BANK LOAN CASE STUDY

PROJECT DESCRIPTION

This case study aims to give you an idea of applying EDA in a real business scenario. In this case study, apart from applying the techniques that you have learnt in the EDA module, you will also develop a basic understanding of risk analytics in banking and financial services and understand how data is used to minimize the risk of losing money while lending to customers.

Business Understanding:

The loan providing companies find it hard to give loans to the people due to their insufficient or non-existent credit history. Because of that, some consumers use it as their advantage by becoming a defaulter. Suppose you work for a consumer finance company which specialises in lending various types of loans to urban customers. You have to use EDA to analyse the patterns present in the data. This will ensure that the applicants capable of repaying the loan are not rejected.

When the company receives a loan application, the company has to decide for loan approval based on the applicant's profile. Two types of risks are associated with the bank's decision:

If the applicant is likely to repay the loan, then not approving the loan results in a loss of business to the company.

If the applicant is not likely to repay the loan, i.e. he/she is likely to default, then approving the loan may lead to a financial loss for the company.

The data given below contains the information about the loan application at the time of applying for the loan. It contains two types of scenarios:

The client with payment difficulties: he/she had late payment more than X days on at least one of the first Y instalments of the loan in our sample

All other cases: All other cases when the payment is paid on time.

When a client applies for a loan, there are four types of decisions that could be taken by the client/company:

Approved: The company has approved loan application

Cancelled: The client cancelled the application sometime during approval. Either the client changed her/his mind about the loan or in some cases due to a higher risk of the client he received worse pricing which he did not want.

Refused: The company had rejected the loan (because the client does not meet their requirements etc.).

Unused Offer: Loan has been cancelled by the client but on different stages of the process.

In this case study, you will use EDA to understand how consumer attributes and loan attributes influence the tendency of default.

Business Objectives:

It aims to identify patterns which indicate if a client has difficulty paying their installments which may be used for taking actions such as denying the loan, reducing the amount of loan, lending (to risky applicants) at a higher interest rate, etc. This will ensure that the consumers capable of repaying the loan are not rejected. Identification of such applicants using EDA is the aim of this case study.

In other words, the company wants to understand the driving factors (or driver variables) behind loan default, i.e. the variables which are strong indicators of default. The company can utilize this knowledge for its portfolio and risk assessment.

To develop your understanding of the domain, you are advised to independently research a little about risk analytics – understanding the types of variables and their significance should be enough).

Data Understanding:

Download the Dataset using the link given under dataset section on the right.

1. `**application_data.csv**` contains all the information of the client at the time of application.
   The data is about wheather a client has payment difficulties.
2. `**previous_application.csv**` contains information about the client's previous loan data. It contains the data whether the previous application had been Approved, Cancelled, Refused or Unused offer.
3. `**columns_descrption.csv**` is data dictionary which describes the meaning of the variables.

PROJECT APPROACH

Initially, my approach involved analyzing the dataset and identifying any missing values or blank entries. I then applied appropriate methods such as mean, median, or mode imputation to fill in these gaps. Additionally, I examined the dataset for outliers, including negative values, which required either deletion or standardization. To further understand the data, I employed Python libraries and basic charts for visualization. Subsequently, I drew insights based on my understanding of the data. Finally, I addressed any grammatical errors and rephrased the content while ensuring that the original meaning remained intact.

TECK STACK USED

Python is a high-level, interpreted programming language that was first released in 1991 by Guido van Rossum. It is known for its simplicity, ease of use, and readability. It is free to use, distribute, and modify. Python has a wide range of applications, including web development, data analysis, scientific computing, artificial intelligence, machine learning, and automation Python is one of the programming languages that can be used in Jupyter Notebook. This means that users can write Python code in a Jupyter Notebook cell, run the code, and see the output immediately in the notebook.

RESULTS AND INSIGHTS

1. **Present the overall approach of the analysis. Mention the problem statement and the analysis approach briefly**

Ans1.

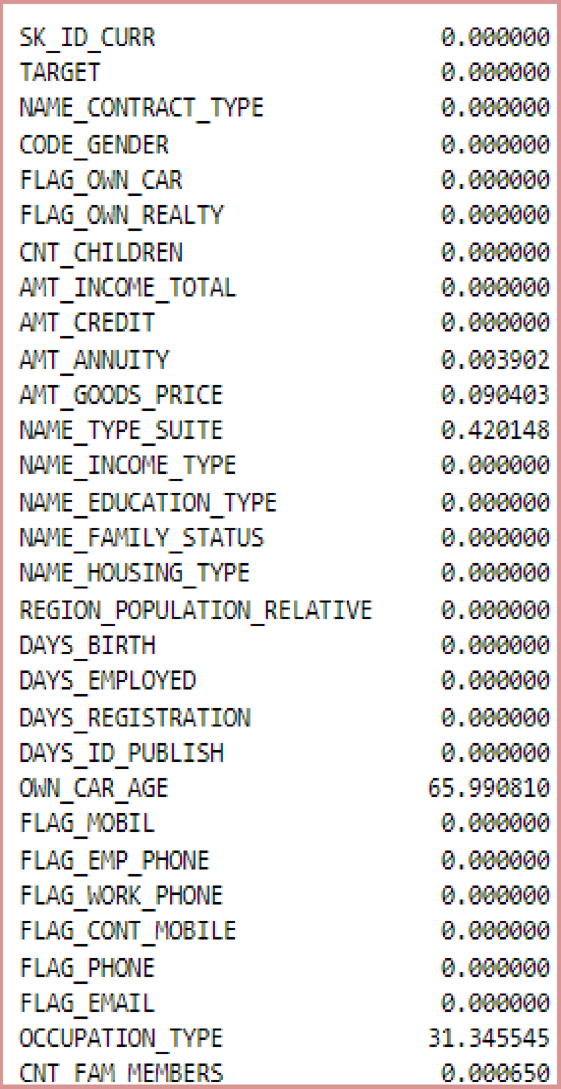
- After downloading the CSV file, I used the Pandas and NumPy libraries to read the data from the CSV file and convert it into a DataFrame by using the following commands:

```
import pandas as pd

import numpy as np

app_df=pd.read_csv("F:\\Trainity\\Bank Loan Case Study\\application_data.csv")

prev_df=pd.read_csv("F:\\Trainity\\Bank Loan Case Study\\previous_application.csv")
```

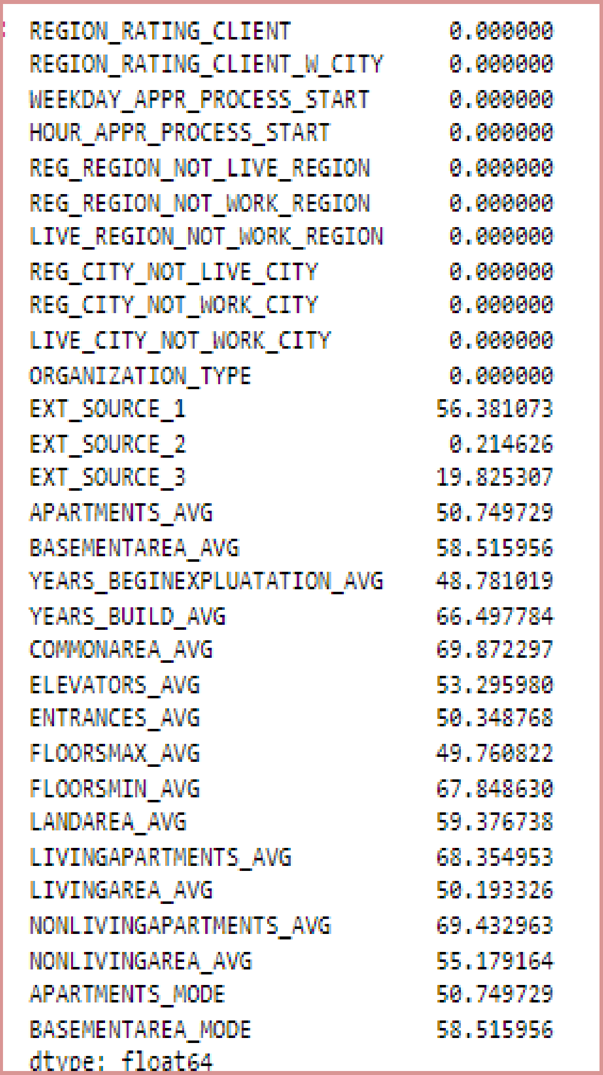
- First, I will read the data and check if there are any null values or errors in the data. As it is raw data, it is very important to clean the data so that it does not cause errors in further analysis.
- After checking the data, I have analyzed that there are null values. Then, I will check if the percentage of null values is greater than 40%.

app_df.iloc[:,0:30].isnull().sum()/len(app_df)*100

app_df.iloc[:,30:60].isnull().sum()/len(app_df)*100



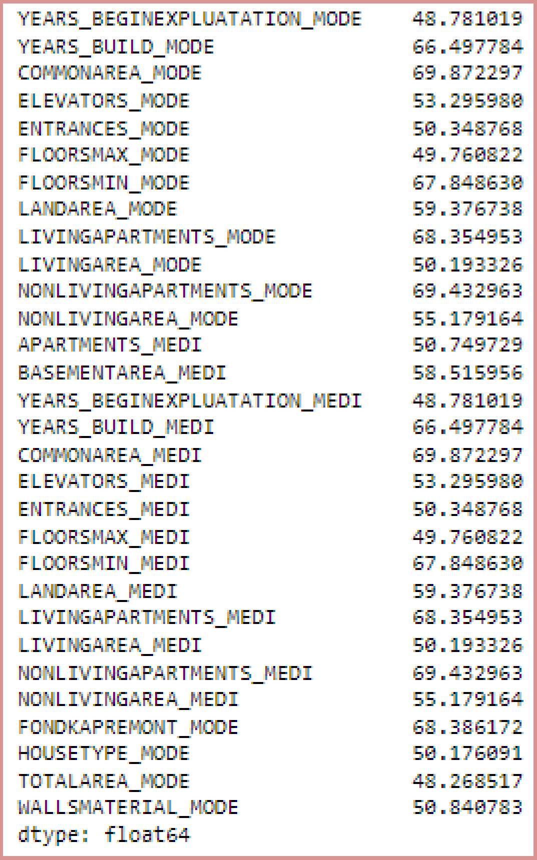
```
SK_ID_CURR                     0.000000
TARGET                         0.000000
NAME_CONTRACT_TYPE             0.000000
CODE_GENDER                    0.000000
FLAG_OWN_CAR                   0.000000
FLAG_OWN_REALTY                0.000000
CNT_CHILDREN                   0.000000
AMT_INCOME_TOTAL               0.000000
AMT_CREDIT                     0.000000
AMT_ANNUITY                    0.003902
AMT_GOODS_PRICE                0.090403
NAME_TYPE_SUITE                0.420148
NAME_INCOME_TYPE               0.000000
NAME_EDUCATION_TYPE            0.000000
NAME_FAMILY_STATUS             0.000000
NAME_HOUSING_TYPE              0.000000
REGION_POPULATION_RELATIVE     0.000000
DAYS_BIRTH                     0.000000
DAYS_EMPLOYED                  0.000000
DAYS_REGISTRATION              0.000000
DAYS_ID_PUBLISH                0.000000
OWN_CAR_AGE                   65.990810
FLAG_MOBIL                     0.000000
FLAG_EMP_PHONE                 0.000000
FLAG_WORK_PHONE                0.000000
FLAG_CONT_MOBILE               0.000000
FLAG_PHONE                     0.000000
FLAG_EMAIL                     0.000000
OCCUPATION_TYPE               31.345545
CNT_FAM_MEMBERS                0.000650
```



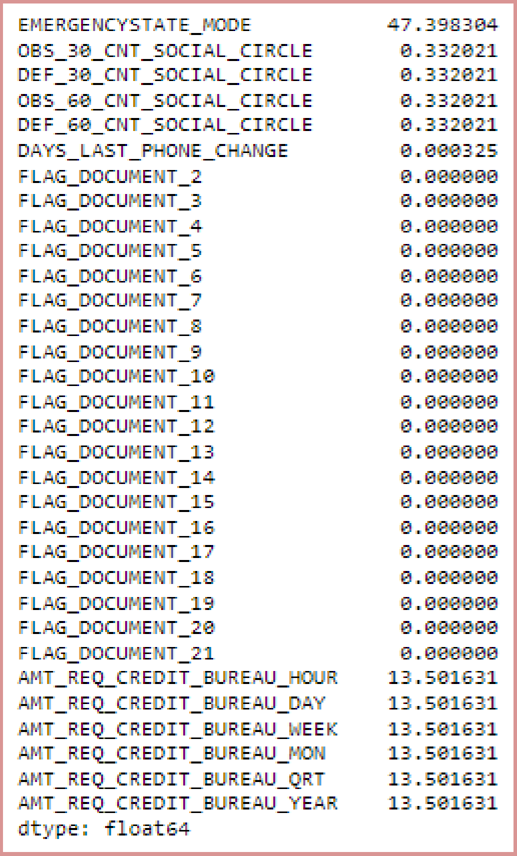
```
REGION_RATING_CLIENT             0.000000
REGION_RATING_CLIENT_W_CITY      0.000000
WEEKDAY_APPR_PROCESS_START       0.000000
HOUR_APPR_PROCESS_START          0.000000
REG_REGION_NOT_LIVE_REGION       0.000000
REG_REGION_NOT_WORK_REGION       0.000000
LIVE_REGION_NOT_WORK_REGION      0.000000
REG_CITY_NOT_LIVE_CITY           0.000000
REG_CITY_NOT_WORK_CITY           0.000000
LIVE_CITY_NOT_WORK_CITY          0.000000
ORGANIZATION_TYPE                0.000000
EXT_SOURCE_1                    56.381073
EXT_SOURCE_2                     0.214626
EXT_SOURCE_3                    19.825307
APARTMENTS_AVG                  50.749729
BASEMENTAREA_AVG                58.515956
YEARS_BEGINEXPLUATATION_AVG     48.781019
YEARS_BUILD_AVG                 66.497784
COMMONAREA_AVG                  69.872297
ELEVATORS_AVG                   53.295980
ENTRANCES_AVG                   50.348768
FLOORSMAX_AVG                   49.760822
FLOORSMIN_AVG                   67.848630
LANDAREA_AVG                    59.376738
LIVINGAPARTMENTS_AVG            68.354953
LIVINGAREA_AVG                  50.193326
NONLIVINGAPARTMENTS_AVG         69.432963
NONLIVINGAREA_AVG               55.179164
APARTMENTS_MODE                 50.749729
BASEMENTAREA_MODE               58.515956
dtype: float64
```

app_df.iloc[:,60:90].isnull().sum()/len(app_df)*100

app_df.iloc[:,90:122].isnull().sum()/len(app_df)*100



```
YEARS_BEGINEXPLUATATION_MODE    48.781019
YEARS_BUILD_MODE                66.497784
COMMONAREA_MODE                 69.872297
ELEVATORS_MODE                  53.295980
ENTRANCES_MODE                  50.348768
FLOORSMAX_MODE                  49.760822
FLOORSMIN_MODE                  67.848630
LANDAREA_MODE                   59.376738
LIVINGAPARTMENTS_MODE           68.354953
LIVINGAREA_MODE                 50.193326
NONLIVINGAPARTMENTS_MODE        69.432963
NONLIVINGAREA_MODE              55.179164
APARTMENTS_MEDI                 50.749729
BASEMENTAREA_MEDI               58.515956
YEARS_BEGINEXPLUATATION_MEDI    48.781019
YEARS_BUILD_MEDI                66.497784
COMMONAREA_MEDI                 69.872297
ELEVATORS_MEDI                  53.295980
ENTRANCES_MEDI                  50.348768
FLOORSMAX_MEDI                  49.760822
FLOORSMIN_MEDI                  67.848630
LANDAREA_MEDI                   59.376738
LIVINGAPARTMENTS_MEDI           68.354953
LIVINGAREA_MEDI                 50.193326
NONLIVINGAPARTMENTS_MEDI        69.432963
NONLIVINGAREA_MEDI              55.179164
FONDKAPREMONT_MODE              68.386172
HOUSETYPE_MODE                  50.176091
TOTALAREA_MODE                  48.268517
WALLSMATERIAL_MODE              50.840783
dtype: float64
```



```
EMERGENCYSTATE_MODE           47.398304
OBS_30_CNT_SOCIAL_CIRCLE       0.332021
DEF_30_CNT_SOCIAL_CIRCLE       0.332021
OBS_60_CNT_SOCIAL_CIRCLE       0.332021
DEF_60_CNT_SOCIAL_CIRCLE       0.332021
DAYS_LAST_PHONE_CHANGE         0.000325
FLAG_DOCUMENT_2                0.000000
FLAG_DOCUMENT_3                0.000000
FLAG_DOCUMENT_4                0.000000
FLAG_DOCUMENT_5                0.000000
FLAG_DOCUMENT_6                0.000000
FLAG_DOCUMENT_7                0.000000
FLAG_DOCUMENT_8                0.000000
FLAG_DOCUMENT_9                0.000000
FLAG_DOCUMENT_10               0.000000
FLAG_DOCUMENT_11               0.000000
FLAG_DOCUMENT_12               0.000000
FLAG_DOCUMENT_13               0.000000
FLAG_DOCUMENT_14               0.000000
FLAG_DOCUMENT_15               0.000000
FLAG_DOCUMENT_16               0.000000
FLAG_DOCUMENT_17               0.000000
FLAG_DOCUMENT_18               0.000000
FLAG_DOCUMENT_19               0.000000
FLAG_DOCUMENT_20               0.000000
FLAG_DOCUMENT_21               0.000000
AMT_REQ_CREDIT_BUREAU_HOUR    13.501631
AMT_REQ_CREDIT_BUREAU_DAY     13.501631
AMT_REQ_CREDIT_BUREAU_WEEK    13.501631
AMT_REQ_CREDIT_BUREAU_MON     13.501631
AMT_REQ_CREDIT_BUREAU_QRT     13.501631
AMT_REQ_CREDIT_BUREAU_YEAR    13.501631
dtype: float64
```

