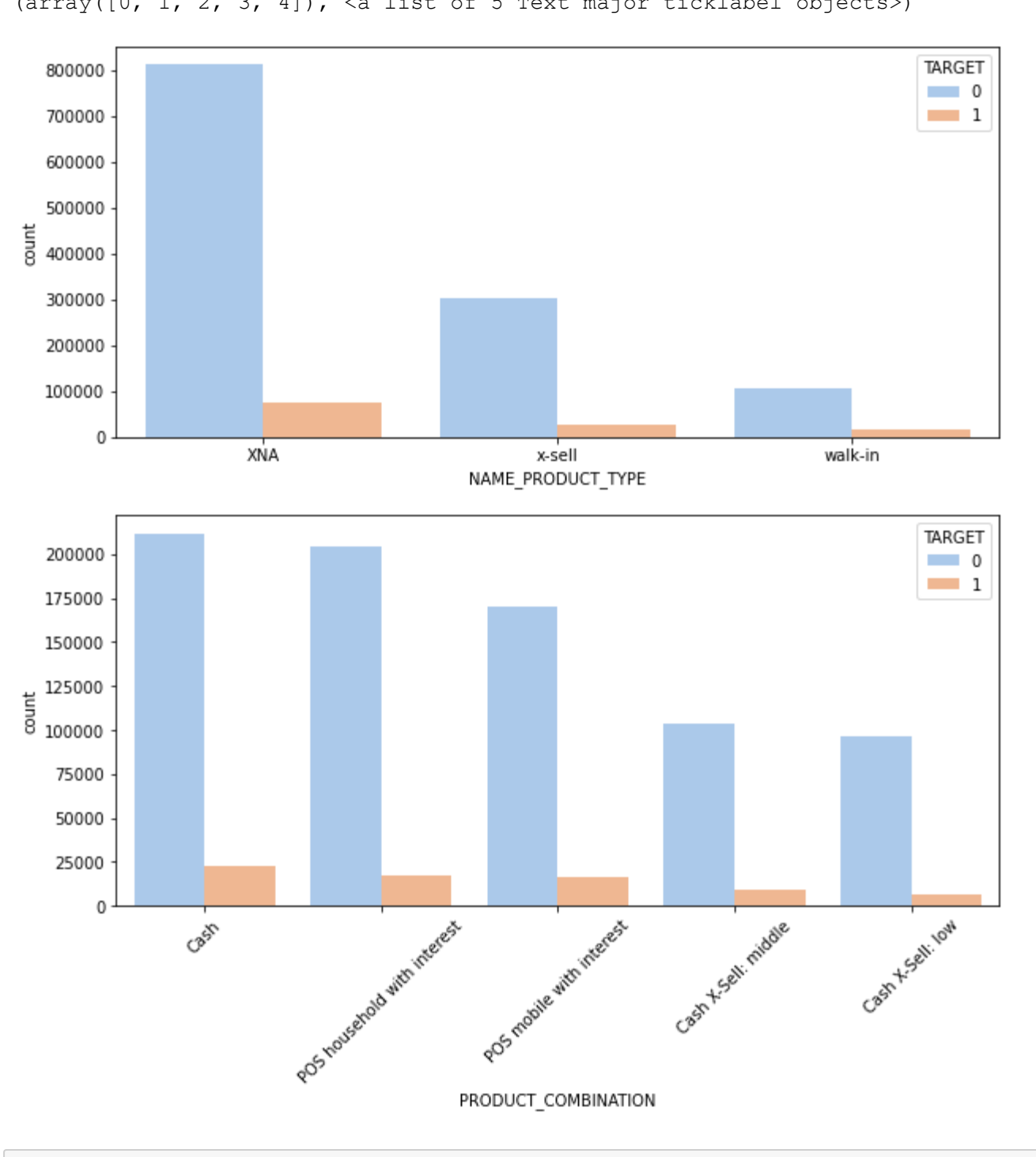
```
New :   19.776
Repeater :   73.319
Refreshed :   6.776
XNA :   0.129
```

In [79]:
```
for i in Final_Data["CODE_REJECT_REASON"].unique():
    defaulters1 = Final_Data[(Final_Data["CODE_REJECT_REASON"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))
```

```
XAP :   73.228
LIMIT :   5.134
HC :   14.635
SCO :   2.793
SCOFR :   1.946
VERIF :   0.245
CLIENT :   1.609
XNA :   0.375
SYSTEM :   0.036
```

In [80]:
```
fig,axes =plt.subplots(2,1,figsize=(10,10))
sns.countplot(data =Final_Data,x="NAME_PRODUCT_TYPE",hue="TARGET",order=Final_Data["NAME_PRODUCT_TYPE"]
.value_counts().sort_values(ascending = False).iloc[:5].index,ax=axes[0],palette="pastel")
sns.countplot(data= Final_Data,x="PRODUCT_COMBINATION",hue="TARGET",order=Final_Data["PRODUCT_COMBINATI
ON"].value_counts().sort_values(ascending = False).iloc[:5].index,ax=axes[1],palette="pastel")
plt.xticks(rotation=45)
```

Out[80]: (array([0, 1, 2, 3, 4]), <a list of 5 Text major ticklabel objects>)



In [81]:
```
for i in Final_Data["NAME_PRODUCT_TYPE"].unique():
    defaulters1 = Final_Data[(Final_Data["NAME_PRODUCT_TYPE"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))
```

