BANK LOAD CASE STUDY

Description

Company faces a challenge: some customers who don't have a sufficient credit history take advantage of this and default on their loans. Your task is to use Exploratory Data Analysis (EDA) to analyze patterns in the data and ensure that capable applicants are not rejected.

When a customer applies for a loan, your company faces two risks:

1. If the applicant can repay the loan but is not approved, the company loses business.
2. If the applicant cannot repay the loan and is approved, the company faces a financial loss.

The dataset you'll be working with contains information about loan applications. It includes two types of scenarios:

1. Customers with payment difficulties: These are customers who had a late payment of more than X days on at least one of the first Y installments of the loan.
2. All other cases: These are cases where the payment was made on time.

When a customer applies for a loan, there are four possible outcomes:

1. Approved: The company has approved the loan application.
2. Cancelled: The customer cancelled the application during the approval process.
3. Refused: The company rejected the loan.
4. Unused Offer: The loan was approved but the customer did not use it.

Your goal in this project is to use EDA to understand how customer attributes and loan attributes influence the likelihood of default.

Importing all the necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import openpyxl
import seaborn as sns
```

Loading the Datasets

```
adt = pd.read_csv(r"D:\Trainity Projects\Project 5\
application_data.csv")


print("about the applications dataset: ",adt.shape)

about the applications dataset:  (307511, 122)

adt.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
 8    AMT_CREDIT                     float64
 9    AMT_ANNUITY                    float64
 10   AMT_GOODS_PRICE                float64
 11   NAME_TYPE_SUITE                object
 12   NAME_INCOME_TYPE               object
 13   NAME_EDUCATION_TYPE            object
 14   NAME_FAMILY_STATUS             object
 15   NAME_HOUSING_TYPE              object
 16   REGION_POPULATION_RELATIVE     float64
 17   DAYS_BIRTH                     int64
 18   DAYS_EMPLOYED                  int64
 19   DAYS_REGISTRATION              float64
 20   DAYS_ID_PUBLISH                int64
 21   OWN_CAR_AGE                    float64
 22   FLAG_MOBIL                     int64
 23   FLAG_EMP_PHONE                 int64
 24   FLAG_WORK_PHONE                int64
 25   FLAG_CONT_MOBILE               int64
 26   FLAG_PHONE                     int64
 27   FLAG_EMAIL                     int64
 28   OCCUPATION_TYPE                object
 29   CNT_FAM_MEMBERS                float64
 30   REGION_RATING_CLIENT           int64
 31   REGION_RATING_CLIENT_W_CITY    int64
 32   WEEKDAY_APPR_PROCESS_START     object
 33   HOUR_APPR_PROCESS_START        int64
 34   REG_REGION_NOT_LIVE_REGION     int64
 35   REG_REGION_NOT_WORK_REGION     int64
 36   LIVE_REGION_NOT_WORK_REGION    int64
 37   REG_CITY_NOT_LIVE_CITY         int64
 38   REG_CITY_NOT_WORK_CITY         int64
 39   LIVE_CITY_NOT_WORK_CITY        int64
 40   ORGANIZATION_TYPE              object
 41   EXT_SOURCE_1                   float64
 42   EXT_SOURCE_2                   float64
 43   EXT_SOURCE_3                   float64
```

```
 44   APARTMENTS_AVG                float64
 45   BASEMENTAREA_AVG              float64
 46   YEARS_BEGINEXPLUATATION_AVG   float64
 47   YEARS_BUILD_AVG               float64
 48   COMMONAREA_AVG                float64
 49   ELEVATORS_AVG                 float64
 50   ENTRANCES_AVG                 float64
 51   FLOORSMAX_AVG                 float64
 52   FLOORSMIN_AVG                 float64
 53   LANDAREA_AVG                  float64
 54   LIVINGAPARTMENTS_AVG          float64
 55   LIVINGAREA_AVG                float64
 56   NONLIVINGAPARTMENTS_AVG       float64
 57   NONLIVINGAREA_AVG             float64
 58   APARTMENTS_MODE               float64
 59   BASEMENTAREA_MODE             float64
 60   YEARS_BEGINEXPLUATATION_MODE  float64
 61   YEARS_BUILD_MODE              float64
 62   COMMONAREA_MODE               float64
 63   ELEVATORS_MODE                float64
 64   ENTRANCES_MODE                float64
 65   FLOORSMAX_MODE                float64
 66   FLOORSMIN_MODE                float64
 67   LANDAREA_MODE                 float64
 68   LIVINGAPARTMENTS_MODE         float64
 69   LIVINGAREA_MODE               float64
 70   NONLIVINGAPARTMENTS_MODE      float64
 71   NONLIVINGAREA_MODE            float64
 72   APARTMENTS_MEDI               float64
 73   BASEMENTAREA_MEDI             float64
 74   YEARS_BEGINEXPLUATATION_MEDI  float64
 75   YEARS_BUILD_MEDI              float64
 76   COMMONAREA_MEDI               float64
 77   ELEVATORS_MEDI                float64
 78   ENTRANCES_MEDI                float64
 79   FLOORSMAX_MEDI                float64
 80   FLOORSMIN_MEDI                float64
 81   LANDAREA_MEDI                 float64
 82   LIVINGAPARTMENTS_MEDI         float64
 83   LIVINGAREA_MEDI               float64
 84   NONLIVINGAPARTMENTS_MEDI      float64
 85   NONLIVINGAREA_MEDI            float64
 86   FONDKAPREMONT_MODE            object
 87   HOUSETYPE_MODE                object
 88   TOTALAREA_MODE                float64
 89   WALLSMATERIAL_MODE            object
 90   EMERGENCYSTATE_MODE           object
 91   OBS_30_CNT_SOCIAL_CIRCLE      float64
 92   DEF_30_CNT_SOCIAL_CIRCLE      float64
```

```
 93   OBS_60_CNT_SOCIAL_CIRCLE      float64
 94   DEF_60_CNT_SOCIAL_CIRCLE      float64
 95   DAYS_LAST_PHONE_CHANGE        float64
 96   FLAG_DOCUMENT_2               int64
 97   FLAG_DOCUMENT_3               int64
 98   FLAG_DOCUMENT_4               int64
 99   FLAG_DOCUMENT_5               int64
 100  FLAG_DOCUMENT_6               int64
 101  FLAG_DOCUMENT_7               int64
 102  FLAG_DOCUMENT_8               int64
 103  FLAG_DOCUMENT_9               int64
 104  FLAG_DOCUMENT_10              int64
 105  FLAG_DOCUMENT_11              int64
 106  FLAG_DOCUMENT_12              int64
 107  FLAG_DOCUMENT_13              int64
 108  FLAG_DOCUMENT_14              int64
 109  FLAG_DOCUMENT_15              int64
 110  FLAG_DOCUMENT_16              int64
 111  FLAG_DOCUMENT_17              int64
 112  FLAG_DOCUMENT_18              int64
 113  FLAG_DOCUMENT_19              int64
 114  FLAG_DOCUMENT_20              int64
 115  FLAG_DOCUMENT_21              int64
 116  AMT_REQ_CREDIT_BUREAU_HOUR    float64
 117  AMT_REQ_CREDIT_BUREAU_DAY     float64
 118  AMT_REQ_CREDIT_BUREAU_WEEK    float64
 119  AMT_REQ_CREDIT_BUREAU_MON     float64
 120  AMT_REQ_CREDIT_BUREAU_QRT     float64
 121  AMT_REQ_CREDIT_BUREAU_YEAR    float64
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

