Case Study Title:

Credit Risk Prediction Model for Loan Applicants

Objective:

Develop an advanced predictive model to accurately assess the credit risk of loan applicants, using historical loan application and repayment data. The goal is to enable the finance company to minimize risks associated with bad loans while maximizing the approval of safe loans.

Key Tasks:

Data Analysis: Historical Data Exploration: Analyze historical loan application and repayment data to understand past trends and borrower behaviors. Feature Identification: Identify key factors contributing to loan default risks.

Loan Type Analysis: Performance Assessment: Perform an in-depth analysis of different loan types and their performance over time. Risk Correlation: Correlate loan types with default rates to understand risk profiles associated with each loan type.

Customer Segmentation: Loan Preferences: Segment customers based on their loan preferences, considering factors like loan amount, term, type, and interest rates. Repayment History: Further segment these groups based on their repayment history, such as timely payments, late payments, or defaults.

Predictive Modeling: Model Development: Build a model to predict credit risk using statistical or machine learning techniques. Model Validation and Optimization: Validate and optimize the model for accuracy and reliability.

Risk Minimization Strategy: Risk Assessment: Evaluate and quantify the risk associated with each loan applicant using the model. Approval Strategy Development: Develop strategies to balance risk minimization with approving a high number of safe loans.

Data Description:

1. 'Application_data.csv': contains all the information of the client at the time of application. The data is about whether a client has payment difficulties.
2. 'Previous_application.csv': contains information about the client’s previous loan data(Historical data). It contains the data whether the previous application had been Approved, Canceled, Refused or Unused offer.

Outcome:

The project aims to deliver a comprehensive tool for credit risk assessment, enhancing the finance company's ability to make informed lending decisions. This tool will not only

contribute to reducing financial losses due to bad loans but also support business growth through the identification and approval of creditworthy applicants.

In [326…
```
# importing libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_rows', None) #To expand the output cell.
%matplotlib inline

import statsmodels
import statsmodels.api as sm
from statsmodels.formula.api import ols
import statsmodels.stats.multicomp
# import statsmodels for statistical modelling
from statsmodels.stats.outliers_influence import variance_inflation_factor



import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.tree import DecisionTreeClassifier
```

In [327…
```
#reading data

appdf = pd.read_csv('application_data.csv')
prevdf = pd.read_csv('previous_application.csv')wq
```

In [328…
```
#checking

appdf.head()
```

Out[328]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_RE
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 × 122 columns

In [329…
```
prevdf.head()
```

Out[329]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CR
0	2030495	271877	Consumer loans	1730.430	17145.0	17
1	2802425	108129	Cash loans	25188.615	607500.0	679
2	2523466	122040	Cash loans	15060.735	112500.0	136
3	2819243	176158	Cash loans	47041.335	450000.0	470
4	1784265	202054	Cash loans	31924.395	337500.0	404

5 rows × 37 columns

In [330…

```
#checking our dataframes

print('no. of rows and columns in appdf: ',appdf.shape)
print('no. of rows and columns in prevdf: ',prevdf.shape)
```

```
no. of rows and columns in appdf:  (307511, 122)
no. of rows and columns in prevdf:  (1670214, 37)
```

In [331…

```
appdf.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 122 columns):
 #    Column                          Dtype
---   ------                          -----
 0    SK_ID_CURR                      int64
 1    TARGET                          int64
 2    NAME_CONTRACT_TYPE              object
 3    CODE_GENDER                     object
 4    FLAG_OWN_CAR                    object
 5    FLAG_OWN_REALTY                 object
 6    CNT_CHILDREN                    int64
 7    AMT_INCOME_TOTAL                float64
 8    AMT_CREDIT                      float64
 9    AMT_ANNUITY                     float64
 10   AMT_GOODS_PRICE                 float64
 11   NAME_TYPE_SUITE                 object
 12   NAME_INCOME_TYPE                object
 13   NAME_EDUCATION_TYPE             object
 14   NAME_FAMILY_STATUS              object
 15   NAME_HOUSING_TYPE               object
 16   REGION_POPULATION_RELATIVE      float64
 17   DAYS_BIRTH                      int64
 18   DAYS_EMPLOYED                   int64
 19   DAYS_REGISTRATION               float64
 20   DAYS_ID_PUBLISH                 int64
 21   OWN_CAR_AGE                     float64
 22   FLAG_MOBIL                      int64
 23   FLAG_EMP_PHONE                  int64
 24   FLAG_WORK_PHONE                 int64
 25   FLAG_CONT_MOBILE                int64
 26   FLAG_PHONE                      int64
 27   FLAG_EMAIL                      int64
 28   OCCUPATION_TYPE                 object
 29   CNT_FAM_MEMBERS                 float64
 30   REGION_RATING_CLIENT            int64
 31   REGION_RATING_CLIENT_W_CITY     int64
 32   WEEKDAY_APPR_PROCESS_START      object
 33   HOUR_APPR_PROCESS_START         int64
 34   REG_REGION_NOT_LIVE_REGION      int64
 35   REG_REGION_NOT_WORK_REGION      int64
 36   LIVE_REGION_NOT_WORK_REGION     int64
 37   REG_CITY_NOT_LIVE_CITY          int64
 38   REG_CITY_NOT_WORK_CITY          int64
 39   LIVE_CITY_NOT_WORK_CITY         int64
 40   ORGANIZATION_TYPE               object
 41   EXT_SOURCE_1                    float64
 42   EXT_SOURCE_2                    float64
 43   EXT_SOURCE_3                    float64
 44   APARTMENTS_AVG                  float64
 45   BASEMENTAREA_AVG                float64
 46   YEARS_BEGINEXPLUATATION_AVG     float64
 47   YEARS_BUILD_AVG                 float64
 48   COMMONAREA_AVG                  float64
 49   ELEVATORS_AVG                   float64
 50   ENTRANCES_AVG                   float64
 51   FLOORSMAX_AVG                   float64
 52   FLOORSMIN_AVG                   float64
 53   LANDAREA_AVG                    float64
 54   LIVINGAPARTMENTS_AVG            float64
 55   LIVINGAREA_AVG                  float64
 56   NONLIVINGAPARTMENTS_AVG         float64
 57   NONLIVINGAREA_AVG               float64
 58   APARTMENTS_MODE                 float64
```

59	BASEMENTAREA_MODE	float64
60	YEARS_BEGINEXPLUATATION_MODE	float64
61	YEARS_BUILD_MODE	float64
62	COMMONAREA_MODE	float64
63	ELEVATORS_MODE	float64
64	ENTRANCES_MODE	float64
65	FLOORSMAX_MODE	float64
66	FLOORSMIN_MODE	float64
67	LANDAREA_MODE	float64
68	LIVINGAPARTMENTS_MODE	float64
69	LIVINGAREA_MODE	float64
70	NONLIVINGAPARTMENTS_MODE	float64
71	NONLIVINGAREA_MODE	float64
72	APARTMENTS_MEDI	float64
73	BASEMENTAREA_MEDI	float64
74	YEARS_BEGINEXPLUATATION_MEDI	float64
75	YEARS_BUILD_MEDI	float64
76	COMMONAREA_MEDI	float64
77	ELEVATORS_MEDI	float64
78	ENTRANCES_MEDI	float64
79	FLOORSMAX_MEDI	float64
80	FLOORSMIN_MEDI	float64
81	LANDAREA_MEDI	float64
82	LIVINGAPARTMENTS_MEDI	float64
83	LIVINGAREA_MEDI	float64
84	NONLIVINGAPARTMENTS_MEDI	float64
85	NONLIVINGAREA_MEDI	float64
86	FONDKAPREMONT_MODE	object
87	HOUSETYPE_MODE	object
88	TOTALAREA_MODE	float64
89	WALLSMATERIAL_MODE	object
90	EMERGENCYSTATE_MODE	object
91	OBS_30_CNT_SOCIAL_CIRCLE	float64
92	DEF_30_CNT_SOCIAL_CIRCLE	float64
93	OBS_60_CNT_SOCIAL_CIRCLE	float64
94	DEF_60_CNT_SOCIAL_CIRCLE	float64
95	DAYS_LAST_PHONE_CHANGE	float64
96	FLAG_DOCUMENT_2	int64
97	FLAG_DOCUMENT_3	int64
98	FLAG_DOCUMENT_4	int64
99	FLAG_DOCUMENT_5	int64
100	FLAG_DOCUMENT_6	int64
101	FLAG_DOCUMENT_7	int64
102	FLAG_DOCUMENT_8	int64
103	FLAG_DOCUMENT_9	int64
104	FLAG_DOCUMENT_10	int64
105	FLAG_DOCUMENT_11	int64
106	FLAG_DOCUMENT_12	int64
107	FLAG_DOCUMENT_13	int64
108	FLAG_DOCUMENT_14	int64
109	FLAG_DOCUMENT_15	int64
110	FLAG_DOCUMENT_16	int64
111	FLAG_DOCUMENT_17	int64
112	FLAG_DOCUMENT_18	int64
113	FLAG_DOCUMENT_19	int64
114	FLAG_DOCUMENT_20	int64
115	FLAG_DOCUMENT_21	int64
116	AMT_REQ_CREDIT_BUREAU_HOUR	float64
117	AMT_REQ_CREDIT_BUREAU_DAY	float64
118	AMT_REQ_CREDIT_BUREAU_WEEK	float64
119	AMT_REQ_CREDIT_BUREAU_MON	float64
120	AMT_REQ_CREDIT_BUREAU_QRT	float64
121	AMT_REQ_CREDIT_BUREAU_YEAR	float64

```
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

In [332…

```
prevdf.info(verbose = True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   SK_ID_PREV                   1670214 non-null  int64  
 1   SK_ID_CURR                   1670214 non-null  int64  
 2   NAME_CONTRACT_TYPE           1670214 non-null  object 
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_DOWN_PAYMENT             774370 non-null   float64
 7   AMT_GOODS_PRICE              1284699 non-null  float64
 8   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object 
 9   HOUR_APPR_PROCESS_START      1670214 non-null  int64  
 10  FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object 
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64  
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object 
 16  NAME_CONTRACT_STATUS         1670214 non-null  object 
 17  DAYS_DECISION                1670214 non-null  int64  
 18  NAME_PAYMENT_TYPE            1670214 non-null  object 
 19  CODE_REJECT_REASON           1670214 non-null  object 
 20  NAME_TYPE_SUITE              849809 non-null   object 
 21  NAME_CLIENT_TYPE             1670214 non-null  object 
 22  NAME_GOODS_CATEGORY          1670214 non-null  object 
 23  NAME_PORTFOLIO               1670214 non-null  object 
 24  NAME_PRODUCT_TYPE            1670214 non-null  object 
 25  CHANNEL_TYPE                 1670214 non-null  object 
 26  SELLERPLACE_AREA             1670214 non-null  int64  
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object 
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object 
 30  PRODUCT_COMBINATION          1669868 non-null  object 
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
```

Data Cleaning and Manipulation

In [333…

```
# creating a function to check the percentage of missing values in dataframe columr

def check_null_values(dataset):
    final_output = round(dataset.isnull().sum()/len(dataset.index)*100,2)
    return final_output
```

In [334…

```
# calling the function to check null values of appdf

check_null_values(appdf)
```

Out[334]:

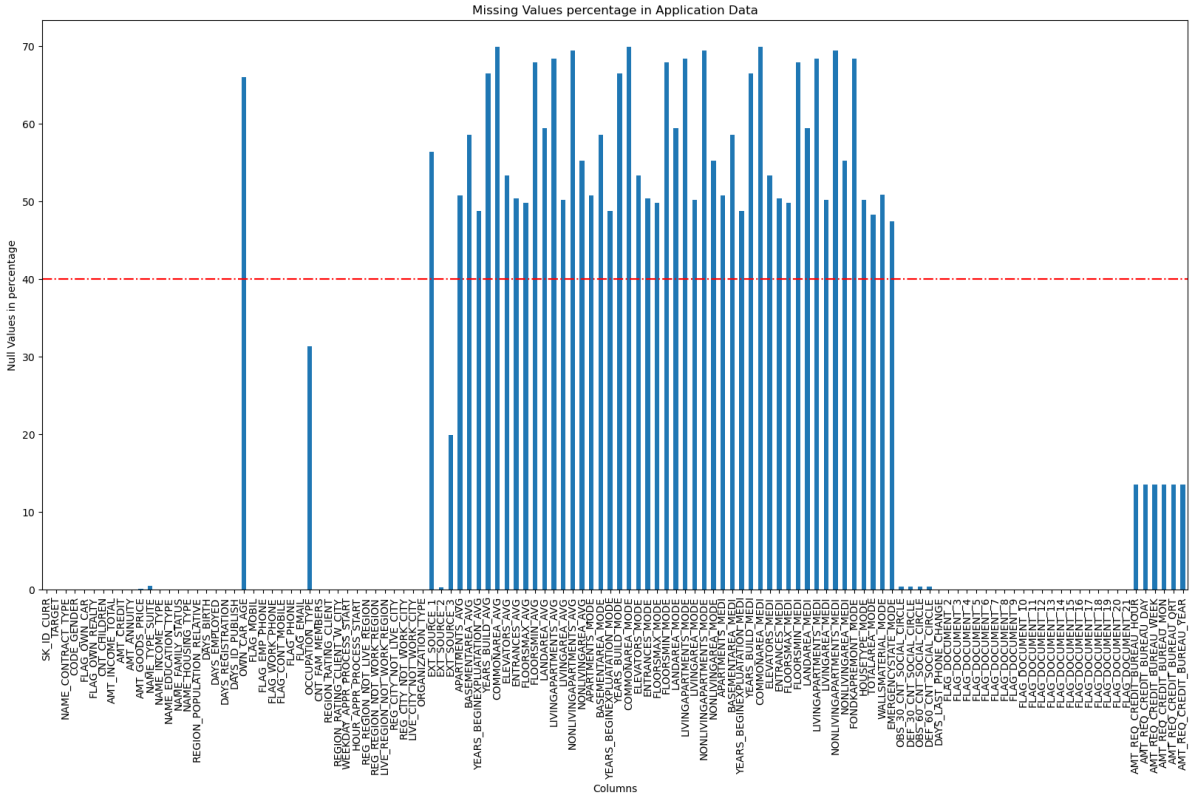
SK_ID_CURR	0.00
TARGET	0.00
NAME_CONTRACT_TYPE	0.00
CODE_GENDER	0.00
FLAG_OWN_CAR	0.00
FLAG_OWN_REALTY	0.00
CNT_CHILDREN	0.00
AMT_INCOME_TOTAL	0.00
AMT_CREDIT	0.00
AMT_ANNUITY	0.00
AMT_GOODS_PRICE	0.09
NAME_TYPE_SUITE	0.42
NAME_INCOME_TYPE	0.00
NAME_EDUCATION_TYPE	0.00
NAME_FAMILY_STATUS	0.00
NAME_HOUSING_TYPE	0.00
REGION_POPULATION_RELATIVE	0.00
DAYS_BIRTH	0.00
DAYS_EMPLOYED	0.00
DAYS_REGISTRATION	0.00
DAYS_ID_PUBLISH	0.00
OWN_CAR_AGE	65.99
FLAG_MOBIL	0.00
FLAG_EMP_PHONE	0.00
FLAG_WORK_PHONE	0.00
FLAG_CONT_MOBILE	0.00
FLAG_PHONE	0.00
FLAG_EMAIL	0.00
OCCUPATION_TYPE	31.35
CNT_FAM_MEMBERS	0.00
REGION_RATING_CLIENT	0.00
REGION_RATING_CLIENT_W_CITY	0.00
WEEKDAY_APPR_PROCESS_START	0.00
HOUR_APPR_PROCESS_START	0.00
REG_REGION_NOT_LIVE_REGION	0.00
REG_REGION_NOT_WORK_REGION	0.00
LIVE_REGION_NOT_WORK_REGION	0.00
REG_CITY_NOT_LIVE_CITY	0.00
REG_CITY_NOT_WORK_CITY	0.00
LIVE_CITY_NOT_WORK_CITY	0.00
ORGANIZATION_TYPE	0.00
EXT_SOURCE_1	56.38
EXT_SOURCE_2	0.21
EXT_SOURCE_3	19.83
APARTMENTS_AVG	50.75
BASEMENTAREA_AVG	58.52
YEARS_BEGINEXPLUATATION_AVG	48.78
YEARS_BUILD_AVG	66.50
COMMONAREA_AVG	69.87
ELEVATORS_AVG	53.30
ENTRANCES_AVG	50.35
FLOORSMAX_AVG	49.76
FLOORSMIN_AVG	67.85
LANDAREA_AVG	59.38
LIVINGAPARTMENTS_AVG	68.35
LIVINGAREA_AVG	50.19
NONLIVINGAPARTMENTS_AVG	69.43
NONLIVINGAREA_AVG	55.18
APARTMENTS_MODE	50.75
BASEMENTAREA_MODE	58.52
YEARS_BEGINEXPLUATATION_MODE	48.78
YEARS_BUILD_MODE	66.50
COMMONAREA_MODE	69.87
ELEVATORS_MODE	53.30

```
ENTRANCES_MODE                  50.35
FLOORSMAX_MODE                  49.76
FLOORSMIN_MODE                  67.85
LANDAREA_MODE                   59.38
LIVINGAPARTMENTS_MODE           68.35
LIVINGAREA_MODE                 50.19
NONLIVINGAPARTMENTS_MODE        69.43
NONLIVINGAREA_MODE              55.18
APARTMENTS_MEDI                 50.75
BASEMENTAREA_MEDI               58.52
YEARS_BEGINEXPLUATATION_MEDI    48.78
YEARS_BUILD_MEDI                66.50
COMMONAREA_MEDI                 69.87
ELEVATORS_MEDI                  53.30
ENTRANCES_MEDI                  50.35
FLOORSMAX_MEDI                  49.76
FLOORSMIN_MEDI                  67.85
LANDAREA_MEDI                   59.38
LIVINGAPARTMENTS_MEDI           68.35
LIVINGAREA_MEDI                 50.19
NONLIVINGAPARTMENTS_MEDI        69.43
NONLIVINGAREA_MEDI              55.18
FONDKAPREMONT_MODE              68.39
HOUSETYPE_MODE                  50.18
TOTALAREA_MODE                  48.27
WALLSMATERIAL_MODE              50.84
EMERGENCYSTATE_MODE             47.40
OBS_30_CNT_SOCIAL_CIRCLE         0.33
DEF_30_CNT_SOCIAL_CIRCLE         0.33
OBS_60_CNT_SOCIAL_CIRCLE         0.33
DEF_60_CNT_SOCIAL_CIRCLE         0.33
DAYS_LAST_PHONE_CHANGE           0.00
FLAG_DOCUMENT_2                  0.00
FLAG_DOCUMENT_3                  0.00
FLAG_DOCUMENT_4                  0.00
FLAG_DOCUMENT_5                  0.00
FLAG_DOCUMENT_6                  0.00
FLAG_DOCUMENT_7                  0.00
FLAG_DOCUMENT_8                  0.00
FLAG_DOCUMENT_9                  0.00
FLAG_DOCUMENT_10                 0.00
FLAG_DOCUMENT_11                 0.00
FLAG_DOCUMENT_12                 0.00
FLAG_DOCUMENT_13                 0.00
FLAG_DOCUMENT_14                 0.00
FLAG_DOCUMENT_15                 0.00
FLAG_DOCUMENT_16                 0.00
FLAG_DOCUMENT_17                 0.00
FLAG_DOCUMENT_18                 0.00
FLAG_DOCUMENT_19                 0.00
FLAG_DOCUMENT_20                 0.00
FLAG_DOCUMENT_21                 0.00
AMT_REQ_CREDIT_BUREAU_HOUR      13.50
AMT_REQ_CREDIT_BUREAU_DAY       13.50
AMT_REQ_CREDIT_BUREAU_WEEK      13.50
AMT_REQ_CREDIT_BUREAU_MON       13.50
AMT_REQ_CREDIT_BUREAU_QRT       13.50
AMT_REQ_CREDIT_BUREAU_YEAR      13.50
dtype: float64
```

In [335…

```
# plotting bar plot to visualize columns of application data which have missing per

plt.figure(figsize=(20,10))
plt.axhline(40,ls='-.',color='red')
```

```
check_null_values(appdf).plot.bar()
plt.title("Missing Values percentage in Application Data")
plt.xlabel("Columns")
plt.ylabel("Null Values in percentage")
plt.show()
```



Inferences:

We can see from the plots that,

the columns above redline mark are the columns with more than 40% missing values.

the columns below redline mark are the columns with less than 40% missing values.

In [336…

