Bank Loan Case Study

By Aayushi Singh

Project Description

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

The Dataset of Project

Application data :
https://drive.google.com/file/d/1rBSWUClUUnHcGeYIWuB77nk-qLjlg3CE/view?usp=share_link
Previous Application data:
https://drive.google.com/file/d/1RRchsIaT7F6MawDBhxSgXV1THwT2mh93/view?usp=share_link
Column Description :
https://drive.google.com/file/d/1iDUWYziwq3FH_MrM79mLgd7-fY-ZEG_6/view?usp=share_link

Approach

By Exploratory Data Analysis we clean the data by removing columns that are not necessary to , perform analysis and give our conclusion on it
We first analyze the application dataframe: for it we clean the data, removing columns that have high percentage of null values or data that does not factor to the analysis ,we fill columns having the null values then analyze it and based on graph/plots observations are made. Similar steps are then taken for previous application data and then both the dataframe are merged together and based on that observations are made

Tech Stack used

Microsoft Excel
Google Docs

Jupyter Notebook

Insights and Result

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

- The client with payment difficulties: he/she had late payment more than X days on at least one of the first Y installments of the loan in our sample
- All other cases: All other cases when the payment is paid on time.

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

1. Approved: The company has approved loan application
2. Canceled: The client canceled the application sometime during approval. Either the client changed her/his mind about the loan or in some cases due to a higher risk of the client he received worse pricing which he did not want.
3. Refused: The company had rejected the loan (because the client does not meet their requirements etc.).
4. Unused Offer: Loan has been canceled by the client but on different stages of the process.

(All the insight and observation made in different steps of EDA are commented along code in the Jupyter Notebook)

2	100004	0	Revolving loans	M	Y	Y	0	67500.0	13500
3	100006	0	Cash loans	F	N	Y	0	135000.0	31268
4	100007	0	Cash loans	M	N	Y	0	121500.0	51300

5 rows × 122 columns

In [5]:
```
prev_app.head()
```

Out[5]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	WEEKI
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	0.0	17145.0	
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	NaN	607500.0	
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	NaN	112500.0	
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	NaN	450000.0	
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	NaN	337500.0	

5 rows × 37 columns

In [6]:
```
app.shape
```

Out[6]:
```
(307511, 122)
```

In [7]:
```
# to display all the columns of the table we will add a code to see all(here maximum 120) columns for data manupulation
pan.set_option('display.max_columns', 122)
```

In [8]:
```
#to preview missing columns
app.isnull()
```

Out[8]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_
0	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	
307506	False	False	False	False	False	False	False	False	
307507	False	False	False	False	False	False	False	False	
307508	False	False	False	False	False	False	False	False	
307509	False	False	False	False	False	False	False	False	
307510	False	False	False	False	False	False	False	False	

307511 rows × 122 columns

In [9]:
```
app.isnull().sum()
```

Out[9]:
```
SK_ID_CURR                      0
TARGET                          0
NAME_CONTRACT_TYPE              0
CODE_GENDER                     0
FLAG_OWN_CAR                    0
                              ...
AMT_REQ_CREDIT_BUREAU_DAY     41519
AMT_REQ_CREDIT_BUREAU_WEEK    41519
AMT_REQ_CREDIT_BUREAU_MON     41519
AMT_REQ_CREDIT_BUREAU_QRT     41519
AMT_REQ_CREDIT_BUREAU_YEAR    41519
Length: 122, dtype: int64
```

In [10]:
```
#Sorting them
app.isnull().sum().sort_values()
```

Out[10]:
```
SK_ID_CURR                        0
HOUR_APPR_PROCESS_START           0
REG_REGION_NOT_WORK_REGION        0
LIVE_REGION_NOT_WORK_REGION       0
REG_CITY_NOT_LIVE_CITY            0
                                ...
NONLIVINGAPARTMENTS_MEDI     213514
NONLIVINGAPARTMENTS_MODE     213514
COMMONAREA_MODE              214865
COMMONAREA_AVG               214865
COMMONAREA_MEDI              214865
Length: 122, dtype: int64
```

In [11]:
```
missing_info=pan.DataFrame(app.isnull().sum().sort_values()).reset_index()
missing_info.rename(columns={'index':'column_name',0:'total_null_values'},inplace=True)
missing_info.head()
```

In [14]:
```
missing_info[missing_info['missing_percentage']>=45]
```

Out[14]:

	column_name	total_null_values	missing_percentage
73	EMERGENCYSTATE_MODE	145755	47.398304
74	TOTALAREA_MODE	148431	48.268517
75	YEARS_BEGINEXPLUATATION_MODE	150007	48.781019
76	YEARS_BEGINEXPLUATATION_AVG	150007	48.781019
77	YEARS_BEGINEXPLUATATION_MEDI	150007	48.781019
78	FLOORSMAX_AVG	153020	49.760822
79	FLOORSMAX_MEDI	153020	49.760822
80	FLOORSMAX_MODE	153020	49.760822