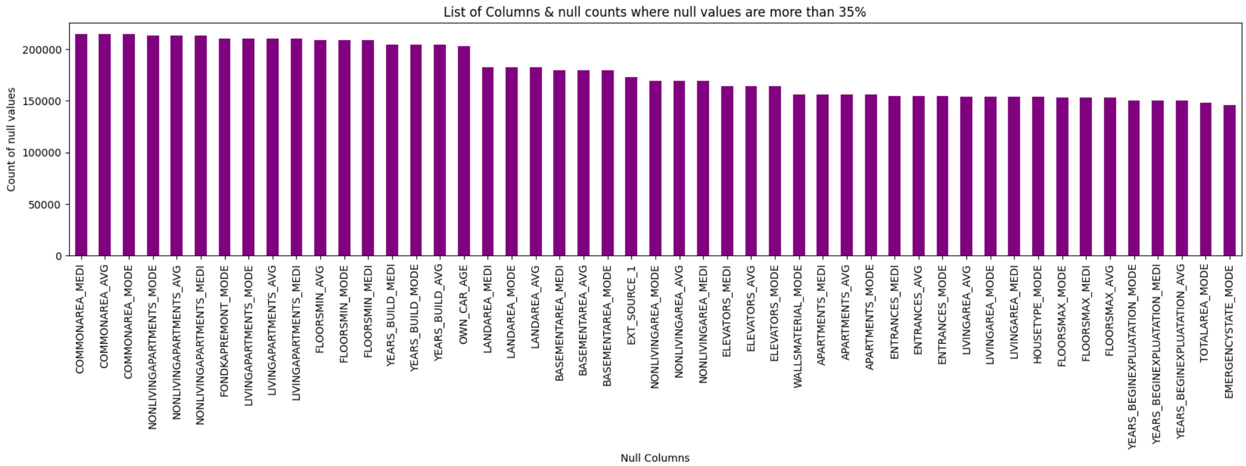
```
n = (adt.isnull().sum()/len(adt)*100).sort_values(ascending =
False).head(50) # gives the null values in the dataset
n
```

```
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
LIVINGAPARTMENTS_MODE           68.354953
LIVINGAPARTMENTS_AVG            68.354953
LIVINGAPARTMENTS_MEDI           68.354953
FLOORSMIN_AVG                   67.848630
FLOORSMIN_MODE                  67.848630
FLOORSMIN_MEDI                  67.848630
YEARS_BUILD_MEDI                66.497784
```

```
YEARS_BUILD_MODE                    66.497784
YEARS_BUILD_AVG                     66.497784
OWN_CAR_AGE                         65.990810
LANDAREA_MEDI                       59.376738
LANDAREA_MODE                       59.376738
LANDAREA_AVG                        59.376738
BASEMENTAREA_MEDI                   58.515956
BASEMENTAREA_AVG                    58.515956
BASEMENTAREA_MODE                   58.515956
EXT_SOURCE_1                        56.381073
NONLIVINGAREA_MODE                  55.179164
NONLIVINGAREA_AVG                   55.179164
NONLIVINGAREA_MEDI                  55.179164
ELEVATORS_MEDI                      53.295980
ELEVATORS_AVG                       53.295980
ELEVATORS_MODE                      53.295980
WALLSMATERIAL_MODE                  50.840783
APARTMENTS_MEDI                     50.749729
APARTMENTS_AVG                      50.749729
APARTMENTS_MODE                     50.749729
ENTRANCES_MEDI                      50.348768
ENTRANCES_AVG                       50.348768
ENTRANCES_MODE                      50.348768
LIVINGAREA_AVG                      50.193326
LIVINGAREA_MODE                     50.193326
LIVINGAREA_MEDI                     50.193326
HOUSETYPE_MODE                      50.176091
FLOORSMAX_MODE                      49.760822
FLOORSMAX_MEDI                      49.760822
FLOORSMAX_AVG                       49.760822
YEARS_BEGINEXPLUATATION_MODE        48.781019
YEARS_BEGINEXPLUATATION_MEDI        48.781019
YEARS_BEGINEXPLUATATION_AVG         48.781019
TOTALAREA_MODE                      48.268517
EMERGENCYSTATE_MODE                 47.398304
OCCUPATION_TYPE                     31.345545
dtype: float64
```

```python
#  Seems like we have alot of null values
nc = adt.isnull().sum().sort_values(ascending=False)
nc = nc[nc.values > (0.35*len(adt))] # contains every column with more than 35% values as null

plt.figure(figsize=(20,4))
nc.plot(kind='bar', color="purple")
plt.title('List of Columns & null counts where null values are more than 35%')
plt.xlabel("Null Columns")                  #Setting X-label and Y-label
```

```
plt.ylabel("Count of null values")
plt.show()
```



we will remove columns with null values of more than 35% after observing those columns.

```
len(nc)
```

```
49
```

We have 49 column with more than 35% null values

Lets remove these columns

```
label = list(nc.index.values) #Making list of column names having null values greater than 35%
adt.drop(labels = label,axis=1,inplace = True)

adt.shape
```

```
(307511, 73)
```

Now that we have successfully removed all the columns with null values over 35%. We will make necessary changes to the "FLAG" columns

```
flag=[col for col in adt.columns if "FLAG" in col]
flag
```

```
['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',
 'FLAG_DOCUMENT_18',
 'FLAG_DOCUMENT_19',
 'FLAG_DOCUMENT_20',
 'FLAG_DOCUMENT_21']
```

```
flag_df=adt[flag+["TARGET"]]
flag_df["TARGET"] = flag_df["TARGET"].replace({1:"Defaulter",
0:"Repayer"}) #according to column description ->1 implies defaulter,
0 implies repayer
```

```
C:\Users\91852\AppData\Local\Temp\ipykernel_15092\3311917187.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  flag_df["TARGET"] = flag_df["TARGET"].replace({1:"Defaulter",
0:"Repayer"}) #according to column description ->1 implies defaulter,
0 implies repayer
```

```
flag_df.head()
```

```
  FLAG_OWN_CAR FLAG_OWN_REALTY  FLAG_MOBIL  FLAG_EMP_PHONE
FLAG_WORK_PHONE
0            N               Y           1               1
0  \
1            N               N           1               1
0
2            Y               Y           1               1
1
3            N               Y           1               1
0
```

```
4             N                Y            1               1
0

   FLAG_CONT_MOBILE  FLAG_PHONE  FLAG_EMAIL  FLAG_DOCUMENT_2
FLAG_DOCUMENT_3
0                 1           1           0                0
1  \
1                 1           1           0                0
1
2                 1           1           0                0
0
3                 1           0           0                0
1
4                 1           0           0                0
0

   ...  FLAG_DOCUMENT_13  FLAG_DOCUMENT_14  FLAG_DOCUMENT_15
0  ...                 0                 0                0  \
1  ...                 0                 0                0
2  ...                 0                 0                0
3  ...                 0                 0                0
4  ...                 0                 0                0

   FLAG_DOCUMENT_16  FLAG_DOCUMENT_17  FLAG_DOCUMENT_18
FLAG_DOCUMENT_19
0                 0                 0                 0
0  \
1                 0                 0                 0
0
2                 0                 0                 0
0
3                 0                 0                 0
0
4                 0                 0                 0
0

   FLAG_DOCUMENT_20  FLAG_DOCUMENT_21     TARGET
0                 0                 0  Defaulter
1                 0                 0    Repayer
2                 0                 0    Repayer
3                 0                 0    Repayer
4                 0                 0    Repayer

[5 rows x 29 columns]
```

```python
import itertools

# Plotting all the graph to find the relation and evaluting for
dropping such columns
```