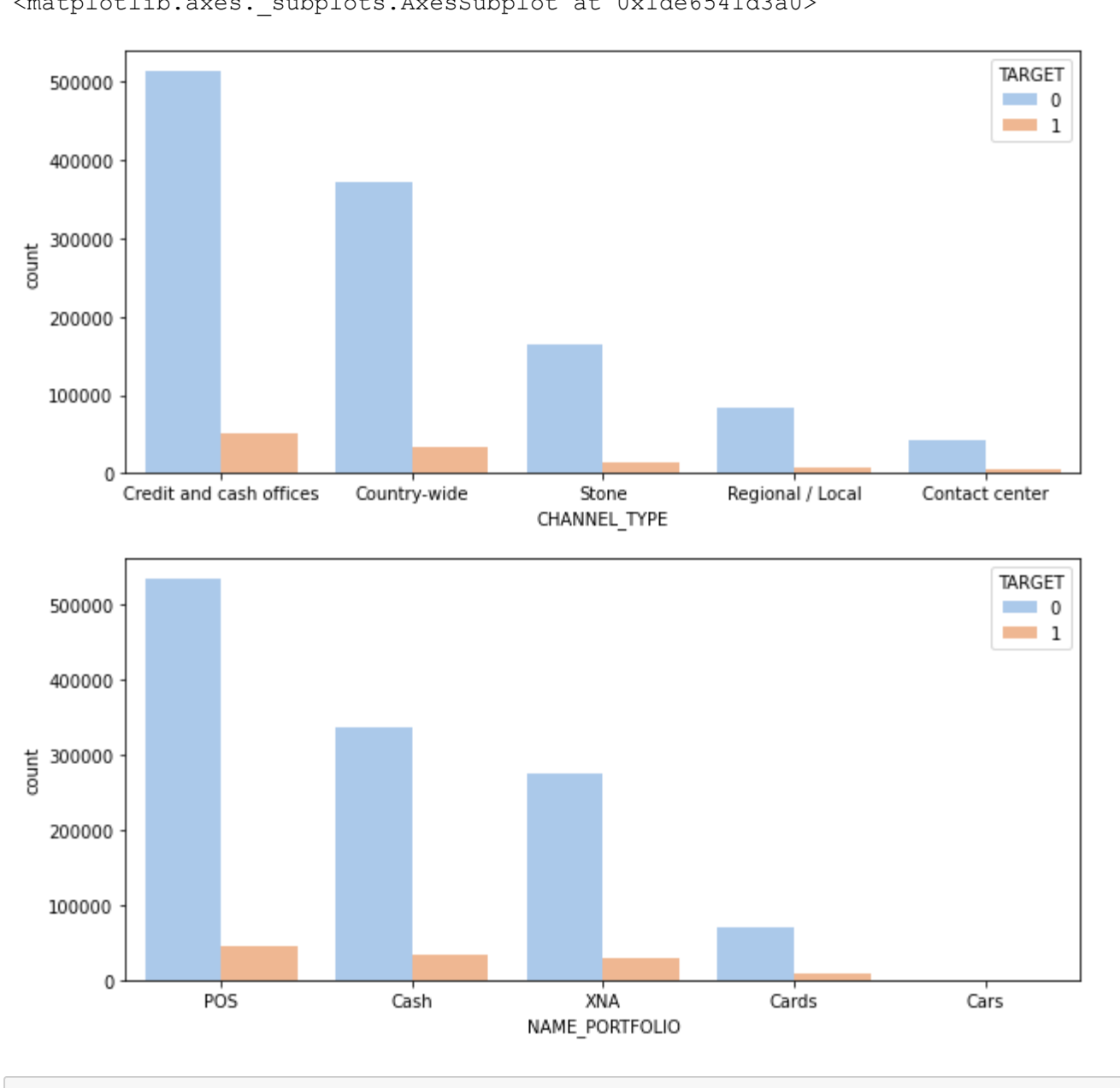
```
XNA :   63.715
x-sell :   23.138
walk-in :   13.147
```

In [82]:
```
for i in Final_Data["PRODUCT_COMBINATION"].unique():
    defaulters1 = Final_Data[(Final_Data["PRODUCT_COMBINATION"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))
```

```
POS other with interest :   1.39
Cash X-Sell: low :   5.931
POS industry with interest :   4.446
POS household with interest :   14.803
POS mobile without interest :   1.311
Card Street :   8.338
Card X-Sell :   3.309
Cash :   19.216
Cash Street: high :   4.884
Cash X-Sell: middle :   7.635
POS mobile with interest :   14.165
POS household without interest :   4.001
POS industry without interest :   0.406
Cash Street: low :   2.425
Cash X-Sell: high :   4.736
Cash Street: middle :   2.88
POS others without interest :   0.125
```

In [83]:
```
fig,axes =plt.subplots(2,1,figsize=(10,10))
sns.countplot(data =Final_Data,x="CHANNEL_TYPE",hue="TARGET",order=Final_Data["CHANNEL_TYPE"].value_cou
nts().sort_values(ascending = False).iloc[:5].index,ax=axes[0],palette="pastel")
sns.countplot(data= Final_Data,x="NAME_PORTFOLIO",hue="TARGET",order=Final_Data["NAME_PORTFOLIO"].value
_counts().sort_values(ascending = False).iloc[:5].index,ax=axes[1],palette="pastel")
```

Out[83]: <matplotlib.axes._subplots.AxesSubplot at 0x1de6541d3a0>



In [84]:
```
for i in Final_Data["CHANNEL_TYPE"].unique():
    defaulters1 = Final_Data[(Final_Data["CHANNEL_TYPE"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))
```

