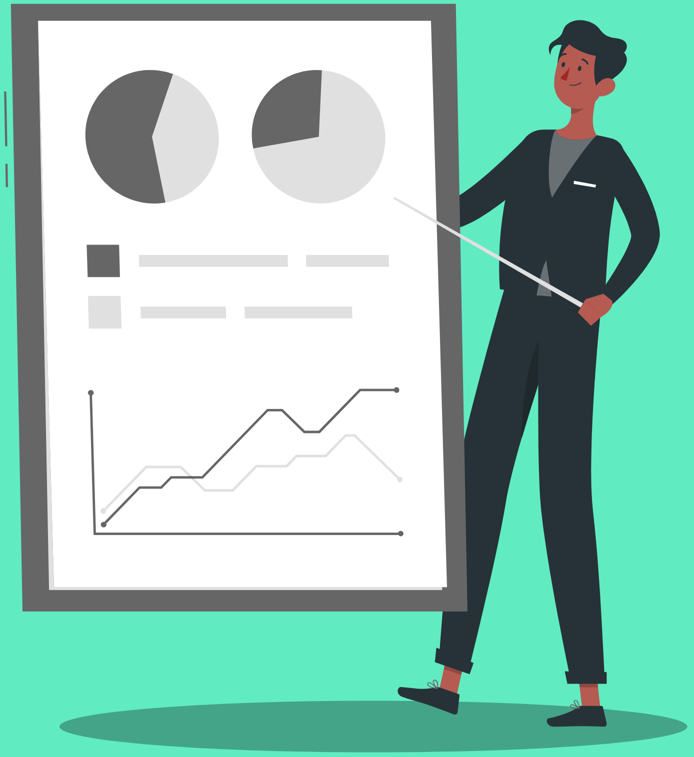


Bank Loan Case Study Report

By Yash Shinde

Jupyter Notebook:- Click Here to Download

Video Presentation:- Click Here to Watch Video Presentation

Project Description!

In this data analytics project, as a data analyst at a finance company specializing in urban loans, my goal is to tackle the challenge of loan default among customers with limited credit history. Through Exploratory Data Analysis (EDA), I will explore the dataset to identify missing data and apply appropriate methods to handle it effectively. My comprehensive EDA will play an important role in enhancing lending decisions and reducing default risks for the company.

Table of Contents

"Unveiling insights through data analysis to enhance lending decisions and mitigate loan default risks."

Tech Stack Used :-

-> Jupyter Notebook
-> PowerPoint
-> Github
-> Google Drive

Importing and Exploring Dataset

Importing and Exploring Dataset

- First, I created a new notebook and imported libraries and dataset.
- application_data = pd.read_csv("application_data.csv")
- Then I explored the dataset using the following programs.
- To see the five rows from top and bottom I used head() and tail() respectively.
- application_data.head()
- application_data.tail()
- Then by Using the column function see the names of all the columns/variables.
- application_data.columns
- Then by using the shape function observe the shape of the DataSet.
- application_data.shape
- By using the describe function obtained all descriptive stats for the numeric variables in the DataSet.
- application_data.describe()
- Same Operations are performed on the previous_application dataset.

```
In [11        prev ous_applic tion.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 49999 entries, 0 to 49998
         Data columns (total 37 columns):
          #   Column                       Non-Null Count  Dtype
         ---  ------                       --------------  -----
          0   SK_ID_PREV                   49999 non-null  int64
          1   SK_ID_CURR                   49999 non-null  int64
          2   NAME_CONTRACT_TYPE           49999 non-null  object
          3   AMT_ANNUITY                  39407 non-null  float64
          4   AMT_APPLICATION              49999 non-null  float64
          5   AMT_CREDIT                   49999 non-null  float64
          6   AMT_DOWN_PAYMENT             24801 non-null  float64
          7   AMT_GOODS_PRICE              39255 non-null  float64
          8   WEEKDAY_APPR_PROCESS_START   49999 non-null  object
          9   HOUR_APPR_PROCESS_START      49999 non-null  int64
          10  FLAG_LAST_APPL_PER_CONTRACT  49999 non-null  object
          11  NFLAG_LAST_APPL_IN_DAY       49999 non-null  int64
          12  RATE_DOWN_PAYMENT            24801 non-null  float64
          13  RATE_INTEREST_PRIMARY        165 non-null    float64
          14  RATE_INTEREST_PRIVILEGED     165 non-null    float64
          15  NAME_CASH_LOAN_PURPOSE       49999 non-null  object
          16  NAME_CONTRACT_STATUS         49999 non-null  object
          17  DAYS_DECISION                49999 non-null  int64
          18  NAME_PAYMENT_TYPE            49999 non-null  object
          19  CODE_REJECT_REASON           49999 non-null  object
          20  NAME_TYPE_SUITE              25756 non-null  object
          21  NAME_CLIENT_TYPE             49999 non-null  object
          22  NAME_GOODS_CATEGORY          49999 non-null  object
          23  NAME_PORTFOLIO               49999 non-null  object
          24  NAME_PRODUCT_TYPE            49999 non-null  object
          25  CHANNEL_TYPE                 49999 non-null  object
          26  SELLERPLACE_AREA             49999 non-null  int64
          27  NAME_SELLER_INDUSTRY         49999 non-null  object
          28  CNT_PAYMENT                  39407 non-null  float64
          29  NAME_YIELD_GROUP             49999 non-null  object
          30  PRODUCT_COMBINATION          49991 non-null  object
          31  DAYS_FIRST_DRAWING           30839 non-null  float64
          32  DAYS_FIRST_DUE               30839 non-null  float64
          33  DAYS_LAST_DUE_1ST_VERSION    30839 non-null  float64
          34  DAYS_LAST_DUE                30839 non-null  float64
          35  DAYS_TERMINATION             30839 non-null  float64
          36  NFLAG_INSURED_ON_APPROVAL    30839 non-null  float64
         dtypes: float64(15), int64(6), object(16)
         memory usage: 14.1+ MB
```

```
appl cation_dat .columns

Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY',
       ...
       'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20',
       'FLAG_DOCUMENT_21', 'AMT_REQ_CREDIT_BUREAU_HOUR',
       'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
       'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT',
       'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object', length=122)
```

```
1  application_data.shape
```

(49999, 122)

```
1  application_data.describe()
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT
count	49999.000000	49999.000000	49999.000000	4.999900e+04	4.999900e+04	499
mean	129013.210584	0.080522	0.419848	1.707676e+05	5.997006e+05	271
std	16690.512048	0.272102	0.724039	5.318191e+05	4.024154e+05	145
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	20
25%	114570.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	164
50%	129076.000000	0.000000	0.000000	1.458000e+05	5.147775e+05	249
75%	143438.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	345
max	157875.000000	1.000000	11.000000	1.170000e+08	4.050000e+06	2580

8 rows × 106 columns

A

Managing Missing Data

Identify Missing Data and Deal with it Appropriately

Managing Missing Data

- First, I calculated the blank cells/Null values in the dataset.
- a = application_data.isnull().sum()
- Then I Created a bar chart to visualize the proportion of missing values for each variable.
- Then I used following code/program to find the columns/variables having more than 50% missing values.
- a = a[a>(len(application_data)*0.5)]
- We don't need this information for our analysis thus removing those columns using the drop function.
- application_data = application_data.drop(columns=a.index)
- Then I performed observation on remaining values and replaced the remaining null values with appropriate values using the following code.
- filldict = dict(zip(keys, values))
- application_data.fillna(value=filldict, inplace=True)application_data.tail()
- Then by Using the duplicated function remove the duplicate rows from the dataset.
- application_data.duplicated().sum()
- Some variables have negative values converted those value into posivtive using the abs() function for better analysis.
- application_data[['DAYS_BIRTH' ,'DAYS_EMPLOYED' ,'DAYS_REGISTRATION' ,'DAYS_ID_PUBLISH']] = application_data[['DAYS_BIRTH' ,'DAYS_EMPLOYED' ,'DAYS_REGISTRATION' ,'DAYS_ID_PUBLISH']].abs()
- In this way I Identified Missing values and Handled them.

In [16]:

```python
a = application_data.isnull().sum()
a
```

Out[16]:

```
SK_ID_CURR                     0
TARGET                         0
NAME_CONTRACT_TYPE             0
CODE_GENDER                    0
FLAG_OWN_CAR                   0
                            ...
AMT_REQ_CREDIT_BUREAU_DAY   6734
AMT_REQ_CREDIT_BUREAU_WEEK  6734
AMT_REQ_CREDIT_BUREAU_MON   6734
AMT_REQ_CREDIT_BUREAU_QRT   6734
AMT_REQ_CREDIT_BUREAU_YEAR  6734
Length: 122, dtype: int64
```

Observation

1) AMT_REQ_CREDIT_BUREAU_HOUR/DAY/WEEK/MON/QRT/YEAR has the Number of enquiries to Credit Bureau about the client before application thus null values in this columns mean that there are 0 enquires thus filling null values in this columns with 0.

2) NAME_TYPE_SUITE has null values where it is not provided by applicant but if it is not porvided then it should be "Unaccompanied".

3) OCCUPATION_TYPE has null values where it is not provied by applicant let's replace it with "Unknown".

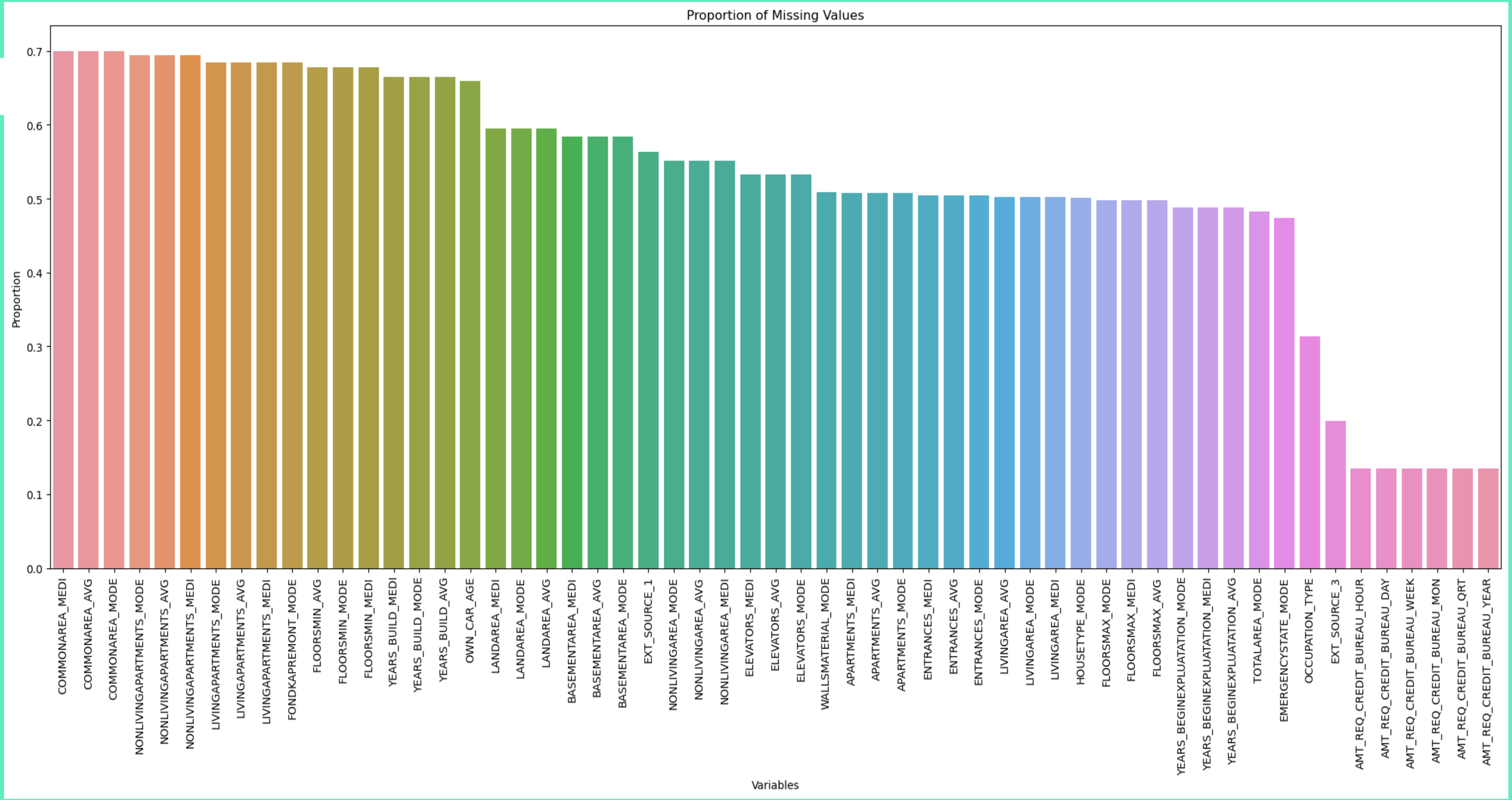
4) CNT_FAM_MEMBERS has 1 null value also its family_details are unknown so it is safe to replce its value to 0

```python
a = a[a>(len(application_data)*0.5)]
print(a)
len(a)
```

```
OWN_CAR_AGE                    32950
EXT_SOURCE_1                   28172
APARTMENTS_AVG                 25385
BASEMENTAREA_AVG               29199
YEARS_BUILD_AVG                33239
COMMONAREA_AVG                 34960
ELEVATORS_AVG                  26651
ENTRANCES_AVG                  25195
FLOORSMIN_AVG                  33894
LANDAREA_AVG                   29721
LIVINGAPARTMENTS_AVG           34226
LIVINGAREA_AVG                 25137
NONLIVINGAPARTMENTS_AVG        34714
NONLIVINGAREA_AVG              27572
APARTMENTS_MODE                25385
BASEMENTAREA_MODE              29199
YEARS_BUILD_MODE               33239
COMMONAREA_MODE                34960
ELEVATORS_MODE                 26651
ENTRANCES_MODE                 25195
FLOORSMIN_MODE                 33894
LANDAREA_MODE                  29721
LIVINGAPARTMENTS_MODE          34226
LIVINGAREA_MODE                25137
NONLIVINGAPARTMENTS_MODE       34714
NONLIVINGAREA_MODE             27572
APARTMENTS_MEDI                25385
BASEMENTAREA_MEDI              29199
YEARS_BUILD_MEDI               33239
COMMONAREA_MEDI                34960
ELEVATORS_MEDI                 26651
ENTRANCES_MEDI                 25195
FLOORSMIN_MEDI                 33894
LANDAREA_MEDI                  29721
LIVINGAPARTMENTS_MEDI          34226
LIVINGAREA_MEDI                25137
NONLIVINGAPARTMENTS_MEDI       34714
NONLIVINGAREA_MEDI             27572
FONDKAPREMONT_MODE             34191
HOUSETYPE_MODE                 25075
WALLSMATERIAL_MODE             25459
dtype: int64
```

Creating bar chart to visualize the proportion of missing values for each variable.

```
# Calculate the proportion of missing values for columns having missing va
mean_val = application_data.isnull().mean()

# Sorting the mean_val for better visualization
mean_val_sorted = mean_val.sort_values(ascending=False)

missing_var_mean = mean_val_sorted[mean_val_sorted>0.1]

# Create a bar chart using seaborn
plt.figure(figsize=(20, 10))
sns.barplot(x=missing_var_mean.index, y=missing_var_mean.values)
plt.title('Proportion of Missing Values')
plt.xlabel('Variables')
plt.ylabel('Proportion')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```


Proportion of Missing Values
Proportion
Variables
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
COMMONAREA_MEDI
COMMONAREA_AVG
COMMONAREA_MODE
NONLIVINGAPARTMENTS_MODE
NONLIVINGAPARTMENTS_AVG
NONLIVINGAPARTMENTS_MEDI
LIVINGAPARTMENTS_MODE
LIVINGAPARTMENTS_AVG
LIVINGAPARTMENTS_MEDI
FONDKAPREMONT_MODE
FLOORSMIN_AVG
FLOORSMIN_MODE
FLOORSMIN_MEDI
YEARS_BUILD_MEDI
YEARS_BUILD_MODE
YEARS_BUILD_AVG
OWN_CAR_AGE
LANDAREA_MEDI
LANDAREA_MODE
LANDAREA_AVG
BASEMENTAREA_MEDI
BASEMENTAREA_AVG
BASEMENTAREA_MODE
EXT_SOURCE_1
NONLIVINGAREA_MODE
NONLIVINGAREA_AVG
NONLIVINGAREA_MEDI
ELEVATORS_MEDI
ELEVATORS_AVG
ELEVATORS_MODE
WALLSMATERIAL_MODE
APARTMENTS_MEDI
APARTMENTS_AVG
APARTMENTS_MODE
ENTRANCES_MEDI
ENTRANCES_AVG
ENTRANCES_MODE
LIVINGAREA_AVG
LIVINGAREA_MODE
LIVINGAREA_MEDI
HOUSETYPE_MODE
FLOORSMAX_MODE
FLOORSMAX_MEDI
FLOORSMAX_AVG
YEARS_BEGINEXPLUATATION_MODE
YEARS_BEGINEXPLUATATION_MEDI
YEARS_BEGINEXPLUATATION_AVG
TOTALAREA_MODE
EMERGENCYSTATE_MODE
OCCUPATION_TYPE
EXT_SOURCE_3
AMT_REQ_CREDIT_BUREAU_HOUR
AMT_REQ_CREDIT_BUREAU_DAY
AMT_REQ_CREDIT_BUREAU_WEEK
AMT_REQ_CREDIT_BUREAU_MON
AMT_REQ_CREDIT_BUREAU_QRT
AMT_REQ_CREDIT_BUREAU_YEAR

B

Detecting Data Outliers

Identify outliers in the loan application dataset.

Detecting Data Outliers

- First, I Selected numerical columns for the analysis of Outliers.
- numerical_columns = application_data.select_dtypes(include=[np.number]).columns
- We know that outliers represent data points that deviate significantly from the bulk of the data due to factors such as measurement inaccuracies and errors during data input.
- I used following steps to calculate Outliers

  Step 1: Calculate Inter Quartile Range (IQR) of the data.

  ```
  q1 = np.percentile(application_data[i], 25)
  q3 = np.percentile(application_data[i], 75)
  iqr = q3 - q1
  ```

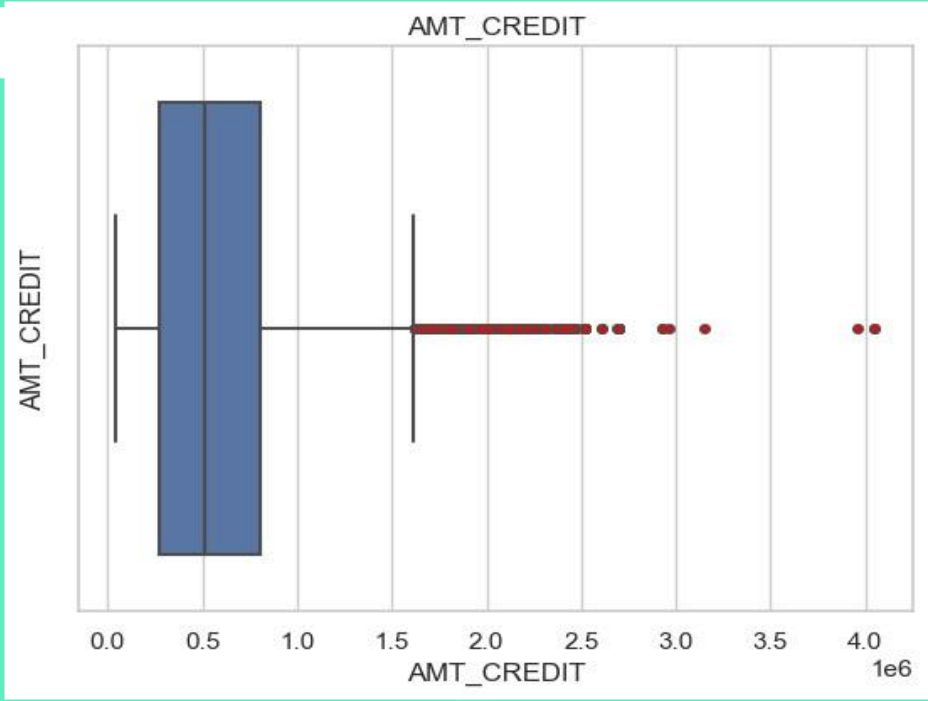
  Step 2: Define lower and upper bounds for outliers using calculated IQR

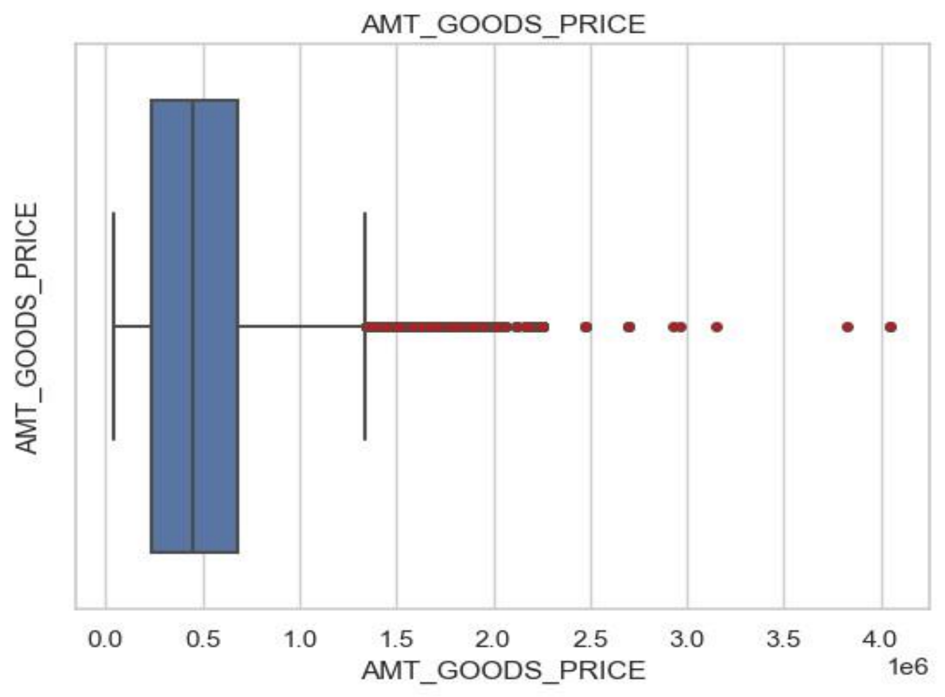
  ```
  lower_bound = q1 - 1.5 * iqr
  upper_bound = q3 + 1.5 * iqr
  ```

  Step 3: Using those find and store outliers in the dataset.

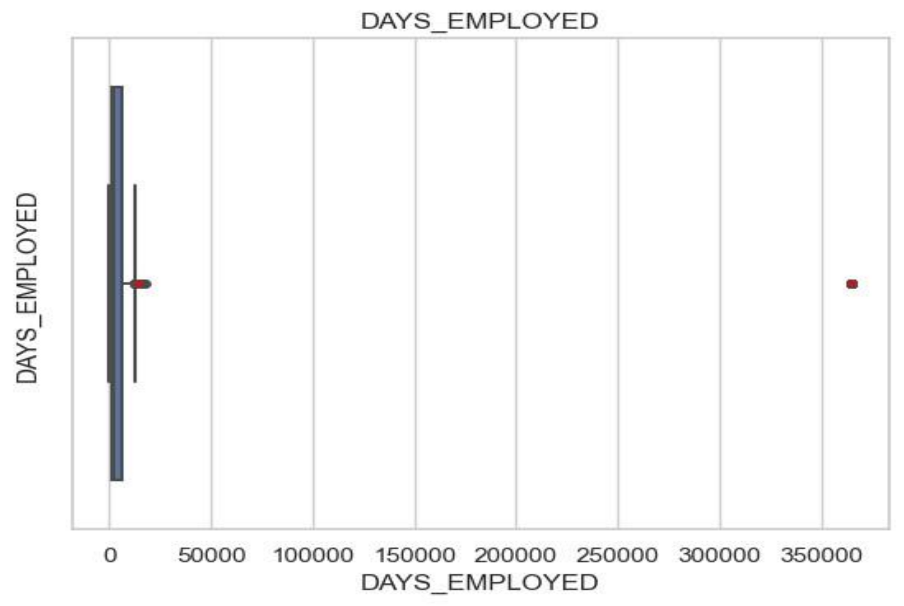
  ```
  outliers = [x for x in application_data[i] if x < lower_bound or x > upper_bound]
  dictoutlier[i] = outliers
  ```
- Then, Plot the box plot with variable name as title and outliers marked with red colour dots

Box Plots

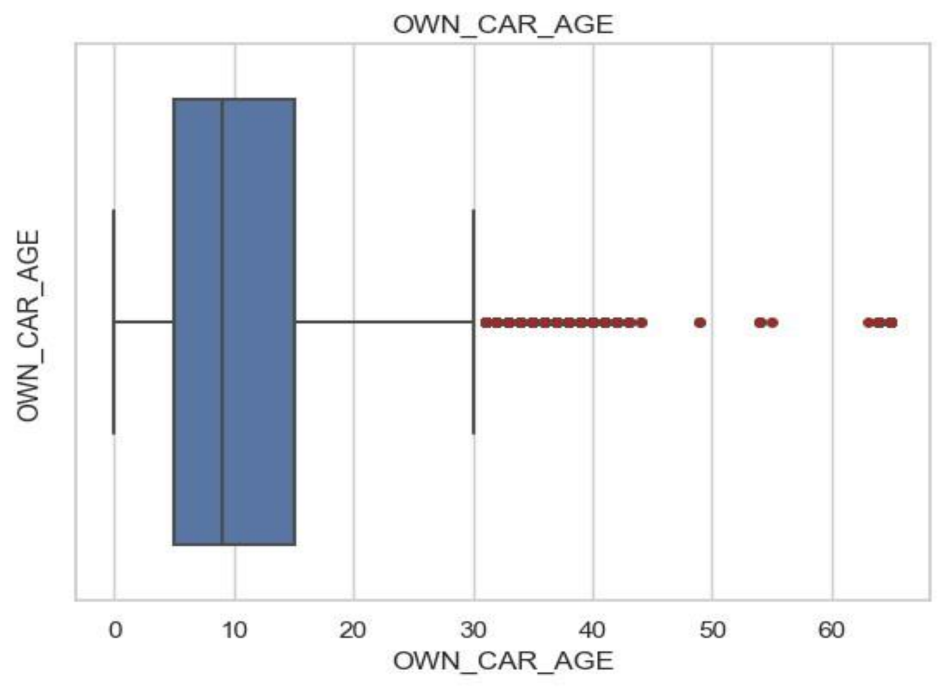




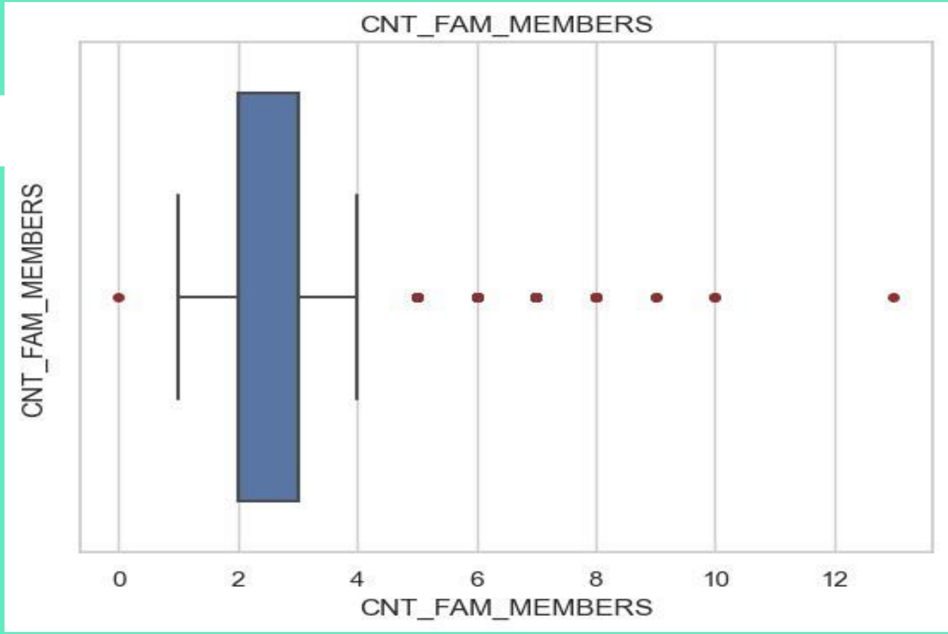
- AMT_CREDIT, AMT_GOODS_PRICE and AMT_ANNUITY have almost the same box plot which shows that all three variables are very closely related.
- As shown in the above Box Plots about 50% of data is inside the box part i.e. between 0.25-0.8.
- Outliers are marked with red colour. Which have a value of more than 1.5



- OWN_CAR_AGE variable has the age of the car owned by the applicants
- As shown in the figure most cars have age in the range of 5-15.
- Some cars between 30-60 are marked by red dots as outliers.

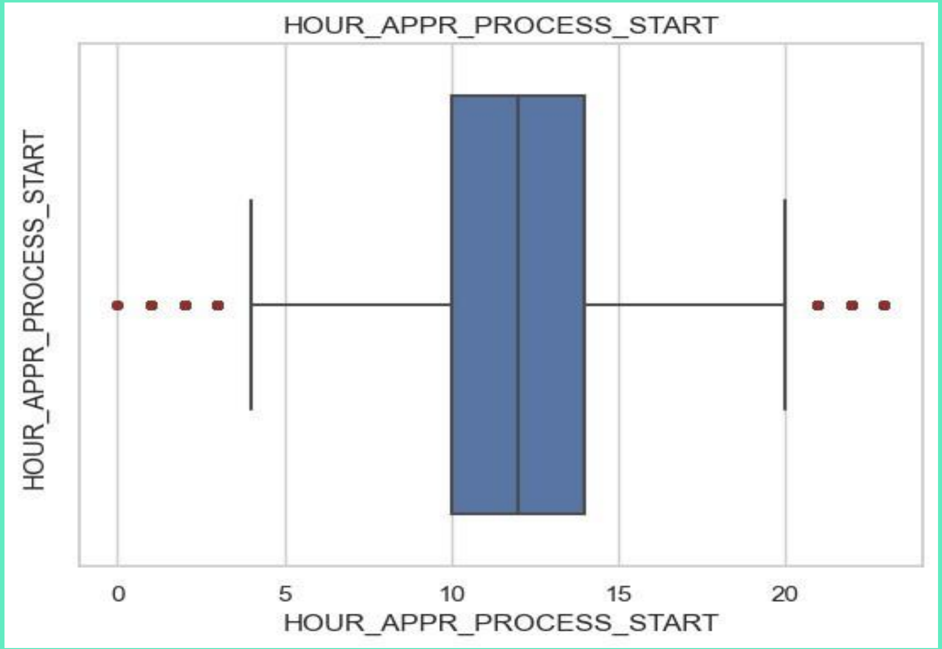


- DAYS_EMPLOYED variable has a value of the total number of days an applicant is employed.
- As shown in the figure most applicants have no of days employed between 0-15000 days.
- Some applicants have about 370000 days employed which is clearly seen in the box plot.
- Thus, they are outliers.



- HOUR_APPR_PROCESS_START variable stores Approximately at what hour did the client apply for the loan?
- As shown in the Box Plot most of the applicants applied between 10 am to 2 pm.
- Applicants who apply between 0 am to 4 am & 8 pm to 12 am are outliers.

- CNT_FAM_MEMBERS variable stores the total number of family members of the applicant.
- It shows that 50% of applicants have 2 or 3 family members.
- Some outliers which can be clearly seen in the box plot are 0,5,6,7,8,9,10 and 11



Total Outliers in each Variable

TARGET: 4026
CNT_CHILDREN: 723
AMT_INCOME_TOTAL: 2295
AMT_CREDIT: 1063
REGION_POPULATION_RELATIVE: 1329
DAYS_EMPLOYED: 9082
DAYS_REGISTRATION: 96
FLAG_MOBIL: 1
FLAG_EMP_PHONE: 8926
FLAG_WORK_PHONE: 9963
FLAG_CONT_MOBILE: 101
FLAG_EMAIL: 2783
CNT_FAM_MEMBERS: 684
REGION_RATING_CLIENT: 13035
REGION_RATING_CLIENT_W_CITY: 12658
HOUR_APPR_PROCESS_START: 353
REG_REGION_NOT_LIVE_REGION: 750
REG_REGION_NOT_WORK_REGION: 2496
LIVE_REGION_NOT_WORK_REGION: 1982
REG_CITY_NOT_LIVE_CITY: 3998
REG_CITY_NOT_WORK_CITY: 11608
LIVE_CITY_NOT_WORK_CITY: 8985
FLAG_DOCUMENT_2: 2
FLAG_DOCUMENT_4: 9
FLAG_DOCUMENT_5: 785
FLAG_DOCUMENT_6: 4335
FLAG_DOCUMENT_7: 11
FLAG_DOCUMENT_8: 4038
FLAG_DOCUMENT_9: 184
FLAG_DOCUMENT_10: 1
FLAG_DOCUMENT_11: 213
FLAG_DOCUMENT_13: 161
FLAG_DOCUMENT_14: 158
FLAG_DOCUMENT_15: 41
FLAG_DOCUMENT_16: 501
FLAG_DOCUMENT_17: 15
FLAG_DOCUMENT_18: 425
FLAG_DOCUMENT_19: 35
FLAG_DOCUMENT_20: 26
FLAG_DOCUMENT_21: 19
AMT_REQ_CREDIT_BUREAU_HOUR: 295
AMT_REQ_CREDIT_BUREAU_DAY: 272
AMT_REQ_CREDIT_BUREAU_WEEK: 1314
AMT_REQ_CREDIT_BUREAU_MON: 7140
AMT_REQ_CREDIT_BUREAU_QRT: 8134
AMT_REQ_CREDIT_BUREAU_YEAR: 552

Count of Unique Outliers

TARGET: 1
CNT_CHILDREN: 8
AMT_INCOME_TOTAL: 120
AMT_CREDIT: 262
REGION_POPULATION_RELATIVE: 1
DAYS_EMPLOYED: 151
DAYS_REGISTRATION: 96
FLAG_MOBIL: 1
FLAG_EMP_PHONE: 1
FLAG_WORK_PHONE: 1
FLAG_CONT_MOBILE: 1
FLAG_EMAIL: 1
CNT_FAM_MEMBERS: 8
REGION_RATING_CLIENT: 2
REGION_RATING_CLIENT_W_CITY: 2
HOUR_APPR_PROCESS_START: 7
REG_REGION_NOT_LIVE_REGION: 1
REG_REGION_NOT_WORK_REGION: 1
LIVE_REGION_NOT_WORK_REGION: 1
REG_CITY_NOT_LIVE_CITY: 1
REG_CITY_NOT_WORK_CITY: 1
LIVE_CITY_NOT_WORK_CITY: 1
FLAG_DOCUMENT_2: 1
FLAG_DOCUMENT_4: 1
FLAG_DOCUMENT_5: 1
FLAG_DOCUMENT_6: 1
FLAG_DOCUMENT_7: 1
FLAG_DOCUMENT_8: 1
FLAG_DOCUMENT_9: 1
FLAG_DOCUMENT_10: 1
FLAG_DOCUMENT_11: 1
FLAG_DOCUMENT_13: 1
FLAG_DOCUMENT_14: 1
FLAG_DOCUMENT_15: 1
FLAG_DOCUMENT_16: 1
FLAG_DOCUMENT_17: 1
FLAG_DOCUMENT_18: 1
FLAG_DOCUMENT_19: 1
FLAG_DOCUMENT_20: 1
FLAG_DOCUMENT_21: 1
AMT_REQ_CREDIT_BUREAU_HOUR: 3
AMT_REQ_CREDIT_BUREAU_DAY: 6
AMT_REQ_CREDIT_BUREAU_WEEK: 6
AMT_REQ_CREDIT_BUREAU_MON: 19
AMT_REQ_CREDIT_BUREAU_QRT: 8
AMT_REQ_CREDIT_BUREAU_YEAR: 11

C

Analyze Data Imbalance

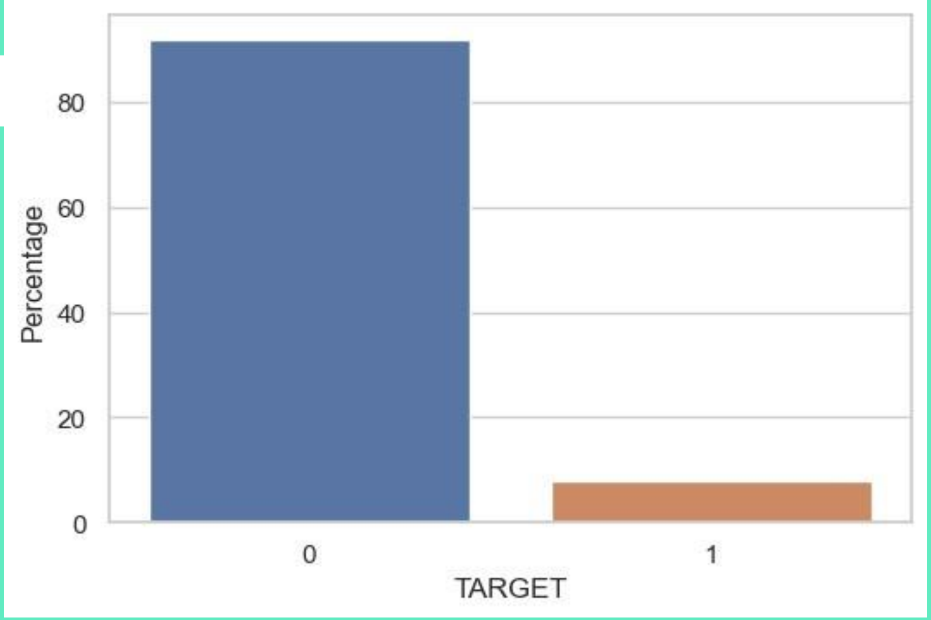
Determine if there is a data imbalance in the loan application dataset.

Analyze Data Imbalance

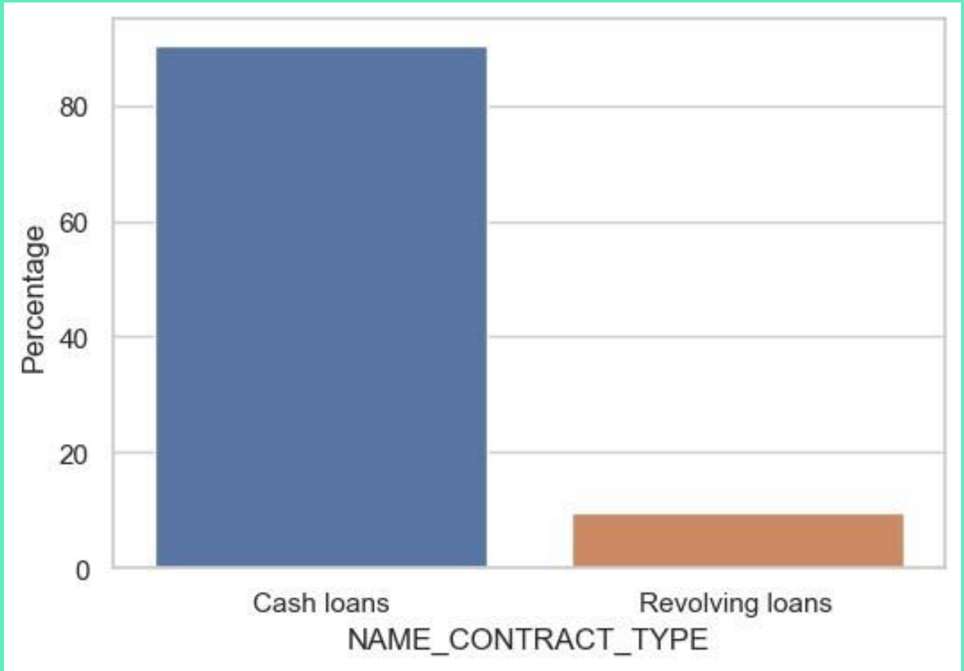
- First, I have Calculated the class frequencies i.e. unique values in each variable and their total count.
- class_frequencies = application_data[i].value_counts()
- Then Created a bar chart to visualize Data imbalance.
- Also, used the if condition to get only categorical variables.

```
if len(class_frequencies)<=3:

    # printing the percentage of each value_count in list
    percentage = np.around((class_frequencies.values)/len(application_
    print(percentage)

    # Create a bar chart to visualize Data imbalance.
    plt.figure(figsize=(6, 4))
    sns.barplot(x = class_frequencies.index, y = (class_frequencies.va
    plt.xlabel(i)
    plt.ylabel("Percentage")
    plt.show()
```

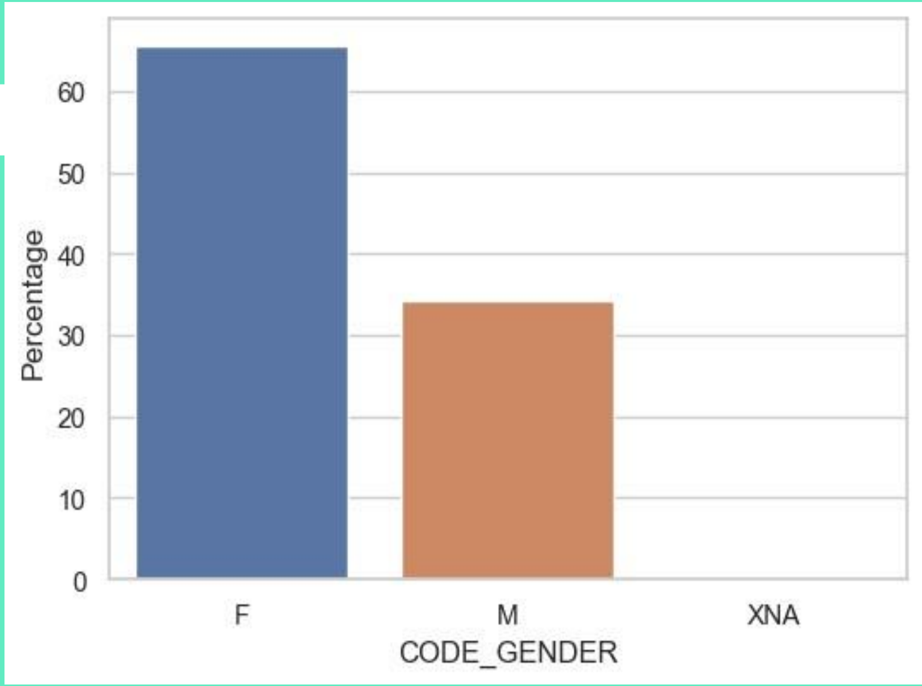
- In the same way also created a bar graph for Variables have class frequencies between 3-15.
- Tables and bar_graphs are shown below.



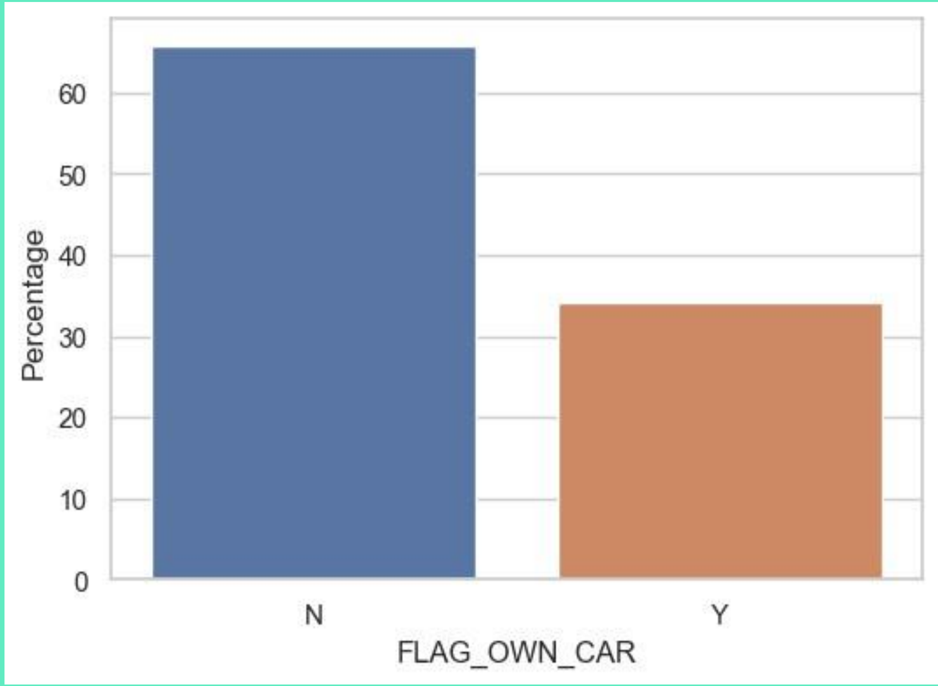
- NAME_CONTRACT_TYPE has the 91% of loans of Cash Loans type.



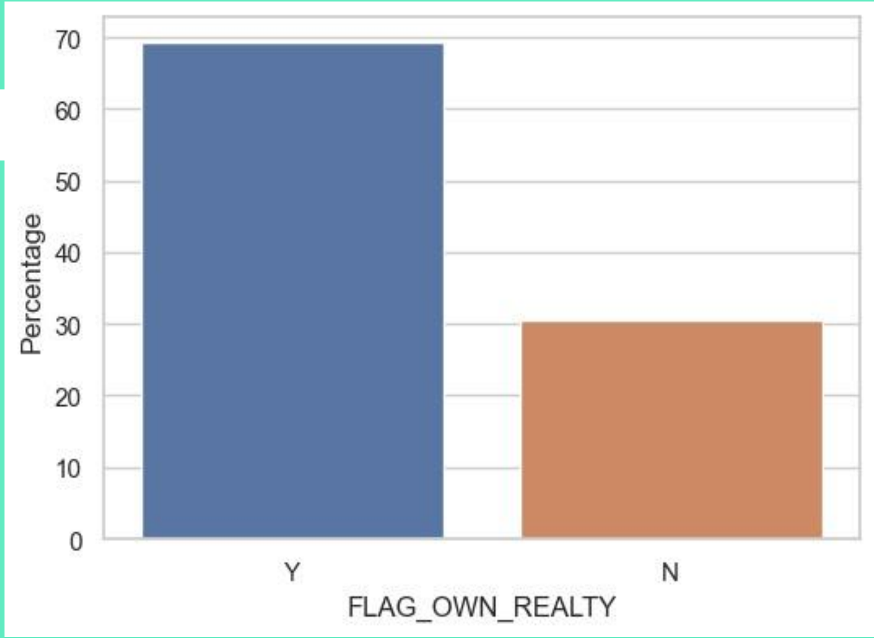
- Target variable shows us that 92% people have the value 1 i.e., clients with payment difficulties



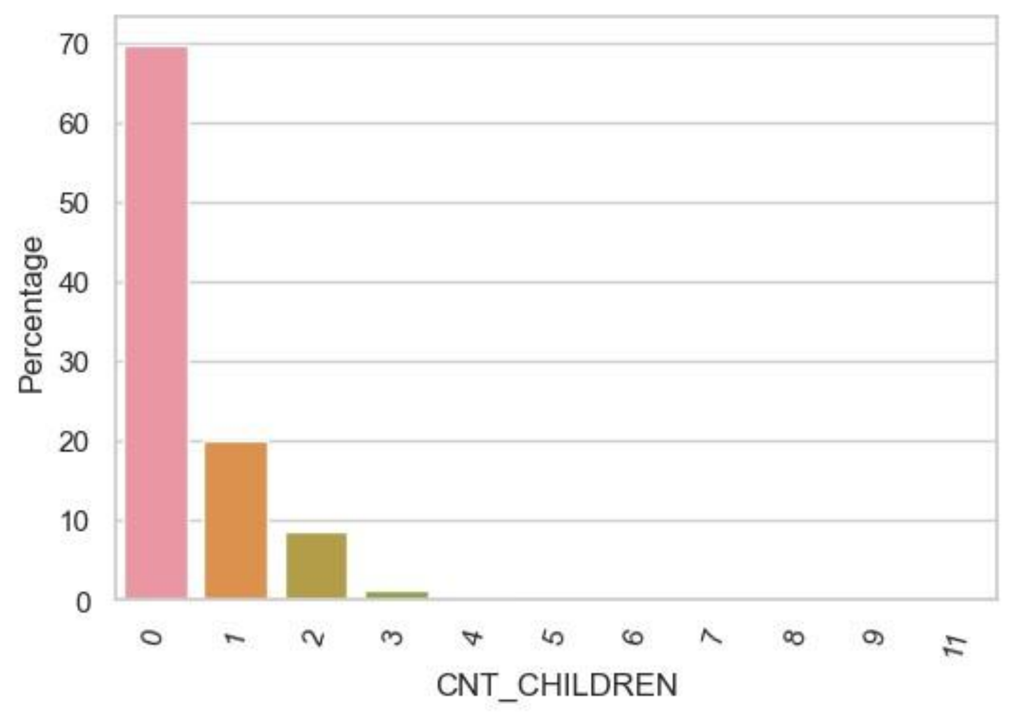
- FLAG_OWN_CAR data imbalance shows us that 66% applicants don't own car.



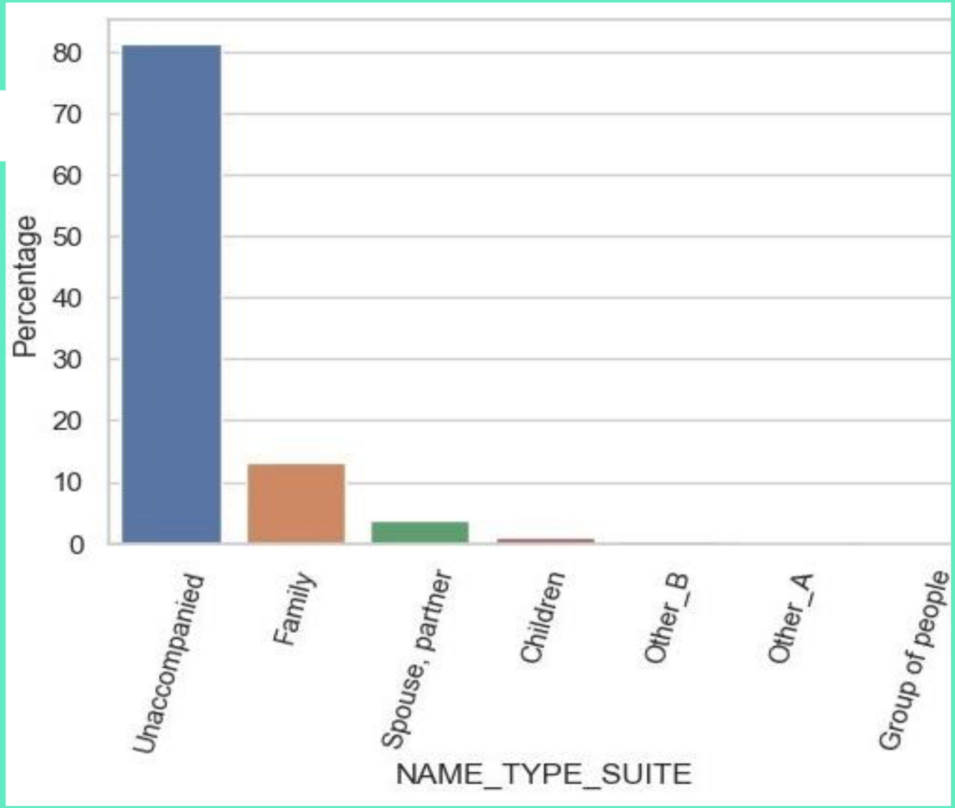
- CODE_GENDER data imbalance shows us that there are 66% female applicatns and 34% male applicants



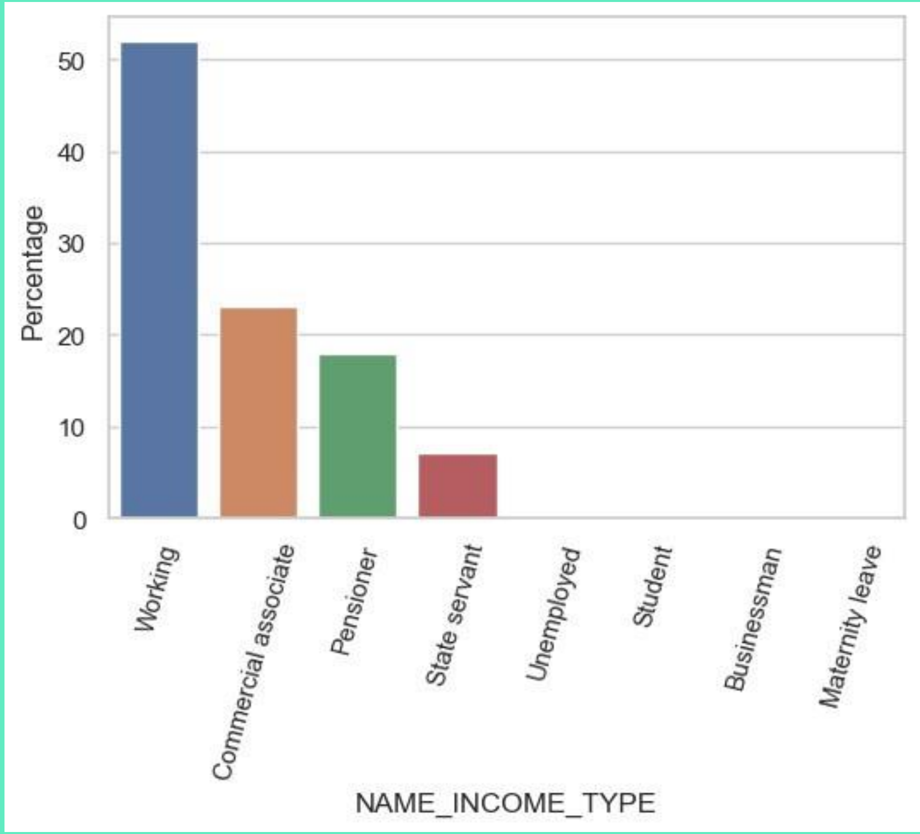
- CNT_CHILDREN data imbalance shows us that 70% applicants don't have childrens.



- FLAG_OWN_REALTY data imbalance shows us that 69% applicants own house.

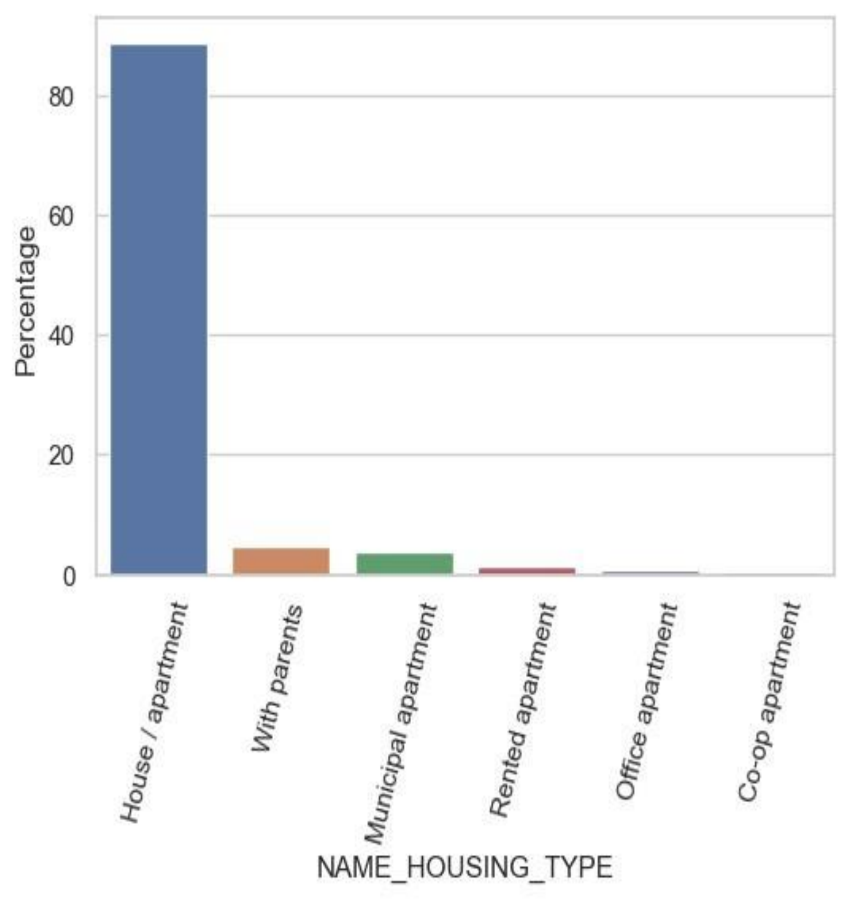


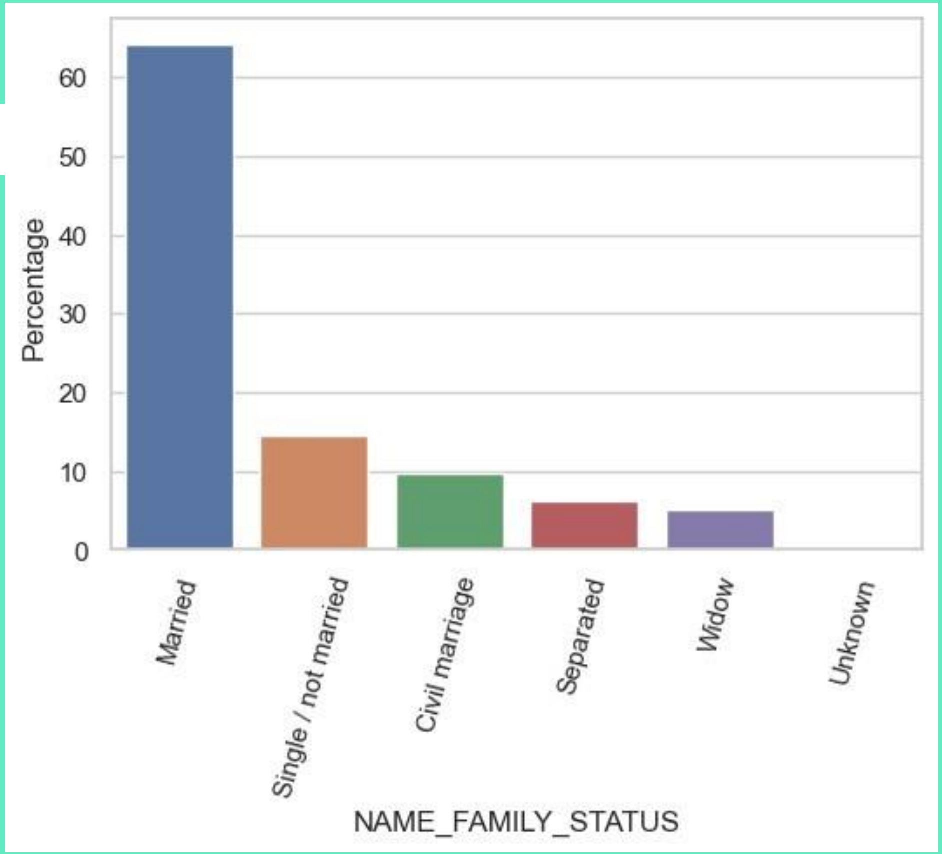
- NAME_INCOME_TYPE data imbalance shows us that 52% applicants are working.



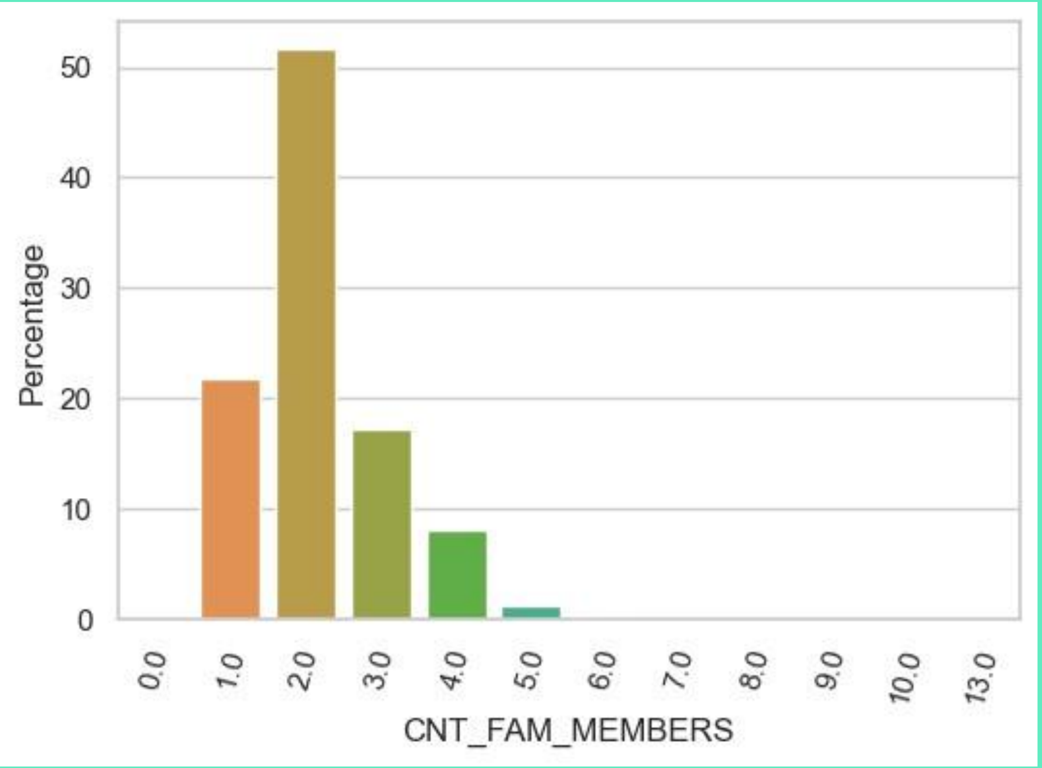
- NAME_TYPE_SUITE data imbalance shows us that 81% of the applicants are unaccompined

- NAME_HOUSINGIN_TYPE data imbalance shows us that 52% applicants are live in House/Apartment.





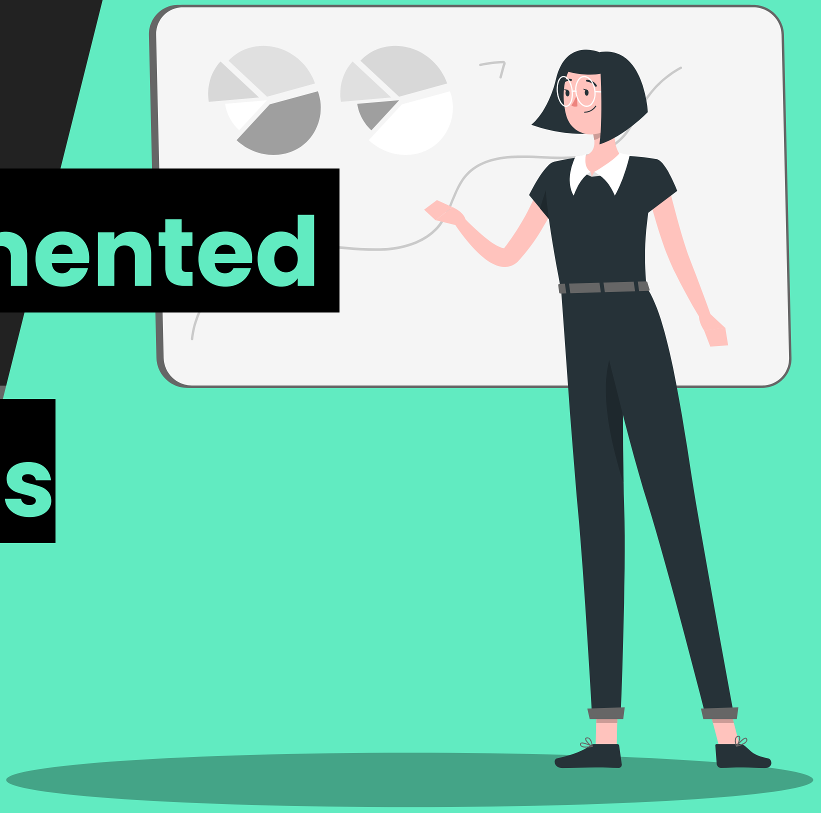
- NAME_FAMILY_STATUS data imbalance shows us that 64% of applicants are married.



- CNT_FAM_MEMBERES data imbalance shows us that 52% of applicants have 2 family members and 22%, 17% applicants have 1 & 3 family members respectively

D

Univariate, Segmented Univariate, and Bivariate Analysis



Univariate Analysis

- We know Univariate analysis focuses on examining the distribution and summary statistics of individual variables.
- summary_stats = application_data.describe()
- Then by using the Tabulate library created the separate table for summary_stats of each variable.

```python
# createing the seperate table using the summary_stats for each variable

from tabulate import tabulate
# create separate tables for each variable
for col in summary_stats.columns:
    col_table = summary_stats[[col]].transpose()

    # Converted the table to a string using tabulate
    table_str = tabulate(col_table, tablefmt='grid')
    print(f"Summary Statistics for {col}:\n{table_str}\n")
    plt.show()
```

- Thus we get following result tables format for each variable.

Result Tables

Summary Statistics for SK_ID_CURR:

SK_ID_CURR	49999	129013	16690.5	100002	114570	129076	143438	157875

Summary Statistics for TARGET:

TARGET	49999	0.0805216	0.272102	0	0	0	0	1

Summary Statistics for CNT_CHILDREN:

CNT_CHILDREN	49999	0.419848	0.724039	0	0	0	1	11

Summary Statistics for AMT_INCOME_TOTAL:

AMT_INCOME_TOTAL	49999	170768	531819	25650	112500	145800	202500	1.17e+08

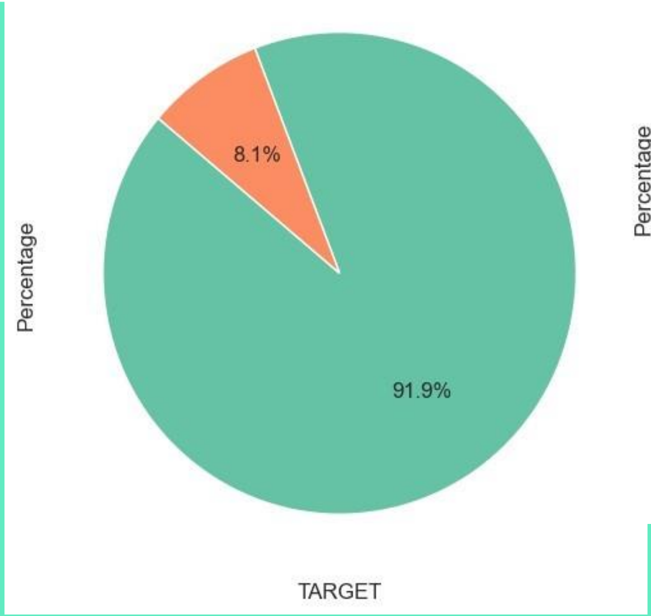
Here, the Summary Statistics for each Table are in the following order:

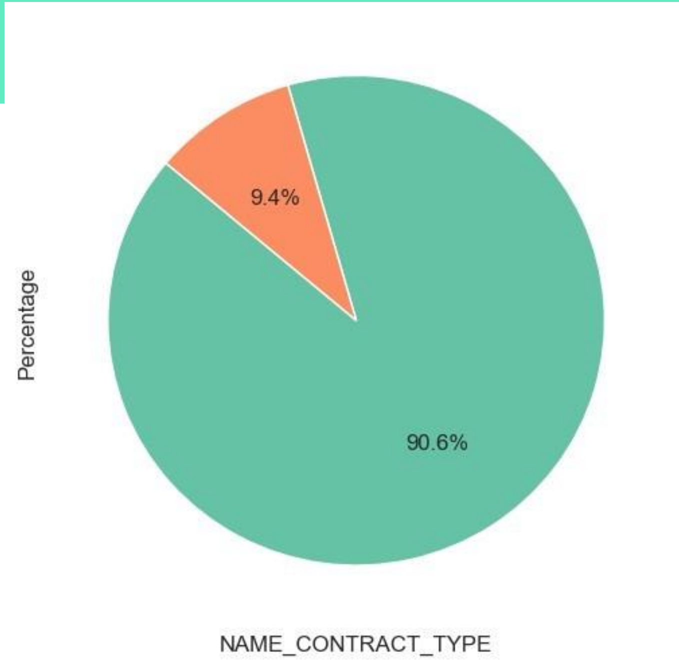
VariableName count mean std min 25% 50% 75% max

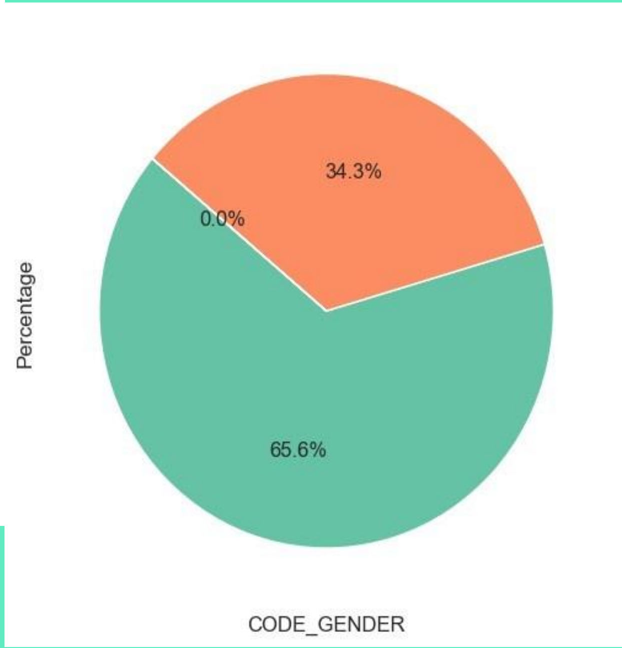
Segmented Univariate Analysis

- We know Segmented univariate analysis involves examining individual variables separately within distinct subgroups to identify patterns or trends.
- Thus, we first Calculated the class frequencies.
- Then Created a pie chart to perform Univariate Analysis.
- Used the following code/program to perform Segmented Univariate Analysis.

```
for i in application_data.columns:
    # Calculate the class frequencies
    class_frequencies = application_data[i].value_counts()

    # Create a pie chart to performt Univariate Analysis.
    # Using if condition to give only those we are categorical variables.

    if len(class_frequencies)<=10:

        print(class_frequencies)
        # printing the percentage of each value_count in list
        percentage = np.around((class_frequencies.values)/len(application_data[i])*100)
        print(percentage)

        # Create a pie chart to visualize Univariate Analysis.
        plt.figure(figsize=(6, 6))
        #sns.pie(x = class_frequencies.index, y = (class_frequencies.values)/len(application_data[i])*100)
        sns.set_palette("Set2")  # Set2 is the inbuilt color palette name
        plt.pie((class_frequencies.values)/len(application_data[i])*100, autopct='%1.1f%%', startangle=140)
        plt.xlabel(i)
        plt.ylabel("Percentage")
        plt.show()
```



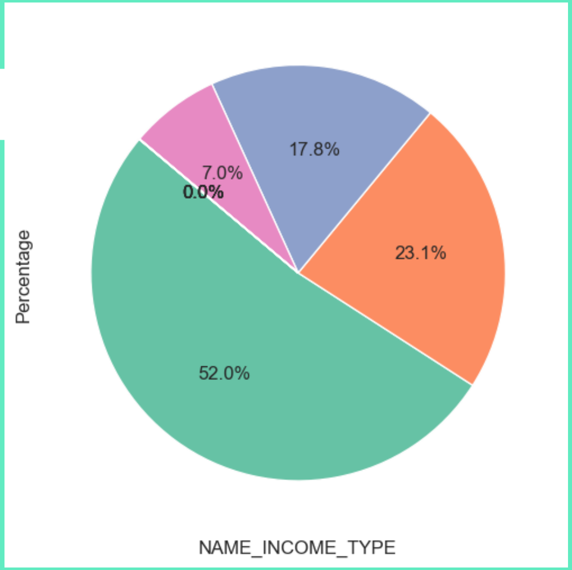


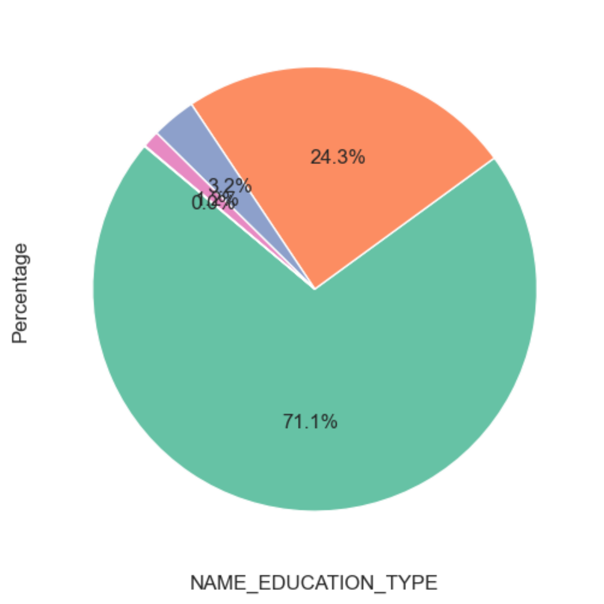


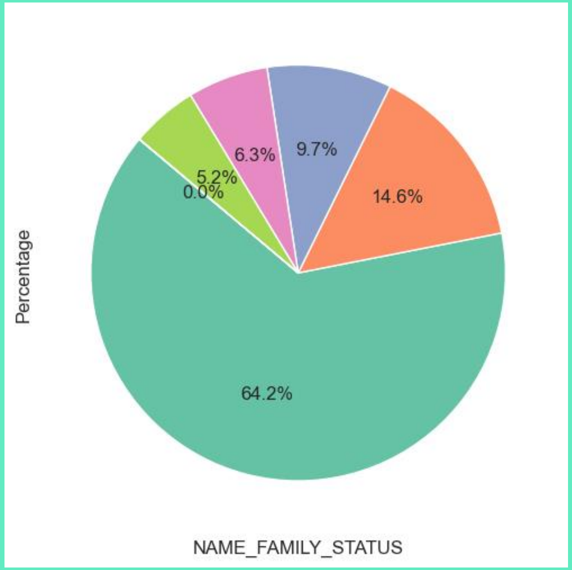
0 45973 1 4026
Name: TARGET,
dtype: int64 [92. 8.]

Cash loans 45276
Revolving loans 4723
Name:
NAME_CONTRACT_TYPE,
dtype: int64 [91. 9.]

F 32823 M 17174
XNA 2 Name:
CODE_GENDER,
dtype: int64 [66.
34. 0.]



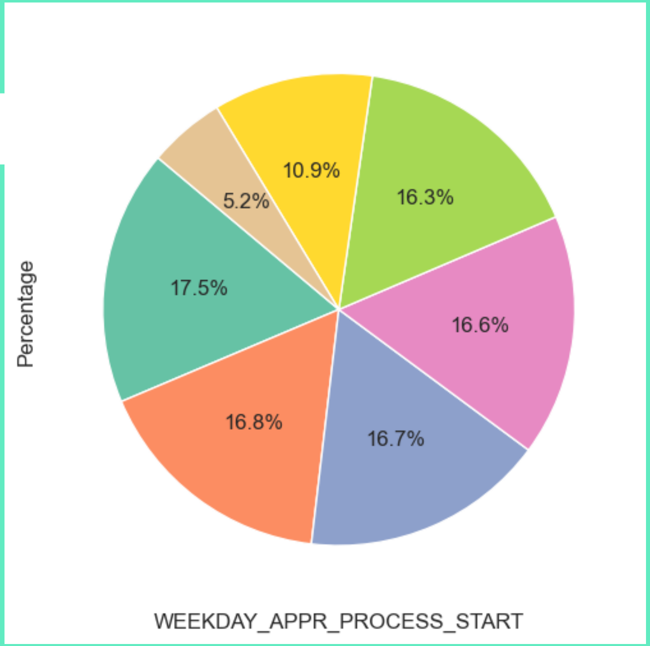





```
Working 26010
Commercial associate 11543
Pensioner 8920
State servant 3512
Unemployed 6
Student 5
Businessman 2
Maternity leave 1
Name: NAME_INCOME_TYPE,
dtype: int64 [52. 23. 18. 7.
0. 0. 0. 0.]
```

```
secondary special 35572 Higher
education 12167
Incomplete higher 1620
Lower secondary 620
Academic degree 20
Name: NAME_EDUCATION_TYPE,
dtype: int64 [71. 24. 3. 1. 0.]
```

```
Married 32094
Single / not married 7306
Civil marriage 4859
Separated 3142
Widow 2597
Unknown 1 Name:
NAME_FAMILY_STATUS, dtype:
int64 [64. 15. 10. 6. 5. 0.]
```




```
TUESDAY 8741
MONDAY 8385
WEDNESDAY 8355
FRIDAY 8286
THURSDAY 8149
SATURDAY 5467
SUNDAY 2616
Name:
WEEKDAY_APPR_PROCESS_START,
dtype: int64 [17. 17. 17. 17.
16. 11. 5.]
```

Inferences

-> target variable shows us that 92% people have the value 1 i.e. clients with payment difficulties
-> NAME_CONTRACT_TYPE has the 91% of loans of Cash Loans type.
-> CODE_GENDER univariate analysis shows us that there are 66% female applicants and 34% male applicants
-> FLAG_OWN_CAR univariate analysis shows us that 66% of applicants don’t own a car.
-> FLAG_OWN_REALTY univariate analysis shows us that 69% of applicants own a house.
-> CNT_CHILDREN univariate analysis shows us that 70% of applicants don't have children.
-> NAME_TYPE_SUITE univariate analysis shows us that 81% of the applicants are unaccompanied
-> NAME_INCOME_TYPE univariate analysis shows us that 52% of applicants are working.
-> NAME_FAMILY_STATUS univariate analysis shows us that 64% of applicants are married.
-> NAME_HOUSINGIN_TYPE univariate analysis shows us that 52% of applicants live in Houses/Apartments.
-> CNT_FAM_MEMBERES univariate analysis shows us that 52% of applicants have 2 family members and 22%, and 17% of applicants have 1 & 3 family members respectively

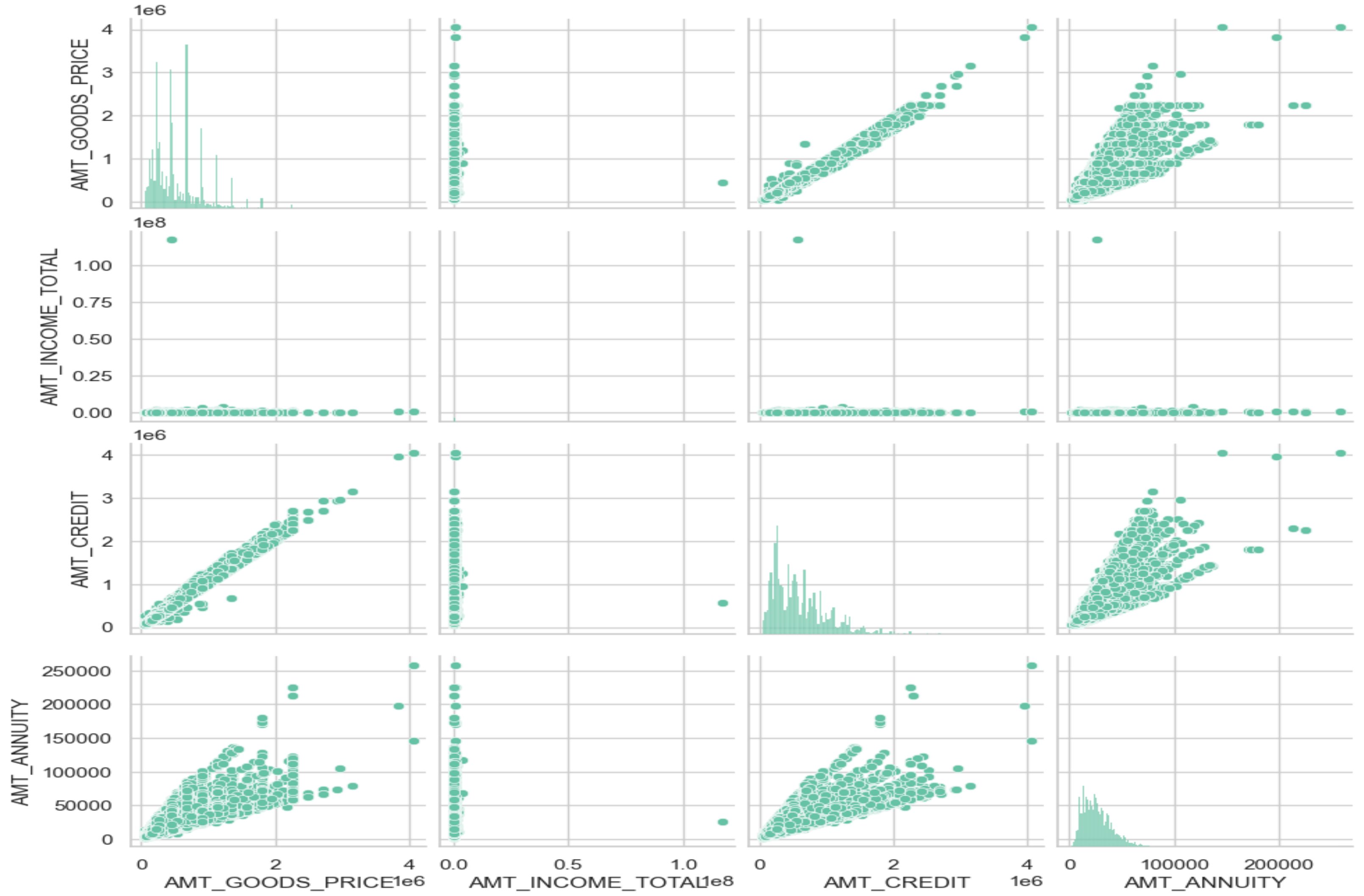
Bivariate Analysis

- We know Bivariate analysis involves analyzing the relationship between two variables to understand their mutual interactions and correlations.
- We created pariplot between some variables using following code.
- sns.pairplot(application_data[['AMT_GOODS_PRICE','AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY']])

❖ Inference of the pairplots

The variables AMT_GOODS_PRICE, AMT_ANNUITY, and AMT_CREDIT show a good positive correlation, which is expected due to the higher cost of goods leading to larger loan amounts and subsequent annuity payments.

- Thus I get following result graphs of pairplots.


AMT_GOODS_PRICE
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
1e6
1e8
1e6
0
1
2
3
4
0.00
0.25
0.50
0.75
1.00
0
50000
100000
150000
200000
250000
0.0
0.5
1.0
100000
200000
AMT_GOODS_PRICE1e6
AMT_INCOME_TOTAL1e8
AMT_CREDIT
1e6
AMT_ANNUITY

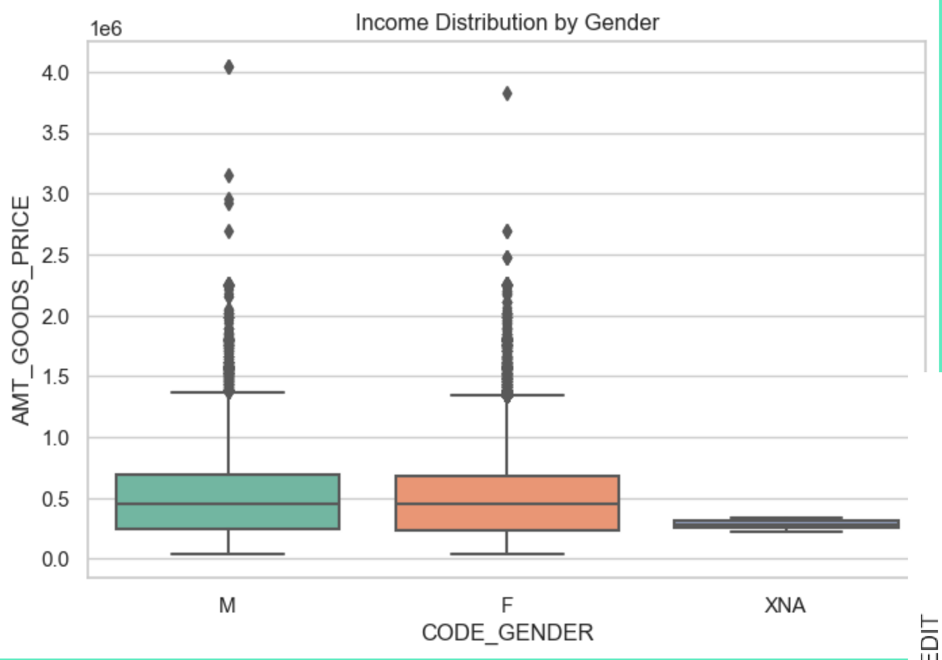
Bivariate Analysis

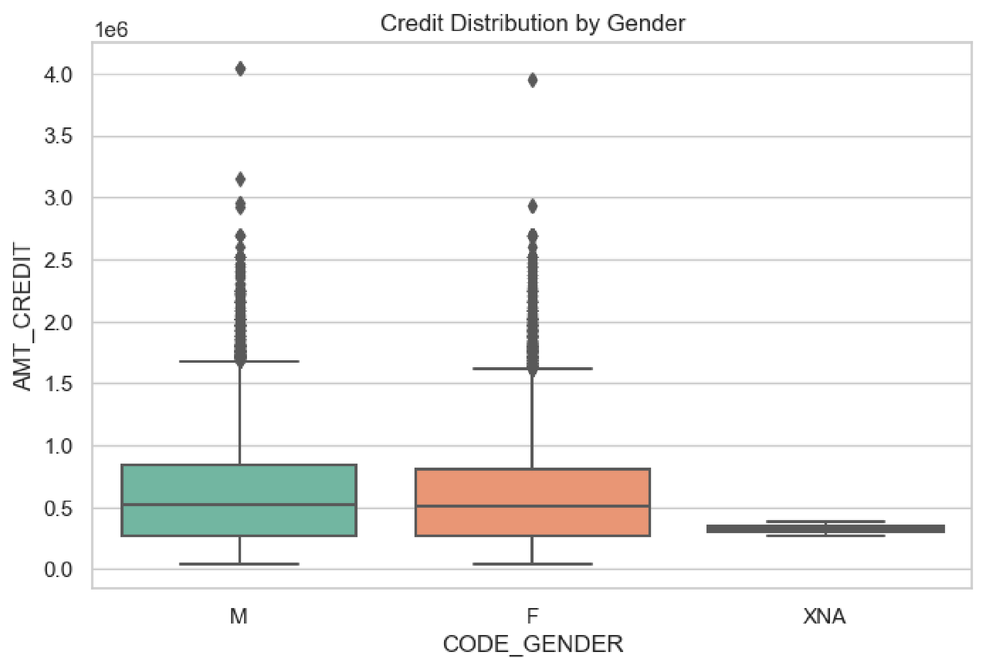
- Then, I created a Boxplot to find Income Distribution by Gender and to find Credit Distribution by Gender.
- I also created a Boxplot to find Credit Distribution by Income Type.

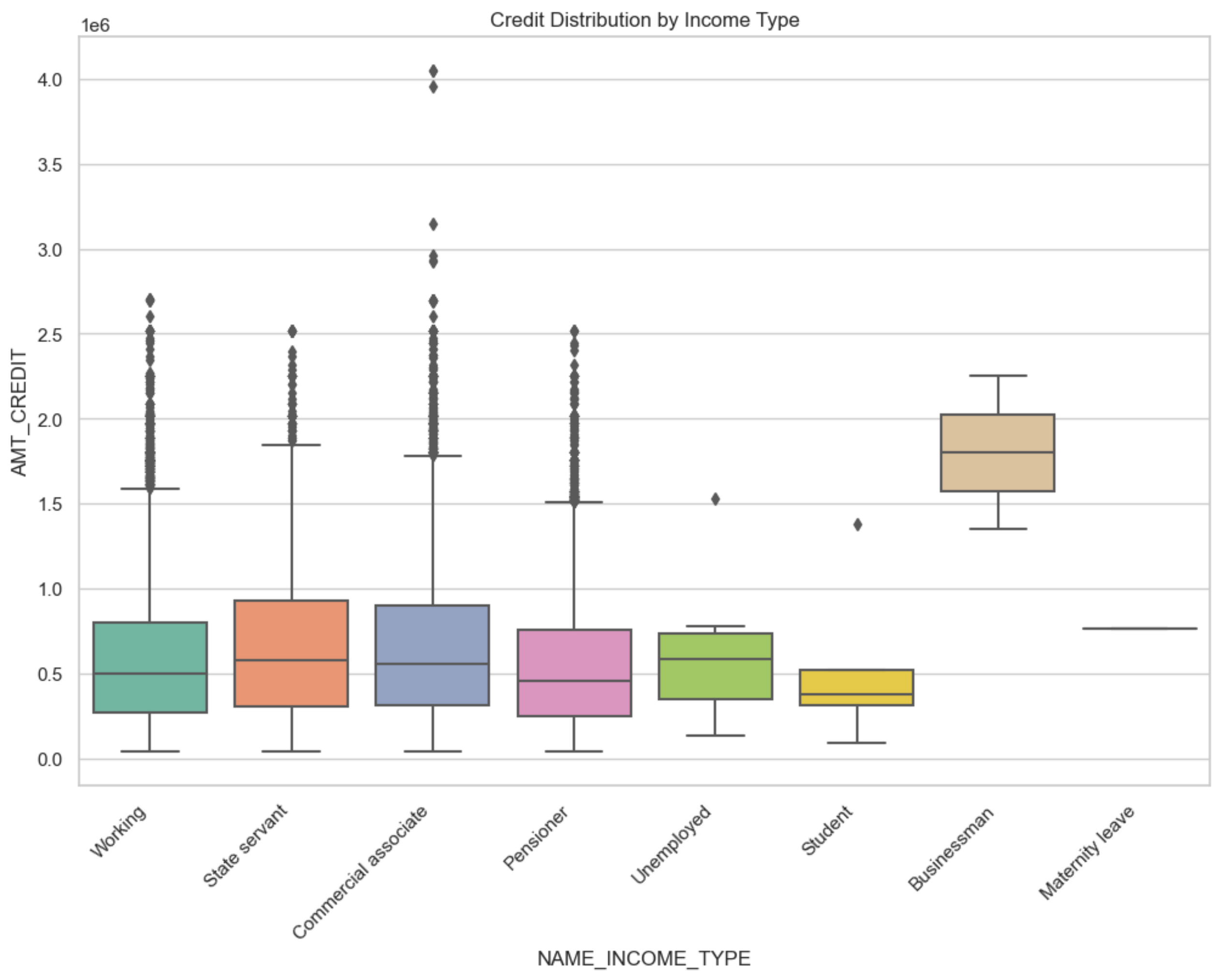
- Inference of this Bivariate Analysis

i. Bivariate analysis between AMT_GOODS_PRICE and AMT_CREDIT through box plots segmented by CODE_GENDER shows that both male and female applicants tend to receive similar loan amounts.

ii. Bivariate analysis of AMT_CREDIT through box plots segmented by NAME_INCOME_TYPE shows that Businessman get more Credit amount than that of other income types.


Income Distribution by Gender
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
AMT_GOODS_PRICE
M
F
XNA
CODE_GENDER


Credit Distribution by Gender
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
AMT_CREDIT
M
F
XNA
CODE_GENDER


Credit Distribution by Income Type
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
AMT_CREDIT
Working
State servant
Commercial associate
Pensioner
Unemployed
Student
Businessman
Maternity leave
NAME_INCOME_TYPE

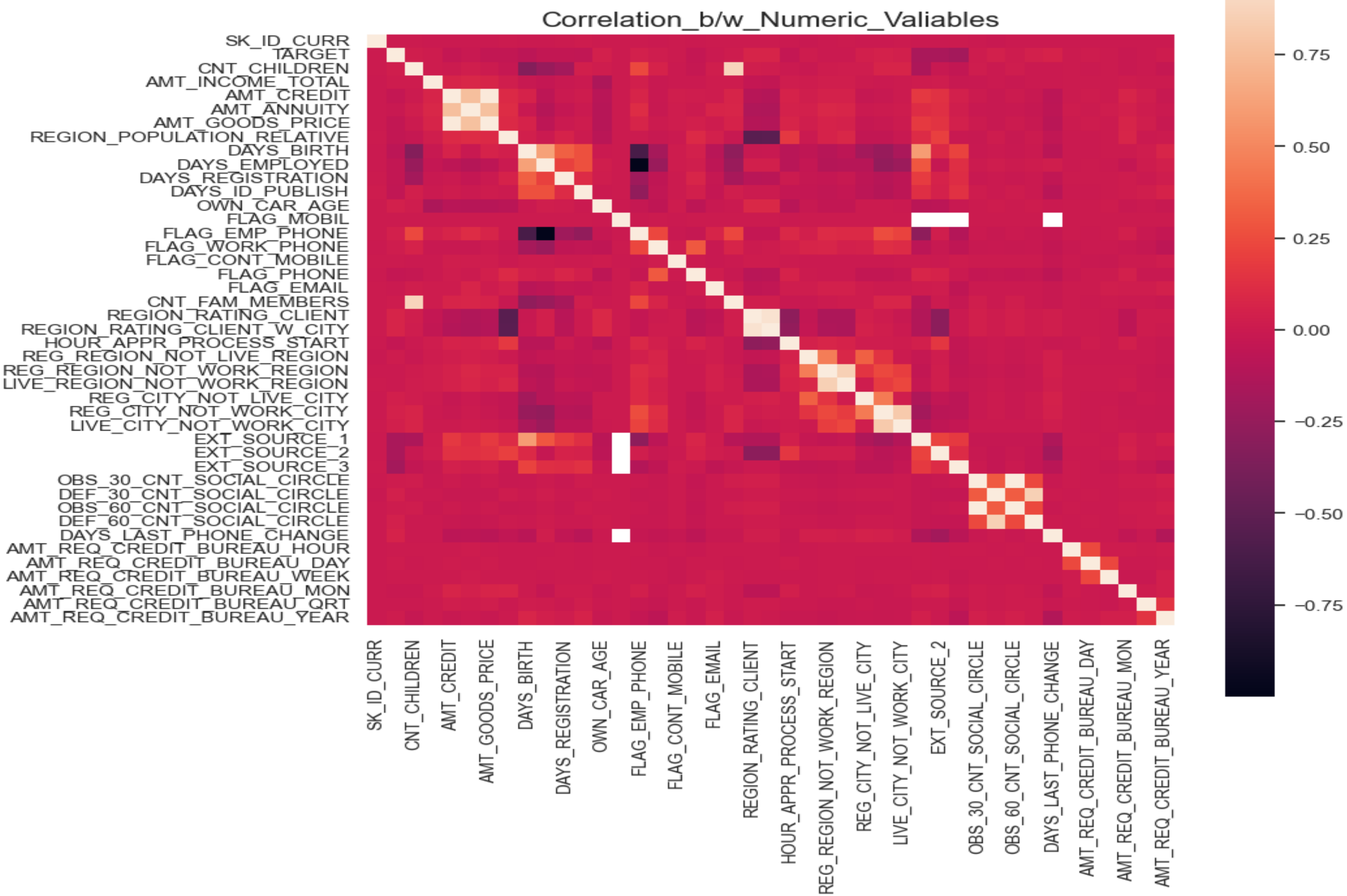
E

Top Correlations for Different Scenarios

Understanding the correlation between variables can provide insights into strong indicators of loan default.

Budget Analysis

- Correlation is a statistical measure that indicates the degree and direction of a linear relationship between two variables.
- Where values closer to +1 or -1 suggest strong correlation, while values closer to 0 suggest weak or no correlation.
- Thus, to find the correlation between the numeric variables first I selected numeric variables and then by using the following code calculated the correlation of it with each other.
- correlation = numerical_columns.corr()
- Then by using the "sns.heatmap(correlation, square = True, vmax = 1, linewidths = 0.000001)" code/program I created the heatmap for the calculated correlations
- Then I stored the upper triangle of the correlation matrix so that there are no same correlation values counted twice.
- upper_triangle_values = correlation.where(np.triu(np.ones(correlation.shape), k=1).astype(bool))
- Finally, by using these "upper_triangle_values" I find the top 5 highest correlations & find the top 5 lowest correlations.


Correlation_b/w_Numeric_Valiables
SK_ID_CURR
TARGET
CNT_CHILDREN
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
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
CNT_FAM_MEMBERS
REGION_RATING_CLIENT
REGION_RATING_CLIENT_W_CITY
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
EXT_SOURCE_1
EXT_SOURCE_2
EXT_SOURCE_3
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE
DEF_60_CNT_SOCIAL_CIRCLE
DAYS_LAST_PHONE_CHANGE
AMT_REQ_CREDIT_BUREAU_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
SK_ID_CURR
CNT_CHILDREN
AMT_CREDIT
AMT_GOODS_PRICE
DAYS_BIRTH
DAYS_REGISTRATION
OWN_CAR_AGE
FLAG_EMP_PHONE
FLAG_CONT_MOBILE
FLAG_EMAIL
REGION_RATING_CLIENT
HOUR_APPR_PROCESS_START
REG_REGION_NOT_WORK_REGION
REG_CITY_NOT_LIVE_CITY
LIVE_CITY_NOT_WORK_CITY
EXT_SOURCE_2
OBS_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE
DAYS_LAST_PHONE_CHANGE
AMT_REQ_CREDIT_BUREAU_DAY
AMT_REQ_CREDIT_BUREAU_MON
AMT_REQ_CREDIT_BUREAU_YEAR
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

```python
# Get the upper triangle of the correlation matrix so that there is no sa
upper_triangle_values = correlation.where(np.triu(np.ones(correlation.sha

# Find the top 5 highest correlations
top_positive_correlations = upper_triangle_values.unstack().sort_values(a

# Find the top 5 lowest correlations
top_negative_correlations = upper_triangle_values.unstack().sort_values(a

print("Top 5 highest correlations:")
print(top_positive_correlations)

print("\nTop 5 lowest correlations:")
print(top_negative_correlations)

```

```
Top 5 highest correlations:
OBS_60_CNT_SOCIAL_CIRCLE      OBS_30_CNT_SOCIAL_CIRCLE       0.998331
AMT_GOODS_PRICE               AMT_CREDIT                     0.986944
REGION_RATING_CLIENT_W_CITY   REGION_RATING_CLIENT           0.950710
CNT_FAM_MEMBERS               CNT_CHILDREN                   0.880430
LIVE_REGION_NOT_WORK_REGION   REG_REGION_NOT_WORK_REGION     0.857142
dtype: float64

Top 5 lowest correlations:
FLAG_EMP_PHONE                DAYS_EMPLOYED                 -0.999746
                              DAYS_BIRTH                    -0.617703
REGION_RATING_CLIENT          REGION_POPULATION_RELATIVE    -0.532667
REGION_RATING_CLIENT_W_CITY   REGION_POPULATION_RELATIVE    -0.530439
DAYS_BIRTH                    CNT_CHILDREN                  -0.329264
dtype: float64
```

Conclusion

- Thus, I have completed a Bank Loan Case Study.
- Given key findings and all meaningful trends or patterns I have discovered.
- I have learned to use Python Libraries (pandas, matplotlib, numpy and seaborn) to analyze the dataset.
- I have learned to use Jupyter Notebook for Data Analysis.
- All the respective Charts and their output are attached to this report.
- GitHub Repository and drive links are given as follows.

GitHub Repository:- https://github.com/ShindeYash/Bank_Loan_Case_Study.git

Jupyter Notebook:- https://drive.google.com/file/d/12Cb27WusFFfHPV2SC-6nYsNTV156JsS2/view?usp=sharing

Drive Link:- https://drive.google.com/drive/folders/15rl6jAJjOlY_UGsrp9pXE084OLBboBXj?usp=sharing

Video Presentation:- https://www.loom.com/share/9c2f8b76127d4670803c25881c832743?sid=a81e1341-2feb-48da-a98d-0c62488c432e

Thanks!

Do you have any questions?

yashpradeepshinde@gmail.com
Yash Shinde