CREDIT EXPLORATORY DATA ANALYSIS CASE STUDY

Executive Post Graduate Programme in Data Science

By SURAJ KHADRI

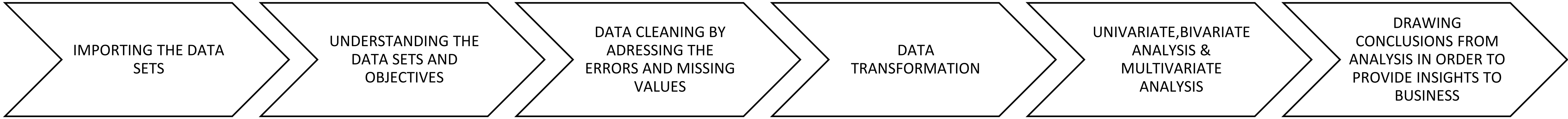
surajkhadri@gmail.com

DS C52 Batch

Business Objective

- This case study aims to identify patterns which indicate if a client has difficulty paying their instalments which may be used for taking actions such as denying the loan, reducing the amount of loan, lending (to risky applicants) at a higher interest rate, etc. This will ensure that the consumers capable of repaying the loan are not rejected. Identification of such applicants using EDA is the aim of this case study.

STEPS INVOLVED IN THE EDA PROCESS



IMPORTING AND UNDERSTANDING ITS STRUCTURES

IMPORT THE NECESSRARY LIBRARIES REQUIRED

```
# imported the libraries.
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="darkgrid")
```

sets the maximum number of columns and rows to be displayed when printing a DataFrame. This is useful when dealing with datasets that have a large number of columns & rows, as it allows you to view more columns & rows at once.

```
pd.set_option('display.max_columns',150)
pd.set_option('display.max_rows',500)
```

LOAD THE DATA

```
# Read the Application Data
application_data =pd.read_csv("application_data.csv")
```

```
# Read the Previous Application Data
previous_application=pd.read_csv("previous_application.csv")
```

DESCRIPTION OF THE DATASET

1. 'application_data.csv' contains all the information of the client at the time of application. The data is about whether a client has payment difficulties.
2. 'previous_application.csv' contains information about the client's previous loan data. It contains the data on whether the previous application had been Approved, Cancelled, Refused or Unused offer.

```
# Checking the loaded data
application_data.head()
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREI
0	100002	1	Cash loans	M	N	Y	0	202500.0	40659
1	100003	0	Cash loans	F	N	N	0	270000.0	129350
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	13500
3	100006	0	Cash loans	F	N	Y	0	135000.0	31268
4	100007	0	Cash loans	M	N	Y	0	121500.0	51300

```
previous_application.head()
```

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	WEEKI
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	0.0	17145.0	
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	NaN	607500.0	
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	NaN	112500.0	

UNDERSTANDING THE STRUCTURE OF THE DATASET

In [9]:
```
application_data.info()
```

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

In [10]:
```
application_data.describe()
```

Out[10]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_POPULATION_RELATIV
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	307499.000000	3.072330e+05	307511.0000
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	27108.573909	5.383962e+05	0.0208
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	14493.737315	3.694465e+05	0.0138
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1615.500000	4.050000e+04	0.0002
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16524.000000	2.385000e+05	0.0100
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	24903.000000	4.500000e+05	0.0188
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.000000	6.795000e+05	0.0286
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	258025.500000	4.050000e+06	0.0725

In [11]:
```
application_data.shape
```

Out[11]: (307511, 122)

In [12]:
```
# Analysing the variables having object data type
application_data.select_dtypes(include=np.object)
```

```
# Analysing the variables having number data type
application_data.select_dtypes(include=np.number)
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_POPULATION_RELATIVE
0	100002	1	0	202500.0	406597.5	24700.5	351000.0	0.018801
1	100003	0	0	270000.0	1293502.5	35698.5	1129500.0	0.003541
2	100004	0	0	67500.0	135000.0	6750.0	135000.0	0.010032
3	100006	0	0	135000.0	312682.5	29686.5	297000.0	0.008019
4	100007	0	0	121500.0	513000.0	21865.5	513000.0	0.028663
...	...	...	...	...	...	...	...	...
307506	456251	0	0	157500.0	254700.0	27558.0	225000.0	0.032561
307507	456252	0	0	72000.0	269550.0	12001.5	225000.0	0.025164
307508	456253	0	0	153000.0	677664.0	29979.0	585000.0	0.005002
307509	456254	1	0	171000.0	370107.0	20205.0	319500.0	0.005313
307510	456255	0	0	157500.0	675000.0	49117.5	675000.0	0.046220

307511 rows × 106 columns

```
application_data.select_dtypes(include=np.object).columns
```

```
Index(['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY',
       'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE',
       'WEEKDAY_APPR_PROCESS_START', 'ORGANIZATION_TYPE', 'FONDKAPREMONT_MODE',
       'HOUSETYPE_MODE', 'WALLSMATERIAL_MODE', 'EMERGENCYSTATE_MODE'],
      dtype='object')
```

DATA CLEANING & MISSING VALUE ANALYSIS AND TREATMENT

DATA CLEANING

MISING VALUE ANALYSIS AND TREATMENT

In [16]:
```
#Checking the  percentage missing values
var_1=round(application_data.isnull().sum().sort_values(ascending=False)*100/application_data.shape[0],2)
var_1
```

```
BASEMENTAREA_AVG            58.52
BASEMENTAREA_MODE           58.52
EXT_SOURCE_1                56.38
NONLIVINGAREA_MODE          55.18
NONLIVINGAREA_AVG           55.18
NONLIVINGAREA_MEDI          55.18
ELEVATORS_MEDI              53.30
ELEVATORS_AVG               53.30
ELEVATORS_MODE              53.30
WALLSMATERIAL_MODE          50.84
APARTMENTS_MEDI             50.75
APARTMENTS_AVG              50.75
APARTMENTS_MODE             50.75
ENTRANCES_MEDI              50.35
ENTRANCES_AVG               50.35
ENTRANCES_MODE              50.35
LIVINGAREA_AVG              50.19
LIVINGAREA_MODE             50.19
LIVINGAREA_MEDI             50.19
HOUSETYPE_MODE              50.18
```

Removing all the columns which consists of missing values more than the treshold limit of 40 pc.

In [17]:
```
for i in application_data:
    if var_1[i] >40:
        application_data.drop(i,axis=1,inplace=True)
```

In [18]:
```
# The Columns having missing values more than 40 pc has been successfully dropped
application_data.shape
```

"Missing" in the place of null values

In [31]:
```
application_data["OCCUPATION_TYPE"]=application_data["OCCUPATION_TYPE"].fillna("Missing")
```

In [32]:
```
# The null values in the categorical variables can be filled with the mode value and the numerical variables can be filled with t
application_data.DAYS_LAST_PHONE_CHANGE = application_data.DAYS_LAST_PHONE_CHANGE.fillna(application_data.DAYS_LAST_PHONE_CHANGE.
application_data.OBS_30_CNT_SOCIAL_CIRCLE = application_data.OBS_30_CNT_SOCIAL_CIRCLE.fillna(application_data.OBS_30_CNT_SOCIAL_C
application_data.DEF_30_CNT_SOCIAL_CIRCLE = application_data.DEF_30_CNT_SOCIAL_CIRCLE.fillna(application_data.DEF_30_CNT_SOCIAL_C
application_data.OBS_60_CNT_SOCIAL_CIRCLE = application_data.OBS_60_CNT_SOCIAL_CIRCLE.fillna(application_data.OBS_60_CNT_SOCIAL_C
application_data.DEF_60_CNT_SOCIAL_CIRCLE = application_data.DEF_60_CNT_SOCIAL_CIRCLE.fillna(application_data.DEF_60_CNT_SOCIAL_C
application_data.EXT_SOURCE_2 = application_data.EXT_SOURCE_2.fillna(application_data.EXT_SOURCE_2.median())
application_data.EXT_SOURCE_3 = application_data.EXT_SOURCE_3.fillna(application_data.EXT_SOURCE_3.median())
application_data.NAME_TYPE_SUITE = application_data.NAME_TYPE_SUITE.fillna(application_data.NAME_TYPE_SUITE.mode()[0])
application_data.AMT_ANNUITY = application_data.AMT_ANNUITY.fillna(application_data.AMT_ANNUITY.median())
application_data.AMT_GOODS_PRICE = application_data.AMT_GOODS_PRICE.fillna(application_data.AMT_GOODS_PRICE.median())
application_data.CNT_FAM_MEMBERS = application_data.CNT_FAM_MEMBERS.fillna(application_data.CNT_FAM_MEMBERS.median())
```

In [33]:
```
application_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

Removing all the columns which consists of missing values more than the threshold limit of 40 pc.

The null values in the categorical variables can be filled with the mode value and the numerical variables can be filled with the median value

DATA TRANSFORMATION & OUTLIERS DETECTION

OUTLIERS DETECTION AND HANDLING

In [49]:
```
application_data.head()
```

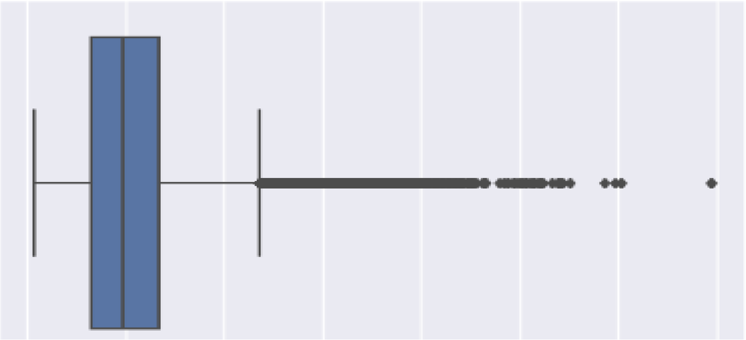
Out[49]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CRE
0	100002	1	Cash loans	M	N	Y	0	202500.0	4065
1	100003	0	Cash loans	F	N	N	0	270000.0	12935
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	1350
3	100006	0	Cash loans	F	N	Y	0	135000.0	3126
4	100007	0	Cash loans	M	N	Y	0	121500.0	5130

In [50]:
```
#creating a function for outliers detection and handling
def outliers_analysis(x):
    print(x.describe())
    print(x.quantile([0.95,0.96,0.97,0.97,0.98,0.99]))
    iqr=x.quantile(0.75)-x.quantile(0.25)
    a=x.quantile(0.25)-1.5*iqr
    b=x.quantile(0.75)+1.5*iqr
    print("All values which are beyond the range of ({},{}) can be considered to be the outliers".format(a,b))
    sns.boxplot(x)
```

In [59]:
```
outliers_analysis(application_data["AMT_ANNUITY"])
```

```
count    295624.000000
mean      26183.569160
std       13215.677577
min        1615.500000
25%       16276.500000
50%       24412.500000
75%       33354.000000
max      173704.500000
Name: AMT_ANNUITY, dtype: float64
0.95    50503.50
0.96    52420.50
0.97    54436.50
0.97    54436.50
0.98    58092.93
0.99    65065.50
Name: AMT_ANNUITY, dtype: float64
All values which are beyond the range of (-9339.75,58970.25) can be considered to be the outliers
```



In [80]:
```
# Here we can categorise all the values above 4 as 4+ using lambda functions
application_data["CNT_CHILDREN"]=application_data["CNT_CHILDREN"].apply(lambda x:"4+" if x >= 4 else x)
```

In [81]:
```
application_data["CNT_CHILDREN"].value_counts()
```

Out[81]:
```
0     200332
1      56584
2      24730
3       3433
4+       502
Name: CNT_CHILDREN, dtype: int64
```

In [82]:
```
application_data["NAME_TYPE_SUITE"].value_counts()
```

Out[82]:
```
Unaccompanied      231841
Family              37337
Spouse, partner     10566
Children             3104
Other_B              1671
Other_A               814
Group of people       248
Name: NAME_TYPE_SUITE, dtype: int64
```

```
#Here we can combine the three columns Other_B,Other_A,Group of people as others column as the the counts are lesser in number
application_data["NAME_TYPE_SUITE"]=application_data["NAME_TYPE_SUITE"].apply(lambda x: "Others" if x == "Other_B" else x)
application_data["NAME_TYPE_SUITE"]=application_data["NAME_TYPE_SUITE"].apply(lambda x: "Others" if x == "Other_A" else x)
application_data["NAME_TYPE_SUITE"]=application_data["NAME_TYPE_SUITE"].apply(lambda x: "Others" if x == "Group of people" else :
```

```
application_data["NAME_TYPE_SUITE"].value_counts()
```

```
Unaccompanied      231841
Family              37337
Spouse, partner     10566
Children             3104
Others               2733
Name: NAME_TYPE_SUITE, dtype: int64
```

```
application_data["NAME_INCOME_TYPE"].value_counts()
```

```
Working                 150069
Commercial associate     62867
Pensioner                52982
State servant            19622
Unemployed                  19
Student                     16
Businessman                  3
Maternity leave              3
Name: NAME_INCOME_TYPE, dtype: int64
```

```
# We can see that the columns such as Unemployed,Student,Businessman and Maternity leave.

application_data["NAME_INCOME_TYPE"]=application_data["NAME_INCOME_TYPE"].apply(lambda x:"Others" if x in ["Unemployed","Student"
```

Function created to do Univariate Analysis on Categorical columns

DEFINING A FUNCTION IN ORDER TO PERFORM UNIVARIATE ANALYSIS ON THE CATEGORICAL VARIABLES

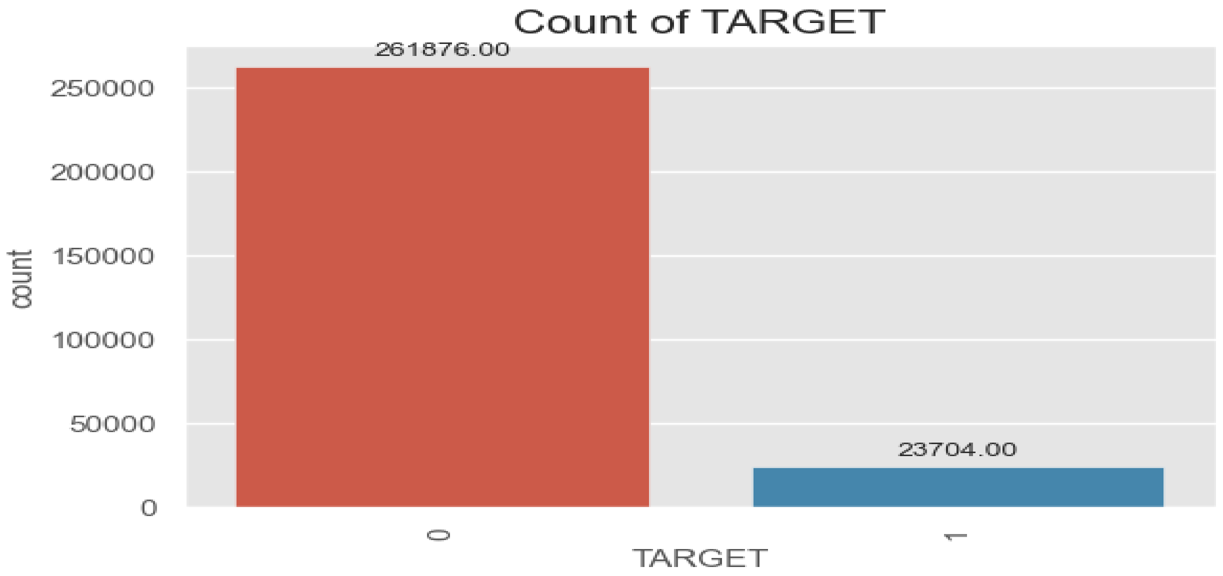
In [181]:
```
def categorical_univariate_analysis(x):
    print('categorical variables univariate analysis of {}'.format(x))

    plt.figure(figsize=[14,4])
    plt.subplot(1,2,1)
    plots = sns.countplot(application_data[x],order = application_data[x].value_counts().index)
    plt.xticks(rotation = 90)
    plt.title('Count of {}'.format(x), fontdict={'fontsize':16})
    for bar in plots.patches:
        plots.annotate(format(bar.get_height(), '.2f'),
                   (bar.get_x() + bar.get_width() / 2,
                    bar.get_height()), ha='center', va='center',
                   size=10, xytext=(0, 8),
                   textcoords='offset points')

    plt.subplot(1,2,2)
    plt.title('Percentage distribution of {}'.format(x), fontdict={'fontsize':18})
    plt.xlabel(x)
    plt.ylabel('Percentage distribution')
    plots = (application_data[x].value_counts()*100/len(application_data[x])).plot.bar()
    for bar in plots.patches:
        plots.annotate(format(bar.get_height(), '.2f'),
                   (bar.get_x() + bar.get_width() / 2,
                    bar.get_height()), ha='center', va='center',
                   size=11, xytext=(0, 8),
                   textcoords='offset points')