```
# Lets get the columns with the null values equal to or greater than 40%

null_columns_appdf = pd.DataFrame(appdf.isnull().sum()/len(appdf.index)*100).reset_
null_columns_appdf.columns=['column names','null values percentage']

null_columns_40_appdf = null_columns_appdf[null_columns_appdf['null values percenta
null_columns_40_appdf
```

Out[336]:

	column names	null values percentage
21	OWN_CAR_AGE	65.990810
41	EXT_SOURCE_1	56.381073
44	APARTMENTS_AVG	50.749729
45	BASEMENTAREA_AVG	58.515956
46	YEARS_BEGINEXPLUATATION_AVG	48.781019
47	YEARS_BUILD_AVG	66.497784
48	COMMONAREA_AVG	69.872297
49	ELEVATORS_AVG	53.295980
50	ENTRANCES_AVG	50.348768
51	FLOORSMAX_AVG	49.760822
52	FLOORSMIN_AVG	67.848630
53	LANDAREA_AVG	59.376738
54	LIVINGAPARTMENTS_AVG	68.354953
55	LIVINGAREA_AVG	50.193326
56	NONLIVINGAPARTMENTS_AVG	69.432963
57	NONLIVINGAREA_AVG	55.179164
58	APARTMENTS_MODE	50.749729
59	BASEMENTAREA_MODE	58.515956
60	YEARS_BEGINEXPLUATATION_MODE	48.781019
61	YEARS_BUILD_MODE	66.497784
62	COMMONAREA_MODE	69.872297
63	ELEVATORS_MODE	53.295980
64	ENTRANCES_MODE	50.348768
65	FLOORSMAX_MODE	49.760822
66	FLOORSMIN_MODE	67.848630
67	LANDAREA_MODE	59.376738
68	LIVINGAPARTMENTS_MODE	68.354953
69	LIVINGAREA_MODE	50.193326
70	NONLIVINGAPARTMENTS_MODE	69.432963
71	NONLIVINGAREA_MODE	55.179164
72	APARTMENTS_MEDI	50.749729
73	BASEMENTAREA_MEDI	58.515956
74	YEARS_BEGINEXPLUATATION_MEDI	48.781019
75	YEARS_BUILD_MEDI	66.497784
76	COMMONAREA_MEDI	69.872297
77	ELEVATORS_MEDI	53.295980

	column names	null values percentage
78	ENTRANCES_MEDI	50.348768
79	FLOORSMAX_MEDI	49.760822
80	FLOORSMIN_MEDI	67.848630
81	LANDAREA_MEDI	59.376738
82	LIVINGAPARTMENTS_MEDI	68.354953
83	LIVINGAREA_MEDI	50.193326
84	NONLIVINGAPARTMENTS_MEDI	69.432963
85	NONLIVINGAREA_MEDI	55.179164
86	FONDKAPREMONT_MODE	68.386172
87	HOUSETYPE_MODE	50.176091
88	TOTALAREA_MODE	48.268517
89	WALLSMATERIAL_MODE	50.840783
90	EMERGENCYSTATE_MODE	47.398304

In [337…
```
len(null_columns_40_appdf)
```

Out[337]:
```
49
```

Inference:

After taking the cut off value as 40% for missing values we can see that 49 columns have missing values more than 40%

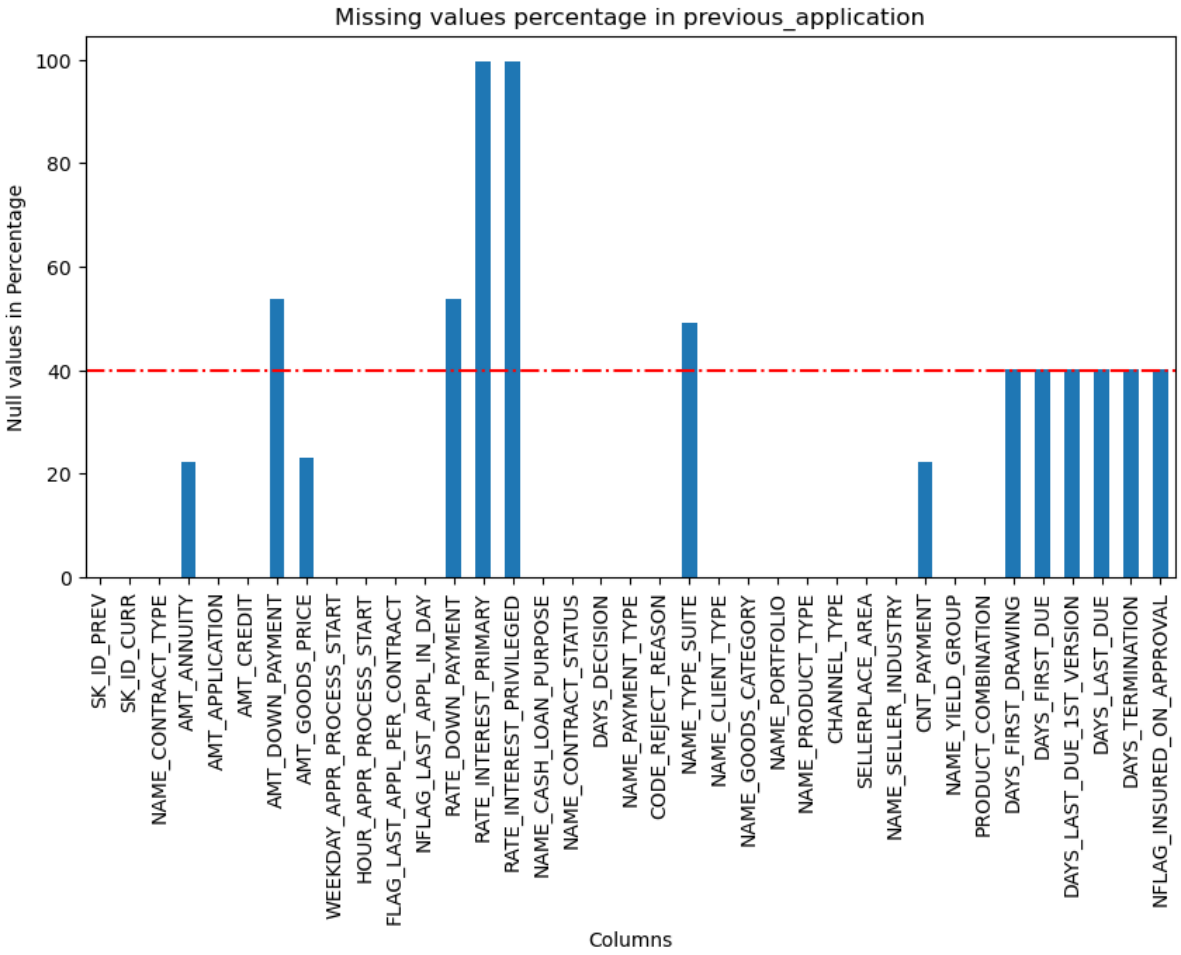
majority of the columns with missing values are related to the estate/property owned by the applicants

PREVDF MISSING VALUES

In [338…
```
# now lets check columns of previous_application dataset for null values

check_null_values(prevdf)
```

Out[338]:
```
SK_ID_PREV                    0.00
SK_ID_CURR                    0.00
NAME_CONTRACT_TYPE            0.00
AMT_ANNUITY                  22.29
AMT_APPLICATION               0.00
AMT_CREDIT                    0.00
AMT_DOWN_PAYMENT             53.64
AMT_GOODS_PRICE              23.08
WEEKDAY_APPR_PROCESS_START    0.00
HOUR_APPR_PROCESS_START       0.00
FLAG_LAST_APPL_PER_CONTRACT   0.00
NFLAG_LAST_APPL_IN_DAY        0.00
RATE_DOWN_PAYMENT            53.64
RATE_INTEREST_PRIMARY        99.64
RATE_INTEREST_PRIVILEGED     99.64
NAME_CASH_LOAN_PURPOSE        0.00
NAME_CONTRACT_STATUS          0.00
DAYS_DECISION                 0.00
NAME_PAYMENT_TYPE             0.00
CODE_REJECT_REASON            0.00
NAME_TYPE_SUITE              49.12
NAME_CLIENT_TYPE              0.00
NAME_GOODS_CATEGORY           0.00
NAME_PORTFOLIO                0.00
NAME_PRODUCT_TYPE             0.00
CHANNEL_TYPE                  0.00
SELLERPLACE_AREA              0.00
NAME_SELLER_INDUSTRY          0.00
CNT_PAYMENT                  22.29
NAME_YIELD_GROUP              0.00
PRODUCT_COMBINATION           0.02
DAYS_FIRST_DRAWING           40.30
DAYS_FIRST_DUE               40.30
DAYS_LAST_DUE_1ST_VERSION    40.30
DAYS_LAST_DUE                40.30
DAYS_TERMINATION             40.30
NFLAG_INSURED_ON_APPROVAL    40.30
dtype: float64
```

In [339…
```
# plotting bar plot to visualize columns of previous application data which have mi

plt.figure(figsize=(10,5))
plt.axhline(40,ls='-.',color='red')
check_null_values(prevdf).plot.bar()
plt.title('Missing values percentage in previous_application')
plt.xlabel("Columns")
plt.ylabel('Null values in Percentage')
plt.show()
```



Inferences:

as we can see their are many columns in previous_application_data whose null values are more than 40%

In [340…

```
# Lets get the columns with the null values equal to or greater than 40%

null_columns_prevdf = pd.DataFrame(prevdf.isnull().sum()/len(prevdf.index)*100).res
null_columns_prevdf.columns=['column names','null values percentage']

null_columns_40_prevdf = null_columns_prevdf[null_columns_prevdf['null values perce
null_columns_40_prevdf
```

Out[340]:

	column names	null values percentage
6	AMT_DOWN_PAYMENT	53.636480
12	RATE_DOWN_PAYMENT	53.636480
13	RATE_INTEREST_PRIMARY	99.643698
14	RATE_INTEREST_PRIVILEGED	99.643698
20	NAME_TYPE_SUITE	49.119754
31	DAYS_FIRST_DRAWING	40.298129
32	DAYS_FIRST_DUE	40.298129
33	DAYS_LAST_DUE_1ST_VERSION	40.298129
34	DAYS_LAST_DUE	40.298129
35	DAYS_TERMINATION	40.298129
36	NFLAG_INSURED_ON_APPROVAL	40.298129

In [341…
```
len(null_columns_40_prevdf)
```

Out[341]:
```
11
```

Inferences:

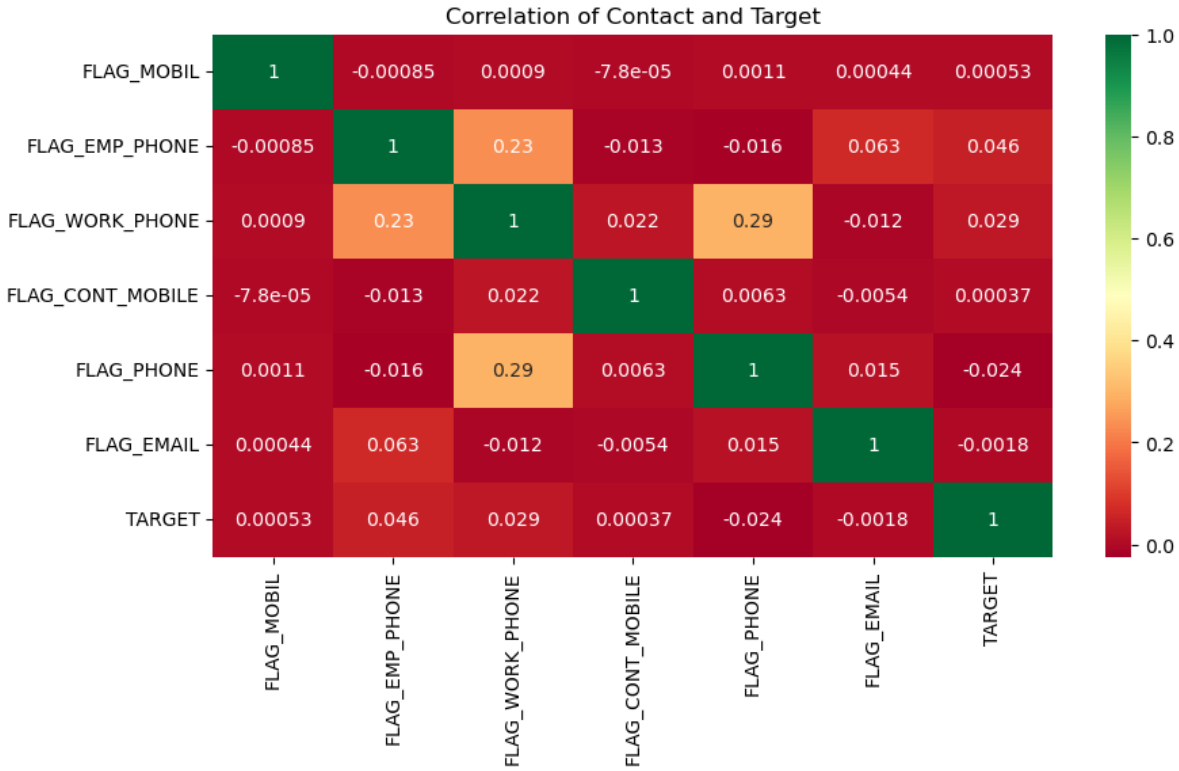
After taking the cut off value as 40% for missing values we can see that 11 columns have missing values more than 40%

Deleting unnecessary data from application_data

Checking correlation of all the contact details columns behavior against Target column.

In [342…
```
# plotting heatmap for correlation of unnecessary columns.

contact_details_columns = ['FLAG_MOBIL','FLAG_EMP_PHONE','FLAG_WORK_PHONE','FLAG_CO
plt.figure(figsize=(10,5))
sns.heatmap(appdf[contact_details_columns].corr(),annot=True,cmap='RdYlGn')
plt.title('Correlation of Contact and Target')
plt.show
```

Out[342]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



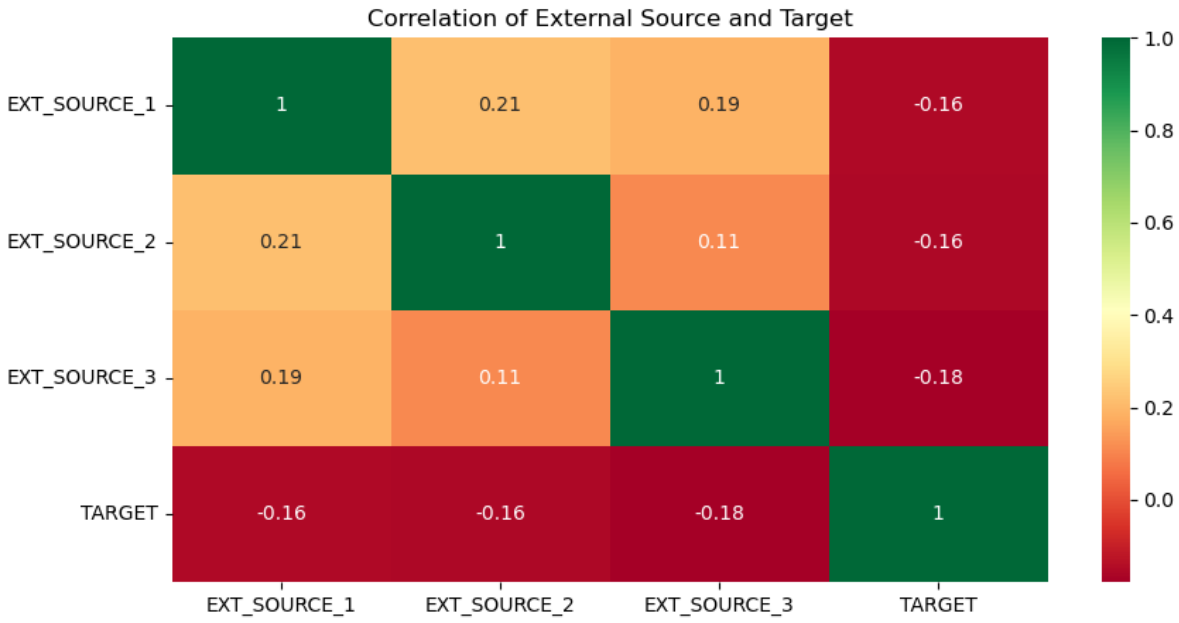
Inferences:

According to above heatmap their is no correlation seen between the contact details columns and target column, so we need to remove these columns also.

Checking correlation of all the external sources columns and target column

In [343…

```
# plotting heatmap for EXT_SOURCE columns and target variable to check the relation

Ext_source_columns = ['EXT_SOURCE_1','EXT_SOURCE_2','EXT_SOURCE_3','TARGET']
plt.figure(figsize=(10,5))
sns.heatmap(appdf[Ext_source_columns].corr(),annot=True,cmap='RdYlGn')
plt.title('Correlation of External Source and Target')
plt.show()
```



Inferences:

Here also according to above heatmap their is no correlation seen between the Ext_source columns and target column. so we need to remove these columns also.

In [344…
```
# getting all the flag_document columns from the lsit of columns in appdf datasets

documents_columns = [x for x in appdf.columns if 'FLAG_DOCUMENT' in x]
documents_columns
```

Out[344]:
```
['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']
```

Checking correlation of all the flag_document columns and target column

In [345…
```
plt.figure(figsize=(20,25))
for i in range(len(documents_columns)):
    plt.subplot(5,4,i+1)
    plt.title(documents_columns[i])
    sns.countplot(x=appdf[documents_columns[i]], hue=appdf['TARGET'],data=appdf)
plt.tight_layout()
plt.show()
```

inferences:

Most of applicants who applied for the loan has not submitted FLAG_DOCUMENT_x except FLAG_DOCUMENT_3. so we can delete rest other columns except FLAG_DOCUMENT_3

In [346…

```
# removing flag_document_3 from the list as we need it for analysis
documents_columns.remove('FLAG_DOCUMENT_3')

#Convert all the column names to list and storing it in variable
to_delete_columns_appdf = null_columns_40_appdf['column names'].to_list()+['EXT_SOU

#adding other unnecessary columns
to_delete_columns_appdf = to_delete_columns_appdf+documents_columns+contact_details
```

```
#removing the target column from the list as we need it for analysis
to_delete_columns_appdf.remove('TARGET')
```

In [347…

```
# checking the columns which are to be removed

to_delete_columns_appdf
```

Out[347]:
```
['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',
 'EXT_SOURCE_2',
 'EXT_SOURCE_3',
 'FLAG_DOCUMENT_2',
 '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',
 'FLAG_MOBIL',
 'FLAG_EMP_PHONE',
 'FLAG_WORK_PHONE',
 'FLAG_CONT_MOBILE',
 'FLAG_PHONE',
 'FLAG_EMAIL']
```

In [348…
```
#dropping the columns which are not needed
appdf.drop(labels=to_delete_columns_appdf,axis=1,inplace=True)
```

In [349…
```
# checking our filtered application_data
appdf.shape
```

Out[349]:
```
(307511, 46)
```

In [350…
```
appdf.head()
```

Out[350]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_RE
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 × 46 columns

Deleting the unnecessary columns from Previous data

In [351…
```
null_columns_40_prevdf
```

Out[351]:

	column names	null values percentage
6	AMT_DOWN_PAYMENT	53.636480
12	RATE_DOWN_PAYMENT	53.636480
13	RATE_INTEREST_PRIMARY	99.643698
14	RATE_INTEREST_PRIVILEGED	99.643698
20	NAME_TYPE_SUITE	49.119754
31	DAYS_FIRST_DRAWING	40.298129
32	DAYS_FIRST_DUE	40.298129
33	DAYS_LAST_DUE_1ST_VERSION	40.298129
34	DAYS_LAST_DUE	40.298129
35	DAYS_TERMINATION	40.298129
36	NFLAG_INSURED_ON_APPROVAL	40.298129

In [352…
```
# getting the list of unwanted columns which needs to be deleted

to_delete_columns=['AMT_DOWN_PAYMENT','RATE_DOWN_PAYMENT','RATE_INTEREST_PRIMARY','
                   'FLAG_LAST_APPL_PER_CONTRACT','NFLAG_LAST_APPL_IN_DAY']

#checking the numer of columns to be deleted.
len(to_delete_columns)
```

Out[352]:
```
15
```

In [353…
```
#Deleting these 15 columns from the previous_data which will be not used for our an
prevdf.drop(labels=to_delete_columns,axis=1,inplace=True)
```

In [354…
```
prevdf.shape
```

Out[354]:
```
(1670214, 22)
```

Standardising Values

After observing all the columns from the datasets I have observed that all the days columns contains negative values, but as we know days cannot be in negative format so we need to convert that into positive.

In [355…
```
appdf['DAYS_BIRTH']=abs(appdf['DAYS_BIRTH'])
appdf['DAYS_EMPLOYED']=abs(appdf['DAYS_EMPLOYED'])
appdf['DAYS_REGISTRATION']=abs(appdf['DAYS_REGISTRATION'])
appdf['DAYS_ID_PUBLISH']=abs(appdf['DAYS_ID_PUBLISH'])
```

In [356…
```
prevdf['DAYS_DECISION']=abs(prevdf['DAYS_DECISION'])
```

Binning Age

In [357…
```
# lets first create a new column for the age group of the clients

appdf['Age_years']=abs(appdf.DAYS_BIRTH//365)
#converting days into years
```

```
#Lets create the parameters required for binning.
bins = [0,20,30,40,50,60,70,80]
slots=['0-20','20-30','30-40','40-50','50-60','60-70','70 and above']
appdf['Age_Group']=pd.cut(appdf['Age_years'],bins=bins,labels=slots)
```

In [358…

```
appdf.head()
```

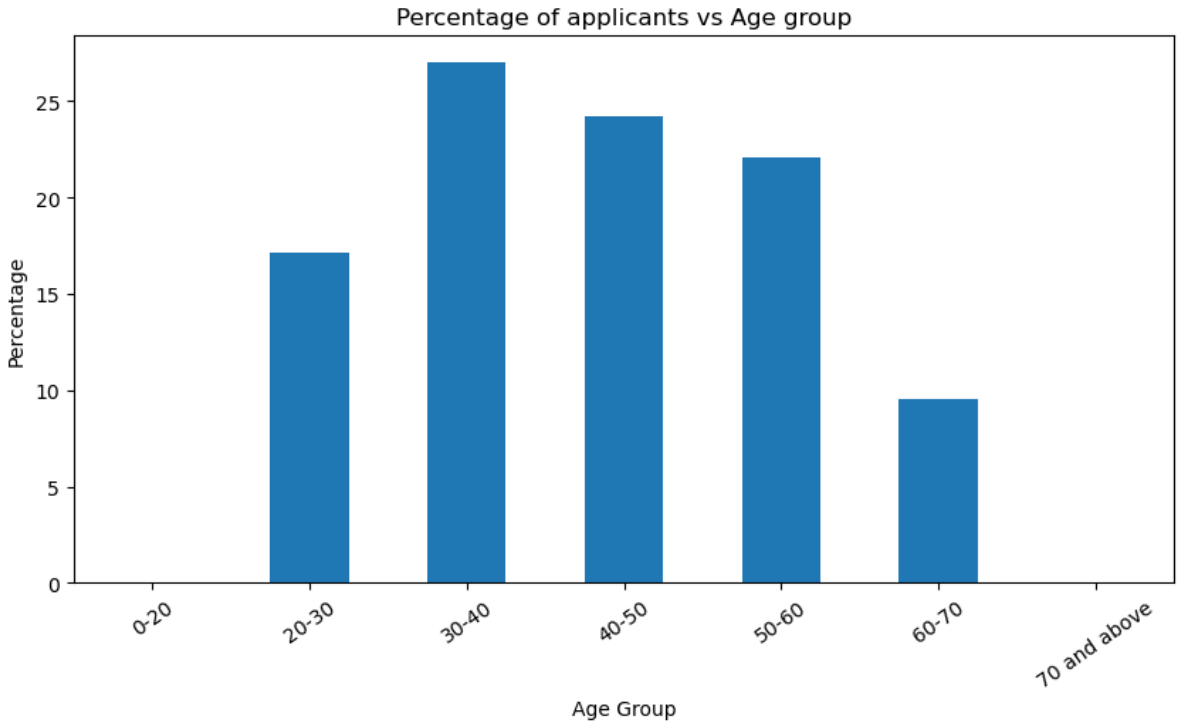
Out[358]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_RE
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 × 48 columns

In [359…

```
# checking percentage of each category of Age_Group

plt.figure(figsize=(10,5))
(appdf['Age_Group'].value_counts(normalize=True)*100).sort_index().plot.bar()
plt.title('Percentage of applicants vs Age group')
plt.xlabel('Age Group')
plt.ylabel('Percentage')
plt.xticks(rotation=35)
plt.show()
```



Inferences:

More than 50% of the applicants belong to age group of 30 to 50 years.

In [360…

```
appdf.head()
```

Out[360]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_RE
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 × 48 columns

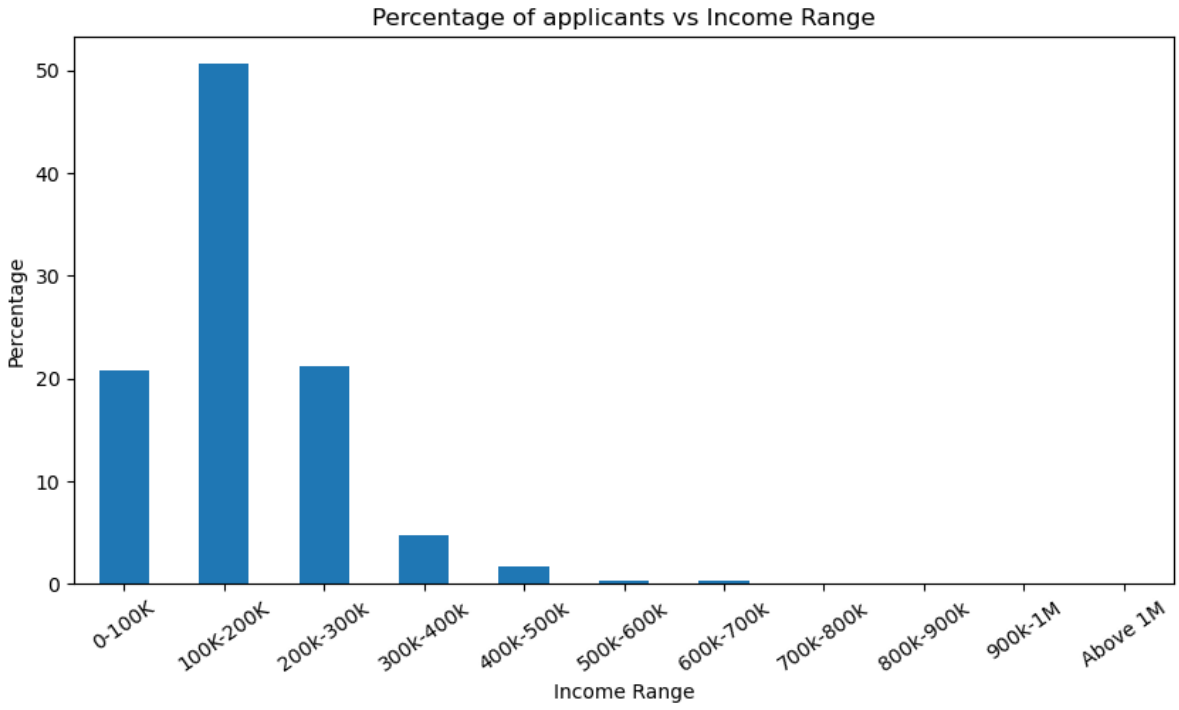
Binning Income

In [361…

