Loan Default Analysis

June 5, 2023

1 Description

This project focused on analyzing bank customers to identify those likely to repay loans and those at risk of defaulting. By performing data cleaning, feature engineering, and in-depth analysis, valuable insights were gained regarding loan repayment trends based on various customer attributes. These findings can aid banks and financial institutions in making informed decisions and implementing strategies to mitigate default risks and improve loan approval processes.

1. Data Import and Platform Setup:

- The project started by importing the necessary libraries and loading the bank dataset onto Jupyter Notebook.

2. Data Overview and Primary Key Identification:

- An initial overview of the data was performed to understand its structure and contents.
- The primary key in the dataset was identified, which serves as a unique identifier for each customer.

3.Data Cleaning:

- Fields and records requiring data cleaning were identified based on their descriptions.
- Null values in the dataset were identified and handled appropriately.
- Fields with more than 45% null values were removed from the dataset to ensure data quality.
- Feature Engineering:

4.New fields such as requests per year and requests per hour were cleaned to provide additional insights.

- Null values in the requests per hour column were replaced with zeros.
- Inappropriate gender values were identified and replaced with null values.
- Similarly, inappropriate organization types were identified and replaced with null values.
- The income range was categorized into five distinct categories for better analysis.
- The days field was converted into years by dividing it by 365, creating a new field.
- Binning was performed to categorize customers based on their birth dates.

5.Defaulters vs. Non-Defaulters Analysis:

- The count of defaulters and non-defaulters was determined, providing a clear understanding of loan repayment trends.

6.Data Visualization:

- Loan applications were visualized based on occupation types, allowing for a comprehensive analysis.
- Bivariate analysis was performed to explore the credit distribution among different income categories and family types.

7.Defaulter Analysis Function:

- A function was created to analyze defaulters in each category, providing valuable insights into default patterns.

==

2 Insights

- The lower the highest education of an applicant, the higher the chance of loan default.
- Labourers and sales staff are areas of major concern, with the highest number of applicants and a significant loan default rate. Drivers also show an alarming combination of counts and default percentages.
- Applicants on maternity leave have a substantial 40% loan default rate. Unemployed applicants also have a 35% loan default rate.
- The low-income range has the highest percentage of loan defaults. As the income range increases, the probability of loan default decreases.
- Among different family statuses, married individuals have the highest likelihood of loan default.
- More men default on loans compared to women.

==

2.0.1 1. Data Import and Platform Setup:

[78]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

Data Ingestion Storing data in DataFrame `df`

[79]:
```
df=pd.read_csv('D:/DS/resume projects/EDA risk analysis/application_data.csv')
```

2.0.2 2. Data Overview and Primary Key Identification:

[65]:
```
df.head()
```

[65]:
```
   SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
0      100002       1         Cash loans           M            N
1      100003       0         Cash loans           F            N
2      100004       0    Revolving loans           M            Y
3      100006       0         Cash loans           F            N
4      100007       0         Cash loans           M            N

  FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  \
0               Y             0          202500.0    406597.5      24700.5
1               N             0          270000.0   1293502.5      35698.5
2               Y             0           67500.0    135000.0       6750.0
3               Y             0          135000.0    312682.5      29686.5
4               Y             0          121500.0    513000.0      21865.5

   …  FLAG_DOCUMENT_18 FLAG_DOCUMENT_19 FLAG_DOCUMENT_20 FLAG_DOCUMENT_21  \
0  …                 0                0                0                0
1  …                 0                0                0                0
2  …                 0                0                0                0
3  …                 0                0                0                0
4  …                 0                0                0                0

  AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_DAY  \
0                        0.0                       0.0
1                        0.0                       0.0
2                        0.0                       0.0
3                        NaN                       NaN
4                        0.0                       0.0

   AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
0                         0.0                        0.0
1                         0.0                        0.0
2                         0.0                        0.0
3                         NaN                        NaN
4                         0.0                        0.0

   AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
0                        0.0                         1.0
1                        0.0                         0.0
2                        0.0                         0.0
3                        NaN                         NaN
4                        0.0                         0.0

[5 rows x 122 columns]
```

- checking consistency of primary key unique ID

[66]:
```
df[df.duplicated('SK_ID_CURR')==True]
```

[66]:
```
Empty DataFrame
Columns: [SK_ID_CURR, TARGET, NAME_CONTRACT_TYPE, CODE_GENDER, FLAG_OWN_CAR,
FLAG_OWN_REALTY, CNT_CHILDREN, AMT_INCOME_TOTAL, AMT_CREDIT, AMT_ANNUITY,
AMT_GOODS_PRICE, NAME_TYPE_SUITE, NAME_INCOME_TYPE, NAME_EDUCATION_TYPE,
NAME_FAMILY_STATUS, NAME_HOUSING_TYPE, REGION_POPULATION_RELATIVE, DAYS_BIRTH,
DAYS_EMPLOYED, DAYS_REGISTRATION, DAYS_ID_PUBLISH, OWN_CAR_AGE, FLAG_MOBIL,
FLAG_EMP_PHONE, FLAG_WORK_PHONE, FLAG_CONT_MOBILE, FLAG_PHONE, FLAG_EMAIL,
OCCUPATION_TYPE, CNT_FAM_MEMBERS, REGION_RATING_CLIENT,
REGION_RATING_CLIENT_W_CITY, WEEKDAY_APPR_PROCESS_START,
HOUR_APPR_PROCESS_START, REG_REGION_NOT_LIVE_REGION, REG_REGION_NOT_WORK_REGION,
LIVE_REGION_NOT_WORK_REGION, REG_CITY_NOT_LIVE_CITY, REG_CITY_NOT_WORK_CITY,
LIVE_CITY_NOT_WORK_CITY, ORGANIZATION_TYPE, EXT_SOURCE_1, EXT_SOURCE_2,
EXT_SOURCE_3, APARTMENTS_AVG, BASEMENTAREA_AVG, YEARS_BEGINEXPLUATATION_AVG,
YEARS_BUILD_AVG, COMMONAREA_AVG, ELEVATORS_AVG, ENTRANCES_AVG, FLOORSMAX_AVG,
FLOORSMIN_AVG, LANDAREA_AVG, LIVINGAPARTMENTS_AVG, LIVINGAREA_AVG,
NONLIVINGAPARTMENTS_AVG, NONLIVINGAREA_AVG, APARTMENTS_MODE, BASEMENTAREA_MODE,
YEARS_BEGINEXPLUATATION_MODE, YEARS_BUILD_MODE, COMMONAREA_MODE, ELEVATORS_MODE,
ENTRANCES_MODE, FLOORSMAX_MODE, FLOORSMIN_MODE, LANDAREA_MODE,
LIVINGAPARTMENTS_MODE, LIVINGAREA_MODE, NONLIVINGAPARTMENTS_MODE,
NONLIVINGAREA_MODE, APARTMENTS_MEDI, BASEMENTAREA_MEDI,
YEARS_BEGINEXPLUATATION_MEDI, YEARS_BUILD_MEDI, COMMONAREA_MEDI, ELEVATORS_MEDI,
ENTRANCES_MEDI, FLOORSMAX_MEDI, FLOORSMIN_MEDI, LANDAREA_MEDI,
LIVINGAPARTMENTS_MEDI, LIVINGAREA_MEDI, NONLIVINGAPARTMENTS_MEDI,
NONLIVINGAREA_MEDI, FONDKAPREMONT_MODE, HOUSETYPE_MODE, TOTALAREA_MODE,
WALLSMATERIAL_MODE, EMERGENCYSTATE_MODE, OBS_30_CNT_SOCIAL_CIRCLE,
DEF_30_CNT_SOCIAL_CIRCLE, OBS_60_CNT_SOCIAL_CIRCLE, DEF_60_CNT_SOCIAL_CIRCLE,
DAYS_LAST_PHONE_CHANGE, FLAG_DOCUMENT_2, FLAG_DOCUMENT_3, FLAG_DOCUMENT_4,
FLAG_DOCUMENT_5, …]
Index: []

[0 rows x 122 columns]
```

2.0.3 3. Data Cleaning

- Fields and records requiring data cleaning were identified based on their descriptions.