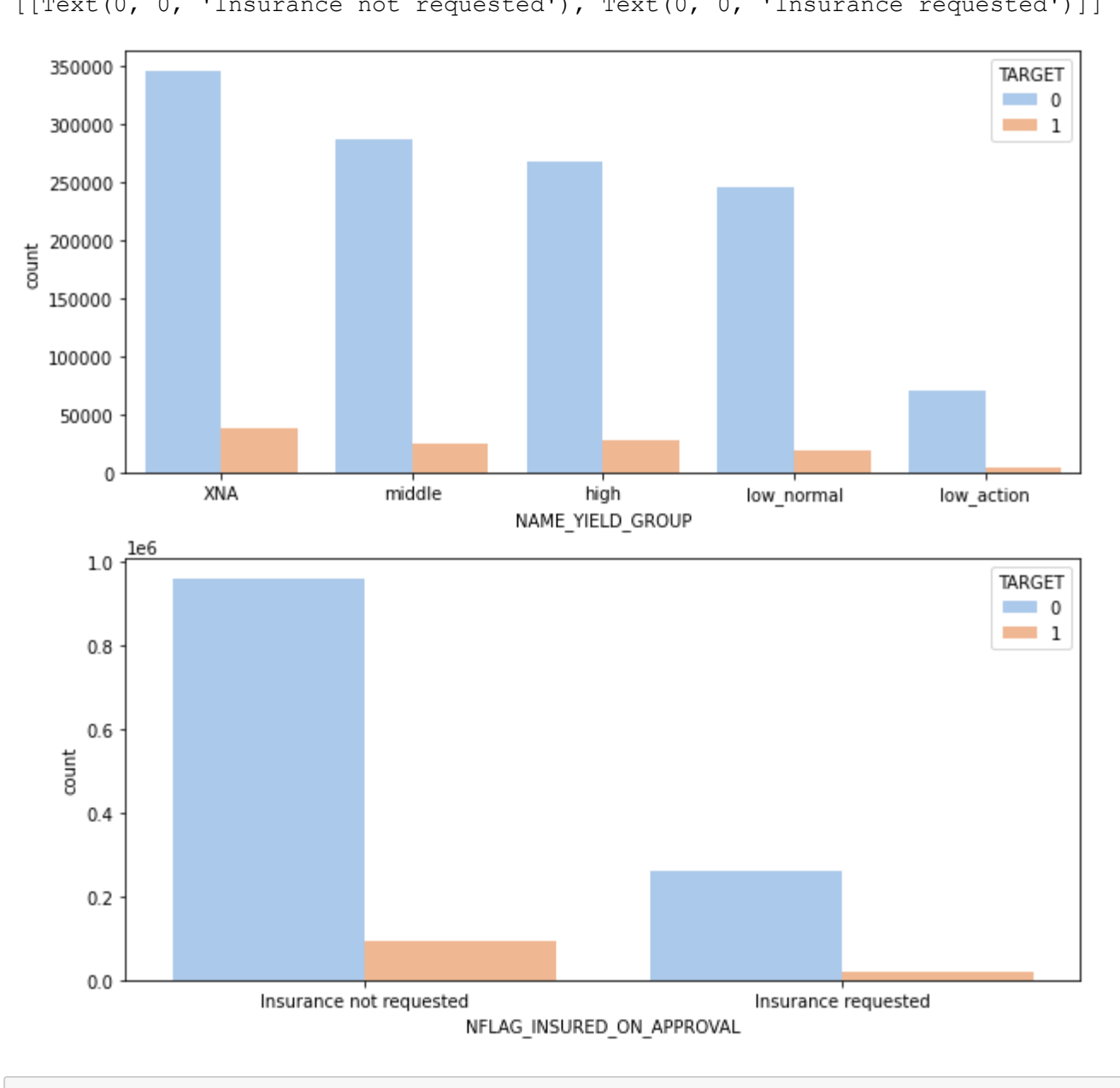
```
Stone :   11.327
Credit and cash offices :   44.256
Country-wide :   28.772
Regional / Local :   5.859
AP+ (Cash loan) :   5.106
Channel of corporate sales :   0.287
Contact center :   4.378
Car dealer :   0.015
```

In [85]:
```
for i in Final_Data["NAME_PORTFOLIO"].unique():
    defaulters1 = Final_Data[(Final_Data["NAME_PORTFOLIO"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))
```

```
POS :   38.112
Cash :   28.49
XNA :   25.588
Cards :   7.795
Cars :   0.015
```

In [86]:
```
fig,axes =plt.subplots(2,1,figsize=(10,10))
sns.countplot(data =Final_Data,x="NAME_YIELD_GROUP",hue="TARGET",order=Final_Data["NAME_YIELD_GROUP"].v
alue_counts().sort_values(ascending = False).iloc[:5].index,ax=axes[0],palette="pastel")
sns.countplot(data= Final_Data,x="NFLAG_INSURED_ON_APPROVAL",hue="TARGET",order=Final_Data["NFLAG_INSUR
ED_ON_APPROVAL"].value_counts().sort_values(ascending = False).iloc[:5].index,ax=axes[1],palette="paste
l").set(xticklabels=["Insurance not requested","Insurance requested"])
```

Out[86]: [[Text(0, 0, 'Insurance not requested'), Text(0, 0, 'Insurance requested')]]



