```python
# Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```python
# In order to view all columns and rows
pd.set_option('display.max_columns', None)
pd.set_option("display.max_rows",None)
```

```python
# Importing the data and store in variable df
df=pd.read_csv("/content/drive/MyDrive/Colab
Notebooks/application_data.csv")
df.head(3)
```

{"type":"dataframe","variable_name":"df"}

```python
# Size of the data
print(df.shape)
print("Number of Rows in the data   :",df.shape[0])
print("Number of columns in the data: ",df.shape[1])
```

```
(307511, 122)
Number of Rows in the data   : 307511
Number of columns in the data:  122
```

```python
# Size of the data
print(df.shape)
print("Number of Rows in the data   :",df.shape[0])
print("Number of columns in the data: ",df.shape[1])
```

```
(307511, 122)
Number of Rows in the data   : 307511
Number of columns in the data:  122
```

```python
df.info(verbose=True)
# dtypes: float64(65), int64(41), object(16)
# there are 65 columns with data type float64
# there are 41 columns with data type int64
# there are 16 columns with data type categorical
```

```
<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
```

Data Cleaning

```
## 5 point summary
# Check the summary for the numeric columns
df.describe().round(1)
```

{"type":"dataframe"}

Data Cleaning

```
# Check for missing values in percentage

round(100 * df.isnull().mean(),2)
```

```
SK_ID_CURR                      0.00
TARGET                          0.00
NAME_CONTRACT_TYPE              0.00
CODE_GENDER                     0.00
```

```
FLAG_OWN_CAR                      0.00
FLAG_OWN_REALTY                   0.00
CNT_CHILDREN                      0.00
AMT_INCOME_TOTAL                  0.00
AMT_CREDIT                        0.00
AMT_ANNUITY                       0.00
AMT_GOODS_PRICE                   0.09
NAME_TYPE_SUITE                   0.42
NAME_INCOME_TYPE                  0.00
NAME_EDUCATION_TYPE               0.00
NAME_FAMILY_STATUS                0.00
NAME_HOUSING_TYPE                 0.00
REGION_POPULATION_RELATIVE        0.00
DAYS_BIRTH                        0.00
DAYS_EMPLOYED                     0.00
DAYS_REGISTRATION                 0.00
DAYS_ID_PUBLISH                   0.00
OWN_CAR_AGE                      65.99
FLAG_MOBIL                        0.00
FLAG_EMP_PHONE                    0.00
FLAG_WORK_PHONE                   0.00
FLAG_CONT_MOBILE                  0.00
FLAG_PHONE                        0.00
FLAG_EMAIL                        0.00
OCCUPATION_TYPE                  31.35
CNT_FAM_MEMBERS                   0.00
REGION_RATING_CLIENT              0.00
REGION_RATING_CLIENT_W_CITY       0.00
WEEKDAY_APPR_PROCESS_START        0.00
HOUR_APPR_PROCESS_START           0.00
REG_REGION_NOT_LIVE_REGION        0.00
REG_REGION_NOT_WORK_REGION        0.00
LIVE_REGION_NOT_WORK_REGION       0.00
REG_CITY_NOT_LIVE_CITY            0.00
REG_CITY_NOT_WORK_CITY            0.00
LIVE_CITY_NOT_WORK_CITY           0.00
ORGANIZATION_TYPE                 0.00
EXT_SOURCE_1                     56.38
EXT_SOURCE_2                      0.21
EXT_SOURCE_3                     19.83
APARTMENTS_AVG                   50.75
BASEMENTAREA_AVG                 58.52
YEARS_BEGINEXPLUATATION_AVG      48.78
YEARS_BUILD_AVG                  66.50
COMMONAREA_AVG                   69.87
ELEVATORS_AVG                    53.30
ENTRANCES_AVG                    50.35
FLOORSMAX_AVG                    49.76
FLOORSMIN_AVG                    67.85
```

```
LANDAREA_AVG                      59.38
LIVINGAPARTMENTS_AVG              68.35
LIVINGAREA_AVG                    50.19
NONLIVINGAPARTMENTS_AVG           69.43
NONLIVINGAREA_AVG                 55.18
APARTMENTS_MODE                   50.75
BASEMENTAREA_MODE                 58.52
YEARS_BEGINEXPLUATATION_MODE      48.78
YEARS_BUILD_MODE                  66.50
COMMONAREA_MODE                   69.87
ELEVATORS_MODE                    53.30
ENTRANCES_MODE                    50.35
FLOORSMAX_MODE                    49.76
FLOORSMIN_MODE                    67.85
LANDAREA_MODE                     59.38
LIVINGAPARTMENTS_MODE             68.35
LIVINGAREA_MODE                   50.19
NONLIVINGAPARTMENTS_MODE          69.43
NONLIVINGAREA_MODE                55.18
APARTMENTS_MEDI                   50.75
BASEMENTAREA_MEDI                 58.52
YEARS_BEGINEXPLUATATION_MEDI      48.78
YEARS_BUILD_MEDI                  66.50
COMMONAREA_MEDI                   69.87
ELEVATORS_MEDI                    53.30
ENTRANCES_MEDI                    50.35
FLOORSMAX_MEDI                    49.76
FLOORSMIN_MEDI                    67.85
LANDAREA_MEDI                     59.38
LIVINGAPARTMENTS_MEDI             68.35
LIVINGAREA_MEDI                   50.19
NONLIVINGAPARTMENTS_MEDI          69.43
NONLIVINGAREA_MEDI                55.18
FONDKAPREMONT_MODE                68.39
HOUSETYPE_MODE                    50.18
TOTALAREA_MODE                    48.27
WALLSMATERIAL_MODE                50.84
EMERGENCYSTATE_MODE               47.40
OBS_30_CNT_SOCIAL_CIRCLE           0.33
DEF_30_CNT_SOCIAL_CIRCLE           0.33
OBS_60_CNT_SOCIAL_CIRCLE           0.33
DEF_60_CNT_SOCIAL_CIRCLE           0.33
DAYS_LAST_PHONE_CHANGE             0.00
FLAG_DOCUMENT_2                    0.00
FLAG_DOCUMENT_3                    0.00
FLAG_DOCUMENT_4                    0.00
FLAG_DOCUMENT_5                    0.00
FLAG_DOCUMENT_6                    0.00
FLAG_DOCUMENT_7                    0.00
```

```
FLAG_DOCUMENT_8                   0.00
FLAG_DOCUMENT_9                   0.00
FLAG_DOCUMENT_10                  0.00
FLAG_DOCUMENT_11                  0.00
FLAG_DOCUMENT_12                  0.00
FLAG_DOCUMENT_13                  0.00
FLAG_DOCUMENT_14                  0.00
FLAG_DOCUMENT_15                  0.00
FLAG_DOCUMENT_16                  0.00
FLAG_DOCUMENT_17                  0.00
FLAG_DOCUMENT_18                  0.00
FLAG_DOCUMENT_19                  0.00
FLAG_DOCUMENT_20                  0.00
FLAG_DOCUMENT_21                  0.00
AMT_REQ_CREDIT_BUREAU_HOUR       13.50
AMT_REQ_CREDIT_BUREAU_DAY        13.50
AMT_REQ_CREDIT_BUREAU_WEEK       13.50
AMT_REQ_CREDIT_BUREAU_MON        13.50
AMT_REQ_CREDIT_BUREAU_QRT        13.50
AMT_REQ_CREDIT_BUREAU_YEAR       13.50
dtype: float64
```

```
# Columns with more then 40 percent null values

col_with_null_40=(df.isnull().sum()*100/df.shape[0])
[(df.isnull().sum()*100/df.shape[0])>40]
col_with_null_40.shape # there are 49 columns which have more than 40
% null values
```

```
(49,)
```

```
# Dropping null values with more than 40 percent of null values
df1=df.drop(columns=col_with_null_40.index)
```

```
df1.shape # After removing the columns with more than
```

```
(307511, 73)
```

```
df1.describe().round(0)
```

```
{"type":"dataframe"}
```

- CNT_FAM_MEMBERS-Majority of customers are having 3 members in their family
- REGION_RATING_CLIENT-25 % of the customers are living in the region which is having higher risk
- REGION_RATING_CLIENT_W_CITY-on average the client applies loan at noon
- Most of the customers gave their permanent adress is same as contact adress

- Most of the customers gave their permanent adress is same as work adress
- Most of the customers contact adress is same as their work adress
- Most of the customers has the registered city is the same as the current living city
- most of the client lives in city where they work
- most of the customers have higher value in DAYS_LAST_PHONE_CHANGE indicates that the client has had the same phone number for a longer time, potentially indicating stability.
- most of the documents were not provided by the customers
- 4 is the maximun Number of inquiries in the last hour before application.
- most people enquired before 1 year of application
- OBS_30_CNT_SOCIAL_CIRCLE:
- Interpretation: This represents the number of observations of the client's social surroundings with a 30-day past due default (DPD).
- The data suggests that most clients have low to moderate numbers of such observations, with some outliers having significantly more.
- DEF_30_CNT_SOCIAL_CIRCLE:
- Interpretation: This represents how many of the client's social surroundings have defaulted on a 30-day DPD.

The five-point summary shows that defaults are generally rare, with most clients having no defaults in their social circles.

- OBS_60_CNT_SOCIAL_CIRCLE:
- Interpretation: This captures the number of observations of the client's social surroundings with a 60-day DPD default.
- The distribution seems similar to the 30-day observation count, with a few outliers having a higher number of such observations.
- DEF_60_CNT_SOCIAL_CIRCLE:
- Interpretation: This variable represents defaults on a 60-day DPD within the client's social circle.
- Similar to the 30-day default count, defaults remain rare, with most individuals having zero defaults, but with outliers having up to 24 defaults.
- EXT_SOURCE_2 and EXT_SOURCE_3: These could be external score sources (e.g., credit scores, risk factors, etc.)

- higher score might suggest lower risk, while a lower score might indicate higher risk.
- NAME_CONTRACT_TYPE: Type of contract (e.g., Cash loans, Consumer loans). Most frequent is "Cash loans" (278,232 cases).
- CODE_GENDER: Gender of the client (e.g., F, M). Most frequent is "F" (202,448 cases).
- FLAG_OWN_CAR: Whether the client owns a car (Y/N). Most frequent is "Y" (202,924 cases).
- FLAG_OWN_REALTY: Whether the client owns real estate (Y/N). Most frequent is "Y" (213,312 cases).
- NAME_TYPE_SUITE: Type of suite the client lives in (e.g., Unaccompanied, Family). Most frequent is "Unaccompanied" (248,526 cases).
- NAME_INCOME_TYPE: Type of income (e.g., Working, Commercial associate). Most frequent is "Working" (158,774 cases).
- NAME_EDUCATION_TYPE: Education level (e.g., Secondary / secondary special, Higher education). Most frequent is "Secondary / secondary special" (218,391 cases).
- NAME_FAMILY_STATUS: Family status (e.g., Married, Single / not married). Most frequent is "Married" (196,432 cases).
- NAME_HOUSING_TYPE: Type of housing (e.g., House / apartment, Rented apartment). Most frequent is "House / apartment" (272,868 cases).
- OCCUPATION_TYPE: Type of occupation (e.g., Laborers, Managers). Most frequent is "Laborers" (55,186 cases).
- WEEKDAY_APPR_PROCESS_START: Weekday when application process starts (e.g., Monday, Tuesday). Most frequent is "Tuesday" (53,901 cases).
- ORGANIZATION_TYPE: Type of organization (e.g., Business Entity Type 3, State owned). Most frequent is "Business Entity Type 3" (67,992 cases).
- FONDKAPREMONT_MODE: Fund repair mode (e.g., Reg oper account, Money transfer). Most frequent is "Reg oper account" (73,830 cases).
- HOUSETYPE_MODE: House type (e.g., Block of flats, Detached house). Most frequent is "Block of flats" (150,503 cases).
- WALLSMATERIAL_MODE: Wall material of the house (e.g., Panel, Monolithic). Most frequent is "Panel" (66,040 cases).
- EMERGENCYSTATE_MODE: Emergency state (e.g., No, Yes). Most frequent is "No" (159,428 cases).

```
# Check for % missing values for remaining columns
round(100 * df1.isnull().sum() / df1.shape[0],2)
```

```
SK_ID_CURR                      0.00
TARGET                          0.00
NAME_CONTRACT_TYPE              0.00
CODE_GENDER                     0.00
FLAG_OWN_CAR                    0.00
FLAG_OWN_REALTY                 0.00
CNT_CHILDREN                    0.00
AMT_INCOME_TOTAL                0.00
AMT_CREDIT                      0.00
AMT_ANNUITY                     0.00
AMT_GOODS_PRICE                 0.09
```

```
NAME_TYPE_SUITE                  0.42
NAME_INCOME_TYPE                 0.00
NAME_EDUCATION_TYPE              0.00
NAME_FAMILY_STATUS               0.00
NAME_HOUSING_TYPE                0.00
REGION_POPULATION_RELATIVE       0.00
DAYS_BIRTH                       0.00
DAYS_EMPLOYED                    0.00
DAYS_REGISTRATION                0.00
DAYS_ID_PUBLISH                  0.00
FLAG_MOBIL                       0.00
FLAG_EMP_PHONE                   0.00
FLAG_WORK_PHONE                  0.00
FLAG_CONT_MOBILE                 0.00
FLAG_PHONE                       0.00
FLAG_EMAIL                       0.00
OCCUPATION_TYPE                 31.35
CNT_FAM_MEMBERS                  0.00
REGION_RATING_CLIENT             0.00
REGION_RATING_CLIENT_W_CITY      0.00
WEEKDAY_APPR_PROCESS_START       0.00
HOUR_APPR_PROCESS_START          0.00
REG_REGION_NOT_LIVE_REGION       0.00
REG_REGION_NOT_WORK_REGION       0.00
LIVE_REGION_NOT_WORK_REGION      0.00
REG_CITY_NOT_LIVE_CITY           0.00
REG_CITY_NOT_WORK_CITY           0.00
LIVE_CITY_NOT_WORK_CITY          0.00
ORGANIZATION_TYPE                0.00
EXT_SOURCE_2                     0.21
EXT_SOURCE_3                    19.83
OBS_30_CNT_SOCIAL_CIRCLE         0.33
DEF_30_CNT_SOCIAL_CIRCLE         0.33
OBS_60_CNT_SOCIAL_CIRCLE         0.33
DEF_60_CNT_SOCIAL_CIRCLE         0.33
DAYS_LAST_PHONE_CHANGE           0.00
FLAG_DOCUMENT_2                  0.00
FLAG_DOCUMENT_3                  0.00
FLAG_DOCUMENT_4                  0.00
FLAG_DOCUMENT_5                  0.00
FLAG_DOCUMENT_6                  0.00
FLAG_DOCUMENT_7                  0.00
FLAG_DOCUMENT_8                  0.00
FLAG_DOCUMENT_9                  0.00
FLAG_DOCUMENT_10                 0.00
FLAG_DOCUMENT_11                 0.00
FLAG_DOCUMENT_12                 0.00
FLAG_DOCUMENT_13                 0.00
FLAG_DOCUMENT_14                 0.00
```

```
FLAG_DOCUMENT_15                 0.00
FLAG_DOCUMENT_16                 0.00
FLAG_DOCUMENT_17                 0.00
FLAG_DOCUMENT_18                 0.00
FLAG_DOCUMENT_19                 0.00
FLAG_DOCUMENT_20                 0.00
FLAG_DOCUMENT_21                 0.00
AMT_REQ_CREDIT_BUREAU_HOUR      13.50
AMT_REQ_CREDIT_BUREAU_DAY       13.50
AMT_REQ_CREDIT_BUREAU_WEEK      13.50
AMT_REQ_CREDIT_BUREAU_MON       13.50
AMT_REQ_CREDIT_BUREAU_QRT       13.50
AMT_REQ_CREDIT_BUREAU_YEAR      13.50
dtype: float64
```

```
# We dropped columns with more than 40 percent of null values since
imputing this columns will lead to biased dataset
# and will not be able to draw proper insights

# After removing columns we left with 73 columns

# In OCCUPATION_TYPE column we have 31.35 percent of null values and
if we impute it with mode dataset will be higly biased so we impute it
with unknown
```

columns with missing values other than occupation type

AMT_REQ_CREDIT_BUREAU_HOUR 13.50 AMT_REQ_CREDIT_BUREAU_DAY 13.50 AMT_REQ_CREDIT_BUREAU_WEEK 13.50 AMT_REQ_CREDIT_BUREAU_MON 13.50 AMT_REQ_CREDIT_BUREAU_QRT 13.50 AMT_REQ_CREDIT_BUREAU_YEAR 13.50 EXT_SOURCE_3 OBS_30_CNT_SOCIAL_CIRCLE 0.33 DEF_30_CNT_SOCIAL_CIRCLE 0.33 OBS_60_CNT_SOCIAL_CIRCLE 0.33 DEF_60_CNT_SOCIAL_CIRCLE

```
df1[['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_CREDI
T_BUREAU_YEAR']].info()

## these columns are number of inquiries made to credit bureau in the
last hour, day, week, month, quarter
## So we need to impute with median since if we impute with mean it
takes float value and will not be having any meaning
## we need to make this columns integer from float variables
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 6 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   AMT_REQ_CREDIT_BUREAU_HOUR  265992 non-null  float64
 1   AMT_REQ_CREDIT_BUREAU_DAY   265992 non-null  float64
 2   AMT_REQ_CREDIT_BUREAU_WEEK  265992 non-null  float64
 3   AMT_REQ_CREDIT_BUREAU_MON   265992 non-null  float64
 4   AMT_REQ_CREDIT_BUREAU_QRT   265992 non-null  float64
 5   AMT_REQ_CREDIT_BUREAU_YEAR  265992 non-null  float64
dtypes: float64(6)
memory usage: 14.1 MB
```

```
columns=['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_CREDI
T_BUREAU_YEAR']
for i in columns:
  # Impute NaN with median and then convert to integer
  df1[i] = df1[i].fillna(df1[i].median()).astype(int)
df1[['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_CREDI
T_BUREAU_YEAR']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 6 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   AMT_REQ_CREDIT_BUREAU_HOUR  307511 non-null  int64
 1   AMT_REQ_CREDIT_BUREAU_DAY   307511 non-null  int64
 2   AMT_REQ_CREDIT_BUREAU_WEEK  307511 non-null  int64
 3   AMT_REQ_CREDIT_BUREAU_MON   307511 non-null  int64
 4   AMT_REQ_CREDIT_BUREAU_QRT   307511 non-null  int64
 5   AMT_REQ_CREDIT_BUREAU_YEAR  307511 non-null  int64
dtypes: int64(6)
memory usage: 14.1 MB
```

```
col1=["OBS_30_CNT_SOCIAL_CIRCLE","DEF_30_CNT_SOCIAL_CIRCLE",
     "OBS_60_CNT_SOCIAL_CIRCLE","DEF_60_CNT_SOCIAL_CIRCLE"]
for i in col1:
  df1[i] = df1[i].fillna(df1[i].median()).astype(int)
##  these columns are Number of observations of people in the client's
##   social circle with 30 days past due on defaults it shoul be in
integer
## we filled null values with median
```

```
df1["EXT_SOURCE_3"].info()
# It likely represents a risk assessment score from a third-party
source,
# used to evaluate a client's creditworthiness.
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 307511 entries, 0 to 307510
Series name: EXT_SOURCE_3
Non-Null Count   Dtype
--------------   -----
246546 non-null  float64
dtypes: float64(1)
memory usage: 2.3 MB
```

```
df1["EXT_SOURCE_3"].skew()
```

```
-0.4093904596160267
```

```
#we need to impute it with median since it is skewed so we cant rely
on mean

df1["EXT_SOURCE_3"] =
df1["EXT_SOURCE_3"].fillna(df1["EXT_SOURCE_3"].median())
df1["EXT_SOURCE_2"] =
df1["EXT_SOURCE_2"].fillna(df1["EXT_SOURCE_2"].median())
```

```
df1[["AMT_ANNUITY","AMT_GOODS_PRICE","NAME_TYPE_SUITE","OCCUPATION_TYP
E","CNT_FAM_MEMBERS","DAYS_LAST_PHONE_CHANGE"]].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 6 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   AMT_ANNUITY             307499 non-null  float64
 1   AMT_GOODS_PRICE         307233 non-null  float64
 2   NAME_TYPE_SUITE         306219 non-null  object
 3   OCCUPATION_TYPE         211120 non-null  object
 4   CNT_FAM_MEMBERS         307509 non-null  float64
 5   DAYS_LAST_PHONE_CHANGE  307510 non-null  float64
dtypes: float64(4), object(2)
memory usage: 14.1+ MB
```

```
for i in
["AMT_ANNUITY","AMT_GOODS_PRICE","CNT_FAM_MEMBERS","DAYS_LAST_PHONE_CH
ANGE"]:
  df1[i] = df1[i].fillna(df1[i].median()) #we need to impute with
median since columns are skewed we need to impute it with median
df1["OCCUPATION_TYPE"] = df1["OCCUPATION_TYPE"].fillna("Unknown") # as
said earlier we impute this column with unknown
df1["NAME_TYPE_SUITE"] =
```

```
df1["NAME_TYPE_SUITE"].fillna(df1["NAME_TYPE_SUITE"].mode()[0])# For
this column we impute null values with mode

df1.isnull().sum() # Null values are removed and treated
```

```
SK_ID_CURR                     0
TARGET                         0
NAME_CONTRACT_TYPE             0
CODE_GENDER                    0
FLAG_OWN_CAR                   0
FLAG_OWN_REALTY                0
CNT_CHILDREN                   0
AMT_INCOME_TOTAL               0
AMT_CREDIT                     0
AMT_ANNUITY                    0
AMT_GOODS_PRICE                0
NAME_TYPE_SUITE                0
NAME_INCOME_TYPE               0
NAME_EDUCATION_TYPE            0
NAME_FAMILY_STATUS             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
FLAG_EMP_PHONE                 0
FLAG_WORK_PHONE                0
FLAG_CONT_MOBILE               0
FLAG_PHONE                     0
FLAG_EMAIL                     0
OCCUPATION_TYPE                0
CNT_FAM_MEMBERS                0
REGION_RATING_CLIENT           0
REGION_RATING_CLIENT_W_CITY    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
ORGANIZATION_TYPE              0
EXT_SOURCE_2                   0
EXT_SOURCE_3                   0
OBS_30_CNT_SOCIAL_CIRCLE       0
DEF_30_CNT_SOCIAL_CIRCLE       0
OBS_60_CNT_SOCIAL_CIRCLE       0
```

```
DEF_60_CNT_SOCIAL_CIRCLE       0
DAYS_LAST_PHONE_CHANGE         0
FLAG_DOCUMENT_2                0
FLAG_DOCUMENT_3                0
FLAG_DOCUMENT_4                0
FLAG_DOCUMENT_5                0
FLAG_DOCUMENT_6                0
FLAG_DOCUMENT_7                0
FLAG_DOCUMENT_8                0
FLAG_DOCUMENT_9                0
FLAG_DOCUMENT_10               0
FLAG_DOCUMENT_11               0
FLAG_DOCUMENT_12               0
FLAG_DOCUMENT_13               0
FLAG_DOCUMENT_14               0
FLAG_DOCUMENT_15               0
FLAG_DOCUMENT_16               0
FLAG_DOCUMENT_17               0
FLAG_DOCUMENT_18               0
FLAG_DOCUMENT_19               0
FLAG_DOCUMENT_20               0
FLAG_DOCUMENT_21               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
AMT_REQ_CREDIT_BUREAU_YEAR     0
dtype: int64
```

Check for Duplicates

```
df1.duplicated().sum() #there are no duplicated in the dataset
```

```
0
```

```
# In these columns we can see the negative values in the DAYS columns

df1[['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH
','DAYS_LAST_PHONE_CHANGE']].describe()
```


```
{"summary":"{\n  \"name\": \"df1[['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH','DAYS_LAST_PHONE_CHANGE']]\",\n  \"rows\": 8,\n  \"fields\": [\n  {\n  \"column\": \"DAYS_BIRTH\",\n  \"properties\": {\n  \"dtype\": \"number\",\n  \"std\": 113721.74256664132,\n  \"min\": -25229.0,\n  \"max\": 307511.0,\n  \"num_unique_values\": 8,\n  \"samples\": [\n  -16036.995066843137,\n  -15750.0,\n  307511.0\n  ],\n  \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"DAYS_EMPLOYED\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 151565.6912219694,\n   \"min\": -17912.0,\n   \"max\": 365243.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   63815.04590404896,\n   -1213.0,\n   307511.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"DAYS_REGISTRATION\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 111067.93567402713,\n   \"min\": -24672.0,\n   \"max\": 307511.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   -4986.120327538419,\n   -4504.0,\n   307511.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"DAYS_ID_PUBLISH\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 109660.34010110334,\n   \"min\": -7197.0,\n   \"max\": 307511.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   -2994.2023732484367,\n   -3254.0,\n   307511.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"DAYS_LAST_PHONE_CHANGE\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 109087.11125815548,\n   \"min\": -4292.0,\n   \"max\": 307511.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   -962.8581188965597,\n   -757.0,\n   307511.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
# We need to convert it into absolute values '
# Make a list of all DAYS columns
col_list = ['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH','DAYS_LAST_PHONE_CHANGE']

# Replace the values with their respective absolute values
for i in col_list:
    df1[i] = abs(df1[i])

# Verify the changes
df1[['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH','DAYS_LAST_PHONE_CHANGE']].describe()
```

{"summary":"{\n \"name\": \"df1[['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH','DAYS_LAST_PHONE_CHANGE']]\",\n \"rows\": 8,\n \"fields\": [\n {\n   \"column\": \"DAYS_BIRTH\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 103830.13734330218,\n   \"min\": 4363.988631785526,\n   \"max\": 307511.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   16036.995066843137,\n   15750.0,\n   307511.0\

n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"DAYS_EMPLOYED\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 147983.32970123301,\n   \"min\":
0.0,\n   \"max\": 365243.0,\n   \"num_unique_values\": 8,\n
\"samples\": [\n   67724.74214906133,\n   2219.0,\n
307511.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"DAYS_REGISTRATION\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 106612.80932614328,\n   \"min\":
0.0,\n   \"max\": 307511.0,\n   \"num_unique_values\": 8,\n
\"samples\": [\n   4986.120327538419,\n   4504.0,\n
307511.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"DAYS_ID_PUBLISH\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 107683.60602472254,\n   \"min\":
0.0,\n   \"max\": 307511.0,\n   \"num_unique_values\": 8,\n
\"samples\": [\n   2994.2023732484367,\n   3254.0,\n
307511.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"DAYS_LAST_PHONE_CHANGE\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 108291.1788159357,\n
\"min\": 0.0,\n   \"max\": 307511.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
962.8581188965597,\n   757.0,\n   307511.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

```
df1['CODE_GENDER'].value_counts() # there are 4 records with XNA which
is to be treated
df1['CODE_GENDER'] = df1['CODE_GENDER'].replace('XNA',np.nan)
```

```
df1['CODE_GENDER'].value_counts()
```

```
CODE_GENDER
F    202448
M    105059
Name: count, dtype: int64
```

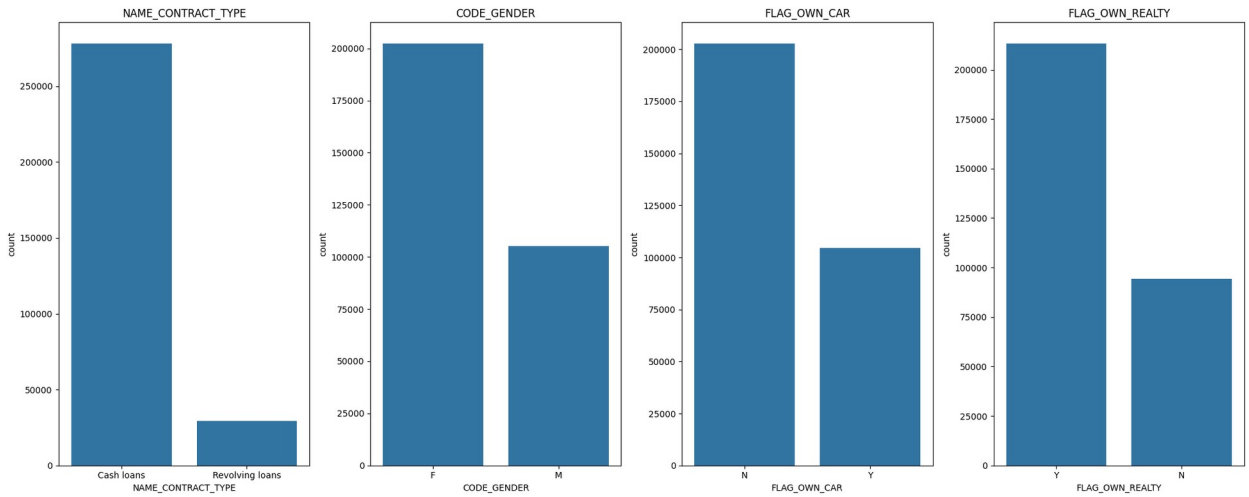
```
df1['CODE_GENDER']=df1['CODE_GENDER'].fillna(df1['CODE_GENDER'].mode()
[0])
```

```
plt.rcParams["figure.figsize"] = (5, 5)
```

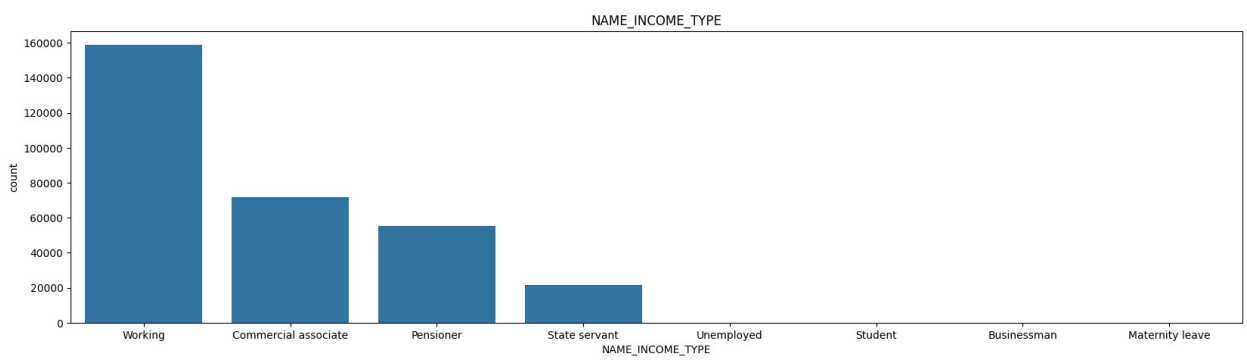
Univariate Analysis

```
plt.figure(figsize =[20,8])
j=1
for i in df1[['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR',
'FLAG_OWN_REALTY']]:
```

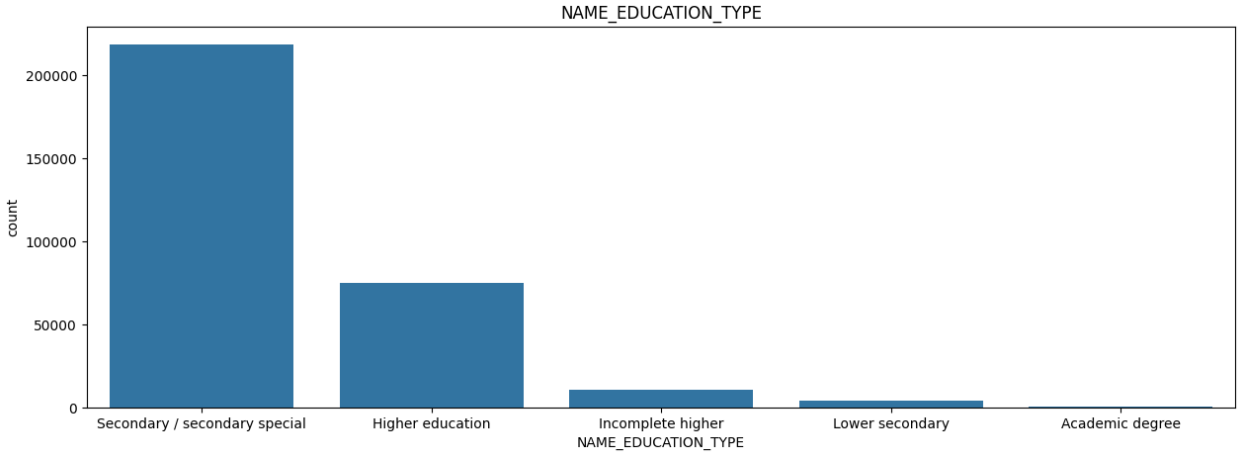
```
    plt.subplot(1,4,j)
    sns.barplot(df1[i].value_counts())
    plt.title(i)
    plt.xticks(rotation=0)
    j += 1
plt.tight_layout()
```



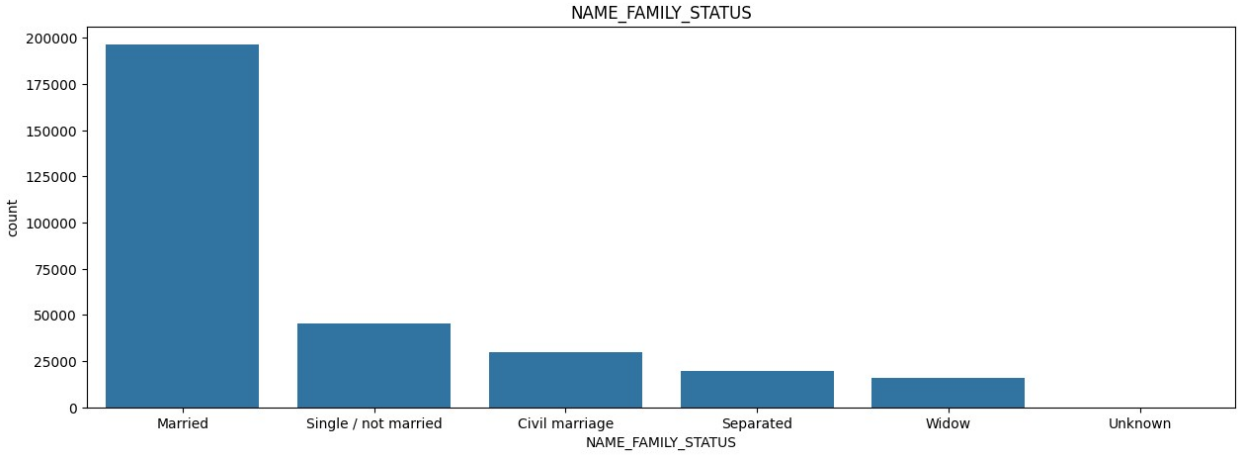

```
plt.figure(figsize =[20,5])
sns.barplot(df1['NAME_INCOME_TYPE'].value_counts())
plt.title('NAME_INCOME_TYPE')
plt.show()# Most clients income is by working
```



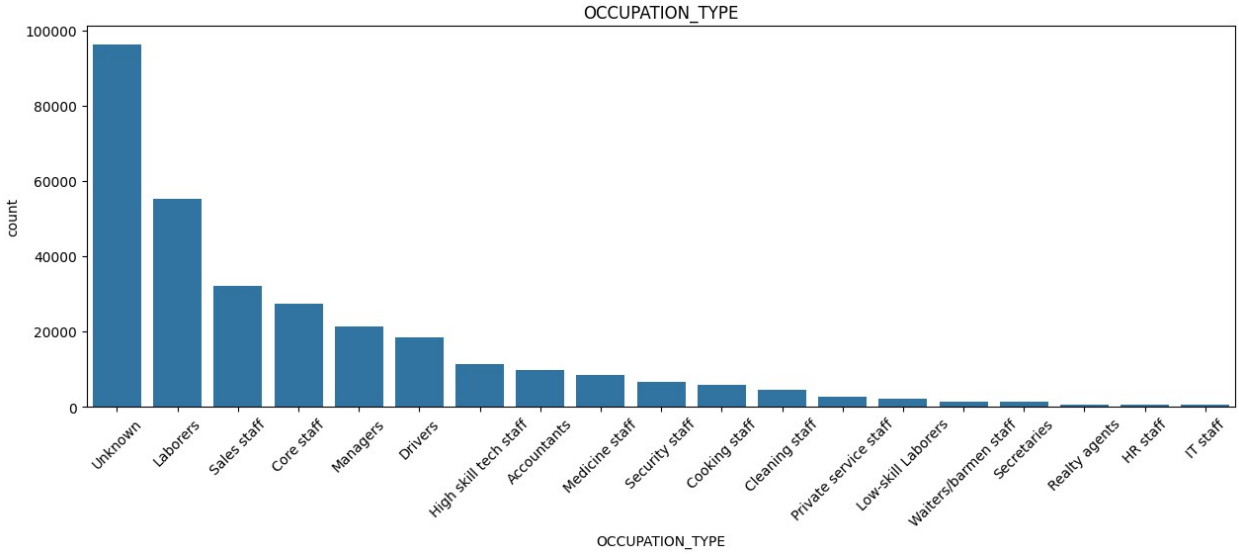

```
plt.figure(figsize =[15,5])
sns.barplot(df1['NAME_EDUCATION_TYPE'].value_counts())
plt.title('NAME_EDUCATION_TYPE')
plt.show()# Most clients education is secondary
```



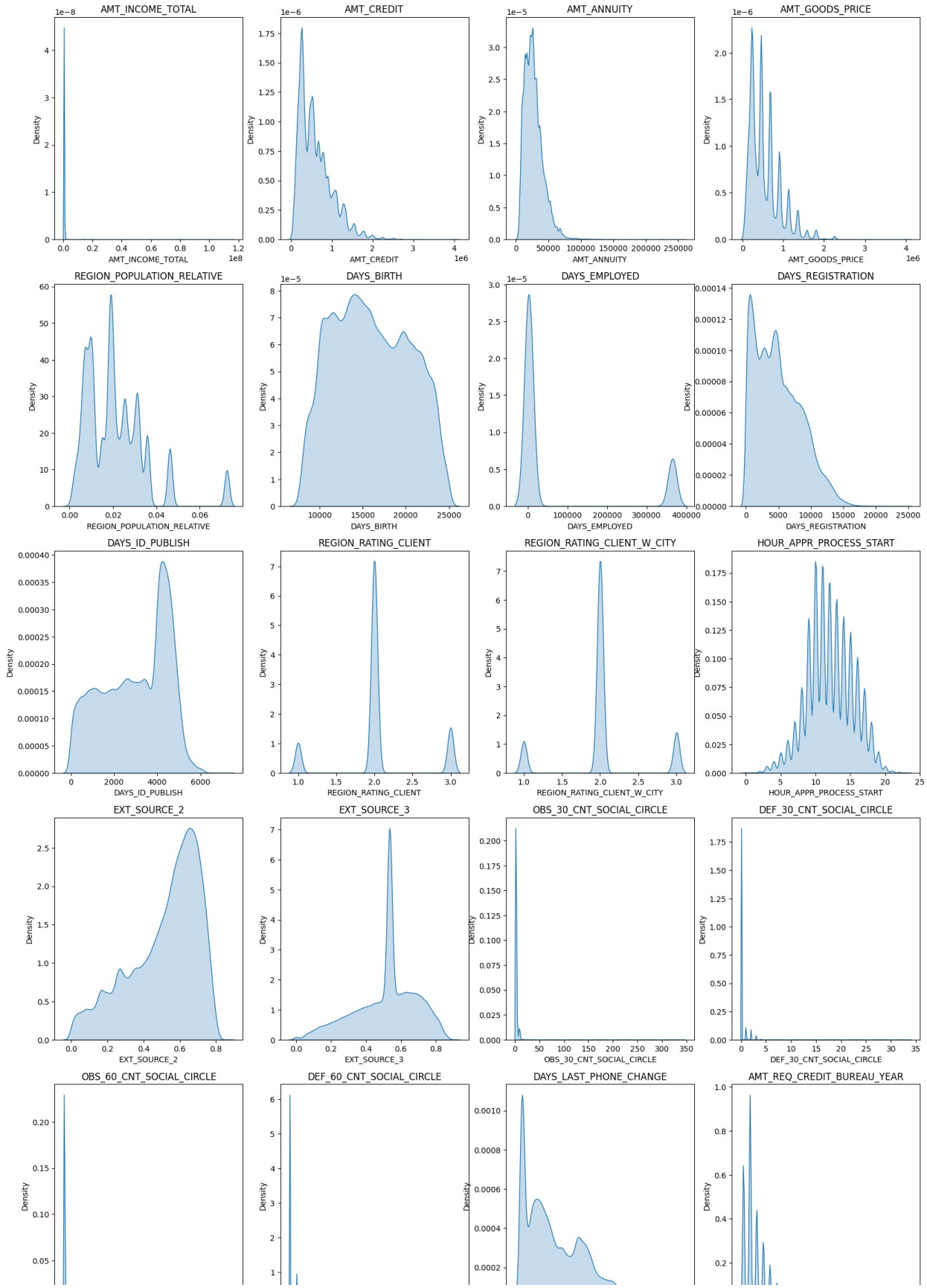

```
plt.figure(figsize =[15,5])
sns.barplot(df1['NAME_FAMILY_STATUS'].value_counts())
plt.title('NAME_FAMILY_STATUS')
plt.show()# Most clients are married
```



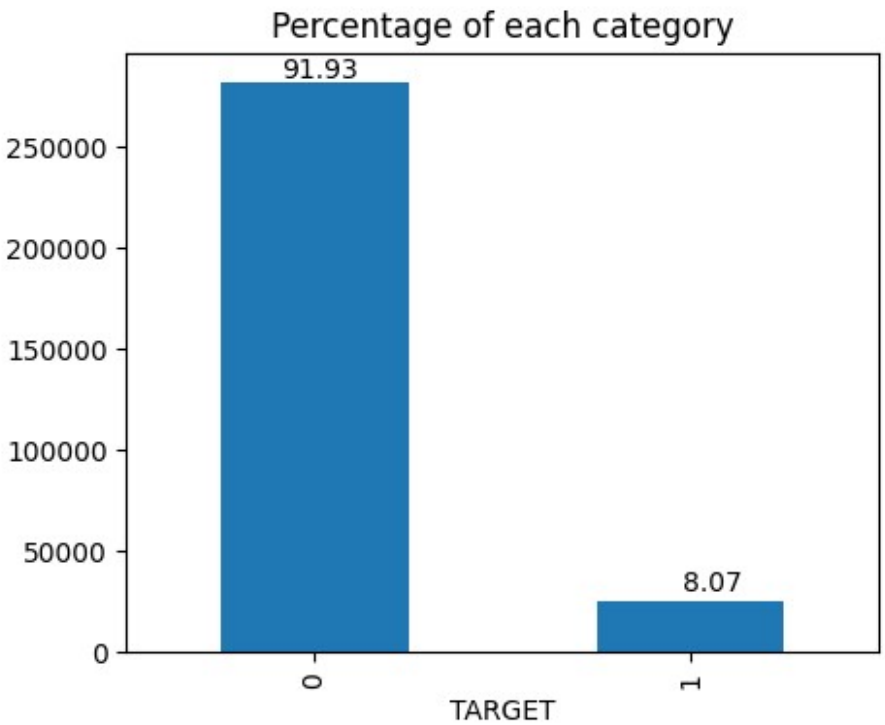

```
plt.figure(figsize =[15,5])
sns.barplot(df1['OCCUPATION_TYPE'].value_counts())
plt.title('OCCUPATION_TYPE')
plt.xticks(rotation=45)
plt.show()# most clients occupation type is labores
```




```
col=["AMT_INCOME_TOTAL","AMT_CREDIT","AMT_ANNUITY","AMT_GOODS_PRICE","
REGION_POPULATION_RELATIVE","DAYS_BIRTH",

"DAYS_EMPLOYED","DAYS_REGISTRATION","DAYS_ID_PUBLISH","REGION_RATING_C
LIENT",
     "REGION_RATING_CLIENT_W_CITY",   "HOUR_APPR_PROCESS_START",
     "EXT_SOURCE_2", "EXT_SOURCE_3",

"OBS_30_CNT_SOCIAL_CIRCLE","DEF_30_CNT_SOCIAL_CIRCLE","OBS_60_CNT_SOCI
AL_CIRCLE","DEF_60_CNT_SOCIAL_CIRCLE","DAYS_LAST_PHONE_CHANGE","AMT_RE
Q_CREDIT_BUREAU_YEAR"]
plt.figure(figsize =[20,30])
j=1
for i in col:
    plt.subplot(5,4,j)
    sns.kdeplot(df1[i],fill =True)
    plt.title(i)
    j+=1
# 1.AMT_INCOME_TOTAL column has high skewness. There is influence of
ouliers here
# 2.
"AMT_CREDIT","AMT_ANNUITY","AMT_GOODS_PRICE","REGION_POPULATION_RELATI
VE" columns are left skewed
# 3.
OBS_30_CNT_SOCIAL_CIRCLE,DEF_30_CNT_SOCIAL_CIRCLE,OBS_60_CNT_SOCIAL_CI
RCLE,DEF_60_CNT_SOCIAL_CIRCLE these columns are skewed highly which
indicates of outliers
# 4. remaing columns are having less skewness nearer to zero
```


AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
REGION_POPULATION_RELATIVE
DAYS_BIRTH
DAYS_EMPLOYED
DAYS_REGISTRATION
DAYS_ID_PUBLISH
REGION_RATING_CLIENT
REGION_RATING_CLIENT_W_CITY
HOUR_APPR_PROCESS_START
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
AMT_REQ_CREDIT_BUREAU_YEAR
Density

```
plt.figure(figsize=[5,4])
df1["TARGET"].value_counts().plot(kind="bar",)
plt.text(x=-0.087, y=285000,s=round(df1["TARGET"].value_counts()
[0]*100/df1["TARGET"].shape[0],2))
plt.text(x=0.98, y=30000,s=round(df1["TARGET"].value_counts()
[1]*100/df1["TARGET"].shape[0],2))
plt.title("Percentage of each category")
plt.show()

# # There is imbalance in the data'
# need to treat imbalance data
```

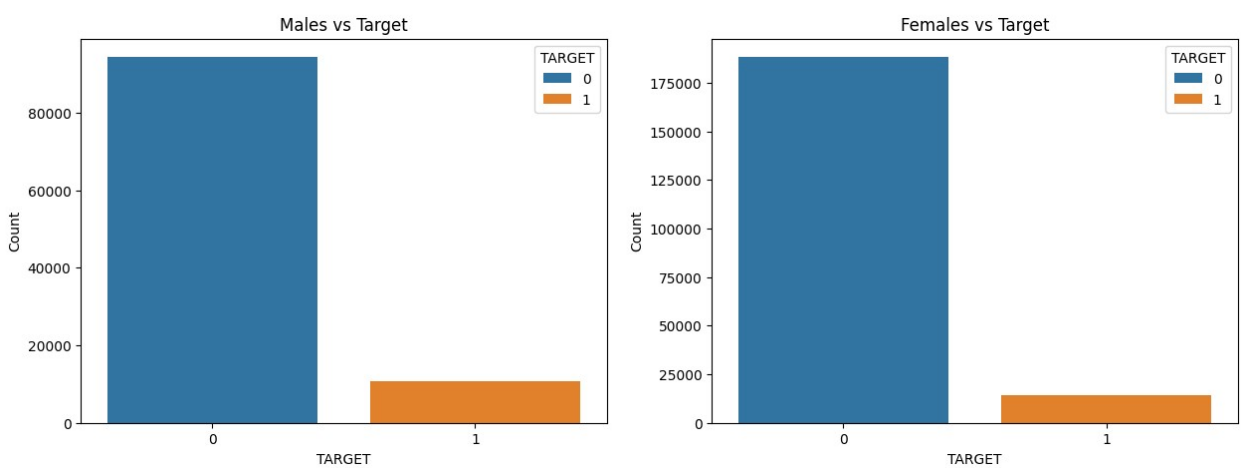


Bivariate Analysis

```
# Code-gender vs Target
import seaborn as sns
import matplotlib.pyplot as plt

# Code-gender vs Target
plt.figure(figsize=[15, 5])

plt.subplot(1, 2, 1)
# Create a DataFrame for male data
male_data = df1[df1["CODE_GENDER"] == "M"]
["TARGET"].value_counts().reset_index()
male_data.columns = ['TARGET', 'Count']
```