[67]:
```
df.describe()
```

[67]:
```
          SK_ID_CURR         TARGET   CNT_CHILDREN  AMT_INCOME_TOTAL  \
count  307511.000000  307511.000000  307511.000000      3.075110e+05
mean   278180.518577       0.080729       0.417052      1.687979e+05
std    102790.175348       0.272419       0.722121      2.371231e+05
min    100002.000000       0.000000       0.000000      2.565000e+04
25%    189145.500000       0.000000       0.000000      1.125000e+05
```

```
50%    278202.000000       0.000000       0.000000      1.471500e+05
75%    367142.500000       0.000000       1.000000      2.025000e+05
max    456255.000000       1.000000      19.000000      1.170000e+08

         AMT_CREDIT    AMT_ANNUITY  AMT_GOODS_PRICE  \
count  3.075110e+05  307499.000000     3.072330e+05
mean   5.990260e+05   27108.573909     5.383962e+05
std    4.024908e+05   14493.737315     3.694465e+05
min    4.500000e+04    1615.500000     4.050000e+04
25%    2.700000e+05   16524.000000     2.385000e+05
50%    5.135310e+05   24903.000000     4.500000e+05
75%    8.086500e+05   34596.000000     6.795000e+05
max    4.050000e+06  258025.500000     4.050000e+06

       REGION_POPULATION_RELATIVE     DAYS_BIRTH  DAYS_EMPLOYED  …  \
count               307511.000000  307511.000000  307511.000000  …
mean                     0.020868  -16036.995067   63815.045904  …
std                      0.013831    4363.988632  141275.766519  …
min                      0.000290  -25229.000000  -17912.000000  …
25%                      0.010006  -19682.000000   -2760.000000  …
50%                      0.018850  -15750.000000   -1213.000000  …
75%                      0.028663  -12413.000000    -289.000000  …
max                      0.072508   -7489.000000  365243.000000  …

       FLAG_DOCUMENT_18  FLAG_DOCUMENT_19  FLAG_DOCUMENT_20  FLAG_DOCUMENT_21  \
count     307511.000000     307511.000000     307511.000000     307511.000000
mean           0.008130          0.000595          0.000507          0.000335
std            0.089798          0.024387          0.022518          0.018299
min            0.000000          0.000000          0.000000          0.000000
25%            0.000000          0.000000          0.000000          0.000000
50%            0.000000          0.000000          0.000000          0.000000
75%            0.000000          0.000000          0.000000          0.000000
max            1.000000          1.000000          1.000000          1.000000

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

       AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
count               265992.000000              265992.000000
mean                     0.034362                   0.267395
```

```
std                    0.204685                  0.916002
min                    0.000000                  0.000000
25%                    0.000000                  0.000000
50%                    0.000000                  0.000000
75%                    0.000000                  0.000000
max                    8.000000                 27.000000

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

[8 rows x 106 columns]
```

[68]:
```
dfna=df.isnull().mean()*100
dfna.head()
```

[68]:
```
SK_ID_CURR            0.0
TARGET                0.0
NAME_CONTRACT_TYPE    0.0
CODE_GENDER           0.0
FLAG_OWN_CAR          0.0
dtype: float64
```

- identifying columns with more than 45% null values

[69]:
```
nn=df.isna().sum().sort_values(ascending=False)
nn=nn[nn.values>0.45*len(df)].reset_index()
#nn=nn.rename(columns={'index':'null'})
nn=nn.rename(columns={'index': 'columns', 0: 'count'})
nn
```

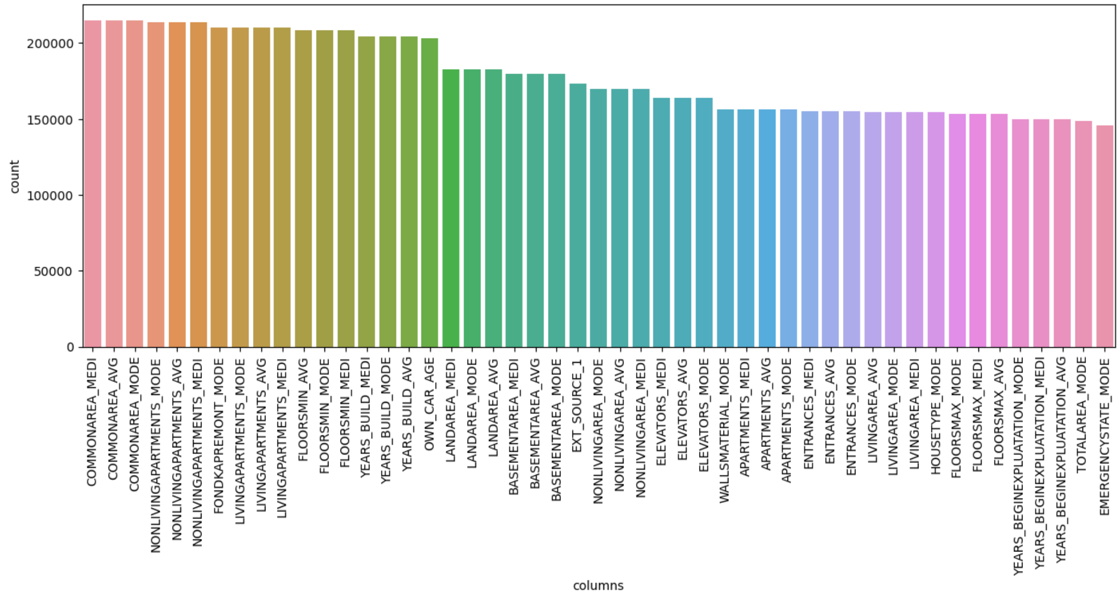
[69]:

	columns	count
0	COMMONAREA_MEDI	214865
1	COMMONAREA_AVG	214865
2	COMMONAREA_MODE	214865
3	NONLIVINGAPARTMENTS_MODE	213514
4	NONLIVINGAPARTMENTS_AVG	213514
5	NONLIVINGAPARTMENTS_MEDI	213514
6	FONDKAPREMONT_MODE	210295
7	LIVINGAPARTMENTS_MODE	210199
8	LIVINGAPARTMENTS_AVG	210199
9	LIVINGAPARTMENTS_MEDI	210199

```
10                 FLOORSMIN_AVG  208642
11                FLOORSMIN_MODE  208642
12                FLOORSMIN_MEDI  208642
13              YEARS_BUILD_MEDI  204488
14              YEARS_BUILD_MODE  204488
15               YEARS_BUILD_AVG  204488
16                   OWN_CAR_AGE  202929
17                 LANDAREA_MEDI  182590
18                 LANDAREA_MODE  182590
19                  LANDAREA_AVG  182590
20             BASEMENTAREA_MEDI  179943
21              BASEMENTAREA_AVG  179943
22             BASEMENTAREA_MODE  179943
23                  EXT_SOURCE_1  173378
24            NONLIVINGAREA_MODE  169682
25             NONLIVINGAREA_AVG  169682
26            NONLIVINGAREA_MEDI  169682
27                ELEVATORS_MEDI  163891
28                 ELEVATORS_AVG  163891
29                ELEVATORS_MODE  163891
30            WALLSMATERIAL_MODE  156341
31               APARTMENTS_MEDI  156061
32                APARTMENTS_AVG  156061
33               APARTMENTS_MODE  156061
34                ENTRANCES_MEDI  154828
35                 ENTRANCES_AVG  154828
36                ENTRANCES_MODE  154828
37                LIVINGAREA_AVG  154350
38               LIVINGAREA_MODE  154350
39               LIVINGAREA_MEDI  154350
40                HOUSETYPE_MODE  154297
41                FLOORSMAX_MODE  153020
42                FLOORSMAX_MEDI  153020
43                 FLOORSMAX_AVG  153020
44  YEARS_BEGINEXPLUATATION_MODE  150007
45  YEARS_BEGINEXPLUATATION_MEDI  150007
46   YEARS_BEGINEXPLUATATION_AVG  150007
47                TOTALAREA_MODE  148431
48           EMERGENCYSTATE_MODE  145755
```

- ploting columns with more than 45% nulls

[70]:
```
plt.figure(figsize=(15,5))
sns.barplot(x='columns', y='count', data=nn)
plt.xticks(rotation=90)
plt.show()
```



- dropping columns with more than 45% null and storing data in dfdd

[71]:
```
l=list(nn['columns'])
dfd=df.drop(l,axis=1)
dfdd=dfd.isna().sum().sort_values(ascending=False)
dfdd=dfdd.reset_index()
dfdd.iloc[:,1]
dfdd['pp']=(100*dfdd.iloc[:,1])/len(df)
dfdd
```

