Vanshita Biyani - 067 - PGDBDA - Statistics Module End Exam

Problem Statement- Loan providing companies face challenges in assessing applicants with insufficient or non-existent credit histories, leading to potential defaults. This case study focuses on identifying patterns in loan repayment behavior to address risks associated with loan approvals and defaults.

```
import pandas as pd #Import necessary library for data manipulation
```

Load Application Data File

```
dfa=pd.read_csv(r'/content/drive/MyDrive/Vanshita/application_data.csv') #load application data into dataframe
```

```
pd.set_option('display.max_columns', None) #set option to display all columns of dataset
```

```
dfa.shape #Get the number of rows and columns for application data
```

```
(307511, 122)
```

```
dfa #Display application data
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CRED
0	100002	1	Cash loans	M	N	Y	0	202500.0	40659
1	100003	0	Cash loans	F	N	N	0	270000.0	129350
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	13500
3	100006	0	Cash loans	F	N	Y	0	135000.0	31268
4	100007	0	Cash loans	M	N	Y	0	121500.0	51300
...	...	...	...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	N	0	157500.0	25470
307507	456252	0	Cash loans	F	N	Y	0	72000.0	26955
307508	456253	0	Cash loans	F	N	Y	0	153000.0	67766
307509	456254	1	Cash loans	F	N	Y	0	171000.0	37010
307510	456255	0	Cash loans	F	N	N	0	157500.0	67500

307511 rows × 122 columns

```
dfa.info() #Get information about columns,count of non-null values,memory usage,etc about application data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

```
for i in dfa.columns:
  print(i)    #print all columns of application data
```

```
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
FLAG_DOCUMENT_6
FLAG_DOCUMENT_7
FLAG_DOCUMENT_8
FLAG_DOCUMENT_9
FLAG_DOCUMENT_10
FLAG_DOCUMENT_11
FLAG_DOCUMENT_12
FLAG_DOCUMENT_13
FLAG_DOCUMENT_14
FLAG_DOCUMENT_15
FLAG_DOCUMENT_16
FLAG_DOCUMENT_17
FLAG_DOCUMENT_18
FLAG_DOCUMENT_19
FLAG_DOCUMENT_20
FLAG_DOCUMENT_21
AMT_REQ_CREDIT_BUREAU_HOUR
AMT_REQ_CREDIT_BUREAU_DAY
AMT_REQ_CREDIT_BUREAU_WEEK
AMT_REQ_CREDIT_BUREAU_MON
AMT_REQ_CREDIT_BUREAU_QRT
AMT_REQ_CREDIT_BUREAU_YEAR
```

```
# List of irrelevant columns to drop from application data
columns_to_drop = [
     "DAYS_BIRTH",
    "DAYS_EMPLOYED", "DAYS_REGISTRATION", "DAYS_ID_PUBLISH", "FLAG_MOBIL",
    "FLAG_EMP_PHONE", "FLAG_WORK_PHONE", "FLAG_CONT_MOBILE", "FLAG_PHONE", "FLAG_EMAIL",
     "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",
     "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", "FLAG_DOCUMENT_6",
    "FLAG_DOCUMENT_7", "FLAG_DOCUMENT_8", "FLAG_DOCUMENT_9", "FLAG_DOCUMENT_10",
    "FLAG_DOCUMENT_11", "FLAG_DOCUMENT_12", "FLAG_DOCUMENT_13", "FLAG_DOCUMENT_14",
    "FLAG_DOCUMENT_15", "FLAG_DOCUMENT_16", "FLAG_DOCUMENT_17", "FLAG_DOCUMENT_18",
    "FLAG_DOCUMENT_19", "FLAG_DOCUMENT_20", "FLAG_DOCUMENT_21"
]
dfa=dfa.drop(columns_to_drop,axis=1)
```

```
dfa #Display application data after removing irrelevant columns
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CRED
0	100002	1	Cash loans	M	N	Y	0	202500.0	40659
1	100003	0	Cash loans	F	N	N	0	270000.0	129350
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	13500
3	100006	0	Cash loans	F	N	Y	0	135000.0	31268
4	100007	0	Cash loans	M	N	Y	0	121500.0	51300
...	...	...	...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	N	0	157500.0	25470
307507	456252	0	Cash loans	F	N	Y	0	72000.0	26955
307508	456253	0	Cash loans	F	N	Y	0	153000.0	67766
307509	456254	1	Cash loans	F	N	Y	0	171000.0	37010
307510	456255	0	Cash loans	F	N	N	0	157500.0	67500

307511 rows × 32 columns

```
dfp=pd.read_csv(r'/content/drive/MyDrive/Vanshita/previous_application.csv') #load previous application data into dataframe
```

```
dfp #Display previous application data into dataframe
```

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	WE
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	0.0	17145.0	
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	NaN	607500.0	
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	NaN	112500.0	
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	NaN	450000.0	
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	NaN	337500.0	
...	...	...	...	...	...	...	...	...	
1670209	2300464	352015	Consumer loans	14704.290	267295.5	311400.0	0.0	267295.5	
1670210	2357031	334635	Consumer loans	6622.020	87750.0	64291.5	29250.0	87750.0	
1670211	2659632	249544	Consumer loans	11520.855	105237.0	102523.5	10525.5	105237.0	
1670212	2785582	400317	Cash loans	18821.520	180000.0	191880.0	NaN	180000.0	
1670213	2418762	261212	Cash loans	16431.300	360000.0	360000.0	NaN	360000.0	

1670214 rows × 37 columns

```
dfp.info()  #Get information about columns,count of non-null values,memory usage,etc about previous application data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   SK_ID_PREV                   1670214 non-null  int64
 1   SK_ID_CURR                   1670214 non-null  int64
 2   NAME_CONTRACT_TYPE           1670214 non-null  object
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_DOWN_PAYMENT             774370 non-null   float64
 7   AMT_GOODS_PRICE              1284699 non-null  float64
```

```
 8   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object
 9   HOUR_APPR_PROCESS_START      1670214 non-null  int64
 10  FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object
 16  NAME_CONTRACT_STATUS         1670214 non-null  object
 17  DAYS_DECISION                1670214 non-null  int64
 18  NAME_PAYMENT_TYPE            1670214 non-null  object
 19  CODE_REJECT_REASON           1670214 non-null  object
 20  NAME_TYPE_SUITE              849809 non-null   object
 21  NAME_CLIENT_TYPE             1670214 non-null  object
 22  NAME_GOODS_CATEGORY          1670214 non-null  object
 23  NAME_PORTFOLIO               1670214 non-null  object
 24  NAME_PRODUCT_TYPE            1670214 non-null  object
 25  CHANNEL_TYPE                 1670214 non-null  object
 26  SELLERPLACE_AREA             1670214 non-null  int64
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object
 30  PRODUCT_COMBINATION          1669868 non-null  object
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
```

```
#Drop irrelevant columns from previous application data
columns_to_drop=[
 'AMT_DOWN_PAYMENT',
 'WEEKDAY_APPR_PROCESS_START',
 'HOUR_APPR_PROCESS_START',
 'FLAG_LAST_APPL_PER_CONTRACT',
 'NFLAG_LAST_APPL_IN_DAY',
 'RATE_DOWN_PAYMENT',
 'RATE_INTEREST_PRIMARY',
 'RATE_INTEREST_PRIVILEGED',
 'DAYS_DECISION',
 'NAME_TYPE_SUITE',
 'NAME_GOODS_CATEGORY',
 'NAME_PORTFOLIO',
 'NAME_PRODUCT_TYPE',
 'CHANNEL_TYPE',
 'SELLERPLACE_AREA',
 'NAME_SELLER_INDUSTRY',
 'CNT_PAYMENT',
 'PRODUCT_COMBINATION',
 'DAYS_FIRST_DRAWING',
 'DAYS_FIRST_DUE',
 'DAYS_LAST_DUE_1ST_VERSION',
 'DAYS_LAST_DUE',
 'DAYS_TERMINATION',
 'NFLAG_INSURED_ON_APPROVAL']

dfp=dfp.drop(columns_to_drop,axis=1)
```

```
dfp #Display previous application data after dropping irrelevant columns
```

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_GOODS_PRICE	NAME_CASH_LOAN_PURPC
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	17145.0	X
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	607500.0	XI
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	112500.0	XI
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	450000.0	XI
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	337500.0	Repa
...	...	...	...	...	...	...	...	
1670209	2300464	352015	Consumer loans	14704.290	267295.5	311400.0	267295.5	X
1670210	2357031	334635	Consumer loans	6622.020	87750.0	64291.5	87750.0	X
1670211	2659632	249544	Consumer loans	11520.855	105237.0	102523.5	105237.0	X
1670212	2785582	400317	Cash loans	18821.520	180000.0	191880.0	180000.0	XI
1670213	2418762	261212	Cash loans	16431.300	360000.0	360000.0	360000.0	XI

1670214 rows × 13 columns

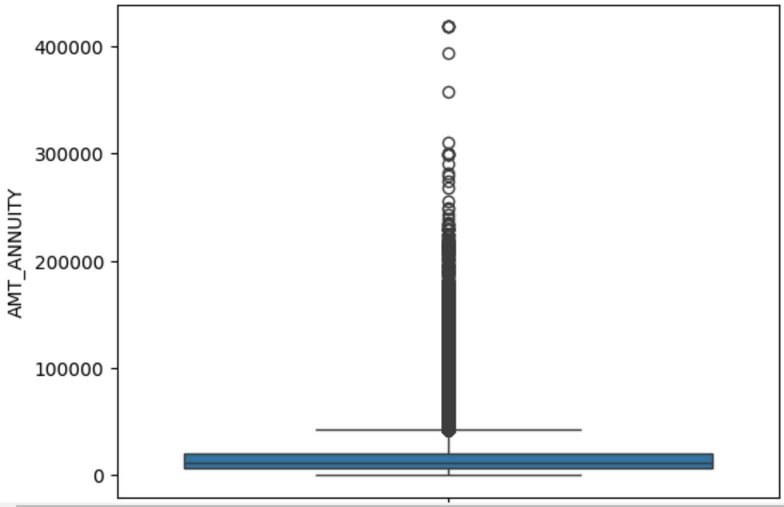
```
dfp.info() #Get information about number of columns,non-null values present in each column,memory usage of previous application data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 13 columns):
 #   Column                  Non-Null Count    Dtype  
---  ------                  --------------    -----  
 0   SK_ID_PREV              1670214 non-null  int64  
 1   SK_ID_CURR              1670214 non-null  int64  
 2   NAME_CONTRACT_TYPE      1670214 non-null  object 
 3   AMT_ANNUITY             1297979 non-null  float64
 4   AMT_APPLICATION         1670214 non-null  float64
 5   AMT_CREDIT              1670213 non-null  float64
 6   AMT_GOODS_PRICE         1284699 non-null  float64
 7   NAME_CASH_LOAN_PURPOSE  1670214 non-null  object 
 8   NAME_CONTRACT_STATUS    1670214 non-null  object 
 9   NAME_PAYMENT_TYPE       1670214 non-null  object 
 10  CODE_REJECT_REASON      1670214 non-null  object 
 11  NAME_CLIENT_TYPE        1670214 non-null  object 
 12  NAME_YIELD_GROUP        1670214 non-null  object 
dtypes: float64(4), int64(2), object(7)
memory usage: 165.7+ MB
```

Finding Null Values & Imputing them with certain calculated values

```
import seaborn as sns
sns.boxplot(y=dfp.AMT_ANNUITY)   # As the boxplot of AMT_ANNUITY column shows that there is presence of outliers so it is not symmetric.
```

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



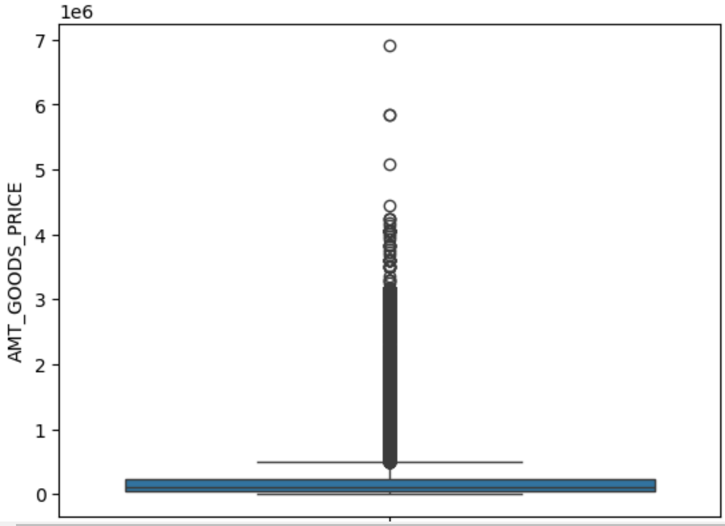
```
dfp['AMT_ANNUITY']=dfp['AMT_ANNUITY'].fillna(dfp.AMT_ANNUITY.median()) #We replace null values with median as graph representing values
```

```
dfp.AMT_ANNUITY.isnull().sum()
```

```
0
```

```
mport seaborn as sns
ns.boxplot(y=dfp.AMT_GOODS_PRICE) # As the boxplot of AMT_GOODS_PRICE column shows that there is presence of outliers so it is not symmet
```

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




```
AMT_GOODS_PRICE'].fillna(dfp.AMT_GOODS_PRICE.median()) #We replace null values with median as graph representing values is not symmetric.
```