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

for i,j in itertools.zip_longest(flag,range(len(flag))):
    plt.subplot(7,4,j+1)
    ax = sns.countplot(x=flag_df[i], hue = flag_df["TARGET"], palette
= ["r","seagreen"])
    plt.xlabel("")
    plt.ylabel("")
    plt.title(i)
```


```python
flag_df.drop(["TARGET","FLAG_OWN_REALTY","FLAG_MOBIL","FLAG_DOCUMENT_3
"], axis=1 , inplace = True)

# dropping the columns of "flag_df" dataframe that is removing more 25
columns from "appl_data" dataframe
```

```
adt.drop(flag_df.columns, axis=1, inplace= True)
adt.shape
```

```
C:\Users\91852\AppData\Local\Temp\ipykernel_15092\3460302222.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy

flag_df.drop(["TARGET","FLAG_OWN_REALTY","FLAG_MOBIL","FLAG_DOCUMENT_3
"], axis=1 , inplace = True)

(307511, 48)
```

```
a = (adt.isnull().sum()/len(adt)*100).sort_values(ascending =
False).head(50)
a.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
DEF_60_CNT_SOCIAL_CIRCLE       0.332021
dtype: float64
```

```
adt['OCCUPATION_TYPE'].value_counts(normalize=True)*100
```

```
OCCUPATION_TYPE
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
```

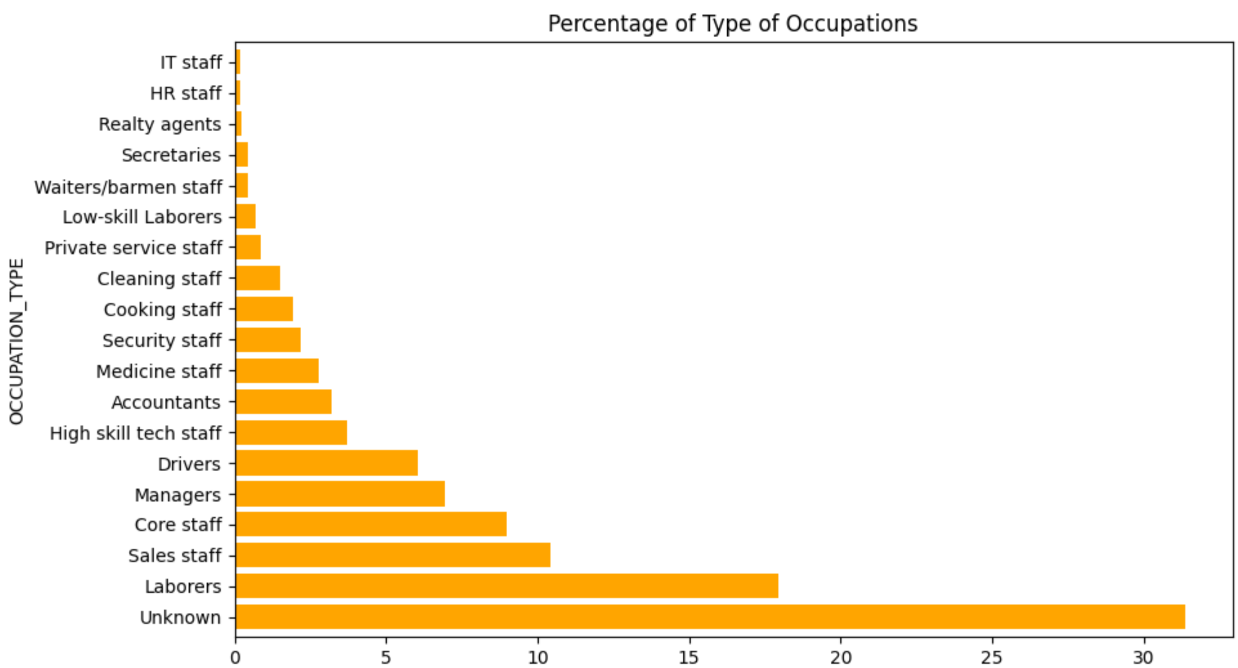
```
HR staff                        0.266673
IT staff                        0.249147
Name: proportion, dtype: float64
```

```
adt['OCCUPATION_TYPE'] = adt['OCCUPATION_TYPE'].fillna("Unknown") #
filling all the null values with unknown
adt['OCCUPATION_TYPE'].isnull().sum()
```

```
0
```

Plotting a graph based on Occupation

```
plt.figure(figsize = (10,6))
plt.title("Percentage of Type of Occupations")
(adt["OCCUPATION_TYPE"].value_counts(normalize=True)*100).plot.barh(co
lor= "orange",width = .8)
plt.show()
```



Now we will tackle the missing values in the numerical columns

```
(adt.isnull().sum()).sort_values(ascending = False)
```

```
EXT_SOURCE_3                  60965
AMT_REQ_CREDIT_BUREAU_YEAR    41519
AMT_REQ_CREDIT_BUREAU_QRT     41519
AMT_REQ_CREDIT_BUREAU_MON     41519
AMT_REQ_CREDIT_BUREAU_WEEK    41519
AMT_REQ_CREDIT_BUREAU_DAY     41519
AMT_REQ_CREDIT_BUREAU_HOUR    41519
```

```
NAME_TYPE_SUITE                    1292
DEF_60_CNT_SOCIAL_CIRCLE           1021
OBS_30_CNT_SOCIAL_CIRCLE           1021
DEF_30_CNT_SOCIAL_CIRCLE           1021
OBS_60_CNT_SOCIAL_CIRCLE           1021
EXT_SOURCE_2                        660
AMT_GOODS_PRICE                     278
AMT_ANNUITY                          12
CNT_FAM_MEMBERS                       2
DAYS_LAST_PHONE_CHANGE                1
HOUR_APPR_PROCESS_START               0
FLAG_DOCUMENT_3                       0
ORGANIZATION_TYPE                     0
LIVE_CITY_NOT_WORK_CITY               0
REG_CITY_NOT_WORK_CITY                0
REG_CITY_NOT_LIVE_CITY                0
LIVE_REGION_NOT_WORK_REGION           0
REG_REGION_NOT_WORK_REGION            0
REG_REGION_NOT_LIVE_REGION            0
SK_ID_CURR                            0
WEEKDAY_APPR_PROCESS_START            0
NAME_FAMILY_STATUS                    0
NAME_CONTRACT_TYPE                    0
CODE_GENDER                           0
FLAG_OWN_REALTY                       0
CNT_CHILDREN                          0
AMT_INCOME_TOTAL                      0
AMT_CREDIT                            0
NAME_INCOME_TYPE                      0
NAME_EDUCATION_TYPE                   0
NAME_HOUSING_TYPE                     0
TARGET                                0
REGION_POPULATION_RELATIVE            0
DAYS_BIRTH                            0
DAYS_EMPLOYED                         0
DAYS_REGISTRATION                     0
DAYS_ID_PUBLISH                       0
FLAG_MOBIL                            0
OCCUPATION_TYPE                       0
REGION_RATING_CLIENT                  0
REGION_RATING_CLIENT_W_CITY           0
dtype: int64
```

```
amt =
["AMT_REQ_CREDIT_BUREAU_YEAR","AMT_REQ_CREDIT_BUREAU_QRT","AMT_REQ_CRE
DIT_BUREAU_MON","AMT_REQ_CREDIT_BUREAU_WEEK",
"AMT_REQ_CREDIT_BUREAU_DAY","AMT_REQ_CREDIT_BUREAU_HOUR"]
adt[amt].describe()
```