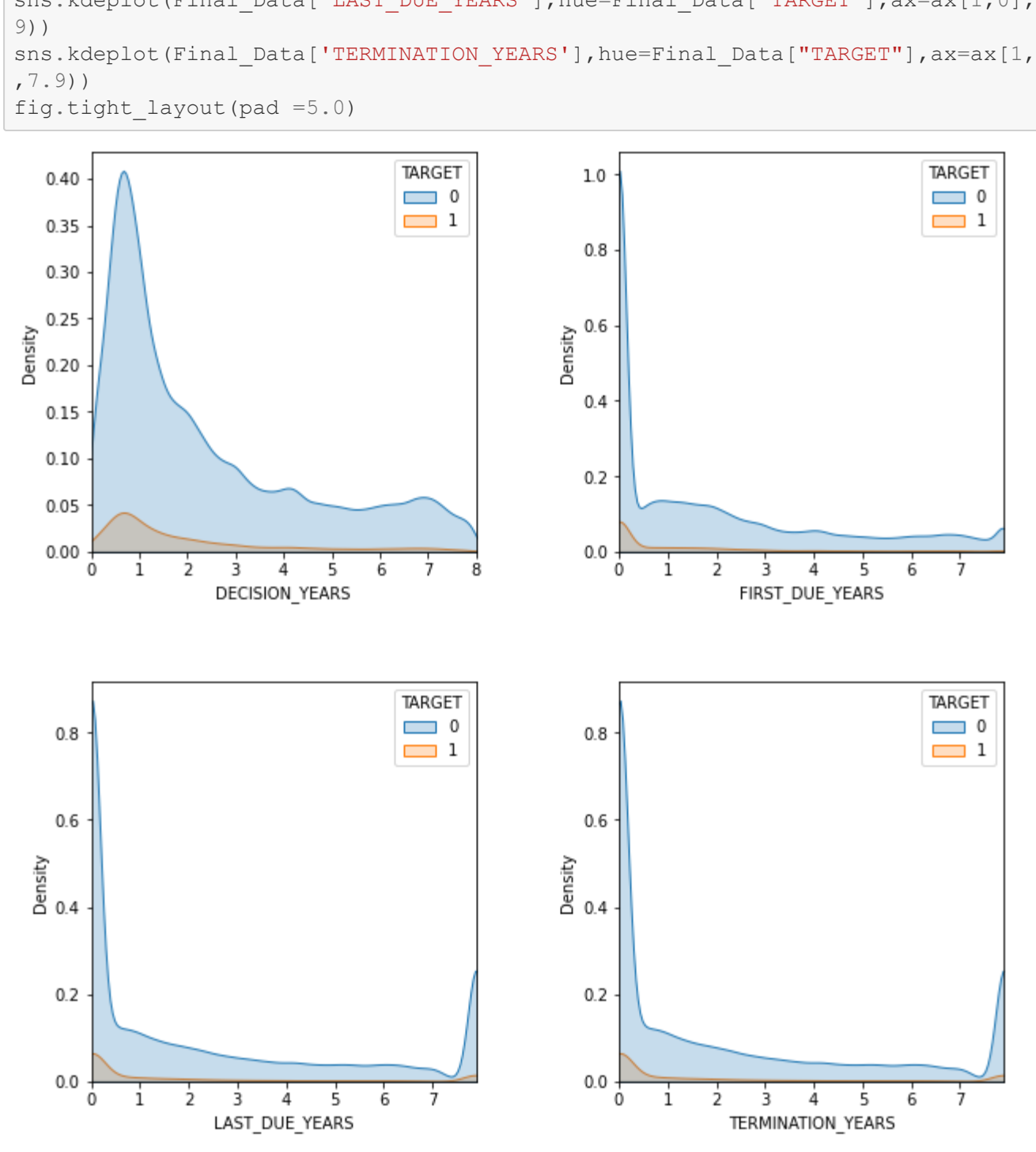
In [87]:
```
for i in Final_Data["NAME_YIELD_GROUP"].unique():
    defaulters1 = Final_Data[(Final_Data["NAME_YIELD_GROUP"] == i) & (Final_Data["TARGET"] ==1)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))
```

```
low_normal :   16.321
middle :   21.687
XNA :   33.383
high :   24.353
low_action :   4.256
```

In [88]:
```
for i in Final_Data["NFLAG_INSURED_ON_APPROVAL"].unique():
    defaulters1 = Final_Data[(Final_Data["NFLAG_INSURED_ON_APPROVAL"] == i) & (Final_Data["TARGET"] ==1
)]
    Total = len(Final_Data[Final_Data["TARGET"] ==1])
    print(i,":  ",round((len(defaulters1)/Total)*100,3))
```