```
dfp.AMT_GOODS_PRICE.isnull().sum()
```

```
0
```

```
dfp.info() #To verify if all null values are filled in previous application data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 13 columns):
 #   Column                  Non-Null Count    Dtype
---  ------                  --------------    -----
 0   SK_ID_PREV              1670214 non-null  int64
 1   SK_ID_CURR              1670214 non-null  int64
 2   NAME_CONTRACT_TYPE      1670214 non-null  object
 3   AMT_ANNUITY             1670214 non-null  float64
 4   AMT_APPLICATION         1670214 non-null  float64
 5   AMT_CREDIT              1670213 non-null  float64
 6   AMT_GOODS_PRICE         1670214 non-null  float64
 7   NAME_CASH_LOAN_PURPOSE  1670214 non-null  object
 8   NAME_CONTRACT_STATUS    1670214 non-null  object
 9   NAME_PAYMENT_TYPE       1670214 non-null  object
 10  CODE_REJECT_REASON      1670214 non-null  object
 11  NAME_CLIENT_TYPE        1670214 non-null  object
 12  NAME_YIELD_GROUP        1670214 non-null  object
dtypes: float64(4), int64(2), object(7)
memory usage: 165.7+ MB
```

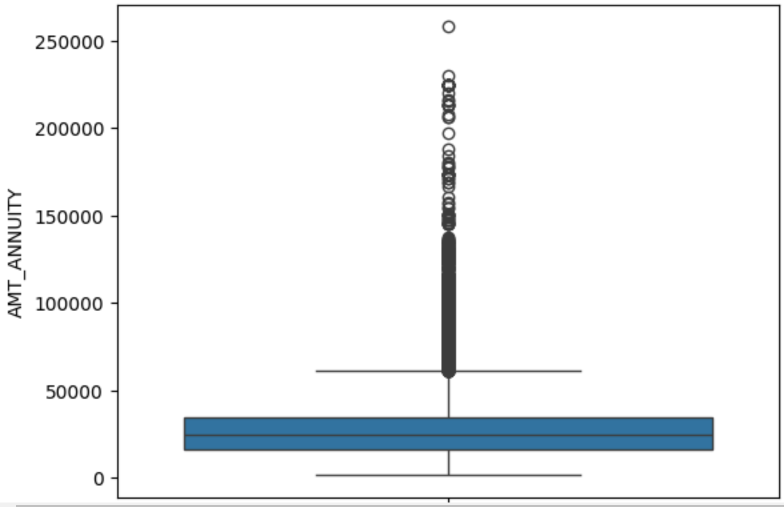
```
dfa.info() #To check columns having null values in application data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 32 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   SK_ID_CURR                    307511 non-null  int64
 1   TARGET                        307511 non-null  int64
 2   NAME_CONTRACT_TYPE            307511 non-null  object
 3   CODE_GENDER                   307511 non-null  object
 4   FLAG_OWN_CAR                  307511 non-null  object
 5   FLAG_OWN_REALTY               307511 non-null  object
 6   CNT_CHILDREN                  307511 non-null  int64
 7   AMT_INCOME_TOTAL              307511 non-null  float64
 8   AMT_CREDIT                    307511 non-null  float64
 9   AMT_ANNUITY                   307499 non-null  float64
 10  AMT_GOODS_PRICE               307233 non-null  float64
 11  NAME_TYPE_SUITE               306219 non-null  object
```

```
 12  NAME_INCOME_TYPE             307511 non-null  object
 13  NAME_EDUCATION_TYPE          307511 non-null  object
 14  NAME_FAMILY_STATUS           307511 non-null  object
 15  NAME_HOUSING_TYPE            307511 non-null  object
 16  REGION_POPULATION_RELATIVE   307511 non-null  float64
 17  OWN_CAR_AGE                  104582 non-null  float64
 18  OCCUPATION_TYPE              211120 non-null  object
 19  CNT_FAM_MEMBERS              307509 non-null  float64
 20  REGION_RATING_CLIENT         307511 non-null  int64
 21  REGION_RATING_CLIENT_W_CITY  307511 non-null  int64
 22  ORGANIZATION_TYPE            307511 non-null  object
 23  EXT_SOURCE_1                 134133 non-null  float64
 24  EXT_SOURCE_2                 306851 non-null  float64
 25  EXT_SOURCE_3                 246546 non-null  float64
 26  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64
 27  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64
 28  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64
 29  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64
 30  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64
 31  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64
dtypes: float64(16), int64(5), object(11)
memory usage: 75.1+ MB
```

```
import seaborn as sns
sns.boxplot(y=dfa.AMT_ANNUITY) # As the boxplot of AMT_ANNUITY column shows that there is presence of outliers so it is not symmetric.
```

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



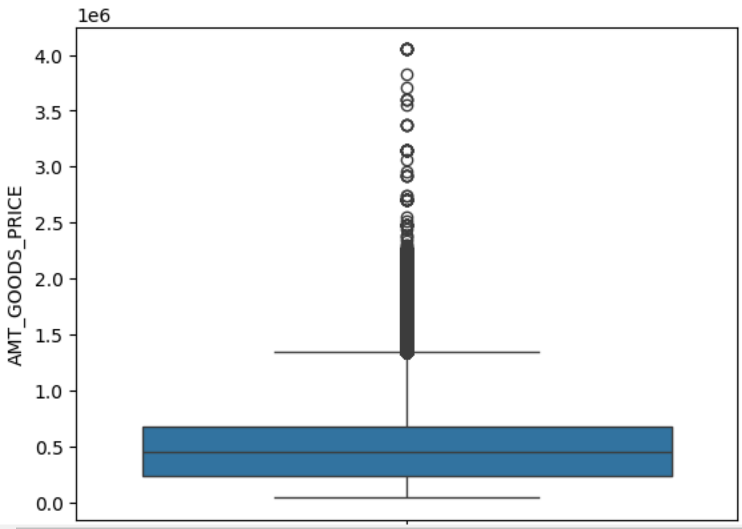

```
dfa['AMT_ANNUITY']=dfa['AMT_ANNUITY'].fillna(dfa['AMT_ANNUITY'].median()) #We replace null values with median as graph representing valu
```

```
dfa['AMT_ANNUITY'].isnull().sum()
```

```
0
```

```
import seaborn as sns
sns.boxplot(y=dfa.AMT_GOODS_PRICE) # As the boxplot of AMT_GOODS_PRICE column shows that there is presence of outliers so it is not symm
```

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



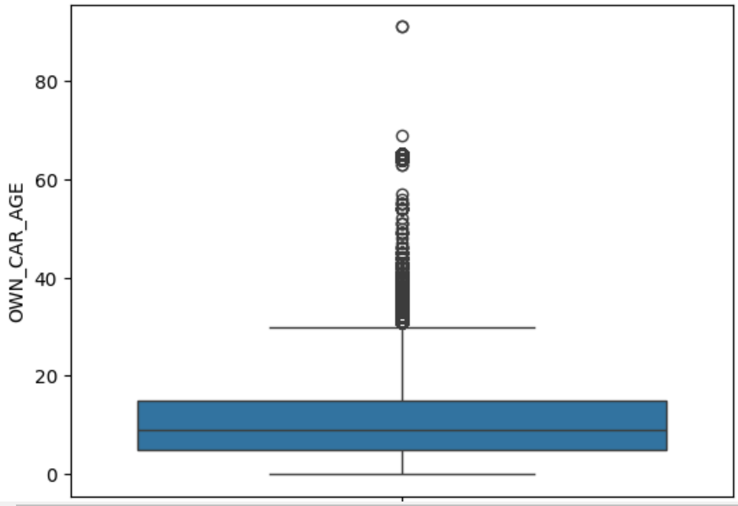
```
dfa['AMT_GOODS_PRICE']=dfa['AMT_GOODS_PRICE'].fillna(dfa.AMT_GOODS_PRICE.median()) #We replace null values with median as graph represen
```

```
dfa.AMT_GOODS_PRICE.isnull().sum()
```

```
0
```

```
import seaborn as sns
sns.boxplot(y=dfa.OWN_CAR_AGE) # As the boxplot of OWN_CAR_AGE column shows that there is presence of outliers so it is not symmetric.
```

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




```
dfa['OWN_CAR_AGE']=dfa['OWN_CAR_AGE'].fillna(dfa.OWN_CAR_AGE.median()) #We replace null values with median as graph representing values
```

```
dfa.OWN_CAR_AGE.isnull().sum()
```

```
0
```

```
dfa=dfa.drop('NAME_TYPE_SUITE',axis=1) #Dropping NAME_TYPE_SUITE column as it is not required for further analysis
```

```
dfa #Displaying application data after filling null values
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CRED
0	100002	1	Cash loans	M	N	Y	0	202500.0	40659
1	100003	0	Cash loans	F	N	N	0	270000.0	129350
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	13500
3	100006	0	Cash loans	F	N	Y	0	135000.0	31268
4	100007	0	Cash loans	M	N	Y	0	121500.0	51300
...	...	...	...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	N	0	157500.0	25470
307507	456252	0	Cash loans	F	N	Y	0	72000.0	26955
307508	456253	0	Cash loans	F	N	Y	0	153000.0	67766
307509	456254	1	Cash loans	F	N	Y	0	171000.0	37010
307510	456255	0	Cash loans	F	N	N	0	157500.0	67500

307511 rows × 31 columns

```
dfa.shape[0]-dfa.OCCUPATION_TYPE.count() #count number of null values present in dfa.OCCUPATION_TYPE column
```

```
96391
```

```
count_occu=dfa.OCCUPATION_TYPE.value_counts().reset_index() #getting frequency of OCCUPATION_TYPE
count_occu
```

	OCCUPATION_TYPE	count
0	Laborers	55186
1	Sales staff	32102
2	Core staff	27570
3	Managers	21371
4	Drivers	18603
5	High skill tech staff	11380
6	Accountants	9813
7	Medicine staff	8537
8	Security staff	6721
9	Cooking staff	5946
10	Cleaning staff	4653
11	Private service staff	2652
12	Low-skill Laborers	2093
13	Waiters/barmen staff	1348
14	Secretaries	1305
15	Realty agents	751
16	HR staff	563
17	IT staff	526

Next steps: View recommended plots New interactive sheet

```
count_occu['prop']=count_occu['count']/211120 #getting proportion of OCCUPATION_TYPE
count_occu
```

	OCCUPATION_TYPE	count	prop
0	Laborers	55186	0.261396
1	Sales staff	32102	0.152056
2	Core staff	27570	0.130589
3	Managers	21371	0.101227
4	Drivers	18603	0.088116
5	High skill tech staff	11380	0.053903
6	Accountants	9813	0.046481
7	Medicine staff	8537	0.040437
8	Security staff	6721	0.031835
9	Cooking staff	5946	0.028164
10	Cleaning staff	4653	0.022040
11	Private service staff	2652	0.012562
12	Low-skill Laborers	2093	0.009914
13	Waiters/barmen staff	1348	0.006385
14	Secretaries	1305	0.006181
15	Realty agents	751	0.003557
16	HR staff	563	0.002667
17	IT staff	526	0.002491

Next steps: View recommended plots New interactive sheet

```
count_occu['tofill']=round((count_occu['prop']*96391),0) #getting count of proportion of total null values to be filled by each OCCUPATI
count_occu
```

	OCCUPATION_TYPE	count	prop	tofill
0	Laborers	55186	0.261396	25196.0
1	Sales staff	32102	0.152056	14657.0
2	Core staff	27570	0.130589	12588.0
3	Managers	21371	0.101227	9757.0
4	Drivers	18603	0.088116	8494.0
5	High skill tech staff	11380	0.053903	5196.0
6	Accountants	9813	0.046481	4480.0
7	Medicine staff	8537	0.040437	3898.0
8	Security staff	6721	0.031835	3069.0
9	Cooking staff	5946	0.028164	2715.0
10	Cleaning staff	4653	0.022040	2124.0
11	Private service staff	2652	0.012562	1211.0
12	Low-skill Laborers	2093	0.009914	956.0
13	Waiters/barmen staff	1348	0.006385	615.0
14	Secretaries	1305	0.006181	596.0
15	Realty agents	751	0.003557	343.0
16	HR staff	563	0.002667	257.0
17	IT staff	526	0.002491	240.0

Next steps: View recommended plots New interactive sheet

```
import numpy as np
index_of_null=np.where(dfa.OCCUPATION_TYPE.isnull())[0] #getting indicies of rows having null values of OCCUPATION_TYPE column
index_of_null
```

```
array([     8,      11,      23, ..., 307500, 307505, 307507])
```

```
len(index_of_null)
```

```
96391
```

```
occupation_data = {
    'Laborers': 25196,
    'Sales staff': 14657,
    'Core staff': 12588,
    'Managers': 9757,
    'Drivers': 8494,
    'High skill tech staff': 5196,
    'Accountants': 4480,
    'Medicine staff': 3898,
    'Security staff': 3069,
    'Cooking staff': 2715,
    'Cleaning staff': 2124,
    'Private service staff': 1211,
    'Low-skill Laborers': 956,
    'Waiters/barmen staff': 615,
    'Secretaries': 596,
    'Realty agents': 342,
    'HR staff': 257,
    'IT staff': 240
}        #get the number of records to be filled for each OCCUPATION_TYPE in a dictionary
```

```
fill_values = []
for category, count in occupation_data.items():
    fill_values.extend([category] * int(count)) #create a new list containing these values of each OCCUPATION_TYPE as many times as ment
```

```
np.random.shuffle(fill_values) #shuffle this list to randomly give values
```

```
dfa.loc[index_of_null,'OCCUPATION_TYPE']=fill_values #update thenull value records with value in list
```

```
dfa.OCCUPATION_TYPE.isnull().sum() #to check if any null value is remaining to be filled
```

```
0
```

```
dfa.OCCUPATION_TYPE.value_counts() #to check new frequency of OCCUPATION_TYPE
```

