PROJECT – 6 :

BANK LOAN CASE STUDY

Name : Shreyash Borkar

DESCRIPTION :

YOU'RE A DATA ANALYST AT A FINANCE COMPANY THAT SPECIALIZES IN LENDING VARIOUS TYPES OF LOANS TO URBAN CUSTOMERS. YOUR COMPANY FACES A CHALLENGE: SOME CUSTOMERS WHO DON'T HAVE A SUFFICIENT CREDIT HISTORY TAKE ADVANTAGE OF THIS AND DEFAULT ON THEIR LOANS. YOUR TASK IS TO USE EXPLORATORY DATA ANALYSIS (EDA) TO ANALYZE PATTERNS IN THE DATA AND ENSURE THAT CAPABLE APPLICANTS ARE NOT REJECTED.

YOUR GOAL IN THIS PROJECT IS TO USE EDA TO UNDERSTAND HOW CUSTOMER ATTRIBUTES AND LOAN ATTRIBUTES INFLUENCE THE LIKELIHOOD OF DEFAULT.

BUSINESS OBJECTIVE :

THE MAIN AIM OF THIS PROJECT IS TO IDENTIFY PATTERNS THAT INDICATE IF A CUSTOMER WILL HAVE DIFFICULTY PAYING THEIR INSTALLMENTS. THIS INFORMATION CAN BE USED TO MAKE DECISIONS SUCH AS DENYING THE LOAN, REDUCING THE AMOUNT OF LOAN, OR LENDING AT A HIGHER INTEREST RATE TO RISKY APPLICANTS. THE COMPANY WANTS TO UNDERSTAND THE KEY FACTORS BEHIND LOAN DEFAULT SO IT CAN MAKE BETTER DECISIONS ABOUT LOAN APPROVAL.

TOOLS & TECHNOLOGY USED :

MICROSOFT EXCEL (CELL REFERENCE)

JUPYTER NOTEBOOK (RUNNING QUERIES)

WORKFLOW:

DATA CLEANING AND PRE-PROCESSING

EDA

DATA ANALYSIS

DATA CLEANING :

- THE FIRST STEP WAS TO CHECK THE GIVEN DATA THOROUGHLY WHICH TELL US STORY ABOUT DATA.

- THEN NEXT STEP WAS TO CHECK FOR NULL VALUES COUNT AND PERCENTAGE OF MISSING VALUES.

- FOR THE APPLICATION DATASET, WE DECIDE A THRESHOLD OF 50% AND DROP COLUMNS HAVING MISSING VALUES > 50%

- NEXT WE AGAIN DROP COLUMNS HAVING MISSING VALUES > 40% OF MISSING VALUES AS THIS COLUMNS WERE MOSTLY NOT NEEDED FOR OUR ANALYSIS PURPOSES.

- NEXT WAS TO CHECK EACH COLUMN HAVING MISSING VALUES MANUALLY AND APPLY BEST STATISTICAL METHOD TO DEAL WITH MISSING VALUES.

- DELETING IRRELEVANT COLUMNS WAS ALSO A STEP PERFORMED DURING THIS ANALYSIS.

- THERE WERE ALSO SOME CATEGORICAL COLUMNS WITH MISSING VALUES SO ALSO IMPUTING THIS NULL VALUES WAS PERFORMED.

- NEXT WAS TO CLEAN THE PREVIOUS APPLICATIONS DATASET WITH STEPS PERFORMED ABOVE.

- NEXT WAS TO PERFORM FEATURE ENGINEERING AND DERIVE NECESSARY COLUMNS FROM EXISTING ONES.

- AFTER PERFORMING THE ABOVE STEPS AND SOME EXTRA STEPS FINALLY, OUR DATA WAS CLEAN AND WAS READY FOR FURTHER ANALYSIS.

Data Cleaning

In [ ]:
```
# checking null values for applications dataset

pd_application.isnull().sum()
```

```
EXT_SOURCE_2                    14
EXT_SOURCE_3                   857
APARTMENTS_AVG                2147
BASEMENTAREA_AVG              2481
YEARS_BEGINEXPLUATATION_AVG   2074
YEARS_BUILD_AVG               2820
COMMONAREA_AVG                2975
ELEVATORS_AVG                 2242
ENTRANCES_AVG                 2118
FLOORSMAX_AVG                 2093
FLOORSMIN_AVG                 2891
LANDAREA_AVG                  2509
LIVINGAPARTMENTS_AVG          2916
LIVINGAREA_AVG                2124
NONLIVINGAPARTMENTS_AVG       2958
NONLIVINGAREA_AVG             2317
APARTMENTS_MODE               2147
BASEMENTAREA_MODE             2481
YEARS_BEGINEXPLUATATION_MODE  2074
YEARS_BUILD_MODE              2820
```

In [ ]:
```
# % missing values
missing = (pd_application.isnull().sum()/len(pd_application)) * 100
missing
```

```
EXT_SOURCE_2                    0.214626
APARTMENTS_AVG                 49.814385
BASEMENTAREA_AVG               57.563805
YEARS_BEGINEXPLUATATION_AVG    48.120650
YEARS_BUILD_AVG                65.429234
COMMONAREA_AVG                 69.025522
ELEVATORS_AVG                  52.018561
ENTRANCES_AVG                  49.141531
FLOORSMAX_AVG                  48.561485
FLOORSMIN_AVG                  67.076566
LANDAREA_AVG                   58.213457
LIVINGAPARTMENTS_AVG           67.656613
LIVINGAREA_AVG                 49.280742
NONLIVINGAPARTMENTS_AVG        68.631090
NONLIVINGAREA_AVG              53.758701
APARTMENTS_MODE                49.814385
BASEMENTAREA_MODE              57.563805
YEARS_BEGINEXPLUATATION_MODE   48.120650
YEARS_BUILD_MODE               65.429234
COMMONAREA_MODE                69.025522
```

In [ ]:
```
# there are few columns who have missing value > 50% of there available data so we will drop those columns

threshold = 50
cols_above_threshold = missing[missing > threshold]
cols_above_threshold
```

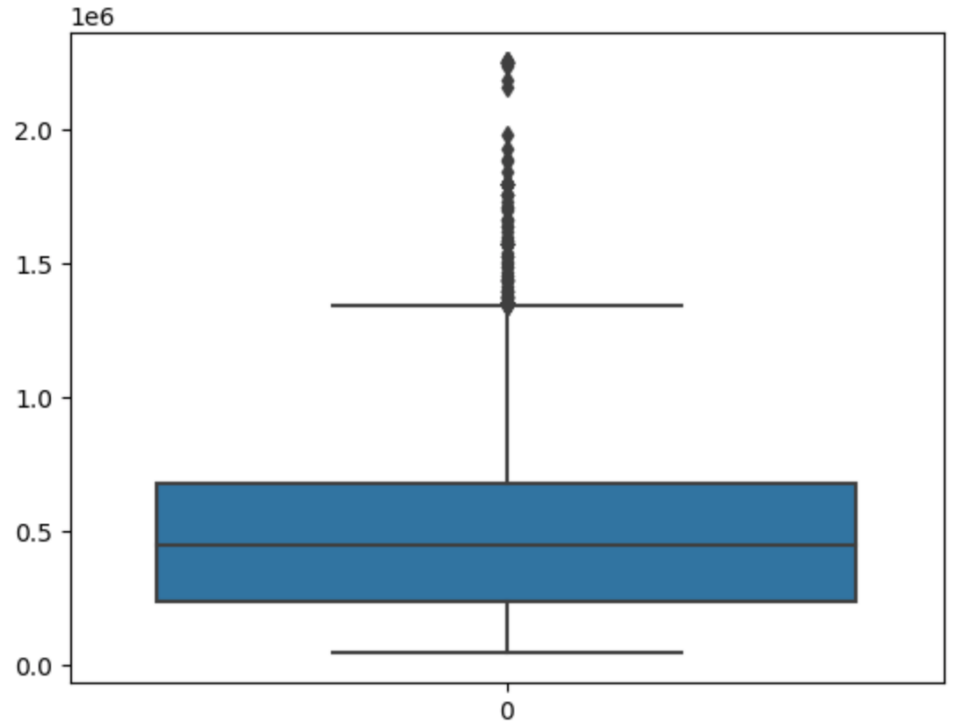
Out[17]:
```
OWN_CAR_AGE                 66.473318
EXT_SOURCE_1                56.635731
BASEMENTAREA_AVG            57.563805
YEARS_BUILD_AVG             65.429234
COMMONAREA_AVG              69.025522
ELEVATORS_AVG               52.018561
FLOORSMIN_AVG               67.076566
LANDAREA_AVG                58.213457
LIVINGAPARTMENTS_AVG        67.656613
NONLIVINGAPARTMENTS_AVG     68.631090
NONLIVINGAREA_AVG           53.758701
BASEMENTAREA_MODE           57.563805
YEARS_BUILD_MODE            65.429234
COMMONAREA_MODE             69.025522
ELEVATORS_MODE              52.018561
FLOORSMIN_MODE              67.076566
LANDAREA_MODE               58.213457
LIVINGAPARTMENTS_MODE       67.656613
NONLIVINGAPARTMENTS_MODE    68.631090
NONLIVINGAREA_MODE          53.758701
BASEMENTAREA_MEDI           57.563805
YEARS_BUILD_MEDI            65.429234
COMMONAREA_MEDI             69.025522
ELEVATORS_MEDI              52.018561
FLOORSMIN_MEDI              67.076566
LANDAREA_MEDI               58.213457
LIVINGAPARTMENTS_MEDI       67.656613
NONLIVINGAPARTMENTS_MEDI    68.631090
NONLIVINGAREA_MEDI          53.758701
FONDKAPREMONT_MODE          67.215777
dtype: float64
```

In [ ]:
```
# there are also certain columns whose missing values are > 40%

threshold = 40
cols_to_drop = missing1[missing1 > threshold]
cols_to_drop
```

Out[22]:
```
APARTMENTS_AVG                 49.814385
YEARS_BEGINEXPLUATATION_AVG    48.120650
ENTRANCES_AVG                  49.141531
FLOORSMAX_AVG                  48.561485
LIVINGAREA_AVG                 49.280742
APARTMENTS_MODE                49.814385
YEARS_BEGINEXPLUATATION_MODE   48.120650
ENTRANCES_MODE                 49.141531
FLOORSMAX_MODE                 48.561485
LIVINGAREA_MODE                49.280742
APARTMENTS_MEDI                49.814385
YEARS_BEGINEXPLUATATION_MEDI   48.120650
ENTRANCES_MEDI                 49.141531
FLOORSMAX_MEDI                 48.561485
LIVINGAREA_MEDI                49.280742
HOUSETYPE_MODE                 49.071926
TOTALAREA_MODE                 47.424594
WALLSMATERIAL_MODE             49.721578
EMERGENCYSTATE_MODE            46.542923
dtype: float64
```

