# CREDIT EDA CASE STUDY


## Abstract

In this case study, we are applying EDA in a real business scenario. Apart from applying the techniques, we are also developing a basic understanding of risk analytics in banking and financial services and understanding how data is used to minimize the risk of losing money while lending to customers.



Prepared by:

Anindita Kundu

(anindita.upgrad@gmail.com)

Daivanshu Gandhi

(daivanshu.gandhi@gmail.com)

## Introduction:

For the purpose of this case study, three data sets were provided, namely:

- Application Data
- Previous Application Data
- Columns description Data

Firstly, we have taken the 'Application Data Set' for our analysis. This contains all the information of the client at the time of application. The data is about whether **a client has payment difficulties**.

```
In [9]: Application_Data.shape
Out[9]: (307511, 41)
```

Here, we can see that the 'Application Data set' is having 307511 Rows and 41 Columns in Total.

## Data Cleaning:

Data Cleaning is done for preparing data for analysis. The following steps are taken for Data cleaning:

1. Firstly, we have found out which all columns are necessary for our analysis.

2. Then, we have tried to find out the NULL values in each column.

```
In [10]: ## FInding the null values in each column
         Application_Data.isnull().sum()

In [11]: ((Application_Data.isnull().sum()*100) / len(Application_Data)).round(2)
```

3. Dropping the columns where null values are more than 20%

```
In [13]: ## droppping columns where null values are more than 20%
         Application_Data.drop(['OCCUPATION_TYPE','EXT_SOURCE_1','EXT_SOURCE_3'],axis=1,inplace=True)

In [14]: Application_Data.shape
Out[14]: (307511, 38)
```

   Now, the number of columns has reduced to 38.

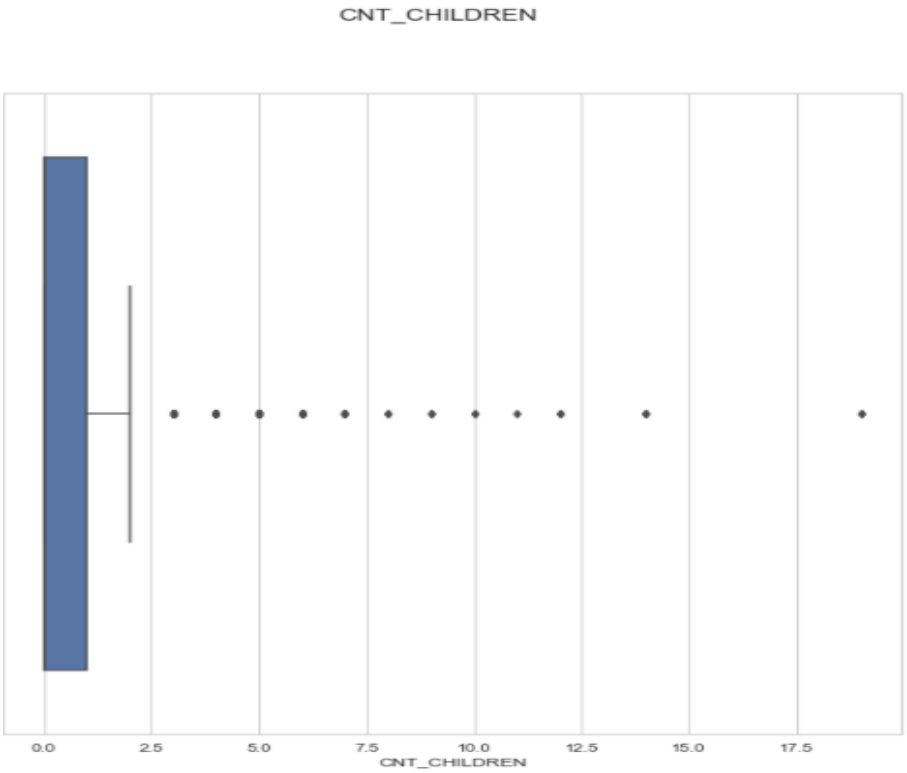
4. Post these actions, for the columns with minimal NULL values, we decided on imputing values on few columns to further make the data set usable.
   We have decided to impute MEAN values to the 'AMT_GOODS_PRICE', 'EXT_SOURCE_2' and 'CNT_FAM_MEMBERS' columns. And MODE for 'NAME_TYPE_SUITE' column.
   Below is the screenshot for reference.

```
In [16]: Application_Data['AMT_GOODS_PRICE'].fillna(Application_Data['AMT_GOODS_PRICE'].mean(),inplace=True)

In [17]: Application_Data['EXT_SOURCE_2'].fillna(Application_Data['EXT_SOURCE_2'].mean(),inplace=True)

In [18]: Application_Data['CNT_FAM_MEMBERS'].fillna(Application_Data['CNT_FAM_MEMBERS'].mean(),inplace=True)

In [19]: Application_Data['NAME_TYPE_SUITE'].value_counts()

Out[19]: Unaccompanied      248526
         Family              40149
         Spouse, partner     11370
         Children             3267
         Other_B              1770
         Other_A               866
         Group of people       271
         Name: NAME_TYPE_SUITE, dtype: int64

In [20]: ## Need to check this column values
         Application_Data['NAME_TYPE_SUITE'].fillna(Application_Data['NAME_TYPE_SUITE'].mode()[0],inplace=True)
```

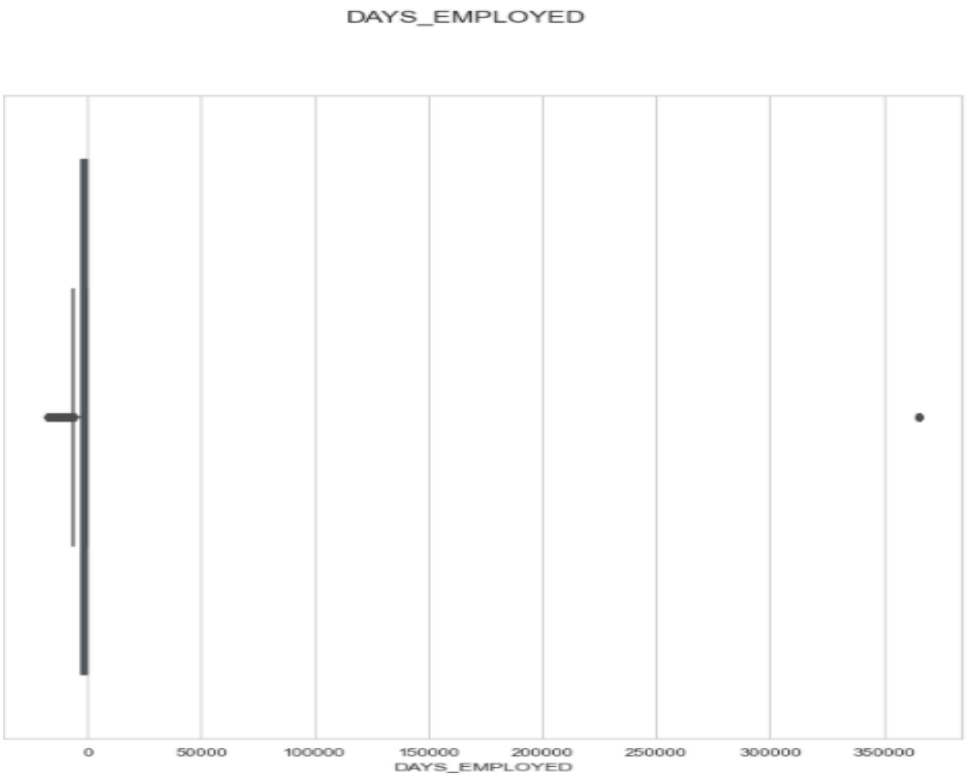
## Outliers:

An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. Here are some of the values we could spot as outliers with the help of plots:

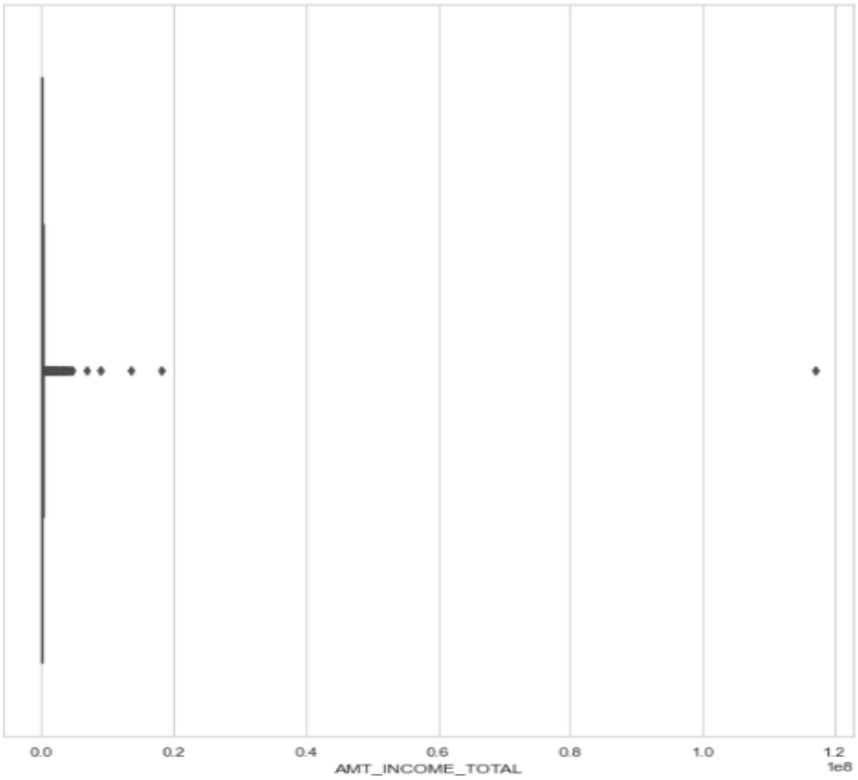
For the below plot of '**CNT_CHILDREN'**, we see that there is an Outlier. Since a family cannot or very rarely have 19 children.



In the plot for '**DAYS_EMPLOYED'**, we also find Outlier at a range of 36,000. This is not possible. We can assume that this error has occurred during data entry.




In the plot **'AMT_INCOME_TOTAL'**, we can visually see that the MAX amount is way larger than the other statistical data [Mean, (25,50,75) percentiles]



In [27]:
```
Application_Data[['CNT_CHILDREN','DAYS_EMPLOYED','AMT_INCOME_TOTAL']].describe().T
```

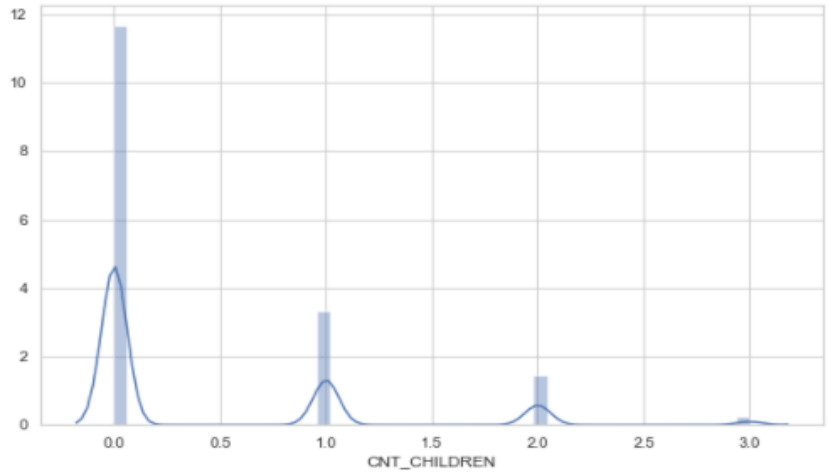
Out[27]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| CNT_CHILDREN | 307511.0 | 0.417052 | 0.722121 | 0.0 | 0.0 | 0.0 | 1.0 | 19.0 |
| DAYS_EMPLOYED | 307511.0 | 63815.045904 | 141275.766519 | -17912.0 | -2760.0 | -1213.0 | -289.0 | 365243.0 |
| AMT_INCOME_TOTAL | 307511.0 | 168797.919297 | 237123.146279 | 25650.0 | 112500.0 | 147150.0 | 202500.0 | 117000000.0 |

Finally, we are done with the identification of the outliers. We have removed them and plotted them again to observer the difference.