OCCUPATION_TYPE	count
Laborers	80382
Sales staff	46759
Core staff	40158
Managers	31128
Drivers	27097
High skill tech staff	16576
Accountants	14293
Medicine staff	12435
Security staff	9790
Cooking staff	8661
Cleaning staff	6777
Private service staff	3863
Low-skill Laborers	3049
Waiters/barmen staff	1963
Secretaries	1901
Realty agents	1093
HR staff	820
IT staff	766

```
dfa.info() #to check if count of non-null values is increased for OCCUPATION_TYPE
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 31 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   SK_ID_CURR                   307511 non-null  int64
 1   TARGET                       307511 non-null  int64
 2   NAME_CONTRACT_TYPE           307511 non-null  object
 3   CODE_GENDER                  307511 non-null  object
 4   FLAG_OWN_CAR                 307511 non-null  object
 5   FLAG_OWN_REALTY              307511 non-null  object
 6   CNT_CHILDREN                 307511 non-null  int64
 7   AMT_INCOME_TOTAL             307511 non-null  float64
 8   AMT_CREDIT                   307511 non-null  float64
 9   AMT_ANNUITY                  307511 non-null  float64
 10  AMT_GOODS_PRICE              307511 non-null  float64
 11  NAME_INCOME_TYPE             307511 non-null  object
 12  NAME_EDUCATION_TYPE          307511 non-null  object
 13  NAME_FAMILY_STATUS           307511 non-null  object
 14  NAME_HOUSING_TYPE            307511 non-null  object
 15  REGION_POPULATION_RELATIVE   307511 non-null  float64
 16  OWN_CAR_AGE                  307511 non-null  float64
 17  OCCUPATION_TYPE              307511 non-null  object
 18  CNT_FAM_MEMBERS              307509 non-null  float64
 19  REGION_RATING_CLIENT         307511 non-null  int64
 20  REGION_RATING_CLIENT_W_CITY  307511 non-null  int64
 21  ORGANIZATION_TYPE            307511 non-null  object
 22  EXT_SOURCE_1                 134133 non-null  float64
 23  EXT_SOURCE_2                 306851 non-null  float64
 24  EXT_SOURCE_3                 246546 non-null  float64
 25  AMT_REQ_CREDIT_BUREAU_HOUR   265992 non-null  float64
 26  AMT_REQ_CREDIT_BUREAU_DAY    265992 non-null  float64
 27  AMT_REQ_CREDIT_BUREAU_WEEK   265992 non-null  float64
 28  AMT_REQ_CREDIT_BUREAU_MON    265992 non-null  float64
 29  AMT_REQ_CREDIT_BUREAU_QRT    265992 non-null  float64
 30  AMT_REQ_CREDIT_BUREAU_YEAR   265992 non-null  float64
dtypes: float64(16), int64(5), object(10)
memory usage: 72.7+ MB
```

```
dfa['EXT_SOURCE_1']=dfa['EXT_SOURCE_1'].fillna(dfa.EXT_SOURCE_1.median()) #We replace null values with median as graph representing valu
```

```
dfa['EXT_SOURCE_2']=dfa['EXT_SOURCE_2'].fillna(dfa.EXT_SOURCE_2.median()) #We replace null values with median as graph representing value
```

```
dfa['EXT_SOURCE_3']=dfa['EXT_SOURCE_3'].fillna(dfa.EXT_SOURCE_3.median()) #We replace null values with median as graph representing value
```

```
dfa['AMT_REQ_CREDIT_BUREAU_HOUR']=dfa['AMT_REQ_CREDIT_BUREAU_HOUR'].fillna(dfa.AMT_REQ_CREDIT_BUREAU_HOUR.median()) #We replace null valu
```

```
dfa['AMT_REQ_CREDIT_BUREAU_DAY']=dfa['AMT_REQ_CREDIT_BUREAU_DAY'].fillna(dfa.AMT_REQ_CREDIT_BUREAU_DAY.median()) #We replace null values
```

```
dfa['AMT_REQ_CREDIT_BUREAU_WEEK']=dfa['AMT_REQ_CREDIT_BUREAU_WEEK'].fillna(dfa.AMT_REQ_CREDIT_BUREAU_WEEK.median()) #We replace null valu
```

```
dfa['AMT_REQ_CREDIT_BUREAU_MON']=dfa['AMT_REQ_CREDIT_BUREAU_MON'].fillna(dfa.AMT_REQ_CREDIT_BUREAU_MON.median()) #We replace null values
```

```
dfa['AMT_REQ_CREDIT_BUREAU_QRT']=dfa['AMT_REQ_CREDIT_BUREAU_QRT'].fillna(dfa.AMT_REQ_CREDIT_BUREAU_QRT.median()) #We replace null values
```

```
dfa['AMT_REQ_CREDIT_BUREAU_YEAR']=dfa['AMT_REQ_CREDIT_BUREAU_YEAR'].fillna(dfa.AMT_REQ_CREDIT_BUREAU_YEAR.median()) #We replace null valu
```

```
dfa.info() #To verify if columns having null values are filled in application data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 31 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   SK_ID_CURR                   307511 non-null  int64  
 1   TARGET                       307511 non-null  int64  
 2   NAME_CONTRACT_TYPE           307511 non-null  object 
 3   CODE_GENDER                  307511 non-null  object 
 4   FLAG_OWN_CAR                 307511 non-null  object 
 5   FLAG_OWN_REALTY              307511 non-null  object 
 6   CNT_CHILDREN                 307511 non-null  int64  
 7   AMT_INCOME_TOTAL             307511 non-null  float64
 8   AMT_CREDIT                   307511 non-null  float64
 9   AMT_ANNUITY                  307511 non-null  float64
 10  AMT_GOODS_PRICE              307511 non-null  float64
 11  NAME_INCOME_TYPE             307511 non-null  object 
 12  NAME_EDUCATION_TYPE          307511 non-null  object 
 13  NAME_FAMILY_STATUS           307511 non-null  object 
 14  NAME_HOUSING_TYPE            307511 non-null  object 
 15  REGION_POPULATION_RELATIVE   307511 non-null  float64
 16  OWN_CAR_AGE                  307511 non-null  float64
 17  OCCUPATION_TYPE              307511 non-null  object 
 18  CNT_FAM_MEMBERS              307509 non-null  float64
 19  REGION_RATING_CLIENT         307511 non-null  int64  
 20  REGION_RATING_CLIENT_W_CITY  307511 non-null  int64  
 21  ORGANIZATION_TYPE            307511 non-null  object 
 22  EXT_SOURCE_1                 307511 non-null  float64
 23  EXT_SOURCE_2                 307511 non-null  float64
 24  EXT_SOURCE_3                 307511 non-null  float64
 25  AMT_REQ_CREDIT_BUREAU_HOUR   307511 non-null  float64
 26  AMT_REQ_CREDIT_BUREAU_DAY    307511 non-null  float64
 27  AMT_REQ_CREDIT_BUREAU_WEEK   307511 non-null  float64
 28  AMT_REQ_CREDIT_BUREAU_MON    307511 non-null  float64
 29  AMT_REQ_CREDIT_BUREAU_QRT    307511 non-null  float64
 30  AMT_REQ_CREDIT_BUREAU_YEAR   307511 non-null  float64
dtypes: float64(16), int64(5), object(10)
memory usage: 72.7+ MB
```

```
merge_df=pd.merge(dfa,dfp,on='SK_ID_CURR',how='inner') #merge both dataframes to get all clients having records of their previous loan/(
```

```
merge_df.shape #get number of rows & number of columns
```

```
(1413701, 43)
```

```
merge_df #Display merged dataframe
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_C
0	100002	1	Cash loans	M	N	Y	0	202500.0	4
1	100003	0	Cash loans	F	N	N	0	270000.0	12
2	100003	0	Cash loans	F	N	N	0	270000.0	12
3	100003	0	Cash loans	F	N	N	0	270000.0	12
4	100004	0	Revolving loans	M	Y	Y	0	67500.0	1
...	...	...	...	...	...	...	...	...	
1413696	456255	0	Cash loans	F	N	N	0	157500.0	6
1413697	456255	0	Cash loans	F	N	N	0	157500.0	6
1413698	456255	0	Cash loans	F	N	N	0	157500.0	6
1413699	456255	0	Cash loans	F	N	N	0	157500.0	6
1413700	456255	0	Cash loans	F	N	N	0	157500.0	6

1413701 rows × 43 columns

```
merge_df.TARGET.value_counts() #get frequency of target column
```

	count
TARGET	
0	1291341
1	122360

Removing Outliers from Merged DataFrame

```
numerical_column = merge_df.select_dtypes(include=[np.number]).columns #Identify numerical columns

upper_fence = {} #store upper fence for each column
for column in numerical_column:
  if column!='TARGET':
       Q1 = merge_df[column].quantile(0.25)  # First quartile (25th percentile)
       Q3 = merge_df[column].quantile(0.75)  # Third quartile (75th percentile)
       IQR = Q3 - Q1  # Interquartile range
       upper_fence[column] = Q3 + 1.5 * IQR    # Calculate upper fence for each numerical column
  else:
      continue

print("Upper fences for numerical columns:")
print(upper_fence) # Print the upper fences for reference

for column, fence in upper_fence.items():
    merge_df = merge_df[merge_df[column] <= fence]  # Remove rows where any numerical column exceeds its upper fence

print("\nFiltered DataFrame:")
print(merge_df) # Display the filtered DataFrame
```

```
1413688                       1.0     1503599       Consumer loans
1413691                       0.0     2016407       Consumer loans
1413692                       0.0     1792910       Consumer loans

         AMT_ANNUITY_y  AMT_APPLICATION  AMT_CREDIT_y  AMT_GOODS_PRICE_y  \
0             9251.775         179055.0      179055.0           179055.0
3             6737.310          68809.5       68053.5            68809.5
6            11250.000              0.0           0.0           112320.0
7            29027.520         334917.0      267930.0           334917.0
8            13500.000         270000.0      270000.0           270000.0
...                ...              ...           ...                ...
1413682       8417.340          39960.0       41940.0            39960.0
1413687       6605.910          40455.0       40455.0            40455.0
1413688      10074.465          57595.5       56821.5            57595.5
1413691      19065.825         223789.5      247423.5           223789.5
1413692       2296.440          18846.0       21456.0            18846.0

        NAME_CASH_LOAN_PURPOSE NAME_CONTRACT_STATUS      NAME_PAYMENT_TYPE  \
0                          XAP             Approved                    XNA
3                          XAP             Approved  Cash through the bank
6                          XAP             Canceled                    XNA
7                          XAP             Approved  Cash through the bank
8                          XAP             Approved                    XNA
...                        ...                  ...                    ...
1413682                    XAP             Approved  Cash through the bank
1413687                    XAP             Approved  Cash through the bank
1413688                    XAP             Approved  Cash through the bank
1413691                    XAP             Approved  Cash through the bank
1413692                    XAP             Approved  Cash through the bank

        CODE_REJECT_REASON NAME_CLIENT_TYPE NAME_YIELD_GROUP
0                      XAP              New       low_normal
3                      XAP        Refreshed           middle
6                      XAP         Repeater              XNA
7                      XAP         Repeater             high
8                      XAP         Repeater              XNA
...                    ...              ...              ...
1413682                XAP              New             high
1413687                XAP              New             high
1413688                XAP              New       low_normal
1413691                XAP         Repeater       low_normal
1413692                XAP              New             high

[378584 rows x 43 columns]
```

```
merge_df.TARGET.value_counts() #to get frequency of target column after removing outliers from all columns of merged dataframe
```

	count
TARGET	
0	342437
1	36147

Final DataFrame after dropping irrelevant columns,removing outliers & imputing null values

```
merge_df.info() #Get final dataframe after merging & removing outliers
```

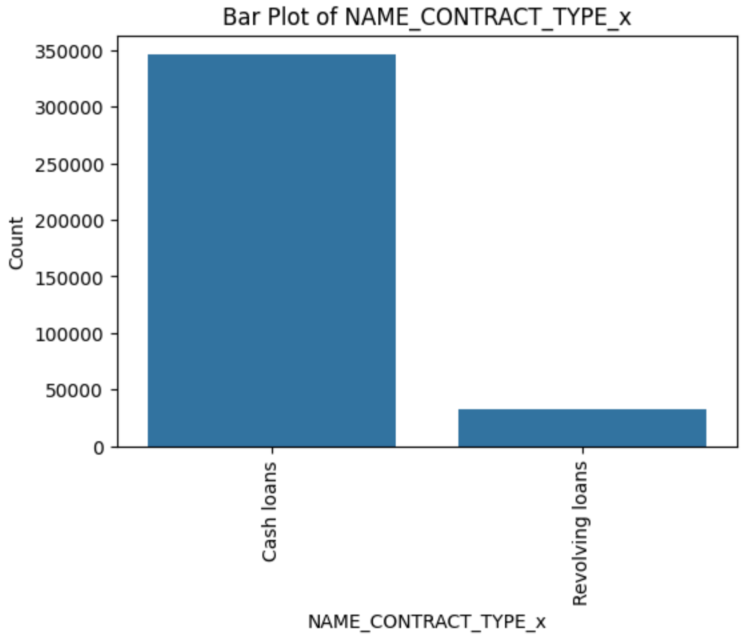
```
<class 'pandas.core.frame.DataFrame'>
Index: 378584 entries, 0 to 1413692
Data columns (total 43 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   SK_ID_CURR                  378584 non-null  int64
 1   TARGET                      378584 non-null  int64
 2   NAME_CONTRACT_TYPE_x        378584 non-null  object
 3   CODE_GENDER                 378584 non-null  object
 4   FLAG_OWN_CAR                378584 non-null  object
 5   FLAG_OWN_REALTY             378584 non-null  object
 6   CNT_CHILDREN                378584 non-null  int64
 7   AMT_INCOME_TOTAL            378584 non-null  float64
 8   AMT_CREDIT_x                378584 non-null  float64
 9   AMT_ANNUITY_x               378584 non-null  float64
 10  AMT_GOODS_PRICE_x           378584 non-null  float64
 11  NAME_INCOME_TYPE            378584 non-null  object
 12  NAME_EDUCATION_TYPE         378584 non-null  object
 13  NAME_FAMILY_STATUS          378584 non-null  object
 14  NAME_HOUSING_TYPE           378584 non-null  object
 15  REGION_POPULATION_RELATIVE  378584 non-null  float64
 16  OWN_CAR_AGE                 378584 non-null  float64
 17  OCCUPATION_TYPE             378584 non-null  object
 18  CNT_FAM_MEMBERS             378584 non-null  float64
```

