PRESENTATION ON CASE STUDY OF EDA

NAME : AMBIKA KUMARI
BATCH :FEB 041

Business objective : This case study aims to identify patterns which indicate if a client has difficulty paying their instalment 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 OF EDA :

- Import /Read data set
- Data Inspection
- Data Cleaning
- Data Analysis

Data inspection : viewing data for verification and getting information about data

```
# importing application_data.csv

appl = pd.read_csv("application_data.csv")
```

```
appl.head()
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_
0	100002	1	Cash loans	M	N	

```
appl.info(all)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 122 columns):
 #     Column                       Dtype
---    ------                       -----
 0     SK_ID_CURR                   int64
 1     TARGET                       int64
 2     NAME_CONTRACT_TYPE           object
 3     CODE_GENDER                  object
 4     FLAG_OWN_CAR                 object
 5     FLAG_OWN_REALTY              object
 6     CNT_CHILDREN                 int64
 7     AMT_INCOME_TOTAL             float64
```

```
#check  sum of null value from each columns

appl.isnull().sum()
```

```
SK_ID_CURR                        0
TARGET                            0
NAME_CONTRACT_TYPE                0
CODE_GENDER                       0
FLAG_OWN_CAR                      0
                                ...
AMT_REQ_CREDIT_BUREAU_DAY     41519
AMT_REQ_CREDIT_BUREAU_WEEK    41519
AMT_REQ_CREDIT_BUREAU_MON     41519
AMT_REQ_CREDIT_BUREAU_QRT     41519
AMT_REQ_CREDIT_BUREAU_YEAR    41519
```

Data Cleaning: Droping null values, missing values, and unwanted columns which are not used

Dealing with Null values more than 50 %

```
#  we will deal with null values more than 50%

app_50= null_value(appl)[null_value(appl)>50]
print(app_50)
print("_"*50)
print(len(app_50))
```

```
COMMONAREA_MEDI              69.872297
COMMONAREA_AVG               69.872297
COMMONAREA_MODE              69.872297
NONLIVINGAPARTMENTS_MODE     69.432963
NONLIVINGAPARTMENTS_AVG      69.432963
NONLIVINGAPARTMENTS_MEDI     69.432963
FONDKAPREMONT_MODE           68.386172
```

Null values which is greater than 50% . Those are droped.

Dealing with Null values more than 15 %

```
# now we will deal with null values more than 15%

app_15 = null_value(appl)[null_value(appl)>15]
```

```
print(app_15)
print()
print(len(app_15))
```

```
FLOORSMAX_AVG                   49.760822
FLOORSMAX_MODE                  49.760822
FLOORSMAX_MEDI                  49.760822
YEARS_BEGINEXPLUATATION_AVG     48.781019
YEARS_BEGINEXPLUATATION_MODE    48.781019
YEARS_BEGINEXPLUATATION_MEDI    48.781019
TOTALAREA_MODE                  48.268517
EMERGENCYSTATE_MODE             47.398304
OCCUPATION_TYPE                 31.345545
EXT_SOURCE_3                    19.825307
dtype: float64

10
```

Null values which is greater than 15% those are droped.

```
# After dropping null_col_15, we have left with 73 columns
appl.shape
```

```
(307511, 73)
```

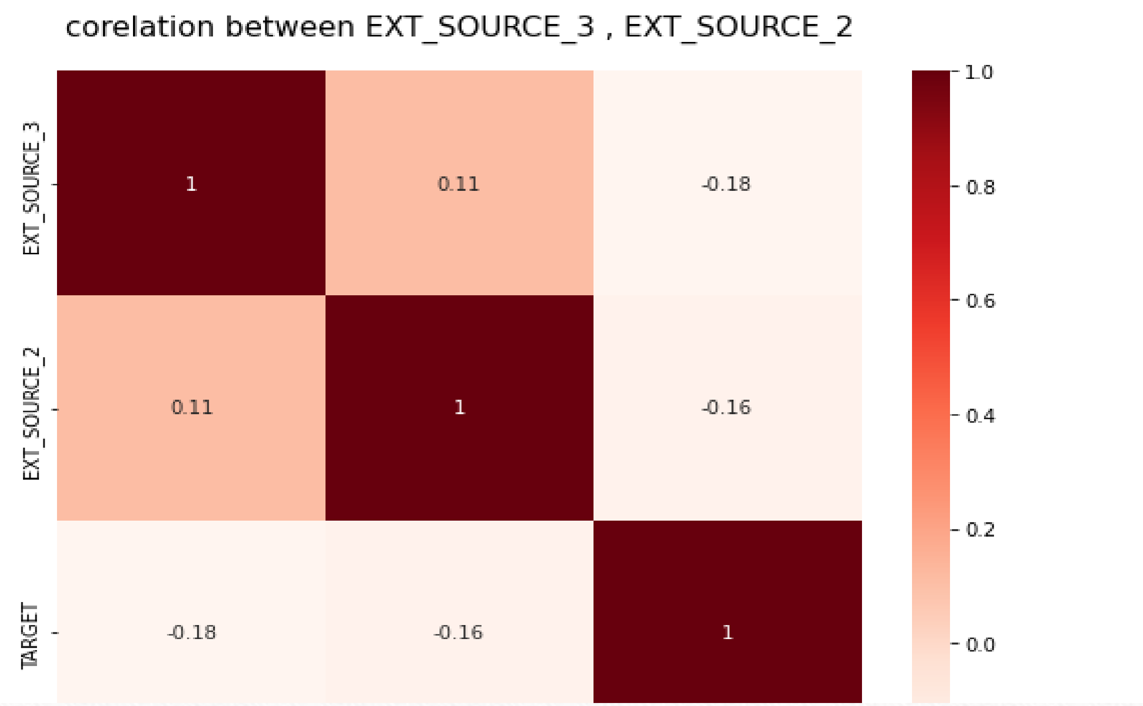
```
null_value(appl).head(10)
```

```
OCCUPATION_TYPE               31.345545
EXT_SOURCE_3                  19.825307
AMT_REQ_CREDIT_BUREAU_YEAR    13.501631
AMT_REQ_CREDIT_BUREAU_QRT     13.501631
AMT_REQ_CREDIT_BUREAU_MON     13.501631
AMT_REQ_CREDIT_BUREAU_WEEK    13.501631
AMT_REQ_CREDIT_BUREAU_DAY     13.501631
AMT_REQ_CREDIT_BUREAU_HOUR    13.501631
NAME_TYPE_SUITE                0.420148
OBS_30_CNT_SOCIAL_CIRCLE       0.332021
dtype: float64
```

Analysis and removing unnecessary columns

Starting with EXT_SOURCE_3 , EXT_SOURCE_2. we can understand the relation between these columns with TARGET column using a heatmap.

```
plt.figure(figsize=[9,7])

sns.heatmap(appl[target+["TARGET"]].corr(),cmap='Reds',annot=True)
plt.title("corelation between EXT_SOURCE_3 , EXT_SOURCE_2",fontsize=15,pad
plt.show()
```



create a dataframe containig all FLAG columns and then plot bar graphs for each column to TARGET column for which "0" will represent as Repayer and "1" will represent as Defaulter

```
# adding and viewing flag_columns
flag_columns = [colum for colum in appl.columns if "FLAG" in col
flag_columns
```

```
['FLAG_OWN_CAR',
 'FLAG_OWN_REALTY',
 'FLAG_MOBIL',
 'FLAG_EMP_PHONE',
 'FLAG_WORK_PHONE',
 'FLAG_CONT_MOBILE',
 'FLAG_PHONE',
 'FLAG_EMAIL',
 'FLAG_DOCUMENT_2',
 'FLAG_DOCUMENT_3',
 'FLAG_DOCUMENT_4',
 'FLAG_DOCUMENT_5',
 'FLAG_DOCUMENT_6',
 'FLAG_DOCUMENT_7',
 'FLAG_DOCUMENT_8',
 'FLAG_DOCUMENT_9',
 'FLAG_DOCUMENT_10',
 'FLAG_DOCUMENT_11',
 'FLAG_DOCUMENT_12',
 'FLAG_DOCUMENT_13',
 'FLAG_DOCUMENT_14',
 'FLAG_DOCUMENT_15',
 'FLAG_DOCUMENT_16',
 'FLAG_DOCUMENT_17'
```