```
# lets first create a new column after converting income amount.
appdf['AMT_INCOME_TOTAL']=appdf.AMT_INCOME_TOTAL/100000

#lets create the parameters required for binning.
bins = [0,1,2,3,4,5,6,7,8,9,10,50]
slots=['0-100K','100K-200K', '200k-300k','300k-400k','400k-500k','500k-600k','600k-
appdf['AMT_INCOME_TOTAL_RANGE']=pd.cut(appdf['AMT_INCOME_TOTAL'],bins=bins,labels=s
```

In [362…

```
# checking percentage of each category of AMT_INCOME_TOTAL

plt.figure(figsize=(10,5))
(appdf['AMT_INCOME_TOTAL_RANGE'].value_counts(normalize=True)*100).sort_index().plo
plt.title('Percentage of applicants vs Income Range')
plt.xlabel('Income Range')
plt.ylabel('Percentage')
plt.xticks(rotation=35)
plt.show()
```



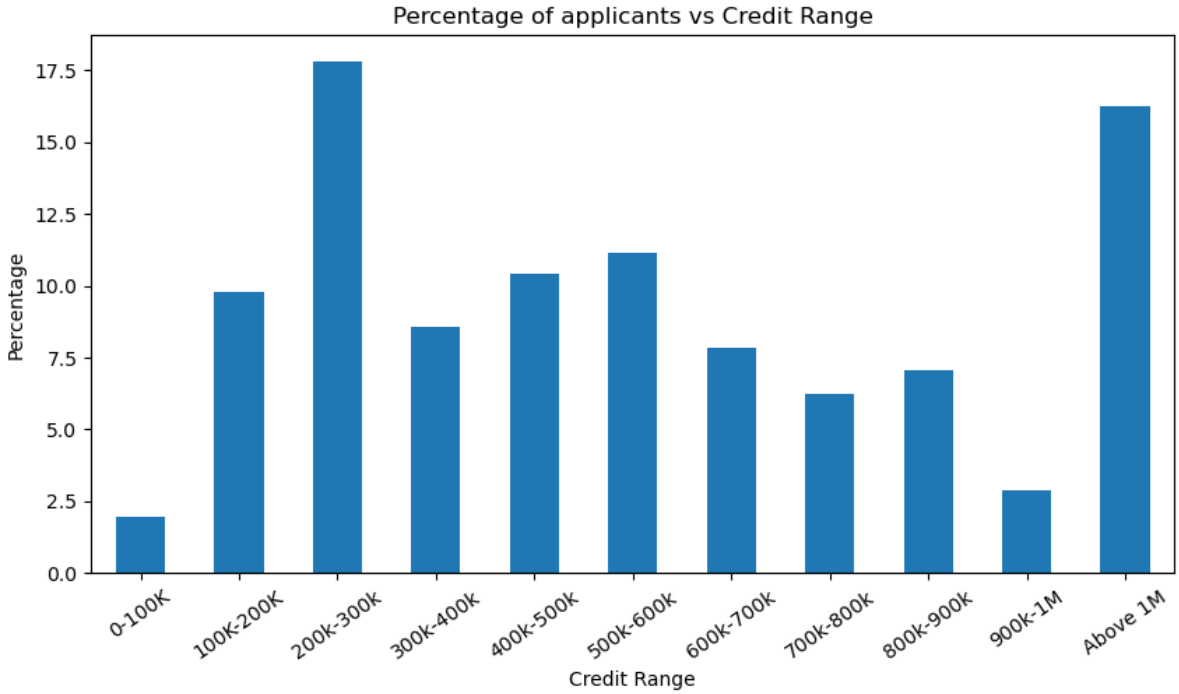
Inferences:

50% applicants have income in the range of 100k to 200k almost 91% applicants have income in the range 0 to 300k less than 10% applicants have income more than 300k

Binning Credit

In [363…
```
# creating column by converting credit amount.
appdf['AMT_CREDIT']=appdf.AMT_CREDIT/100000

#Lets create the parameters required for binning.
bins=[0,1,2,3,4,5,6,7,8,9,10,50]
slots=['0-100K','100K-200K', '200k-300k','300k-400k','400k-500k','500k-600k','600k-
appdf['AMT_CREDIT_RANGE']=pd.cut(appdf['AMT_CREDIT'],bins=bins,labels=slots)
```

In [364…
```
#Checking percentage of each category of AMT_CREDIT

plt.figure(figsize=(10,5))
(appdf['AMT_CREDIT_RANGE'].value_counts(normalize=True)*100).sort_index().plot.bar(
plt.title('Percentage of applicants vs Credit Range')
plt.xlabel('Credit Range')
plt.ylabel('Percentage')
plt.xticks(rotation=35)
plt.show()
```



inference:

Around 16% applicants took loan of above 1M

Almost 18% applicants took loan in the range of 200k to 300k

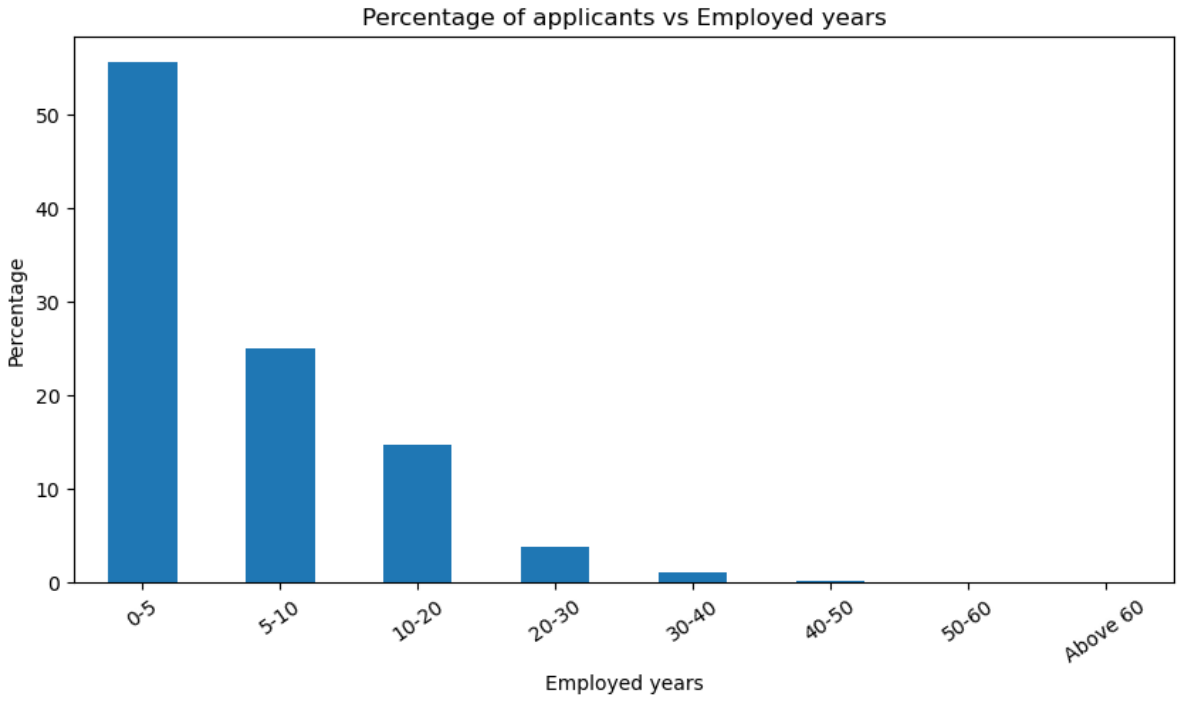
Binning Employment Years

In [365…
```
# creating column by converting days into years.
appdf['employed_years']=appdf.DAYS_EMPLOYED//365
```

```
# creating parameters required for binning.
bins=[0,5,10,20,30,40,50,60,100]
slots=['0-5','5-10','10-20','20-30','30-40','40-50','50-60','Above 60']
appdf['YEARS_EMPLOYED']=pd.cut(appdf['employed_years'],bins=bins,labels=slots)
```

In [366…

```
#Checking percentage of each category of YEARS_EMPLOYED

plt.figure(figsize=(10,5))
(appdf['YEARS_EMPLOYED'].value_counts(normalize=True)*100).sort_index().plot.bar()
plt.title('Percentage of applicants vs Employed years')
plt.xlabel('Employed years')
plt.ylabel('Percentage')
plt.xticks(rotation=35)
plt.show()
```



Inference:

More than 50% applicants are employed for last 5 years whereas almost 80% applicants have 10 years employment.

NULL VALUE IMPUTATION

In [367…

```
# checking null values in all the columns of application data

check_null_values(appdf)
```

Out[367]:

```
SK_ID_CURR                    0.00
TARGET                        0.00
NAME_CONTRACT_TYPE            0.00
CODE_GENDER                   0.00
FLAG_OWN_CAR                  0.00
FLAG_OWN_REALTY               0.00
CNT_CHILDREN                  0.00
AMT_INCOME_TOTAL              0.00
AMT_CREDIT                    0.00
AMT_ANNUITY                   0.00
AMT_GOODS_PRICE               0.09
NAME_TYPE_SUITE               0.42
NAME_INCOME_TYPE              0.00
NAME_EDUCATION_TYPE           0.00
NAME_FAMILY_STATUS            0.00
NAME_HOUSING_TYPE             0.00
REGION_POPULATION_RELATIVE    0.00
DAYS_BIRTH                    0.00
DAYS_EMPLOYED                 0.00
DAYS_REGISTRATION             0.00
DAYS_ID_PUBLISH               0.00
OCCUPATION_TYPE              31.35
CNT_FAM_MEMBERS               0.00
REGION_RATING_CLIENT          0.00
REGION_RATING_CLIENT_W_CITY   0.00
WEEKDAY_APPR_PROCESS_START    0.00
HOUR_APPR_PROCESS_START       0.00
REG_REGION_NOT_LIVE_REGION    0.00
REG_REGION_NOT_WORK_REGION    0.00
LIVE_REGION_NOT_WORK_REGION   0.00
REG_CITY_NOT_LIVE_CITY        0.00
REG_CITY_NOT_WORK_CITY        0.00
LIVE_CITY_NOT_WORK_CITY       0.00
ORGANIZATION_TYPE             0.00
OBS_30_CNT_SOCIAL_CIRCLE      0.33
DEF_30_CNT_SOCIAL_CIRCLE      0.33
OBS_60_CNT_SOCIAL_CIRCLE      0.33
DEF_60_CNT_SOCIAL_CIRCLE      0.33
DAYS_LAST_PHONE_CHANGE        0.00
FLAG_DOCUMENT_3               0.00
AMT_REQ_CREDIT_BUREAU_HOUR   13.50
AMT_REQ_CREDIT_BUREAU_DAY    13.50
AMT_REQ_CREDIT_BUREAU_WEEK   13.50
AMT_REQ_CREDIT_BUREAU_MON    13.50
AMT_REQ_CREDIT_BUREAU_QRT    13.50
AMT_REQ_CREDIT_BUREAU_YEAR   13.50
Age_years                     0.00
Age_Group                     0.00
AMT_INCOME_TOTAL_RANGE        0.00
AMT_CREDIT_RANGE              0.00
employed_years                0.00
YEARS_EMPLOYED               27.08
dtype: float64
```

Null Values Imputation Suggestions for Application data

1. As seen in the percentage data above as occupation_type has the highest missing percentage (i.e. 31.35) We can create a new seperate column for this as replacing this column with any other values may lead to a misleading data.

2. WE can also impute numerical variables with the median as there are no outliers that can be seen from results of describe() and mean() also from boxplots.

3. Impute 'NAME_TYPE_SUIT' which is (0.42 %) a categorical variable with mode. (categorical value)

In [368…
```
appdf.AMT_GOODS_PRICE.describe()
```

Out[368]:
```
count    3.072330e+05
mean     5.383962e+05
std      3.694465e+05
min      4.050000e+04
25%      2.385000e+05
50%      4.500000e+05
75%      6.795000e+05
max      4.050000e+06
Name: AMT_GOODS_PRICE, dtype: float64
```

In [369…
```
appdf['AMT_GOODS_PRICE'].fillna(53839.62, inplace=True)
```

In [370…
```
appdf['AMT_ANNUITY'].describe()
```

Out[370]:
```
count    307499.000000
mean      27108.573909
std       14493.737315
min        1615.500000
25%       16524.000000
50%       24903.000000
75%       34596.000000
max      258025.500000
Name: AMT_ANNUITY, dtype: float64
```

In [371…
```
appdf['AMT_ANNUITY'].fillna(27108, inplace=True)
```

In [372…
```
appdf['NAME_TYPE_SUITE'].mode()
```

Out[372]:
```
0    Unaccompanied
Name: NAME_TYPE_SUITE, dtype: object
```

In [373…
```
appdf['NAME_TYPE_SUITE'].fillna('unaccompanied', inplace=True)
```

In [374…
```
appdf['OCCUPATION_TYPE'].mode()
```

Out[374]:
```
0    Laborers
Name: OCCUPATION_TYPE, dtype: object
```

In [375…
```
appdf['OCCUPATION_TYPE'].fillna('unaccompanied', inplace=True)
```

In [376…
```
appdf['OBS_30_CNT_SOCIAL_CIRCLE'].describe()
```

Out[376]:
```
count    306490.000000
mean          1.422245
std           2.400989
min           0.000000
25%           0.000000
50%           0.000000
75%           2.000000
max         348.000000
Name: OBS_30_CNT_SOCIAL_CIRCLE, dtype: float64
```

In [377…
```
appdf['OBS_30_CNT_SOCIAL_CIRCLE'].fillna(1.422245, inplace=True)
```

In [378…
```
appdf['DEF_30_CNT_SOCIAL_CIRCLE'].describe()
```

Out[378]:
```
count    306490.000000
mean          0.143421
std           0.446698
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max          34.000000
Name: DEF_30_CNT_SOCIAL_CIRCLE, dtype: float64
```

In [379…
```
appdf['DEF_30_CNT_SOCIAL_CIRCLE'].fillna(0.143421, inplace=True)
```

In [380…
```
appdf['OBS_60_CNT_SOCIAL_CIRCLE'].describe()
```

Out[380]:
```
count    306490.000000
mean          1.405292
std           2.379803
min           0.000000
25%           0.000000
50%           0.000000
75%           2.000000
max         344.000000
Name: OBS_60_CNT_SOCIAL_CIRCLE, dtype: float64
```

In [381…
```
appdf['OBS_60_CNT_SOCIAL_CIRCLE'].fillna(1.405292, inplace=True)
```

In [382…
```
appdf['DEF_60_CNT_SOCIAL_CIRCLE'].describe()
```

Out[382]:
```
count    306490.000000
mean          0.100049
std           0.362291
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max          24.000000
Name: DEF_60_CNT_SOCIAL_CIRCLE, dtype: float64
```

In [383…
```
appdf['DEF_60_CNT_SOCIAL_CIRCLE'].fillna(0.100049, inplace=True)
```

In [384…
```
appdf['AMT_REQ_CREDIT_BUREAU_HOUR'].describe()
```

Out[384]:
```
count    265992.000000
mean          0.006402
std           0.083849
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           4.000000
Name: AMT_REQ_CREDIT_BUREAU_HOUR, dtype: float64
```

In [385…
```
appdf['AMT_REQ_CREDIT_BUREAU_HOUR'].fillna(0.006402, inplace=True)
```

In [386…
```
appdf['AMT_REQ_CREDIT_BUREAU_DAY'].describe()
```

Out[386]:
```
count    265992.000000
mean          0.007000
std           0.110757
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           9.000000
Name: AMT_REQ_CREDIT_BUREAU_DAY, dtype: float64
```

In [387…
```
appdf['AMT_REQ_CREDIT_BUREAU_DAY'].fillna(0.007000, inplace=True)
```

In [388…
```
appdf['AMT_REQ_CREDIT_BUREAU_WEEK'].describe()
```

Out[388]:
```
count    265992.000000
mean          0.034362
std           0.204685
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           8.000000
Name: AMT_REQ_CREDIT_BUREAU_WEEK, dtype: float64
```

In [389…
```
appdf['AMT_REQ_CREDIT_BUREAU_WEEK'].fillna(0.034362, inplace=True)
```

In [390…
```
appdf['AMT_REQ_CREDIT_BUREAU_MON'].describe()
```

Out[390]:
```
count    265992.000000
mean          0.267395
std           0.916002
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max          27.000000
Name: AMT_REQ_CREDIT_BUREAU_MON, dtype: float64
```

In [391…
```
appdf['AMT_REQ_CREDIT_BUREAU_MON'].fillna(0.267395, inplace=True)
```

In [392…
```
appdf['AMT_REQ_CREDIT_BUREAU_QRT'].describe()
```

Out[392]:
```
count    265992.000000
mean          0.265474
std           0.794056
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max         261.000000
Name: AMT_REQ_CREDIT_BUREAU_QRT, dtype: float64
```

In [393…
```
appdf['AMT_REQ_CREDIT_BUREAU_QRT'].fillna(0.265474, inplace=True)
```

In [394…
```
appdf['AMT_REQ_CREDIT_BUREAU_YEAR'].describe()
```

Out[394]:
```
count    265992.000000
mean          1.899974
std           1.869295
min           0.000000
25%           0.000000
50%           1.000000
75%           3.000000
max          25.000000
Name: AMT_REQ_CREDIT_BUREAU_YEAR, dtype: float64
```

In [395…
```
appdf['AMT_REQ_CREDIT_BUREAU_YEAR'].fillna(1.899974, inplace=True)
```

In [396…
```
appdf['YEARS_EMPLOYED'].describe()
```

Out[396]:
```
count     224233
unique         6
top          0-5
freq      124634
Name: YEARS_EMPLOYED, dtype: object
```

In [397…
```
appdf['YEARS_EMPLOYED'].fillna('0-5', inplace=True)
```

In [398…
```
# checking null values in all the columns of previous application data

check_null_values(prevdf)
```

Out[398]:
```
SK_ID_PREV                 0.00
SK_ID_CURR                 0.00
NAME_CONTRACT_TYPE         0.00
AMT_ANNUITY               22.29
AMT_APPLICATION            0.00
AMT_CREDIT                 0.00
AMT_GOODS_PRICE           23.08
NAME_CASH_LOAN_PURPOSE     0.00
NAME_CONTRACT_STATUS       0.00
DAYS_DECISION              0.00
NAME_PAYMENT_TYPE          0.00
CODE_REJECT_REASON         0.00
NAME_CLIENT_TYPE           0.00
NAME_GOODS_CATEGORY        0.00
NAME_PORTFOLIO             0.00
NAME_PRODUCT_TYPE          0.00
CHANNEL_TYPE               0.00
SELLERPLACE_AREA           0.00
NAME_SELLER_INDUSTRY       0.00
CNT_PAYMENT               22.29
NAME_YIELD_GROUP           0.00
PRODUCT_COMBINATION        0.02
dtype: float64
```

Null Values Imputation Suggestions for Previous Application Data.

1. AMT_ANNUITY column in previous data can be imputed with mean but if the data is too skewed then we can impute it with median.

2. AMT_GOOD_PRICE can be imputed with mode as it reflects the most common goods price present in the dataset.

In [399…
```
prevdf['AMT_ANNUITY'].describe()
```

Out[399]:
```
count    1.297979e+06
mean     1.595512e+04
std      1.478214e+04
min      0.000000e+00
25%      6.321780e+03
50%      1.125000e+04
75%      2.065842e+04
max      4.180581e+05
Name: AMT_ANNUITY, dtype: float64
```

In [400…
```
prevdf['AMT_ANNUITY'].fillna(18440.73, inplace=True)
```

In [401…
```
prevdf['AMT_GOODS_PRICE'].mode()
```

Out[401]:
```
0    45000.0
Name: AMT_GOODS_PRICE, dtype: float64
```

In [402…
```
prevdf['AMT_GOODS_PRICE'].fillna(450000, inplace=True)
```

In [403…
```
prevdf['CNT_PAYMENT'].describe()
```

Out[403]:
```
count    1.297984e+06
mean     1.605408e+01
std      1.456729e+01
min      0.000000e+00
25%      6.000000e+00
50%      1.200000e+01
75%      2.400000e+01
max      8.400000e+01
Name: CNT_PAYMENT, dtype: float64
```

In [404…
```
prevdf['CNT_PAYMENT'].fillna(16054.08, inplace=True)
```

In [405…
```
prevdf['PRODUCT_COMBINATION'].mode()
```

Out[405]:
```
0    Cash
Name: PRODUCT_COMBINATION, dtype: object
```

In [406…
```
prevdf['PRODUCT_COMBINATION'].fillna('Cash', inplace=True)
```

In [407…
```
#### cross checking if there are any null values.

check_null_values(appdf)
```

Out[407]:

```
SK_ID_CURR                    0.0
TARGET                        0.0
NAME_CONTRACT_TYPE            0.0
CODE_GENDER                   0.0
FLAG_OWN_CAR                  0.0
FLAG_OWN_REALTY               0.0
CNT_CHILDREN                  0.0
AMT_INCOME_TOTAL              0.0
AMT_CREDIT                    0.0
AMT_ANNUITY                   0.0
AMT_GOODS_PRICE               0.0
NAME_TYPE_SUITE               0.0
NAME_INCOME_TYPE              0.0
NAME_EDUCATION_TYPE           0.0
NAME_FAMILY_STATUS            0.0
NAME_HOUSING_TYPE             0.0
REGION_POPULATION_RELATIVE    0.0
DAYS_BIRTH                    0.0
DAYS_EMPLOYED                 0.0
DAYS_REGISTRATION             0.0
DAYS_ID_PUBLISH               0.0
OCCUPATION_TYPE               0.0
CNT_FAM_MEMBERS               0.0
REGION_RATING_CLIENT          0.0
REGION_RATING_CLIENT_W_CITY   0.0
WEEKDAY_APPR_PROCESS_START    0.0
HOUR_APPR_PROCESS_START       0.0
REG_REGION_NOT_LIVE_REGION    0.0
REG_REGION_NOT_WORK_REGION    0.0
LIVE_REGION_NOT_WORK_REGION   0.0
REG_CITY_NOT_LIVE_CITY        0.0
REG_CITY_NOT_WORK_CITY        0.0
LIVE_CITY_NOT_WORK_CITY       0.0
ORGANIZATION_TYPE             0.0
OBS_30_CNT_SOCIAL_CIRCLE      0.0
DEF_30_CNT_SOCIAL_CIRCLE      0.0
OBS_60_CNT_SOCIAL_CIRCLE      0.0
DEF_60_CNT_SOCIAL_CIRCLE      0.0
DAYS_LAST_PHONE_CHANGE        0.0
FLAG_DOCUMENT_3               0.0
AMT_REQ_CREDIT_BUREAU_HOUR    0.0
AMT_REQ_CREDIT_BUREAU_DAY     0.0
AMT_REQ_CREDIT_BUREAU_WEEK    0.0
AMT_REQ_CREDIT_BUREAU_MON     0.0
AMT_REQ_CREDIT_BUREAU_QRT     0.0
AMT_REQ_CREDIT_BUREAU_YEAR    0.0
Age_years                     0.0
Age_Group                     0.0
AMT_INCOME_TOTAL_RANGE        0.0
AMT_CREDIT_RANGE              0.0
employed_years                0.0
YEARS_EMPLOYED                0.0
dtype: float64
```

In [408…