```
 19  REGION_RATING_CLIENT         378584 non-null  int64
 20  REGION_RATING_CLIENT_W_CITY  378584 non-null  int64
 21  ORGANIZATION_TYPE            378584 non-null  object
 22  EXT_SOURCE_1                 378584 non-null  float64
 23  EXT_SOURCE_2                 378584 non-null  float64
 24  EXT_SOURCE_3                 378584 non-null  float64
 25  AMT_REQ_CREDIT_BUREAU_HOUR   378584 non-null  float64
 26  AMT_REQ_CREDIT_BUREAU_DAY    378584 non-null  float64
 27  AMT_REQ_CREDIT_BUREAU_WEEK   378584 non-null  float64
 28  AMT_REQ_CREDIT_BUREAU_MON    378584 non-null  float64
 29  AMT_REQ_CREDIT_BUREAU_QRT    378584 non-null  float64
 30  AMT_REQ_CREDIT_BUREAU_YEAR   378584 non-null  float64
 31  SK_ID_PREV                   378584 non-null  int64
 32  NAME_CONTRACT_TYPE_y         378584 non-null  object
 33  AMT_ANNUITY_y                378584 non-null  float64
 34  AMT_APPLICATION              378584 non-null  float64
 35  AMT_CREDIT_y                 378584 non-null  float64
 36  AMT_GOODS_PRICE_y            378584 non-null  float64
 37  NAME_CASH_LOAN_PURPOSE       378584 non-null  object
 38  NAME_CONTRACT_STATUS         378584 non-null  object
 39  NAME_PAYMENT_TYPE            378584 non-null  object
 40  CODE_REJECT_REASON           378584 non-null  object
 41  NAME_CLIENT_TYPE             378584 non-null  object
 42  NAME_YIELD_GROUP             378584 non-null  object
dtypes: float64(20), int64(6), object(17)
memory usage: 127.1+ MB
```

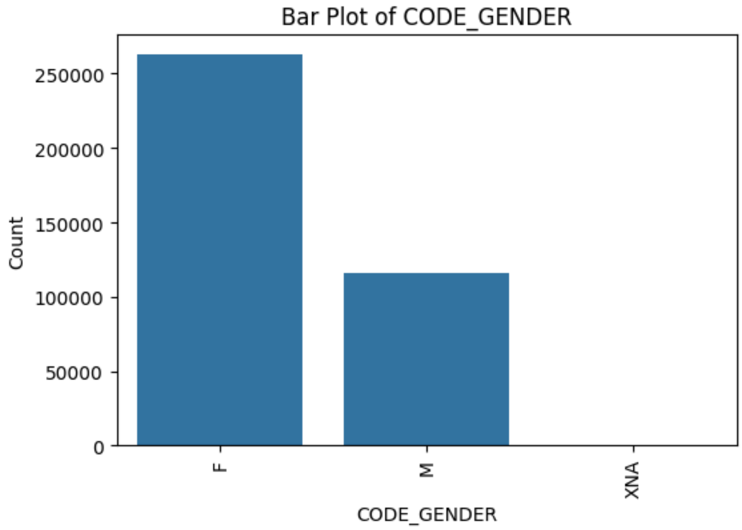
Univariate Analysis

```
import matplotlib.pyplot as plt
list_no_graph=['SK_ID_CURR','TARGET','REGION_RATING_CLIENT_W_CITY','AMT_REQ_CREDIT_BUREAU_HOUR','AMT_REQ_CREDIT_BUREAU_DAY','AMT_REQ_CRE
# Loop through each column
for column in merge_df.columns:
    if column not in list_no_graph:
    # Check the data type of the column
      if merge_df[column].dtype == 'object':  # Categorical column
        print(f"Analysis for {column}:")
        # Bar plot for categorical data using seaborn
        plt.figure(figsize=(6, 4))
        sns.barplot(x=merge_df[column].value_counts().index, y=merge_df[column].value_counts().values)
        plt.xticks(rotation=90)
        plt.title(f"Bar Plot of {column}")
        plt.xlabel(column)
        plt.ylabel('Count')
        plt.show()

      else:  # Numerical column
         print(f"Analysis for {column}:")
         print("Summary statistics:")
         print(merge_df[column].describe())  # Summary statistics (mean, std, min, etc.)
         print()

        # Histogram with KDE (Kernel Density Estimate) for numerical data
         plt.figure(figsize=(6, 4))
         sns.histplot(merge_df[column], kde=True)
         plt.title(f"Distribution of {column}")
         plt.show()
```

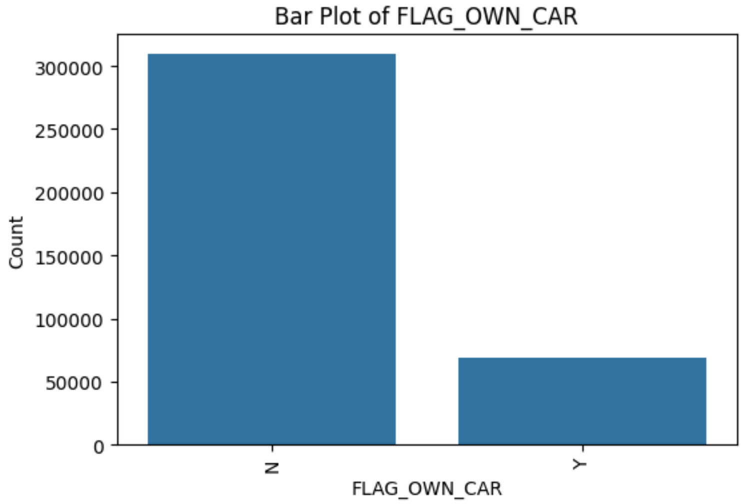
Analysis for NAME_CONTRACT_TYPE_x:



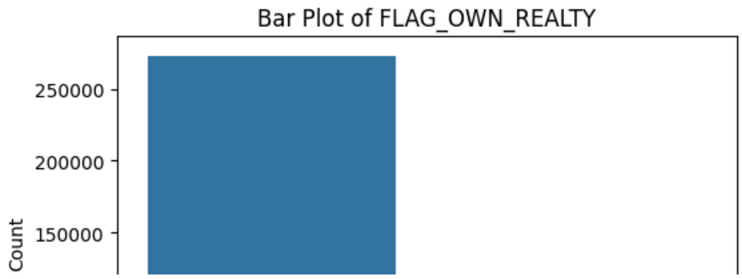
Analysis for CODE_GENDER:

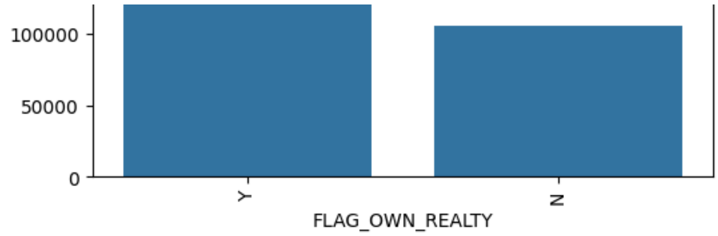


Analysis for FLAG_OWN_CAR:

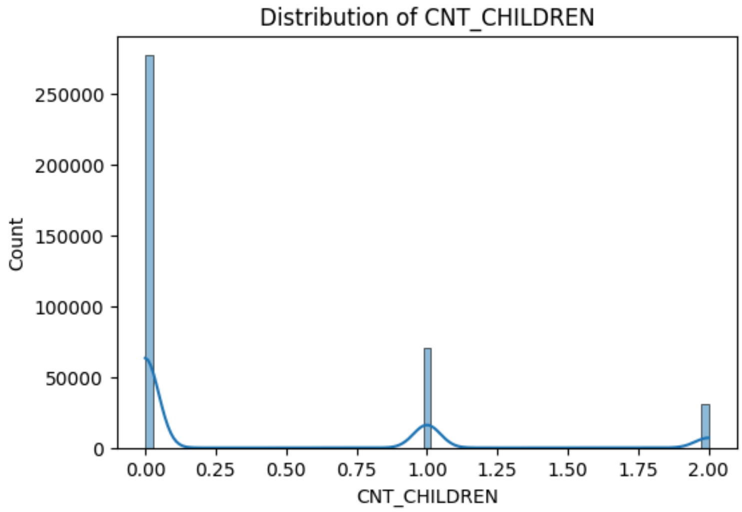


Analysis for FLAG_OWN_REALTY:

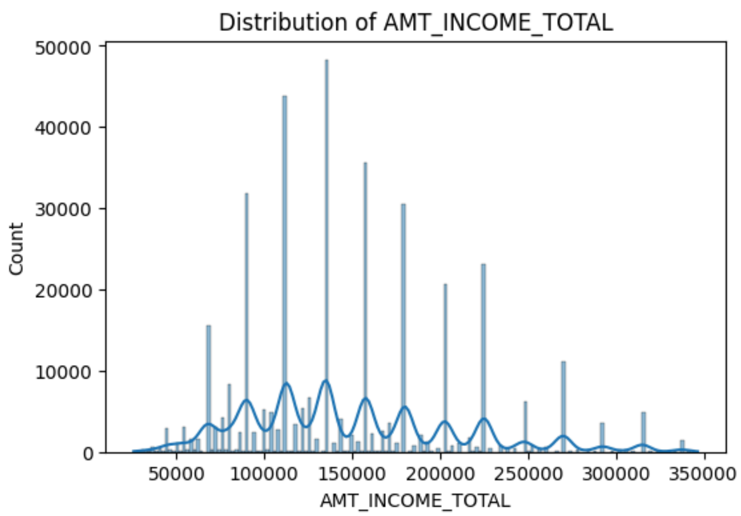




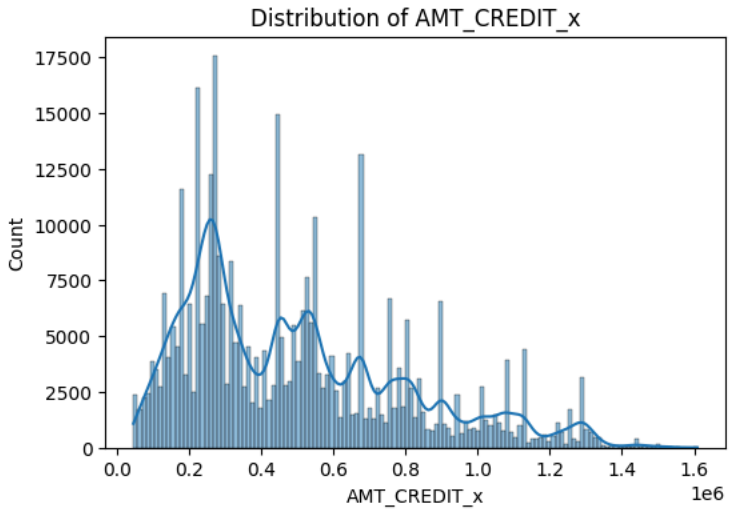

```
Analysis for CNT_CHILDREN:
Summary statistics:
count    378584.000000
mean          0.348684
std           0.624183
min           0.000000
25%           0.000000
50%           0.000000
75%           1.000000
max           2.000000
Name: CNT_CHILDREN, dtype: float64
```



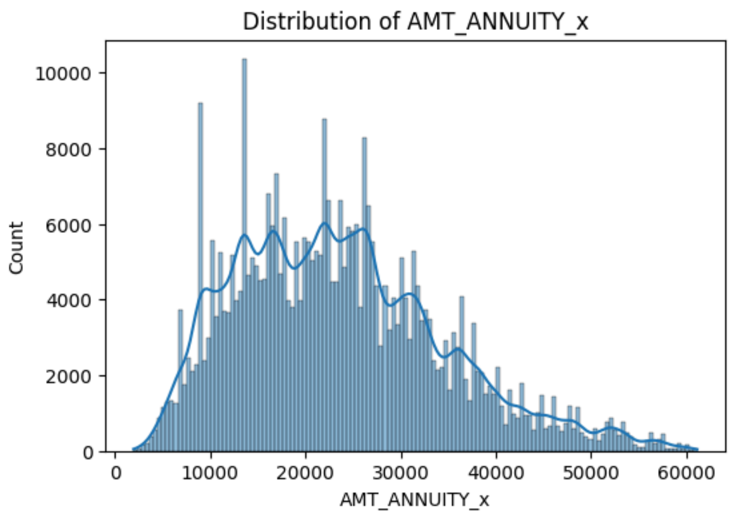

```
Analysis for AMT_INCOME_TOTAL:
Summary statistics:
count    378584.000000
mean     148248.575359
std       60613.424017
min       25650.000000
25%      108000.000000
50%      135000.000000
75%      180000.000000
max      346500.000000
Name: AMT_INCOME_TOTAL, dtype: float64
```



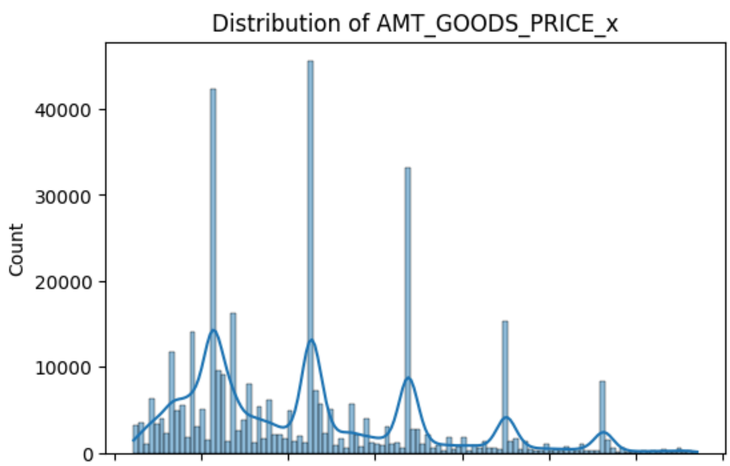