In [ ]:
```
sns.boxplot(pd_application['AMT_GOODS_PRICE'])
```

Out[28]: <Axes: >



In [ ]:
```
print(pd_application['AMT_GOODS_PRICE'].mean())
print(pd_application['AMT_GOODS_PRICE'].median())
```

```
539396.0339986075
450000.0
```

In [ ]:
```
pd_application['NAME_TYPE_SUITE'].value_counts()
```

Out[33]:
```
Unaccompanied      3479
Family              577
Spouse, partner     157
Children             40
Other_B              29
Other_A              10
Group of people       3
Name: NAME_TYPE_SUITE, dtype: int64
```

In [ ]:
```
# clearly Unaccompanied is the most occuring variable so, we will impute null with mode

pd_application['NAME_TYPE_SUITE'].fillna('Unaccompanied',inplace=True)
```

In [ ]:
```
pd_application['NAME_TYPE_SUITE'].isnull().sum()
```

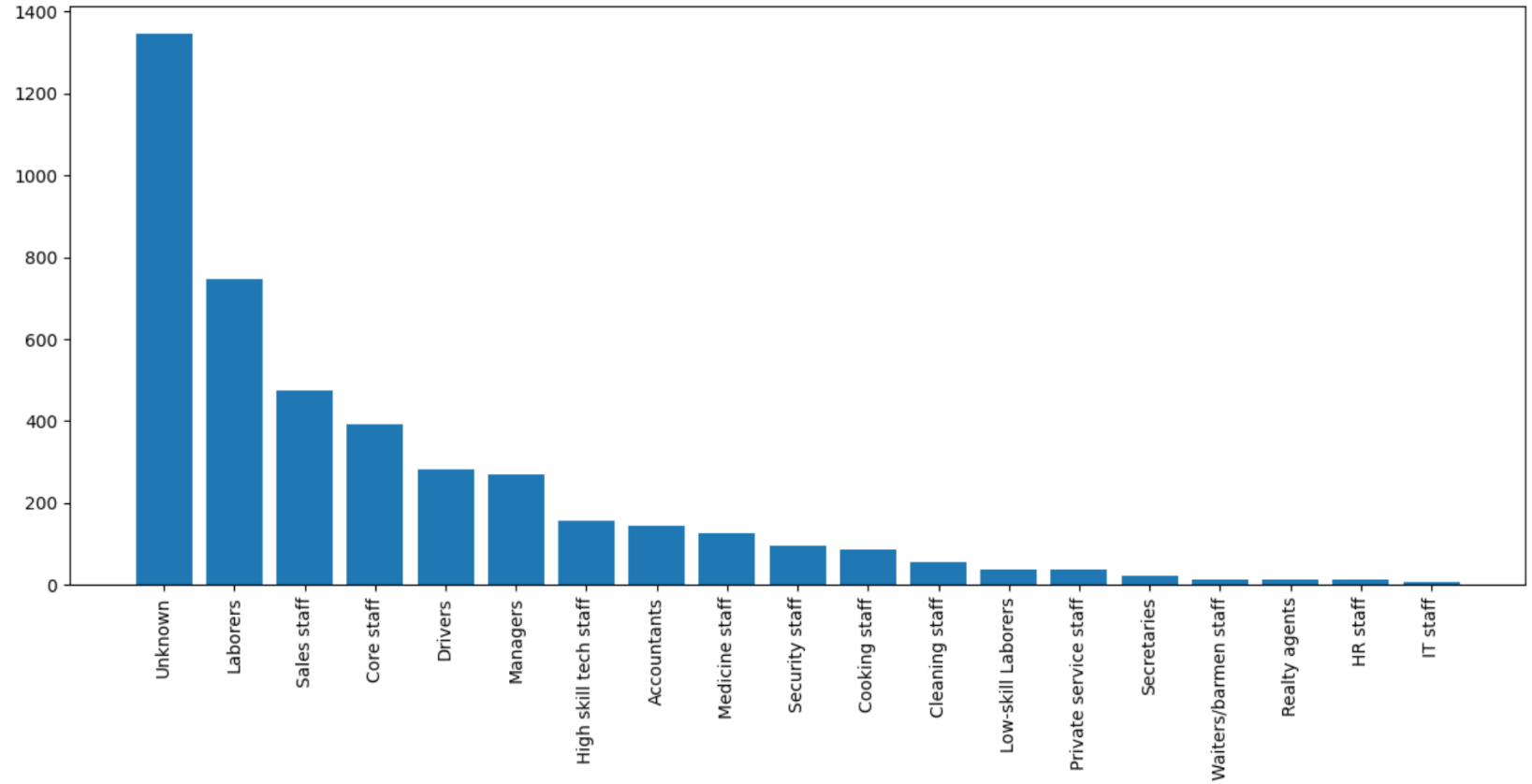
Out[36]:
```
0
```

In [ ]:
```
pd_application['OCCUPATION_TYPE'].fillna('Unknown',inplace=True)
```

In [ ]:
```
count1 = pd_application['OCCUPATION_TYPE'].value_counts()
count1
```

Out[55]:
```
Unknown                 1346
Laborers                 745
Sales staff              473
Core staff               393
Drivers                  282
Managers                 269
High skill tech staff    155
Accountants              144
Medicine staff           127
Security staff            95
Cooking staff             87
Cleaning staff            54
Low-skill Laborers        38
Private service staff     38
Secretaries               23
Waiters/barmen staff      13
Realty agents             11
HR staff                  11
IT staff                   6
Name: OCCUPATION_TYPE, dtype: int64
```

In [ ]:
```
plt.figure(figsize=[15, 6])
plt.bar(count1.index, count1.values)
plt.xticks(rotation=90)
plt.show()
```



In [ ]:

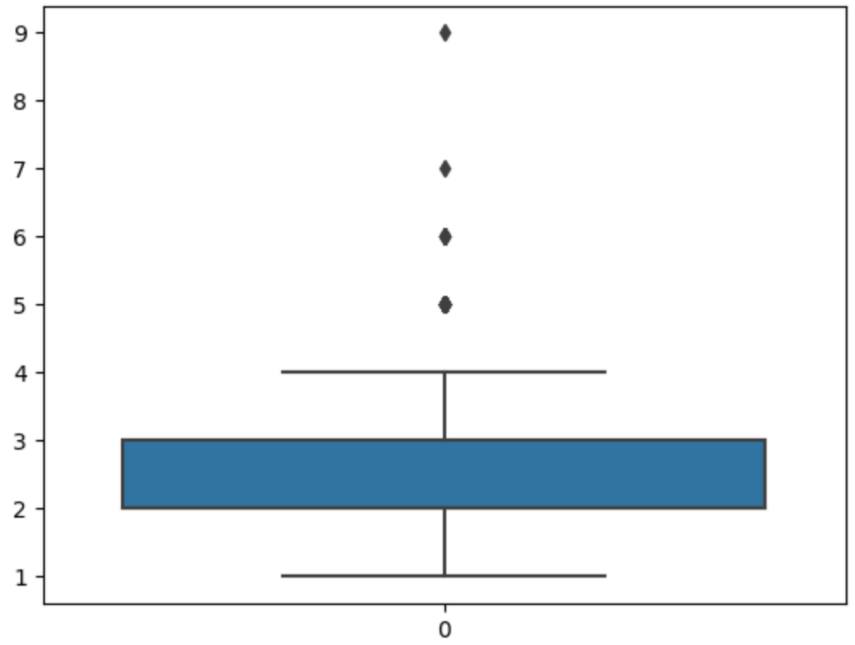
```
# deleting irrelevant columns which are not required for furthur analysis

irr_cols = ['WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START', 'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LI
            'LIVE_CITY_NOT_WORK_CITY', 'DAYS_LAST_PHONE_CHANGE','OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCL
            'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'REG_CITY_NOT_WORK_CITY','HOUR_APPR_PROCESS_STAR
            'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_C
            'DAYS_ID_PUBLISH']

pd_application = pd_application.drop(irr_cols,axis=1)
```

In [ ]:
```
pd_application['CNT_FAM_MEMBERS'].value_counts()
```

Out[63]:
```
2.0    2243
1.0     962
3.0     739
4.0     299
5.0      59
6.0       5
9.0       1
7.0       1
Name: CNT_FAM_MEMBERS, dtype: int64
```

In [ ]:
```
sns.boxplot(pd_application['CNT_FAM_MEMBERS'])
```

Out[64]:
```
<Axes: >
```



Previous Application Data Cleaning

In [ ]:
```
pd_previous.isnull().sum()
```

Out[105]:

SK_ID_PREV	0
SK_ID_CURR	0
NAME_CONTRACT_TYPE	0
AMT_ANNUITY	10592
AMT_APPLICATION	0
AMT_CREDIT	0
AMT_DOWN_PAYMENT	25198
AMT_GOODS_PRICE	10744
WEEKDAY_APPR_PROCESS_START	0
HOUR_APPR_PROCESS_START	0
FLAG_LAST_APPL_PER_CONTRACT	0
NFLAG_LAST_APPL_IN_DAY	0
RATE_DOWN_PAYMENT	25198
RATE_INTEREST_PRIMARY	49834
RATE_INTEREST_PRIVILEGED	49834
NAME_CASH_LOAN_PURPOSE	0
NAME_CONTRACT_STATUS	0
DAYS_DECISION	0
NAME_PAYMENT_TYPE	0
CODE_REJECT_REASON	0
NAME_TYPE_SUITE	24243
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	10592
NAME_YIELD_GROUP	0
PRODUCT_COMBINATION	8
DAYS_FIRST_DRAWING	19160
DAYS_FIRST_DUE	19160
DAYS_LAST_DUE_1ST_VERSION	19160
DAYS_LAST_DUE	19160
DAYS_TERMINATION	19160

In [ ]:
```
missing = pd_previous.isnull().sum()/len(pd_previous) * 100
missing
```

Out[111]:
```
SK_ID_PREV                    0.000000
SK_ID_CURR                    0.000000
NAME_CONTRACT_TYPE            0.000000
AMT_ANNUITY                  21.184424
AMT_APPLICATION               0.000000
AMT_CREDIT                    0.000000
AMT_DOWN_PAYMENT             50.397008
AMT_GOODS_PRICE              21.488430
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
RATE_DOWN_PAYMENT            50.397008
RATE_INTEREST_PRIMARY        99.669993
RATE_INTEREST_PRIVILEGED     99.669993
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_TYPE_SUITE              48.486970
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                  21.184424
NAME_YIELD_GROUP              0.000000
PRODUCT_COMBINATION           0.016000
DAYS_FIRST_DRAWING           38.320766
DAYS_FIRST_DUE               38.320766
DAYS_LAST_DUE_1ST_VERSION    38.320766
DAYS_LAST_DUE                38.320766
DAYS_TERMINATION             38.320766
NFLAG_INSURED_ON_APPROVAL    38.320766
dtype: float64
```

In [ ]:
```
# dropping > 50%
threshold = 50
cols_to_drop = missing[missing > threshold]
cols_to_drop
```