```
check_null_values(prevdf)
```

Out[408]:
```
SK_ID_PREV                0.0
SK_ID_CURR                0.0
NAME_CONTRACT_TYPE        0.0
AMT_ANNUITY               0.0
AMT_APPLICATION           0.0
AMT_CREDIT                0.0
AMT_GOODS_PRICE           0.0
NAME_CASH_LOAN_PURPOSE    0.0
NAME_CONTRACT_STATUS      0.0
DAYS_DECISION             0.0
NAME_PAYMENT_TYPE         0.0
CODE_REJECT_REASON        0.0
NAME_CLIENT_TYPE          0.0
NAME_GOODS_CATEGORY       0.0
NAME_PORTFOLIO            0.0
NAME_PRODUCT_TYPE         0.0
CHANNEL_TYPE              0.0
SELLERPLACE_AREA          0.0
NAME_SELLER_INDUSTRY      0.0
CNT_PAYMENT               0.0
NAME_YIELD_GROUP          0.0
PRODUCT_COMBINATION       0.0
dtype: float64
```

Inferences:

Both appdf and prevdf dataframes are cleaned now.

FINDING OUTLIERS

Finding the outliers for the columns in application_data dataset.

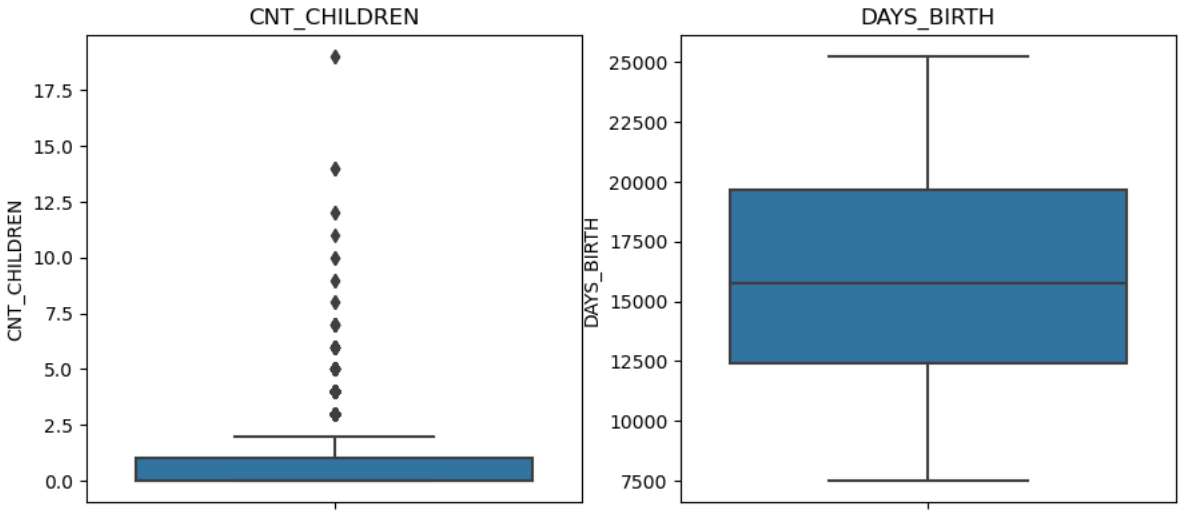
In [2]:
```
plt.figure(figsize=(22,10))

appdf_outliers = ['AMT_INCOME_TOTAL','AMT_ANNUITY','AMT_GOODS_PRICE','AMT_CREDIT','
for i in enumerate(appdf_outliers):
    plt.subplot(2,4,i[0]+1)
    sns.boxplot(y=appdf[i[1]])
    plt.title(i[1])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 1
----> 1 plt.figure(figsize=(22,10))
      3 appdf_outliers = ['AMT_INCOME_TOTAL','AMT_ANNUITY','AMT_GOODS_PRICE','AMT_
CREDIT','DAYS_EMPLOYED']
      4 for i in enumerate(appdf_outliers):

NameError: name 'plt' is not defined
```

INFERENCES:

1. AMT_INCOME_TOTAL has some outliers which shows that there are some applicants with very high income
2. AMT_ANNUITY, AMT_GOODS_PRICE, AMT_CREDIT have very high number of outliers
3. DAYS_EMPLOYED column has less number of outliers with value around 350000 days which is impossible hence it is incorrect entry

In [410…
```
plt.figure(figsize=(22,10))
appdf_outliers_2 = ['CNT_CHILDREN','DAYS_BIRTH']
for i in enumerate(appdf_outliers_2):
    plt.subplot(2,4,i[0]+6)
    sns.boxplot(y=appdf[i[1]])
    plt.title(i[1])
```



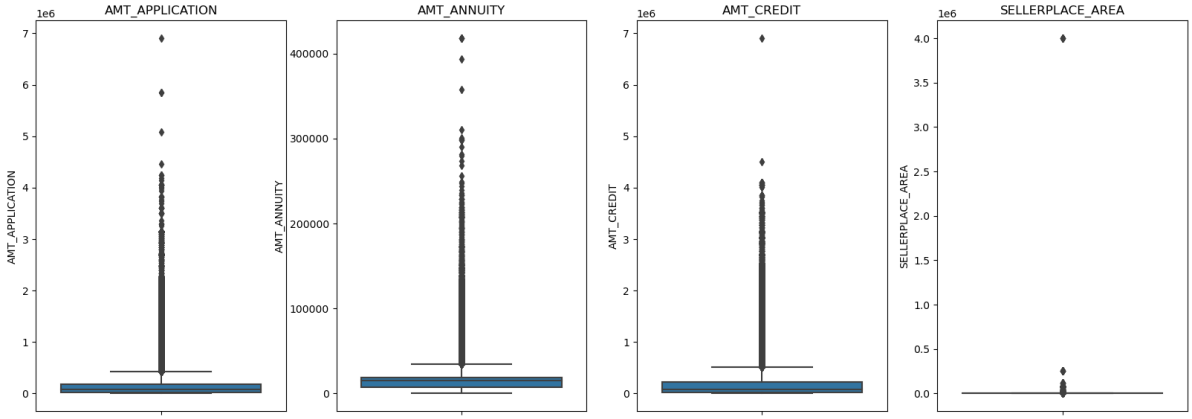
Inferences:

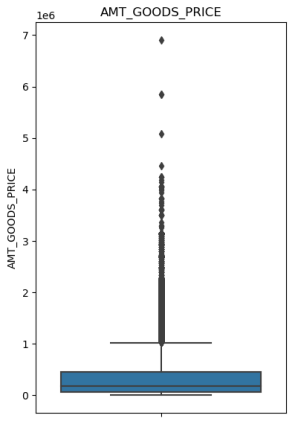
CNT_CHILDREN have few outliers with value almost 20.

DAY_BIRTH have no outliers.

Finding the outliers for the columns in previous application dataset

In [411…
```
plt.figure(figsize=(20,15))

prevdf_outliers = ['AMT_APPLICATION','AMT_ANNUITY','AMT_CREDIT','SELLERPLACE_AREA',
for i in enumerate(prevdf_outliers):
    plt.subplot(2,4,i[0]+1)
    sns.boxplot(y=prevdf[i[1]])
    plt.title(i[1])
```



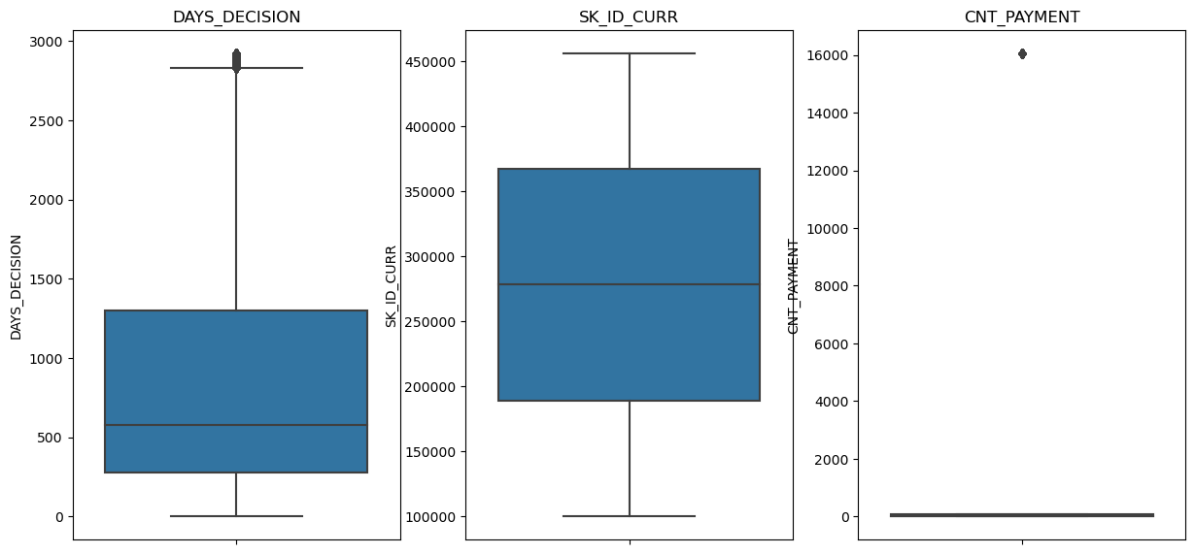


Inferences:

AMT_APPLICATION, AMT_ANNUITY, AMT_CREDIT, SELLERPLACE_AREA and AMT_GOODS_PRICE, have huge number of outliers.

In [412…

```
plt.figure(figsize=(20,15))
prevdf_outliers_1 = ['DAYS_DECISION','SK_ID_CURR','CNT_PAYMENT']

for i in enumerate(prevdf_outliers_1):
    plt.subplot(2,4,i[0]+6)
    sns.boxplot(y=prevdf[i[1]])
    plt.title(i[1])
```



Inferences:

DAYS_DECISION and CNT_PAYMENT have very few ouliers

SK_ID_CURR has no outliers since it is an ID column so it is understood

Logistic Regression Prediction

In [413…
```
appdf.head()
```

Out[413]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_RE
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 × 52 columns

In [414…
```
appdf.columns
```

Out[414]:
```
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', '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', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', '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_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_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', 'Age_years',
       'Age_Group', 'AMT_INCOME_TOTAL_RANGE', 'AMT_CREDIT_RANGE',
       'employed_years', 'YEARS_EMPLOYED'],
      dtype='object')
```

In [415…
```
appdf['TARGET'].describe()
# it has min=0 & max=1, seems like binary column
```

Out[415]:
```
count    307511.000000
mean          0.080729
std           0.272419
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           1.000000
Name: TARGET, dtype: float64
```

In [416…
```
appdf['TARGET'].unique()
```

Out[416]:
```
array([1, 0], dtype=int64)
```

In [417…
```
for i in appdf.columns:
    print(i, appdf[i].dtype)
```

```
SK_ID_CURR int64
TARGET int64
NAME_CONTRACT_TYPE object
CODE_GENDER object
FLAG_OWN_CAR object
FLAG_OWN_REALTY object
CNT_CHILDREN int64
AMT_INCOME_TOTAL float64
AMT_CREDIT float64
AMT_ANNUITY float64
AMT_GOODS_PRICE float64
NAME_TYPE_SUITE object
NAME_INCOME_TYPE object
NAME_EDUCATION_TYPE object
NAME_FAMILY_STATUS object
NAME_HOUSING_TYPE object
REGION_POPULATION_RELATIVE float64
DAYS_BIRTH int64
DAYS_EMPLOYED int64
DAYS_REGISTRATION float64
DAYS_ID_PUBLISH int64
OCCUPATION_TYPE object
CNT_FAM_MEMBERS float64
REGION_RATING_CLIENT int64
REGION_RATING_CLIENT_W_CITY int64
WEEKDAY_APPR_PROCESS_START object
HOUR_APPR_PROCESS_START int64
REG_REGION_NOT_LIVE_REGION int64
REG_REGION_NOT_WORK_REGION int64
LIVE_REGION_NOT_WORK_REGION int64
REG_CITY_NOT_LIVE_CITY int64
REG_CITY_NOT_WORK_CITY int64
LIVE_CITY_NOT_WORK_CITY int64
ORGANIZATION_TYPE object
OBS_30_CNT_SOCIAL_CIRCLE float64
DEF_30_CNT_SOCIAL_CIRCLE float64
OBS_60_CNT_SOCIAL_CIRCLE float64
DEF_60_CNT_SOCIAL_CIRCLE float64
DAYS_LAST_PHONE_CHANGE float64
FLAG_DOCUMENT_3 int64
AMT_REQ_CREDIT_BUREAU_HOUR float64
AMT_REQ_CREDIT_BUREAU_DAY float64
AMT_REQ_CREDIT_BUREAU_WEEK float64
AMT_REQ_CREDIT_BUREAU_MON float64
AMT_REQ_CREDIT_BUREAU_QRT float64
AMT_REQ_CREDIT_BUREAU_YEAR float64
Age_years int64
Age_Group category
AMT_INCOME_TOTAL_RANGE category
AMT_CREDIT_RANGE category
employed_years int64
YEARS_EMPLOYED category
```

In [418…
```
appdf = appdf.astype({'AMT_CREDIT': int, 'AMT_ANNUITY': int, 'AMT_GOODS_PRICE': int
# changing float types to int
```

In [419…
```
predlist = ['SK_ID_CURR', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_AN
# selecting predictor columns which are with numerical data
```

In [420…
```
x_train, x_test, y_train, y_test = train_test_split(appdf[predlist], appdf['TARGET'
# creating 4-way split for training & testing dataset
```

In [421…
```
x_train.head()
```

Out[421]:

	SK_ID_CURR	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_G
163071	289054	0	1.530	1	7695	
82646	195849	0	0.675	2	10125	
191560	322116	0	1.350	3	10188	
248029	386995	0	1.350	10	31653	
94866	210153	0	2.475	7	38250	

In [422…
```
for co in x_train.columns:
    print(co, '\t\t\t', x_train[co].dtype)
```

```
SK_ID_CURR                      int64
CNT_CHILDREN                    int64
AMT_INCOME_TOTAL                        float64
AMT_CREDIT                      int32
AMT_ANNUITY                     int32
AMT_GOODS_PRICE                         int32
DAYS_EMPLOYED                   int64
AMT_REQ_CREDIT_BUREAU_QRT                       int32
```

In [423…
```
y_train.head()
```

Out[423]:
```
163071    0
82646     1
191560    0
248029    0
94866     0
Name: TARGET, dtype: int64
```

Logistic Regression

In [424…
```
clf = LogisticRegression()
```

In [425…
```
clf.fit(x_train, y_train)
```

Out[425]:
```
▾ LogisticRegression
LogisticRegression()
```

In [426…
```
predictions = clf.predict(x_test)
```

In [427…
```
accuracy_score(y_test, predictions)
```

Out[427]:
```
0.9215660770031218
```

In [428…

```
mod1 = sm.OLS(y_train, x_train).fit()
print(mod1.summary())
```

```
                                 OLS Regression Results
================================================================================
=====
Dep. Variable:                 TARGET   R-squared (uncentered):
0.077
Model:                            OLS   Adj. R-squared (uncentered):
0.077
Method:                 Least Squares   F-statistic:
3133.
Date:                Wed, 27 Dec 2023   Prob (F-statistic):
0.00
Time:                        14:35:32   Log-Likelihood:                         -3
6227.
No. Observations:              299823   AIC:                                 7.24
7e+04
Df Residuals:                  299815   BIC:                                 7.25
6e+04
Df Model:                           8
Covariance Type:            nonrobust
================================================================================
===========
                                 coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----------
SK_ID_CURR                 1.996e-07   3.25e-09     61.414      0.000    1.93e-07
2.06e-07
CNT_CHILDREN                  0.0105      0.001     15.024      0.000       0.009
0.012
AMT_INCOME_TOTAL              0.0010      0.000      4.902      0.000       0.001
0.001
AMT_CREDIT                    0.0128      0.001     18.105      0.000       0.011
0.014
AMT_ANNUITY                1.681e-06    5.2e-08     32.355      0.000    1.58e-06
1.78e-06
AMT_GOODS_PRICE           -1.901e-07   7.88e-09    -24.120      0.000   -2.06e-07
-1.75e-07
DAYS_EMPLOYED             -4.057e-08   3.61e-09    -11.236      0.000   -4.76e-08
-3.35e-08
AMT_REQ_CREDIT_BUREAU_QRT     0.0013      0.001      1.965      0.049    3.23e-06
0.003
==============================================================================
Omnibus:                   178703.327   Durbin-Watson:                   1.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1118723.400
Skew:                           3.020   Prob(JB):                         0.00
Kurtosis:                      10.284   Cond. No.                     9.92e+05
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain
a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
[3] The condition number is large, 9.92e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Multiple Regression

In [429…
```
df = appdf[['SK_ID_CURR', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_AN
```

In [430…
```
x_train, x_test, y_train, y_test = train_test_split(df.drop('TARGET', axis = 1),df.
```

In [431…
```
x_train = sm.add_constant(x_train, prepend = False)
```

In [432…
```
mod1 = sm.OLS(y_train,x_train).fit()
```

In [433…
```
print(mod1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 TARGET   R-squared:                       0.007
Model:                            OLS   Adj. R-squared:                  0.007
Method:                 Least Squares   F-statistic:                     227.7
Date:                Wed, 27 Dec 2023   Prob (F-statistic):               0.00
Time:                        14:35:33   Log-Likelihood:                -28572.
No. Observations:              246008   AIC:                         5.716e+04
Df Residuals:                  245999   BIC:                         5.725e+04
Df Model:                           8
Covariance Type:            nonrobust
================================================================================
===========
                                 coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----------
SK_ID_CURR                -5.522e-09   5.33e-09     -1.036      0.300    -1.6e-08
4.93e-09
CNT_CHILDREN                  0.0020      0.001      2.561      0.010       0.000
0.004
AMT_INCOME_TOTAL             -0.0050      0.001     -8.692      0.000      -0.006
-0.004
AMT_CREDIT                    0.0188      0.001     23.964      0.000       0.017
0.020
AMT_ANNUITY                8.066e-07   6.21e-08     12.983      0.000    6.85e-07
9.28e-07
AMT_GOODS_PRICE            -2.54e-07   8.75e-09    -29.033      0.000   -2.71e-07
-2.37e-07
DAYS_EMPLOYED             -9.362e-08   4.09e-09    -22.865      0.000   -1.02e-07
-8.56e-08
AMT_REQ_CREDIT_BUREAU_QRT    -0.0012      0.001     -1.699      0.089      -0.003
0.000
const                         0.1078      0.002     52.582      0.000       0.104
0.112
==============================================================================
Omnibus:                   147387.557   Durbin-Watson:                   2.001
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           928372.753
Skew:                           3.038   Prob(JB):                         0.00
Kurtosis:                      10.324   Cond. No.                     2.63e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
[2] The condition number is large, 2.63e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

DECISION TREE

In [434…
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

In [435…
```
df = appdf[['SK_ID_CURR', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_AN
```

In [436…
```
x_train, x_test, y_train, y_test = train_test_split(df.drop('TARGET', axis = 1),df.
```

In [437…
```
DTmod = DecisionTreeClassifier(max_depth=4).fit(x_train, y_train)
```

In [438…
```
y_pred = DTmod.predict(x_test)
```

In [439…
```
from sklearn.metrics import confusion_matrix
```

In [440…
```
clf = DecisionTreeClassifier()
```

In [441…
```
clf.fit(x_train, y_train)
```

Out[441]:
▾ DecisionTreeClassifier

DecisionTreeClassifier()

In [442…
```
predictions = clf.predict(x_test)
```

In [443…
```
accuracy_score(y_test, predictions)
```

Out[443]:
```
0.847519633188625
```

In [444…
```
print(confusion_matrix(y_test,y_pred))
```

```
[[56574     0]
 [ 4929     0]]
```

Random Forest

In [445…
```
from sklearn.ensemble import RandomForestClassifier
```

In [446…
```
df = appdf[['SK_ID_CURR', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_AN
```

In [447…
```
x_train, x_test, y_train, y_test = train_test_split(df.drop('TARGET', axis = 1),df.
```

In [448…
```
RFmod = RandomForestClassifier(n_estimators=50).fit(x_train, y_train)
```

In [449…
```
y_pred = DTmod.predict(x_test)
```

In [450…
```
clf = RandomForestClassifier()
```

In [451…
```
clf.fit(x_train, y_train)
```

Out[451]:
▾ RandomForestClassifier

RandomForestClassifier()

In [452…
```
predictions = clf.predict(x_test)
```

In [453…
```
accuracy_score(y_test, predictions)
```

Out[453]:
```
0.9176137749378079
```

In [454…
```
y_pred = RFmod.predict(x_test)
```

In [455…
```
print(confusion_matrix(y_test,y_pred))
```

```
[[56423    60]
 [ 5007    13]]
```

Naive Bayes Classifier

In [456…
```
from sklearn.naive_bayes import GaussianNB
```

In [457…
```
x_train, x_test, y_train, y_test = train_test_split(df.drop('TARGET', axis = 1),df.
```

In [458…
```
NBmod = GaussianNB().fit(x_train, y_train)
```

In [459…
```
y_pred = NBmod.predict(x_test)
```

In [460…
```
print(confusion_matrix(y_test,y_pred))
```

```
[[56482     1]
 [ 5020     0]]
```

In [461…
```
from sklearn.metrics import accuracy_score
```

In [462…
```
accuracy_score(y_test,y_pred)
#91%
```

Out[462]:
```
0.9183617059330439
```

In [463…
```
total_predictions = len(y_test)
correct_predictions = sum(y_test == y_pred)

accuracy_manual = correct_predictions / total_predictions
accuracy_manual
```

Out[463]:
```
0.9183617059330439
```

KNN

In [464…
```
from sklearn.neighbors import KNeighborsClassifier
```

In [465…
```
x_train, x_test, y_train, y_test = train_test_split(df.drop('TARGET', axis = 1),df.
```

In [466…
```
KNNmod = KNeighborsClassifier(n_neighbors=5).fit(x_train, y_train)
```

In [467…
```
y_pred = KNNmod.predict(x_test)
```

In [468…
```
print(confusion_matrix(y_test,y_pred))
```