```
       AMT_REQ_CREDIT_BUREAU_YEAR  AMT_REQ_CREDIT_BUREAU_QRT  \
count               265992.000000              265992.000000
mean                     1.899974                   0.265474
std                      1.869295                   0.794056
min                      0.000000                   0.000000
25%                      0.000000                   0.000000
50%                      1.000000                   0.000000
75%                      3.000000                   0.000000
max                     25.000000                 261.000000

       AMT_REQ_CREDIT_BUREAU_MON  AMT_REQ_CREDIT_BUREAU_WEEK  \
count              265992.000000               265992.000000
mean                    0.267395                    0.034362
std                     0.916002                    0.204685
min                     0.000000                    0.000000
25%                     0.000000                    0.000000
50%                     0.000000                    0.000000
75%                     0.000000                    0.000000
max                    27.000000                    8.000000

       AMT_REQ_CREDIT_BUREAU_DAY  AMT_REQ_CREDIT_BUREAU_HOUR
count              265992.000000               265992.000000
mean                    0.007000                    0.006402
std                     0.110757                    0.083849
min                     0.000000                    0.000000
25%                     0.000000                    0.000000
50%                     0.000000                    0.000000
75%                     0.000000                    0.000000
max                     9.000000                    4.000000
```

```
m = adt[amt].median()
adt.fillna(m, inplace=True)
(adt.isnull().sum()).sort_values(ascending = False)
```

```
EXT_SOURCE_3                     60965
NAME_TYPE_SUITE                   1292
OBS_30_CNT_SOCIAL_CIRCLE          1021
DEF_30_CNT_SOCIAL_CIRCLE          1021
DEF_60_CNT_SOCIAL_CIRCLE          1021
OBS_60_CNT_SOCIAL_CIRCLE          1021
EXT_SOURCE_2                       660
AMT_GOODS_PRICE                    278
AMT_ANNUITY                         12
CNT_FAM_MEMBERS                      2
DAYS_LAST_PHONE_CHANGE               1
REG_CITY_NOT_WORK_CITY               0
LIVE_CITY_NOT_WORK_CITY              0
REG_CITY_NOT_LIVE_CITY               0
ORGANIZATION_TYPE                    0
LIVE_REGION_NOT_WORK_REGION          0
```

```
REG_REGION_NOT_WORK_REGION          0
SK_ID_CURR                          0
HOUR_APPR_PROCESS_START             0
FLAG_DOCUMENT_3                     0
AMT_REQ_CREDIT_BUREAU_HOUR          0
AMT_REQ_CREDIT_BUREAU_DAY           0
AMT_REQ_CREDIT_BUREAU_WEEK          0
AMT_REQ_CREDIT_BUREAU_MON           0
AMT_REQ_CREDIT_BUREAU_QRT           0
REG_REGION_NOT_LIVE_REGION          0
REGION_RATING_CLIENT_W_CITY         0
WEEKDAY_APPR_PROCESS_START          0
NAME_FAMILY_STATUS                  0
NAME_CONTRACT_TYPE                  0
CODE_GENDER                         0
FLAG_OWN_REALTY                     0
CNT_CHILDREN                        0
AMT_INCOME_TOTAL                    0
AMT_CREDIT                          0
NAME_INCOME_TYPE                    0
NAME_EDUCATION_TYPE                 0
NAME_HOUSING_TYPE                   0
TARGET                              0
REGION_POPULATION_RELATIVE          0
DAYS_BIRTH                          0
DAYS_EMPLOYED                       0
DAYS_REGISTRATION                   0
DAYS_ID_PUBLISH                     0
FLAG_MOBIL                          0
OCCUPATION_TYPE                     0
REGION_RATING_CLIENT                0
AMT_REQ_CREDIT_BUREAU_YEAR          0
dtype: int64
```

```
adt['NAME_TYPE_SUITE'].value_counts()
```

```
NAME_TYPE_SUITE
Unaccompanied      248526
Family              40149
Spouse, partner     11370
Children             3267
Other_B              1770
Other_A               866
Group of people       271
Name: count, dtype: int64
```

```
adt['NAME_TYPE_SUITE'] =
adt['NAME_TYPE_SUITE'].fillna("Unaccompanied")
adt['NAME_TYPE_SUITE'].isnull().sum()
```

```
0

del adt['EXT_SOURCE_3']
del adt['EXT_SOURCE_2']

(adt.isnull().sum()).sort_values(ascending = False)

OBS_60_CNT_SOCIAL_CIRCLE       1021
DEF_60_CNT_SOCIAL_CIRCLE       1021
DEF_30_CNT_SOCIAL_CIRCLE       1021
OBS_30_CNT_SOCIAL_CIRCLE       1021
AMT_GOODS_PRICE                 278
AMT_ANNUITY                      12
CNT_FAM_MEMBERS                   2
DAYS_LAST_PHONE_CHANGE            1
ORGANIZATION_TYPE                 0
REG_REGION_NOT_LIVE_REGION        0
REG_REGION_NOT_WORK_REGION        0
LIVE_REGION_NOT_WORK_REGION       0
REG_CITY_NOT_LIVE_CITY            0
REG_CITY_NOT_WORK_CITY            0
LIVE_CITY_NOT_WORK_CITY           0
SK_ID_CURR                        0
WEEKDAY_APPR_PROCESS_START        0
FLAG_DOCUMENT_3                   0
AMT_REQ_CREDIT_BUREAU_HOUR        0
AMT_REQ_CREDIT_BUREAU_DAY         0
AMT_REQ_CREDIT_BUREAU_WEEK        0
AMT_REQ_CREDIT_BUREAU_MON         0
AMT_REQ_CREDIT_BUREAU_QRT         0
HOUR_APPR_PROCESS_START           0
REGION_RATING_CLIENT              0
REGION_RATING_CLIENT_W_CITY       0
NAME_EDUCATION_TYPE               0
NAME_CONTRACT_TYPE                0
CODE_GENDER                       0
FLAG_OWN_REALTY                   0
CNT_CHILDREN                      0
AMT_INCOME_TOTAL                  0
AMT_CREDIT                        0
NAME_TYPE_SUITE                   0
NAME_INCOME_TYPE                  0
NAME_FAMILY_STATUS                0
TARGET                            0
NAME_HOUSING_TYPE                 0
REGION_POPULATION_RELATIVE        0
DAYS_BIRTH                        0
DAYS_EMPLOYED                     0
DAYS_REGISTRATION                 0
DAYS_ID_PUBLISH                   0
```

```
FLAG_MOBIL                         0
OCCUPATION_TYPE                    0
AMT_REQ_CREDIT_BUREAU_YEAR         0
dtype: int64
```

```python
d=["DEF_60_CNT_SOCIAL_CIRCLE","OBS_60_CNT_SOCIAL_CIRCLE","DEF_30_CNT_SOCIAL_CIRCLE","OBS_30_CNT_SOCIAL_CIRCLE"]
adt[d].describe()
```

```
       DEF_60_CNT_SOCIAL_CIRCLE  OBS_60_CNT_SOCIAL_CIRCLE
count             306490.000000             306490.000000  \
mean                   0.100049                  1.405292
std                    0.362291                  2.379803
min                    0.000000                  0.000000
25%                    0.000000                  0.000000
50%                    0.000000                  0.000000
75%                    0.000000                  2.000000
max                   24.000000                344.000000