    plt.show()
    print('-------------------------------------------------------------------------------------------------------------------
```

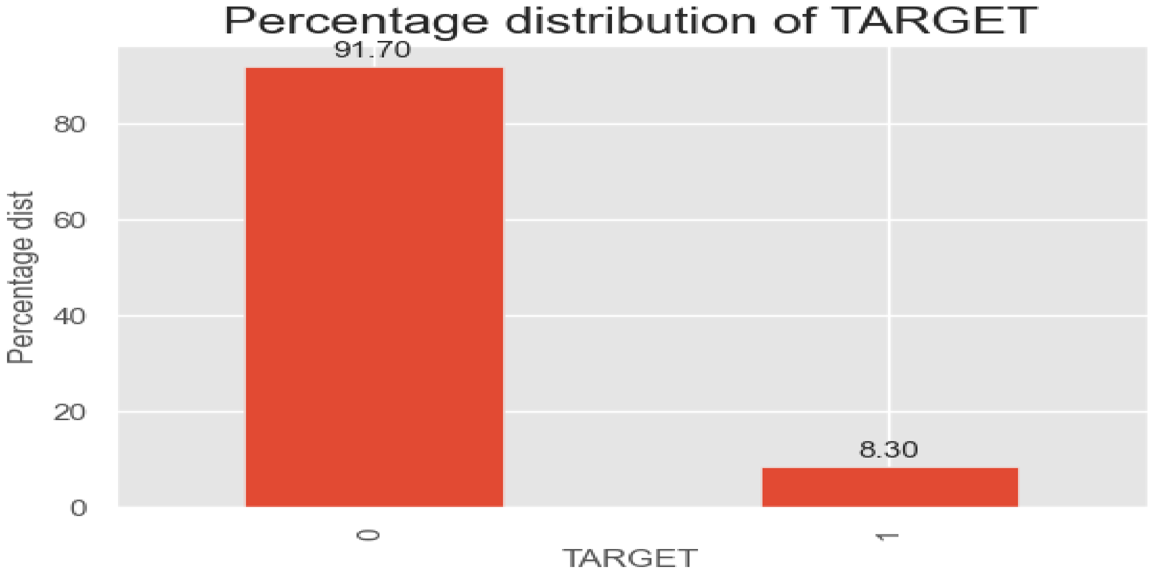
Function created to do Univariate Analysis on Numerical columns

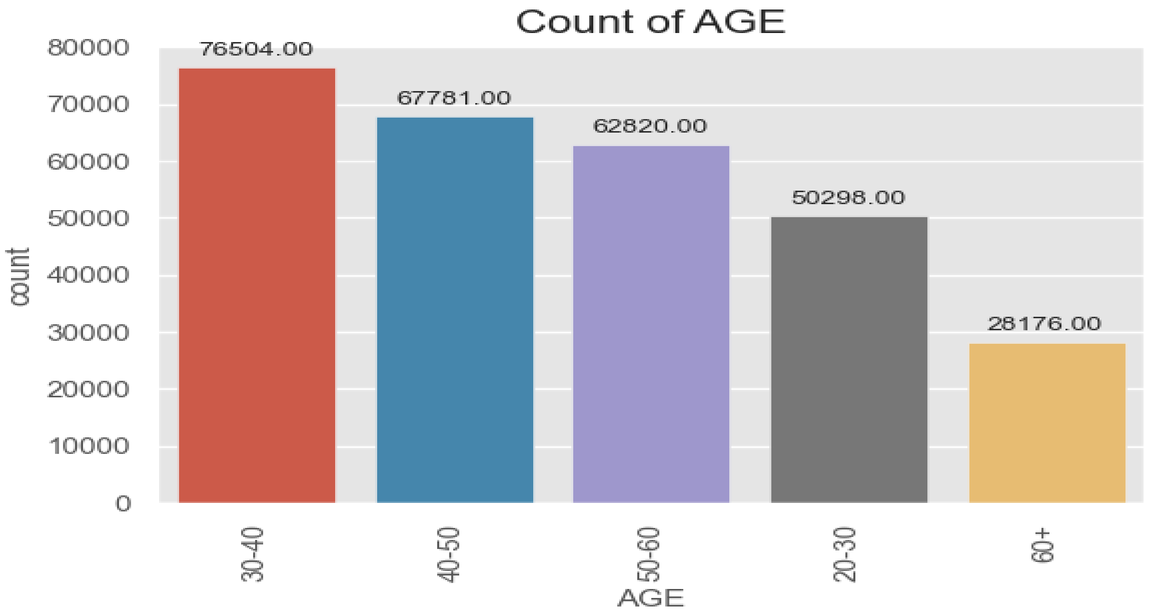
DEFINING A FUNCTION IN ORDER TO PERFORM UNIVARIATE ANALYSIS ON THE NUMERICAL VARIABLES

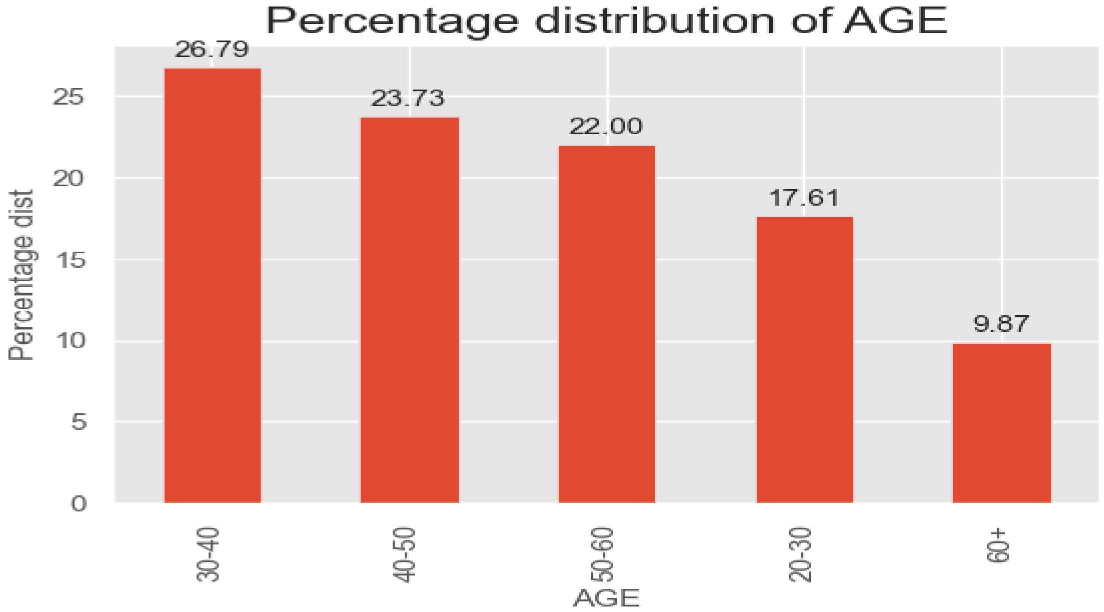
In [182]:
```
def numerical_univariate_analysis(x):
    print('Numerical Univariate Analysis of {}'.format(x))
    print(application_data[x].describe())
    plt.figure(figsize=[16,6])
    plt.subplot(1,2,1)
    sns.boxplot(application_data[x])
    plt.title('Boxplot of {}:'.format(x),fontdict={'fontsize':16})

    plt.subplot(1,2,2)
    sns.distplot(application_data[x],hist=False)
    plt.title('distplot of {}:'.format(x),fontdict={'fontsize':16})


    plt.show()
    print('-----------------------------------------------------------------------------------------------------------
```

DATA VISUALISATION OBTAINED FROM THE FUNCTIONS ON CATEGORICAL COLUMNS

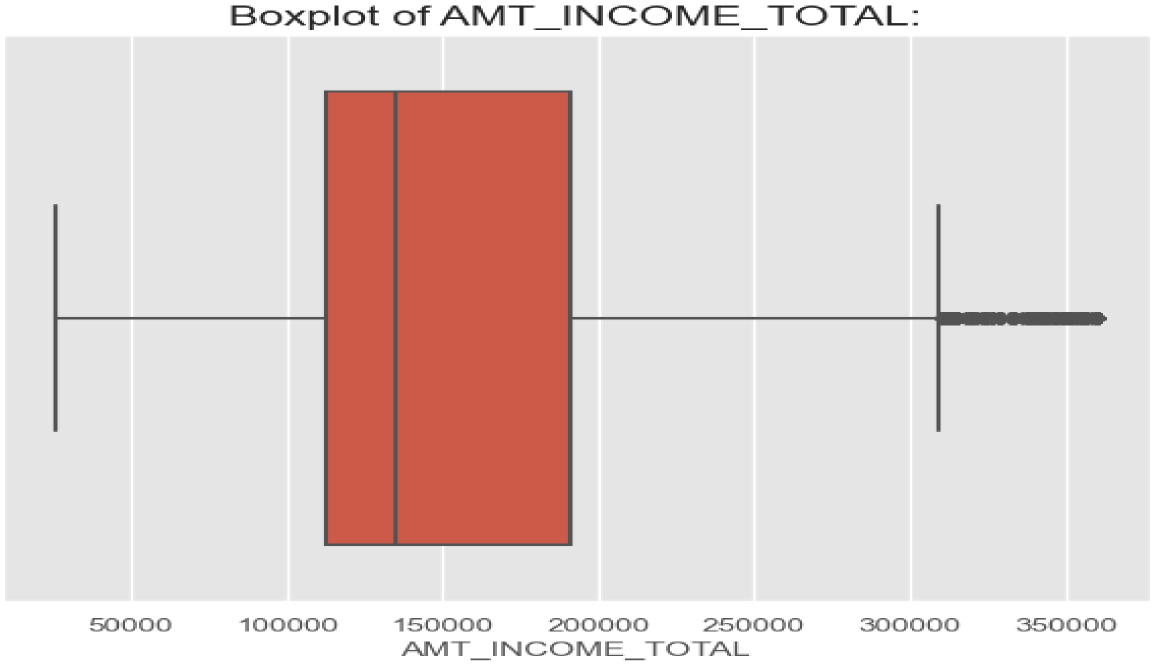


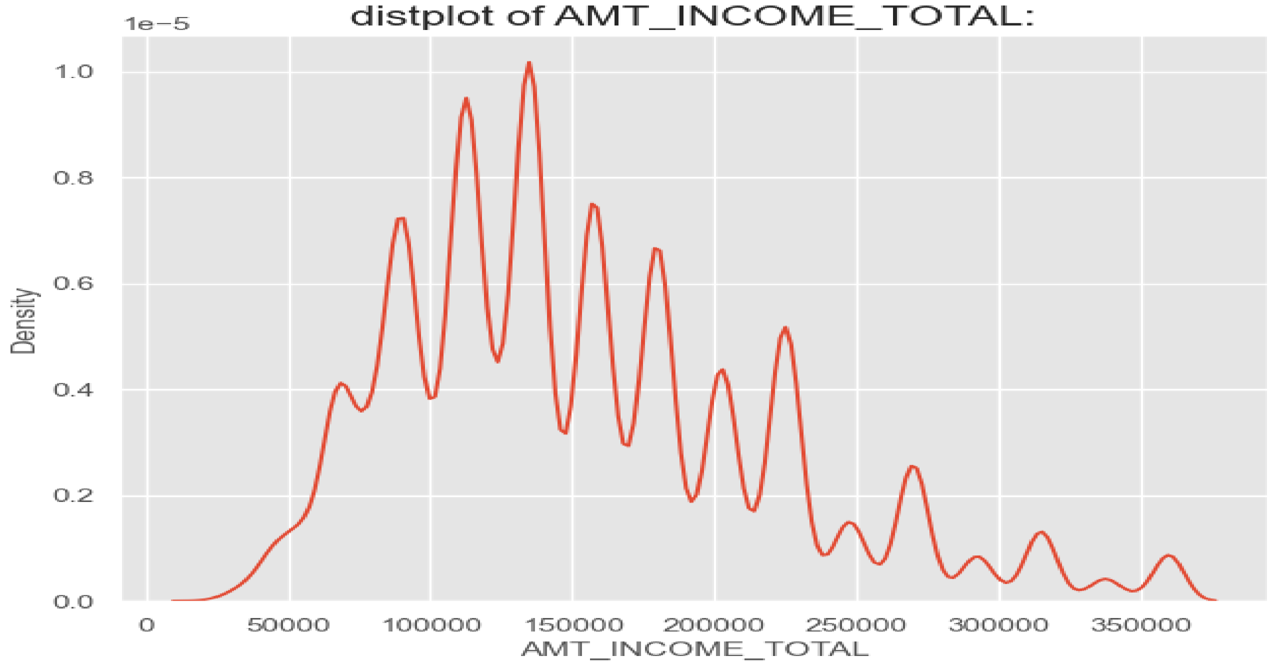


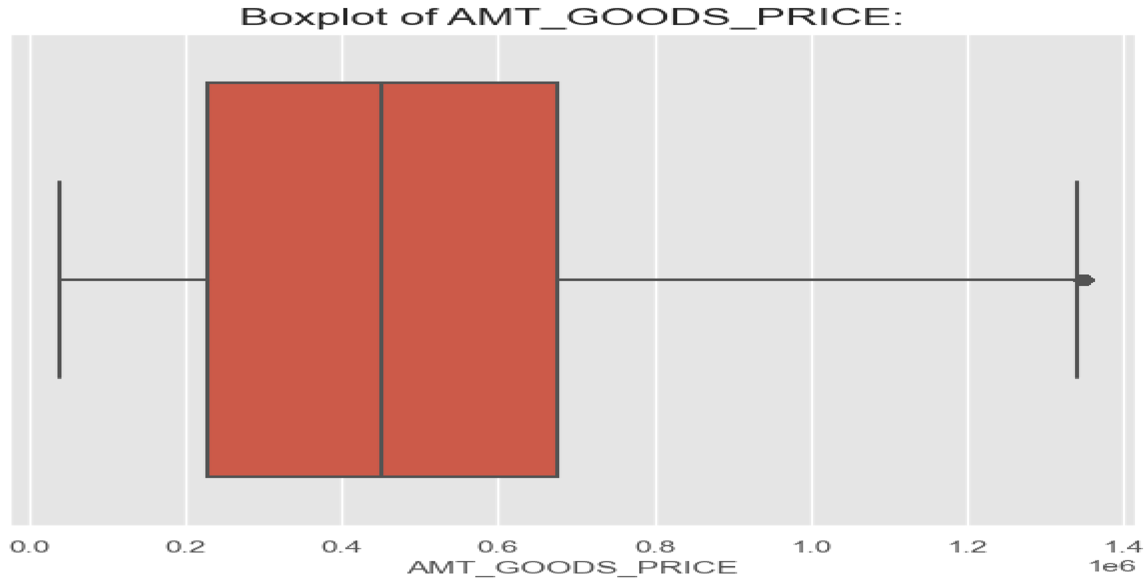


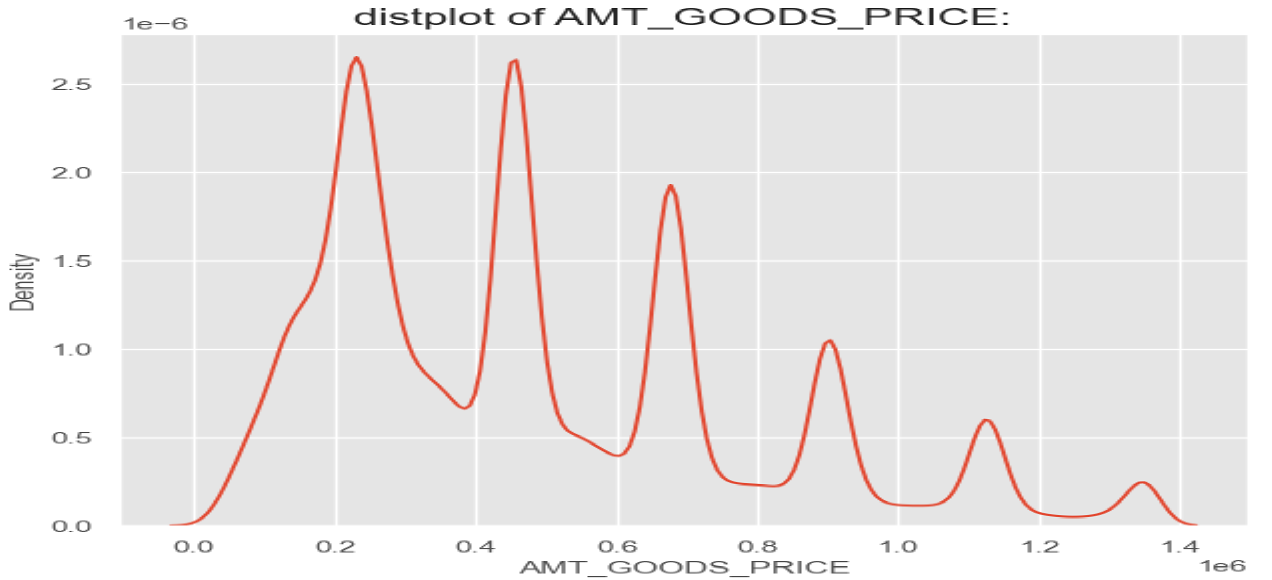


DATA VISUALISATION OBTAINED FROM THE FUNCTIONS ON NUMERICAL COLUMNS







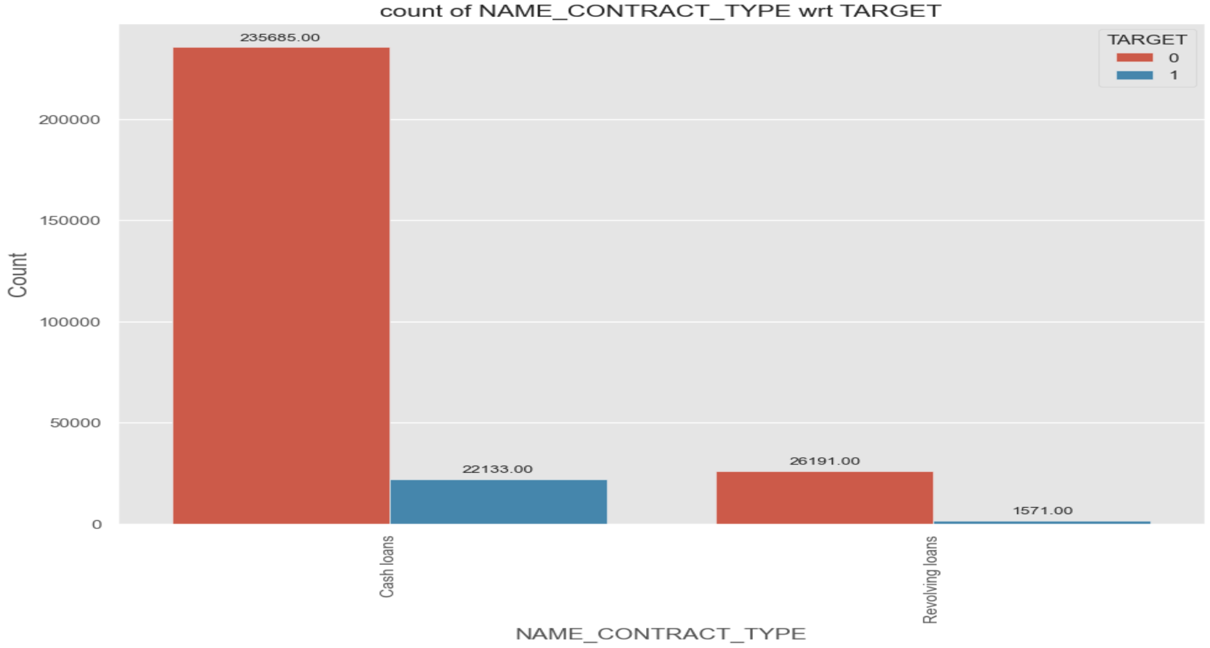


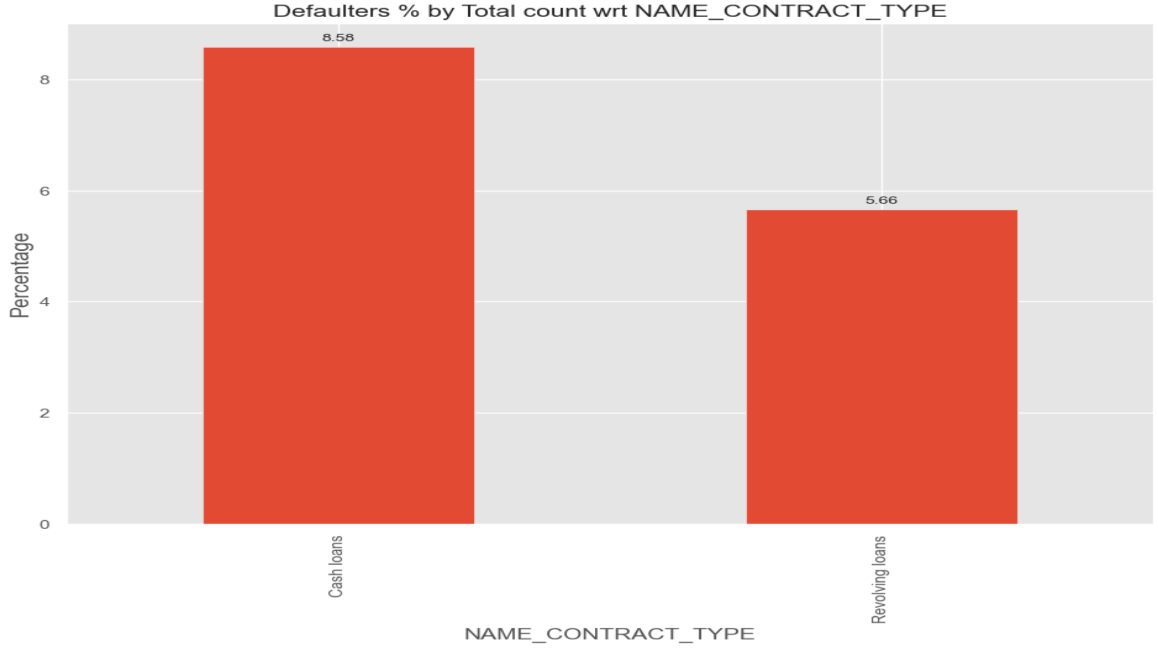
Segmented Univariate Analysis and Bivariate Analysis

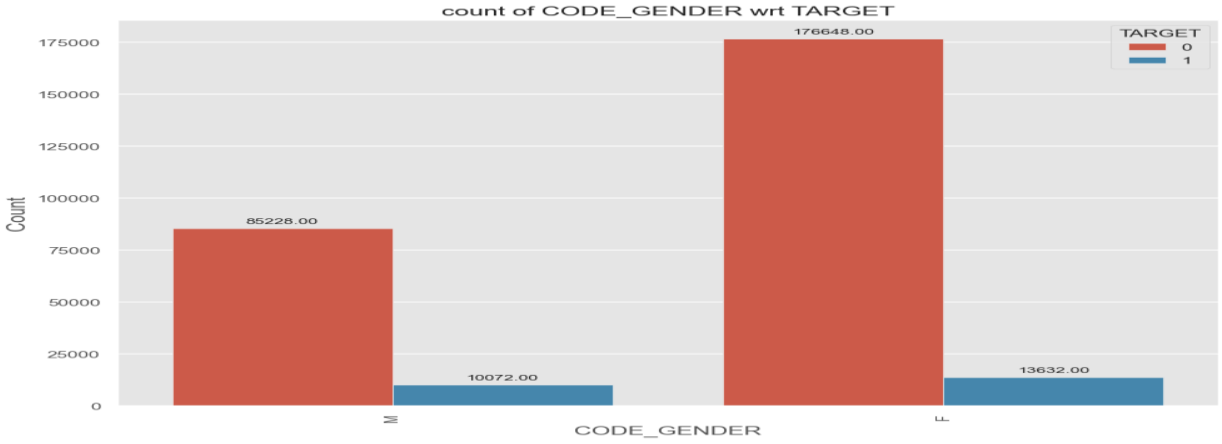
Creating the functions for categorical varialbles by segmenting the defaulters and Non-defaulters and performing the univariate analysis in order to understand the distribution of diffrents categories and analysis based on segments(defaulters and Non-defaulters)

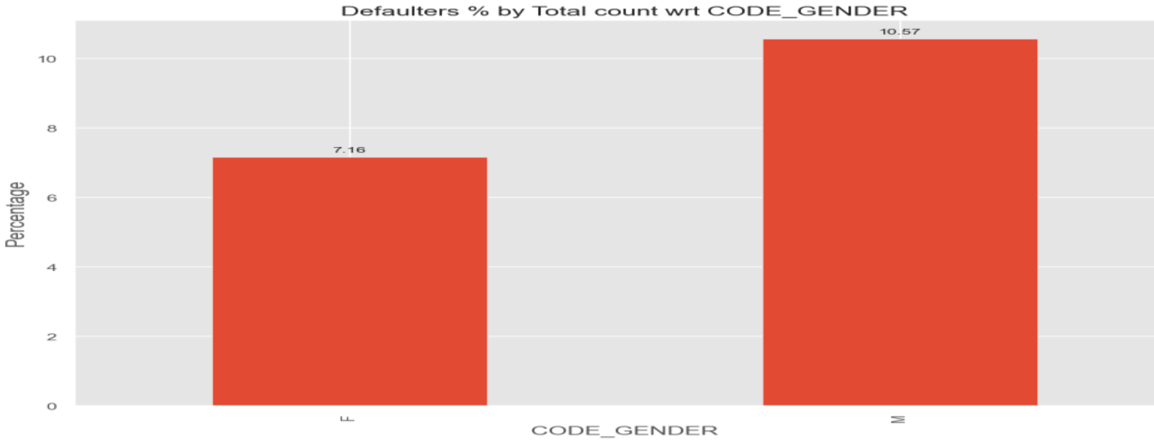
```
def categorical_univariate_analysis_segmented(x):
    print('categorical univariate analysis (segmented) of {}'.format(x))
    print('')