```
[[56158   444]
 [ 4840    61]]
```

In [469…
```
accuracy_score(y_test,y_pred)
```

Out[469]:
```
0.9140854917646294
```

K-Means

In [470…
```
from sklearn.cluster import KMeans
```

In [471…
```
KMmod = KMeans(n_clusters=3).fit(df)
```

In [472…
```
KMmod.cluster_centers_
```

Out[472]:
```
array([[2.78174354e+05, 4.14604166e-01, 1.89665425e+00, 7.80240967e+00,
        3.43893649e+04, 7.38737587e+05, 6.26876976e+04, 2.41165853e-01,
        7.26775412e-02],
       [2.78195868e+05, 4.20051313e-01, 1.45280986e+00, 2.85907875e+00,
        1.97014329e+04, 2.96809964e+05, 7.25389682e+04, 2.17418247e-01,
        9.01732910e-02],
       [2.78112313e+05, 4.07099880e-01, 2.42262567e+00, 1.38370638e+01,
        4.83218740e+04, 1.32714481e+06, 5.47643541e+04, 2.65884477e-01,
        5.03309266e-02]])
```

In [473…
```
KMmod.labels_
```

Out[473]:
```
array([1, 2, 1, ..., 0, 1, 0])
```

In [474…
```
KMmod.inertia_
```

Out[474]:
```
1.620972295901575e+16
```

In [ ]:
```

```

UNIVARIATE ANALYSIS

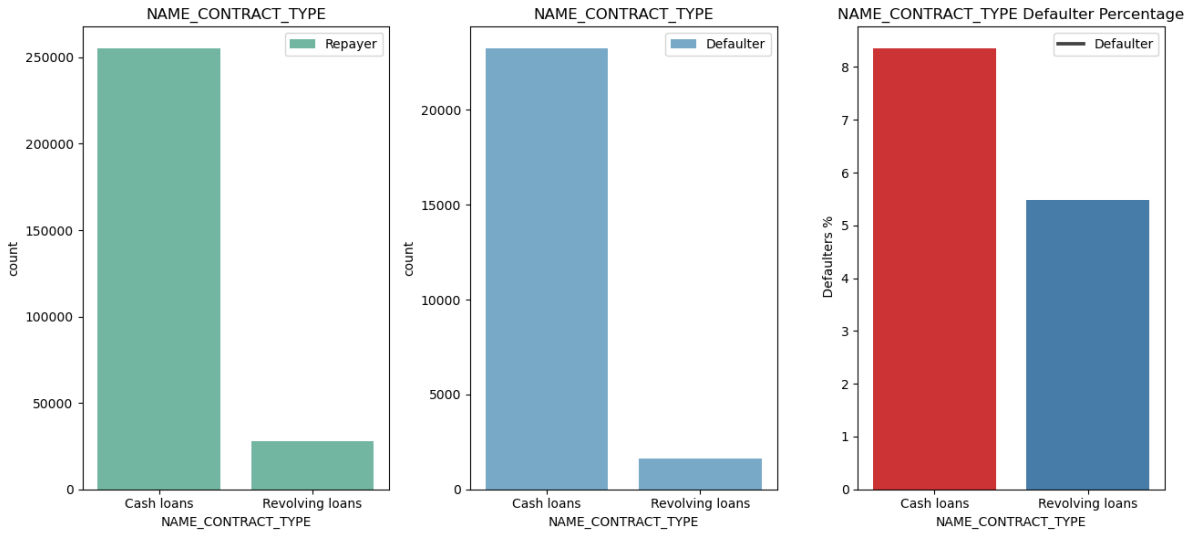
In [476…
```
appdf_0=appdf[appdf['TARGET']==0]
appdf_1=appdf[appdf['TARGET']==1]

appdf_0['TARGET'].replace({0:'Repayer',1:'Defaulter'},inplace=True)
appdf_1['TARGET'].replace({0:'Repayer',1:'Defaulter'},inplace=True)
clients=['Repayer','Defaulter']
```

In [477…
```
def uni_cat(attribute,label_rotation=False,horizontal_layout=True): # label_rotatio
                                                                    # Horizontal_La
                                                                    # made true to

    if horizontal_layout:
        fig, (ax1, ax2, ax3) = plt.subplots(ncols=3, figsize=(13,6))
    else:
        fig, (ax1, ax2, ax3) = plt.subplots(nrows=3, figsize=(20,18))

    #Using groupby function to get the percentage of defaulters the selected catego
    temp = appdf[[attribute, 'TARGET']].groupby(attribute).mean().reset_index()
    temp["TARGET"] = temp["TARGET"]*100
    temp.sort_values(by='TARGET', ascending=False, inplace=True) #sorting values ir
```

```
    #### PLOT 1
    s = sns.countplot(ax=ax1, x=attribute, data=appdf_0, hue="TARGET",palette='Set2
    ax1.set_title(attribute)
    ax1.legend(['Repayer'])

    if label_rotation:
        s.set_xticklabels(s.get_xticklabels(),rotation=90) #label_rotation paramete

    #### PLOT 2
    s = sns.countplot(ax=ax2, x=attribute, data=appdf_1,hue='TARGET',palette='Blues
    plt.ylabel(' Defaulters %', fontsize=10)
    ax2.set_title(attribute)
    ax2.legend(['Defaulter'])

    if label_rotation:
        s.set_xticklabels(s.get_xticklabels(),rotation=90) #label_rotation paramete

    #### PLOT 3
    s = sns.barplot(ax=ax3, x=attribute, y='TARGET', data=temp,palette='Set1') #plo
    plt.ylabel(' Defaulters %', fontsize=10)
    ax3.set_title(attribute + " Defaulter Percentage")
    ax3.legend(['Defaulter'])

    if label_rotation:
        s.set_xticklabels(s.get_xticklabels(),rotation=90) #label_rotation paramete


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

In [478…

```
#using fucntion to plot countplots and barplot.
uni_cat('NAME_CONTRACT_TYPE')
```



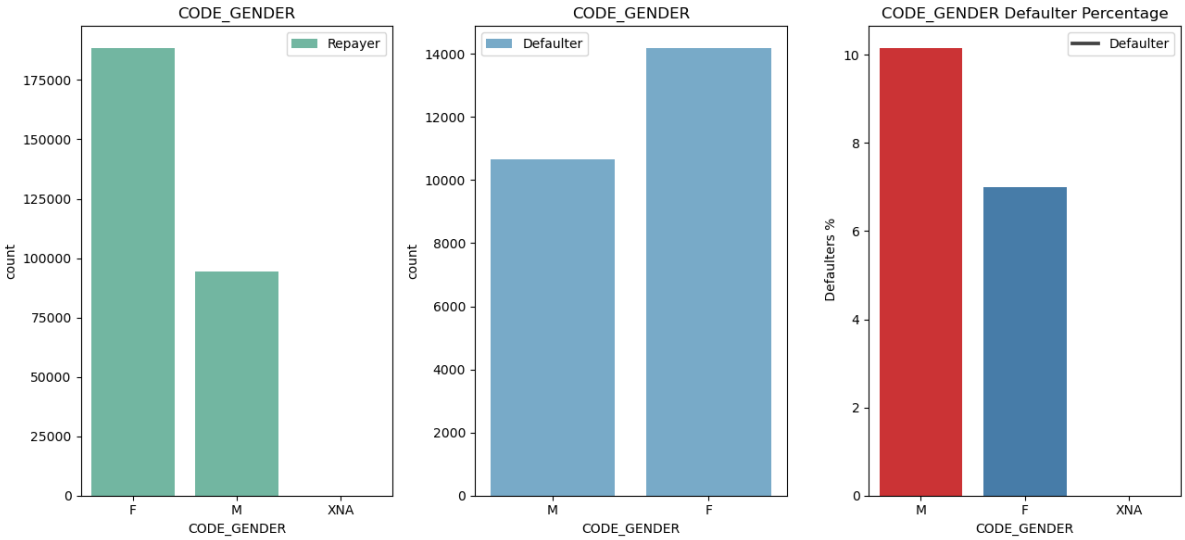
INFERENCES:

1.As we can see their are less amount of Revolving loans but still majority of the loans are not repaid.

2.Large amount of cash loans are seen and also defaulter percenatge is higher for cash loans.

In [479…

```
#using fucntion to plot countplots and barplot.
uni_cat('CODE_GENDER')
```

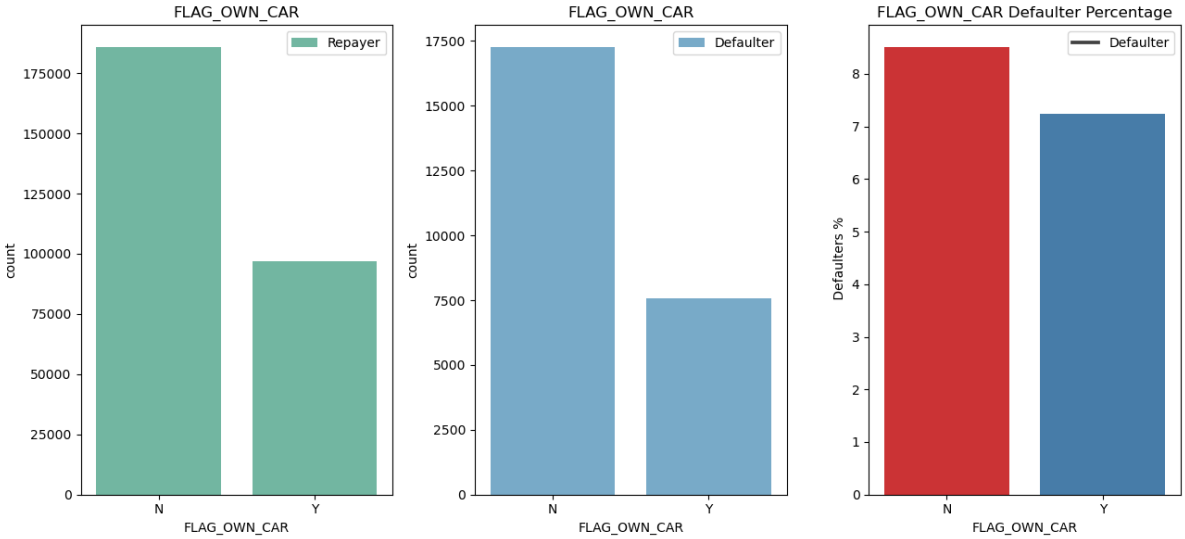


INFERENCES:

1.We can see number of female clients is more than male clients.

2.But in Defaulter % plot we can see that defaulter percentage of male clients (approx-10%) is more than female clients. This means males have less chance of returning their loans.

In [480…
```
#using fucntion to plot countplots and barplot.
uni_cat('FLAG_OWN_CAR')
```

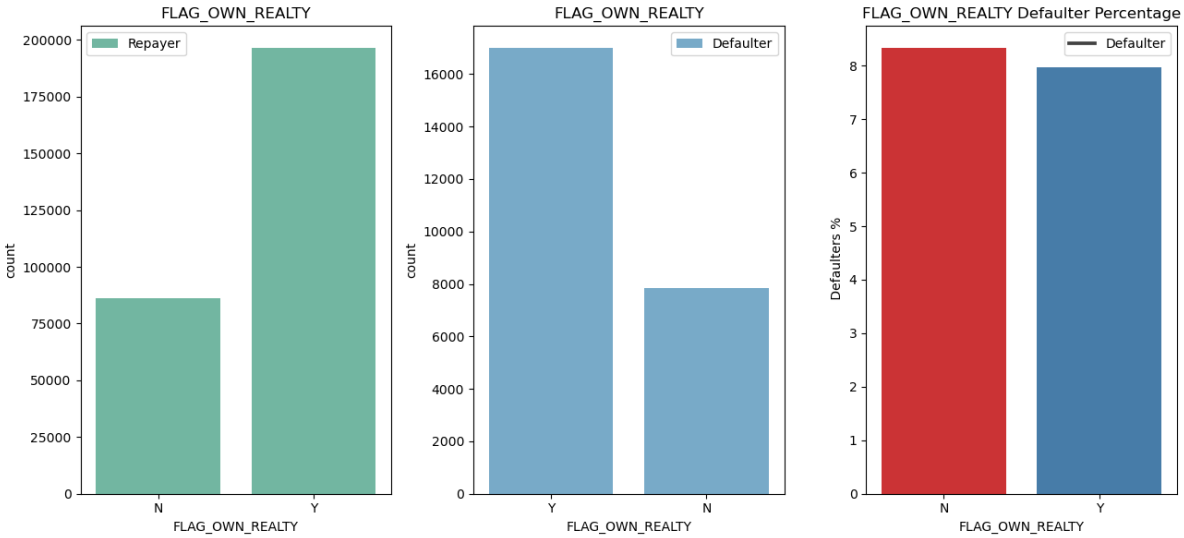


INFERENCES:

1.We can observe that Large Number of clients does not own cars.

2.In Defaulter plot client who does not own car have slightly high chances of not repaying the loan than client who owns the car but still we cant say theirs coorelation bewteen them as they both have almost same percentage of defaulters.

In [481…
```
#using fucntion to plot countplots and barplot.
uni_cat('FLAG_OWN_REALTY')
```

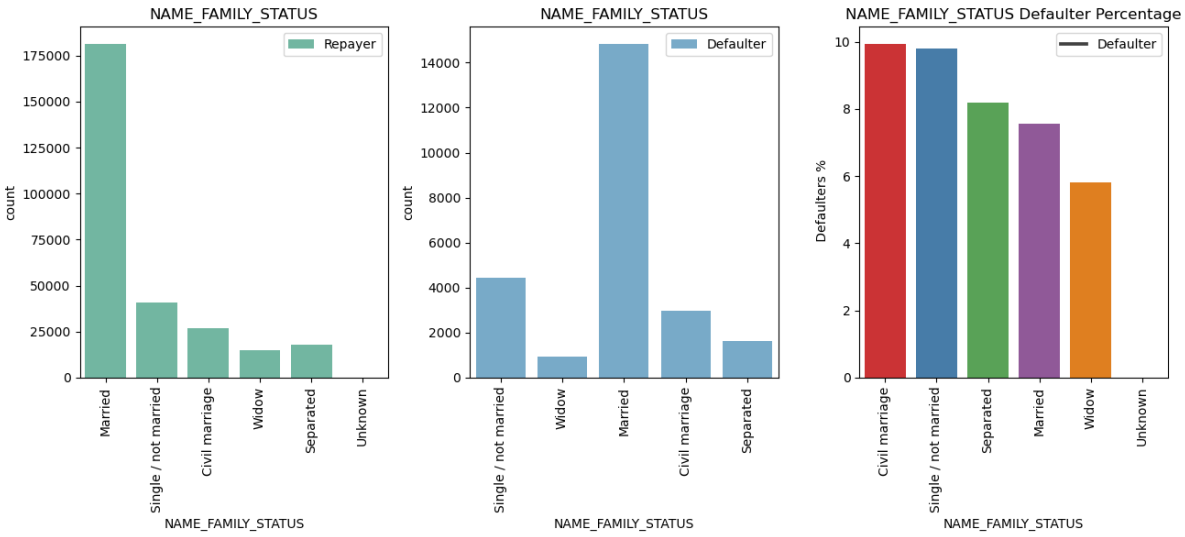


INFERENCES:

1.Here we can see clients who own real estate is more than half of the clients who dont own real estate.This means that majority of the clients own real etsate.

2.Their is no correlation seen between the client owning real estate and defaulter of loan as their percentage of defaulters is almost same.

In [482…
```
#using fucntion to plot countplots and barplot.
uni_cat('NAME_FAMILY_STATUS',True,True)
```



INFERENCES:

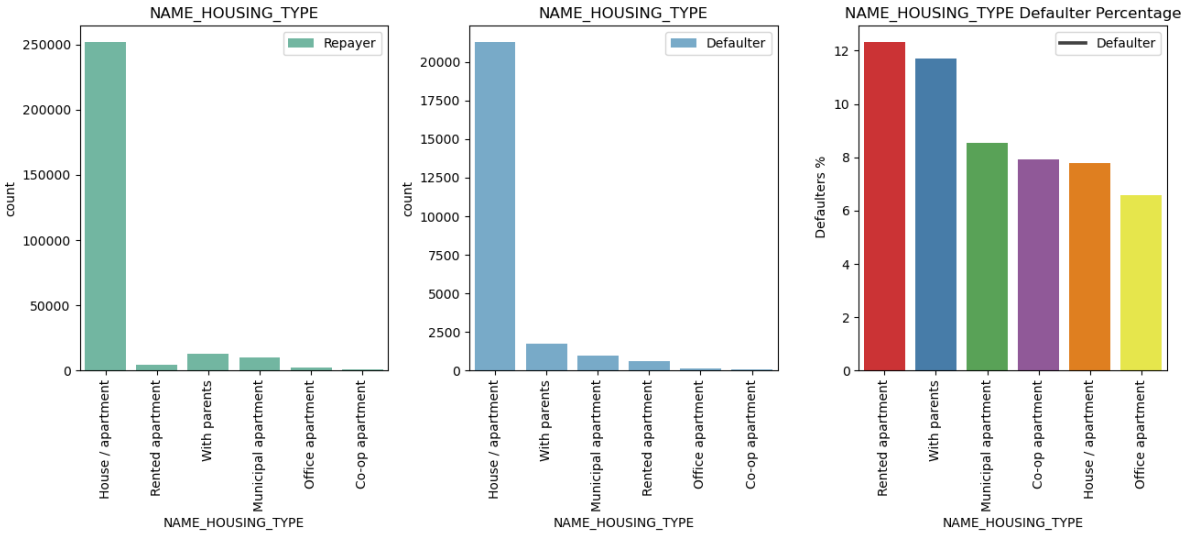
1.Majority of the loans are take by married clients as seen in the plot.

2.Second highest for taking the loans are Single/not Married clients and then Civil Marriages, Seperated and Widow.

3.As seen in the Defaulter % plot Civil marriage (10%) has the highest percenatage of defaulters and then comes Single/not married.

4.Widow has the lowest percentage of defaulter.

In [483…
```
#using fucntion to plot countplots and barplot.
uni_cat('NAME_HOUSING_TYPE',True)
```



INFERENCES:

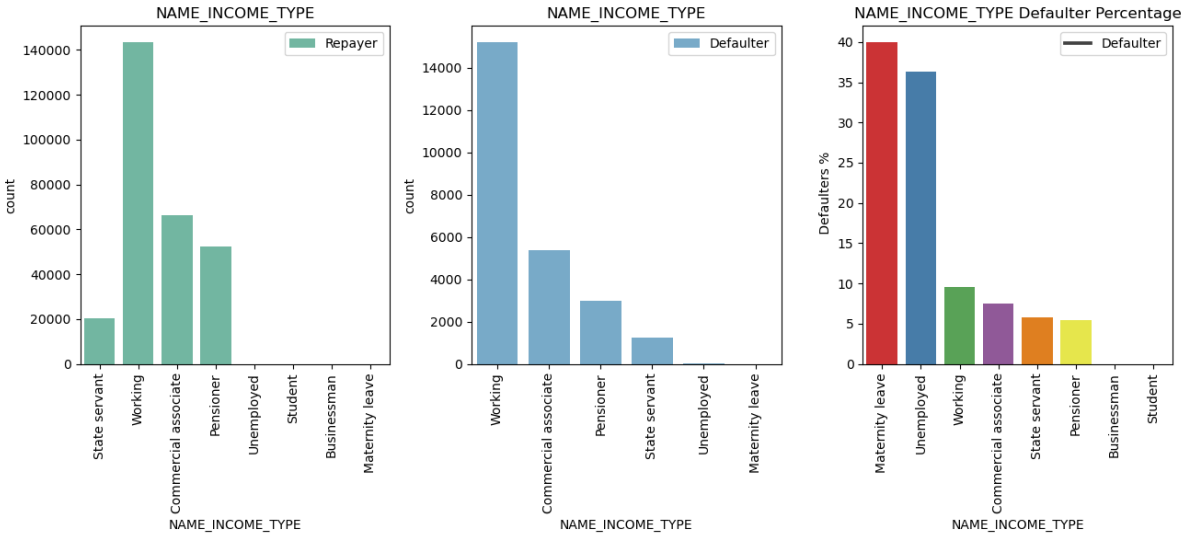
1.From this plot we get to know that majority of the clients stays in their own House/Apartment.

2.Then comes clients who lives with their parents and then comes clients who live in Muncipal apartment.

3.Clients living in Rented apartment(12%) and clients living with parents(approx:11.8) have Highest default rates.

In [484…

```
#using fucntion to plot countplots and barplot.
uni_cat('NAME_INCOME_TYPE',True)
```



INFERENCES:

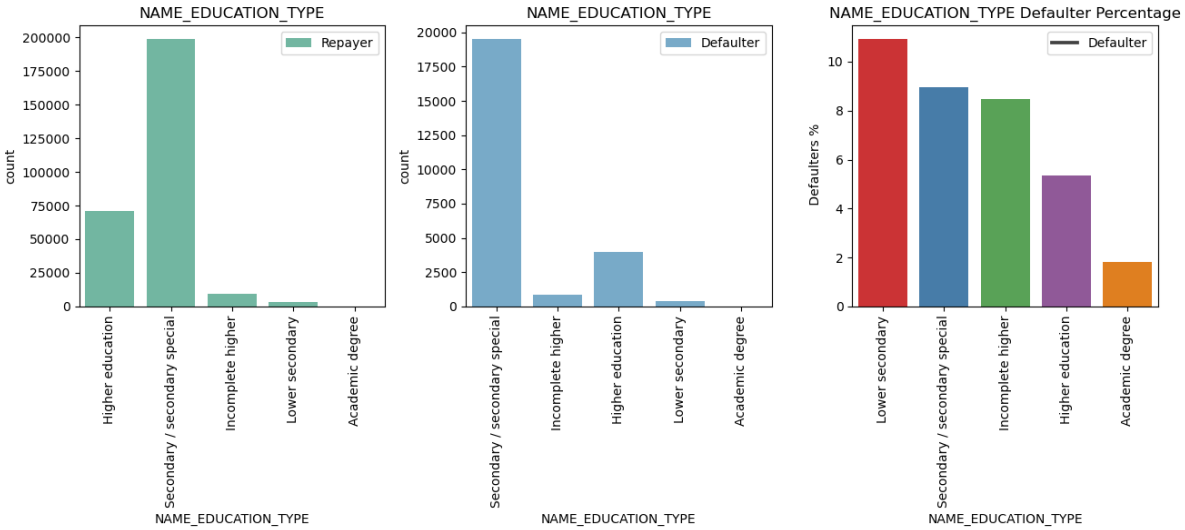
1.Large number of clients who are taking loans are working working clients followed by Commercial associate and pensioner.

2.Clients with the type of Maternity leave (40%) are the clients not repaying the loan.

3.Loan can be provided to students and businessmans as theirs almost no record of defaulters for them.So it could be the most reliable and safest option.