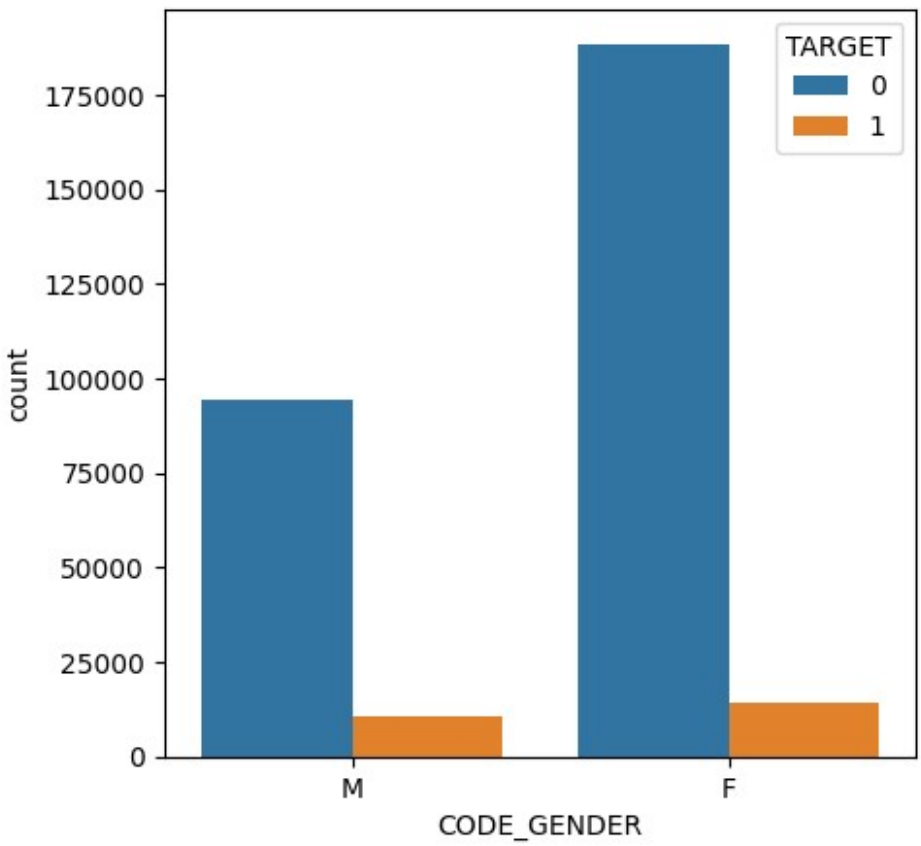
```
sns.barplot(x='TARGET', y='Count', data=male_data, hue='TARGET')
plt.title("Males vs Target")

plt.subplot(1, 2, 2)
Female_data = df1[df1["CODE_GENDER"] == "F"]
["TARGET"].value_counts().reset_index()
Female_data.columns = ['TARGET', 'Count']
sns.barplot(x='TARGET', y='Count', data=Female_data, hue='TARGET')
plt.title("Females vs Target")
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)

Text(0.5, 1.0, 'Females vs Target')
```

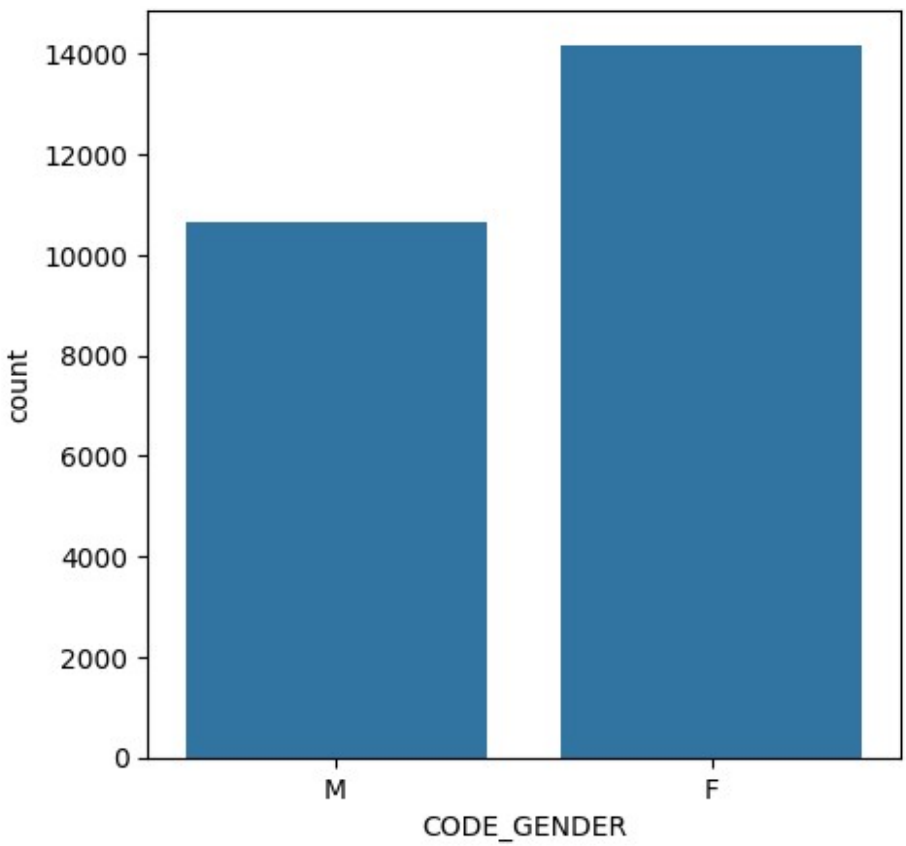



```
sns.countplot(x="CODE_GENDER",hue="TARGET",data=df1)
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)

<Axes: xlabel='CODE_GENDER', ylabel='count'>
```

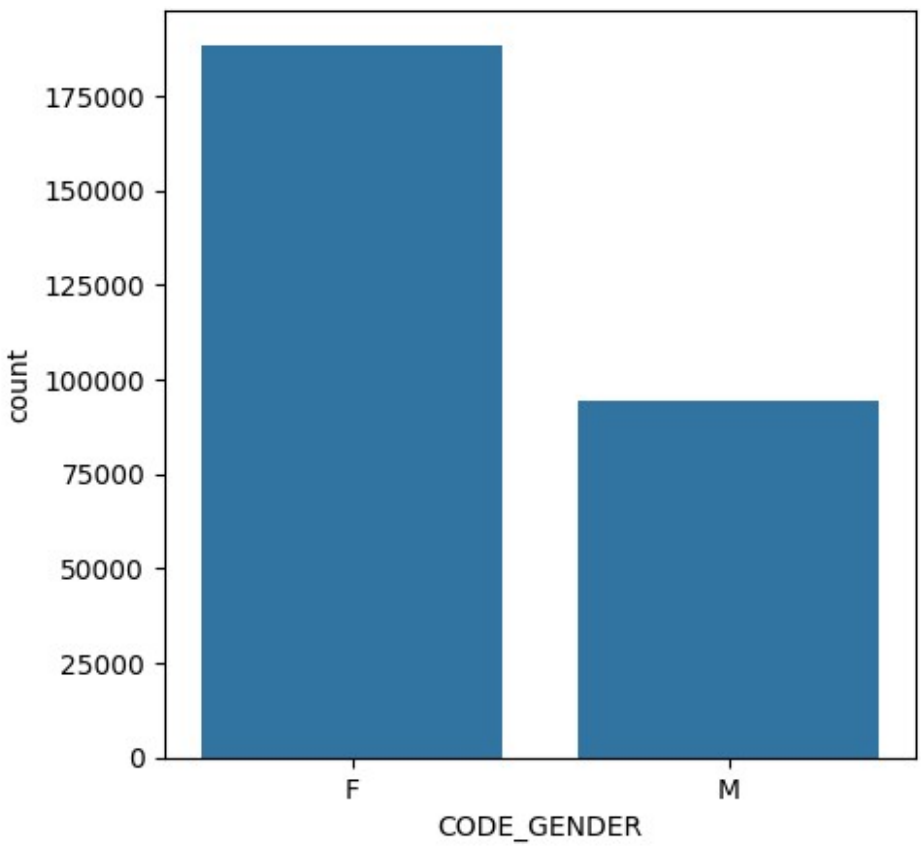



```
defaults=df1[df1["TARGET"]==1]
others=df1[df1["TARGET"]==0]
sns.countplot(data=defaults,x="CODE_GENDER")

<Axes: xlabel='CODE_GENDER', ylabel='count'>
```

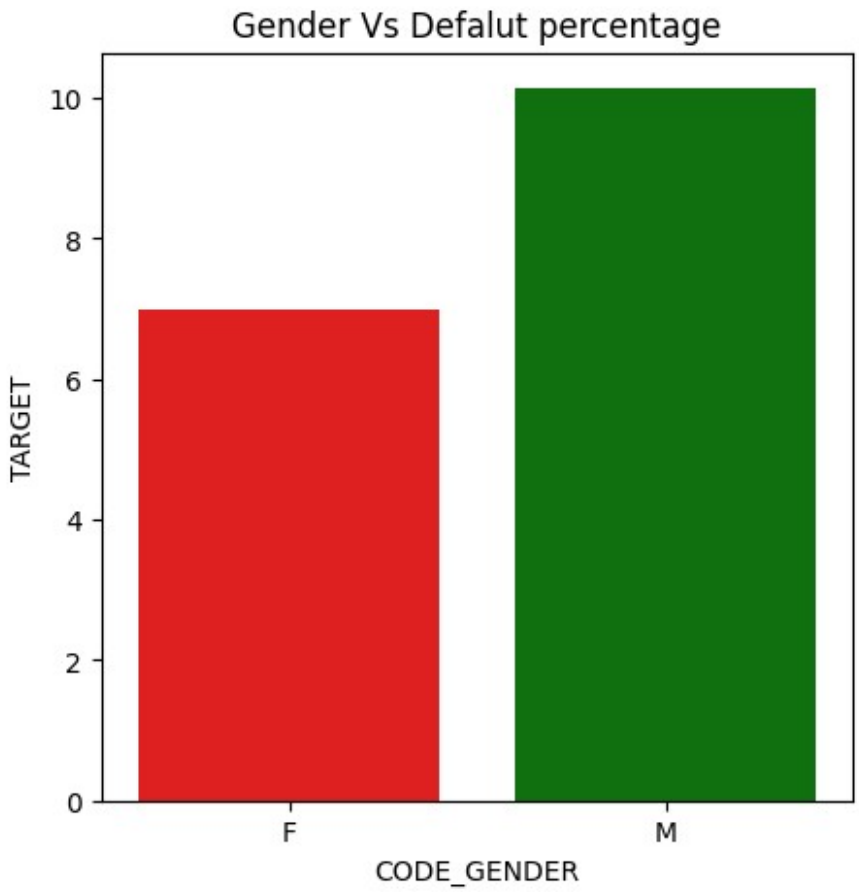



```
sns.countplot(data=others,x="CODE_GENDER")
```

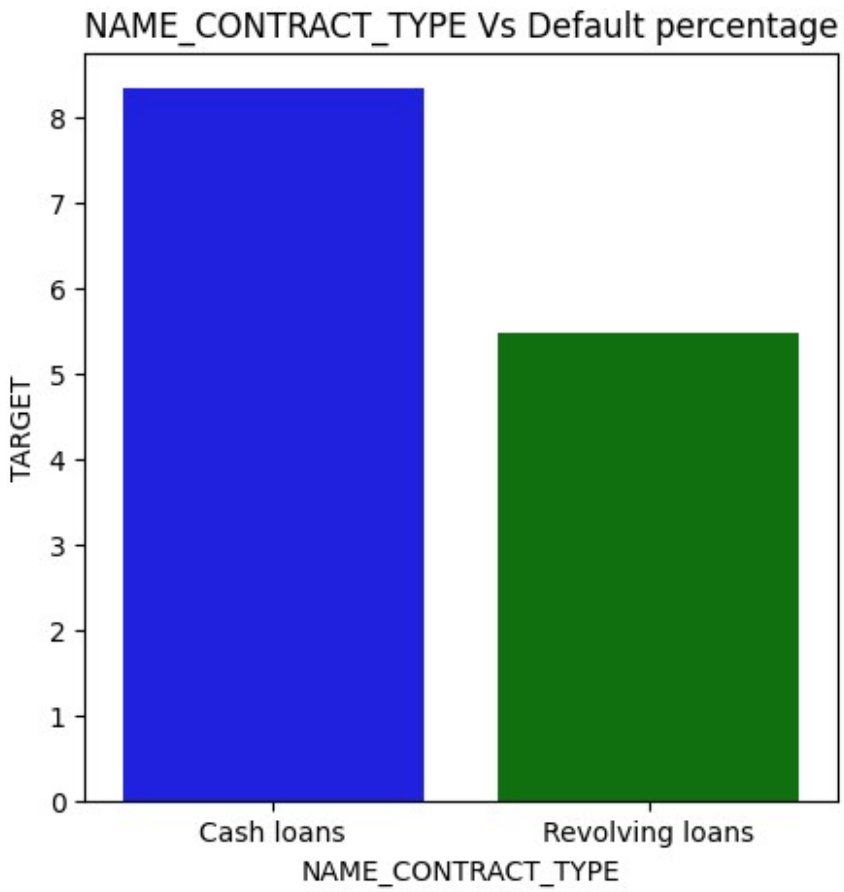
```
<Axes: xlabel='CODE_GENDER', ylabel='count'>
```



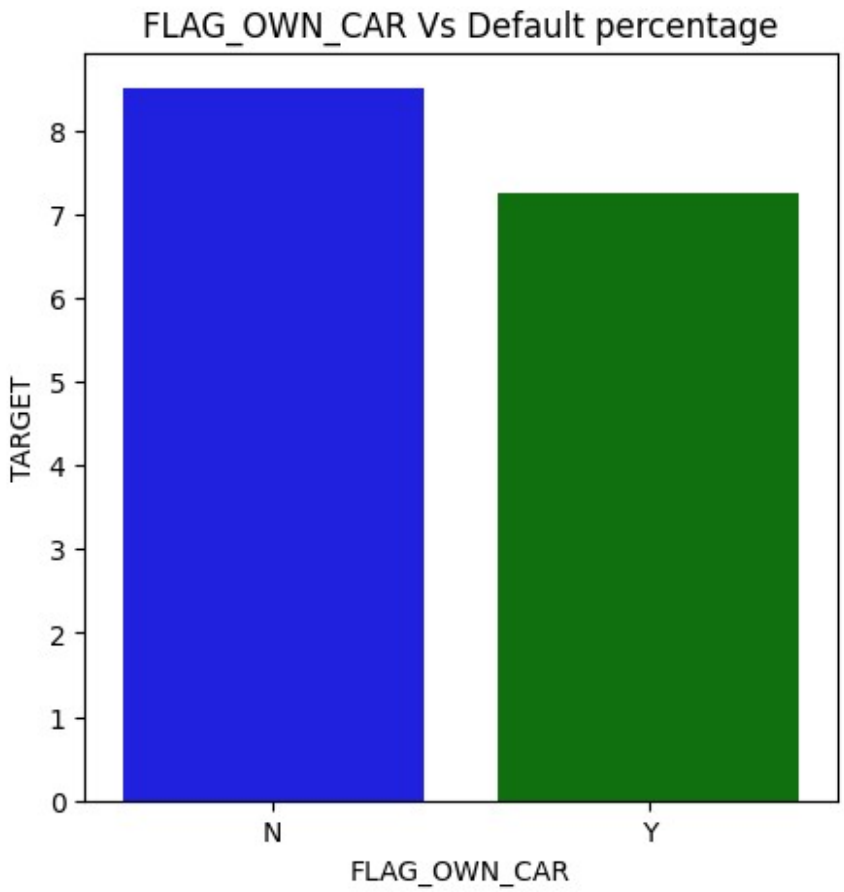

```
import warnings
warnings.filterwarnings("ignore")

tempdf = df1[["TARGET","CODE_GENDER"]].groupby(["CODE_GENDER"],
as_index=False).mean()
tempdf["TARGET"] = tempdf["TARGET"]*100
sns.barplot(x="CODE_GENDER", y = "TARGET", data = tempdf,palette=["r",
"g"])
plt.title("Gender Vs Defalut percentage")
plt.show()
#As seen from the above diagram, males have comparatively higher
percentage of being defaulters than females.
```



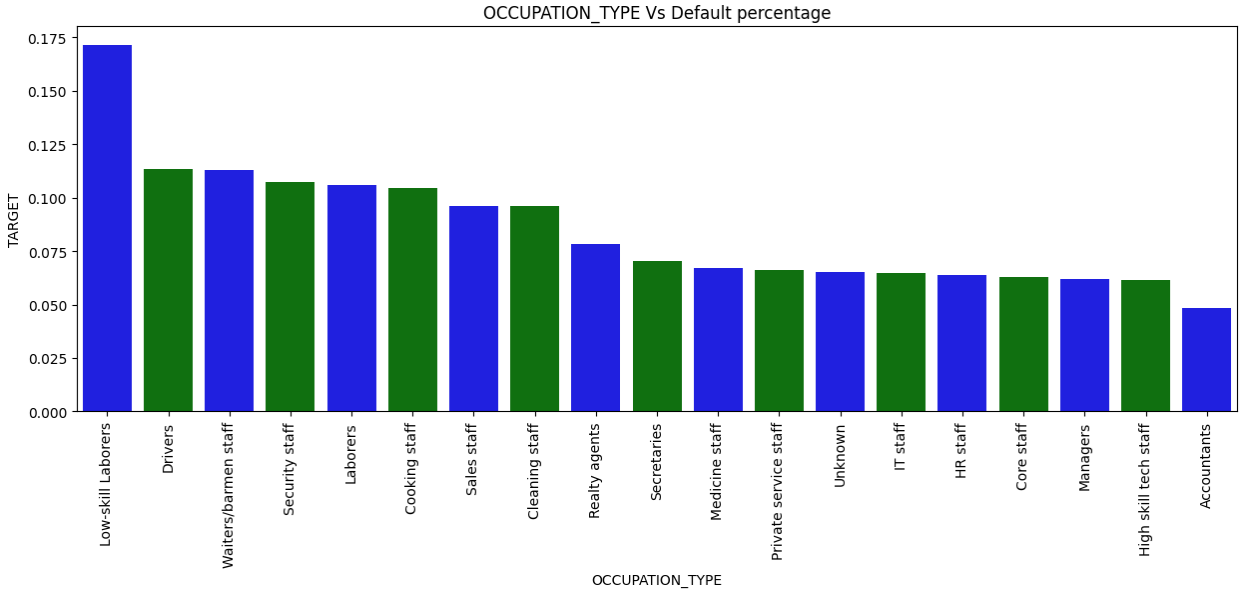

```
df1.select_dtypes(include="object").columns

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

tempdf =
df1[["TARGET","NAME_CONTRACT_TYPE"]].groupby(["NAME_CONTRACT_TYPE"],
as_index=False).mean()
tempdf["TARGET"] = tempdf["TARGET"]*100
sns.barplot(x="NAME_CONTRACT_TYPE", y = "TARGET", data =
tempdf,palette=["b", "g"])
plt.title("NAME_CONTRACT_TYPE Vs Default percentage")
plt.show()
#As seen from the below diagram, cash loans have comparatively higher
percentage of being defaulters than Revovling loans.
```



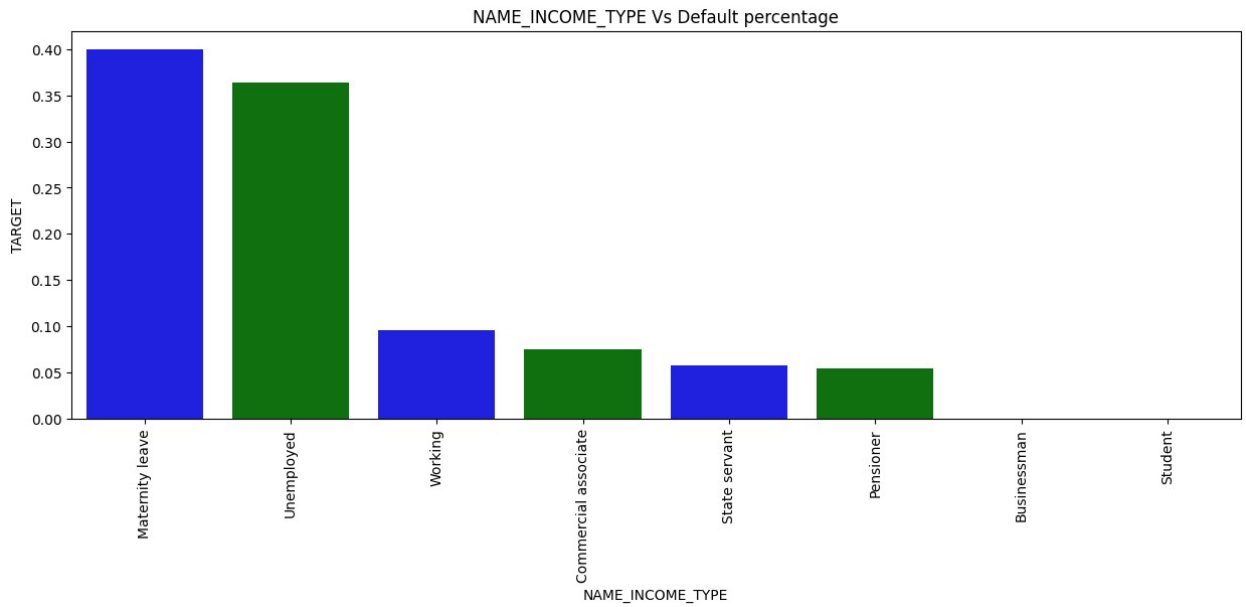

```
tempdf = df1[["TARGET","FLAG_OWN_CAR"]].groupby(["FLAG_OWN_CAR"],
as_index=False).mean()
tempdf["TARGET"] = tempdf["TARGET"]*100
sns.barplot(x="FLAG_OWN_CAR", y = "TARGET", data =
tempdf,palette=["b", "g"])
plt.title("FLAG_OWN_CAR Vs Default percentage")
plt.show()
#As seen from the below diagram, clients with no car have
comparatively higher percentage of being defaulters than the clients
who have car.
```



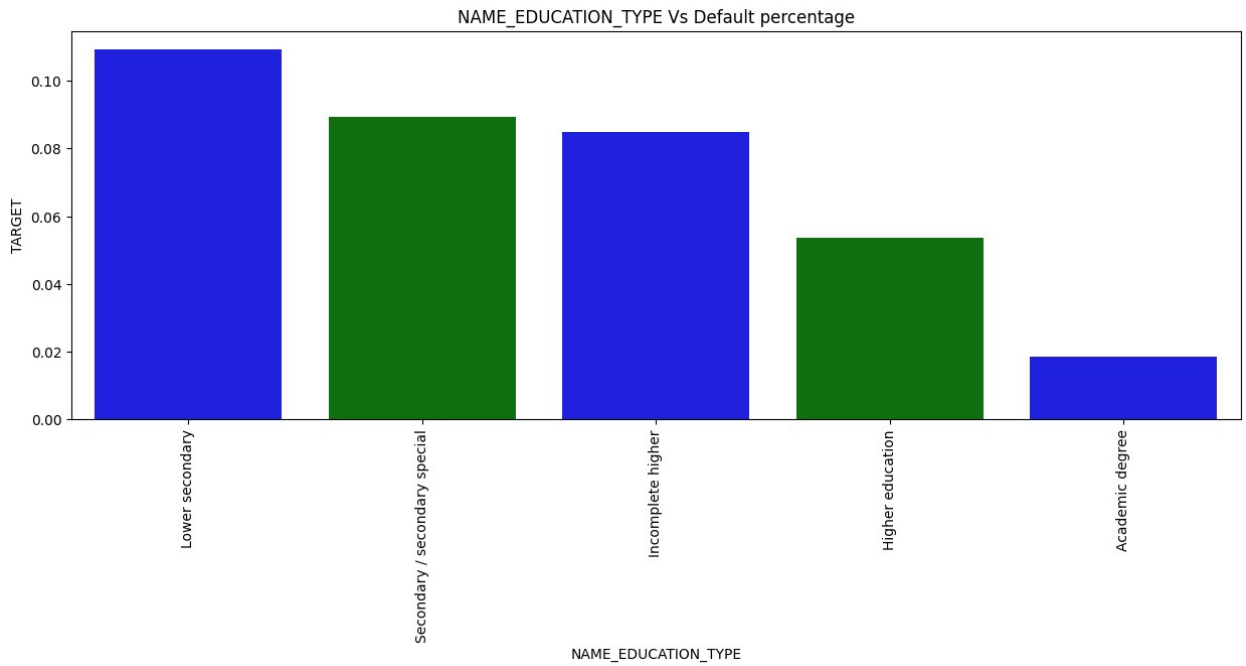

```
plt.figure(figsize=[15, 5])
tempdf =
df1[["TARGET","OCCUPATION_TYPE"]].groupby(["OCCUPATION_TYPE"],
as_index=False).mean()
tempdf.sort_values(by="TARGET", ascending=False, inplace=True)
sns.barplot(x="OCCUPATION_TYPE", y = "TARGET", data =
tempdf,palette=["b", "g"])
plt.title("OCCUPATION_TYPE Vs Default percentage")
plt.xticks(rotation=90)
plt.show()
#As seen from the below diagram we can say that low skill labors are
high chance of being defaulters.
```



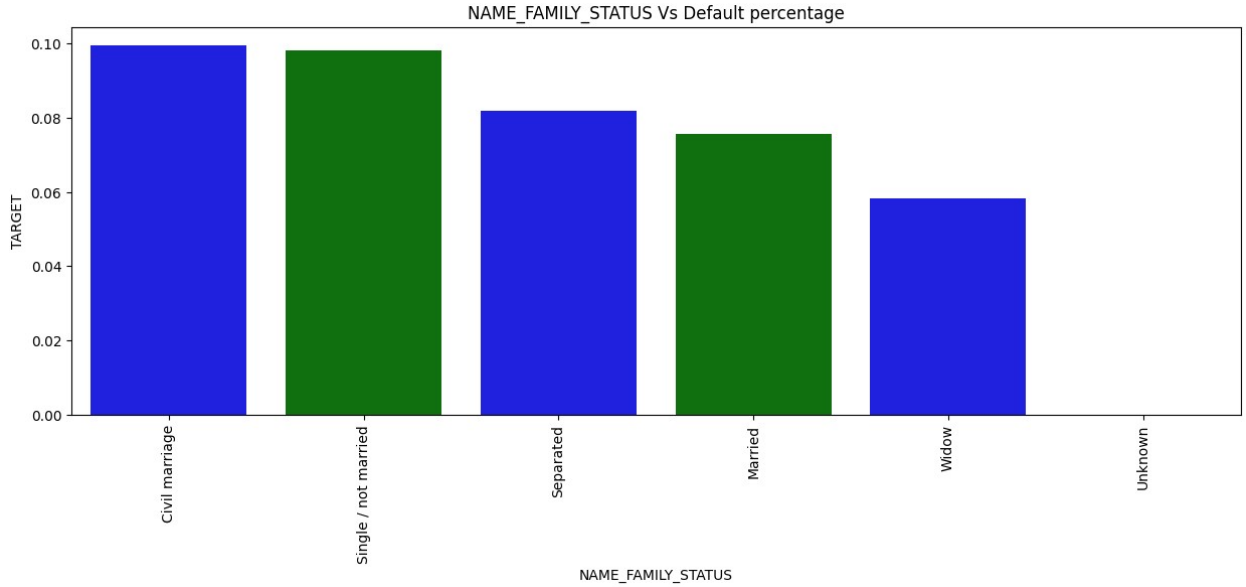

```
plt.figure(figsize=[15, 5])
tempdf =
df1[["TARGET","NAME_INCOME_TYPE"]].groupby(["NAME_INCOME_TYPE"],
as_index=False).mean()
tempdf.sort_values(by="TARGET", ascending=False, inplace=True)
sns.barplot(x="NAME_INCOME_TYPE", y = "TARGET", data =
tempdf,palette=["b", "g"])
plt.title("NAME_INCOME_TYPE Vs Default percentage")
plt.xticks(rotation=90)
plt.show()
# The people on Maternity leave and unemployed categories are more
likely to fail to repay.
# Businessman and Students have the lowest chances of defaulting.
```



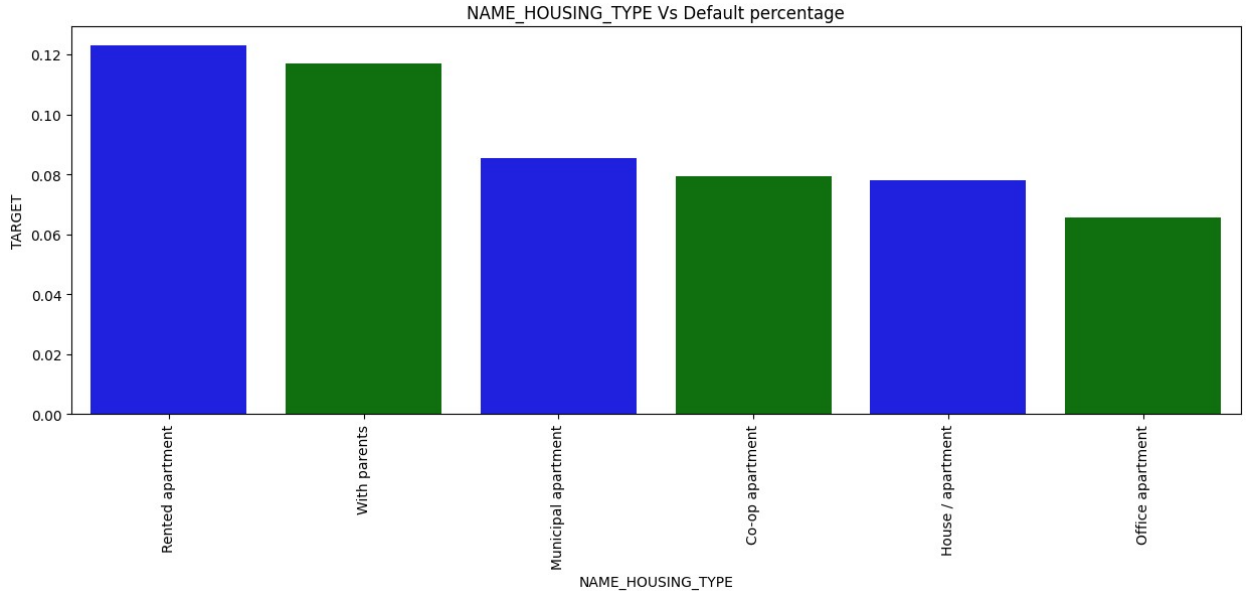

```
plt.figure(figsize=[15, 5])
tempdf =
df1[["TARGET","NAME_EDUCATION_TYPE"]].groupby(["NAME_EDUCATION_TYPE"],
as_index=False).mean()
tempdf.sort_values(by="TARGET", ascending=False, inplace=True)
sns.barplot(x="NAME_EDUCATION_TYPE", y = "TARGET", data =
tempdf,palette=["b", "g"])
plt.title("NAME_EDUCATION_TYPE Vs Default percentage")
plt.xticks(rotation=90)
plt.show()
# People with lower secondary education level are more likely to fail
repayment.
# People with academic degree or higher education, however, are mostly
able to repay on time.
```



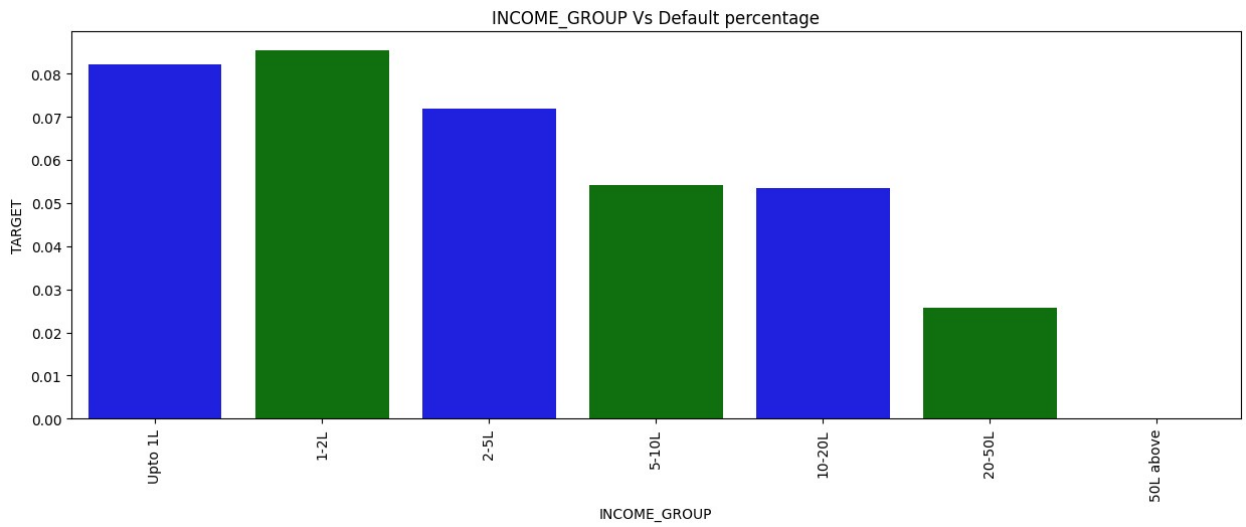

```
plt.figure(figsize=[15, 5])
tempdf =
df1[["TARGET","NAME_FAMILY_STATUS"]].groupby(["NAME_FAMILY_STATUS"],
as_index=False).mean()
tempdf.sort_values(by="TARGET", ascending=False, inplace=True)
sns.barplot(x="NAME_FAMILY_STATUS", y = "TARGET", data =
tempdf,palette=["b", "g"])
plt.title("NAME_FAMILY_STATUS Vs Default percentage")
plt.xticks(rotation=90)
plt.show()
# People in a civil marriage or those who are singles, are more likely
to default.
```



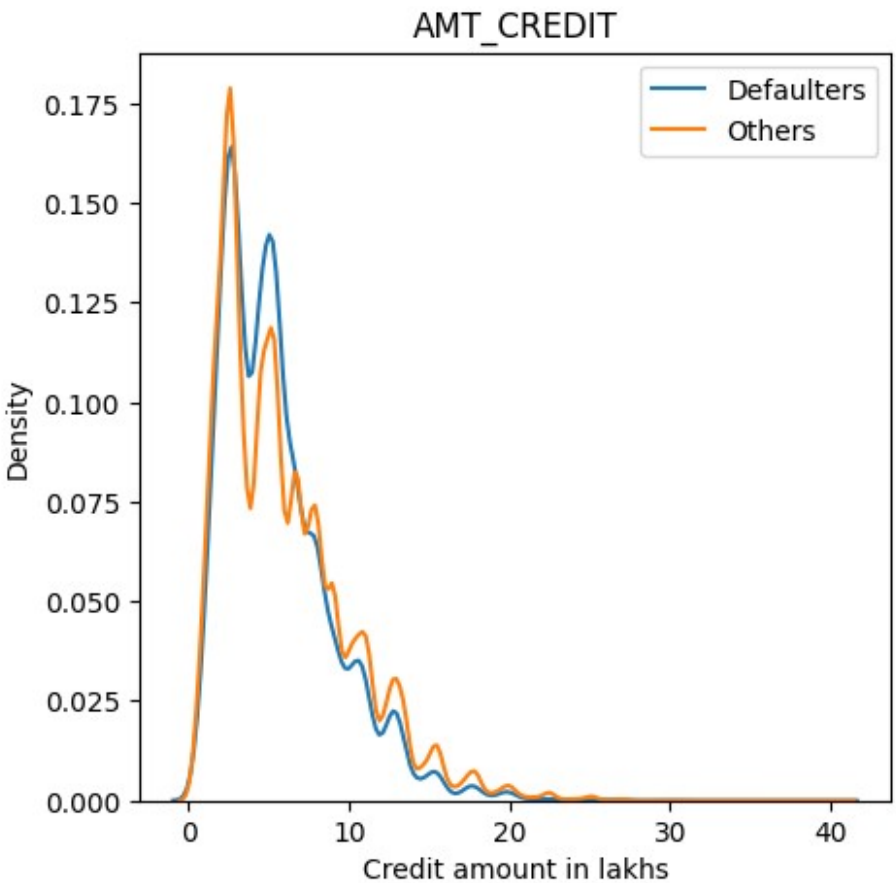

```
plt.figure(figsize=[15, 5])
tempdf =
df1[["TARGET","NAME_HOUSING_TYPE"]].groupby(["NAME_HOUSING_TYPE"],
as_index=False).mean()
tempdf.sort_values(by="TARGET", ascending=False, inplace=True)
sns.barplot(x="NAME_HOUSING_TYPE", y = "TARGET", data =
tempdf,palette=["b", "g"])
plt.title("NAME_HOUSING_TYPE Vs Default percentage")
plt.xticks(rotation=90)
plt.show()
#Most of the applicants who are likely to default are either staying
in a rented apartment or with parents compared to other housing types
```



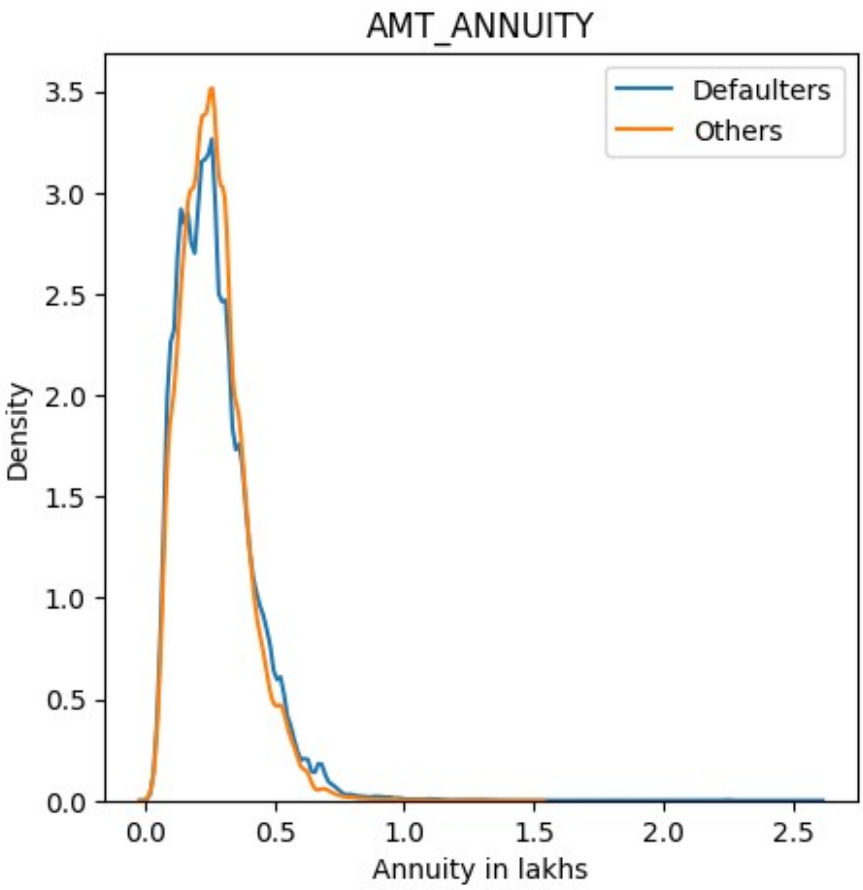
```
bins = [0,1,2,5,10,20,50,1000]
labels = ['Upto 1L','1-2L','2-5L','5-10L','10-20L','20-50L','50L
above']
df1['INCOME_GROUP'] = pd.cut(df1['AMT_INCOME_TOTAL'] / 100000, bins =
bins, labels = labels )

plt.figure(figsize=[15, 5])
tempdf = df1[["TARGET","INCOME_GROUP"]].groupby(["INCOME_GROUP"],
as_index=False).mean()
tempdf.sort_values(by="TARGET", ascending=False, inplace=True)
sns.barplot(x="INCOME_GROUP", y = "TARGET", data =
tempdf,palette=["b", "g"])
plt.title("INCOME_GROUP Vs Default percentage")
plt.xticks(rotation=90)
plt.show()
```



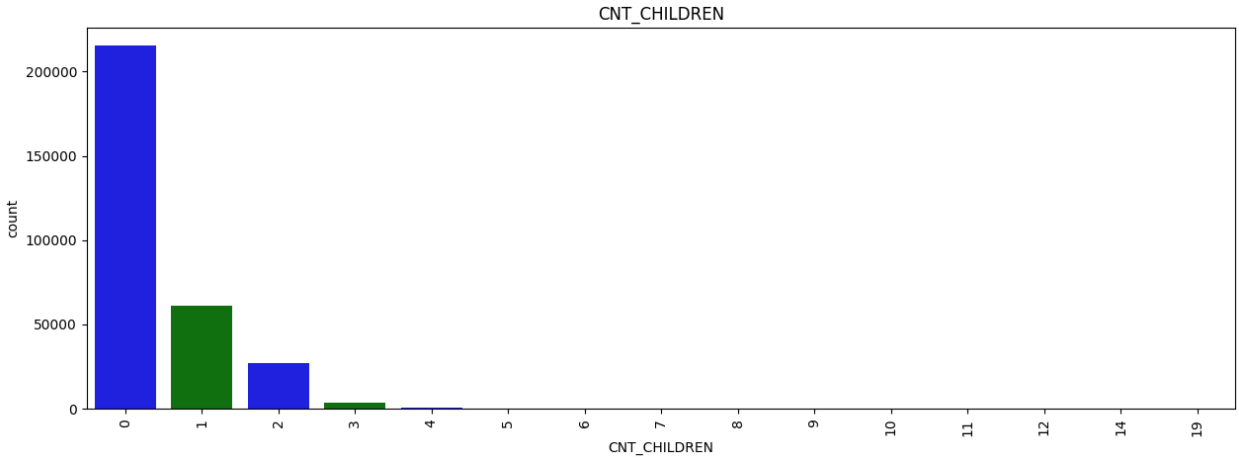

```
## Clients having less income are high likely defaulters

c1=df1[df1["TARGET"]==1]["AMT_CREDIT"]/100000
c2=df1[df1["TARGET"]==0]["AMT_CREDIT"]/100000
sns.kdeplot(c1 , label='Defaulters')
sns.kdeplot(c2 , label='Others')
plt.title("AMT_CREDIT")
plt.xlabel('Credit amount in lakhs')
plt.legend()
plt.show()
```



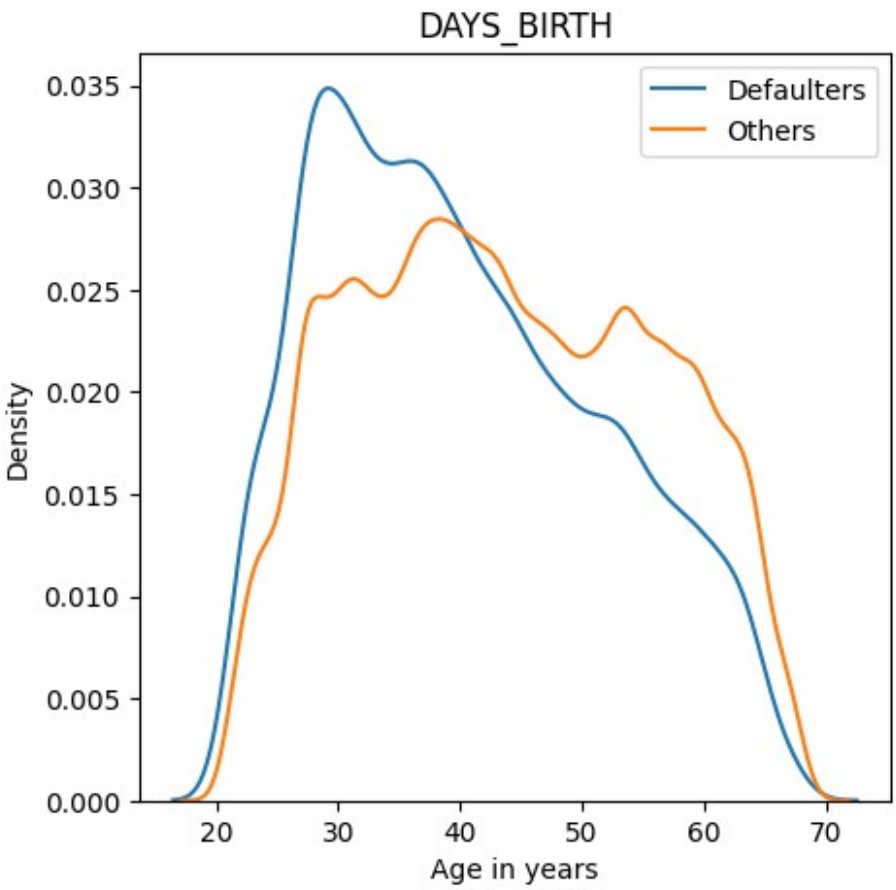

```
#Most of the loans are given with credit amount less than 10 lakhs.

c1=df1[df1["TARGET"]==0]["AMT_ANNUITY"]/100000
c2=df1[df1["TARGET"]==1]["AMT_ANNUITY"]/100000
sns.kdeplot(c1 , label='Defaulters')
sns.kdeplot(c2 , label='Others')
plt.title("AMT_ANNUITY")
plt.xlabel('Annuity in lakhs')
plt.legend()
plt.show()
#Most of the loan annuity is less than 75, 000.
```




```
plt.figure(figsize=[15, 5])
sns.countplot(x="CNT_CHILDREN", data = df1,palette=["b", "g"])
plt.title("CNT_CHILDREN")
plt.xticks(rotation=90)
plt.show()
#Majority of the applicants are having no children. Mostly they have 3
or less children.
```



```
col1 = df1[df1["TARGET"]==1]['DAYS_BIRTH']//365
col2 = df1[df1["TARGET"]==0]['DAYS_BIRTH']//365

sns.kdeplot(col1 , label='Defaulters')
sns.kdeplot(col2 , label='Others')
plt.title("DAYS_BIRTH")
plt.xlabel('Age in years')
plt.legend()
plt.show()
```

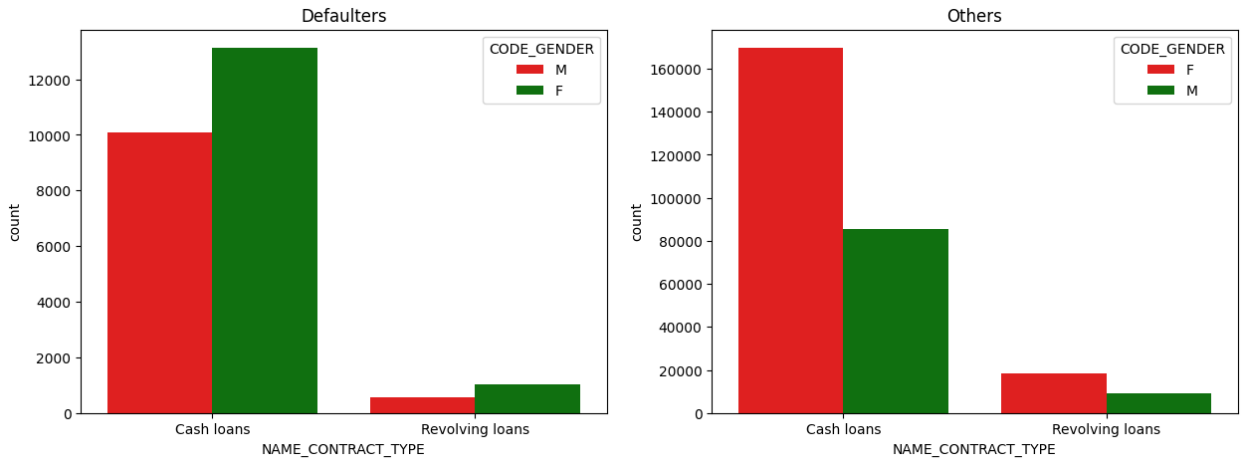



```
#Defaulters are more in 25-40 age group. Above 40, the number of
defaulters tends to decrease.
```