Out[113]:
```
AMT_DOWN_PAYMENT            50.397008
RATE_DOWN_PAYMENT           50.397008
RATE_INTEREST_PRIMARY       99.669993
RATE_INTEREST_PRIVILEGED    99.669993
dtype: float64
```

In [ ]:
```
pd_previous.drop(columns=cols_to_drop.index,inplace=True)
```

In [ ]:
```
pd_previous.isnull().sum()/len(pd_previous) * 100
```

Out[115]:

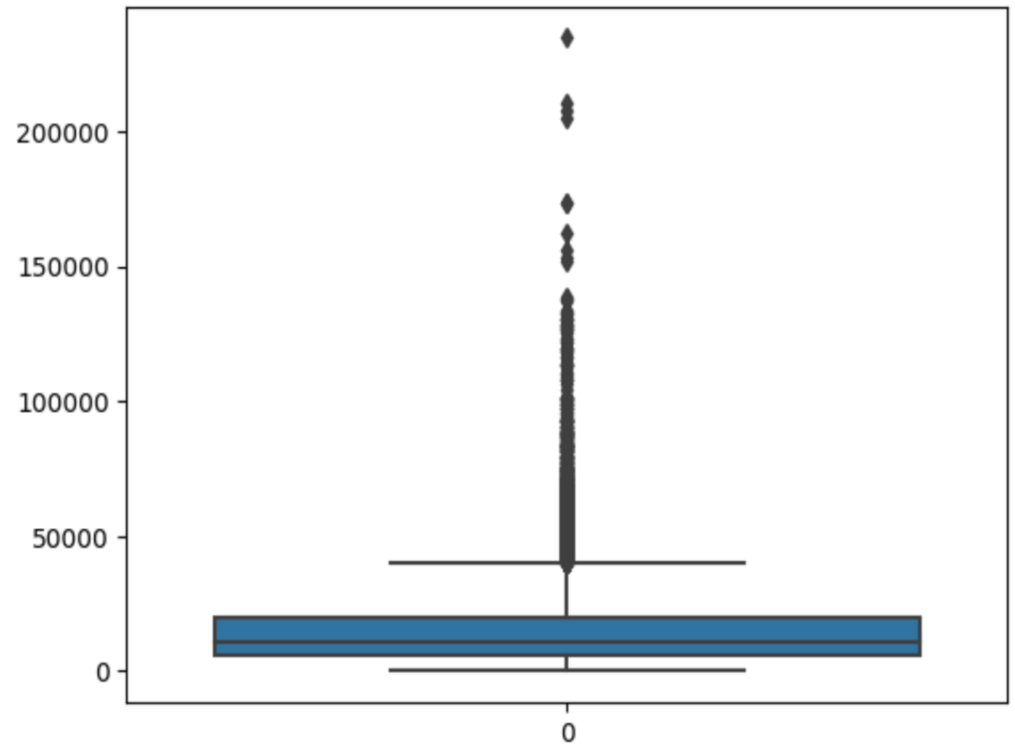
SK_ID_PREV	0.000000
SK_ID_CURR	0.000000
NAME_CONTRACT_TYPE	0.000000
AMT_ANNUITY	21.184424
AMT_APPLICATION	0.000000
AMT_CREDIT	0.000000
AMT_GOODS_PRICE	21.488430
WEEKDAY_APPR_PROCESS_START	0.000000
HOUR_APPR_PROCESS_START	0.000000
FLAG_LAST_APPL_PER_CONTRACT	0.000000
NFLAG_LAST_APPL_IN_DAY	0.000000
NAME_CASH_LOAN_PURPOSE	0.000000
NAME_CONTRACT_STATUS	0.000000
DAYS_DECISION	0.000000
NAME_PAYMENT_TYPE	0.000000
CODE_REJECT_REASON	0.000000
NAME_TYPE_SUITE	48.486970
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	21.184424
NAME_YIELD_GROUP	0.000000
PRODUCT_COMBINATION	0.016000
DAYS_FIRST_DRAWING	38.320766
DAYS_FIRST_DUE	38.320766
DAYS_LAST_DUE_1ST_VERSION	38.320766
DAYS_LAST_DUE	38.320766
DAYS_TERMINATION	38.320766
NFLAG_INSURED_ON_APPROVAL	38.320766

dtype: float64

In [ ]:
```
pd_previous['AMT_ANNUITY'].value_counts()
```

In [ ]:
```
sns.boxplot(pd_previous['AMT_ANNUITY'])
```

Out[108]: <Axes: >



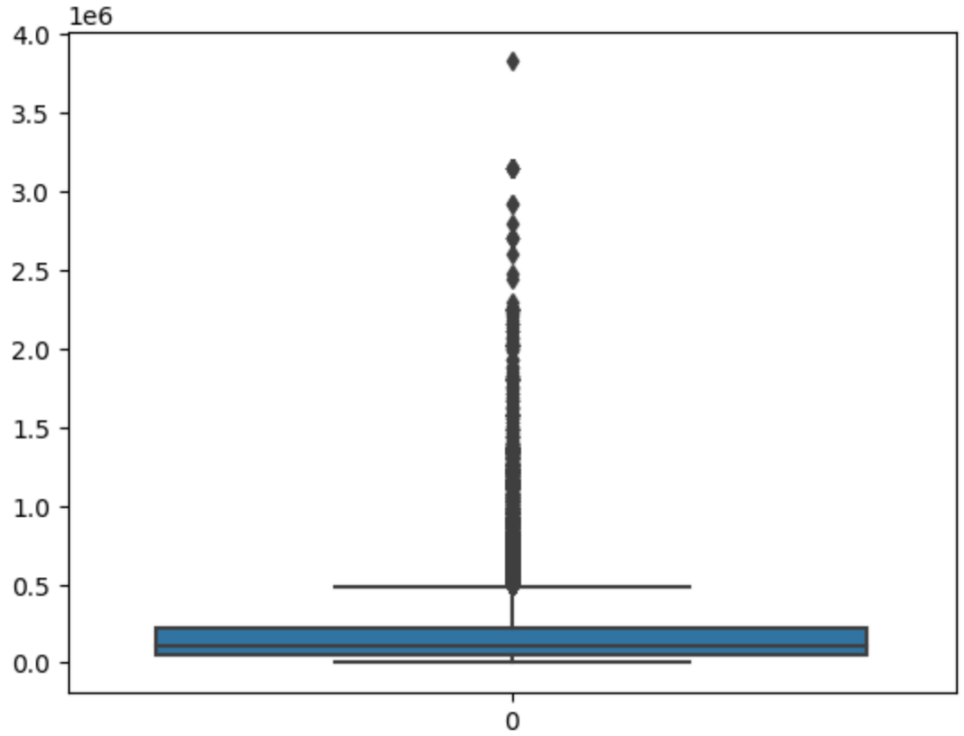
In [ ]:
```
pd_previous['AMT_ANNUITY'].median()
```

Out[109]: 10879.92

In [ ]:
```
pd_previous['AMT_ANNUITY'].fillna(pd_previous['AMT_ANNUITY'].median(),inplace=True)
```

In [ ]:
```
sns.boxplot(pd_previous['AMT_GOODS_PRICE'])
```

Out[117]: <Axes: >



In [ ]:
```
pd_previous['AMT_GOODS_PRICE'].median()
```

Out[118]: 104017.5

In [ ]:
```
pd_previous['AMT_GOODS_PRICE'].mean()
```

Out[119]: 215141.41734887278

In [ ]:
```
pd_previous['AMT_GOODS_PRICE'].fillna(pd_previous['AMT_GOODS_PRICE'].median(),inplace=True)
```

In [ ]:
```
pd_previous['PRODUCT_COMBINATION'].value_counts(normalize=True) * 100
```

Out[134]:
```
POS household with interest       17.023064
Cash                              15.880859
POS mobile with interest          14.060531
Cash X-Sell: middle                7.907423
Cash X-Sell: low                   7.079274
Card Street                        6.647196
POS industry with interest         6.463163
POS household without interest     5.599008
Card X-Sell                        4.604829
Cash Street: high                  3.504631
Cash X-Sell: high                  3.314597
Cash Street: low                   2.112380
Cash Street: middle                1.920346
POS mobile without interest        1.462263
POS other with interest            1.456262
POS industry without interest      0.780140
POS others without interest        0.184033
Name: PRODUCT_COMBINATION, dtype: float64
```

In [ ]:
```
pd_previous['PRODUCT_COMBINATION'].fillna('Unknown',inplace=True)
```

In [ ]:
```
pd_previous['DAYS_FIRST_DRAWING'].value_counts()
```

In [ ]:
```
# converting days to absolute number
pd_previous[['DAYS_DECISION','DAYS_FIRST_DRAWING' ,'DAYS_FIRST_DUE','DAYS_LAST_DUE_1ST_VERSION', 'DAYS_LAST_DUE','DA
```

In [ ]:
```
pd_previous[['DAYS_DECISION_YEARS','DAYS_FIRST_DRAWING_YEARS','DAYS_FIRST_DUE_YEARS','DAYS_LAST_DUE_1ST_VERSION_YEAR
```