- I will delete the rows where null values greater than 40% as if have more than sufficient data so it will not cause an problem for us further null values greater than 40% will not show us any analysis as it is already have null values more than the data.

null_perc=app_df.isnull().sum()/len(app_df)

null_cols = null_perc[null_perc > 0.4].index.tolist()

app_df.drop(labels=null_cols,axis=1,inplace=True)

- After that, I will delete the columns which are not necessary for our analysis. Hence, I found out that the following are irrelevant columns. We will drop these columns and make our data cleaner.

```
app_df=app_df.drop(['Flag_Mobil','Flag_Emp_Phone','Flag_Work_Phone','Flag_Cont_Mobile','Flag_Phone','Flag_Email','Region_Rating_Client','Cnt_Fam_Members','Region_Rating_Client_W_City','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_Year','Amt_Req_Credit_Bureau_Week','Amt_Req_Credit_Bureau_Day','Amt_Req_Credit_Bureau_Qrt','Amt_Req_Credit_Bureau_Mon'],axis=1)
```

Checking for undefined values

- After checking the undefined values, I have replaced them with proper values to make the data cleaner and avoid any errors in further analysis.

```
app_df.loc[app_df['CODE_GENDER']=='XNA','CODE_GENDER']='F'

app_df.NAME_FAMILY_STATUS=app_df.NAME_FAMILY_STATUS.apply(lambda x: x.replace('/','or'))

app_df.loc[app_df['NAME_FAMILY_STATUS']=='Unknown','NAME_FAMILY_STATUS']='Widow'

app_df.NAME_HOUSING_TYPE=app_df.NAME_HOUSING_TYPE.apply(lambda x: x.replace('/','or'))
```

app_df=app_df[~(app_df.ORGANIZATION_TYPE=='XNA')]

- I checked that the values in these columns are negative and do not make any sense. Hence, they should be positive, so I replaced all negative values with positive values.

app_df['DAYS_BIRTH'] = app_df['DAYS_BIRTH'].abs()

app_df['DAYS_EMPLOYED'] = app_df['DAYS_EMPLOYED'].abs()

app_df['DAYS_REGISTRATION'] = app_df['DAYS_REGISTRATION'].abs()

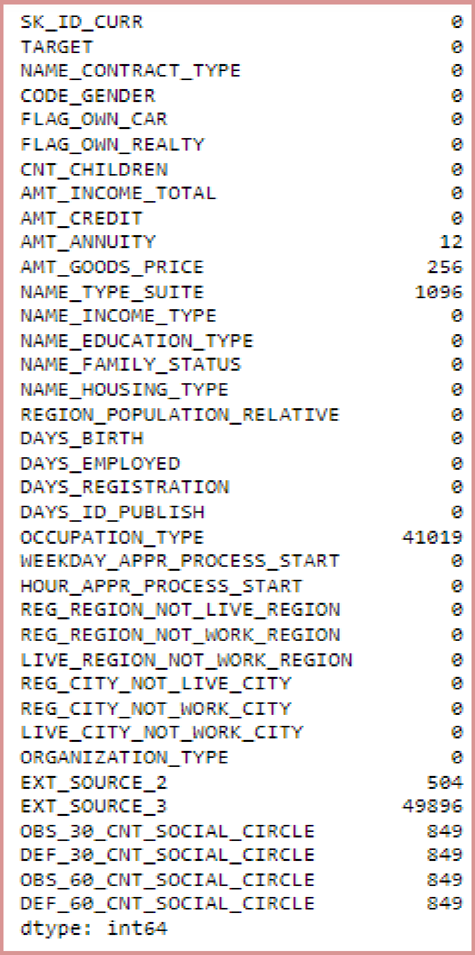
app_df['DAYS_ID_PUBLISH'] = app_df['DAYS_ID_PUBLISH'].abs()

- The values in these columns are in days, so I converted them into years to improve our analysis.

app_df.DAYS_BIRTH=app_df.DAYS_BIRTH.values/365

app_df.DAYS_EMPLOYED=app_df.DAYS_EMPLOYED.values/365

- After deleting all the unnecessary columns and null value columns with greater than 40%, we are left with these values which are important for our analysis.
- So, we should replace these null values with **mean, median, or mode.**



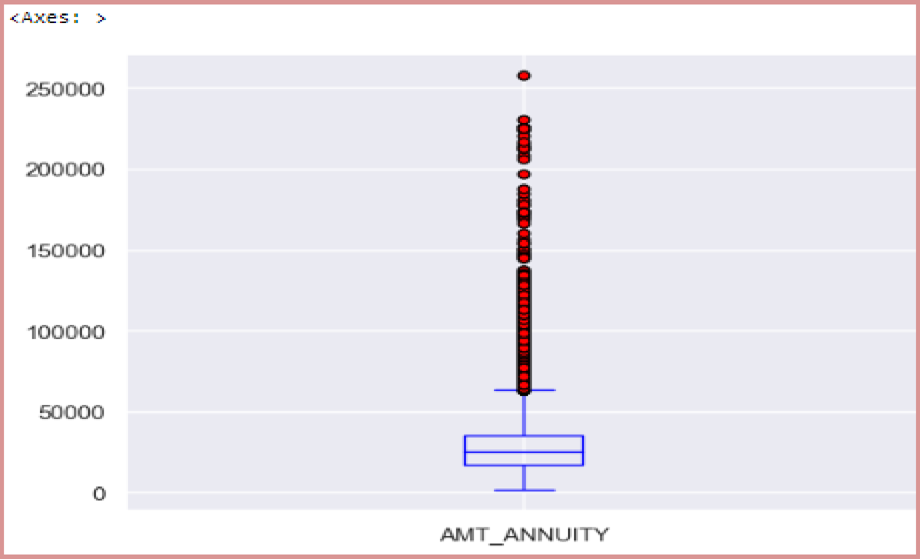
```
SK_ID_CURR                        0
TARGET                            0
NAME_CONTRACT_TYPE                0
CODE_GENDER                       0
FLAG_OWN_CAR                      0
FLAG_OWN_REALTY                   0
CNT_CHILDREN                      0
AMT_INCOME_TOTAL                  0
AMT_CREDIT                        0
AMT_ANNUITY                      12
AMT_GOODS_PRICE                 256
NAME_TYPE_SUITE                1096
NAME_INCOME_TYPE                  0
NAME_EDUCATION_TYPE               0
NAME_FAMILY_STATUS                0
NAME_HOUSING_TYPE                 0
REGION_POPULATION_RELATIVE        0
DAYS_BIRTH                        0
DAYS_EMPLOYED                     0
DAYS_REGISTRATION                 0
DAYS_ID_PUBLISH                   0
OCCUPATION_TYPE               41019
WEEKDAY_APPR_PROCESS_START        0
HOUR_APPR_PROCESS_START           0
REG_REGION_NOT_LIVE_REGION        0
REG_REGION_NOT_WORK_REGION        0
LIVE_REGION_NOT_WORK_REGION       0
REG_CITY_NOT_LIVE_CITY            0
REG_CITY_NOT_WORK_CITY            0
LIVE_CITY_NOT_WORK_CITY           0
ORGANIZATION_TYPE                 0
EXT_SOURCE_2                    504
EXT_SOURCE_3                  49896
OBS_30_CNT_SOCIAL_CIRCLE        849
DEF_30_CNT_SOCIAL_CIRCLE        849
OBS_60_CNT_SOCIAL_CIRCLE        849
DEF_60_CNT_SOCIAL_CIRCLE        849
dtype: int64
```

2. **Indentify the missing data and use appropriate method to deal with it. (Remove columns/or replace it with an appropriate value)**
   Hint: Note that in EDA, since it is not necessary to replace the missing value, but if you have to replace the missing value, what should be the approach. Clearly mention the approach.

Ans 2.

1. **app_df.boxplot(column='AMT_ANNUITY', color='blue', flierprops=dict(marker='o', markerfacecolor='red', markersize=5))**

```
<Axes: >
```

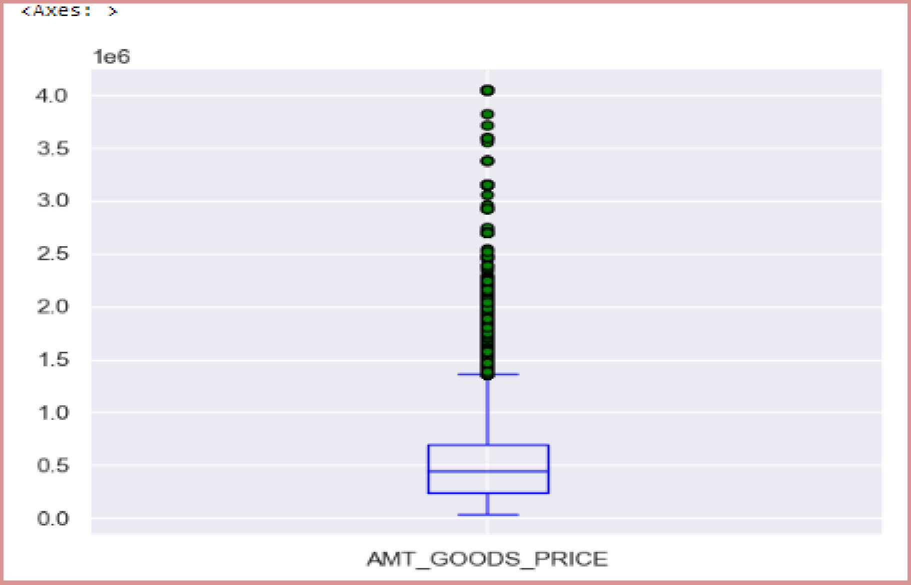


- Since the box plot shows a large number of outliers, we will consider the median as a measure of central tendency.

app_df['AMT_ANNUITY']=app_df['AMT_ANNUITY'].fillna(app_df['AMT_ANNUITY'].median())

2. **app_df.boxplot(column='AMT_GOODS_PRICE', color='blue', flierprops=dict(marker='o', markerfacecolor='green', markersize=5))**

```
<Axes: >
```



- Since the box plot shows a large number of outliers, we will consider the median as a measure of central tendency.

app_df['AMT_GOODS_PRICE']=app_df['AMT_GOODS_PRICE'].fillna(app_df ['AMT_GOODS_PRICE'].median())

3. **mode_value = app_df['NAME_TYPE_SUITE'].mode()[0]**

- Since it is a categorical value, considering mode measure to impute missing values.

  app_df['NAME_TYPE_SUITE'].fillna(mode_value, inplace=True)

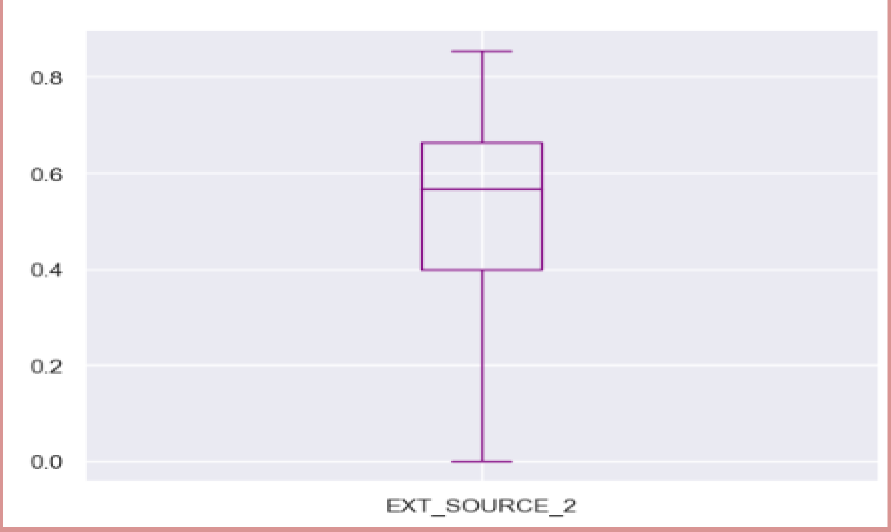
4. **mode_value1 = app_df['OCCUPATION_TYPE'].mode()[0]**

- Since it is a categorical value, considering mode measure to impute missing values.

  app_df['OCCUPATION_TYPE'].fillna(mode_value, inplace=True)

5. **app_df.boxplot(column='EXT_SOURCE_2', color='purple')**

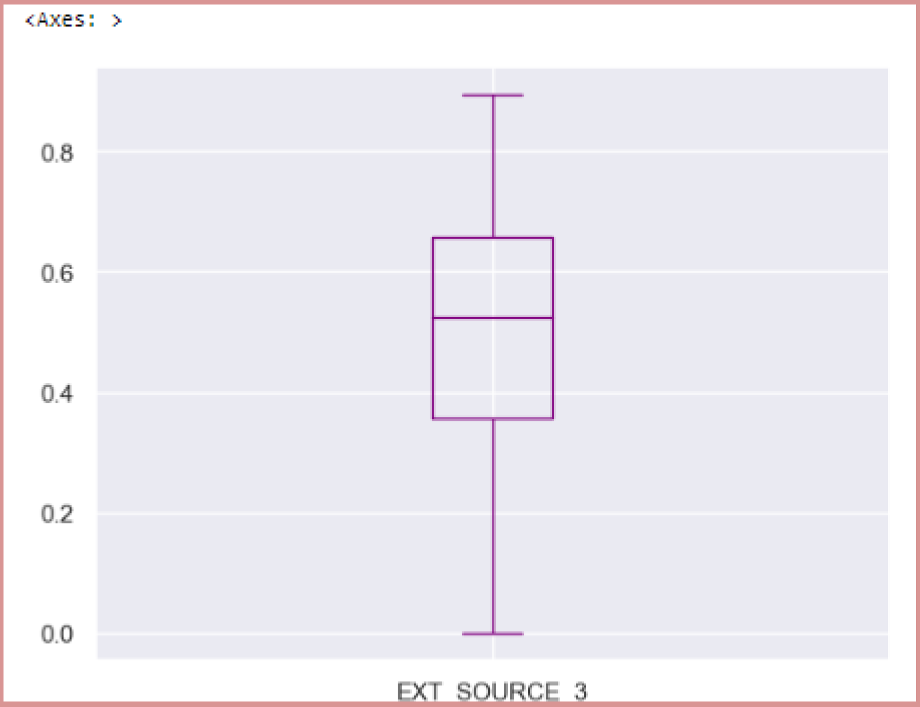
```
<Axes: >
```



- In a boxplot, the middle line in the box represents the median of the data. If the median is positioned exactly in the middle of the box, it means that the data is symmetrically distributed around the median.
- In a box plot, if the median is not centered in the box, it suggests that the data is skewed to one side, and the mean may not be the most appropriate measure of central tendency. In this case, it may be more appropriate to use the median as the measure of central tendency, as it is less sensitive to outliers than the mean.

app_df['EXT_SOURCE_2']=app_df['EXT_SOURCE_2'].fillna(app_df['EXT_SOURCE_2'].median())