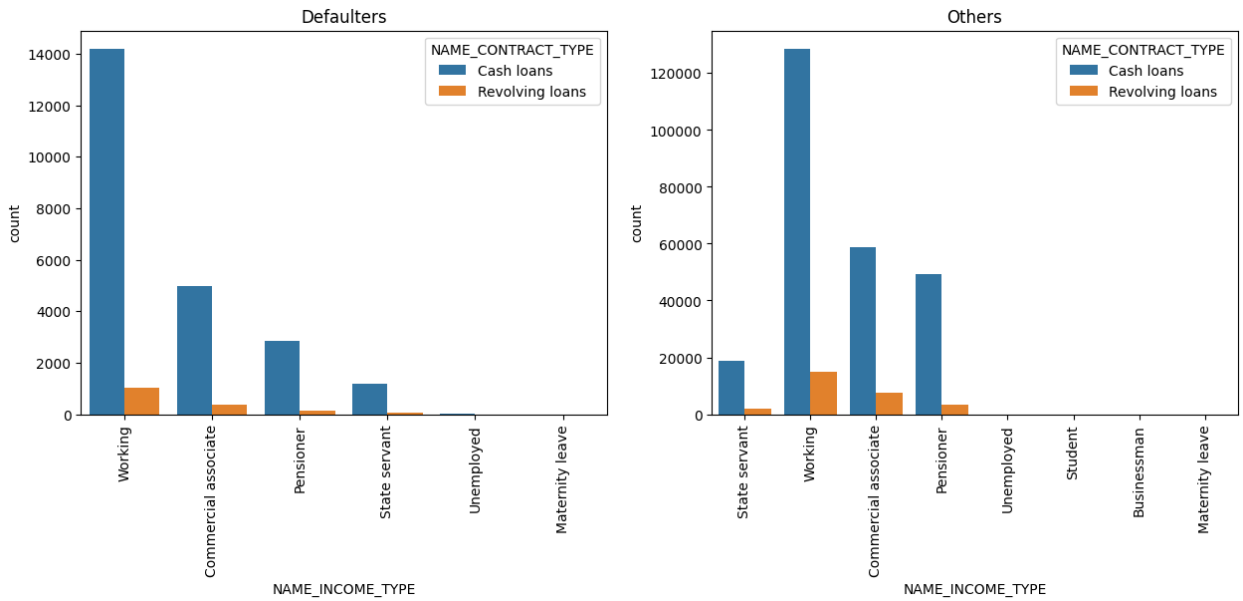
Bivariate Analysis

```
defaults=df1[df1["TARGET"]==1]
others=df1[df1["TARGET"]==0]

# NAME_CONTRACT_TYPE - CODE_GENDER
plt.figure(figsize=[15, 5])
plt.subplot(1,2,1)
```

```
sns.countplot(data=defaults,x="NAME_CONTRACT_TYPE",hue="CODE_GENDER",p
alette=("red","green"))
plt.title("Defaulters")
plt.subplot(1,2,2)
sns.countplot(data=others,x="NAME_CONTRACT_TYPE",hue="CODE_GENDER",pal
ette=("red","green"))
plt.title("Others")
plt.show()
```

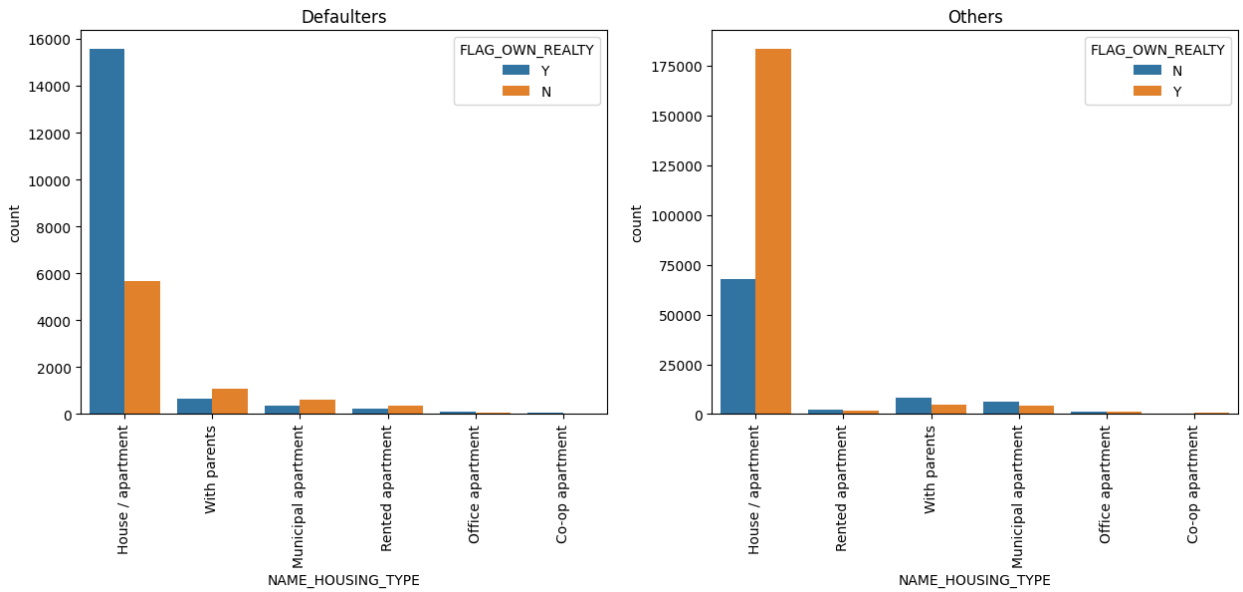



```
# Most of the applications have applied for cash loans.
# Female applicants are more than males.

#NAME_INCOME_TYPE - NAME_CONTRACT_TYPE
plt.figure(figsize=[15, 5])
plt.subplot(1,2,1)
sns.countplot(data=defaults,x="NAME_INCOME_TYPE",hue="NAME_CONTRACT_TY
PE")
plt.title("Defaulters")
plt.xticks(rotation=90)
plt.subplot(1,2,2)
sns.countplot(data=others,x="NAME_INCOME_TYPE",hue="NAME_CONTRACT_TYPE
")
plt.title("Others")
plt.xticks(rotation=90)
plt.show()
```



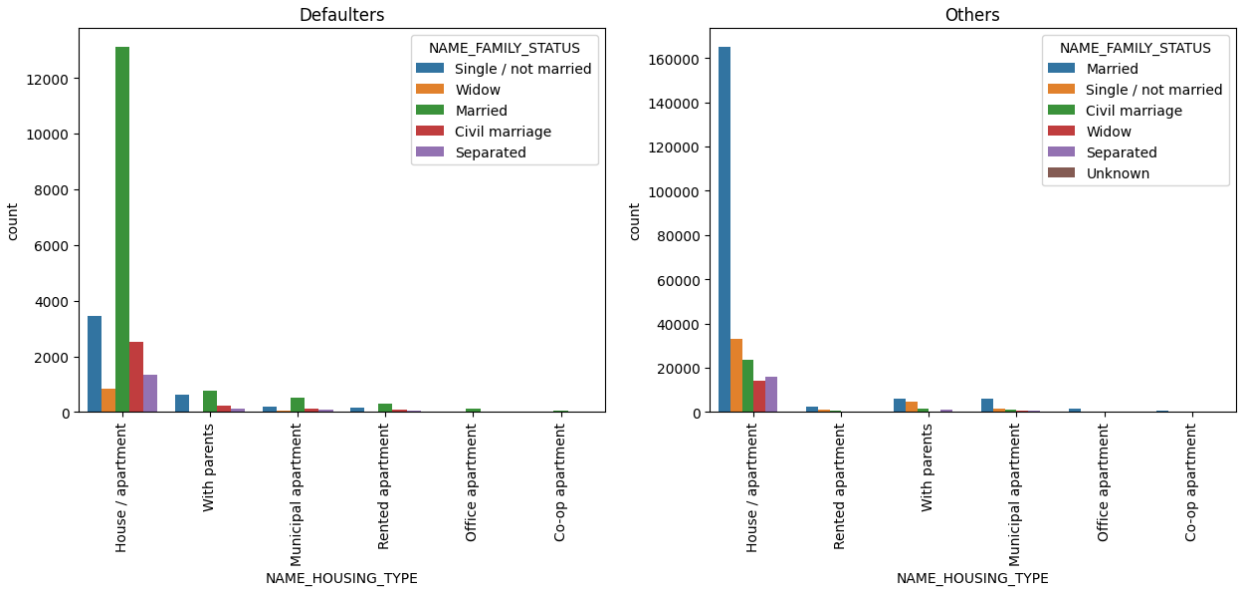
- Across income types, cash loan seems to be the popular contract type.
- Most of the people who have taken loans are working class and they have taken cash loans mostly compared to revolving loans.
- People who have taken cash loans are likely to default as well

```
# NAME_HOUSING_TYPE - FLAG_OWN_REALTY

plt.figure(figsize=[15, 5])
plt.subplot(1,2,1)
sns.countplot(data=defaults,x="NAME_HOUSING_TYPE",hue="FLAG_OWN_REALTY
")
plt.title("Defaulters")
plt.xticks(rotation=90)
plt.subplot(1,2,2)
sns.countplot(data=others,x="NAME_HOUSING_TYPE",hue="FLAG_OWN_REALTY")
plt.title("Others")
plt.xticks(rotation=90)
plt.show()
```



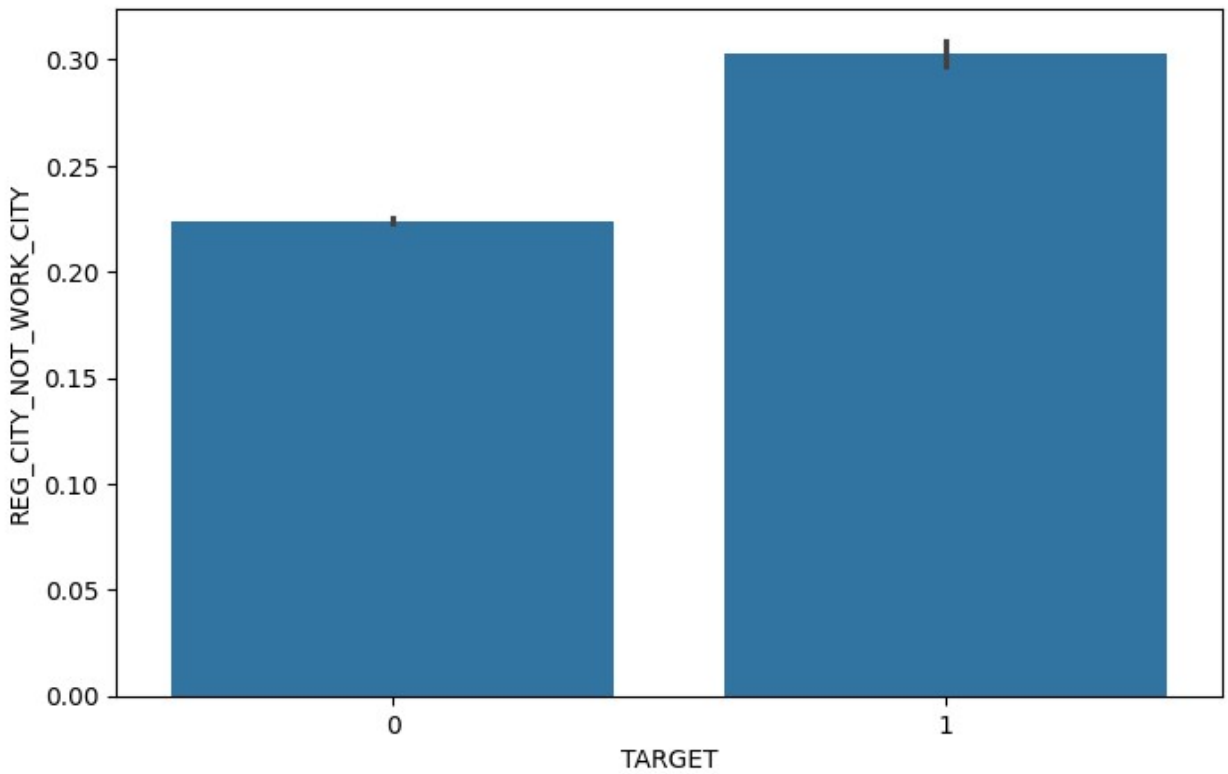
- People who own a house/flat and are staying in own property are likely to make repayments.

```
# NAME_HOUSING_TYPE - NAME_FAMILY_STATUS

plt.figure(figsize=[15, 5])
plt.subplot(1,2,1)
sns.countplot(data=defaults,x="NAME_HOUSING_TYPE",hue="NAME_FAMILY_STA
TUS")
plt.title("Defaulters")
plt.xticks(rotation=90)
plt.subplot(1,2,2)
sns.countplot(data=others,x="NAME_HOUSING_TYPE",hue="NAME_FAMILY_STATU
S",)
plt.title("Others")
plt.xticks(rotation=90)
plt.show()
```

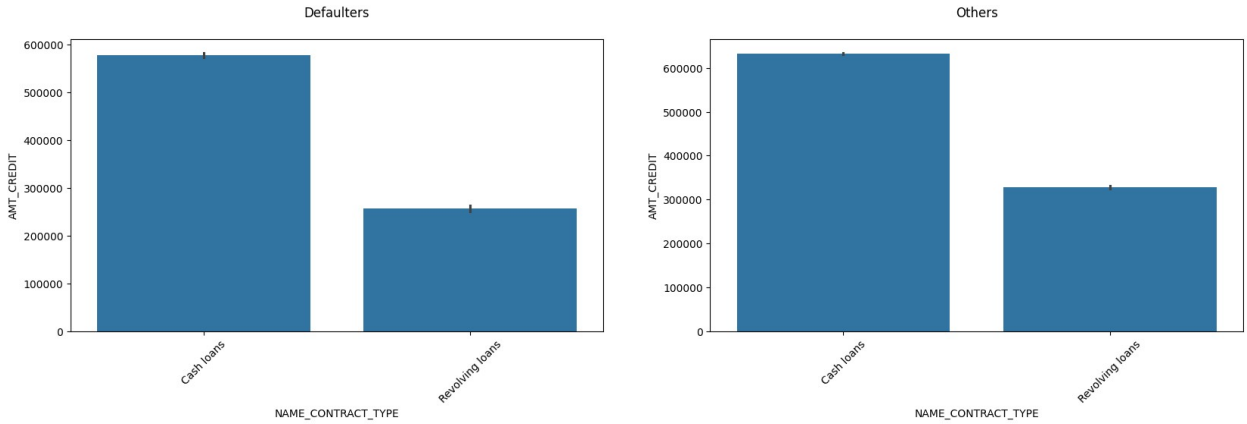


- Married loan applicants are mostly staying in house/apartment.
- Married people staying in house/apartments are the group with maximum number of loan applications.
- Single and civil marriage applicants are more likely to default.

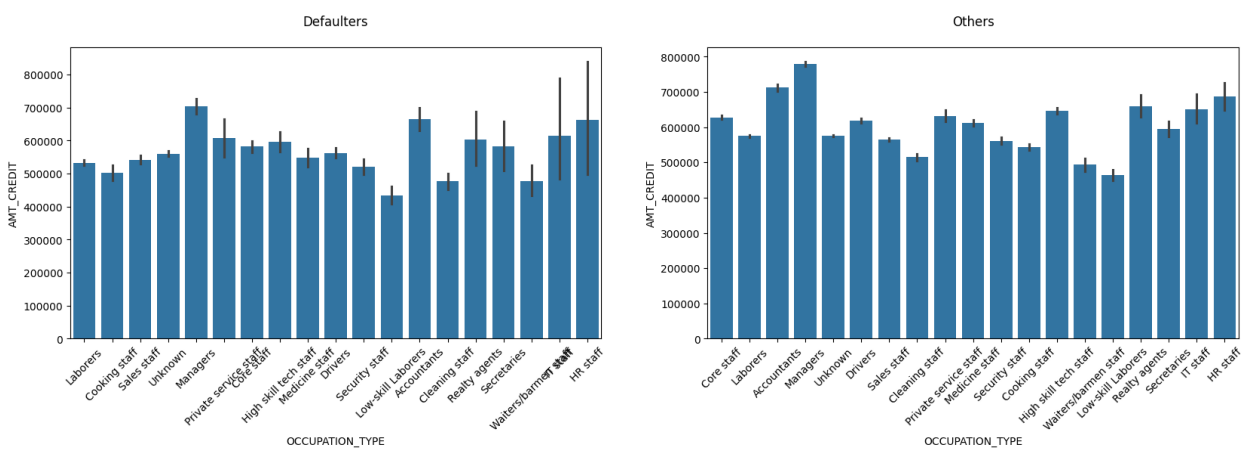
```
plt.figure(figsize=[8,5])
sns.barplot(data=df1,y='REG_CITY_NOT_WORK_CITY',x="TARGET")
plt.show()

# Number of people who have reg city differnet from their work city
has more defaulters than non defaulters.
```



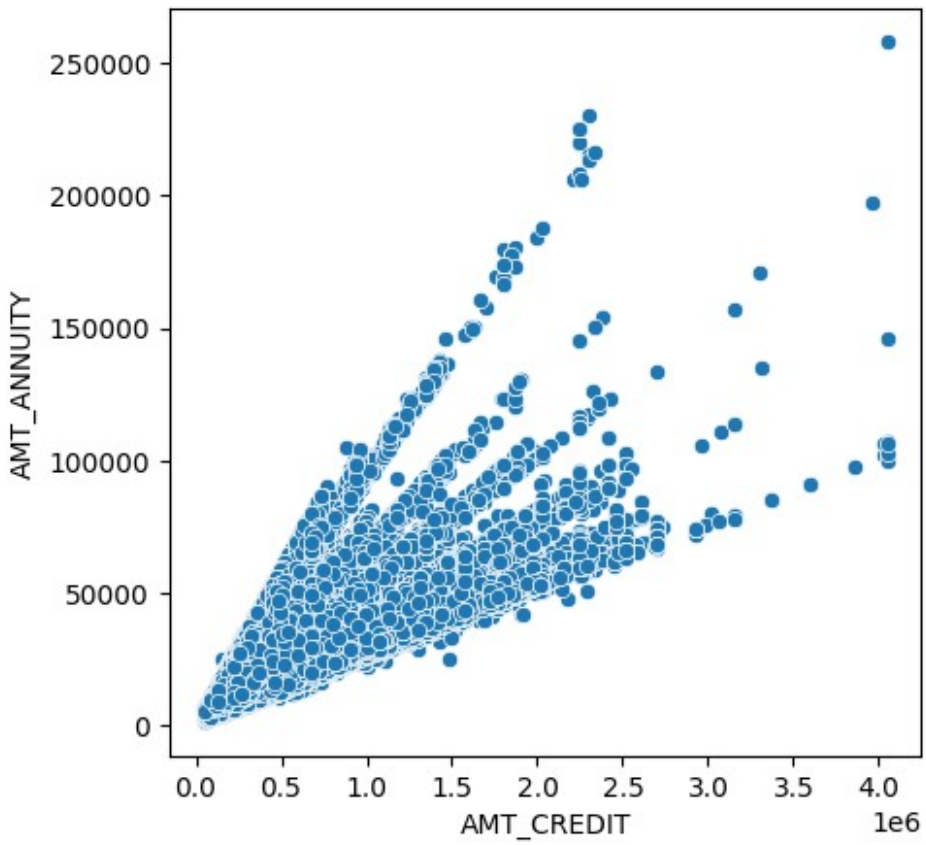

```
# NAME_CONTRACT_TYPE - AMT_CREDIT
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
plt.title('Defaulters\n')
sns.barplot(x="NAME_CONTRACT_TYPE",y="AMT_CREDIT", data=defaults)
plt.xlabel("NAME_CONTRACT_TYPE")
plt.ylabel("AMT_CREDIT")
plt.xticks(rotation=45)

plt.subplot(1,2,2)
plt.title('Others\n')
sns.barplot(x="NAME_CONTRACT_TYPE",y="AMT_CREDIT", data=others)
plt.xlabel("NAME_CONTRACT_TYPE")
plt.ylabel("AMT_CREDIT")
plt.xticks(rotation=45)
plt.show()
```



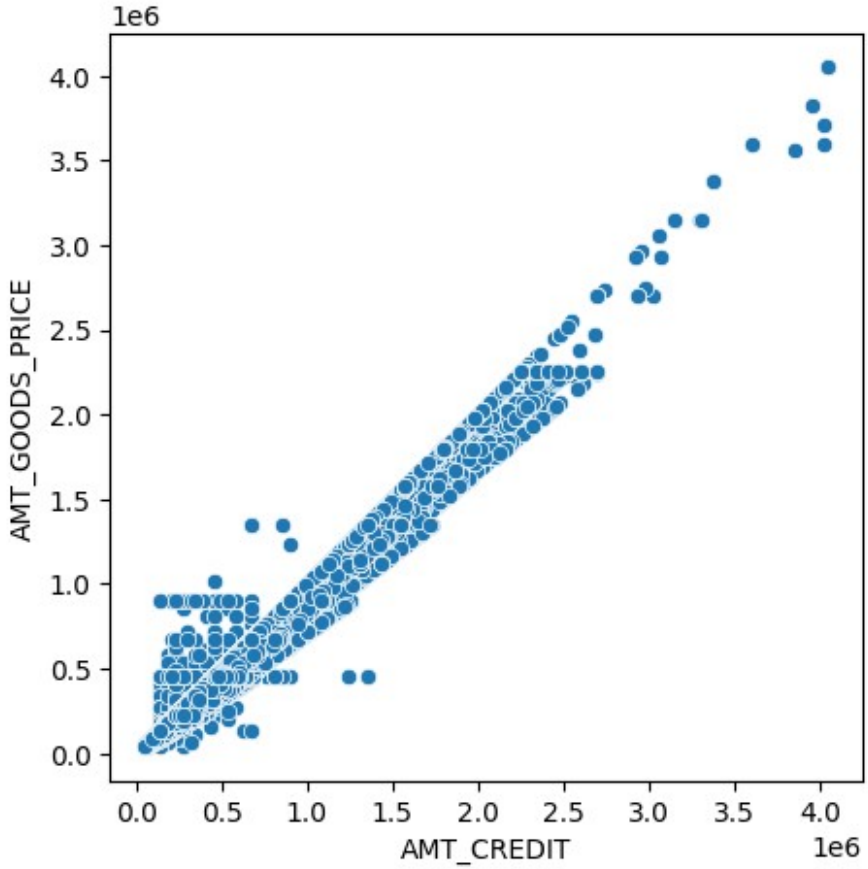

```
# Loan credit amount for cash loan is higher than that of revolving
loans.
# Cash loan is favourite among all genders.

# OCCUPATION_TYPE - AMT_CREDIT
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
plt.title('Defaulters\n')
sns.barplot(x="OCCUPATION_TYPE",y="AMT_CREDIT", data=defaults)
plt.xlabel("OCCUPATION_TYPE")
plt.ylabel("AMT_CREDIT")
plt.xticks(rotation=45)

plt.subplot(1,2,2)
plt.title('Others\n')
sns.barplot(x="OCCUPATION_TYPE",y="AMT_CREDIT", data=others)
plt.xlabel("OCCUPATION_TYPE")
plt.ylabel("AMT_CREDIT")
plt.xticks(rotation=45)
plt.show()
```



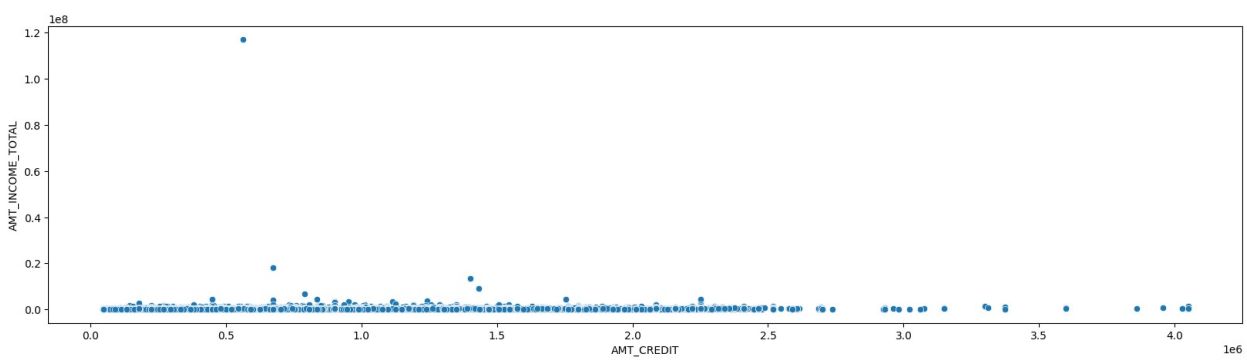
```
# Managers and Accountants have comparatively higher credit amount.
# Defaulters have high credit amount

# AMT_CREDIT-AMT_ANNUITY
sns.scatterplot(x = "AMT_CREDIT", y = "AMT_ANNUITY", data = df1)
plt.show()
```




```
# # AMT_CREDIT-AMT_GOODS_PRICE
sns.scatterplot(x = "AMT_CREDIT", y = "AMT_GOODS_PRICE", data = df1)
plt.show()
# AMT_CREDIT-AMT_GOODS_PRICE
```

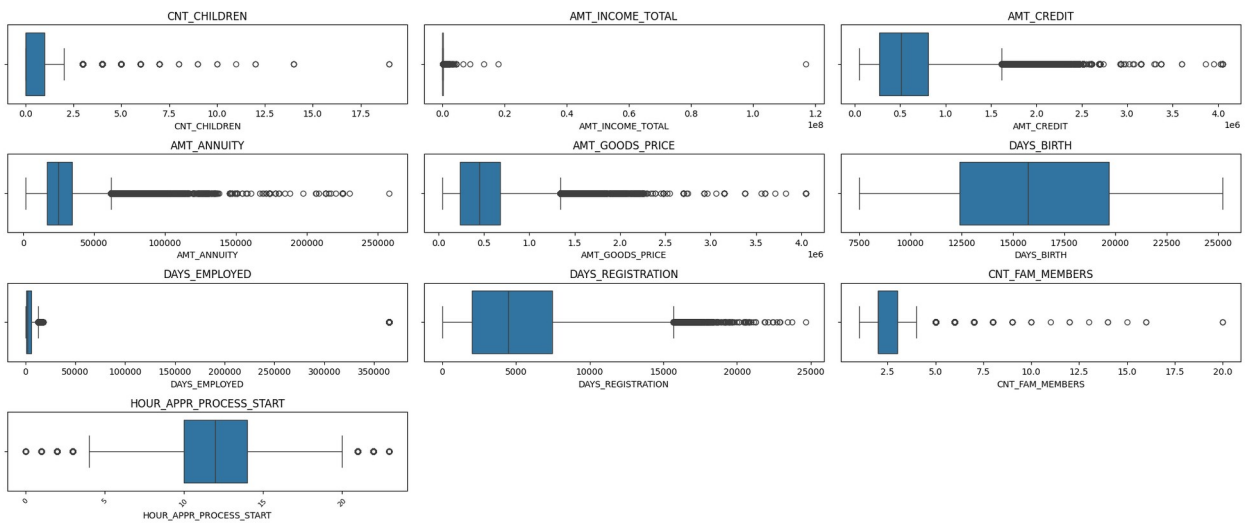



```
# # AMT_CREDIT-AMT_GOODS_PRICE
plt.figure(figsize=(20,5))
sns.scatterplot(x = "AMT_CREDIT", y = "AMT_INCOME_TOTAL", data = df1)
plt.show()
# AMT_CREDIT and AMT_INCOME_TOTAL does not seem to be correlated.
```




```
cols=['CNT_CHILDREN','AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','AM
T_GOODS_PRICE','DAYS_BIRTH',

'DAYS_EMPLOYED','DAYS_REGISTRATION','CNT_FAM_MEMBERS','HOUR_APPR_PROCE
SS_START']
```

```
plt.figure(figsize=(20,10))
j = 1
for i in cols:
  plt.subplot(5, 3, j)
  sns.boxplot(x=df1.loc[:, i])
  plt.title(i)
  j += 1

plt.xticks(rotation=45, fontsize=8)
plt.tight_layout()
plt.show()

# All numerical columns are having outliers which might need
treatement
```




```
corr_df1 =
defaults[['AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','AMT_GOODS_PRI
CE','DAYS_BIRTH','DAYS_EMPLOYED','CNT_CHILDREN']].corr()
corr_df1
```


```
{"summary":"{\n  \"name\": \"corr_df1\",\n  \"rows\": 7,\n
\"fields\": [\n  {\n  \"column\": \"AMT_INCOME_TOTAL\",\n
\"properties\": {\n  \"dtype\": \"number\",\n  \"std\":
0.37184769035509635,\n  \"min\": -0.014977396491471881,\n
\"max\": 1.0,\n  \"num_unique_values\": 7,\n  \"samples\":
[\n  1.0,\n  0.03813143463007386,\n  -
0.014977396491471881\n  ],\n  \"semantic_type\": \"\",\n
\"description\": \"\"\n  }\n  },\n  {\n  \"column\":
\"AMT_CREDIT\",\n  \"properties\": {\n  \"dtype\":
\"number\",\n  \"std\": 0.47307483573834835,\n  \"min\": -
0.0016749611094153205,\n  \"max\": 1.0,\n
\"num_unique_values\": 7,\n  \"samples\": [\n
0.03813143463007386,\n  1.0,\n  0.0019301832548641012\
```

n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"AMT_ANNUITY\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.4542177710284441,\n   \"min\": -
0.08120712012550896,\n   \"max\": 1.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n
0.046421056510901545,\n   0.7521947353758889,\n   -
0.08120712012550896\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"AMT_GOODS_PRICE\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.47338669530151983,\n   \"min\": -
0.008110740552729026,\n   \"max\": 1.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n
0.03759087313053715,\n   0.9827833228776286,\n
0.006647516021066061\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"DAYS_BIRTH\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.42325480146307637,\n   \"min\": -
0.259108666117443,\n   \"max\": 1.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   -
0.0030962446816019847,\n   0.13531636902889027,\n
0.5821851479036093\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"DAYS_EMPLOYED\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.4360689987181553,\n   \"min\": -
0.192863828031626,\n   \"max\": 1.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   -
0.014977396491471881,\n   0.0019301832548641012,\n
1.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"CNT_CHILDREN\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.4199773820777796,\n   \"min\": -
0.259108666117443,\n   \"max\": 1.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n
0.00479578732616832,\n   -0.0016749611094153205,\n   -
0.192863828031626\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"corr_df1"}

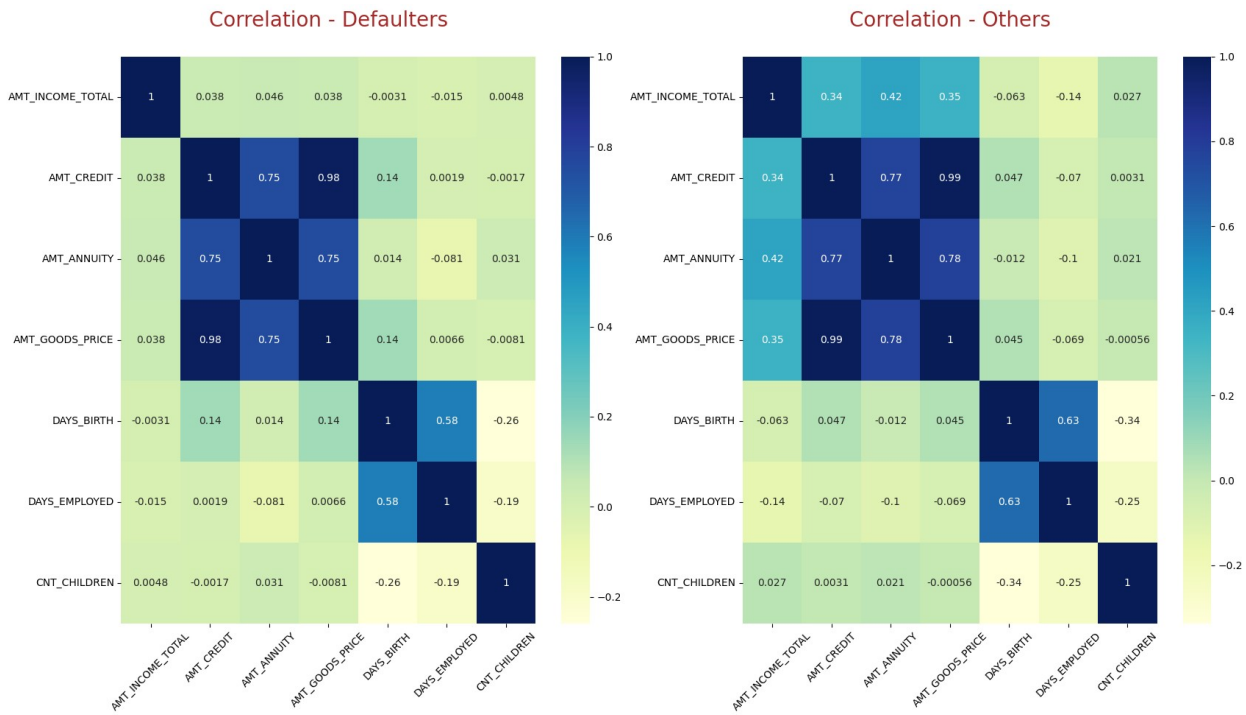
```
corr_df0 =
others[['AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','AMT_GOODS_PRICE
','DAYS_BIRTH','DAYS_EMPLOYED','CNT_CHILDREN']].corr()
corr_df0
```

{"summary":"{\n \"name\": \"corr_df0\",\n \"rows\": 7,\n
\"fields\": [\n   {\n   \"column\": \"AMT_INCOME_TOTAL\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.388223325280255,\n   \"min\": -0.1403924655915397,\n
\"max\": 1.0,\n   \"num_unique_values\": 7,\n   \"samples\":
[\n   1.0,\n   0.3427994498748986,\n   -

0.1403924655915397\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"AMT_CREDIT\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.472104229635207,\n   \"min\": -
0.07010431389260673,\n   \"max\": 1.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n
0.3427994498748986,\n   1.0,\n   -0.07010431389260673\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"AMT_ANNUITY\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.44867848298642565,\n   \"min\": -0.10497471490015325,\n
\"max\": 1.0,\n   \"num_unique_values\": 7,\n   \"samples\":
[\n   0.4189480860003718,\n   0.7712969450810268,\n
-0.10497471490015325\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"AMT_GOODS_PRICE\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.4731452829781594,\n   \"min\": -
0.06852725590409603,\n   \"max\": 1.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n
0.3494261146119611,\n   0.987021502499697,\n   -
0.06852725590409603\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"DAYS_BIRTH\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.4600900478792162,\n   \"min\": -
0.3369664839606809,\n   \"max\": 1.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   -
0.06260915814078147,\n   0.04737783126515827,\n
0.6261138779530819\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"DAYS_EMPLOYED\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.47442680282121086,\n   \"min\": -
0.24517406473567363,\n   \"max\": 1.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   -
0.1403924655915397,\n   -0.07010431389260673,\n   1.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"CNT_CHILDREN\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.4365533752471343,\n   \"min\": -0.3369664839606809,\n
\"max\": 1.0,\n   \"num_unique_values\": 7,\n   \"samples\":
[\n   0.027397187930142647,\n   0.0030812246529976295,\n
-0.24517406473567363\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"corr_df0"}

```
# Plot correlation heatmap for numerical variables

plt.figure(figsize=[20,10])

plt.subplot(1,2,1)
sns.heatmap(corr_df1, cmap="YlGnBu", annot = True)
```

```
plt.title('Correlation - Defaulters\n', fontdict={'fontsize': 20,
'fontweight' : 5, 'color' : 'Brown'})
plt.xticks(rotation=45)

plt.subplot(1,2,2)
sns.heatmap(corr_df0, cmap="YlGnBu", annot = True)
plt.title('Correlation - Others\n', fontdict={'fontsize': 20,
'fontweight' : 5, 'color' : 'Brown'})
plt.xticks(rotation=45)

plt.show()
```




```
 # AMT_CREDIT is strongly correlated to AMT_ANNUITY and
AMT_GOODS_PRICE in both cases
```

Statistical test

```
import statistics
from scipy import stats
from statsmodels.stats import weightstats as stests
from scipy.stats import shapiro
from statsmodels.stats import power

df1['TARGET']=df1['TARGET'].astype('O')

df1.drop(columns=['TARGET','AMT_ANNUITY','AMT_GOODS_PRICE','CNT_FAM_ME
MBERS','AMT_REQ_CREDIT_BUREAU_HOUR','AMT_REQ_CREDIT_BUREAU_DAY','AMT_R
```

```
EQ_CREDIT_BUREAU_WEEK','AMT_REQ_CREDIT_BUREAU_MON','AMT_REQ_CREDIT_BUR
EAU_QRT','AMT_REQ_CREDIT_BUREAU_YEAR']).select_dtypes(exclude=np.numbe
r).dtypes
```

```
NAME_CONTRACT_TYPE              object
CODE_GENDER                     object
FLAG_OWN_CAR                    object
FLAG_OWN_REALTY                 object
NAME_TYPE_SUITE                 object
NAME_INCOME_TYPE                object
NAME_EDUCATION_TYPE             object
NAME_FAMILY_STATUS              object
NAME_HOUSING_TYPE               object
OCCUPATION_TYPE                 object
WEEKDAY_APPR_PROCESS_START      object
ORGANIZATION_TYPE               object
INCOME_GROUP                  category
dtype: object
```

```
for i in df1.select_dtypes(exclude=np.number):
    a=pd.crosstab(df1[i],df1["TARGET"])
    print(i,":",stats.chi2_contingency(a))
# Null Hypothesis (H₀): The variables are independent (no association
between them).
# Alternative Hypothesis (H₁): The variables are not independent
(there is an association between them).
# his is the p-value associated with the test. Since this p-value is
extremely small (much smaller than a typical significance level like
0.05),
#you would reject the null hypothesis, indicating that the variables
are not independent, meaning there is a significant association
between them.
```

```
TARGET : Chi2ContingencyResult(statistic=307497.52517928014,
pvalue=0.0, dof=1, expected_freq=array([[259865.09294302,
22820.90705698],
       [ 22820.90705698,   2004.09294302]]))
NAME_CONTRACT_TYPE :
Chi2ContingencyResult(statistic=293.15054219964554,
pvalue=1.0235150721172847e-65, dof=1,
expected_freq=array([[255770.65910488,  22461.34089512],
       [ 26915.34089512,   2363.65910488]]))
CODE_GENDER : Chi2ContingencyResult(statistic=920.1038473528081,
pvalue=4.183493188620687e-202, dof=1,
expected_freq=array([[186108.28904332,  16343.71095668],
       [ 96577.71095668,   8481.28904332]]))
FLAG_OWN_CAR : Chi2ContingencyResult(statistic=146.65601825576024,
pvalue=9.330994431109667e-34, dof=1,
expected_freq=array([[186542.18504054,  16381.81495946],
       [ 96143.81495946,   8443.18504054]]))
```

```
FLAG_OWN_REALTY : Chi2ContingencyResult(statistic=11.57582729808418,
pvalue=0.0006681470317545887, dof=1,
expected_freq=array([[ 86594.42593598,   7604.57406402],
       [196091.57406402,  17220.42593598]]))
NAME_TYPE_SUITE : Chi2ContingencyResult(statistic=31.947946557874488,
pvalue=1.669680344640335e-05, dof=6,
expected_freq=array([[3.00325895e+03, 2.63741053e+02],
       [3.69078186e+04, 3.24118137e+03],
       [2.49122490e+02, 2.18775101e+01],
       [7.96088842e+02, 6.99111576e+01],
       [1.62710999e+03, 1.42890010e+02],
       [1.04521133e+04, 9.17886677e+02],
       [2.29650488e+05, 2.01675122e+04]]))
NAME_INCOME_TYPE : Chi2ContingencyResult(statistic=1253.4708080924986,
pvalue=1.9281456056861122e-266, dof=7,
expected_freq=array([[9.19271181e+00, 8.07288195e-01],
       [6.58354441e+04, 5.78155586e+03],
       [4.59635590e+00, 4.03644097e-01],
       [5.08926911e+04, 4.46930890e+03],
       [1.99509424e+04, 1.75205757e+03],
       [1.65468812e+01, 1.45311875e+00],
       [2.02239660e+01, 1.77603403e+00],
       [1.45956362e+05, 1.28176376e+04]]))
NAME_EDUCATION_TYPE :
Chi2ContingencyResult(statistic=1019.2131873088356,
pvalue=2.4476812052198174e-219, dof=4,
expected_freq=array([[1.50760474e+02, 1.32395264e+01],
       [6.88193984e+04, 6.04360161e+03],
       [9.44734992e+03, 8.29650078e+02],
       [3.50793882e+03, 3.08061175e+02],
       [2.00760552e+05, 1.76304476e+04]]))
NAME_FAMILY_STATUS :
Chi2ContingencyResult(statistic=504.6940825563211,
pvalue=7.744841561414037e-107, dof=5,
expected_freq=array([[2.73712994e+04, 2.40370060e+03],
       [1.80574277e+05, 1.58577235e+04],
       [1.81739912e+04, 1.59600876e+03],
       [4.17753595e+04, 3.66864047e+03],
       [1.83854236e+00, 1.61457639e-01],
       [1.47892348e+04, 1.29876525e+03]]))
NAME_HOUSING_TYPE :
Chi2ContingencyResult(statistic=420.55618983894664,
pvalue=1.0990890032617707e-88, dof=5,
expected_freq=array([[1.03142226e+03, 9.05777354e+01],
       [2.50839688e+05, 2.20283115e+04],
       [1.02802096e+04, 9.02790388e+02],
       [2.40573268e+03, 2.11267321e+02],
       [4.48696263e+03, 3.94037368e+02],
       [1.36419843e+04, 1.19801568e+03]]))
```

```
OCCUPATION_TYPE : Chi2ContingencyResult(statistic=1975.0827518430256,
pvalue=0.0, dof=18, expected_freq=array([[9.02080809e+03,
7.92191905e+02],
       [4.27736880e+03, 3.75631197e+02],
       [5.46598644e+03, 4.80013560e+02],
       [2.53443064e+04, 2.22569355e+03],
       [1.71012018e+04, 1.50179823e+03],
       [5.17549675e+02, 4.54503254e+01],
       [1.04613060e+04, 9.18693965e+02],
       [4.83536641e+02, 4.24633590e+01],
       [5.07308994e+04, 4.45510063e+03],
       [1.92403458e+03, 1.68965419e+02],
       [1.96457444e+04, 1.72525560e+03],
       [7.84781807e+03, 6.89181932e+02],
       [2.43790717e+03, 2.14092829e+02],
       [6.90372657e+02, 6.06273434e+01],
       [2.95104434e+04, 2.59155656e+03],
       [1.19964889e+03, 1.05351109e+02],
       [6.17842160e+03, 5.42578396e+02],
       [8.86094684e+04, 7.78153164e+03],
       [1.23917755e+03, 1.08822449e+02]]))
WEEKDAY_APPR_PROCESS_START :
Chi2ContingencyResult(statistic=15.387556913044653,
pvalue=0.01744736931389504, dof=6,
expected_freq=array([[46274.27268618,  4063.72731382],
       [46619.91865006,  4094.08134994],
       [31119.16800375,  2732.83199625],
       [14874.72697237,  1306.27302763],
       [46506.84829486,  4084.15170514],
       [49549.63590246,  4351.36409754],
       [47741.42949033,  4192.57050967]]))
ORGANIZATION_TYPE :
Chi2ContingencyResult(statistic=1609.2406359645206,
pvalue=5.224541090300172e-299, dof=57,
expected_freq=array([[3.94367336e+02, 3.46326635e+01],
       [2.25589148e+03, 1.98108523e+02],
       [2.30461285e+03, 2.02387150e+02],
       [5.50091874e+03, 4.83081256e+02],
       [9.70106877e+03, 8.51931232e+02],
       [6.25030861e+04, 5.48891389e+03],
       [2.39010507e+02, 2.09894931e+01],
       [6.17842160e+03, 5.42578396e+02],
       [3.48403777e+02, 3.05962226e+01],
       [8.73307622e+02, 7.66923785e+01],
       [5.14791861e+02, 4.52081389e+01],
       [9.56409736e+03, 8.39902638e+02],
       [8.88015960e+02, 7.79840396e+01],
       [2.71920415e+03, 2.38795848e+02],
       [9.55122757e+02, 8.38772434e+01],
```

```
       [1.00200559e+02, 8.79944132e+00],
       [2.48570927e+03, 2.18290728e+02],
       [3.39211066e+02, 2.97889344e+01],
       [6.15911691e+01, 5.40883090e+00],
       [4.21026201e+02, 3.69737993e+01],
       [3.01337093e+03, 2.64629070e+02],
       [8.06200825e+02, 7.07991747e+01],
       [5.50643437e+02, 4.83565629e+01],
       [1.02958372e+02, 9.04162778e+00],
       [1.20148743e+03, 1.05512567e+02],
       [2.20625083e+01, 1.93749167e+00],
       [3.09610534e+03, 2.71894664e+02],
       [5.48804895e+02, 4.81951052e+01],
       [6.32458572e+03, 5.55414278e+02],
       [2.80377710e+02, 2.46222899e+01],
       [1.02894023e+04, 9.03597676e+02],
       [2.42136029e+03, 2.12639710e+02],
       [2.91408964e+02, 2.55910358e+01],
       [1.53362011e+04, 1.34679889e+03],
       [2.15201383e+03, 1.88986166e+02],
       [1.98286794e+03, 1.74132064e+02],
       [3.64031387e+02, 3.19686125e+01],
       [7.81380503e+01, 6.86194965e+00],
       [1.66480011e+03, 1.46199892e+02],
       [8.17507861e+03, 7.17921391e+02],
       [2.98487352e+03, 2.62126477e+02],
       [1.81464131e+03, 1.59358690e+02],
       [3.53110446e+04, 3.10095541e+03],
       [1.44785211e+03, 1.27147891e+02],
       [5.30419471e+02, 4.65805288e+01],
       [3.19906371e+02, 2.80936292e+01],
       [1.74661524e+03, 1.53384757e+02],
       [3.21009496e+03, 2.81905038e+02],
       [5.88333556e+01, 5.16664445e+00],
       [4.50442878e+01, 3.95571215e+00],
       [5.80060115e+02, 5.09398851e+01],
       [7.19881261e+03, 6.32187385e+02],
       [1.84773507e+02, 1.62264927e+01],
       [2.02607368e+03, 1.77926318e+02],
       [1.09117489e+03, 9.58251087e+01],
       [4.96222583e+03, 4.35774167e+02],
       [1.21987286e+03, 1.07127143e+02],
       [5.09037224e+04, 4.47027765e+03]]))
INCOME_GROUP : Chi2ContingencyResult(statistic=165.27574873290746,
pvalue=4.5150776508532995e-33, dof=6,
expected_freq=array([[5.85559261e+04, 5.14207392e+03],
       [1.43313005e+05, 1.25849955e+04],
       [7.83341098e+04, 6.87889016e+03],
       [2.25406026e+03, 1.97939735e+02],
       [1.89370479e+02, 1.66295210e+01],
```

```
       [3.58516926e+01, 3.14830737e+00],
       [3.67709668e+00, 3.22903320e-01]]))
```

```
from scipy.stats import shapiro
col=["AMT_INCOME_TOTAL","AMT_CREDIT","AMT_ANNUITY","AMT_GOODS_PRICE","
REGION_POPULATION_RELATIVE","DAYS_BIRTH",

"DAYS_EMPLOYED","DAYS_REGISTRATION","DAYS_ID_PUBLISH","REGION_RATING_C
LIENT",
     "REGION_RATING_CLIENT_W_CITY",   "HOUR_APPR_PROCESS_START",
     "EXT_SOURCE_2", "EXT_SOURCE_3",

"OBS_30_CNT_SOCIAL_CIRCLE","DEF_30_CNT_SOCIAL_CIRCLE","OBS_60_CNT_SOCI
AL_CIRCLE","DEF_60_CNT_SOCIAL_CIRCLE","DAYS_LAST_PHONE_CHANGE","AMT_RE
Q_CREDIT_BUREAU_YEAR"]
for i in col:
    g1=df1[df1["TARGET"]==1][i]
    g2=df1[df1["TARGET"]==0][i]
    print(i,stats.shapiro(g1),stats.shapiro(g2))
```

```
AMT_INCOME_TOTAL ShapiroResult(statistic=0.01677782971852504,
pvalue=6.662375322133307e-142)
ShapiroResult(statistic=0.6192600058599327, pvalue=2.347277203672878e-
177)
AMT_CREDIT ShapiroResult(statistic=0.9049141805327058,
pvalue=2.5634663814507962e-80)
ShapiroResult(statistic=0.906051406373492, pvalue=6.906539376853772e-
132)
AMT_ANNUITY ShapiroResult(statistic=0.950335616967387,
pvalue=2.4323014446805237e-66)
ShapiroResult(statistic=0.9103069513528247, pvalue=1.694215084066444e-
130)
AMT_GOODS_PRICE ShapiroResult(statistic=0.8820380162995517,
pvalue=2.931472402851972e-85)
ShapiroResult(statistic=0.8903838302919731,
pvalue=1.4503528520418792e-136)
REGION_POPULATION_RELATIVE ShapiroResult(statistic=0.8964173270912867,
pvalue=2.916808606630239e-82)
ShapiroResult(statistic=0.8760926768509598,
pvalue=2.4311089073991435e-140)
DAYS_BIRTH ShapiroResult(statistic=0.962374384665713,
pvalue=8.769366684236648e-61)
ShapiroResult(statistic=0.970101969383913, pvalue=9.548528103092476e-
100)
DAYS_EMPLOYED ShapiroResult(statistic=0.3907498014316627,
pvalue=7.651324535820128e-128)
ShapiroResult(statistic=0.4846139205864164, pvalue=4.326184382825252e-
188)
DAYS_REGISTRATION ShapiroResult(statistic=0.9456712868471129,
pvalue=3.4332256903263684e-68)
```

```
ShapiroResult(statistic=0.954175613281937, pvalue=3.351416663784814e-
111)
DAYS_ID_PUBLISH ShapiroResult(statistic=0.950711109801341,
pvalue=3.4774291383025534e-66)
ShapiroResult(statistic=0.9400183640778638, pvalue=9.504077743523667e-
119)
REGION_RATING_CLIENT ShapiroResult(statistic=0.6788179937173893,
pvalue=3.3806580768183284e-110)
ShapiroResult(statistic=0.686589838733871, pvalue=1.2622882328752519e-
170)
REGION_RATING_CLIENT_W_CITY
ShapiroResult(statistic=0.6740554011890485,
pvalue=1.3748551665575398e-110)
ShapiroResult(statistic=0.6800208116685726, pvalue=2.454710404469502e-
171)
HOUR_APPR_PROCESS_START ShapiroResult(statistic=0.9905443883531981,
pvalue=7.58019364975599e-37)
ShapiroResult(statistic=0.9887943911134505, pvalue=8.036296612176762e-
76)
EXT_SOURCE_2 ShapiroResult(statistic=0.953531729416534,
pvalue=5.4992940033749706e-65)
ShapiroResult(statistic=0.924782746956069, pvalue=2.7404932678457945e-
125)
EXT_SOURCE_3 ShapiroResult(statistic=0.9594879026265539,
pvalue=3.0743180895246348e-62)
ShapiroResult(statistic=0.9654503866927961,
pvalue=1.4857157720665516e-103)
OBS_30_CNT_SOCIAL_CIRCLE ShapiroResult(statistic=0.6784973279135061,
pvalue=3.1807772678083075e-110)
ShapiroResult(statistic=0.6177332695759722,
pvalue=1.7012679345230269e-177)
DEF_30_CNT_SOCIAL_CIRCLE ShapiroResult(statistic=0.41911427937352796,
pvalue=1.7521283116663653e-126)
ShapiroResult(statistic=0.3484517290518495,
pvalue=1.2772492041897993e-196)
OBS_60_CNT_SOCIAL_CIRCLE ShapiroResult(statistic=0.6772679272460036,
pvalue=2.5192534166018172e-110)
ShapiroResult(statistic=0.6164521245406203, pvalue=1.299736741123094e-
177)
DEF_60_CNT_SOCIAL_CIRCLE ShapiroResult(statistic=0.3588685602754883,
pvalue=2.6347662340004317e-129)
ShapiroResult(statistic=0.2910433683585099 7,
pvalue=9.706134089914555e-200)
DAYS_LAST_PHONE_CHANGE ShapiroResult(statistic=0.8960402174221515,
pvalue=2.40904978549676e-82)
ShapiroResult(statistic=0.9246218506064995, pvalue=2.372675591912799e-
125)
AMT_REQ_CREDIT_BUREAU_YEAR ShapiroResult(statistic=0.8346970869311088,
pvalue=2.6911989352416408e-93)
```

```
ShapiroResult(statistic=0.8377357208383164,
pvalue=7.8145700166249495e-149)
```

```python
# Given the p-value is extraordinarily small,
# you would reject the null hypothesis at any reasonable significance
level (e.g., 0.05, 0.01).
# This means there is strong evidence that the two groups do not come
from the same distribution, suggesting that there is a
# statistically significant difference between the two groups you
compared.

from scipy.stats import mannwhitneyu
for i in
df1[["AMT_CREDIT","AMT_ANNUITY","AMT_GOODS_PRICE","REGION_POPULATION_R
ELATIVE","DAYS_BIRTH","DAYS_EMPLOYED","DAYS_REGISTRATION","DAYS_ID_PUB
LISH","REGION_RATING_CLIENT",
     "REGION_RATING_CLIENT_W_CITY",   "HOUR_APPR_PROCESS_START",
      "EXT_SOURCE_2", "EXT_SOURCE_3",
"OBS_30_CNT_SOCIAL_CIRCLE","DEF_30_CNT_SOCIAL_CIRCLE","OBS_60_CNT_SOCI
AL_CIRCLE","DEF_60_CNT_SOCIAL_CIRCLE","DAYS_LAST_PHONE_CHANGE","AMT_RE
Q_CREDIT_BUREAU_YEAR"]]:
    g1=df1[df1["TARGET"]==1][i]
    g2=df1[df1["TARGET"]==0][i]
    stat, p_value = mannwhitneyu(g1, g2)
    print(i,"stats:",stat,"p_value:",p_value)
```

```
AMT_CREDIT stats: 3378434520.0 p_value: 2.36018881652632e-22
AMT_ANNUITY stats: 3508297762.0 p_value: 0.9677479203683577
AMT_GOODS_PRICE stats: 3274917364.5 p_value: 2.821334043506046e-68
REGION_POPULATION_RELATIVE stats: 3281647585.5 p_value:
1.9999665809732712e-64
DAYS_BIRTH stats: 2926354691.5 p_value: 0.0
DAYS_EMPLOYED stats: 2866924137.5 p_value: 0.0
DAYS_REGISTRATION stats: 3210111966.0 p_value: 6.304065912692224e-110
DAYS_ID_PUBLISH stats: 3118165526.0 p_value: 1.3917745563748238e-186
REGION_RATING_CLIENT stats: 3846132118.5 p_value: 4.555791548665177e-
234
REGION_RATING_CLIENT_W_CITY stats: 3853982907.5 p_value:
2.6798603110992284e-250
HOUR_APPR_PROCESS_START stats: 3339172571.5 p_value:
5.433090599121058e-37
EXT_SOURCE_2 stats: 2414683836.5 p_value: 0.0
EXT_SOURCE_3 stats: 2460886559.0 p_value: 0.0
OBS_30_CNT_SOCIAL_CIRCLE stats: 3575901262.0 p_value:
4.896756666134539e-08
DEF_30_CNT_SOCIAL_CIRCLE stats: 3641655396.5 p_value:
5.093631155153175e-72
OBS_60_CNT_SOCIAL_CIRCLE stats: 3574774897.0 p_value:
7.863423087584917e-08
DEF_60_CNT_SOCIAL_CIRCLE stats: 3619686965.5 p_value:
```

```
2.064553521295198e-66
DAYS_LAST_PHONE_CHANGE stats: 3109788371.0 p_value:
6.183497950494181e-195
AMT_REQ_CREDIT_BUREAU_YEAR stats: 3603082793.0 p_value:
4.3996169061348015e-13
```

```
# H₀: The two groups come from the same distribution (or have the same
median).
# H₁: The two groups come from different distributions (or have
different medians).
 #The p-value is extremely small (essentially 0 when considering
typical significance levels, like 0.05).
#This means that the result is statistically significant, and you can
reject the null hypothesis,
#which usually states that there is no difference between the two
groups.
```

```
df_p = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/previous_application.csv')
```

```
df_p.shape
```

```
(1670214, 37)
```

```
df_p.describe().T
```


```
{"summary":"{\n  \"name\": \"df_p\",\n  \"rows\": 21,\n  \"fields\":
[\n    {\n      \"column\": \"count\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 512237.10644422047,\n
\"min\": 5951.0,\n        \"max\": 1670214.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
1297979.0,\n          5951.0,\n          1670214.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"mean\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 417552.0186896905,\n
\"min\": -880.6796679946402,\n        \"max\": 1923089.1353305625,\n
\"num_unique_values\": 21,\n        \"samples\": [\n
1923089.1353305625,\n          33767.77405382746,\n
342209.85503871535\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"std\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 146016.87868102104,\n        \"min\": 0.05932963128712511,\n
\"max\": 532597.9586955433,\n        \"num_unique_values\": 21,\n
\"samples\": [\n          532597.9586955433,\n
106857.03478910643,\n          88916.11583368349\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"min\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 218429.7507997795,\n
\"min\": -2922.0,\n        \"max\": 1000001.0,\n
\"num_unique_values\": 13,\n        \"samples\": [\n          -
2889.0,\n          -2892.0,\n          1000001.0\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"25%\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 323874.28943008685,\n   \"min\": -1628.0,\n   \"max\": 1461857.25,\n   \"num_unique_values\": 19,\n   \"samples\": [\n   1461857.25,\n   0.0,\n   -1300.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"50%\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 421268.92450452177,\n   \"min\": -831.0,\n   \"max\": 1923110.5,\n   \"num_unique_values\": 20,\n   \"samples\": [\n   1923110.5,\n   -537.0,\n   -831.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"75%\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 520049.5579420275,\n   \"min\": -411.0,\n   \"max\": 2384279.75,\n   \"num_unique_values\": 20,\n   \"samples\": [\n   2384279.75,\n   129.0,\n   365243.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"max\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 2496808.87949573,\n   \"min\": -1.0,\n   \"max\": 6905160.0,\n   \"num_unique_values\": 11,\n   \"samples\": [\n   23.0,\n   2845382.0,\n   84.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
df_null=(df_p.isnull().sum()/
df_p.shape[0]*100).reset_index().sort_values(by=0,ascending=False)
```

```
df_null=df_null[df_null[0]>40].set_index("index")
```

```
df_null.index
```

```
Index(['RATE_INTEREST_PRIVILEGED', 'RATE_INTEREST_PRIMARY',
'AMT_DOWN_PAYMENT',
       'RATE_DOWN_PAYMENT', 'NAME_TYPE_SUITE',
'NFLAG_INSURED_ON_APPROVAL',
       'DAYS_TERMINATION', 'DAYS_LAST_DUE',
'DAYS_LAST_DUE_1ST_VERSION',
       'DAYS_FIRST_DUE', 'DAYS_FIRST_DRAWING'],
      dtype='object', name='index')
```

```
col_with_40=['RATE_INTEREST_PRIVILEGED', 'RATE_INTEREST_PRIMARY',
'AMT_DOWN_PAYMENT',
       'RATE_DOWN_PAYMENT', 'NAME_TYPE_SUITE',
'NFLAG_INSURED_ON_APPROVAL',
       'DAYS_TERMINATION', 'DAYS_LAST_DUE',
'DAYS_LAST_DUE_1ST_VERSION',
       'DAYS_FIRST_DUE', 'DAYS_FIRST_DRAWING']
```

```
for i in col_with_40:
  df_p.drop(columns=i,inplace=True)
```

```
df_p.shape
```

```
(1670214, 26)
```

```
df_p.isnull().sum()/df_p.shape[0]*100
```