In [485…

```
#using fucntion to plot countplots and barplot.
uni_cat('NAME_EDUCATION_TYPE',True)
```



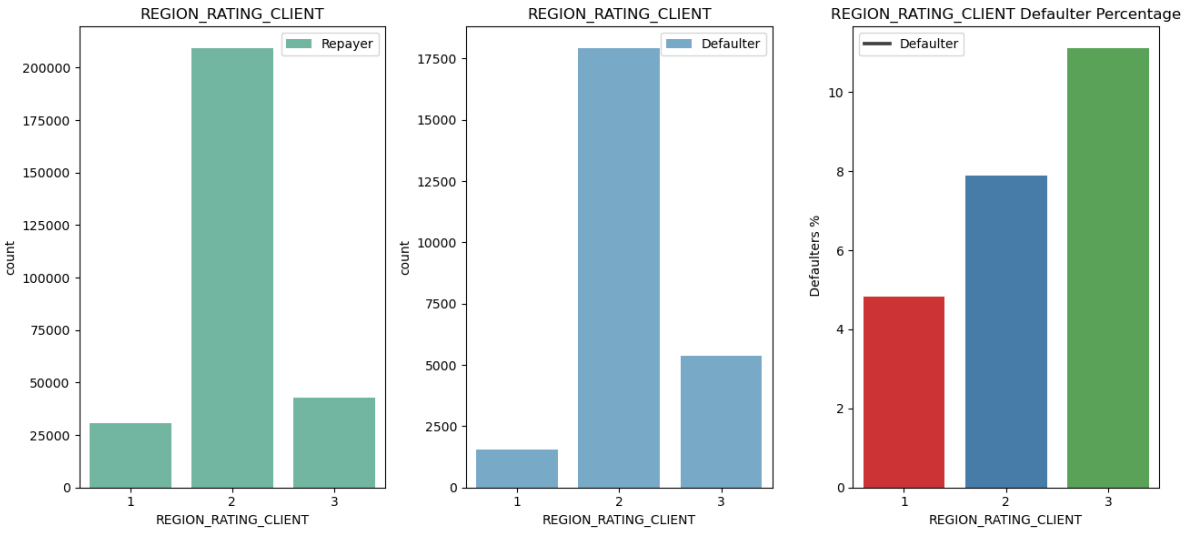
INFERENCES:

1.Majority of the clients have Secondary/Secondary special education then comes Higher education.

2.Defaulter % (approx:11%) of Lower secondary edcuation type is the highest.This means that clients with lower education have higher chances of not repaying the loans.

3.Clients with Academic degree have the lowest defaulter %.

In [486…
```
#using fucntion to plot countplots and barplot.
uni_cat('REGION_RATING_CLIENT')
```

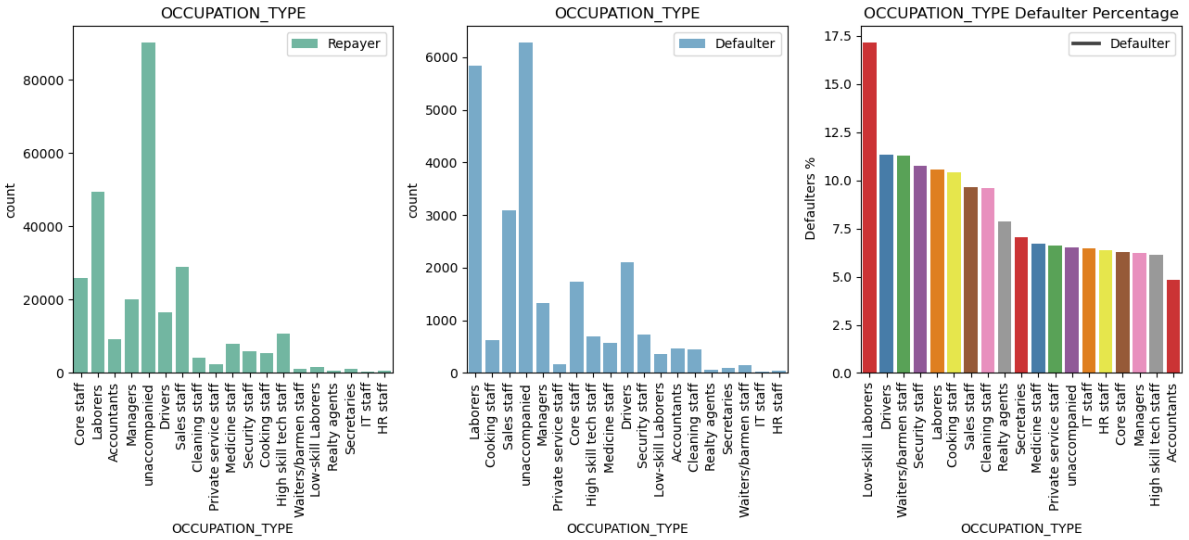


INFERENCES:

1.Most of the clients are living in Region Rating 2 plcaes also we can say tier 2 palces.

2.Highest % of defaulters can be seeon in clients coming from Rating Region 3(Approx: 11%) also we can say tier 3 places.

In [487…
```
#using fucntion to plot countplots and barplot.
uni_cat('OCCUPATION_TYPE',True)
```



INFERENCES:

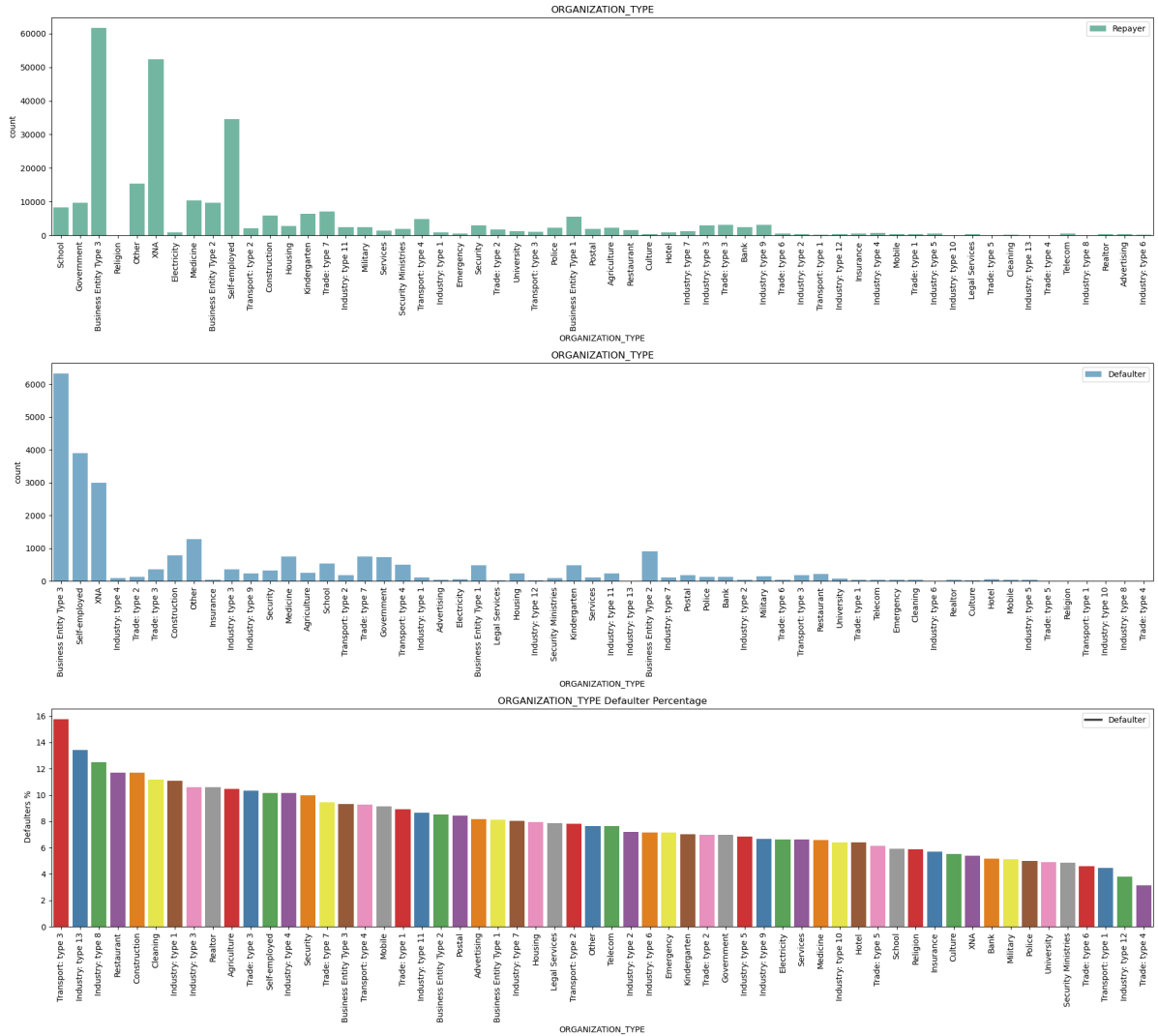
1.Majority of the loans are taken by Laborers followed by Sales Staff , Core Staff,Drivers,Accountants.

2.Though the percentage of loan take by low-skilled laborers are less, but it has the highest percentage of not repaying the loan.

3.Then followed by Drivers, Waiters/barmen staff, security staff, laborers,cooking staff and on.

In [488…

```
#using fucntion to plot countplots and barplot.
uni_cat('ORGANIZATION_TYPE',True,False)
```



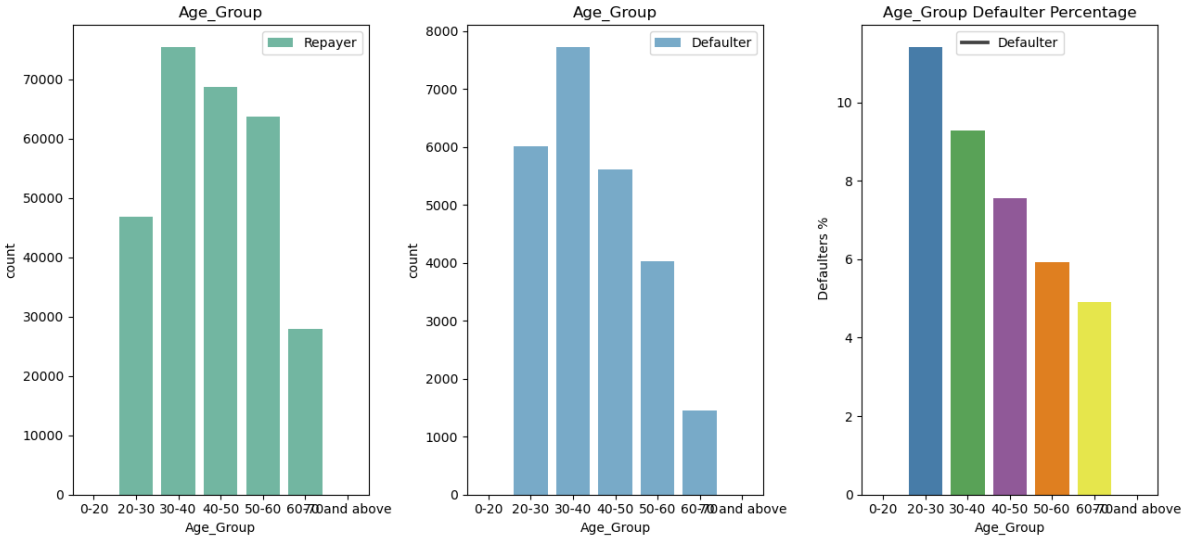
INFERENCES:

1.Most of the applicants are from Business Entity type 3 followed by XNA, self-employed.

2.Organization with higher percentage of defaulters are Transport type 3 (approx:16%) followed by industry type 13 (approx 13%), industry type 8 (approx:12.5%).

3.Trade type 4 and industry type 12 has the lowest defaulter rate, so we could say that loan can be provided to these organization\s without any concern.

In [489…
```
#using fucntion to plot countplots and barplot.
uni_cat('Age_Group')
```

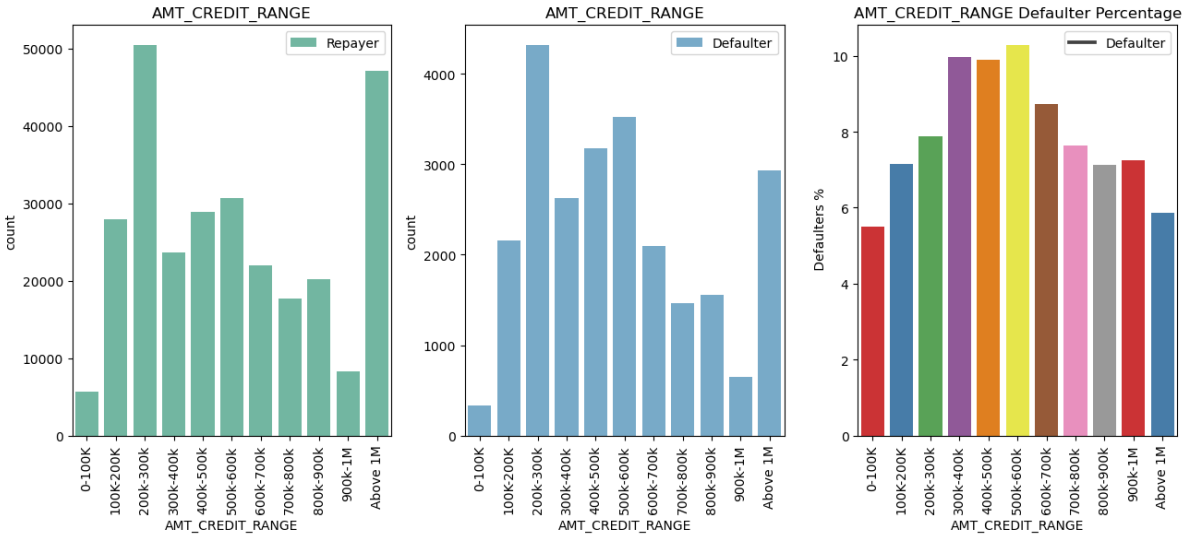


INFERENCES:

1.Clients within age group of 30 to 60 have the highest applicants.

2.Clients with age group of 20-30 have higher chances of not repaying the loan or we can say higher chances of being defaulters.

In [490…
```
#using fucntion to plot countplots and barplot.
uni_cat('AMT_CREDIT_RANGE',True)
```



INFERENCES:

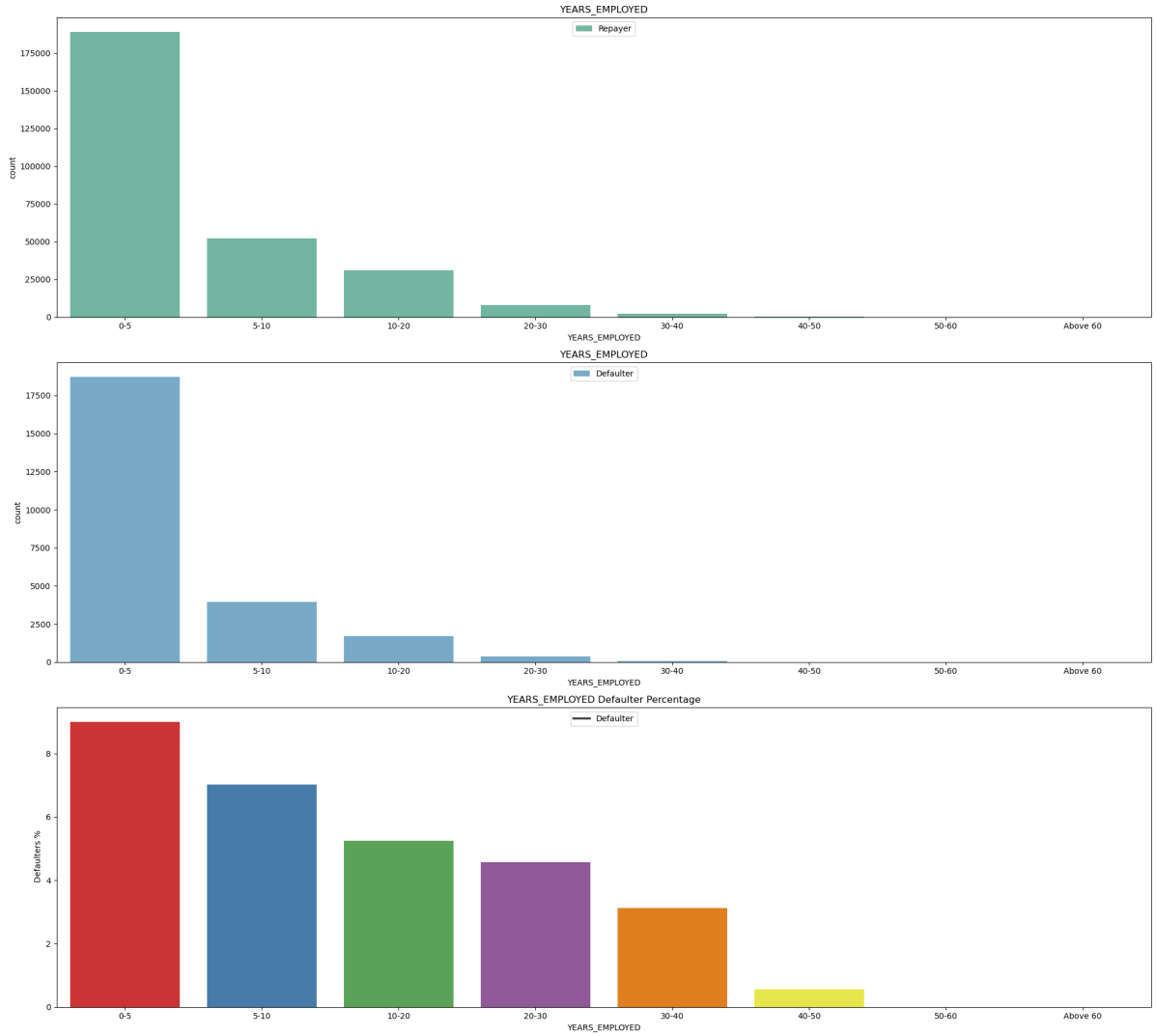
1.Majority of the clients have been granted loan for amount between 200k-300k.

2.very less amounts of clients have beenn credited with the loan amount more than 1M.

3.Mostly loan credited to the clients is less than 900k.

4.Cleints who have been credited with the loan between 300k to 600k has more percentage of defaulter.

In [491…
```
#using fucntion to plot countplots and barplot.
uni_cat('YEARS_EMPLOYED',False,False)
```



INFERENCES:

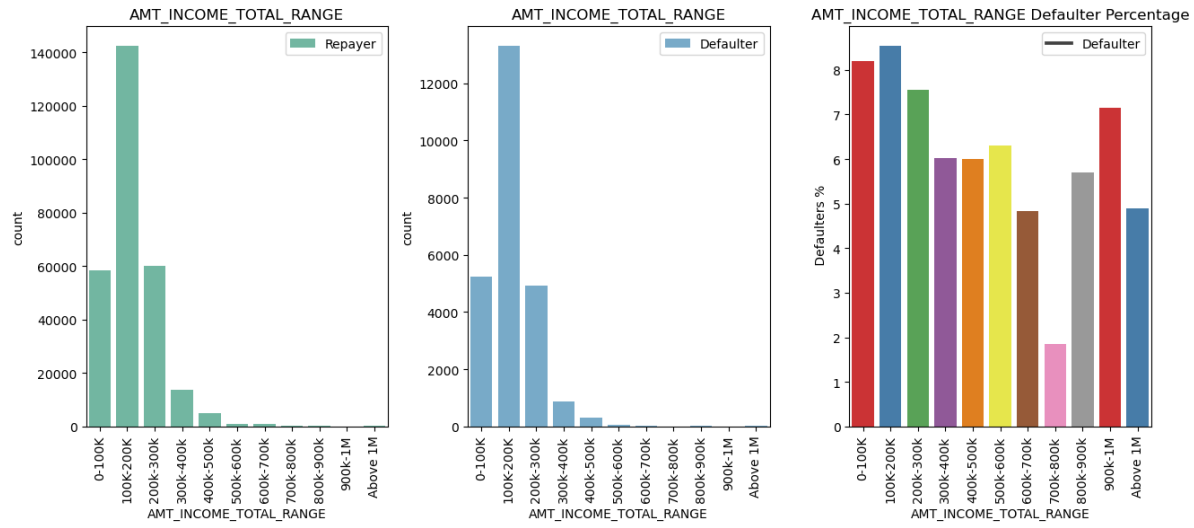
1. Very high number of applicants lies in the range of work experience of 0 to 5 years

2. Default rate for applicants with work experience of 0-5 years is high

3. As the work experience goes on increasing default rate goes down

In [492…

```
#using fucntion to plot countplots and barplot.
uni_cat('AMT_INCOME_TOTAL_RANGE',True)
```

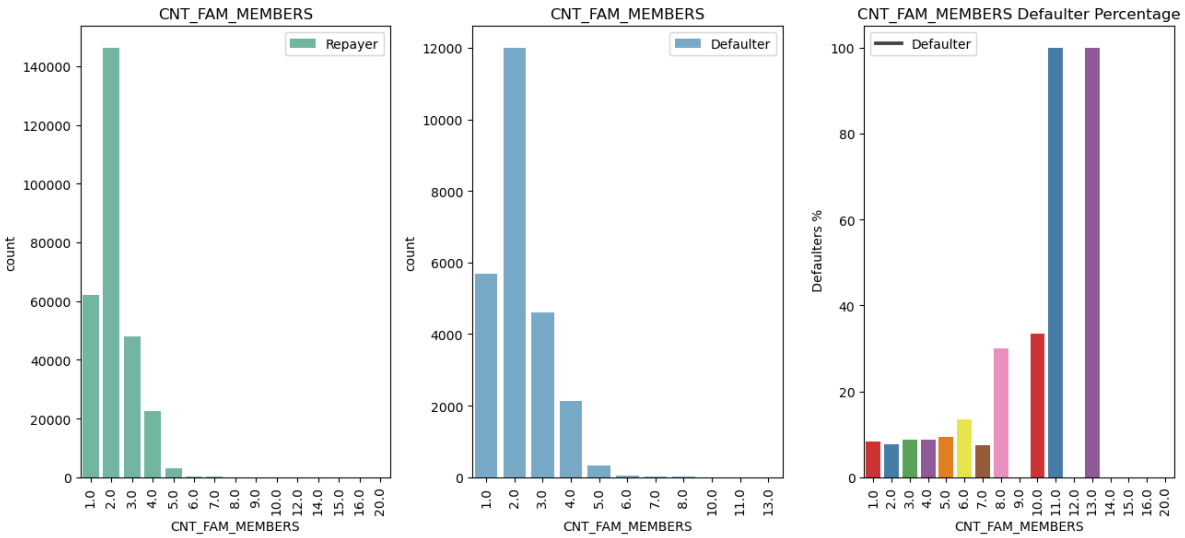


INFERENCES

1.Majority of the clients have the income range bewteen 100k-200k.

2.Highest default rate is also for the clients with income range of 100k-200k followed by the clients with income range of 0-100k.

In [493…
```
#using fucntion to plot countplots and barplot.
uni_cat('CNT_FAM_MEMBERS', True)
```

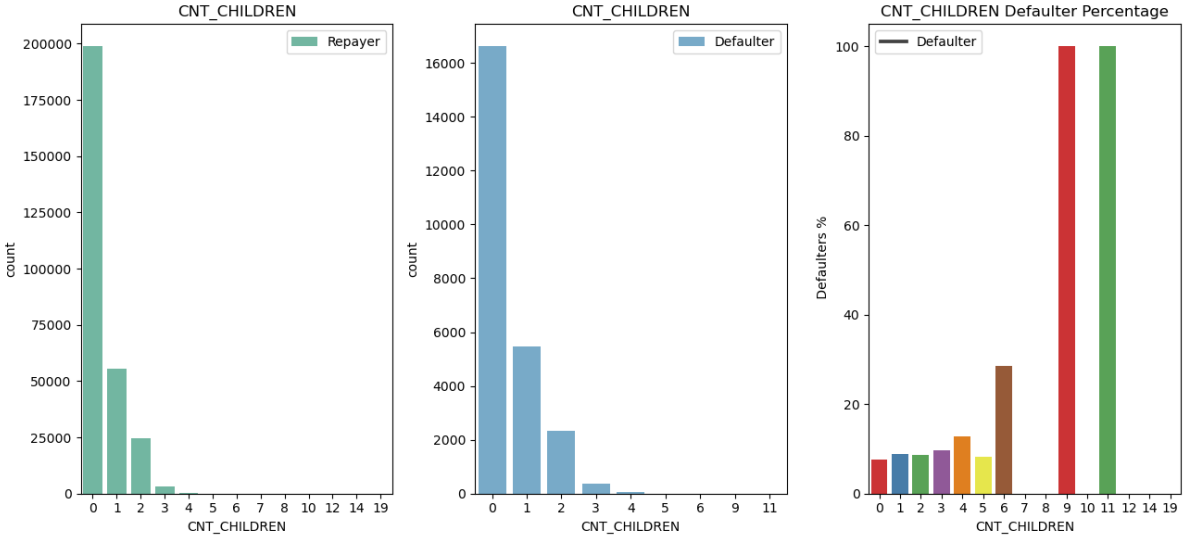


INFERENCES:

1.Majority of the clients are having 2 members in the family followed by 1,3 and 4.

