In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

#reading the data of application_data
data_c = pd.read_csv('C:/Intellipaat/ThinkEvolve/Credit EDA Case Study-20220210T0526
data_c
```

Out[1]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100004	0	Revolving loans	M	Y	
3	100006	0	Cash loans	F	N	
4	100007	0	Cash loans	M	N	
...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	
307507	456252	0	Cash loans	F	N	
307508	456253	0	Cash loans	F	N	
307509	456254	1	Cash loans	F	N	
307510	456255	0	Cash loans	F	N	

307511 rows × 122 columns

In [2]:
```
#information of the dataset
data_c.info(verbose=True)
```

```
<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
```

In [3]:

```
#to get the description of the numeric data
data_c.describe()
```

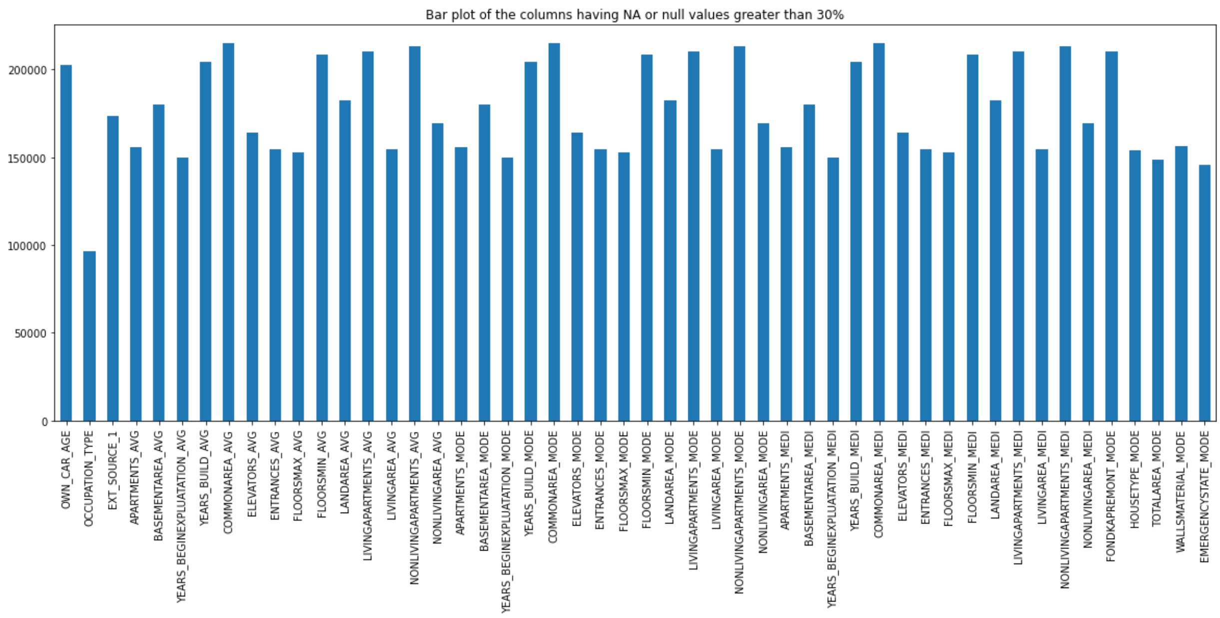
Out[3]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANI
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	307499.(
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	27108.5
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	14493.7
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1615.5
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16524.(
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	24903.(
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.(
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	258025.5

8 rows × 106 columns

In [4]:

```
#check for missing values and plot graph for the null value columns where null value
data_na = data_c.isna().sum()
data_na = data_na[data_na.values > (0.3 * len(data_c))]
plt.figure(figsize=(20,7))
data_na.plot(kind='bar')
plt.title('Bar plot of the columns having NA or null values greater than 30%')
plt.show()
```



As it is difficult and time consuming to look at each and every column of the dataset for null or NA values, here it is mentioned that display the columns having null values greater than 30%, so that we can ignore these columns.

So, in the above graph there are more than 200000 null values in 50 columns, hence, we will remove those columns as it is not feasible to impute missing values for those columns.

In [5]:

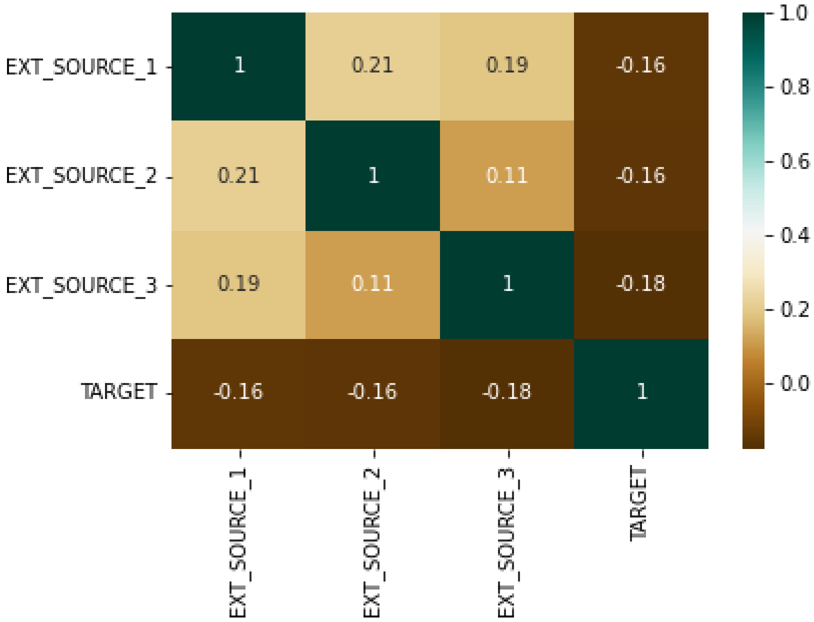
```
#to get the column names having null values more than 30%
print(data_na)
print('\nNo. of columns in the dataset having null values more than 30% = ',len(data
```

```
OWN_CAR_AGE                     202929
OCCUPATION_TYPE                  96391
EXT_SOURCE_1                    173378
APARTMENTS_AVG                  156061
BASEMENTAREA_AVG                179943
YEARS_BEGINEXPLUATATION_AVG     150007
YEARS_BUILD_AVG                 204488
COMMONAREA_AVG                  214865
ELEVATORS_AVG                   163891
ENTRANCES_AVG                   154828
FLOORSMAX_AVG                   153020
FLOORSMIN_AVG                   208642
LANDAREA_AVG                    182590
LIVINGAPARTMENTS_AVG            210199
LIVINGAREA_AVG                  154350
NONLIVINGAPARTMENTS_AVG         213514
NONLIVINGAREA_AVG               169682
APARTMENTS_MODE                 156061
BASEMENTAREA_MODE               179943
```

```
YEARS_BEGINEXPLUATATION_MODE    150007
YEARS_BUILD_MODE                204488
COMMONAREA_MODE                 214865
ELEVATORS_MODE                  163891
ENTRANCES_MODE                  154828
FLOORSMAX_MODE                  153020
FLOORSMIN_MODE                  208642
LANDAREA_MODE                   182590
LIVINGAPARTMENTS_MODE           210199
LIVINGAREA_MODE                 154350
NONLIVINGAPARTMENTS_MODE        213514
NONLIVINGAREA_MODE              169682
APARTMENTS_MEDI                 156061
BASEMENTAREA_MEDI               179943
YEARS_BEGINEXPLUATATION_MEDI    150007
YEARS_BUILD_MEDI                204488
COMMONAREA_MEDI                 214865
ELEVATORS_MEDI                  163891
ENTRANCES_MEDI                  154828
FLOORSMAX_MEDI                  153020
FLOORSMIN_MEDI                  208642
LANDAREA_MEDI                   182590
LIVINGAPARTMENTS_MEDI           210199
LIVINGAREA_MEDI                 154350
NONLIVINGAPARTMENTS_MEDI        213514
NONLIVINGAREA_MEDI              169682
FONDKAPREMONT_MODE              210295
HOUSETYPE_MODE                  154297
TOTALAREA_MODE                  148431
WALLSMATERIAL_MODE              156341
EMERGENCYSTATE_MODE             145755
dtype: int64

No. of columns in the dataset having null values more than 30% =  50 columns
```

In [6]:

```
#to check the corelation between EXT_SOURCE_1, EXT_SOURCE_2,EXT_SOURCE_3 and the Tar
#so if there is no such corelation then we can drop the column if it has null values

#get the required coulumns and store in one variable
data_tar = data_c[['EXT_SOURCE_1','EXT_SOURCE_2','EXT_SOURCE_3','TARGET']]
data_cor = data_tar.corr()
cor = sns.heatmap(data_cor, xticklabels=data_cor.columns, yticklabels=data_cor.colum
```



From the above plot we can see that there is no corelation between External sources and Target variables. So these columns of External sources can be dropped from the dataset.

In [7]:

```
#creating a function to remove columns and rows having null values greater than 30%
def removeNulls(dataframe, axis =1, percent=0.3):
    df = dataframe.copy()
    ishape = df.shape
    if axis == 0:
        rownames = df.transpose().isnull().sum()
        rownames = list(rownames[rownames.values > percent*len(df)].index)
        df.drop(df.index[rownames],inplace=True)
        print("\nNumber of Rows dropped\t: ",len(rownames))
    else:
        colnames = (df.isnull().sum()/len(df))
        colnames = list(colnames[colnames.values>=percent].index)
        df.drop(labels = colnames,axis =1,inplace=True)
        print("Number of Columns dropped\t: ",len(colnames))

    print("\nOld dataset rows,columns",ishape,"\nNew dataset rows,columns",df.shape)

    return df
```

In [8]:

```
#remove columns having nuull values greater than 30%
data_null = removeNulls(data_c, axis = 1, percent = 0.3)
# data_null
```

```
Number of Columns dropped	:  50

Old dataset rows,columns (307511, 122)
New dataset rows,columns (307511, 72)
```

There are 50 columns that were dropped that had null values more than 30%

In [9]:

```
#remove rows having nuull values greater than 30%
data_null = removeNulls(data_null, axis = 0, percent = 0.3)
```

```
Number of Rows dropped	:  0

Old dataset rows,columns (307511, 72)
New dataset rows,columns (307511, 72)
```

There are no rows with null values greater than 30%

In [10]:

```
#to get the column names after removing null values from the dataset
data_null.columns
```

Out[10]:

```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'FLAG_MOBIL',
       'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE',
       'FLAG_EMAIL', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START',
       'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION',
       'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION',
       'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY',
       'LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_TYPE', 'EXT_SOURCE_2',
       'EXT_SOURCE_3', 'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
```

```
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', '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',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')
```

In [11]:

```
#to remove other columns of External sources columns
data_null.drop(['EXT_SOURCE_2','EXT_SOURCE_3'],axis=1, inplace=True)
data_null.columns
```

Out[11]:

```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'FLAG_MOBIL',
       'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE',
       'FLAG_EMAIL', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START',
       'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION',
       'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION',
       'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY',
       'LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_TYPE',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', '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',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')
```

In [12]:

```
data_tar = data_null['TARGET'].replace({1:'Non-repayer', 0:'Repayer'})
data_tar = pd.DataFrame(data_tar)
data_tar
```

Out[12]:

	TARGET
0	Non-repayer
1	Repayer
2	Repayer
3	Repayer
4	Repayer
...	...
307506	Repayer

	TARGET
307507	Repayer
307508	Repayer
307509	Non-repayer
307510	Repayer

307511 rows × 1 columns

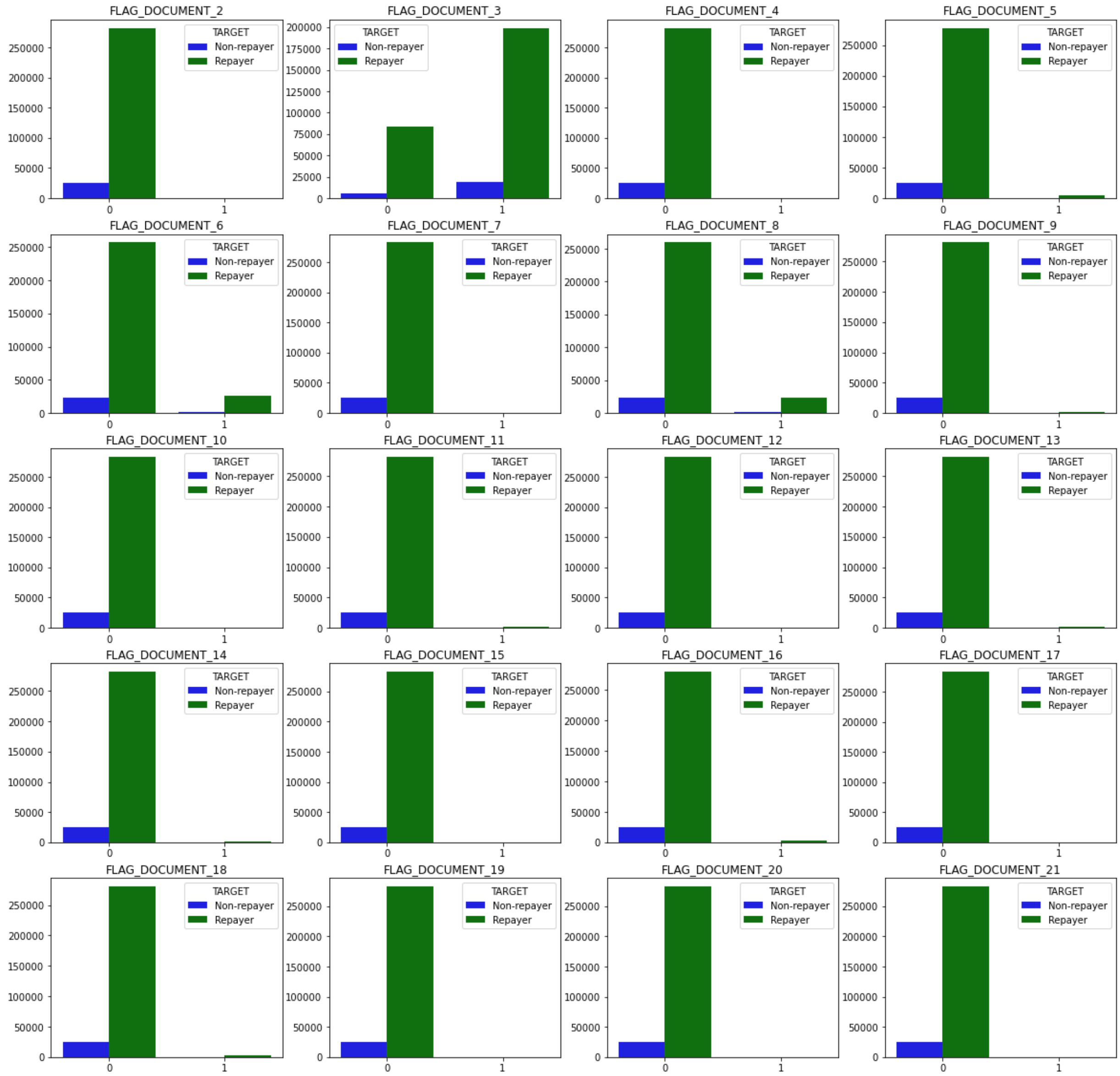
In [13]:

```
#to check the importance of the Flag document with respect to Target variable

#to store the Flag document variables in data_flag
data_flag = ['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']


data_tar = data_c[data_flag+['TARGET']]

# df = pd.DataFrame(data_tar, columns = ['column_name'])

#to change the values in Target columns as 1 = Non-repayer, 0 = repayer
data_tar['TARGET'] = data_tar['TARGET'].replace({1:'Non-repayer', 0:'Repayer'})

#plotting the graph
plt.figure(figsize=(20,20))

import itertools
for i,j in itertools.zip_longest(data_flag,range(len(data_flag))):
    plt.subplot(5,4,j+1)
    ax = sns.countplot(data_tar[i],hue=data_tar["TARGET"],palette=["b","g"])
    plt.yticks(fontsize=10)
    plt.xlabel("")
    plt.ylabel("")
    plt.title(i)
```

```
C:\Users\priyanka\AppData\Local\Temp/ipykernel_15992/367016558.py:18: SettingWithCop
yWarning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  data_tar['TARGET'] = data_tar['TARGET'].replace({1:'Non-repayer', 0:'Repayer'})
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
```

```
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
```



From the above plots we can see that majority of the documents are not submitted by the people/clients who have taken loans. Only, incase of Document 3 clients have submitted the documents. So, we can remove all the other document columns except for Document 3. Plot of document 3 describes that if clients submit documents then less chance of loan defaulters.

In [15]:

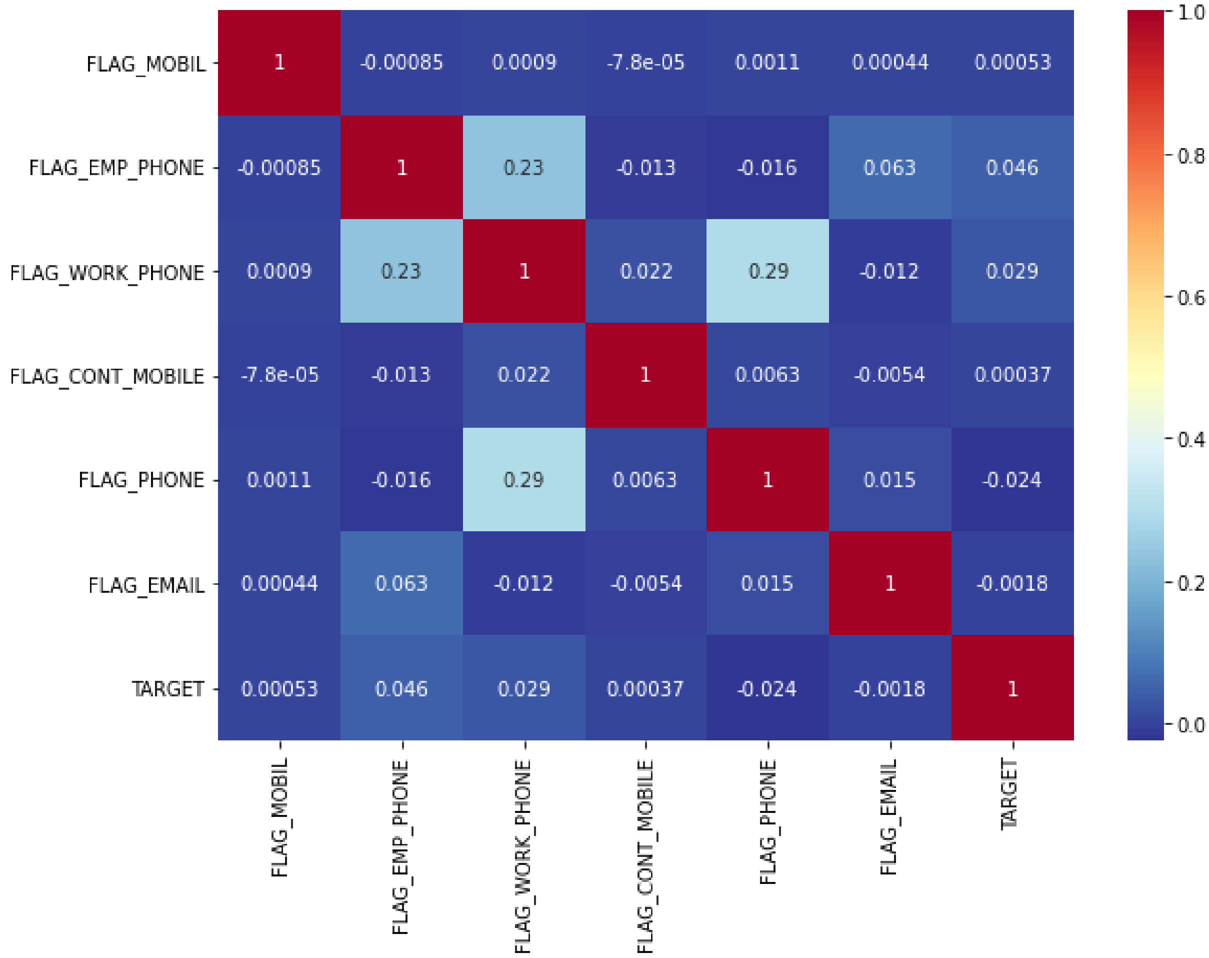
```
# dropping the FLAG_DOCUMENT columns except FLAG_DOCUMENT_3

data_null.drop(['FLAG_DOCUMENT_2',
       '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'], axis = 1, inplac

#columns after dropping FLAG_DOCUMENT
data_null.columns
```

Out[15]:

```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'FLAG_MOBIL',
       'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE',
       'FLAG_EMAIL', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START',
       'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION',
       'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION',
       'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY',
       'LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_TYPE',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_3',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')
```

In [16]:

```
#Now to find relationship between other columns like FLAG_MOBIL, FLAG_EMP_PHONE, FLA
#FLAG_PHONE, FLAG_EMAIL, TARGET column

#to store the relevant column in data_con variable
data_con = ['FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE',
       'FLAG_PHONE', 'FLAG_EMAIL','TARGET']

#to get the corelation of the mentioned columns
data_relation = data_c[data_con].corr()
data_relation

#to plot the heatmap
plt.figure(figsize=(10,7))
sns.heatmap(data_relation, xticklabels = data_relation.columns, yticklabels = data_r
            cmap ="RdYlBu_r")
plt.show()
```



From the above heat map we can mention that there is no corelation between the selected columns and the Target column (i.e. loan repayment). So, we can delete/drop these columns as well

In [17]:
```
#to drop the unwanted columns i.e. 'FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE'
#'FLAG_PHONE', 'FLAG_EMAIL'

data_null.drop(['FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE
        'FLAG_PHONE', 'FLAG_EMAIL'],axis = 1, inplace=True)

#columns after dropping the unwanted columns
data_null.columns
```

Out[17]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH',
       'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START',
       'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION',
       'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION',
       'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY',
       'LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_TYPE',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_3',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')
```

In [18]:

```
#final shape of the dataset after removing unwanted columns
data_null.shape
```