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

```
df_p.select_dtypes(np.number).skew()

# all are skewed we can impute null values with medin
```

```
SK_ID_PREV                  -0.000573
SK_ID_CURR                  -0.003303
AMT_ANNUITY                  2.692572
AMT_APPLICATION              3.391442
AMT_CREDIT                   3.245815
AMT_GOODS_PRICE              3.073690
HOUR_APPR_PROCESS_START     -0.025629
NFLAG_LAST_APPL_IN_DAY     -16.735924
DAYS_DECISION               -1.053080
SELLERPLACE_AREA           529.620279
```

```
CNT_PAYMENT                  1.531403
dtype: float64
```

```
df_p.select_dtypes(exclude=np.number).describe()
```


```
{"summary":"{\n  \"name\": \"df_p\",\n  \"rows\": 4,\n  \"fields\": [\
n    {\n      \"column\": \"NAME_CONTRACT_TYPE\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          4,\n
\"747553\",\n          \"1670214\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"WEEKDAY_APPR_PROCESS_START\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          7,\n
\"255118\",\n          \"1670214\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"FLAG_LAST_APPL_PER_CONTRACT\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2,\n
\"1661739\",\n          \"1670214\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NAME_CASH_LOAN_PURPOSE\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          25,\n
\"922661\",\n          \"1670214\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NAME_CONTRACT_STATUS\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          4,\n
\"1036781\",\n          \"1670214\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NAME_PAYMENT_TYPE\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          4,\n
\"1033552\",\n          \"1670214\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CODE_REJECT_REASON\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          9,\n
\"1353093\",\n          \"1670214\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NAME_CLIENT_TYPE\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          4,\n
\"1231261\",\n          \"1670214\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NAME_GOODS_CATEGORY\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          28,\n
\"950809\",\n          \"1670214\"\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"NAME_PORTFOLIO\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n   \"samples\": [\n   5,\n
\"691011\",\n   \"1670214\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"NAME_PRODUCT_TYPE\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n   \"samples\": [\n   3,\n
\"1063666\",\n   \"1670214\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"CHANNEL_TYPE\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n   \"samples\": [\n   8,\n
\"719968\",\n   \"1670214\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"NAME_SELLER_INDUSTRY\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n   \"samples\": [\n   11,\n
\"855720\",\n   \"1670214\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"NAME_YIELD_GROUP\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n   \"samples\": [\n   5,\n
\"517215\",\n   \"1670214\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"PRODUCT_COMBINATION\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n   \"samples\": [\n   17,\n
\"285990\",\n   \"1669868\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

```
df_p.select_dtypes(exclude=np.number).isnull().sum()
```

```
NAME_CONTRACT_TYPE               0
WEEKDAY_APPR_PROCESS_START       0
FLAG_LAST_APPL_PER_CONTRACT      0
NAME_CASH_LOAN_PURPOSE           0
NAME_CONTRACT_STATUS             0
NAME_PAYMENT_TYPE                0
CODE_REJECT_REASON               0
NAME_CLIENT_TYPE                 0
NAME_GOODS_CATEGORY              0
NAME_PORTFOLIO                   0
NAME_PRODUCT_TYPE                0
CHANNEL_TYPE                     0
NAME_SELLER_INDUSTRY             0
NAME_YIELD_GROUP                 0
```

```
PRODUCT_COMBINATION            346
dtype: int64
```

```
df_p["PRODUCT_COMBINATION"]=df_p["PRODUCT_COMBINATION"].fillna(df_p["P
RODUCT_COMBINATION"].mode()[0])
```

```
for i in df_p.select_dtypes(np.number):
  df_p[i]=df_p[i].fillna(df_p[i].median())
```

```
df_p.isnull().sum()
```

```
SK_ID_PREV                     0
SK_ID_CURR                     0
NAME_CONTRACT_TYPE             0
AMT_ANNUITY                    0
AMT_APPLICATION                0
AMT_CREDIT                     0
AMT_GOODS_PRICE                0
WEEKDAY_APPR_PROCESS_START     0
HOUR_APPR_PROCESS_START        0
FLAG_LAST_APPL_PER_CONTRACT    0
NFLAG_LAST_APPL_IN_DAY         0
NAME_CASH_LOAN_PURPOSE         0
NAME_CONTRACT_STATUS           0
DAYS_DECISION                  0
NAME_PAYMENT_TYPE              0
CODE_REJECT_REASON             0
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                    0
NAME_YIELD_GROUP               0
PRODUCT_COMBINATION            0
dtype: int64
```

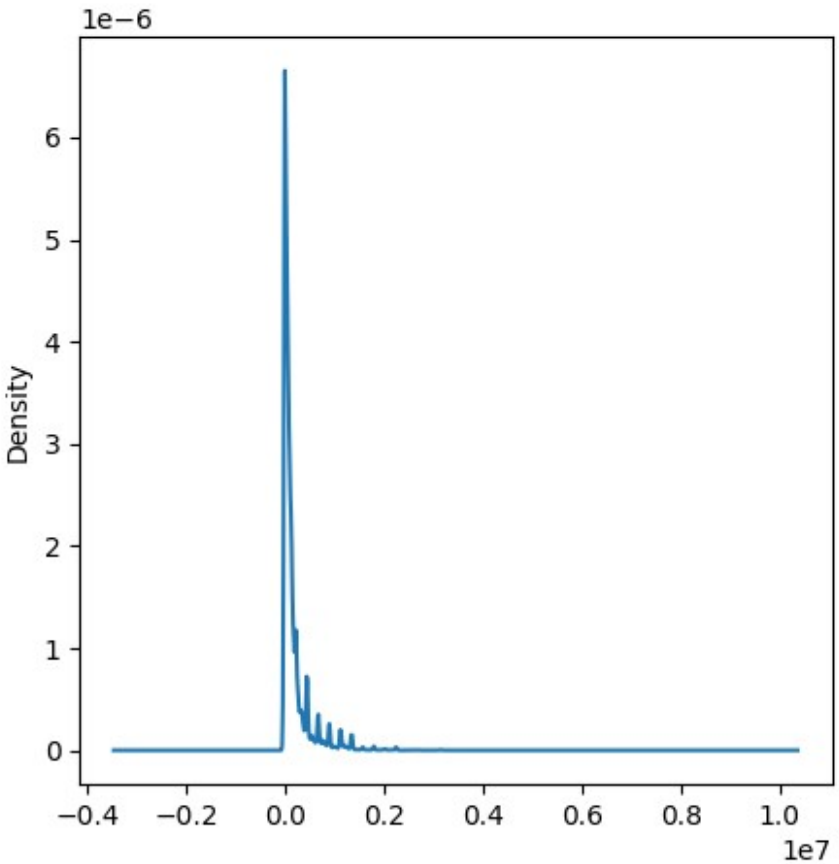
```
df_p.select_dtypes(np.number).columns
```

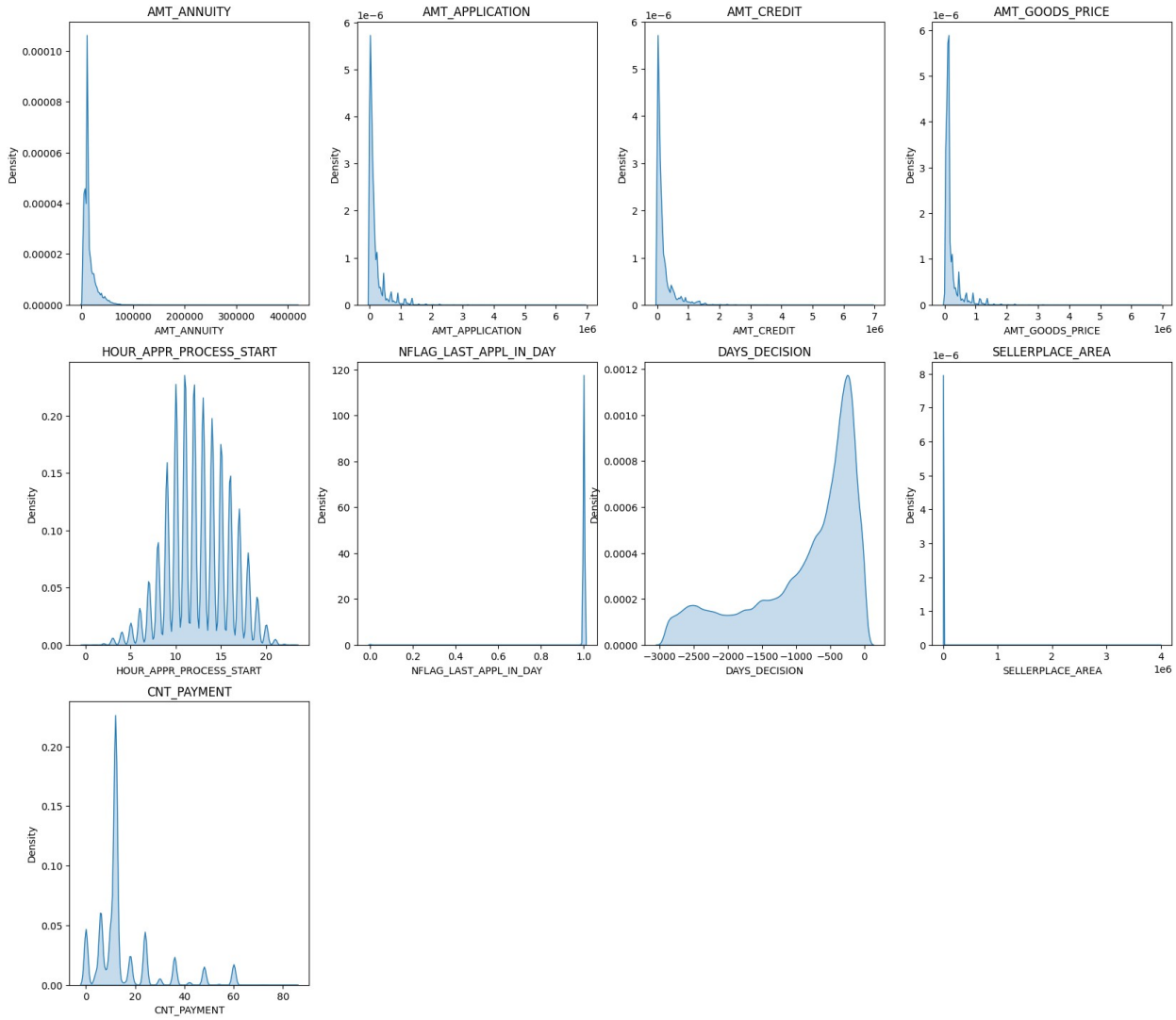
```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'AMT_ANNUITY', 'AMT_APPLICATION',
       'AMT_CREDIT', 'AMT_GOODS_PRICE', 'HOUR_APPR_PROCESS_START',
       'NFLAG_LAST_APPL_IN_DAY', 'DAYS_DECISION', 'SELLERPLACE_AREA',
       'CNT_PAYMENT'],
      dtype='object')
```

```
df_p['AMT_APPLICATION'].plot(kind='kde')
```

```
cols=df_p.select_dtypes(np.number).drop(columns=["SK_ID_PREV","SK_ID_C
URR"]).columns
```

```
plt.figure(figsize =[20,30])
j=1
for i in cols:
    plt.subplot(5,4,j)
    sns.kdeplot(df_p[i],fill =True)
    plt.title(i)
    j+=1
```






```
pd.set_option('display.max_columns',None)


df_p.head()
```

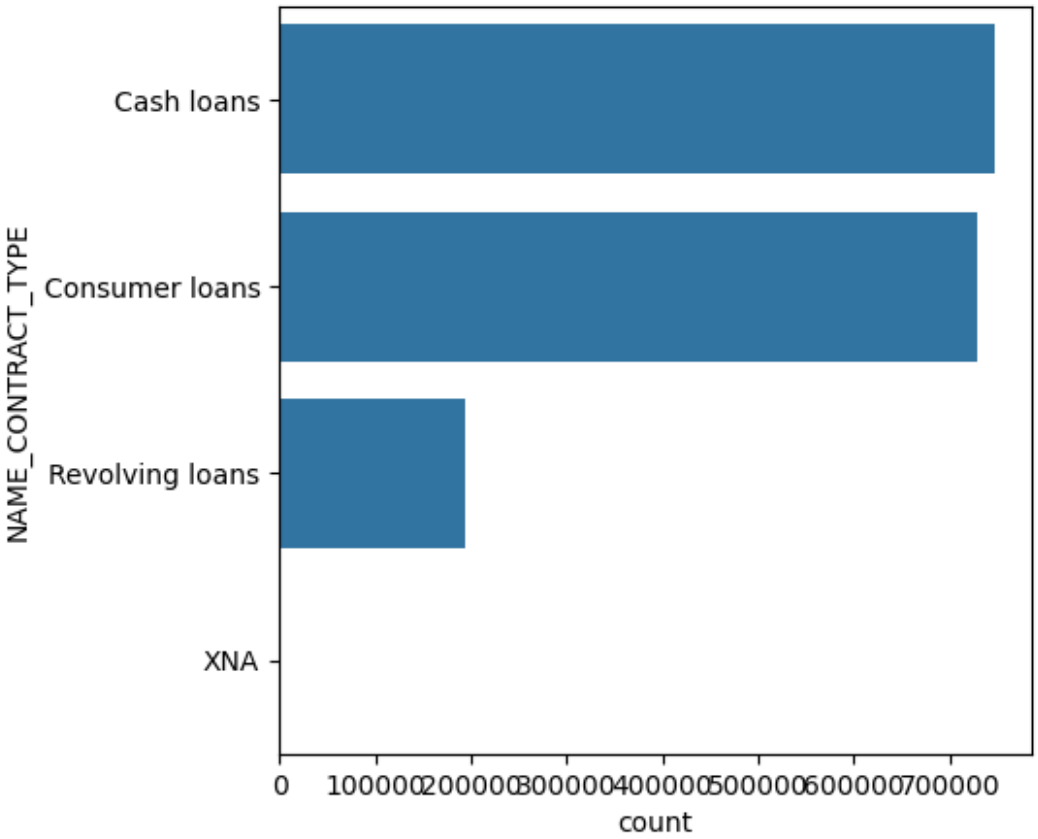
{"type":"dataframe","variable_name":"df_p"}

```
df_p.select_dtypes(exclude=np.number).columns
```

```
Index(['NAME_CONTRACT_TYPE', 'WEEKDAY_APPR_PROCESS_START',
       'FLAG_LAST_APPL_PER_CONTRACT', 'NAME_CASH_LOAN_PURPOSE',
       'NAME_CONTRACT_STATUS', 'NAME_PAYMENT_TYPE',
'CODE_REJECT_REASON',
       'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO',
       'NAME_PRODUCT_TYPE', 'CHANNEL_TYPE', 'NAME_SELLER_INDUSTRY',
       'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION'],
      dtype='object')
```

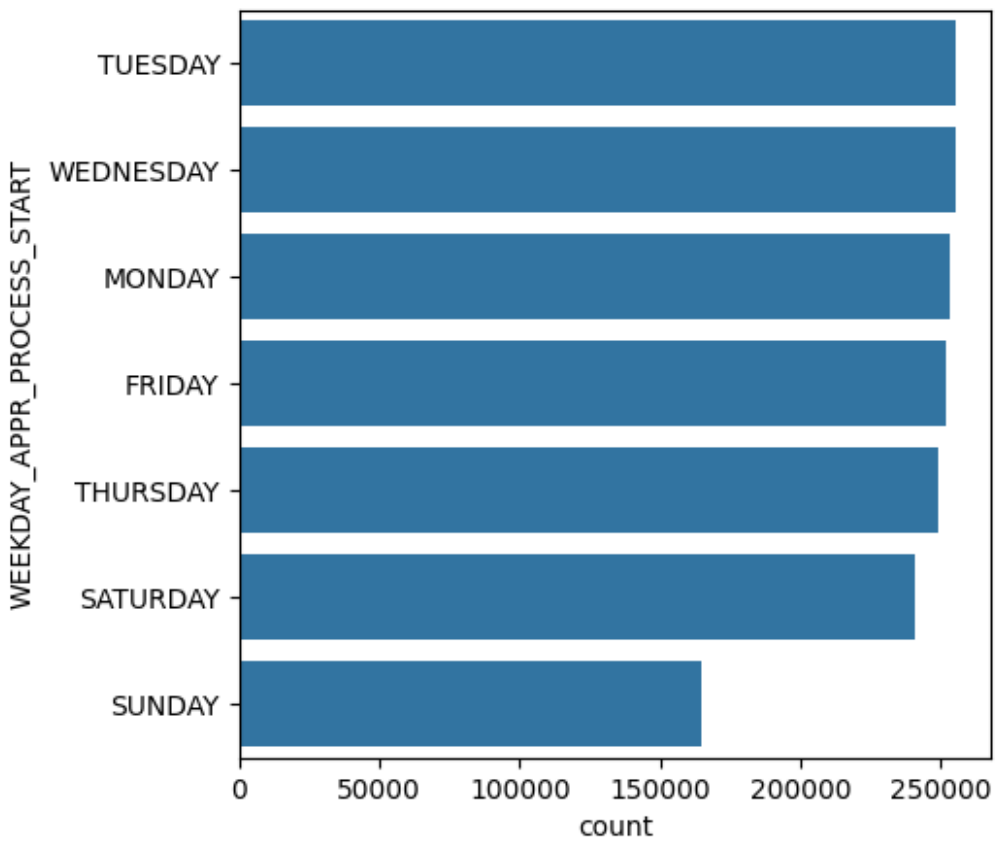
```
sns.countplot(df_p['NAME_CONTRACT_TYPE'],order=df_p['NAME_CONTRACT_TYPE'].value_counts().index)
```

```
<Axes: xlabel='count', ylabel='NAME_CONTRACT_TYPE'>
```



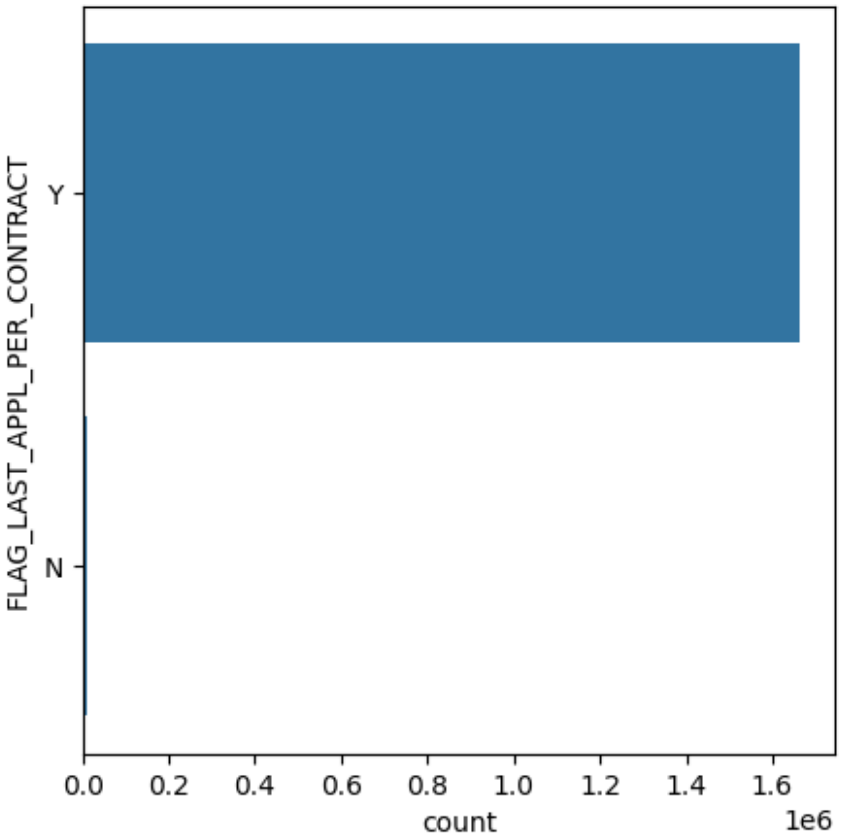

```
sns.countplot(df_p['WEEKDAY_APPR_PROCESS_START'],order=df_p['WEEKDAY_APPR_PROCESS_START'].value_counts().index)
```

```
<Axes: xlabel='count', ylabel='WEEKDAY_APPR_PROCESS_START'>
```



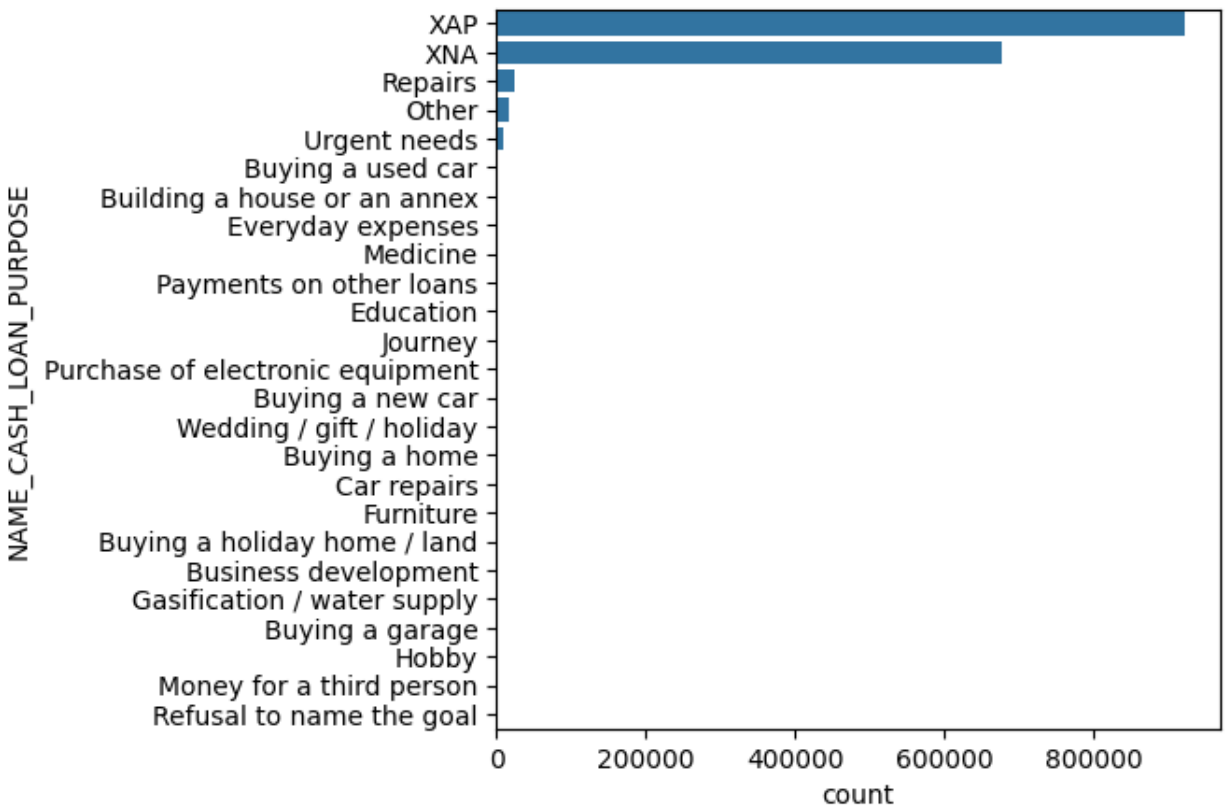

```
sns.countplot(df_p['FLAG_LAST_APPL_PER_CONTRACT'],order=df_p['FLAG_LAS
T_APPL_PER_CONTRACT'].value_counts().index)
```

```
<Axes: xlabel='count', ylabel='FLAG_LAST_APPL_PER_CONTRACT'>
```



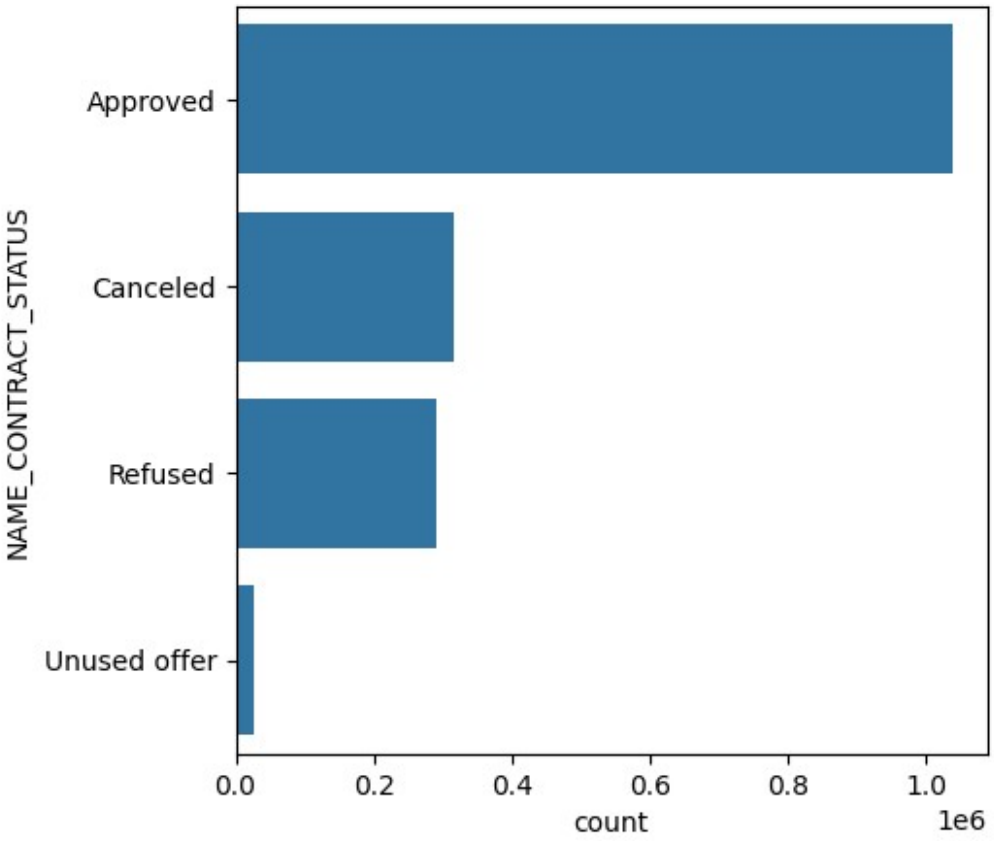

```
sns.countplot(df_p['NAME_CASH_LOAN_PURPOSE'],order=df_p['NAME_CASH_LOA
N_PURPOSE'].value_counts().index)
```

```
<Axes: xlabel='count', ylabel='NAME_CASH_LOAN_PURPOSE'>
```



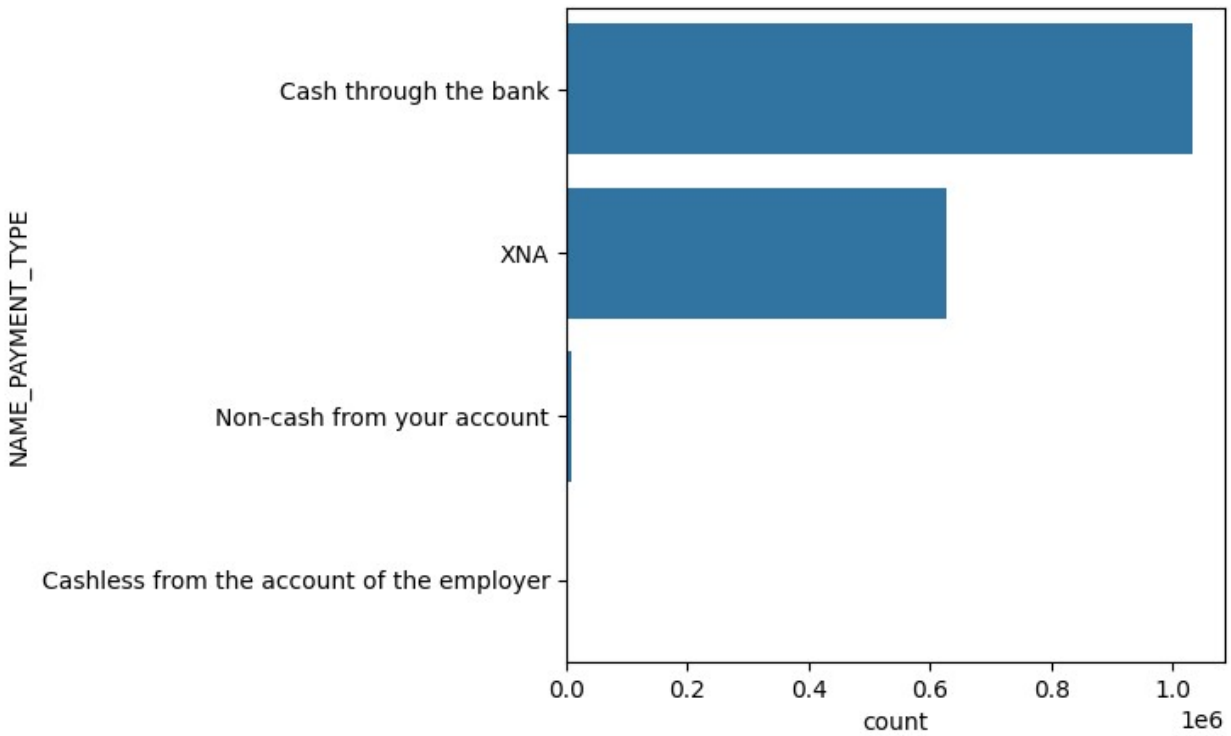

```
sns.countplot(df_p['NAME_CONTRACT_STATUS'],order=df_p['NAME_CONTRACT_S
TATUS'].value_counts().index)
```

```
<Axes: xlabel='count', ylabel='NAME_CONTRACT_STATUS'>
```



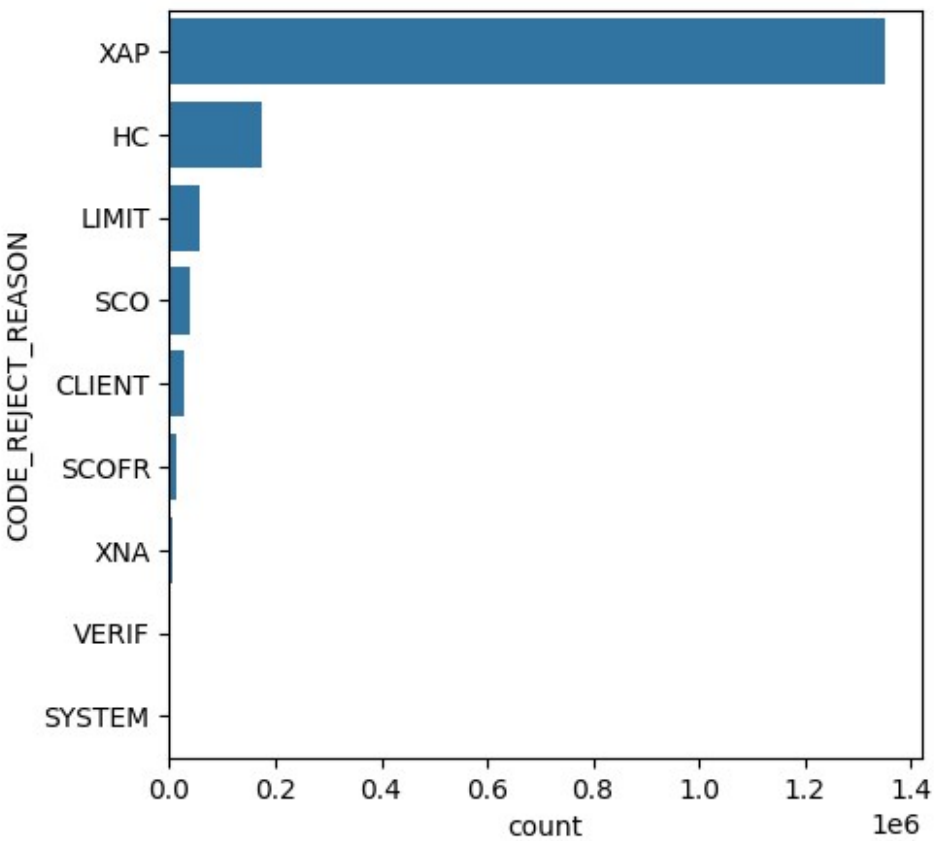

```
sns.countplot(df_p['NAME_PAYMENT_TYPE'],order=df_p['NAME_PAYMENT_TYPE'
].value_counts().index)
```

```
<Axes: xlabel='count', ylabel='NAME_PAYMENT_TYPE'>
```



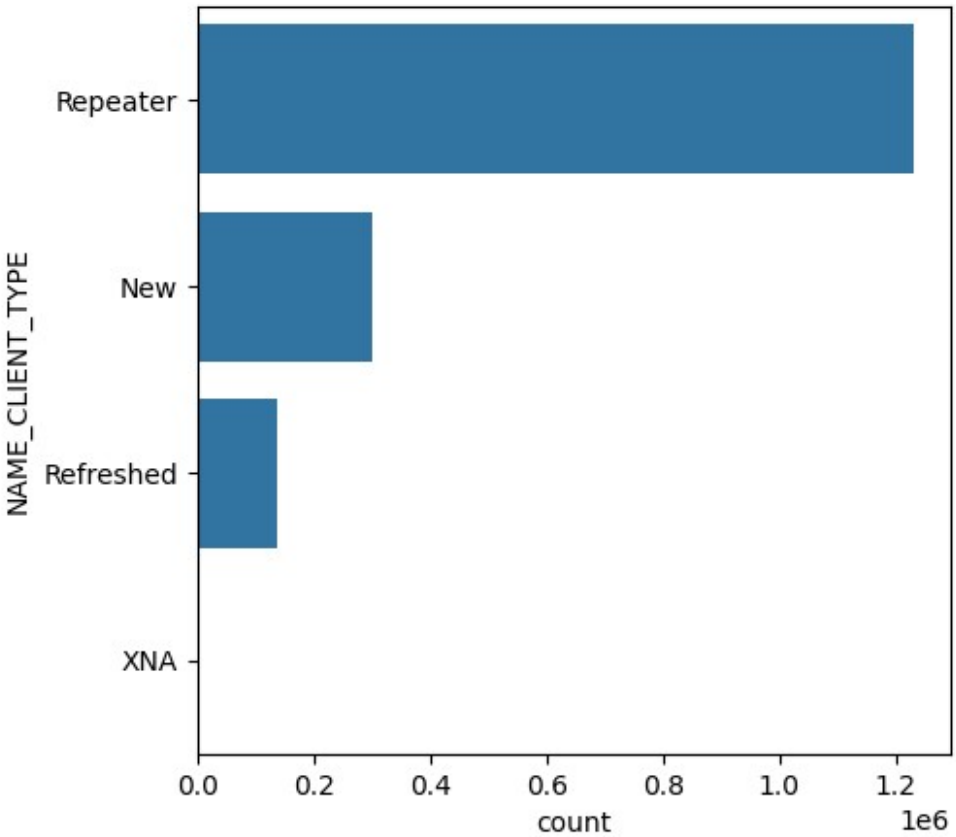

```
sns.countplot(df_p['CODE_REJECT_REASON'],order=df_p['CODE_REJECT_REASO
N'].value_counts().index)
```