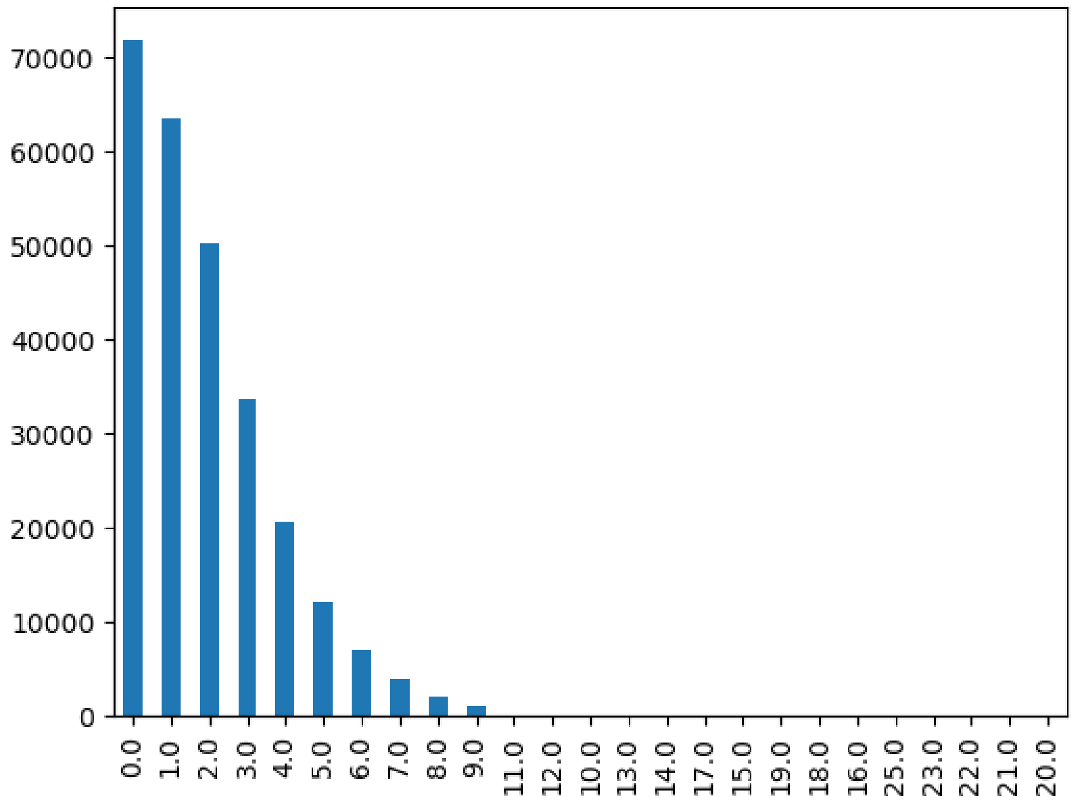
[71]:
```
                          index      0         pp
0               OCCUPATION_TYPE  96391  31.345545
1                  EXT_SOURCE_3  60965  19.825307
2    AMT_REQ_CREDIT_BUREAU_YEAR  41519  13.501631
3     AMT_REQ_CREDIT_BUREAU_QRT  41519  13.501631
4     AMT_REQ_CREDIT_BUREAU_MON  41519  13.501631
..                          …      …          …
68   REG_REGION_NOT_LIVE_REGION      0   0.000000
69   REG_REGION_NOT_WORK_REGION      0   0.000000
70  LIVE_REGION_NOT_WORK_REGION      0   0.000000
71                       TARGET      0   0.000000
72         REG_CITY_NOT_LIVE_CITY      0   0.000000

[73 rows x 3 columns]
```

2.0.4 4.New fields such as requests per year and requests per hour were cleaned to provide additional insights.

- Null values in the requests per hour column were replaced with zeros.

[72]:
```
df.AMT_REQ_CREDIT_BUREAU_YEAR.value_counts().plot(kind='bar')
```

[72]:
```
<Axes: >
```

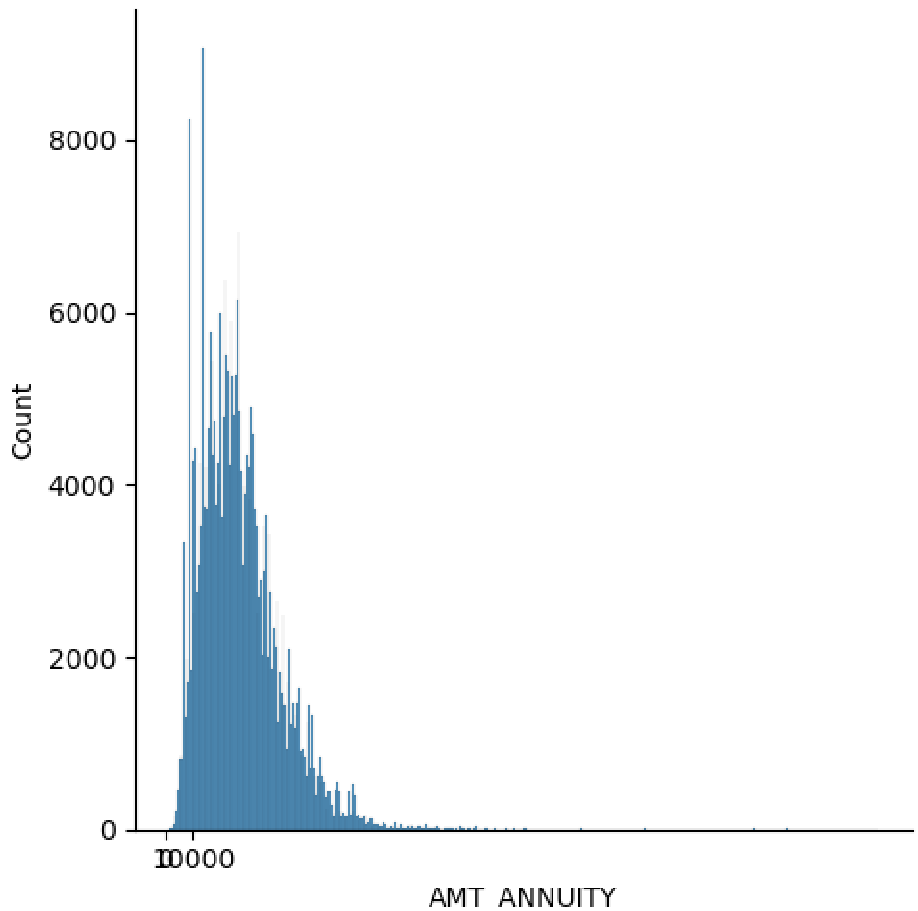


- Creating a new DataFrame called `dfd1` for data wrangling.

[73]:
```
dfd1=dfd
dfd1[['AMT_REQ_CREDIT_BUREAU_HOUR','AMT_REQ_CREDIT_BUREAU_DAY','AMT_REQ_CREDIT_BUREAU_WEEK','A
↪fillna(0)
```

[74]:
```
# cleaning AMT_ANNUITY column
plt.figure(figsize=(10,1))
sns.displot(dfd1.AMT_ANNUITY)
plt.xticks(range(0,20000,10000))
plt.show()
```

```
<Figure size 1000x100 with 0 Axes>
```



[75]:
```
dfd1.AMT_ANNUITY.isna().sum()
```

[75]: 12

[76]:
```
dfd1['AMT_ANNUITY']=dfd['AMT_ANNUITY'].fillna(dfd.AMT_ANNUITY.median())
```

- Inappropriate gender values were identified and replaced with null values.

[53]:
```
dfd1.CODE_GENDER.value_counts()
```

[53]:
```
F      202448
M      105059
XNA         4
Name: CODE_GENDER, dtype: int64
```

[54]:
```
dfd1.loc[dfd1.CODE_GENDER=='XNA','CODE_GENDER']=np.NaN
```

[55]:
```
dfd1.CODE_GENDER.value_counts()
```

[55]:
```
F    202448
M    105059
Name: CODE_GENDER, dtype: int64
```

- Similarly, inappropriate organization types were identified and replaced with null values.

[56]:
```python
dfd1.ORGANIZATION_TYPE
```

[56]:
```
0         Business Entity Type 3
1                         School
2                     Government
3         Business Entity Type 3
4                       Religion
                   …
307506                  Services
307507                       XNA
307508                    School
307509    Business Entity Type 1
307510    Business Entity Type 3
Name: ORGANIZATION_TYPE, Length: 307511, dtype: object
```

[57]:
```python
dfd1.loc[dfd1.ORGANIZATION_TYPE=='XNA','ORGANIZATION_TYPE']=np.NaN
```

- The AMT_INCOME_RANGE field is dividing into diffrent category, creating a new field.