       DEF_30_CNT_SOCIAL_CIRCLE  OBS_30_CNT_SOCIAL_CIRCLE
count             306490.000000             306490.000000
mean                   0.143421                  1.422245
std                    0.446698                  2.400989
min                    0.000000                  0.000000
25%                    0.000000                  0.000000
50%                    0.000000                  0.000000
75%                    0.000000                  2.000000
max                   34.000000                348.000000
```

```python
adt.fillna(adt[d].median(), inplace=True)
```

```python
(adt.isnull().sum()).sort_values(ascending = False).head(10)
```

```
AMT_GOODS_PRICE                   278
AMT_ANNUITY                        12
CNT_FAM_MEMBERS                     2
DAYS_LAST_PHONE_CHANGE              1
SK_ID_CURR                          0
OBS_30_CNT_SOCIAL_CIRCLE            0
REG_REGION_NOT_LIVE_REGION          0
REG_REGION_NOT_WORK_REGION          0
LIVE_REGION_NOT_WORK_REGION         0
REG_CITY_NOT_LIVE_CITY              0
dtype: int64
```

```python
print(adt['DAYS_LAST_PHONE_CHANGE'].head(10))
adt['CNT_FAM_MEMBERS'].describe()
```

```
0   -1134.0
1    -828.0
2    -815.0
```

```
3    -617.0
4   -1106.0
5   -2536.0
6   -1562.0
7   -1070.0
8       0.0
9   -1673.0
Name: DAYS_LAST_PHONE_CHANGE, dtype: float64
```

```
count    307509.000000
mean          2.152665
std           0.910682
min           1.000000
25%           2.000000
50%           2.000000
75%           3.000000
max          20.000000
Name: CNT_FAM_MEMBERS, dtype: float64
```

```
adt['CNT_FAM_MEMBERS'].fillna(2,inplace=True)
adt['DAYS_LAST_PHONE_CHANGE'].dropna(inplace=True) # since it had only
one null value
adt['DAYS_LAST_PHONE_CHANGE'] = adt['DAYS_LAST_PHONE_CHANGE'].abs() #
to convert the values from negative to positive
```

```
adt['DAYS_LAST_PHONE_CHANGE'].describe()
```

```
count    307510.000000
mean        962.858788
std         826.808487
min           0.000000
25%         274.000000
50%         757.000000
75%        1570.000000
max        4292.000000
Name: DAYS_LAST_PHONE_CHANGE, dtype: float64
```

```
am = ["AMT_GOODS_PRICE","AMT_ANNUITY"]
adt[am].describe()
```

	AMT_GOODS_PRICE	AMT_ANNUITY
count	3.072330e+05	307499.000000
mean	5.383962e+05	27108.573909
std	3.694465e+05	14493.737315
min	4.050000e+04	1615.500000
25%	2.385000e+05	16524.000000
50%	4.500000e+05	24903.000000
75%	6.795000e+05	34596.000000
max	4.050000e+06	258025.500000

```
adt.fillna(adt[am].median, inplace=True)
(adt.isnull().sum()).sort_values(ascending = False).head(10)
```

```
SK_ID_CURR                        0
OBS_30_CNT_SOCIAL_CIRCLE          0
WEEKDAY_APPR_PROCESS_START        0
HOUR_APPR_PROCESS_START           0
REG_REGION_NOT_LIVE_REGION        0
REG_REGION_NOT_WORK_REGION        0
LIVE_REGION_NOT_WORK_REGION       0
REG_CITY_NOT_LIVE_CITY            0
REG_CITY_NOT_WORK_CITY            0
LIVE_CITY_NOT_WORK_CITY           0
dtype: int64
```

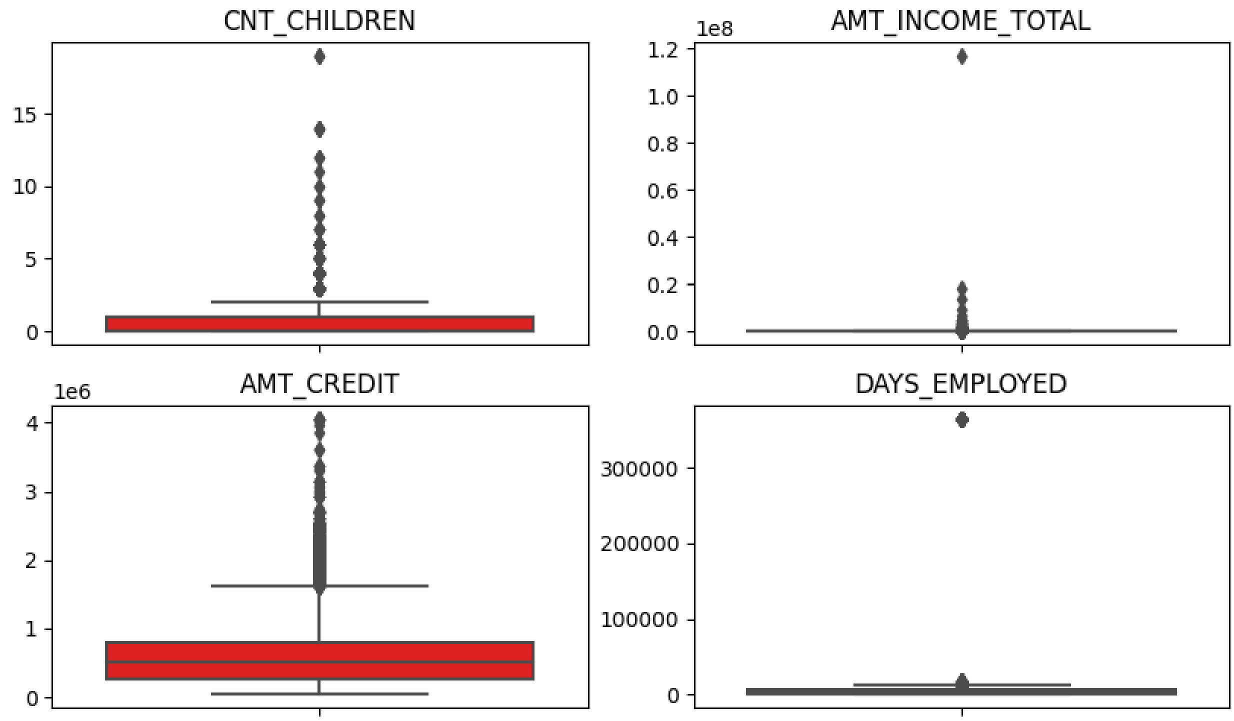
```
#  There are some column which are still left with negative values
l =
['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH']
adt[l] = adt[l].abs()
```

SHOWCASING OUTLIERS

```
outlier = ["CNT_CHILDREN","AMT_INCOME_TOTAL", "AMT_CREDIT",
"DAYS_EMPLOYED"]
plt.figure(figsize=(10,15))
for i,j in itertools.zip_longest(outlier, range(len(outlier))):
    plt.subplot(5,2,j+1)
    sns.boxplot(y = adt[i], orient = "h", color = "red")
    plt.xlabel("")
    plt.ylabel("")
    plt.title(i)
```

```
C:\Users\91852\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\seaborn\_oldcore.py:1592:
UserWarning: Horizontal orientation ignored with only `y` specified.
  warnings.warn(single_var_warning.format("Horizontal", "y"))
C:\Users\91852\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\seaborn\_oldcore.py:1592:
UserWarning: Horizontal orientation ignored with only `y` specified.
  warnings.warn(single_var_warning.format("Horizontal", "y"))
C:\Users\91852\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\seaborn\_oldcore.py:1592:
UserWarning: Horizontal orientation ignored with only `y` specified.
  warnings.warn(single_var_warning.format("Horizontal", "y"))
C:\Users\91852\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\seaborn\_oldcore.py:1592:
```