```
for i in flag_dataframe:
    if i !="TARGET":
        flag_dataframe[i]=flag_dataframe[i].replace({1:"Y",0:"
```

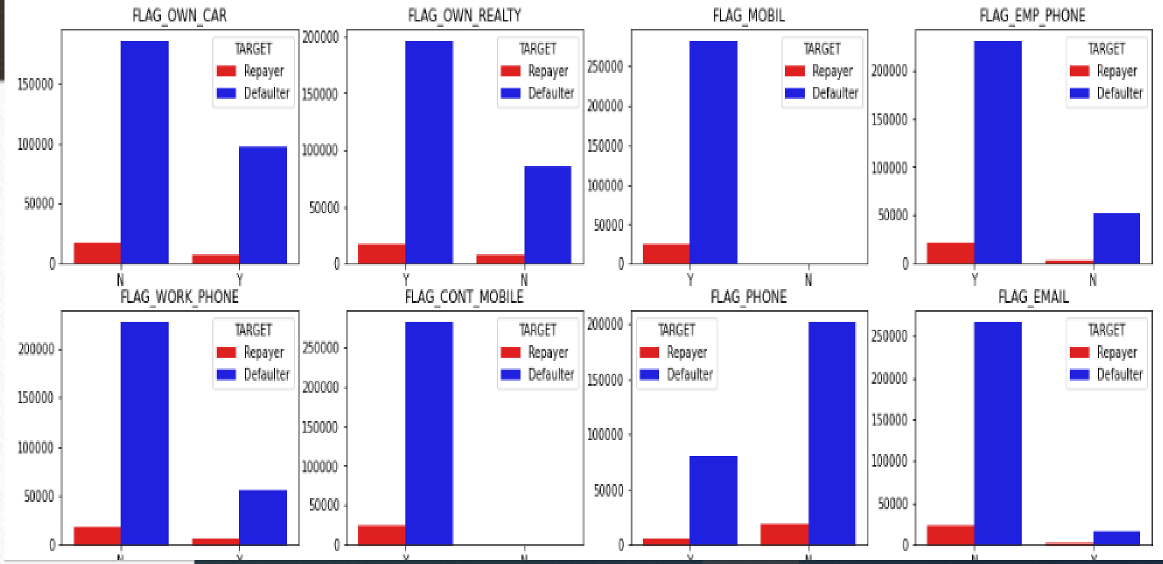
```
flag_dataframe.head()
```

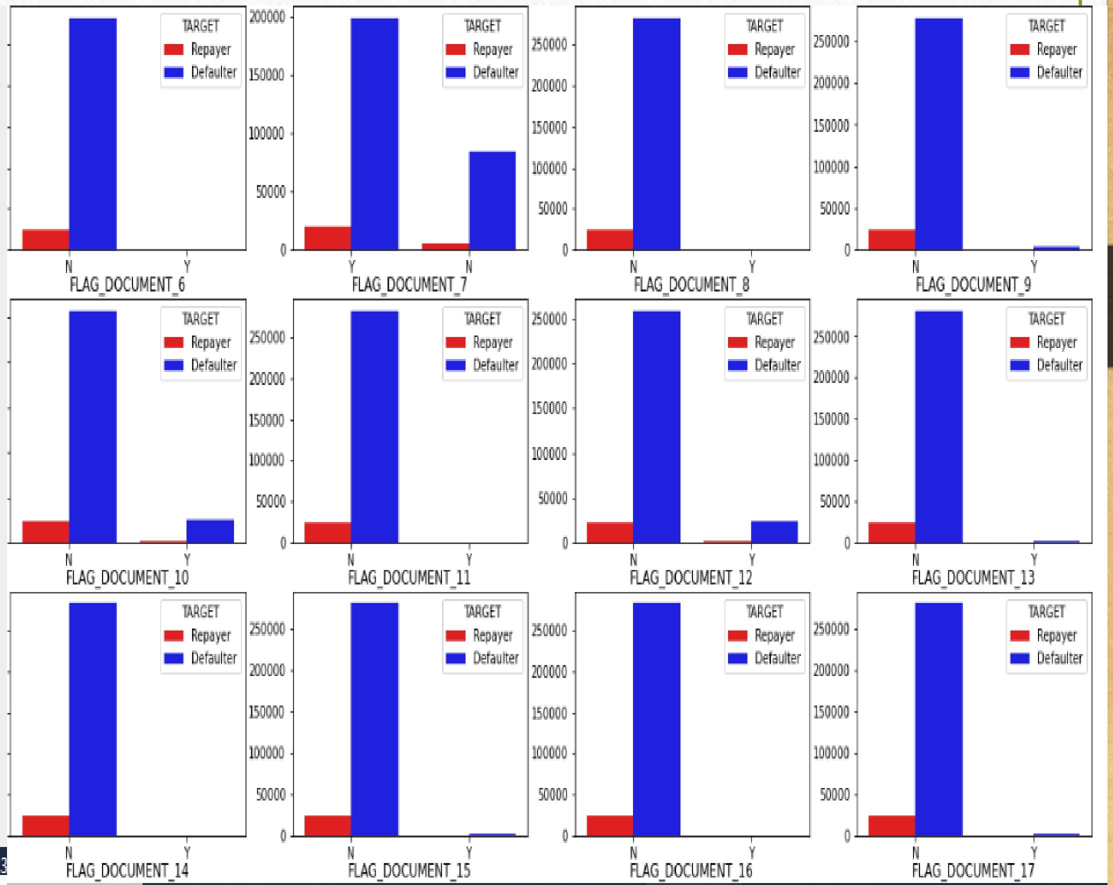
	FLAG_OWN_CAR	FLAG_OWN_REALTY	FLAG_MOBIL	FLAG_EMP_PHONE
0	N	Y	Y	Y
1	N	N	Y	Y
2	Y	Y	Y	Y
3	N	Y	Y	Y
4	N	Y	Y	Y

5 rows × 29 columns

All the graph to find the relation and evaluting for dropping such columns and *Columns (FLAG_OWN_REALTY, FLAG_MOBIL ,FLAG_EMP_PHONE, FLAG_CONT_MOBILE, FLAG_DOCUMENT_3) have more repayers and then defaulter and from these keeping FLAG_DOCUMENT_3,FLAG_OWN_REALTY, FLAG_MOBIL..*

```
plt.figure(figsize = [20,24])

for i,j in itertools.zip_longest(flag_columns,range(len(flag_columns))):
    plt.subplot(7,4,j+1)
    ax = sns.countplot(flag_dataframe[i], hue = flag_dataframe["TARGET"], palette = ["r","b"])
    #plt.yticks(fontsize=8)
    plt.xlabel("")
    plt.ylabel("")
    plt.title(i)
```

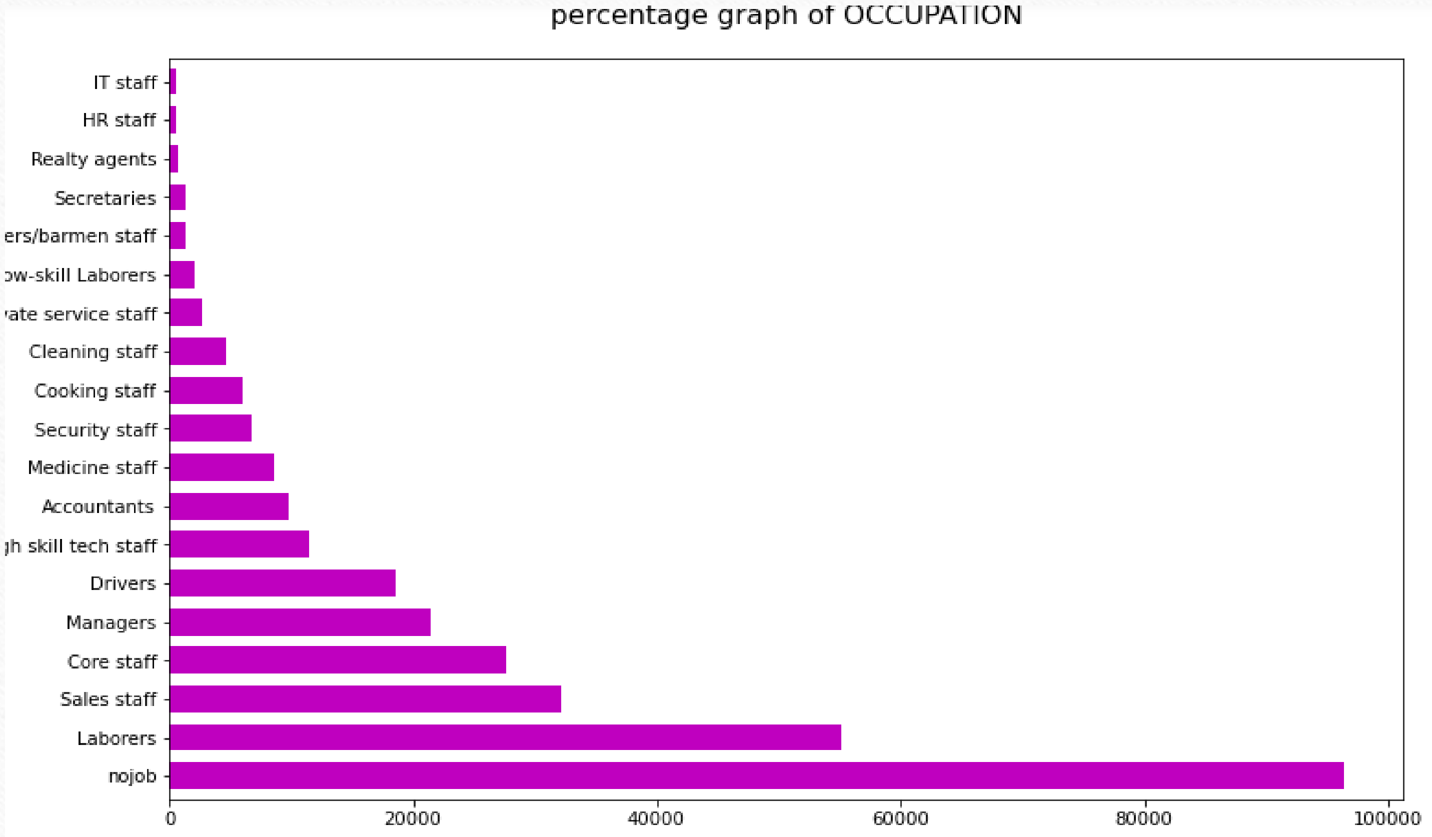




Imputing value of occupation and in the occupation _type which has null _value that rows is imputed as no job.

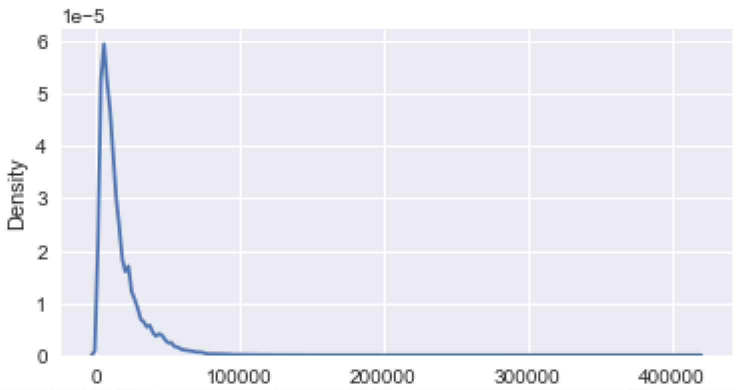
No job column is higher than all jobs columns

