

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

In [2]: os.chdir("D:\Lohith\DA\HansonRisk_Analytics")

In [3]: Client_Data = pd.read_csv("application_data.csv") #Reading the Client data
print(Client_Data.is_read_csv("application_data.csv", encoding="cp1252"))

By default it will show only 20 columns, since we have more than 20 columns, this function is used

In [4]: pd.set_option('display.max_columns','None', 'display.max_row','None')

In [5]: Description.drop(columns="Unnamed: 0", axis =1, inplace = True) # Description of our Columns
Description[1:22]

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 diff... 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 loan NaN
8 application_data AMT_CREDIT Credit amount of the client 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 (... NaN
16 application_data REGION_POPULATION_RELATIVE Normalized population of region where client li... normalized
17 application_data DAYS_BIRTH Clients 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 client... time only relative to the application
20 application_data DAYS_ID_PUBLISH How many days before the application did client... time only relative to the application
21 application_data OWN_CAR_AGE Age of clients 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 region where client lives (1... NaN
31 application_data HOUR_APPR_PROCESS_START Our rating of the region where client lives w... 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 apply... 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_BEGINEXPLUATION_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 FLOORMIN_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_BEGINEXPLUATION_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 FLOORMIN_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_BEGINEXPLUATION_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 FLOORMIN_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 FONDAPPRMOT_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 EMERGENCYSITATE_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 did client call 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_INCOI
0 100002 1 Cash loans M N Y 0
1 100003 0 Cash loans F N Y 0
2 100004 0 Revolving loans M Y N 0
3 100006 0 Cash loans F N Y 0
4 100007 0 Cash loans M N Y 0

In [7]: print(Client_Data.shape)

Client_Data: (307511, 122)

In [8]: mmo=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]: LIVE_REGION,"REG_REGION_NOT_WORK_REGION","LIVE_REGION_NOT_WORK_REGION","OCCUPATION_TYPE","AMT_REQ_CRE
DIT_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(cthresh=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 other columns."OCCUPATION_TYPE" can be filled by comparing data
• But as we can get insights directly from "NAME_INCOME_TYPE" which is highly depandent,we have dropped "OCCUPATION_TYPE".

In [12]: print(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
HOUR_APPR_PROCESS_START 0
WEEKDAY_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 object
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
HOUR_APPR_PROCESS_START int64
WEEKDAY_APPR_PROCESS_START object
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 int64
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"] = Client_Data["CODE_GENDER","DAYS_LAST_PHONE_CHANGE","DAYS_BIRTH","DAYS_EMPLOYED","DAYS_ID_PUBLISH","AMT
INCOME_TOTAL"].abs()

Out[18]:
CODE_GENDER DAYS_LAST_PHONE_CHANGE DAYS_BIRTH DAYS_EMPLOYED DAYS_ID_PUBLISH AMT_INCOME_TOTAL
0 M 1138.0 1461 637 2120 20290.0
1 F 828.0 16765 168 291 27000.0
2 M 816.0 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]:
count 305185.000000 305185.000000 305185.000000
mean 43.147825 185.592190 4.132549
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"] = Client_Data["DAYS_ID_PUBLISH"].replace(to_replace=0,value=0.1,inplace=True)
Client_Data["DAYS_EMPLOYED"] = Client_Data["DAYS_EMPLOYED"].replace(to_replace=1000,value=70,inplace=True)

Checking 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 5.992185e+04
min 2.376652e+05 4.020170e+05
std 2.565000e+04 4.500000e+04
25% 1.125000e+05 2.700000e+05
50% 1.458000e+05 5.147750e+05
75% 2.025000e+05 8.086500e+05
max 1.170000e+06 4.050000e+06

In [24]: Client_Data.loc[Client_Data["AMT_INCOME_TOTAL"]>600000,["TARGET","AMT_INCOME_TOTAL","AMT_CREDIT"]].hea
d(10)

Out[24]:
TARGET AMT_INCOME_TOTAL AMT_CREDIT
12840 1 117000000.0 562491.0
77768 0 9000000.0 143153.0
131127 0 675000.0 790830.0
203693 0 18000090.0 675000.0
246555 0 1350000.0 1405933.5

• Income Data has an outlier between 6 and 120 Millions, But the "Amt_Credit"(Loan amount provided) has 1Million Maximum and
• As "Analysis" gets effected 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"]>600000,["TARGET","AMT_INCOME_TOTAL","AMT_CREDIT"]].hea
d(10)

Out[26]:
TARGET AMT_INCOME_TOTAL AMT_CREDIT
12840 1 117000000.0 562491.0
77768 0 9000000.0 143153.0
131127 0 675000.0 790830.0
203693 0 18000090.0 675000.0
246555 0 1350000.0 1405933.5