```
UserWarning: Horizontal orientation ignored with only `y` specified.
  warnings.warn(single_var_warning.format("Horizontal", "y"))
```




```
adt['DAYS_EMPLOYED'].describe()
```

```
count    307511.000000
mean      67724.742149
std      139443.751806
min           0.000000
25%         933.000000
50%        2219.000000
75%        5707.000000
max      365243.000000
Name: DAYS_EMPLOYED, dtype: float64
```

We have no more null values anymore and the Dataset has been cleaned

SAVING THE CLEANED DATASET ON SYSTEM

```
adt.to_csv("cleaned_application_data.csv")
```

Now we will clean the previous_applications dataset

```
prev = pd.read_csv(r'D:\Trainity Projects\Project 5\
previous_application.csv')
prev.shape
```

```
(1670214, 37)
```

```
(prev.isnull().sum()/len(prev)*100).sort_values(ascending=False)
```

```
RATE_INTEREST_PRIVILEGED       99.643698
RATE_INTEREST_PRIMARY          99.643698
AMT_DOWN_PAYMENT               53.636480
RATE_DOWN_PAYMENT              53.636480
NAME_TYPE_SUITE                49.119754
NFLAG_INSURED_ON_APPROVAL      40.298129
DAYS_TERMINATION               40.298129
DAYS_LAST_DUE                  40.298129
DAYS_LAST_DUE_1ST_VERSION      40.298129
DAYS_FIRST_DUE                 40.298129
DAYS_FIRST_DRAWING             40.298129
AMT_GOODS_PRICE                23.081773
AMT_ANNUITY                    22.286665
CNT_PAYMENT                    22.286366
PRODUCT_COMBINATION             0.020716
AMT_CREDIT                      0.000060
NAME_YIELD_GROUP                0.000000
NAME_PORTFOLIO                  0.000000
NAME_SELLER_INDUSTRY            0.000000
SELLERPLACE_AREA                0.000000
CHANNEL_TYPE                    0.000000
NAME_PRODUCT_TYPE               0.000000
SK_ID_PREV                      0.000000
NAME_GOODS_CATEGORY             0.000000
NAME_CLIENT_TYPE                0.000000
CODE_REJECT_REASON              0.000000
SK_ID_CURR                      0.000000
DAYS_DECISION                   0.000000
NAME_CONTRACT_STATUS            0.000000
NAME_CASH_LOAN_PURPOSE          0.000000
NFLAG_LAST_APPL_IN_DAY          0.000000
FLAG_LAST_APPL_PER_CONTRACT     0.000000
HOUR_APPR_PROCESS_START         0.000000
WEEKDAY_APPR_PROCESS_START      0.000000
AMT_APPLICATION                 0.000000
NAME_CONTRACT_TYPE              0.000000
NAME_PAYMENT_TYPE               0.000000
dtype: float64
```

```
l =
['RATE_INTEREST_PRIVILEGED','RATE_INTEREST_PRIMARY','AMT_DOWN_PAYMENT'
,'RATE_DOWN_PAYMENT','NAME_TYPE_SUITE','NFLAG_INSURED_ON_APPROVAL','DA
YS_TERMINATION','DAYS_LAST_DUE','DAYS_FIRST_DUE','DAYS_LAST_DUE_1ST_VE
RSION','DAYS_FIRST_DRAWING']
prev_clean = prev.drop(columns=l)
prev_clean.shape
```

```
(1670214, 26)

Unnecessary_col =
['WEEKDAY_APPR_PROCESS_START','HOUR_APPR_PROCESS_START','FLAG_LAST_APP
L_PER_CONTRACT', 'NFLAG_LAST_APPL_IN_DAY']
prev_clean = prev_clean.drop(columns=Unnecessary_col)
prev_clean.shape

(1670214, 22)
```

DEALING WITH MISSING/NULL VALUES

```
prev_clean.head(10)

   SK_ID_PREV  SK_ID_CURR NAME_CONTRACT_TYPE  AMT_ANNUITY
AMT_APPLICATION
0     2030495      271877     Consumer loans     1730.430
17145.0  \
1     2802425      108129         Cash loans    25188.615
607500.0
2     2523466      122040         Cash loans    15060.735
112500.0
3     2819243      176158         Cash loans    47041.335
450000.0
4     1784265      202054         Cash loans    31924.395
337500.0
5     1383531      199383         Cash loans    23703.930
315000.0
6     2315218      175704         Cash loans          NaN
0.0
7     1656711      296299         Cash loans          NaN
0.0
8     2367563      342292         Cash loans          NaN
0.0
9     2579447      334349         Cash loans          NaN
0.0

   AMT_CREDIT  AMT_GOODS_PRICE NAME_CASH_LOAN_PURPOSE
NAME_CONTRACT_STATUS
0     17145.0          17145.0                    XAP
Approved  \
1    679671.0         607500.0                    XNA
Approved
2    136444.5         112500.0                    XNA
Approved
3    470790.0         450000.0                    XNA
Approved
4    404055.0         337500.0                Repairs
Refused
5    340573.5         315000.0     Everyday expenses
```

```
Approved
6         0.0             NaN                      XNA
Canceled
7         0.0             NaN                      XNA
Canceled
8         0.0             NaN                      XNA
Canceled
9         0.0             NaN                      XNA
Canceled

   DAYS_DECISION  ... NAME_CLIENT_TYPE NAME_GOODS_CATEGORY
NAME_PORTFOLIO
0            -73  ...         Repeater              Mobile
POS  \
1           -164  ...         Repeater                 XNA
Cash
2           -301  ...         Repeater                 XNA
Cash
3           -512  ...         Repeater                 XNA
Cash
4           -781  ...         Repeater                 XNA
Cash
5           -684  ...         Repeater                 XNA
Cash
6            -14  ...         Repeater                 XNA
XNA
7            -21  ...         Repeater                 XNA
XNA
8           -386  ...         Repeater                 XNA
XNA
9            -57  ...         Repeater                 XNA
XNA

  NAME_PRODUCT_TYPE             CHANNEL_TYPE SELLERPLACE_AREA
0               XNA             Country-wide               35  \
1            x-sell           Contact center               -1
2            x-sell  Credit and cash offices               -1
3            x-sell  Credit and cash offices               -1
4           walk-in  Credit and cash offices               -1
5            x-sell  Credit and cash offices               -1
6               XNA  Credit and cash offices               -1
7               XNA  Credit and cash offices               -1
8               XNA  Credit and cash offices               -1
9               XNA  Credit and cash offices               -1

  NAME_SELLER_INDUSTRY  CNT_PAYMENT NAME_YIELD_GROUP
PRODUCT_COMBINATION
0         Connectivity         12.0           middle  POS mobile with
interest
1                  XNA         36.0       low_action          Cash X-
```

```
Sell: low
2                   XNA          12.0                 high         Cash X-
Sell: high
3                   XNA          12.0               middle       Cash X-
Sell: middle
4                   XNA          24.0                 high         Cash
Street: high
5                   XNA          18.0           low_normal            Cash X-
Sell: low
6                   XNA           NaN                  XNA
Cash
7                   XNA           NaN                  XNA
Cash
8                   XNA           NaN                  XNA
Cash
9                   XNA           NaN                  XNA
Cash

[10 rows x 22 columns]
```

```
prev_clean['DAYS_DECISION'] = prev['DAYS_DECISION'].abs()
```

```
prev_clean['DAYS_DECISION'].head()
```

```
0     73
1    164
2    301
3    512
4    781
Name: DAYS_DECISION, dtype: int64
```