2.We can see clients with members 11 and 13 have 100% of defaulter rate.

In [494…
```
#using fucntion to plot countplots and barplot.
uni_cat('CNT_CHILDREN')
```



INFERENCES:

1.Majority of the clients have no childrens.

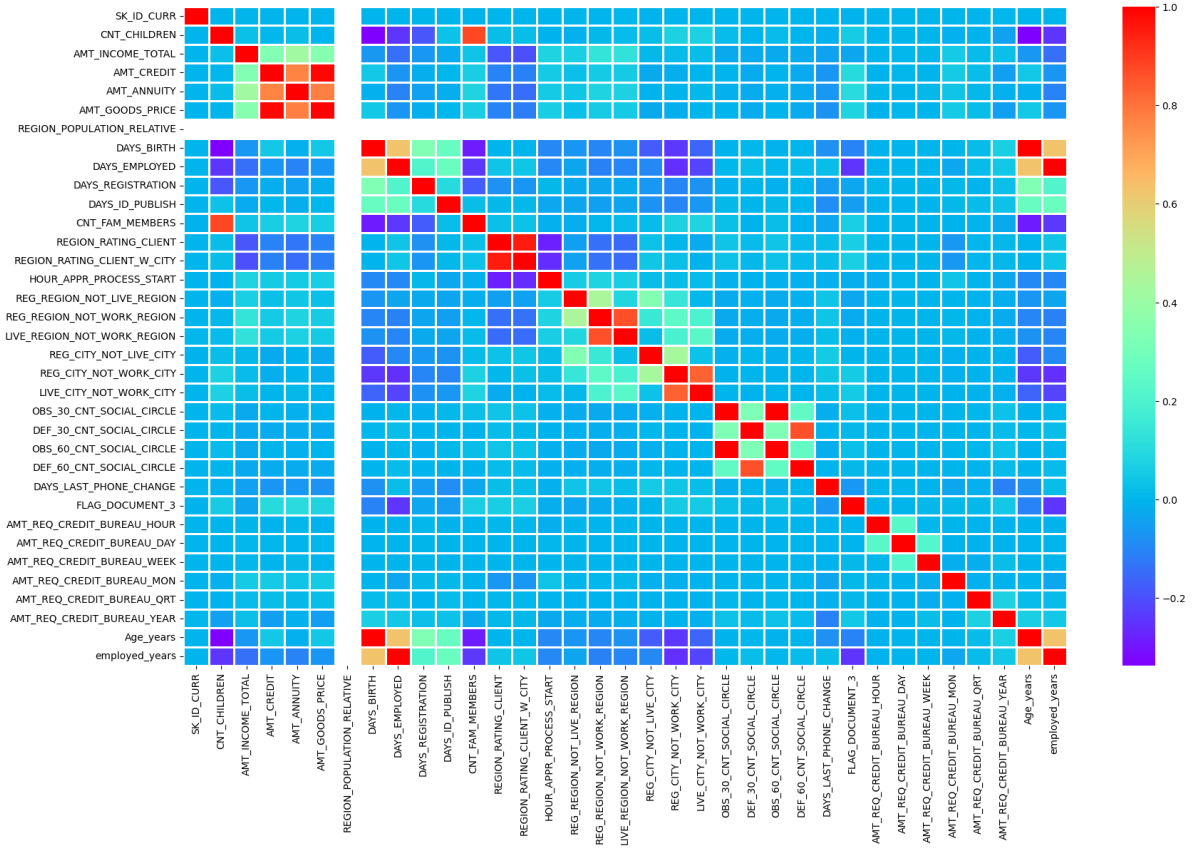
2.Very less clients can be seen who has 3 childrens.

3.Clients with the child count 9 and 11 are showing 100% defaulter rate..

FINDING THE CORRELATION FOR NUMERICAL COLUMNS OF APPLICATION DATA

In [495…

```
# Selecting only numerical columns
numeric_df = appdf_0.select_dtypes(include='number')

# Plotting the correlation heatmap
fig = plt.figure(figsize=(20, 12))
ax = sns.heatmap(numeric_df.corr(), annot=False, linewidth=1, cmap='rainbow')
plt.show()
```



INFERENCES:

AMT_GOODS_PRICE and AMT_CREDIT credit are highly correlated

CNT_FAM_MEMBERS and CNT_CHILDREN are also highly correlated

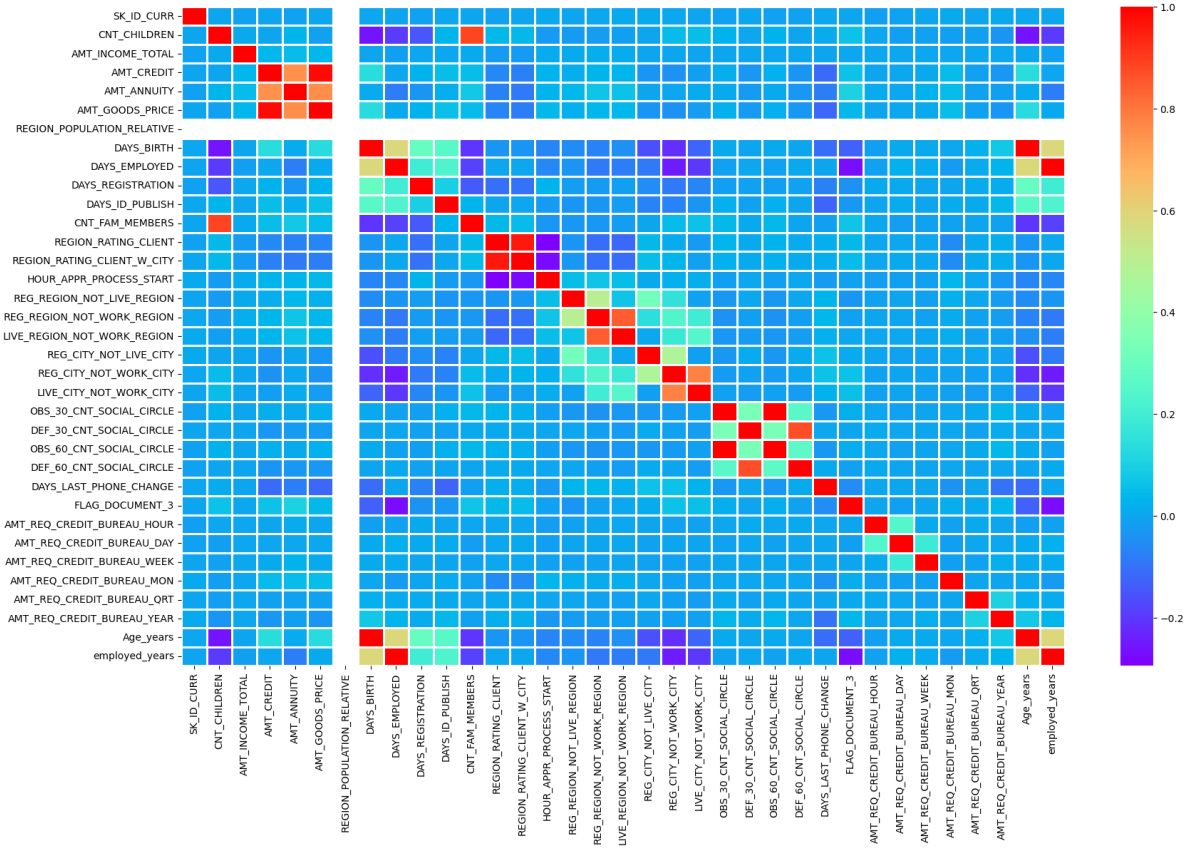
AMT_GOODS_PRICE and AMT_ANNUITY are Moderately correalted

AMT_ANNUITY and AMT_CREDIT are also moderately correlated

DAYS_EMPLOYED and DAYS_BIRTH are also correlated with each other

In [496…

```
import pandas as pd

# Filtering only the numeric columns
numeric_columns_1 = appdf_1.select_dtypes(include='number').columns.tolist()

# Creating a DataFrame with only numeric columns
numeric_df_1 = appdf_1[numeric_columns_1]

# Plotting the correlation heatmap
fig = plt.figure(figsize=(20, 12))
ax = sns.heatmap(numeric_df_1.corr(), annot=False, linewidth=1, cmap='rainbow')
plt.show()
```



INFERENCES:

AMT_GOODS_PRICE AMT_CREDIT are higly correlated

CNT_FAM_MEMBERS and CNT_CHILDREN are moderately correlated

AMT_GOODS_PRICE and AMT_ANNUITY are moderately correlated

AMT_ANNUITY and AMT_CREDIT are also correlated with each other

BIVARIATE ANALYSIS

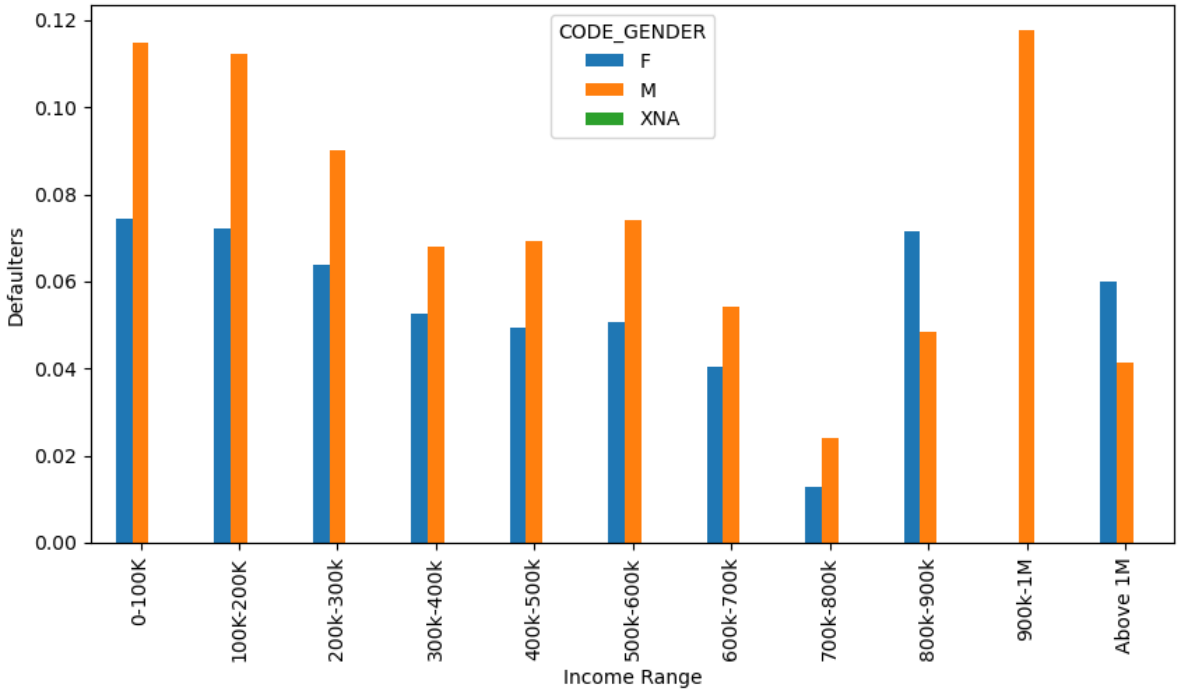
COMPARING INCOME RANGE AND GENDER

In [497…

```
#using pivot table to plot the barplot
appdf.pivot_table(values='TARGET',index='AMT_INCOME_TOTAL_RANGE',columns='CODE_GEND
plt.xlabel('Income Range')
plt.ylabel('Defaulters')
```

Out[497]:

```
Text(0, 0.5, 'Defaulters')
```



INFERENCES

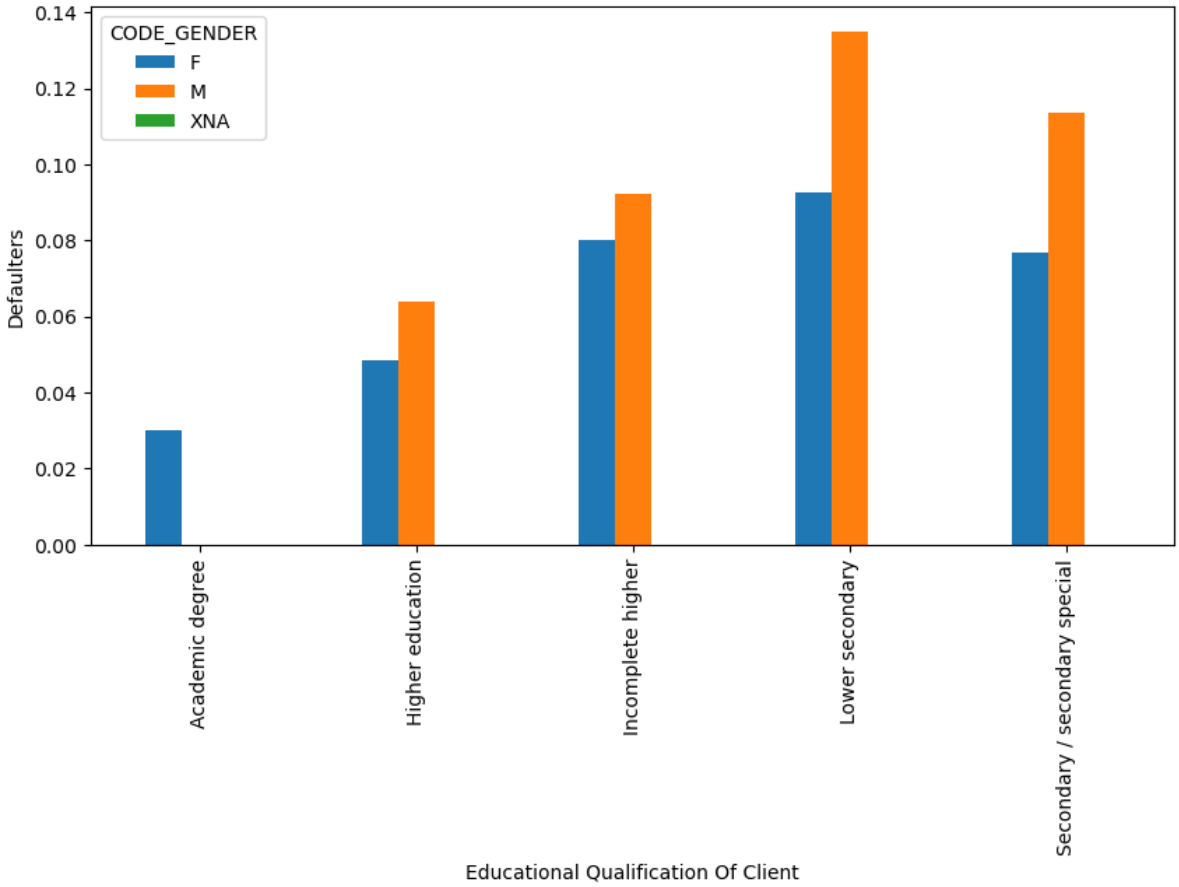
1. Male and female in the income range 0 to 100k and 100k to 200k have equal default rate around 10%

2. Females earning above 1M are more likely to default than males

COMPARING EDUCATION TYPE AND GENDER

In [498…
```
#using pivot table to plot the barplot
appdf.pivot_table(values='TARGET',index='NAME_EDUCATION_TYPE',columns='CODE_GENDER'
plt.xlabel('Educational Qualification Of Client')
plt.ylabel('Defaulters')
```

Out[498]:
```
Text(0, 0.5, 'Defaulters')
```



INFERENCES:

1.Males and female clients with Lower secondary Educational Qualification have higher defaulter rates.

2.Then comes males clients with Secondary/secondary special who have high defaulter rate.

3.People with higher education have low defaulter rate.

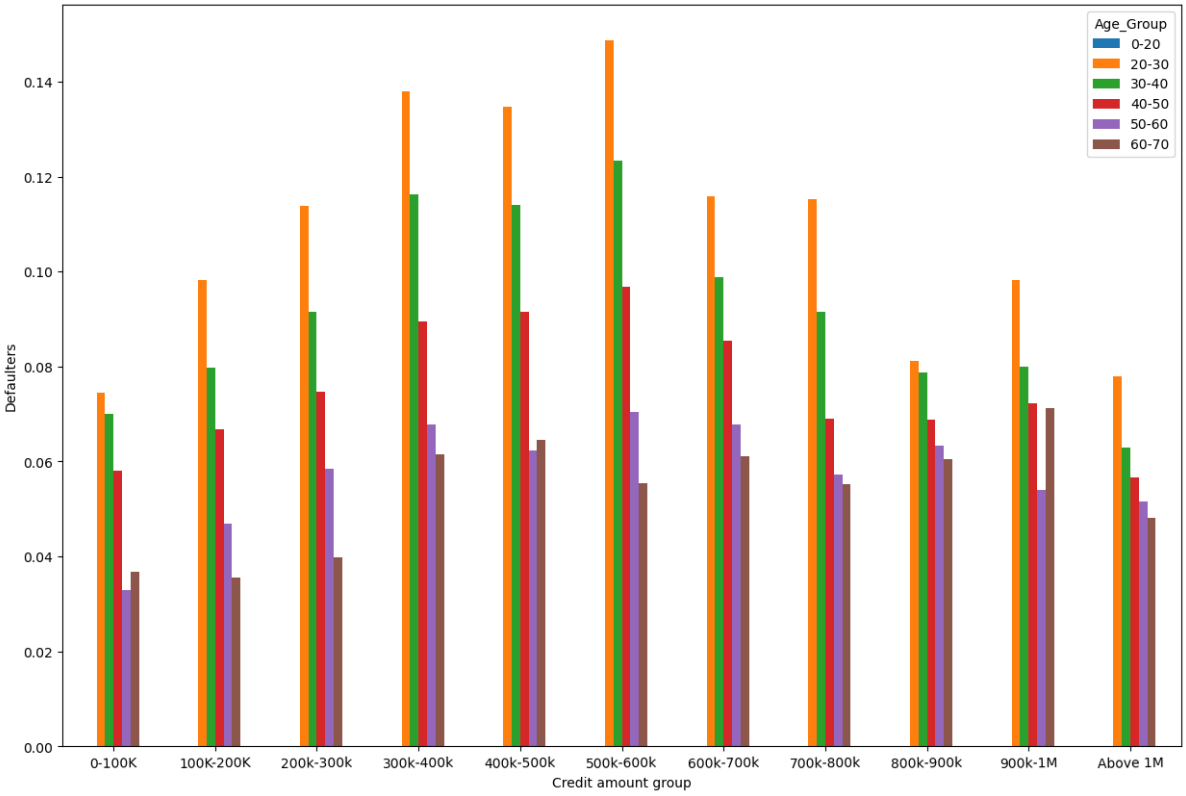
COMPARING AMOUNT CREDIT RANGE AND AGE GROUP

In [499…

```
#using pivot table to plot the barplot
appdf.pivot_table(values='TARGET',index='AMT_CREDIT_RANGE',columns='Age_Group',aggf
plt.xlabel('Credit amount group')
plt.ylabel('Defaulters')
```

Out[499]:

```
Text(0, 0.5, 'Defaulters')
```



INFERENCES:

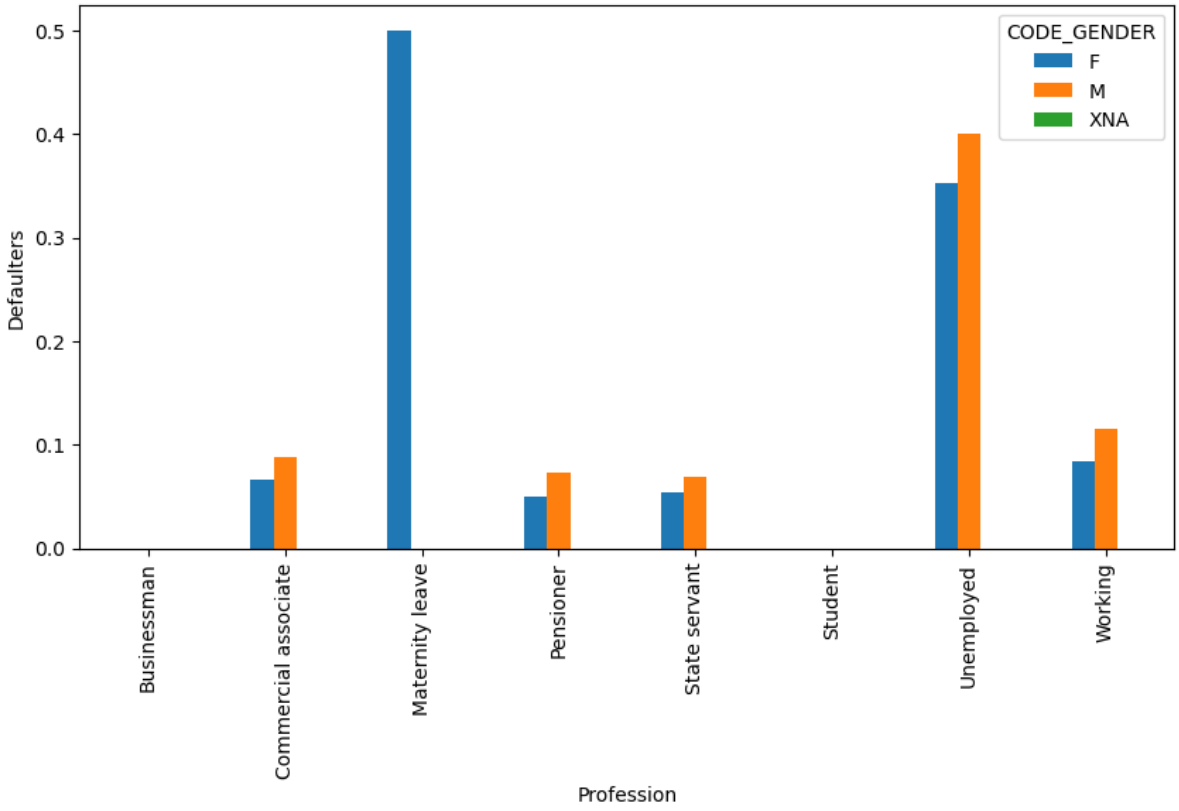
1.Form this plot we can see that yooung clients with the average income have the most default rates.

2.All the senior citizens are less liekly to be defaults.

COMPARING INCOME TYPE AND GENDER

In [500…
```
#using pivot table to plot the barplot
appdf.pivot_table(values='TARGET',index='NAME_INCOME_TYPE',columns='CODE_GENDER',ag
plt.xlabel('Profession')
plt.ylabel('Defaulters')
```

Out[500]:
```
Text(0, 0.5, 'Defaulters')
```



INFERENCES:

1.Unemployed clients have more percentage of defaulter rate.

2.Also clients who are at maternity leave have the highest cahnces of defaulter rate.