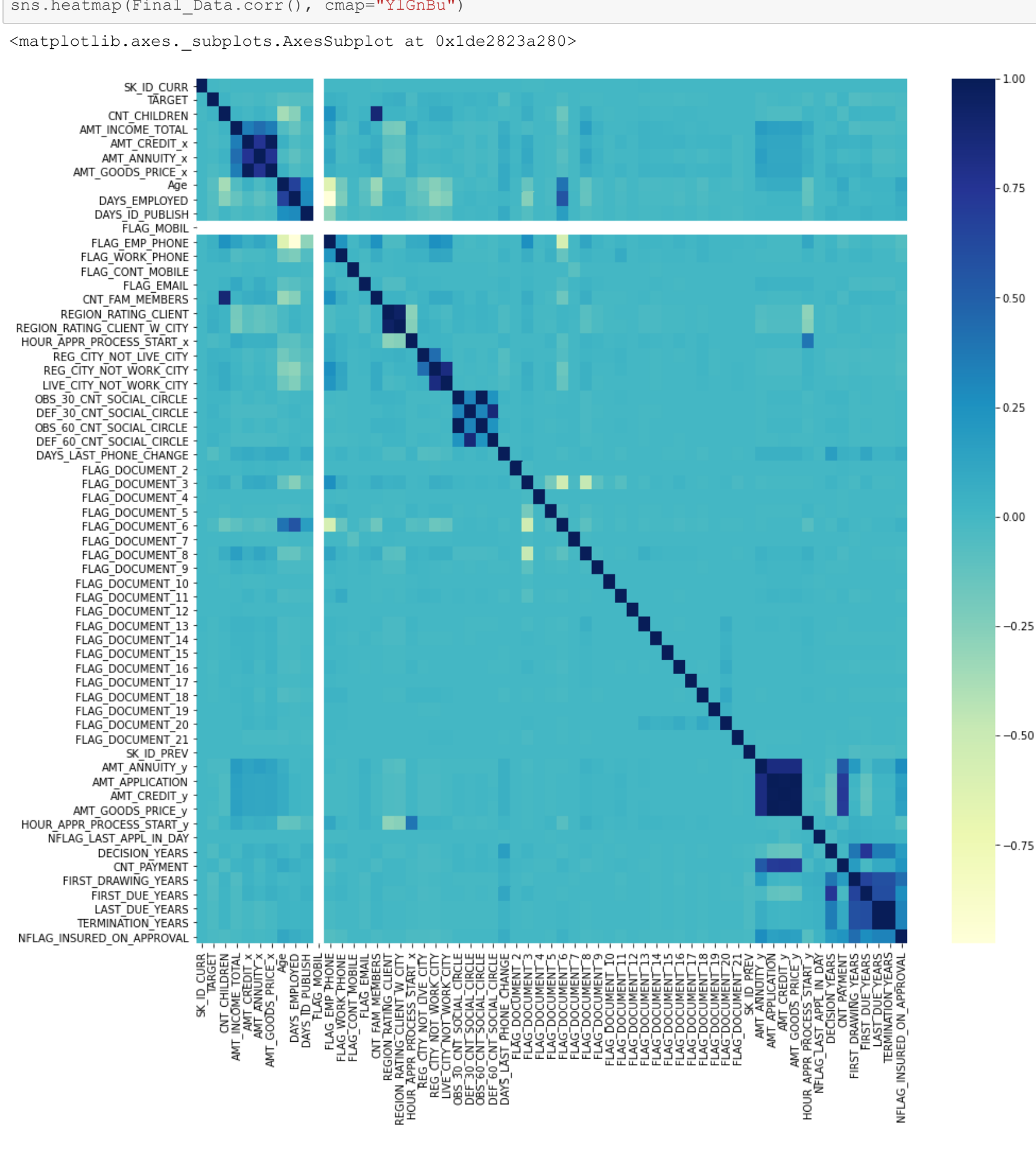
```
0.0 :   81.647
1.0 :   18.353
```

In [89]:
```
fig,ax =plt.subplots(2,2,figsize=(10,10))
sns.kdeplot(Final_Data['DECISION_YEARS'],hue=Final_Data["TARGET"],ax=ax[0,0],shade=True).set(xlim=(0,8.
0))
sns.kdeplot(Final_Data['FIRST_DUE_YEARS'],hue=Final_Data["TARGET"],ax=ax[0,1],shade=True).set(xlim=(0,
7.9))
sns.kdeplot(Final_Data['LAST_DUE_YEARS'],hue=Final_Data["TARGET"],ax=ax[1,0],shade=True).set(xlim=(0,7.
9))
sns.kdeplot(Final_Data['TERMINATION_YEARS'],hue=Final_Data["TARGET"],ax=ax[1,1],shade=True).set(xlim=(0
,7.9))
fig.tight_layout(pad =5.0)
```



In [90]:
```
plt.figure(figsize=(15,15))
sns.heatmap(Final_Data.corr(), cmap="YlGnBu")
```

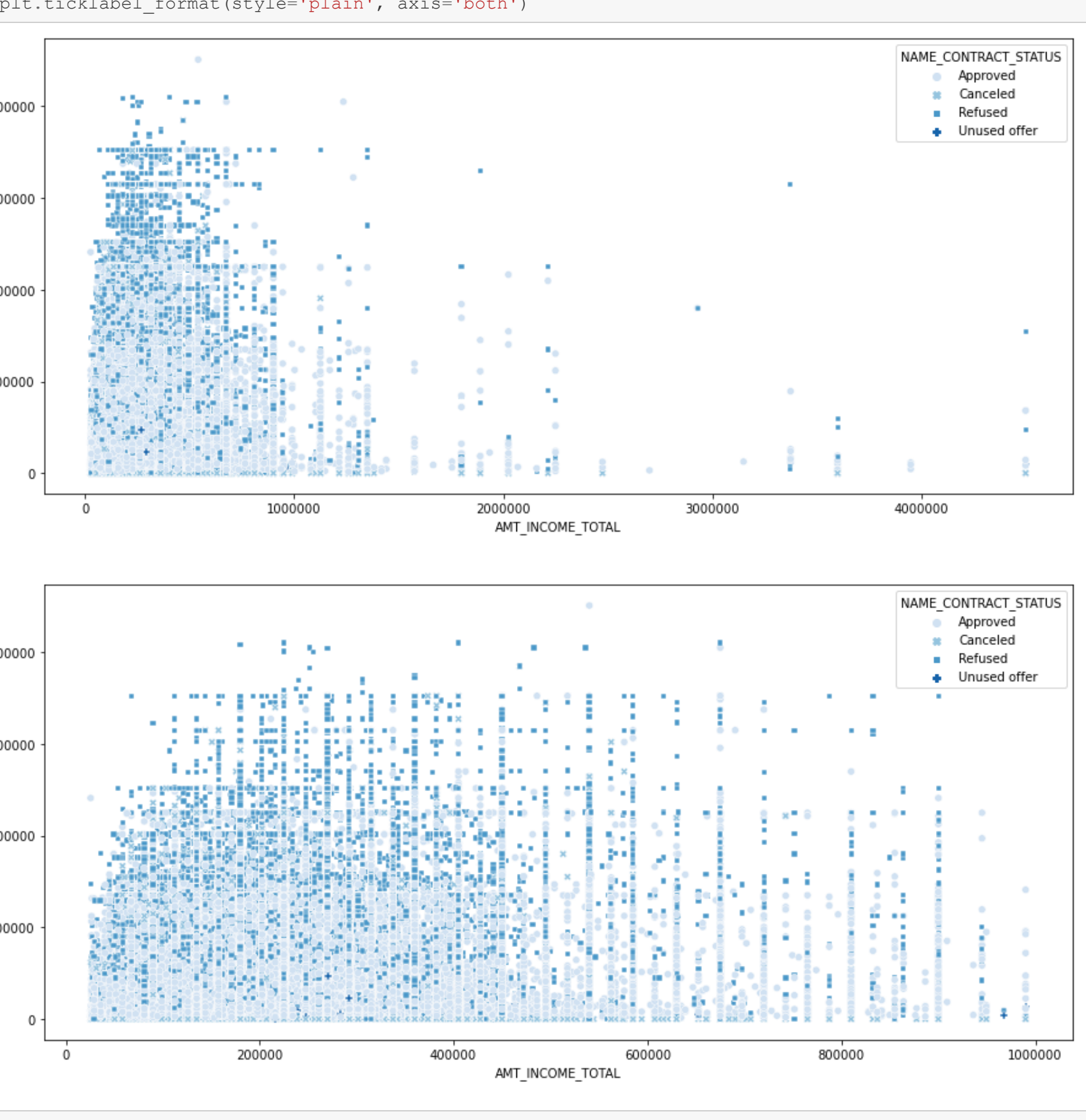
Out[90]: <matplotlib.axes._subplots.AxesSubplot at 0x1de2823a280>