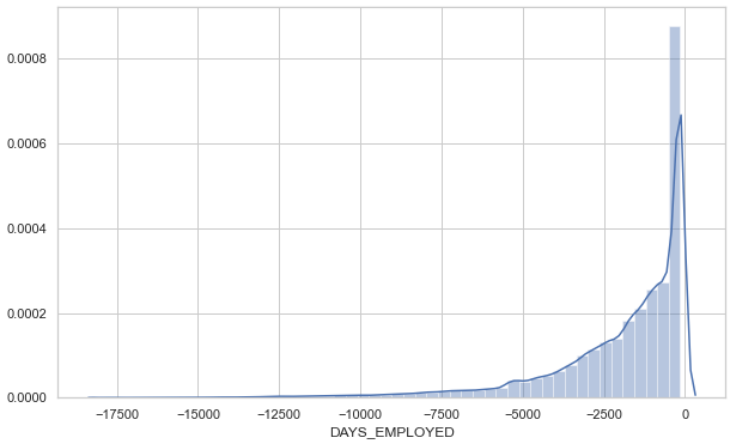
- *Restricting the Count of Children to maximum value of 3*

In [28]:
```
q1 = Application_Data['CNT_CHILDREN'].quantile(0.99)
Application_Data['CNT_CHILDREN'] = Application_Data['CNT_CHILDREN'].apply(lambda x: q1 if x>q1 else x)

# displaying a plot to show that the values have been restricted to a max value as 3
f, ax = plt.subplots(figsize=(10,6))
outlier_plot_1 = sns.distplot(Application_Data["CNT_CHILDREN"])
```



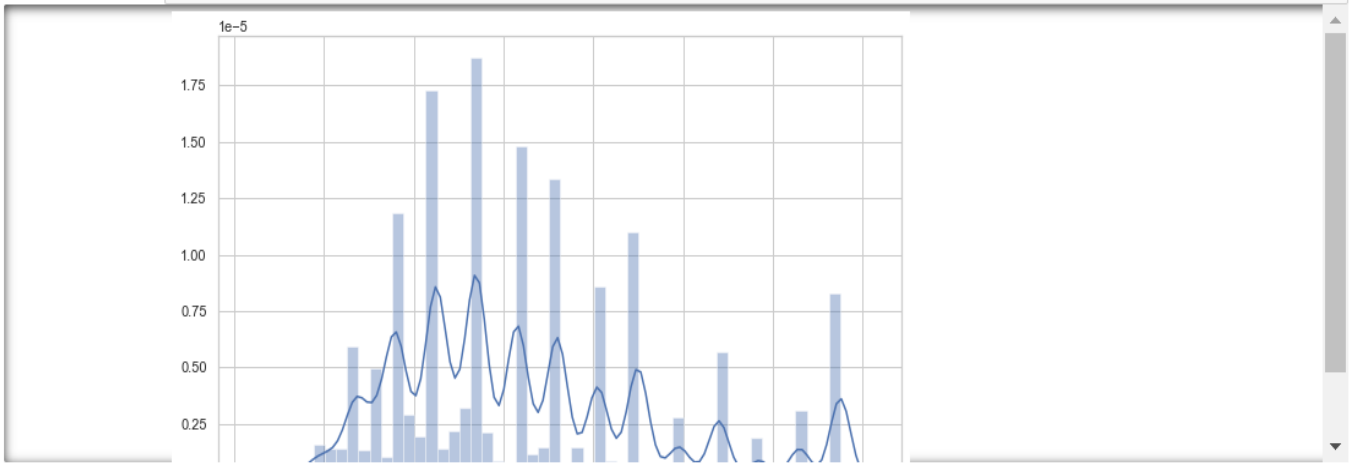

- *Restricting the Number of Days employed so that we do not get highly distributed values*

In [29]:
```
# displaying a plot to show that the values have been restricted so h=that we dont get highly distributed values


q2=Application_Data["DAYS_EMPLOYED"].quantile(0.80)
Application_Data["DAYS_EMPLOYED"] =  Application_Data.DAYS_EMPLOYED.apply(lambda x: q2 if x>q2 else x)

f, ax = plt.subplots(figsize=(10,6))
outlier_plot_2 = sns.distplot(Application_Data["DAYS_EMPLOYED"])
```



- *Restricting the total amount of income, so that we do not get a highly distributed value*

```
In [30]:  #  displaying a plot to show that the values have been restricted so that we dont get highly distributed values

          q3=Application_Data["AMT_INCOME_TOTAL"].quantile(0.95)
          Application_Data["AMT_INCOME_TOTAL"] =  Application_Data.AMT_INCOME_TOTAL.apply(lambda x: q3 if x>q3 else x)

          f, ax = plt.subplots(figsize=(10,6))
          outlier_plot_3 = sns.distplot(Application_Data["AMT_INCOME_TOTAL"])
```




Furthermore, we have done some modification of values, in order to make our analysis of data easier. We have taken the below two steps.

- Converting the Date of Birth from column '**DAYS_BIRTH'** to **AGE**

```
In [33]:  # Converting DAYS_BIRTH to AGE
          Application_Data["AGE"] = Application_Data.DAYS_BIRTH.apply(lambda x :round(abs(x)/365),0)
          Application_Data["AGE"]
          Application_Data["AGE"] = pd.to_numeric(Application_Data["AGE"])
```

```
In [34]:  #Dropping the DAYS_Birth column since we have created an AGE column for the same
          Application_Data.drop('DAYS_BIRTH',axis=1,inplace=True)
```

- And secondly, **binning of salaries** into High, Medium and Moderate for more clarity.

In [38]:
```
Application_Data.AMT_INCOME_TOTAL.quantile([0.25,0.5,0.85,1])
```

Out[38]:
```
0.25    112500.0
0.50    147150.0
0.85    234000.0
1.00    337500.0
Name: AMT_INCOME_TOTAL, dtype: float64
```

In [39]:
```
# Based on the Quantile Values , segregating the values to its respective categories
def salary_category_func(x):
    if x>=234000:
        return('HIGH')
    elif x<234000 and x>=147150:
        return('MODERATE')
    elif x<147150 and x>=112500:
        return('LOW')
    else:
        return("EXTREMLY LOW")
Application_Data["SALARY_CATEGORY"] = Application_Data.AMT_INCOME_TOTAL.apply(salary_category_func)
Application_Data["SALARY_CATEGORY"]
```

Out[39]:
```
0             MODERATE
1                 HIGH
2          EXTREMLY LOW
3                  LOW
4                  LOW
                 ...
307506        MODERATE
307507     EXTREMLY LOW
307508        MODERATE
307509        MODERATE
307510        MODERATE
Name: SALARY_CATEGORY, Length: 307511, dtype: object
```

In [40]:
```
Application_Data["SALARY_CATEGORY"].value_counts()
```

Out[40]:
```
MODERATE        106966
LOW              84194
EXTREMLY LOW     69559
HIGH             46792
Name: SALARY_CATEGORY, dtype: int64
```

In [41]:
```
#Dropping "AMT_INCOME_TOTAL" Column,because have Binned those Salary Values
Application_Data.drop("AMT_INCOME_TOTAL",axis=1,inplace=True)
```

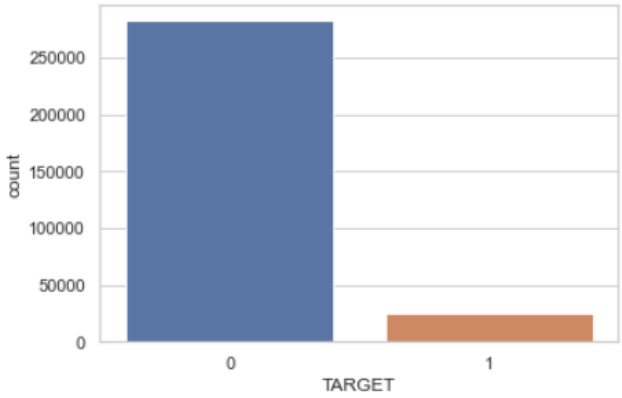
## ANALYSIS OF APPLICATION DATA

Then we proceeded with the analysis of data.

### Checking the **TARGET** Variable

If we plot the count of the TARGET Variable, we observer that there is a huge imbalance in our Target variable.

```
In [42]: Target_count= sns.countplot("TARGET",data =Application_Data)
```




So, we can segregate the TARGET variable into two different data frames – Defaulters and Good Clients.

```
In [43]: good_client = Application_Data[Application_Data.TARGET == 0]
         defaulter_client = Application_Data[Application_Data.TARGET == 1]
```

From the data,

**TARGET value 0** indicates that the client is not a defaulter thus a good client. And **TARGET value 1** indicates, client with payment difficulties, i.e., the client has late payment with more than X days on at least one of the first Y instalments of the loan in our sample.

### Checking for Client who are unlikely to pay their loans

```
In [46]: numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
         Univariate_defaulter_Num_1_df = defaulter_client.select_dtypes(include=numerics)
         Univariate_defaulter_Num_1_df
```

Out[46]:

| | SK_ID_CURR | TARGET | CNT_CHILDREN | AMT_CREDIT | AMT_ANNUITY | AMT_GOODS_PRICE | REGION_POPULATION_RELATIVE | DAYS_EMPLOYED |
|---|---|---|---|---|---|---|---|---|
| 0 | 100002 | 1 | 0.0 | 406597.5 | 24700.5 | 351000.0 | 0.018801 | -637.0 |
| 26 | 100031 | 1 | 0.0 | 979992.0 | 27076.5 | 702000.0 | 0.018029 | -2628.0 |
| 40 | 100047 | 1 | 0.0 | 1193580.0 | 35028.0 | 855000.0 | 0.025164 | -1262.0 |
| 42 | 100049 | 1 | 0.0 | 288873.0 | 16258.5 | 238500.0 | 0.007305 | -3597.0 |
| 81 | 100096 | 1 | 0.0 | 252000.0 | 14593.5 | 252000.0 | 0.028663 | -144.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 307448 | 456186 | 1 | 1.0 | 450000.0 | 32746.5 | 450000.0 | 0.020246 | -3048.0 |
| 307475 | 456215 | 1 | 1.0 | 1303200.0 | 46809.0 | 1125000.0 | 0.007330 | -2405.0 |
| 307481 | 456225 | 1 | 0.0 | 297000.0 | 19975.5 | 297000.0 | 0.008575 | -3147.0 |
| 307489 | 456233 | 1 | 0.0 | 521280.0 | 23089.5 | 450000.0 | 0.014464 | -286.0 |
| 307509 | 456254 | 1 | 0.0 | 370107.0 | 20205.0 | 319500.0 | 0.005313 | -4786.0 |

24825 rows × 26 columns

In [47]:
```
categorical = ["object"]
Univariate_defaulter_Cat_1_df =defaulter_client.select_dtypes(include=categorical)
Univariate_defaulter_Cat_1_df
```

Out[47]:

| | NAME_CONTRACT_TYPE | CODE_GENDER | FLAG_OWN_CAR | FLAG_OWN_REALTY | NAME_TYPE_SUITE | NAME_INCOME_TYPE | NAME_EDUCATION_ |
|---|---|---|---|---|---|---|---|
| 0 | Cash loans | M | N | Y | Unaccompanied | Working | Secondary / secc s |
| 26 | Cash loans | F | N | Y | Unaccompanied | Working | Secondary / secc s |
| 40 | Cash loans | M | N | Y | Unaccompanied | Commercial associate | Secondary / secc s |
| 42 | Cash loans | F | N | N | Unaccompanied | Working | Secondary / secc s |
| 81 | Cash loans | F | N | Y | Unaccompanied | Pensioner | Secondary / secc s |
| ... | ... | ... | ... | ... | ... | ... | |
| 307448 | Cash loans | M | N | N | Unaccompanied | Working | Secondary / secc s |
| 307475 | Cash loans | F | N | N | Unaccompanied | Working | Higher edu |
| 307481 | Cash loans | M | N | Y | Family | Working | Secondary / secc s |
| 307489 | Cash loans | F | N | Y | Unaccompanied | Commercial associate | Secondary / secc s |
| 307509 | Cash loans | F | N | Y | Unaccompanied | Commercial associate | Secondary / secc s |

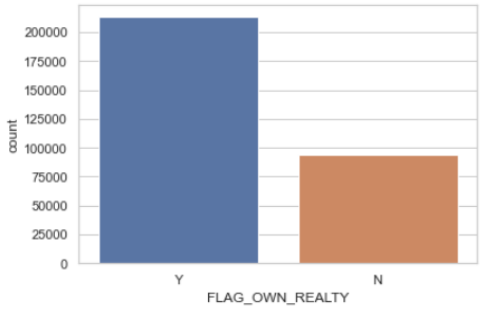
24825 rows × 12 columns

## Univariate Analysis of Categorical and Numerical Data

### Based on FLAG_OWN_REALTY:

- We will now compare and check how the possession of a property affects the repayment of loans. From the below plot, we can see that clients having some sort of real estate acquirements are more than the ones who don't have any property.

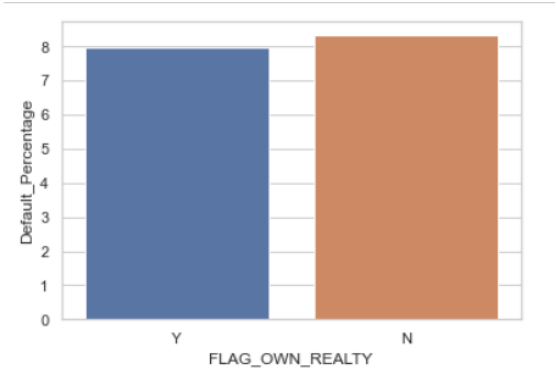
In [48]:
```
#graph to plot Number of property owners and non-property owners in the entire polulation
PropertyOwners_vs_Total = sns.countplot("FLAG_OWN_REALTY",data =Application_Data)
```




- Next, we found out the percentage of property and non-property owners in the defaulter list and plotting to show the number of property and non-property owners vs. Target variable. Here, Target =1.

Out[49]:

| | FLAG_OWN_REALTY | Default_Percentage |
|---|---|---|
| 0 | Y | 7.96 |
| 1 | N | 8.32 |




*Inference:*

We can see that the number of Non-Payers of Loan i.e., Defaulters are very close almost equal to 9%. It is difficult to decide a target based on this metric.