```
Analysis for AMT_CREDIT_x:
Summary statistics:
count    3.785840e+05
mean     4.952886e+05
std      3.055971e+05
min      4.500000e+04
25%      2.547000e+05
50%      4.500000e+05
75%      6.750000e+05
max      1.609272e+06
Name: AMT_CREDIT_x, dtype: float64
```

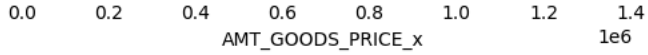



```
Analysis for AMT_ANNUITY_x:
Summary statistics:
count    378584.000000
mean      23616.284858
std       10947.905364
min        1993.500000
25%       15115.500000
50%       22455.000000
75%       30393.000000
max       61123.500000
Name: AMT_ANNUITY_x, dtype: float64
```

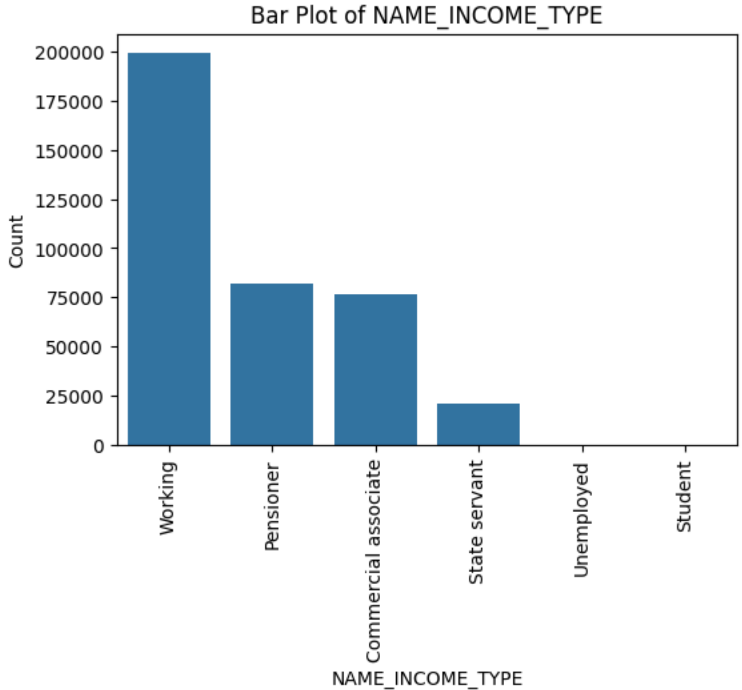



```
Analysis for AMT_GOODS_PRICE_x:
Summary statistics:
count    3.785840e+05
mean     4.424000e+05
std      2.743457e+05
min      4.500000e+04
25%      2.250000e+05
50%      4.050000e+05
75%      6.525000e+05
max      1.341000e+06
Name: AMT_GOODS_PRICE_x, dtype: float64
```

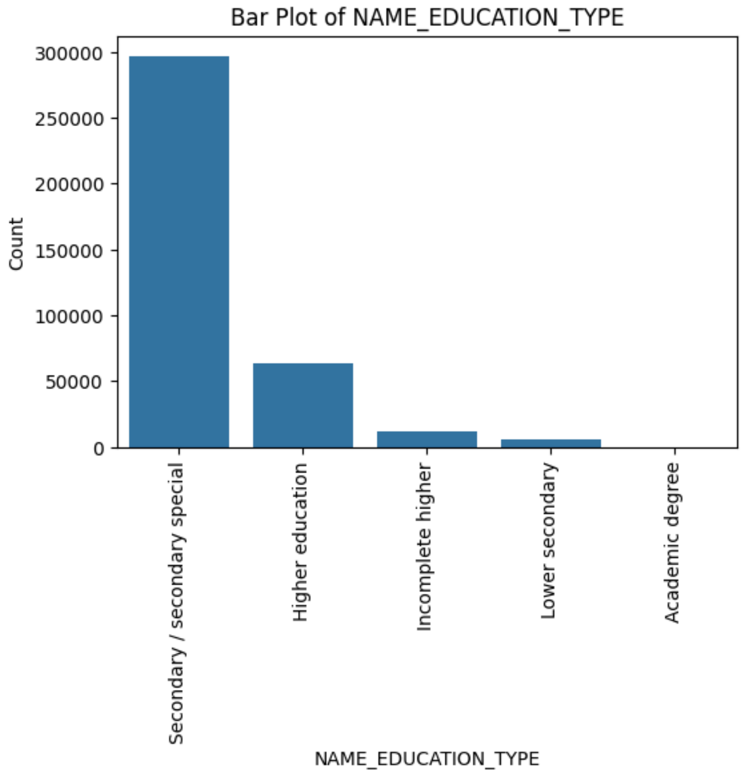




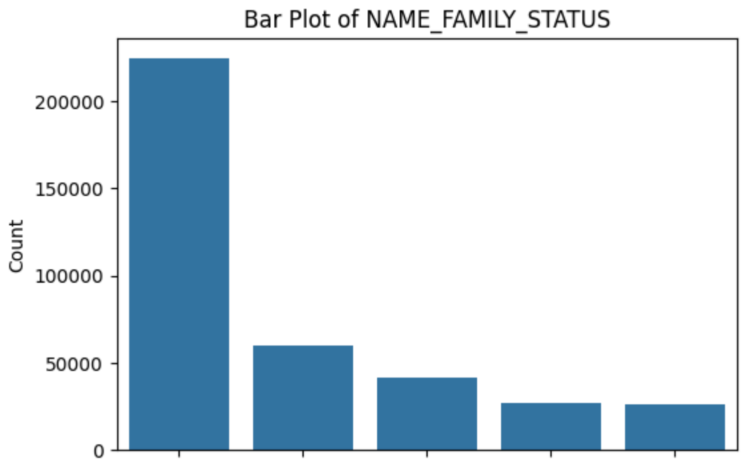

```
Analysis for NAME_INCOME_TYPE:
```



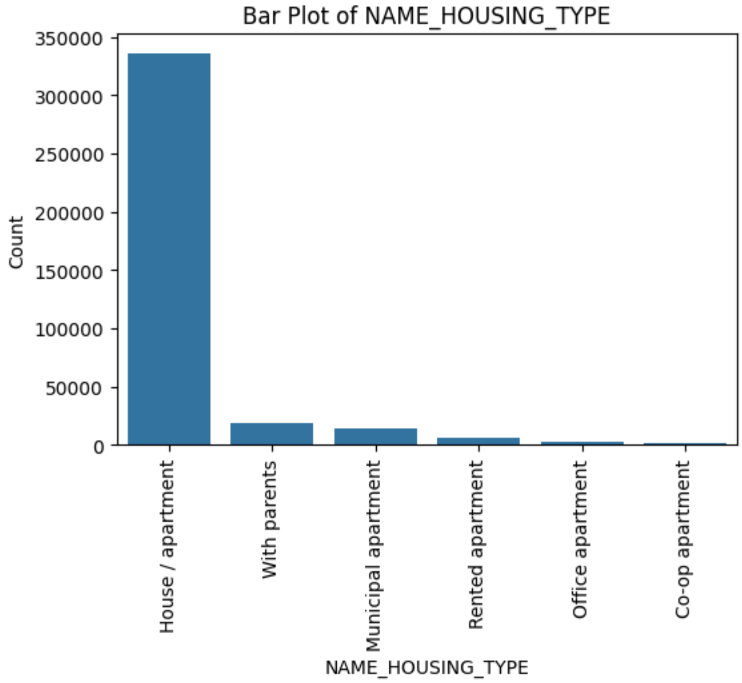

```
Analysis for NAME_EDUCATION_TYPE:
```



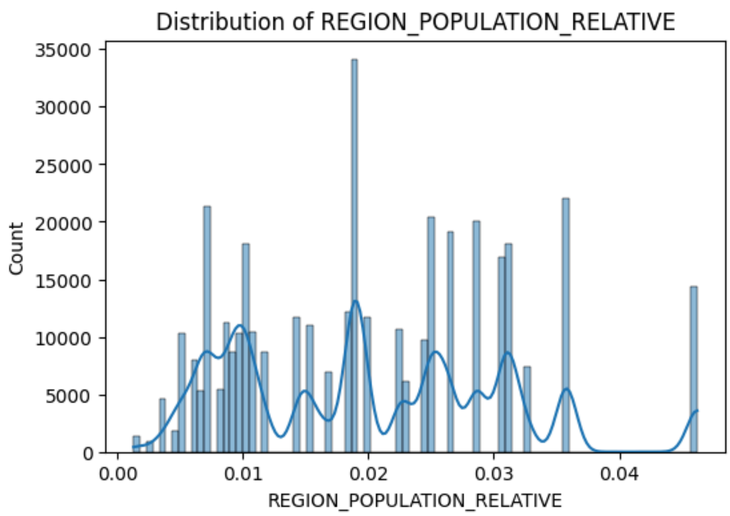

```
Analysis for NAME_FAMILY_STATUS:
```



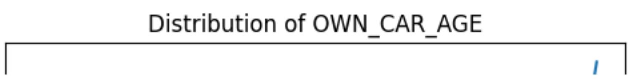

```
Analysis for NAME_HOUSING_TYPE:
```

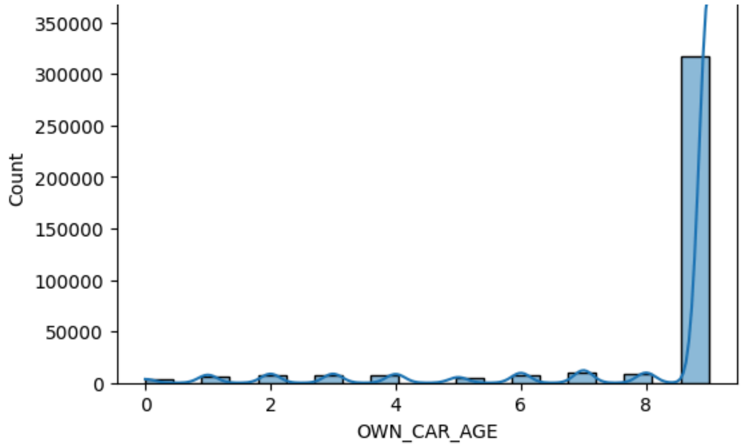



```
Analysis for REGION_POPULATION_RELATIVE:
Summary statistics:
count    378584.000000
mean          0.020029
std           0.010648
min           0.001276
25%           0.010032
50%           0.019101
75%           0.028663
max           0.046220
Name: REGION_POPULATION_RELATIVE, dtype: float64
```

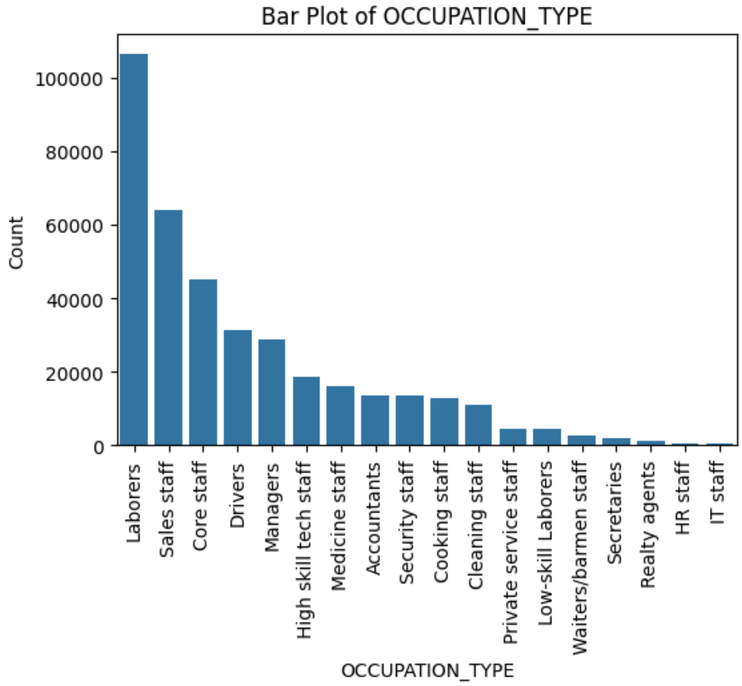



```
Analysis for OWN_CAR_AGE:
Summary statistics:
count    378584.000000
mean          8.261696
std           1.950995
min           0.000000
25%           9.000000
50%           9.000000
75%           9.000000
max           9.000000
Name: OWN_CAR_AGE, dtype: float64
```

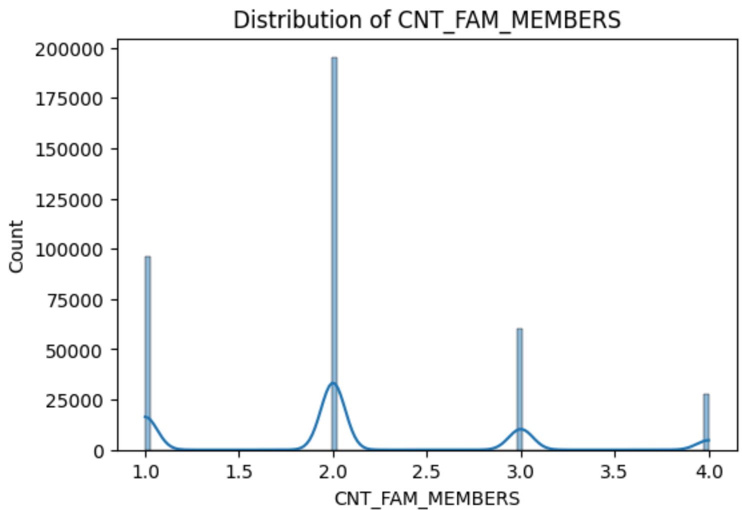