```
<Axes: xlabel='count', ylabel='CODE_REJECT_REASON'>
```



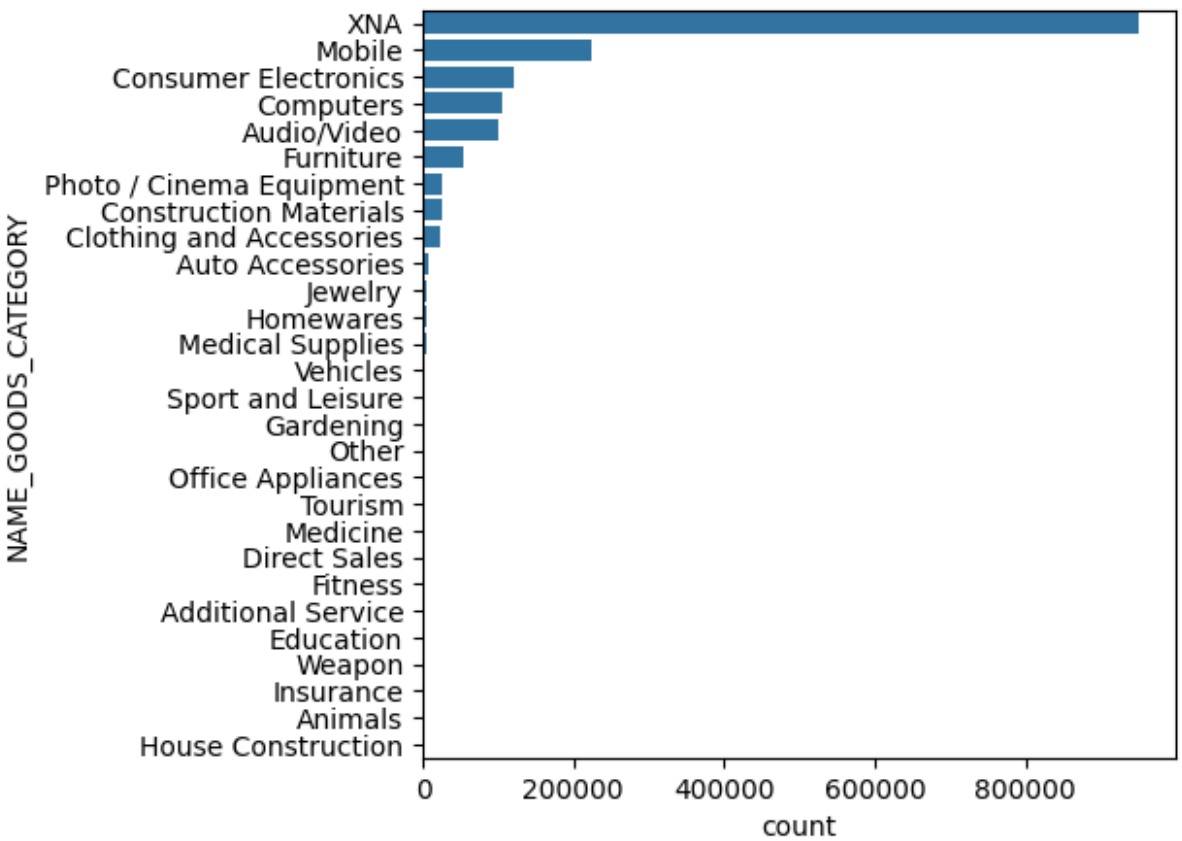

```
sns.countplot(df_p['NAME_CLIENT_TYPE'],order=df_p['NAME_CLIENT_TYPE'].
value_counts().index)
```

```
<Axes: xlabel='count', ylabel='NAME_CLIENT_TYPE'>
```



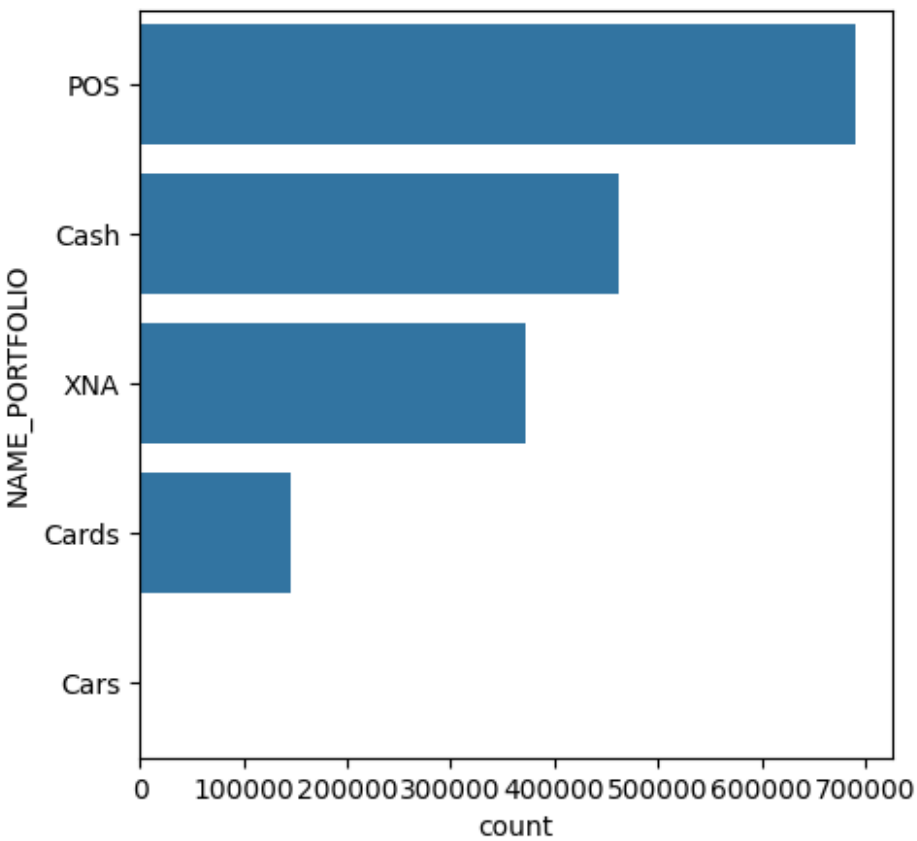

```
sns.countplot(df_p['NAME_GOODS_CATEGORY'],order=df_p['NAME_GOODS_CATEG
ORY'].value_counts().index)
```

```
<Axes: xlabel='count', ylabel='NAME_GOODS_CATEGORY'>
```



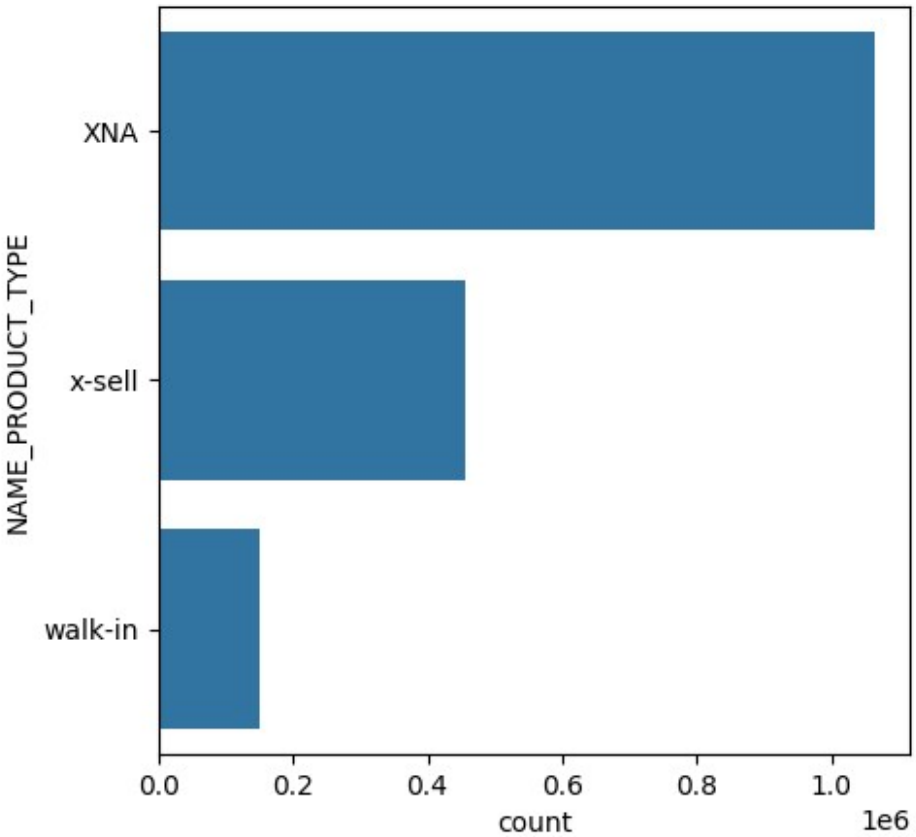

```
sns.countplot(df_p['NAME_PORTFOLIO'],order=df_p['NAME_PORTFOLIO'].valu
e_counts().index)
```

```
<Axes: xlabel='count', ylabel='NAME_PORTFOLIO'>
```



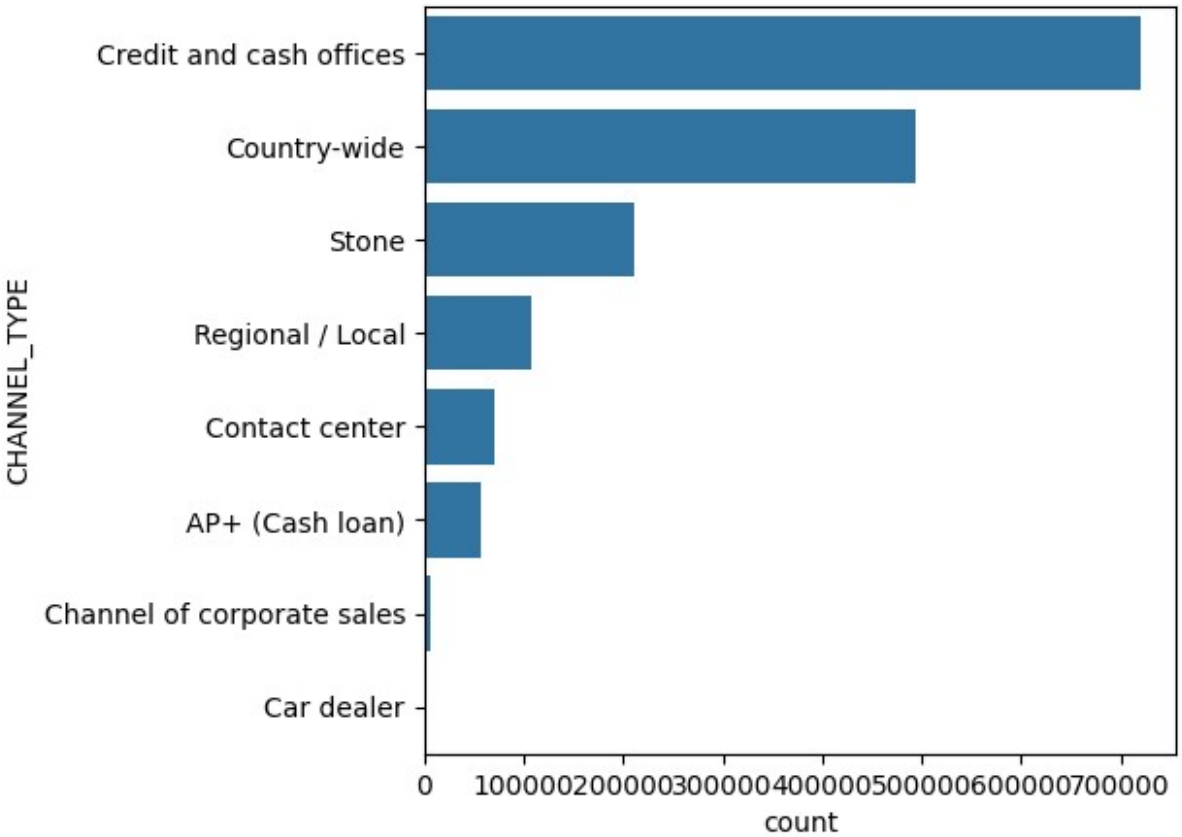

```
sns.countplot(df_p['NAME_PRODUCT_TYPE'],order=df_p['NAME_PRODUCT_TYPE'
].value_counts().index)
```

```
<Axes: xlabel='count', ylabel='NAME_PRODUCT_TYPE'>
```



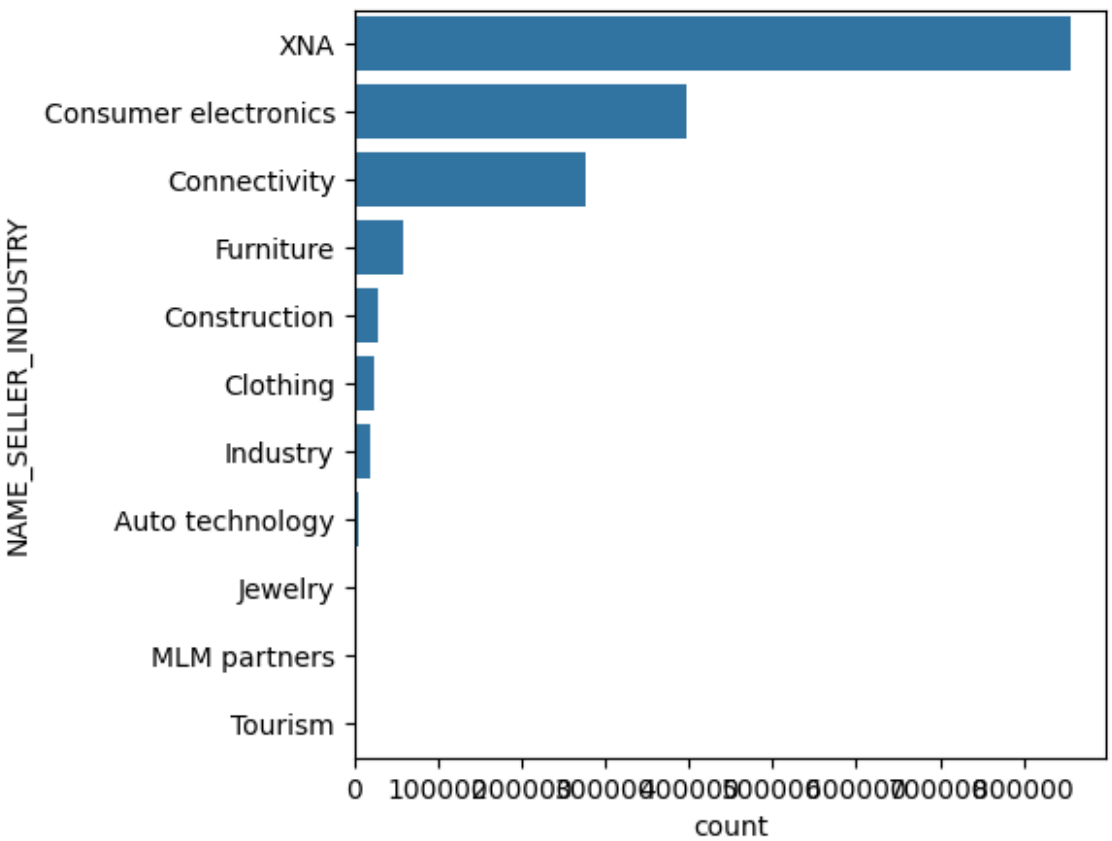

```
sns.countplot(df_p['CHANNEL_TYPE'],order=df_p['CHANNEL_TYPE'].value_co
unts().index)
```

```
<Axes: xlabel='count', ylabel='CHANNEL_TYPE'>
```



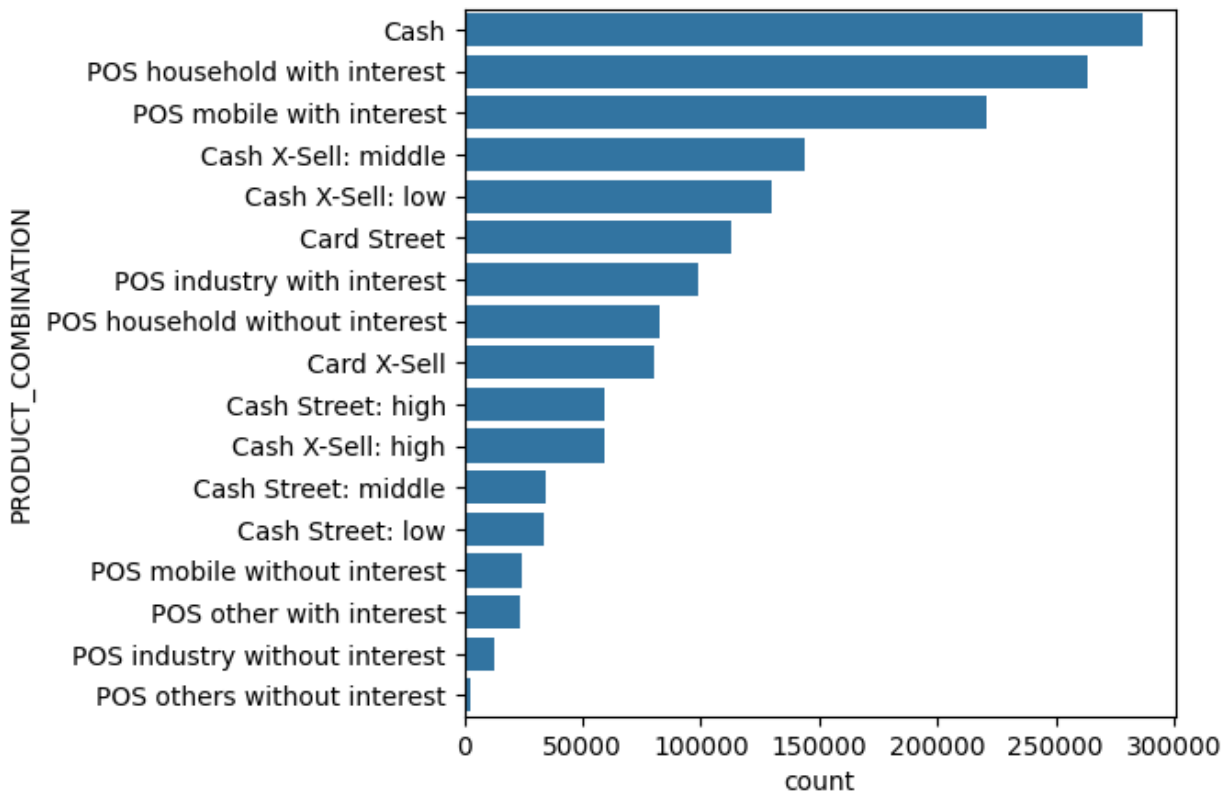

```
sns.countplot(df_p['NAME_SELLER_INDUSTRY'],order=df_p['NAME_SELLER_IND
USTRY'].value_counts().index)
```

```
<Axes: xlabel='count', ylabel='NAME_SELLER_INDUSTRY'>
```




```
sns.countplot(df_p['PRODUCT_COMBINATION'],order=df_p['PRODUCT_COMBINAT
ION'].value_counts().index)
```

```
<Axes: xlabel='count', ylabel='PRODUCT_COMBINATION'>
```




```
df_p['CODE_REJECT_REASON'].value_counts()
```

```
CODE_REJECT_REASON
XAP       1353093
HC         175231
LIMIT       55680
SCO         37467
CLIENT      26436
SCOFR       12811
XNA          5244
VERIF        3535
SYSTEM        717
Name: count, dtype: int64
```

XAP (1353093): This could refer to a rejection related to an application processing error, such as missing or incorrect information in the application.

HC (175231): This might indicate a health-related issue, potentially involving compliance with health regulations or requirements.

LIMIT (55680): This likely pertains to exceeding a predefined limit, such as financial thresholds or operational limits set by a regulatory body.

SCO (37467): This could refer to a score-related issue, possibly indicating that a score (like a credit score) fell below acceptable levels.

CLIENT (26436): This may relate to client-specific issues, such as account status or client eligibility.

SCOFR (12811): This could refer to a specific scoring framework or criteria that was not met.

XNA (5244): This might indicate a rejection due to an unknown or unspecified reason, often used when other codes do not apply.

VERIF (3535): This likely refers to verification issues, such as problems confirming identity or credentials.

cols=df_p.select_dtypes(np.number).drop(columns=["SK_ID_PREV","SK_ID_CURR"]).columns plt.figure(figsize=(20,10)) j = 1 for i in cols: plt.subplot(5, 3, j) sns.boxplot(x=df_p.loc[:, i]) plt.title(i) j += 1

plt.xticks(rotation=45, fontsize=8) plt.tight_layout() plt.show()

POS CASH BALANCE EDA

```
df=pd.read_csv("/content/drive/MyDrive/Colab
Notebooks/POS_CASH_balance.csv")
df.head()
```

```
{"type":"dataframe","variable_name":"df"}
```

```
df.shape
```

```
(10001358, 8)
```

```
df.isnull().sum()
```

```
SK_ID_PREV                   0
SK_ID_CURR                   0
MONTHS_BALANCE               0
CNT_INSTALMENT           26071
CNT_INSTALMENT_FUTURE    26087
NAME_CONTRACT_STATUS         0
SK_DPD                       0
SK_DPD_DEF                   0
dtype: int64
```

```
df.isnull().sum()*100/df.shape[0]
```

```
SK_ID_PREV               0.000000
SK_ID_CURR               0.000000
MONTHS_BALANCE           0.000000
CNT_INSTALMENT           0.260675
CNT_INSTALMENT_FUTURE    0.260835
NAME_CONTRACT_STATUS     0.000000
SK_DPD                   0.000000
SK_DPD_DEF               0.000000
dtype: float64
```

```
df.select_dtypes(include=np.number).describe().round()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 8,\n \"fields\": [\n {\n \"column\": \"SK_ID_PREV\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 3020814.5656346893,\n \"min\": 535847.0,\n \"max\": 10001358.0,\n \"num_unique_values\": 8,\n \"samples\": [\n 1903217.0,\n 1896565.0,\n 10001358.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"SK_ID_CURR\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 3448651.3957597744,\n \"min\": 100001.0,\n \"max\": 10001358.0,\n \"num_unique_values\": 8,\n \"samples\": [\n 278404.0,\n 278654.0,\n 10001358.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"MONTHS_BALANCE\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 3536024.1836563903,\n \"min\": -96.0,\n \"max\": 10001358.0,\n \"num_unique_values\": 8,\n \"samples\": [\n -35.0,\n -28.0,\n 10001358.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"CNT_INSTALMENT\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 3526788.055824791,\n \"min\": 1.0,\n \"max\": 9975287.0,\n \"num_unique_values\": 7,\n \"samples\": [\n 9975287.0,\n 17.0,\n 24.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"CNT_INSTALMENT_FUTURE\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 3526784.3182522696,\n \"min\": 0.0,\n \"max\": 9975271.0,\n \"num_unique_values\": 8,\n \"samples\": [\n 10.0,\n 7.0,\n 9975271.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"SK_DPD\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 3535793.316792337,\n \"min\": 0.0,\n \"max\": 10001358.0,\n \"num_unique_values\": 5,\n \"samples\": [\n 12.0,\n 4231.0,\n 133.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"SK_DPD_DEF\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 3535830.9623564114,\n \"min\": 0.0,\n \"max\": 10001358.0,\n \"num_unique_values\": 5,\n \"samples\": [\n 1.0,\n 3595.0,\n 33.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n }\n]\n}","type":"dataframe"}

```
df.select_dtypes(exclude=np.number).describe()
```

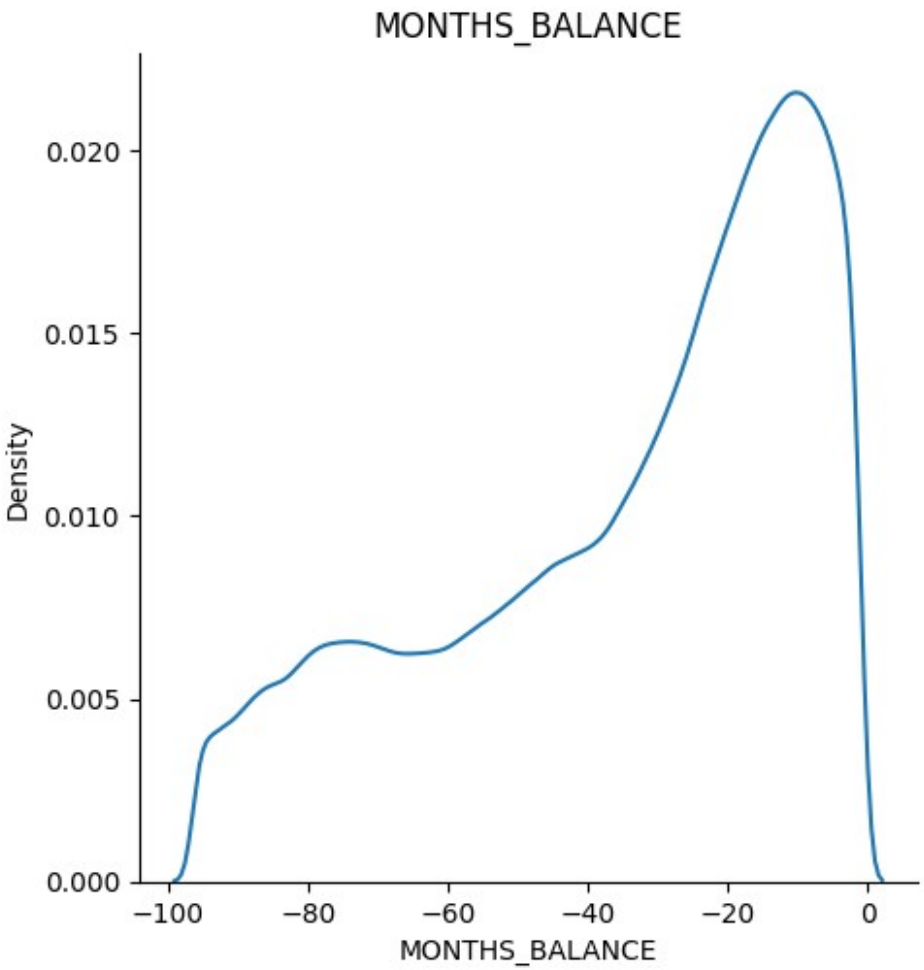
{"summary":"{\n \"name\": \"df\",\n \"rows\": 4,\n \"fields\": [\n {\n \"column\": \"NAME_CONTRACT_STATUS\",\n \"properties\": {\n \"dtype\": \"string\",\n \"num_unique_values\": 4,\n \"samples\": [\n 9,\n \"9151119\",\n \"10001358\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n }\n]\n}","type":"dataframe"}

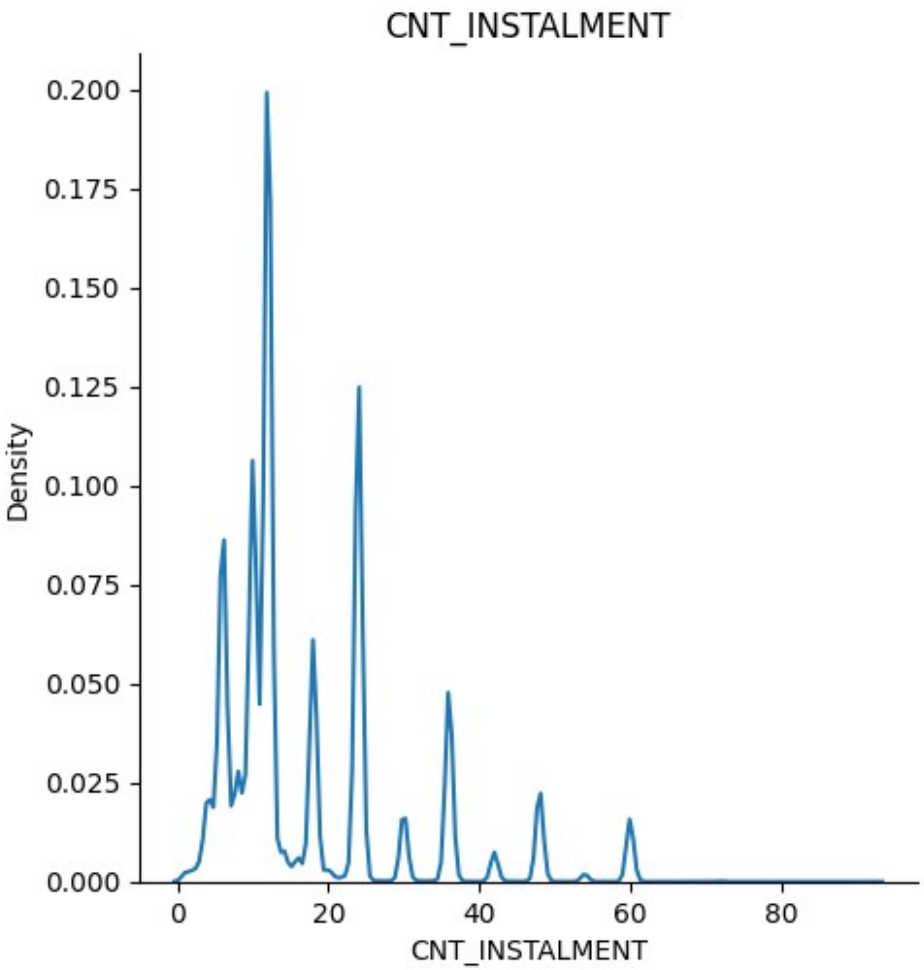
- SK_ID_PREV :ID of previous credit in Home Credit related to loan in our sample. (One loan in our sample can have 0,1,2 or more previous loans in Home Credit)
- SK_ID_CURR : ID of loan in our sample
- MONTHS_BALANCE : Month of balance relative to application date (-1 means the information to the freshest monthly snapshot, 0 means the information at application - often it will be the same as -1 as many banks are not updating the information to Credit Bureau regularly)
- CNT_INSTALMENT : Term of previous credit (can change over time)
- CNT_INSTALMENT_FUTURE : Installments left to pay on the previous credit
- NAME_CONTRACT_STATUS : Contract status during the month
- SK_DPD : DPD (days past due) during the month of previous credit
- SK_DPD_DEF : DPD during the month with tolerance (debts with low loan amounts are ignored) of the previous credit

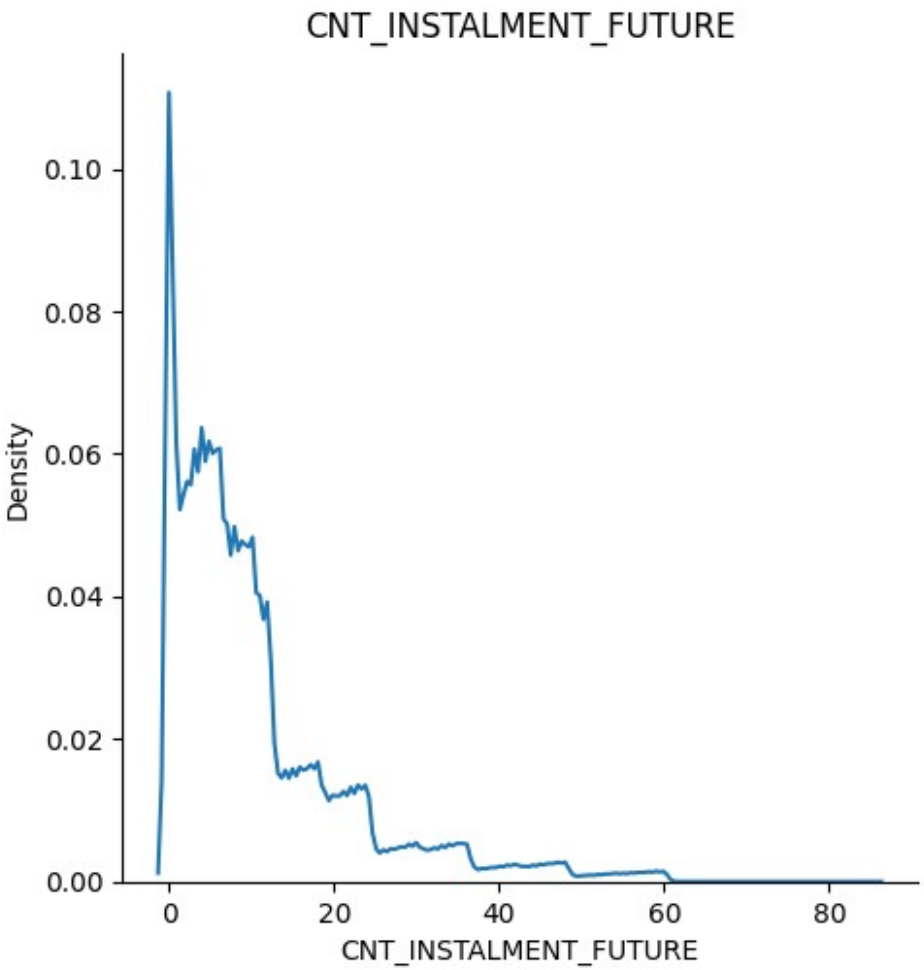
```
df.columns
```

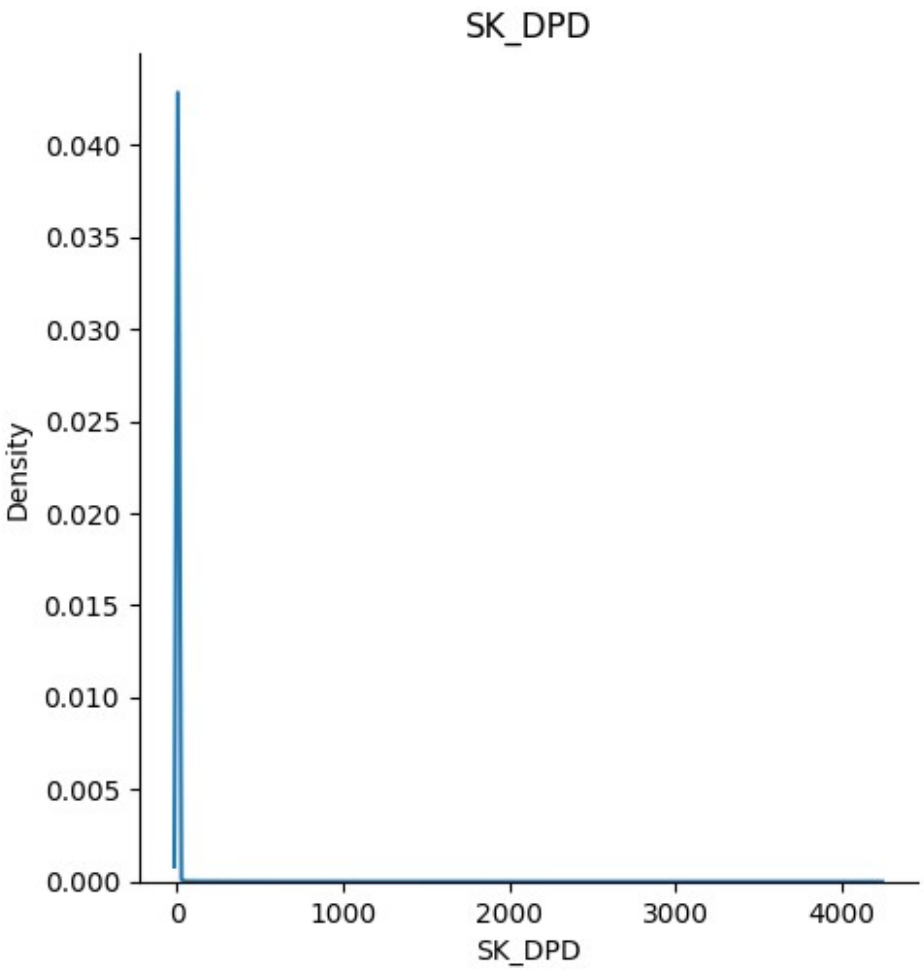
```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'MONTHS_BALANCE', 'CNT_INSTALMENT',
       'CNT_INSTALMENT_FUTURE', 'NAME_CONTRACT_STATUS', 'SK_DPD',
       'SK_DPD_DEF'],
      dtype='object')
```

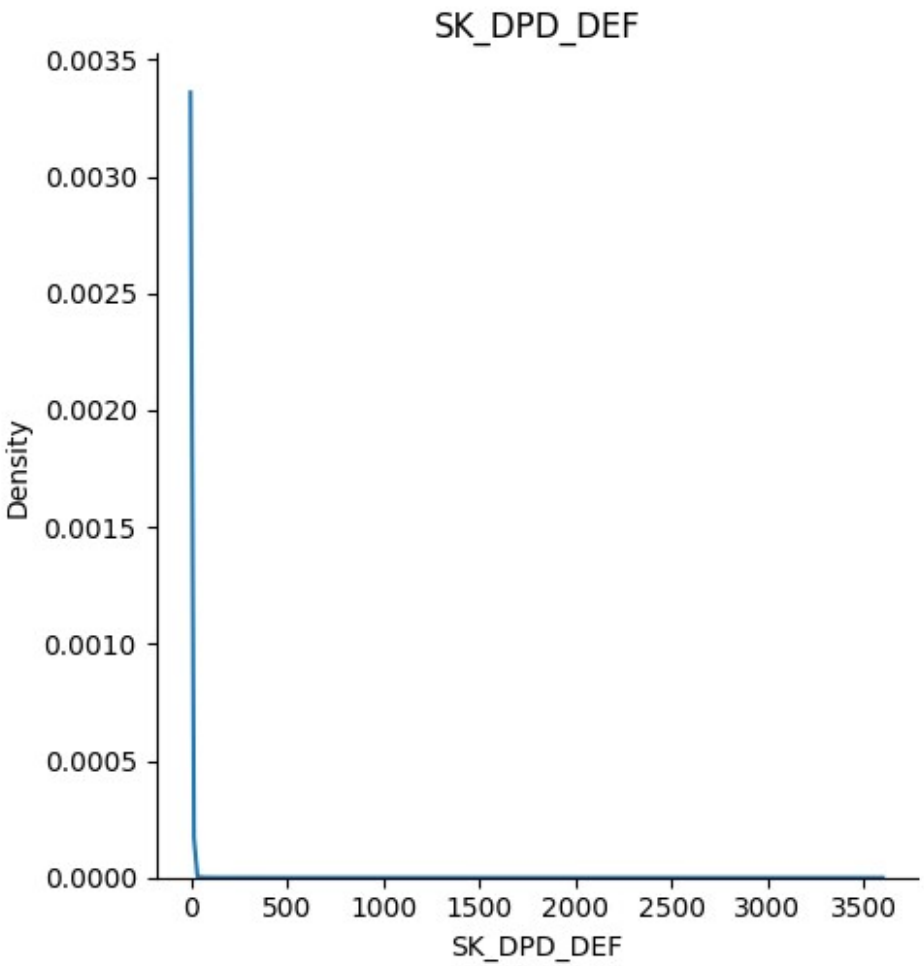
```
cols=['MONTHS_BALANCE','CNT_INSTALMENT','CNT_INSTALMENT_FUTURE','SK_DPD','SK_DPD_DEF']

for i in cols:
    sns.displot(df[i],kind='kde')
    plt.title(i)
    plt.show()
```


MONTHS_BALANCE
Density
0.020
0.015
0.010
0.005
0.000
−100
−80
−60
−40
−20
0
MONTHS_BALANCE

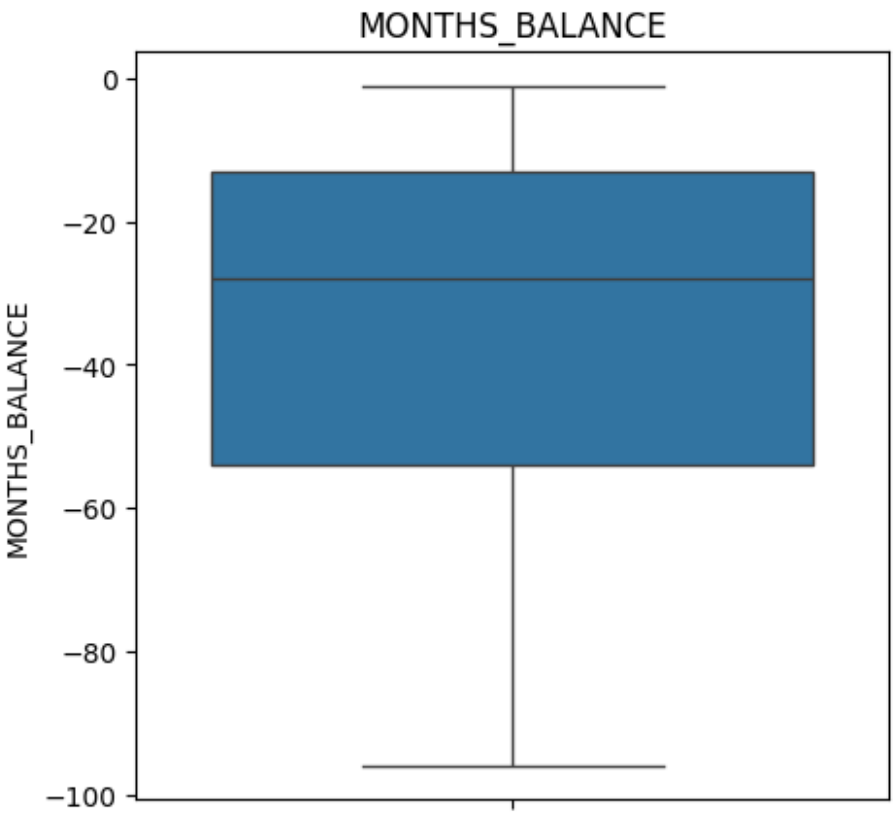

CNT_INSTALMENT
Density
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
20
40
60
80
CNT_INSTALMENT

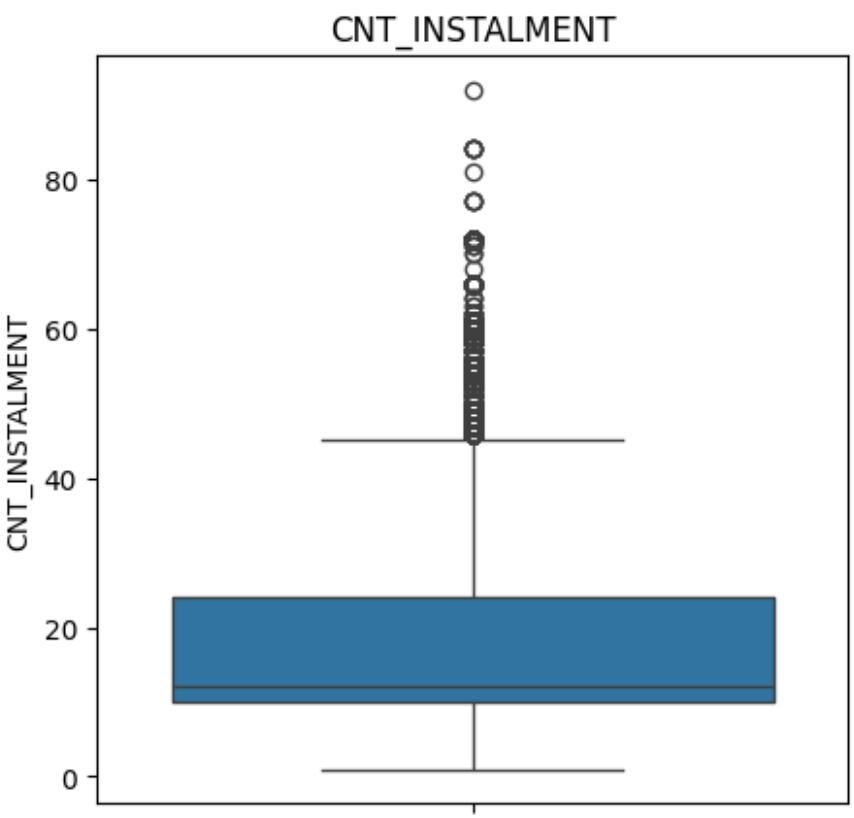

CNT_INSTALMENT_FUTURE
Density
0.10
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80
CNT_INSTALMENT_FUTURE

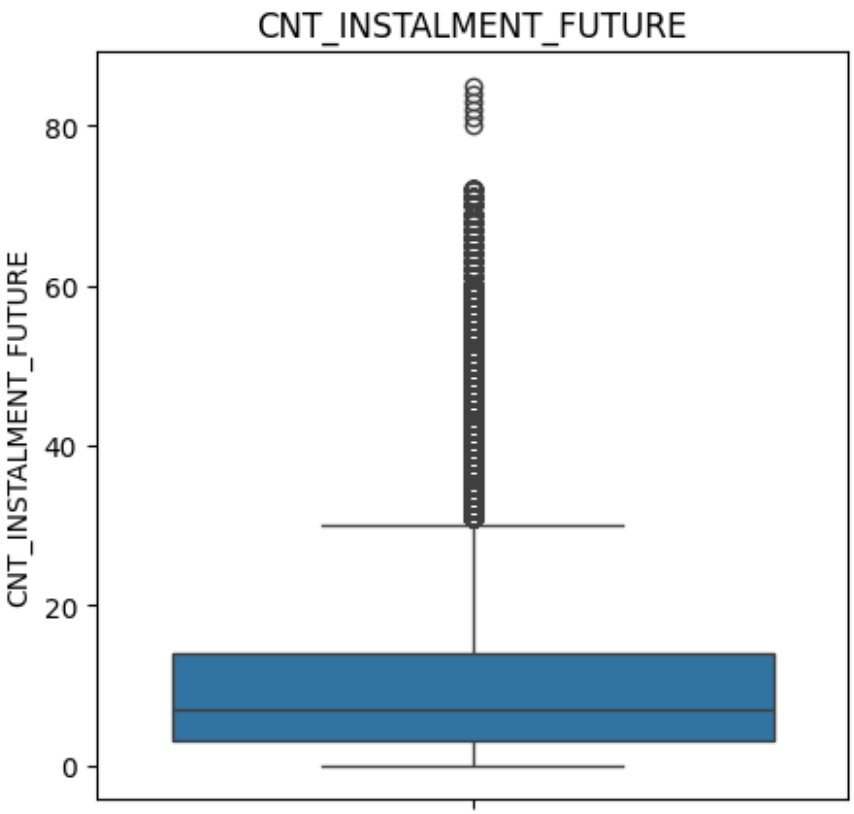

SK_DPD
Density
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
1000
2000
3000
4000
SK_DPD

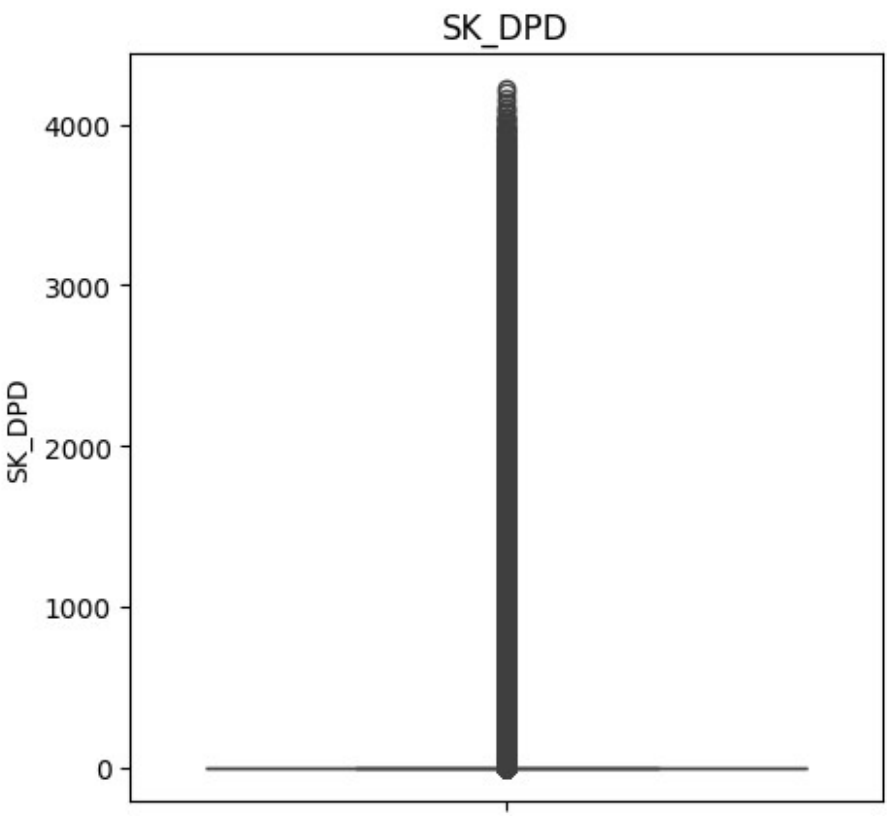


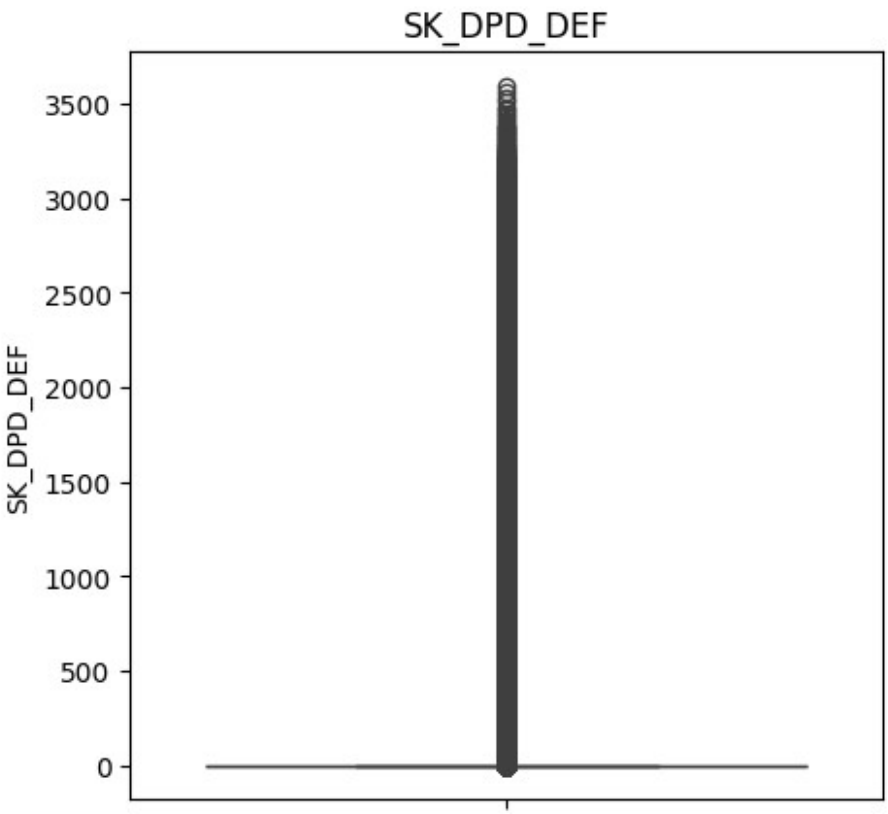

```
cols=['MONTHS_BALANCE','CNT_INSTALMENT','CNT_INSTALMENT_FUTURE','SK_DP
D','SK_DPD_DEF']

for i in cols:
    sns.boxplot(df[i])
    plt.title(i)
    plt.show()
```