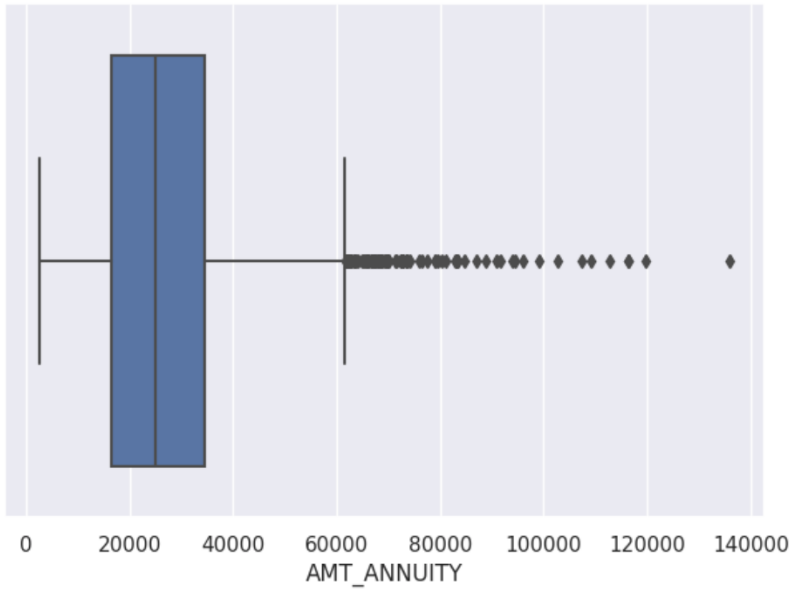
Univariate Analysis

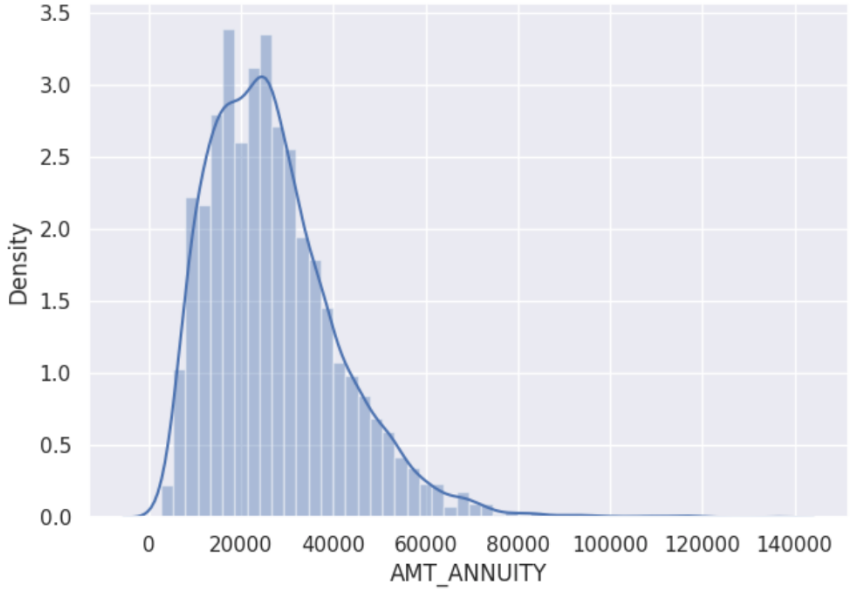
In [ ]:
```
Numarical_Data = ['AMT_ANNUITY','AMT_GOODS_PRICE','CNT_FAM_MEMBERS']
```

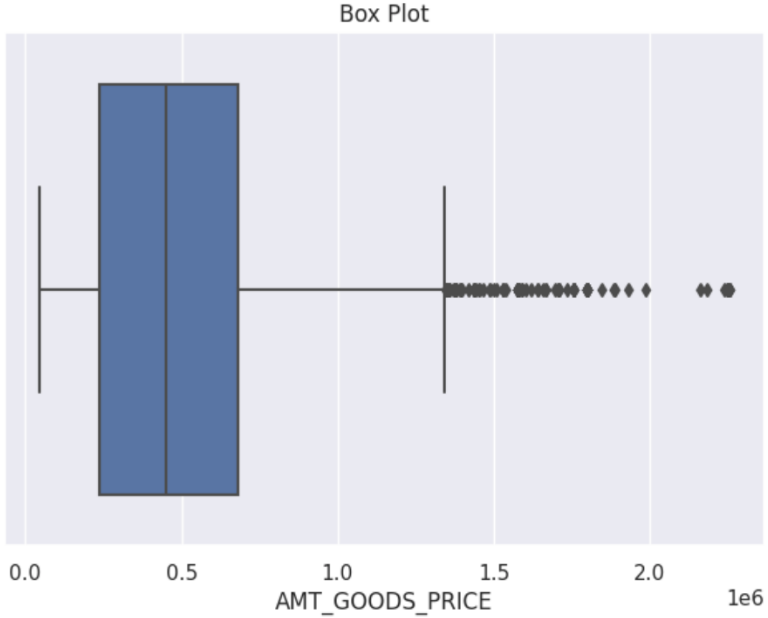
In [ ]:
```
Categorical_Data = ['FLAG_OWN_CAR','FLAG_OWN_REALTY','NAME_TYPE_SUITE','NAME_INCOME_TYPE','NAME_EDUCATION_TYPE', 'NA
```

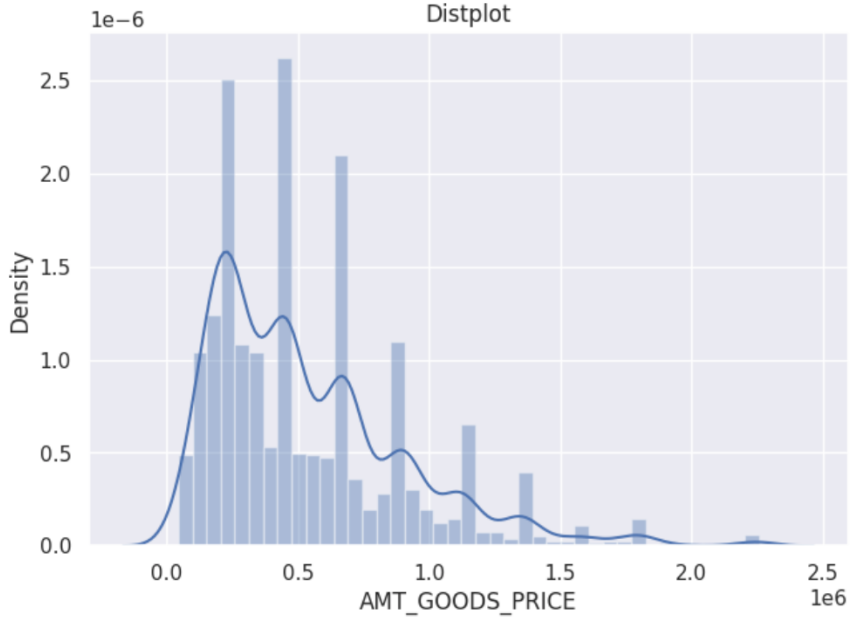
In [ ]:
```
def Uni_Analysis_Numarical(dataframe, column):
  sns.set(style='darkgrid')
  plt.figure(figsize=(25, 5))
  plt.subplot(1, 3, 1)
  sns.boxplot(data=dataframe, x=column, orient='v').set(title='Box Plot')
  plt.subplot(1, 3, 2)
  sns.distplot(dataframe[column].dropna()).set(title='Distplot')
  plt.show()
```

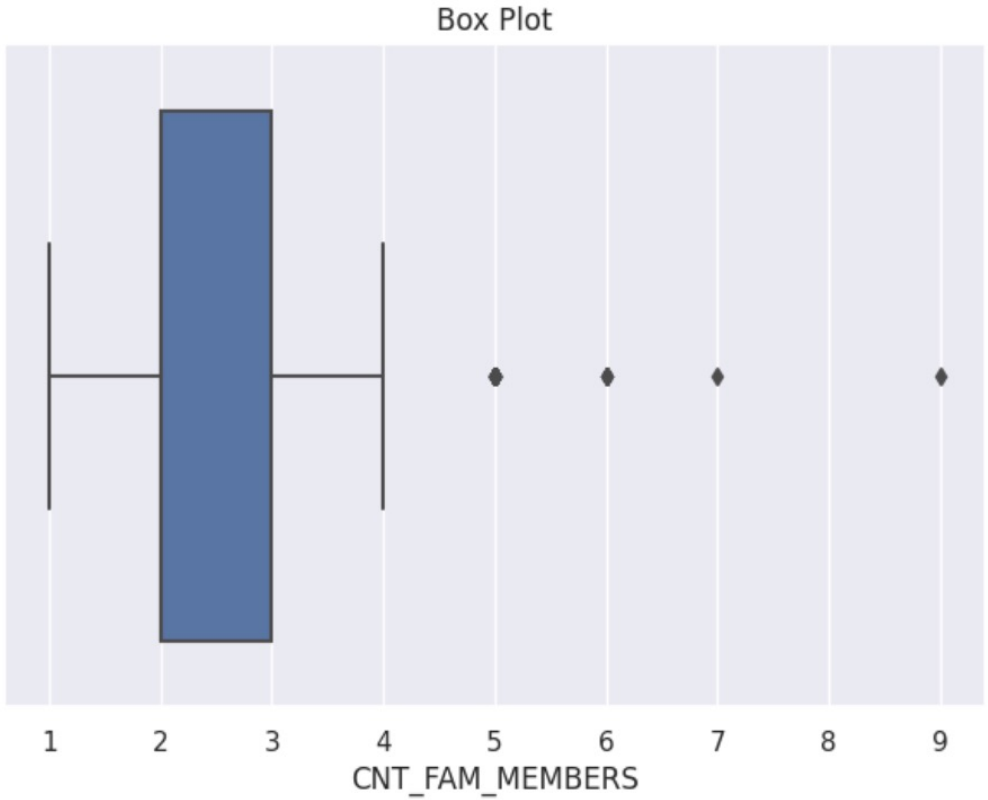
In [ ]:
```
def Uni_Analysis_Categorcal(dataframe, column):
  sns.set(style='darkgrid')
  plt.figure(figsize = [12,5])
  dataframe[column].value_counts().plot.barh(width = 0.8)
  plt.title(column)
  plt.show()
```

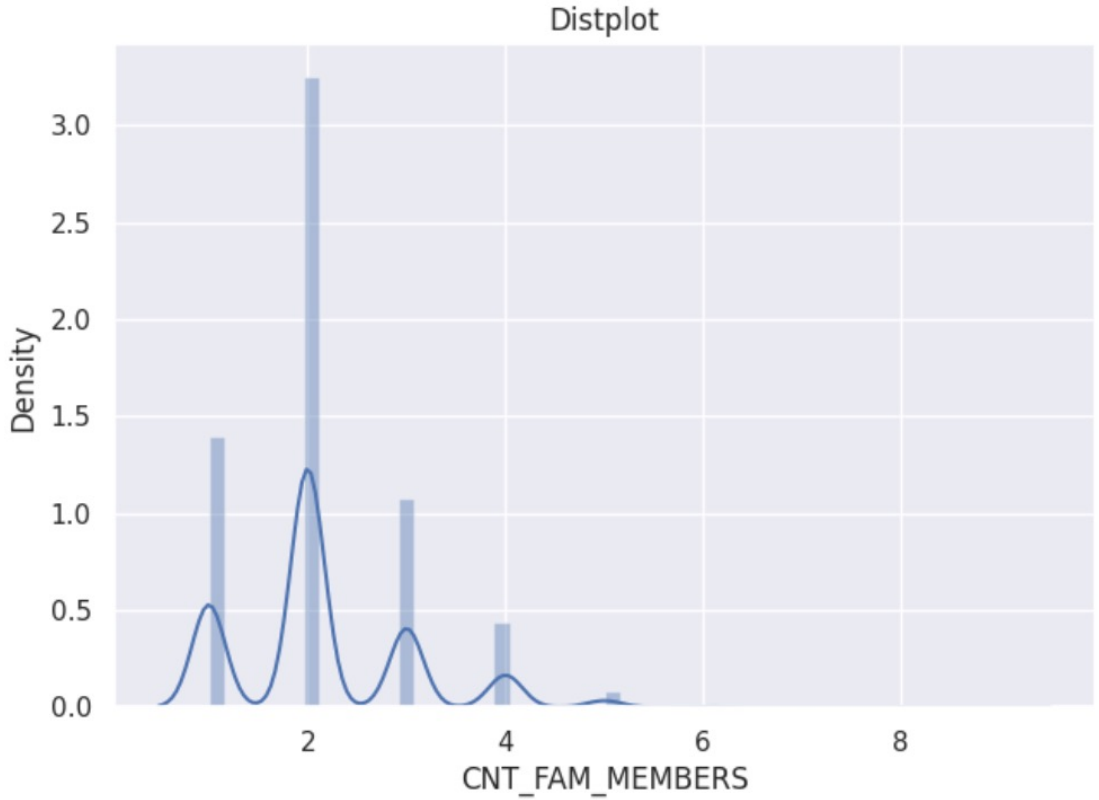

Box Plot
0
20000
40000
60000
80000
100000
120000
140000
AMT_ANNUITY


Distplot
1e−5
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Density
0
20000
40000
60000
80000
100000
120000
140000
AMT_ANNUITY