```
Analysis for OCCUPATION_TYPE:
```

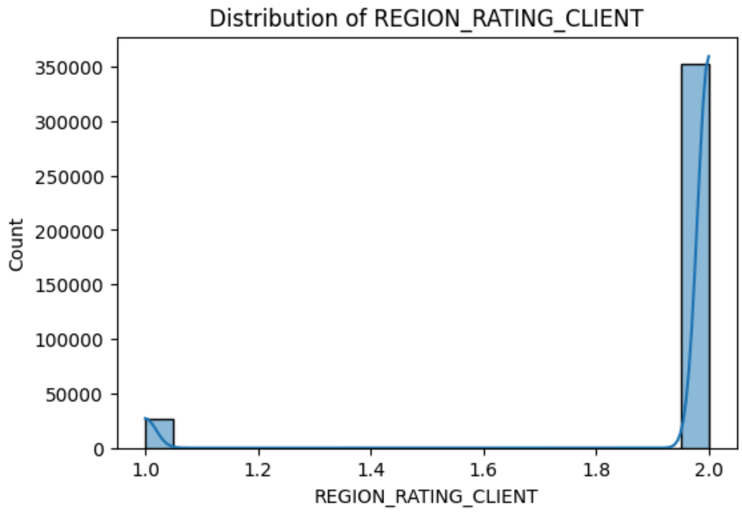



```
Analysis for CNT_FAM_MEMBERS:
Summary statistics:
count    378584.000000
mean          2.051893
std           0.837677
min           1.000000
25%           1.000000
50%           2.000000
75%           2.000000
max           4.000000
Name: CNT_FAM_MEMBERS, dtype: float64
```

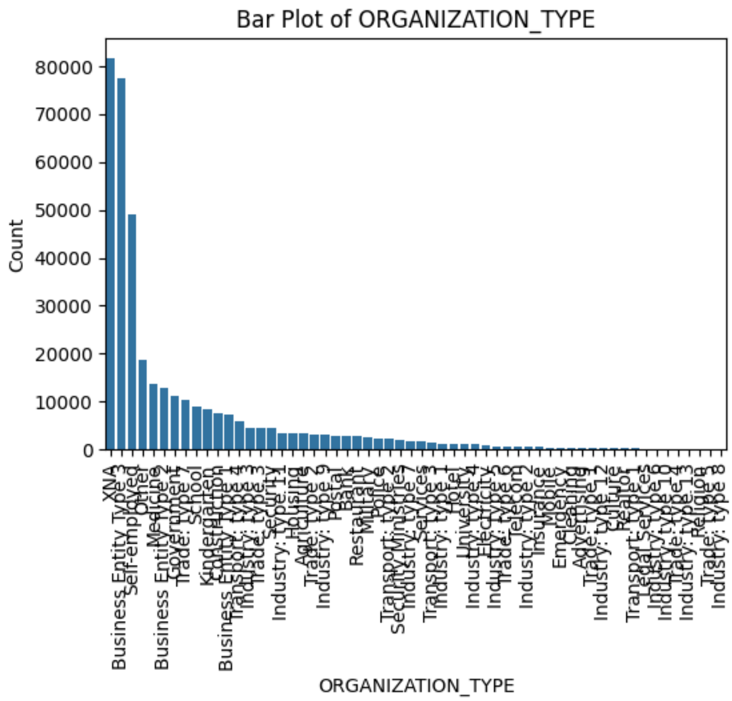



```
Analysis for REGION_RATING_CLIENT:
Summary statistics:
count    378584.000000
mean          1.929992
std           0.255161
```

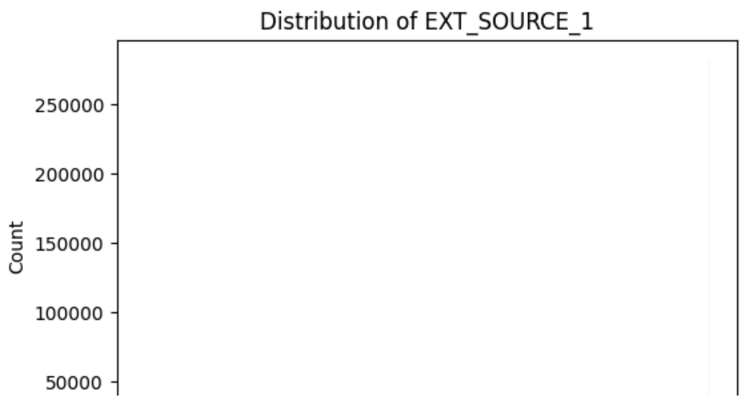
```
min          1.000000
25%          2.000000
50%          2.000000
75%          2.000000
max          2.000000
Name: REGION_RATING_CLIENT, dtype: float64
```

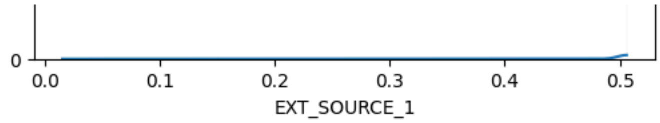



```
Analysis for ORGANIZATION_TYPE:
```

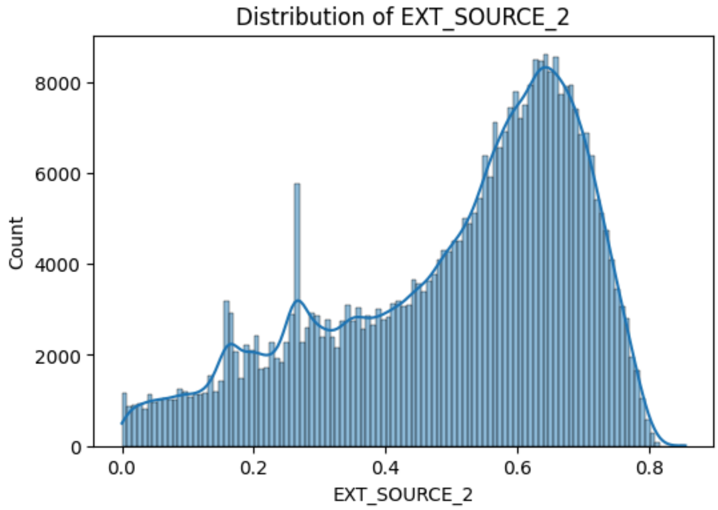



```
Analysis for EXT_SOURCE_1:
Summary statistics:
count    378584.000000
mean          0.459480
std           0.098710
min           0.015600
25%           0.499316
50%           0.505998
75%           0.505998
max           0.505998
Name: EXT_SOURCE_1, dtype: float64
```

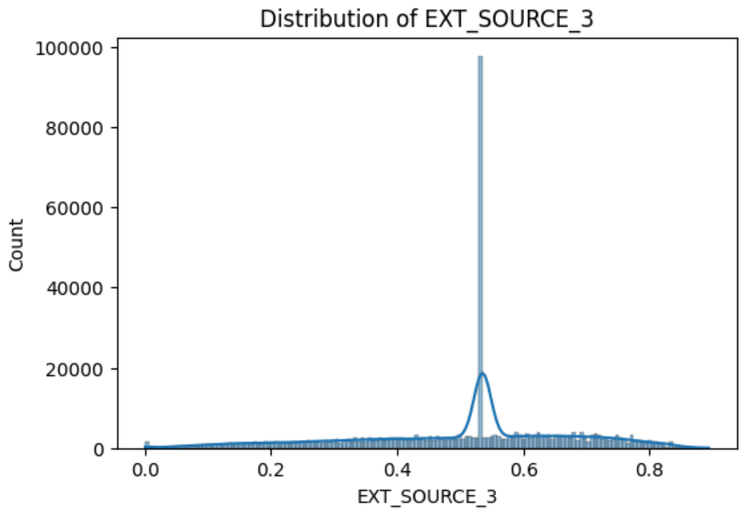




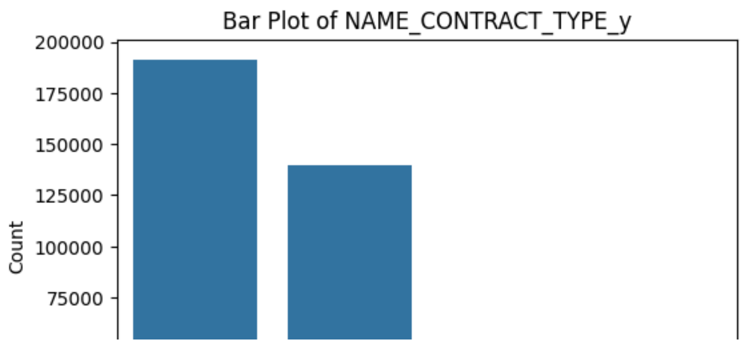

```
Analysis for EXT_SOURCE_2:
Summary statistics:
count    3.785840e+05
mean     5.046930e-01
std      1.904159e-01
min      8.173617e-08
25%      3.736136e-01
50%      5.582592e-01
75%      6.540971e-01
max      8.549997e-01
Name: EXT_SOURCE_2, dtype: float64
```

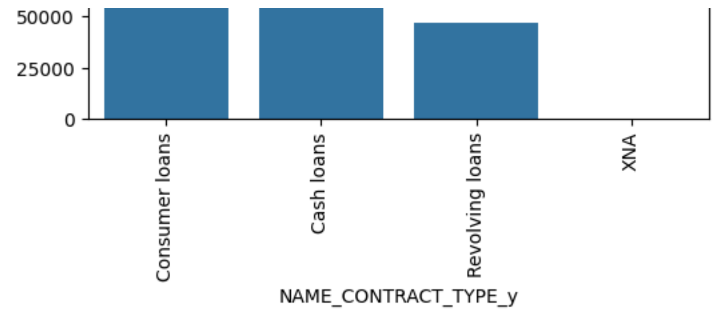



```
Analysis for EXT_SOURCE_3:
Summary statistics:
count    378584.000000
mean          0.505385
std           0.174863
min           0.000527
25%           0.408359
50%           0.535276
75%           0.614414
max           0.893976
Name: EXT_SOURCE_3, dtype: float64
```

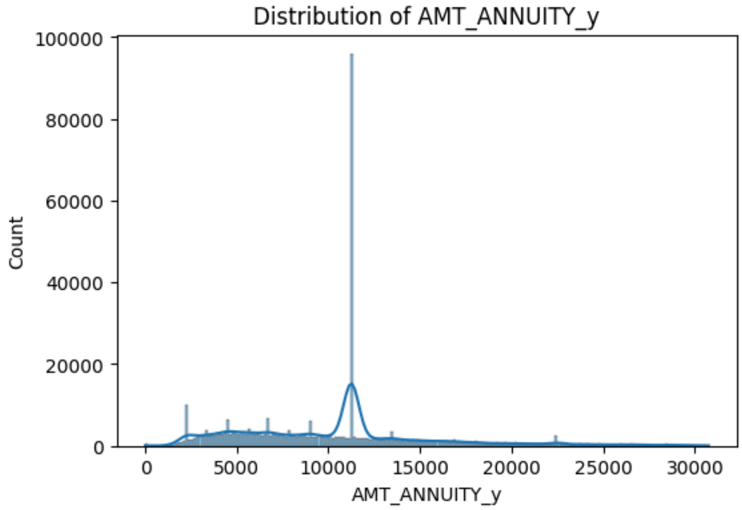



```
Analysis for NAME_CONTRACT_TYPE_y:
```

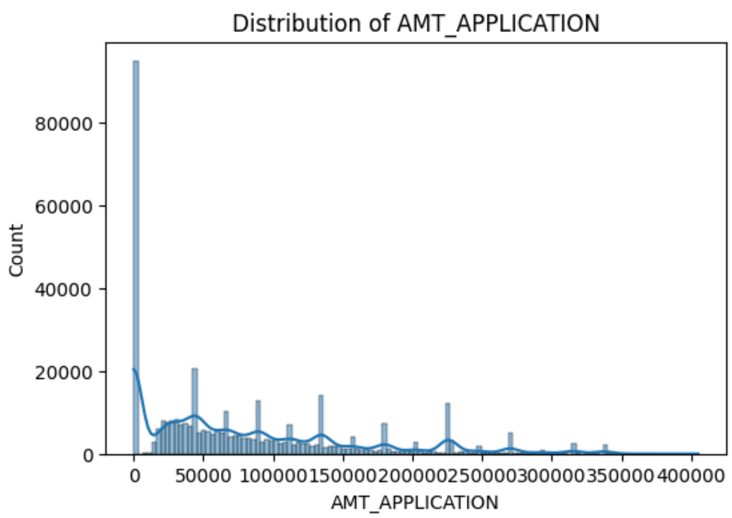





```
Analysis for AMT_ANNUITY_y:
Summary statistics:
count    378584.000000
mean      10385.661505
std        5419.469488
min           0.000000
25%        6325.335000
50%       11250.000000
75%       11674.530000
max       30760.650000
Name: AMT_ANNUITY_y, dtype: float64
```

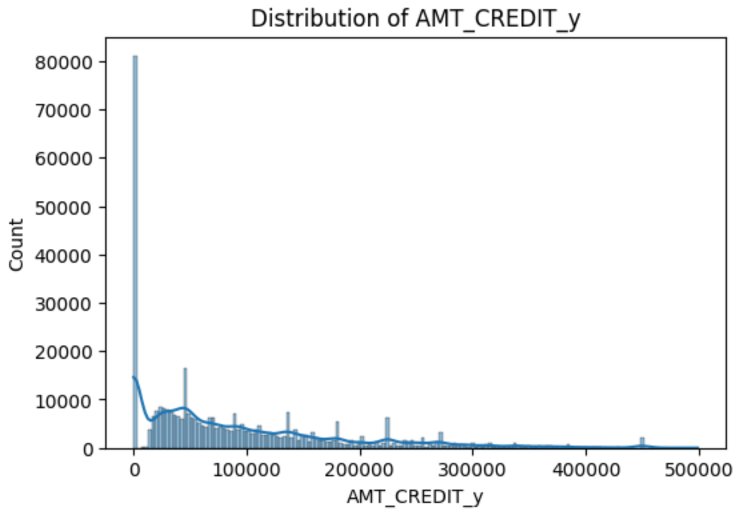