    plt.figure(figsize=[25,8])

    plt.subplot(1,2,1)
    plots = sns.countplot(x,hue='TARGET',data=application_data)
    plt.xticks(rotation = 90)
    plt.title('count of {} wrt TARGET'.format(x), fontdict={'fontsize':15})
    plt.xlabel(x,fontdict={'fontsize':14})
    plt.ylabel('Count',fontdict={'fontsize':15})
    for bar in plots.patches:
        plots.annotate(format(bar.get_height(), '.2f'),
                       (bar.get_x() + bar.get_width() / 2,
                        bar.get_height()), ha='center', va='center',
                       size=10, xytext=(0, 8),
                       textcoords='offset points')

    plt.subplot(1,2,2)
    plots = (application_data[x][application_data['TARGET']==1].value_counts()*100/application_data[x].value_counts()).plot.bar(
    plt.xticks(rotation = 90)
    plt.title('Defaulters % by Total count wrt {}'.format(x), fontdict={'fontsize':15})
    plt.xlabel(x,fontdict={'fontsize':14})
    plt.ylabel('Percentage',fontdict={'fontsize':15})
    for bar in plots.patches:
        plots.annotate(format(bar.get_height(), '.2f'),
                       (bar.get_x() + bar.get_width() / 2,
                        bar.get_height()), ha='center', va='center',
                       size=10, xytext=(0, 8),
                       textcoords='offset points')

    plt.show()
    print('-----------------------------------------------------------------------------------------------------
```

DATA VISUALISATION OBTAINED FROM THE FUNCTIONS ON CATEGORICAL SEGMENTED COLUMNS









INFERENCES AND CONCLUSIONS FROM ANALYSIS

- 1) Male gender tends to default the loan when compared to the female genders
- 2) Lesser the number of children, occurence of defaulting the loan is also is less
- 3) From NAME_TYPE_SUITE column it can be noted that others which consists of have higher percentage of the defaulters
- 4) From NAME_INCOME_TYPE column it can be noted that the others column consisting of
- ['Unemployed', 'Student', 'Businessman', 'Maternity leave'] have highest defaulters followed by the working class and the pensioners are the least defaulters
- 5) From NAME_EDUCATION_TYPE coulmn it can be noted that the people having education level of lower secondary,incomplete higher and secondary education are among the high defaulters whereas the people with academic degree tends to be least defaulters
- 6) From NAME_HOUSING_TYPE it can be noted that people living in rented apartmets or people living with their parents have higher defaulting rates. Whereas people living in Office Apartments and House have lesser defaulting rates maybe because the fact office apartment charges may be borne by the office
- 7) From REGION_POPULATION_RELATIVE it can be noted that as the population increases the defaulters percentage decreases thi may be due to the growth factor in the populated regions

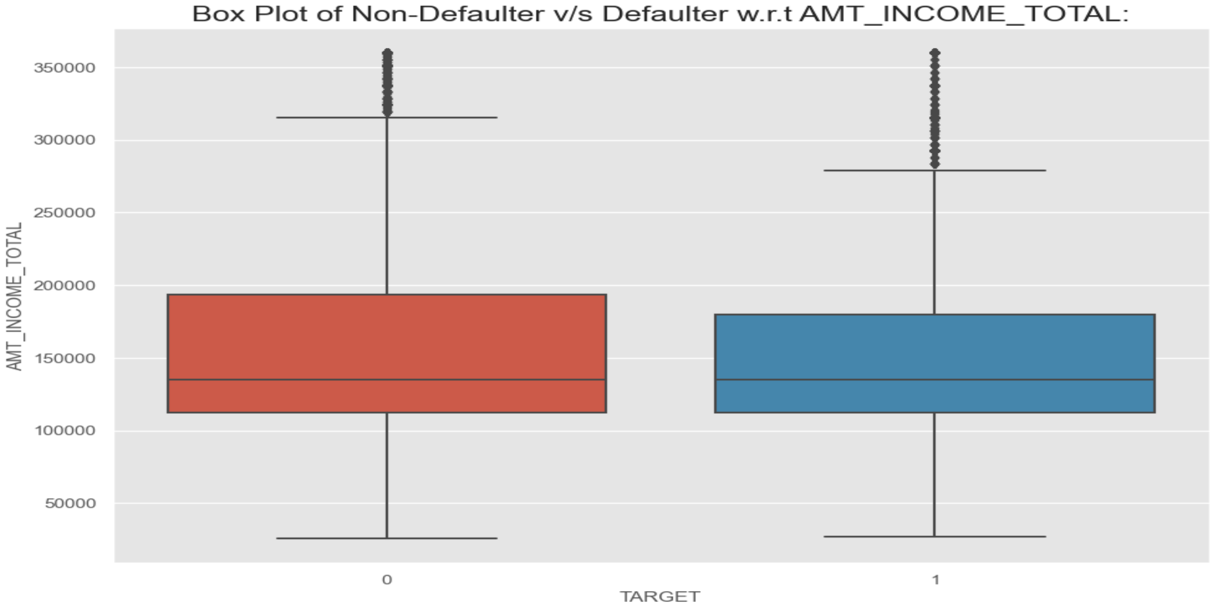
- 8) From AGE column it can be noted that as the age group increases the defaulters percentage decreases which means that the young people have difficulty in making the repayment
- 9) From YEARS_EMPLOYED it can be noted that as the years employed increases the percentage of defaulters decreases may be due to the fact that as people gets more experienced they can pay easily the repayments when compared to the freshers.
- 10) From YEARS_ID_PUBLISH it can be noted that as the age group increases the defaulters percentage decreases
- 11) From OCCUPATION_TY it can be noted that the occupations such as Low skill laborer's, drivers, waiters, laborer, security staff etc. have higher percentage of defaulters when compared to accountants ,core staffs, HR staff, managers, it staff and medical staffs this may be due to the fact that the accountants, core staffs, HR staff, managers, it staff and medical staffs have higher income as compared to Low skill laborer's, drivers, waiters ,laborer's ,security staff
- 12) From INCOME_GROUPS we can note that the people having very low income finds it difficult to repay the loan when compared to very income groups
- 13) From OWNER_CAR_OR_REALTY we can note that people who have no car or realty tends to be in higher percentage of defaulters whereas people who own both car and realty are in lower percentage of defaulters.

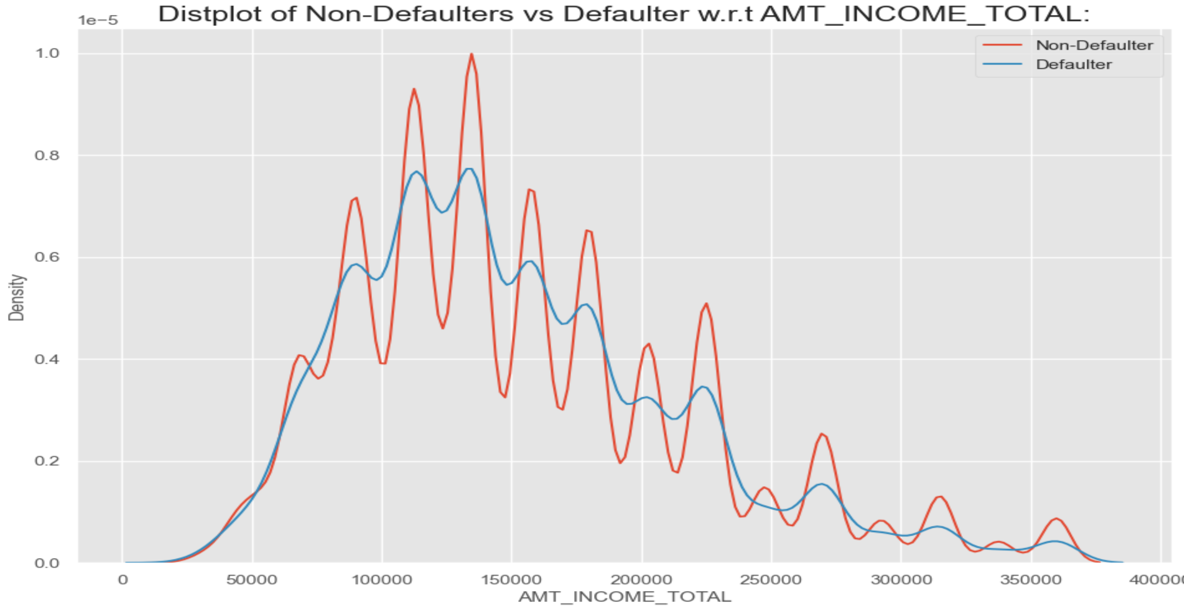
Creating the functions for Numerical varialbles by segmenting the defaulters and Non-defaulters and performing the univariate analysis in order to understand the distribution of diffrents categories and analysis based on segments(defaulters and Non-defaulters

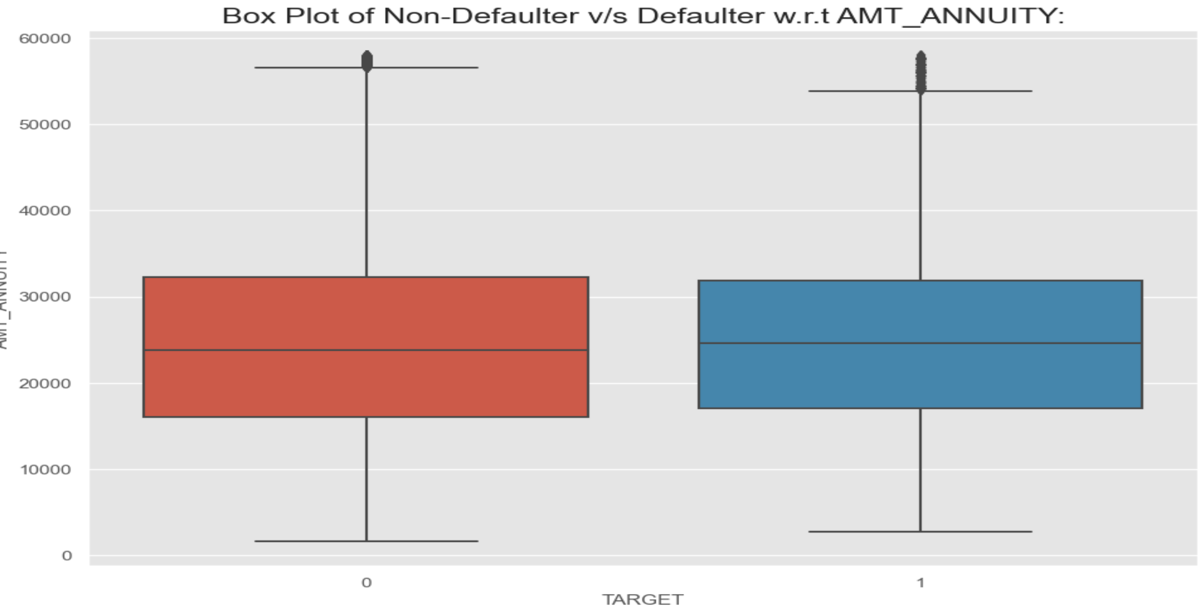
In [205]:
```
def segmented_numerical_univariate_analysis(x):
    print('Numerical Univariate Analysis of Defaulters with regard to {}'.format(x))
    print(round(df1[x].describe(),2))
    print('')
    print('Numerical Univariate Analysis of Non-Defaulters with regard to {}'.format(x))
    print(round(df0[x].describe(),2))