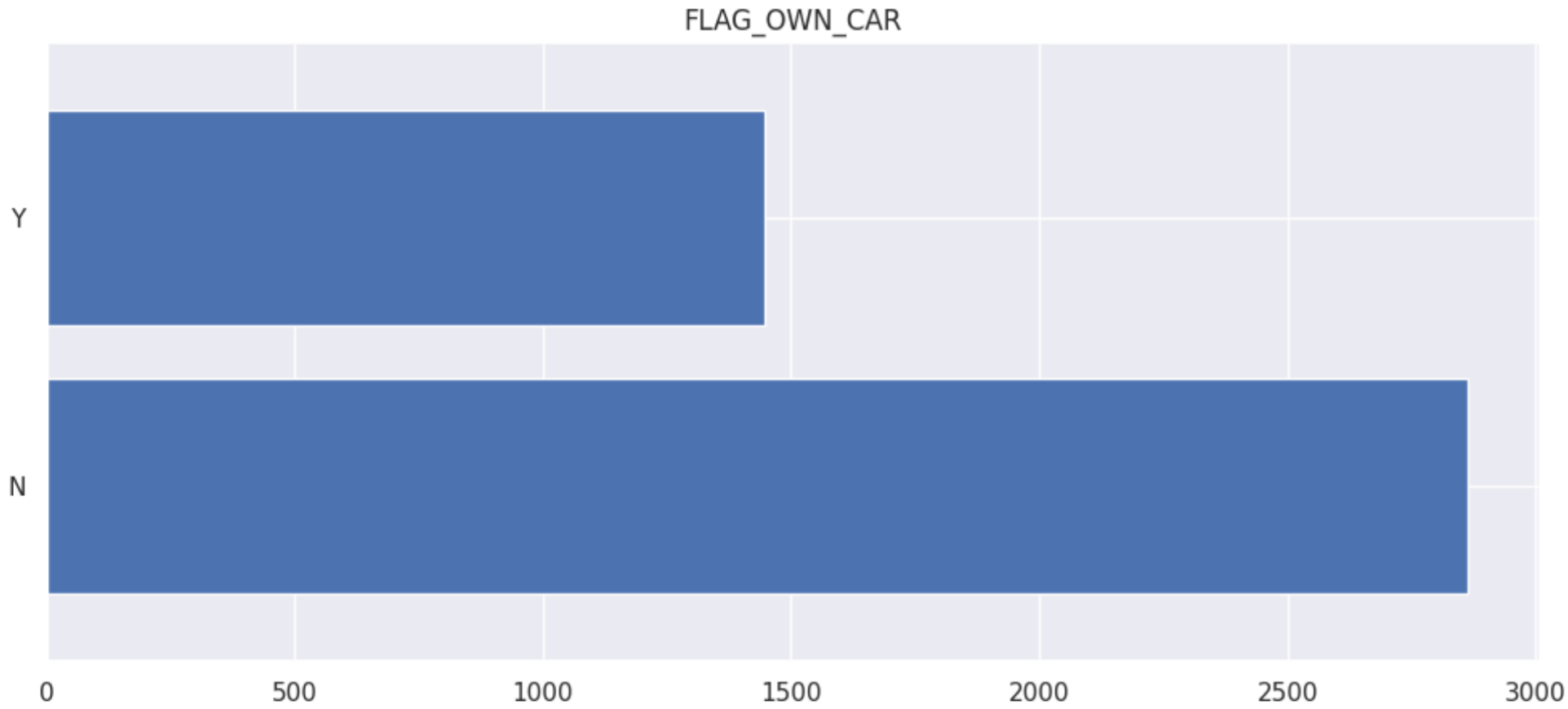

Box Plot
0.0
0.5
1.0
1.5
2.0
AMT_GOODS_PRICE
1e6

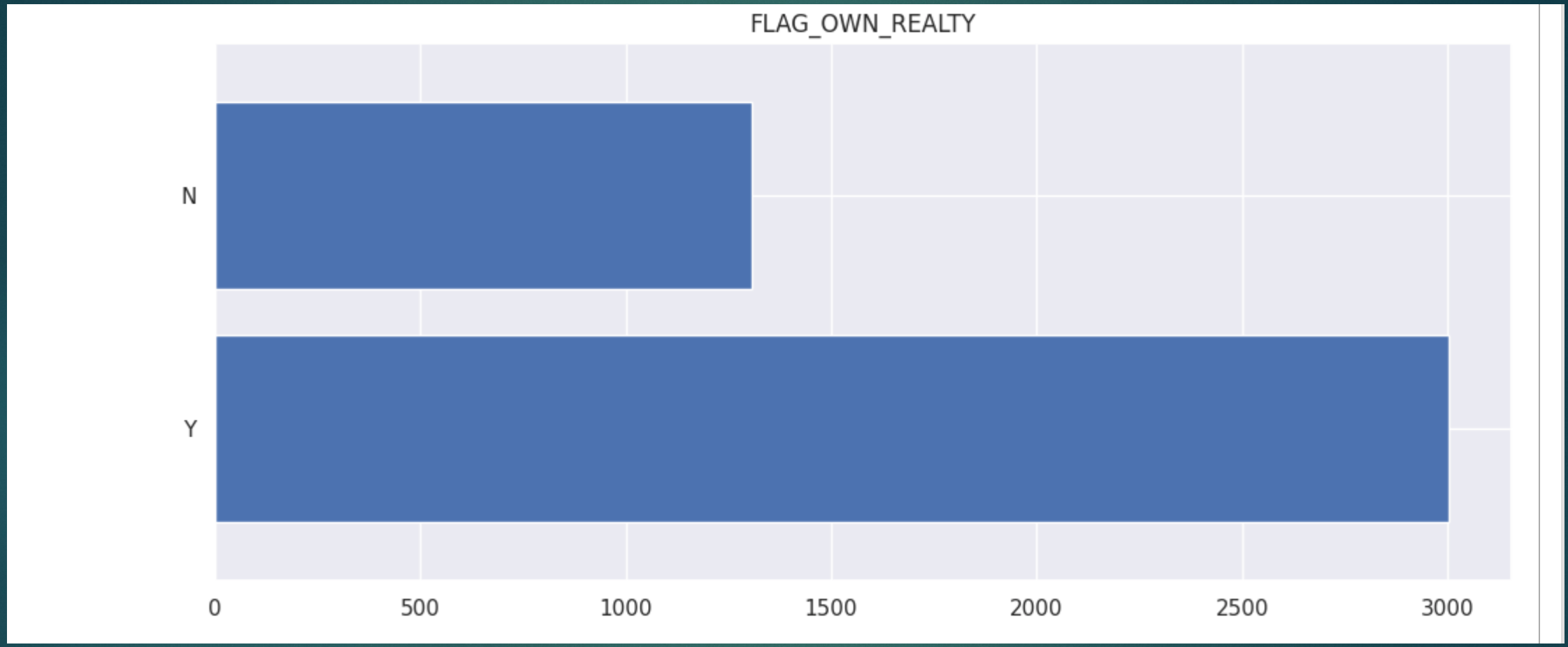

Distplot
1e−6
2.5
2.0
1.5
1.0
0.5
0.0
Density
0.0
0.5
1.0
1.5
2.0
2.5
AMT_GOODS_PRICE
1e6