Out[18]:

```
(307511, 45)
```

In [19]:

```
#displying the information of columns in the dataset after removal of unwanted colum
data_null.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 307511 entries, 0 to 307510
Data columns (total 45 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   SK_ID_CURR                   307511 non-null  int64  
 1   TARGET                       307511 non-null  int64  
 2   NAME_CONTRACT_TYPE           307511 non-null  object 
 3   CODE_GENDER                  307511 non-null  object 
 4   FLAG_OWN_CAR                 307511 non-null  object 
 5   FLAG_OWN_REALTY              307511 non-null  object 
 6   CNT_CHILDREN                 307511 non-null  int64  
 7   AMT_INCOME_TOTAL             307511 non-null  float64
 8   AMT_CREDIT                   307511 non-null  float64
 9   AMT_ANNUITY                  307499 non-null  float64
 10  AMT_GOODS_PRICE              307233 non-null  float64
 11  NAME_TYPE_SUITE              306219 non-null  object 
 12  NAME_INCOME_TYPE             307511 non-null  object 
 13  NAME_EDUCATION_TYPE          307511 non-null  object 
 14  NAME_FAMILY_STATUS           307511 non-null  object 
 15  NAME_HOUSING_TYPE            307511 non-null  object 
 16  REGION_POPULATION_RELATIVE   307511 non-null  float64
 17  DAYS_BIRTH                   307511 non-null  int64  
 18  DAYS_EMPLOYED                307511 non-null  int64  
 19  DAYS_REGISTRATION            307511 non-null  float64
 20  DAYS_ID_PUBLISH              307511 non-null  int64  
 21  CNT_FAM_MEMBERS              307509 non-null  float64
 22  REGION_RATING_CLIENT         307511 non-null  int64  
 23  REGION_RATING_CLIENT_W_CITY  307511 non-null  int64  
 24  WEEKDAY_APPR_PROCESS_START   307511 non-null  object 
 25  HOUR_APPR_PROCESS_START      307511 non-null  int64  
 26  REG_REGION_NOT_LIVE_REGION   307511 non-null  int64  
 27  REG_REGION_NOT_WORK_REGION   307511 non-null  int64  
 28  LIVE_REGION_NOT_WORK_REGION  307511 non-null  int64  
 29  REG_CITY_NOT_LIVE_CITY       307511 non-null  int64  
 30  REG_CITY_NOT_WORK_CITY       307511 non-null  int64  
 31  LIVE_CITY_NOT_WORK_CITY      307511 non-null  int64  
 32  ORGANIZATION_TYPE            307511 non-null  object 
 33  OBS_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 34  DEF_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 35  OBS_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 36  DEF_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 37  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64
 38  FLAG_DOCUMENT_3              307511 non-null  int64  
 39  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64
 40  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64
 41  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64
 42  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64
 43  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64
 44  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64
dtypes: float64(18), int64(16), object(11)
memory usage: 107.9+ MB
```

In [20]:
```
#to get description of the dataset after data cleaning
data_null.describe()
```

Out[20]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANI
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	307499.(
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	27108.5
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	14493.7
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1615.5
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16524.(
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	24903.(
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.(
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	258025.5

8 rows × 34 columns

In [21]:
```
#to standardize values of the dataset

#to convert values of the dataset like value of dates --- these cannot be negative,
data_app_date = ['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH']


for col in data_app_date:
    data_null[col] = abs(data_null[col])
```

In [22]:
```
#to check for the percent of income amount of the loan applicants
#for this we will convert numerical value to categorical values by binning process

#look for maximum amount in AMT_INCOME_TOTAL column
data_null['AMT_INCOME_TOTAL'].max()
```

Out[22]:
```
117000000.0
```

In [23]:
```
#so we will divide the AMT_INCOME_TOTAL by 100000 to create bins
data_null['AMT_INCOME_TOTAL'] = data_null['AMT_INCOME_TOTAL']/100000

#to create 10 bins
data_bins = [0,1,2,3,4,5,6,7,8,9,10]

#creating binning slots
data_slot = ['0-100k','100K-200k', '200k-300k','300k-400k','400k-500k','500k-600k','
        '900k above']

#to create column INCOME_RANGE
data_null['INCOME_RANGE'] = pd.cut(data_null['AMT_INCOME_TOTAL'], data_bins, labels

#to get the count and percent values of the income range of the loan applicants
data_null['INCOME_RANGE'].value_counts(normalize=True)*100
```

Out[23]:
```
100K-200k     50.737972
200k-300k     21.211934
0-100k        20.730910
300k-400k      4.776395
```

```
400k-500k      1.744771
500k-600k      0.356375
600k-700k      0.282821
800k-900k      0.096986
700k-800k      0.052724
900k above     0.009113
Name: INCOME_RANGE, dtype: float64
```

From the above percent value we can make out that all the loan applicants are present in the income range of 0-100k

In [24]:

```
#in similar manner to create bins for credit amount --- to check what is the range o
#loan applicants

#so we will divide the AMT_CREDIT by 100000 to create bins
data_null['AMT_CREDIT'] = data_null['AMT_CREDIT']/100000

#creating 10 bins
data_bins = [0,1,2,3,4,5,6,7,8,9,10]

data_slots = ['0-100k','100K-200k', '200k-300k','300k-400k','400k-500k','500k-600k',
        '900k above']

#creating new column CREDIT_RANGE
data_null['CREDIT_RANGE'] = pd.cut(data_null['AMT_CREDIT'], data_bins, labels = data

#to get the count and percent values of the credit range of the loan applicants
data_null['CREDIT_RANGE'].value_counts(normalize=True)*100
```

Out[24]:

```
200k-300k     21.284453
500k-600k     13.292638
400k-500k     12.440686
100K-200k     11.703673
300k-400k     10.227317
600k-700k      9.338475
800k-900k      8.462058
700k-800k      7.452840
900k above     3.466446
0-100k         2.331415
Name: CREDIT_RANGE, dtype: float64
```

From the above value range we can see that there are more than 50% of loan applicants who have taken loan ranging 0-600k

In [25]:

```
#to check work experience of the people who are applying for loans
data_null['YEARS_EMPLOYED'] = data_null['DAYS_EMPLOYED'] // 365

#creating bins
data_bins = [0, 5, 10, 20, 30, 40, 50, 60, 150]

data_slots = ['0-5','5-10','10-20','20-30','30-40','40-50','50-60','60 above']

#create new column WORK_EXP
data_null['WORK_EXP'] = pd.cut(data_null['YEARS_EMPLOYED'], data_bins, labels = data

#to get the count and percent values of the work exp. of the loan applicants
data_null['WORK_EXP'].value_counts(normalize=True)*100
```

Out[25]:

```
0-5         55.582363
5-10        24.966441
10-20       14.564315
20-30        3.750117
```

```
30-40        1.058720
40-50        0.078044
50-60        0.000000
60 above     0.000000
Name: WORK_EXP, dtype: float64
```

From above values we can see that 50% of the loan applicants have work exp. of 0-5 yrs.

In [26]:

```
#to check for the age group of the loan applicants
data_null['AGE'] = data_null["DAYS_BIRTH"] // 365

#to create bins
data_bins = [0, 20, 30, 40, 50, 100]

data_slots = ['0-20','20-30','30-40','40-50','50 above']

#to create new column
data_null['AGE_RANGE'] = pd.cut(data_null['AGE'], data_bins, labels = data_slots)

#to get the count and percent values of the age range of the loan applicants
data_null['AGE_RANGE'].value_counts(normalize=True)*100
```

Out[26]:

```
50 above    31.604398
30-40       27.028952
40-50       24.194582
20-30       17.171743
0-20         0.000325
Name: AGE_RANGE, dtype: float64
```

From the above values we can see that more than 50% of loan applicants are in age range of 40 and above

In [27]:

```
#converting the non-categorical value columns to categorical values

#check the datatype of the dataset
data_null.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 307511 entries, 0 to 307510
Data columns (total 51 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   SK_ID_CURR                   307511 non-null  int64  
 1   TARGET                       307511 non-null  int64  
 2   NAME_CONTRACT_TYPE           307511 non-null  object 
 3   CODE_GENDER                  307511 non-null  object 
 4   FLAG_OWN_CAR                 307511 non-null  object 
 5   FLAG_OWN_REALTY              307511 non-null  object 
 6   CNT_CHILDREN                 307511 non-null  int64  
 7   AMT_INCOME_TOTAL             307511 non-null  float64
 8   AMT_CREDIT                   307511 non-null  float64
 9   AMT_ANNUITY                  307499 non-null  float64
 10  AMT_GOODS_PRICE              307233 non-null  float64
 11  NAME_TYPE_SUITE              306219 non-null  object 
 12  NAME_INCOME_TYPE             307511 non-null  object 
 13  NAME_EDUCATION_TYPE          307511 non-null  object 
 14  NAME_FAMILY_STATUS           307511 non-null  object 
 15  NAME_HOUSING_TYPE            307511 non-null  object 
 16  REGION_POPULATION_RELATIVE   307511 non-null  float64
 17  DAYS_BIRTH                   307511 non-null  int64  
 18  DAYS_EMPLOYED                307511 non-null  int64  
 19  DAYS_REGISTRATION            307511 non-null  float64
 20  DAYS_ID_PUBLISH              307511 non-null  int64  
```

```
 21  CNT_FAM_MEMBERS              307509 non-null  float64
 22  REGION_RATING_CLIENT         307511 non-null  int64
 23  REGION_RATING_CLIENT_W_CITY  307511 non-null  int64
 24  WEEKDAY_APPR_PROCESS_START   307511 non-null  object
 25  HOUR_APPR_PROCESS_START      307511 non-null  int64
 26  REG_REGION_NOT_LIVE_REGION   307511 non-null  int64
 27  REG_REGION_NOT_WORK_REGION   307511 non-null  int64
 28  LIVE_REGION_NOT_WORK_REGION  307511 non-null  int64
 29  REG_CITY_NOT_LIVE_CITY       307511 non-null  int64
 30  REG_CITY_NOT_WORK_CITY       307511 non-null  int64
 31  LIVE_CITY_NOT_WORK_CITY      307511 non-null  int64
 32  ORGANIZATION_TYPE            307511 non-null  object
 33  OBS_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 34  DEF_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 35  OBS_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 36  DEF_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 37  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64
 38  FLAG_DOCUMENT_3              307511 non-null  int64
 39  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64
 40  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64
 41  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64
 42  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64
 43  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64
 44  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64
 45  INCOME_RANGE                 307261 non-null  category
 46  CREDIT_RANGE                 257526 non-null  category
 47  YEARS_EMPLOYED               307511 non-null  int64
 48  WORK_EXP                     224233 non-null  category
 49  AGE                          307511 non-null  int64
 50  AGE_RANGE                    307511 non-null  category
dtypes: category(4), float64(18), int64(18), object(11)
memory usage: 113.8+ MB
```

From the above information there are few columns that has datatype as object. So converting these columns into categorical values

In [28]:

```
#categorical conversion
data_cat = ['NAME_CONTRACT_TYPE','CODE_GENDER','NAME_TYPE_SUITE','NAME_INCOME_TYPE',
            'NAME_FAMILY_STATUS','NAME_HOUSING_TYPE','ORGANIZATION_TYPE','WEEKDAY_APP
            'ORGANIZATION_TYPE','FLAG_OWN_CAR','FLAG_OWN_REALTY','LIVE_CITY_NOT_WORK_
            'REG_CITY_NOT_LIVE_CITY','REG_CITY_NOT_WORK_CITY','REG_REGION_NOT_WORK_RE
            'LIVE_REGION_NOT_WORK_REGION','REGION_RATING_CLIENT','WEEKDAY_APPR_PROCES
            'REGION_RATING_CLIENT_W_CITY']

#using for loop for conversion
for cat in data_cat:
    data_null[cat] = pd.Categorical(data_null[cat])
```

In [29]:

```
#to check the data type of the data after conversion
data_null.info()
```