```
(prev_clean.isnull().sum()/
len(prev_clean)*100).sort_values(ascending=False)
```

```
AMT_GOODS_PRICE           23.081773
AMT_ANNUITY               22.286665
CNT_PAYMENT               22.286366
PRODUCT_COMBINATION        0.020716
AMT_CREDIT                 0.000060
NAME_GOODS_CATEGORY        0.000000
NAME_YIELD_GROUP           0.000000
NAME_SELLER_INDUSTRY       0.000000
SELLERPLACE_AREA           0.000000
CHANNEL_TYPE               0.000000
NAME_PRODUCT_TYPE          0.000000
NAME_PORTFOLIO             0.000000
SK_ID_PREV                 0.000000
NAME_CLIENT_TYPE           0.000000
SK_ID_CURR                 0.000000
NAME_PAYMENT_TYPE          0.000000
```

```
DAYS_DECISION                  0.000000
NAME_CONTRACT_STATUS           0.000000
NAME_CASH_LOAN_PURPOSE         0.000000
AMT_APPLICATION                0.000000
NAME_CONTRACT_TYPE             0.000000
CODE_REJECT_REASON             0.000000
dtype: float64
```

```
prev_clean['AMT_CREDIT'].describe()
```

```
count    1.670213e+06
mean     1.961140e+05
std      3.185746e+05
min      0.000000e+00
25%      2.416050e+04
50%      8.054100e+04
75%      2.164185e+05
max      6.905160e+06
Name: AMT_CREDIT, dtype: float64
```

```
prev_clean['AMT_CREDIT'].mode()
```

```
0    0.0
Name: AMT_CREDIT, dtype: float64
```

```
prev_clean['AMT_CREDIT'].fillna(0.0,inplace=True)
```

```
prev_clean['PRODUCT_COMBINATION'].head()
prev_clean['PRODUCT_COMBINATION'].describe()
```

```
count     1669868
unique         17
top          Cash
freq       285990
Name: PRODUCT_COMBINATION, dtype: object
```

```
prev_clean['PRODUCT_COMBINATION'].fillna('Cash',inplace=True)
```

```
prev_clean['CNT_PAYMENT'].head(10)
```

```
0    12.0
1    36.0
2    12.0
3    12.0
4    24.0
5    18.0
6     NaN
7     NaN
8     NaN
9     NaN
Name: CNT_PAYMENT, dtype: float64
```

```
prev_clean['CNT_PAYMENT'].fillna(prev_clean['CNT_PAYMENT'].median(),inplace=True)
```

```
# prev_clean['AMT_ANNUITY'].describe()
print(prev_clean['AMT_ANNUITY'].mode())
print(prev_clean['AMT_ANNUITY'].mean())
print(prev_clean['AMT_ANNUITY'].median())
```

```
0    2250.0
Name: AMT_ANNUITY, dtype: float64
15955.120659452119
11250.0
```

```
prev_clean['AMT_ANNUITY'].fillna(prev_clean['AMT_ANNUITY'].median(),inplace=True)
```

```
# prev_clean['AMT_GOODS_PRICE'].describe()
print(prev_clean['AMT_GOODS_PRICE'].mode())
print(prev_clean['AMT_GOODS_PRICE'].mean())
print(prev_clean['AMT_GOODS_PRICE'].median())
```

```
0    45000.0
Name: AMT_GOODS_PRICE, dtype: float64
227847.27928334562
112320.0
```

```
prev_clean['AMT_GOODS_PRICE'].fillna(prev_clean['AMT_GOODS_PRICE'].median(),inplace=True)
```

```
(prev_clean.isnull().sum()/
len(prev_clean)*100).sort_values(ascending=False)
```

```
SK_ID_PREV                0.0
SK_ID_CURR                0.0
NAME_YIELD_GROUP          0.0
CNT_PAYMENT               0.0
NAME_SELLER_INDUSTRY      0.0
SELLERPLACE_AREA          0.0
CHANNEL_TYPE              0.0
NAME_PRODUCT_TYPE         0.0
NAME_PORTFOLIO            0.0
NAME_GOODS_CATEGORY       0.0
NAME_CLIENT_TYPE          0.0
CODE_REJECT_REASON        0.0
NAME_PAYMENT_TYPE         0.0
DAYS_DECISION             0.0
NAME_CONTRACT_STATUS      0.0
NAME_CASH_LOAN_PURPOSE    0.0
AMT_GOODS_PRICE           0.0
AMT_CREDIT                0.0
```

```
AMT_APPLICATION            0.0
AMT_ANNUITY                0.0
NAME_CONTRACT_TYPE         0.0
PRODUCT_COMBINATION        0.0
dtype: float64
```

SAVING ON LOCAL SYSTEM

```
prev_clean.to_csv('clean_prev_application')
```

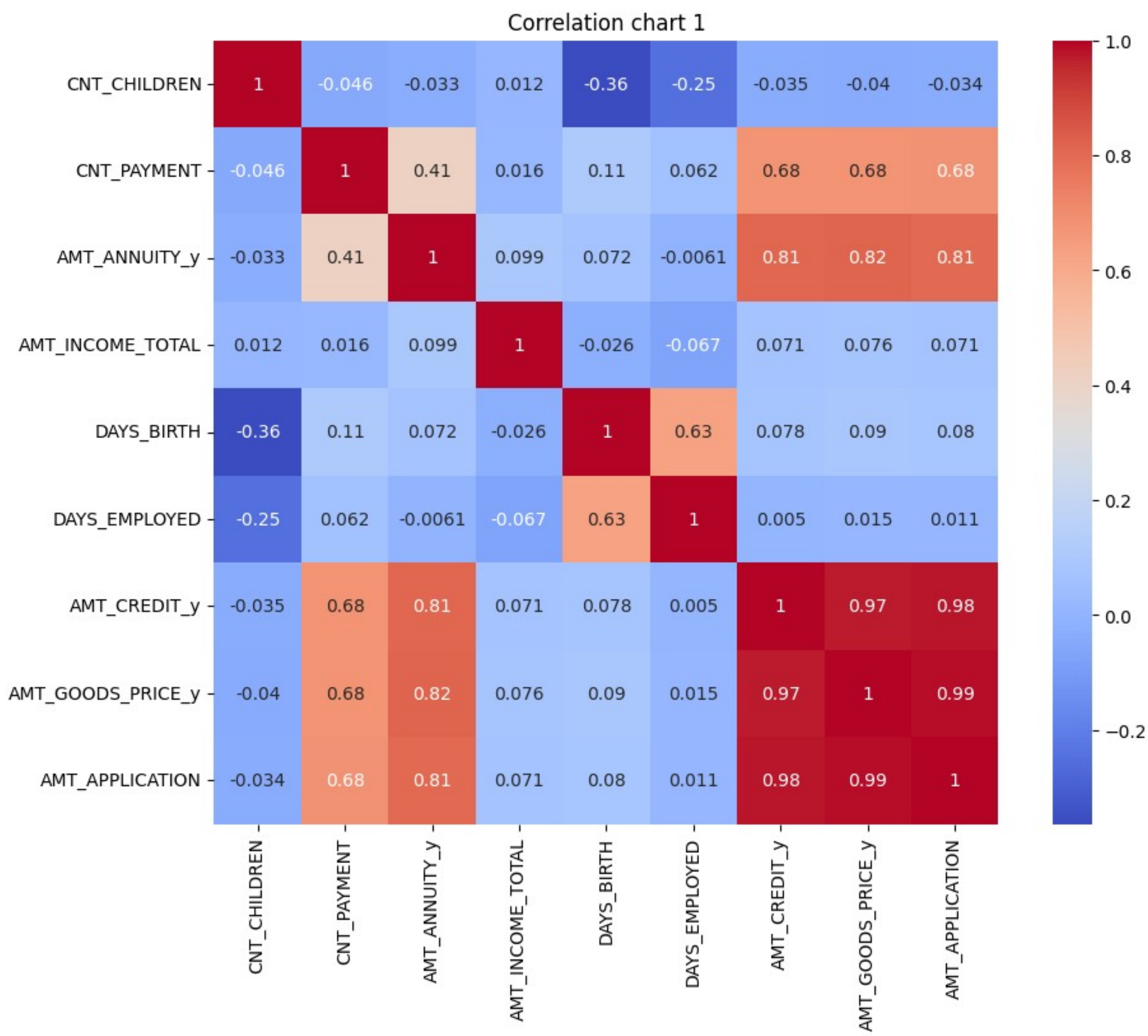
MERGING THE DATSETS