```
#Percentage of each category present in "OCCUPATION_TYPE

appl["OCCUPATION_TYPE"].value_counts(normalize=True)*100

Laborers                 26.139636
Sales staff              15.205570
Core staff               13.058924
Managers                 10.122679
Drivers                   8.811576
High skill tech staff     5.390299
Accountants               4.648067
Medicine staff            4.043672
Security staff            3.183498
Cooking staff             2.816408
Cleaning staff            2.203960
Private service staff     1.256158
Low-skill Laborers        0.991379
Waiters/barmen staff      0.638499
Secretaries               0.618132
Realty agents             0.355722
HR staff                  0.266673
IT staff                  0.249147
Name: OCCUPATION_TYPE, dtype: float64
```

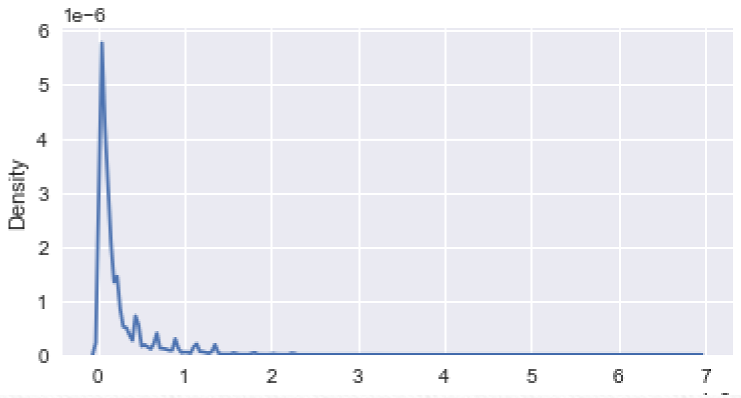


To impute null values in continuous variables, we plotted the distribution of the columns and used.

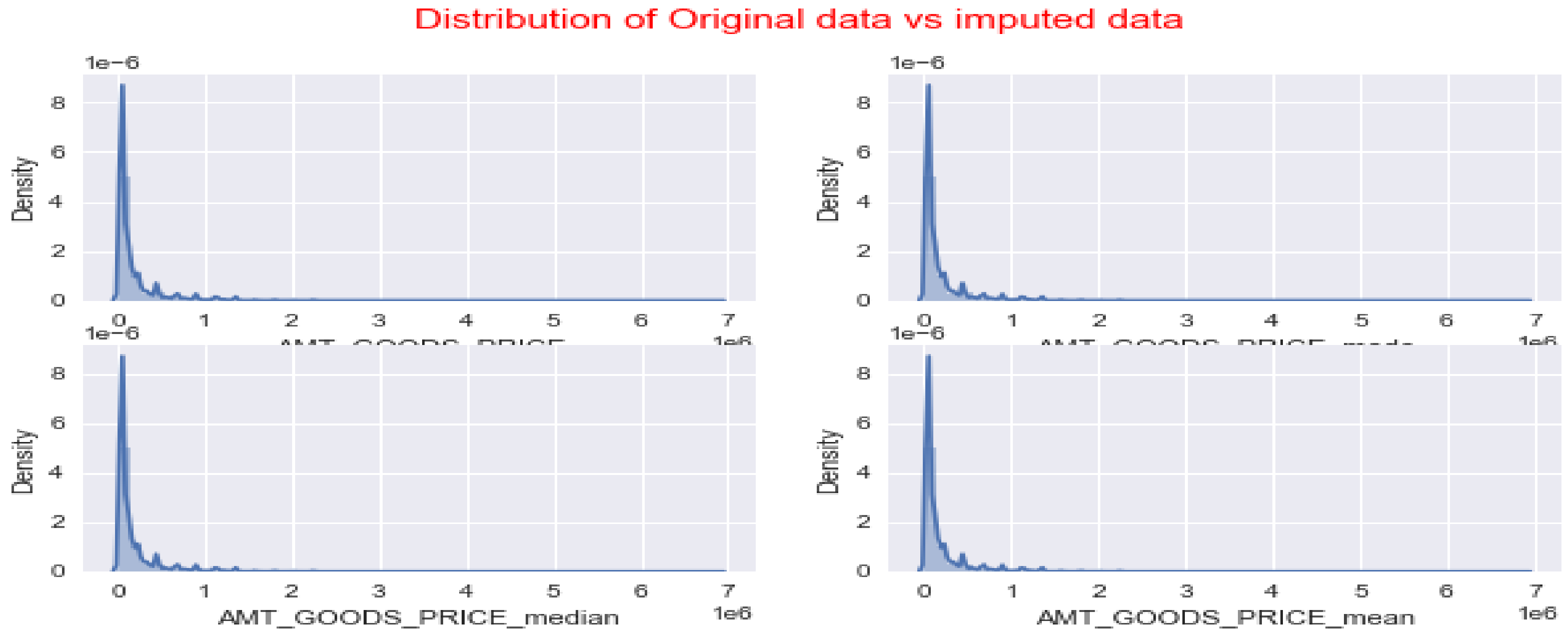
```python
#plotting a kdeplot to understand distribution of "AMT_ANNU

plt.figure(figsize=(6,3))
sns.kdeplot(pa_df['AMT_ANNUITY'])
plt.show()
```




```python
# Plotting kde plot for "AMT_GOODS_PRICE" to understand the distribution

plt.figure(figsize=(6,3))
sns.kdeplot(pa_df['AMT_GOODS_PRICE'])
plt.show()
```



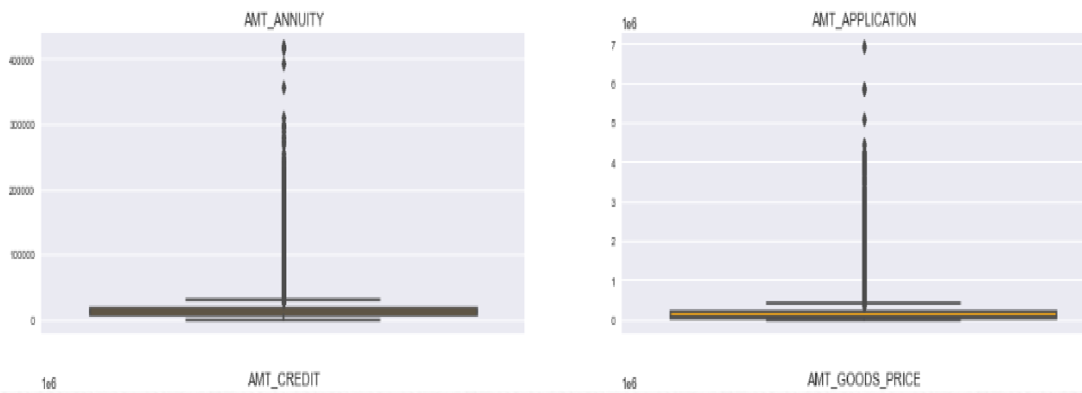
```
colo = ['AMT_GOODS_PRICE_mode', 'AMT_GOODS_PRICE_median','AMT_GOODS_PRICE_mean']

plt.figure(figsize=(10,5))
plt.suptitle('Distribution of Original data vs imputed data',fontsize=15,color='r')
plt.subplot(221)
sns.distplot(pa_df['AMT_GOODS_PRICE'][pd.notnull(pa_df['AMT_GOODS_PRICE'])]);
for i in enumerate(colo):
    plt.subplot(2,2,i[0]+2)
    sns.distplot(statsDF[i[1]])
```

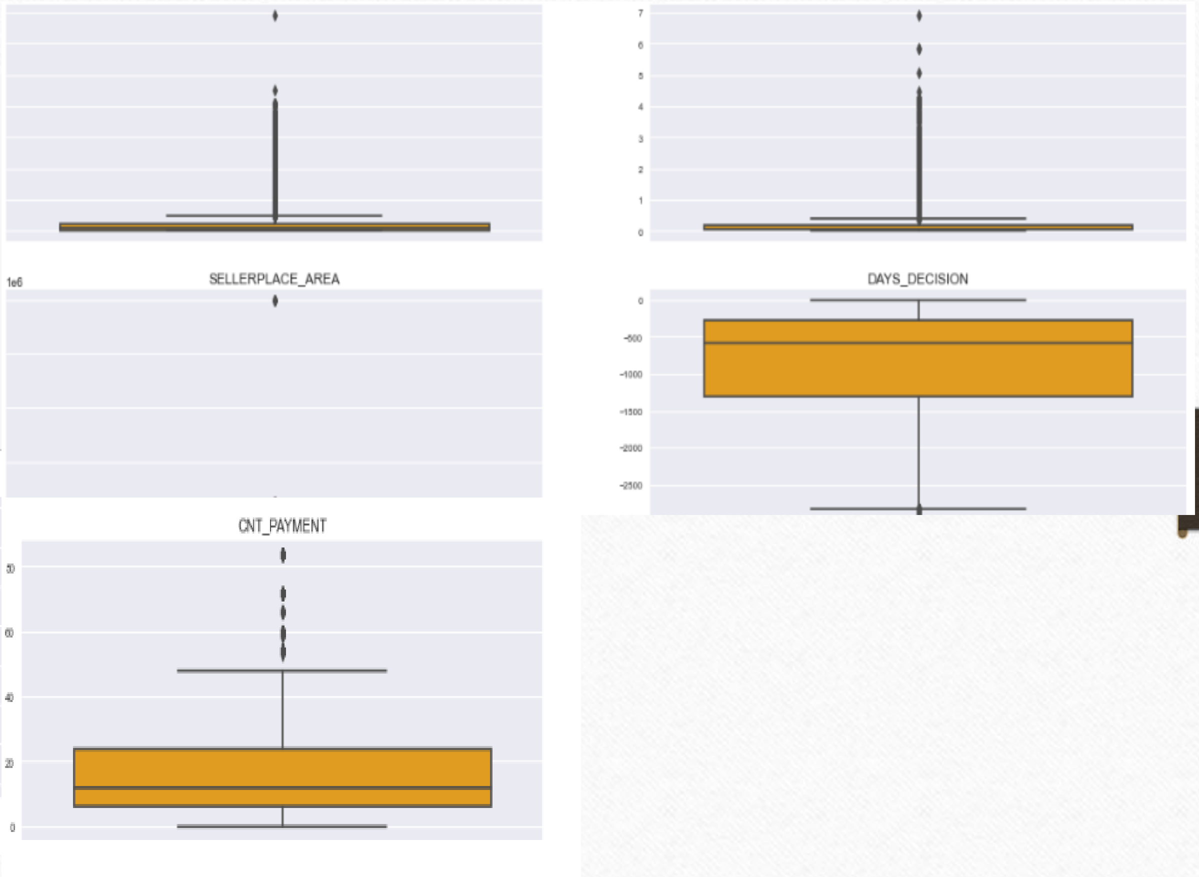


The original distribution is closer with the distribution of data imputed with mode in this case, thus will impute mode for missing values.

Outlier of columns:

```
p_outlier_col = ['AMT_ANNUITY','AMT_APPLICATION','AMT_CREDIT','AMT_GOODS_PRICE',
                 'SELLERPLACE_AREA','DAYS_DECISION','CNT_PAYMENT']

plt.figure(figsize=[20,15])
for i,j in itertools.zip_longest(p_outlier_col, range(len(p_outlier_col))):
    plt.subplot(4,2,j+1)
    sns.boxplot(y = pa_df[i], orient = "h", color = "orange")
    plt.yticks(fontsize=8)
    plt.xlabel("")
    plt.ylabel("")
    plt.title(i)
```

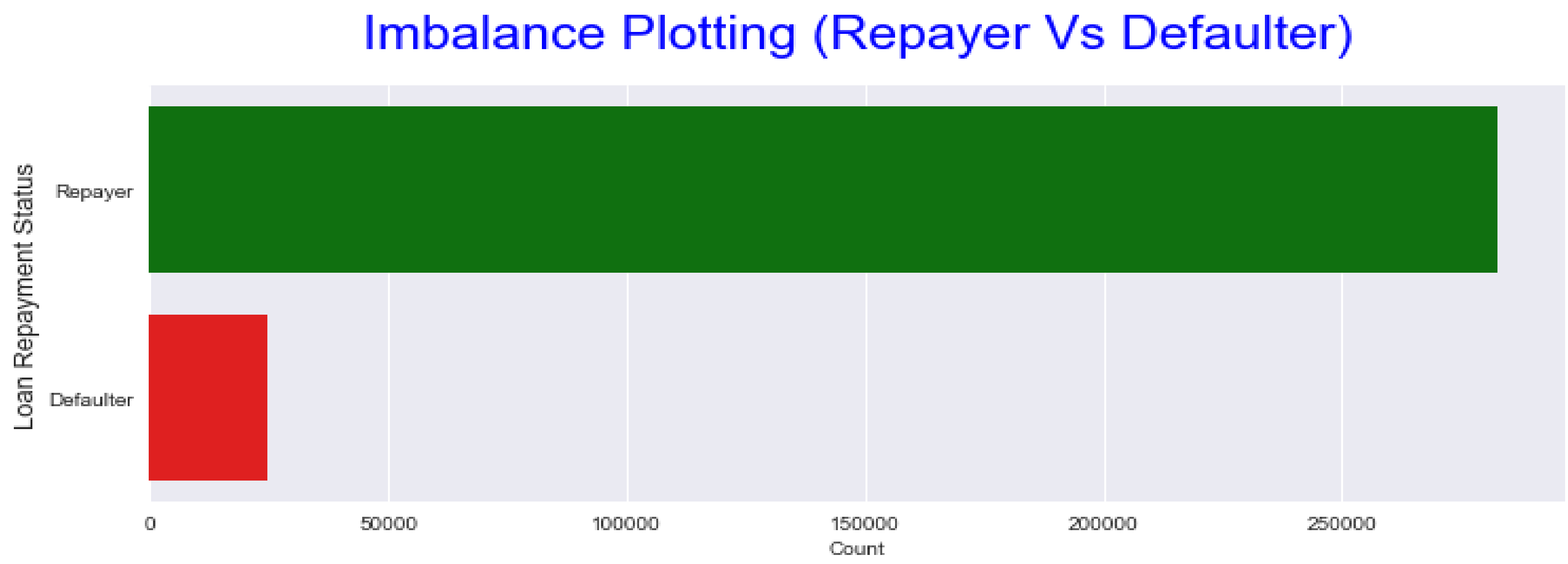




AMT_ANNUITY, AMT_APPLICATION, AMT_CREDIT, AMT_GOODS_PRICE, SELLERPLACE_AREA have huge number of outliers. .CNT_PAYMENT has few outlier values. .DAYS_DECISION has little number of outliers indicating that these previous applications decisions

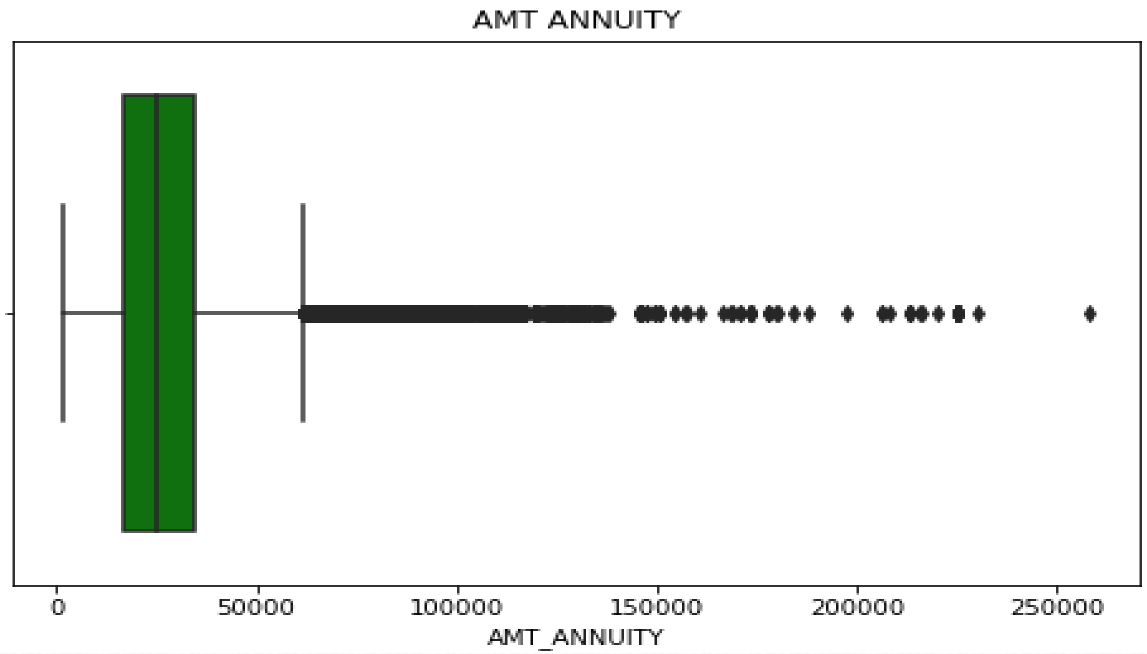
Imbalance Data :

```
plt.figure(figsize= [12,4])
sns.barplot(y=["Repayer","Defaulter"], x = appl["TARGET"].value_counts(), palette = ["g","r"],orient="h")
plt.ylabel("Loan Repayment Status",fontdict = {"fontsize":13})
plt.xlabel("Count",fontdict = {"fontsize":10})
plt.title("Imbalance Plotting (Repayer Vs Defaulter)", fontdict = {"fontsize":25}, pad = 20,color='b')
plt.show()
```



Univariate analysis : univariate analysis of outlier columes of numerical values.

```
plt.figure(figsize=[8,5])
sns.boxplot(appl.AMT_ANNUITY,color="g")
plt.title("AMT ANNUITY")
plt.show()
```

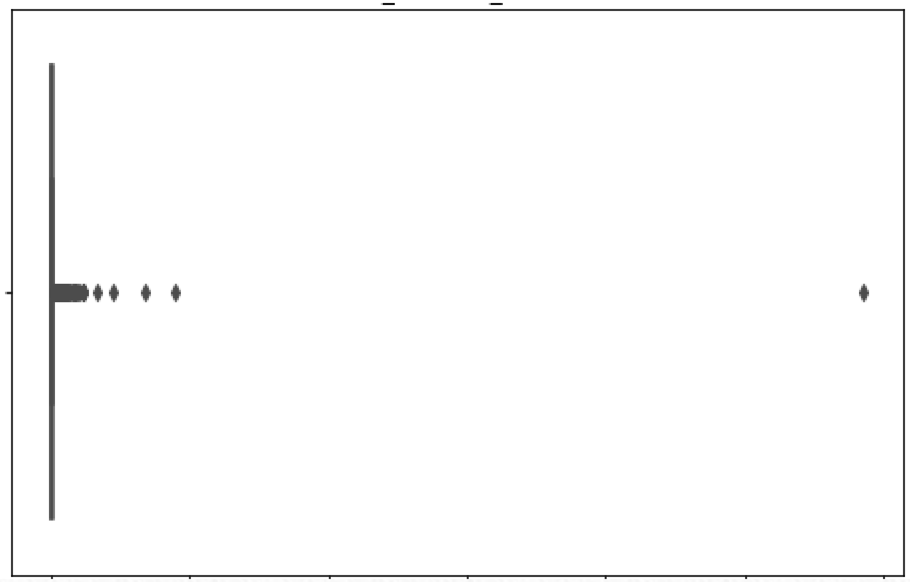



```
appl["AMT_ANNUITY"].describe()
```

```
count    307499.000000
mean      27108.573909
std       14493.737315
min        1615.500000
25%       16524.000000
50%       24903.000000
75%       34596.000000
max      258025.500000
Name: AMT_ANNUITY, dtype: float64
```

Analysis of Amt_income_total