Bi-Variate Analysis

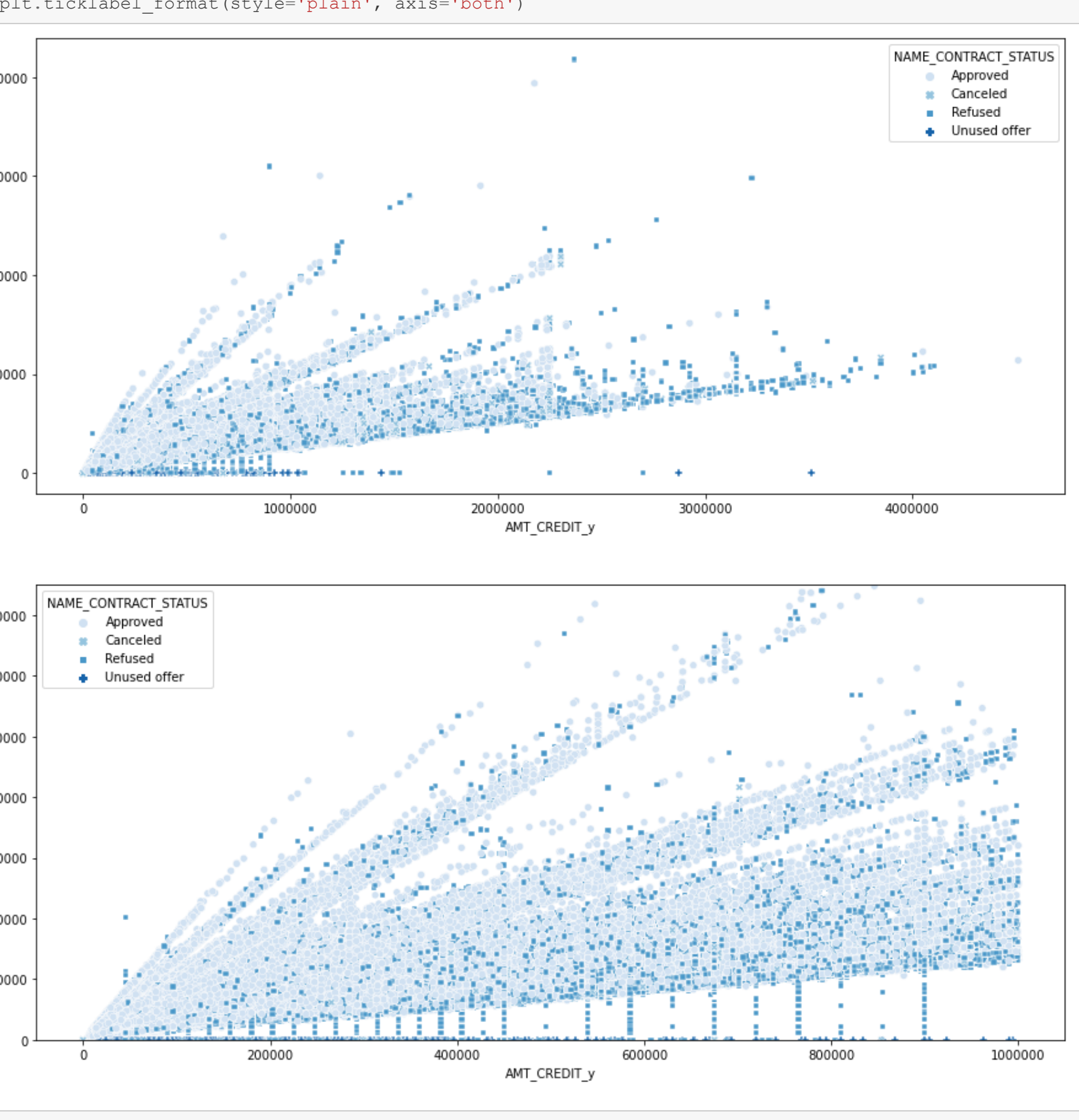
In [91]:
```
fig,ax =plt.subplots(2,1,figsize=(15,10))
sns.scatterplot(data=Final_Data, x="AMT_INCOME_TOTAL", y="AMT_CREDIT_y",hue ="TARGET",ax=ax[0],style="T
ARGET",palette="YlOrBr")
Inc =Final_Data.loc[Final_Data["AMT_INCOME_TOTAL"]<1000000,["AMT_INCOME_TOTAL","TARGET","AMT_CREDIT_y"
]]
sns.scatterplot(data=Inc, x="AMT_INCOME_TOTAL", y="AMT_CREDIT_y",hue ="TARGET",ax=ax[1],style="TARGET",
palette="YlOrBr")
for ax in fig.axes:
    plt.sca(ax)
    plt.ticklabel_format(style='plain', axis='both')
```

```
In [92]: fig,ax =plt.subplots(2,1,figsize=(15,15))
sns.scatterplot(data=Final_Data, x="AMT_INCOME_TOTAL", y="AMT_CREDIT_y",hue ="NAME_CONTRACT_STATUS",ax=ax[0],style="NAME_CONTRACT_STATUS",palette="Blues")
Inc =Final_Data.loc[Final_Data["AMT_INCOME_TOTAL"]<1000000,["AMT_INCOME_TOTAL","NAME_CONTRACT_STATUS","AMT_CREDIT_y"]]
sns.scatterplot(data=Inc, x="AMT_INCOME_TOTAL", y="AMT_CREDIT_y",hue ="NAME_CONTRACT_STATUS",ax=ax[1],style="NAME_CONTRACT_STATUS",palette="Blues")
for ax in fig.axes:
    plt.sca(ax)
    plt.ticklabel_format(style='plain', axis='both')
```