MONTHS_BALANCE
MONTHS_BALANCE
0
−20
−40
−60
−80
−100


CNT_INSTALMENT
CNT_INSTALMENT
80
60
40
20
0


CNT_INSTALMENT_FUTURE
CNT_INSTALMENT_FUTURE
0
20
40
60
80


SK_DPD
SK_DPD
0
1000
2000
3000
4000




```
df['NAME_CONTRACT_STATUS'].unique()
```

```
array(['Active', 'Completed', 'Signed', 'Approved',
       'Returned to the store', 'Demand', 'Canceled', 'XNA',
       'Amortized debt'], dtype=object)
```

```
df['NAME_CONTRACT_STATUS'].value_counts()
```

```
NAME_CONTRACT_STATUS
Active                   9151119
Completed                 744883
Signed                     87260
Demand                      7065
Returned to the store       5461
Approved                    4917
Amortized debt               636
Canceled                      15
XNA                            2
Name: count, dtype: int64
```

```
df['SK_DPD'].value_counts()
```

```
SK_DPD
0       9706131
1         21872
2         17358
3         14403
4         12350
5         11046
6          9615
7          8332
8          7360
9          6668
10         6049
11         5392
12         4907
13         4468
14         3944
15         3438
16         3169
17         2980
18         2759
19         2532
20         2322
21         1996
22         1820
23         1586
24         1386
25         1258
26         1101
27          923
28          864
29          700
30          571
31          518
32          512
33          439
35          428
34          401
36          338
37          321
39          289
38          288
40          284
41          274
43          267
44          250
42          237
49          237
45          226
50          225
48          225
```

47	217
46	216
53	210
51	210
59	201
69	196
74	194
67	194
54	194
63	192
78	188
73	187
68	183
52	180
80	179
65	178
66	178
77	175
98	175
109	172
61	172
128	172
97	172
71	172
99	172
79	171
94	171
64	170
111	170
104	170
70	169
125	167
81	167
158	166
100	165
93	164
84	163
62	162
120	161
186	161
130	161
135	161
72	160
141	160
89	159
56	159
60	157
82	156
90	156
85	156

105	156
92	155
139	155
110	155
155	154
126	153
76	153
124	153
133	153
189	152
151	152
96	151
159	151
112	151
102	150
108	150
172	150
57	148
91	147
140	147
58	147
83	146
103	146
115	146
55	146
166	143
161	143
212	143
95	142
134	142
114	142
75	141
127	141
194	141
106	141
229	140
129	139
170	139
165	138
191	137
203	137
217	137
107	136
146	136
201	136
119	135
131	134
204	134
87	133
171	133

163	133
188	133
154	133
142	133
143	133
192	133
156	133
164	132
220	132
168	132
150	131
137	131
101	131
121	131
222	131
123	130
169	129
181	128
216	128
157	128
160	127
231	127
202	127
196	127
145	127
187	126
179	125
132	125
247	125
262	125
173	125
198	125
136	124
86	124
113	124
152	123
246	123
176	123
162	123
118	123
253	123
232	123
233	122
182	122
138	122
372	122
321	122
88	122
219	121
395	120

190	120
200	120
261	119
185	119
284	119
225	119
144	119
292	119
167	119
148	119
197	118
250	118
116	118
207	118
199	117
224	117
245	117
215	116
260	116
290	116
117	115
183	115
249	115
184	114
122	114
264	114
314	114
153	114
353	114
210	113
195	113
277	113
175	113
180	112
227	112
193	112
223	112
218	112
414	112
276	111
257	111
255	111
177	111
323	110
453	110
273	110
352	110
248	110
259	110
230	109

214	109
242	109
252	109
149	108
397	108
316	108
147	108
281	108
234	108
425	107
569	107
278	107
307	107
174	107
377	107
294	107
271	107
283	107
286	106
311	106
322	106
243	105
228	105
263	105
279	105
211	105
308	104
312	104
235	103
291	103
404	103
542	103
240	103
266	103
450	103
206	103
325	103
280	102
632	102
342	102
265	102
226	102
293	102
237	102
373	102
306	101
626	101
428	101
382	101
256	101

221	101
236	100
238	100
315	100
285	100
455	99
332	99
585	99
347	99
335	99
296	99
288	99
304	99
309	98
282	98
295	98
338	98
383	98
406	98
559	97
388	97
328	97
254	97
393	97
463	97
420	97
343	96
239	96
244	96
378	96
367	96
251	96
481	96
686	96
576	95
354	95
268	95
396	95
340	95
418	95
289	95
337	95
258	95
375	95
297	95
413	95
267	94
274	94
305	94
384	94

345	94
209	94
324	94
655	94
213	93
208	93
339	93
784	93
506	93
415	93
649	93
419	93
275	93
400	93
205	93
358	93
327	93
299	93
477	93
475	92
272	92
567	92
467	92
474	92
351	91
178	91
445	91
344	91
357	91
424	91
241	91
346	91
369	90
386	90
363	90
333	90
444	90
476	90
443	90
684	89
595	89
452	89
302	89
300	89
507	89
659	88
417	88
287	88
451	88
313	88

365	88
359	88
356	88
815	88
326	88
303	88
269	88
618	88
433	88
355	87
748	87
385	87
449	87
743	87
364	87
651	87
301	87
320	87
436	87
689	86
341	86
537	86
391	86
298	86
318	86
485	86
671	86
403	85
536	85
421	85
663	85
336	85
566	85
317	85
598	85
505	85
1004	85
426	85
405	85
330	84
349	84
447	84
699	84
766	84
469	84
394	84
392	83
334	83
389	83
410	83

720	83
581	83
823	83
732	83
402	83
534	83
1084	83
310	83
661	83
437	83
329	82
361	82
408	82
374	82
348	82
376	82
442	82
457	81
498	81
366	81
368	81
370	81
412	81
785	81
493	81
464	81
513	81
533	81
416	81
491	81
502	81
687	81
497	81
465	80
1081	80
473	80
270	80
509	80
697	80
434	80
319	80
435	80
516	80
510	79
483	79
494	79
524	79
609	79
528	79
1462	79

563	79
431	79
594	79
668	79
480	78
628	78
456	78
503	78
718	78
541	78
432	78
458	78
851	78
1097	78
430	77
398	77
478	77
479	77
407	77
422	77
588	77
738	77
472	77
572	77
546	77
387	76
379	76
460	76
554	76
409	76
1174	76
438	76
439	76
558	76
446	76
710	75
459	75
511	75
362	75
723	75
610	75
504	75
441	75
565	75
589	75
1371	74
462	74
832	74
401	74
578	74

448	74
380	74
488	74
607	74
371	74
540	74
521	74
350	74
731	74
495	74
429	74
360	73
529	73
512	73
553	73
555	73
527	73
620	73
399	73
500	73
526	73
712	73
692	73
440	73
486	73
544	72
1465	72
602	72
487	72
518	72
523	72
631	72
520	72
517	72
1127	71
548	71
575	71
535	71
634	71
1052	71
471	71
557	71
665	71
713	71
552	71
1647	71
492	71
490	71
411	71

787	71
690	70
570	70
1023	70
525	70
514	70
943	70
501	70
496	70
508	70
590	70
753	70
681	70
597	70
790	69
641	69
759	69
1035	69
556	69
1096	69
882	69
1554	69
390	69
715	69
613	69
547	69
600	69
806	69
706	68
845	68
762	68
751	68
579	68
577	68
568	68
427	68
603	68
1493	68
564	68
454	68
1401	68
539	68
381	68
615	68
466	68
461	68
644	67
519	67
1012	67

650	67
1091	67
531	67
587	67
470	67
662	67
865	67
489	67
789	67
779	66
1524	66
624	66
601	66
639	66
680	66
792	66
657	66
582	66
693	66
637	65
538	65
778	65
1066	65
760	65
543	65
747	65
625	65
616	65
989	65
840	65
1630	65
679	65
532	65
670	65
629	65
530	65
701	64
754	64
550	64
1340	64
630	64
604	64
619	64
1559	64
809	64
522	64
729	64
839	64
596	64

571	64
871	64
764	64
593	64
611	64
562	63
638	63
669	63
1309	63
967	63
1272	63
716	63
812	63
622	63
482	63
771	63
705	63
749	63
709	63
707	63
656	63
658	63
835	63
821	63
499	62
913	62
683	62
549	62
617	62
781	62
797	62
648	62
647	62
586	62
1028	62
756	62
810	62
802	62
605	62
883	62
1046	62
814	62
876	62
869	62
331	62
717	62
1402	62
584	62
672	62

860	62
932	61
695	61
1223	61
847	61
664	61
763	61
1602	61
1188	61
884	61
545	61
722	61
902	61
980	61
1827	61
804	61
1026	61
608	60
974	60
1579	60
1545	60
996	60
468	60
758	60
645	60
627	60
975	60
740	60
635	60
633	60
1171	60
640	60
560	60
696	60
677	59
574	59
728	59
580	59
726	59
561	59
906	59
674	59
1157	59
1638	59
599	59
950	59
654	59
999	59
1121	59

750	58
916	58
614	58
642	58
551	58
765	58
926	58
698	58
822	58
870	58
702	58
1180	58
963	58
794	58
773	58
591	57
1088	57
636	57
623	57
848	57
583	57
901	57
573	57
682	57
1216	57
1235	57
853	57
938	56
934	56
782	56
1274	56
1327	56
1643	56
846	56
1149	56
700	56
1006	56
724	56
646	56
484	56
735	56
1173	56
711	56
808	56
1432	56
515	56
1363	56
727	56
621	56

1036	56
1136	56
678	56
685	56
1126	55
962	55
886	55
774	55
937	55
1027	55
742	55
1020	55
965	55
741	55
666	55
852	55
841	55
739	55
893	55
694	55
909	55
1128	55
1221	55
1030	55
940	55
1361	55
1187	55
1431	55
1328	55
675	55
676	55
768	55
978	54
961	54
843	54
912	54
719	54
998	54
793	54
761	54
816	54
606	54
2192	54
1024	54
947	54
824	54
1858	54
1306	54
919	54

969	54
1215	54
862	54
1370	54
592	54
770	54
725	53
688	53
1570	53
791	53
673	53
1241	53
1555	53
1404	53
780	53
858	53
776	53
1248	53
935	53
643	53
721	53
730	53
612	53
1159	53
875	53
825	52
667	52
956	52
1369	52
1051	52
834	52
691	52
917	52
993	52
1278	52
1487	52
842	52
703	52
1766	52
968	52
959	52
772	52
911	52
1582	52
873	52
733	52
1217	52
899	52
1203	52

2223	52
1615	52
1218	51
1194	51
874	51
777	51
1349	51
1119	51
877	51
945	51
660	51
818	51
1317	51
863	51
744	51
653	51
737	51
1310	51
1005	51
972	51
931	51
1341	51
811	51
976	51
944	51
894	51
1037	51
1067	51
1637	51
907	50
1083	50
830	50
966	50
928	50
801	50
855	50
736	50
775	50
704	50
1350	50
799	50
819	50
800	50
2923	50
1443	50
960	50
1065	50
1490	50
1509	50

1422	50
827	50
757	50
708	50
423	50
829	50
1547	50
844	50
755	50
783	50
896	50
866	50
1107	50
769	49
981	49
837	49
1461	49
904	49
1108	49
923	49
1245	49
867	49
914	49
1249	49
803	49
1054	49
992	49
878	49
1055	49
898	49
924	49
915	49
745	49
868	49
1158	49
954	49
1057	49
859	49
1636	49
1523	49
850	49
885	49
990	48
979	48
849	48
1117	48
854	48
900	48
1759	48
828	48

817	48
1312	48
1447	48
1103	48
820	48
746	48
1064	48
1115	48
991	48
1058	48
881	48
1607	48
955	47
714	47
798	47
903	47
1175	47
833	47
2131	47
788	47
889	47
807	47
1302	47
1650	47
1211	47
1590	47
652	47
888	47
1131	47
1601	47
1697	47
872	47
1391	47
1060	47
1077	47
1000	46
1247	46
1002	46
1551	46
880	46
1039	46
1520	46
1189	46
1150	46
1416	46
2247	46
879	46
957	46
864	46

752	46
1614	46
856	46
1205	46
1219	46
1015	46
1177	46
929	46
1113	45
1118	45
1208	45
1021	45
1271	45
831	45
908	45
1210	45
1042	45
1156	45
1279	45
1452	45
941	45
1600	45
1146	45
1014	45
988	45
838	45
1629	45
1098	44
1303	44
1421	44
1584	44
983	44
1730	44
1284	44
1072	44
964	44
984	44
1050	44
1696	44
1797	44
1190	44
920	44
1705	44
1250	44
933	44
1144	44
1155	44
1315	44
1152	44

925	43
953	43
2039	43
905	43
994	43
1070	43
2557	43
1016	43
796	43
1074	43
805	43
1125	43
1577	43
930	43
1280	43
1674	43
813	43
734	42
1105	42
1101	42
1068	42
1477	42
1282	42
1073	42
1919	42
942	42
1031	42
1343	42
1162	42
958	42
1139	42
2588	42
1828	42
1573	42
1100	42
1033	42
1613	42
1209	42
1345	42
1044	42
1078	41
1227	41
973	41
1129	41
892	41
1043	41
1110	41
1220	41
970	41

1888	41
1413	41
1116	41
1251	41
997	41
1299	41
1011	41
987	41
767	41
897	41
1980	41
1337	41
1395	41
1034	41
1122	41
1181	41
1007	41
1553	41
1082	41
1207	41
910	41
1522	41
2404	41
891	41
1059	41
1167	41
1047	40
1200	40
857	40
1429	40
1463	40
2496	40
1585	40
1364	40
1297	40
1424	40
995	40
1264	40
2435	40
1049	40
1978	40
1062	40
836	40
1141	40
2070	40
985	40
1019	40
1018	40
1266	40

1333	39
1213	39
2770	39
1104	39
1796	39
946	39
927	39
1093	39
1147	39
1497	39
2862	39
1022	39
971	39
1232	39
1300	39
1085	39
1325	39
1254	39
1332	39
1008	39
1552	39
1134	39
795	39
1646	39
1735	39
1087	39
1164	39
1079	39
890	38
1373	38
2162	38
1045	38
1003	38
1143	38
1061	38
1276	38
1092	38
1239	38
1202	38
1186	38
1330	38
1095	38
1053	38
1086	38
1514	38
1192	38
1258	38
1089	38
1358	38

2343	38
1779	38
1285	38
1407	38
951	38
1009	38
1242	38
1294	38
1236	38
1025	37
1153	37
1176	37
1001	37
1165	37
1393	37
1169	37
1676	37
1453	37
1029	37
1707	37
1195	37
1114	37
1621	37
1120	37
1392	37
1094	37
1492	37
1990	37
1268	37
936	37
1112	37
1179	37
1041	37
1204	37
1137	37
1380	37
1334	37
922	37
2253	37
1263	37
1010	36
1135	36
1142	36
921	36
2284	36
1075	36
952	36
1658	36
861	36

1183	36
1063	36
1667	36
1515	36
918	36
1133	36
1612	36
2954	36
1736	36
895	36
1417	36
1140	36
1456	35
1389	35
1296	35
986	35
1388	35
1281	35
1109	35
1339	35
1124	35
1197	35
1419	35
1231	35
887	35
1468	35
1593	35
1949	35
786	35
1400	35
1398	35
1198	35
1686	35
1480	35
1233	35
2527	35
1356	35
1099	35
1331	35
1353	35
1372	34
1288	34
1238	34
1168	34
1591	34
2801	34
1305	34
3288	34
1448	34

1295	34
1680	34
1123	34
1324	34
1644	34
1576	34
1038	34
1737	34
1606	34
1013	34
1918	34
1495	34
1485	34
1244	34
1184	34
1308	34
1435	34
1484	34
939	34
1575	34
1275	34
1265	34
982	34
1496	34
1494	33
1214	33
1521	33
1292	33
1229	33
1090	33
1539	33
1438	33
3227	33
1616	33
1145	33
1199	33
1367	33
1889	33
1269	33
1352	33
1166	33
1243	33
1437	33
1253	33
1346	33
1172	33
1498	33
949	33
1677	33

1423	33
2314	33
1455	33
1469	33
1225	33
1311	33
1056	33
1259	33
1290	33
1185	33
1323	33
1224	32
1319	32
1111	32
1583	32
1161	32
1193	32
1581	32
1102	32
1486	32
1741	32
1196	32
1950	32
2011	32
1675	32
1106	32
1017	32
1293	32
1076	32
1365	32
1917	32
1770	32
1478	32
2100	32
1706	32
2161	32
1228	32
1425	32
2831	32
2374	32
1357	32
1080	32
1375	32
826	31
1649	31
1698	31
1454	31
1474	31
1529	31

1887	31
2282	31
1789	31
1565	31
1368	31
1138	31
1237	31
1377	31
2193	31
1257	31
1546	31
1517	31
1449	31
1385	31
1262	31
1032	31
1732	31
2009	31
948	31
1499	30
977	30
1857	30
1322	30
1460	30
1403	30
1483	30
1426	30
1261	30
1289	30
1411	30
1362	30
2465	30
2008	30
1234	30
1491	30
1408	30
2015	30
1376	30
1925	30
1619	30
1069	30
1342	30
1354	30
1768	30
2345	30
1355	30
1833	30
1240	30
1592	30

1728	30
1394	30
1252	30
1384	30
1212	30
1511	30
1170	30
1439	30
1578	30
1270	29
1335	29
1336	29
1772	29
1329	29
2648	29
1230	29
1569	29
1415	29
1201	29
1374	29
1442	29
2892	29
1589	29
1307	29
1267	29
1396	29
1382	29
2101	29
1277	29
1386	29
1782	29
1530	29
2041	29
1316	29
1163	29
1598	29
1246	29
1729	29
1760	29
1751	29
1160	29
1321	29
3166	29
1711	28
2739	28
1182	28
1473	28
1588	28
2312	28
1433	28

1406	28
1508	28
1880	28
1506	28
1864	28
1560	28
2102	28
1273	28
1753	28
1542	28
1947	28
1414	28
2709	28
1318	28
1412	28
1567	28
1071	28
2618	28
1399	28
1948	28
3319	28
1326	28
1894	28
2649	28
1255	28
1459	27
1821	27
2072	27
1527	27
1668	27
1673	27
1526	27
2283	27
1436	27
1405	27
2558	27
1659	27
1820	27
2466	27
1722	27
1802	27
1314	27
1558	27
1040	27
1535	27
1512	27
1366	27
1516	27
1859	27

1790	27
1148	27
1338	27
1758	27
1286	27
1410	27
1360	27
1886	27
1518	27
1981	27
2076	27
1450	27
1505	27
1920	26
1378	26
2984	26
1557	26
1283	26
1301	26
1819	26
1984	26
1665	26
1541	26
1645	26
1348	26
2526	26
1718	26
1430	26
2020	26
1951	26
1206	26
1642	26
1609	26
2046	26
2224	26
1504	26
1572	26
3258	26
1479	26
1767	26
1538	26
1738	26
2467	26
1222	26
1048	26
2376	26
1596	26
1427	26
1260	26

1470	25
1814	25
1881	25
1783	25
1178	25
1911	25
1989	25
1608	25
1132	25
2313	25
1383	25
1500	25
1743	25
2111	25
1871	25
3135	25
1531	25
1467	25
1720	25
1256	25
1537	25
1151	25
1154	25
1620	25
1379	25
1536	25
1804	25
1528	25
1409	25
1942	25
1434	25
1475	25
2373	25
2251	25
2255	25
2133	25
1904	25
1420	25
1298	25
1704	25
1700	25
1387	25
2106	25
2071	25
1347	25
1359	25
1850	25
1849	25
1757	25

1130	25
1287	24
1877	24
1191	24
1708	24
2063	24
1502	24
1765	24
1631	24
1749	24
1561	24
1803	24
1507	24
1902	24
2347	24
2040	24
1446	24
1692	24
1935	24
1533	24
2893	24
1544	24
1852	24
1594	24
1813	24
1633	24
1798	24
1640	24
1548	24
1657	24
1843	24
1320	24
1856	24
1503	24
1651	24
1844	24
1954	24
1605	24
1690	24
2026	24
1457	24
2074	24
1681	24
2095	24
1418	24
2132	24
1458	24
1788	24
1445	23

1890	23
1703	23
2222	23
1755	23
1693	23
1861	23
2057	23
1851	23
3592	23
1661	23
2678	23
2043	23
1883	23
1291	23
1795	23
1710	23
2617	23
1622	23
1466	23
2679	23
2168	23
1761	23
1818	23
1967	23
1834	23
1226	23
3076	23
1663	23
1959	23
1901	23
1826	23
2254	23
2405	23
1839	23
1313	23
1428	23
2013	23
2051	23
1774	23
1836	23
1540	23
1867	23
1562	23
1390	23
1351	23
3015	23
1441	23
1549	23
1734	23

2028	23
2042	23
1727	23
1639	23
2089	22
1773	22
1840	22
2137	22
2156	22
2047	22
2135	22
1344	22
1982	22
1816	22
1604	22
3531	22
1586	22
1618	22
1682	22
1688	22
1683	22
1972	22
2030	22
1699	22
2202	22
2045	22
2166	22
1695	22
1873	22
2163	22
2497	22
2315	22
1835	22
1910	22
1724	22
1662	22
2436	22
1913	22
3074	22
1810	22
1501	22
1626	22
2087	22
1481	22
1830	22
1808	22
1715	22
2179	22
1745	22

1627	22
1451	22
1908	22
2118	22
1938	22
1874	22
1930	22
2437	21
1898	21
1726	21
2017	21
1670	21
1472	21
1747	21
1992	21
1687	21
1986	21
1712	21
2120	21
1689	21
2141	21
1550	21
1932	21
2033	21
2050	21
1653	21
1780	21
1863	21
2173	21
2061	21
1304	21
2710	21
1623	21
3653	21
2124	21
2832	21
1777	21
1807	21
2323	21
1471	21
2589	21
2093	21
2010	21
1694	21
1534	21
1740	21
2406	21
1476	21
1444	21

1381	21
2227	21
3196	21
1812	21
1785	21
2080	21
1838	21
2620	21
1482	21
1654	21
2012	21
1892	21
1985	21
2016	20
2107	20
1936	20
1603	20
1624	20
1628	20
1895	20
2246	20
1714	20
1664	20
1685	20
2104	20
1941	20
2185	20
2003	20
1799	20
2049	20
1769	20
1632	20
1870	20
1995	20
2034	20
2680	20
2741	20
1996	20
1966	20
1955	20
1564	20
3349	20
1869	20
2498	20
2286	20
2024	20
2170	20
2245	20
2103	20

1776	20
1926	20
1489	20
2081	20
3105	20
2138	20
1721	20
1733	20
2408	20
1595	20
2316	20
1510	20
2194	20
1568	20
1953	20
1801	20
2619	20
1912	20
2294	20
2134	19
1896	19
2252	19
2053	19
1866	19
1974	19
1791	19
1655	19
1809	19
2196	19
1940	19
1963	19
2346	19
1979	19
1921	19
1800	19
1997	19
1713	19
1784	19
2153	19
1771	19
1792	19
1717	19
1742	19
1825	19
1599	19
1764	19
1731	19
1563	19
2181	19

2078	19
2201	19
1532	19
1845	19
2378	19
1923	19
1860	19
2055	19
2073	19
2147	19
1846	19
2239	19
1957	19
2165	19
2002	19
1635	19
1841	19
2233	19
1973	19
1566	19
1571	19
2242	19
2216	19
2069	19
2740	19
1660	19
1977	19
2248	19
1965	19
2439	19
1928	19
2142	19
2058	19
1971	19
1868	19
3197	19
3013	19
1956	19
1669	19
2263	19
2123	19
1781	19
1488	19
1580	19
1999	19
1853	18
2708	18
2276	18
2110	18

3500	18
2150	18
1634	18
2148	18
2257	18
1969	18
1931	18
1987	18
2054	18
2214	18
1805	18
1875	18
2300	18
1865	18
2184	18
2022	18
1440	18
1944	18
2559	18
2221	18
1691	18
2306	18
1970	18
1723	18
1829	18
1943	18
1822	18
2154	18
2285	18
1900	18
3045	18
2032	18
2005	18
1397	18
1961	18
1666	18
3441	18
1897	18
1672	18
2367	18
1946	18
1543	18
1823	18
1893	18
1905	18
1671	18
2232	18
1464	18
2172	18
1993	18

2083	18
3380	18
2262	18
1748	18
3623	18
1625	18
1702	18
2092	18
1882	18
1879	18
2361	18
2082	18
2145	18
2384	17
1794	17
1787	17
2344	17
2210	17
2258	17
1914	17
1716	17
2273	17
2229	17
2112	17
1848	17
2077	17
2460	17
1746	17
1934	17
1806	17
2985	17
1752	17
1754	17
1929	17
3257	17
2398	17
1610	17
2105	17
3014	17
1994	17
2441	17
1831	17
2288	17
1656	17
2007	17
2098	17
1916	17
2296	17
3439	17

2215	17
2062	17
1574	17
2023	17
2267	17
1786	17
1855	17
2319	17
1775	17
1513	17
1958	17
1899	17
2217	17
1587	17
1778	17
2309	17
2307	17
2528	17
1725	17
2155	16
2325	16
2380	16
1597	16
2293	16
2281	16
2178	16
2500	16
2084	16
2125	16
1750	16
2332	16
2449	16
1744	16
2126	16
2208	16
2108	16
2277	16
1684	16
1793	16
1811	16
1611	16
1709	16
1679	16
1862	16
3470	16
1847	16
1763	16
2019	16
2014	16

2354	16
2143	16
1962	16
1924	16
1641	16
1906	16
2038	16
1988	16
1891	16
2355	16
2130	16
1832	16
2418	16
2199	16
2399	16
2651	16
2187	16
2230	16
2048	16
2200	16
3684	16
2226	16
2338	16
2114	16
2269	16
3044	16
1817	16
2271	16
2457	16
2712	15
2174	15
2518	15
2726	15
2151	15
2116	15
2018	15
2353	15
2198	15
2000	15
2236	15
2031	15
2337	15
2261	15
2426	15
2677	15
2177	15
1927	15
2036	15
2647	15

1922	15
2235	15
2416	15
2191	15
2375	15
2086	15
2091	15
2001	15
2128	15
1964	15
2264	15
2160	15
2587	15
2586	15
2419	15
1885	15
1525	15
2211	15
2231	15
2064	15
2159	15
2434	15
2066	15
3046	15
2204	15
1739	15
2800	15
2357	15
2027	15
2429	15
2590	15
2175	15
2711	15
2298	15
2004	15
2065	15
2531	15
2476	15
2059	15
2292	15
2206	15
1975	14
2579	14
2391	14
2186	14
2301	14
1756	14
2953	14
2334	14

2115	14
1960	14
2275	14
1824	14
2037	14
2318	14
2075	14
2802	14
2183	14
2358	14
2304	14
2469	14
2317	14
2385	14
2119	14
2225	14
2234	14
2190	14
1652	14
2085	14
2113	14
2582	14
2342	14
2212	14
1933	14
2616	14
2266	14
2447	14
2240	14
2331	14
2290	14
2430	14
1519	14
1678	14
2203	14
2571	14
2407	14
2146	14
2237	14
3137	14
2205	14
2415	14
2060	14
1815	14
1903	14
1842	14
3561	14
1952	14
1876	14

1907	14
2122	14
1719	14
2604	14
2260	14
2410	14
2613	14
2445	13
2428	13
2350	13
1915	13
1762	13
2463	13
2386	13
2402	13
1939	13
2611	13
1998	13
2674	13
2401	13
2164	13
3107	13
2552	13
2442	13
2197	13
2265	13
2533	13
2067	13
2411	13
1909	13
2209	13
2771	13
3562	13
2368	13
2452	13
1617	13
2665	13
2510	13
2390	13
2220	13
2176	13
2270	13
2329	13
2359	13
2326	13
2495	13
2743	13
3378	13
2278	13

2480	13
2330	13
2079	13
2006	13
2491	13
2535	13
2524	13
2512	13
2365	13
2140	13
2139	13
3410	13
2521	13
2088	13
2773	13
2377	13
2349	13
2392	13
1878	13
2167	13
2472	13
2021	13
2169	13
2094	13
2422	12
2238	12
2650	12
2702	12
2149	12
2158	12
2387	12
2519	12
2543	12
1945	12
2256	12
2478	12
2171	12
2025	12
2096	12
2481	12
2529	12
2109	12
2479	12
2136	12
1701	12
2682	12
2924	12
2388	12
2097	12

2336	12
2311	12
1854	12
2610	12
2379	12
1991	12
2573	12
2327	12
3472	12
2099	12
2451	12
1872	12
2189	12
2592	12
1884	12
2641	12
2369	12
2291	12
2339	12
2537	12
2861	12
2487	12
3409	12
2328	12
2483	12
2688	12
2321	12
2863	12
2446	12
2671	12
2525	12
2602	12
2453	12
2144	12
2489	12
2983	12
2561	12
2370	12
2423	12
2340	12
3136	12
2195	12
1837	12
2414	11
2769	11
2490	11
2493	11
3411	11
2207	11

3104	11
2420	11
2289	11
2297	11
1648	11
2772	11
2052	11
2438	11
2127	11
2117	11
2458	11
2764	11
2804	11
2372	11
2643	11
2488	11
2056	11
2129	11
2393	11
2433	11
2982	11
2068	11
2482	11
2324	11
2554	11
2448	11
2395	11
2646	11
2705	11
2188	11
1937	11
2542	11
2581	11
2504	11
2506	11
2672	11
2576	11
2738	11
2681	11
2551	11
1976	11
2432	11
2279	11
1556	11
2360	11
2514	11
2219	11
2684	11
2351	11

2268	11
2833	11
2520	11
2470	11
2749	11
2585	11
2389	11
2396	11
2035	11
2865	11
2550	10
3622	10
2544	10
2295	10
2776	10
2716	10
2661	10
2745	10
1983	10
2511	10
2243	10
2090	10
2837	10
2450	10
2412	10
2459	10
2322	10
2507	10
2596	10
2421	10
2484	10
2302	10
2695	10
2494	10
2555	10
2644	10
3167	10
2503	10
2540	10
2556	10
2371	10
2896	10
3502	10
2732	10
2464	10
2632	10
3168	10
3289	10
2694	10

2634	10
2180	10
3350	10
2303	10
2044	10
2259	10
3043	10
2513	10
2549	10
2382	10
2462	10
2723	9
2736	9
2879	9
3714	9
2244	9
3745	9
2417	9
2308	9
2515	9
2565	9
2320	9
2468	9
2502	9
2635	9
2381	9
2603	9
3106	9
2626	9
2029	9
2473	9
2703	9
2841	9
2409	9
3379	9
2475	9
2580	9
2692	9
2335	9
2631	9
2733	9
2659	9
2642	9
2623	9
2622	9
2364	9
2363	9
2834	9
2539	9
2806	9

1968	9
2570	9
2272	9
2655	9
2707	9
2548	9
2633	9
2249	9
2456	9
2657	9
2299	9
2663	9
2640	9
2461	9
2676	9
2424	8
2609	8
3501	8
2594	8
2362	8
2957	8
2508	8
2523	8
2228	8
2425	8
2250	8
2545	8
2431	8
2443	8
2699	8
2814	8
2828	8
2454	8
2696	8
3049	8
2121	8
2890	8
2352	8
2795	8
2598	8
2356	8
2753	8
2653	8
2607	8
2568	8
2757	8
2787	8
2871	8
2848	8

2686	8
2583	8
2600	8
2784	8
2541	8
3228	8
2883	8
2567	8
2638	8
2492	8
2383	8
2797	8
2152	8
2614	8
2287	8
2955	8
2756	8
2690	8
2766	8
2816	8
2810	8
2794	8
2566	8
3654	8
2182	8
2763	8
2627	8
2241	8
2902	8
2218	8
3198	8
2827	8
2471	8
2348	8
2735	8
2305	7
3318	7
2856	7
2601	7
2720	7
2341	7
2825	7
2547	7
2737	7
2917	7
2477	7
2894	7
3006	7
2538	7

2574	7
2605	7
3348	7
3533	7
2900	7
2595	7
2858	7
2869	7
2845	7
2807	7
3226	7
2798	7
2767	7
2818	7
2280	7
2719	7
2366	7
2444	7
2747	7
2213	7
2572	7
2952	7
2564	7
2474	7
2403	7
2563	7
2830	7
2599	7
3053	7
2625	7
2742	7
2799	7
2427	7
2976	7
2889	7
2898	7
3075	7
2857	7
2668	7
2822	7
2704	7
2693	7
2615	7
2628	7
2780	7
2397	7
2788	7
2919	7
2662	7

2724	6
3743	6
2400	6
2915	6
3138	6
2796	6
2909	6
2936	6
2485	6
2729	6
2670	6
3110	6
2812	6
2967	6
3866	6
2501	6
3469	6
2808	6
3159	6
3774	6
3037	6
2926	6
2988	6
2996	6
2951	6
2965	6
2394	6
2760	6
2645	6
2981	6
2751	6
2777	6
2904	6
2946	6
2843	6
2826	6
3199	6
2910	6
2932	6
2612	6
2875	6
3016	6
2718	6
3229	6
3024	6
2947	6
2755	6
2413	6
2666	6

2722	6
2157	6
2715	6
2725	6
2522	6
2562	6
2803	6
2852	6
3806	6
2578	6
2868	6
2941	6
2636	6
2560	6
2758	6
2499	6
2584	6
2629	6
2673	6
2486	6
2597	6
2546	6
2621	6
2553	6
2575	6
2669	6
2880	6
2768	6
3067	5
2823	5
2978	5
2817	5
2914	5
2997	5
2440	5
2944	5
3317	5
2717	5
2675	5
2608	5
2854	5
3114	5
2859	5
2779	5
2959	5
3744	5
2516	5
2811	5
2654	5

2274	5
3085	5
2509	5
2783	5
2762	5
2752	5
3169	5
2730	5
2855	5
2658	5
3141	5
3068	5
2824	5
2849	5
2530	5
2791	5
2992	5
2517	5
2569	5
3988	5
2975	5
3713	5
2835	5
2884	5
2846	5
2536	5
2656	5
2577	5
3835	5
2790	5
2872	5
2455	5
3408	5
2873	5
3100	5
2697	5
2961	5
3018	5
2685	5
2734	5
2886	5
3563	5
2591	5
3347	5
3165	5
2505	5
2786	5
2945	5
3775	5

2532	5
2333	5
2781	5
2847	5
2754	5
2310	5
2534	5
2933	5
3532	5
2782	5
2935	5
2929	5
2731	5
3896	5
3128	5
3715	5
2819	5
2922	5
3077	5
2701	5
3471	5
3084	5
2839	5
2793	5
2829	5
3007	5
2950	4
3683	4
3440	4
2885	4
3042	4
2948	4
2940	4
3927	4
4019	4
3124	4
3776	4
3103	4
3134	4
2913	4
2700	4
3091	4
3027	4
2963	4
2906	4
2867	4
3564	4
2774	4
3594	4

3195	4
2891	4
3001	4
2785	4
3055	4
2765	4
2746	4
3108	4
2908	4
3145	4
3062	4
2706	4
2990	4
2931	4
3030	4
2921	4
2789	4
2714	4
3320	4
2920	4
2969	4
2887	4
3022	4
2864	4
2960	4
2639	4
2850	4
2876	4
3593	4
2853	4
2637	4
2667	4
2844	4
2664	4
3036	4
2624	4
3019	4
2727	4
2877	4
2974	4
3039	4
2964	4
3097	4
2821	4
2907	4
2938	4
2775	4
2888	4
2713	4

3131	4
3038	3
2930	3
3051	3
3172	3
3837	3
2836	3
2838	3
2943	3
3069	3
2895	3
3003	3
3072	3
3020	3
3928	3
2778	3
3034	3
3200	3
2721	3
3000	3
3804	3
2878	3
2987	3
3957	3
3176	3
3116	3
3011	3
3129	3
3152	3
3028	3
2916	3
3234	3
3122	3
2860	3
2687	3
3023	3
2792	3
3080	3
2698	3
3685	3
3060	3
2970	3
3958	3
2630	3
3032	3
2903	3
3117	3
2840	3
2993	3

2994	3
2991	3
2934	3
3079	3
2691	3
3206	3
2748	3
3026	3
3260	3
2683	3
3058	3
3867	3
3155	3
2899	3
2927	3
3088	3
3009	3
3008	3
2759	3
3682	3
3012	3
2966	3
2815	3
3250	3
3256	3
3164	3
2911	3
3220	3
3312	3
2652	3
3189	3
3897	3
3073	3
3805	3
3047	2
3162	2
2866	2
3959	2
3340	2
2918	2
2851	2
3503	2
3070	2
3276	2
3061	2
2973	2
3184	2
3089	2
3204	2

2881	2
3368	2
3215	2
3245	2
3123	2
2593	2
3175	2
2972	2
3202	2
3002	2
3183	2
2942	2
2905	2
3081	2
3186	2
2728	2
3033	2
3095	2
3211	2
3064	2
2689	2
3094	2
2999	2
3092	2
3153	2
3306	2
3230	2
3173	2
3322	2
2949	2
3021	2
3156	2
2805	2
3112	2
2977	2
3261	2
3119	2
2761	2
3150	2
3148	2
2744	2
3096	2
3287	2
3190	2
3438	2
3187	2
3098	2
3214	2
2606	2
2928	2

3057	2
3281	2
2820	2
3160	2
3625	2
3208	2
3251	2
3102	2
3216	2
2842	2
3530	2
3099	2
2939	2
3041	2
3147	2
3010	2
2912	2
3083	2
2986	2
3591	2
3031	2
3056	2
3063	2
2813	2
2901	2
2660	2
2979	2
3898	2
2971	2
3712	2
3180	2
2750	2
2874	2
3209	2
2882	2
2962	2
3773	2
3836	2
3291	2
4050	2
3115	1
3506	1
4172	1
4078	1
3316	1
3377	1
2937	1
2980	1
3716	1

3126	1
3989	1
3337	1
3475	1
3247	1
3987	1
4200	1
4080	1
3865	1
3929	1
3277	1
3414	1
3225	1
3534	1
3381	1
3146	1
3275	1
3219	1
3118	1
3442	1
3357	1
3351	1
3259	1
3746	1
3149	1
3065	1
4108	1
4047	1
3236	1
3154	1
3371	1
3388	1
3203	1
3238	1
3231	1
3807	1
3595	1
3059	1
3267	1
3280	1
3244	1
3177	1
3315	1
3191	1
3329	1
3353	1
3444	1
4110	1
3352	1

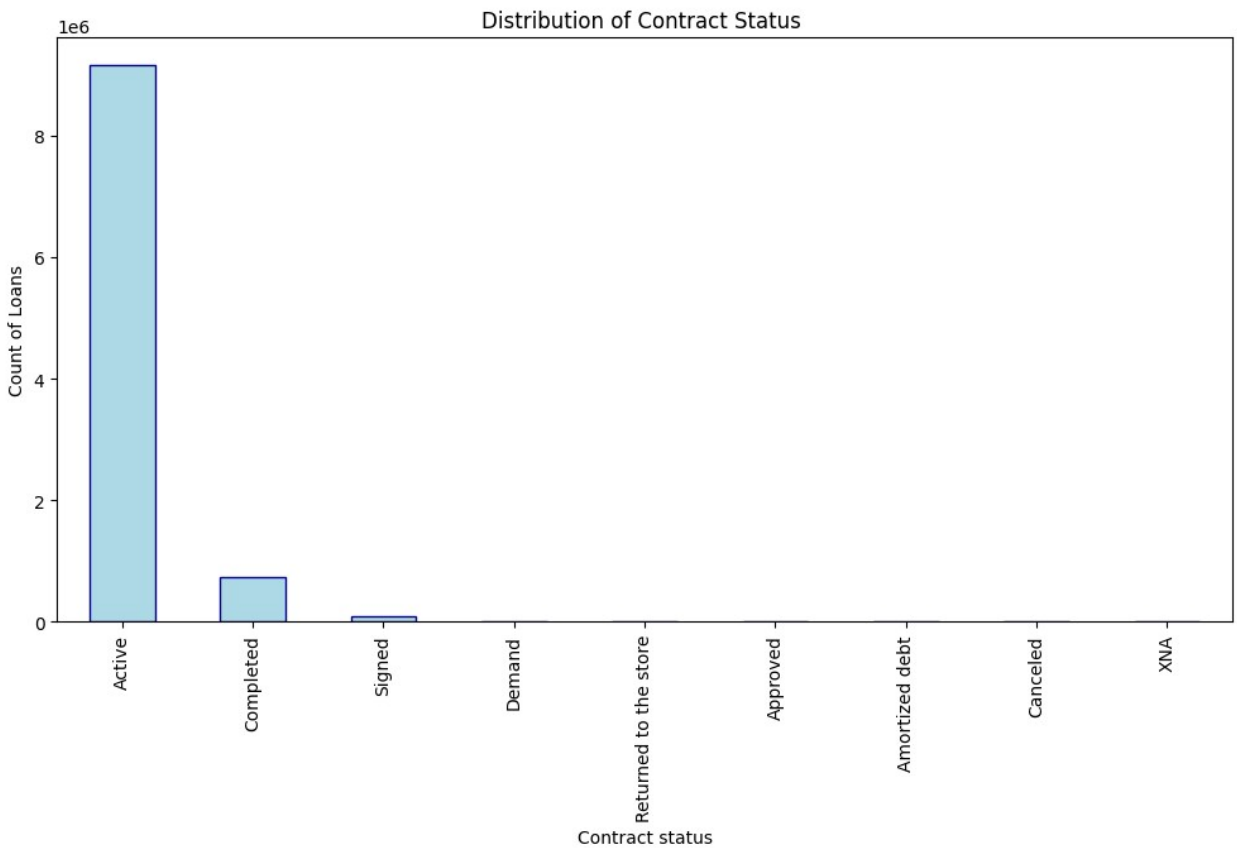
3142	1
2809	1
3396	1
3273	1
4049	1
3132	1
3113	1
2989	1
3163	1
4141	1
3054	1
2998	1
3235	1
3052	1
3071	1
3004	1
3144	1
3093	1
2870	1
3549	1
3731	1
3641	1
3580	1
3488	1
3335	1
3307	1
3383	1
3218	1
3264	1
3221	1
3246	1
3300	1
3330	1
3373	1
3342	1
3407	1
3361	1
3269	1
3468	1
3239	1
3025	1
3086	1
3178	1
3185	1
3298	1
3237	1
3048	1
3242	1
3310	1

4231	1
3565	1
3303	1
3284	1
3282	1
3427	1
3240	1
3125	1
3233	1
3309	1
3301	1
3253	1
3265	1
3272	1
3066	1
3087	1
3133	1
3457	1
3519	1
3610	1
3127	1
3292	1
3181	1
3101	1
3624	1
3120	1
3341	1
3192	1
3035	1
3296	1
3248	1
3222	1
3241	1
3278	1
3270	1
3429	1
3111	1
3143	1
3040	1
3179	1
3158	1
2995	1
3672	1
3700	1
2897	1
2958	1
3005	1
3366	1
3161	1

```
3217          1
3323          1
3399          1
3082          1
Name: count, dtype: int64
```

Distribution of Contract Status

```
df["NAME_CONTRACT_STATUS"].unique()

plt.figure(figsize=(12,6))
df['NAME_CONTRACT_STATUS'].value_counts().plot(kind='bar',color='light
blue',edgecolor='darkblue')
plt.title('Distribution of Contract Status ')
plt.xlabel('Contract status')
plt.ylabel('Count of Loans')
plt.show()
```



- Majority of the clients have "Active" status in NAME_CONTRACT_STATUS.