```
plt.figure(figsize=[8,5])
sns.boxplot(appl.AMT_INCOME_TOTAL, color="red")
plt.title("AMT_INCOME_TOTAL")
plt.show()
```

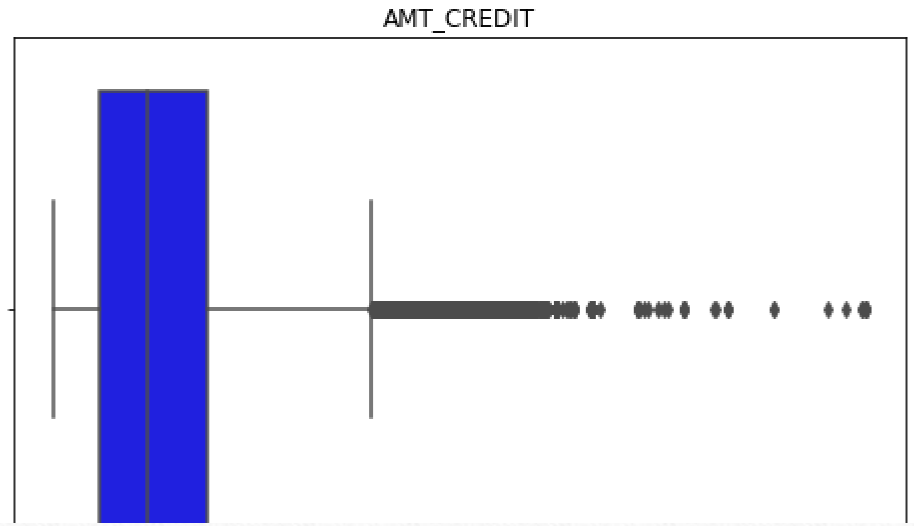



```
appl["AMT_INCOME_TOTAL"].describe()
```

```
count    307511.000000
mean          1.687979
std           2.371231
min           0.256500
25%           1.125000
50%           1.471500
75%           2.025000
max        1170.000000
Name: AMT_INCOME_TOTAL, dtype: float64
```

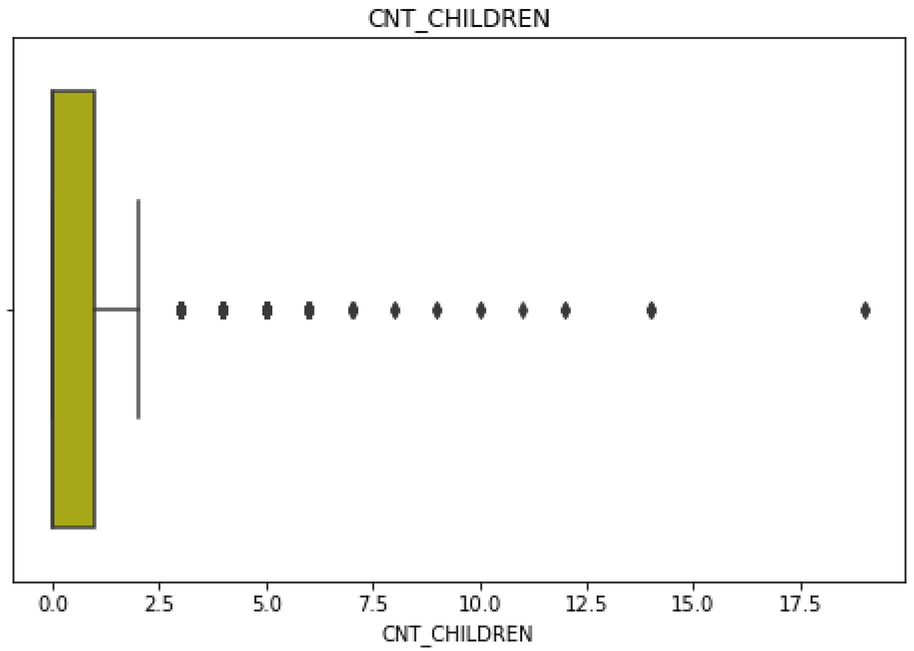
Analysis of "AMT_CREDIT"

```
plt.figure(figsize=[8,5])
sns.boxplot(appl.AMT_CREDIT, color="b")
plt.title("AMT_CREDIT")
plt.show()
```



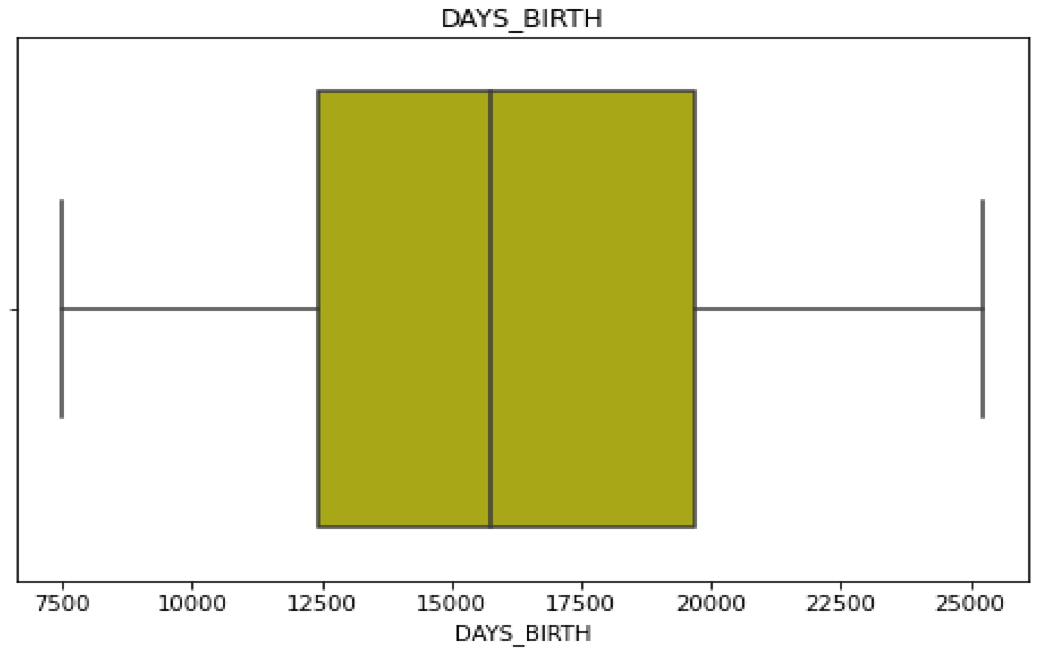
Analysis of "CNT_CHILDREN"

```
plt.figure(figsize=[8,5])
sns.boxplot(appl.CNT_CHILDREN, color="y")
plt.title("CNT_CHILDREN")
plt.show()
```



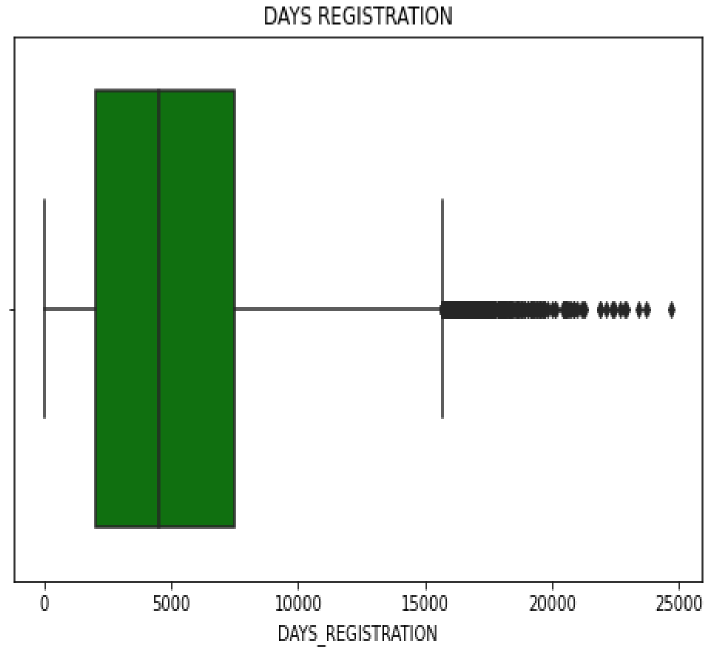
Analysis of "DAYS_BIRTH"

```
plt.figure(figsize=[8,5])
sns.boxplot(appl.DAYS_BIRTH, color="y")
plt.title("DAYS_BIRTH")
plt.show()
```



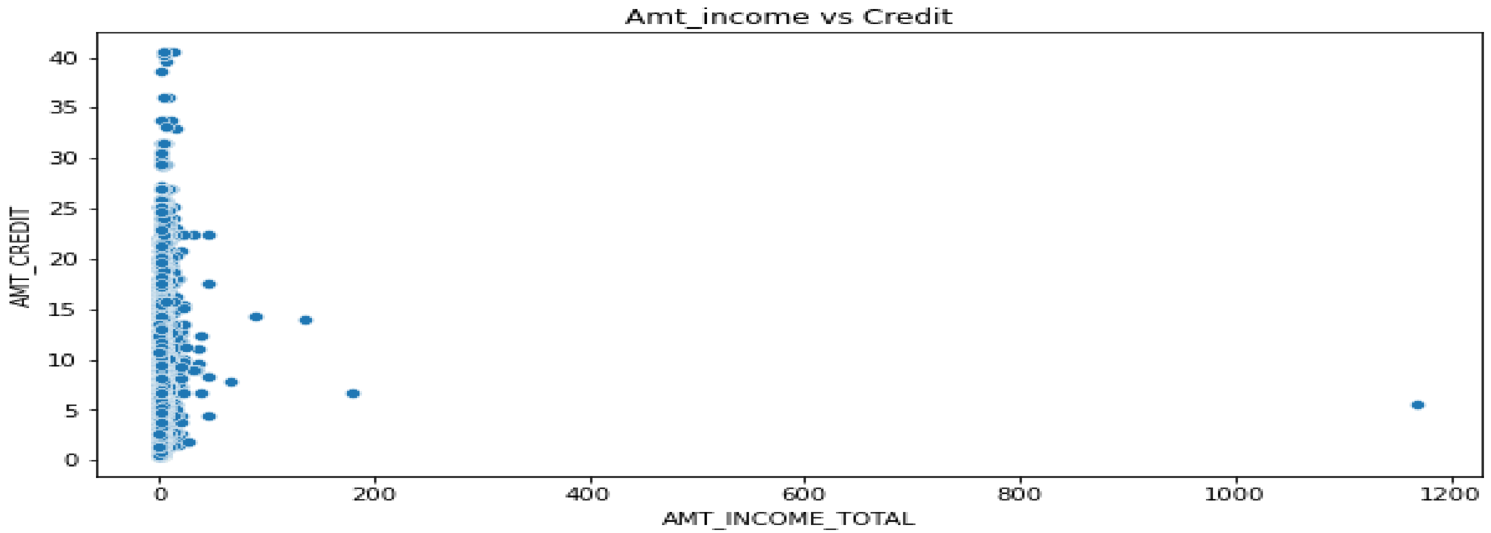
Analysis of "DAYS REGISTRATION"