    plt.figure(figsize=[25,8])

    plt.subplot(1,2,1)
    sns.boxplot(x=application_data.TARGET,y=application_data[x])
    plt.title('Box Plot of Non-Defaulter v/s Defaulter w.r.t {}:'.format(x),fontdict={'fontsize':18})

    plt.subplot(1,2,2)
    sns.distplot(df0[x],hist=False,label='Non-Defaulter')
    sns.distplot(df1[x],hist=False,label='Defaulter')
    plt.title('Distplot of Non-Defaulters vs Defaulter w.r.t {}:'.format(x),fontdict={'fontsize':18})


    plt.legend()
    plt.show()
    print('-----------------------------------------------------------------------------------------
```

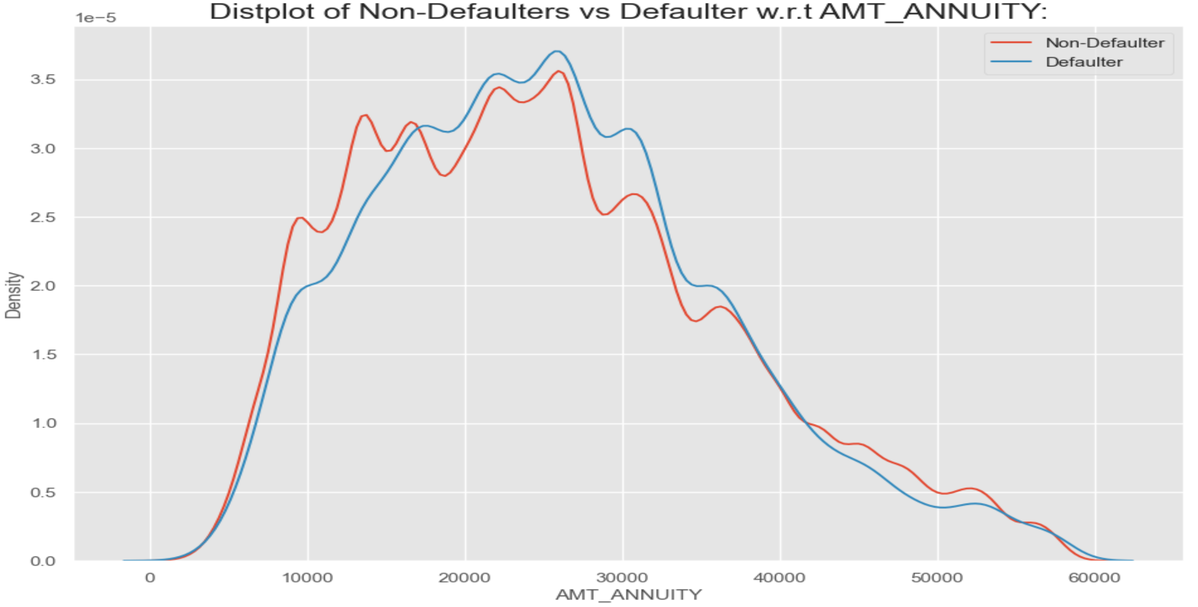
In [207]:
```
#Listing out the numerical columns in the dataset
Numerical_columns=['AMT_INCOME_TOTAL','AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE','EXT_SOURCE_2','EXT_SOURCE_3']
```

DATA VISUALISATION OBTAINED FROM THE FUNCTIONS ON NUMERICAL SEGMENTED COLUMNS





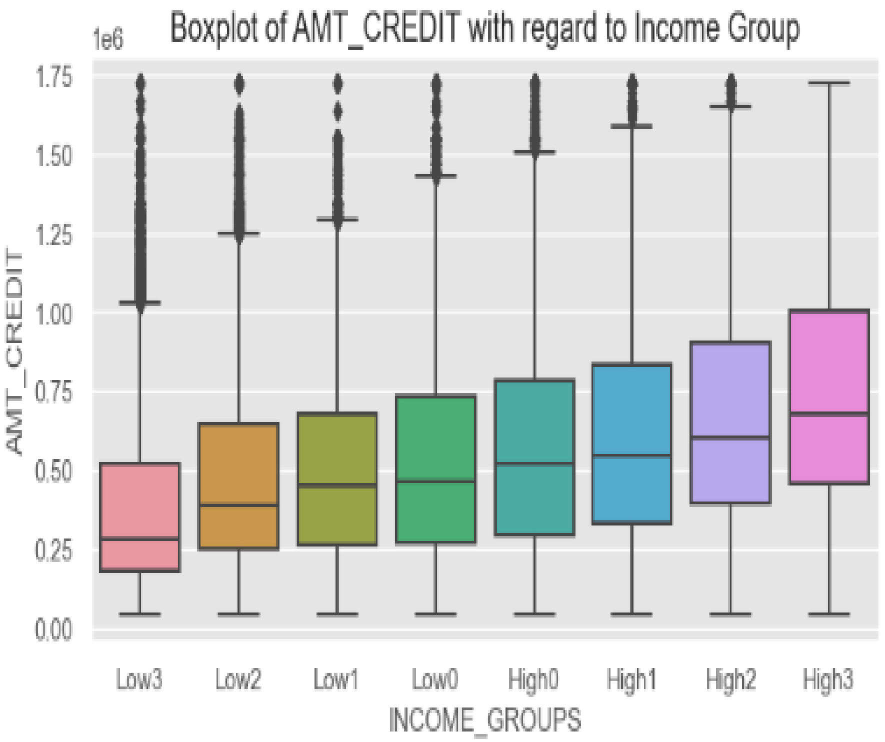


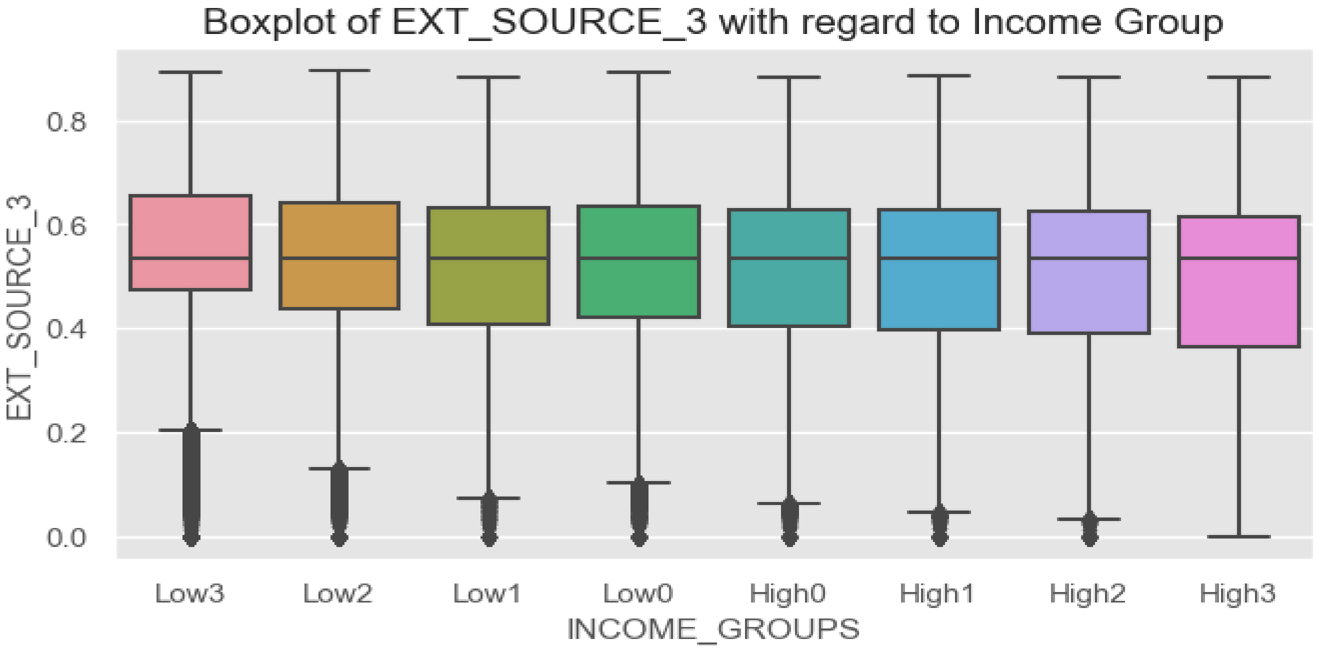


Box plot to analyse between numerical variables and income group

BIVARIATE ANALYSIS BETWEEN TWO VARIABLES

```
#lets create box plot to analyse between numerical variables and income group
a=0
numerical = ['AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE','EXT_SOURCE_2','EXT_SOURCE_3']
for i in numerical:
    a = a + 1
    fig = plt.figure(figsize=(17,13))
    plt.subplot(3,2,a)
    plt.title('Boxplot of {} with regard to Income Group'.format(i))
    plt.xlabel(i)
    sns.boxplot(x='INCOME_GROUPS',y=i,data=application_data)