```
app_clean = pd.read_csv(r'cleaned_application_data.csv')

C:\Users\91852\AppData\Local\Temp\ipykernel_15092\3820721257.py:1:
DtypeWarning: Columns (9,39) have mixed types. Specify dtype option on
import or set low_memory=False.
  app_clean = pd.read_csv(r'cleaned_application_data.csv')

merged_data =
pd.merge(app_clean,prev_clean,how='inner',on='SK_ID_CURR')
merged_data.head()

   Unnamed: 0  SK_ID_CURR  TARGET NAME_CONTRACT_TYPE_x CODE_GENDER
0           0      100002       1           Cash loans           M  \
1           1      100003       0           Cash loans           F
2           1      100003       0           Cash loans           F
3           1      100003       0           Cash loans           F
4           2      100004       0      Revolving loans           M

  FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT_x
AMT_ANNUITY_x
0               Y             0          202500.0      406597.5
24700.5  \
1               N             0          270000.0     1293502.5
35698.5
2               N             0          270000.0     1293502.5
35698.5
3               N             0          270000.0     1293502.5
35698.5
4               Y             0           67500.0      135000.0
6750.0

   ... NAME_CLIENT_TYPE   NAME_GOODS_CATEGORY NAME_PORTFOLIO
0  ...              New              Vehicles            POS  \
1  ...         Repeater                   XNA           Cash
2  ...        Refreshed             Furniture            POS
3  ...        Refreshed  Consumer Electronics            POS
4  ...              New                Mobile            POS
```

```
  NAME_PRODUCT_TYPE              CHANNEL_TYPE SELLERPLACE_AREA
0               XNA                     Stone              500  \
1            x-sell  Credit and cash offices                -1
2               XNA                     Stone             1400
3               XNA              Country-wide              200
4               XNA          Regional / Local               30

   NAME_SELLER_INDUSTRY  CNT_PAYMENT  NAME_YIELD_GROUP
0       Auto technology         24.0        low_normal  \
1                   XNA         12.0        low_normal
2             Furniture          6.0            middle
3  Consumer electronics         12.0            middle
4          Connectivity          4.0            middle

           PRODUCT_COMBINATION
0      POS other with interest
1             Cash X-Sell: low
2   POS industry with interest
3  POS household with interest
4  POS mobile without interest

[5 rows x 68 columns]

# merged_data.to_csv('merged_data.csv')
```

CORRELATION CHART

```
col =
['CNT_CHILDREN','CNT_PAYMENT','AMT_ANNUITY_y','AMT_INCOME_TOTAL','DAYS
_BIRTH','DAYS_EMPLOYED','AMT_CREDIT_y','AMT_GOODS_PRICE_y','AMT_APPLIC
ATION']

correlation = merged_data[col].corr()
plt.figure(figsize=(11,8))
plt.title("Correlation chart 1")
sns.heatmap(correlation,annot=True,cmap='coolwarm',square=True)
plt.show()
```

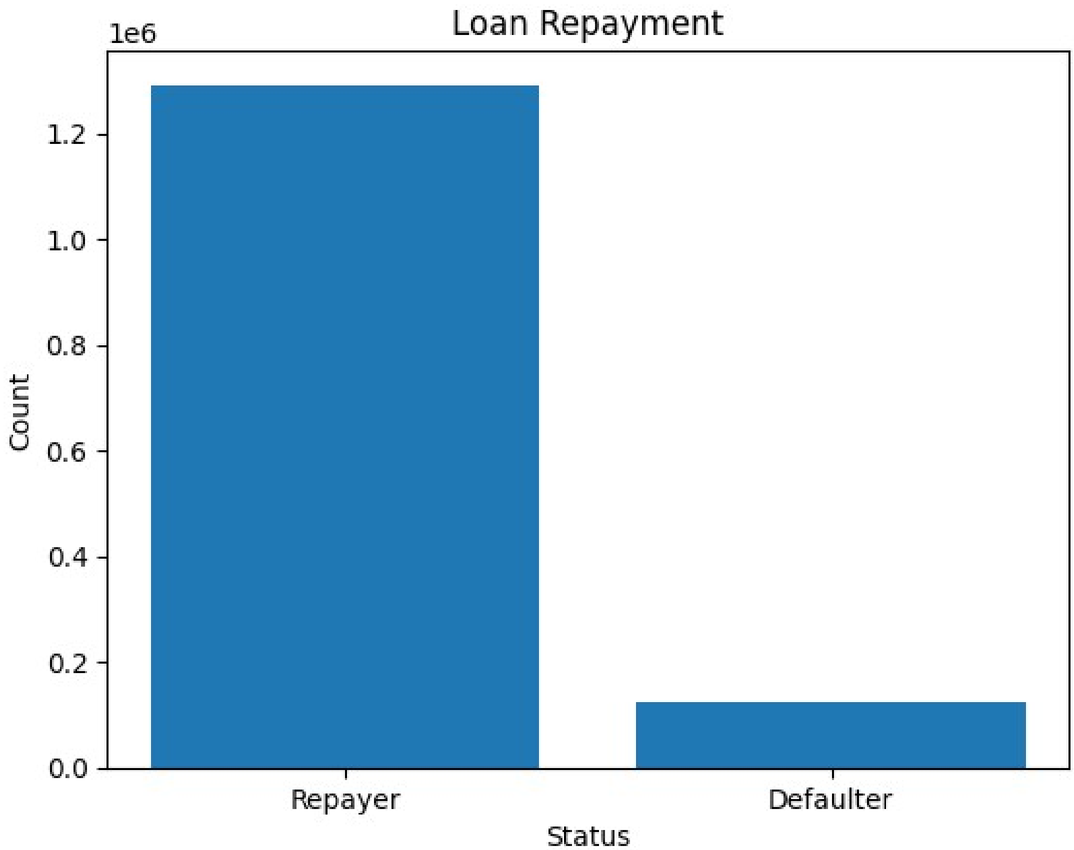


RATIO ON IMBALANCE

```
count = merged_data['TARGET'].value_counts()
ratio = count[1]/count[0]
print("Imbalance ratio:%.5f"%(ratio))
```

```
Imbalance ratio:0.09475
```

```
lt = [count[0],count[1]]
x = ['Repayer','Defaulter']
plt.title('Loan Repayment')
plt.xlabel('Status')
plt.ylabel('Count')
plt.bar(x,lt)
plt.show()
```


Loan Repayment
1e6
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Count
Repayer
Defaulter
Status