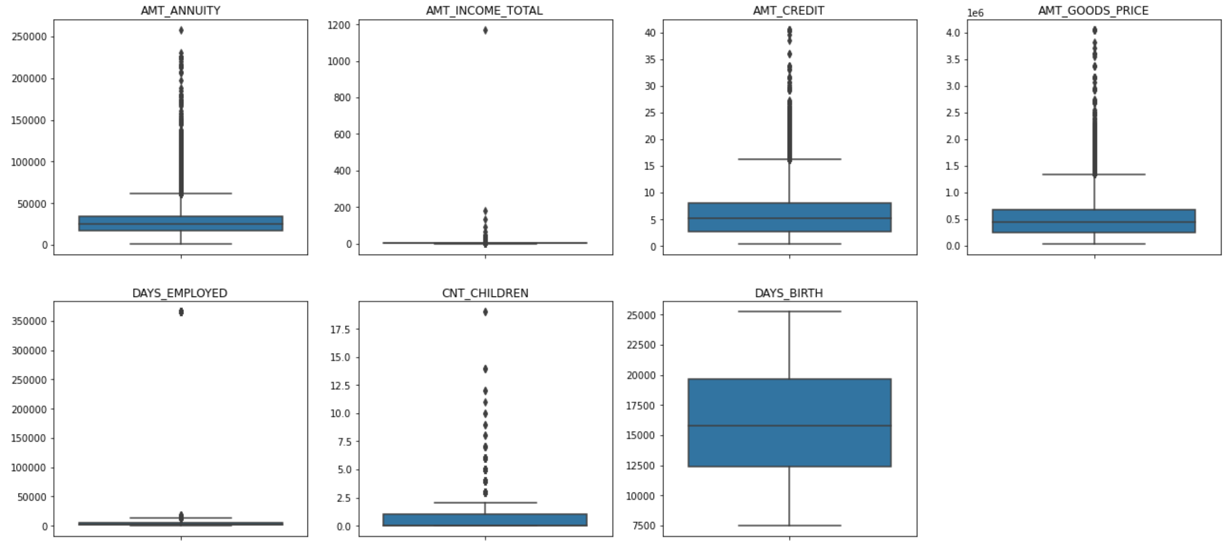
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 307511 entries, 0 to 307510
Data columns (total 51 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   SK_ID_CURR                   307511 non-null  int64
 1   TARGET                       307511 non-null  int64
 2   NAME_CONTRACT_TYPE           307511 non-null  category
 3   CODE_GENDER                  307511 non-null  category
 4   FLAG_OWN_CAR                 307511 non-null  category
 5   FLAG_OWN_REALTY              307511 non-null  category
```

```
 6   CNT_CHILDREN                 307511 non-null  int64
 7   AMT_INCOME_TOTAL             307511 non-null  float64
 8   AMT_CREDIT                   307511 non-null  float64
 9   AMT_ANNUITY                  307499 non-null  float64
 10  AMT_GOODS_PRICE              307233 non-null  float64
 11  NAME_TYPE_SUITE              306219 non-null  category
 12  NAME_INCOME_TYPE             307511 non-null  category
 13  NAME_EDUCATION_TYPE          307511 non-null  category
 14  NAME_FAMILY_STATUS           307511 non-null  category
 15  NAME_HOUSING_TYPE            307511 non-null  category
 16  REGION_POPULATION_RELATIVE   307511 non-null  float64
 17  DAYS_BIRTH                   307511 non-null  int64
 18  DAYS_EMPLOYED                307511 non-null  int64
 19  DAYS_REGISTRATION            307511 non-null  float64
 20  DAYS_ID_PUBLISH              307511 non-null  int64
 21  CNT_FAM_MEMBERS              307509 non-null  float64
 22  REGION_RATING_CLIENT         307511 non-null  category
 23  REGION_RATING_CLIENT_W_CITY  307511 non-null  category
 24  WEEKDAY_APPR_PROCESS_START   307511 non-null  category
 25  HOUR_APPR_PROCESS_START      307511 non-null  int64
 26  REG_REGION_NOT_LIVE_REGION   307511 non-null  int64
 27  REG_REGION_NOT_WORK_REGION   307511 non-null  category
 28  LIVE_REGION_NOT_WORK_REGION  307511 non-null  category
 29  REG_CITY_NOT_LIVE_CITY       307511 non-null  category
 30  REG_CITY_NOT_WORK_CITY       307511 non-null  category
 31  LIVE_CITY_NOT_WORK_CITY      307511 non-null  category
 32  ORGANIZATION_TYPE            307511 non-null  category
 33  OBS_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 34  DEF_30_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 35  OBS_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 36  DEF_60_CNT_SOCIAL_CIRCLE     306490 non-null  float64
 37  DAYS_LAST_PHONE_CHANGE       307510 non-null  float64
 38  FLAG_DOCUMENT_3              307511 non-null  int64
 39  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64
 40  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64
 41  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64
 42  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64
 43  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64
 44  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64
 45  INCOME_RANGE                 307261 non-null  category
 46  CREDIT_RANGE                 257526 non-null  category
 47  YEARS_EMPLOYED               307511 non-null  int64
 48  WORK_EXP                     224233 non-null  category
 49  AGE                          307511 non-null  int64
 50  AGE_RANGE                    307511 non-null  category
dtypes: category(22), float64(18), int64(11)
memory usage: 76.8 MB
```

From the above inoformation we can see that the the column datatype is converted from object to categorical

In [30]:

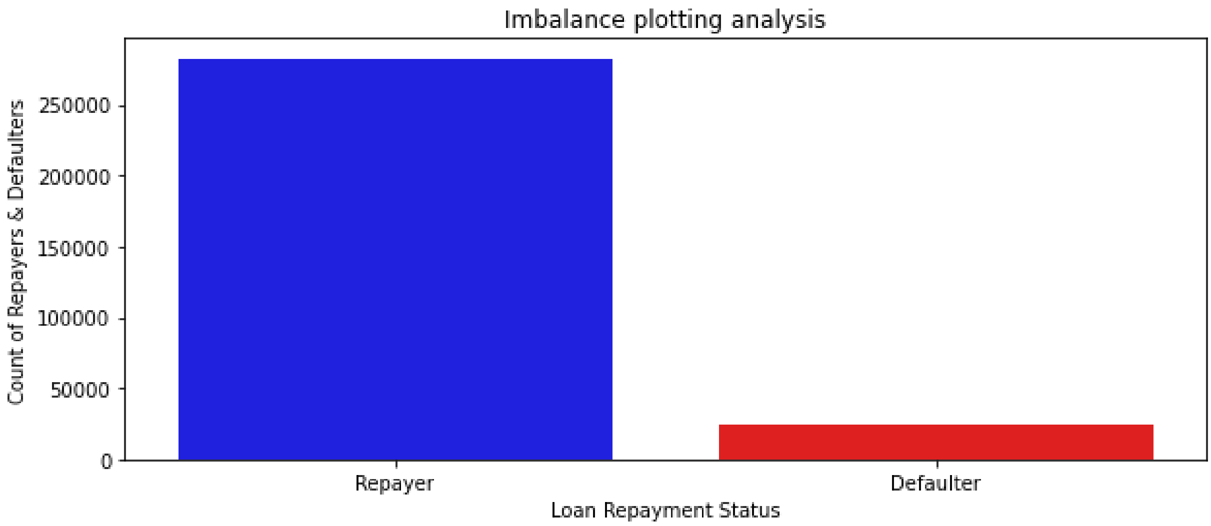
```
#to check for the outliers in the dataset
plt.figure(figsize=(22,10))

outlier_1 = ['AMT_ANNUITY', 'AMT_INCOME_TOTAL','AMT_CREDIT','AMT_GOODS_PRICE','DAYS_

for i in enumerate(outlier_1):
    plt.subplot(2,4,i[0]+1)
    sns.boxplot(y=data_null[i[1]])
    plt.title(i[1])
    plt.ylabel("")
```



AMT_ANNUITY, AMT_CREDIT, AMT_GOODS_PRICE, CNT_CHILDREN columns have few outliers, while AMT_INCOME_TOTAL column has large no. of outliers. DAYS_BIRTH column has no outliers. DAYS_EMPLOYED has outlier, but may be incorrect entry as the range is showed as more than 350k (which is not possible)

In [31]:

```
#analysis of the TARGET data
data_target = data_null["TARGET"].value_counts().reset_index()

plt.figure(figsize=(10,4))
x = ['Repayer','Defaulter']
sns.barplot(x = x, y = "TARGET", data = data_target, palette= ['b','r'])
plt.xlabel("Loan Repayment Status")
plt.ylabel("Count of Repayers & Defaulters")
plt.title("Imbalance plotting analysis")
plt.show()
```



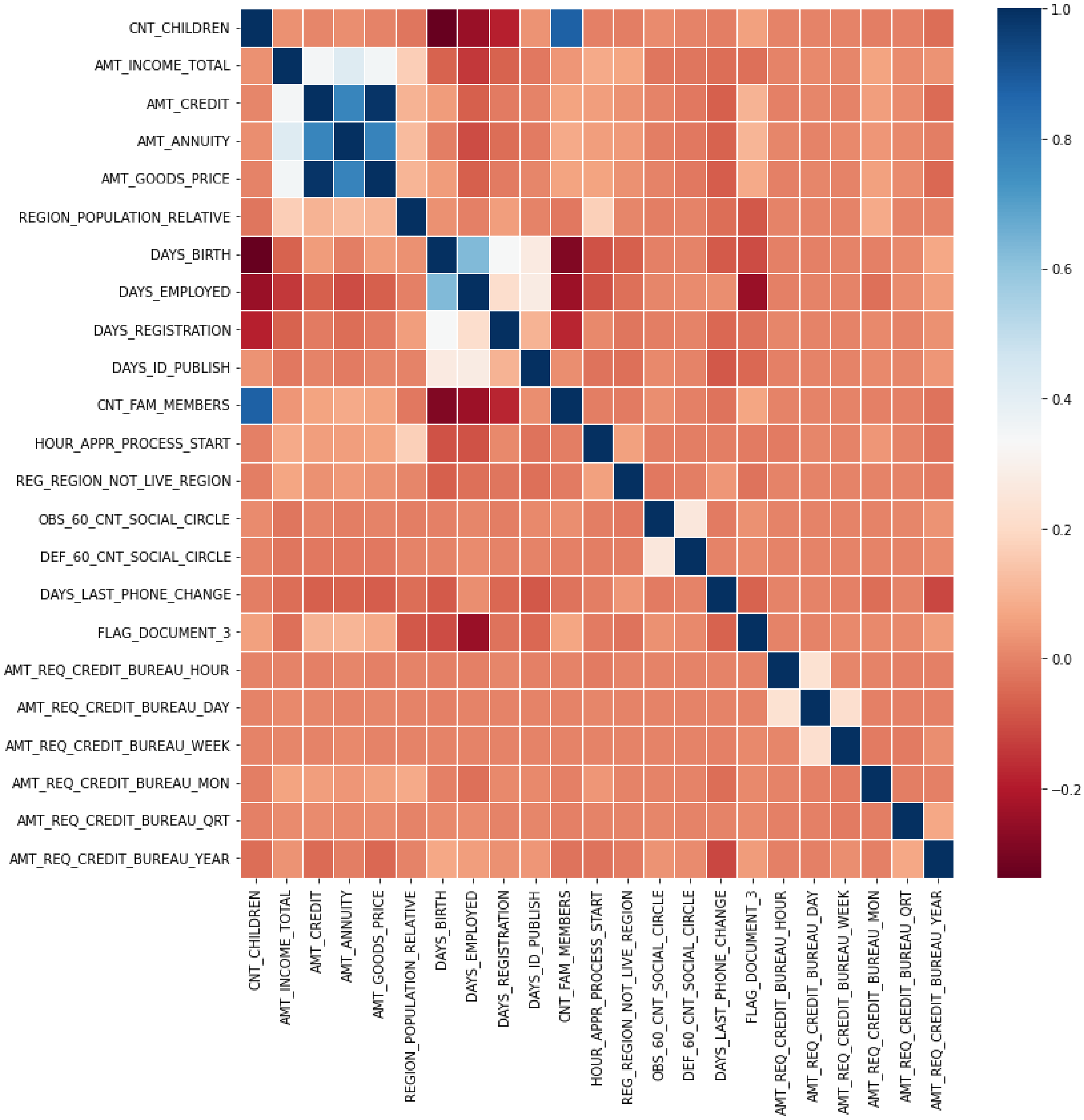
In [32]:

```
#to check corelation between Repayers and Defaulters

#storing the desired columns in data_col variable
data_col = ['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY',
            'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GO
            'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMI
            'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_E
            'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'CNT_FAM_MEMBERS', 'REGION_RATIN
            'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_
            'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION
```

```
            'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_
            'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE
            'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CRED
            'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDI

#Repayers data store in data_repayer variable
data_repayer = data_null.loc[data_null['TARGET'] == 0, data_col]