plt.show()
```





CONCLUSIONS AND INFERENCES DRAWN FROM THE ANALYSIS

1) As the income increases from low to high the credit amount will also increase

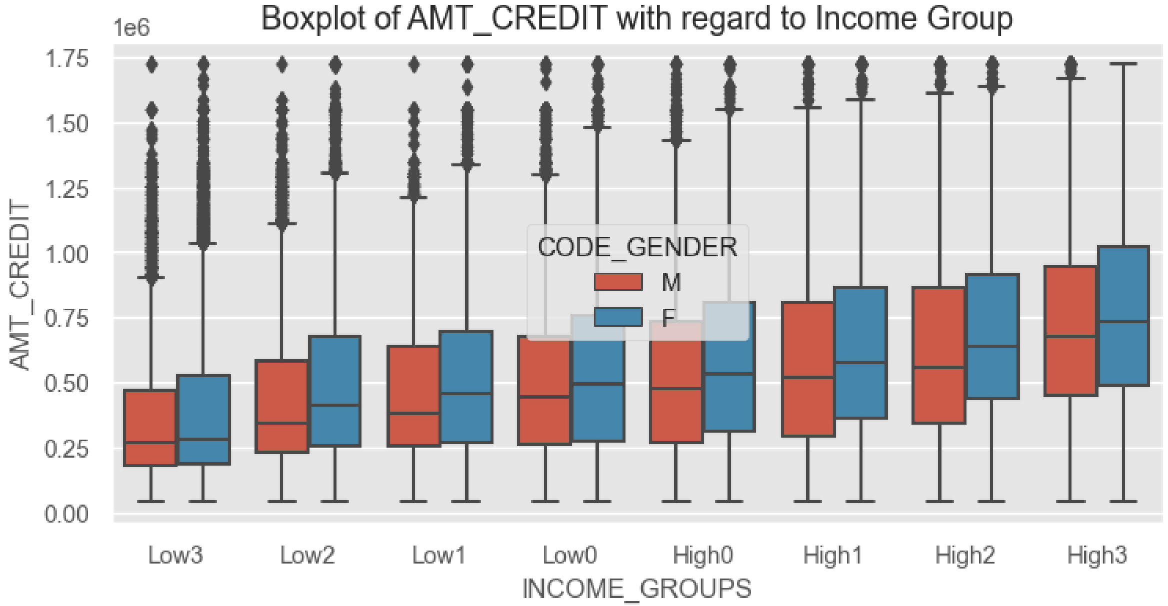
2) As the income increases from low to high the annuity amount will also increase this is because the credit would also get increased since they both are co-related

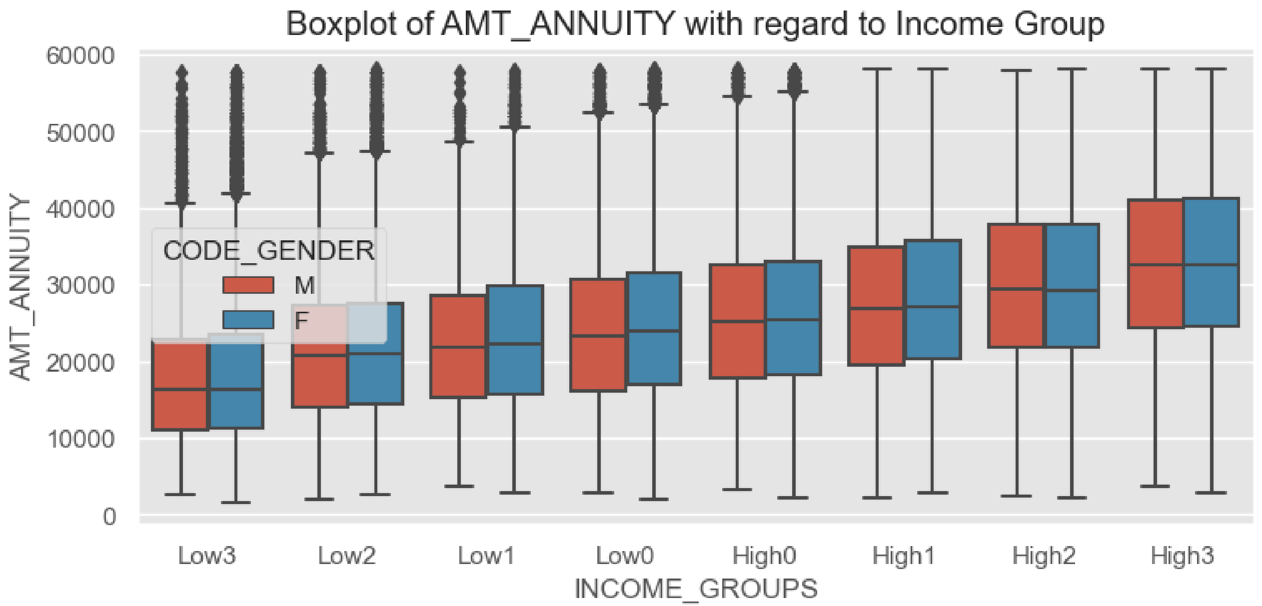
3) People with higher income buy goods with heavy prices thereby as the income increases the amount borrowed for the goods will also increase

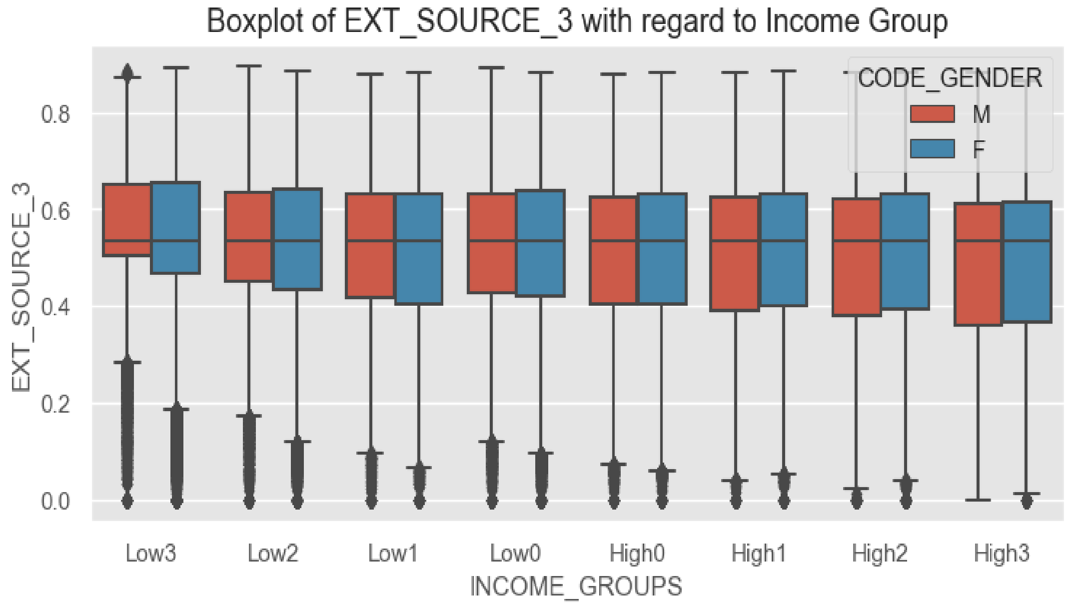
4)As Income increases, EXT_SOURCE_2 also increases.

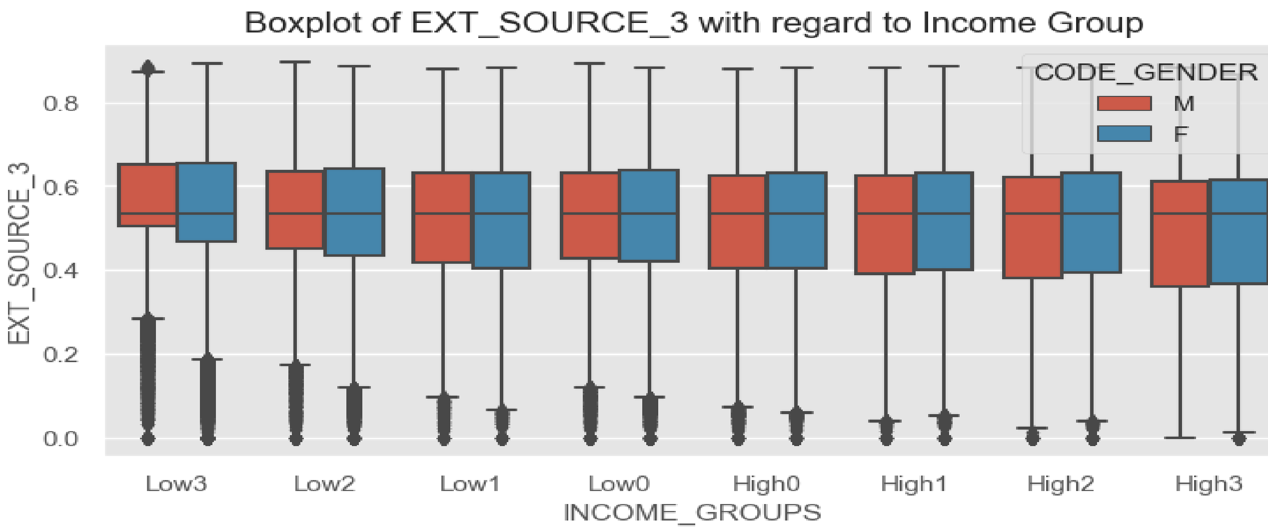
5)As Income increases, EXT_SOURCE_3 decreases.

created box plot to analyze between numerical variables and income group in order to understand the gender difference between the men and women

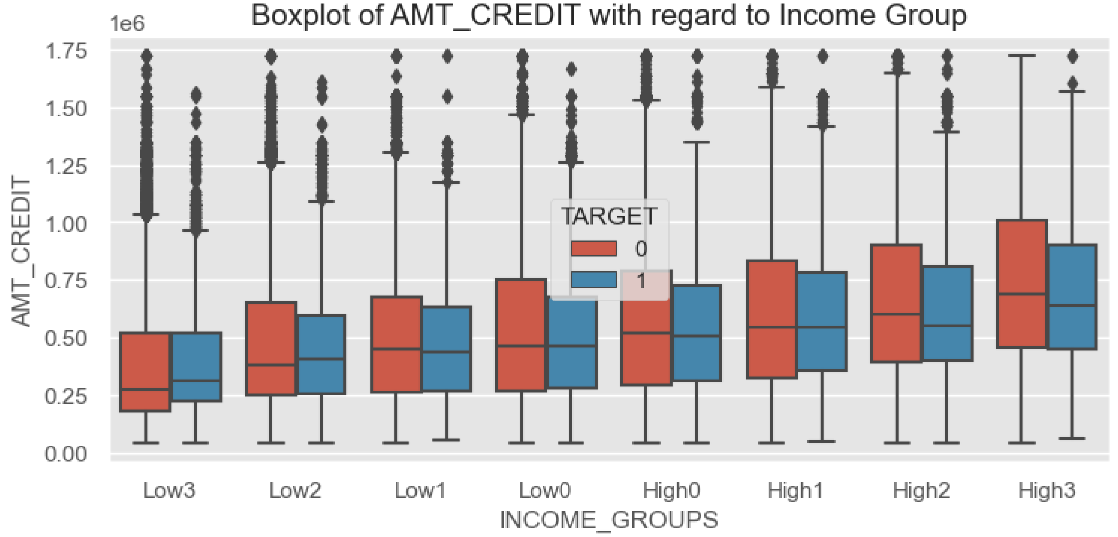


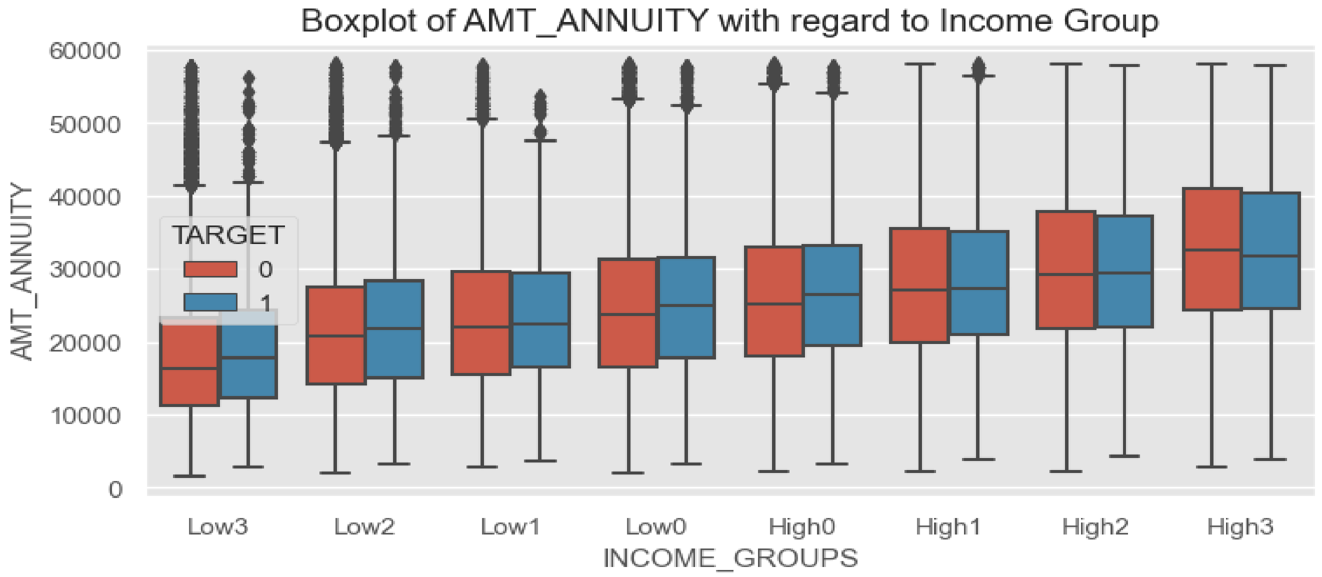


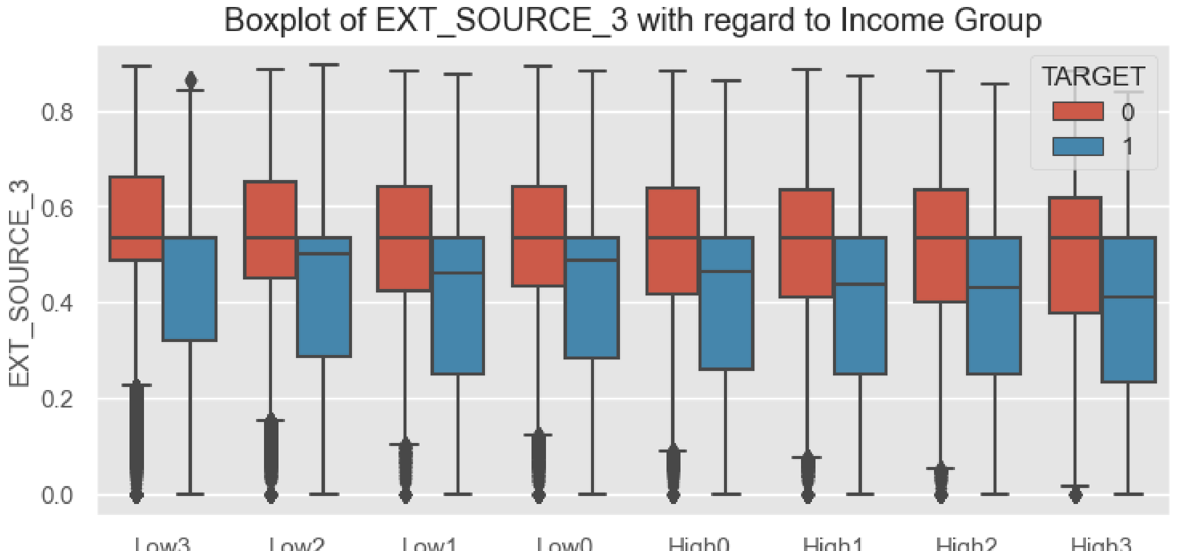




created box plot to analyze between numerical variables and income group in order to understand the difference between the defaulters and non-defaulters







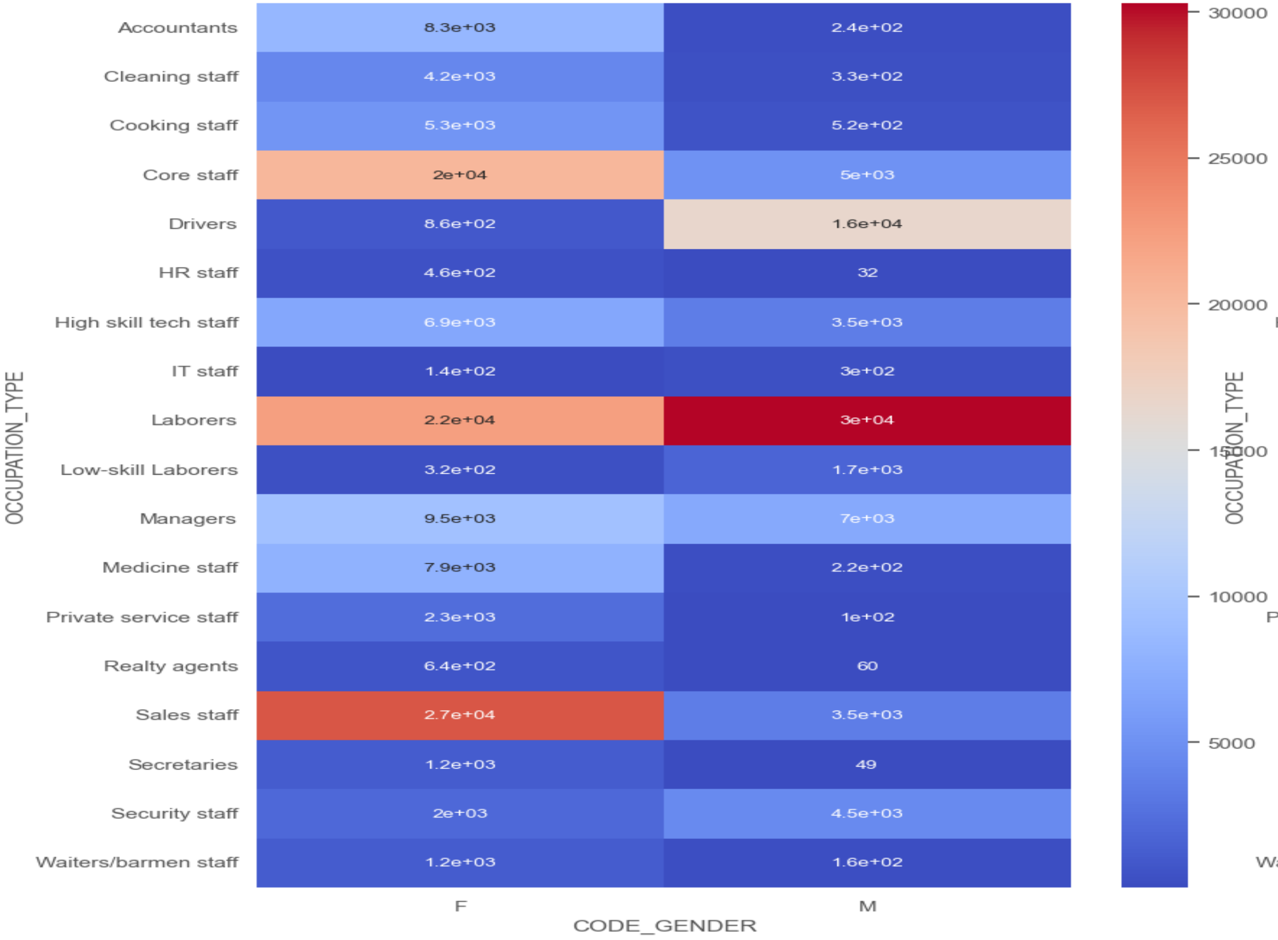
CONCLUSIONS AND INFERENCES DRAWN FROM THE ANALYSIS

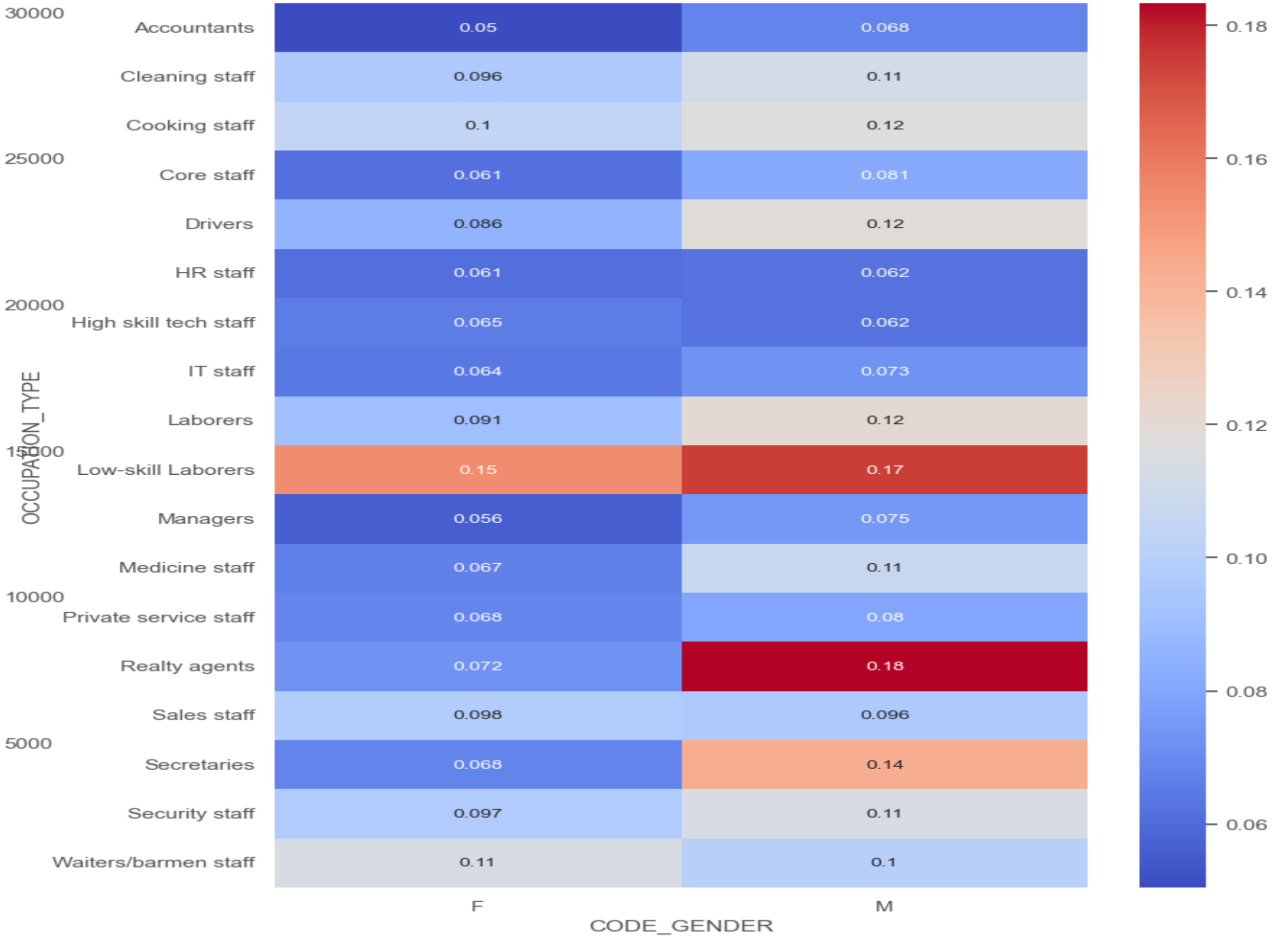
1) Female gender generally have taken comparatively more amount of credit across all the income groups

2) Female gender generally have taken more amount on goods purchased on compared to the male gender across all the income groups

3) Irrespective of the incomes and annuity amounts,defaulters exists across all the categories

Heatmap is plotted to understand why Men are higher Defaulters than Women and to check if whether the occupation type has its role



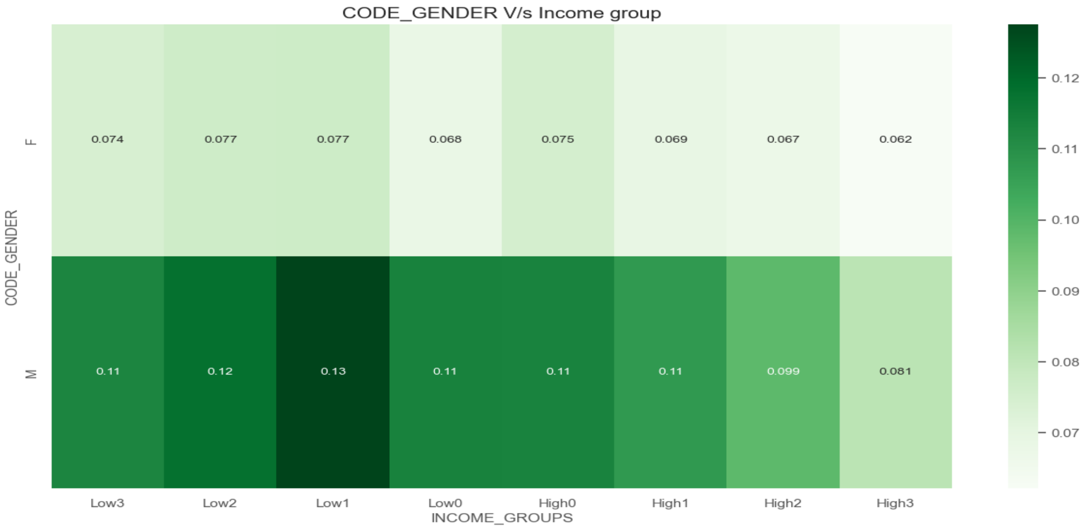


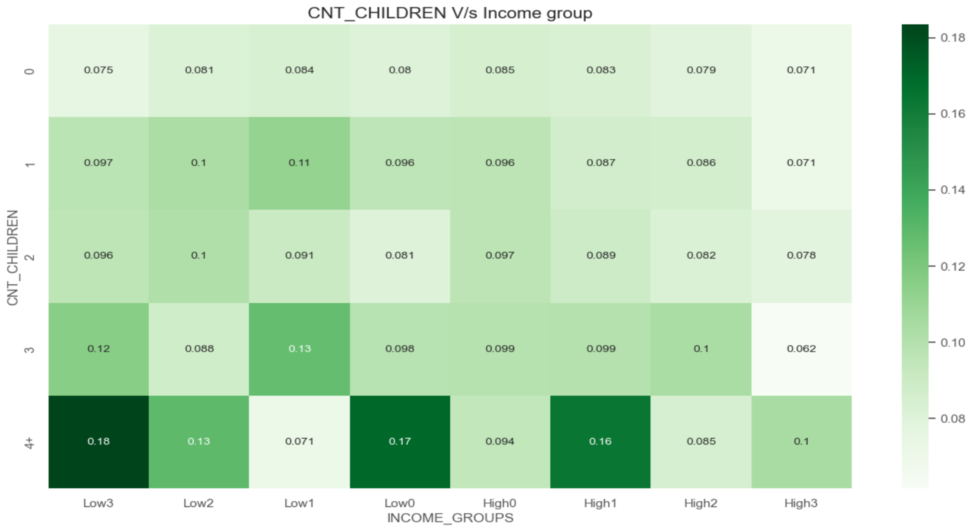
CONCLUSIONS AND INFERENCES DRAWN FROM THE ANALYSIS

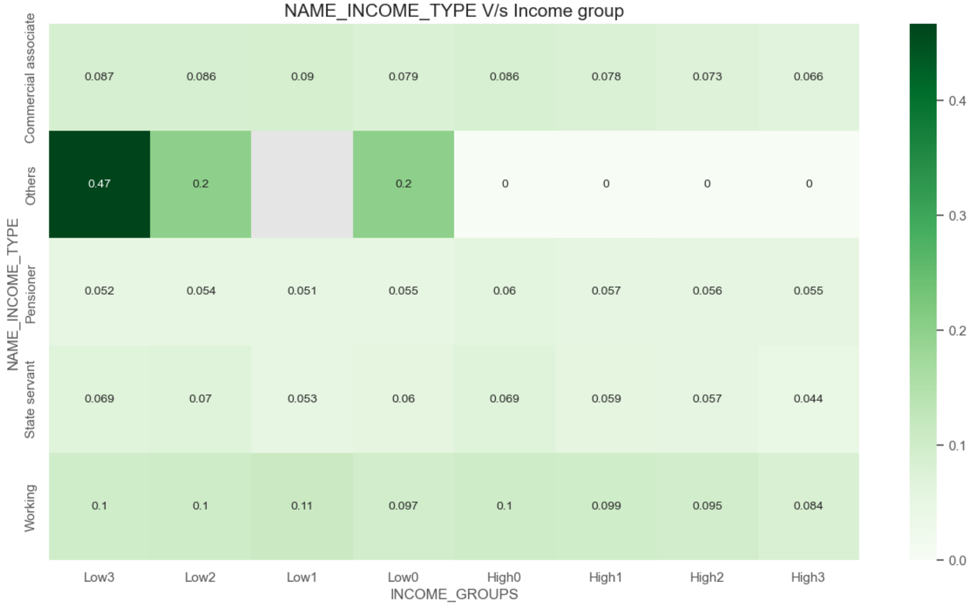
1) From the tables we can note that the men are more in number for the occupation types such as Labourers ,Drivers and these occupations accounts for the highest number of defaulters

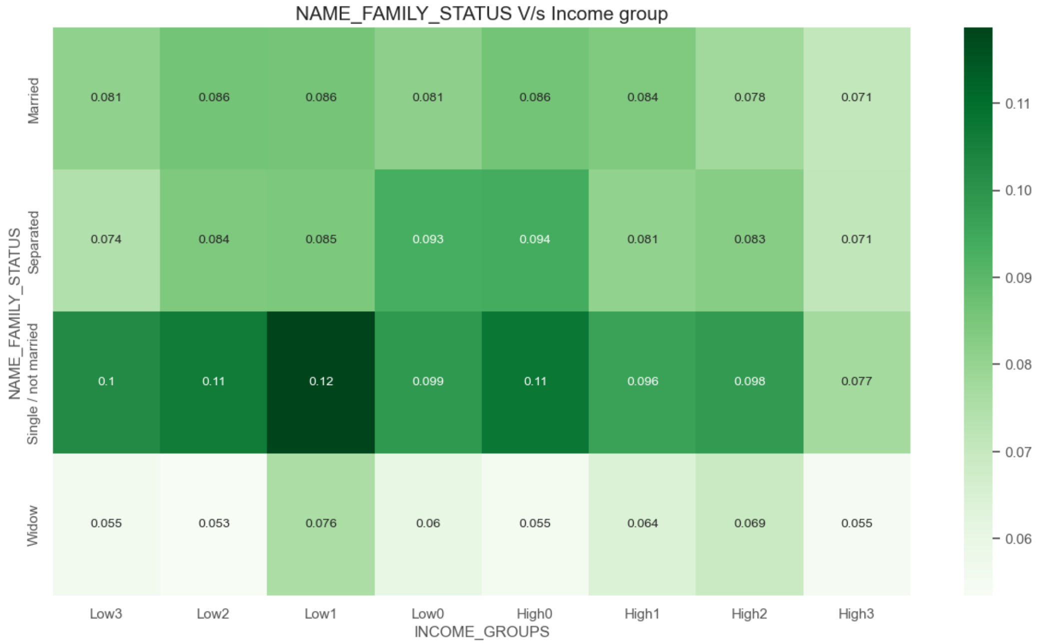
2) There are more number of female accountants, core staffs and sales staffs when compared to the males and they have relatively lower defaulting rates

Few Examples of Heatmap plotted to find the relation between Categorical columns vs 'Income Group' vs 'TARGET'









CONCLUSIONS AND INFERENCES DRAWN FROM THE ANALYSIS

1) From CODE_GENDER heatplot we can observe that the males have higher percentage of defaulters even when the incomes are
among high income groups,whereas females tends to pay off their dues promptly irrespective of the income groups
2) From CNT_CHILDREN heatplot we can observe that as the count of children increases the defaulting percentage also increases
as people 4+ children have high defaulting percentage even at high1 income
3) From NAME_INCOME_TYPE others having less income tends to be in higher defaulters range.
4) From NAME_EDUCATION_TYPE, as the education level increases the defaulters percentage drops significantly and people having lower secondary education and are in high income group also tend to be higher defaulters percentage
5) From NAME_FAMILY_STATUS singles or not married people are in higher defaulters percentage whereas the widows irrespective of the incomes are paying the dues promptly
6) From NAME_HOUSING_TYPE,people residing in the rented apartment and having low income and high income are in higher defaulters

- 7) From REGION_POPULATION_RELATIVE people residing in the regions having very high population are among low defaulters whereas the people residing in the relatively lower populated regions tends to be in relatively higher defaulters.
- 8) From AGE people who are young ie 20-30 irrespective of their incomes constitutes high defaulters whereas old people ie 60+ irrespective of the incomes pay their dues properly as the age increases the people are paying the dues more promptly
- 9) From 'YEARS_EMPLOYED freshers having less experience are finding it difficult to pay the dues irrespective of the income levels whereas the experiences are being prompt in paying the dues, as the experience years increases the defaulting percentage decreases
- 10) From OCCUPATION_TYPE Low skill labourers and drivers find it difficult to repay whereas the accountants,Hr staffs are being prompt in repaying
- 11) From CNT_FAM_MEMBERS people having more no. of children finds it difficult in repaying the dues even at a slightly higher income groups and peope with lesser no. of childrens are fairly repaying the dues promptly
- 12) From OWNER_CAR_OR_REALTY people having no car or realty tends to make more payent defaults
- 13) From AMT_CREDIT_RANGE people having credit range of 400000-600000 have repayment difficulties.

Previous_application Dataset import and cleaning

In [295]:
```
previous_application.head()
```

Out[295]:

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

In [298]:
```
# fetching the information on the dataset
previous_application.info()
```

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