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
mean 278164.964965 0.069095 4.417147 1.680786e+05 5.992175e+05 27135.854142 5.382210e+05
std 102784.507591 0.272688 0.722266 9.867711e+04 4.020147e+05 14473.432700 3.688252e+05
min 100276.800000 0.000000 0.000000 2.565000e+04 4.500000e+04 1615.500000 4.050000e+04
25% 189128.750000 0.000000 0.000000 1.125000e+05 2.700000e+05 16573.500000 2.385000e+05
50% 278169.500000 0.000000 0.000000 1.458000e+05 5.147750e+05 34619.000000 4.500000e+05
75% 367120.000000 0.000000 1.000000 2.025000e+05 8.086500e+05 34614.000000 6.785000e+05
max 456255.250000 1.000000 19.000000 4.500000e+06 4.050000e+06 258025.500000 4.050000e+06

Univariate Analysis

In [28]: sns.countplot(Client_Data["TARGET"],palette="YlOrBr").set(sticklabels=["Non Defaulters","Defaulters"])
print("Defaulters: (1) : ",Client_Data["TARGET"].value_counts()[1],",",Client_Data["TARGET"].value_counts
(0))
print("Non-Defaulters (0): ",Client_Data["TARGET"].value_counts()[0],",",Client_Data["TARGET"].value_co
unts()[1])
print("Defaulters: (1) : 24718 , 9.09482272757061 %
Non-Defaulters(0) : 280462 , 91.9051772724294 %

In [29]: Document_Status.head(10)
Document_Status[19:24]
plt.figure(figsize = (10,5))
sns.boxplot(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]),
<list of 20 Text major ticklabel objects>)

In [30]: 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 highly neglected in all Applications.

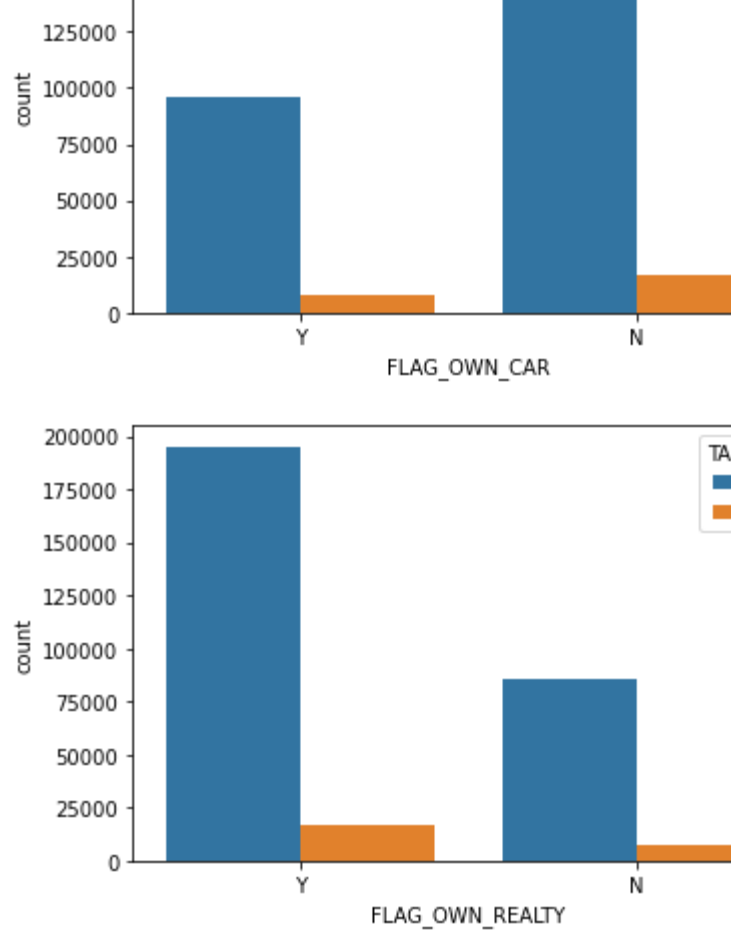
Asset Status

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 highly neglected in all Applications.


```
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"]]:
defaulters = 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))

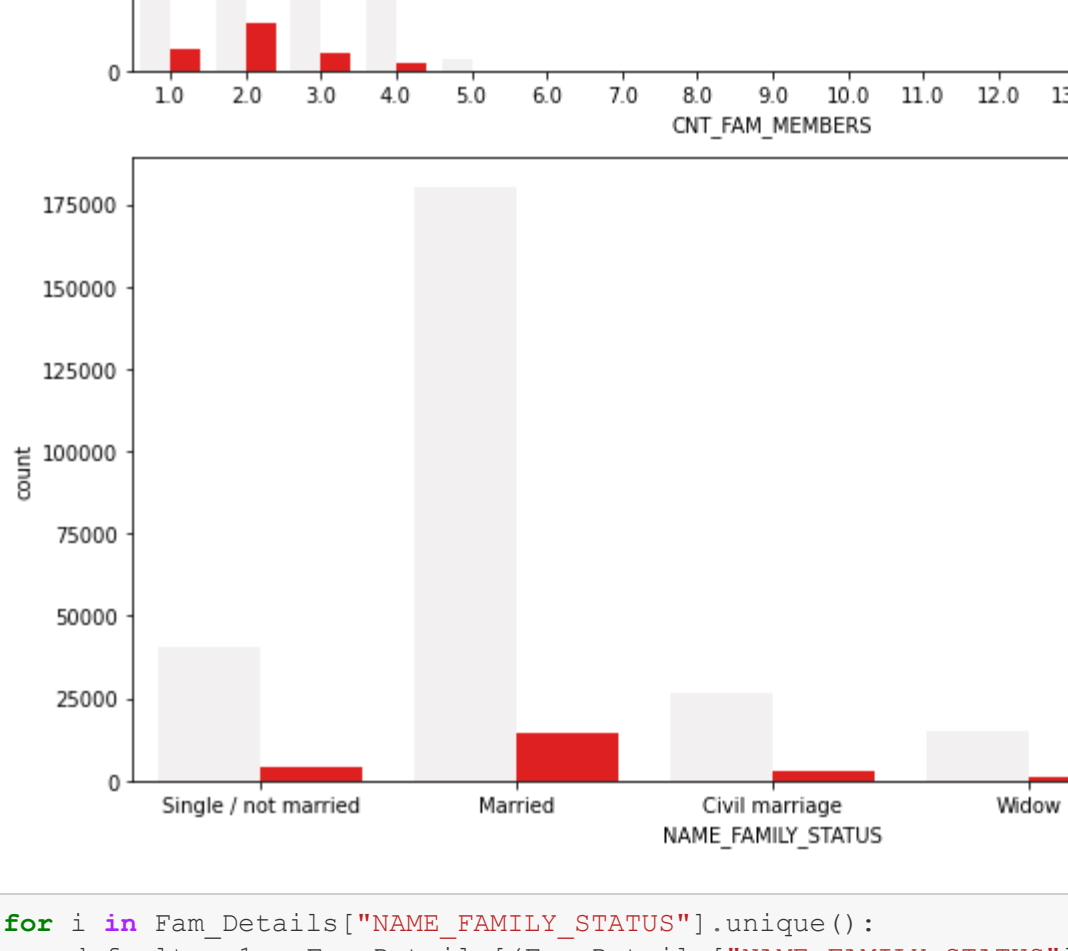
FLAG_OWN_CAR (H/C): 30.532
FLAG_OWN_CAR (NH/NC): 69.468
FLAG_OWN_REALTY (H/C): 68.452
FLAG_OWN_REALTY (NH/NC): 31.548
```

- 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),)
defaulters1 = Fam_Details[(Fam_Details["NAME_CONTRACT_TYPE"] == "TARGET",ax=axes[0])
sns.countplot(data =Client_Data,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)]
defaulters2 = Fam_Details[(Fam_Details["NAME_CONTRACT_TYPE"] == i) & (Client_Data["TARGET"] ==0)]
Total = len(Client_Data[Client_Data["TARGET"] ==1])
print(i," ",(len(defaulters1)/Total)*100))

Cash loans : 93.745486608949
Revolving loans : 6.25455139105105
```

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

```
In [37]: fig,axes=plt.subplots(2,1,figsize=(10,10))
sns.countplot(data =Client_Data,x="NAME_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)]
defaulters2 = Fam_Details[(Fam_Details["NAME_FAMILY_STATUS"] == i) & (Client_Data["TARGET"] ==0)]
Total = len(Client_Data[Client_Data["TARGET"] ==1])
print(i," ",(len(defaulters1)/Total)*100,3))