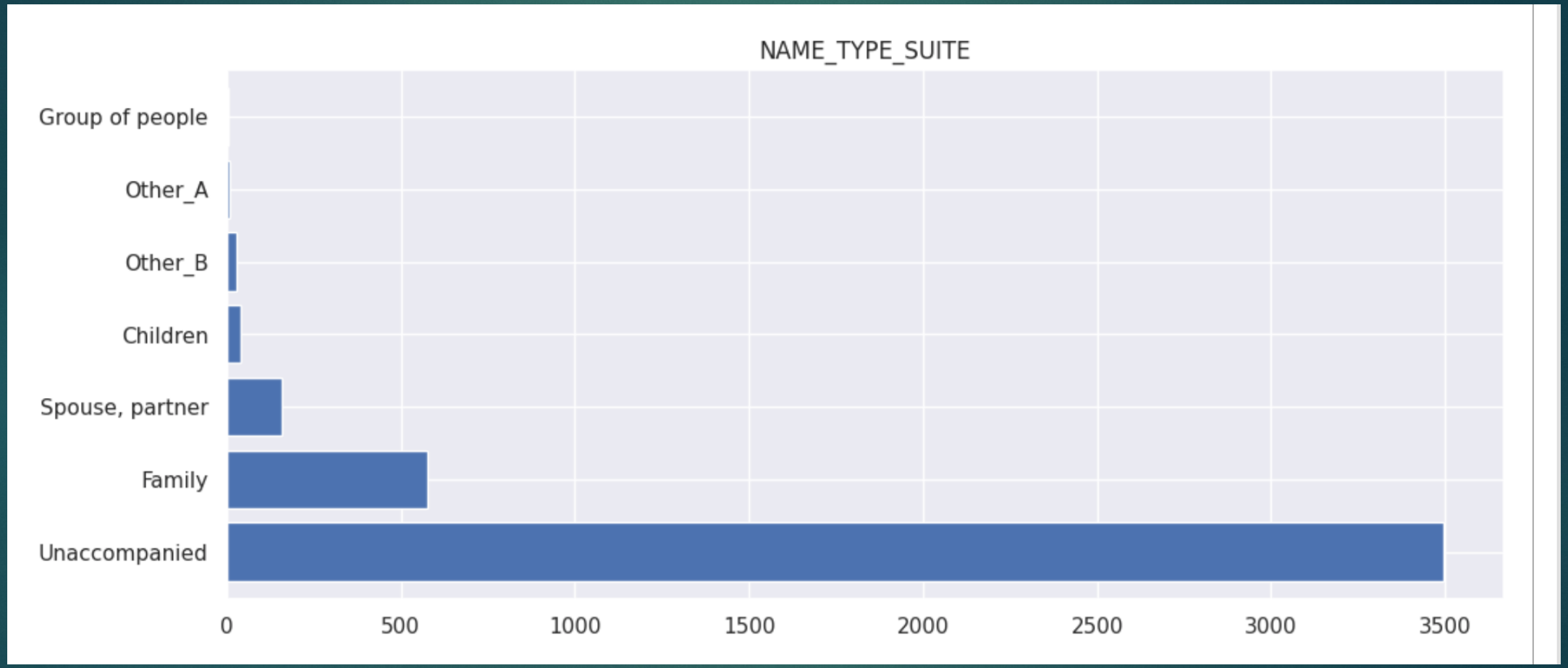

Box Plot
1
2
3
4
5
6
7
8
9
CNT_FAM_MEMBERS

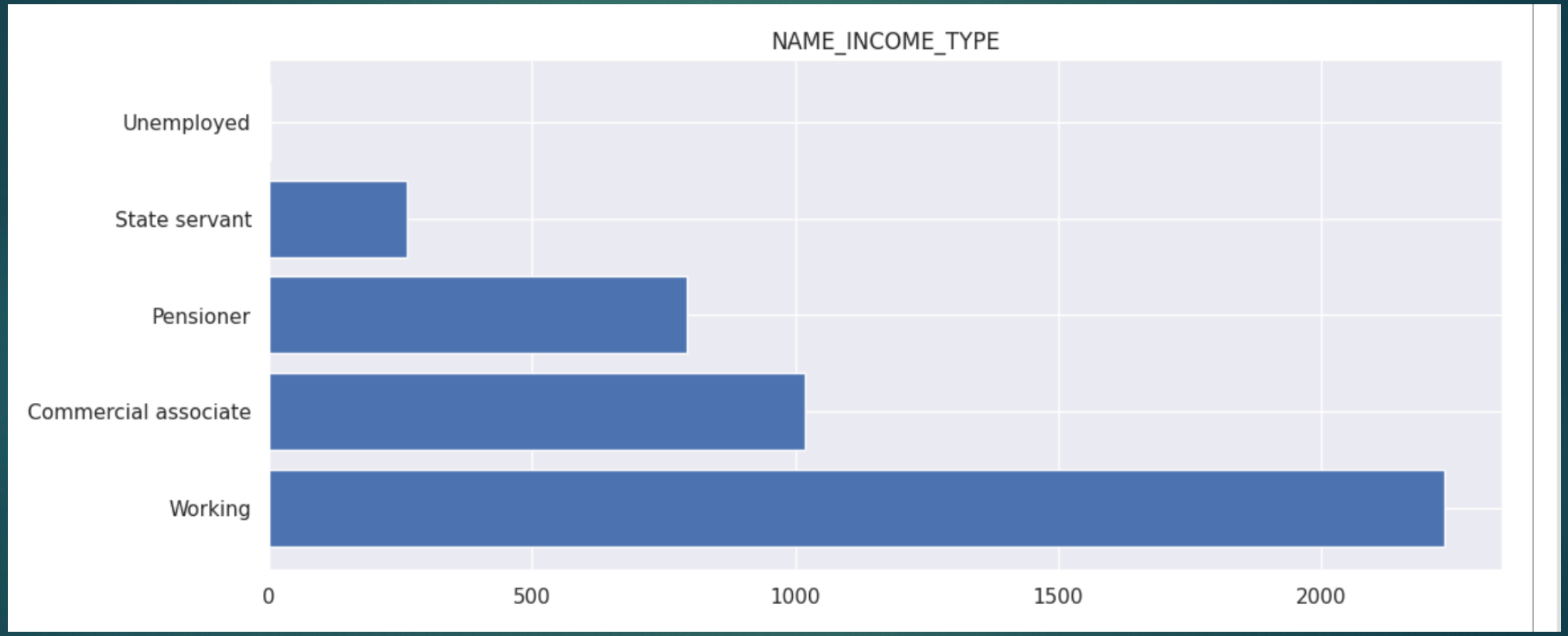

Distplot
Density
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
8
CNT_FAM_MEMBERS

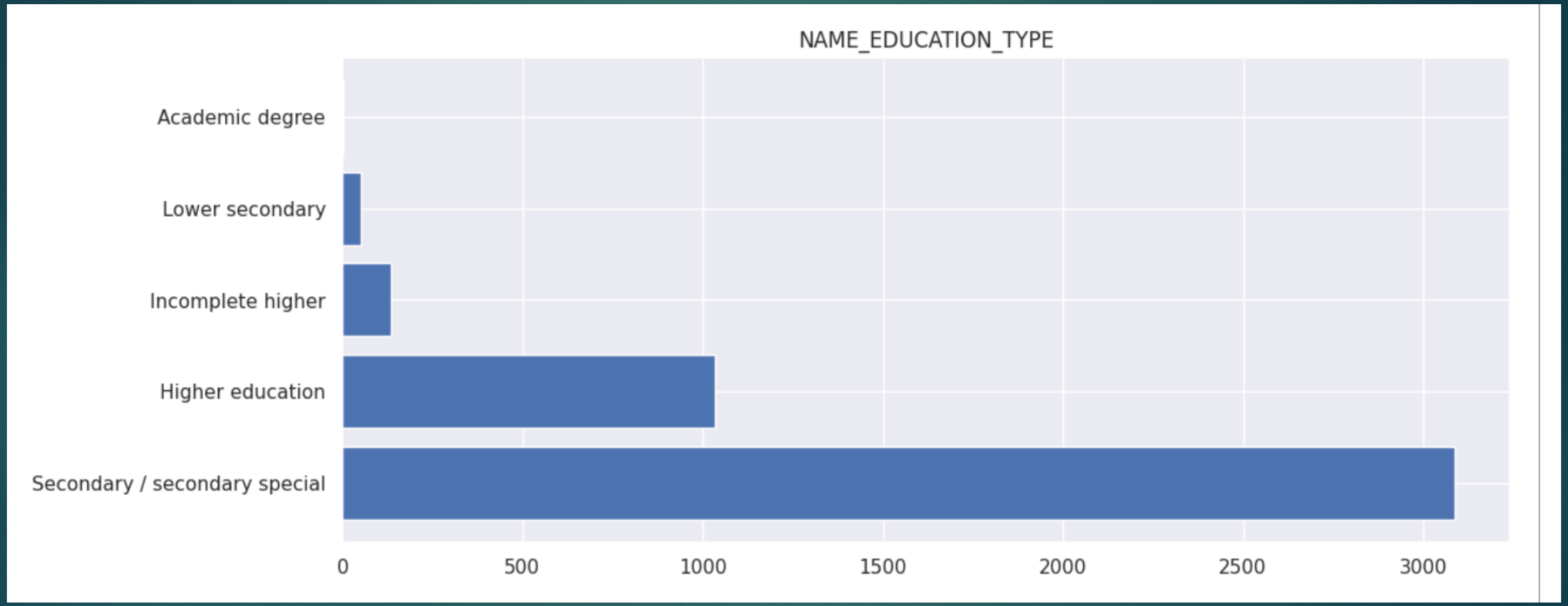
In [ ]:
```
for i in Categorical_Data:
  Uni_Analysis_Categorcal(pd_application,i)
```