#Defaulters data store in data_defaulter variable
data_defaulter = data_null.loc[data_null['TARGET'] == 1, data_col]
```

In [33]:

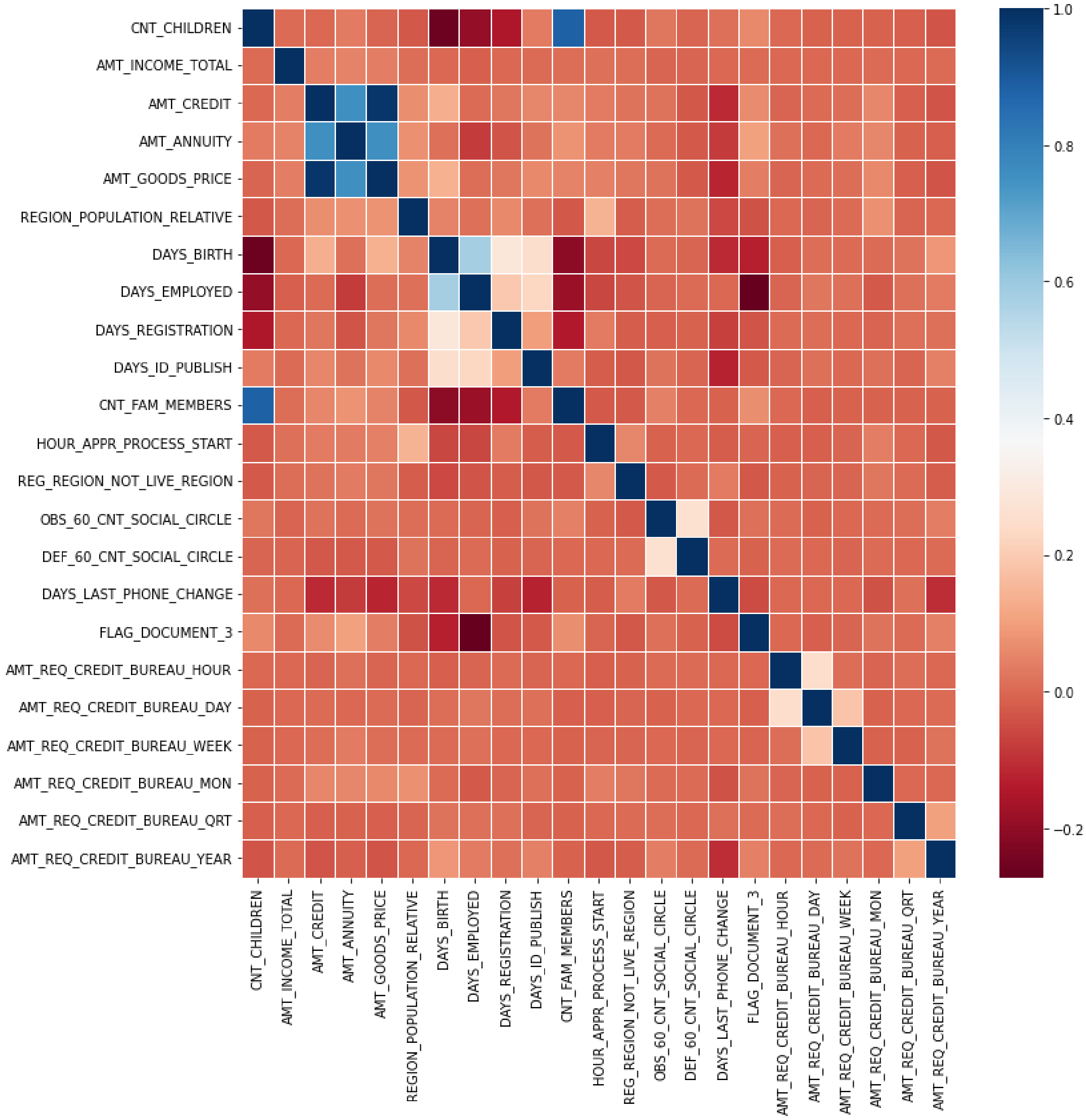
```
#to find the corelation for repayers data where TARGET value is 0
fig = plt.figure(figsize = (12,12))
ax = sns.heatmap(data_repayer.corr(), cmap = "RdBu", annot = False, linewidth = 1)
```



From the above heatmap we can see that Credit amount is highly correlated with amount of goods price, loan annuity and total income

In [34]:

```
#to find the corelation for Defaulter data where TARGET value is 1
fig = plt.figure(figsize = (12,12))
ax = sns.heatmap(data_defaulter.corr(), cmap="RdBu", annot = False, linewidth = 1)
```



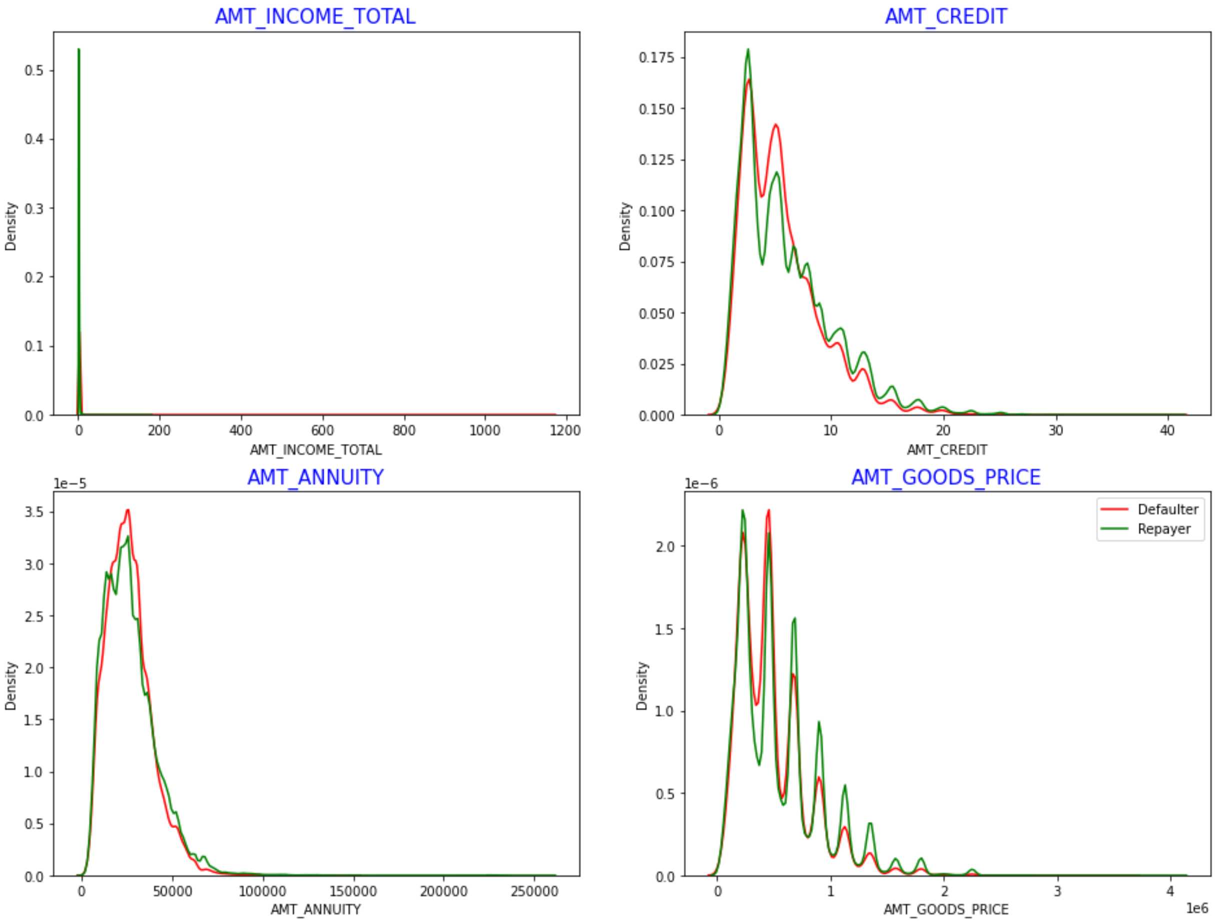
Credit amount is highly correlated with amount of goods price which is same as repayers. But in case of loan annuity correlation with credit amount has slightly reduced when compared to repayers There is a drop in the correlation between total income of the client and the credit amount amongst defaulters whereas it is among repayers. Days_birth and number of children correlation has reduced to in defaulters when compared to in repayers.

In [35]:

```
# Plotting the numerical columns related to amount as distribution plot to see densi
amount = data_null[[ 'AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY', 'AMT_GOODS_PRICE

fig = plt.figure(figsize=(16,12))

for i in enumerate(amount):
    plt.subplot(2,2,i[0]+1)
    sns.distplot(data_defaulter[i[1]], hist = False, color='r', label = "Defaulter")
    sns.distplot(data_repayer[i[1]], hist = False, color='g', label = "Repayer")
    plt.title(i[1], fontdict={'fontsize' : 15, 'fontweight' : 5, 'color' : 'Blue'})

plt.legend()
plt.show()
```

```
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future version.
```

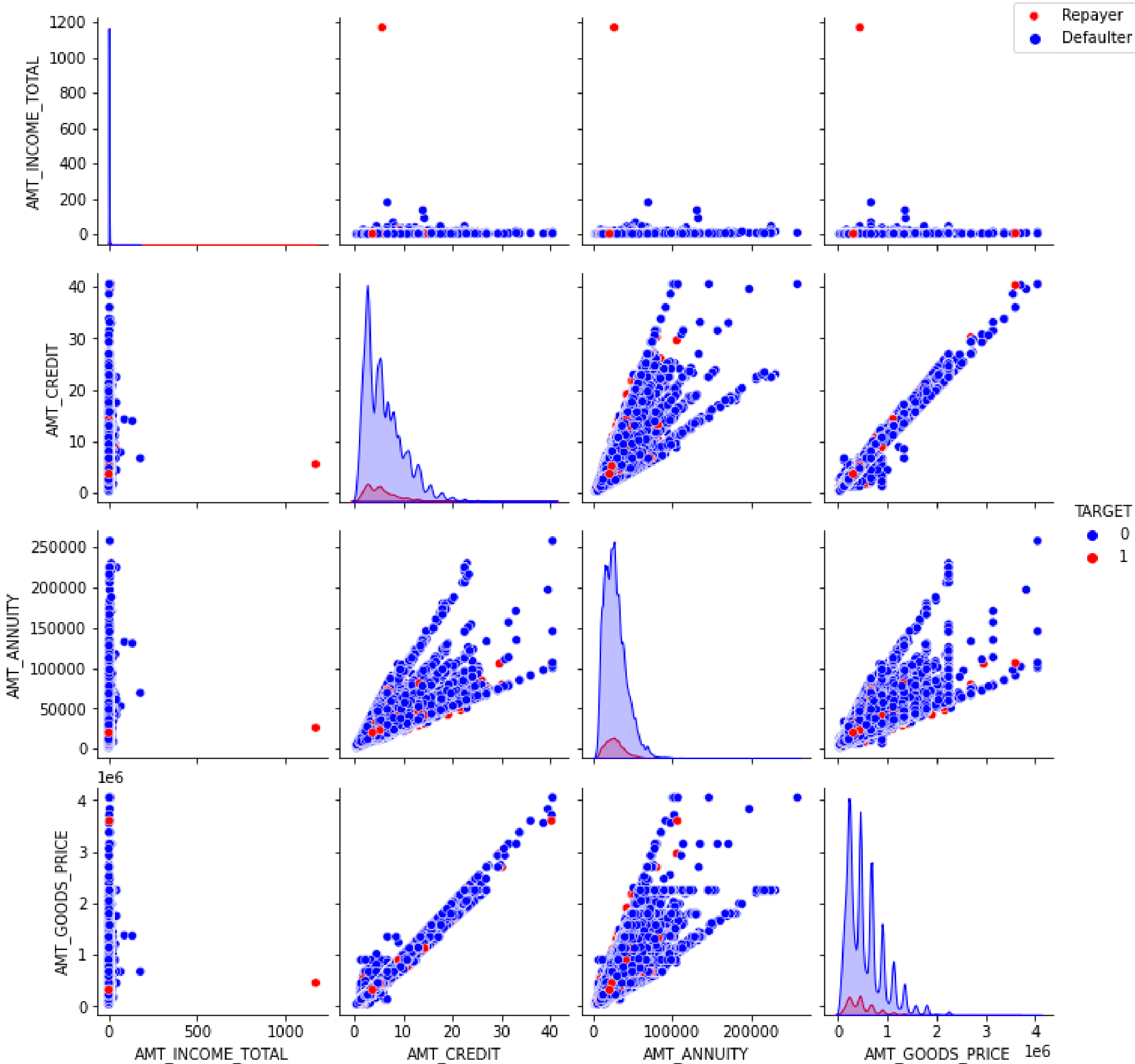
```
Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\priyanka\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



From above graph we can see that most no of loans are given for goods price below 10 lakhs and most people pay annuity below 50000 for the credit loan. Also, credit amount of the loan is mostly less then 10 lakhs. The repayers and defaulters distribution overlap in all the plots and hence we cannot use any of these variables to make a decision.