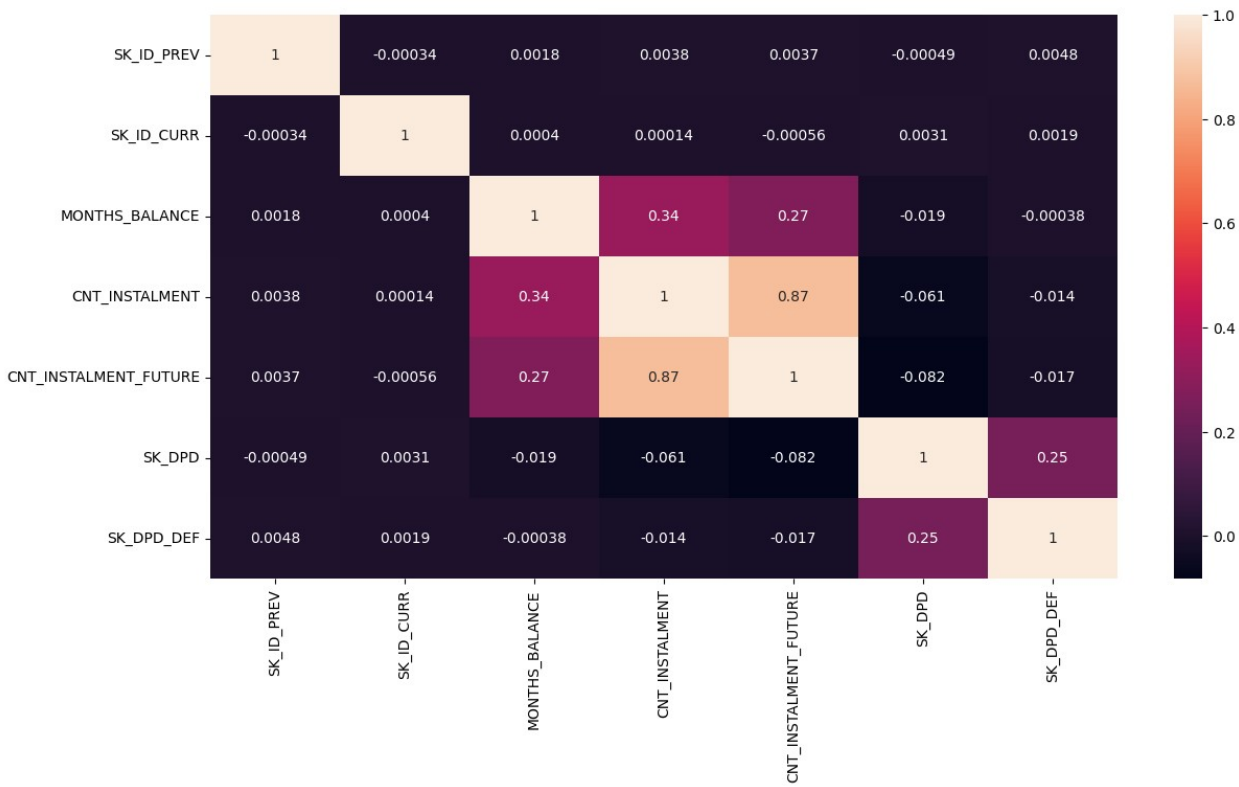
```
a=df.select_dtypes(include=np.number).corr()
a
```

{"summary":"{\n \"name\": \"a\",\n \"rows\": 7,\n \"fields\": [\n {\n \"column\": \"SK_ID_PREV\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.37712856641467885,\n \"min\": -0.00048650127831204997,\n \"max\": 1.0,\n \"num_unique_values\": 7,\n \"samples\": [\n 1.0,\n -0.00033640186135633197,\n -0.00048650127831204997\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"SK_ID_CURR\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.3776695599401592,\n \"min\": -0.000559450889491026,\n \"max\": 1.0,\n \"num_unique_values\": 7,\n \"samples\": [\n -0.00033640186135633197,\n 1.0,\n 0.003118111100230418\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"MONTHS_BALANCE\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.37095978975537525,\n \"min\": -0.018939226282118773,\n \"max\": 1.0,\n \"num_unique_values\": 7,\n \"samples\": [\n 0.0018352433208916623,\n 0.00040396365483740015,\n -0.018939226282118773\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"CNT_INSTALMENT\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.45166026788362235,\n \"min\": -0.06080313776025321,\n \"max\": 1.0,\n \"num_unique_values\": 7,\n \"samples\": [\n 0.003819799424511613,\n 0.00014396700246291138,\n -0.06080313776025321\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"CNT_INSTALMENT_FUTURE\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.45489741679065016,\n \"min\": -0.0820041344934497,\n \"max\": 1.0,\n \"num_unique_values\": 7,\n \"samples\": [\n 0.003678917910923811,\n -0.000559450889491026,\n -0.0820041344934497\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"SK_DPD\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.3878373776534058,\n \"min\": -0.0820041344934497,\n \"max\": 1.0,\n \"num_unique_values\": 7,\n \"samples\": [\n -0.00048650127831204997,\n 0.003118111100230418,\n 1.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"SK_DPD_DEF\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.37596546688745985,\n \"min\": -0.017436043627572374,\n \"max\": 1.0,\n \"num_unique_values\": 7,\n \"samples\": [\n 0.004848362321909511,\n 0.0019478986718724082,\n 0.2457822529945811\n],\n \"semantic_type\": \"\",\n

\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"a"}

```
plt.figure(figsize=(14,7))
sns.heatmap(a,annot=True)
```

```
<Axes: >
```



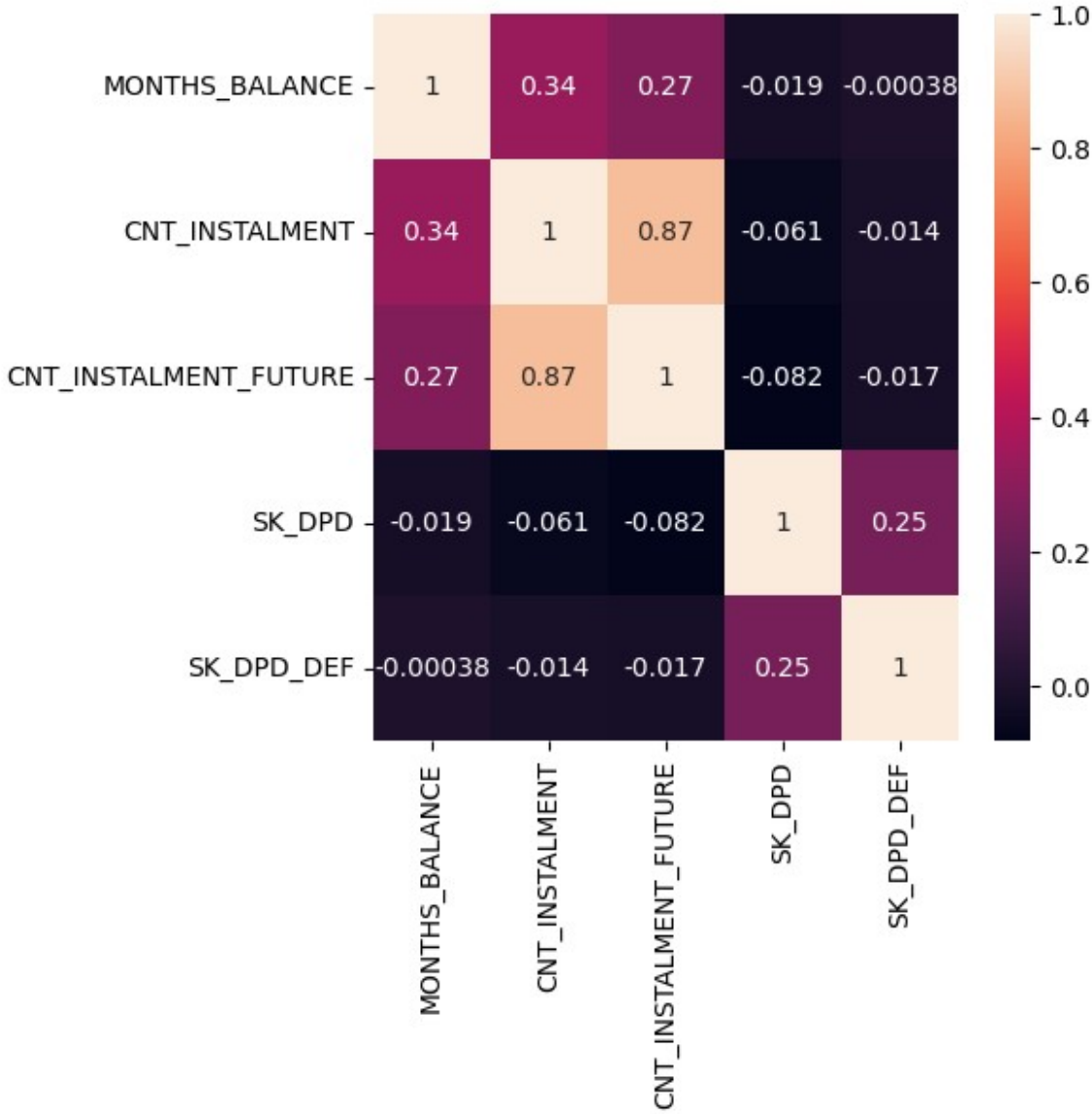

```
df.drop(columns=['SK_ID_PREV', 'SK_ID_CURR'], inplace=True)

b = df.select_dtypes(include=np.number).corr()

b
```

{"summary":"{\n \"name\": \"b\",\n \"rows\": 5,\n \"fields\": [\n
{\n   \"column\": \"MONTHS_BALANCE\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0.4130682128402052,\n
\"min\": -0.018939226282118773,\n   \"max\": 1.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n
0.33616300074310984,\n   -0.00038102329153132244,\n
0.27159451034013016\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"CNT_INSTALMENT\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.4915609397447816,\n   \"min\": -
0.06080313776025321,\n   \"max\": 1.0,\n

\"num_unique_values\": 5,\n \"samples\": [\n 1.0,\n -0.014153584302564555,\n 0.8712759853337174\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"CNT_INSTALMENT_FUTURE\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.5011967233080064,\n \"min\": -0.0820041344934497,\n \"max\": 1.0,\n \"num_unique_values\": 5,\n \"samples\": [\n 0.8712759853337174,\n -0.017436043627572374,\n 1.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"SK_DPD\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.45720963638843376,\n \"min\": -0.0820041344934497,\n \"max\": 1.0,\n \"num_unique_values\": 5,\n \"samples\": [\n -0.06080313776025321,\n 0.2457822529945811,\n -0.0820041344934497\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"SK_DPD_DEF\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.4376773011341028,\n \"min\": -0.017436043627572374,\n \"max\": 1.0,\n \"num_unique_values\": 5,\n \"samples\": [\n -0.014153584302564555,\n 1.0,\n -0.017436043627572374\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n }\n]\n}","type":"dataframe","variable_name":"b"}

```
sns.heatmap(b,annot=True)
```

```
<Axes: >
```




```
df.isnull().sum()
```

```
MONTHS_BALANCE                0
CNT_INSTALMENT            26071
CNT_INSTALMENT_FUTURE     26087
NAME_CONTRACT_STATUS          0
SK_DPD                        0
SK_DPD_DEF                    0
dtype: int64
```

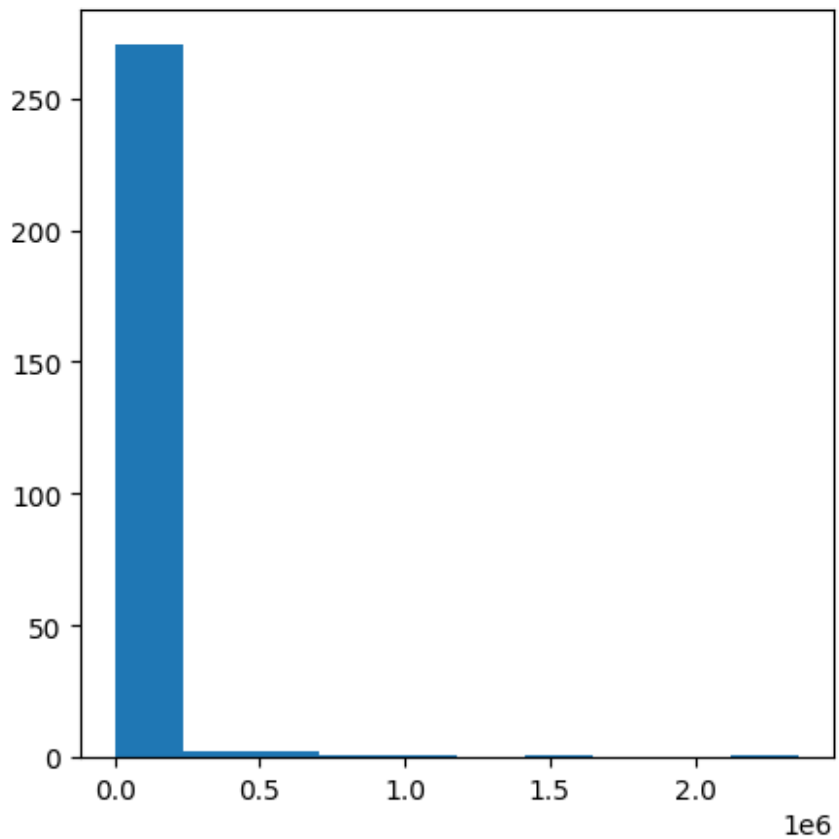
```
df["CNT_INSTALMENT"]=df["CNT_INSTALMENT"].fillna(df["CNT_INSTALMENT"].
median())
```

```
df["CNT_INSTALMENT_FUTURE"]=df["CNT_INSTALMENT_FUTURE"].fillna(df["CNT
_INSTALMENT_FUTURE"].median())

df.columns
```

```
Index(['MONTHS_BALANCE', 'CNT_INSTALMENT', 'CNT_INSTALMENT_FUTURE',
       'NAME_CONTRACT_STATUS', 'SK_DPD', 'SK_DPD_DEF'],
      dtype='object')
```

```
plt.hist(df.groupby('NAME_CONTRACT_STATUS')
["CNT_INSTALMENT"].value_counts())
```

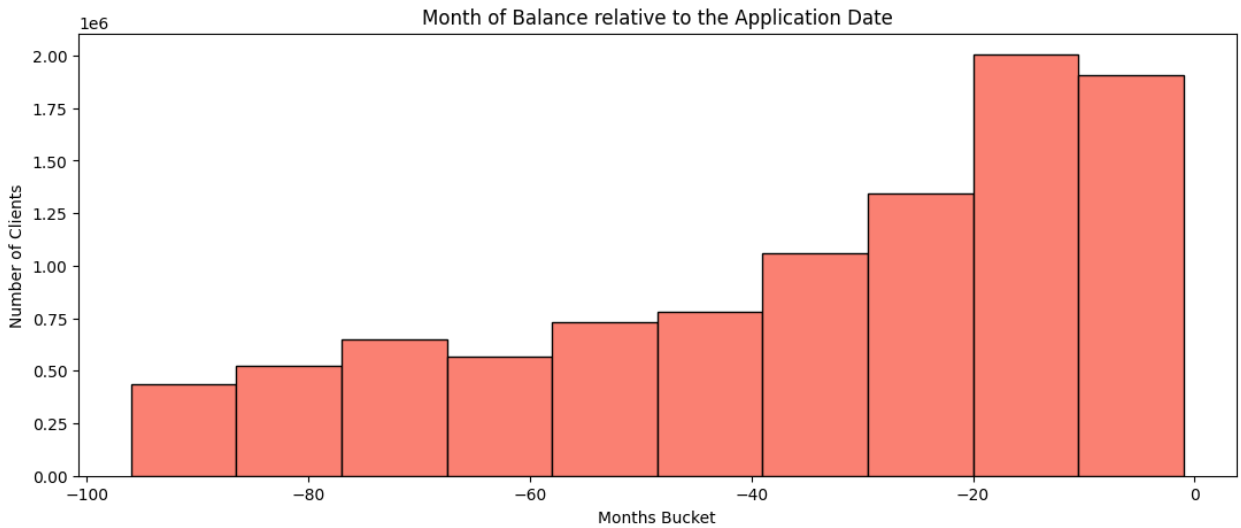
```
(array([270.,   2.,   2.,   1.,   1.,   0.,   1.,   0.,   0.,   1.]),
 array([1.000000e+00, 2.357040e+05, 4.714070e+05, 7.071100e+05,
        9.428130e+05, 1.178516e+06, 1.414219e+06, 1.649922e+06,
        1.885625e+06, 2.121328e+06, 2.357031e+06]),
 <BarContainer object of 10 artists>)
```



MONTHS_BALANCE

```
plt.figure(figsize=(13,5))
plt.hist((df['MONTHS_BALANCE']).values,edgecolor='black',color='salmon
')
```

```
plt.title('Month of Balance relative to the Application Date')
plt.xlabel('Months Bucket')
plt.ylabel('Number of Clients')
plt.show()
```



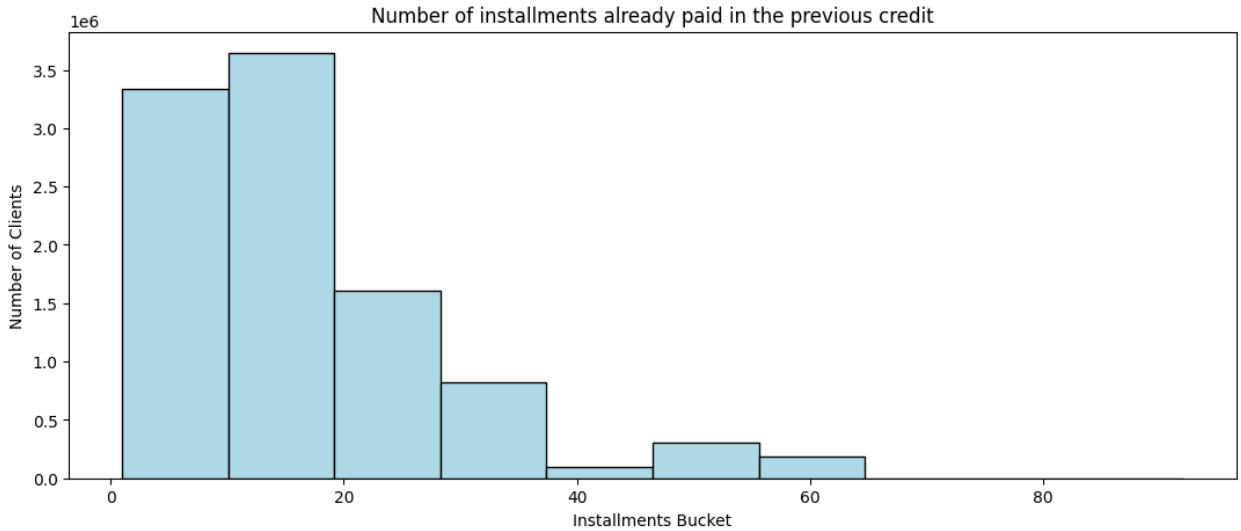
- MONTHS_BALANCE for majority of the clients is between 10 and 20 months before the date of application.

```
df.columns
```

```
Index(['MONTHS_BALANCE', 'CNT_INSTALMENT', 'CNT_INSTALMENT_FUTURE',
       'NAME_CONTRACT_STATUS', 'SK_DPD', 'SK_DPD_DEF'],
      dtype='object')
```

Number of installments already paid.

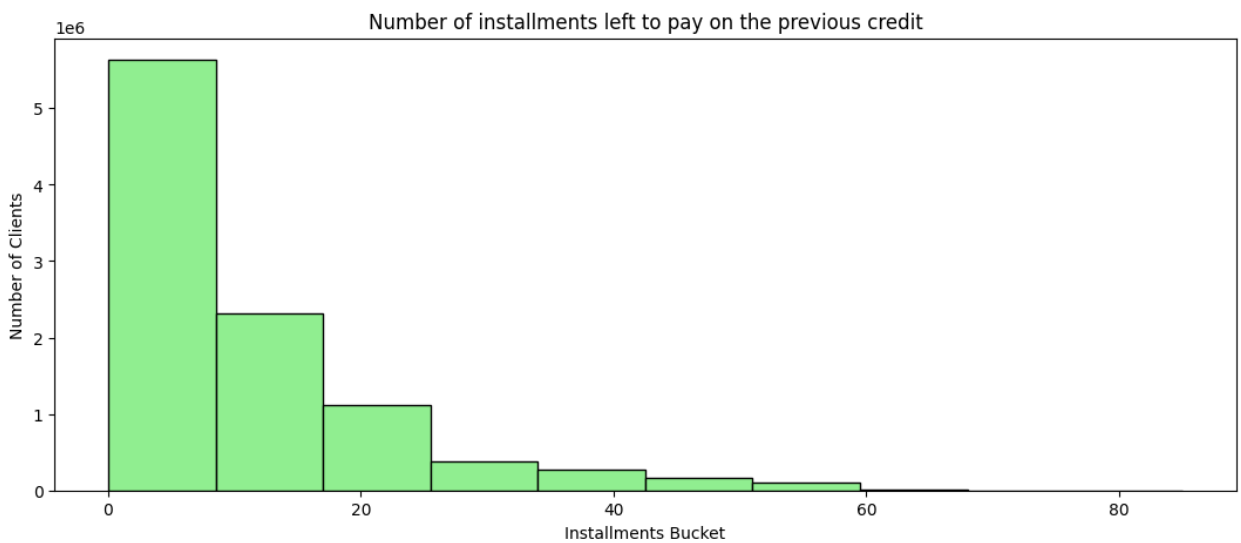
```
plt.figure(figsize=(13,5))
plt.hist((df['CNT_INSTALMENT']).values,edgecolor='black',color='lightb
lue')
plt.title('Number of installments already paid in the previous
credit')
plt.xlabel('Installments Bucket')
plt.ylabel('Number of Clients')
plt.show()
```



- Majority of the clients have number of installments less than equal 20

Number of installments left to pay

```
plt.figure(figsize=(13,5))
plt.hist((df['CNT_INSTALMENT_FUTURE']).values,edgecolor='black',color=
'lightgreen')
plt.title('Number of installments left to pay on the previous credit')
plt.xlabel('Installments Bucket')
plt.ylabel('Number of Clients')
plt.show()
```



- Majority of the the clients have less than 10 installments left to pay off.

Installments Balance

```
df_inst=pd.read_csv("/content/drive/MyDrive/Colab
Notebooks/installments_payments.csv")
df_inst.head()
```

{"type":"dataframe","variable_name":"df_inst"}

```
df_inst.columns
```

```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NUM_INSTALMENT_VERSION',
       'NUM_INSTALMENT_NUMBER', 'DAYS_INSTALMENT',
'DAYS_ENTRY_PAYMENT',
       'AMT_INSTALMENT', 'AMT_PAYMENT'],
      dtype='object')
```

```
df_inst.isnull().sum()
```

```
SK_ID_PREV                   0
SK_ID_CURR                   0
NUM_INSTALMENT_VERSION       0
NUM_INSTALMENT_NUMBER        0
DAYS_INSTALMENT              0
DAYS_ENTRY_PAYMENT        2905
AMT_INSTALMENT               0
AMT_PAYMENT               2905
dtype: int64
```

```
df_inst.isnull().sum()/df_inst.shape[0]*100
```

```
SK_ID_PREV                 0.000000
SK_ID_CURR                 0.000000
NUM_INSTALMENT_VERSION     0.000000
NUM_INSTALMENT_NUMBER      0.000000
DAYS_INSTALMENT            0.000000
DAYS_ENTRY_PAYMENT         0.021352
AMT_INSTALMENT             0.000000
AMT_PAYMENT                0.021352
dtype: float64
```

```
df_inst.shape
```

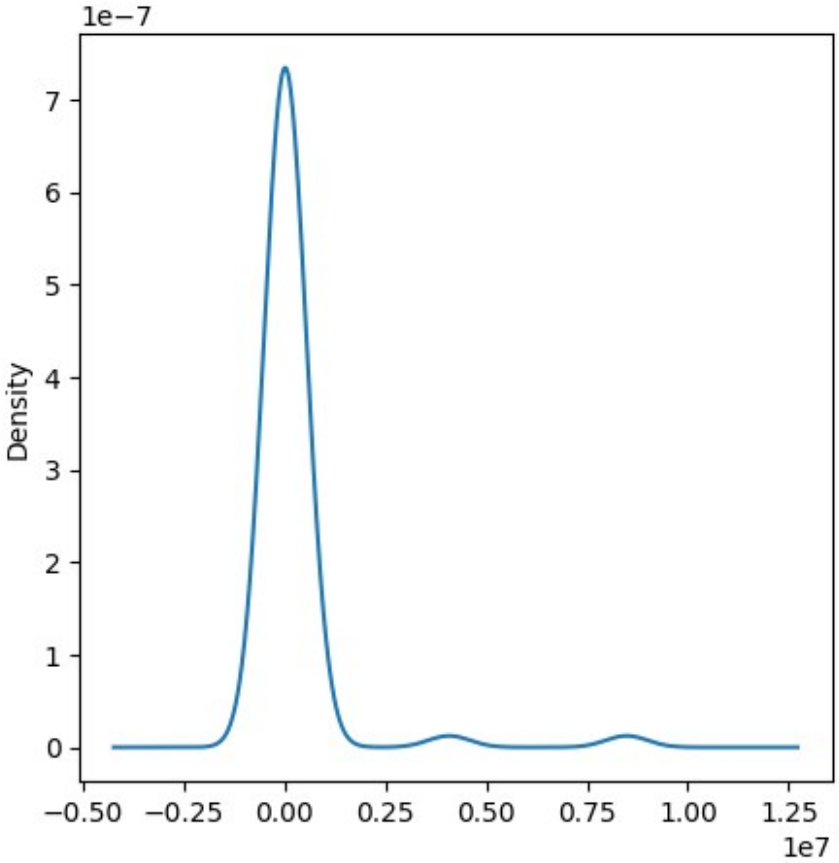
```
(13605401, 8)
```

```
df_inst.dropna(inplace=True)
df_inst.shape
```

```
(13602496, 8)
```

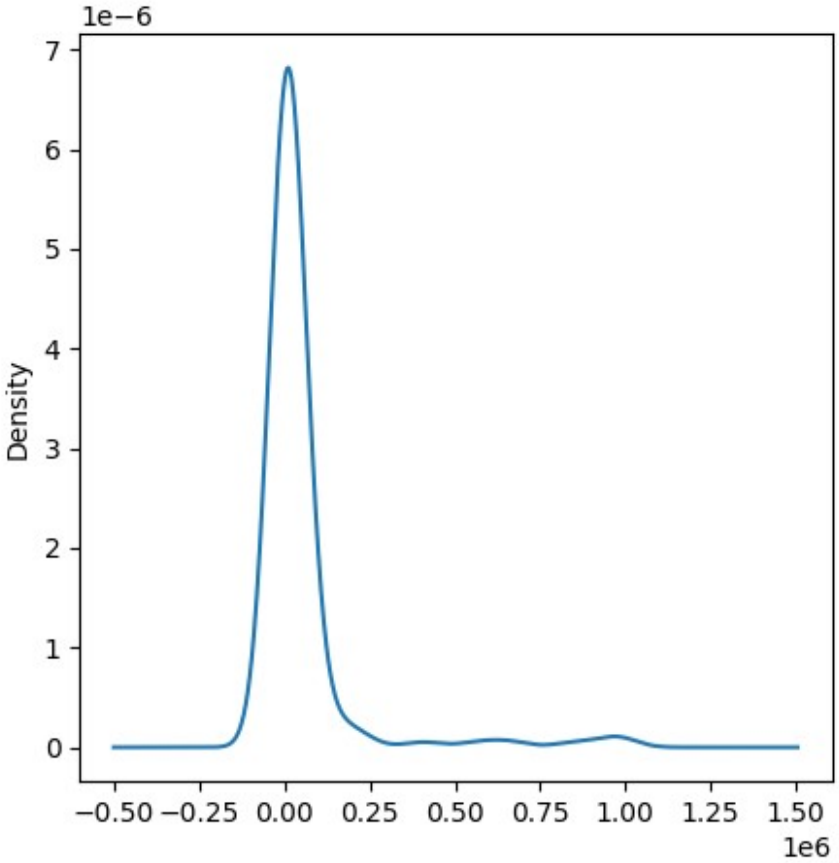
```
df_inst['NUM_INSTALMENT_VERSION'].value_counts().plot(kind='kde')
```

```
<Axes: ylabel='Density'>
```



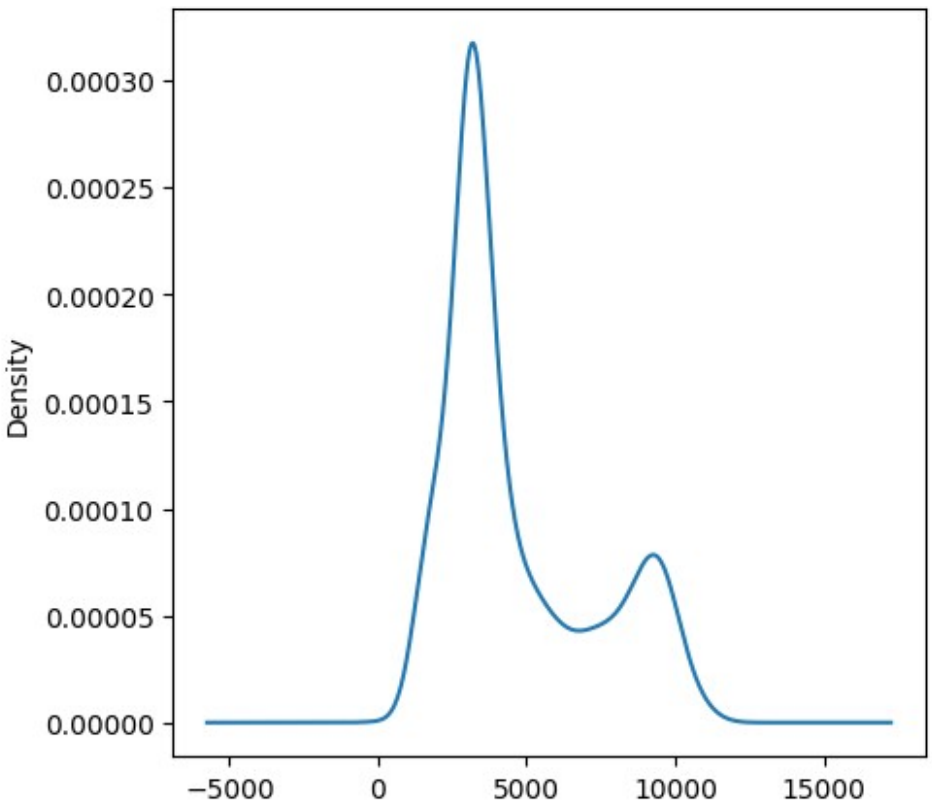

```
df_inst['NUM_INSTALMENT_NUMBER'].value_counts().plot(kind='kde')
```

```
<Axes: ylabel='Density'>
```



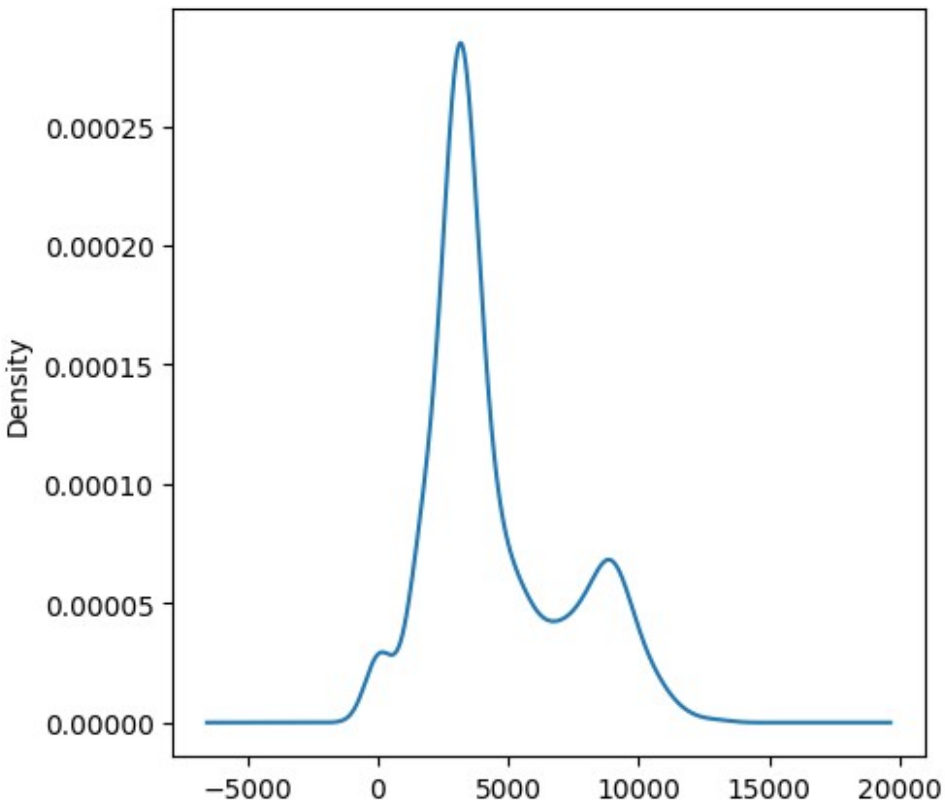

```
df_inst['DAYS_INSTALMENT'].value_counts().plot(kind='kde')
```

```
<Axes: ylabel='Density'>
```




```
df_inst['DAYS_ENTRY_PAYMENT'].value_counts().plot(kind='kde')
```

```
<Axes: ylabel='Density'>
```



CREDIT CARD BALANCE

```
credit_card_balance = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/credit_card_balance.csv')
credit_card_balance.head()
```

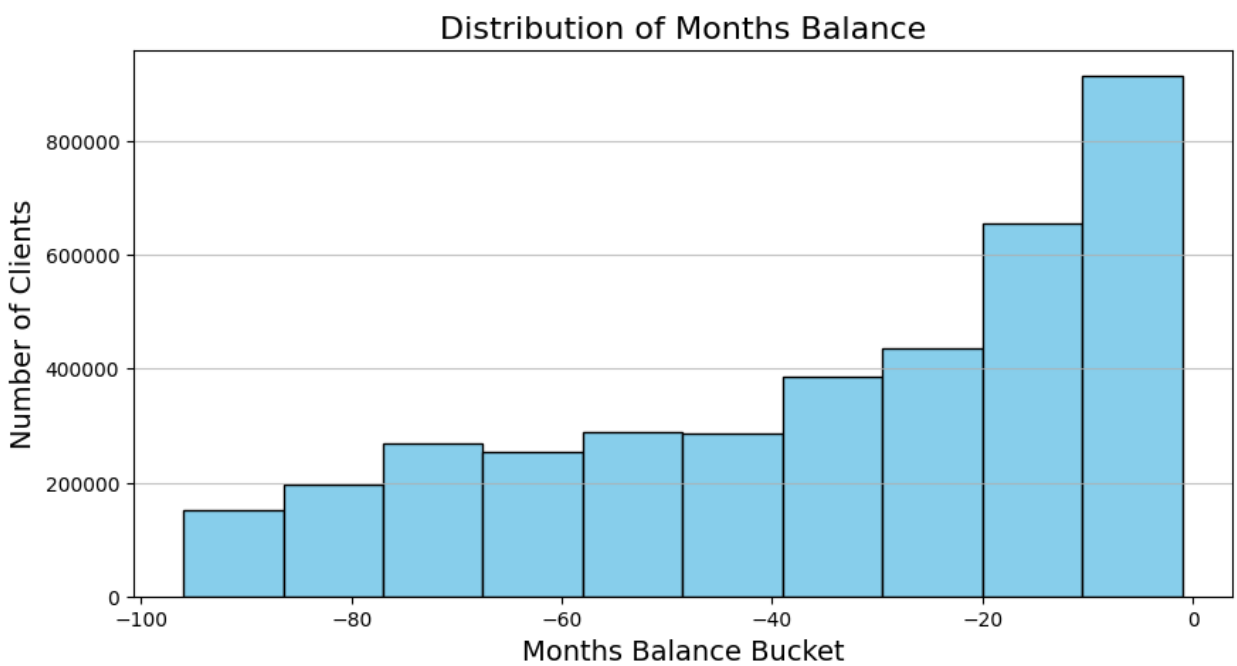
{"type":"dataframe","variable_name":"credit_card_balance"}

```
credit_card_balance.columns
```

```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'MONTHS_BALANCE', 'AMT_BALANCE',
       'AMT_CREDIT_LIMIT_ACTUAL', 'AMT_DRAWINGS_ATM_CURRENT',
       'AMT_DRAWINGS_CURRENT', 'AMT_DRAWINGS_OTHER_CURRENT',
       'AMT_DRAWINGS_POS_CURRENT', 'AMT_INST_MIN_REGULARITY',
       'AMT_PAYMENT_CURRENT', 'AMT_PAYMENT_TOTAL_CURRENT',
       'AMT_RECEIVABLE_PRINCIPAL', 'AMT_RECIVABLE',
'AMT_TOTAL_RECEIVABLE',
       'CNT_DRAWINGS_ATM_CURRENT', 'CNT_DRAWINGS_CURRENT',
       'CNT_DRAWINGS_OTHER_CURRENT', 'CNT_DRAWINGS_POS_CURRENT',
       'CNT_INSTALMENT_MATURE_CUM', 'NAME_CONTRACT_STATUS', 'SK_DPD',
       'SK_DPD_DEF'],
      dtype='object')
```

```
## UNIVARIATE ANALYSIS
```

```python
#### MONTH_BALANCE :

import matplotlib.pyplot as plt

# Set figure size
plt.figure(figsize=(10, 5))

# Create histogram
plt.hist(credit_card_balance['MONTHS_BALANCE'].values, bins=10,
edgecolor='black', color='skyblue')

# Add title and labels
plt.title('Distribution of Months Balance', fontsize=16)
plt.xlabel('Months Balance Bucket', fontsize=14)
plt.ylabel('Number of Clients', fontsize=14)

# Add grid for better readability
plt.grid(axis='y', alpha=0.75)

# Show plot
plt.show()
```



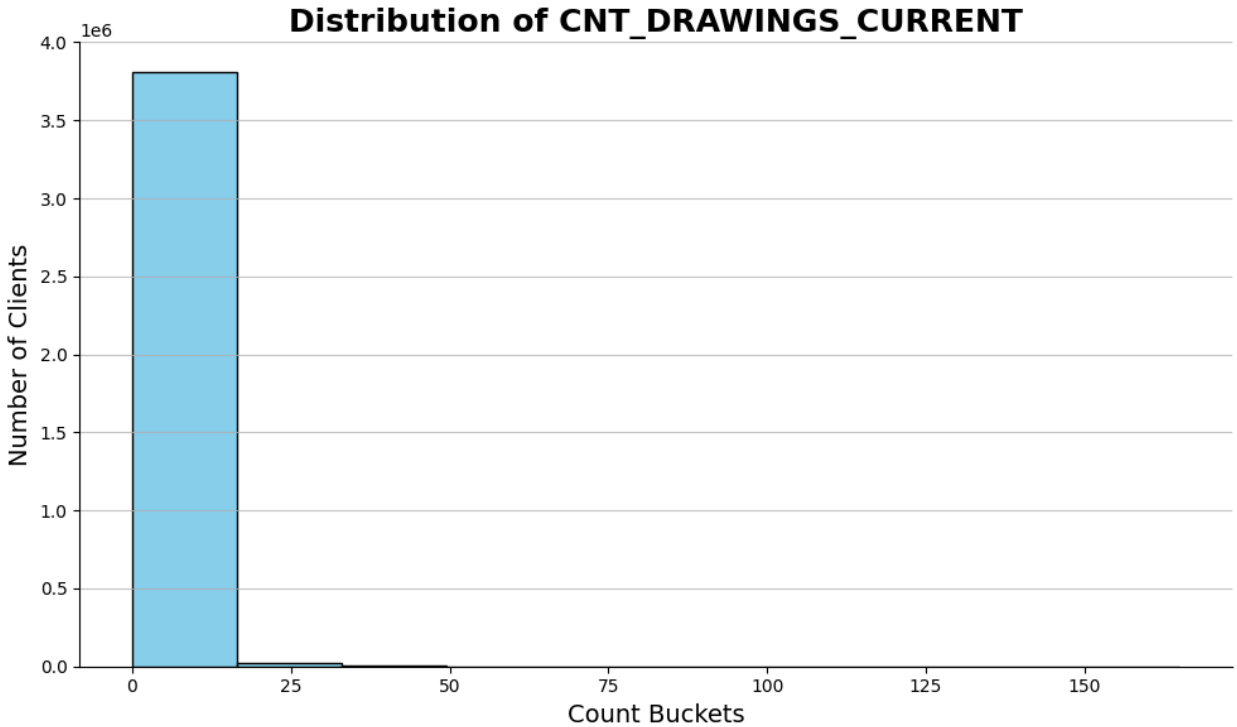
INFERENCES FOR THE ABOVE VISUALISATION:

New Application: Most clients have recent balances between 0 to -10 months from the application date, indicating active or recently closed credit accounts.

Fewer Long-Term Data: Fewer clients have balances older than 20 months, likely due to account closures or inactivity.

Data Recency Focus: The most relevant data for analysis is within the last 20 months, with a high density in the 0 to -10 months range.

```
#### CNT_DRAWINGS_CURRENT:

# Set figure size
plt.figure(figsize=(10, 6))

# Create histogram
plt.hist(credit_card_balance['CNT_DRAWINGS_CURRENT'].values, bins=10,
edgecolor='black', color='skyblue')

# Add title and labels with increased font sizes
plt.title('Distribution of CNT_DRAWINGS_CURRENT', fontsize=18,
fontweight='bold')
plt.xlabel('Count Buckets', fontsize=14)
plt.ylabel('Number of Clients', fontsize=14)

# Add grid lines for better readability
plt.grid(axis='y', alpha=0.75)

# Add a box around the plot for better definition
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Show plot
plt.tight_layout()  # Adjust layout to make room for labels
plt.show()
```



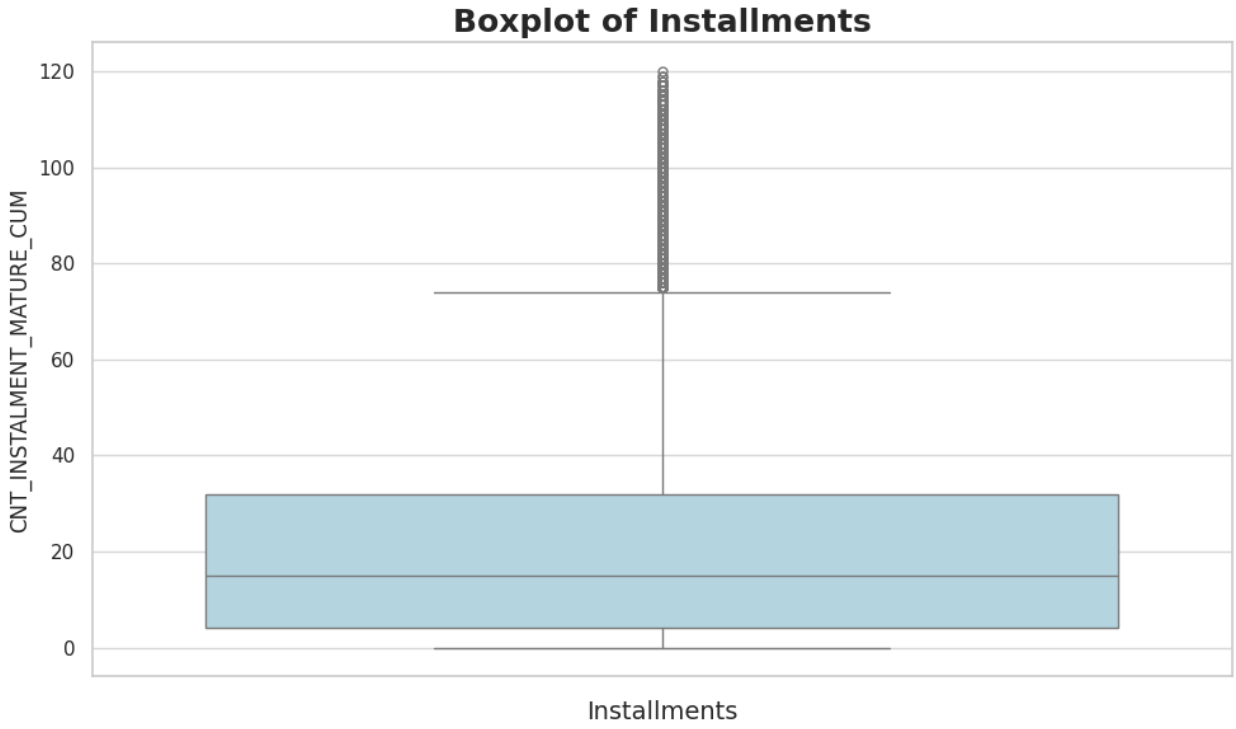
OBSERVATIONS:

Majority Low Usage: Most clients have fewer than 25 withdrawals in the current month on their credit accounts, indicating minimal drawing activity.

Rare Outliers: Only a small number of clients have high withdrawal counts, suggesting these are exceptions rather than the norm.

```
#### CNT_INSTALMENT_MATURE_CUM :


# Set the style for the plot
sns.set(style='whitegrid')

# Create the boxplot
plt.figure(figsize=(10, 6))
sns.boxplot(y='CNT_INSTALMENT_MATURE_CUM', data=credit_card_balance,
color='lightblue', fliersize=5)

# Add title and labels with increased font sizes
plt.title('Boxplot of Installments', fontsize=18, fontweight='bold')
plt.xlabel('Installments', fontsize=14)

# Add grid lines for better readability
plt.grid(axis='y', alpha=0.75)

# Show the plot
plt.tight_layout()  # Adjust layout to make room for labels
```

```
plt.show()
```



OBSERVATIONS

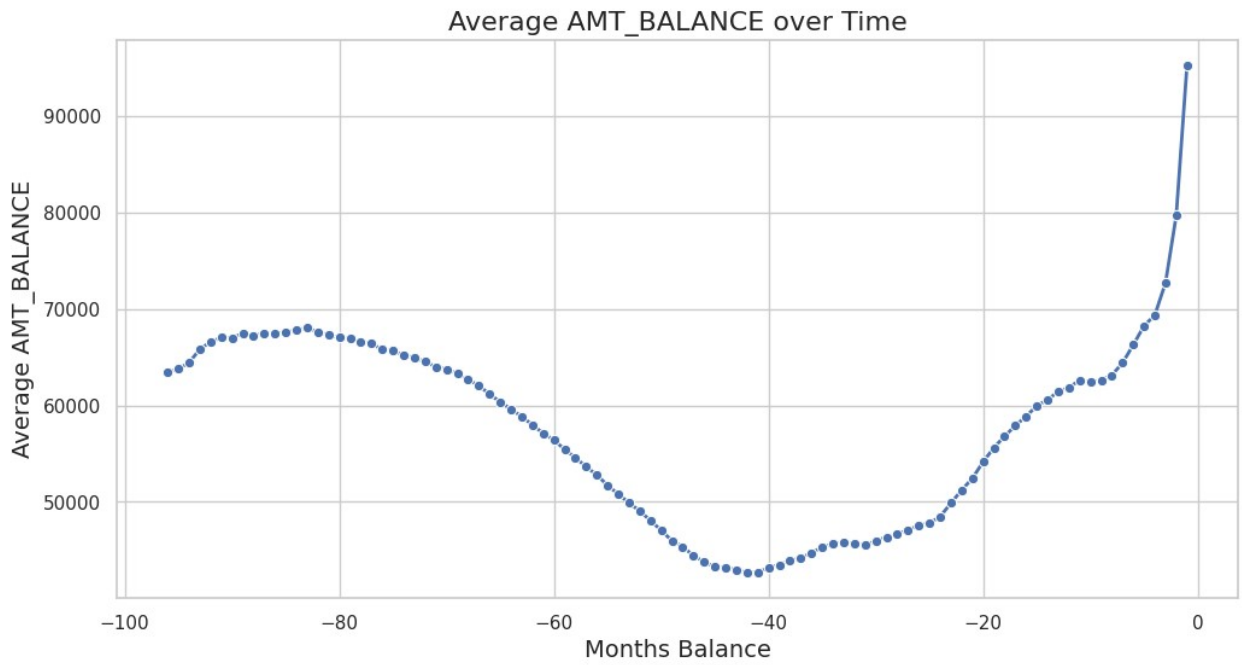
Installment Distribution: Most clients have fewer than 32 paid installments, with 50% having 15 or fewer.

A few clients (outliers) have made up to 120 installments, but this is uncommon.

Key Takeaway: The bulk of clients have relatively few installments, indicating shorter credit histories or less reliance on installment-based repayments.