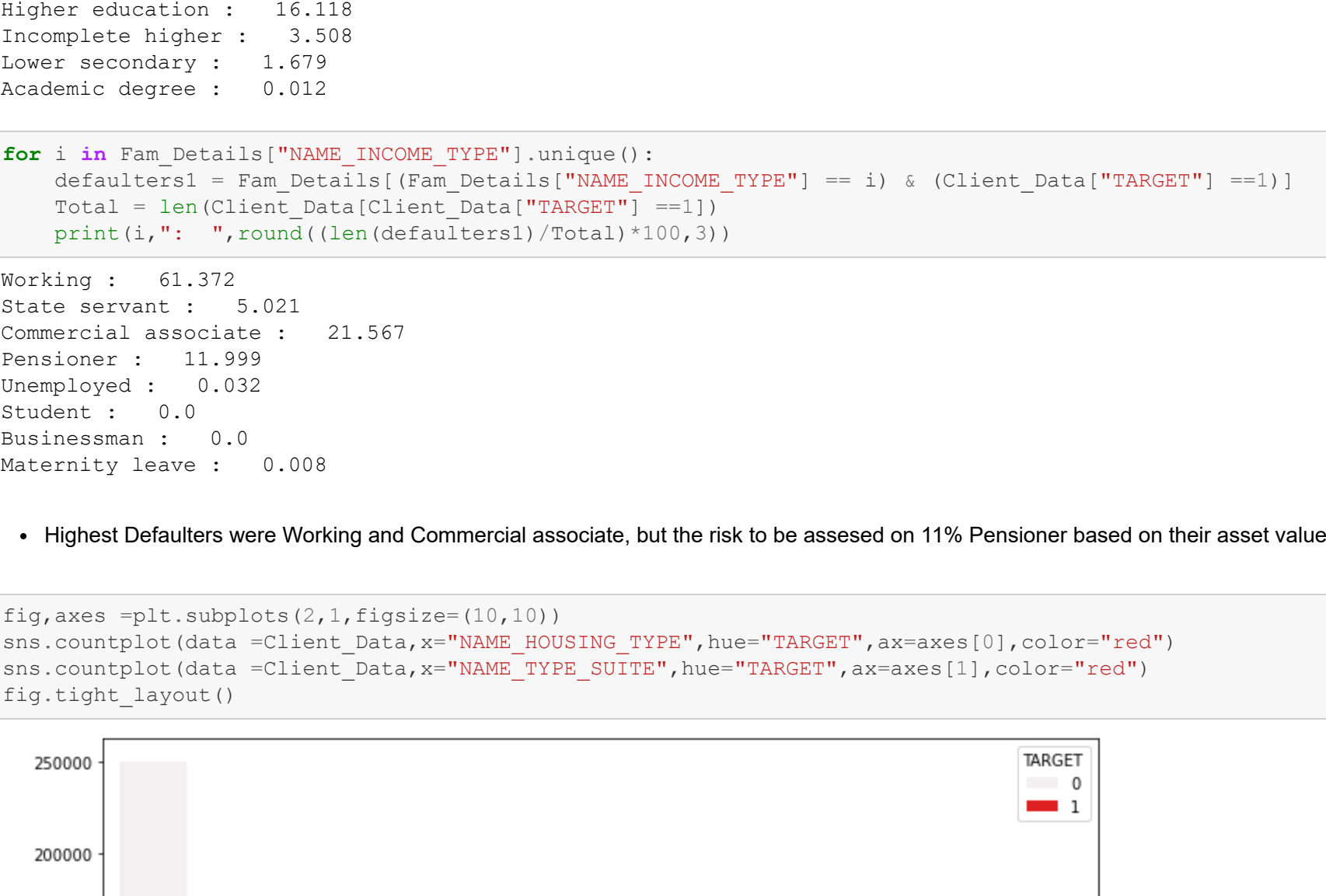
Single / not married : 17.967
Married : 59.815
Civil marriage : 11.922
Widow : 3.772
Separated : 6.518
```

```
In [39]: for i in Fam_Details[["CNT_FAM_MEMBERS"]].unique():
defaulters1 = Fam_Details[(Fam_Details["CNT_FAM_MEMBERS"] == i) & (Client_Data["TARGET"] ==1)]
defaulters2 = Fam_Details[(Fam_Details["CNT_FAM_MEMBERS"] == i) & (Client_Data["TARGET"] ==0)]
Total = len(Client_Data[Client_Data["TARGET"] ==1])
print(i," ",(len(defaulters1)/Total)*100,3))