In [62]:

```
amount = data_null[[ 'AMT_INCOME_TOTAL','AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRIC
amount = amount[(amount["AMT_GOODS_PRICE"].notnull()) & (amount["AMT_ANNUITY"].notnu
ax= sns.pairplot(amount, hue = "TARGET", palette = ["b","r"])
ax.fig.legend(labels=['Repayer','Defaulter'])
plt.show()
```



From the above plots we can see that when AMT_ANNUITY > 15000 AMT_GOODS_PRICE > 3M, there is a lesser chance of defaulters AMT_CREDIT and AMT_GOODS_PRICE are highly correlated with eachother.

In [ ]:

EDA of previous application data

In [36]:

```
#reading previous_application data
data_p = pd.read_csv('C:/Intellipaat/ThinkEvolve/Credit EDA Case Study-20220210T0526
data_p
```

Out[36]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AI
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	

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AI
...	...	...	...	...	...	
1670209	2300464	352015	Consumer loans	14704.290	267295.5	
1670210	2357031	334635	Consumer loans	6622.020	87750.0	
1670211	2659632	249544	Consumer loans	11520.855	105237.0	
1670212	2785582	400317	Cash loans	18821.520	180000.0	
1670213	2418762	261212	Cash loans	16431.300	360000.0	

1670214 rows × 37 columns

In [37]:

```
#get the information of the columns in the dataset
data_p.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 
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64  
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object 
 16  NAME_CONTRACT_STATUS         1670214 non-null  object 
 17  DAYS_DECISION                1670214 non-null  int64  
 18  NAME_PAYMENT_TYPE            1670214 non-null  object 
 19  CODE_REJECT_REASON           1670214 non-null  object 
 20  NAME_TYPE_SUITE              849809 non-null   object 
 21  NAME_CLIENT_TYPE             1670214 non-null  object 
 22  NAME_GOODS_CATEGORY          1670214 non-null  object 
 23  NAME_PORTFOLIO               1670214 non-null  object 
 24  NAME_PRODUCT_TYPE            1670214 non-null  object 
 25  CHANNEL_TYPE                 1670214 non-null  object 
 26  SELLERPLACE_AREA             1670214 non-null  int64  
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object 
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object 
 30  PRODUCT_COMBINATION          1669868 non-null  object 
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
```

```
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
```

In [38]:

```
data_p.columns
```

Out[38]:

```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_ANNUITY',
       'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_DOWN_PAYMENT', 'AMT_GOODS_PRICE',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       'FLAG_LAST_APPL_PER_CONTRACT', 'NFLAG_LAST_APPL_IN_DAY',
       'RATE_DOWN_PAYMENT', 'RATE_INTEREST_PRIMARY',
       'RATE_INTEREST_PRIVILEGED', 'NAME_CASH_LOAN_PURPOSE',
       'NAME_CONTRACT_STATUS', 'DAYS_DECISION', 'NAME_PAYMENT_TYPE',
       'CODE_REJECT_REASON', 'NAME_TYPE_SUITE', 'NAME_CLIENT_TYPE',
       'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO', 'NAME_PRODUCT_TYPE',
       'CHANNEL_TYPE', 'SELLERPLACE_AREA', 'NAME_SELLER_INDUSTRY',
       'CNT_PAYMENT', 'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION',
       'DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION',
       'DAYS_LAST_DUE', 'DAYS_TERMINATION', 'NFLAG_INSURED_ON_APPROVAL'],
      dtype='object')
```

In [39]:

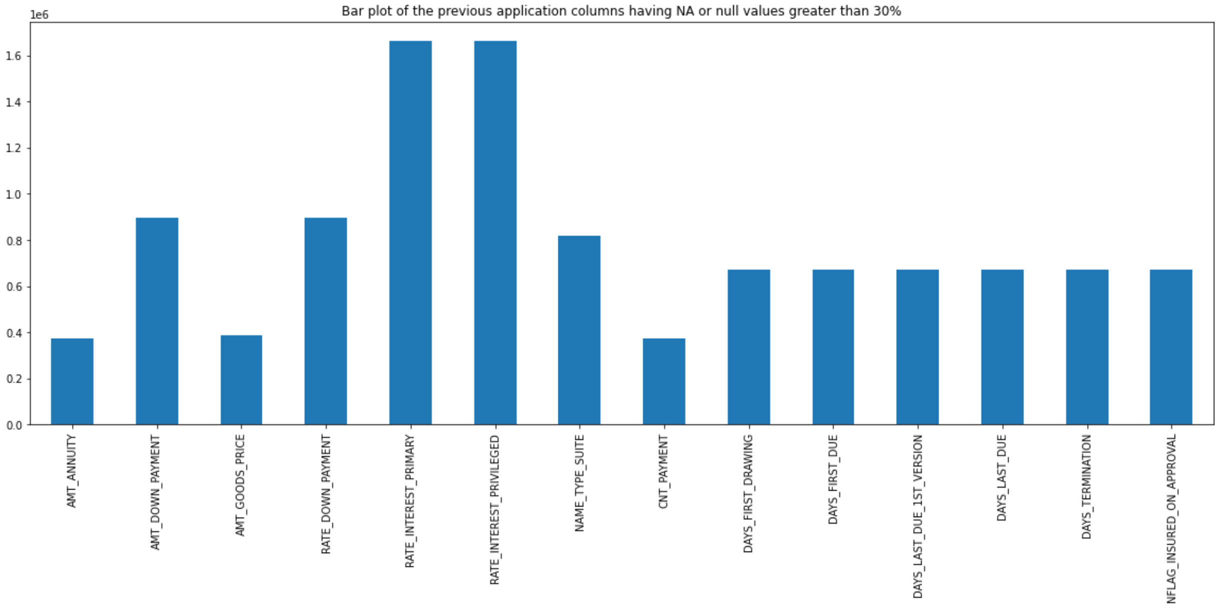
```
#to check for any null values
data_p.isna().sum()
```

Out[39]:

```
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 [40]:

```
#to check for missing values and plot graph for the null value columns where null va
data_na_p = data_p.isna().sum()
data_na_p = data_na_p[data_na_p.values > (0.3 * len(data_c))]
plt.figure(figsize=(20,7))
data_na_p.plot(kind='bar')
plt.title('Bar plot of the previous application columns having NA or null values gre
plt.show()
```



From the above plot we can see that there are 14 columns that have null values that are greater than 30%. So, in this case we can remove these columns.

In [41]:

```
#removing the above columns and other unwanted columns like 'WEEKDAY_APPR_PROCESS_ST
# 'FLAG_LAST_APPL_PER_CONTRACT','NFLAG_LAST_APPL_IN_DAY'

#dropping the unwanted columns

data_p.drop(['DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION','DA
         'HOUR_APPR_PROCESS_START','FLAG_LAST_APPL_PER_CONTRACT','NFLAG_LAST_APPL_IN_
         'AMT_DOWN_PAYMENT','CNT_PAYMENT', 'NAME_TYPE_SUITE','RATE_INTEREST_PRIVILEGE
         'RATE_DOWN_PAYMENT','AMT_GOODS_PRICE'], axis = 1, inplace = True)

#to get columns after removing unwanted columns
data_p.columns
```

Out[41]:

```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_APPLICATION',
       'AMT_CREDIT', 'NAME_CASH_LOAN_PURPOSE', 'NAME_CONTRACT_STATUS',
       'DAYS_DECISION', 'NAME_PAYMENT_TYPE', 'CODE_REJECT_REASON',
       'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO',
       'NAME_PRODUCT_TYPE', 'CHANNEL_TYPE', 'SELLERPLACE_AREA',
       'NAME_SELLER_INDUSTRY', 'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION'],
      dtype='object')
```

In [42]:

```
#to get the shape of the dataset
data_p.shape
```

Out[42]:

```
(1670214, 19)
```

In [43]:

```
#to get the description of the dataset
data_p.describe()
```

Out[43]:

	SK_ID_PREV	SK_ID_CURR	AMT_APPLICATION	AMT_CREDIT	DAYS_DECISION	SELLERPLACI
count	1.670214e+06	1.670214e+06	1.670214e+06	1.670213e+06	1.670214e+06	1.6702
mean	1.923089e+06	2.783572e+05	1.752339e+05	1.961140e+05	-8.806797e+02	3.1395
std	5.325980e+05	1.028148e+05	2.927798e+05	3.185746e+05	7.790997e+02	7.1274
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00	-2.922000e+03	-1.0000
25%	1.461857e+06	1.893290e+05	1.872000e+04	2.416050e+04	-1.300000e+03	-1.0000
50%	1.923110e+06	2.787145e+05	7.104600e+04	8.054100e+04	-5.810000e+02	3.0000
75%	2.384280e+06	3.675140e+05	1.803600e+05	2.164185e+05	-2.800000e+02	8.2000
max	2.845382e+06	4.562550e+05	6.905160e+06	6.905160e+06	-1.000000e+00	4.0000

In [44]:

```
#conversion of days from negetive to positive values
data_p['DAYS_DECISION'] = abs(data_p['DAYS_DECISION'])
data_p['DAYS_DECISION']
```

Out[44]:

```
0            73
1           164
2           301
3           512
4           781
           ...
1670209     544
1670210    1694
1670211    1488
1670212    1185
1670213    1193
Name: DAYS_DECISION, Length: 1670214, dtype: int64
```

In [45]:

```
#conversion of object data type to categorical datatype

#to get the information of the dataset
data_p.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 19 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_APPLICATION          1670214 non-null  float64
 4   AMT_CREDIT               1670213 non-null  float64
 5   NAME_CASH_LOAN_PURPOSE   1670214 non-null  object
 6   NAME_CONTRACT_STATUS     1670214 non-null  object
 7   DAYS_DECISION            1670214 non-null  int64
 8   NAME_PAYMENT_TYPE        1670214 non-null  object
 9   CODE_REJECT_REASON       1670214 non-null  object
 10  NAME_CLIENT_TYPE         1670214 non-null  object
 11  NAME_GOODS_CATEGORY      1670214 non-null  object
 12  NAME_PORTFOLIO           1670214 non-null  object
 13  NAME_PRODUCT_TYPE        1670214 non-null  object
 14  CHANNEL_TYPE             1670214 non-null  object
 15  SELLERPLACE_AREA         1670214 non-null  int64
 16  NAME_SELLER_INDUSTRY     1670214 non-null  object
```

```
 17  NAME_YIELD_GROUP        1670214 non-null  object
 18  PRODUCT_COMBINATION     1669868 non-null  object
dtypes: float64(2), int64(4), object(13)
memory usage: 242.1+ MB
```

From above information we can see that there are some of the columns that have data type as object which needs to be converted to categorical datatype

In [46]:

```
#get all the object columns & store in a variable
data_p_cat = ['NAME_CASH_LOAN_PURPOSE','NAME_CONTRACT_TYPE','NAME_PAYMENT_TYPE',
             'CODE_REJECT_REASON','NAME_CLIENT_TYPE','NAME_GOODS_CATEGORY','NAME_PORT
            'NAME_PRODUCT_TYPE','CHANNEL_TYPE','NAME_SELLER_INDUSTRY','NAME_YIELD_GRO
             'NAME_CONTRACT_STATUS']

#using for loop for conversion
for c in data_p_cat:
    data_p[c] = pd.Categorical(data_p[c])
```

In [47]:

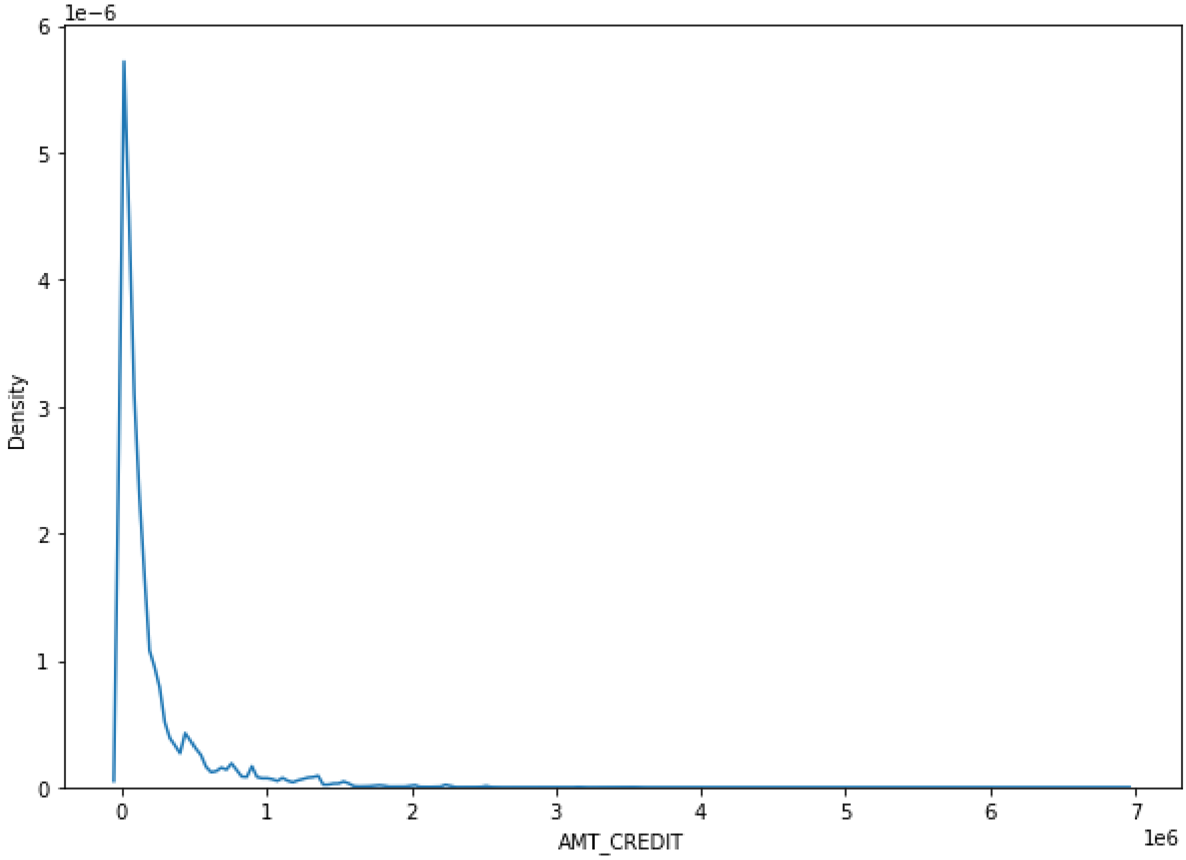
```
#to get the information of the dataset after conversion
data_p.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 19 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  category
 3   AMT_APPLICATION         1670214 non-null  float64
 4   AMT_CREDIT              1670213 non-null  float64
 5   NAME_CASH_LOAN_PURPOSE  1670214 non-null  category
 6   NAME_CONTRACT_STATUS    1670214 non-null  category
 7   DAYS_DECISION           1670214 non-null  int64
 8   NAME_PAYMENT_TYPE       1670214 non-null  category
 9   CODE_REJECT_REASON      1670214 non-null  category
 10  NAME_CLIENT_TYPE        1670214 non-null  category
 11  NAME_GOODS_CATEGORY     1670214 non-null  category
 12  NAME_PORTFOLIO          1670214 non-null  category
 13  NAME_PRODUCT_TYPE       1670214 non-null  category
 14  CHANNEL_TYPE            1670214 non-null  category
 15  SELLERPLACE_AREA        1670214 non-null  int64
 16  NAME_SELLER_INDUSTRY    1670214 non-null  category
 17  NAME_YIELD_GROUP        1670214 non-null  category
 18  PRODUCT_COMBINATION     1669868 non-null  category
dtypes: category(13), float64(2), int64(4)
memory usage: 97.2 MB
```

From the above information we can see that the object type data has been converted to categorical datatype

In [48]:

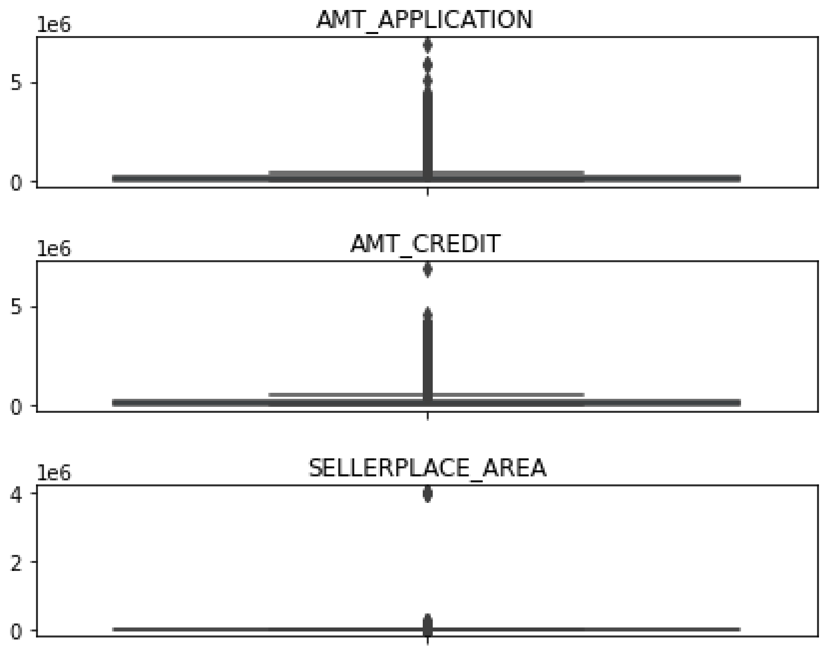
```
#to plot AMT_CREDIT column values inorder to check the skewness
plt.figure(figsize=(10,7))
sns.kdeplot(data_p['AMT_CREDIT'])
plt.show()
```

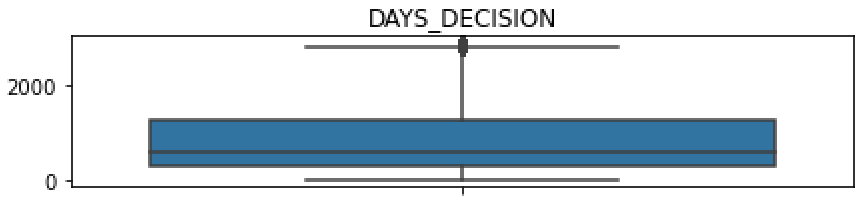


From the above plot the peak is on left side meaning positive skewed data.

In [49]:

```
#check outliers for the dataset

data_p_out = ['AMT_APPLICATION','AMT_CREDIT','SELLERPLACE_AREA', 'DAYS_DECISION']

for i in enumerate(data_p_out):
    plt.figure(figsize=(7,8))
    plt.subplot(5,1,i[0]+1)
    sns.boxplot(y=data_p[i[1]])
    plt.title(i[1])
    plt.ylabel("")
```





From above plot we can see that AMT_ANNUITY, AMT_APPLICATION, AMT_CREDIT, AMT_GOODS_PRICE, SELLERPLACE_AREA have more number of outliers. DAYS_DECISION column has few outliers that indicates loan application decision were taken long back.

In [50]:
```
data_p.columns
```

Out[50]:
```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_APPLICATION',
       'AMT_CREDIT', 'NAME_CASH_LOAN_PURPOSE', 'NAME_CONTRACT_STATUS',
       'DAYS_DECISION', 'NAME_PAYMENT_TYPE', 'CODE_REJECT_REASON',
       'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO',
       'NAME_PRODUCT_TYPE', 'CHANNEL_TYPE', 'SELLERPLACE_AREA',
       'NAME_SELLER_INDUSTRY', 'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION'],
      dtype='object')
```

Merging both the datasets

In [51]:
```
#to merge both the datasets using inner join and SK_ID_CURR as primary key
data_merge = pd.merge(data_null, data_p, how = 'inner', on = 'SK_ID_CURR')
data_merge
```

Out[51]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG_C
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100003	0	Cash loans	F	N	
3	100003	0	Cash loans	F	N	
4	100004	0	Revolving loans	M	Y	
...	...	...	...	...	...	
1413696	456255	0	Cash loans	F	N	
1413697	456255	0	Cash loans	F	N	
1413698	456255	0	Cash loans	F	N	
1413699	456255	0	Cash loans	F	N	
1413700	456255	0	Cash loans	F	N	

1413701 rows × 69 columns

In [52]:
```
#to get information of the merged dataset
```