[58]:
```python
dfd1['AMT_INCOME_RANGE']=pd.qcut(dfd1.AMT_INCOME_TOTAL,q=[0,0.2,0.5,0.85,0.
↪95,1],labels=['VERY_LOW','LOW','MEDIUM','HIGH','VERY HIGH'])
```

[59]:
```python
dfd1['AMT_INCOME_RANGE']
```

[59]:
```
0           MEDIUM
1             HIGH
2         VERY_LOW
3              LOW
4              LOW
            …
307506      MEDIUM
307507    VERY_LOW
307508      MEDIUM
307509      MEDIUM
307510      MEDIUM
Name: AMT_INCOME_RANGE, Length: 307511, dtype: category
Categories (5, object): ['VERY_LOW' < 'LOW' < 'MEDIUM' < 'HIGH' < 'VERY HIGH']
```

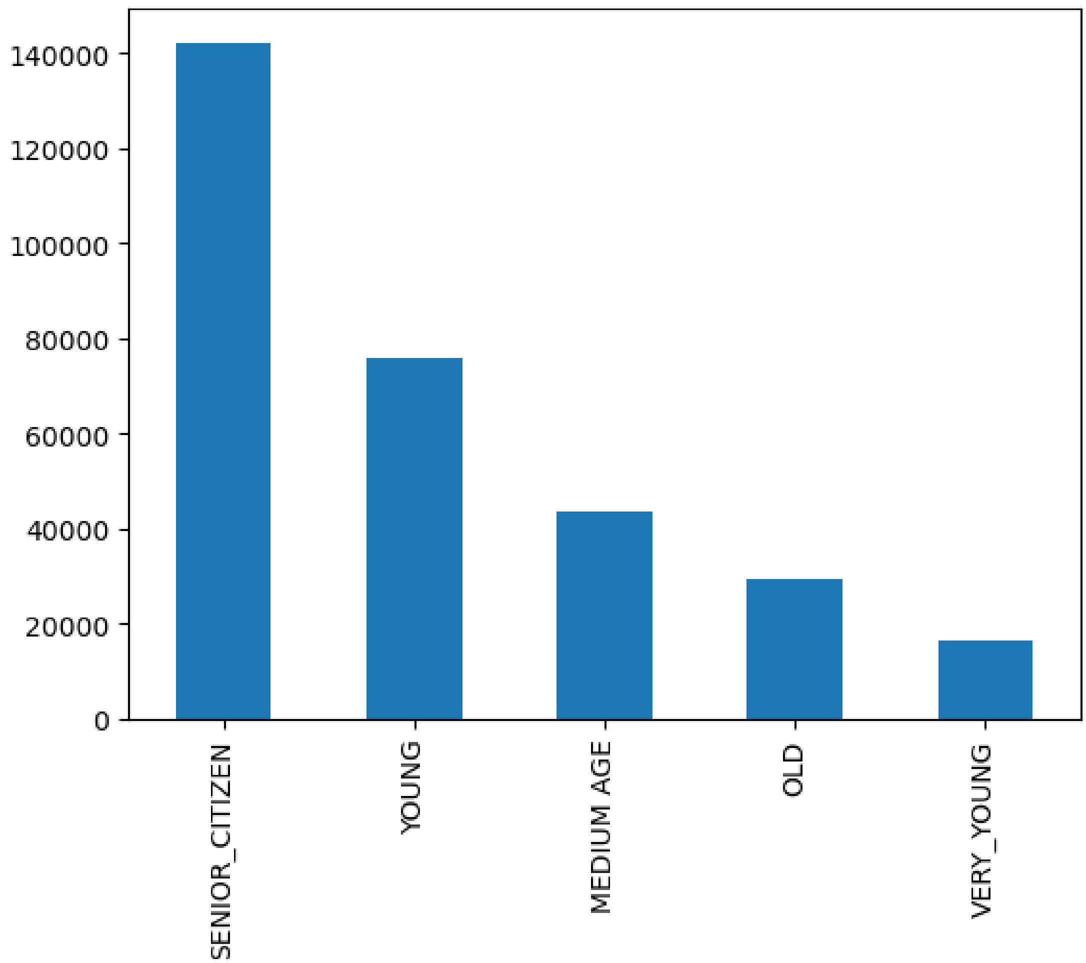
- The days field was converted into years by dividing it by 365, creating a new field.
- Binning was performed to categorize customers based on their birth dates.

[60]:
```python
err=[i for i in dfd1 if i.startswith('DAYS')]
dfd1[err]=abs(dfd1[err])
```

```
dfd1.DAYS_BIRTH=(dfd1.DAYS_BIRTH/365).astype(int)
dfd1['DAYS_BIRTH_BINS']=pd.cut(dfd1.
 ↪DAYS_BIRTH,bins=[16,25,35,40,60,100],labels=['VERY_YOUNG','YOUNG','MEDIUM␣
 ↪AGE','SENIOR_CITIZEN','OLD'])
```

[61]:
```
dfd1['DAYS_BIRTH_BINS'].value_counts().plot(kind='bar')
```

[61]: <Axes: >



2.0.5 5.Defaulters vs. Non-Defaulters Analysis:

- The count of defaulters and non-defaulters was determined, providing a clear understanding of loan repayment trends.

[29]:
```
dfd1.TARGET.value_counts()
```

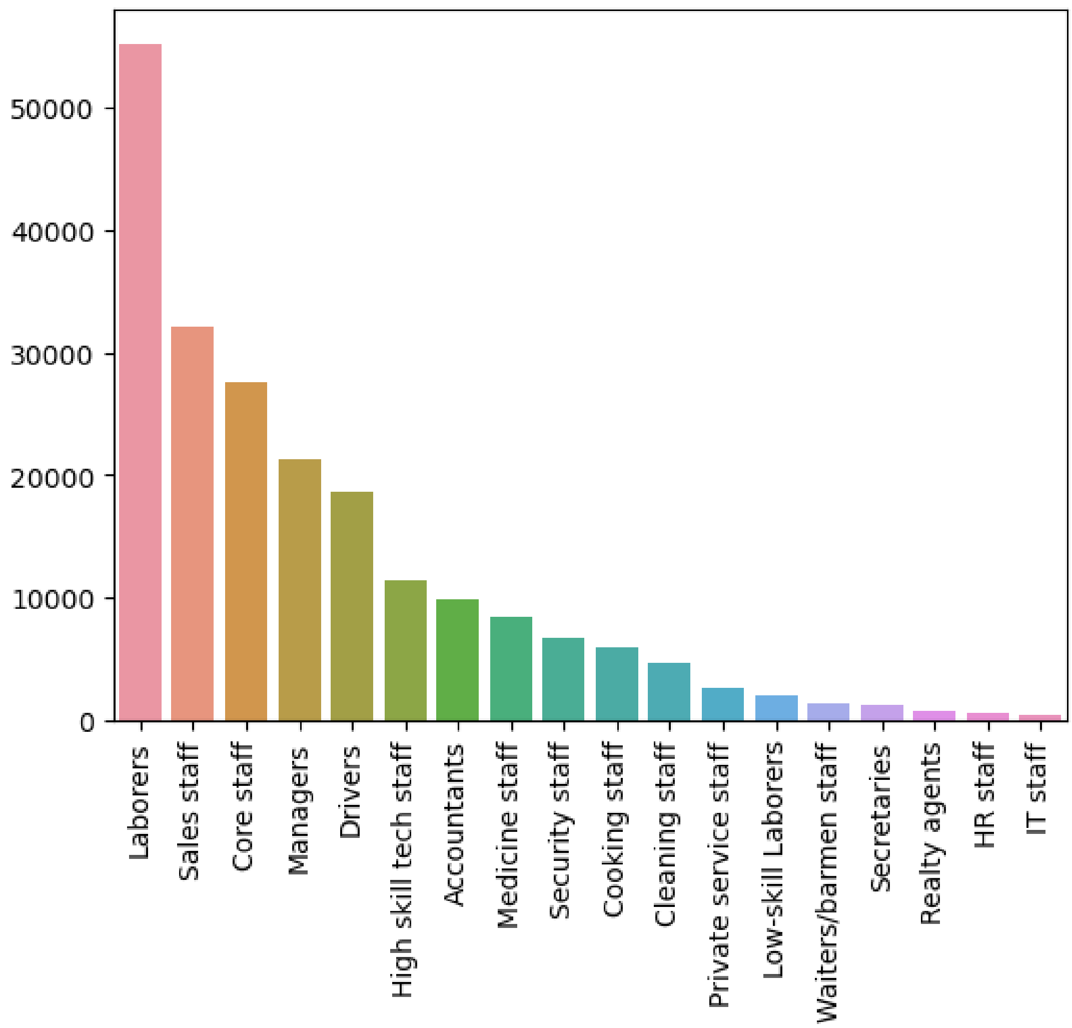
[29]:
```
0    282686
1     24825
Name: TARGET, dtype: int64
```

[30]:
```
defolter=dfd1[dfd1.TARGET==1]
non_defolter=dfd1[dfd1.TARGET==0]
```

2.0.6 6.Data Visualization:

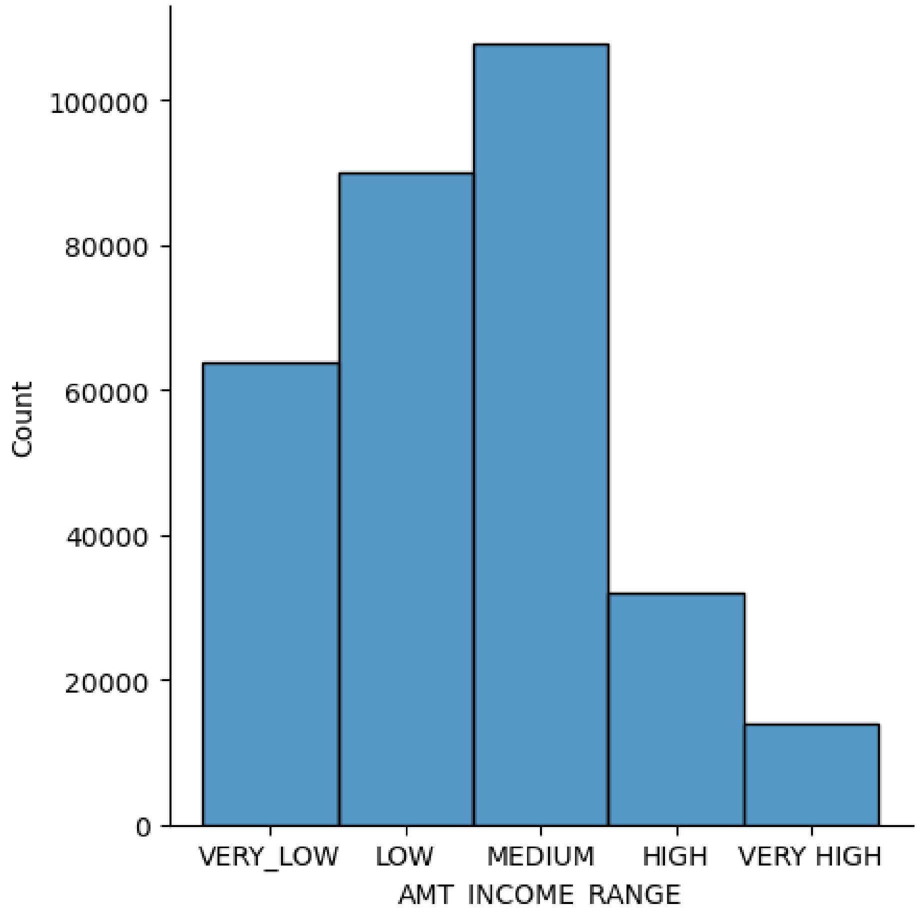
- Loan applications were visualized based on occupation types, allowing for a comprehensive analysis.

[31]:
```
c=dfd1.OCCUPATION_TYPE.value_counts()
sns.barplot(x=c.index,y=c.values)
plt.xticks(rotation=90)
plt.show()
```



[32]:
```python
sns.displot(dfd1['AMT_INCOME_RANGE'])
```

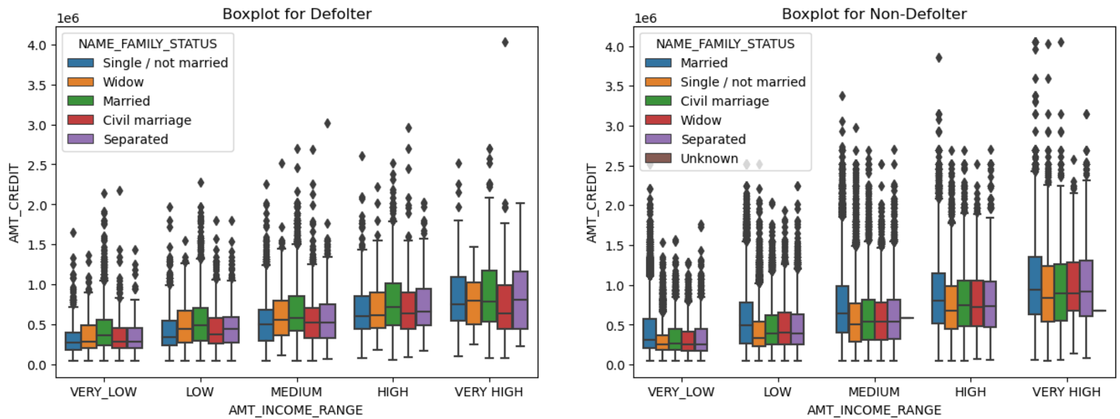
[32]: <seaborn.axisgrid.FacetGrid at 0x2071080c370>



- Bivariate analysis was performed to explore the credit distribution among different income categories and family types.

[33]:
```python
def ss(col1, col2, col3):
    f,ax=plt.subplots(1,2,figsize=(15,5))
    ax[0].set_title('Boxplot for Defolter')
    sns.boxplot(data=defolter,x=col1, y=col2, hue=col3, ax=ax[0])

    ax[1].set_title('Boxplot for Non-Defolter')
    sns.boxplot(data=non_defolter,x=col1, y=col2, hue=col3, ax=ax[1])

    plt.show()
```

```
ss('AMT_INCOME_RANGE', 'AMT_CREDIT', 'NAME_FAMILY_STATUS')
```

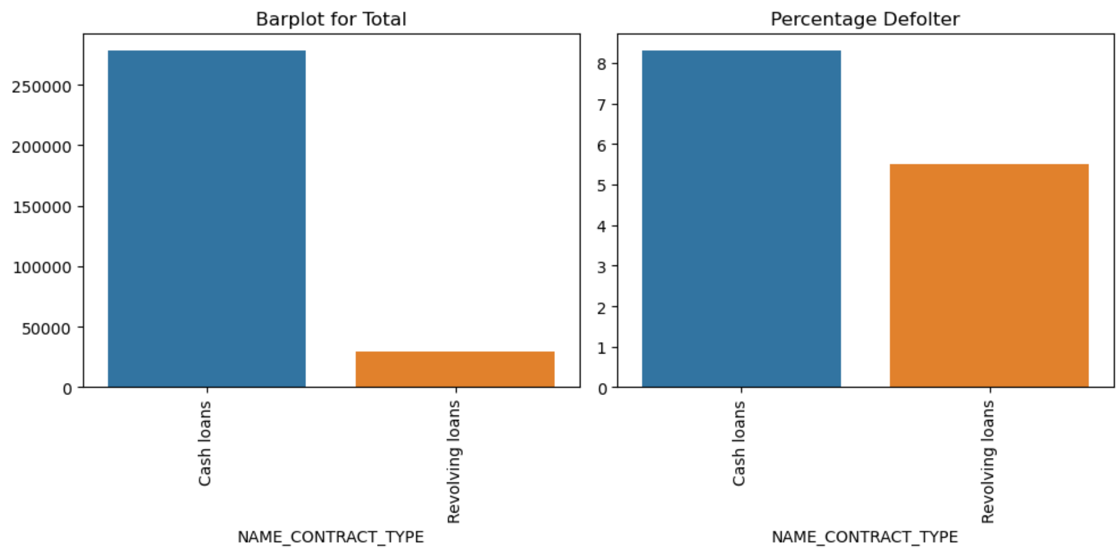


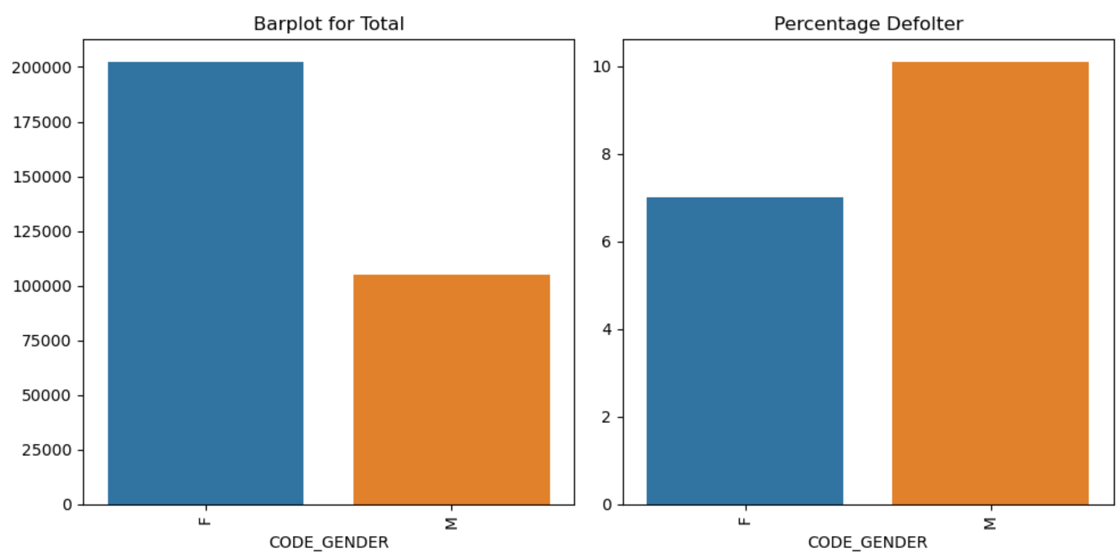
2.0.7 7.Defaulter Analysis Function:

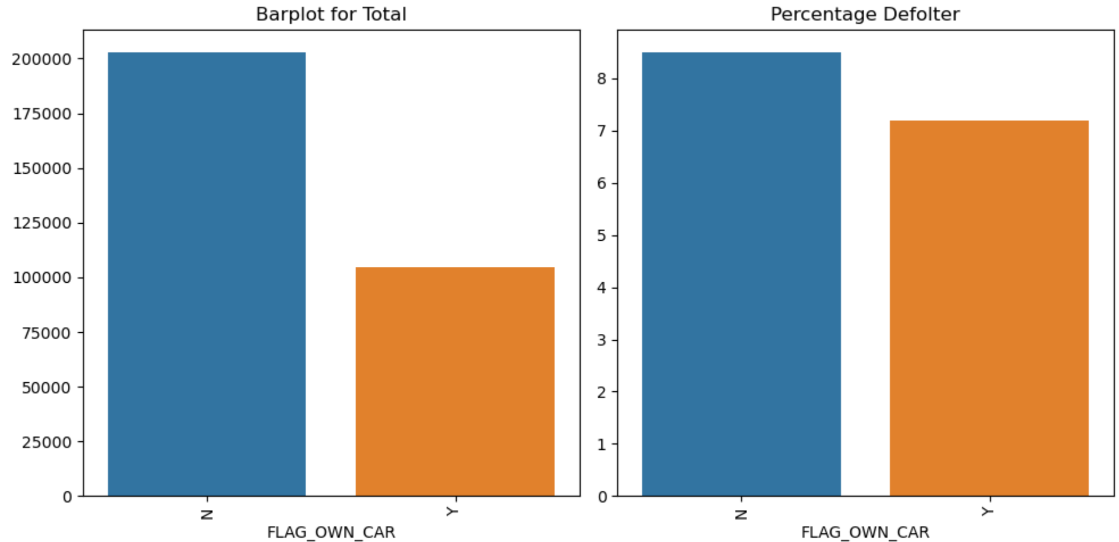
- A function was created to analyze defaulters in each category, providing valuable insights into default patterns.

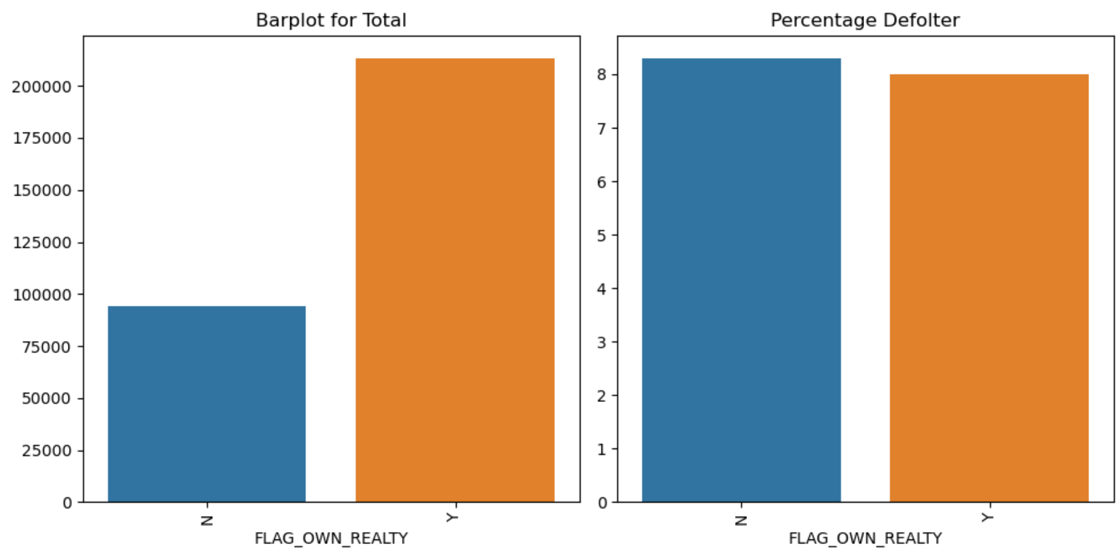
[34]:
```
def Category_analysis(col):
    f0 = dfd1.groupby(col)['TARGET'].count()
    f = defolter.groupby(col)['TARGET'].count()
    pct = round(f * 100 / f0, 1)

    fig, ax = plt.subplots(1, 2, figsize=(10, 5))  # Updated figsize

    ax[0].set_title('Barplot for Total')
    sns.barplot(x=f0.index, y=f0.values, ax=ax[0])
    ax[0].tick_params(axis='x', rotation=90)  # Updated tick_params

    ax[1].set_title('Percentage Defolter')
    sns.barplot(x=pct.index, y=pct.values, ax=ax[1])
    ax[1].tick_params(axis='x', rotation=90)  # Updated tick_params

    plt.tight_layout()
    plt.show()
```

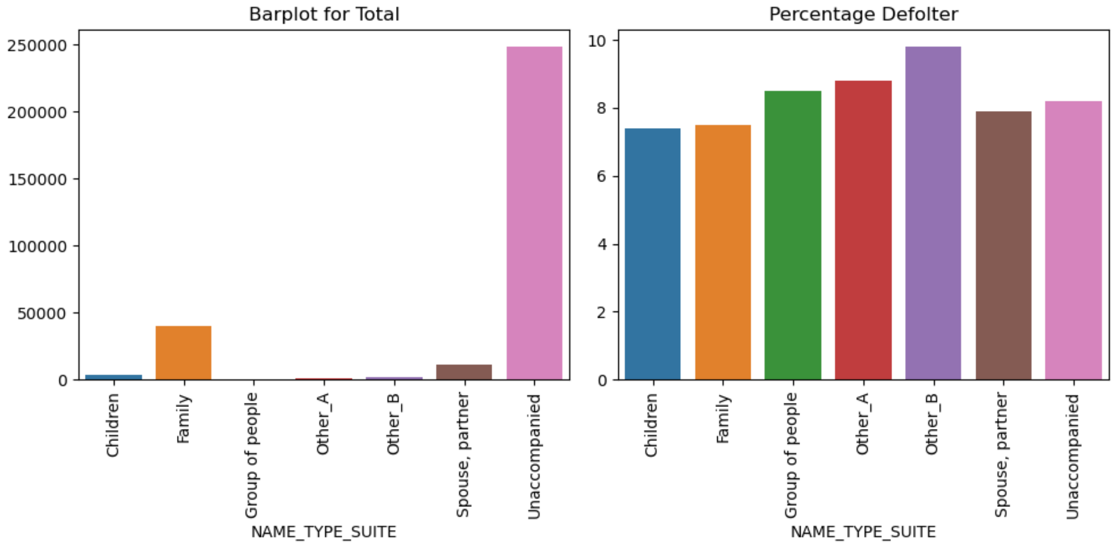
[35]:
```
catcol1=dfd1.columns[dfd1.dtypes=='O']
for i in catcol1:
    Category_analysis(i)
    plt.show()
```

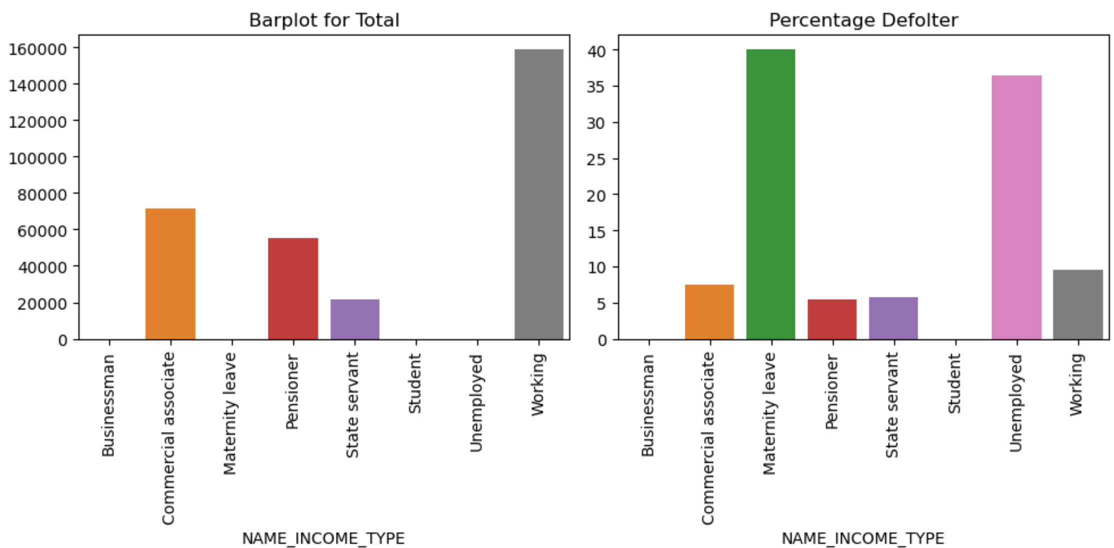

Barplot for Total
250000
200000
150000
100000
50000
0
Cash loans
Revolving loans
NAME_CONTRACT_TYPE
Percentage Defolter
8
7
6
5
4
3
2
1
0
Cash loans
Revolving loans
NAME_CONTRACT_TYPE

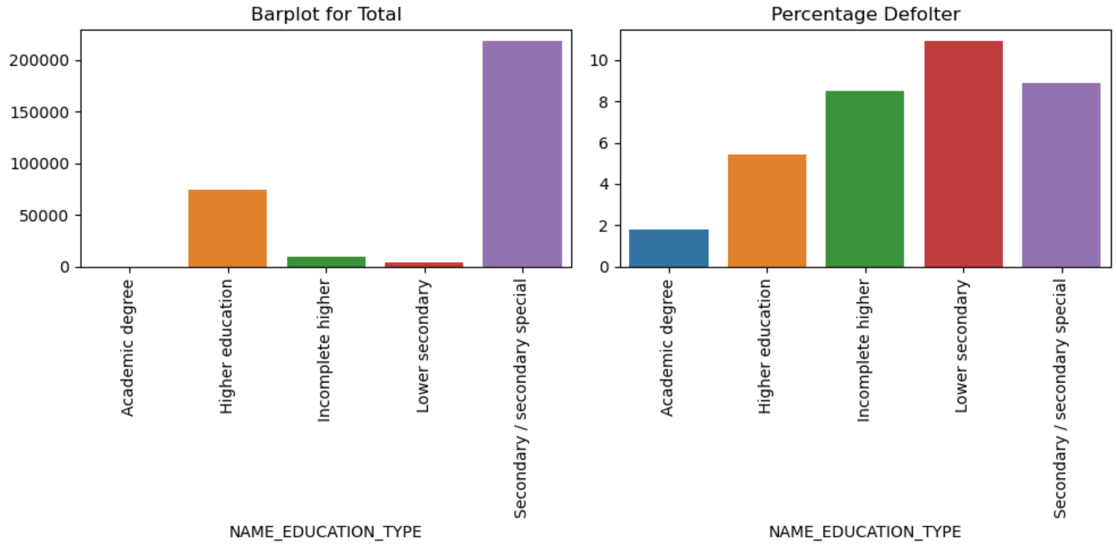

Barplot for Total
200000
175000
150000
125000
100000
75000
50000
25000
0
F
M
CODE_GENDER
Percentage Defolter
10
8
6
4
2
0
F
M
CODE_GENDER

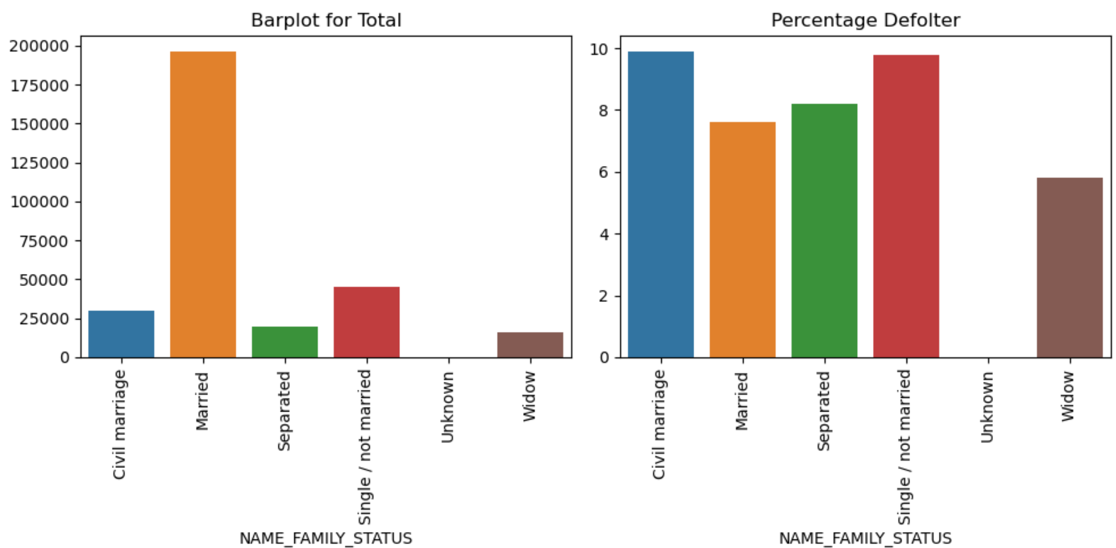

Barplot for Total
Percentage Defolter
200000
175000
150000
125000
100000
75000
50000
25000
0
8
7
6
5
4
3
2
1
0
N
Y
FLAG_OWN_CAR
FLAG_OWN_CAR

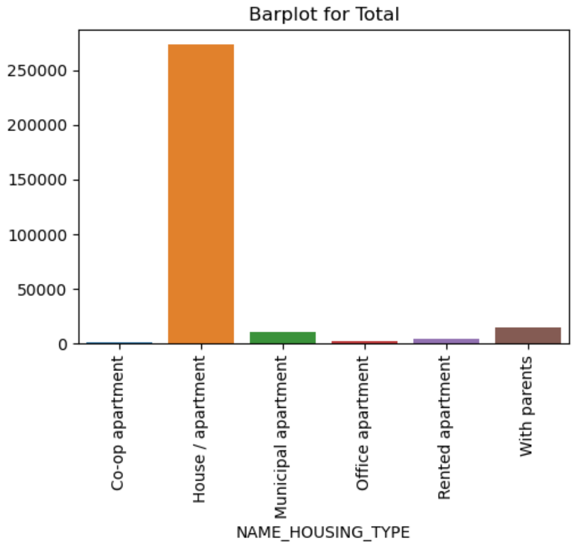

Barplot for Total
Percentage Defolter
200000
175000
150000
125000
100000
75000
50000
25000
0
8
7
6
5
4
3
2
1
0
N
Y
FLAG_OWN_REALTY
FLAG_OWN_REALTY

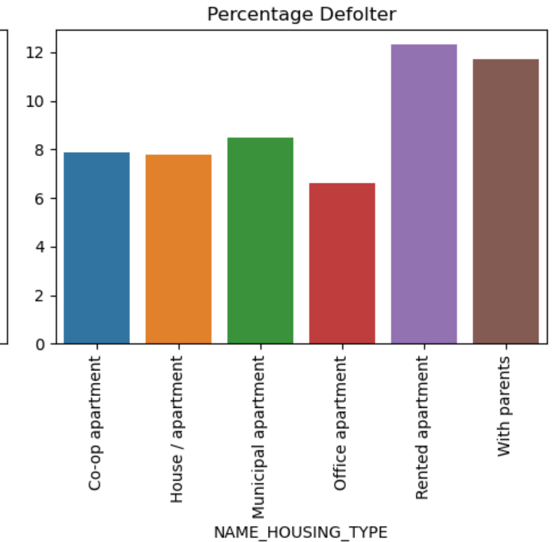

Barplot for Total
Percentage Defolter
NAME_TYPE_SUITE
Children
Family
Group of people
Other_A
Other_B
Spouse, partner
Unaccompanied
0
50000
100000
150000
200000
250000
0
2
4
6
8
10