81	HOUSETYPE_MODE	154297	50.176091
82	LIVINGAREA_AVG	154350	50.193326
83	LIVINGAREA_MODE	154350	50.193326
84	LIVINGAREA_MEDI	154350	50.193326
85	ENTRANCES_AVG	154828	50.348768
86	ENTRANCES_MODE	154828	50.348768
87	ENTRANCES_MEDI	154828	50.348768

Out[11]:

	column_name	total_null_values
0	SK_ID_CURR	0
1	HOUR_APPR_PROCESS_START	0
2	REG_REGION_NOT_WORK_REGION	0
3	LIVE_REGION_NOT_WORK_REGION	0
4	REG_CITY_NOT_LIVE_CITY	0

In [12]:
```
# Now to filter out which column to keep we find the percentage of null values in column based
#on that we decide what range to keep
missing_info['missing_percentage']=missing_info['total_null_values']/app.shape[0]*100
missing_info.head()
```

Out[12]:

	column_name	total_null_values	missing_percentage
0	SK_ID_CURR	0	0.0
1	HOUR_APPR_PROCESS_START	0	0.0
2	REG_REGION_NOT_WORK_REGION	0	0.0
3	LIVE_REGION_NOT_WORK_REGION	0	0.0
4	REG_CITY_NOT_LIVE_CITY	0	0.0

In [13]:
```
whole data is not visible on jupter for analysis we create an excel file
fo.to_excel(r"C:\Users\Sanjeev Kumar Singh\Desktop\Aayushi\projects\Trainity\Bank Loan Case Study\missing_info.xlsx",index=False)
```

In [14]:
```
missing_info[missing_info['missing_percentage']>=45]
```

Out[14]:

	column_name	total_null_values	missing_percentage
73	EMERGENCYSTATE_MODE	145755	47.398304
74	TOTALAREA_MODE	148431	48.268517
75	YEARS_BEGINEXPLUATATION_MODE	150007	48.781019
76	YEARS_BEGINEXPLUATATION_AVG	150007	48.781019
77	YEARS_BEGINEXPLUATATION_MEDI	150007	48.781019
78	FLOORSMAX_AVG	153020	49.760822
79	FLOORSMAX_MEDI	153020	49.760822
80	FLOORSMAX_MODE	153020	49.760822
81	HOUSETYPE_MODE	154297	50.176091
82	LIVINGAREA_AVG	154350	50.193326
83	LIVINGAREA_MODE	154350	50.193326
84	LIVINGAREA_MEDI	154350	50.193326
85	ENTRANCES_AVG	154828	50.348768
86	ENTRANCES_MODE	154828	50.348768
87	ENTRANCES_MEDI	154828	50.348768
108	YEARS_BUILD_MEDI	204488	66.497784
109	FLOORSMIN_AVG	208642	67.848630
110	FLOORSMIN_MODE	208642	67.848630
111	FLOORSMIN_MEDI	208642	67.848630
112	LIVINGAPARTMENTS_AVG	210199	68.354953
113	LIVINGAPARTMENTS_MODE	210199	68.354953
114	LIVINGAPARTMENTS_MEDI	210199	68.354953
115	FONDKAPREMONT_MODE	210295	68.386172
116	NONLIVINGAPARTMENTS_AVG	213514	69.432963
117	NONLIVINGAPARTMENTS_MEDI	213514	69.432963
118	NONLIVINGAPARTMENTS_MODE	213514	69.432963
119	COMMONAREA_MODE	214865	69.872297
120	COMMONAREA_AVG	214865	69.872297
121	COMMONAREA_MEDI	214865	69.872297

In [15]:
```
#data Imbalance Ratio
Re_payer  = app[app["TARGET"] == 0]
default  = app[app["TARGET"] == 1]

print("No. of defaulters: ", default.shape[0])
print("No. of Repayers: ", Re_payer.shape[0])
```

```
No. of defaulters:  24825
No. of Repayers:  282686
```

In [16]:
```
print("defaulters Percentile: ", default.shape[0]*100/(default.shape[0]+Re_payer.shape[0]))
```

```
defaulters Percentile:  8.072881945686495
```

In [17]:
```
#since with such a high percentage of null data we cannot use this in analysis so we remove them
to_drop_column=missing_info[missing_info['missing_percentage']>=45]['column_name'].to_list()
app_missing_removal=app.drop(labels=to_drop_column,axis=1)
app_missing_removal.shape
app_missing_removal.head()
```

Out[17]: (307511, 73)

Out[17]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREI
0	100002	1	Cash loans	M	N	Y	0	202500.0	40659