BIVARIATE ANALYSIS

```
# Line plot showing AMT_BALANCE over time
df_time = credit_card_balance.groupby('MONTHS_BALANCE')
['AMT_BALANCE'].mean().reset_index()

plt.figure(figsize=(12, 6))
sns.lineplot(x='MONTHS_BALANCE', y='AMT_BALANCE', data=df_time,
marker='o', color='b', linestyle='-', linewidth=2)
plt.title('Average AMT_BALANCE over Time', fontsize=16)
plt.xlabel('Months Balance', fontsize=14)
plt.ylabel('Average AMT_BALANCE', fontsize=14)
plt.grid(True)
plt.show()
```



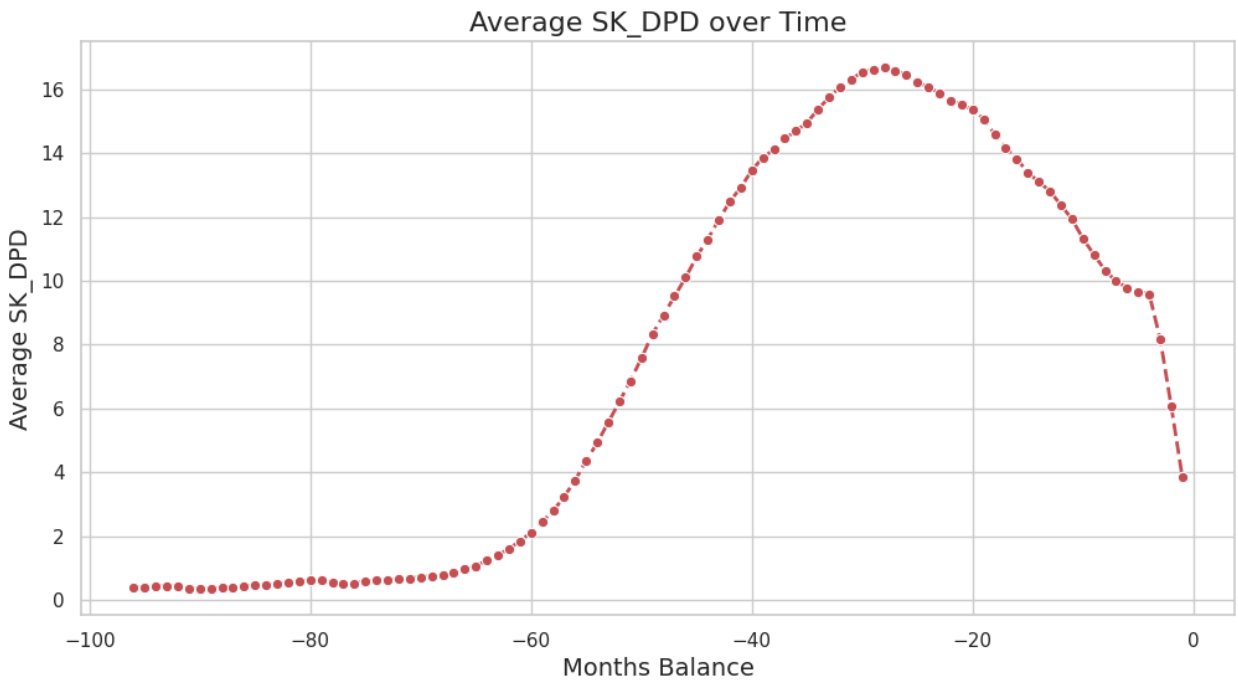
OBSERVATIONS

Initial Decline (-100 to -60 months): Average balance starts above BHD 60,000 and steadily decreases, possibly due to active repayment or reduced spending.

Lowest Point (-40 to -20 months): Balances hit a low of around BHD 45,000, likely indicating debt clearance or minimal credit usage.

Recent Spike (Last 20 months): A sharp rise to over BHD 90,000 suggests increased spending or unpaid balances, reflecting greater credit reliance or financial strain.

```
# Line plot showing SK_DPD over time
df_dpd_time = credit_card_balance.groupby('MONTHS_BALANCE')
['SK_DPD'].mean().reset_index()

plt.figure(figsize=(12, 6))
sns.lineplot(x='MONTHS_BALANCE', y='SK_DPD', data=df_dpd_time,
marker='o', color='r', linestyle='--', linewidth=2)
plt.title('Average SK_DPD over Time', fontsize=16)
plt.xlabel('Months Balance', fontsize=14)
plt.ylabel('Average SK_DPD', fontsize=14)
plt.grid(True)
plt.show()
```



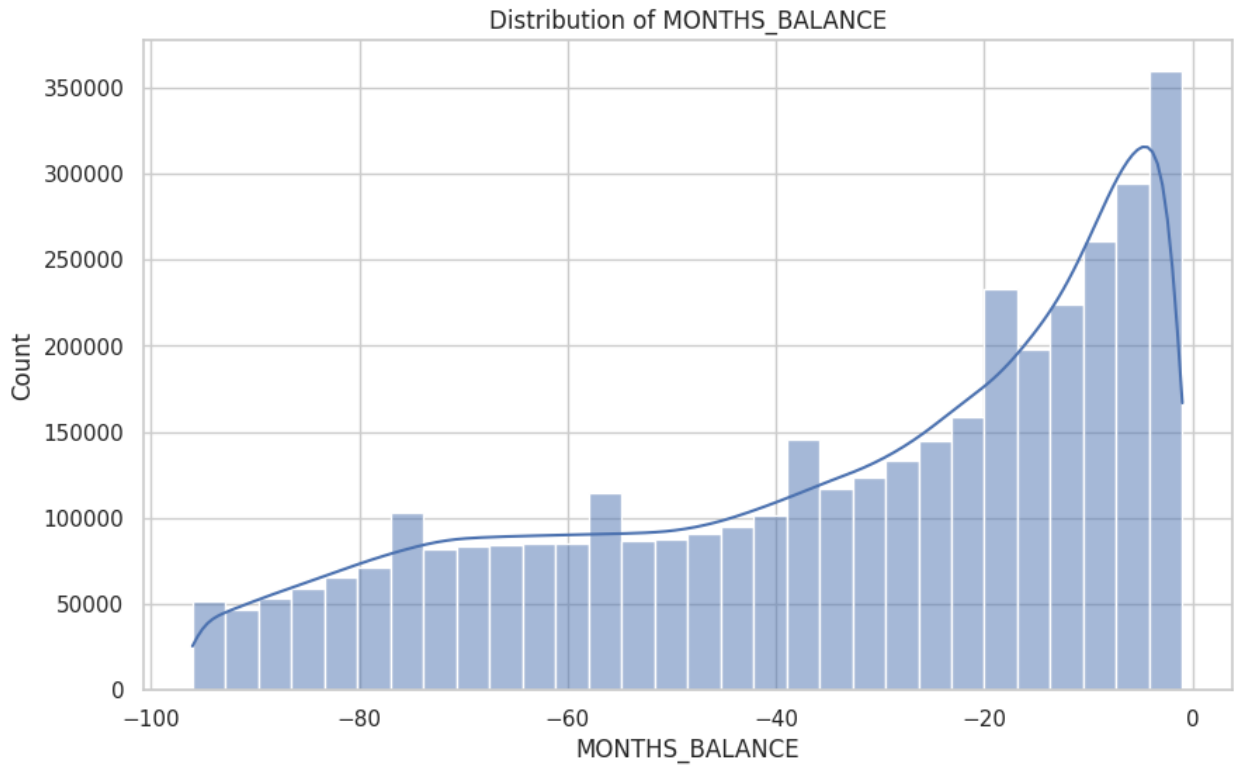
OBSERVATIONS

Correlation: As AMT_BALANCE rises, SK_DPD (delinquencies) may increase, indicating repayment difficulties.

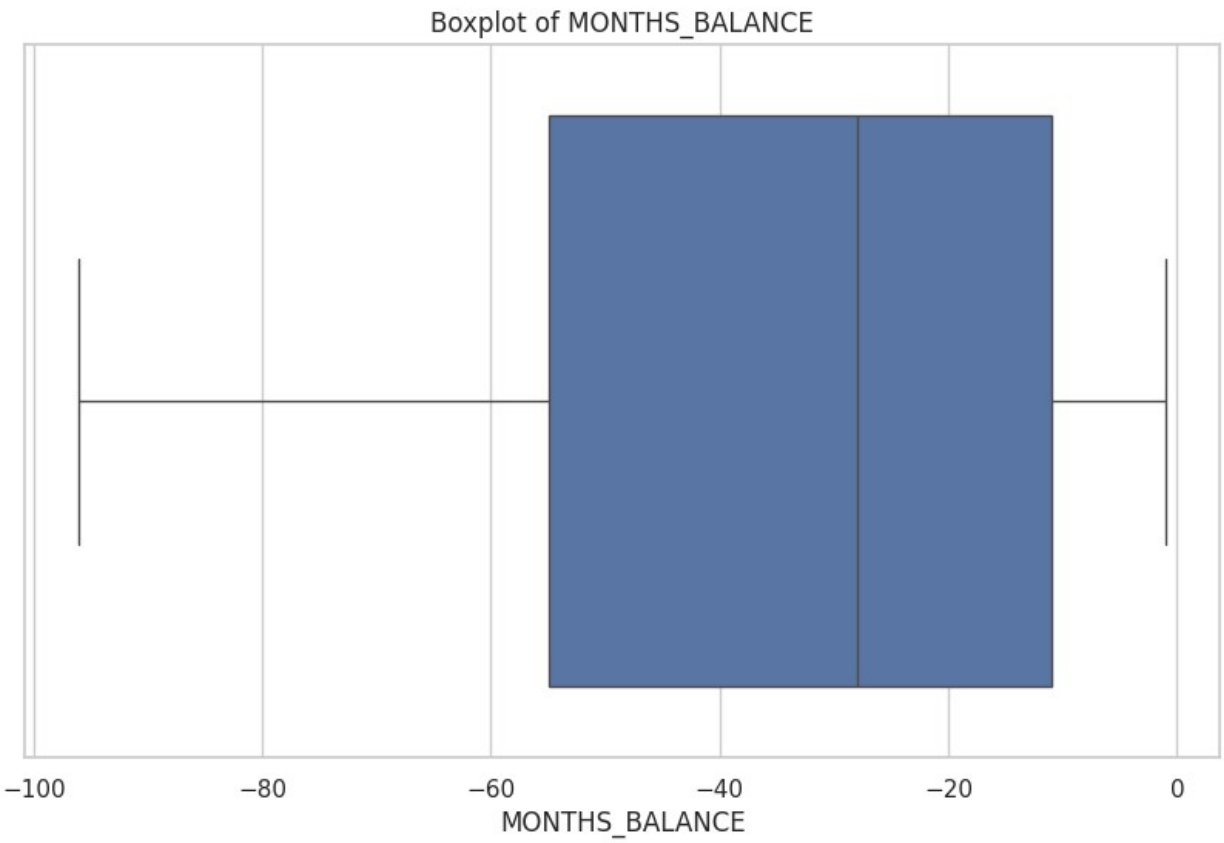
Minimal Delinquency (-40 to -20 months): During low balance periods, fewer overdue payments are expected.

Increase in Delinquencies (Recent months): As balances rise near the present, overdue payments likely increase, signaling financial stress.

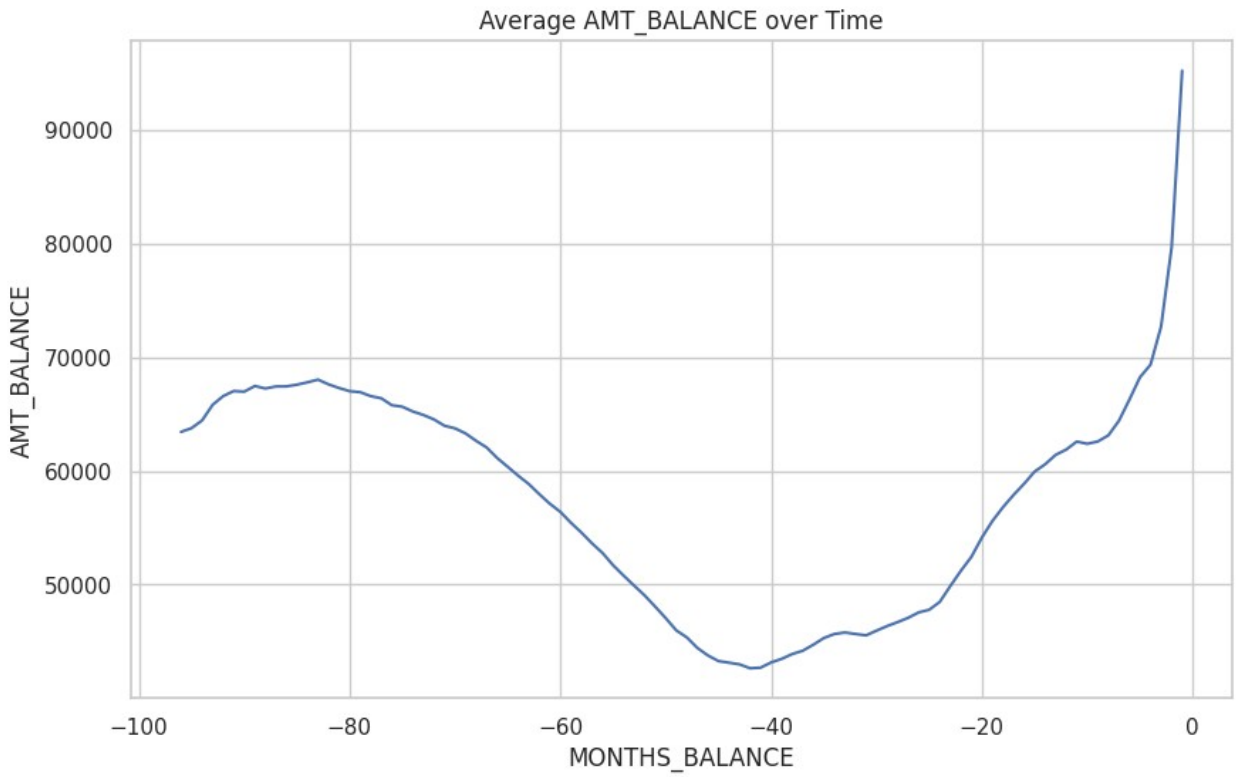
```
# Histogram for MONTHS_BALANCE
plt.figure(figsize=(10, 6))
sns.histplot(credit_card_balance['MONTHS_BALANCE'], bins=30, kde=True)
plt.title('Distribution of MONTHS_BALANCE')
plt.show()
```




```
# Boxplot for MONTHS_BALANCE
plt.figure(figsize=(10, 6))
sns.boxplot(x=credit_card_balance['MONTHS_BALANCE'])
plt.title('Boxplot of MONTHS_BALANCE')
plt.show()
```




```
# Line plot for MONTHS_BALANCE
df_time = credit_card_balance.groupby('MONTHS_BALANCE')
['AMT_BALANCE'].mean().reset_index()
plt.figure(figsize=(10, 6))
sns.lineplot(x='MONTHS_BALANCE', y='AMT_BALANCE', data=df_time)
plt.title('Average AMT_BALANCE over Time')
plt.show()
```



Machine Learning Model Building

```
df1.head()
```

{"type":"dataframe","variable_name":"df1"}

```
cat_cols=df1.select_dtypes(exclude=np.number).columns
cat_cols
```

```
Index(['TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR',
       'FLAG_OWN_REALTY', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE',
       'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
'NAME_HOUSING_TYPE',
       'OCCUPATION_TYPE', 'WEEKDAY_APPR_PROCESS_START',
'ORGANIZATION_TYPE',
       'INCOME_GROUP'],
      dtype='object')
```

```
for i in cat_cols:
  print(df1[i].value_counts())
  print()
```

```
TARGET
0    282686
1     24825
```

```
Name: count, dtype: int64

NAME_CONTRACT_TYPE
Cash loans         278232
Revolving loans     29279
Name: count, dtype: int64

CODE_GENDER
F    202452
M    105059
Name: count, dtype: int64

FLAG_OWN_CAR
N    202924
Y    104587
Name: count, dtype: int64

FLAG_OWN_REALTY
Y    213312
N     94199
Name: count, dtype: int64

NAME_TYPE_SUITE
Unaccompanied      249818
Family              40149
Spouse, partner     11370
Children             3267
Other_B              1770
Other_A               866
Group of people       271
Name: count, dtype: int64

NAME_INCOME_TYPE
Working                 158774
Commercial associate     71617
Pensioner                55362
State servant            21703
Unemployed                  22
Student                     18
Businessman                 10
Maternity leave              5
Name: count, dtype: int64

NAME_EDUCATION_TYPE
Secondary / secondary special    218391
Higher education                  74863
Incomplete higher                 10277
Lower secondary                    3816
Academic degree                     164
Name: count, dtype: int64
```

```
NAME_FAMILY_STATUS
Married                 196432
Single / not married     45444
Civil marriage           29775
Separated                19770
Widow                    16088
Unknown                      2
Name: count, dtype: int64

NAME_HOUSING_TYPE
House / apartment      272868
With parents            14840
Municipal apartment     11183
Rented apartment         4881
Office apartment         2617
Co-op apartment          1122
Name: count, dtype: int64

OCCUPATION_TYPE
Unknown                  96391
Laborers                 55186
Sales staff              32102
Core staff               27570
Managers                 21371
Drivers                  18603
High skill tech staff    11380
Accountants               9813
Medicine staff            8537
Security staff            6721
Cooking staff             5946
Cleaning staff            4653
Private service staff     2652
Low-skill Laborers        2093
Waiters/barmen staff      1348
Secretaries               1305
Realty agents              751
HR staff                   563
IT staff                   526
Name: count, dtype: int64

WEEKDAY_APPR_PROCESS_START
TUESDAY      53901
WEDNESDAY    51934
MONDAY       50714
THURSDAY     50591
FRIDAY       50338
SATURDAY     33852
SUNDAY       16181
Name: count, dtype: int64
```

```
ORGANIZATION_TYPE
Business Entity Type 3    67992
XNA                       55374
Self-employed             38412
Other                     16683
Medicine                  11193
Business Entity Type 2    10553
Government                10404
School                     8893
Trade: type 7              7831
Kindergarten               6880
Construction               6721
Business Entity Type 1     5984
Transport: type 4          5398
Trade: type 3              3492
Industry: type 9           3368
Industry: type 3           3278
Security                   3247
Housing                    2958
Industry: type 11          2704
Military                   2634
Bank                       2507
Agriculture                2454
Police                     2341
Transport: type 2          2204
Postal                     2157
Security Ministries        1974
Trade: type 2              1900
Restaurant                 1811
Services                   1575
University                 1327
Industry: type 7           1307
Transport: type 3          1187
Industry: type 1           1039
Hotel                       966
Electricity                 950
Industry: type 4            877
Trade: type 6               631
Industry: type 5            599
Insurance                   597
Telecom                     577
Emergency                   560
Industry: type 2            458
Advertising                 429
Realtor                     396
Culture                     379
Industry: type 12           369
Trade: type 1               348
```

```
Mobile                        317
Legal Services                305
Cleaning                      260
Transport: type 1             201
Industry: type 6              112
Industry: type 10             109
Religion                       85
Industry: type 13              67
Trade: type 4                  64
Trade: type 5                  49
Industry: type 8               24
Name: count, dtype: int64

INCOME_GROUP
1-2L         155898
2-5L          85213
Upto 1L       63698
5-10L          2452
10-20L          206
20-50L           39
50L above         4
Name: count, dtype: int64
```

```
df1["ORGANIZATION_TYPE"].nunique()
```

```
58
```

```
df1.to_csv('/content/drive/MyDrive/Colab
Notebooks/new_file.csv',index=False)
```

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# 1. Label Encoding for Ordinal Variables (if applicable)
# Assuming INCOME_GROUP is ordinal
income_order = ['Upto 1L', '1-2L', '2-5L', '5-10L', '10-20L', '20-
50L', '50L above']
df1['INCOME_GROUP'] = pd.Categorical(df1['INCOME_GROUP'],
categories=income_order, ordered=True).codes

# 2. One-Hot Encoding for Nominal Variables
# List of nominal variables to one-hot encode
nominal_vars = [
    'NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR',
    'FLAG_OWN_REALTY', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE',
    'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
    'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE',
    'WEEKDAY_APPR_PROCESS_START', 'ORGANIZATION_TYPE'
]
```

```
# Apply one-hot encoding
df1 = pd.get_dummies(df1, columns=nominal_vars, drop_first=True)

# Display the encoded DataFrame
df1.head()
```

{"type":"dataframe","variable_name":"df1"}

```
df1.shape
```

```
(307511, 174)
```

```
bool_cols=df1.select_dtypes('bool').columns

df1[bool_cols]=df1[bool_cols].astype('int')

df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 174 entries, SK_ID_CURR to ORGANIZATION_TYPE_XNA
dtypes: float64(10), int64(162), int8(1), object(1)
memory usage: 406.2+ MB
```

```
df1.select_dtypes('object').columns
```

```
Index(['TARGET'], dtype='object')
```

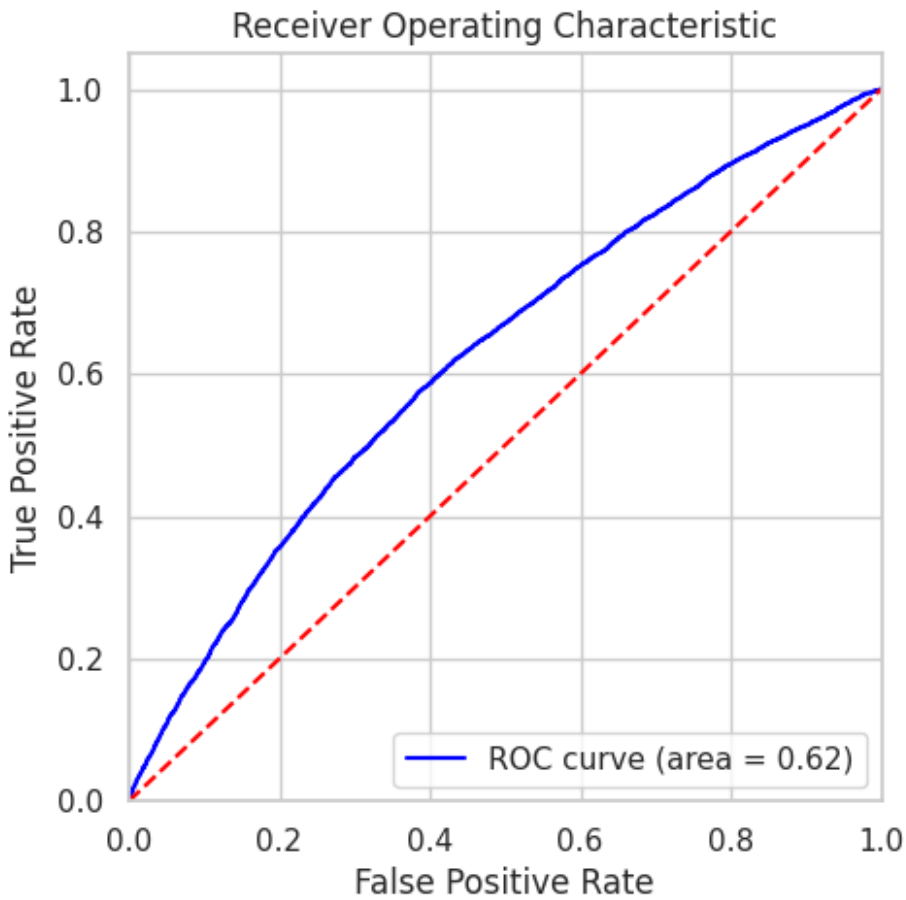
```
x=df1.drop(columns=['SK_ID_CURR','TARGET'])
y=df1['TARGET'].astype('int')

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, roc_curve

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_st
ate=42)

model = LogisticRegression()
model.fit(xtrain, ytrain)
```

```
LogisticRegression()
```

```
y_probs = model.predict_proba(xtest)[:, 1]

roc_auc = roc_auc_score(ytest, y_probs)
print(f"ROC-AUC: {roc_auc:.4f}")

# Optionally, plot the ROC curve
fpr, tpr, thresholds = roc_curve(ytest, y_probs)

plt.figure()
plt.plot(fpr, tpr, color='blue', label='ROC curve (area = %0.2f)' %
```

```
roc_auc)
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.show()
```

```
ROC-AUC: 0.6231
```

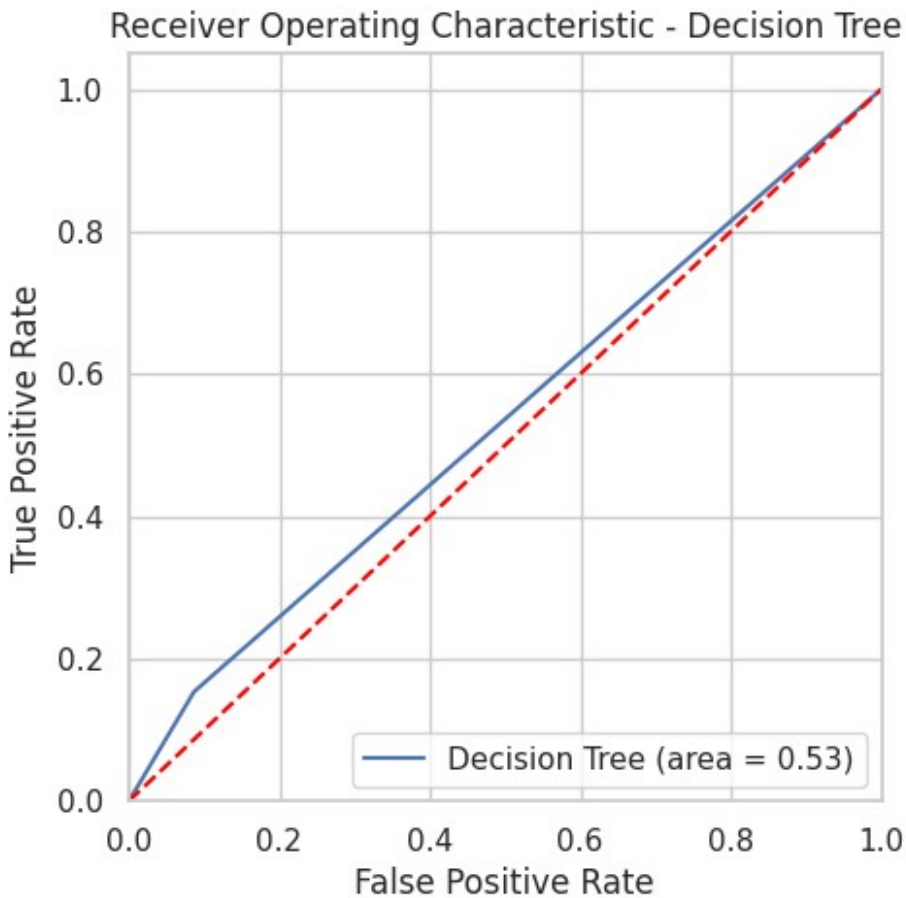



```
import tensorflow as tf

# Check if GPU is available
gpu_available = tf.config.list_physical_devices('GPU')
print("GPU Available:", gpu_available)
```

```
GPU Available: []
```

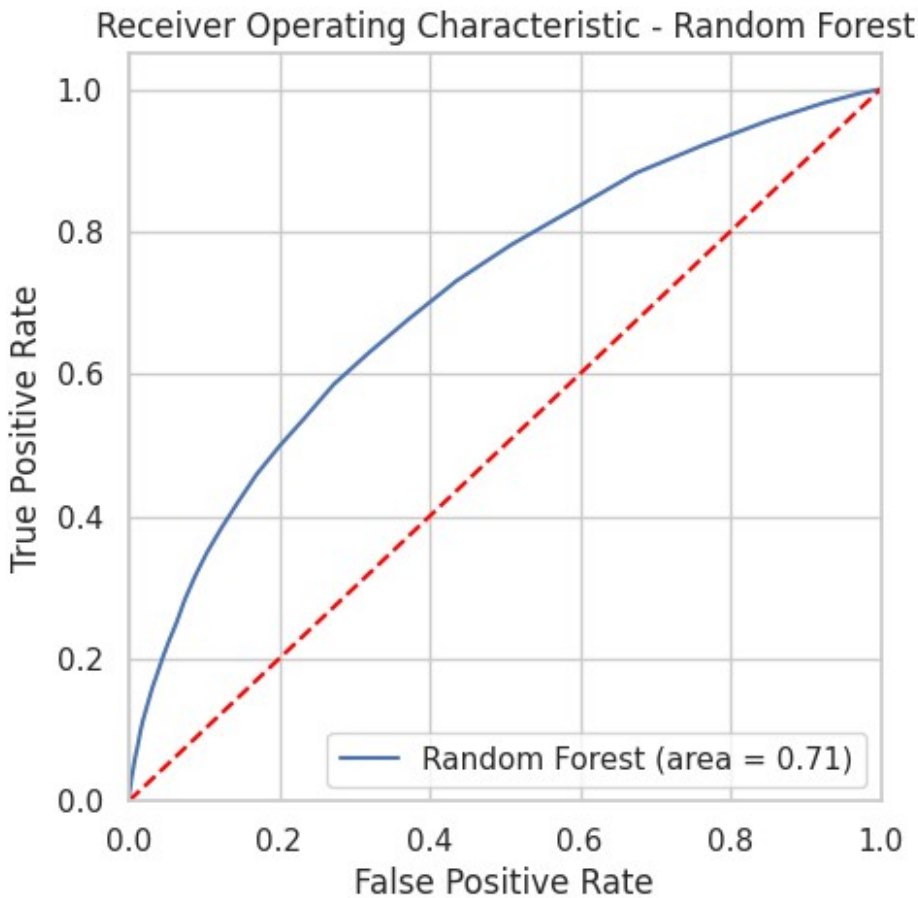
```
def plot_roc_auc(y_test, y_probs, model_name):
    """Function to plot ROC-AUC curve for a given model."""
```

```python
    roc_auc = roc_auc_score(y_test, y_probs)
    fpr, tpr, _ = roc_curve(y_test, y_probs)

    plt.plot(fpr, tpr, label=f'{model_name} (area = {roc_auc:.2f})')

    return roc_auc

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier

dt_model = DecisionTreeClassifier()
dt_model.fit(xtrain, ytrain)

# Predict probabilities
dt_y_probs = dt_model.predict_proba(xtest)[:, 1]

# Plot ROC-AUC curve
dt_roc_auc = plot_roc_auc(ytest, dt_y_probs, 'Decision Tree')

# Plot diagonal line for random classifier
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Decision Tree')
plt.legend(loc='lower right')
plt.show()

print(f"Decision Tree ROC-AUC: {dt_roc_auc:.4f}")
```



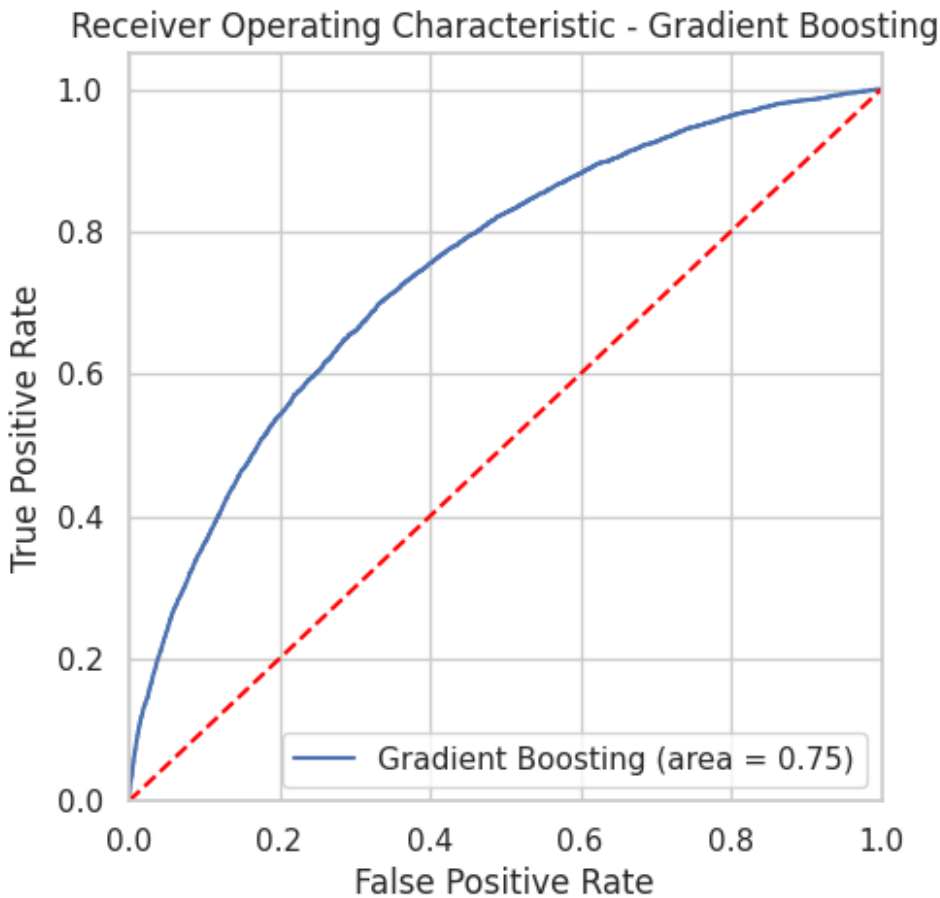

```
Decision Tree ROC-AUC: 0.5335
```

```
rf_model = RandomForestClassifier()
rf_model.fit(xtrain, ytrain)

# Predict probabilities
rf_y_probs = rf_model.predict_proba(xtest)[:, 1]

# Plot ROC-AUC curve
rf_roc_auc = plot_roc_auc(ytest, rf_y_probs, 'Random Forest')

# Plot diagonal line for random classifier
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Random Forest')
plt.legend(loc='lower right')
plt.show()

print(f"Random Forest ROC-AUC: {rf_roc_auc:.4f}")
```




```
Random Forest ROC-AUC: 0.7114
```

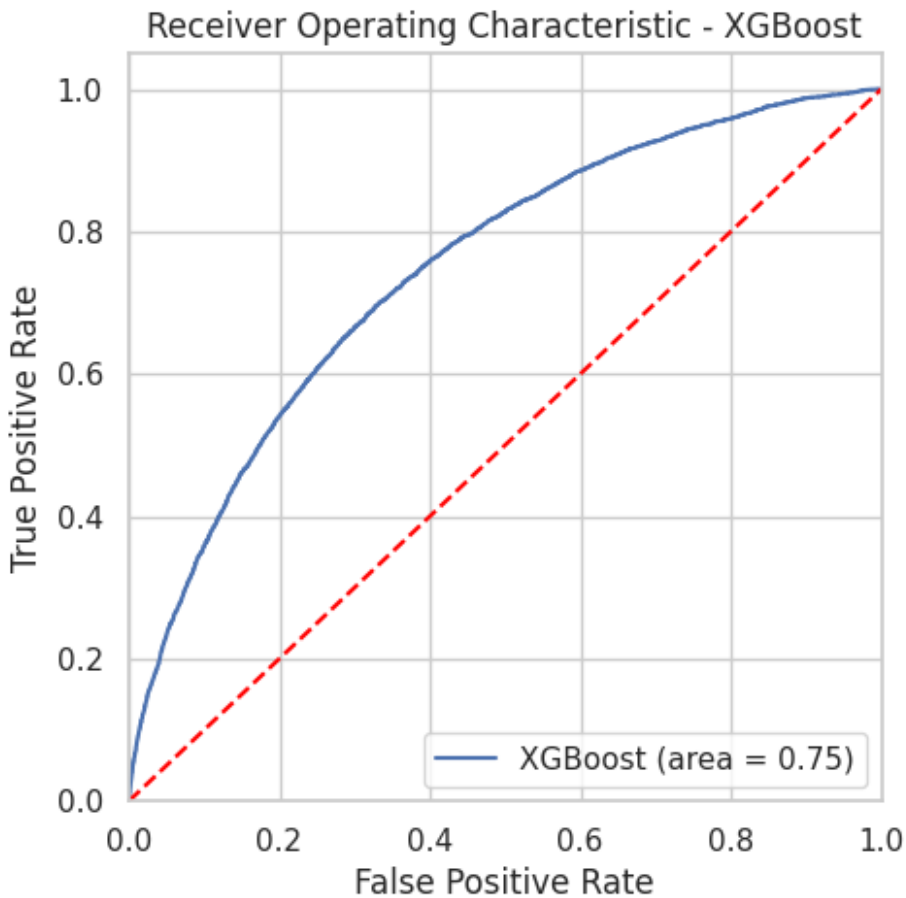
```
gb_model = GradientBoostingClassifier()
gb_model.fit(xtrain, ytrain)

# Predict probabilities
gb_y_probs = gb_model.predict_proba(xtest)[:, 1]

# Plot ROC-AUC curve
gb_roc_auc = plot_roc_auc(ytest, gb_y_probs, 'Gradient Boosting')

# Plot diagonal line for random classifier
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Gradient Boosting')
plt.legend(loc='lower right')
plt.show()

print(f"Gradient Boosting ROC-AUC: {gb_roc_auc:.4f}")
```




```
Gradient Boosting ROC-AUC: 0.7461
```

```
import xgboost as xgb

xgb_model = xgb.XGBClassifier(use_label_encoder=False,
eval_metric='auc')
xgb_model.fit(xtrain, ytrain)

# Predict probabilities
xgb_y_probs = xgb_model.predict_proba(xtest)[:, 1]

# Plot ROC-AUC curve
xgb_roc_auc = plot_roc_auc(ytest, xgb_y_probs, 'XGBoost')

# Plot diagonal line for random classifier
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - XGBoost')
plt.legend(loc='lower right')
```

```
plt.show()

print(f"XGBoost ROC-AUC: {xgb_roc_auc:.4f}")
```

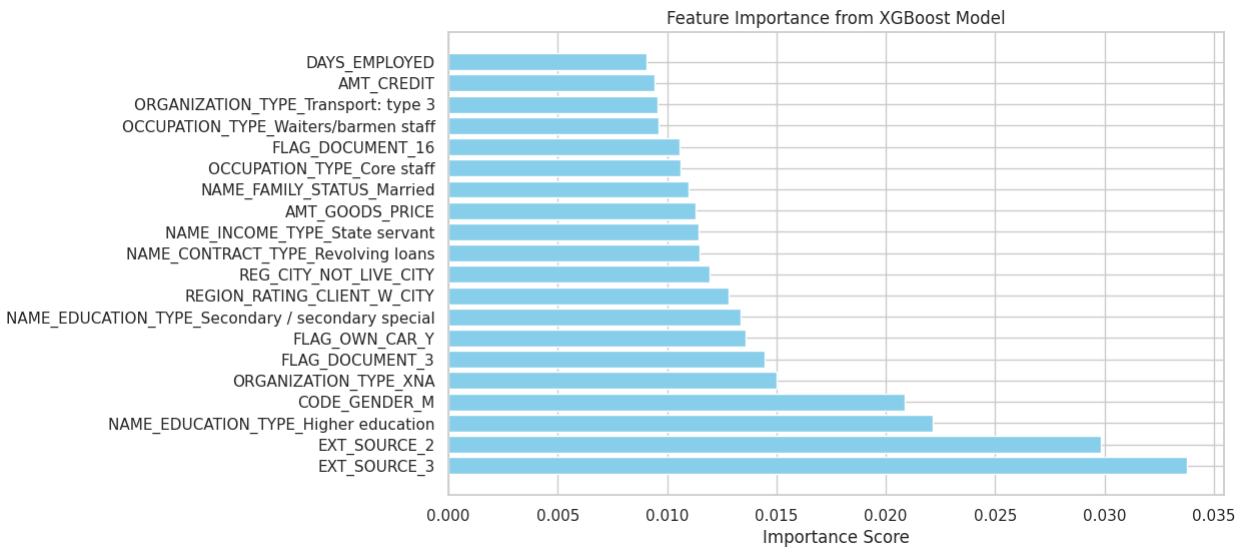



```
XGBoost ROC-AUC: 0.7464

# Get feature importance
importance = xgb_model.feature_importances_


feature_names = x.columns
importance_df = pd.DataFrame({'Feature': feature_names, 'Importance':
importance})
importance_df = importance_df.sort_values(by='Importance',
ascending=False)

importance_df.head(10)
```

{"summary":"{\n \"name\": \"importance_df\",\n \"rows\": 172,\n
\"fields\": [\n {\n \"column\": \"Feature\",\n
\"properties\": {\n \"dtype\": \"string\",\n
\"num_unique_values\": 172,\n \"samples\": [\n

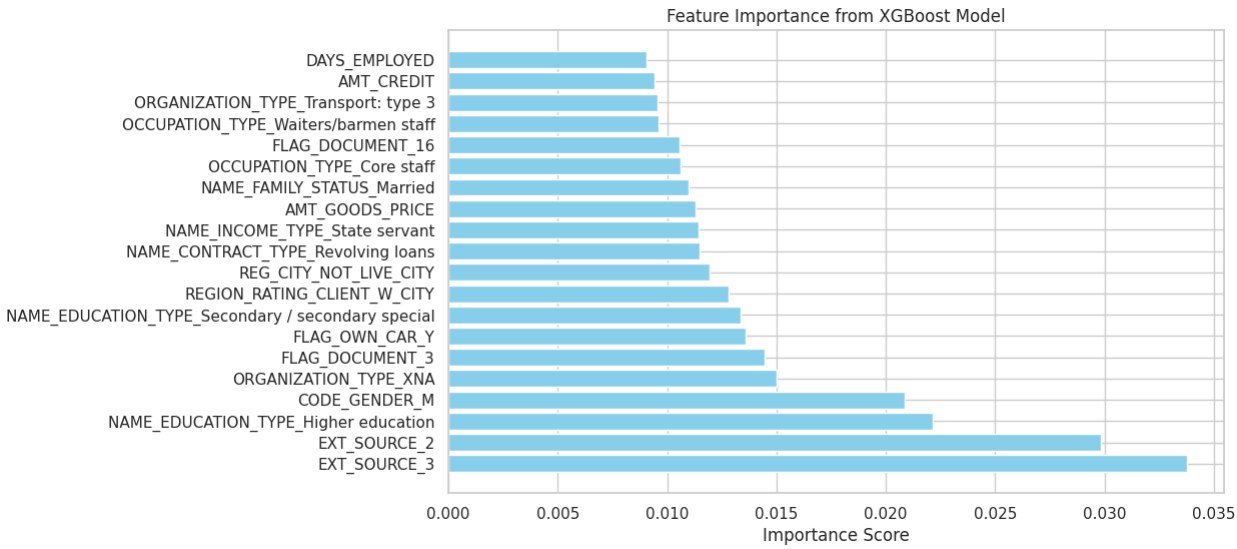
\"FLAG_OWN_REALTY_Y\",\n \"ORGANIZATION_TYPE_Hotel\",\n \"ORGANIZATION_TYPE_Services\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Importance\",\n \"properties\": {\n \"dtype\": \"float32\",\n \"num_unique_values\": 150,\n \"samples\": [\n 0.005802934523671865 5,\n 0.009425627067685127,\n 0.004412588197737932\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n }\n]\n}","type":"dataframe","variable_name":"importance_df"}

```
# Plot feature importance
plt.figure(figsize=(10, 6))
plt.barh(importance_df.head(20)['Feature'], importance_df.head(20)
['Importance'], color='skyblue')
plt.xlabel('Importance Score')
plt.title('Feature Importance from XGBoost Model')
plt.show()
```




```
# RF feature importance
importance = rf_model.feature_importances_

# Create a DataFrame for better visualization
importance_rf = pd.DataFrame({'Feature': x.columns, 'Importance':
importance})
importance_rf = importance_df.sort_values(by='Importance',
ascending=False)

# Plot feature importance
plt.figure(figsize=(10, 6))
plt.barh(importance_rf.head(20)['Feature'], importance_rf.head(20)
['Importance'], color='skyblue')
plt.xlabel('Importance Score')
```

```
plt.title('Feature Importance from XGBoost Model')
plt.show()
```




```
importance_df
```

{"summary":"{\n \"name\": \"importance_df\",\n \"rows\": 172,\n \"fields\": [\n {\n \"column\": \"Feature\",\n \"properties\": {\n \"dtype\": \"string\",\n \"num_unique_values\": 172,\n \"samples\": [\n \"FLAG_OWN_REALTY_Y\",\n \"ORGANIZATION_TYPE_Hotel\",\n \"ORGANIZATION_TYPE_Services\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Importance\",\n \"properties\": {\n \"dtype\": \"float32\",\n \"num_unique_values\": 150,\n \"samples\": [\n 0.005802934523671865 5,\n 0.009425627067685127,\n 0.004412588197737932\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n }\n]\n}","type":"dataframe","variable_name":"importance_df"}

```
from statsmodels.stats.outliers_influence import
variance_inflation_factor
import pandas as pd
# Calculate VIF for each feature
vif_data = pd.DataFrame()
vif_data["Feature"] = x.columns
vif_data["VIF"] = [variance_inflation_factor(x.values, i) for i in
range(x.shape[1])]

vif_data
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-199-f0e81c4a4c79> in <cell line: 6>()
      4 vif_data = pd.DataFrame()
      5 vif_data["Feature"] = x.columns
----> 6 vif_data["VIF"] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
      7
      8 vif_data

<ipython-input-199-f0e81c4a4c79> in <listcomp>(.0)
      4 vif_data = pd.DataFrame()
      5 vif_data["Feature"] = x.columns
----> 6 vif_data["VIF"] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
      7
      8 vif_data

/usr/local/lib/python3.10/dist-packages/statsmodels/stats/outliers_influence.py in variance_inflation_factor(exog, exog_idx)
    194     mask = np.arange(k_vars) != exog_idx
    195     x_noti = exog[:, mask]
--> 196     r_squared_i = OLS(x_i, x_noti).fit().rsquared
    197     vif = 1. / (1. - r_squared_i)
    198     return vif

/usr/local/lib/python3.10/dist-packages/statsmodels/regression/linear_model.py in __init__(self, endog, exog, missing, hasconst, **kwargs)
    919                     "An exception will be raised in the next version.")
    920             warnings.warn(msg, ValueWarning)
--> 921         super().__init__(endog, exog, missing=missing,
    922                          hasconst=hasconst, **kwargs)
    923         if "weights" in self._init_keys:

/usr/local/lib/python3.10/dist-packages/statsmodels/regression/linear_model.py in __init__(self, endog, exog, weights, missing, hasconst, **kwargs)
    744         else:
    745             weights = weights.squeeze()
--> 746         super().__init__(endog, exog, missing=missing,
    747                          weights=weights, hasconst=hasconst, **kwargs)
    748         nobs = self.exog.shape[0]

/usr/local/lib/python3.10/dist-packages/statsmodels/regression/linear_model.py in __init__(self, endog, exog, **kwargs)
    198     """
```

```
    199     def __init__(self, endog, exog, **kwargs):
--> 200         super().__init__(endog, exog, **kwargs)
    201         self.pinv_wexog: Float64Array | None = None
    202         self._data_attr.extend(['pinv_wexog', 'wendog',
'wexog', 'weights'])

/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py in
__init__(self, endog, exog, **kwargs)
    268
    269     def __init__(self, endog, exog=None, **kwargs):
--> 270         super().__init__(endog, exog, **kwargs)
    271         self.initialize()
    272

/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py in
__init__(self, endog, exog, **kwargs)
     93         missing = kwargs.pop('missing', 'none')
     94         hasconst = kwargs.pop('hasconst', None)
---> 95         self.data = self._handle_data(endog, exog, missing,
hasconst,
     96                                       **kwargs)
     97         self.k_constant = self.data.k_constant

/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py in
_handle_data(self, endog, exog, missing, hasconst, **kwargs)
    133
    134     def _handle_data(self, endog, exog, missing, hasconst,
**kwargs):
--> 135         data = handle_data(endog, exog, missing, hasconst,
**kwargs)
    136         # kwargs arrays could have changed, easier to just
attach here
    137         for key in kwargs:

/usr/local/lib/python3.10/dist-packages/statsmodels/base/data.py in
handle_data(endog, exog, missing, hasconst, **kwargs)
    673
    674     klass = handle_data_class_factory(endog, exog)
--> 675     return klass(endog, exog=exog, missing=missing,
hasconst=hasconst,
    676                  **kwargs)

/usr/local/lib/python3.10/dist-packages/statsmodels/base/data.py in
__init__(self, endog, exog, missing, hasconst, **kwargs)
     86         self.const_idx = None
     87         self.k_constant = 0
---> 88         self._handle_constant(hasconst)
     89         self._check_integrity()
     90         self._cache = {}
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/base/data.py in
_handle_constant(self, hasconst)
    176                              (np.ones(self.exog.shape[0]),
self.exog))
    177                 rank_augm =
np.linalg.matrix_rank(augmented_exog)
--> 178                 rank_orig = np.linalg.matrix_rank(self.exog)
    179                 self.k_constant = int(rank_orig == rank_augm)
    180                 self.const_idx = None

/usr/local/lib/python3.10/dist-packages/numpy/linalg/linalg.py in
matrix_rank(A, tol, hermitian)
   1920     if A.ndim < 2:
   1921         return int(not all(A==0))
-> 1922     S = svd(A, compute_uv=False, hermitian=hermitian)
   1923     if tol is None:
   1924         tol = S.max(axis=-1, keepdims=True) * max(A.shape[-
2:]) * finfo(S.dtype).eps

/usr/local/lib/python3.10/dist-packages/numpy/linalg/linalg.py in
svd(a, full_matrices, compute_uv, hermitian)
   1691
   1692         signature = 'D->d' if isComplexType(t) else 'd->d'
-> 1693         s = gufunc(a, signature=signature, extobj=extobj)
   1694         s = s.astype(_realType(result_t), copy=False)
   1695         return s

KeyboardInterrupt:
```

```
df1.to_csv('/content/drive/MyDrive/Colab
Notebooks/df1.csv',index=False)
```