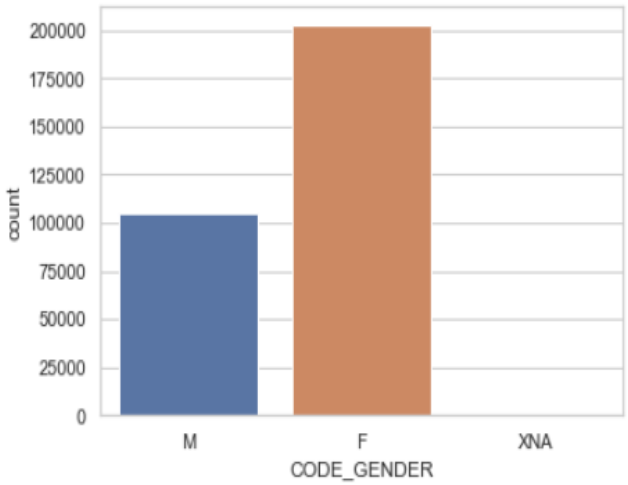
## Based on GENDER:

Now here we have compared and checked, how the GENDER of client affects the repayment of loans.

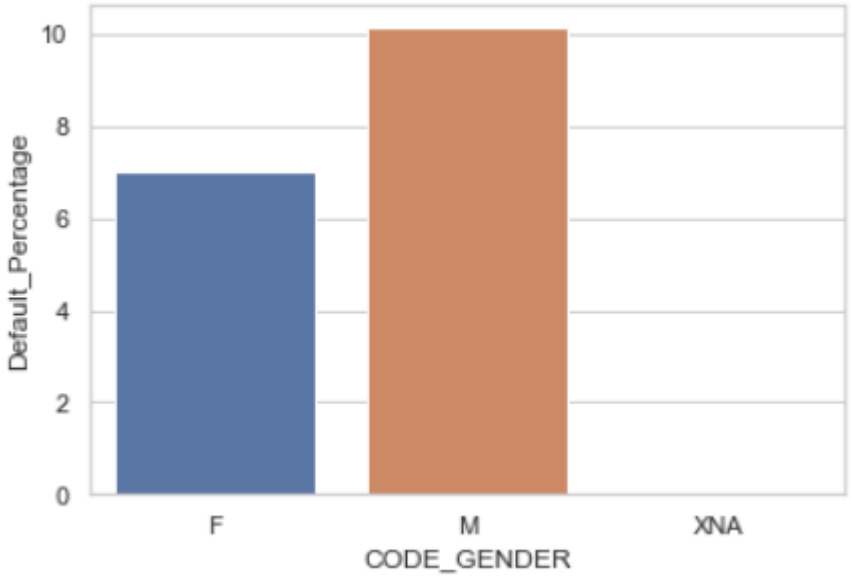
We have found out the percentage of males and females in the defaulter list.

Out[52]:

| | CODE_GENDER | Default_Percentage |
|---|---|---|
| 0 | F | 7.00 |
| 1 | M | 10.14 |
| 2 | XNA | NaN |



The below plot shows the number male and female clients vs. Target variable. Here, Target =1




*Inference:*

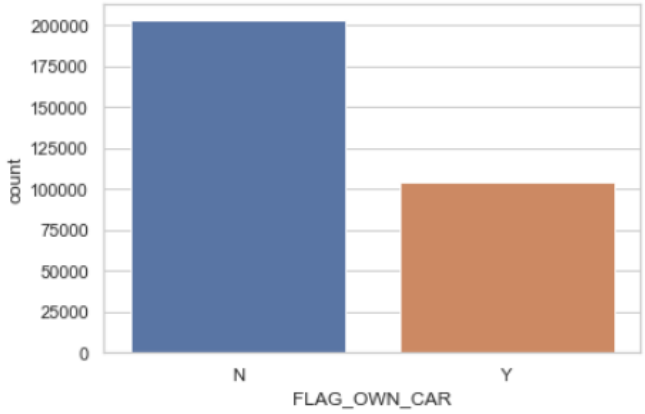
So, from the above Plots and Data it is visible that the Female clients are a better TARGET as compared to the Male clients.

Observing the percent of defaulted credits, male client has a higher chance of not returning their loans [10.14%], compared to the female clients [7%].

## Based on **_FLAG_OWN_CAR _ :**

Here we are comparing and check how the car owners and non-car owners differ in their repayment of loans.

```
In [54]: #graph to plot car owners and non-carOwners in the entire polulation
         CarOwner_vs_Total = sns.countplot("FLAG_OWN_CAR",data =Application_Data)
```

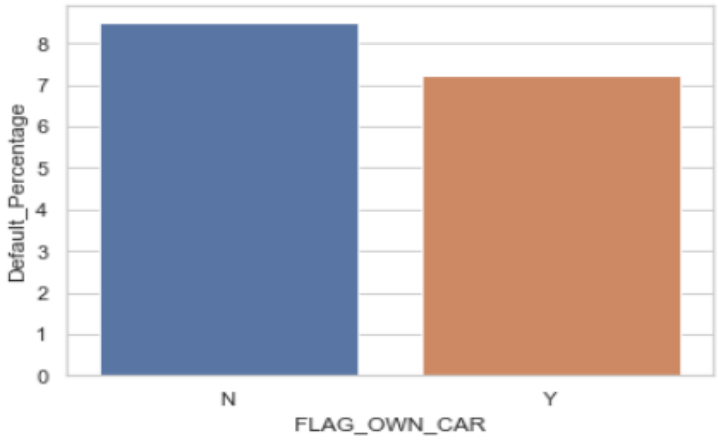



From the above plot, we can see that Clients who don't own cars/vehicles are more in the given population.

We found out the percentage of car owners and non-car Owners in the defaulter list and plotting to show the car owners and non-car Owners vs. Target variable. Here, Target =1

Out[55]:

| | FLAG_OWN_CAR | Default_Percentage |
|---|---|---|
| 0 | N | 8.50 |
| 1 | Y | 7.24 |




*Inference:*

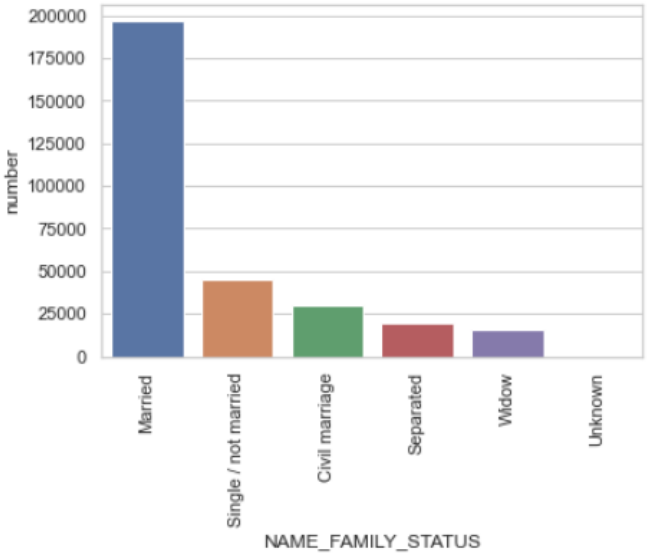
As we can see from above graph, the clients that own a car are less likely to not repay the loan when compared to the ones that do not own a car. The loan non-repayment rates of both the Car Owners and Non-Car Owners are very close. This is interesting to see and indicates that probably this metric will **not be a suitable** one when targeting a client.

## Based on **NAME_FAMILY_STATUS:**

Here we are comparing and checking how the family status of clients affect their repayment of loans.

Out[57]:

| | NAME_FAMILY_STATUS | number |
|---|---|---|
| 0 | Married | 196432 |
| 1 | Single / not married | 45444 |
| 2 | Civil marriage | 29775 |
| 3 | Separated | 19770 |
| 4 | Widow | 16088 |
| 5 | Unknown | 2 |

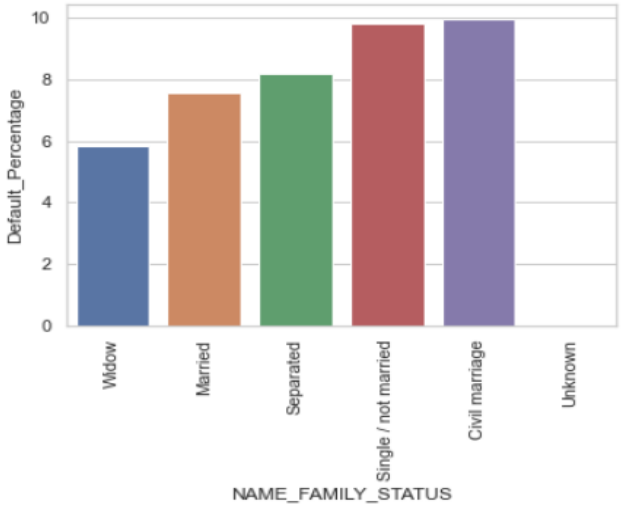



Here from the bar chart, it is evident that most of the clients are Married, followed by Single and Civil marriage.

Next, we are the percentage of clients according to family status in the defaulter list and then plotting to show the family status of client vs. Target variable. Here, Target =1

Out[59]:

| | NAME_FAMILY_STATUS | Default_Percentage |
|---|---|---|
| 5 | Widow | 5.82 |
| 1 | Married | 7.56 |
| 2 | Separated | 8.19 |
| 3 | Single / not married | 9.81 |
| 0 | Civil marriage | 9.94 |
| 4 | Unknown | NaN |



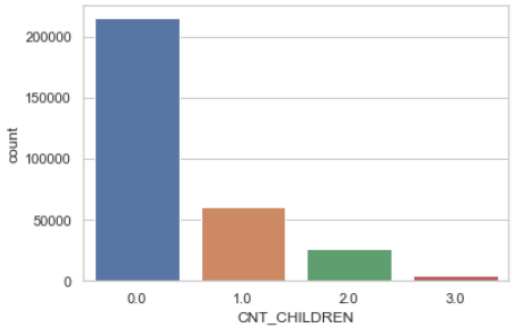

*Inference:*

From above graph, we can say that the percentage of non-repayment of loan is at highest for civil marriage and lowest for widows. This is interesting to see because you expect widows to not payback their loans but it is the opposite here.

## Based on **CNT_CHILDREN :**

Now, comparing how the number of children in a family affects the non-repayment of loans.

In [61]:
```
#graph to plot Number of children per client in the entire polulation
NoOfChildren_vs_Total = sns.countplot("CNT_CHILDREN",data =Application_Data)
```

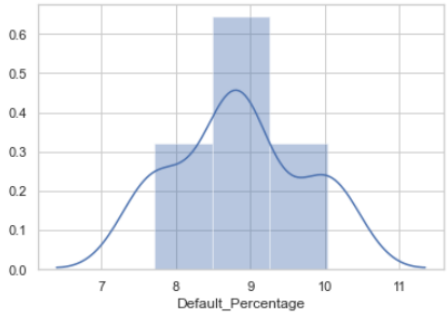


Out[62]:

| | CNT_CHILDREN | Default_Percentage |
|---|---|---|
| **3** | 3.0 | 10.04 |
| **1** | 1.0 | 8.92 |
| **2** | 2.0 | 8.72 |
| **0** | 0.0 | 7.71 |

In [63]:

```
#plot to show the number of children per client vs. Target variable. Here, Target =1

#NoOfChildren_vs_Target= sns.barplot(x="CNT_CHILDREN",y="Default_Percentage",data=test_df5,order=test_df5['CNT_CHILDREN'])

NoOfChildren_vs_Target = sns.distplot(test_df5["Default_Percentage"])
plt.show()
```

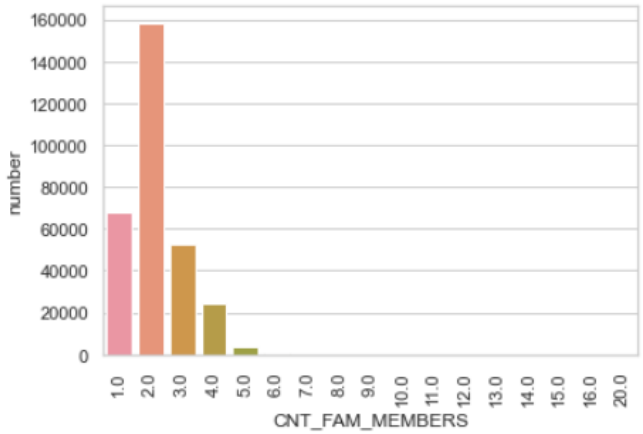



*Inference:*

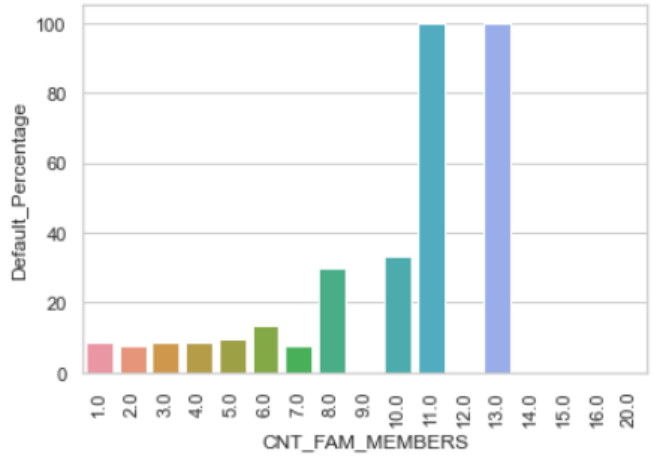
There is more chance for a client with more children to not repay the loan back. This can be because of the more liability that is on the client. The more the number of children the more difficult it is for the client to repay the loan due to more personal expenditures.

## Based on **CNT_FAM_MEMBERS:**

The below plot is about the Number of family members per client in the entire population.




The percentage number of family members per client the defaulter list:

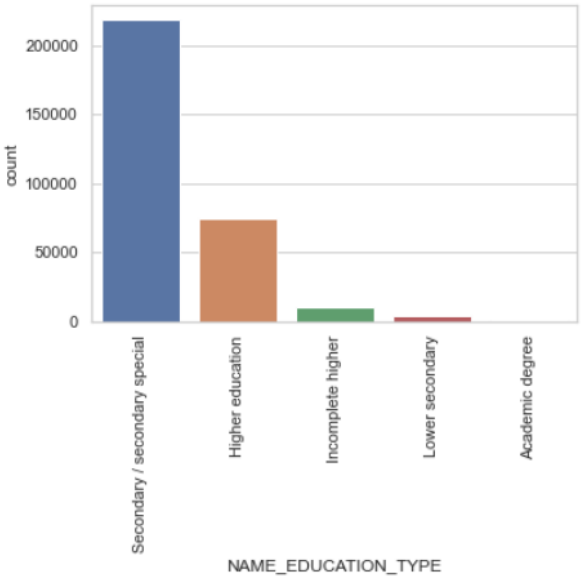



*Inference:*

By analysing the data with Count of Family Members Metrics, we can observe that though we can see that family with 11 and 13 members shows highest default rate, but their count is very less [2].

## Based on **NAME_EDUCATION_TYPE:**

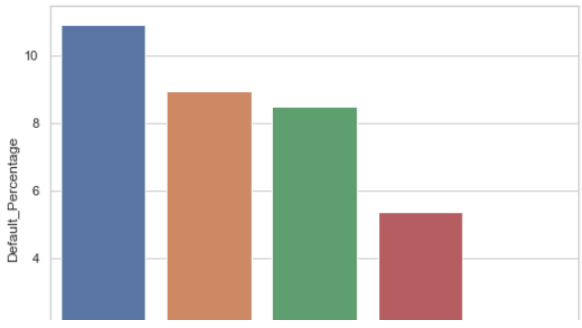
Plot of education type in the entire population.




Finding out the percentage education level of clients in the defaulter list and plotting the education type of each client vs. Target variable. Here, Target =1

)ut[69]:

| | NAME_EDUCATION_TYPE | Default_Percentage |
|---|---|---|
| 3 | Lower secondary | 10.93 |
| 0 | Secondary / secondary special | 8.94 |
| 2 | Incomplete higher | 8.48 |
| 1 | Higher education | 5.36 |
| 4 | Academic degree | 1.83 |



*Inference:*

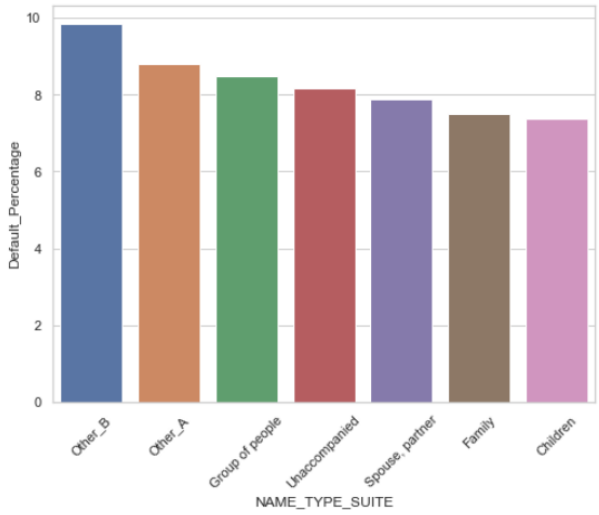
It can be seen from above graph that the more educated clients are likely to repay their loans because they will be having more stable jobs with monthly income.

## Based on **NAME_TYPE_SUITE**

Here we are analysing the data for **NAME_TYPE_SUITE**, i.e. Who was accompanying client when he was applying for the loan. And how this metrics affects the non-repayment of loans

Out[72]:

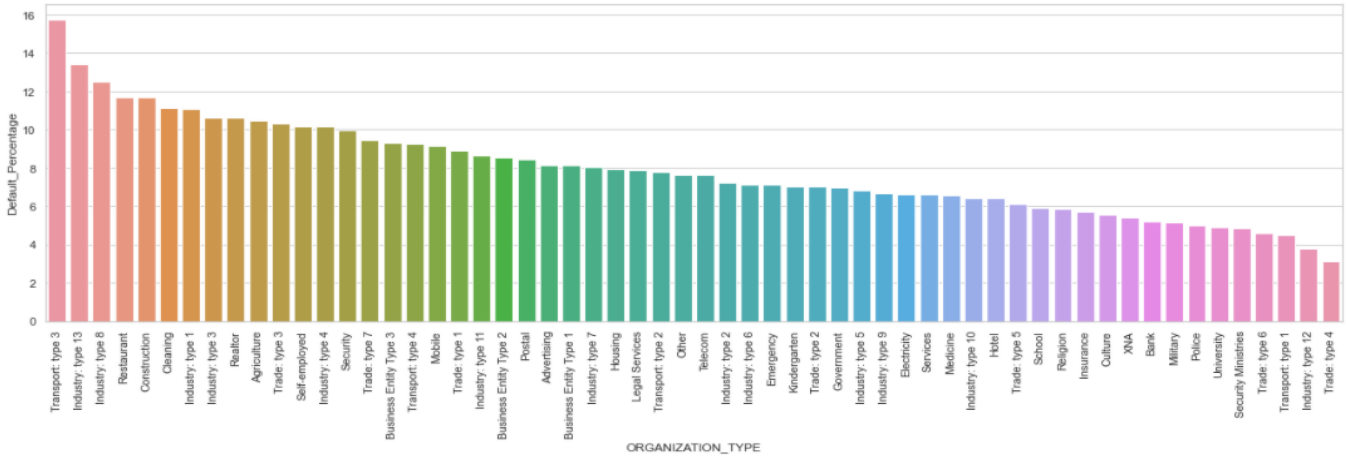
| | NAME_TYPE_SUITE | Default_Percentage |
|---|---|---|
| **4** | Other_B | 9.83 |
| **5** | Other_A | 8.78 |
| **6** | Group of people | 8.49 |
| **0** | Unaccompanied | 8.17 |
| **2** | Spouse, partner | 7.87 |
| **1** | Family | 7.49 |
| **3** | Children | 7.38 |




*Inference:*

Most clients who were occupied by **Other_B** and followed by **Other_A** are unlikely to pay back their loans.

## Based on **ORGANISATION_TYPE:**

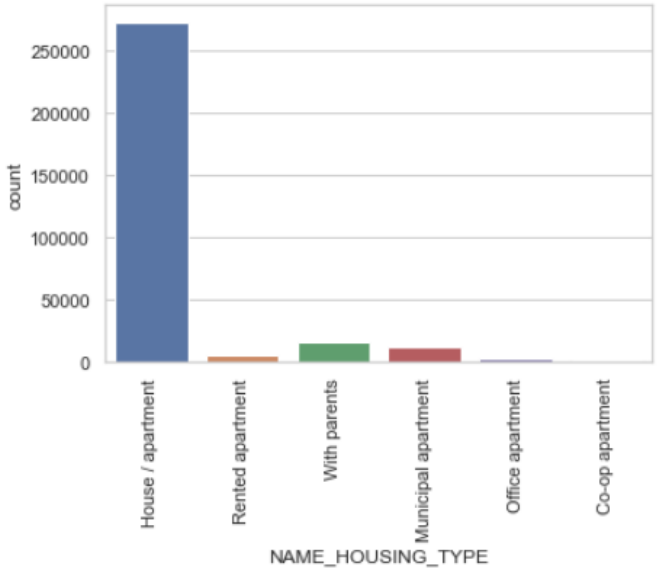


*Inference:*

From above graph, highest number of non-repayments can be seen in Applicants who work in **Transport Type3**.

## Based on **NAME_HOUSING_TYPE:**

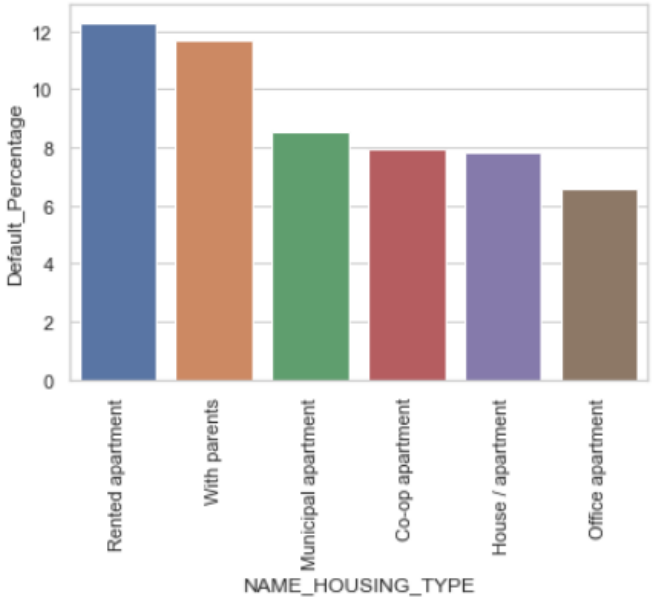
The below plot shows the Housing type in the entire population.




Calculating the percentage housing type of each client the defaulter list and plotting to show the housing type of each client vs. Target variable. Here, Target =1

Out[77]:

| | NAME_HOUSING_TYPE | Default_Percentage |
|---|---|---|
| 3 | Rented apartment | 12.31 |
| 1 | With parents | 11.70 |
| 2 | Municipal apartment | 8.54 |
| 5 | Co-op apartment | 7.93 |
| 0 | House / apartment | 7.80 |
| 4 | Office apartment | 6.57 |



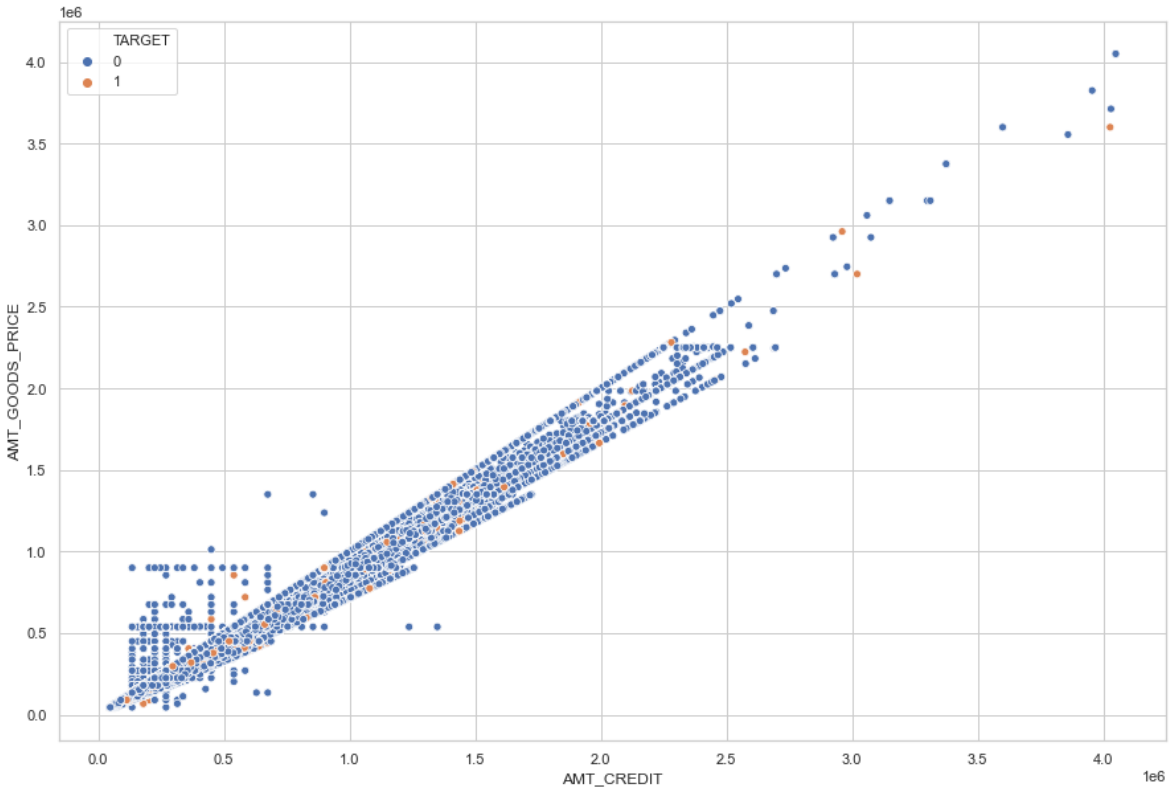

*Inference:*

From above graph it can be seen clearly that people with rented apartments are less likely to pay back their loans. This can be because they already have more liabilities compared to other type of people who do not have this liability.

# Bivariate Analysis

## AMT_CREDIT vs AMT_GOODS_PRICE

```
In [79]: f, ax = plt.subplots(figsize=(15,10))
         sns.scatterplot("AMT_CREDIT","AMT_GOODS_PRICE",data=Application_Data,hue="TARGET")
         plt.show()
```

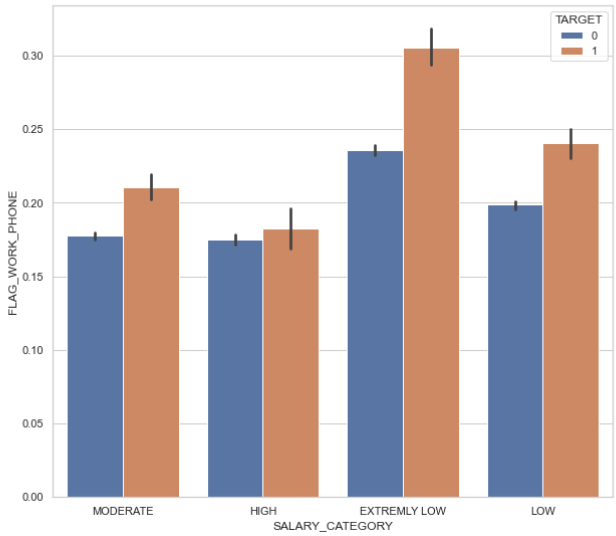



*Inference:*

We found that Credit amount and the Amount goods price are more correlated with the Defaulters. The Defaulters are linearly increasing as these both variable increases.