1.0 : 22.87
2.0 : 48.357
3.0 : 18.357
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
```

- Even though highest Defaulters lie around Married, but risk factor is high with ~5 - ~10% Single,Civil marriages and Separated
- 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")
fig.tight_layout()
```



```
In [41]: for i in Fam_Details[["NAME_EDUCATION_TYPE"]].unique():
defaulters1 = Fam_Details[(Fam_Details["NAME_EDUCATION_TYPE"] == i) & (Client_Data["TARGET"] ==1)]
defaulters2 = Fam_Details[(Fam_Details["NAME_EDUCATION_TYPE"] == i) & (Client_Data["TARGET"] ==0)]
Total = len(Client_Data[Client_Data["TARGET"] ==1])
print(i," ",(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)]
defaulters2 = Fam_Details[(Fam_Details["NAME_INCOME_TYPE"] == i) & (Client_Data["TARGET"] ==0)]
Total = len(Client_Data[Client_Data["TARGET"] ==1])
print(i," ",(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 assessed 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)]
defaulters2 = Fam_Details[(Fam_Details["NAME_HOUSING_TYPE"] == i) & (Client_Data["TARGET"] ==0)]
Total = len(Client_Data[Client_Data["TARGET"] ==1])
print(i," ",(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)]
defaulters2 = Fam_Details[(Fam_Details["NAME_TYPE_SUITE"] == i) & (Client_Data["TARGET"] ==0)]
Total = len(Client_Data[Client_Data["TARGET"] ==1])
print(i," ",(len(defaulters1)/Total)*100,3))

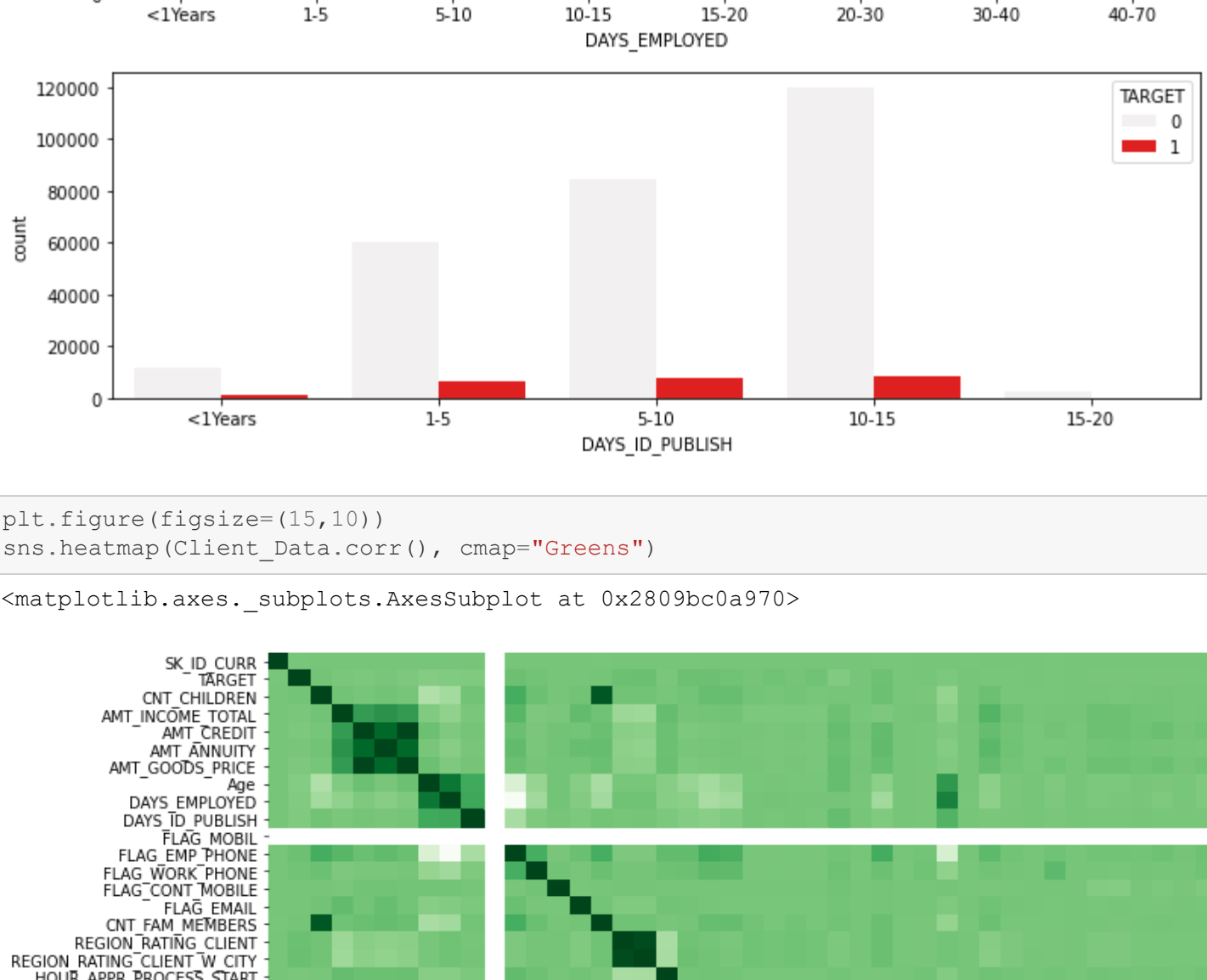
Unaccompanied : 82.13
Family : 12.169
Spouse, partner : 3.621
Children : 0.975
Other_A : 0.307
Other_B : 0.704
Group of people : 0.093
```