```
Analysis for AMT_APPLICATION:
Summary statistics:
count    378584.00000
mean      77146.12950
std       78668.90396
min           0.00000
25%           0.00000
50%       52605.00000
75%      117562.50000
max      405000.00000
Name: AMT_APPLICATION, dtype: float64
```

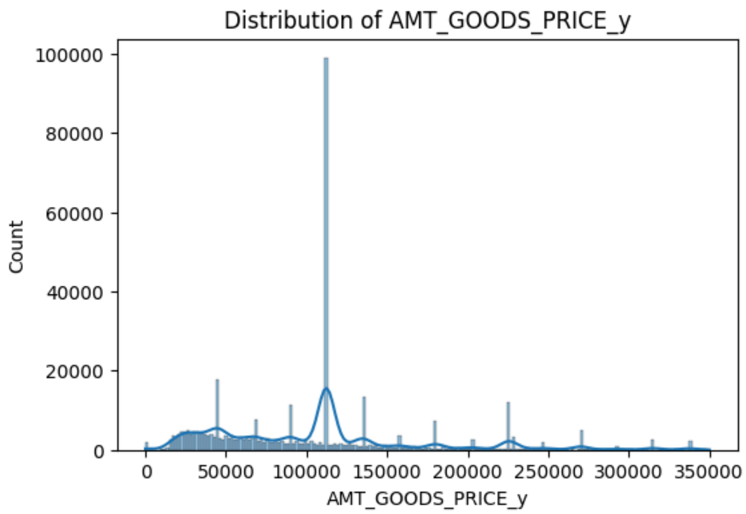



```
Analysis for AMT_CREDIT_y:
Summary statistics:
count    378584.000000
mean      88222.075376
std       91778.667008
min           0.000000
25%       20070.000000
50%       58450.500000
```

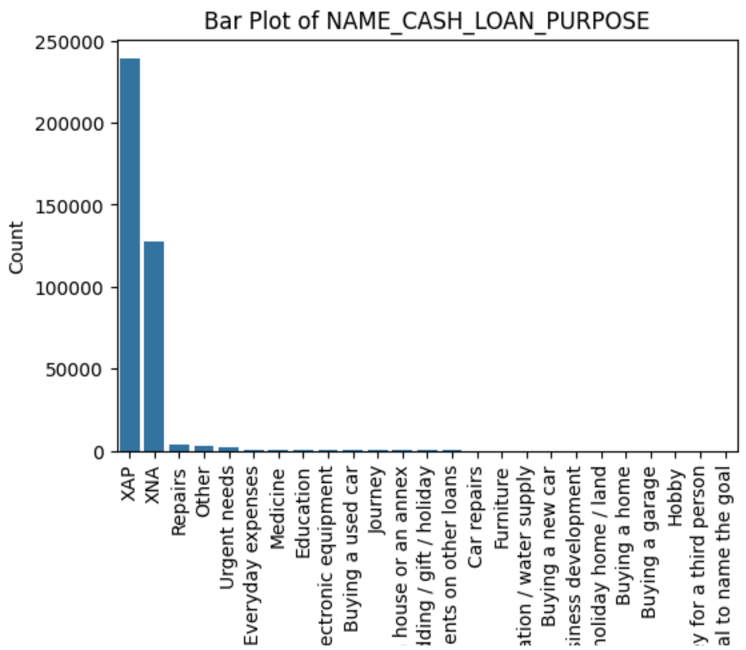
```
75%      134316.000000
max      499099.500000
Name: AMT_CREDIT_y, dtype: float64
```

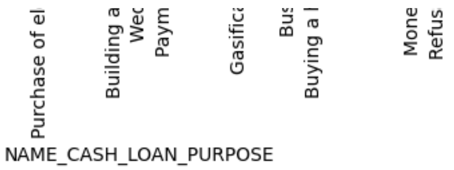



```
Analysis for AMT_GOODS_PRICE_y:
Summary statistics:
count     378584.000000
mean      104666.755623
std        65383.039386
min            0.000000
25%        51470.662500
50%       112320.000000
75%       117554.625000
max       350131.500000
Name: AMT_GOODS_PRICE_y, dtype: float64
```

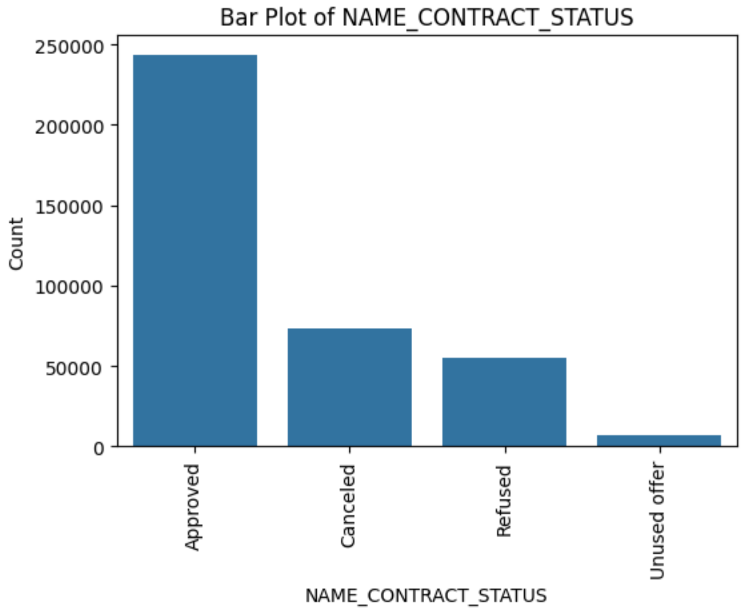



```
Analysis for NAME_CASH_LOAN_PURPOSE:
```

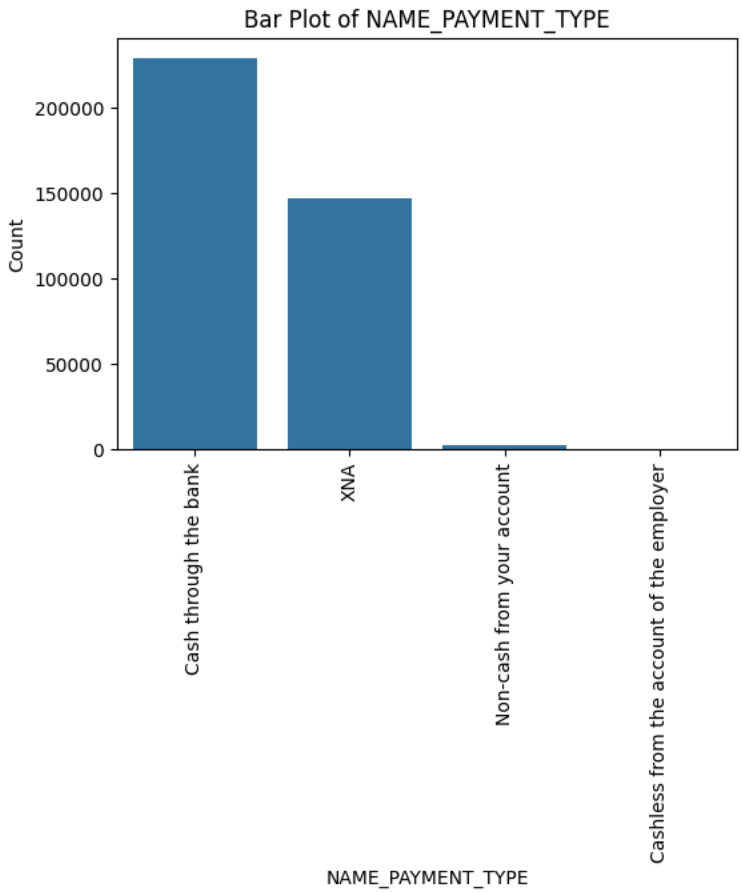




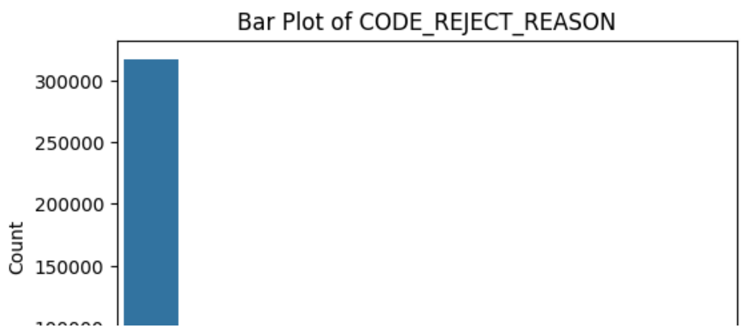
Analysis for NAME_CONTRACT_STATUS:



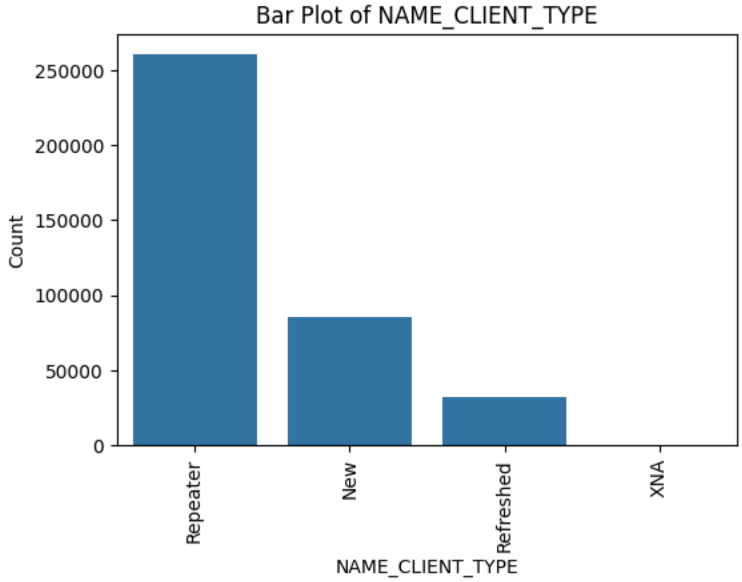
Analysis for NAME_PAYMENT_TYPE:



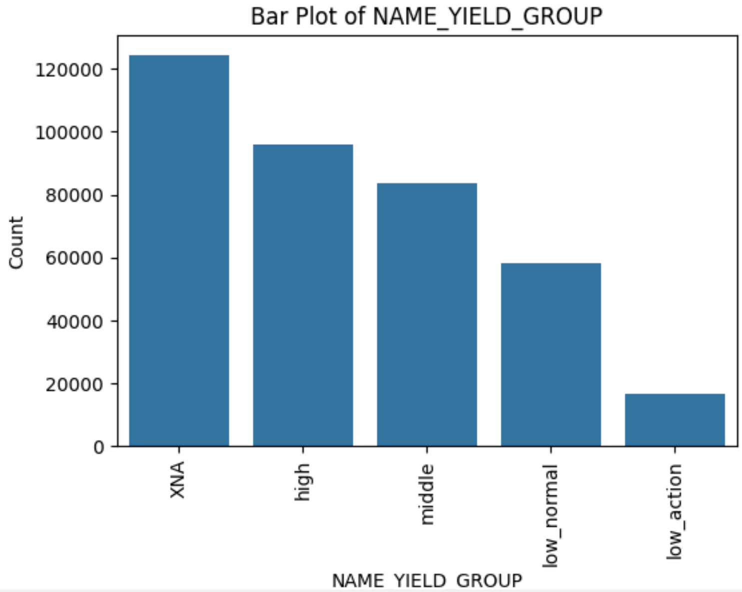
Analysis for CODE_REJECT_REASON:




```
Analysis for NAME_CLIENT_TYPE:
```




```
Analysis for NAME_YIELD_GROUP:
```



```
merge_df.TARGET.value_counts()
```

TARGET	count
0	342437
1	36147

Setting Threshold Value for Becoming Defaulter

```
round(36147*100/(36147+342437),0) #Percentage of clients becoming defaulter in overall data
```

```
10.0
```

Bivariate Analysis

```
pd.crosstab(merge_df.CODE_GENDER,merge_df.TARGET) #to get frequencies of target column for gender
```

CODE_GENDER \ TARGET	0	1
F	240829	22129
M	101600	14018
XNA	8	0

```
22129/(240829+22129)  # Percentage of females defaulting from total clients
```

```
0.08415412347218948
```

```
22129/(22129+14018)  # Percentage of females defaulting from total deafulting clients
```

```
0.6121946496251418
```

```
14018/(14018+101600)  # Percentage of males defaulting from total clients
```

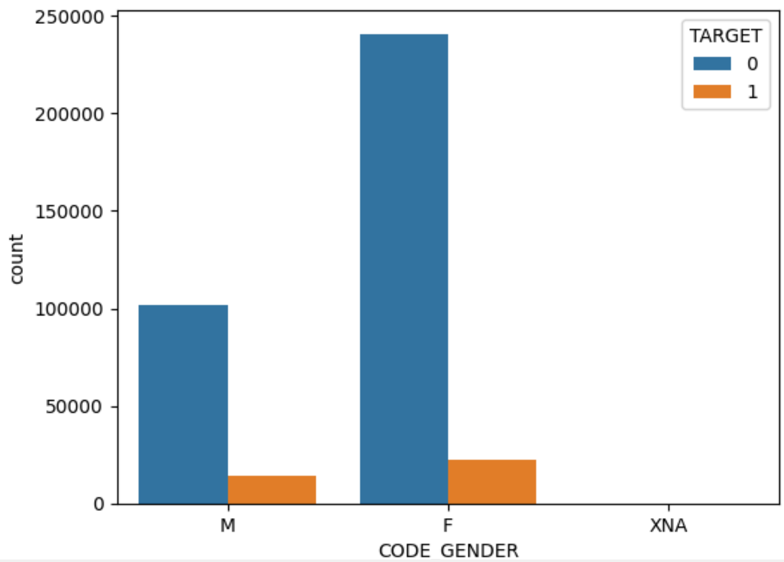
```
0.12124409693992284
```

```
14018/(22129+14018)  # Percentage of males defaulting from total deafulting clients
```