```
plt.figure(figsize=[8,5])
sns.boxplot(appl.DAYS_REGISTRATION,color="g")
plt.title("DAYS REGISTRATION")
plt.show()
```



Bivariate Analysis

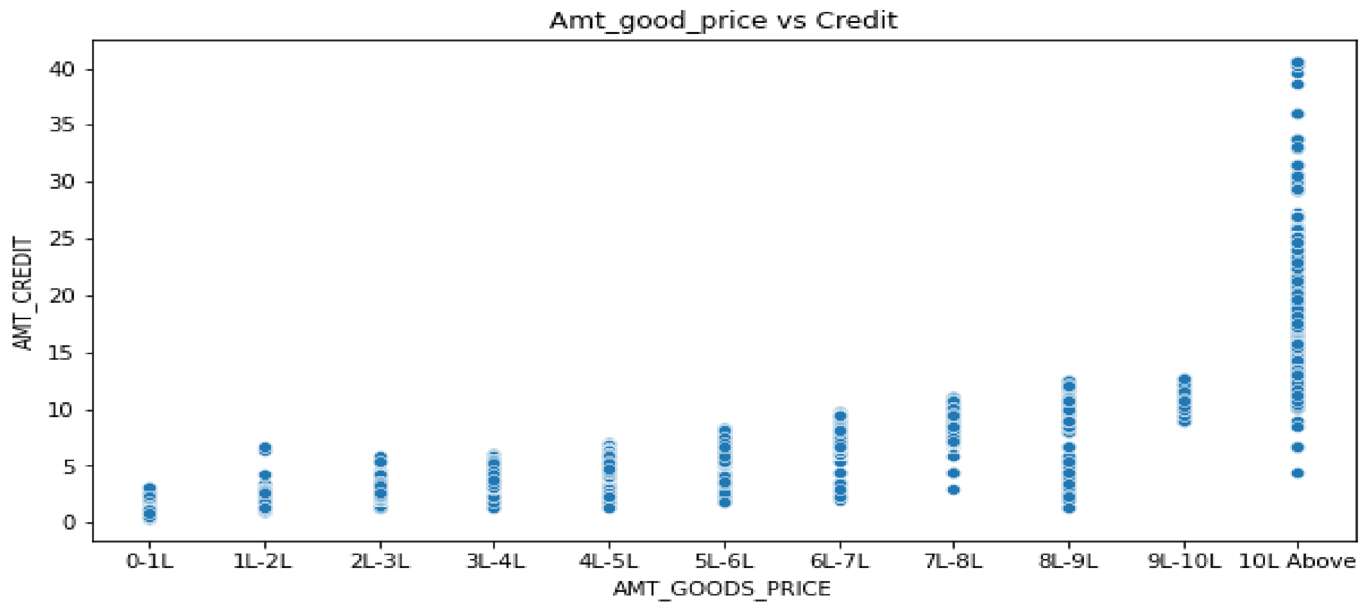
```
plt.figure(figsize=[10,5])
sns.scatterplot(y = appl.AMT_CREDIT, x = appl.AMT_INCOME_TOTAL, data =appl[appl.AMT_CREDIT<18.54000])
plt.title("Amt_income vs Credit")
plt.show()
```



In this bivariate analysis Amt_income_total is greater than Amt _credit

Analysis of "Amt_good_price vs Credit"

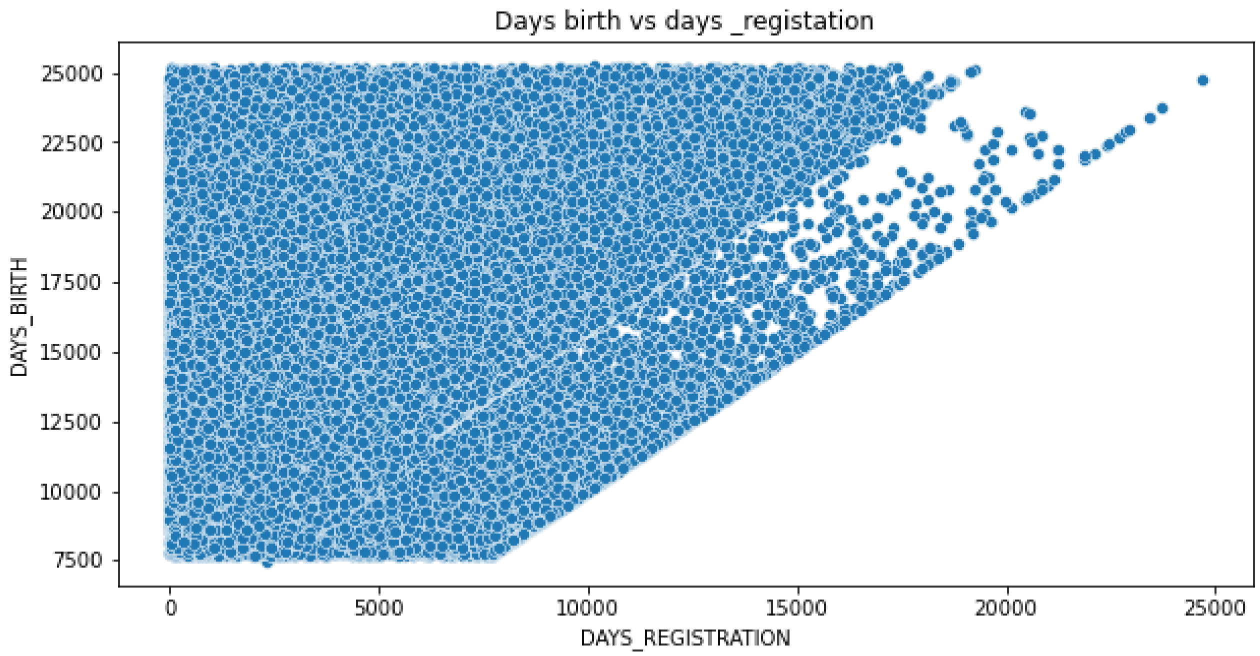
```
plt.figure(figsize=[10,5])
sns.scatterplot(y = appl.AMT_CREDIT, x = appl.AMT_GOODS_PRICE, data =appl[appl.AMT_CREDIT<18.54000])
plt.title("Amt_good_price vs Credit")
plt.show()
```



In this bivariate analysis used to find out that in which range amount high.

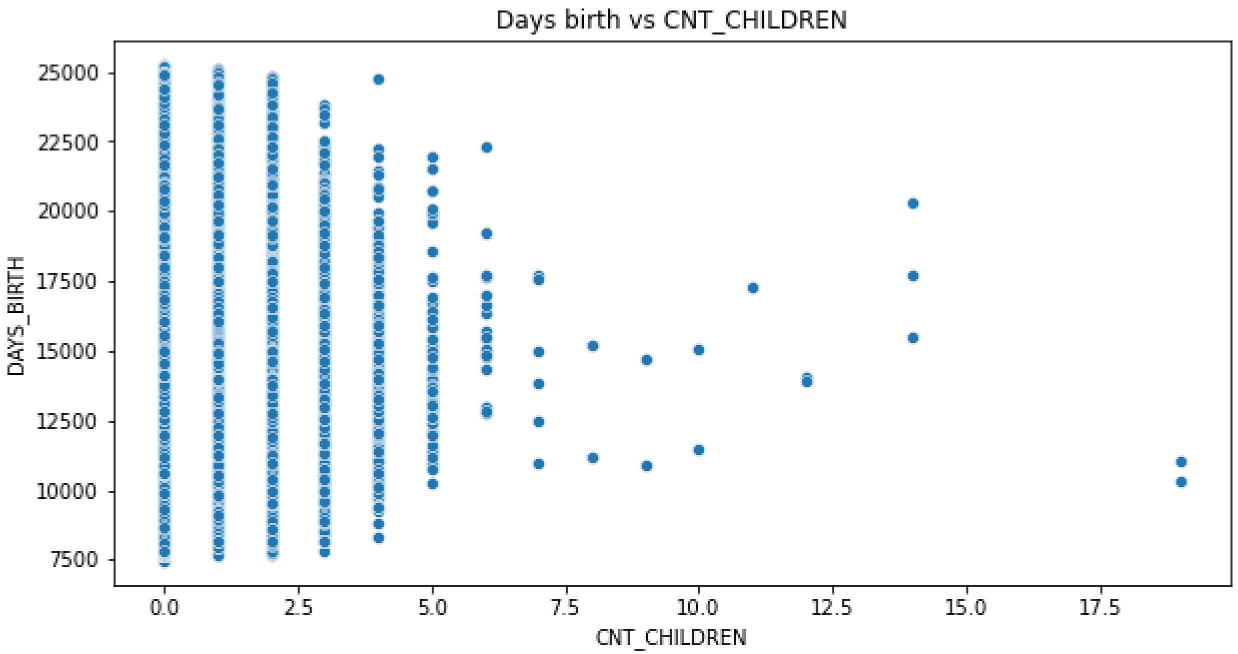
Analysis of Days birth vs Days _registation