```
# Dropping the columns having more than 40 percentage of null values
for i in previous_application.columns:
    if (previous_application[i].isnull().sum()*100/previous_application.shape[0])> 40:
        previous_application.drop(i,inplace=True,axis=1)
```

```
# Finding the percentage of null values
previous_application.isnull().sum()*100/previous_application.shape[0]
```

```
SK_ID_PREV                     0.000000
SK_ID_CURR                     0.000000
NAME_CONTRACT_TYPE             0.000000
AMT_ANNUITY                   22.286665
AMT_APPLICATION                0.000000
AMT_CREDIT                     0.000060
AMT_GOODS_PRICE               23.081773
WEEKDAY_APPR_PROCESS_START     0.000000
HOUR_APPR_PROCESS_START        0.000000
FLAG_LAST_APPL_PER_CONTRACT    0.000000
NFLAG_LAST_APPL_IN_DAY         0.000000
NAME_CASH_LOAN_PURPOSE         0.000000
NAME_CONTRACT_STATUS           0.000000
DAYS_DECISION                  0.000000
NAME_PAYMENT_TYPE              0.000000
CODE_REJECT_REASON             0.000000
NAME_CLIENT_TYPE               0.000000
NAME_GOODS_CATEGORY            0.000000
NAME_PORTFOLIO                 0.000000
NAME_PRODUCT_TYPE              0.000000
CHANNEL_TYPE                   0.000000
SELLERPLACE_AREA               0.000000
NAME_SELLER_INDUSTRY           0.000000
CNT_PAYMENT                   22.286366
NAME_YIELD_GROUP               0.000000
PRODUCT_COMBINATION            0.020716
dtype: float64
```

```
# Listing out the categorical columns
categorical_col_1 = ['NAME_CONTRACT_TYPE','NAME_PAYMENT_TYPE','NAME_CASH_LOAN_PURPOSE','NAME_CONTRACT_STATUS',
        'CODE_REJECT_REASON', 'NAME_CLIENT_TYPE','NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO', 'NAME_PRODUCT_TYPE',
           'CHANNEL_TYPE','NAME_SELLER_INDUSTRY','NAME_YIELD_GROUP', 'PRODUCT_COMBINATION']
for i in categorical_col_1:
    print(previous_application[i].value_counts())
    print("-----------------------------------END-----------------------------------
```

```
Cash loans         747553
Consumer loans     729151
Revolving loans    193164
XNA                   346
Name: NAME_CONTRACT_TYPE, dtype: int64
```

In [301]:
```
previous_application.isnull().sum()
```

Out[301]:
```
SK_ID_PREV                           0
SK_ID_CURR                           0
NAME_CONTRACT_TYPE                   0
AMT_ANNUITY                     372235
AMT_APPLICATION                      0
AMT_CREDIT                           1
AMT_DOWN_PAYMENT                895844
AMT_GOODS_PRICE                 385515
WEEKDAY_APPR_PROCESS_START           0
HOUR_APPR_PROCESS_START              0
FLAG_LAST_APPL_PER_CONTRACT          0
NFLAG_LAST_APPL_IN_DAY               0
RATE_DOWN_PAYMENT               895844
RATE_INTEREST_PRIMARY          1664263
RATE_INTEREST_PRIVILEGED       1664263
NAME_CASH_LOAN_PURPOSE               0
NAME_CONTRACT_STATUS                 0
DAYS_DECISION                        0
NAME_PAYMENT_TYPE                    0
CODE_REJECT_REASON                   0
NAME_TYPE_SUITE                 820405
NAME_CLIENT_TYPE                     0
NAME_GOODS_CATEGORY                  0
NAME_PORTFOLIO                       0
NAME_PRODUCT_TYPE                    0
CHANNEL_TYPE                         0
SELLERPLACE_AREA                     0
NAME_SELLER_INDUSTRY                 0
CNT_PAYMENT                     372230
NAME_YIELD_GROUP                     0
PRODUCT_COMBINATION                346
DAYS_FIRST_DRAWING              673065
DAYS_FIRST_DUE                  673065
DAYS_LAST_DUE_1ST_VERSION       673065
DAYS_LAST_DUE                   673065
DAYS_TERMINATION                673065
NFLAG_INSURED_ON_APPROVAL       673065
dtype: int64
```

In [302]:
```
# Finding the percentage of null values
previous_application.isnull().sum()*100/previous_application
```

We can huge numbers of XAP and XNA in the dataset which has no value this can be replaced to null values

In [317]:
```
categorical_col_1 = ['NAME_CONTRACT_TYPE','NAME_PAYMENT_TYPE','NAME_CASH_LOAN_PURPOSE','NAME_CONTRACT_STATUS',
        'CODE_REJECT_REASON', 'NAME_CLIENT_TYPE','NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO', 'NAME_PRODUCT_TYPE',
           'CHANNEL_TYPE','NAME_SELLER_INDUSTRY','NAME_YIELD_GROUP', 'PRODUCT_COMBINATION']
for i in categorical_col_1:
    previous_application[i]=previous_application[i].apply(lambda x:np.nan if x in ["XNA","XAP"] else x)
```

In [318]:
```
previous_application.head()
```

Out[318]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_GOODS_PRICE	WEEKDAY_APPR_PROCESS_ST
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	17145.0	SATUF
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	607500.0	THURS
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	112500.0	TUES
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	450000.0	MON
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	337500.0	THURS

STEPS TAKEN TO CLEAN THE DATA

The columns which were having the missing values more than 40% has been dropped
the values XAP and XNA has been replaced with NaN

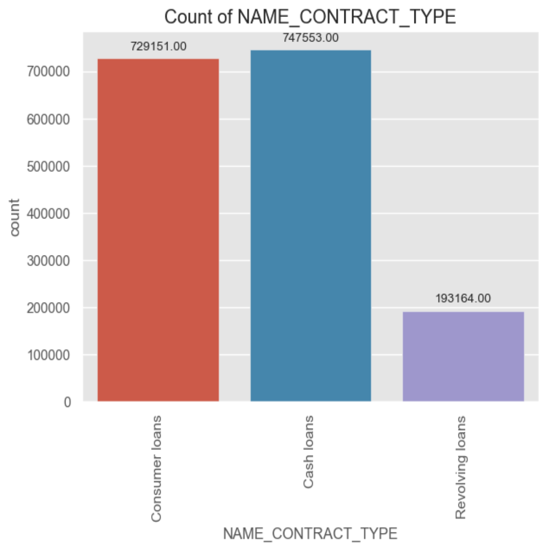
Creating the function to obtain the count plot and bar chart in order to analyze

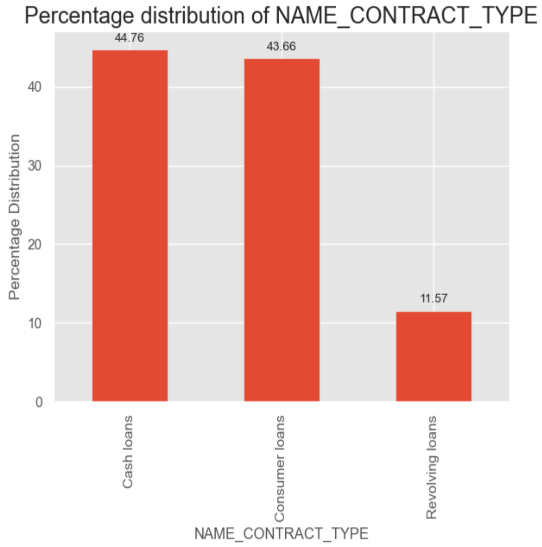
CATEGORICAL UNIVARIATE ANALYSIS

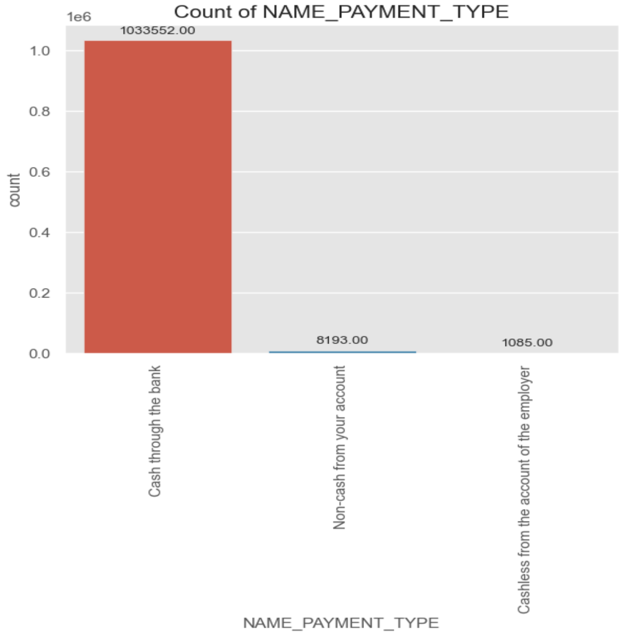
```
# Creating the function to obtain the count plot and bar chart in order to analyse
def categorical_univariate_Analysis_ds2(x):
    print('Categorical Univariate Analysis of {}'.format(x))