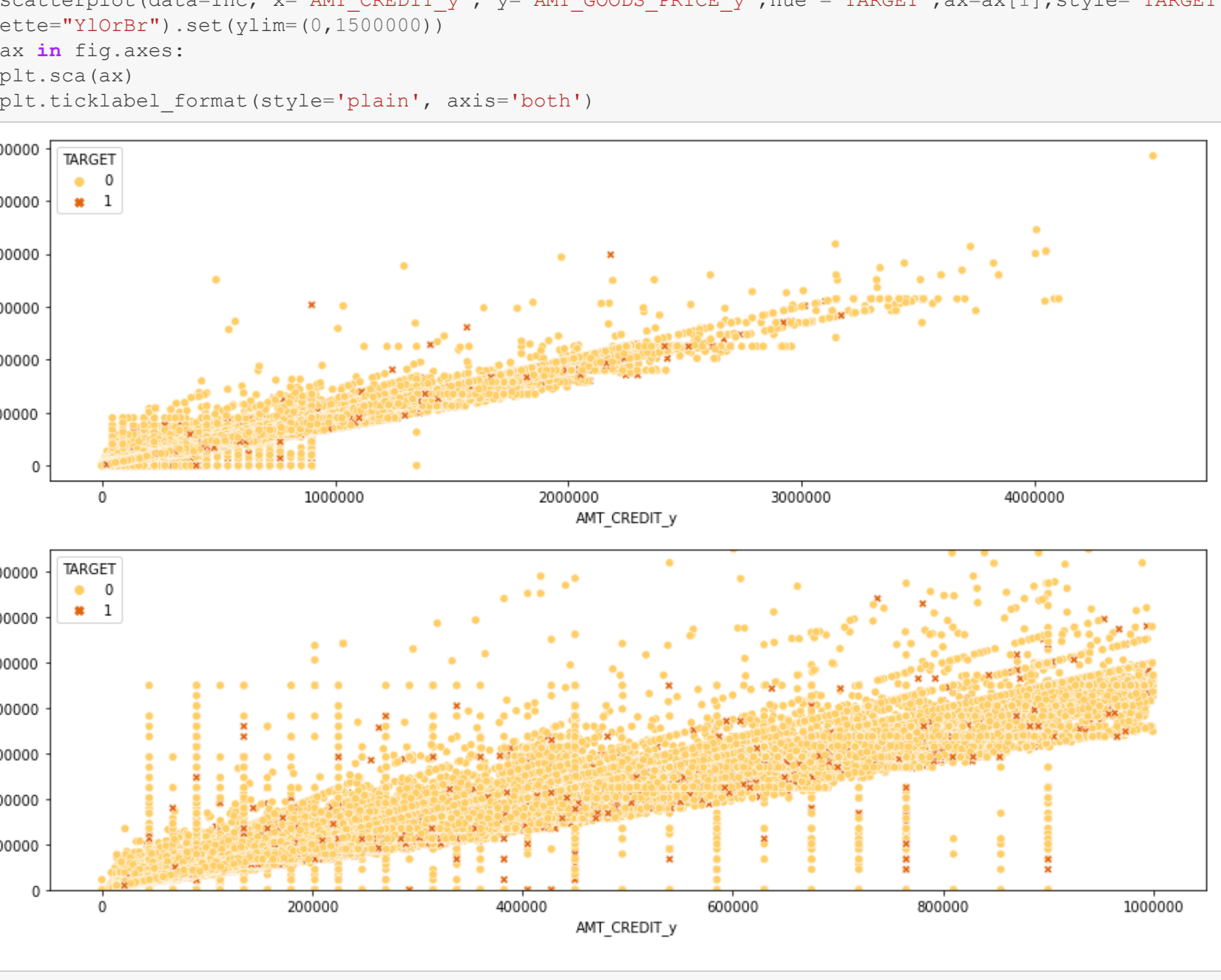



```
In [93]: fig,ax =plt.subplots(2,1,figsize=(15,10))
sns.scatterplot(data=Final_Data, x="AMT_CREDIT_y", y="AMT_ANNUITY_y",hue ="TARGET",ax=ax[0],style="TARGET",palette="YlOrBr")
Inc =Final_Data.loc[Final_Data["AMT_CREDIT_y"]<1000000,["AMT_CREDIT_y","TARGET","AMT_ANNUITY_y"]]
sns.scatterplot(data=Inc, x="AMT_CREDIT_y", y="AMT_ANNUITY_y",hue ="TARGET",ax=ax[1],style="TARGET",palette="YlOrBr").set(ylim=(0,150000))
for ax in fig.axes:
    plt.sca(ax)
    plt.ticklabel_format(style='plain', axis='both')
```