3.Overall males have more defaulter rate in all the Income types.

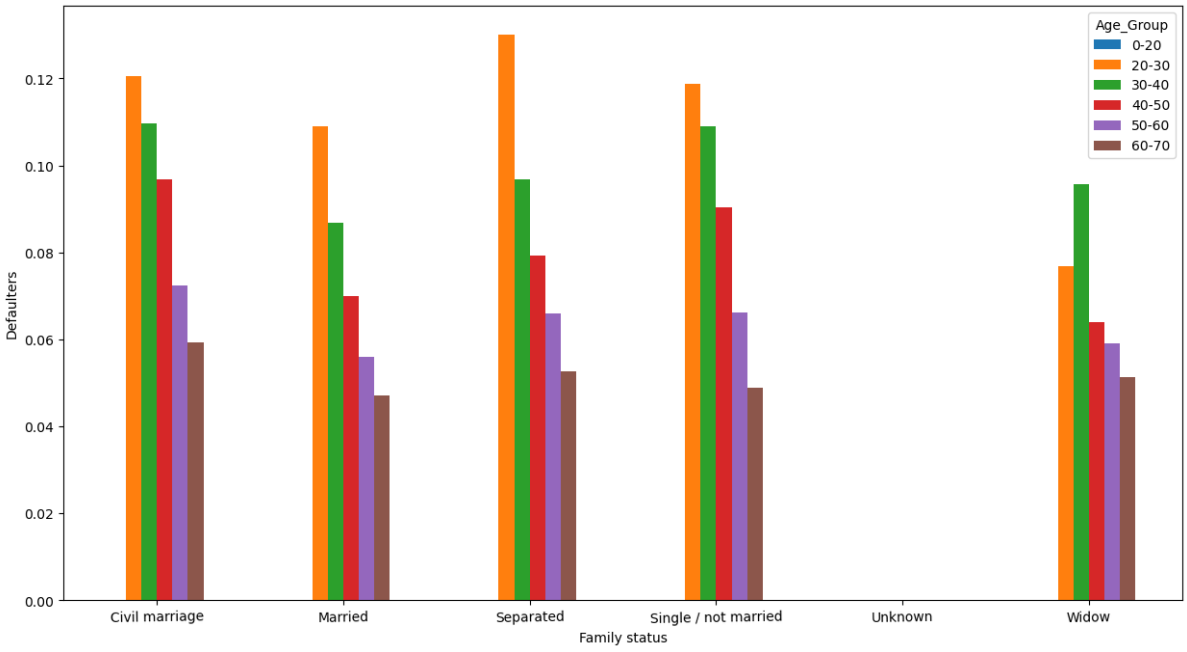
COMPARING FAMILY STATUS AND AGE GROUP

In [501…

```
#using pivot table to plot the barplot
appdf.pivot_table(values='TARGET',index='NAME_FAMILY_STATUS',columns='Age_Group',ag
plt.xlabel('Family status')
plt.ylabel('Defaulters')
```

Out[501]:

```
Text(0, 0.5, 'Defaulters')
```



INFERENCES:

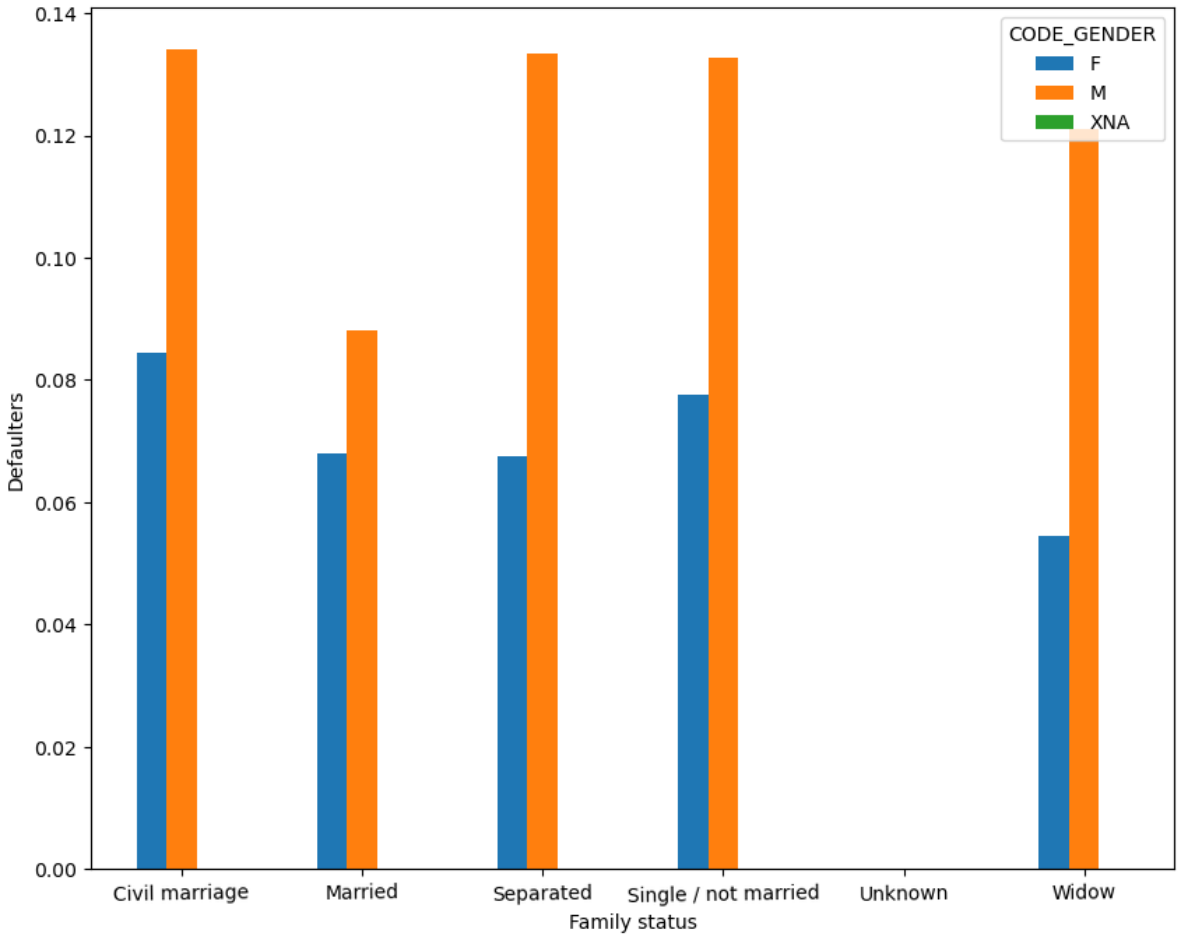
1.In this plot we can see that younger generations have higher percentage of defaulters in all the status except widow.

2.Lowest defaulter rate are the cleints who are elderly age.

COMPARING FAMILY STATUS AND CODE GENDER

In [502…
```
#using pivot table to plot the barplot
appdf.pivot_table(values='TARGET',index='NAME_FAMILY_STATUS',columns='CODE_GENDER',
plt.xlabel('Family status')
plt.ylabel('Defaulters')
```

Out[502]:
```
Text(0, 0.5, 'Defaulters')
```



INFERENCES:

1.We can see that whichever the family status is itn, male clients are the ones with high default rate.

2.If we see females have higher default rates with the family status of civil marriage followed by single/not married

MERGING APPLICATION DATA AND PREVIOUS DATA

In [503…
```
application_data_columns= ['SK_ID_CURR','TARGET','CODE_GENDER','NAME_EDUCATION_TYPE
```

In [504…
```
# Creating a dataset from current application for merging
appdf_merge = appdf[application_data_columns]
appdf_merge.head()
```

Out[504]:

	SK_ID_CURR	TARGET	CODE_GENDER	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	Age_Gr
0	100002	1	M	Secondary / secondary special	Single / not married	2(
1	100003	0	F	Higher education	Married	4(
2	100004	0	M	Secondary / secondary special	Single / not married	5(
3	100006	0	F	Secondary / secondary special	Civil marriage	5(
4	100007	0	M	Secondary / secondary special	Single / not married	5(

In [505…

```
#Merging the two datasets.
final_df_merge = pd.merge(prevdf,appdf_merge, on='SK_ID_CURR', how='left')
final_df_merge.head()
```

Out[505]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CI
0	2030495	271877	Consumer loans	1730.430	17145.0	17
1	2802425	108129	Cash loans	25188.615	607500.0	679
2	2523466	122040	Cash loans	15060.735	112500.0	136
3	2819243	176158	Cash loans	47041.335	450000.0	470
4	1784265	202054	Cash loans	31924.395	337500.0	404

5 rows × 28 columns

In [506…

```
final_df_merge.shape
```

Out[506]:

```
(1670214, 28)
```

In [507…

```
#Removing the rows containing nan values in the target column.
final_df_merge= final_df_merge[(~(np.isnan(final_df_merge['TARGET'])))]
```

In [508…

```
#Getting the info of the merged columns.
final_df_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1413701 entries, 0 to 1670213
Data columns (total 28 columns):
 #   Column                  Non-Null Count    Dtype
---  ------                  --------------    -----
 0   SK_ID_PREV              1413701 non-null  int64
 1   SK_ID_CURR              1413701 non-null  int64
 2   NAME_CONTRACT_TYPE      1413701 non-null  object
 3   AMT_ANNUITY             1413701 non-null  float64
 4   AMT_APPLICATION         1413701 non-null  float64
 5   AMT_CREDIT              1413700 non-null  float64
 6   AMT_GOODS_PRICE         1413701 non-null  float64
 7   NAME_CASH_LOAN_PURPOSE  1413701 non-null  object
 8   NAME_CONTRACT_STATUS    1413701 non-null  object
 9   DAYS_DECISION           1413701 non-null  int64
 10  NAME_PAYMENT_TYPE       1413701 non-null  object
 11  CODE_REJECT_REASON      1413701 non-null  object
 12  NAME_CLIENT_TYPE        1413701 non-null  object
 13  NAME_GOODS_CATEGORY     1413701 non-null  object
 14  NAME_PORTFOLIO          1413701 non-null  object
 15  NAME_PRODUCT_TYPE       1413701 non-null  object
 16  CHANNEL_TYPE            1413701 non-null  object
 17  SELLERPLACE_AREA        1413701 non-null  int64
 18  NAME_SELLER_INDUSTRY    1413701 non-null  object
 19  CNT_PAYMENT             1413701 non-null  float64
 20  NAME_YIELD_GROUP        1413701 non-null  object
 21  PRODUCT_COMBINATION     1413701 non-null  object
 22  TARGET                  1413701 non-null  float64
 23  CODE_GENDER             1413701 non-null  object
 24  NAME_EDUCATION_TYPE     1413701 non-null  object
 25  NAME_FAMILY_STATUS      1413701 non-null  object
 26  Age_Group               1413701 non-null  category
 27  AMT_INCOME_TOTAL_RANGE  1413683 non-null  category
dtypes: category(2), float64(6), int64(4), object(16)
memory usage: 293.9+ MB
```

UNIVARTIE ANALYSIS ON OUR FINAL MERGED DATASET

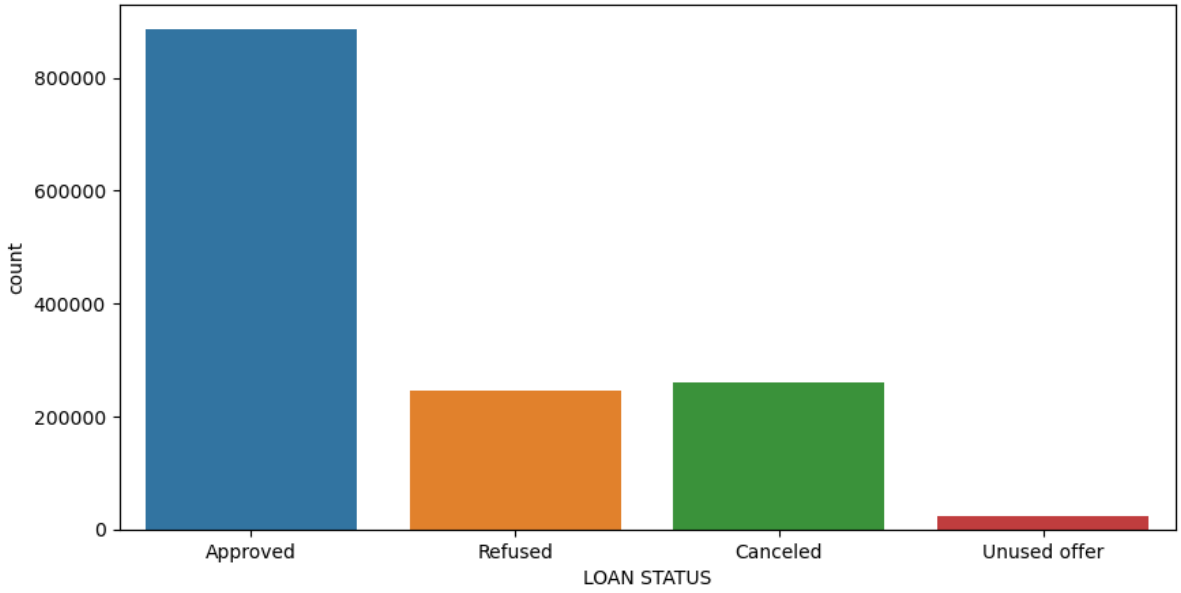
NAME_CONTRACT_STATUS

In [509…

```
#Ploting the countplot for univariate analysis.
plt.figure(figsize=(10,5))
ax = sns.countplot(x = 'NAME_CONTRACT_STATUS',data=final_df_merge)
ax.set(xlabel='LOAN STATUS')
```

Out[509]:

```
[Text(0.5, 0, 'LOAN STATUS')]
```



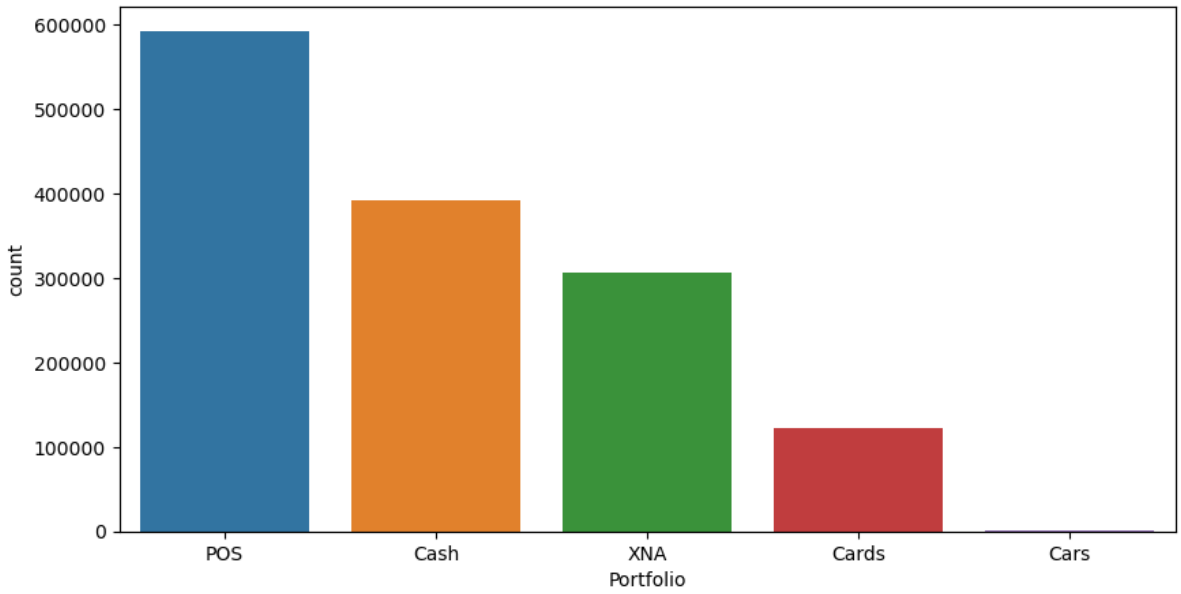
INFERENCES:

1.Mostly Loans are being approved

2.Amost same number loans are being refused or either got canceled

In [510…
```
#Ploting the countplot for univariate analysis.
plt.figure(figsize=(10,5))
ax = sns.countplot(x = 'NAME_PORTFOLIO',data=final_df_merge)
ax.set(xlabel='Portfolio')
```

Out[510]:
```
[Text(0.5, 0, 'Portfolio')]
```



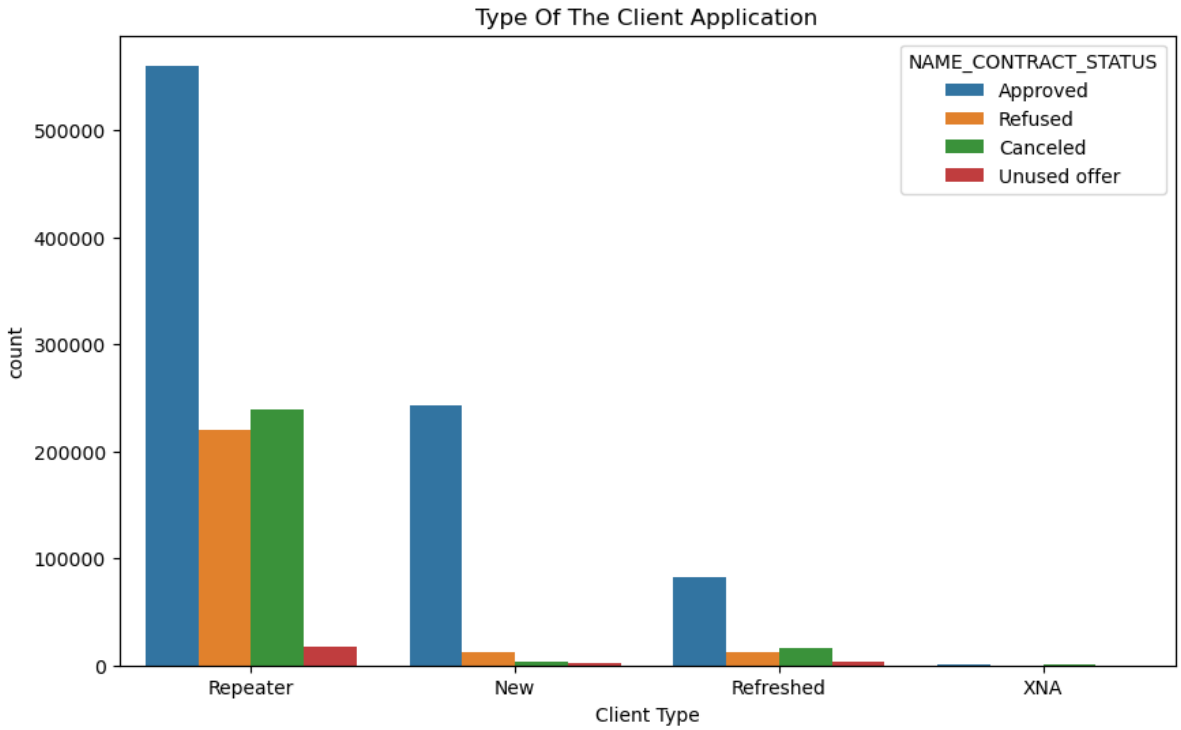
INFERENCES:

Most of the previous applciations were for POS followed by cash.

BIVARIATE ANALYSIS

In [511…
```
#Plotting Countplot for comparison
plt.figure(figsize=(10,6))

sns.countplot(x = 'NAME_CLIENT_TYPE' ,hue = 'NAME_CONTRACT_STATUS', data = final_df
plt.title('Type Of The Client Application')
plt.xlabel('Client Type')
```

Out[511]:
```
Text(0.5, 0, 'Client Type')
```

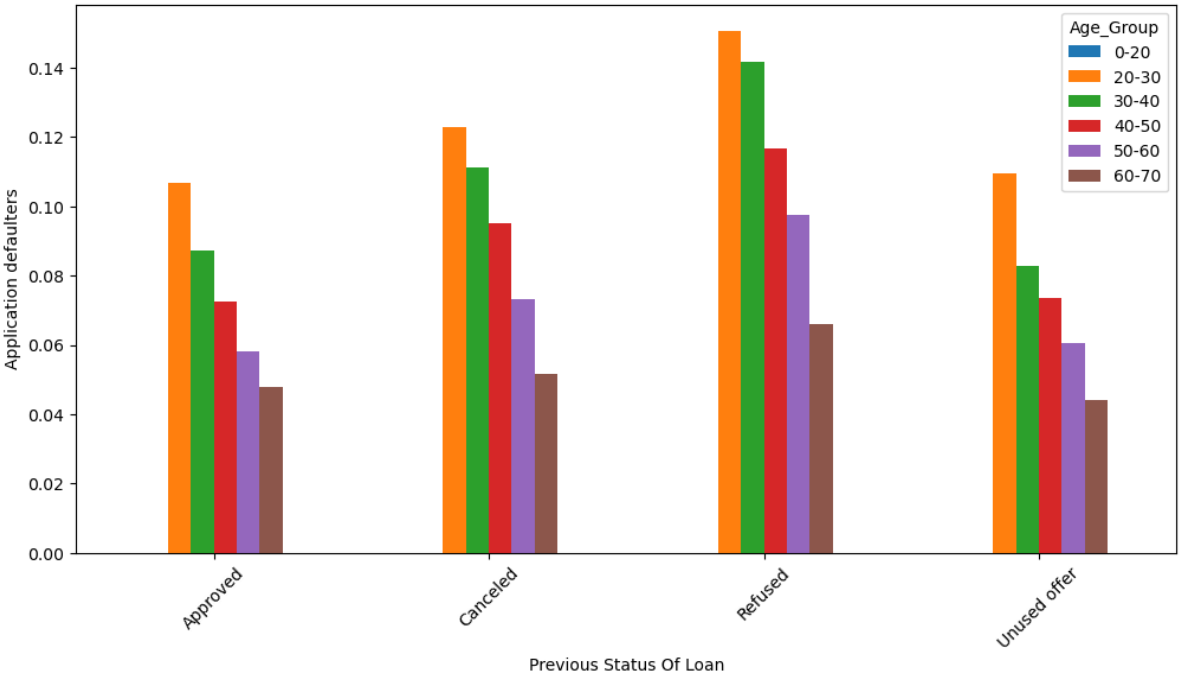


INFERENCES:

Most of the approved loans are off the repeater clients.

In [512…
```
#using pivot table to plot the barplot
final_df_merge.pivot_table(values='TARGET',index='NAME_CONTRACT_STATUS',columns='Ag
plt.xlabel('Previous Status Of Loan')
plt.ylabel('Application defaulters')
```

Out[512]:
```
Text(0, 0.5, 'Application defaulters')
```



INFERENCES:

1.As we can see in all the cases higher precentagews of young clients can be seen.

2.Refused clients are more defaulted than the previosuly approved clients.

FINAL RECOMMENDATIONS

Overall Recommendations

1. It is more safe to grant the loan to mid age clients and senior citizen clients with higher income.
2. Loan can be granteed to highly educated clients because their is very less chance of them being a defaulter.
3. Overall females have less chance of being a defaulter than males so loan can be granted to them.
4. Married cleints should also be given loans beacuse it has less defaulter rate as compared to other family status.

Overall Risks

1. Clients with low income groups should be avoided because of higher chances of being defaulter.
2. Unemployed clients can also be a big risk factor for providing with loan.
3. External credit score should be also considered before approving the clients appplication as it consits of clients credit score.
4. Lower secondary and secondary educated clients should be avoided for loan as they have high defaulter rates.