```
plt.figure(figsize=[10,5])
sns.scatterplot(y = appl.DAYS_BIRTH, x = appl.DAYS_REGISTRATION, data =appl[appl.DAYS_BIRTH<24419.0])
plt.title("Days birth vs days _registation")
plt.show()
```

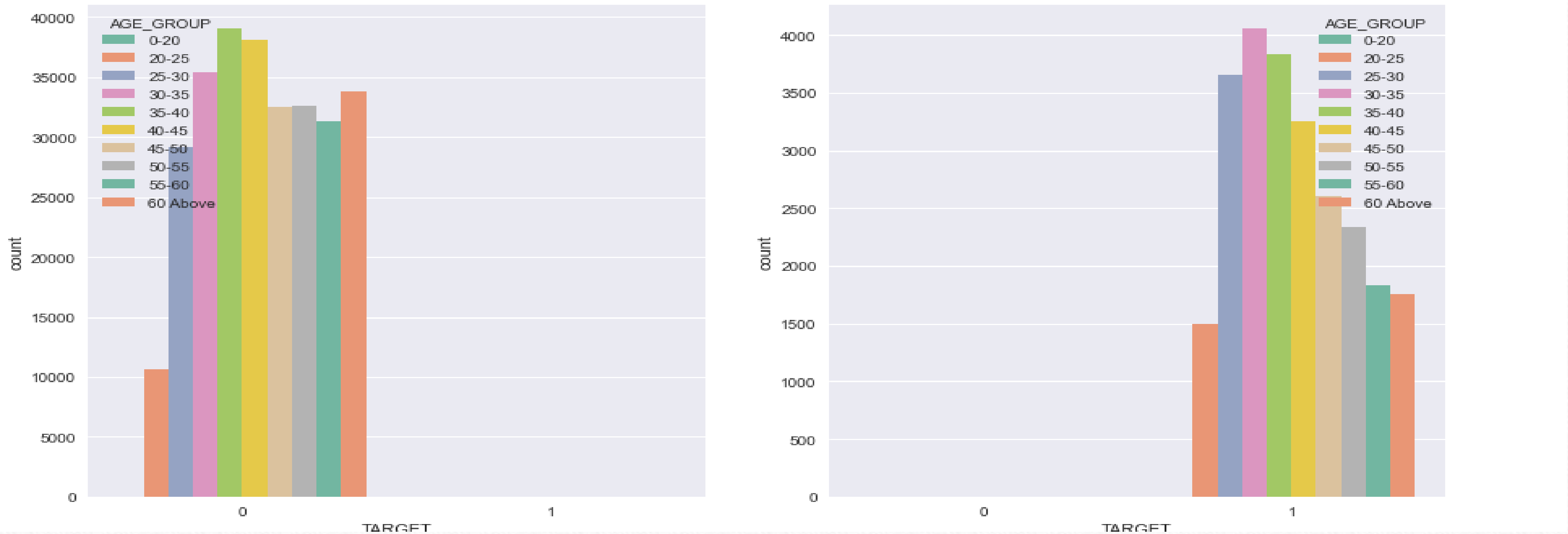


Analysis of Days birth vs Cnt_Children

```
plt.figure(figsize=[10,5])
sns.scatterplot(y = appl.DAYS_BIRTH, x = appl.CNT_CHILDREN, data =appl[appl.DAYS_BIRTH<24419.0])
plt.title("Days birth vs CNT_CHILDREN")
plt.show()
```



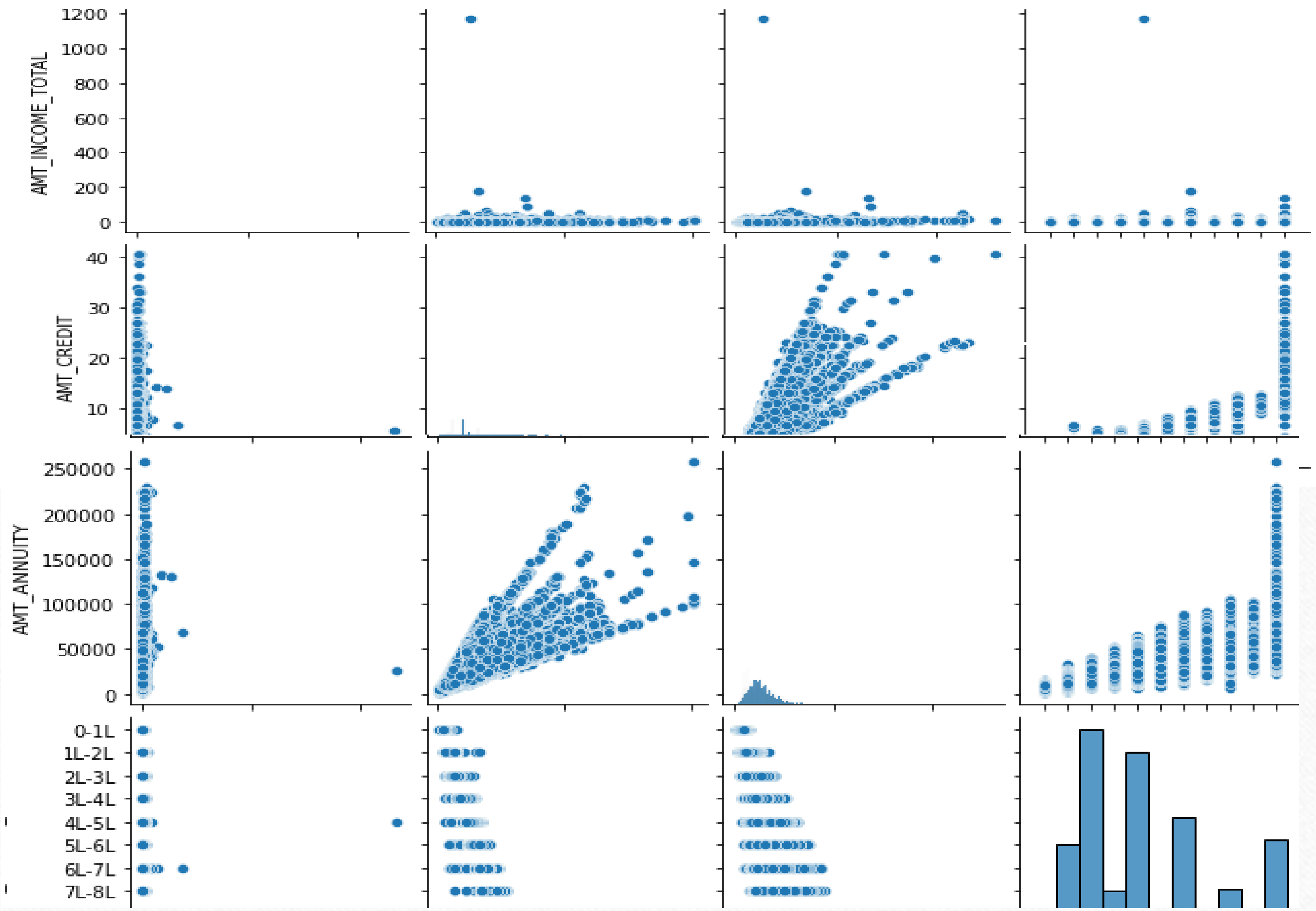
```
plt.figure(figsize=(15,7))
plt.subplot(121)
sns.countplot(x='TARGET',hue='AGE_GROUP',data=data0,palette='Set2')
plt.subplot(122)
sns.countplot(x='TARGET',hue='AGE_GROUP',data=data1,palette='Set2')
plt.show()
```



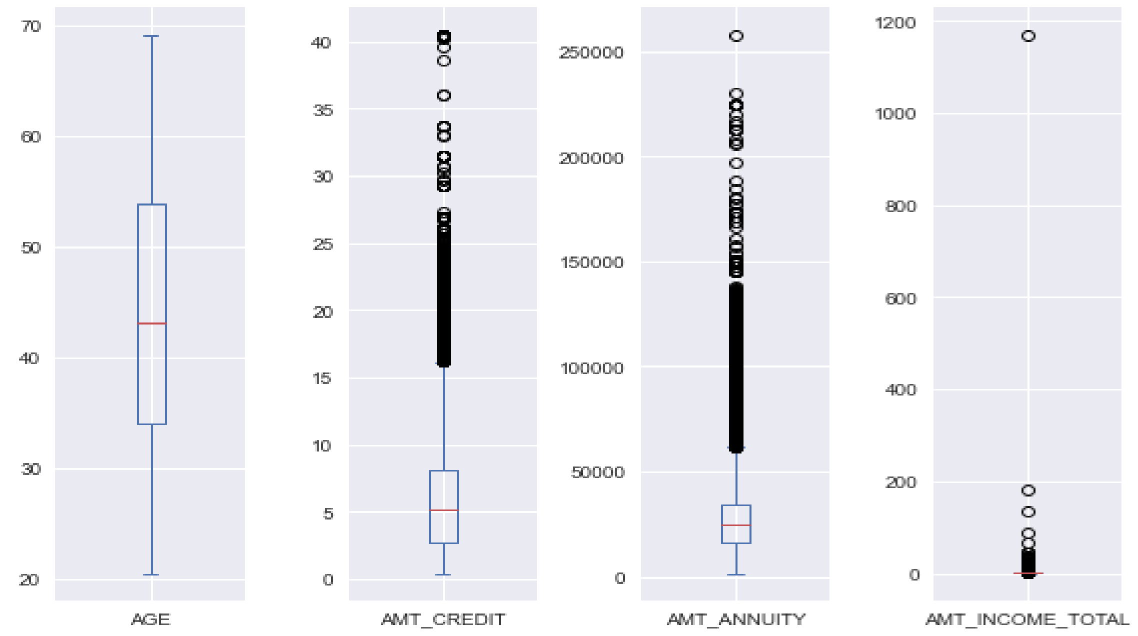
This plot used for age group which are defaulter or repayer

Multi Variant analysis :