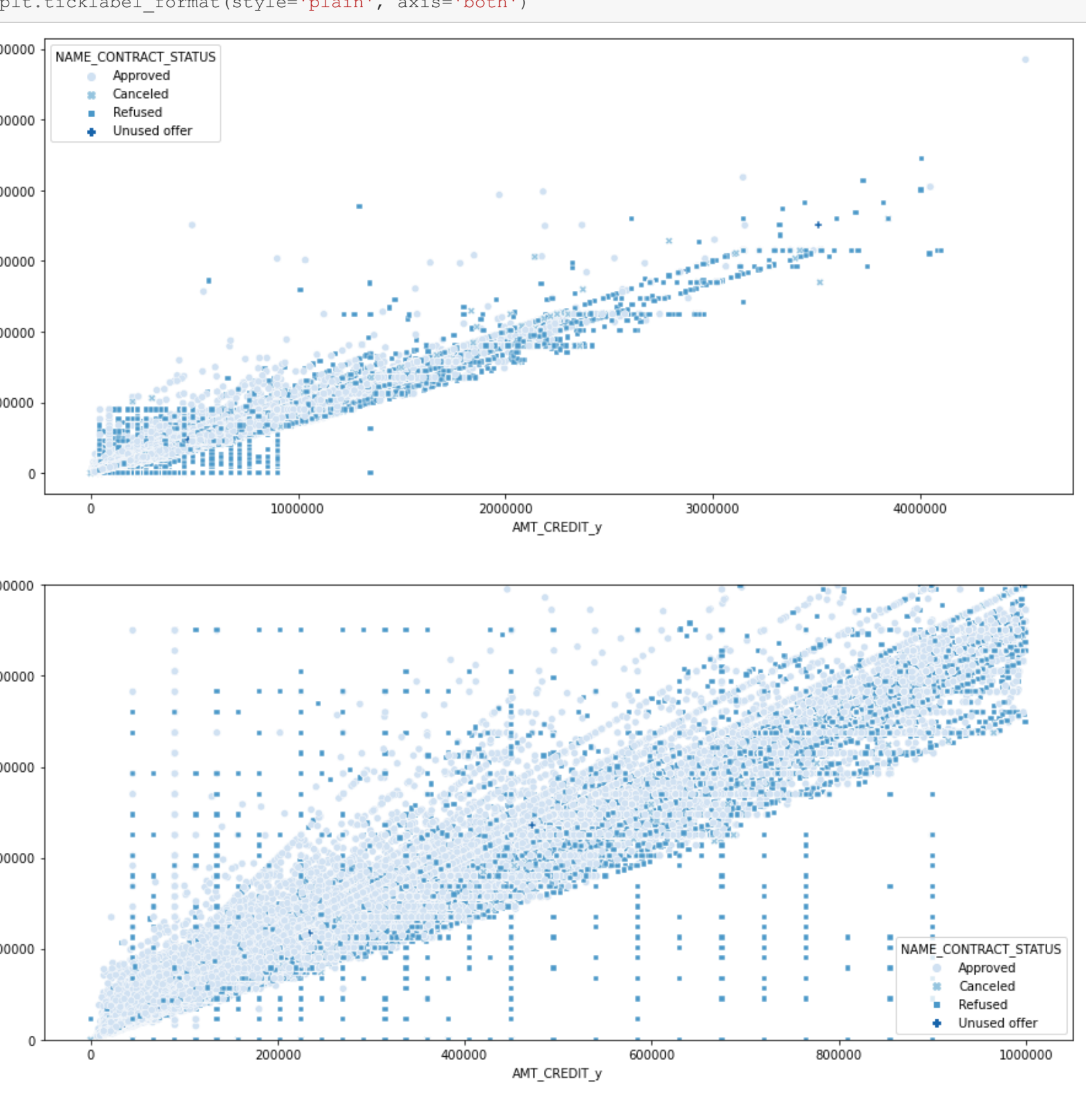

```
In [94]: fig,ax =plt.subplots(2,1,figsize=(15,15))
sns.scatterplot(data=Final_Data, x="AMT_CREDIT_y", y="AMT_ANNUITY_y",hue ="NAME_CONTRACT_STATUS",ax=ax[0],style="NAME_CONTRACT_STATUS",palette="Blues")
Inc =Final_Data.loc[Final_Data["AMT_CREDIT_y"]<1000000,["AMT_ANNUITY_y","NAME_CONTRACT_STATUS","AMT_CREDIT_y"]]
sns.scatterplot(data=Inc, x="AMT_CREDIT_y", y="AMT_ANNUITY_y",hue ="NAME_CONTRACT_STATUS",ax=ax[1],style="NAME_CONTRACT_STATUS",palette="Blues").set(ylim=(0,150000))
for ax in fig.axes:
    plt.sca(ax)
    plt.ticklabel_format(style='plain', axis='both')
```




```
In [95]: fig,ax =plt.subplots(2,1,figsize=(15,10))
sns.scatterplot(data=Final_Data, x="AMT_CREDIT_y", y="AMT_GOODS_PRICE_y",hue ="TARGET",ax=ax[0],style="TARGET",palette="YlOrBr")
Inc =Final_Data.loc[Final_Data["AMT_CREDIT_y"]<1000000,["AMT_CREDIT_y","TARGET","AMT_GOODS_PRICE_y"]]
sns.scatterplot(data=Inc, x="AMT_CREDIT_y", y="AMT_GOODS_PRICE_y",hue ="TARGET",ax=ax[1],style="TARGET",palette="YlOrBr").set(ylim=(0,1500000))
for ax in fig.axes:
    plt.sca(ax)
    plt.ticklabel_format(style='plain', axis='both')
```




```
In [96]: fig,ax =plt.subplots(2,1,figsize=(15,15))
sns.scatterplot(data=Final_Data, x="AMT_CREDIT_y", y="AMT_GOODS_PRICE_y",hue ="NAME_CONTRACT_STATUS",ax=ax[0],style="NAME_CONTRACT_STATUS",palette="Blues")
Inc =Final_Data.loc[Final_Data["AMT_CREDIT_y"]<1000000,["AMT_GOODS_PRICE_y","NAME_CONTRACT_STATUS","AMT_CREDIT_y"]]
sns.scatterplot(data=Inc, x="AMT_CREDIT_y", y="AMT_GOODS_PRICE_y",hue ="NAME_CONTRACT_STATUS",ax=ax[1],style="NAME_CONTRACT_STATUS",palette="Blues").set(ylim=(0,1000000))
for ax in fig.axes:
    plt.sca(ax)
    plt.ticklabel_format(style='plain', axis='both')
```



Conclusion:

9% were Defaulters out of 1.3 Million Applications.

Data_Discrepancies:

- Lot of columns has 'XAP', 'XNA' other unique codes didnt find any particular meaning for it, But our Analysis Continued with those deafult values.
- The columns with Time line data has 1000 years as maximum which was imputed with 2nd highest value in columns as removing those columns will reduce the record count for our analysis.
- 4 Columns has missing data of about 50%, as the values are dynamic they were removed from our anlysis.

Findings:

- Highest Applicants were from Consumer loans & Cash loans but Defaulters were quite high from Cash loan applicants.
- Highest risk with applicants whose loan application were Canceled & Refused in past.
- Refused Applicants were high in the income range below 0.4 Millions for the credited amount below 2.5 Million.Low priority should be given to the customers who refused offers more than 2 times.
- Payment dificulties the most deafulters fall in the incomerange below 0.4Millions.
- Most of the applicants requested another loan within the span of 1 year, which is a risky factor.
- Highest Risk with Applicants who requested loan for Everyday expenses,Mobile,Refusal to name the goal and Medicine.
- Defaulters were high in Cash,POS mobile with interest,POS household with interest and Card Street.
- Applicants who obtained loan through Cash,POS were facing difficulty in paying.
- Highest deafuters were in insurance requested applicants rather than non-insured.
- Many clients were from Stone,Credit and cash offices,Country-wide Channels.

```
In [97]: Final_Data.to_csv("Final_data.csv",index=False)
```

```
In [ ]:
```