## SALARY CATEGORY vs CLIENT WHO PROVIDED HOME NUMBER

```
In [81]: f, ax = plt.subplots(figsize=(10,9))
         plot_1=sns.barplot("SALARY_CATEGORY","FLAG_WORK_PHONE",data=Application_Data,hue="TARGET")
         plt.show()
```

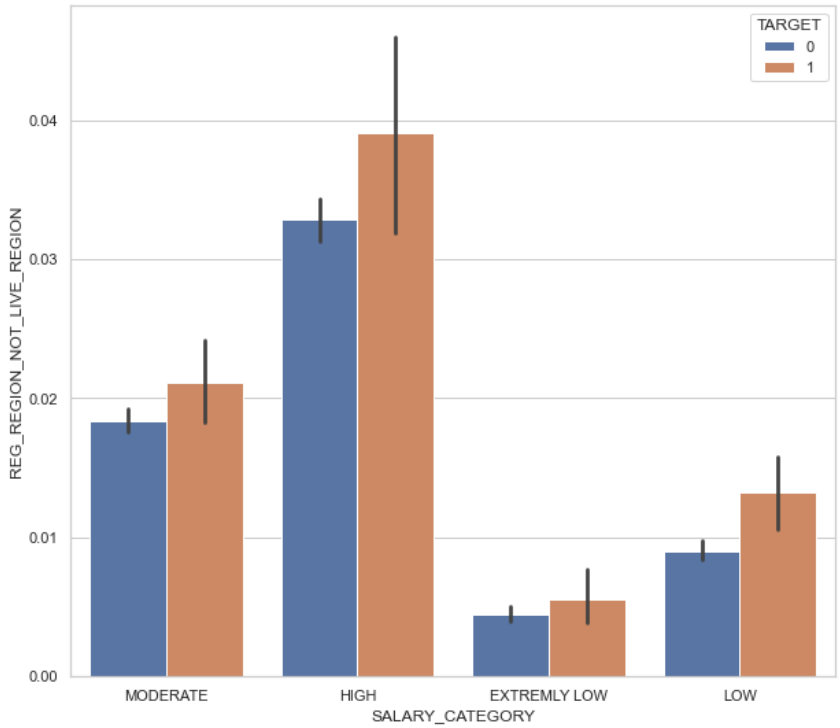


*Inference:*

Client with extremely low salary has more chance to be a Defaulter, when he did not provide the Home phone number. Here approximately 30% people only provided the phone number.

## SALARY VS CLIENT WHOSE PERMANENT ADDRESS NOT MATCH WITH CONTACT ADDRESS -REGION LEVEL

```
In [83]: f, ax = plt.subplots(figsize=(10,9))
         plot_1=sns.barplot("SALARY_CATEGORY","REG_REGION_NOT_LIVE_REGION",data=Application_Data,hue="TARGET")
         plt.show()
```

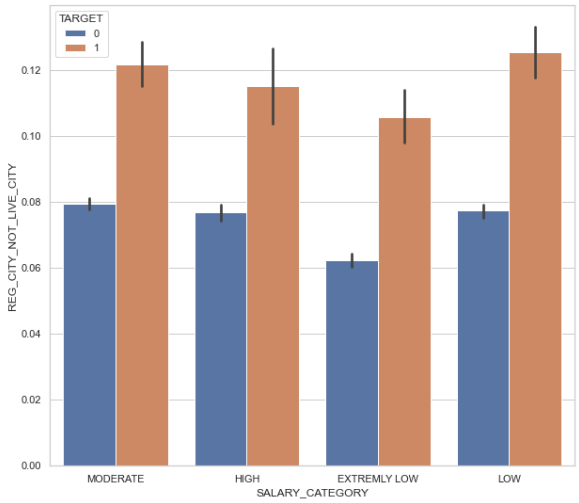



*Inference:*

When Client gets Extremely lower salary and if his/her address does not match, then there is a Higher chance for him/her to be defaulter.

## SALARY VS CLIENT WHOSE PERMANENT ADDRESS DOES NOT MATCH WITH WORK ADDRESS – REGION LEVEL

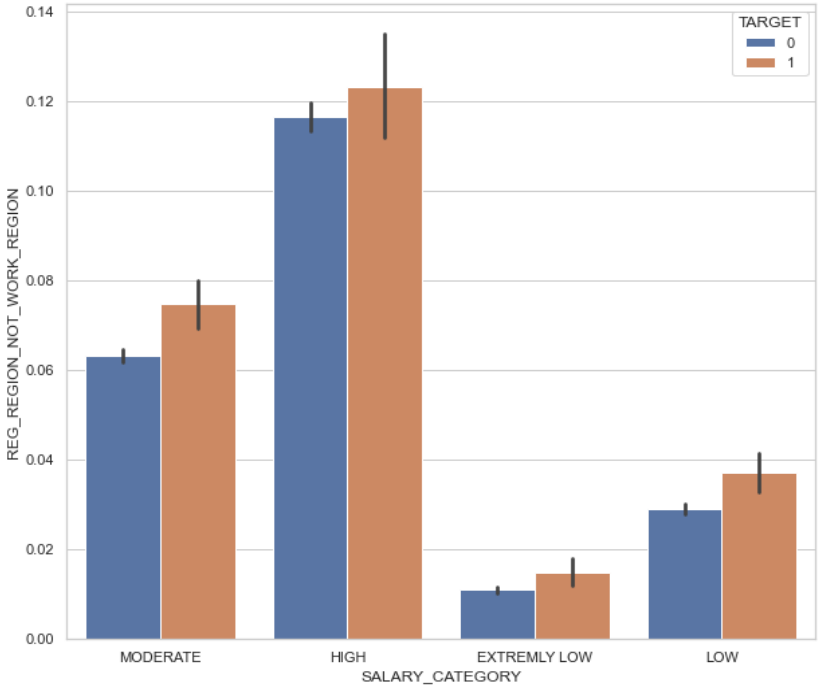
```
In [85]: f, ax = plt.subplots(figsize=(10,9))
         plot_1=sns.barplot("SALARY_CATEGORY","REG_CITY_NOT_LIVE_CITY",data=Application_Data,hue="TARGET")
         plt.show()
```



*Inference:*

When Client gets LOWER salary and if his/her CONTACT address (CITY-LEVEL) does not match, then there is a Higher chance for him/her to be defaulter.

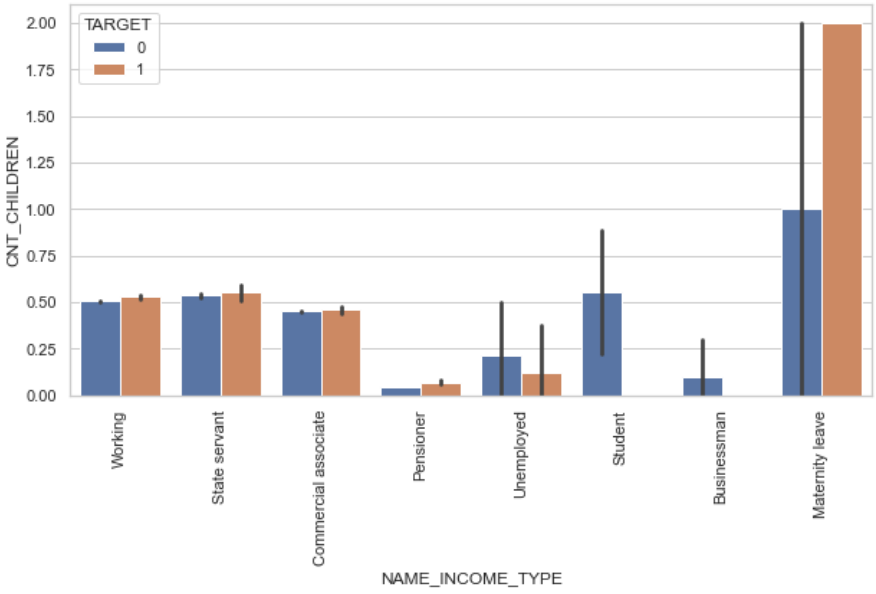
### Salary vs Client whose Permanent Address not match with Contact Address -City Level




*Inference:*

When Client gets HIGH salary and if his/her WORK address (CITY-LEVEL) doesn't match, then there is a Higher chance for him/her to be defaulter.

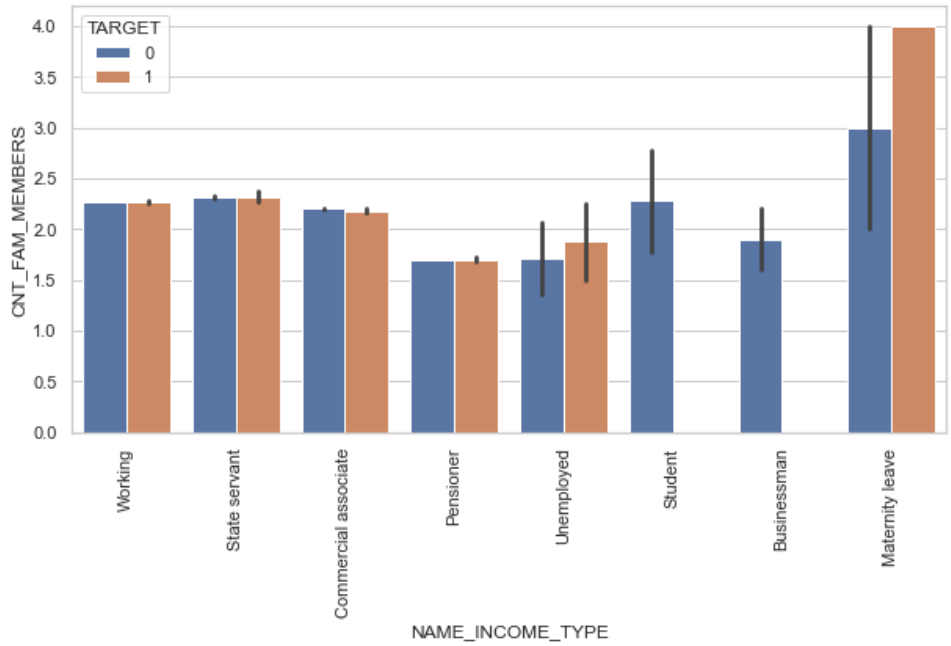
### INCOME vs CHILDREN Count



*Inference:*

People who getting income via Maternity Leave tends to be more Defaulter when they have more children.

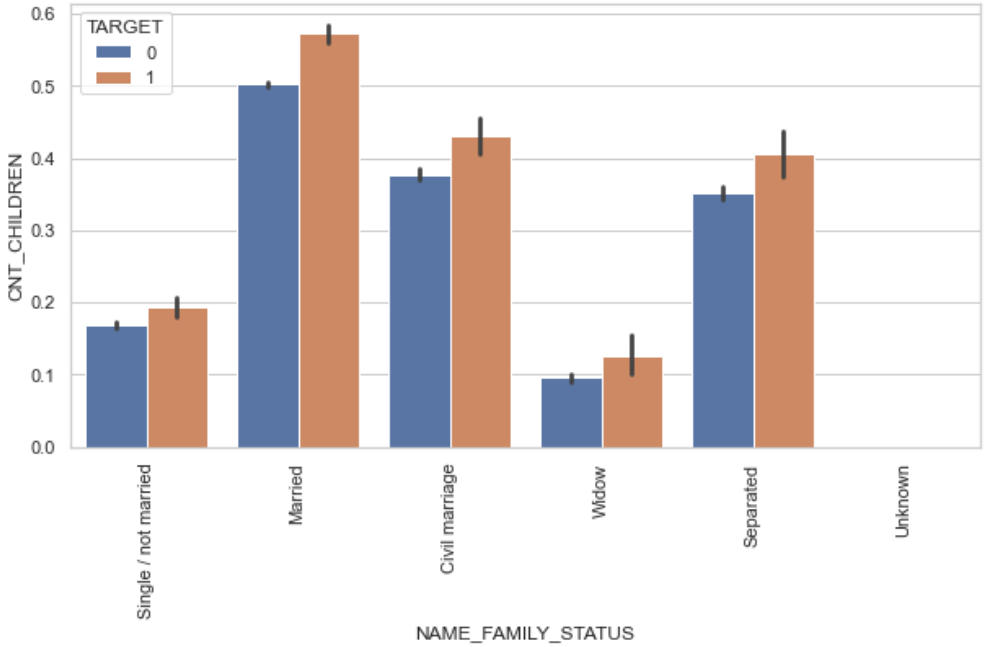
### Income vs Number of Family Members




*Inference:*

People who getting income via Maternity Leave tends to be more Defaulter when they have more Family Members

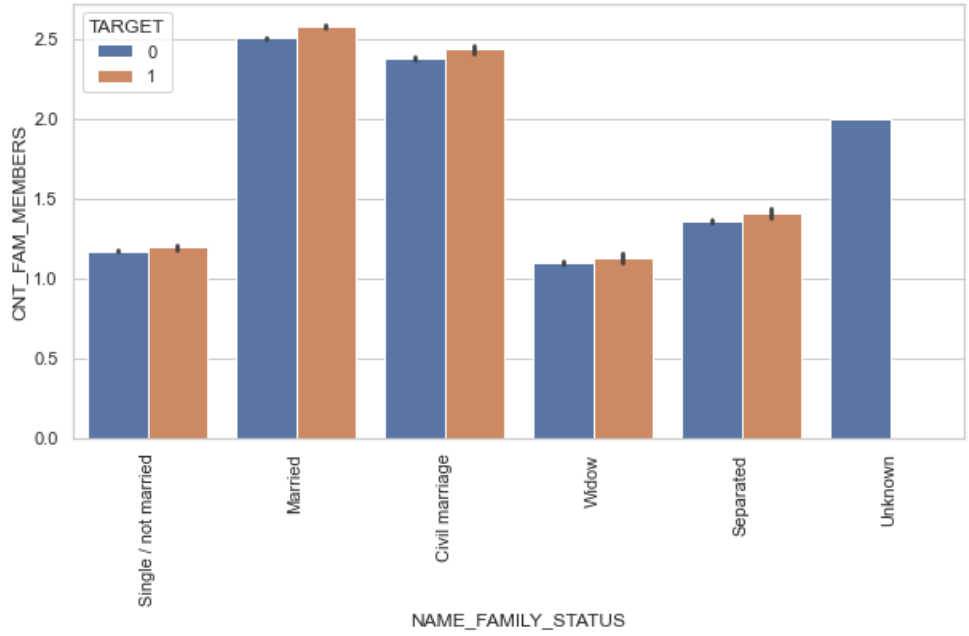
### Family Status vs Count of Children



*Inference:*

Client who are married and has more children (5+), chances to be a defaulter in High. This may be due to the Economic situation of their family, because of more children.