```
sns.pairplot(data = appl , vars = ['AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','AMT_GOODS_PRICE'])
plt.show()
```

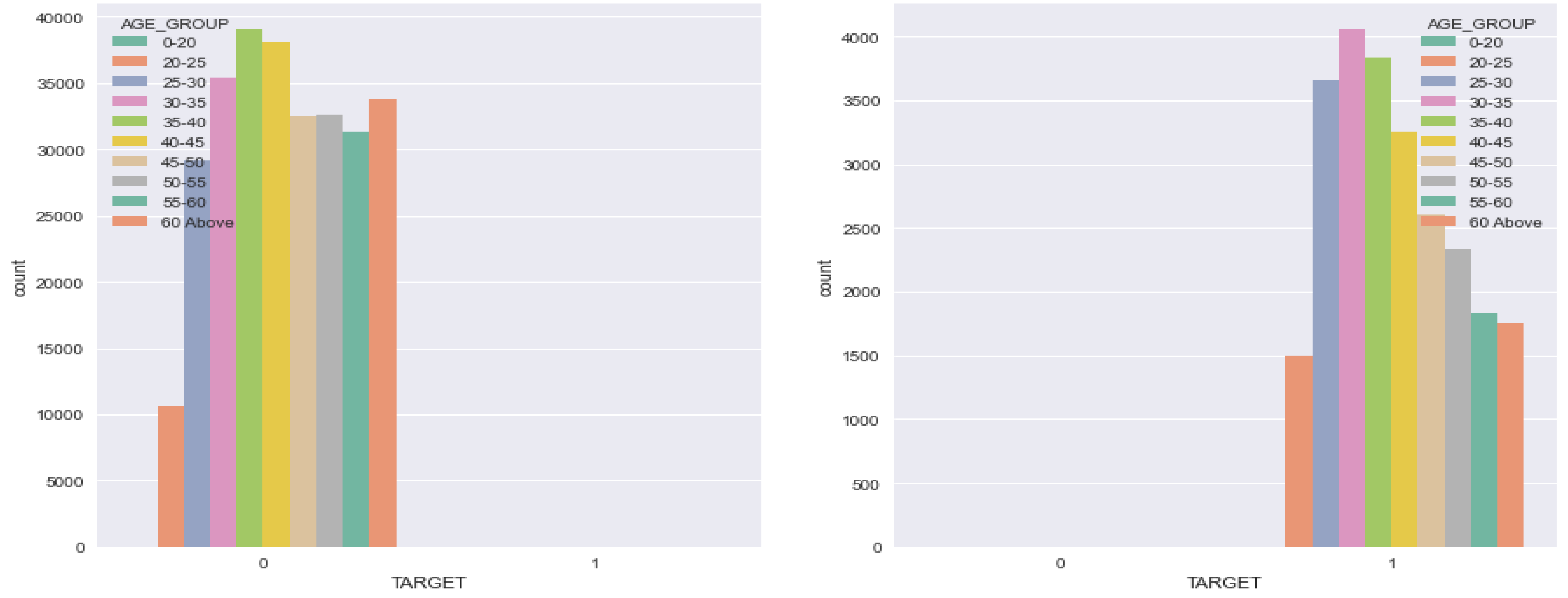


```
plt.style.use('seaborn')
appl[['AGE','AMT_CREDIT','AMT_ANNUITY','AMT_INCOME_TOTAL']].plot(kind='box', subplots=True)
plt.tight_layout()
plt.show()
```



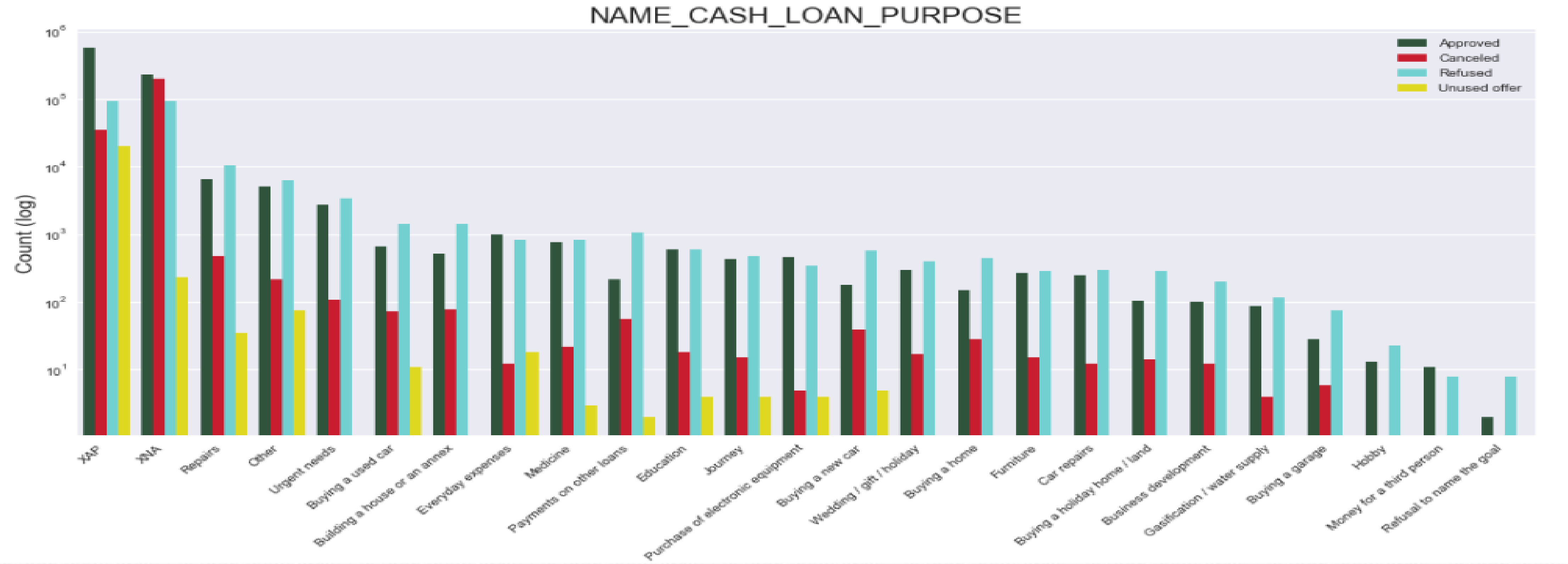
This plot is used the outlier of Age, amt_credit amt_annuity, amt_income_total by the box plot.

```
plt.figure(figsize=(15,7))
plt.subplot(121)
sns.countplot(x='TARGET',hue='AGE_GROUP',data=data0,palette='Set2')
plt.subplot(122)
sns.countplot(x='TARGET',hue='AGE_GROUP',data=data1,palette='Set2')
plt.show()
```

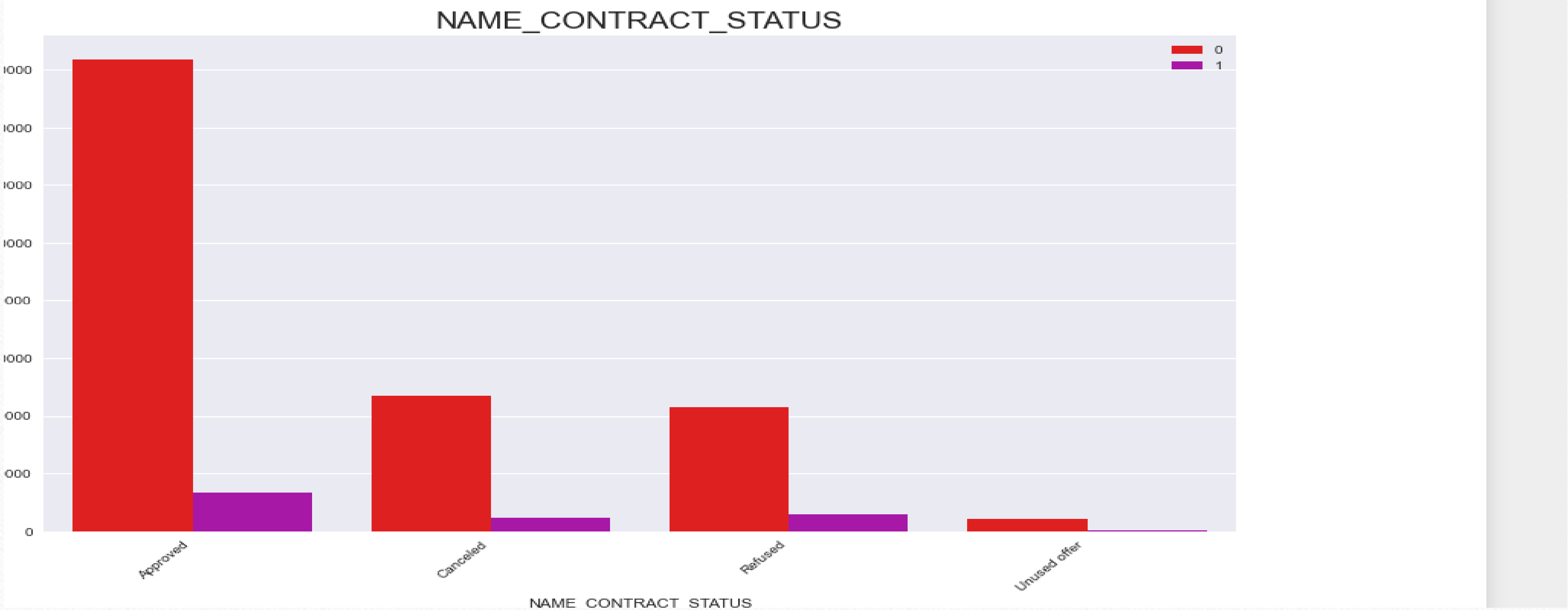


This plot denotes that age group of deafaulter and repayer

```
univariate_merge("NAME_CASH_LOAN_PURPOSE",L0,"NAME_CONTRACT_STATUS",["#295939","#e40017","#64dfdf","#fff600"],True,(18,7))
```



This plot denotes that number of approved, refused, canceled,unused application form.



This plot is used to Approved,canceled , refused, unused offer by the name of contract status

CONCULISION :

- we have extensively covered pre-processing steps required to analyze data
- We have covered Null value imputation methods
- We have also covered step by step analyzing techniques such as Univariate analysis, Bivariate analysis, Multivariate analysis, etc

Thank you