- 82% Defaulters were Unaccompanied while submitting 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()
```

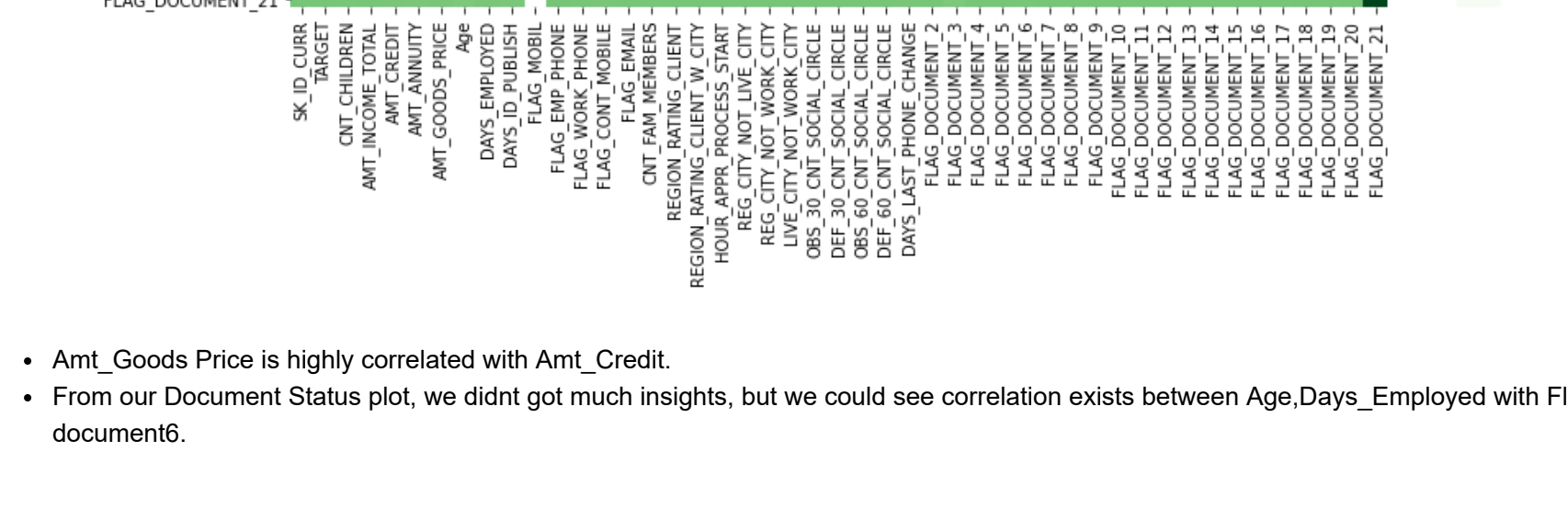
```
In [47]: Client_Data1["Age"] = pd.cut(x= Client_Data1["Age"],bins = [20,25,30,35,40,45,50,55,60,65,70],labels =
["20-25","25-30","30-35","35-40","40-45","45-50","50-55","55-60","60-65","65-70"])
Client_Data1["Days_Employed"] = pd.cut(x= Client_Data1["DAYS_EMPLOYED"],bins = [0,1,1,0,5,0,10,0,15,0,2
0,0,30,0,40,0,70,0],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 = [0,1,1,0,5,0,10,0,1
5,0,30,0],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_Data1.corr(), cmap="Greens")