### Family Status vs Count of Family Members




*Inference:*

Client who are married and has more children (5+), chances to be a defaulter in High. This may be due to the Economic situation of their family, because of more children

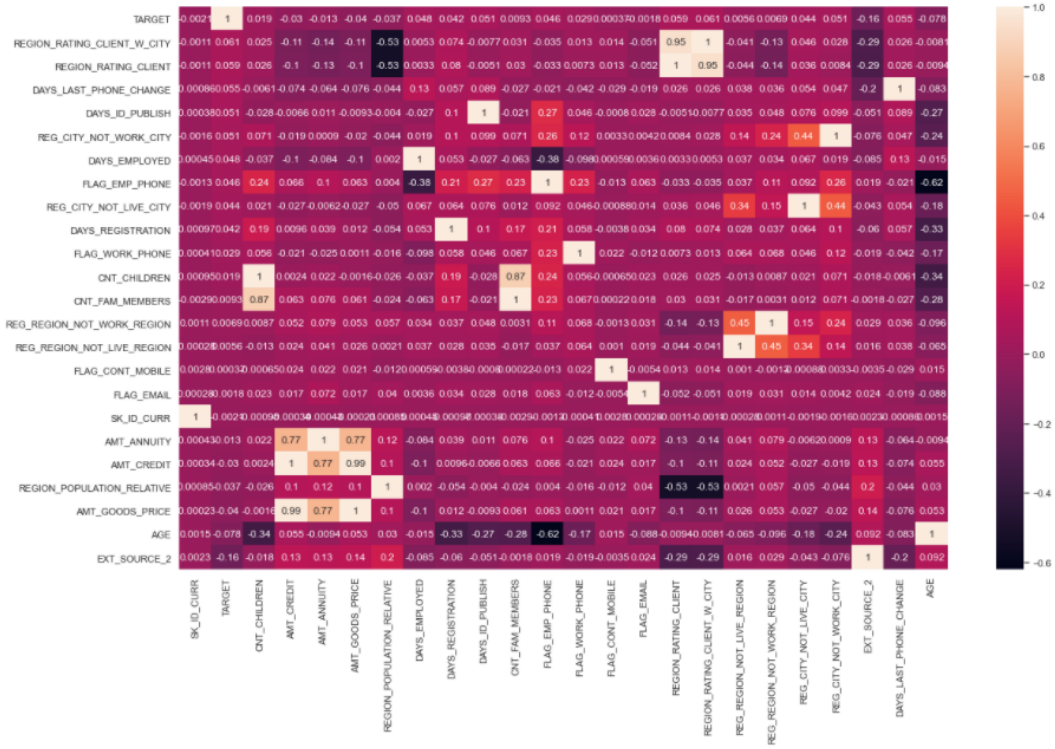
Based on the Bivariate analyses, few columns proved to be of no use, so we are dropping them.

```
In [92]: Application_Data.drop(["HOUR_APPR_PROCESS_START","FLAG_MOBIL"],axis=1,inplace=True)
```

## ANALYSING CORRELATION OF TARGET VARIABLE VS OTHER VARIABLES

```
In [98]: Correlation = Application_Data.corr()
         Correlation.sort_values(by=["TARGET"],ascending=False,inplace=True)
```

```
In [99]: f, ax = plt.subplots(figsize=(20,12))
         sns.heatmap(Correlation,annot=True)
         plt.show()
```




```
In [100]: Correlation.head(6)["TARGET"][1:]

Out[100]: REGION_RATING_CLIENT_W_CITY    0.060893
          REGION_RATING_CLIENT           0.058899
          DAYS_LAST_PHONE_CHANGE         0.055218
          DAYS_ID_PUBLISH                0.051457
          REG_CITY_NOT_WORK_CITY         0.050994
          Name: TARGET, dtype: float64
```

```
In [102]: Correlation.tail(5)["TARGET"]

Out[102]: AMT_CREDIT                  -0.030369
          REGION_POPULATION_RELATIVE  -0.037227
          AMT_GOODS_PRICE             -0.039628
          AGE                         -0.078263
          EXT_SOURCE_2                -0.160303
          Name: TARGET, dtype: float64
```

We observe the Highly Correlated Variables are:

- AMT_CREDIT and AMT_GOODS_PRICE =0.99
- REGION_RATING_CLIENT_W_CITY and REGION_RATING_CLIENT = 0.95
- CNT_FAM_MEMBERS and CNT_CHILDREN = 0.87
- AMT_ANNUITY and AMT_CREDIT = 0.77

## PREVIOUS APPLICATION ANALYSIS

Now, we have taken the second data set -- 'Previous Application Data Set' for our analysis. This contains information about the client's previous loan data. It contains the data whether the previous application had been **Approved, Cancelled, Refused or Unused offer.**

This data set contains 1670214 rows × 37 columns

Similarly, we have proceeded here with Data Cleaning:

- Found out the percentage of null values in each column, to determine what needs to be done as part of data cleaning.
- Calculated the percentage of **NAN** entries in the given Data Frame.
- Iterating over columns in Data Frame and deleting those with where >20% of the values are **NULL**.
- Determining the percentage of **NULL** values in each column.

After doing the above steps, the Shape of the cleaned data frame has come to 23 columns.

```
In [110]: previous_data.shape
Out[110]: (1670214, 23)
```

- After this, we have filled 2% missing value with the Highest Mode in **PRODUCT_COMBINATION** column.

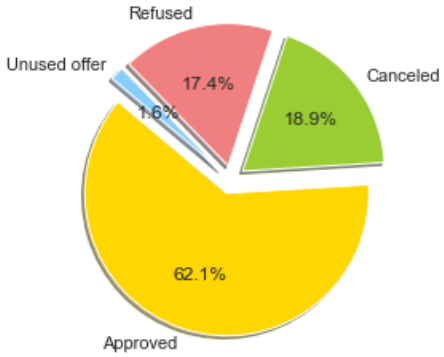
```
In [111]: # Filling 2% missing value with the Highest Mode in PRODUCT_COMBINATION column
previous_data["PRODUCT_COMBINATION"].fillna(previous_data["PRODUCT_COMBINATION"].mode()[0],inplace=True)
```

## As the Data set is ready for analysis, Let's start visualising so as to get some viable inference

### Based on **Contract Status**

Out[113]:

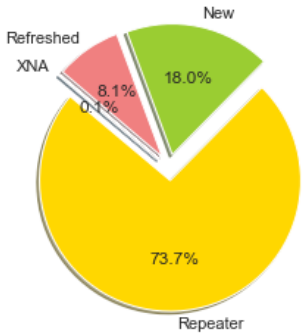
| | NAME_CONTRACT_STATUS | Percentage_of_Values |
|---|---|---|
| 0 | Approved | 62.07 |
| 1 | Canceled | 18.94 |
| 2 | Refused | 17.40 |
| 3 | Unused offer | 1.58 |




### Based on **Client Type**

Out[116]:

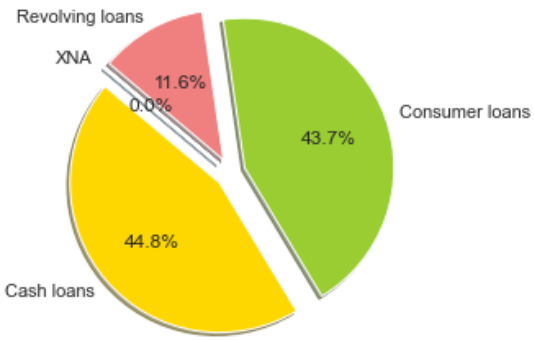
| | NAME_CLIENT_TYPE | Percentage_of_Values |
|---|---|---|
| 0 | Repeater | 73.72 |
| 1 | New | 18.04 |
| 2 | Refreshed | 8.12 |
| 3 | XNA | 0.12 |



Based on **Contract Type**

Out[119]:

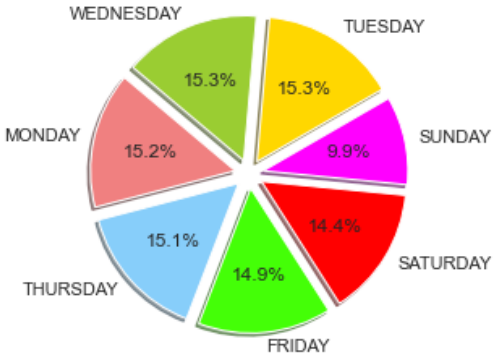
| | NAME_CONTRACT_TYPE | Percentage_of_Values |
|---|---|---|
| 0 | Cash loans | 44.76 |
| 1 | Consumer loans | 43.66 |
| 2 | Revolving loans | 11.57 |
| 3 | XNA | 0.02 |




Based on **Days of Approval - WEEKDAY_APPR_PROCESS_START**

Out[122]:

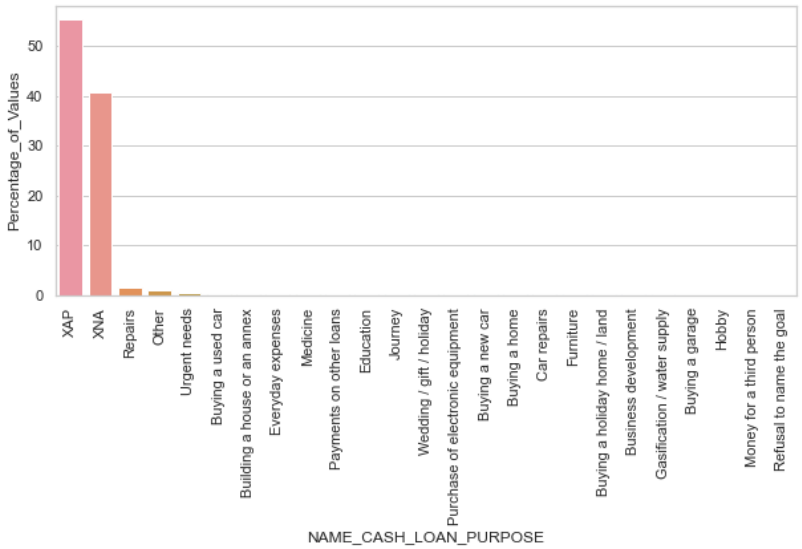
| | WEEKDAY_APPR_PROCESS_START | Percentage_of_Values |
|---|---|---|
| 0 | TUESDAY | 15.27 |
| 1 | WEDNESDAY | 15.27 |
| 2 | MONDAY | 15.18 |
| 3 | FRIDAY | 15.09 |
| 4 | THURSDAY | 14.91 |
| 5 | SATURDAY | 14.41 |
| 6 | SUNDAY | 9.86 |




*Inference:*

Most of the clients have opted to apply loan on Tuesday. It is very interesting to see that applicants are very low on weekends. We would otherwise assume that the applicants would prefer weekends to apply.

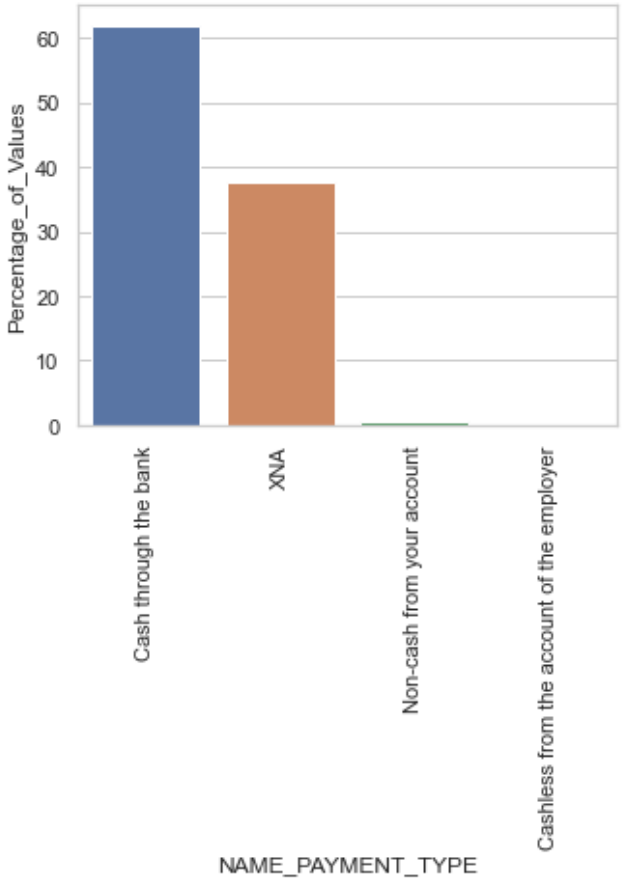
Based on **Purpose of loan - NAME_CASH_LOAN_PURPOSE**



*Inference:*

Most Loan purpose was not recorded. **XAP** and **XNA** values are highest.