    plt.figure(figsize=[15,5])
    plt.subplot(1,2,1)
    plots = sns.countplot(previous_application[x])
    plt.xticks(rotation = 90)
    plt.title('Count of {}'.format(x), fontdict={'fontsize':15})
    for bar in plots.patches:
        plots.annotate(format(bar.get_height(), '.2f'),
                       (bar.get_x() + bar.get_width() / 2,
                        bar.get_height()), ha='center', va='center',
                       size=10, xytext=(0, 8),
                       textcoords='offset points')

    plt.subplot(1,2,2)
    plt.title('Percentage distribution of {}'.format(x), fontdict={'fontsize':18})
    plt.xlabel(x)
    plt.ylabel('Percentage Distribution')
    plots = (previous_application[x].value_counts()*100/len(previous_application[previous_application[x]!=np.nan])).plot.bar()
    for bar in plots.patches:
        plots.annotate(format(bar.get_height(), '.2f'),
                       (bar.get_x() + bar.get_width() / 2,
                        bar.get_height()), ha='center', va='center',
                       size=10, xytext=(0, 8),
                       textcoords='offset points')

    plt.show()
    print('-----------------------------------------------------------------------------------------------------------------
```

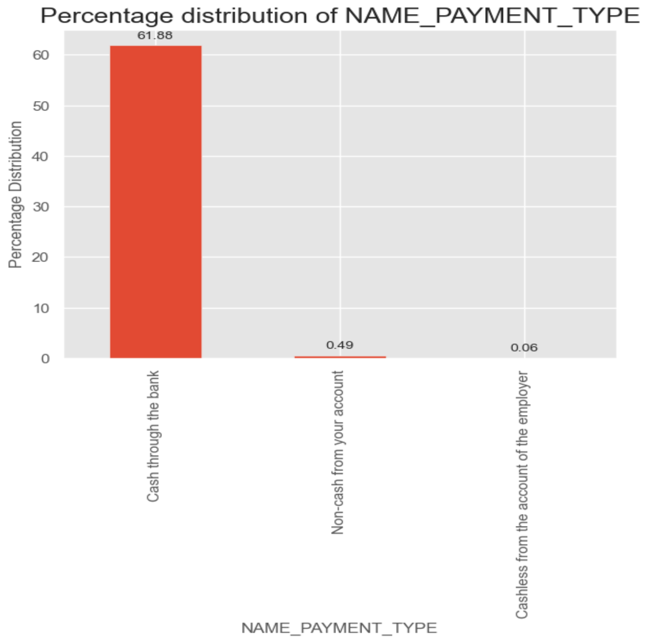
```
categorical_col_1 = ['NAME_CONTRACT_TYPE','NAME_PAYMENT_TYPE','NAME_CASH_LOAN_PURPOSE','NAME_CONTRACT_STATUS',
       'CODE_REJECT_REASON', 'NAME_CLIENT_TYPE','NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO', 'NAME_PRODUCT_TYPE',
           'CHANNEL_TYPE','NAME_SELLER_INDUSTRY','NAME_YIELD_GROUP', 'PRODUCT_COMBINATION']
```

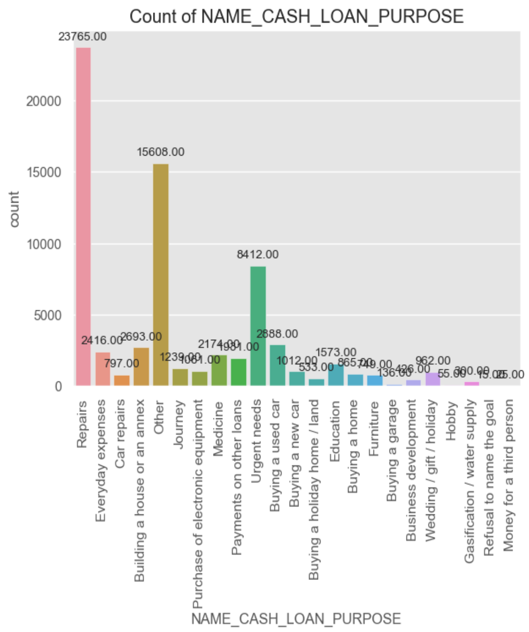
Univariate Analysis for Previous_Application Dataset

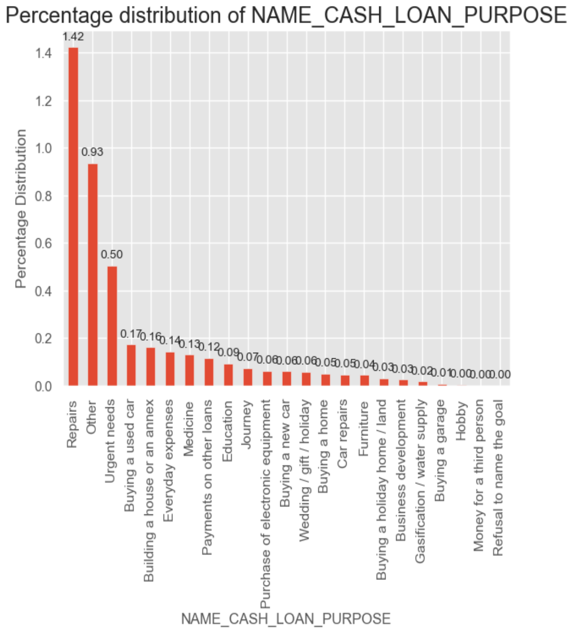


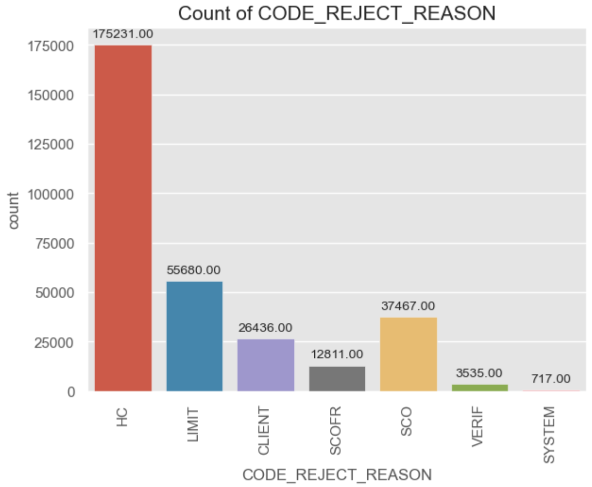


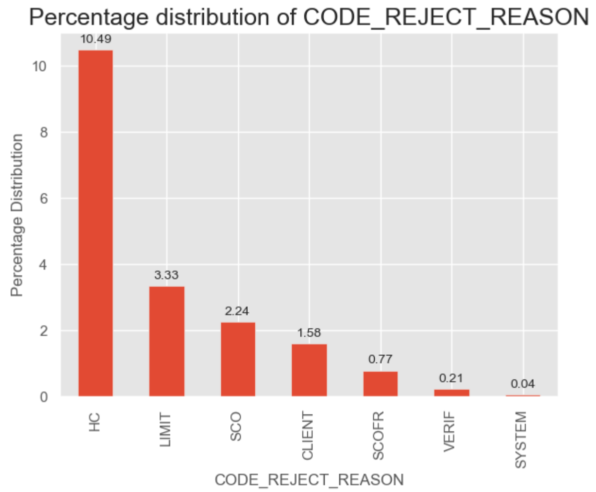












- CONCLUSIONS AND INFERENCES DRAWN FROM THE ANALYSIS
- 1) From NAME_CONTRACT_TYPE it can be noted that significant number of people obtain consumer loans and cash loans
- 2) From NAME_PAYMENT_TYPE it can be noted that huge number of people borrow in the form of cash through the bank
- 3) From NAME_CASH_LOAN_PURPOSE it can be noted that repairs,others and urgent needs are the top 3 purposes for obtaining the loan
- 4) From NAME_CONTRACT_STATUS it can be noted that majority of the loans are approved.
- 5) From CODE_REJECT_REASON it can be noted that HC accounts for rejection reason
- 6) From NAME_CLIENT_TYPE it can be loan repeaters are more in number
- 7) From NAME_GOODS_CATEGORY it can be noted that the loan is being taken for the purpose of procuring goods such as mobiles,consumer electronics,computers,and audio,video devices
- 8) From NAME_PORTFOLIO it can be noted that pos (point of sale) accounts for more loan disbursed
- 9) From NAME_PRODUCT_TYPE it can be noted that credit and cash offices,country-wide Through which channel we acquired the client on the previous application is highest
- 10) From NAME_SELLER_INDUSTRY we can see that seller industry is majorly consumer-electronics and connectivity

Merging Data and Bivariate Analysis for Merged Dataset

MERGING THE DATASETS BASED ON THE COMMON COLUMN

In [347]:
```
Final_Merge=application_data_1.merge(previous_application_1, left_on='SK_ID_CURR',right_on='SK_ID_CURR', how='inner')
```

In [349]:
```
# Reviewing the merged dataset
Final_Merge.head()
```

Out[349]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CF
0	100002	1	Cash loans	M	N	Y	0	202500.0	4
1	100003	0	Cash loans	F	N	N	0	270000.0	12
2	100003	0	Cash loans	F	N	N	0	270000.0	12
3	100003	0	Cash loans	F	N	N	0	270000.0	12
4	100004	0	Revolving loans	M	Y	Y	0	67500.0	1

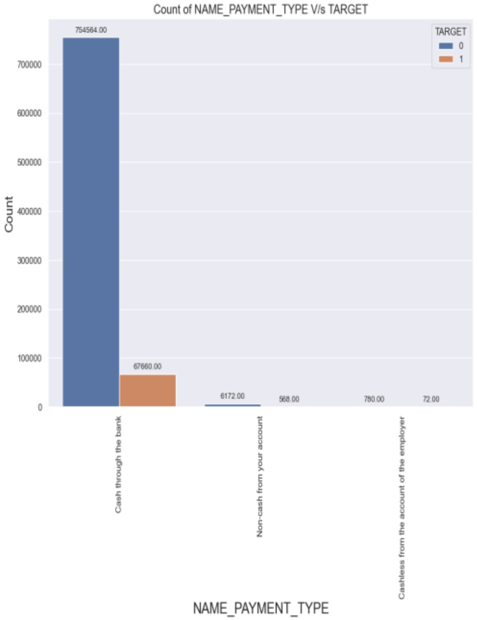
SEGMENTING THE DATA BASED ON DEFAULTERS AND NON-DEFAULTERS

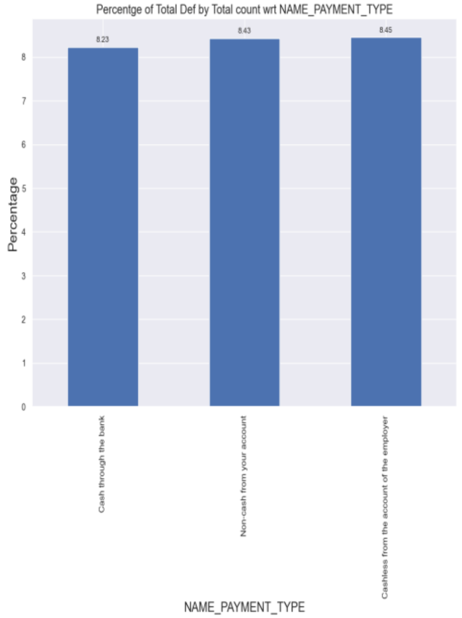
In [358]:
```
Final_Merge_0=Final_Merge[Final_Merge["TARGET"]==0]
```

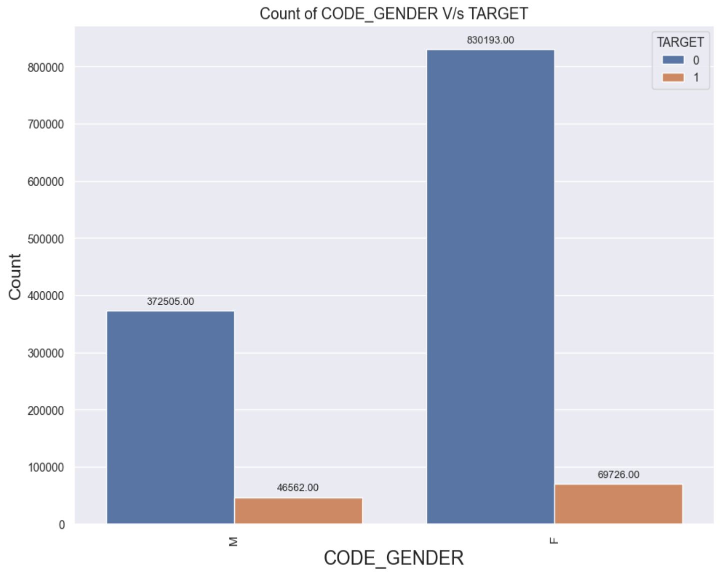
In [359]:
```
Final_Merge_1=Final_Merge[Final_Merge["TARGET"]==1]
```

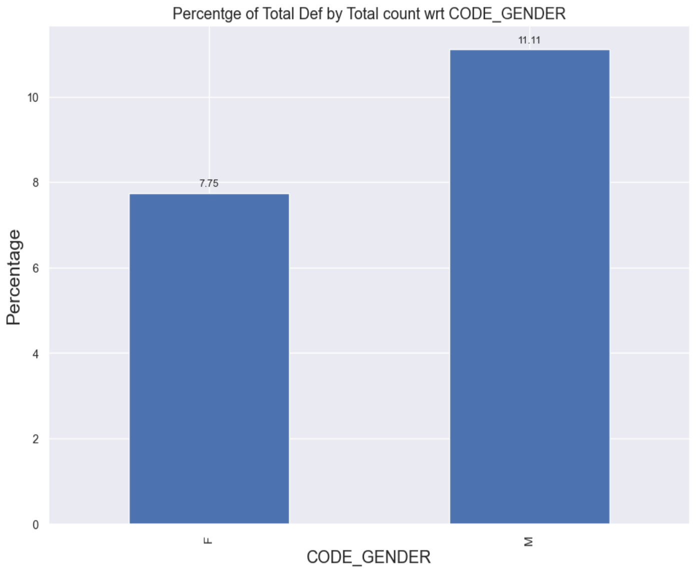
BIVARIATE ANALYSIS

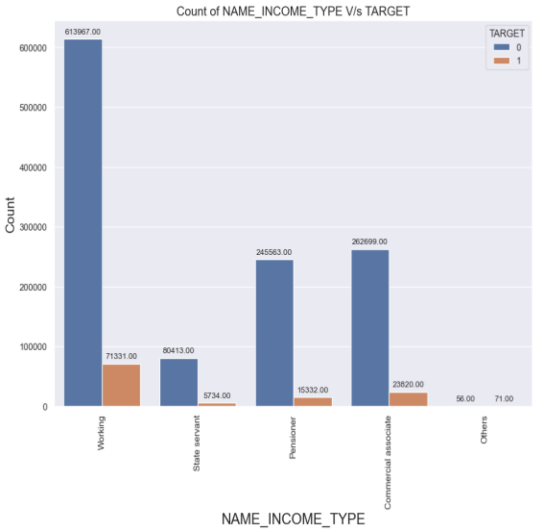
In [361]:
```
# Create a function to perform bivariate analysis

def final_segmented_categorical_uni_analysis(x):
    print('Segmented Categorical Univariate Analysis of {}'.format(x))
    print("")

    plt.figure(figsize=[25,8])

    plt.subplot(1,2,1)
    plots = sns.countplot(x,hue='TARGET',data=Final_Merge)
    plt.xticks(rotation = 90)
    plt.title('Count of {} V/s TARGET'.format(x), fontdict={'fontsize':15})
    plt.xlabel(x,fontdict={'fontsize':18})
    plt.ylabel('Count',fontdict={'fontsize':16})
    for bar in plots.patches:
        plots.annotate(format(bar.get_height(), '.2f'),
                       (bar.get_x() + bar.get_width() / 2,
                        bar.get_height()), ha='center', va='center',
                       size=10, xytext=(0, 8),
                       textcoords='offset points')