```

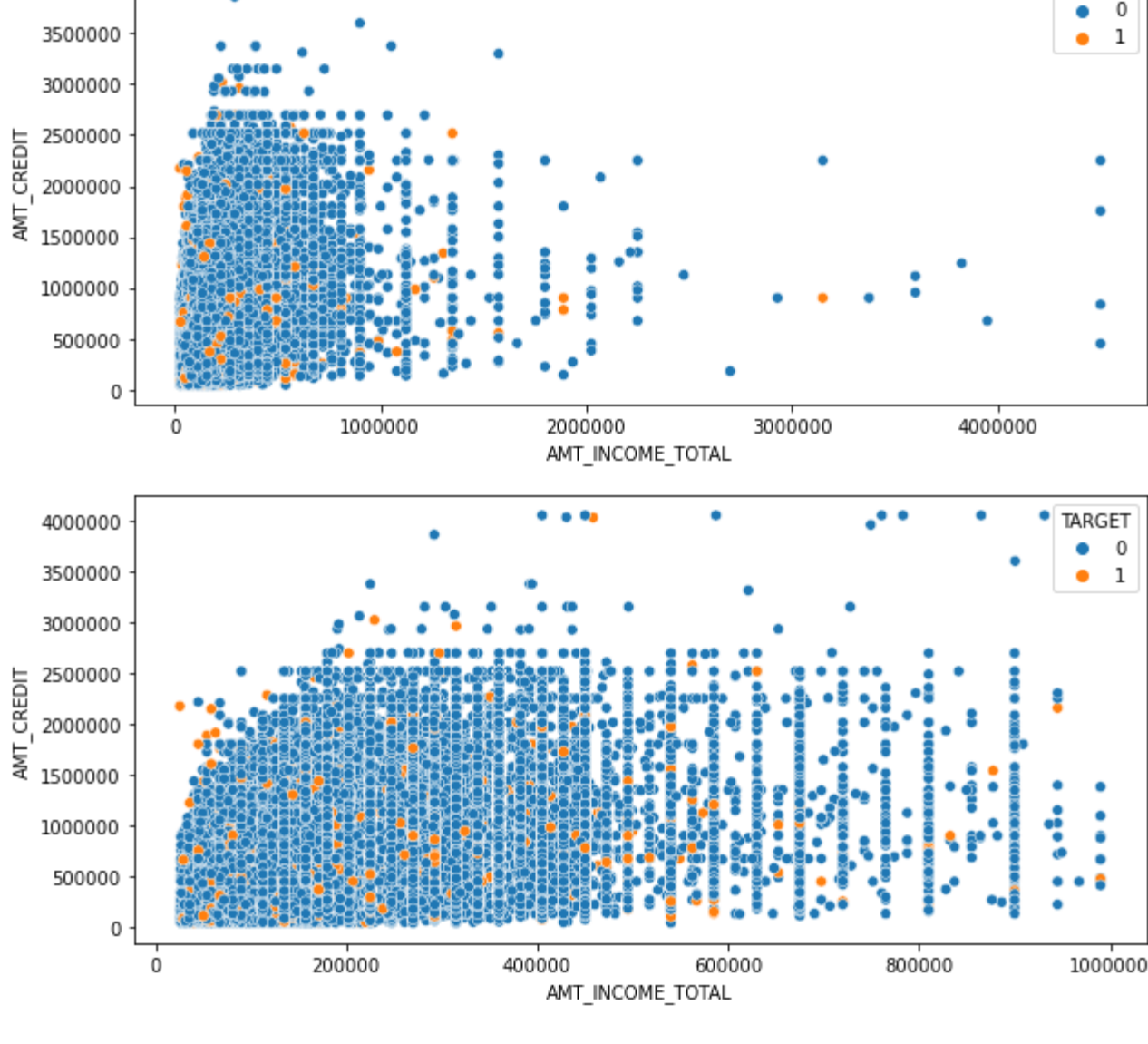
```
Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x2809bc0a970>
```



- 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 document.

Bi-Variate Analysis

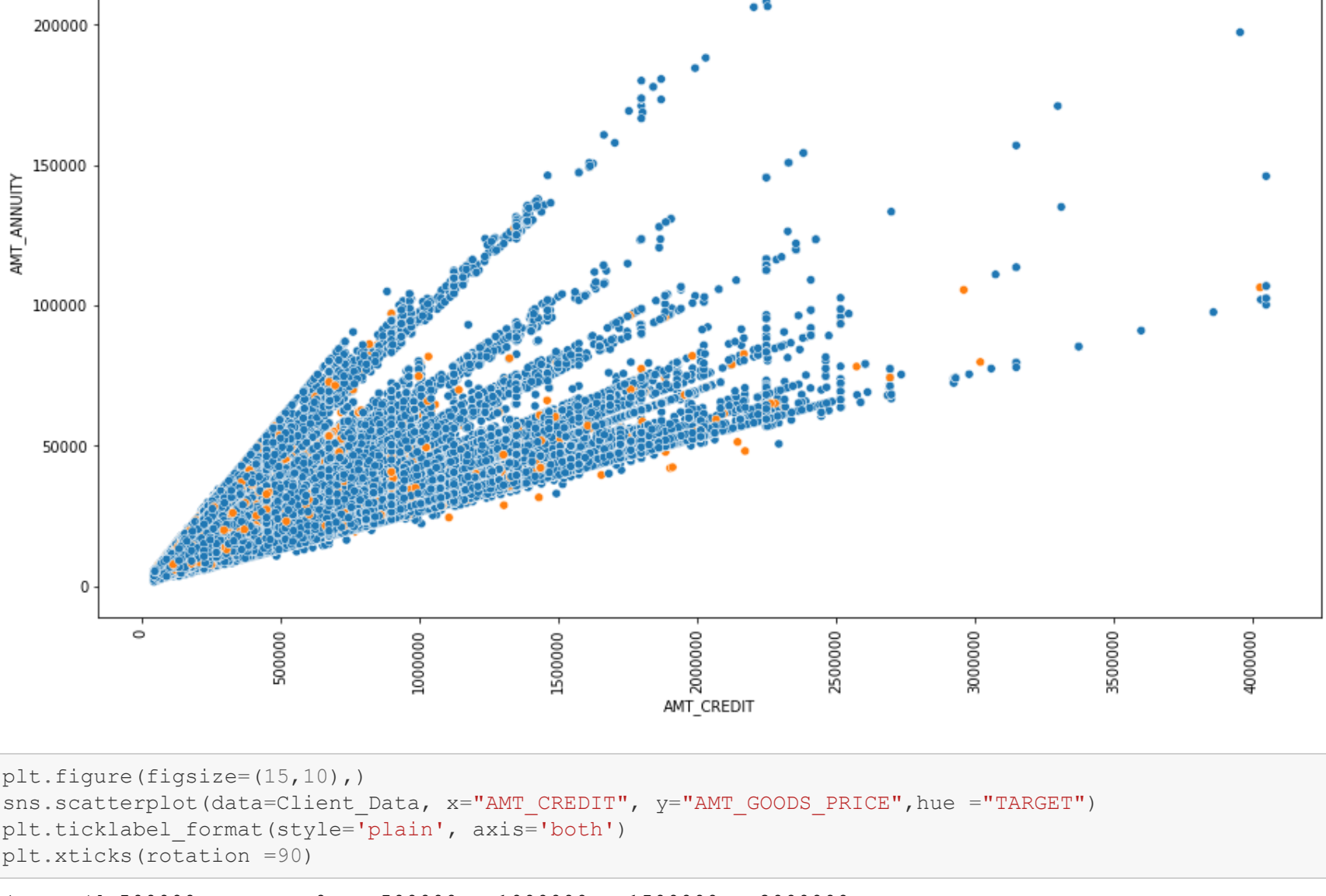
```
In [50]: fig,ax=plt.subplots(2,1,figsize=(10,10))
sns.scatterplot(data=Client_Data, x="AMT_INCOME_TOTAL", y="AMT_CREDIT",hue = "TARGET",ax=ax[0])
sns.scatterplot(data =Client_Data1,x="AMT_INCOME_TOTAL",y="AMT_INCOME_TOTAL",hue="TARGET",ax=ax[1])
sns.scatterplot(data=Inc, x="AMT_INCOME_TOTAL", y="AMT_CREDIT",hue = "TARGET",ax=ax[1])
fig.tight_layout()
```