Based on **Payment type - NAME_PAYMENT_TYPE**



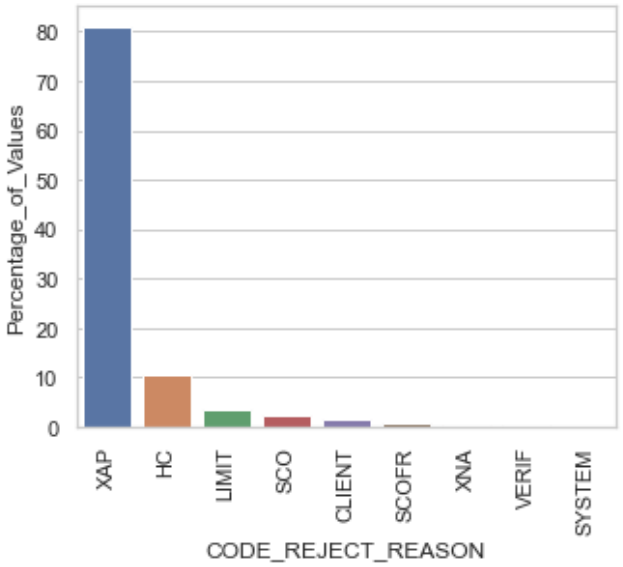

*Inference:*

Most people preferred **CASH (62.44%)** as the mode of Payment.

Based on **Reason of rejection of loan - CODE_REJECT_REASON**

Out[131]:

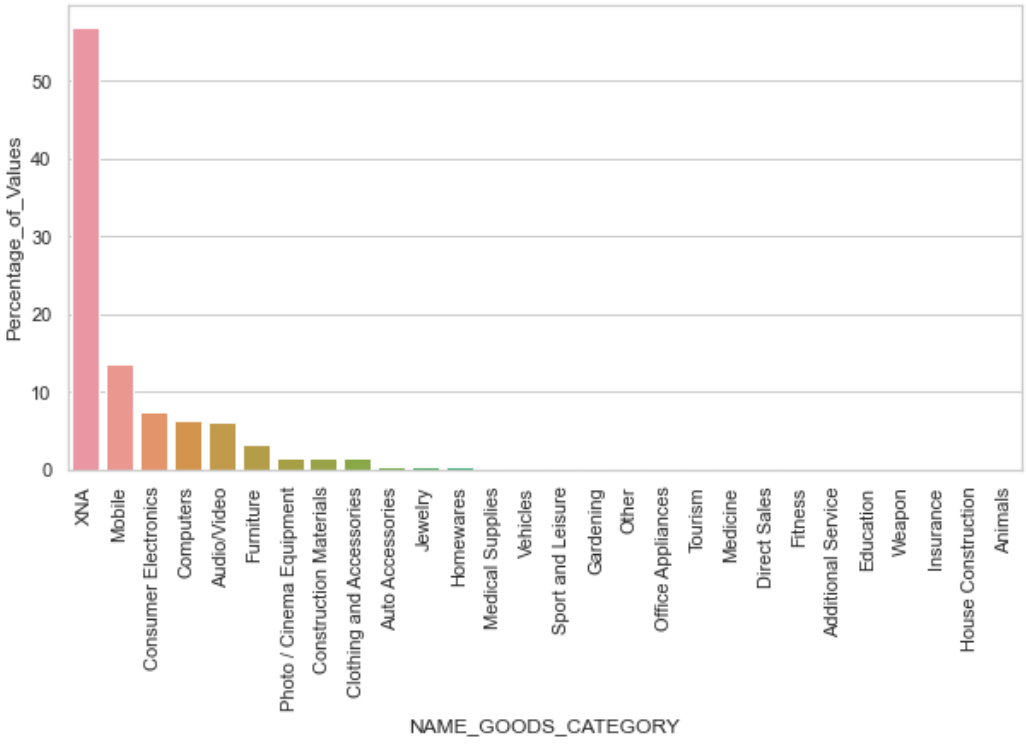
| | CODE_REJECT_REASON | Percentage_of_Values |
|---|---|---|
| 0 | XAP | 81.01 |
| 1 | HC | 10.49 |
| 2 | LIMIT | 3.33 |
| 3 | SCO | 2.24 |
| 4 | CLIENT | 1.58 |
| 5 | SCOFR | 0.77 |
| 6 | XNA | 0.31 |
| 7 | VERIF | 0.21 |
| 8 | SYSTEM | 0.04 |



*Inference:*

Primary reason for the Loan to get rejected is not recorded **(XAP (81%))** followed by **HC**.

Based on **What kind of goods did the client apply for in the previous application - NAME_GOODS_CATEGORY**



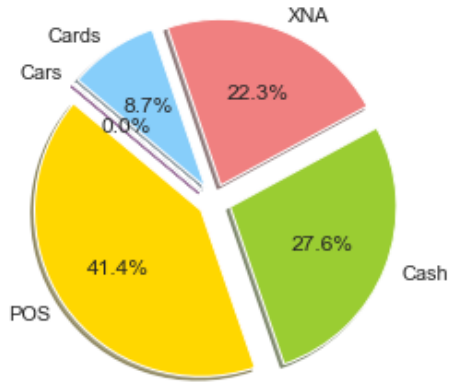

*Inference:*

Most clients applied for Mobile and **53.96%** of the data is not recorded **(XNA).**

Based on **Was the previous application for CASH, POS, CAR, … - NAME_PORTFOLIO**

Out[135]:

| | NAME_PORTFOLIO | Percentage_of_Values |
|---|---|---|
| 0 | POS | 41.37 |
| 1 | Cash | 27.63 |
| 2 | XNA | 22.29 |
| 3 | Cards | 8.68 |
| 4 | Cars | 0.03 |



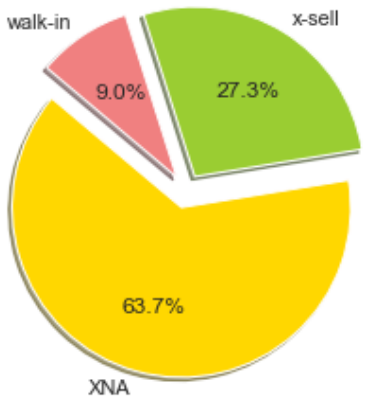

*Inference:*

**41.4%** of the applications were for **POS**.

Based on **Was the previous application x-sell or walk-in - NAME_PRODUCT_TYPE**

Out[137]:

| | NAME_PRODUCT_TYPE | Percentage_of_Values |
|---|---|---|
| 0 | XNA | 63.68 |
| 1 | x-sell | 27.32 |
| 2 | walk-in | 9.00 |



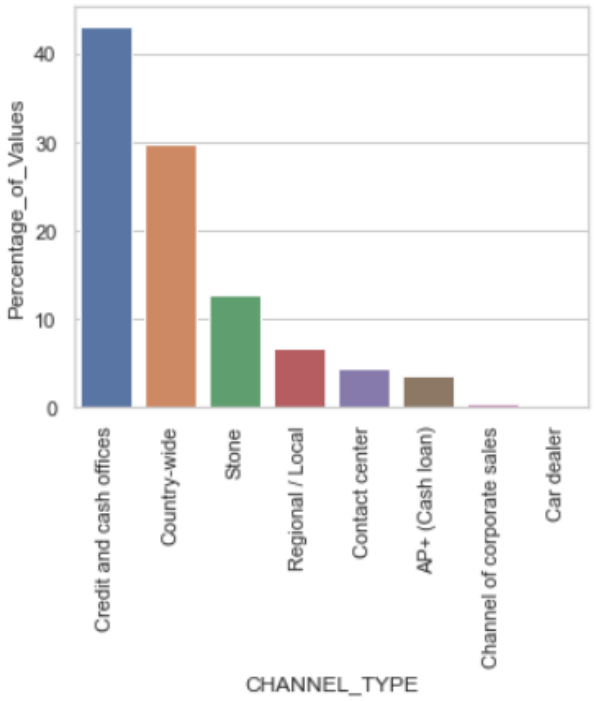

*Inference:*

X-sell applications were more than walk-in

Based on **Through which channel we acquired the client on the previous application - CHANNEL_TYPE**

Out[139]:

| | CHANNEL_TYPE | Percentage_of_Values |
|---|---|---|
| 0 | Credit and cash offices | 43.11 |
| 1 | Country-wide | 29.62 |
| 2 | Stone | 12.70 |
| 3 | Regional / Local | 6.50 |
| 4 | Contact center | 4.27 |
| 5 | AP+ (Cash loan) | 3.42 |
| 6 | Channel of corporate sales | 0.37 |
| 7 | Car dealer | 0.03 |



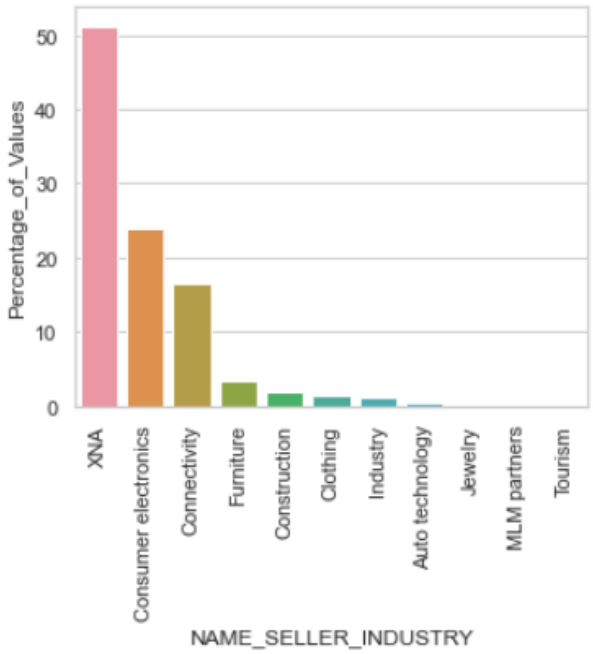

*Inference:*

Most clients were acquired from **Credit and Cash Offices**

Based on **the industry of the seller - NAME_SELLER_INDUSTRY**

Out[141]:

| | NAME_SELLER_INDUSTRY | Percentage_of_Values |
|---|---|---|
| 0 | XNA | 51.23 |
| 1 | Consumer electronics | 23.85 |
| 2 | Connectivity | 16.53 |
| 3 | Furniture | 3.46 |
| 4 | Construction | 1.78 |
| 5 | Clothing | 1.43 |
| 6 | Industry | 1.15 |
| 7 | Auto technology | 0.30 |
| 8 | Jewelry | 0.16 |
| 9 | MLM partners | 0.07 |
| 10 | Tourism | 0.03 |



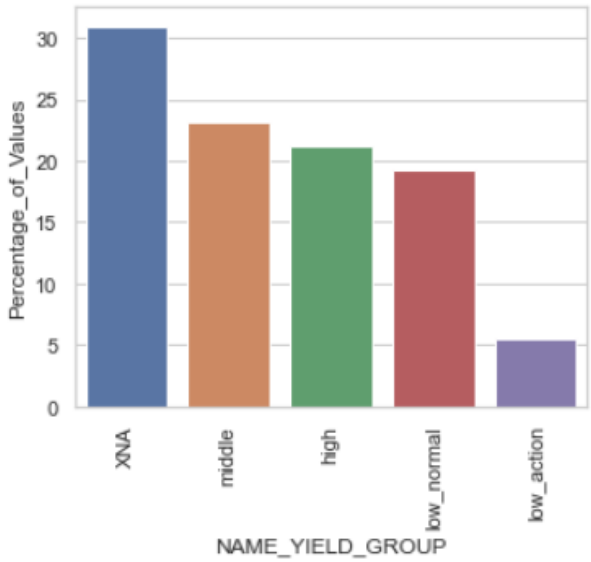

*Inference:*

Most Sellers are from **Consumer electronics**

Based on **Grouped interest rate into small medium and high of the previous application - NAME_YIELD_GROUP**

Out[143]:

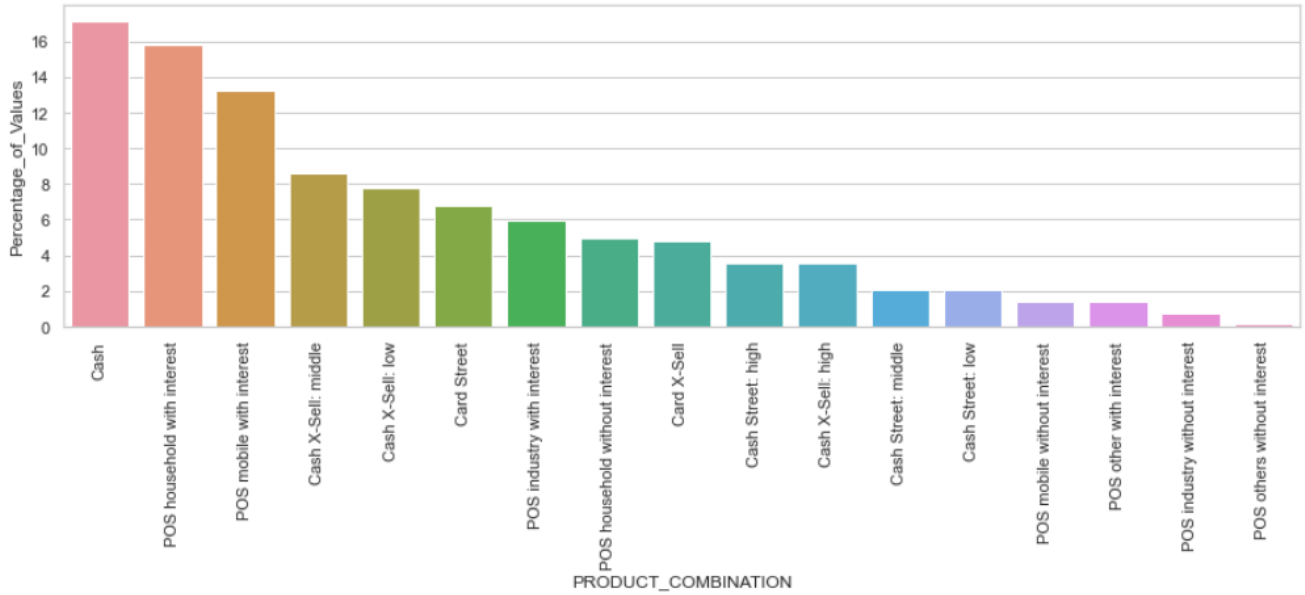
| | NAME_YIELD_GROUP | Percentage_of_Values |
|---|---|---|
| 0 | XNA | 30.97 |
| 1 | middle | 23.08 |
| 2 | high | 21.15 |
| 3 | low_normal | 19.28 |
| 4 | low_action | 5.51 |




*Inference:*

Most group interest rates lie in middle.

Based on **Detailed product combination of the previous application - PRODUCT_COMBINATION**



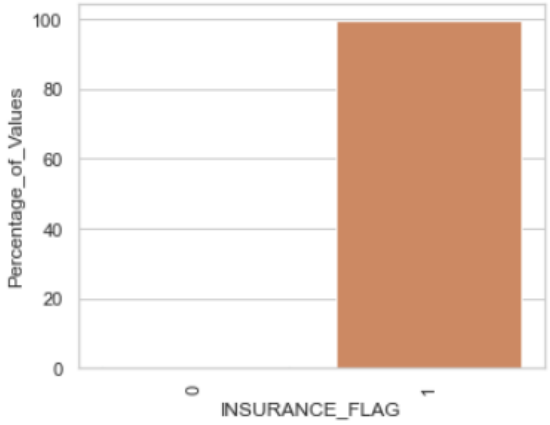

*Inference:*

Highest product combination is **Cash** followed by **POS household with interest**

Based on **Flag if the application was the last application per day of the client - NFLAG_LAST_APPL_IN_DAY**

Out[147]:

| | INSURANCE_FLAG | Percentage_of_Values |
|---|---|---|
| 0 | 1 | 99.65 |
| 1 | 0 | 0.35 |




*Inference:*

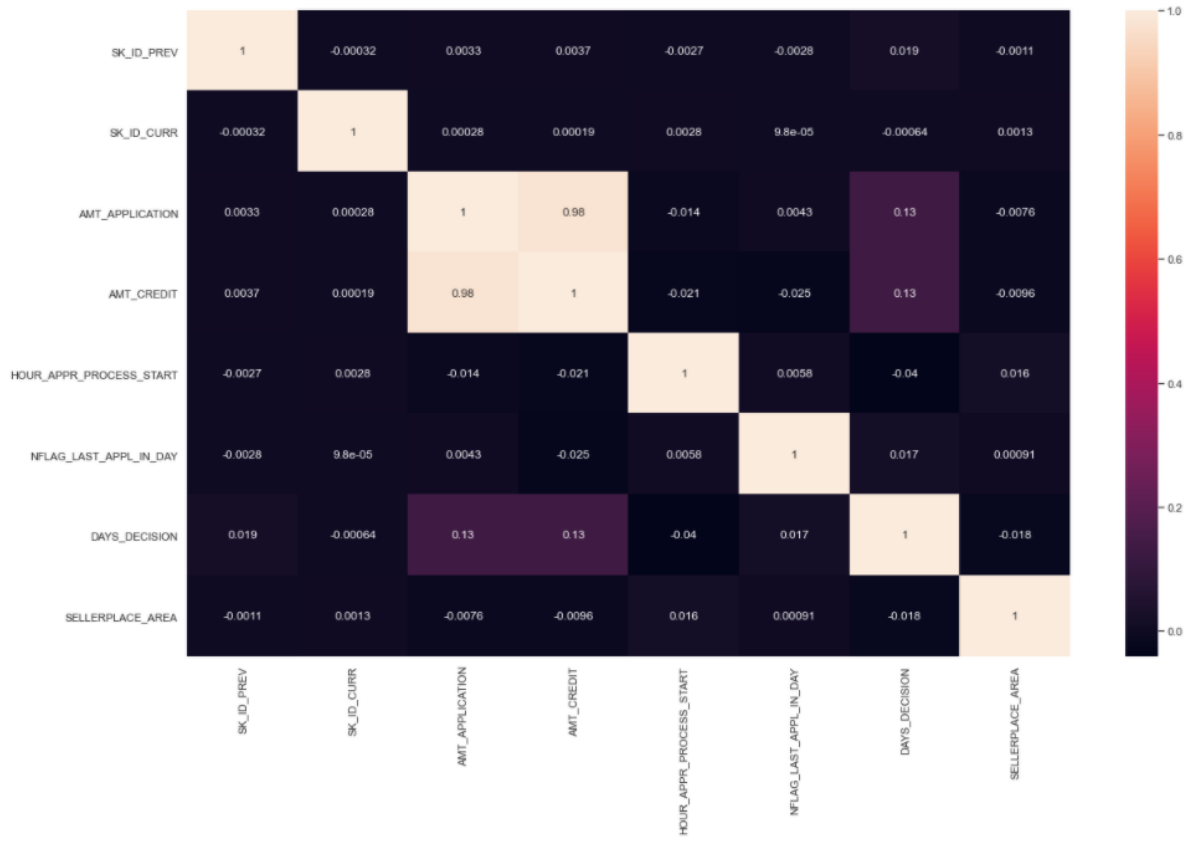
For most clients it was the last application of the day.

## MERGING APPLICATION DATA AND PREVIOUS APPLICATION

### Finding the **Correlation in 'Previous_data'** data frame

In [157]:
```
Correlation = previous_data.corr()
#Correlation.sort_values(by=["TARGET"],ascendingb=False,inplace=True)
f, ax = plt.subplots(figsize=(20,12))
sns.heatmap(Correlation,annot=True)
plt.show()
```

The below plot shows the Correlation of variables in Previous Application



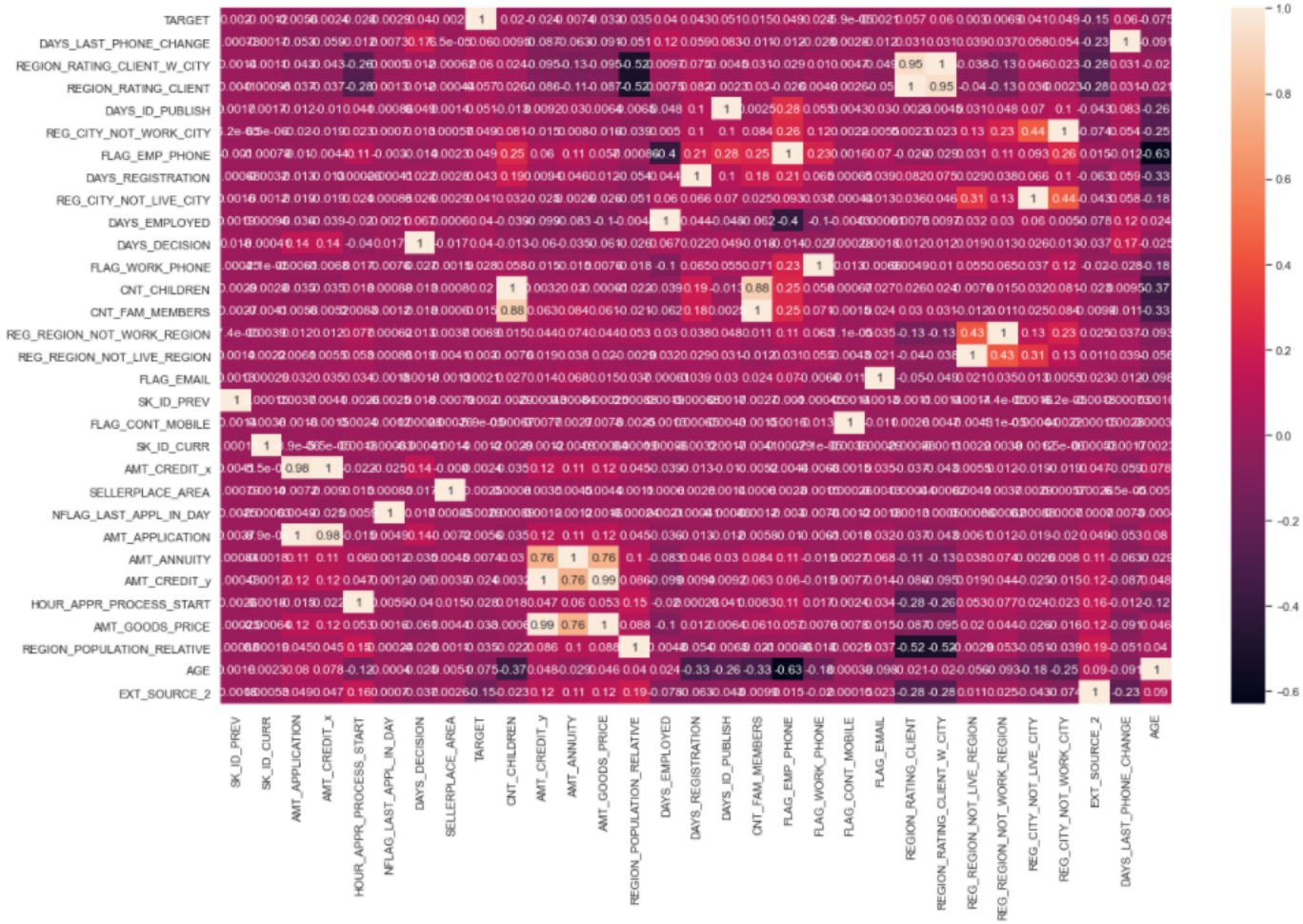
Merging the Application_Data and Previous_Data:

```
In [159]: prev_current_app_df = pd.merge(previous_data,Application_Data,how="inner",on="SK_ID_CURR")
          prev_current_app_df.info()
```

**Correlation between previous_data and application_data data frames**

```
In [162]: Correlation = prev_current_app_df.corr()
          Correlation.sort_values(by=["TARGET"],ascending=False,inplace=True)
          f, ax = plt.subplots(figsize=(20,12))
          sns.heatmap(Correlation,annot=True)
          plt.show()
```

The below shows the Correlation between **Previous_data and Application_data** data frames:




After analyzing all the previous and current applications, we once again checked the correlation of the variable with respect to the Target variable. We got the following results.

```
In [163]: Correlation.head(6)["TARGET"][1:]

Out[163]: DAYS_LAST_PHONE_CHANGE         0.059721
          REGION_RATING_CLIENT_W_CITY    0.059700
          REGION_RATING_CLIENT           0.056932
          DAYS_ID_PUBLISH                0.051037
          REG_CITY_NOT_WORK_CITY         0.049353
          Name: TARGET, dtype: float64
```

```
In [164]: Correlation.tail(6)["TARGET"][1:]

Out[164]: HOUR_APPR_PROCESS_START      -0.027809
          AMT_GOODS_PRICE              -0.032550
          REGION_POPULATION_RELATIVE   -0.035028
          AGE                          -0.074927
          EXT_SOURCE_2                 -0.154919
          Name: TARGET, dtype: float64
```

# MERGING APPLICATION DATA AND PREVIOUS APPLICATION

After analyzing all the previous and current applications, we once again checked the correlation of the variable with respect to the Target variable. We got the following results.

## TOP COORELATION VARIABLES

| | |
|---|---|
| DAYS_LAST_PHONE_CHANGE | 0.059721 |
| REGION_RATING_CLIENT_W_CITY | 0.059700 |
| REGION_RATING_CLIENT | 0.056932 |
| DAYS_ID_PUBLISH | 0.051037 |
| REG_CITY_NOT_WORK_CITY | 0.049353 |

## LOW COORELATION VARIABLES

| | |
|---|---|
| DAYS_LAST_PHONE_CHANGE | 0.059721 |
| REGION_RATING_CLIENT_W_CITY | 0.059700 |
| REGION_RATING_CLIENT | 0.056932 |
| DAYS_ID_PUBLISH | 0.051037 |
| REG_CITY_NOT_WORK_CITY | 0.049353 |

Even here also the same variables, as we seen in our Application data, has been contributing more to the **DEFAULTERS** Identification.