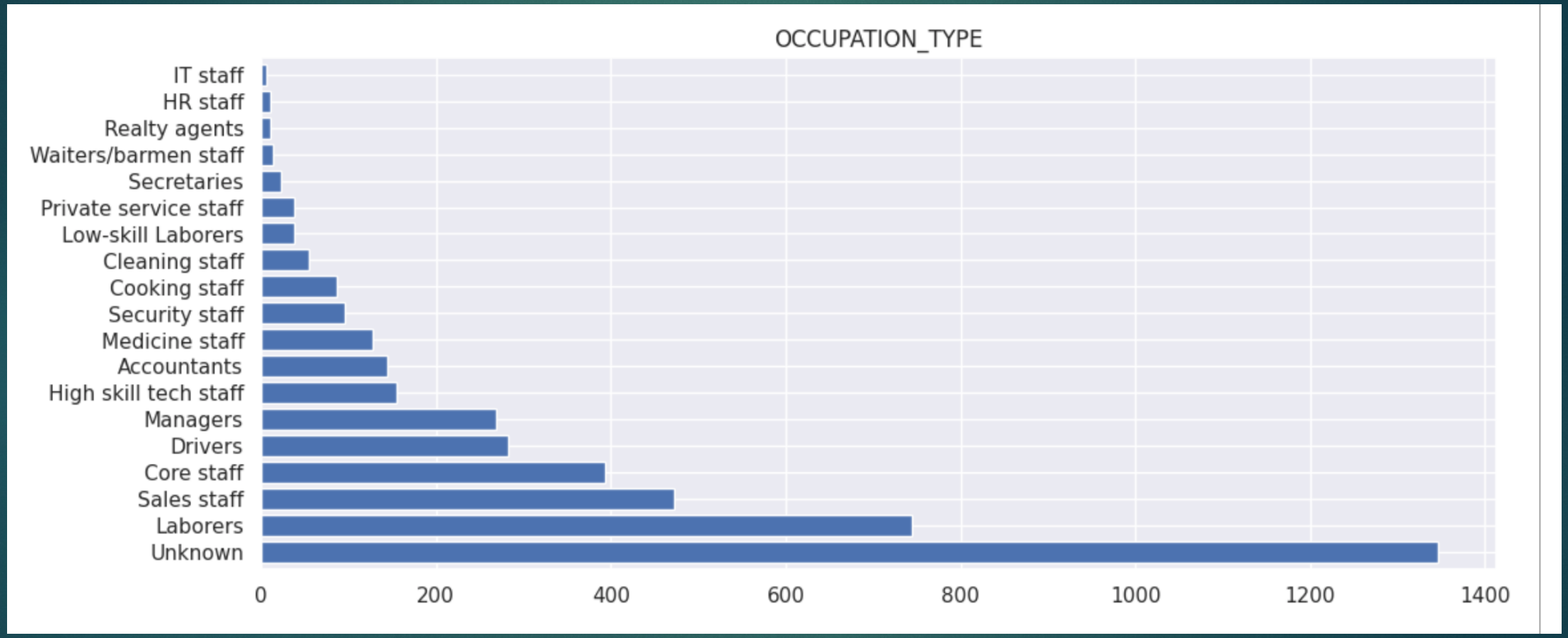


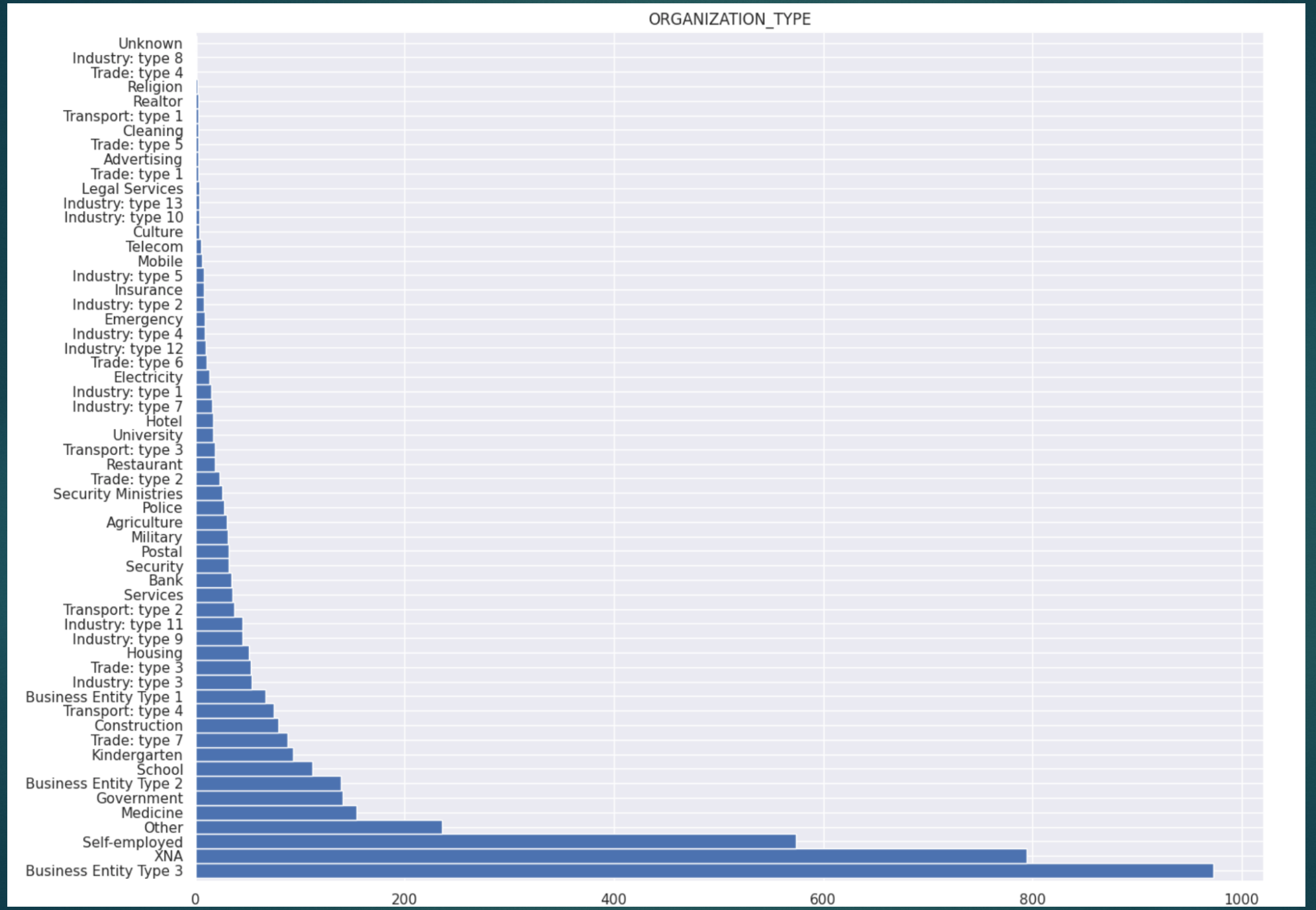

FLAG_OWN_REALTY
N
Y
0
500
1000
1500
2000
2500
3000


NAME_TYPE_SUITE
Group of people
Other_A
Other_B
Children
Spouse, partner
Family
Unaccompanied
0
500
1000
1500
2000
2500
3000
3500


NAME_INCOME_TYPE
Unemployed
State servant
Pensioner
Commercial associate
Working
0
500
1000
1500
2000

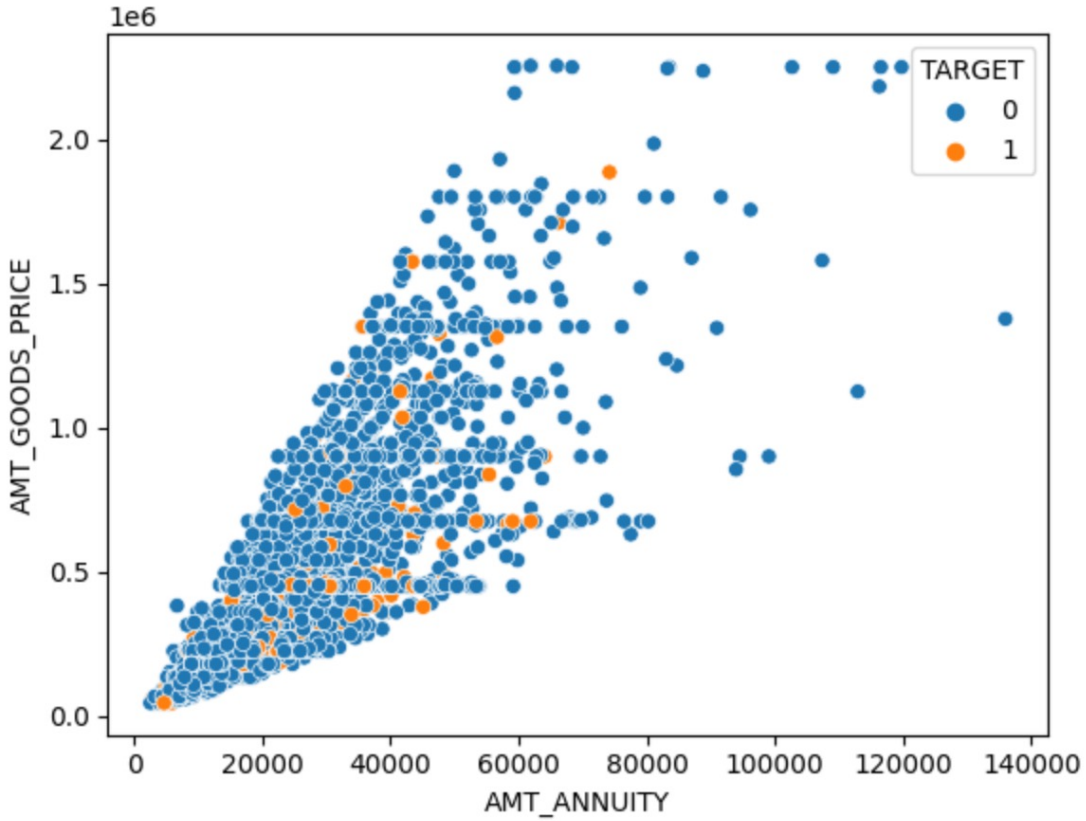

NAME_EDUCATION_TYPE
Academic degree
Lower secondary
Incomplete higher
Higher education
Secondary / secondary special
0
500
1000
1500
2000
2500
3000


OCCUPATION_TYPE
IT staff
HR staff
Realty agents
Waiters/barmen staff
Secretaries
Private service staff
Low-skill Laborers
Cleaning staff
Cooking staff
Security staff
Medicine staff
Accountants
High skill tech staff
Managers
Drivers
Core staff
Sales staff
Laborers
Unknown
0
200
400
600
800
1000
1200
1400


ORGANIZATION_TYPE
Unknown
Industry: type 8
Trade: type 4
Religion
Realtor
Transport: type 1
Cleaning
Trade: type 5
Advertising
Trade: type 1
Legal Services
Industry: type 13
Industry: type 10
Culture
Telecom
Mobile
Industry: type 5
Insurance
Industry: type 2
Emergency
Industry: type 4
Industry: type 12
Trade: type 6
Electricity
Industry: type 1
Industry: type 7
Hotel
University
Transport: type 3
Restaurant
Trade: type 2
Security Ministries
Police
Agriculture
Military
Postal
Security
Bank
Services
Transport: type 2
Industry: type 11
Industry: type 9
Housing
Trade: type 3
Industry: type 3
Business Entity Type 1
Transport: type 4
Construction
Trade: type 7
Kindergarten
School
Business Entity Type 2
Government
Medicine
Other
Self-employed
XNA
Business Entity Type 3
0
200
400
600
800
1000

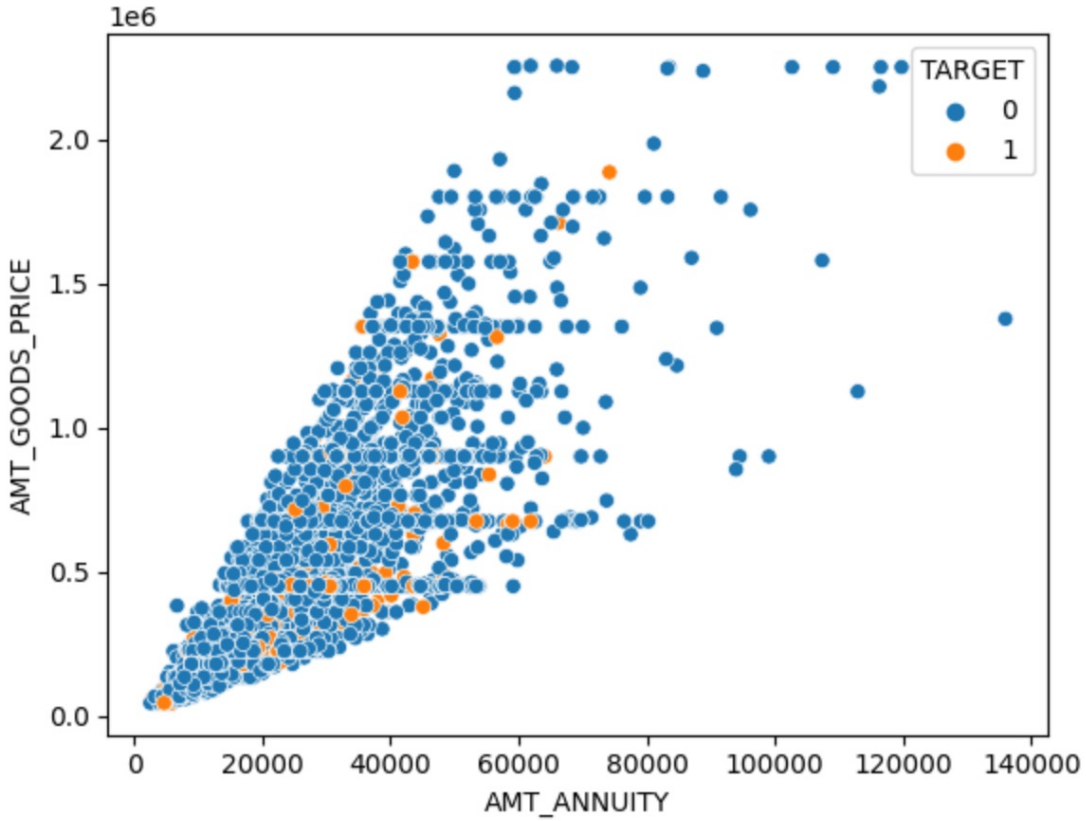
```
In [7]: sns.scatterplot(x=pd_application['AMT_ANNUITY'], y = pd_application['AMT_GOODS_PRICE'], data=pd_application,hue = pd
```

```
Out[7]: <Axes: xlabel='AMT_ANNUITY', ylabel='AMT_GOODS_PRICE'>
```