6. **app_df.boxplot(column='EXT_SOURCE_3', color='purple')**



app_df['EXT_SOURCE_3']=app_df['EXT_SOURCE_3'].fillna(app_df['EXT_SOURCE_2'].median())

7. **app_df['OBS_30_CNT_SOCIAL_CIRCLE']=app_df['OBS_30_CNT_SOCIA L_CIRCLE'].fillna(app_df['OBS_30_CNT_SOCIAL_CIRCLE'].median())**

8. **app_df['DEF_30_CNT_SOCIAL_CIRCLE']=app_df['DEF_30_CNT_SOCIAL _CIRCLE'].fillna(app_df['DEF_30_CNT_SOCIAL_CIRCLE'].median())**

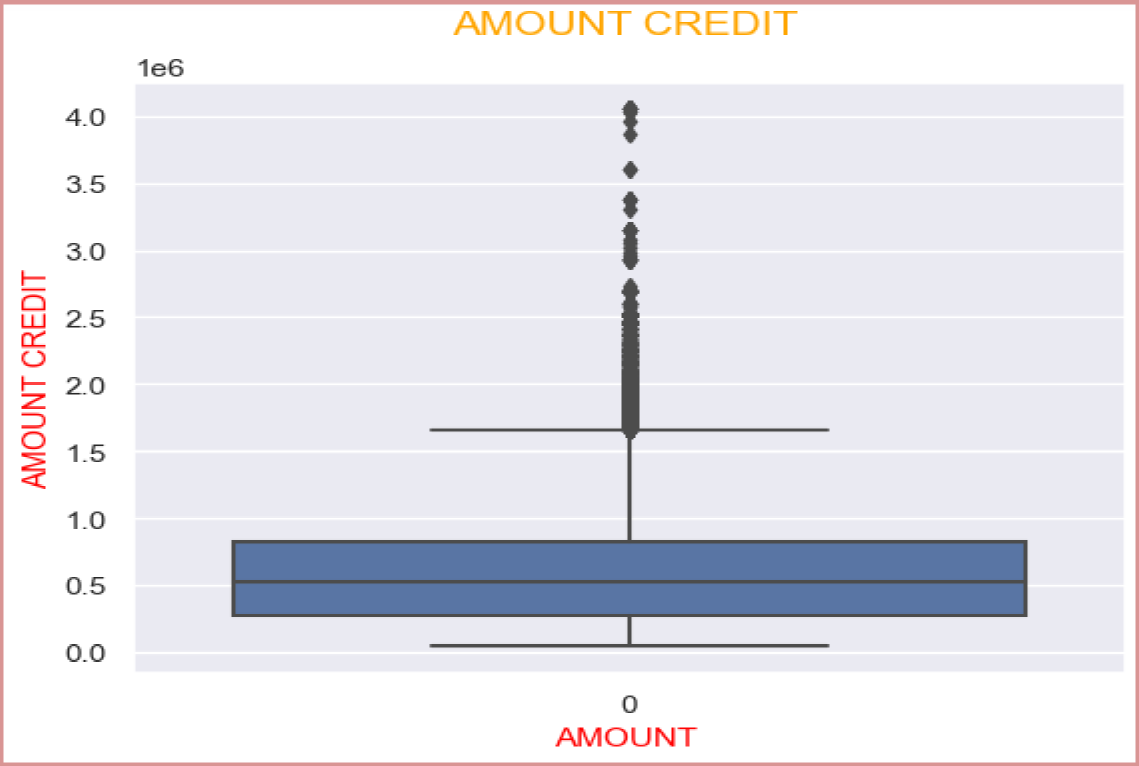
9. **app_df['OBS_60_CNT_SOCIAL_CIRCLE']=app_df['OBS_60_CNT_SOCIA L_CIRCLE'].fillna(app_df['OBS_60_CNT_SOCIAL_CIRCLE'].median())**

10. **app_df['DEF_60_CNT_SOCIAL_CIRCLE']=app_df['DEF_60_CNT_SOCIA L_CIRCLE'].fillna(app_df['DEF_60_CNT_SOCIAL_CIRCLE'].median())**

3. **Identify if there are outliers in the dataset. Also, mention why do you think it is an outlier. Again, remember that for this exercise, it is not necessary to remove any data points.**

Ans3. We have checked for outliers in the numerical columns and identified outliers for at least 5 variables. Values above the upper whisker and below the lower whisker are considered outliers. We observed that the outliers are only present above the upper whisker. Therefore, we will consider only those values as outliers.

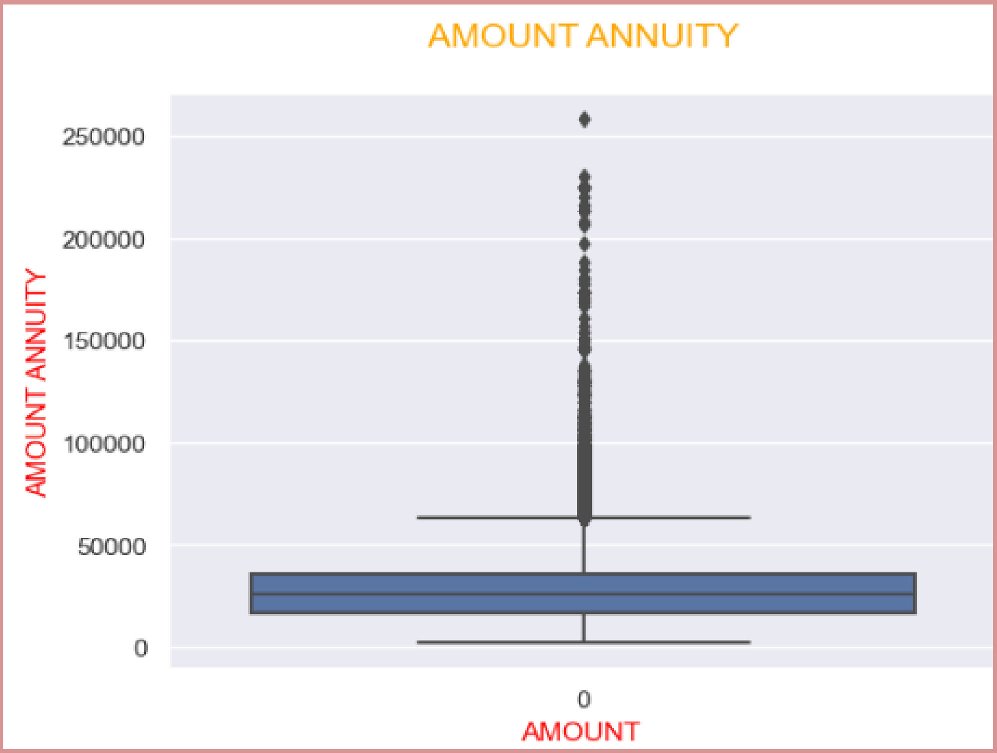
1. sns.boxplot(app_df.**AMT_CREDIT**)
   plt.xlabel("AMOUNT ", fontdict={'fontsize': 12, 'color' : 'Red'})
   plt.ylabel("AMOUNT CREDIT", fontdict={'fontsize': 12, 'color' : 'Red'})
   plt.title('AMOUNT CREDIT \n',fontdict={'fontsize': 15, 'color' : 'Orange'})
   plt.show()



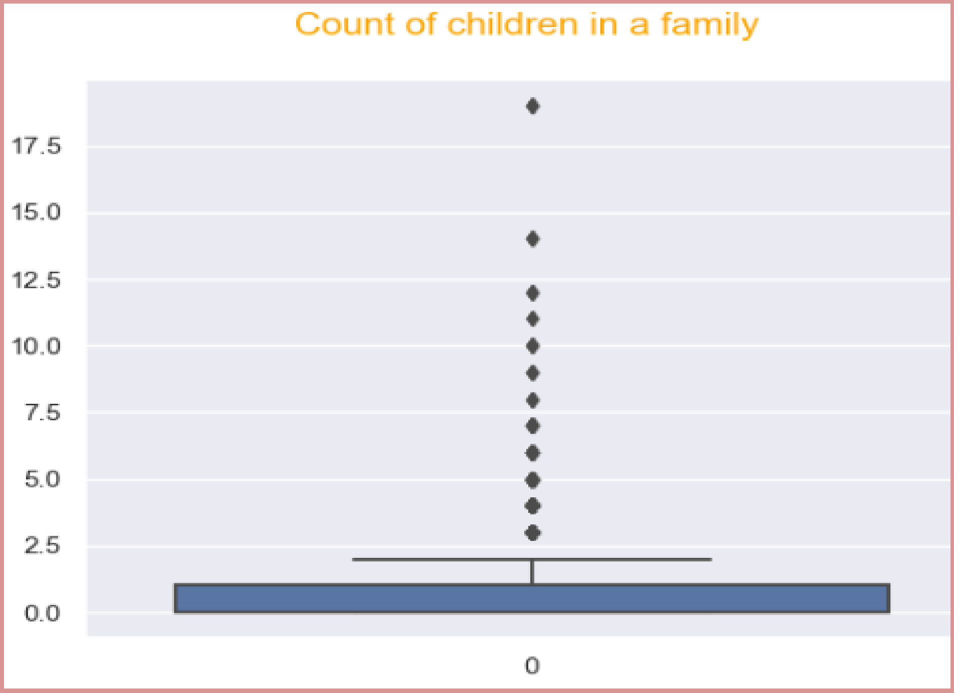
- An outlier is a data point that is located outside the whiskers of a box plot. Any value that falls beyond the whiskers is considered an outlier.
- The amount credit **greater than 1.6** can be considered as an outlier.

2. sns.boxplot(app_df.**AMT_ANNUITY**)
   plt.xlabel("AMOUNT ", fontdict={'fontsize': 12, 'color' : 'Red'})
   plt.ylabel("AMOUNT ANNUITY", fontdict={'fontsize': 12, 'color' : 'Red'})
   plt.title(' AMOUNT ANNUITY \n',fontdict={'fontsize': 15, 'color' : 'Orange'})
   plt.show()

- The amount Annuity greater than **60000+** can be considered as an outlier.

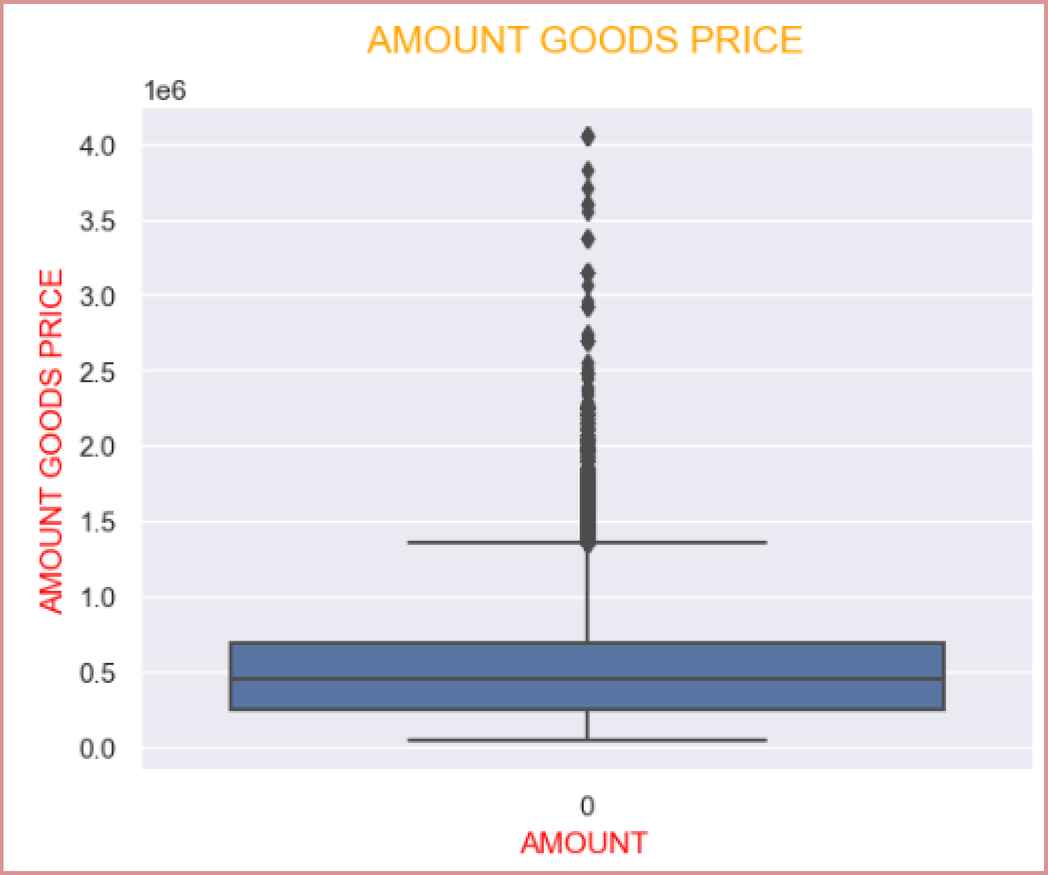


3. sns.boxplot(app_df.**CNT_CHILDREN**)
   plt.title('Count of children in a family \n',fontdict={'fontsize': 15, 'color' : 'Orange'})
   plt.show()



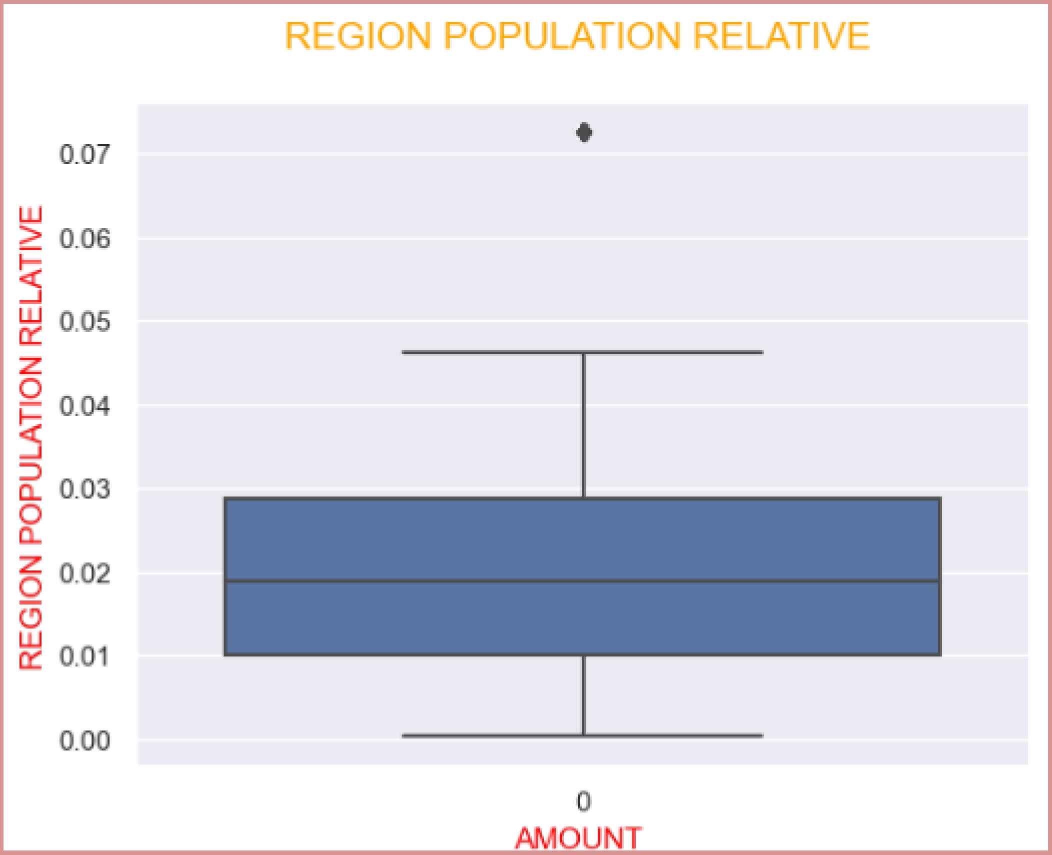
- Values greater than **3** can be considered outliers since the count of children cannot be in decimals. Therefore, we can conclude that a count greater than **3** can be an outlier.

4.
   ```
   sns.boxplot(app_df.AMT_GOODS_PRICE)
   plt.xlabel("AMOUNT ", fontdict={'fontsize': 12, 'color' : 'Red'})
   plt.ylabel("AMOUNT GOODS PRICE", fontdict={'fontsize': 12, 'color' :
   'Red'})
   plt.title(' AMOUNT GOODS PRICE \n',fontdict={'fontsize': 15, 'color' :
   'Orange'})
   plt.show()
   ```



- The amount goods price greater than **1.3** can be considered as an outlier.

5.
   ```
   sns.boxplot(app_df.REGION_POPULATION_RELATIVE)
   plt.xlabel("AMOUNT ", fontdict={'fontsize': 12,'color' : 'Red'})
   plt.ylabel("REGION POPULATION RELATIVE", fontdict={'fontsize': 12,
   'color' : 'Red'})
   plt.title('REGION POPULATION RELATIVE \n',fontdict={'fontsize': 15,
   'color' : 'Orange'})
   plt.show()
   ```

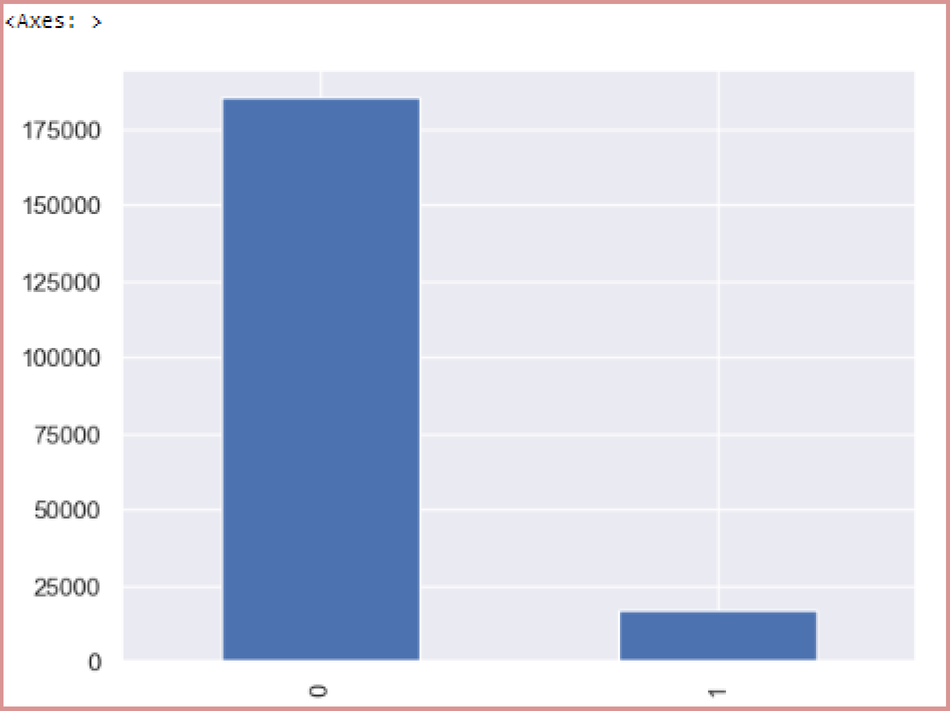


- Population relative count greater than 0.0566 is considered to be an outlier

4. **Identify if there is data imbalance in the data. Find the ratio of data imbalance.**
   Hint: Since there are a lot of columns, you can run your analysis in loops for the appropriate columns and find the insights.

Ans 4.

1. **app_df['TARGET'].value_counts().plot(kind='bar')**


```
<Axes: >
```

- <u>Target 0</u>

✓ ratio=((val.TARGET[0])/ (val1.TARGET[0]+val.TARGET[0]))*100

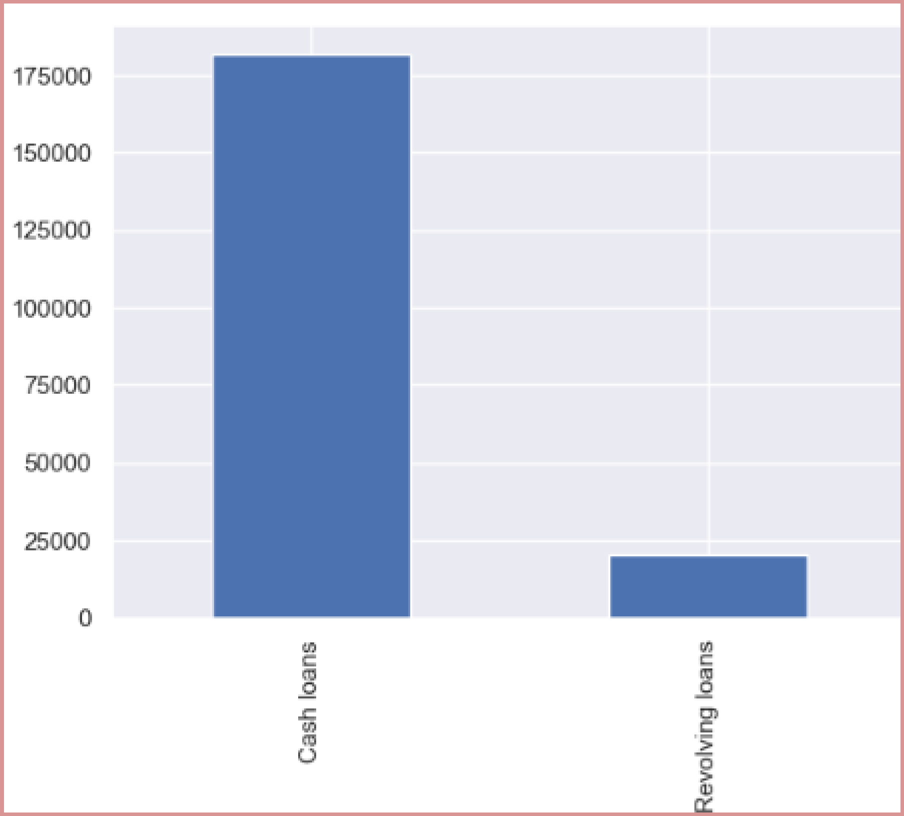
✓ ratio=91.35%

- <u>Target 1</u>

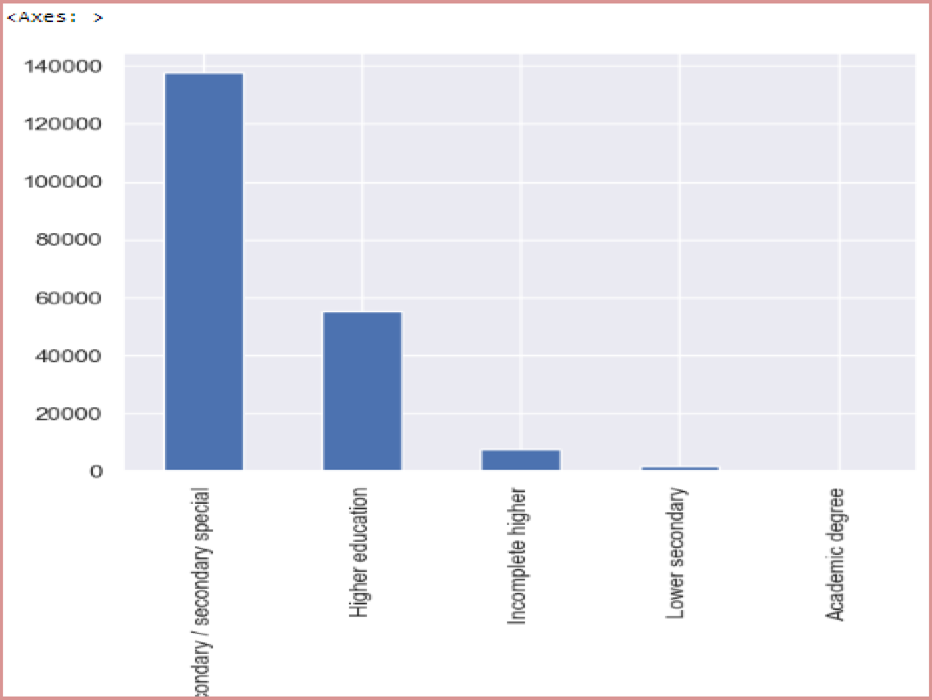
✓ ratio=((val1.TARGET[0])/ (val1.TARGET[0]+val.TARGET[0]))*100

✓ ratio=8.65%

2. app_df['NAME_CONTRACT_TYPE'].value_counts().plot(kind='bar')



3. app_df['NAME_EDUCATION_TYPE'].value_counts().plot(kind='bar')



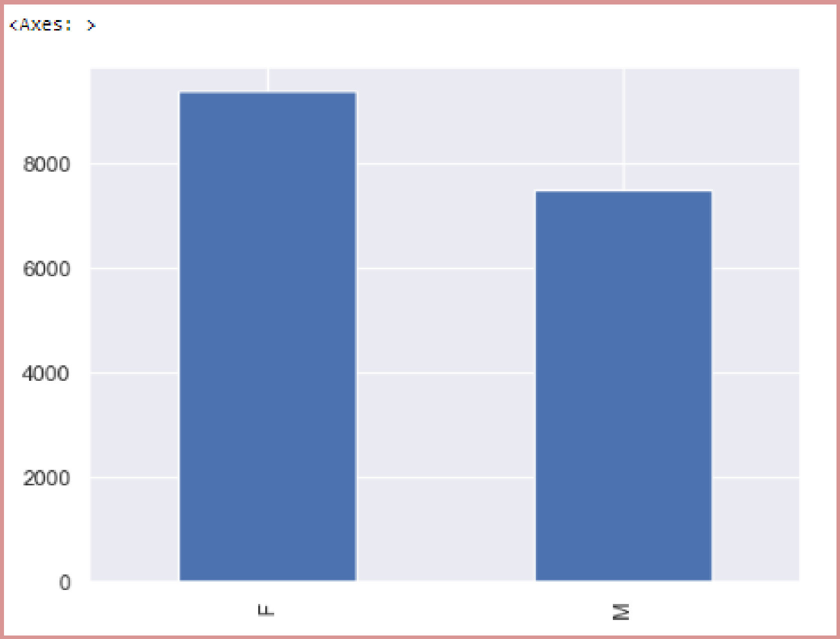
5. **Explain the results of univariate, segmented univariate, bivariate analysis, etc. in business terms.**

Ans 5.

- **Univariate Analysis**: First, we divide the data into two sets, i.e., **Target-1 and Target-0**. Then we perform univariate analysis on the categorical columns in both tables.

- def_=app_df[app_df['TARGET']==1]
  def_['**CODE_GENDER**'].value_counts().plot(kind='bar')

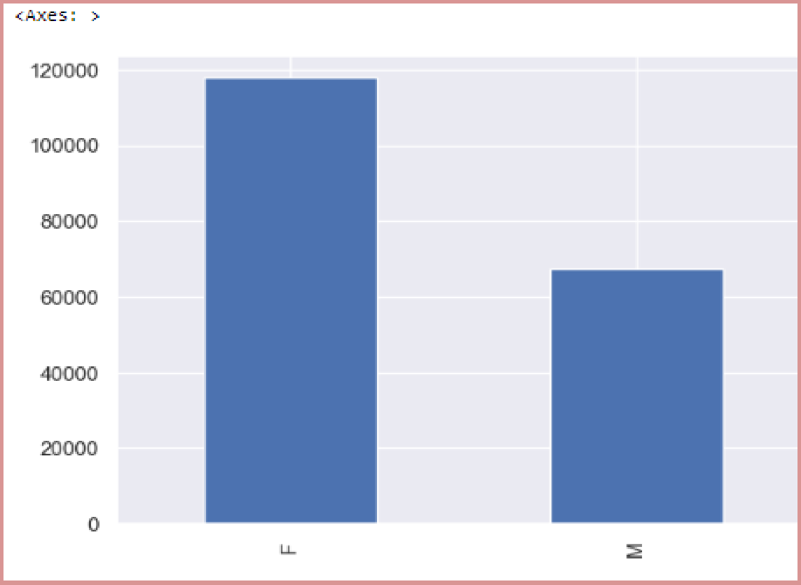
DEFAULTER



```
<Axes: >
```

- ndef_=app_df[app_df['TARGET']==0]
  ndef_['**CODE_GENDER**'].value_counts().plot(kind='bar')

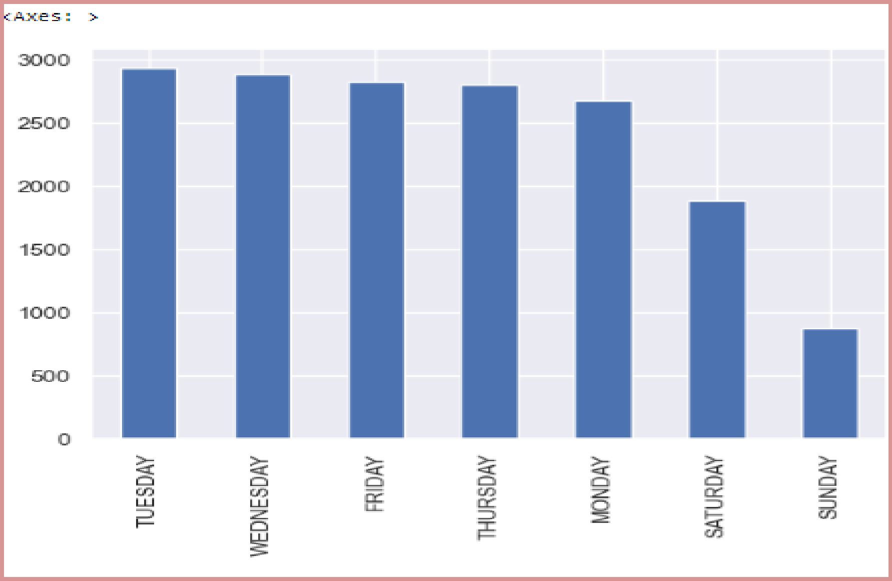
NON-DEAFULTER



- def_=app_df[app_df['TARGET']==1]

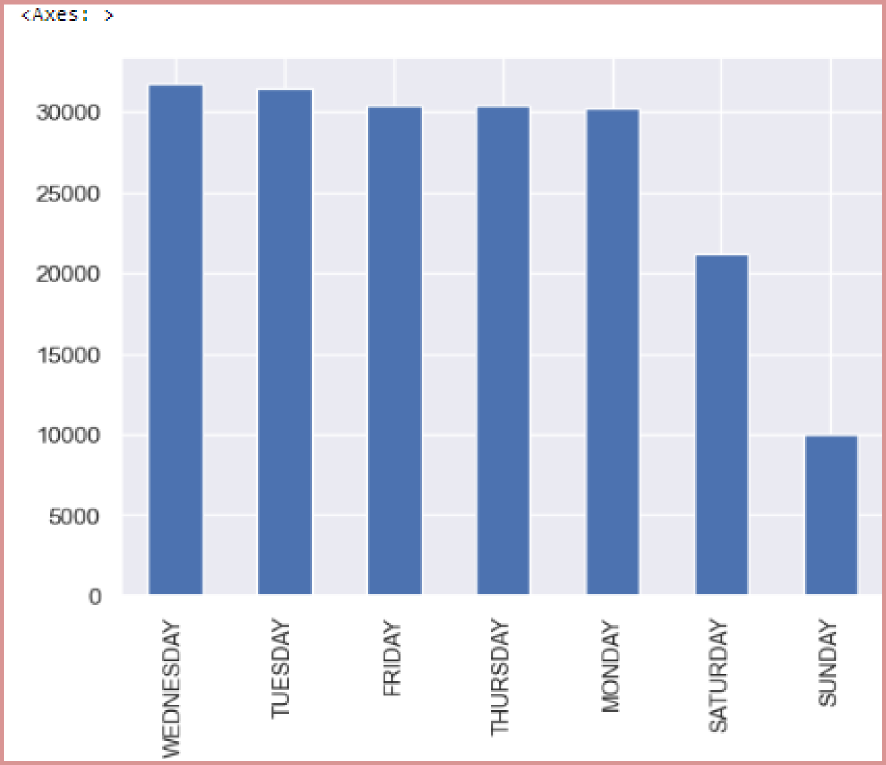
def_['**WEEKDAY_APPR_PROCESS_START**'].value_counts().plot(kind='bar')

DEFAULTER

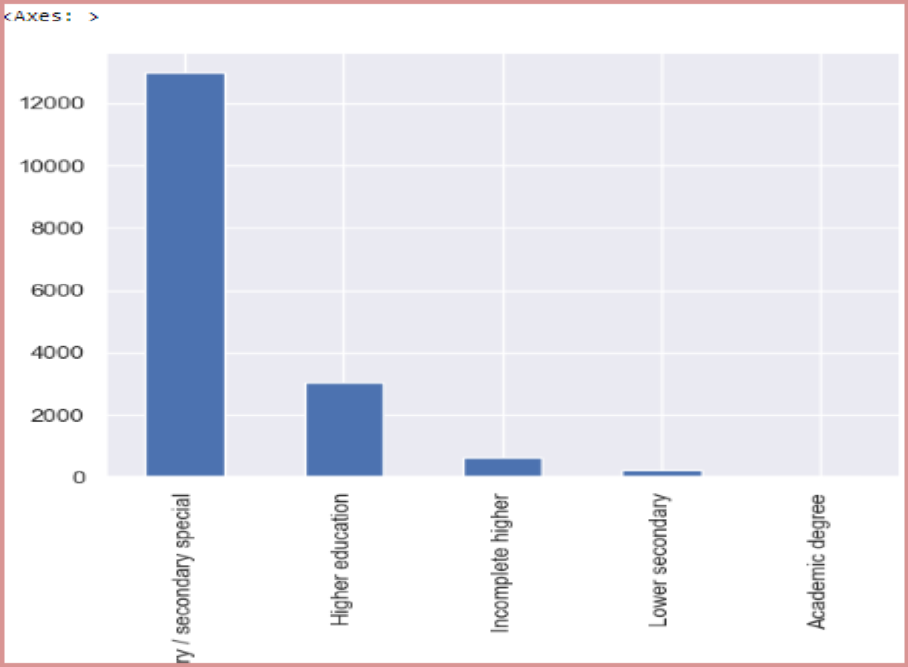


- ndef_=app_df[app_df['TARGET']==0]
  ndef_['**WEEKDAY_APPR_PROCESS_START**'].value_counts().plot(kind='bar')

NON-DEFAULTER

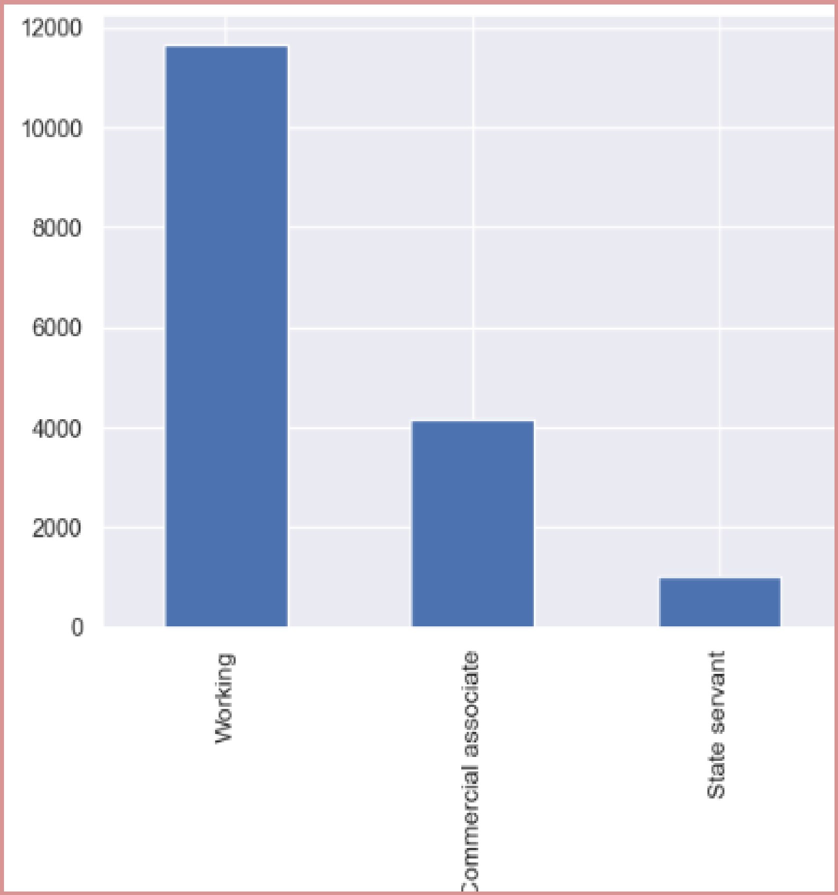


- def_=app_df[app_df['TARGET']==1]
  def_['**NAME_EDUCATION_TYPE**'].value_counts().plot(kind='bar')



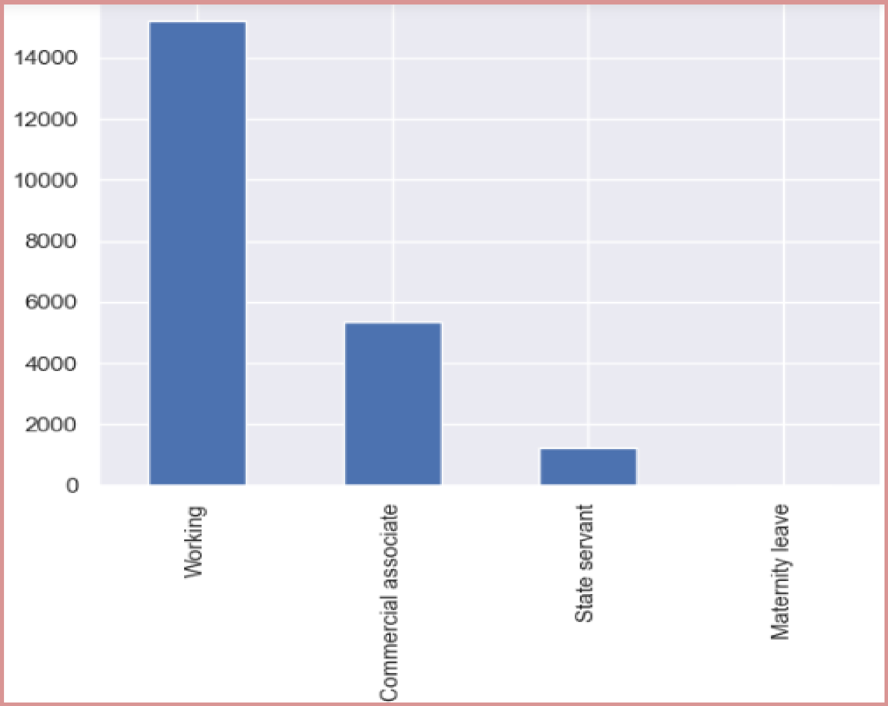
- ndef_=app_df[app_df['TARGET']==0]
  ndef_[**'NAME_EDUCATION_TYPE'**].value_counts().plot(kind='bar')

NON-DEFAULTER



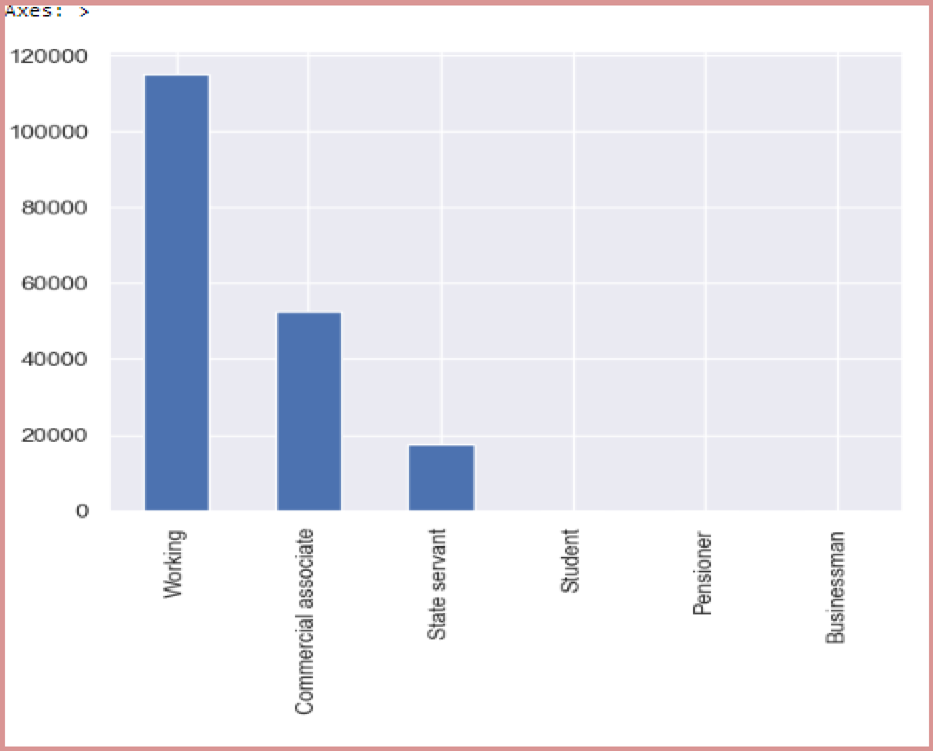
- def_=app_df[app_df['TARGET']==1]
  def_[**'NAME_INCOME_TYPE'**].value_counts().plot(kind='bar')

DEFAULTER



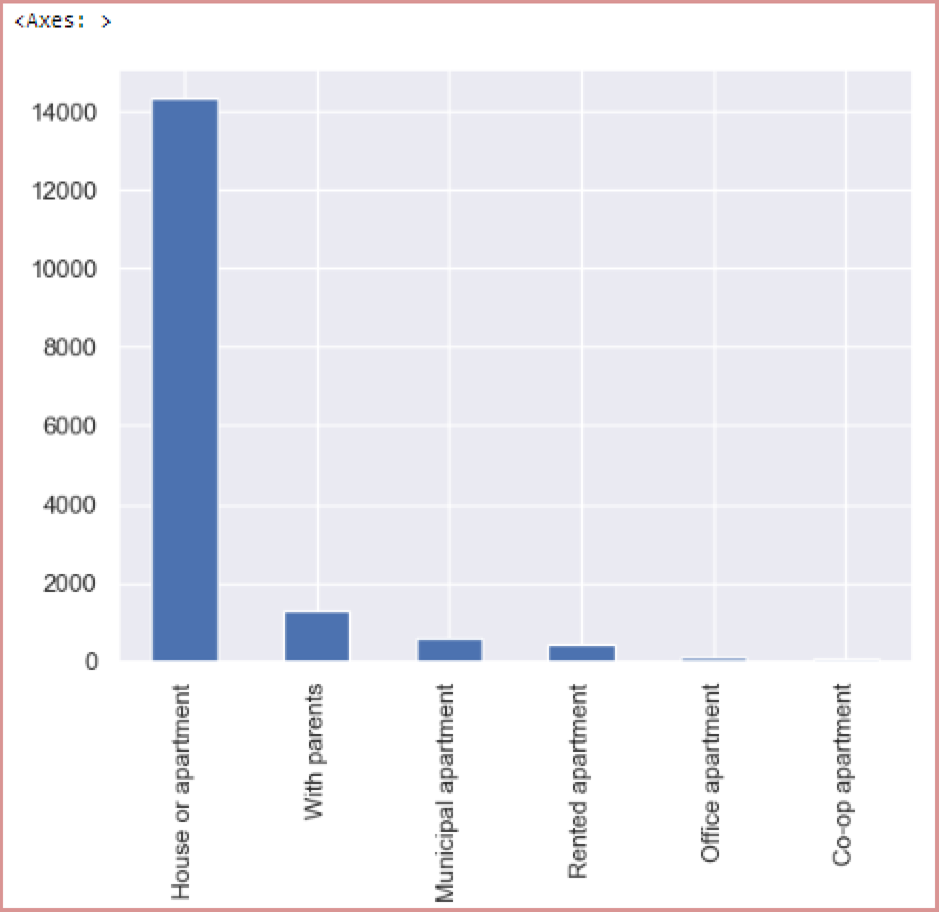
- ndef_=app_df[app_df['TARGET']==0]
  ndef_[**'NAME_INCOME_TYPE'**].value_counts().plot(kind='bar')

NON-DEFAULTER


```
Axes: >
```

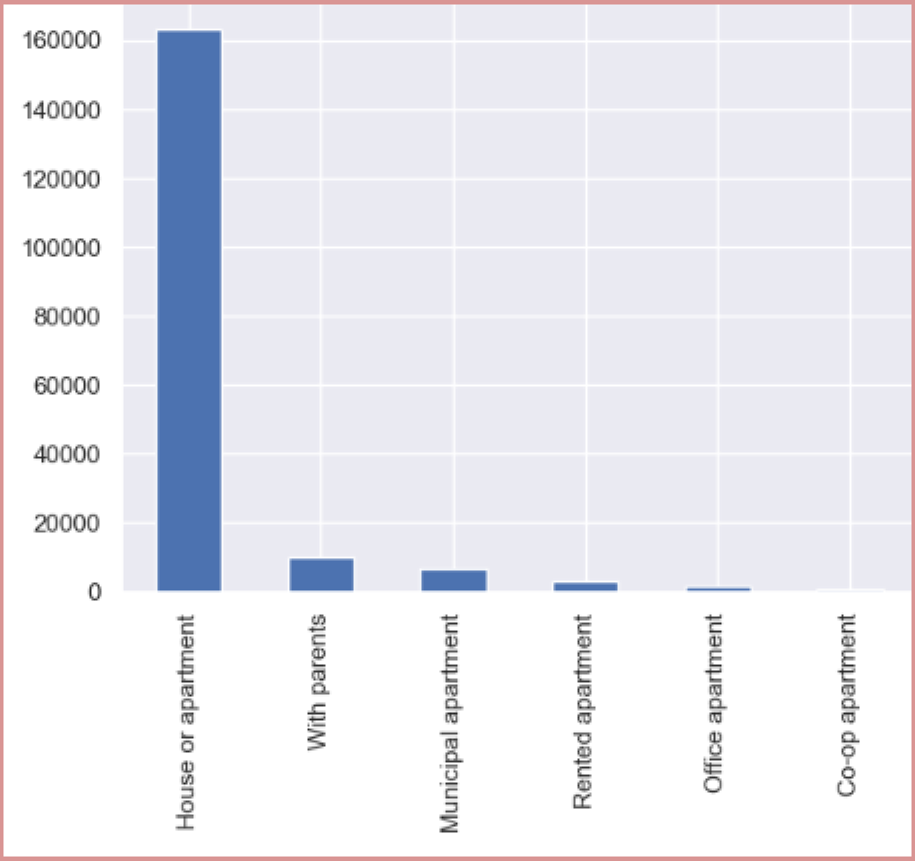
- def_=app_df[app_df['TARGET']==1]
  def_['**NAME_HOUSING_TYPE**'].value_counts().plot(kind='bar')

DEFAULTER


```
<Axes: >
```

- ndef_=app_df[app_df['TARGET']==0]
  ndef_['**NAME_HOUSING_TYPE**'].value_counts().plot(kind='bar')

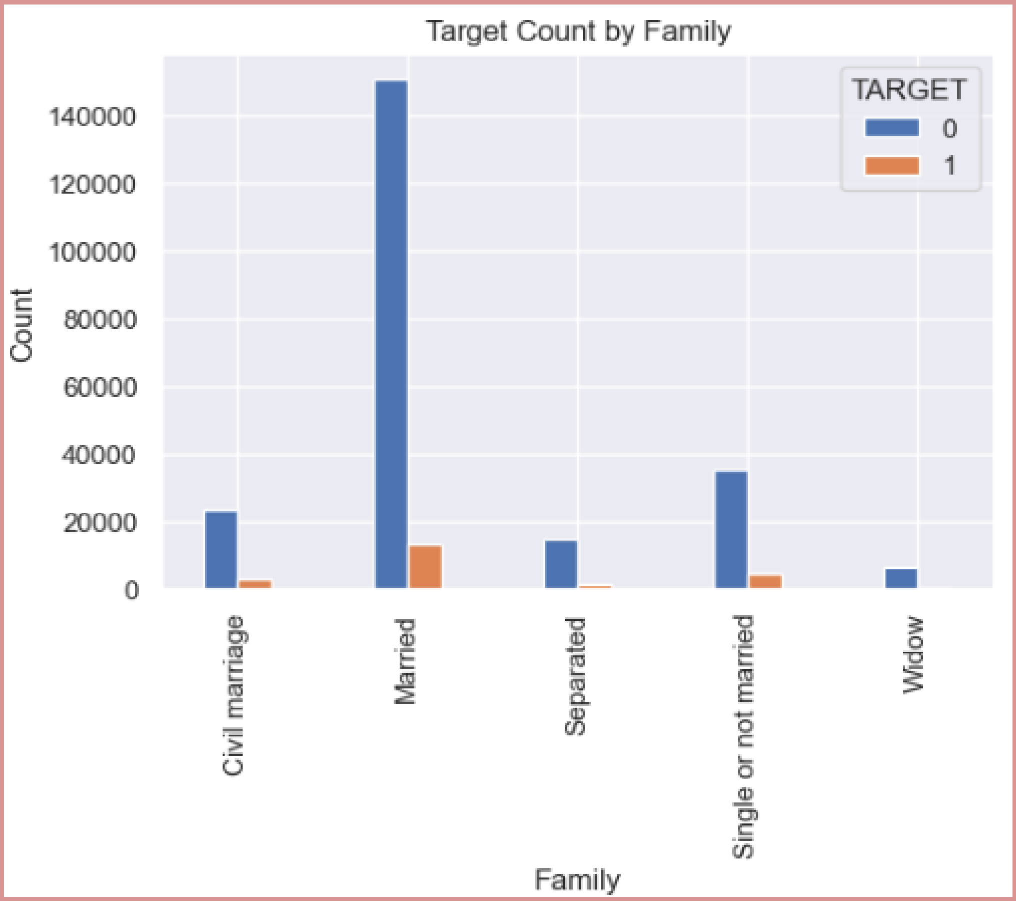
NON-DEFAULTER



2. Bivariate Analysis

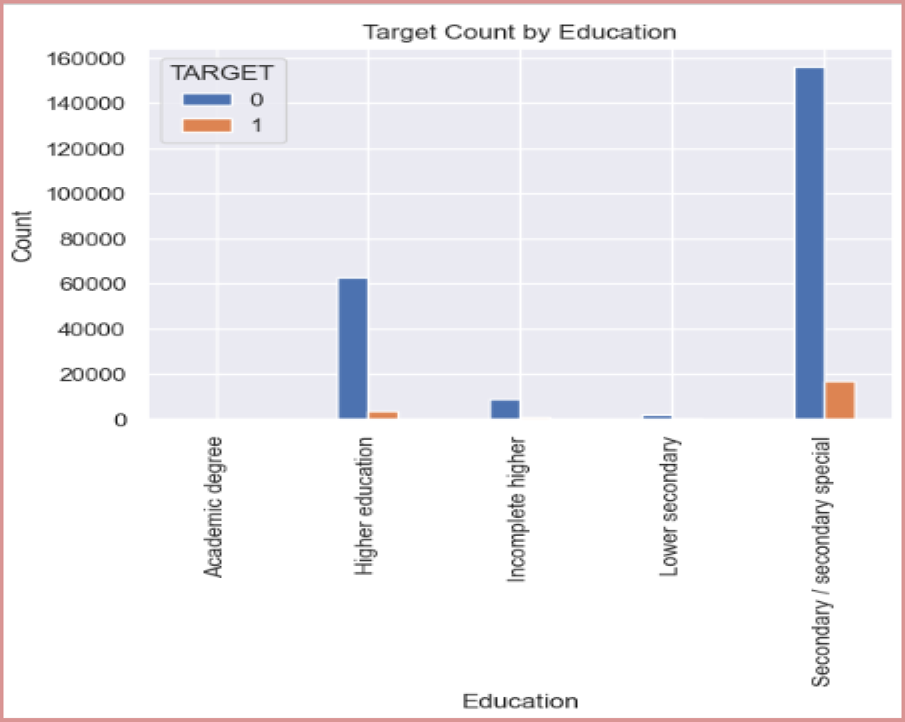
1. grouped = app_df.groupby(['**NAME_FAMILY_STATUS**', 'TARGET']).
size().unstack()
fig, ax = plt.subplots(figsize=(10, 6))
grouped.plot (kind = 'bar', width = 0.4 , ax=ax)

ax.set_xlabel('Family')
ax.set_ylabel('Count')
ax.set_title('Target Count by Family')
plt.show()



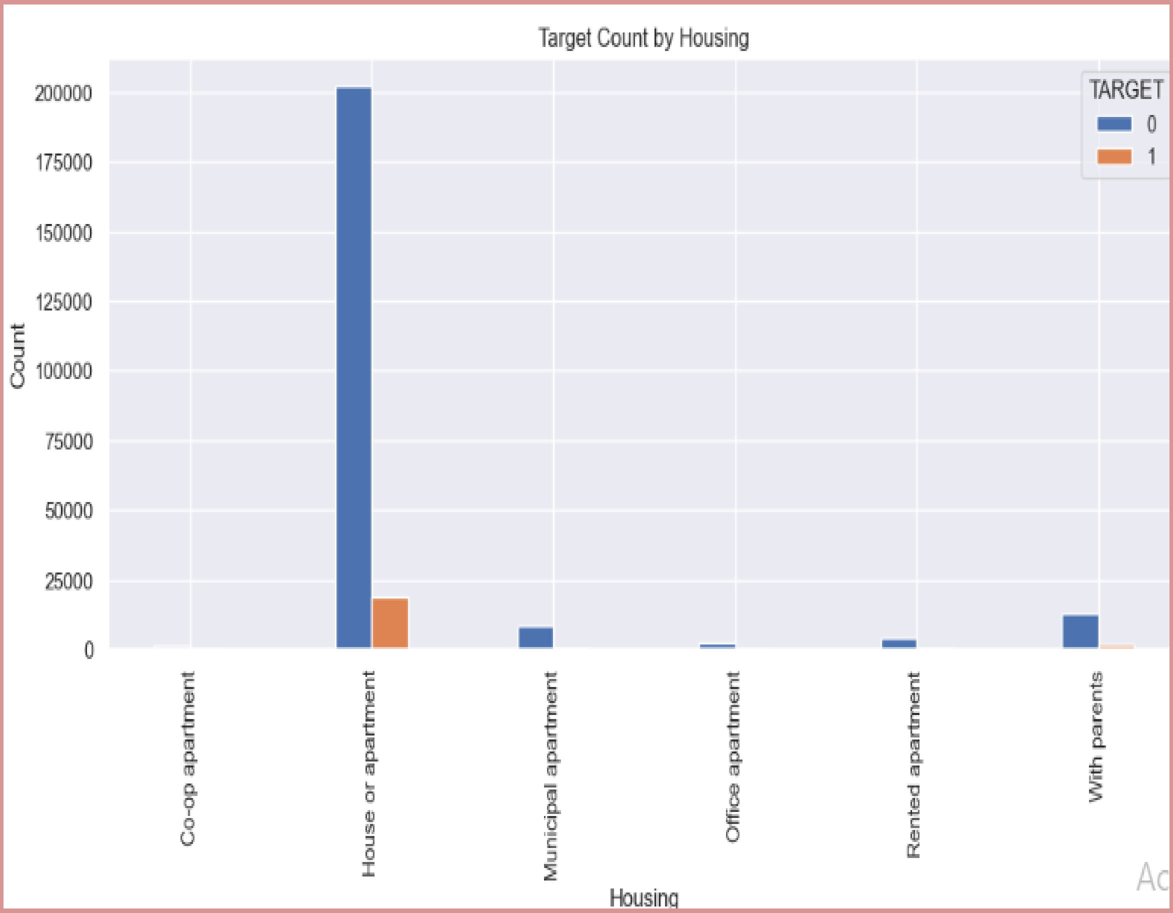
2. grouped = app_df.groupby([**'NAME_EDUCATION_TYPE'**, 'TARGET'])
.size().unstack()
fig, ax = plt.subplots(figsize=(6, 4))
grouped.plot(kind='bar', width=0.4, ax=ax)

ax.set_xlabel('Education')
ax.set_ylabel('Count')
ax.set_title('Target Count by Education')
plt.show()



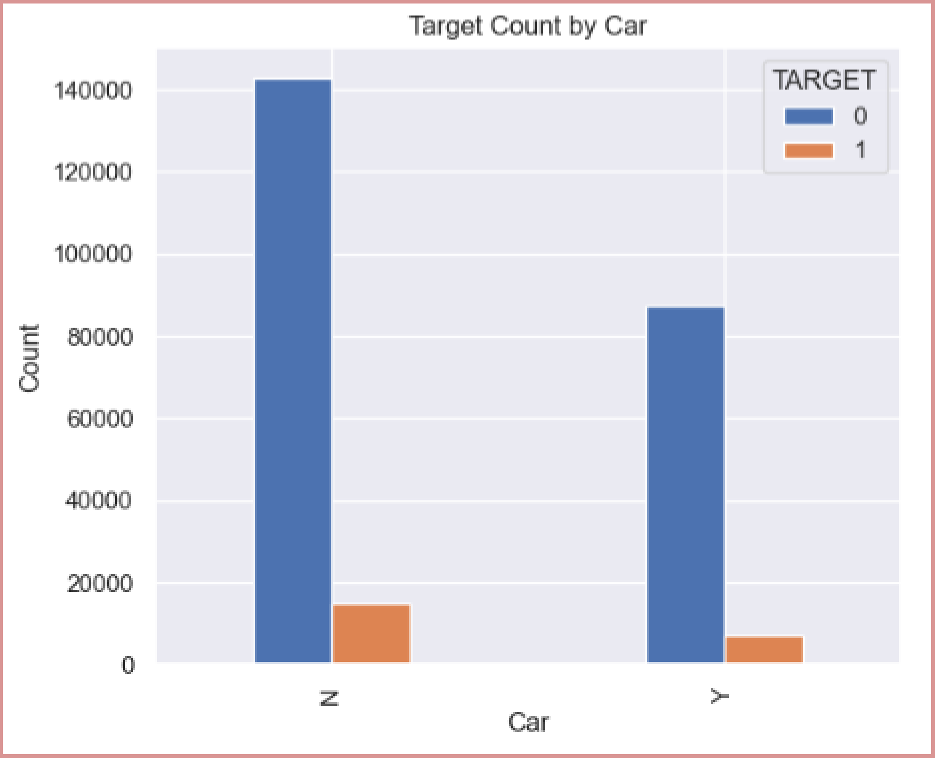
3. grouped = app_df.groupby([**'NAME_HOUSING_TYPE'**, 'TARGET'])
.size().unstack()

```
fig, ax = plt.subplots(figsize=(12, 5))

grouped.plot(kind='bar', width=0.4, ax=ax)


ax.set_xlabel('Housing')
ax.set_ylabel('Count')
ax.set_title('Target Count by Housing')
plt.show()
```



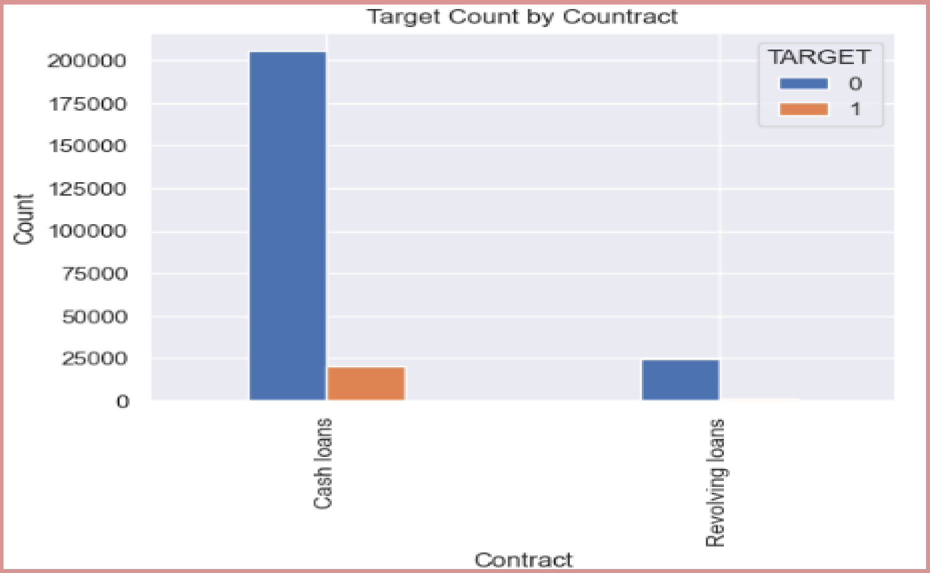
4. grouped = app_df.groupby(['**FLAG_OWN_CAR**', 'TARGET']).size().unstack()
   fig, ax = plt.subplots(figsize=(6, 5))
   grouped.plot(kind='bar', width=0.4, ax=ax)

   ax.set_xlabel('Car')
   ax.set_ylabel('Count')
   ax.set_title('Target Count by Car')
   plt.show()



5. grouped = app_df.groupby(['**NAME_CONTRACT_TYPE**', 'TARGET']).size().unstack()
   fig, ax = plt.subplots(figsize=(6, 4))
   grouped.plot(kind='bar', width=0.4, ax=ax)

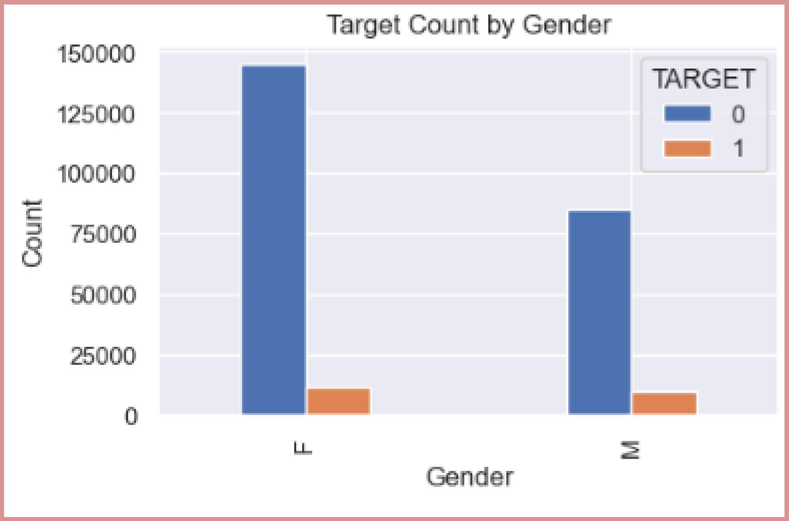
   ax.set_xlabel('Contract')
   ax.set_ylabel('Count')
   ax.set_title('Target Count by Contract')
   plt.show()



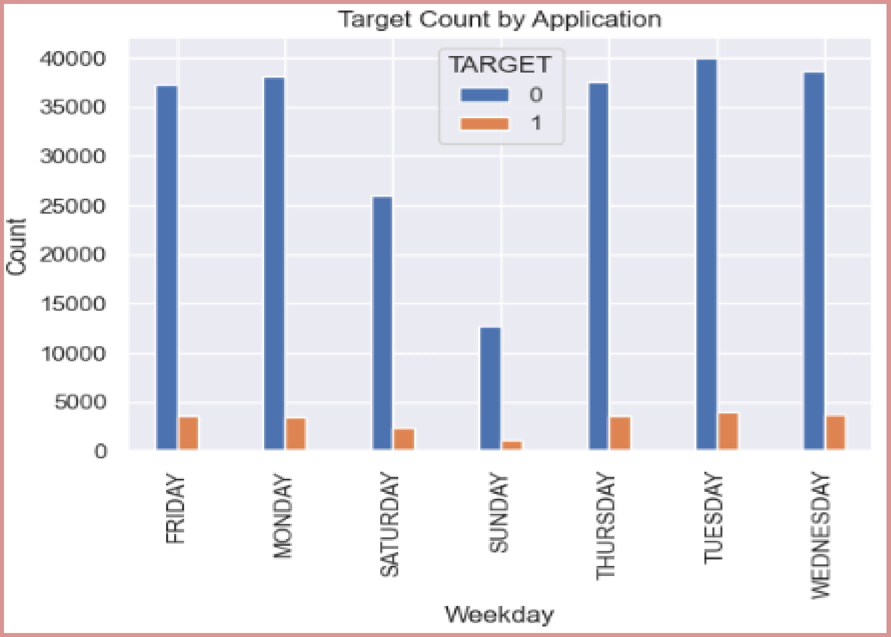
6. grouped = app_df.groupby(['**CODE_GENDER**',
   'TARGET']).size().unstack()
   fig, ax = plt.subplots(figsize=(5, 3))
   grouped.plot(kind='bar', width=0.4, ax=ax)

   ax.set_xlabel('Gender')
   ax.set_ylabel('Count')
   ax.set_title('Target Count by Gender')

   plt.show()



```
7. grouped = app_df.groupby(['WEEKDAY_APPR_PROCESS_START',
   'TARGET']).size().unstack()
   fig, ax = plt.subplots(figsize=(6, 4))
   grouped.plot(kind='bar', width=0.4, ax=ax)

   ax.set_xlabel('Weekday')
   ax.set_ylabel('Count')
   ax.set_title('Target Count by Application')
   plt.show()
```



3. Segmented Analysis

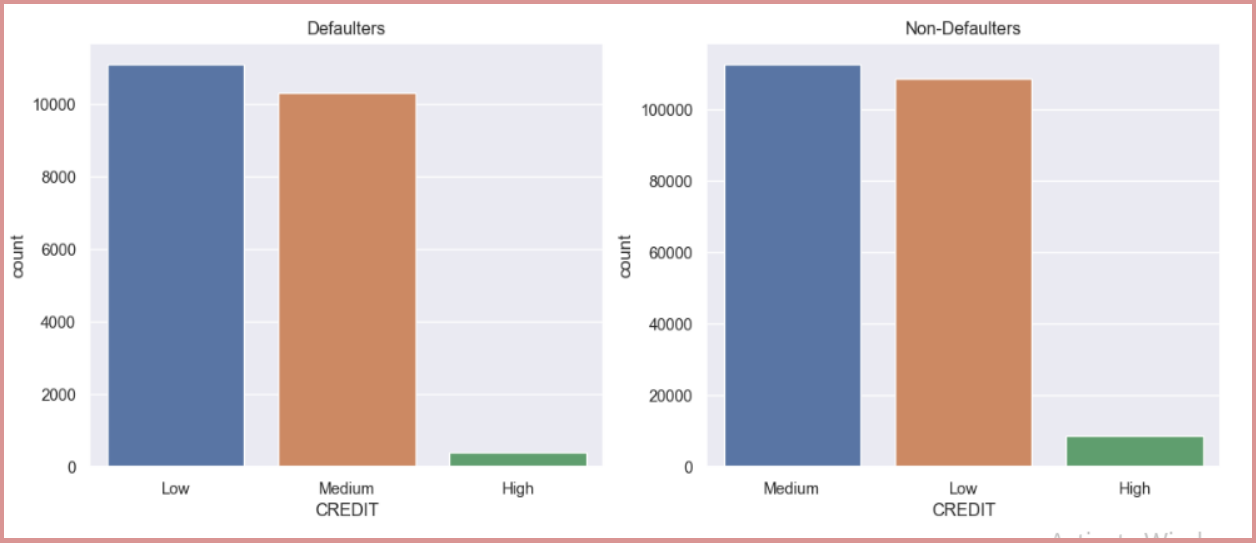
1. AMT_CREDIT

```
def credit(x):
    if (x < 500000):
        return 'Low'
    elif (x >= 500000 and x < 750000):
        return 'Medium'
    else:
        return 'High'
```

```
app_df['CREDIT'] = app_df['AMT_CREDIT'].apply(credit)
plt.figure(figsize=(14,5))
```

```
plt.subplot(1,2,1)
ax = sns.countplot(x = 'CREDIT',data=df_target_1)
plt.title('Defaulters')
```

```
plt.subplot(1,2,2)
ax = sns.countplot(x = 'CREDIT',data=df_target_0)
plt.title('Non-Defaulters')
```



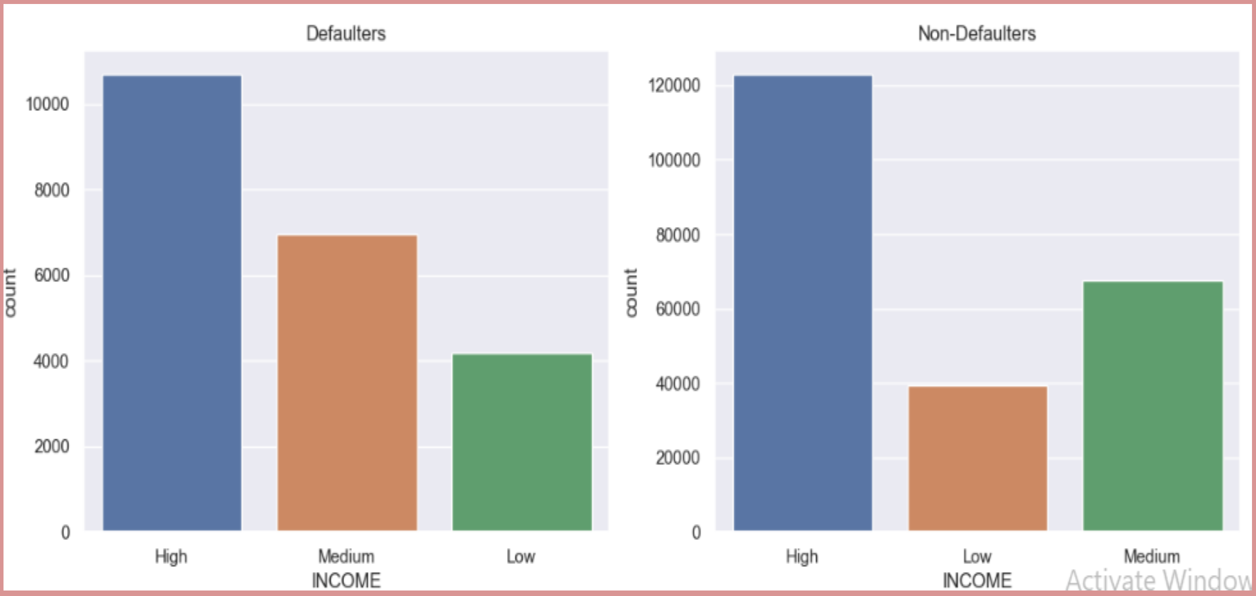
2. AMT_INCOME

```
def income(x):
    if (x < 100000):
        return 'Low'
    elif(x >= 100000 and x < 150000):
        return 'Medium'
    else:
        return 'High'
```

- app_df['INCOME'] = app_df['AMT_INCOME_TOTAL'].apply(income)
  plt.figure(figsize=(14,5))

- plt.subplot(1,2,1)
  ax = sns.countplot(x = 'INCOME',data=df_target_1)
  plt.title('Defaulters')

- plt.subplot(1,2,2)
  ax = sns.countplot(x = 'INCOME',data=df_target_0)
  plt.title('Non-Defaulters')



6. **Find the top 10 correlation for the Client with payment difficulties and all other cases (Target variable). Note that you have to find the top correlation by segmenting the data frame w.r.t to the target variable and then find the top correlation for each of the segmented data and find if any insight is there. Say, there are 5+1(target) variables in a dataset: Var1, Var2, Var3, Var4, Var5, Target. And if you have to find top 3 correlation, it can be: Var1 & Var2, Var2 & Var3, Var1 & Var3. Target variable will not feature in this correlation as it is a categorical variable and not a continuous variable which is increasing or decreasing.**

Ans 6.

- **col** = ['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_ OWN_ REALTY' , 'CNT_CHILDREN' , 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS' ,'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'OCCUPATION_TYPE', 'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION' , 'LIVE_REGION_NOT_WORK_REGION','REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_TYPE','OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE']

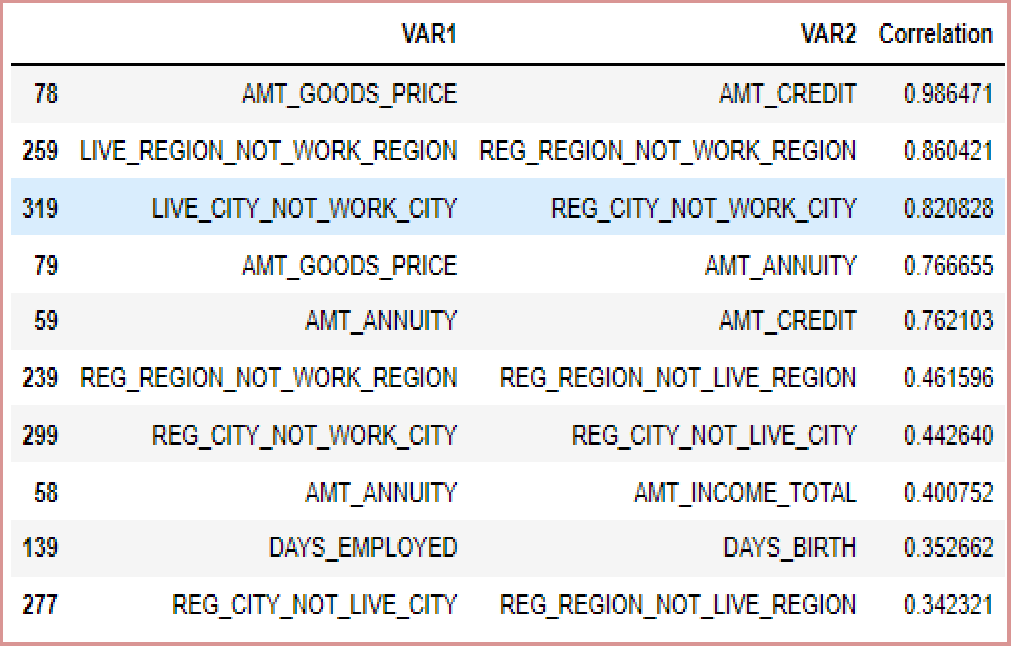
  - **Non_Deafulter** = app_df.loc[app_df['TARGET']==0, col]
  - **Defaulter** = app_df.loc[app_df['TARGET']==1, col]

1. **TARGET 0**

```
corr_non_deafulter = non_deafulter.corr()

corr_non_deafulter =
corr_non_deafulter.where(np.triu(np.ones(corr_non_deafulter.shape),k=1).a
stype(bool)).unstack().reset_index()

corr_non_deafulter.columns =['VAR1','VAR2','Correlation']

corr_non_deafulter.dropna(subset = ["Correlation"], inplace = True)

corr_non_deafulter["Correlation"]=corr_non_deafulter["Correlation"].abs()

corr_non_deafulter.sort_values(by='Correlation', ascending=False,
inplace=True)
```

- **corr_non_deafulter.head(10)**

Non-Deafulter

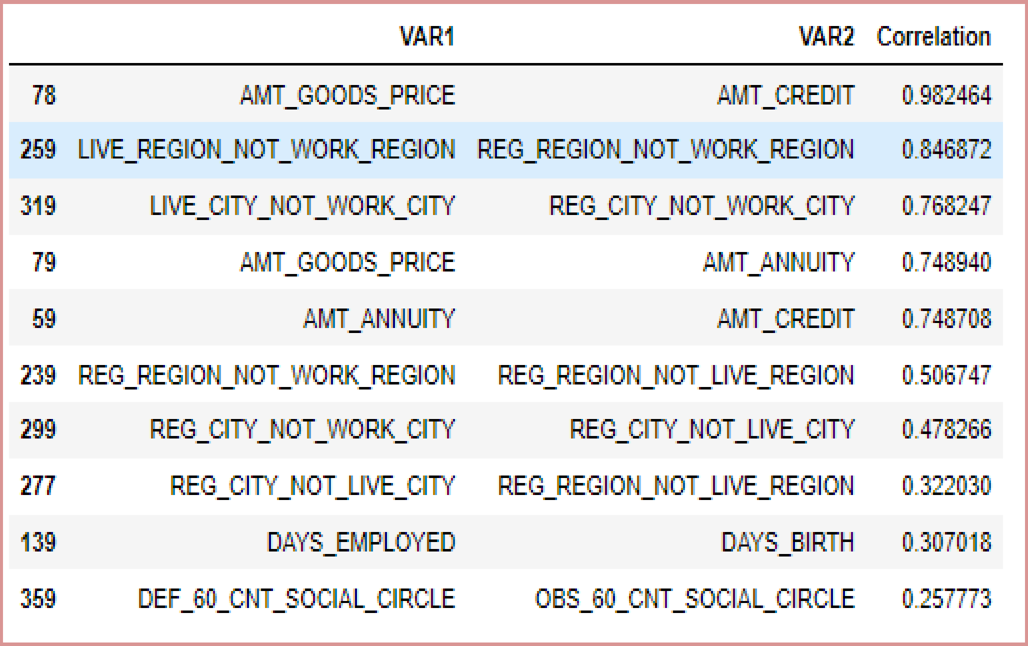


	VAR1	VAR2	Correlation
78	AMT_GOODS_PRICE	AMT_CREDIT	0.986471
259	LIVE_REGION_NOT_WORK_REGION	REG_REGION_NOT_WORK_REGION	0.860421
319	LIVE_CITY_NOT_WORK_CITY	REG_CITY_NOT_WORK_CITY	0.820828
79	AMT_GOODS_PRICE	AMT_ANNUITY	0.766655
59	AMT_ANNUITY	AMT_CREDIT	0.762103
239	REG_REGION_NOT_WORK_REGION	REG_REGION_NOT_LIVE_REGION	0.461596
299	REG_CITY_NOT_WORK_CITY	REG_CITY_NOT_LIVE_CITY	0.442640
58	AMT_ANNUITY	AMT_INCOME_TOTAL	0.400752
139	DAYS_EMPLOYED	DAYS_BIRTH	0.352662
277	REG_CITY_NOT_LIVE_CITY	REG_REGION_NOT_LIVE_REGION	0.342321

2. Target 1

```
corr_Defaulter = Defaulter.corr()

corr_Defaulter = corr_Defaulter.where(np.triu(np.ones(corr_Defaulter.shape),k=1).astype(bool))

corr_df_Defaulter = corr_Defaulter.unstack().reset_index()

corr_df_Defaulter.columns =['VAR1','VAR2','Correlation']

corr_df_Defaulter.dropna(subset = ["Correlation"], inplace = True)

corr_df_Defaulter["Correlation"]=corr_df_Defaulter["Correlation"].abs()

corr_df_Defaulter.sort_values(by='Correlation', ascending=False, inplace=True)

corr_df_Defaulter.head(10)
```

Defaulter

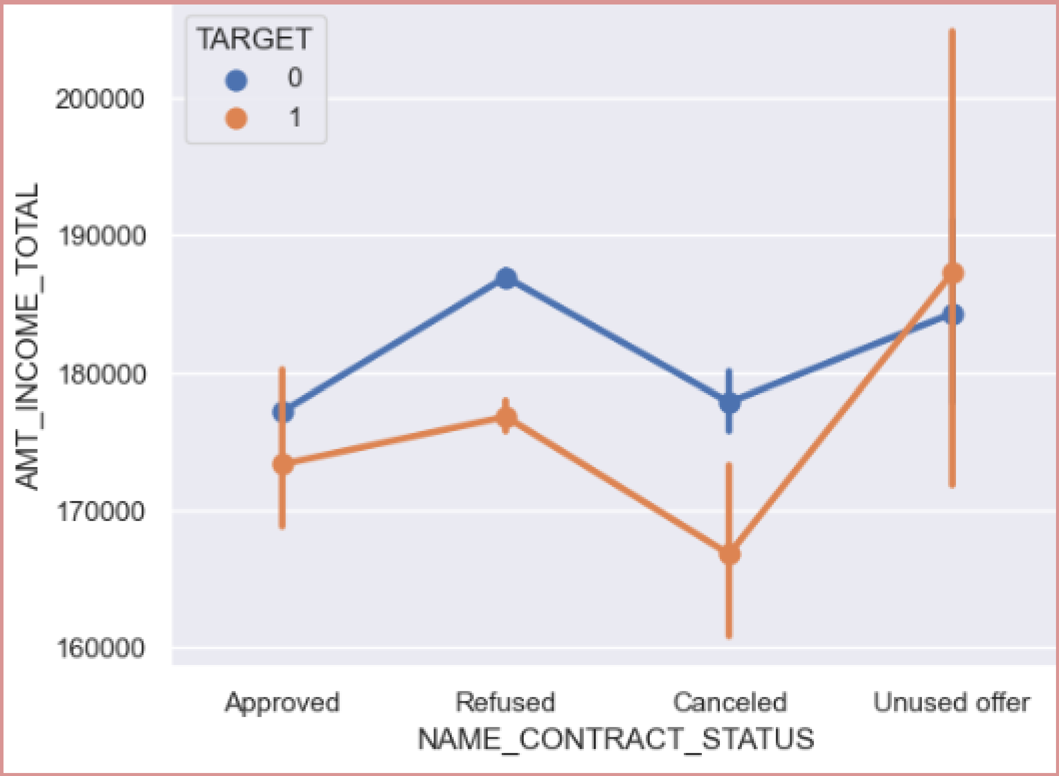


	VAR1	VAR2	Correlation
78	AMT_GOODS_PRICE	AMT_CREDIT	0.982464
259	LIVE_REGION_NOT_WORK_REGION	REG_REGION_NOT_WORK_REGION	0.846872
319	LIVE_CITY_NOT_WORK_CITY	REG_CITY_NOT_WORK_CITY	0.768247
79	AMT_GOODS_PRICE	AMT_ANNUITY	0.748940
59	AMT_ANNUITY	AMT_CREDIT	0.748708
239	REG_REGION_NOT_WORK_REGION	REG_REGION_NOT_LIVE_REGION	0.506747
299	REG_CITY_NOT_WORK_CITY	REG_CITY_NOT_LIVE_CITY	0.478266
277	REG_CITY_NOT_LIVE_CITY	REG_REGION_NOT_LIVE_REGION	0.322030
139	DAYS_EMPLOYED	DAYS_BIRTH	0.307018
359	DEF_60_CNT_SOCIAL_CIRCLE	OBS_60_CNT_SOCIAL_CIRCLE	0.257773

7. **Include visualizations and summarize the most important results in the presentation. You are free to choose the graphs which explain the numerical/categorical variables. Insights should explain why the variable is important for differentiating the clients with payment difficulties with all other cases.**

Ans 7.

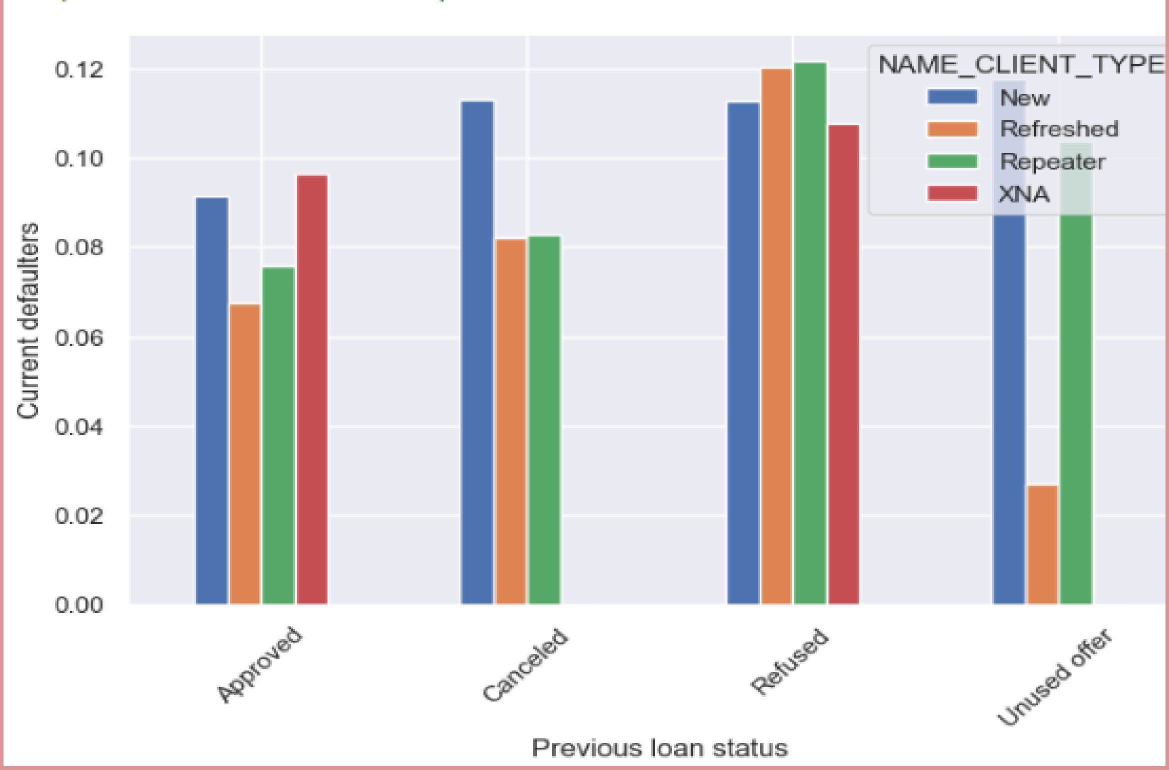
1. sns.pointplot(data=com_df, x="**NAME_CONTRACT_STATUS**",y="**AMT_INCOME_TOTAL**",hue="**TARGET**")



- The point plot indicates that individuals who did not use the offer previously defaulted despite having higher incomes than others.

2.
   ```
   com_df.pivot_table(values='TARGET',index='NAME_CONTRACT_STATUS',columns='NAME_CLIENT_TYPE',aggfunc='mean').plot.bar(figsize=(8,5),rot=45)
   plt.xlabel('Previous loan status')
   plt.ylabel('Current defaulters')
   ```

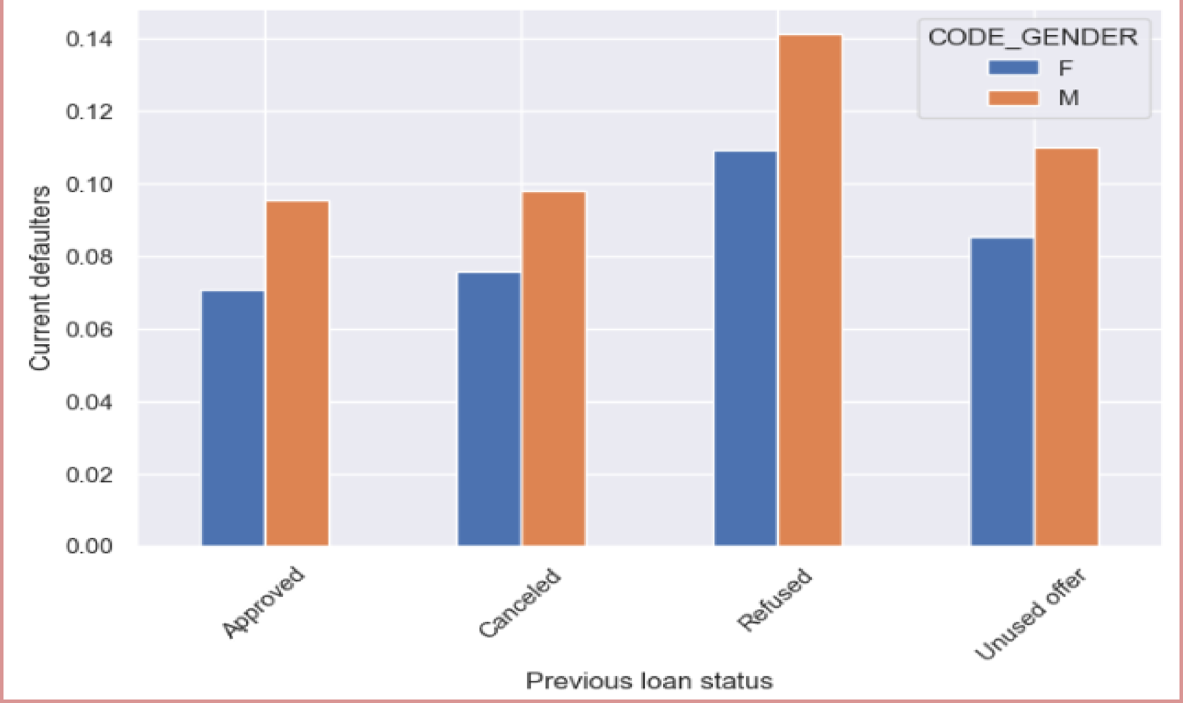
```
Text(0, 0.5, 'Current defaulters')
```



- We can see that the Defaulters are more for previously Unused offers loan status clients, who were New.
- For previously Approved status the New clients were more defaulted followed by Repeater.
- For previously Refused applicants the Defaulters are more Refreshed clients.
- For previously Canceled applicants the Defaulters are more New clients.

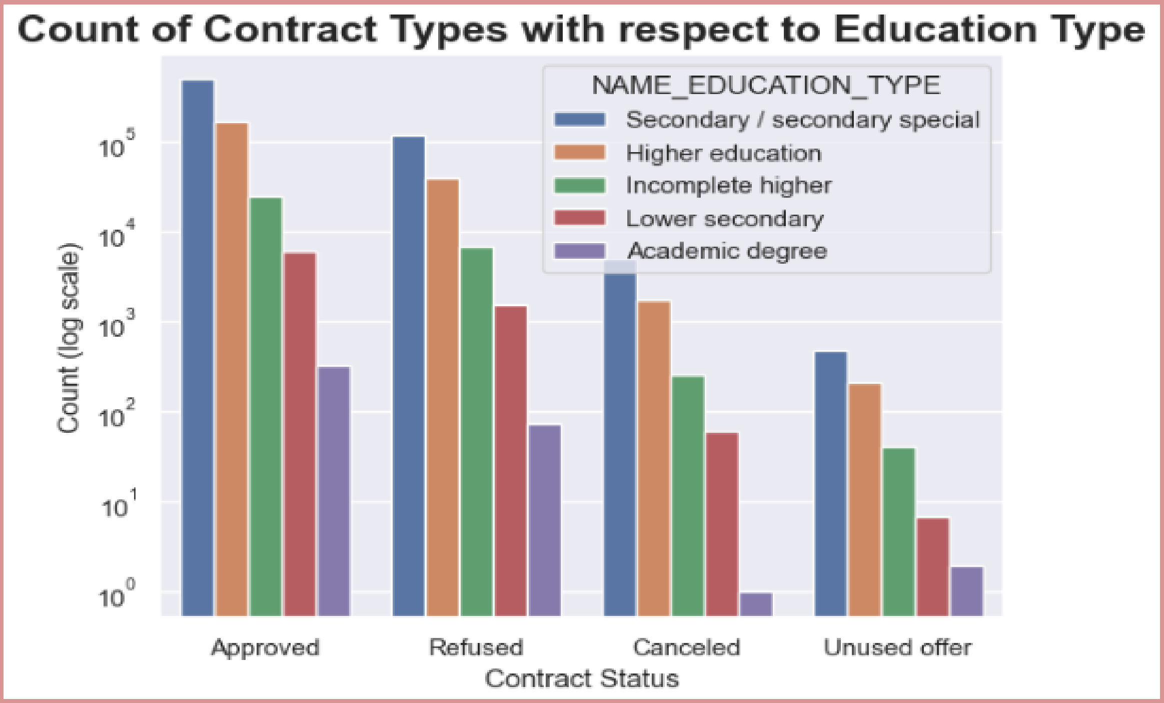
3. com_df.pivot_table(values=**'TARGET'**,index=**'NAME_CONTRACT_STATUS'**,columns=**'CODE_GENDER'**,aggfunc='mean').plot.bar(figsize=(8,5),rot=45)
   plt.xlabel('Previous loan status')
   plt.ylabel('Current defaulters')

```
Text(0, 0.5, 'Current defaulters')
```



- We see that previously Refused client is more defaulted than previously Approved clients. Also, in all the cases the Males are more defaulted than Females.

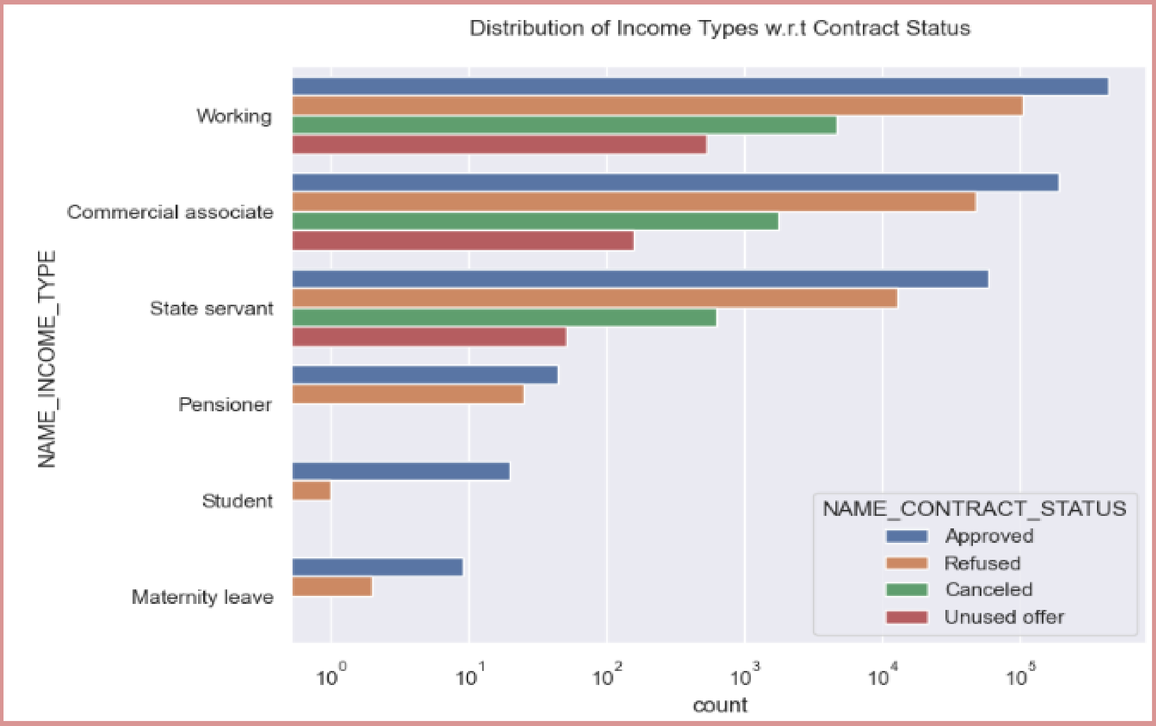
4. sns.countplot(x='NAME_CONTRACT_STATUS', hue= 'NAME_EDUCATION_TYPE' , data=com_df)
   plt.title('Count of Contract Types with respect to Education Type', fontsize=18, fontweight='bold')
   plt.xlabel('Contract Status', fontsize=12)
   plt.ylabel('Count (log scale)', fontsize=12
   plt.yscale('log')
   plt.show()



- Most clients with all kinds of education types apply mostly for cash loans rather than revolving loans.

5. sns.countplot(data=com_df,y='NAME_INCOME_TYPE',order=com_df['
   NAME_INCOME_TYPE'].value_counts().index,hue='NAME_CONTRACT_
   STATUS')
   plt.title('Distribution of Income Types w.r.t Contract Status\n')
   plt.xlabel("count")
   plt.ylabel("NAME_INCOME_TYPE")
   plt.xscale('log')plt.show()

- There are no nused offers for students and pensioners
- The number of approved loans for state servants is almost equal to the refusal or canceled loans for Commercial associates
- Maximum unused offers is by working clients



PROJECT LINK

https://drive.google.com/file/d/1-yOW-_efKtTeejhGS7KpghsZUyxIUrMf/view?usp=share_link

INSIGHTS

A. Decisive Factor whether an applicant will be **Repayer:**

1. **NAME_EDUCATION_TYPE:** Academic degree has less defaults.
2. **NAME_INCOME_TYPE:** Student and Businessmen have no defaults.
3. **ORGANIZATION_TYPE:** Clients with Trade Type 4 and 5 and Industry type 8 have defaulted less than 3%
4. **AMT_INCOME_TOTAL:** Applicant with Income more than 700,000 are less likely to default.
5. **CNT_CHILDREN:** People with zero to two children tend to repay the loans.

B. Decisive Factor whether an applicant will be **Defaulter:**

1. **CODE_GENDER:** Men are at relatively higher default rate
2. **NAME_FAMILY_STATUS :** People who have civil marriage or who are single default a lot.
3. **NAME_EDUCATION_TYPE:** People with Lower Secondary & Secondary education.
4. **NAME_INCOME_TYPE:** Clients who are either at Maternity leave OR Unemployed default a lot.
5. **OCCUPATION_TYPE:** Avoid Low-skill Laborers, Drivers and Waiters/barmen staff, Security staff, Laborers and Cooking staff as their default rate is huge.
6. **CNT_CHILDREN :** Client who have children equal to or more than 9 default 100% and hence their applications are to be rejected.
7. **AMT_GOODS_PRICE:** When the credit amount goes beyond 3lakhs, there is an increase in defaulters.

SUGGESSTIONS

- 90% of the previously cancelled client have actually repayed the loan. Record the reason for cancellation which might help the bank to determine and negotiate terms with these repaying customers in future for increase business opportunity.
- 85% of the clients who were refused by bank for loan earlier have now turned into a repaying client. Hence documenting the reason for rejection could mitigate the business loss and these clients could be contacted for further loans.

RESULTS

1. **NAME_HOUSING_TYPE:** High number of loan applications are from the category of people who live in Rented apartments & living with parents and hence offering the loan would mitigate the loss if any of those default.
2. **AMT_CREDIT:** People who get loan for 3-6 Lakhs tend to default more than others and hence having higher interest specifically for this credit range would be ideal.
3. **AMT_INCOME:** Since 90% of the applications have Income total less than 3Lakhs and they have high probability of defaulting, they could be offered loan with higher interest compared to other income category.
4. **CNT_CHILDREN:** Clients who have 4 to 8 children has a very high default rate and hence higher interest should be imposed on their loans.

THANK YOU