```
0.3878053503748582
```

```
import seaborn as sns
sns.countplot(x='CODE_GENDER',hue='TARGET',data=merge_df) #to check influence of gender column on target column
```

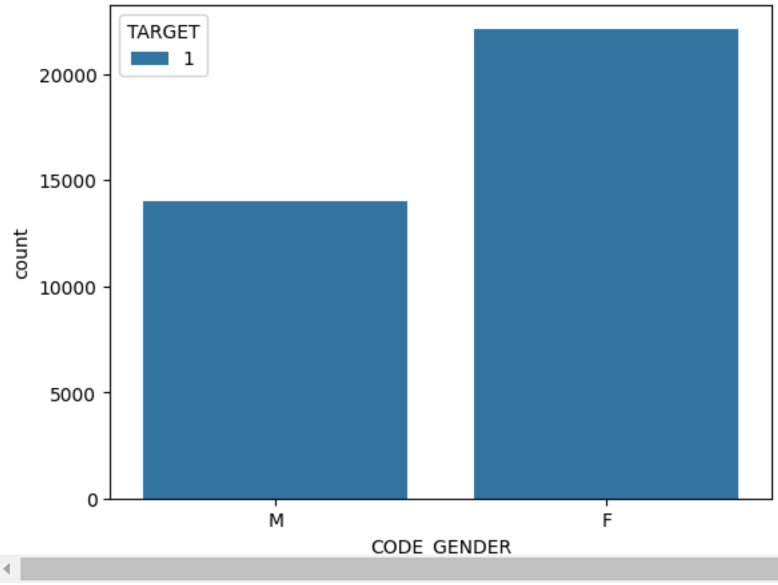
```
<Axes: xlabel='CODE_GENDER', ylabel='count'>
```



```
df1=merge_df.loc[merge_df.TARGET==1] #to create new dataframe containing records having only 1(defaulter) as target column value
```

```
import seaborn as sns
sns.countplot(x='CODE_GENDER',hue='TARGET',data=df1) #to check influence of gender column on target column value of 1
```

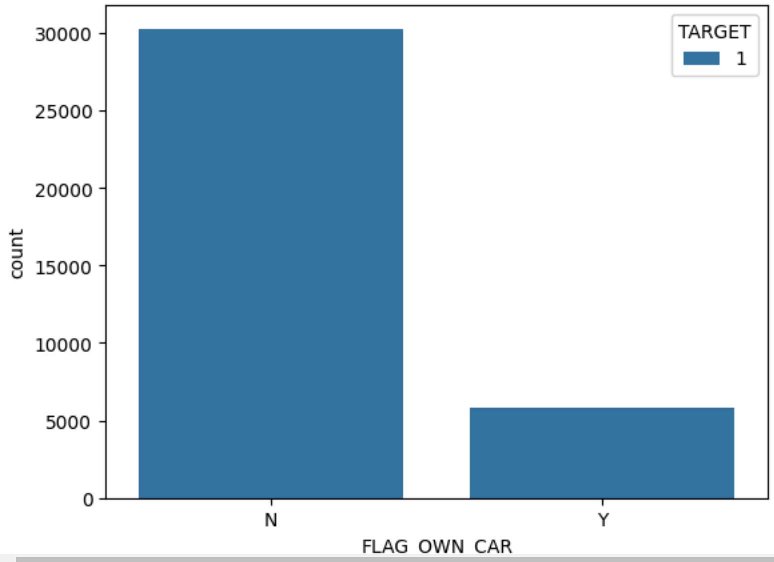
<Axes: xlabel='CODE_GENDER', ylabel='count'>




```
'''
 Conclusion-Overall there is 10% of defaulter credits.
 According to data, we observe that 4% more males   have a higher   chance of   not returning   their   loans default than females.
 But females contributes 61% of total defaulters.
'''
```

```
import seaborn as sns
sns.countplot(x='FLAG_OWN_CAR',hue='TARGET',data=df1) #to check influence of flag_own_car column on target column value of 1
```

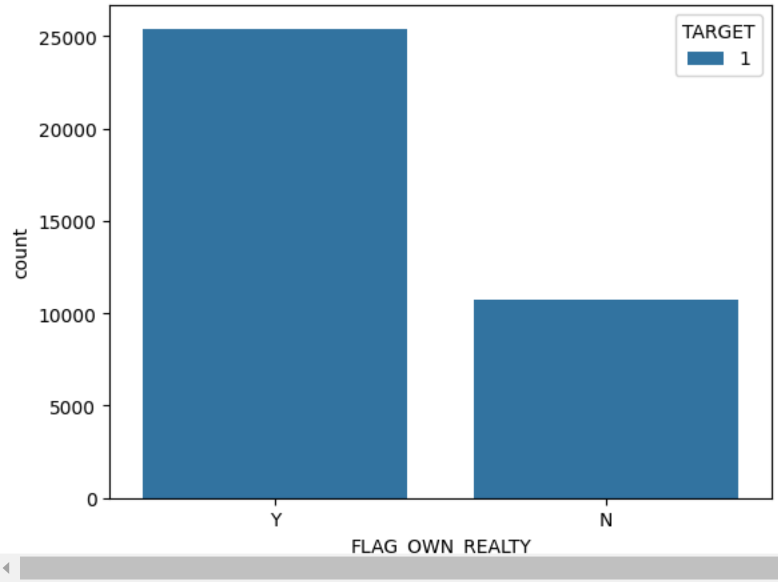
<Axes: xlabel='FLAG_OWN_CAR', ylabel='count'>




```
'''Conclusion-Clients those who own Car are less likely to default than those who don't own.'''
```

```
import seaborn as sns
sns.countplot(x='FLAG_OWN_REALTY',hue='TARGET',data=df1) #to check influence of flag_own_realty column on target column value of 1
```

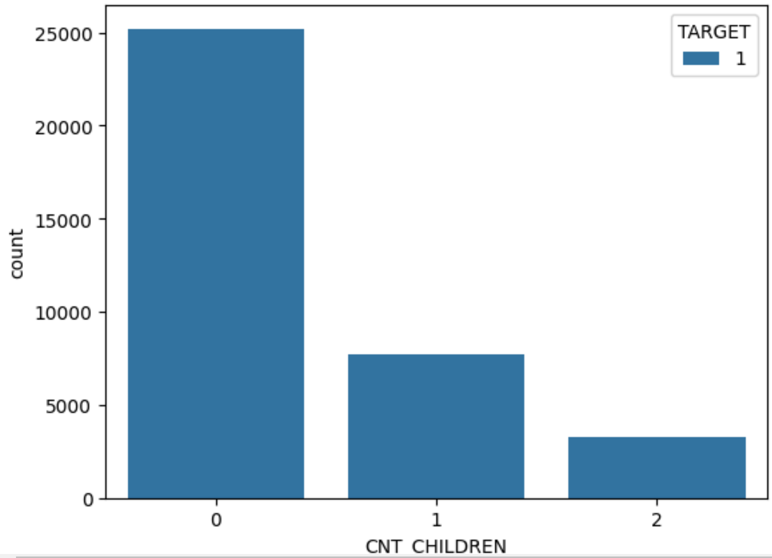
```
<Axes: xlabel='FLAG_OWN_REALTY', ylabel='count'>
```




```
'''Conclusion-Clients those who own Real Estate are more likely to default than those who don't own.'''
```

```
import seaborn as sns
sns.countplot(x='CNT_CHILDREN',hue='TARGET',data=df1) #to check influence of cnt_children column on target column value of 1
```

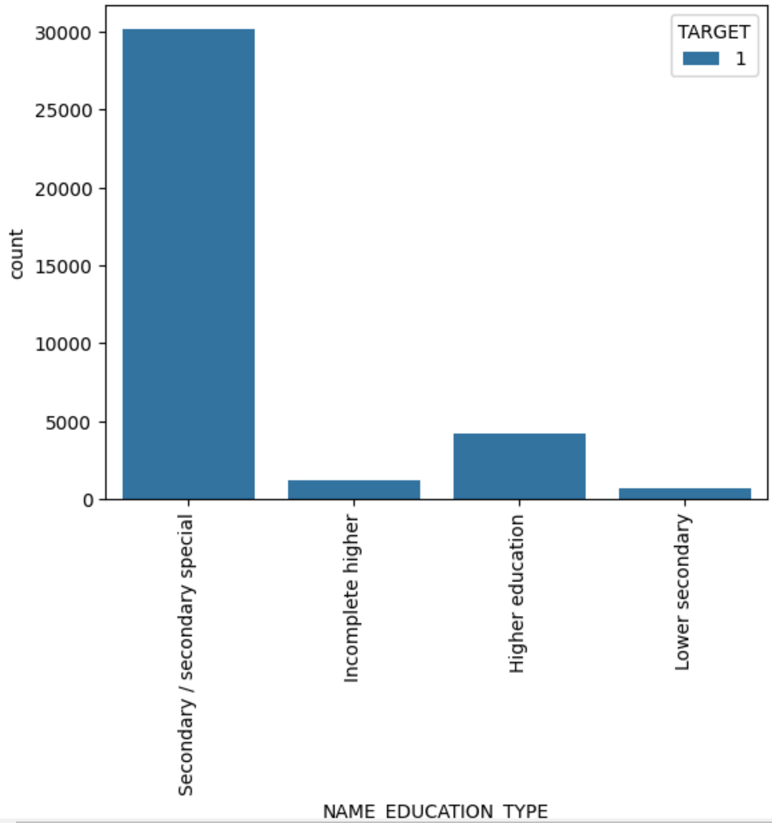
```
<Axes: xlabel='CNT_CHILDREN', ylabel='count'>
```




```
'''Conclusion-Clients those who don't have children are more likely to default than those have 1 or 2.'''
```

```
import seaborn as sns
sns.countplot(x='NAME_EDUCATION_TYPE',hue='TARGET',data=df1) #to check influence of name_education_type column on target column value 1
plt.xticks(rotation=90)
```

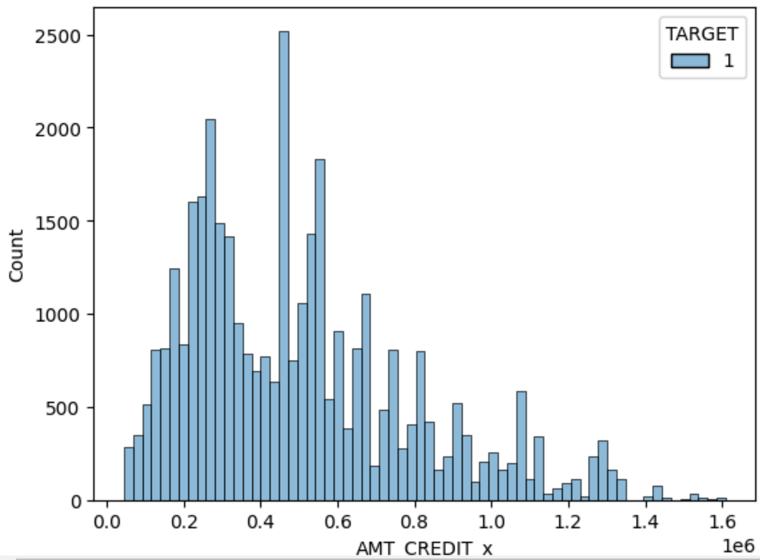
```
([0, 1, 2, 3],
 [Text(0, 0, 'Secondary / secondary special'),
  Text(1, 0, 'Incomplete higher'),
  Text(2, 0, 'Higher education'),
  Text(3, 0, 'Lower secondary')])
```




```
'''Conclusion-Clients those who have completed Secondary Education & Higher Eduaction are more likely to default than those have not cor
```

```
import seaborn as sns
sns.histplot(x='AMT_CREDIT_x',hue='TARGET',data=df1) #to check influence of amt_credit_x column on target column value 1
```

```
<Axes: xlabel='AMT_CREDIT_x', ylabel='Count'>
```




```
'''Conclusion-Clients having credit amount in present applied loan in range of 3 to 6 lakhs are more likely to default than others.'''
```

Multivariate Analysis

```
crosstab = pd.crosstab(
    [merge_df['CODE_GENDER'], merge_df['NAME_FAMILY_STATUS']],  # Predictors: Gender and Family Status
    merge_df['TARGET'],                                         # Response: Target
    values=None,                                                # No aggregation
    aggfunc=None,                                               # Frequency count
)
```

```
crosstab.rename(columns={0: 'Good Client', 1: 'Defaulter Client', 'All': 'Total'}, inplace=True) #To create table having counts of tager
```

```
crosstab.rename_axis(['Gender', 'Family Status'], inplace=True) #Rename existing columns
print(crosstab)
```

```
TARGET                        Good Client  Defaulter Client
Gender Family Status
F      Civil marriage               26730              2864
       Married                     136956             12674
       Separated                    18988              1518
       Single / not married         34017              3608
       Widow                        24138              1465
M      Civil marriage                9968              1734
       Married                      67234              8056
       Separated                     4585               780
       Single / not married         18990              3292
       Widow                          823               156
XNA    Married                          8                 0
```

```
multi_df=crosstab.reset_index() #Expand the Target column that is customised index
multi_df
```

TARGET	Gender	Family Status	Good Client	Defaulter Client
0	F	Civil marriage	26730	2864
1	F	Married	136956	12674
2	F	Separated	18988	1518
3	F	Single / not married	34017	3608
4	F	Widow	24138	1465
5	M	Civil marriage	9968	1734
6	M	Married	67234	8056
7	M	Separated	4585	780
8	M	Single / not married	18990	3292
9	M	Widow	823	156
10	XNA	Married	8	0

Next steps: View recommended plots New interactive sheet

```
multi_df=multi_df.drop(np.where(multi_df.Gender=='XNA')[0],axis=0) #Drop XNA value as gender because it will not be visible in graph
```

```
multi_df #To create table having counts of tagert column according to gender & then further divided by family status
```

TARGET	Gender	Family Status	Good Client	Defaulter Client
0	F	Civil marriage	26730	2864