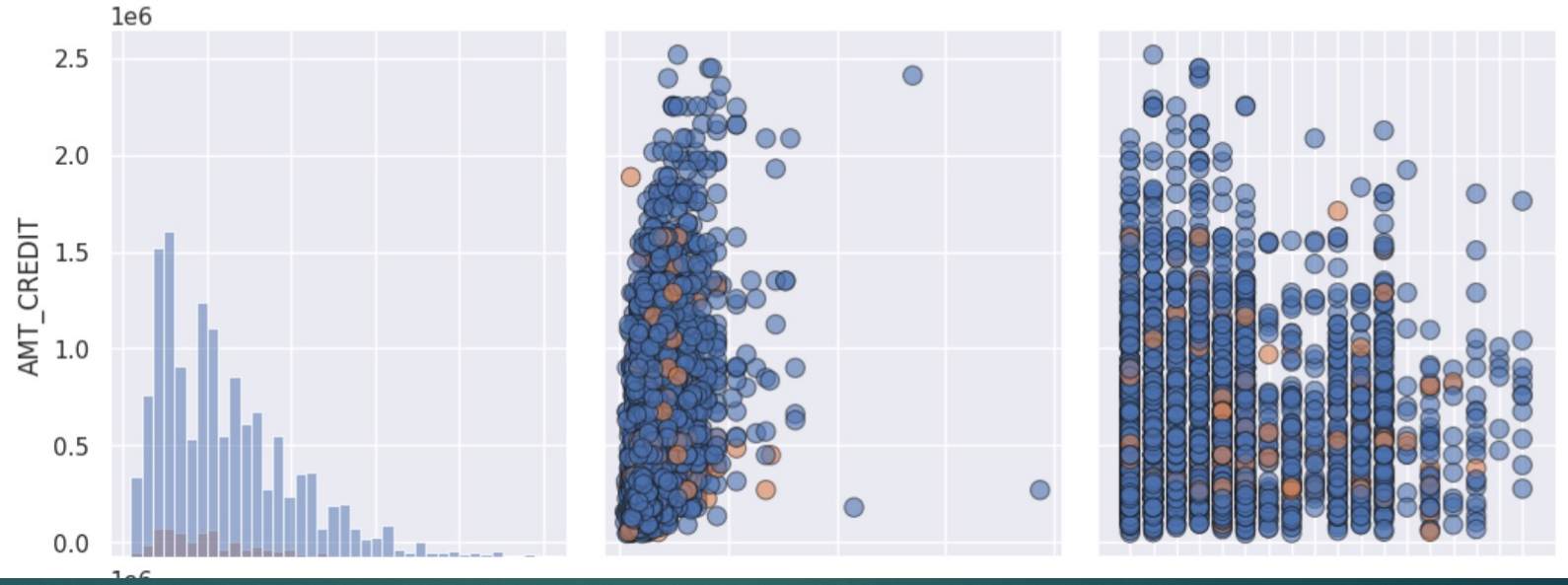


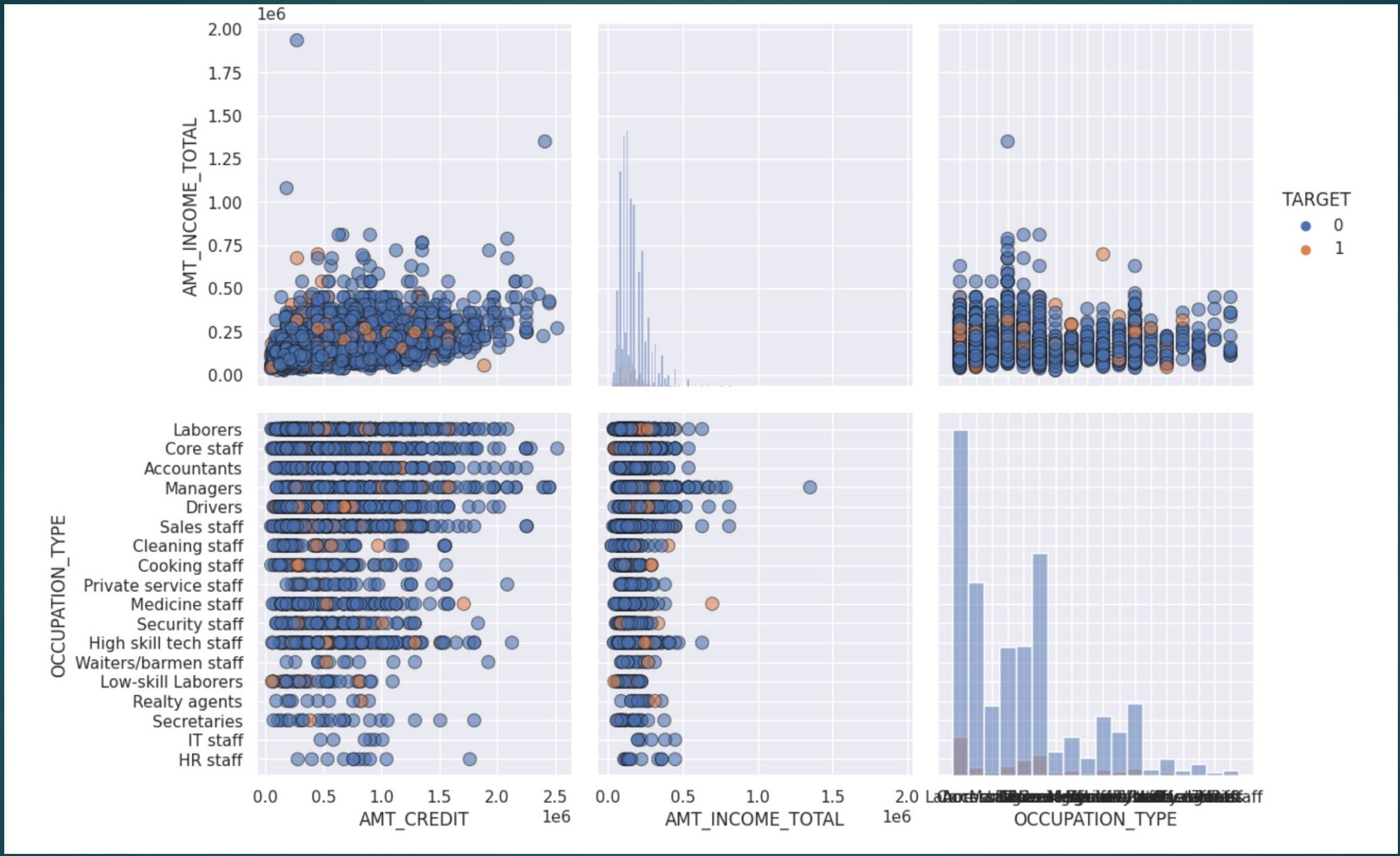
```
In [7]: sns.scatterplot(x=pd_application['AMT_ANNUITY'], y = pd_application['AMT_GOODS_PRICE'], data=pd_application,hue = pd
```

```
Out[7]: <Axes: xlabel='AMT_ANNUITY', ylabel='AMT_GOODS_PRICE'>
```

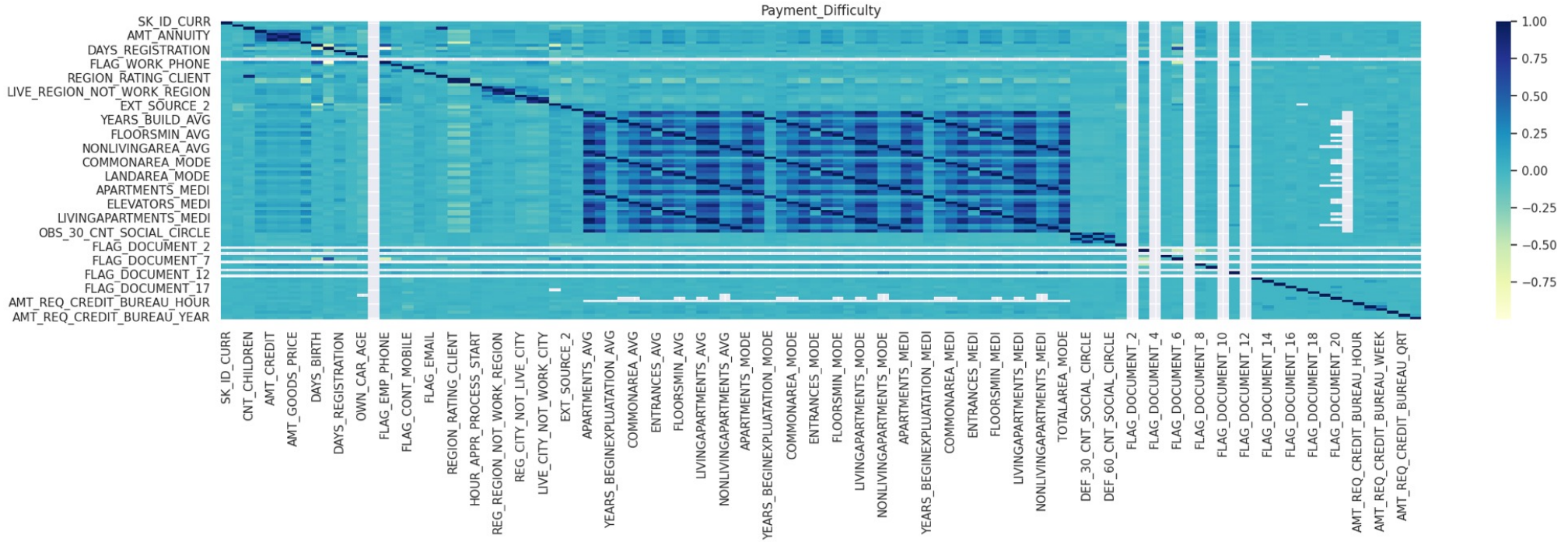


In [12]:
```
sns.pairplot(pd_application,vars = ['AMT_CREDIT','AMT_INCOME_TOTAL','OCCUPATION_TYPE'],diag_kind = 'hist', hue = 'TA
plt.xticks(rotation=90)
plt.show()
```




1e6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
AMT_INCOME_TOTAL
TARGET
0
1
Laborers
Core staff
Accountants
Managers
Drivers
Sales staff
Cleaning staff
Cooking staff
Private service staff
Medicine staff
Security staff
High skill tech staff
Waiters/barmen staff
Low-skill Laborers
Realty agents
Secretaries
IT staff
HR staff
OCCUPATION_TYPE
0.0
0.5
1.0
1.5
2.0
2.5
1e6
AMT_CREDIT
0.0
0.5
1.0
1.5
2.0
1e6
AMT_INCOME_TOTAL
OCCUPATION_TYPE

In [13]:
```
plt.figure(figsize=(25, 5))
sns.heatmap(pd_application.corr(),cmap="YlGnBu")
plt.title('Payment_Difficulty')
plt.show()
```



THANK YOU