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

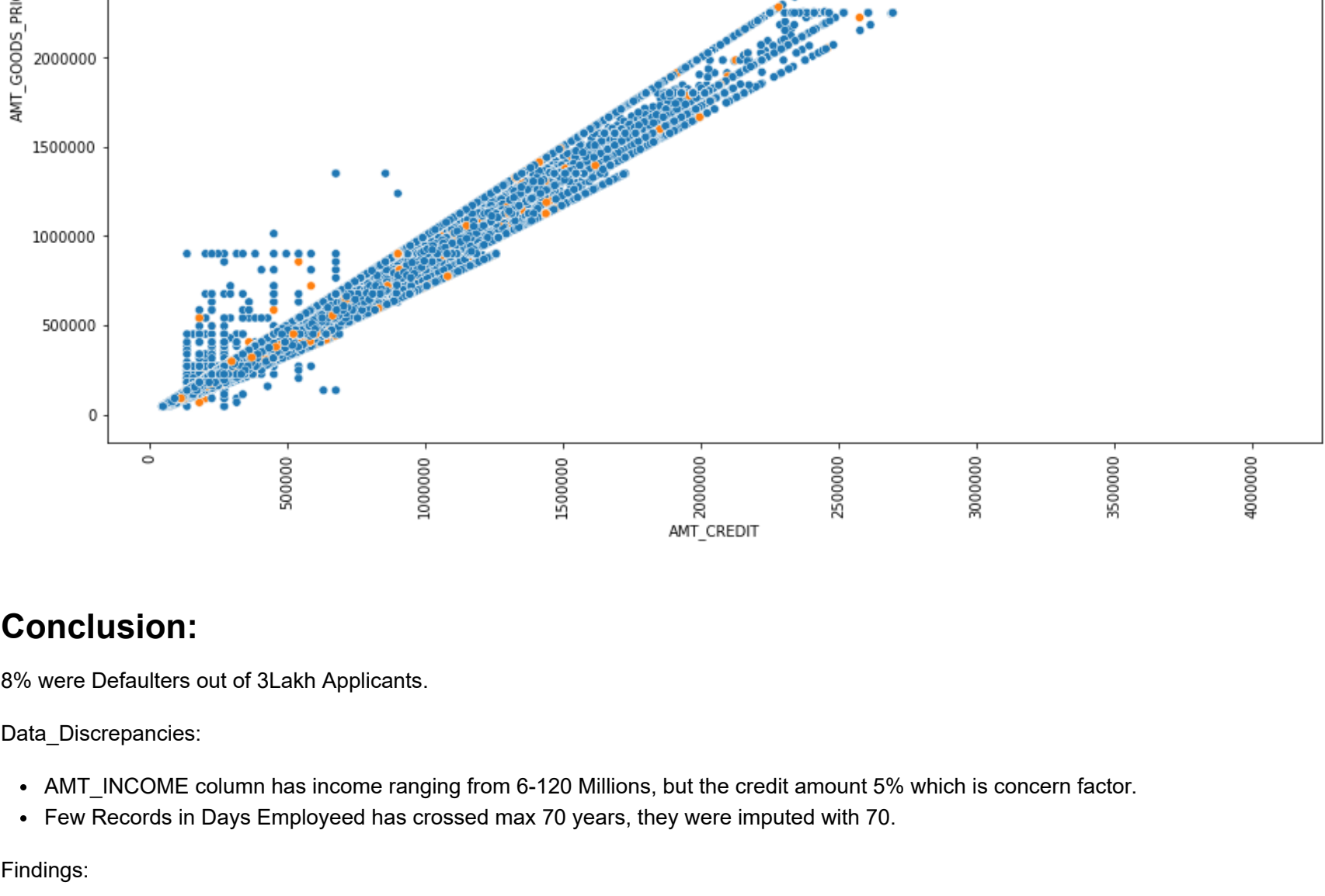
```
In [51]: plt.figure(figsize=(15,10))
sns.scatterplot(data=Client_Data, x="AMT_CREDIT", y="AMT_ANNUITY",hue = "TARGET")
plt.ticklabel_format(style='plain', axis='both')
plt.xticks(rotation =90)
```

```
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_Employed has crossed max 70 years, they were imputed with 70.

Findings:

- Maximum loans sanctioned were Cash loans.
- Highest Applicants are female, but defaulters 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 experience < 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 [ ] :
```