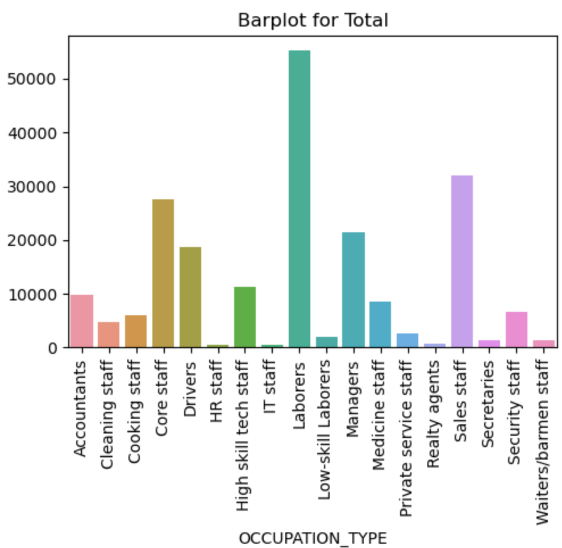

Barplot for Total
Percentage Defolter
NAME_INCOME_TYPE
Businessman
Commercial associate
Maternity leave
Pensioner
State servant
Student
Unemployed
Working
0
20000
40000
60000
80000
100000
120000
140000
160000
0
5
10
15
20
25
30
35
40

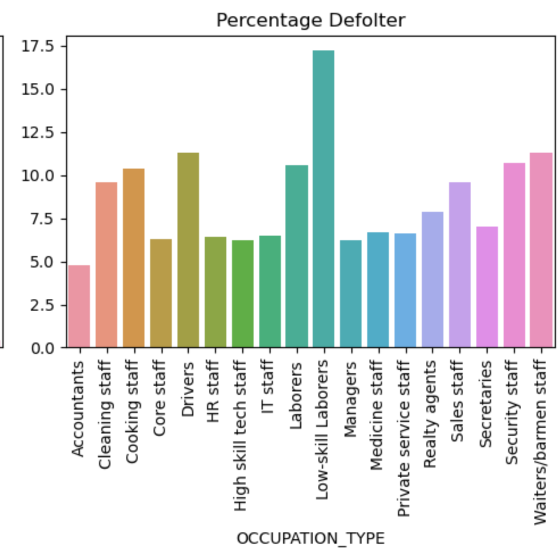

Barplot for Total
200000
150000
100000
50000
0
Academic degree
Higher education
Incomplete higher
Lower secondary
Secondary / secondary special
NAME_EDUCATION_TYPE
Percentage Defolter
10
8
6
4
2
0
Academic degree
Higher education
Incomplete higher
Lower secondary
Secondary / secondary special
NAME_EDUCATION_TYPE

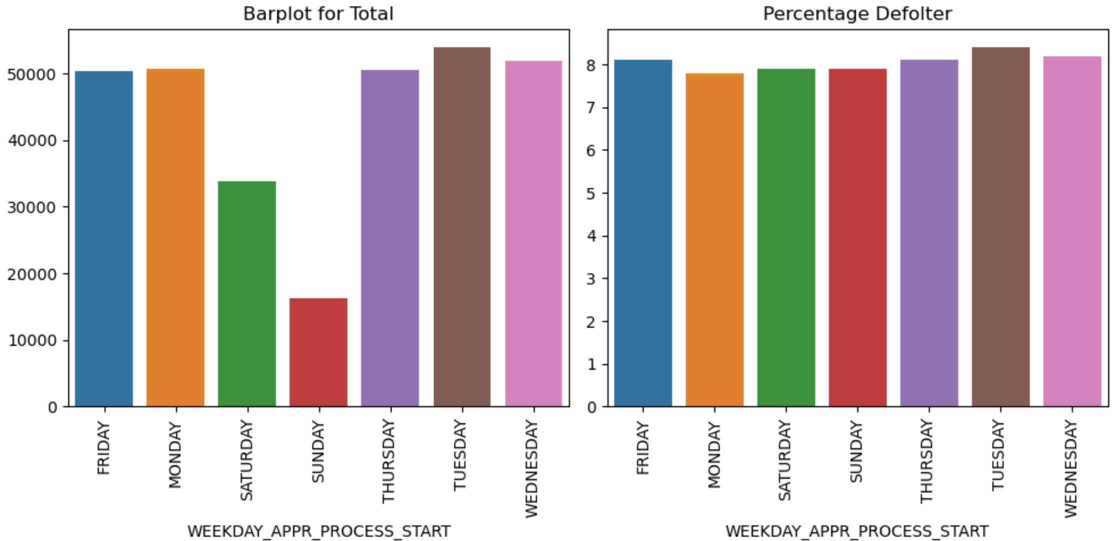

Barplot for Total
200000
175000
150000
125000
100000
75000
50000
25000
0
Civil marriage
Married
Separated
Single / not married
Unknown
Widow
NAME_FAMILY_STATUS
Percentage Defolter
10
8
6
4
2
0
Civil marriage
Married
Separated
Single / not married
Unknown
Widow
NAME_FAMILY_STATUS

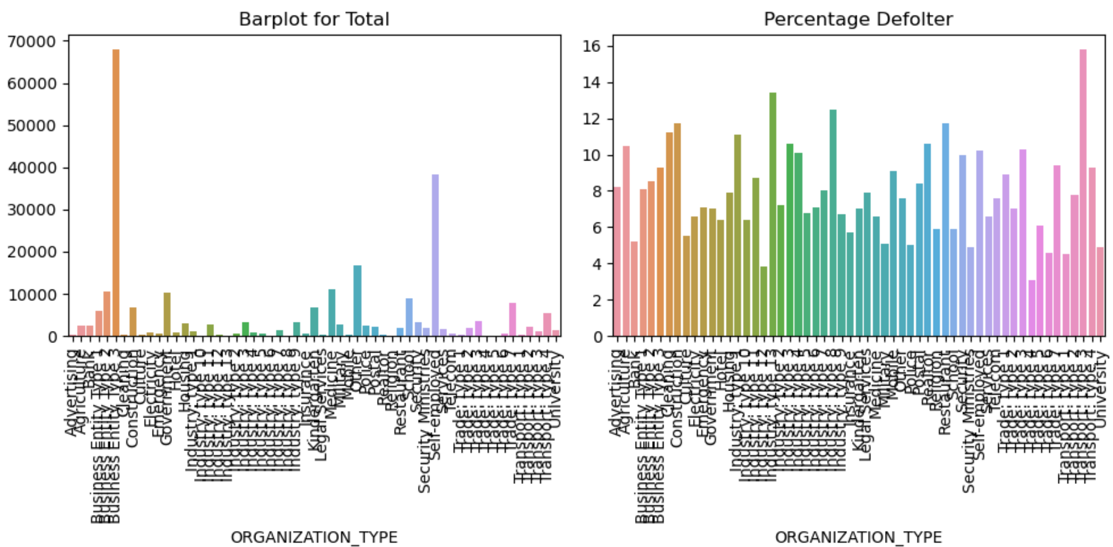

Barplot for Total
250000
200000
150000
100000
50000
0
Co-op apartment
House / apartment
Municipal apartment
Office apartment
Rented apartment
With parents
NAME_HOUSING_TYPE


Percentage Defolter
12
10
8
6
4
2
0
Co-op apartment
House / apartment
Municipal apartment
Office apartment
Rented apartment
With parents
NAME_HOUSING_TYPE


Barplot for Total
50000
40000
30000
20000
10000
0
Accountants
Cleaning staff
Cooking staff
Core staff
Drivers
HR staff
High skill tech staff
IT staff
Laborers
Low-skill Laborers
Managers
Medicine staff
Private service staff
Realty agents
Sales staff
Secretaries
Security staff
Waiters/barmen staff
OCCUPATION_TYPE


Percentage Defolter
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Accountants
Cleaning staff
Cooking staff
Core staff
Drivers
HR staff
High skill tech staff
IT staff
Laborers
Low-skill Laborers
Managers
Medicine staff
Private service staff
Realty agents
Sales staff
Secretaries
Security staff
Waiters/barmen staff
OCCUPATION_TYPE


Barplot for Total
50000
40000
30000
20000
10000
0
FRIDAY
MONDAY
SATURDAY
SUNDAY
THURSDAY
TUESDAY
WEDNESDAY
WEEKDAY_APPR_PROCESS_START
Percentage Defolter
8
7
6
5
4
3
2
1
0
FRIDAY
MONDAY
SATURDAY
SUNDAY
THURSDAY
TUESDAY
WEDNESDAY
WEEKDAY_APPR_PROCESS_START


Barplot for Total
70000
60000
50000
40000
30000
20000
10000
0
ORGANIZATION_TYPE
Percentage Defolter
16
14
12
10
8
6
4
2
0
ORGANIZATION_TYPE