```
data_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1413701 entries, 0 to 1413700
Data columns (total 69 columns):
 #   Column                       Non-Null Count    Dtype   
---  ------                       --------------    -----   
 0   SK_ID_CURR                   1413701 non-null  int64   
 1   TARGET                       1413701 non-null  int64   
 2   NAME_CONTRACT_TYPE_x         1413701 non-null  category
 3   CODE_GENDER                  1413701 non-null  category
 4   FLAG_OWN_CAR                 1413701 non-null  category
 5   FLAG_OWN_REALTY              1413701 non-null  category
 6   CNT_CHILDREN                 1413701 non-null  int64   
 7   AMT_INCOME_TOTAL             1413701 non-null  float64 
 8   AMT_CREDIT_x                 1413701 non-null  float64 
 9   AMT_ANNUITY                  1413608 non-null  float64 
 10  AMT_GOODS_PRICE              1412493 non-null  float64 
 11  NAME_TYPE_SUITE              1410175 non-null  category
 12  NAME_INCOME_TYPE             1413701 non-null  category
 13  NAME_EDUCATION_TYPE          1413701 non-null  category
 14  NAME_FAMILY_STATUS           1413701 non-null  category
 15  NAME_HOUSING_TYPE            1413701 non-null  category
 16  REGION_POPULATION_RELATIVE   1413701 non-null  float64 
 17  DAYS_BIRTH                   1413701 non-null  int64   
 18  DAYS_EMPLOYED                1413701 non-null  int64   
 19  DAYS_REGISTRATION            1413701 non-null  float64 
 20  DAYS_ID_PUBLISH              1413701 non-null  int64   
 21  CNT_FAM_MEMBERS              1413701 non-null  float64 
 22  REGION_RATING_CLIENT         1413701 non-null  category
 23  REGION_RATING_CLIENT_W_CITY  1413701 non-null  category
 24  WEEKDAY_APPR_PROCESS_START   1413701 non-null  category
 25  HOUR_APPR_PROCESS_START      1413701 non-null  int64   
 26  REG_REGION_NOT_LIVE_REGION   1413701 non-null  int64   
 27  REG_REGION_NOT_WORK_REGION   1413701 non-null  category
 28  LIVE_REGION_NOT_WORK_REGION  1413701 non-null  category
 29  REG_CITY_NOT_LIVE_CITY       1413701 non-null  category
 30  REG_CITY_NOT_WORK_CITY       1413701 non-null  category
 31  LIVE_CITY_NOT_WORK_CITY      1413701 non-null  category
 32  ORGANIZATION_TYPE            1413701 non-null  category
 33  OBS_30_CNT_SOCIAL_CIRCLE     1410555 non-null  float64 
 34  DEF_30_CNT_SOCIAL_CIRCLE     1410555 non-null  float64 
 35  OBS_60_CNT_SOCIAL_CIRCLE     1410555 non-null  float64 
 36  DEF_60_CNT_SOCIAL_CIRCLE     1410555 non-null  float64 
 37  DAYS_LAST_PHONE_CHANGE       1413701 non-null  float64 
 38  FLAG_DOCUMENT_3              1413701 non-null  int64   
 39  AMT_REQ_CREDIT_BUREAU_HOUR   1250074 non-null  float64 
 40  AMT_REQ_CREDIT_BUREAU_DAY    1250074 non-null  float64 
 41  AMT_REQ_CREDIT_BUREAU_WEEK   1250074 non-null  float64 
 42  AMT_REQ_CREDIT_BUREAU_MON    1250074 non-null  float64 
 43  AMT_REQ_CREDIT_BUREAU_QRT    1250074 non-null  float64 
 44  AMT_REQ_CREDIT_BUREAU_YEAR   1250074 non-null  float64 
 45  INCOME_RANGE                 1413001 non-null  category
 46  CREDIT_RANGE                 1195774 non-null  category
 47  YEARS_EMPLOYED               1413701 non-null  int64   
 48  WORK_EXP                     1032756 non-null  category
 49  AGE                          1413701 non-null  int64   
 50  AGE_RANGE                    1413701 non-null  category
 51  SK_ID_PREV                   1413701 non-null  int64   
 52  NAME_CONTRACT_TYPE_y         1413701 non-null  category
 53  AMT_APPLICATION              1413701 non-null  float64 
 54  AMT_CREDIT_y                 1413700 non-null  float64 
 55  NAME_CASH_LOAN_PURPOSE       1413701 non-null  category
 56  NAME_CONTRACT_STATUS         1413701 non-null  category
```

```
 57  DAYS_DECISION              1413701 non-null  int64
 58  NAME_PAYMENT_TYPE          1413701 non-null  category
 59  CODE_REJECT_REASON         1413701 non-null  category
 60  NAME_CLIENT_TYPE           1413701 non-null  category
 61  NAME_GOODS_CATEGORY        1413701 non-null  category
 62  NAME_PORTFOLIO             1413701 non-null  category
 63  NAME_PRODUCT_TYPE          1413701 non-null  category
 64  CHANNEL_TYPE               1413701 non-null  category
 65  SELLERPLACE_AREA           1413701 non-null  int64
 66  NAME_SELLER_INDUSTRY       1413701 non-null  category
 67  NAME_YIELD_GROUP           1413701 non-null  category
 68  PRODUCT_COMBINATION        1413388 non-null  category
dtypes: category(35), float64(20), int64(14)
memory usage: 424.7 MB
```

In [53]:

```
data_merge.shape
```

Out[53]:

```
(1413701, 69)
```

In [54]:

```
#description of the dataset
data_merge.describe()
```

Out[54]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT_x	AMT_ANI
count	1.413701e+06	1.413701e+06	1.413701e+06	1.413701e+06	1.413701e+06	1.41360
mean	2.784813e+05	8.655296e-02	4.048933e-01	1.733160e+00	5.875537e+00	2.70170
std	1.028118e+05	2.811789e-01	7.173454e-01	1.985734e+00	3.849173e+00	1.39511
min	1.000020e+05	0.000000e+00	0.000000e+00	2.565000e-01	4.500000e-01	1.61550
25%	1.893640e+05	0.000000e+00	0.000000e+00	1.125000e+00	2.700000e+00	1.68210
50%	2.789920e+05	0.000000e+00	0.000000e+00	1.575000e+00	5.084955e+00	2.49255
75%	3.675560e+05	0.000000e+00	1.000000e+00	2.070000e+00	8.079840e+00	3.45420
max	4.562550e+05	1.000000e+00	1.900000e+01	1.170000e+03	4.050000e+01	2.25000

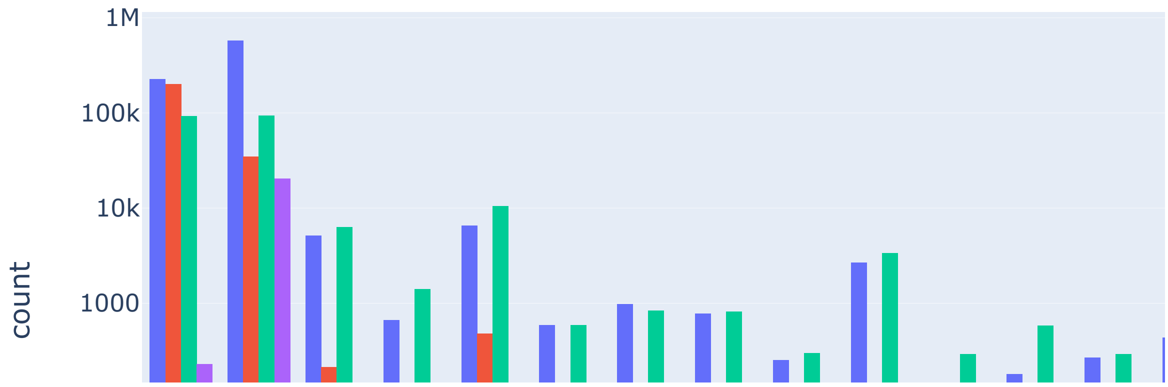
8 rows × 34 columns

In [55]:

```
#to categorize the Target values into repayers and defaulters

#loan repayers
data_R = data_merge[data_merge['TARGET'] == 0]

#loan defaulters
data_D = data_merge[data_merge['TARGET'] == 1]
```

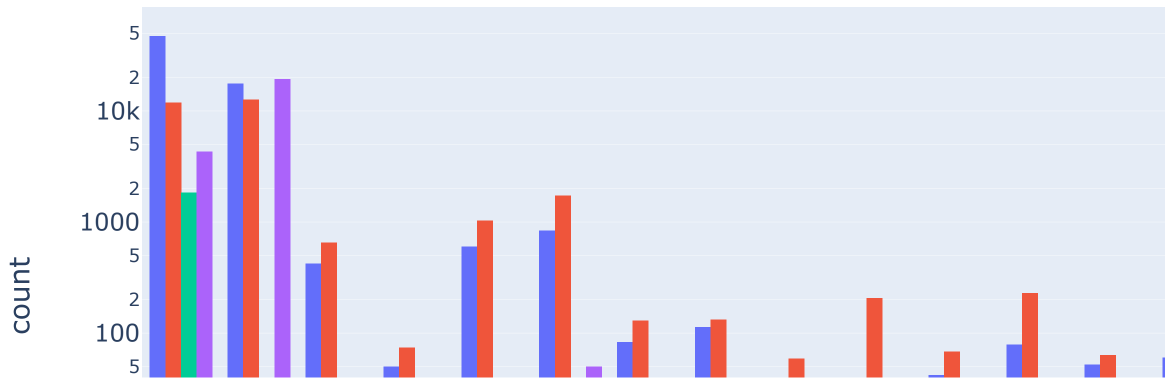
In [64]:

```
#plot grouped bar graph for the data having Target value as 0 that is loan repayers
import plotly.express as px
from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)

fig_r = px.histogram(data_merge, x = data_R["NAME_CASH_LOAN_PURPOSE"],
                     color = data_R['NAME_CONTRACT_STATUS'], barmode = 'group', height
fig_r.update_yaxes(type="log")
fig_r.show()
```



In [65]:

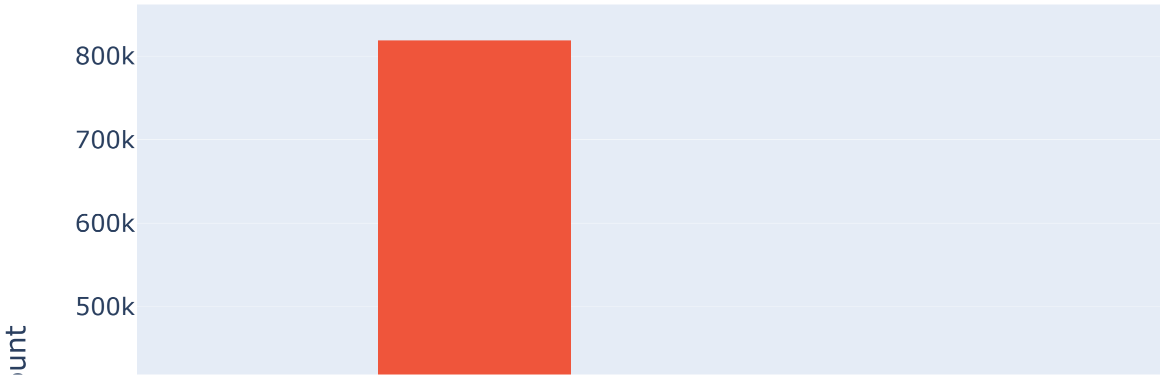
```
#plot grouped bar graph for the data having Target value as 1 that is loan defaulter
fig_d = px.histogram(data_merge, x = data_D["NAME_CASH_LOAN_PURPOSE"],
                     color = data_D['NAME_CONTRACT_STATUS'], barmode = 'group', height
fig_d.update_yaxes(type="log")
fig_d.show()
```



From the above two plots we can see that mostly purpose of the loan are for unknown values like XAP, XNA. Loan taken for other purpose is higher in case of loan repayers where as in case of loan defaulters the loan taken is higher for urgent needs.

In [66]:

```
# to check the Contract Status based on loan repayment status and whether there is a
target = data_merge['TARGET'].replace({1:'Loan defaulter', 0:'Loan repayer'})

fig_l = px.histogram(data_merge, x = 'NAME_CONTRACT_STATUS', color = target, barmode
fig_l.show()
```



In [59]:

```
#to get the percentage and count of the NAME_CONTRACT_STATUS with respent to Target

#applying groupby on NAME_CONTRACT_STATUS and Target
data_g = data_merge.groupby('NAME_CONTRACT_STATUS')['TARGET']
data_g

#concatinating counts and percent values by rounding of upto two decimal points. Als
#Counts and Percentage
data_1 = pd.concat([data_g.value_counts(),round(data_g.value_counts(normalize = True
                axis = 1, keys = ('Counts','Percentage'))

#adding % sign after percent value
```

```
data_1['Percentage'] = data_1['Percentage'].astype(str) +"%"
print(data_1)
```

```
                             Counts Percentage
NAME_CONTRACT_STATUS TARGET
Approved             0       818856     92.41%
                     1        67243      7.59%
Canceled             0       235641     90.83%
                     1        23800      9.17%
Refused              0       215952      88.0%
                     1        29438      12.0%
Unused offer         0        20892     91.75%
                     1         1879      8.25%
```

From the above plot and data we can see that approx. 90 - 92% of the previously cancelled client have repayed the loan. Revisiting the interest rates would increase business opoortunity for these clients 88% of the clients who have been previously refused a loan has payed back the loan in current case. Refusal reason should be recorded for further analysis as these clients may turn into repaying customers.

In [ ]: