Name: Naim Kalekhan

Description:

Imagine 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.

In [ ]:

Aim:

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.

In [ ]:

Dataset Overview:

The dataset provides details about the current loan applications like the type of contract, annuity amount, credit amount etc.

The Dataset details are:

- Number of Data-Points: 49,999
- Number of Features: 122

Import required libraries

In [372...

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Read the dataset

In [373...

```
df= pd.read_csv(r"C:\Users\Fahim\Downloads\application_data.csv")
```

Understand the Dataset

In [374...

```
df.head()
```

Out[374...

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REAL
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100004	0	Revolving loans	M	Y	

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REAL
3	100006	0	Cash loans	F	N	
4	100007	0	Cash loans	M	N	

5 rows × 122 columns

In [375...
```
df.shape
```

Out[375...
```
(49999, 122)
```

In [376...
```
df.isnull().sum()
```

Out[376...
```
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
```

In [377...
```
df.describe()
```

Out[377...

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNU
count	49999.000000	49999.000000	49999.000000	4.999900e+04	4.999900e+04	49998.00C
mean	129013.210584	0.080522	0.419848	1.707676e+05	5.997006e+05	27107.377
std	16690.512048	0.272102	0.724039	5.318191e+05	4.024154e+05	14562.944
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	2052.00C
25%	114570.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16456.50C
50%	129076.000000	0.000000	0.000000	1.458000e+05	5.147775e+05	24939.00C
75%	143438.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.00C
max	157875.000000	1.000000	11.000000	1.170000e+08	4.050000e+06	258025.50C

8 rows × 106 columns

In [378...
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49999 entries, 0 to 49998
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(64), int64(42), object(16)
memory usage: 46.5+ MB
```

In [379...
```
print(df.isnull().sum().head(50))
```

```
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                          1
AMT_GOODS_PRICE                     38
NAME_TYPE_SUITE                    192
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
OWN_CAR_AGE                      32950
FLAG_MOBIL                           0
FLAG_EMP_PHONE                       0
FLAG_WORK_PHONE                      0
FLAG_CONT_MOBILE                     0
FLAG_PHONE                           0
FLAG_EMAIL                           0
OCCUPATION_TYPE                  15654
CNT_FAM_MEMBERS                      1
REGION_RATING_CLIENT                 0
REGION_RATING_CLIENT_W_CITY          0
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_1                     28172
EXT_SOURCE_2                       126
EXT_SOURCE_3                      9944
APARTMENTS_AVG                   25385
BASEMENTAREA_AVG                 29199
YEARS_BEGINEXPLUATATION_AVG      24394
YEARS_BUILD_AVG                  33239
COMMONAREA_AVG                   34960
ELEVATORS_AVG                    26651
dtype: int64
```

Data Preprocessing

Check for the duplicates

In [380…

```
df[df['SK_ID_CURR'].duplicated()]
```

Out[380…

SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALT

0 rows × 122 columns

As per above result we can say that, there is no duplicate loan application id's

Feature Engineering

We created new columns which showed duration in years. We also created columns which showed one feature's value as percentage of another feature's value.

In [381...

```
df['AGE']=df['DAYS_BIRTH'].abs().div(365).round(3)
df['EXP_YRS']=df['DAYS_EMPLOYED'].abs().div(365).round(3)
df['REG_YRS']=df['DAYS_REGISTRATION'].abs().div(365).round(3)
df['REG_YRS_ID']=df['DAYS_ID_PUBLISH'].abs().div(3365).round(3)
```

In [382...

```
df['CREDIT%INCOME'] = df['AMT_CREDIT'].div(df['AMT_INCOME_TOTAL']).multiply(100).rou
df['AMUNITY%INCOME'] = df['AMT_ANNUITY'].div(df['AMT_INCOME_TOTAL']).multiply(100).r
df['AMUNITY%CREDIT'] = df['AMT_ANNUITY'].div(df['AMT_CREDIT']).multiply(100).round(3
df['CREDIT%GOODS_PRICE']=df['AMT_CREDIT'].div(df['AMT_GOODS_PRICE']).multiply(100).r
```

We have successfully made changes accordingly. And hence need to drop respective unwanted columns from the original dataset.

Drop columns DAYS_BIRTH, DAYS_EMPLOYED, DAYS_REGISTRATION and DAYS_ID_PUBLISH as we have created equivalent columns

In [383...

```
df = df.drop(['DAYS_BIRTH','DAYS_EMPLOYED','DAYS_REGISTRATION','DAYS_ID_PUBLISH'],ax
```

In [384...

```
df.head(5)
```

Out[384...

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REAL
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100004	0	Revolving loans	M	Y	
3	100006	0	Cash loans	F	N	
4	100007	0	Cash loans	M	N	

5 rows × 126 columns

In [ ]:

Data Analytics Tasks:

A.

Identify Missing Data and Deal with it Appropriately: As a data analyst, you come across missing data in the loan application dataset. It is essential to handle missing data effectively to ensure the accuracy of the analysis.

Task: Identify the missing data in the dataset and decide on an appropriate method to deal with it using Excel built-in functions and features.

In [ ]:

1. Drop Unwanted columns from the dataset.

In [385...

```
df = df.drop(['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',
                 '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','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','REGION_RATING_CLIENT',
                 'REGION_RATING_CLIENT_W_CITY','SK_ID_CURR'], axis=1)
```

1. Now,we are going to drop columns which has missing values / null values more than 40 percentage

In [386…

```
l=df.columns[df.isnull().mean()>0.40]
l
```

Out[386…

```
Index(['OWN_CAR_AGE', 'EXT_SOURCE_1', 'APARTMENTS_AVG', 'BASEMENTAREA_AVG',
       'YEARS_BEGINEXPLUATATION_AVG', 'YEARS_BUILD_AVG', 'COMMONAREA_AVG',
       'ELEVATORS_AVG', 'ENTRANCES_AVG', 'FLOORSMAX_AVG', 'FLOORSMIN_AVG',
       'LANDAREA_AVG', 'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_AVG',
       'NONLIVINGAPARTMENTS_AVG', 'NONLIVINGAREA_AVG', 'APARTMENTS_MODE',
       'BASEMENTAREA_MODE', 'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BUILD_MODE',
       'COMMONAREA_MODE', 'ELEVATORS_MODE', 'ENTRANCES_MODE', 'FLOORSMAX_MODE',
       'FLOORSMIN_MODE', 'LANDAREA_MODE', 'LIVINGAPARTMENTS_MODE',
       'LIVINGAREA_MODE', 'NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAREA_MODE',
       'APARTMENTS_MEDI', 'BASEMENTAREA_MEDI', 'YEARS_BEGINEXPLUATATION_MEDI',
       'YEARS_BUILD_MEDI', 'COMMONAREA_MEDI', 'ELEVATORS_MEDI',
       'ENTRANCES_MEDI', 'FLOORSMAX_MEDI', 'FLOORSMIN_MEDI', 'LANDAREA_MEDI',
       'LIVINGAPARTMENTS_MEDI', 'LIVINGAREA_MEDI', 'NONLIVINGAPARTMENTS_MEDI',
       'NONLIVINGAREA_MEDI', 'FONDKAPREMONT_MODE', 'HOUSETYPE_MODE',
       'TOTALAREA_MODE', 'WALLSMATERIAL_MODE', 'EMERGENCYSTATE_MODE'],
      dtype='object')
```

In [387…

```
df = df.drop(l, axis = 1)
```

Result: We have sccessfully drop all the columns which has missing values / null values more than 40 percentage

In [ ]:

1. Now, check for the nulls present in the dataset after above 2 data cleaning steps.

In [388…

```
df.isnull().sum().sort_values()
```

Out[388…

```
TARGET                         0
ORGANIZATION_TYPE              0
AGE                            0
FLAG_DOCUMENT_2                0
FLAG_DOCUMENT_3                0
FLAG_DOCUMENT_4                0
FLAG_DOCUMENT_5                0
FLAG_DOCUMENT_6                0
FLAG_DOCUMENT_7                0
FLAG_DOCUMENT_8                0
HOUR_APPR_PROCESS_START        0
FLAG_DOCUMENT_9                0
FLAG_DOCUMENT_11               0
FLAG_DOCUMENT_12               0
FLAG_DOCUMENT_13               0
FLAG_DOCUMENT_14               0
FLAG_DOCUMENT_15               0
FLAG_DOCUMENT_16               0
FLAG_DOCUMENT_17               0
FLAG_DOCUMENT_18               0
FLAG_DOCUMENT_19               0
```

```
FLAG_DOCUMENT_10                 0
WEEKDAY_APPR_PROCESS_START       0
EXP_YRS                          0
REG_YRS                          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
CREDIT%INCOME                    0
REG_YRS_ID                       0
FLAG_DOCUMENT_20                 0
NAME_INCOME_TYPE                 0
NAME_FAMILY_STATUS               0
NAME_HOUSING_TYPE                0
REGION_POPULATION_RELATIVE       0
FLAG_MOBIL                       0
FLAG_EMP_PHONE                   0
FLAG_WORK_PHONE                  0
FLAG_CONT_MOBILE                 0
FLAG_PHONE                       0
FLAG_EMAIL                       0
NAME_EDUCATION_TYPE              0
FLAG_DOCUMENT_21                 0
AMUNITY%INCOME                   1
DAYS_LAST_PHONE_CHANGE           1
CNT_FAM_MEMBERS                  1
AMT_ANNUITY                      1
AMUNITY%CREDIT                   1
AMT_GOODS_PRICE                 38
CREDIT%GOODS_PRICE              38
NAME_TYPE_SUITE                192
OCCUPATION_TYPE              15654
dtype: int64
```

- As we can see there are still some nulls present in the columns AMUNITY%INCOME,DAYS_LAST_PHONE_CHANGE,CNT_FAM_MEMBERS,AMT_ANNUITY,AMUNIT
- We need to handle these nulls as well.

In [ ]:

1. For CODE_GENDER column there are some records with entry XNA. For this one we are going to replace these records with most repeatative entries i.e.F

In [389…

```
df['CODE_GENDER'].value_counts()
```

Out[389…

```
F      32823
M      17174
XNA        2
Name: CODE_GENDER, dtype: int64
```

In [390…

```
df['CODE_GENDER'].value_counts()
df['CODE_GENDER'] = df['CODE_GENDER'].replace('XNA','F')
```

1. Same method we are going to use in case of ORGANIZATION_TYPE filed. In this column there are some records with XNA entry. We are going to replace these records with 'Not Working'.

In [391…

```
df['ORGANIZATION_TYPE'].value_counts()
```

Out[391…

```
Business Entity Type 3    11101
XNA                        8924
Self-employed              6240
Other                      2717
Medicine                   1817
Government                 1716
Business Entity Type 2     1704
School                     1450
Trade: type 7              1210
Kindergarten               1090
Construction               1066
Business Entity Type 1      953
Transport: type 4           837
Trade: type 3               550
Security                    550
Industry: type 3            542
Industry: type 9            537
Industry: type 11           489
Housing                     489
Military                    458
Bank                        435
Transport: type 2           392
Agriculture                 392
Postal                      370
Police                      366
Security Ministries         331
Trade: type 2               307
Restaurant                  289
Services                    284
University                  222
Industry: type 7            209
Transport: type 3           191
Hotel                       182
Industry: type 1            159
Electricity                 147
Industry: type 4            140
Trade: type 6               108
Telecom                     106
Industry: type 5            103
Emergency                    93
Insurance                    89
Industry: type 2             78
Advertising                  68
Trade: type 1                66
Culture                      64
Realtor                      61
Mobile                       56
Industry: type 12            53
Legal Services               44
Cleaning                     40
Transport: type 1            28
Industry: type 10            21
Industry: type 13            15
Religion                     14
Industry: type 6             12
Industry: type 8              8
Trade: type 4                 8
Trade: type 5                 8
Name: ORGANIZATION_TYPE, dtype: int64
```

In [392…

```
df['ORGANIZATION_TYPE'].value_counts()
df['ORGANIZATION_TYPE']=df['ORGANIZATION_TYPE'].replace('XNA','Not Working')
```

- For null values in OCCUPATION_TYPE column, we replaced them with values which have the maximum count of the value of NAME_INCOME_TYPE and NAME_EDUCATION_TYPE of the corresponding null value. For the null values in OCCUPATION_TYPE column which are still present after above step, we replaced them with Laborers which is the most common Occupation type.

In [393…
```
df.OCCUPATION_TYPE.value_counts()
df.OCCUPATION_TYPE = df.OCCUPATION_TYPE.fillna('Laborers')
```

- For null values in NAME_TYPE_SUITE column, we replaced them with values which have the maximum count of the value of NAME_INCOME_TYPE and NAME_FAMILY_STATUS of the corresponding null value. For all the column values of NAME_INCOME_TYPE and NAME_FAMILY_STATUS, the most common column value of NAME_TYPE_SUITE is Unaccompanied. So replaced all null values of NAME_TYPE_SUITE with Unaccompanied.

In [394…
```
df.NAME_TYPE_SUITE.value_counts()
df.NAME_TYPE_SUITE = df.NAME_TYPE_SUITE.fillna('Unaccompanied')
```

- For null values in AMT_GOODS_PRICE column, we replaced them with the mean value of the AMT_GOODS_PRICE recrods.

In [395…
```
df.AMT_GOODS_PRICE.value_counts()
df.AMT_GOODS_PRICE = df.AMT_GOODS_PRICE.fillna(df.AMT_GOODS_PRICE.mean())
```

In [ ]:
```

```

- For the rest of the columns we have have replaced the null with Mean, Median or Mode, as per analysis.

In [396…
```
df.AMT_ANNUITY.value_counts()
df.AMT_ANNUITY = df.AMT_ANNUITY.fillna(df.AMT_ANNUITY.mean())
```

In [397…
```
df.CNT_FAM_MEMBERS.value_counts()
df.CNT_FAM_MEMBERS = df.CNT_FAM_MEMBERS.fillna('2.0')
```

In [398…
```
df.DAYS_LAST_PHONE_CHANGE.value_counts()
df.DAYS_LAST_PHONE_CHANGE=df.DAYS_LAST_PHONE_CHANGE.fillna('0.0')
```

In [399…
```
df['AMUNITY%INCOME'].value_counts()
df['AMUNITY%INCOME'] = df['AMUNITY%INCOME'].fillna(df['AMUNITY%INCOME'].mean())
```

In [400…
```
df['AMUNITY%CREDIT'].value_counts()
df['AMUNITY%CREDIT']=df['AMUNITY%CREDIT'].fillna(df['AMUNITY%CREDIT'].mean())
```

In [401…
```
df['CREDIT%GOODS_PRICE'].value_counts()
df['CREDIT%GOODS_PRICE']=df['CREDIT%GOODS_PRICE'].fillna(df['CREDIT%GOODS_PRICE'].me
```

- Now, we can check our dataset is cleaned.No nulls or any unwanted values are present in our dataset.

In [402…
```
df.isnull().sum()
```

Out[402…
```
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                    0
AMT_GOODS_PRICE                0
NAME_TYPE_SUITE                0
NAME_INCOME_TYPE               0
NAME_EDUCATION_TYPE            0
NAME_FAMILY_STATUS             0
NAME_HOUSING_TYPE              0
REGION_POPULATION_RELATIVE     0
FLAG_MOBIL                     0
FLAG_EMP_PHONE                 0
FLAG_WORK_PHONE                0
FLAG_CONT_MOBILE               0
FLAG_PHONE                     0
FLAG_EMAIL                     0
OCCUPATION_TYPE                0
CNT_FAM_MEMBERS                0
WEEKDAY_APPR_PROCESS_START     0
HOUR_APPR_PROCESS_START        0
ORGANIZATION_TYPE              0
DAYS_LAST_PHONE_CHANGE         0
FLAG_DOCUMENT_2                0
FLAG_DOCUMENT_3                0
FLAG_DOCUMENT_4                0
FLAG_DOCUMENT_5                0
FLAG_DOCUMENT_6                0
FLAG_DOCUMENT_7                0
FLAG_DOCUMENT_8                0
FLAG_DOCUMENT_9                0
FLAG_DOCUMENT_10               0
FLAG_DOCUMENT_11               0
FLAG_DOCUMENT_12               0
FLAG_DOCUMENT_13               0
FLAG_DOCUMENT_14               0
FLAG_DOCUMENT_15               0
FLAG_DOCUMENT_16               0
FLAG_DOCUMENT_17               0
FLAG_DOCUMENT_18               0
FLAG_DOCUMENT_19               0
FLAG_DOCUMENT_20               0
FLAG_DOCUMENT_21               0
AGE                            0
EXP_YRS                        0
REG_YRS                        0
REG_YRS_ID                     0
CREDIT%INCOME                  0
AMUNITY%INCOME                 0
AMUNITY%CREDIT                 0
CREDIT%GOODS_PRICE             0
dtype: int64
```

In [ ]:

B.

Identify Outliers in the Dataset: Outliers can significantly impact the analysis and distort the results. You need to identify outliers in the loan application dataset.

Task: Detect and identify outliers in the dataset using Excel statistical functions and features, focusing on numerical variables.

1. AGE

- For Outliers in DAYS_BIRTH column, we checked the maximum and minimum age. No issue found.

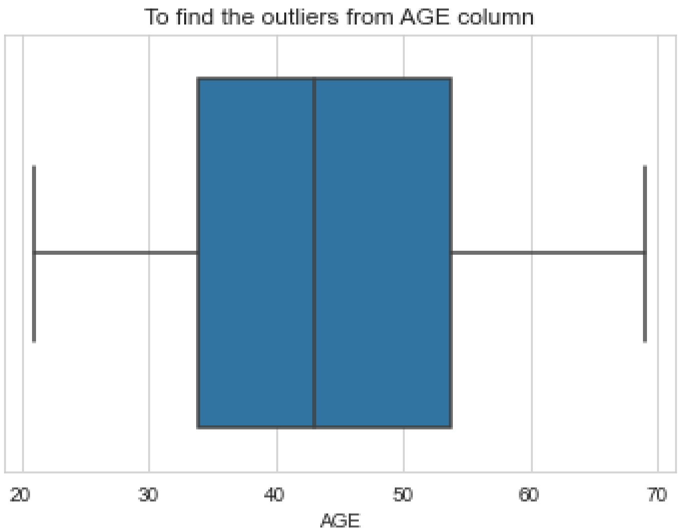
In [403…

```
print("Maximum Age of Cutomers = ",df['AGE'].max())
print("Minimum Age of customers = ",df['AGE'].min())
```

```
Maximum Age of Cutomers =  68.997
Minimum Age of customers =  21.041
```

In [404…

```
plt.title("To find the outliers from AGE column")
sns.boxplot(x=df['AGE'])
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```

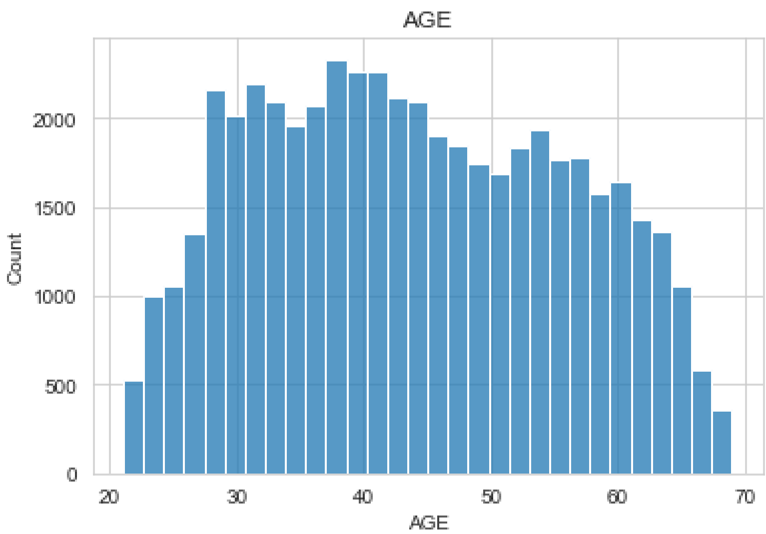



```
<Figure size 864x1440 with 0 Axes>
```

- As per above result we didnt found any issues with the AGE column

In [405…

```
plt.title("AGE")
sns.histplot(x=df['AGE'],bins=30)
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

Insights:

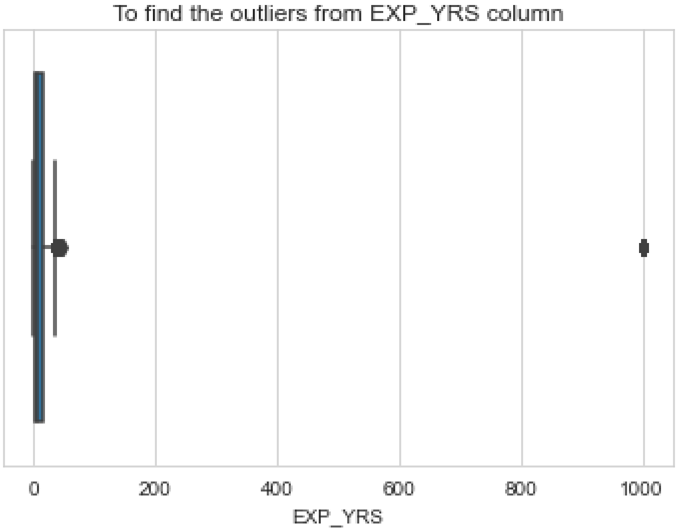
- As per above chart we can conclude that most of the customers have age between 30 to 40.
- We can observe that AGE column somewhat follows a normal distribution and most applicants are between age 27 and 65 i.e. most were in the working age group.

1. EXP_YRS

- As per below result we found 1 outlier in column EXP_YRS with entry 1000.666. To remove this outlier we have replaced this one with the median value of column EXP_YRS

In [406…

```
plt.title("To find the outliers from EXP_YRS column")
sns.boxplot(x=df['EXP_YRS'])
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

In [407…

```
l= df['EXP_YRS'].median()
l
```

Out[407…

```
6.071
```

In [408…

```
df['EXP_YRS'] = df['EXP_YRS'].replace(1000.666,6.071)
```

In [409…

```
df['EXP_YRS'].max()
```

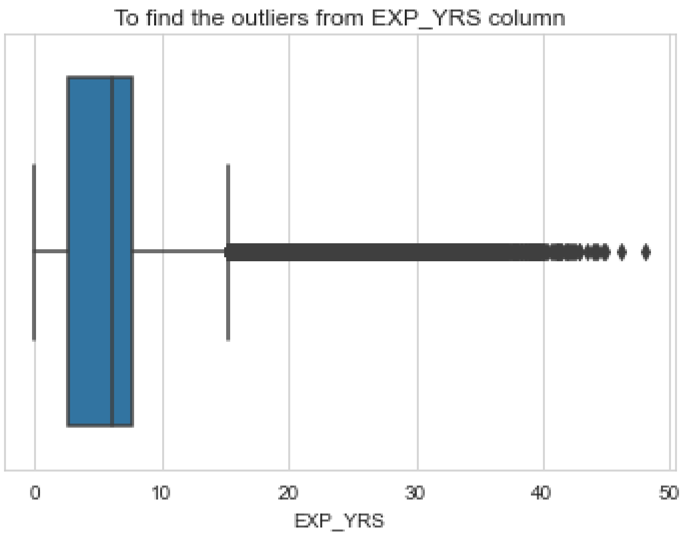
Out[409…

```
48.03
```

- After hadnling outliers of the column Expereince years again we cross verified by generating boxplot for it.

In [410…

```
plt.title("To find the outliers from EXP_YRS column")
sns.boxplot(x=df['EXP_YRS'])
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```

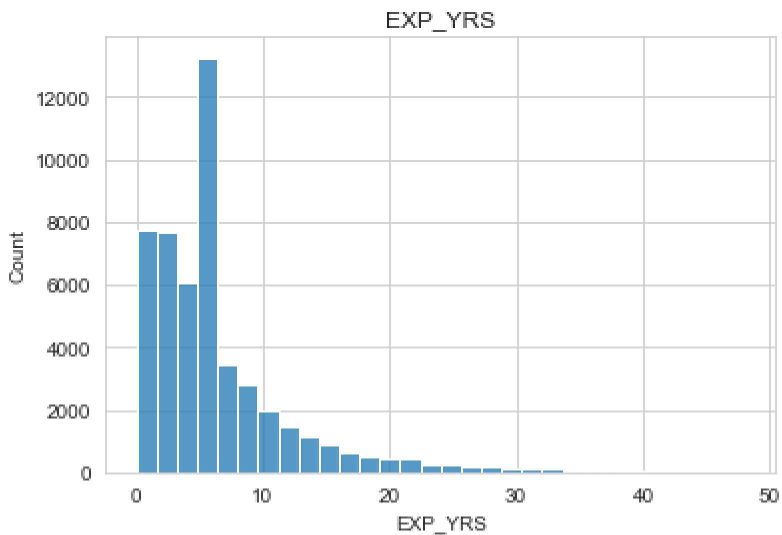



```
<Figure size 864x1440 with 0 Axes>
```

- As we can we had removed outliers from the column EXP_YRS and replaced it with the median value.

In [411...

```
plt.title("EXP_YRS")
sns.histplot(x=df['EXP_YRS'],bins=30)
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

Insights:

- As we can see, most of the customers have experience of 8 years and field ranges is between 0 to 50

In [ ]:

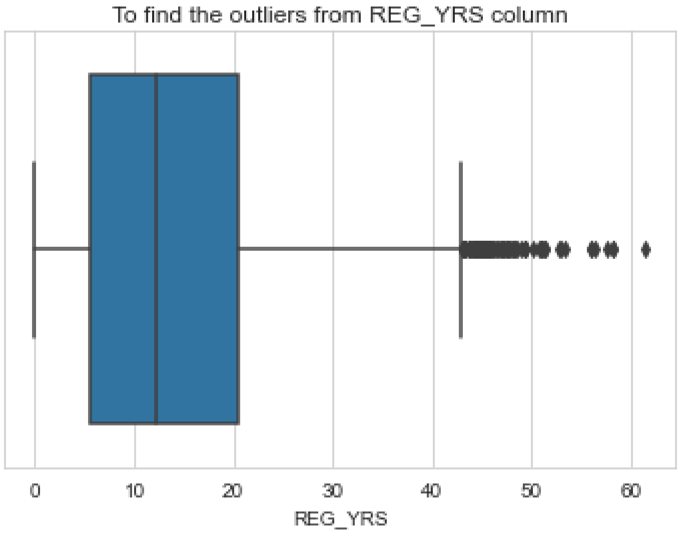
```

```

1. REG_YRS

In [412...

```
plt.title("To find the outliers from REG_YRS column")
sns.boxplot(x=df['REG_YRS'])
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```



<Figure size 864x1440 with 0 Axes>

- There are no outliers present in this REG_EXP column. RAnge is between o 65, which is normal range.

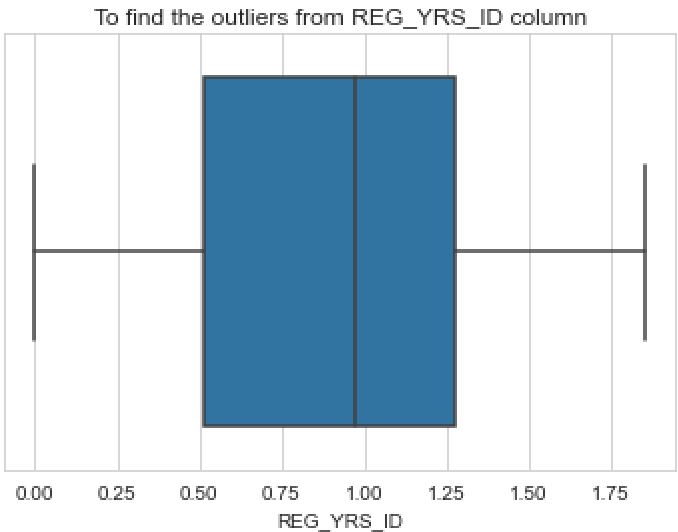
In [ ]:

1. REG_YRS_ID

In [ ]:

In [413…

```
plt.title("To find the outliers from REG_YRS_ID column")
sns.boxplot(x=df['REG_YRS_ID'])
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```



<Figure size 864x1440 with 0 Axes>

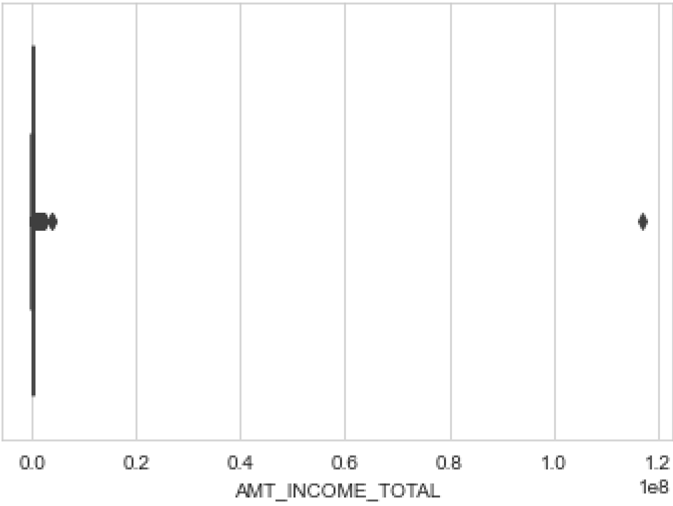
- As we can see there is no outliers present in this REG_YRS_ID column

In [ ]:

1. AMT_INCOME_TOTAL

In [414...

```
plt.title("To find the outliers from AMT_INCOME_TOTAL column")
sns.boxplot(x=df['AMT_INCOME_TOTAL'])
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

- As we can see there is outliers present in this AMT_INCOME_TOTAL field.

In [415...

```
df_1 = df.copy() # to perform some check created copy of original cleaned dataset
```

In [416...

```
df_1[df_1['AMT_INCOME_TOTAL']>1000000][['AMT_INCOME_TOTAL','AMT_CREDIT','AMT_GOODS_P
```

Out[416...

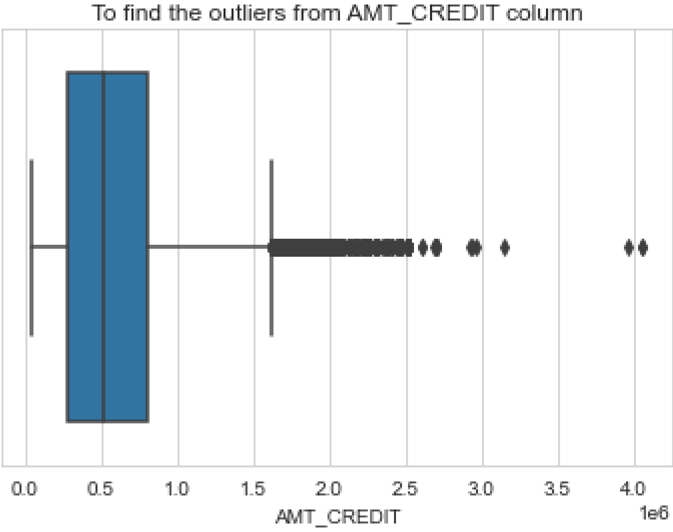
	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_GOODS_PRICE	AMT_ANNUITY
1504	1080000.0	180000.0	180000.0	9000.0
1723	1935000.0	269550.0	225000.0	10534.5
3371	1350000.0	2410380.0	2250000.0	109053.0
4603	1350000.0	405000.0	405000.0	20250.0
7061	1035000.0	2695500.0	2250000.0	74254.5

- We again checked for outliers in AMT_INCOME_TOTAL using Box plot and considered values greater than 1000000 as outliers and checked for any possible issues for rows with AMT_INCOME_TOTAL greater than 1000000. Found no issues.

1. AMT_CREDIT

In [417...

```
plt.title("To find the outliers from AMT_CREDIT column")
sns.boxplot(x=df['AMT_CREDIT'])
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

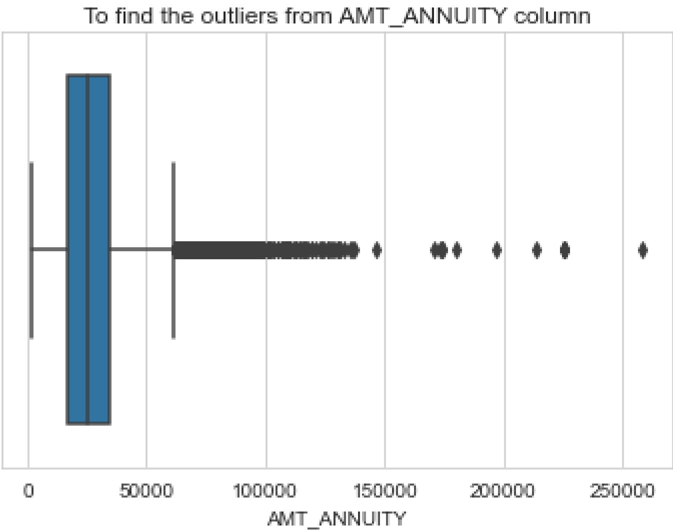
In [ ]:

- For Outliers in AMT_CREDIT column we checked for any possible issues.Found no issues.

1. AMT_ANNUITY

In [418...

```python
plt.title("To find the outliers from AMT_ANNUITY column")
sns.boxplot(x=df['AMT_ANNUITY'])
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

- For Outliers in AMT_ANNUITY column we checked for any possible issues.Found no issues.

In [ ]:

*** Checked if values of YRS_EXP, YRS_REG_CNG, YRS_ID_CNG are more than AGE.

In [419...

```python
df_1[df_1['AGE']< df_1['EXP_YRS']]
```

Out[419...

TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILD

0 rows × 56 columns

In [420…

```
df_1[df_1['AGE']< df_1['REG_YRS']]
```

Out[420…

TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILD

0 rows × 56 columns

In [421…

```
df_1[df_1['AGE']< df_1['REG_YRS_ID']]
```

Out[421…

TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILD

0 rows × 56 columns

In [ ]:

In [ ]:

C.

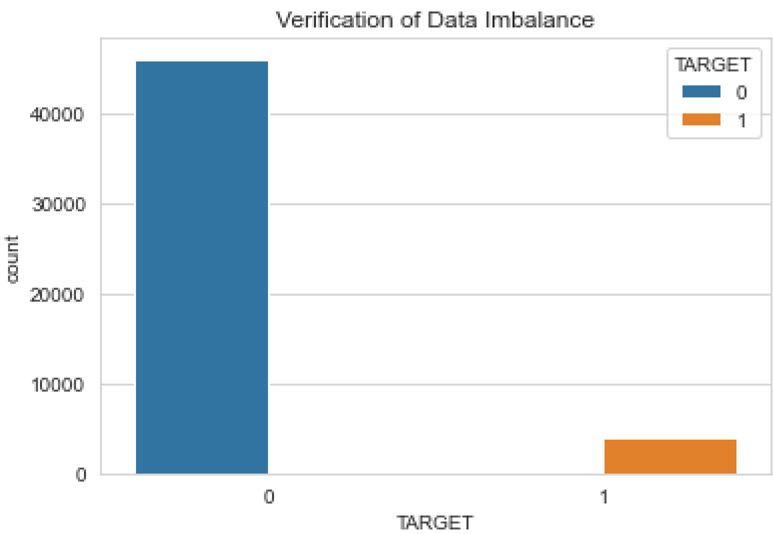
Analyze Data Imbalance: Data imbalance can affect the accuracy of the analysis, especially for binary classification problems. Understanding the data distribution is crucial for building reliable models.

Task: Determine if there is data imbalance in the loan application dataset and calculate the ratio of data imbalance using Excel functions.

In [ ]:

In [422…

```
plt.title("Verification of Data Imbalance")
sns.countplot(x=df['TARGET'],hue=df['TARGET'],dodge=True,)
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

- The Dataset is highly imbalanced, skewed more towards Class Label 0 (No Payment Difficulty). From above Bar Chart, we can see that around 45000 of Applicants didn't had any difficulty in paying loan installments and around 5000 (Class Label 1) of Applicants had difficulty in paying loan installments.

In [ ]:

D.

Perform Univariate, Segmented Univariate, and Bivariate Analysis: To gain insights into the driving factors of loan default, it is important to conduct various analyses on consumer and loan attributes.

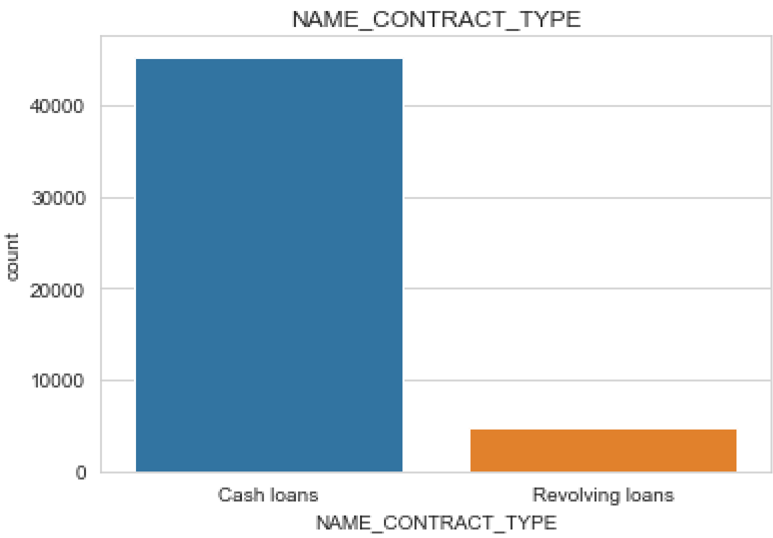
Task: Perform univariate analysis to understand the distribution of individual variables, segmented univariate analysis to compare variable distributions for different scenarios, and bivariate analysis to explore relationships between variables and the target variable using Excel functions and features.

Univariate Analysis:

1. NAME_CONTRACT_TYPE

In [423…

```
plt.title("NAME_CONTRACT_TYPE")
sns.countplot(x=df['NAME_CONTRACT_TYPE'])
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

Insights:

- Most of the loan applications are for Cash Loans and very less are for Revolving Loans. This is True in reality as well.

1. CODE_GENDER

In [424…

```
plt.title("CODE_GENDER")
sns.countplot(x=df['CODE_GENDER'])
```

```
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
#### Insights:plt.show()
```

Out[424…

```
<Figure size 864x1440 with 0 Axes>
```


```
<Figure size 864x1440 with 0 Axes>
```

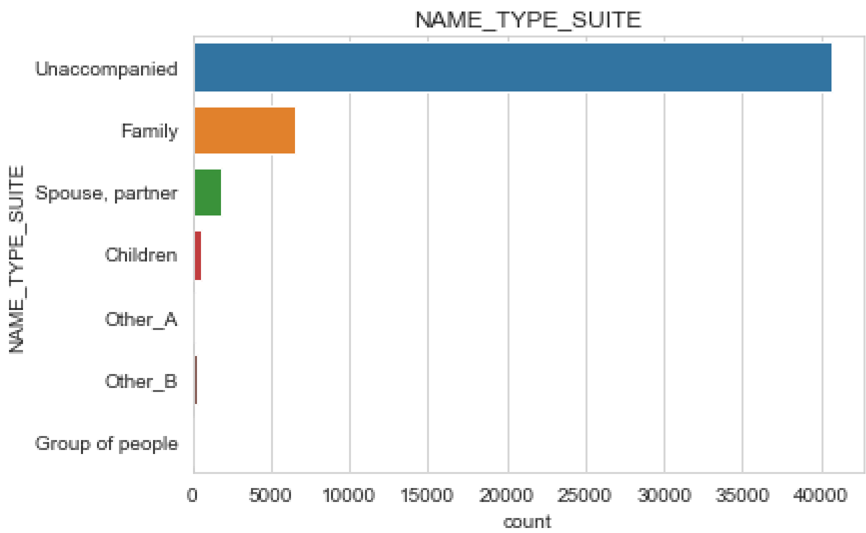
Insights:

- Most of the loan applicants are Females and less are Males.

1. NAME_TYPE_SUITE

In [425…

```
plt.title("NAME_TYPE_SUITE")
sns.countplot(y=df['NAME_TYPE_SUITE'],orient='h')
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

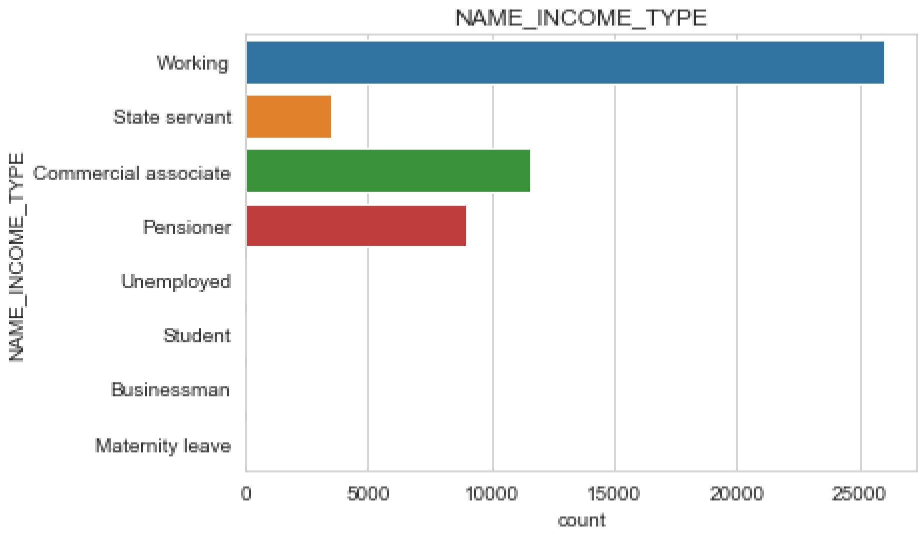
Insights:

- Most of the applicants were not accompanied by anyone else followed by applicants who were accompanies by family members.

1. NAME_INCOME_TYPE

In [426…

```
plt.title("NAME_INCOME_TYPE")
sns.countplot(y=df['NAME_INCOME_TYPE'],orient='h')
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

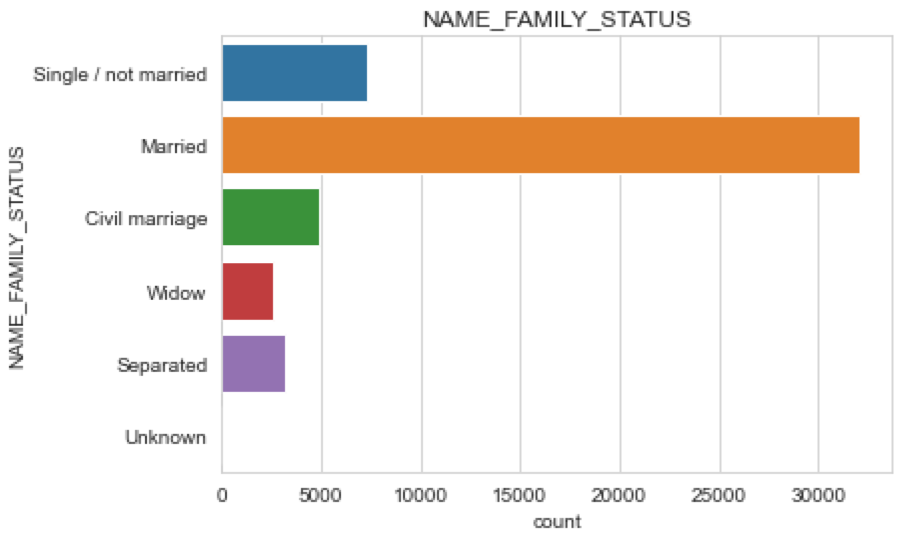
Insights:

- Most of the applicants had Working Income Type followed by Commercial Associate Income Type.

1. NAME_FAMILY_STATUS

In [427…

```
plt.title("NAME_FAMILY_STATUS")
sns.countplot(y=df['NAME_FAMILY_STATUS'],orient='h')
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

Insights:

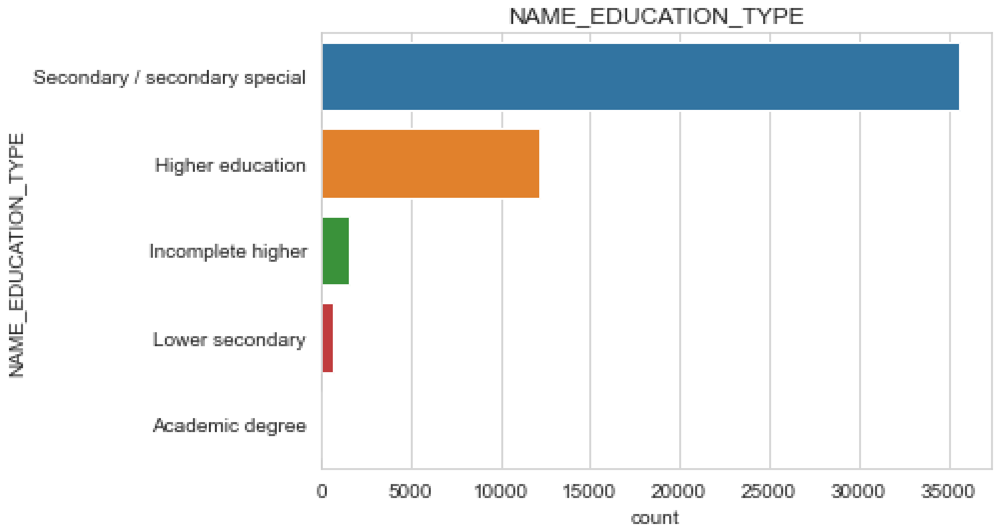
- Most of the applicants were normally Married followed by Single or Not Married.

In [ ]:

1. NAME_EDUCATION_TYPE

In [428…

```
plt.title("NAME_EDUCATION_TYPE")
sns.countplot(y=df['NAME_EDUCATION_TYPE'],orient='h')
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

Insights:

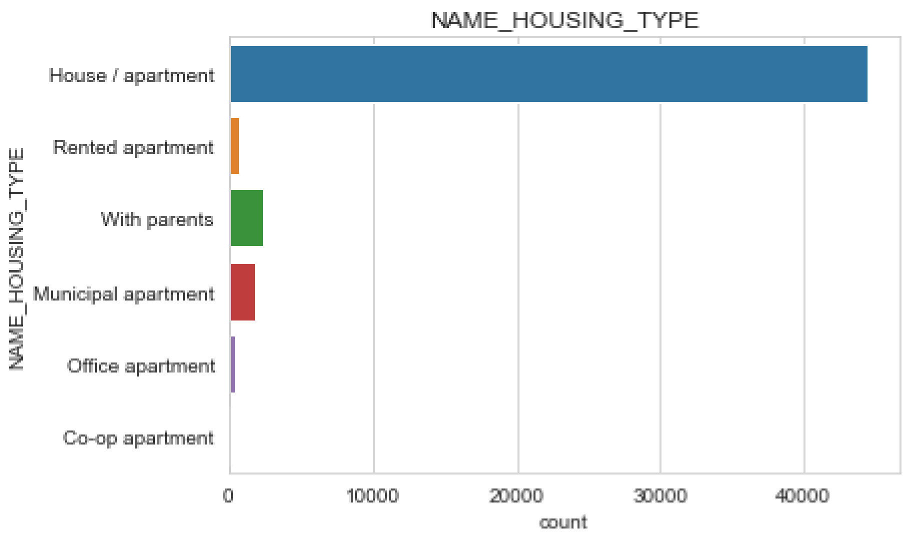
- Most of the applicants had education upto Secondary followed by Higher education.

In [ ]:

1. NAME_HOUSING_TYPE

In [429…

```
plt.title("NAME_HOUSING_TYPE")
sns.countplot(y=df['NAME_HOUSING_TYPE'],orient='h')
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

Insights:

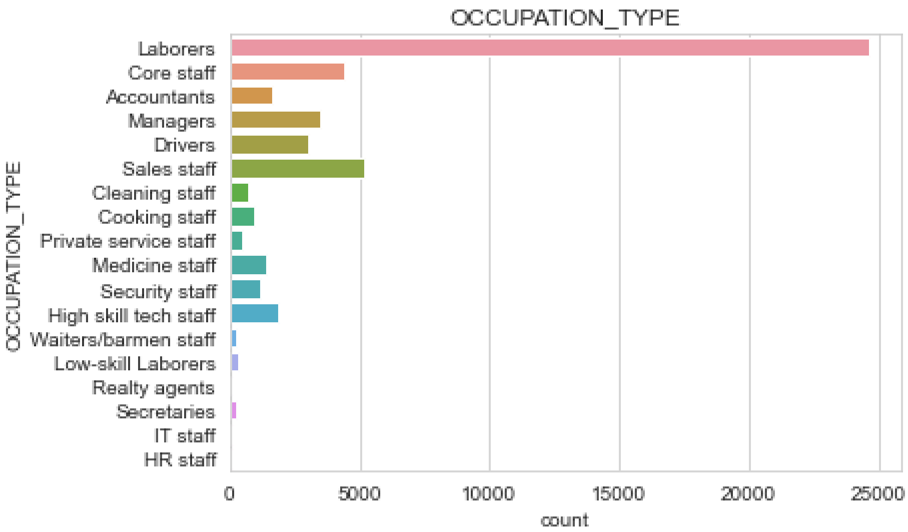
- Most of the applicants had own House followed by applicant who were living with parents.

In [ ]:

1. OCCUPATION_TYPE

In [430…

```
plt.title("OCCUPATION_TYPE")
sns.countplot(y=df['OCCUPATION_TYPE'],orient='h')
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
plt.show()
```




```
<Figure size 864x1440 with 0 Axes>
```

Insights:

- Most of the applicants were Laborers followed by Core staff and accountants

1. FLAG_OWN_CAR and FLAG_OWN_REALTY

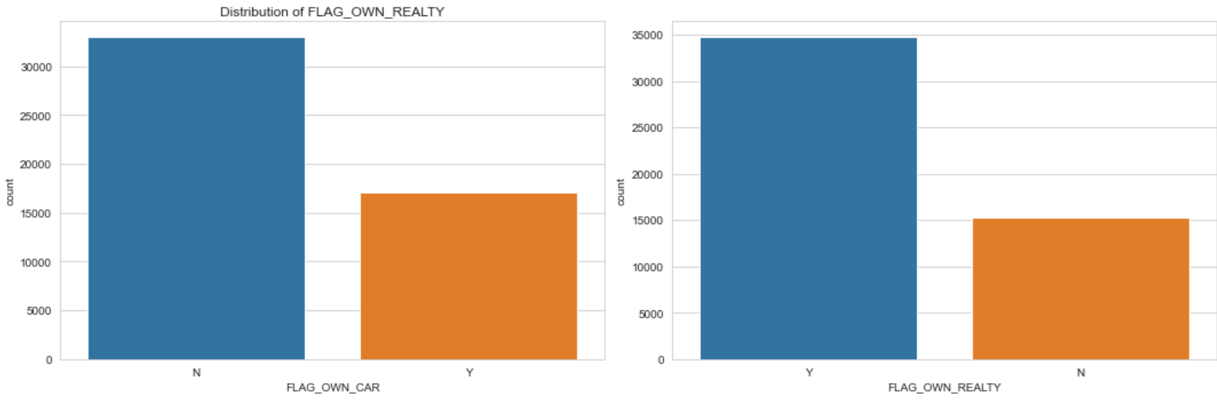
In [431…

```
fig, axs = plt.subplots(1, 2,figsize=(15, 5))

sns.countplot(x=df['FLAG_OWN_CAR'], ax=axs[0])
axs[0].set_title("Distribution of FLAG_OWN_CAR")

sns.countplot(x=df['FLAG_OWN_REALTY'], ax=axs[1])
axs[0].set_title("Distribution of FLAG_OWN_REALTY")

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



Insights:

- Most of the applicants didn't own car but had own property.

In [ ]:

1. FLAG_MOBIL and FLAG_EMP_PHONE

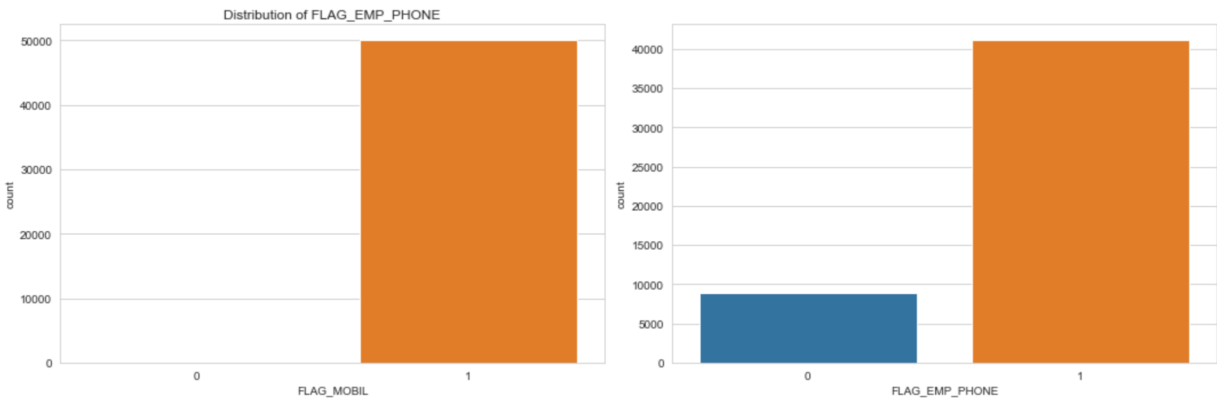
In [432…

```
fig, axs = plt.subplots(1, 2,figsize=(15, 5))

sns.countplot(x=df['FLAG_MOBIL'], ax=axs[0])
axs[0].set_title("Distribution of FLAG_MOBIL")

sns.countplot(x=df['FLAG_EMP_PHONE'], ax=axs[1])
axs[0].set_title("Distribution of FLAG_EMP_PHONE")


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



In [ ]:

Insights:

- Almost everyone had a Mobile Phone and most applicant had a Phone from their employer.

In [ ]:

1. FLAG_WORK_PHONE and FLAG_CONT_MOBILE

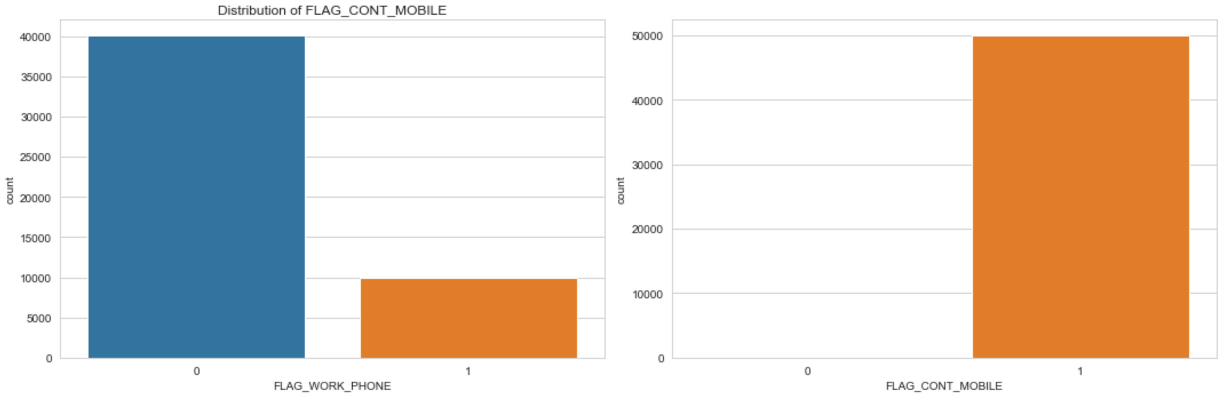
In [433…

```
fig, axs = plt.subplots(1, 2,figsize=(15, 5))

sns.countplot(x=df['FLAG_WORK_PHONE'], ax=axs[0])
axs[0].set_title("Distribution of FLAG_WORK_PHONE")
```

```
sns.countplot(x=df['FLAG_CONT_MOBILE'], ax=axs[1])
axs[0].set_title("Distribution of FLAG_CONT_MOBILE")


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



Insights:

- Most of the applicants didn't had/provided home phone but most of the applicants phone provided were reachable.

In [ ]:

1. FLAG_PHONE and FLAG_EMAIL

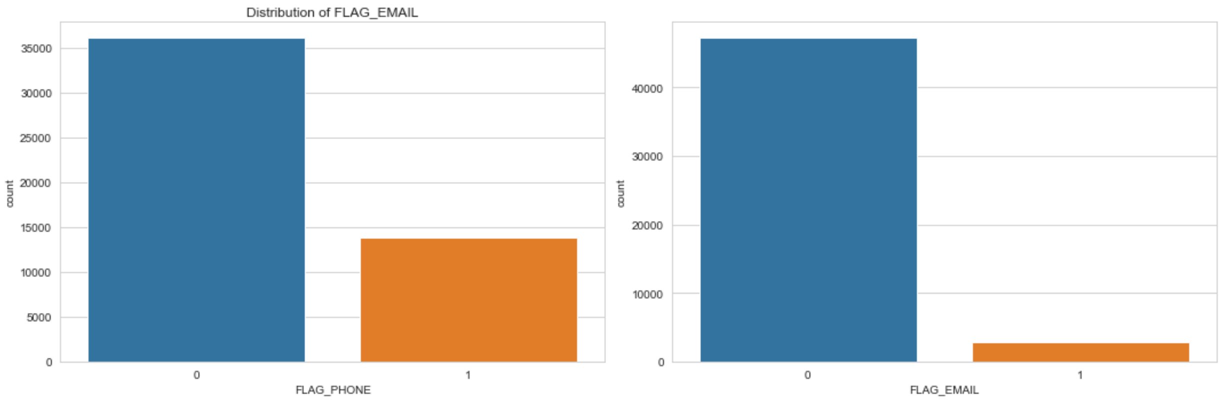
In [434...

```
fig, axs = plt.subplots(1, 2,figsize=(15, 5))

ax=sns.countplot(x=df['FLAG_PHONE'], ax=axs[0])
axs[0].set_title("Distribution of FLAG_PHONE")

ax=sns.countplot(x=df['FLAG_EMAIL'], ax=axs[1])
axs[0].set_title("Distribution of FLAG_EMAIL")


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



Insights:

- Also most applicants didn't had/provided their Email IDs.

In [ ]:

1. AMT_ANNUITY and AMT_CREDIT

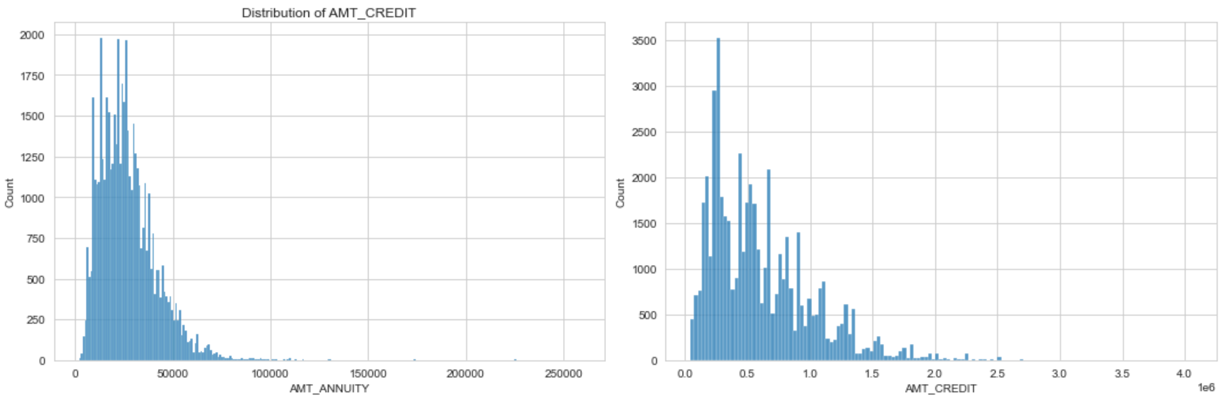
In [435…

```
fig, axs = plt.subplots(1, 2,figsize=(15, 5))

sns.histplot(x=df['AMT_ANNUITY'], ax=axs[0])
axs[0].set_title("Distribution of AMT_ANNUITY")

sns.histplot(x=df['AMT_CREDIT'], ax=axs[1])
axs[0].set_title("Distribution of AMT_CREDIT")


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



Insights:

- Most of AMT_AMMUNITY lies between 0 to 55000 whereas AMT_CREDIT lies between 0.0 to 2.0

Bivariate Analysis:

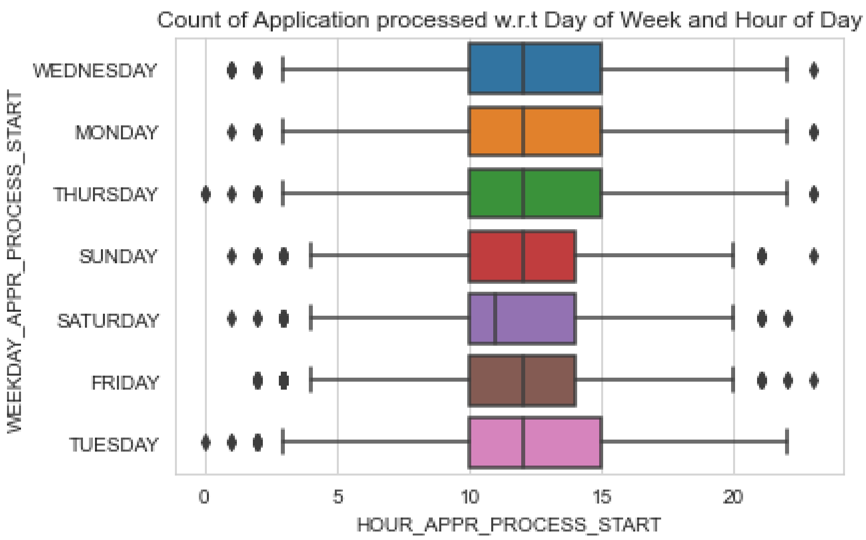
1. HOUR_APPR_PROCESS_START vs WEEKDAY_APPR_PROCESS_START

In [436…

```
plt.title("Count of Application processed w.r.t Day of Week and Hour of Day")
sns.boxplot(x=df['HOUR_APPR_PROCESS_START'],y=df['WEEKDAY_APPR_PROCESS_START'],orien
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
```

Out[436…

```
<Figure size 864x1440 with 0 Axes>
```

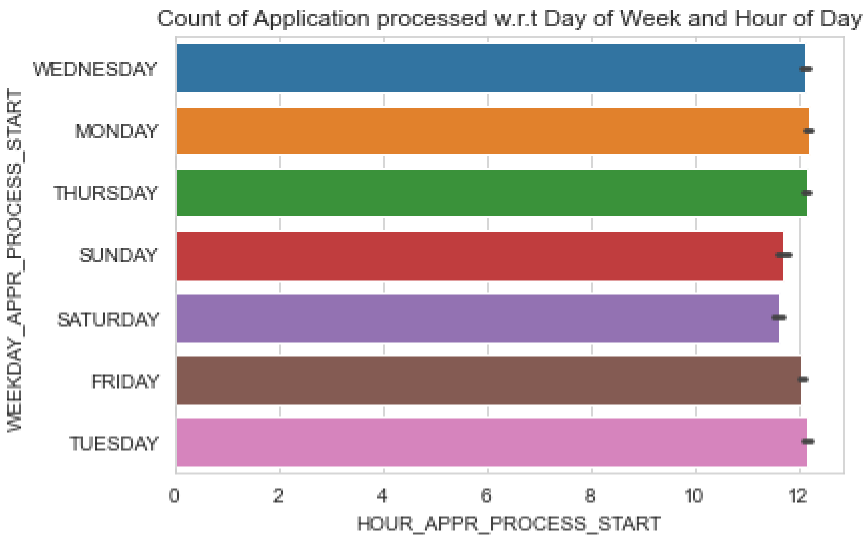



```
<Figure size 864x1440 with 0 Axes>
```

In [ ]:

In [437…
```
plt.title("Count of Application processed w.r.t Day of Week and Hour of Day")
sns.barplot(x=df['HOUR_APPR_PROCESS_START'],y=df['WEEKDAY_APPR_PROCESS_START'],orien
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
```

Out[437…
```
<Figure size 864x1440 with 0 Axes>
```

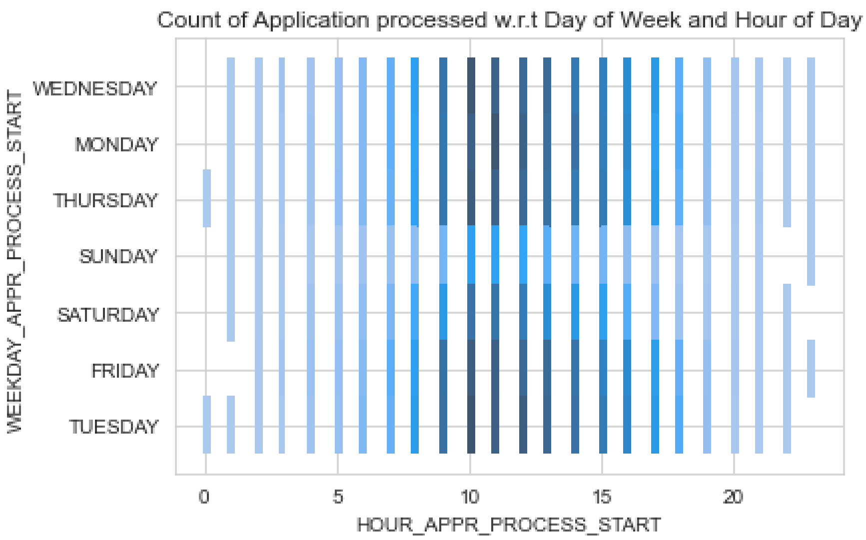



```
<Figure size 864x1440 with 0 Axes>
```

In [ ]:

In [438…
```
plt.title("Count of Application processed w.r.t Day of Week and Hour of Day")
sns.histplot(x=df['HOUR_APPR_PROCESS_START'],y=df['WEEKDAY_APPR_PROCESS_START'],)
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
```

Out[438…
```
<Figure size 864x1440 with 0 Axes>
```




```
<Figure size 864x1440 with 0 Axes>
```

Insights:

- From the above 3 charts we can see that, We can observe that most clients applied for Loans during Weekdays and between 9 A.M and 4 P.M. But there are also very few clients who applied for Loans late at night as well.

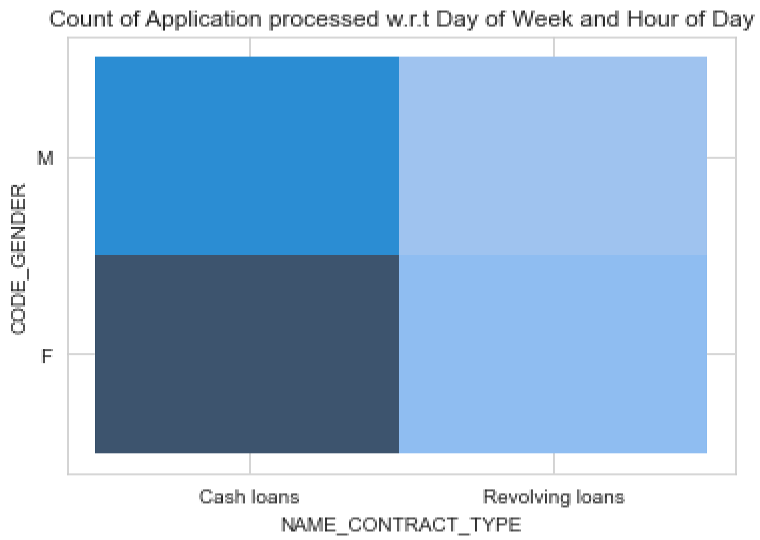
1. NAME_CONTRACT_TYPE vs CODE_GENDER

In [439…

```
plt.title("Count of Application processed w.r.t Day of Week and Hour of Day")
sns.histplot(x=df['NAME_CONTRACT_TYPE'],y=df['CODE_GENDER'],)
sns.set_style('whitegrid')
plt.figure(figsize=(12,20))
```

Out[439…

```
<Figure size 864x1440 with 0 Axes>
```




```
<Figure size 864x1440 with 0 Axes>
```

In [ ]:

In [ ]:

Multivariate Analysis

E.

Identify Top Correlations for Different Scenarios: Understanding the correlation between variables and the target variable can provide insights into strong indicators of loan default.

Task: Segment the dataset based on different scenarios (e.g., clients with payment difficulties and all other cases) and identify the top correlations for each segmented data using Excel functions.

- To get this we have calculated the correlation between Target column 'TARGET' and other dependant column.

In [440…

```
df2 = df[['TARGET','CNT_CHILDREN','AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','AMT

corr1 = df2.corr()
corr1
```

Out[440…

	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_
TARGET	1.000000	0.026364	0.010894	-0.032428	
CNT_CHILDREN	0.026364	1.000000	0.009589	0.004972	
AMT_INCOME_TOTAL	0.010894	0.009589	1.000000	0.069316	
AMT_CREDIT	-0.032428	0.004972	0.069316	1.000000	

	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_
AMT_ANNUITY	-0.012399	0.026179	0.083009	0.769498	
AMT_GOODS_PRICE	-0.041296	0.000253	0.069885	0.986607	
REGION_POPULATION_RELATIVE	-0.040799	-0.025556	0.029841	0.095111	
FLAG_MOBIL	0.001323	0.002593	0.000376	0.003568	
FLAG_EMP_PHONE	0.041408	0.240678	0.031568	0.069063	
FLAG_WORK_PHONE	0.021302	0.055881	-0.009181	-0.015115	
FLAG_WORK_PHONE	0.021302	0.055881	-0.009181	-0.015115	
FLAG_CONT_MOBILE	0.006766	-0.002827	-0.003246	0.024481	
FLAG_PHONE	-0.032679	-0.030654	-0.002044	0.019460	
FLAG_EMAIL	-0.001312	0.026816	0.015215	0.010812	
HOUR_APPR_PROCESS_START	-0.032036	-0.006254	0.018464	0.056677	
AGE	-0.076788	-0.329264	-0.016003	0.059343	
EXP_YRS	-0.070001	-0.052654	0.004480	0.091794	
REG_YRS	-0.042343	-0.181217	-0.009952	-0.003449	
REG_YRS_ID	-0.046933	0.032112	-0.003502	0.012230	
CREDIT%INCOME	-0.006197	-0.013865	-0.053295	0.642609	
AMUNITY%INCOME	0.018013	0.000362	-0.075282	0.369951	
AMUNITY%CREDIT	0.015015	0.020507	-0.014306	-0.558646	
CREDIT%GOODS_PRICE	0.069634	0.025555	-0.006143	0.016090	

23 rows × 23 columns

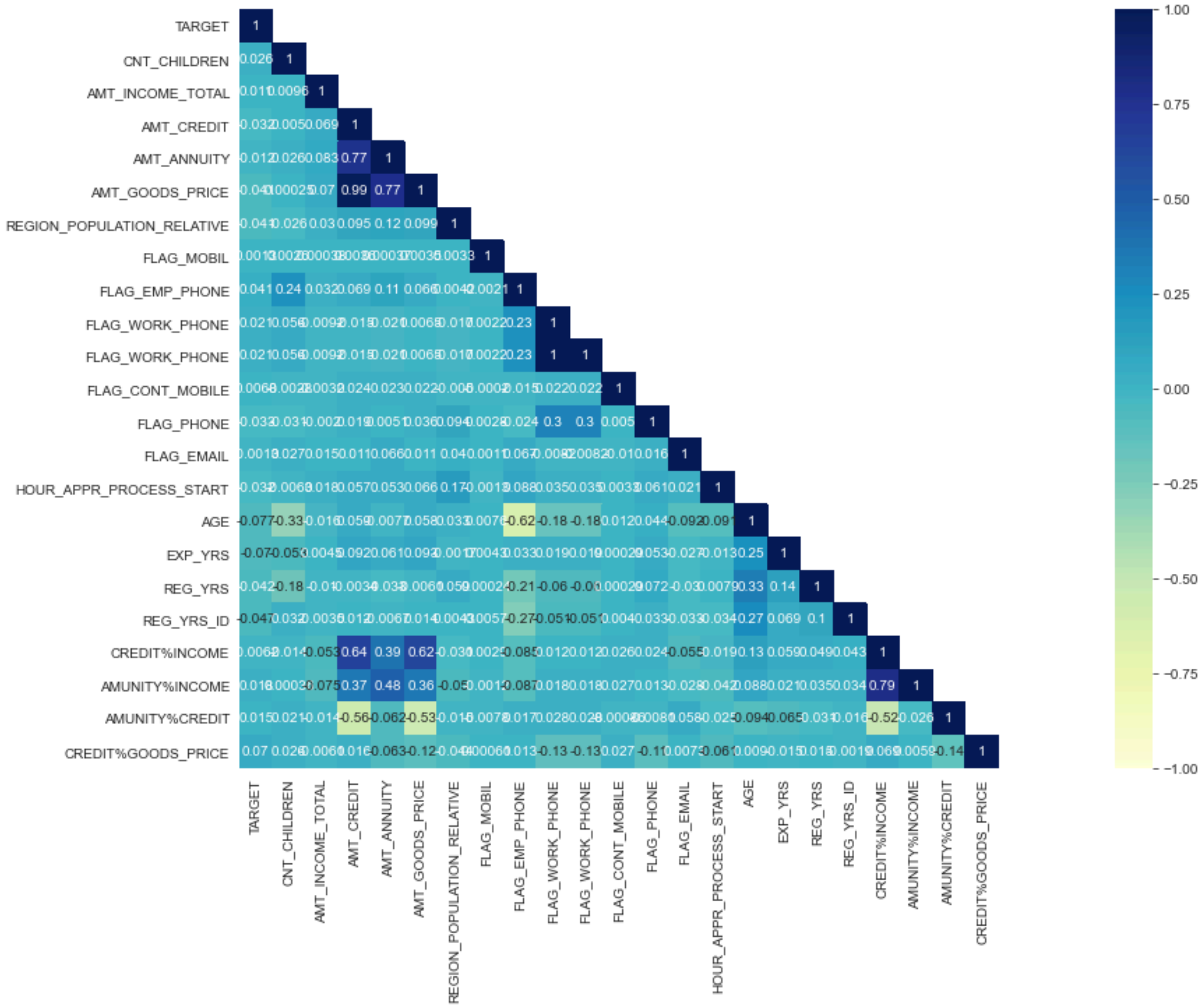
In [ ]:

In [441…

```
mask = np.array(corr1)
mask[np.tril_indices_from(mask)] = False
fig,ax= plt.subplots()
fig.set_size_inches(30,10)
sns.heatmap(corr1, vmin=-1, vmax=1, mask=mask, square=True,annot=True, cmap="YlGnBu"
```

Out[441…

<AxesSubplot:>



- AMT_CREDIT and AMT_GOODS_PRICE are highly and positively correlated as the Credit amount request is for the Goods whose price is in AMT_GOODS_PRICE column.
- CNT_FAM_MEMBERS and CNT_CHILDREN are highly and positively correlated as we observed before that all applicants were either Single Parent or had Nuclear Family.
- CREDIT%INCOME and ANNUITY%INCOME, AMT_ANNUITY and AMT_GOODS_PRICE, AMT_CREDIT and AMT_ANNUITY are highly and positively correlated and have almost same Correlation values.
- AMT_CREDIT and CREDIT%INCOME, AMT_GOODS_PRICE and CREDIT%INCOME are highly and positively correlated as CREDIT%INCOME is a derived feature which is directly proportional to AMT_CREDIT and AMT_CREDIT and AMT_GOODS_PRICE are highly and positively correlated as in point 1.
- FLAG_EMP_PHONE and AGE are highly and negatively correlated possibly because older generation people are less used to phones.
- AMT_CREDIT and ANNUIT%CREDIT, AMT_GOODS_PRICE and ANNUITY%CREDIT are highly and negatively correlated as ANNUITY%CREDIT is a derived feature which is inversely proportional to AMT_CREDIT and AMT_CREDIT and AMT_GOODS_PRICE are highly and positively correlated as in point 1.

In [ ]:

Thank you

In [ ]:

```

```