    plt.subplot(1,2,2)
    plots = (Final_Merge[x][Final_Merge['TARGET']==1].value_counts()*100/Final_Merge[x].value_counts()).plot.bar()
    plt.xticks(rotation = 90)
    plt.title('Percentge of Total Def by Total count wrt {}'.format(x), fontdict={'fontsize':15})
    plt.xlabel(x,fontdict={'fontsize':16})
    plt.ylabel('Percentage',fontdict={'fontsize':17})
    for bar in plots.patches:
        plots.annotate(format(bar.get_height(), '.2f'),
                       (bar.get_x() + bar.get_width() / 2,
                        bar.get_height()), ha='center', va='center',
                       size=10, xytext=(0, 8),
                       textcoords='offset points')
```

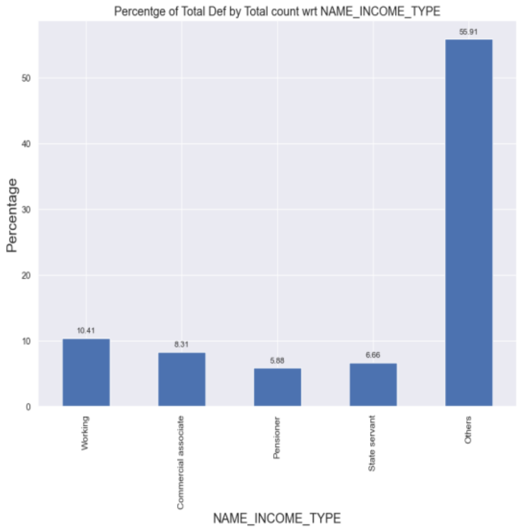


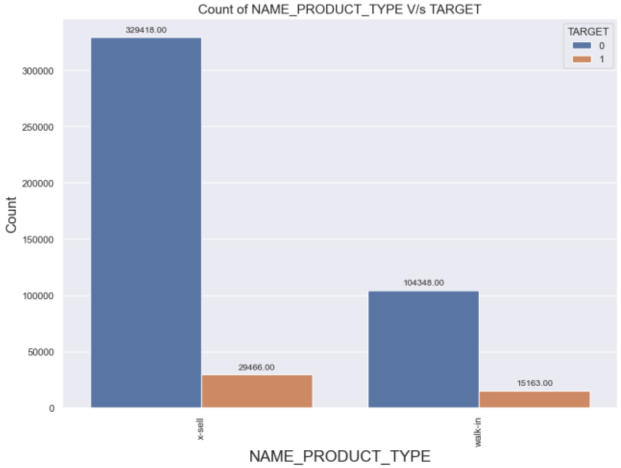


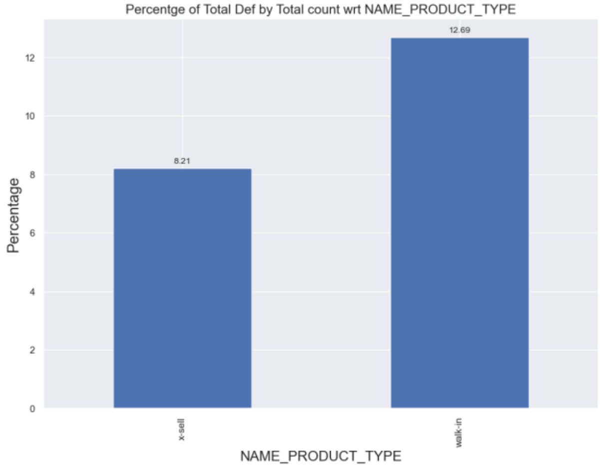












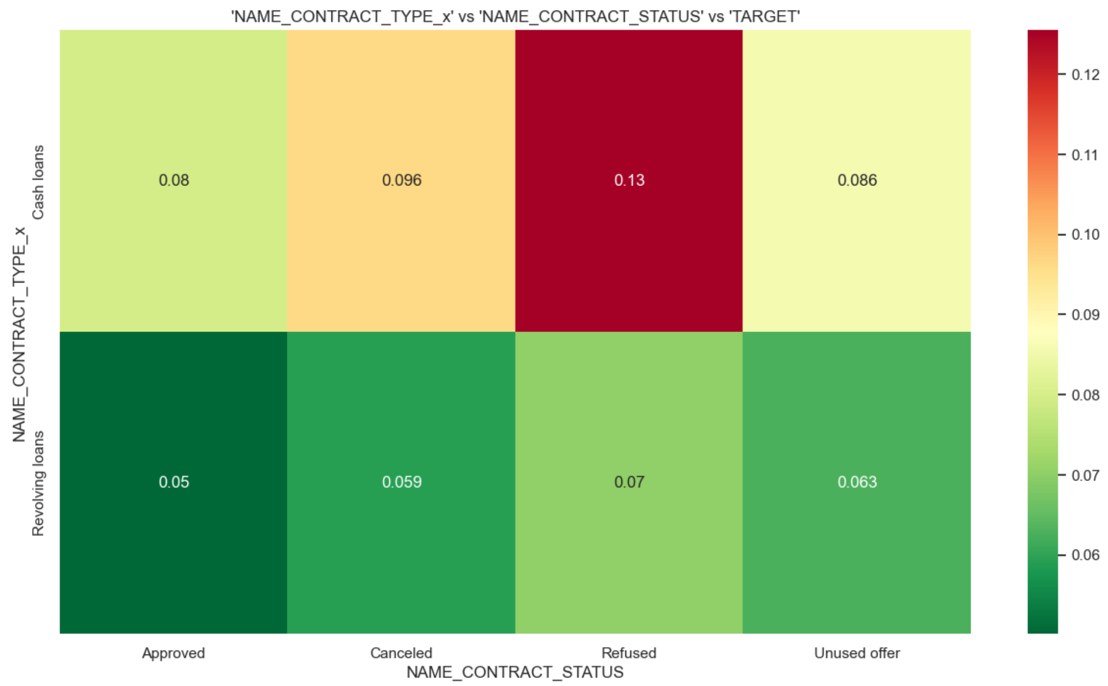
Bivariate Analysis for Merged Dataset

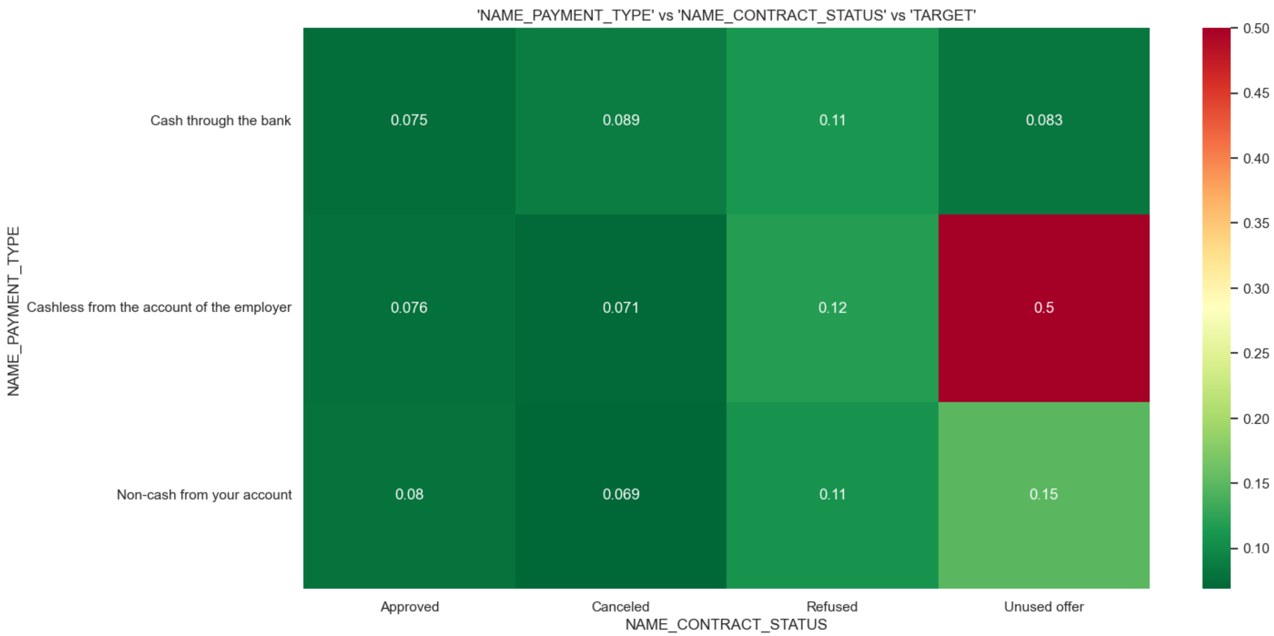
Create a heatmap to analyze the relationship between the name_contract_status ,target and the categorical columns to understand its relationship

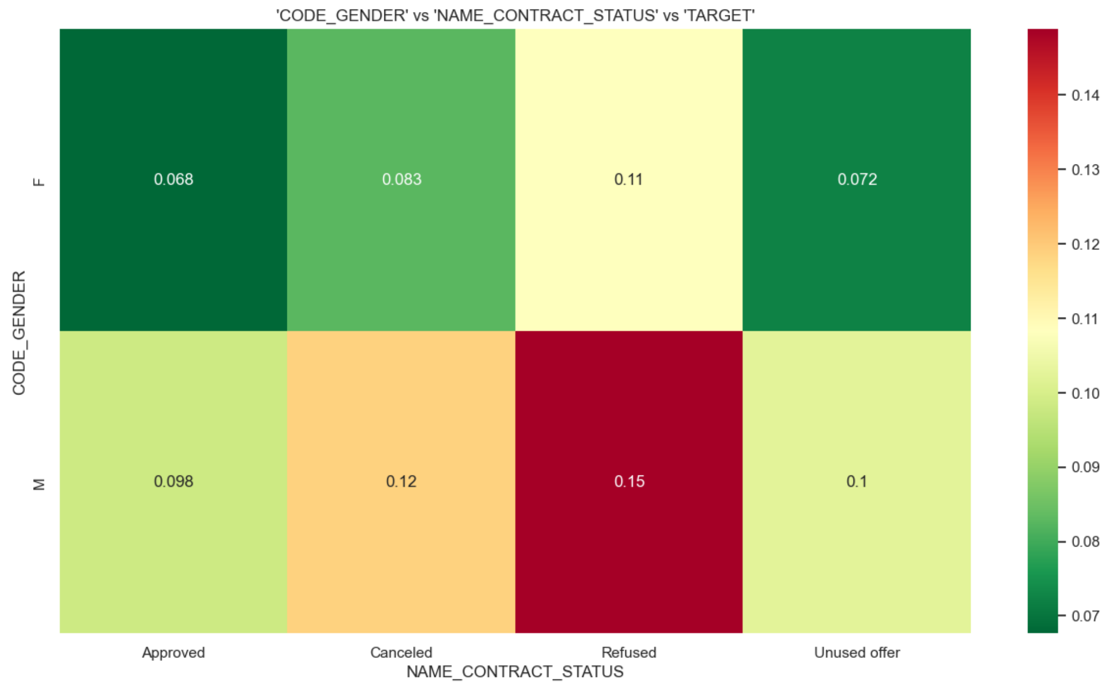
CHANNEL_TYPE CHANNEL_TYPE

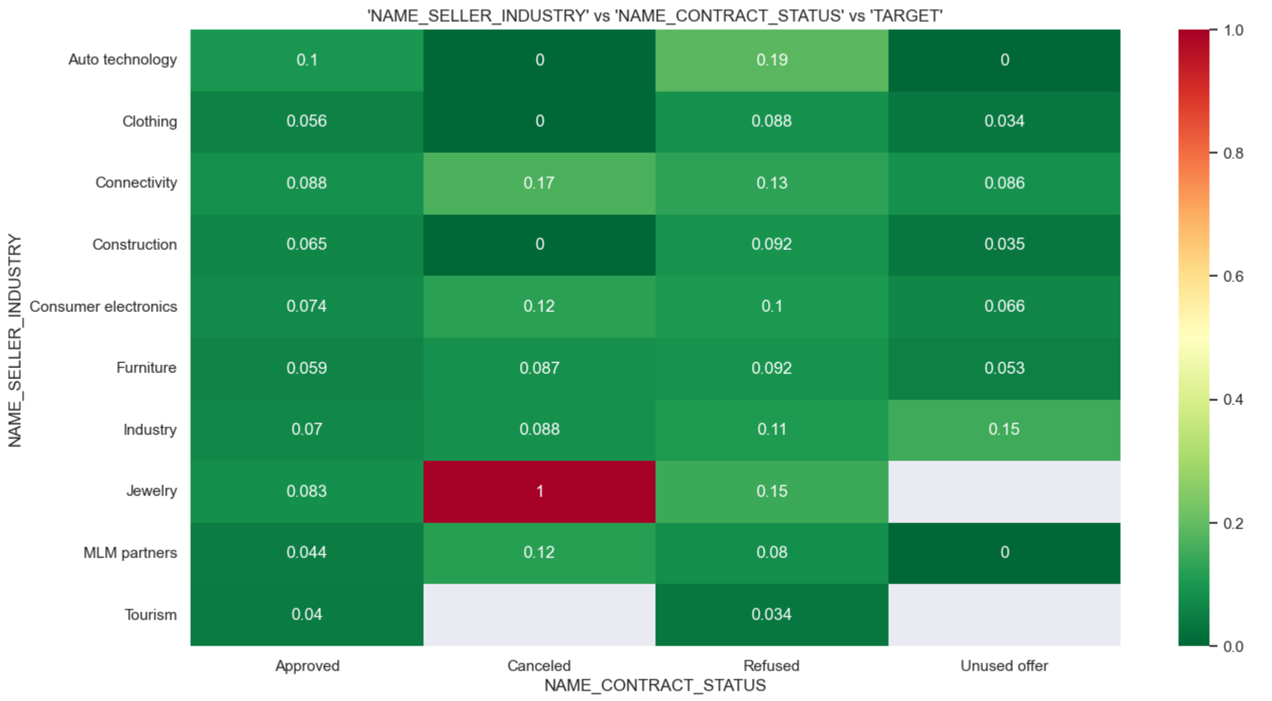
In [381]:

```
# Create a heatmap to analyse the relationship between the name_contract_status ,target and the categorical columns to understar

categorical_col = ['NAME_CONTRACT_TYPE_x','NAME_PAYMENT_TYPE','NAME_CLIENT_TYPE','CODE_GENDER', 'CNT_CHILDREN', 'NAME_INCOME_TYP
       'CODE_REJECT_REASON','NAME_PRODUCT_TYPE','CHANNEL_TYPE','NAME_SELLER_INDUSTRY','PRODUCT_COMBINATION','NAME_FAMILY_STATUS'
       'AGE', 'YEARS_EMPLOYED','NAME_CASH_LOAN_PURPOSE']
for i in categorical_col:
    plt.figure(figsize=[15,8])
    sns.heatmap(pd.pivot_table(data=Final_Merge,index=i,columns="NAME_CONTRACT_STATUS",values="TARGET",aggfunc="mean"),annot=Tru
    plt.title("'{}' vs 'NAME_CONTRACT_STATUS' vs 'TARGET'".format(i))

    plt.show()
    print("-------------------------------------------------------------------------------------------------------------------
```


'NAME_CONTRACT_TYPE_x' vs 'NAME_CONTRACT_STATUS' vs 'TARGET'
NAME_CONTRACT_TYPE_x
Cash loans
Revolving loans
0.08
0.096
0.13
0.086
0.05
0.059
0.07
0.063
Approved
Canceled
Refused
Unused offer
NAME_CONTRACT_STATUS
0.12
0.11
0.10
0.09
0.08
0.07
0.06


'NAME_PAYMENT_TYPE' vs 'NAME_CONTRACT_STATUS' vs 'TARGET'
NAME_PAYMENT_TYPE
Cash through the bank
Cashless from the account of the employer
Non-cash from your account
0.075
0.089
0.11
0.083
0.076
0.071
0.12
0.5
0.08
0.069
0.11
0.15
Approved
Canceled
Refused
Unused offer
NAME_CONTRACT_STATUS
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10


'CODE_GENDER' vs 'NAME_CONTRACT_STATUS' vs 'TARGET'
CODE_GENDER
F
M
0.068
0.083
0.11
0.072
0.098
0.12
0.15
0.1
Approved
Canceled
Refused
Unused offer
NAME_CONTRACT_STATUS
0.14
0.13
0.12
0.11
0.10
0.09
0.08
0.07


'NAME_SELLER_INDUSTRY' vs 'NAME_CONTRACT_STATUS' vs 'TARGET'
NAME_SELLER_INDUSTRY
Auto technology
Clothing
Connectivity
Construction
Consumer electronics
Furniture
Industry
Jewelry
MLM partners
Tourism
0.1
0
0.19
0
0.056
0
0.088
0.034
0.088
0.17
0.13
0.086
0.065
0
0.092
0.035
0.074
0.12
0.1
0.066
0.059
0.087
0.092
0.053
0.07
0.088
0.11
0.15
0.083
1
0.15
0.044
0.12
0.08
0
0.04
0.034
Approved
Canceled
Refused
Unused offer
NAME_CONTRACT_STATUS
1.0
0.8
0.6
0.4
0.2
0.0

- Conclusions and Inferences

- Ideally the Approved boxes should be in dark greener in color which implies that the approval are being done for someone who's known not to be doing the default payments
- refused, cancelled columns should ideally be in red color which implies that the defaulters are not given the loan.
- From the heatmaps we can see that there are few columns where eventhough the chance of defaulting is less the loan is being refused or cancelled which again is a missed business oppurtunity to the banks and also few instances where the defaulting chances are more but are given the loan by approving the banks can look into this as either ways its a loss to the banks.