1	100003	0	Cash loans	F	N	N	0	270000.0	129350
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	13500
3	100006	0	Cash loans	F	N	Y	0	135000.0	31268
4	100007	0	Cash loans	M	N	Y	0	121500.0	51300

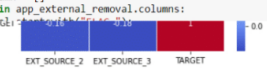
In [20]:
```
app_external_removal=app_missing_removal.drop(labels=['EXT_SOURCE_2','EXT_SOURCE_3'],axis=1)
app_external_removal.shape
app_external_removal.head()
```

Out[20]: (307511, 71)

Out[20]:

SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CRE
0 100002	1	Cash loans	M	N	Y	0	202500.0	40659
1 100003	0	Cash loans	F	N	N	0	270000.0	129350
2 100004	0	Revolving loans	M	Y	Y	0	67500.0	13500
3 100006	0	Cash loans	F	N	Y	0	135000.0	31268
4 100007	0	Cash loans	M	N	Y	0	121500.0	51300

In [21]:
```
#Working with flags
flag_column = []
for col in app_external_removal.columns:
```



In [20]:
```
app_external_removal=app_missing_removal.drop(labels=['EXT_SOURCE_2','EXT_SOURCE_3'],axis=1)
app_external_removal.shape
app_external_removal.head()
```

Out[20]: (307511, 71)

Out[20]:

SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CRE
0 100002	1	Cash loans	M	N	Y	0	202500.0	40659
1 100003	0	Cash loans	F	N	N	0	270000.0	129350
2 100004	0	Revolving loans	M	Y	Y	0	67500.0	13500
3 100006	0	Cash loans	F	N	Y	0	135000.0	31268
4 100007	0	Cash loans	M	N	Y	0	121500.0	51300

In [21]:
```
#Working with flags
flag_column = []
for col in app_external_removal.columns:
    'FLAG_DOCUMENT_19',
    'FLAG_DOCUMENT_20',
    'FLAG_DOCUMENT_21']
```

In [22]:
```
flag_target_col=app_external_removal[flag_column+['TARGET']].head()
flag_target_col.head()
```

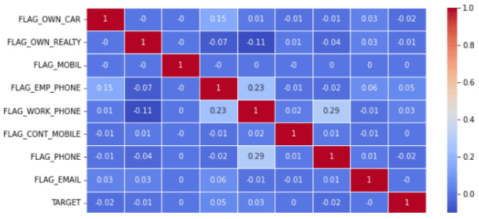
Out[22]:

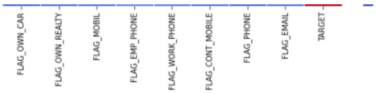
FLAG_OWN_CAR	FLAG_OWN_REALTY	FLAG_MOBIL	FLAG_EMP_PHONE	FLAG_WORK_PHONE	FLAG_CONT_MOBILE	FLAG_PHONE	FLAG_EMAIL	FLAG_
0 N	Y	1	1	0	1	1	0	
1 N	N	1	1	0	1	1	0	
2 Y	Y	1	1	1	1	1	0	
3 N	Y	1	1	0	1	0	0	
4 N	Y	1	1	0	1	0	0	

In [23]:
```
#plotting indiviusal bar graph of flag with respect to target
mplot.figure(figsize=(30,30)) # helps to configure
for i,col in enumerate(flag_column):
    mplot.subplot(7,4,i+1)
    sea.countplot(data=flag_target_col,x=col,hue='TARGET')
```

Out[26]: <Figure size 720x360 with 0 Axes>

Out[26]: <AxesSubplot:>





In [27]:
```
# as per Observation these are not required for analysis of data as they do not factor the results so we remove them
 'FLAG_DOCUMENT_19',
 'FLAG_DOCUMENT_20',
 'FLAG_DOCUMENT_21']
```

In [22]:
```
flag_target_col=app_external_removal[flag_column+['TARGET']].head()
flag_target_col.head()
```

Out[22]:

	FLAG_OWN_CAR	FLAG_OWN_REALTY	FLAG_MOBIL	FLAG_EMP_PHONE	FLAG_WORK_PHONE	FLAG_CONT_MOBILE	FLAG_PHONE	FLAG_EMAIL	FLAG_I
0	N	Y	1	1	0	1	1	0	
1	N	N	1	1	0	1	1	0	
2	Y	Y	1	1	1	1	1	0	
3	N	Y	1	1	0	1	0	0	
4	N	Y	1	1	0	1	0	0	

In [23]:
```
#plotting indiviusal bar graph of flag with respect to target
mplot.figure(figsize=(30,30)) # helps to configure
for i,col in enumerate(flag_column):
    mplot.subplot(7,4,i+1)
    sea.countplot(data=flag_target_col,x=col,hue='TARGET')
```

4	N	Y	1	1	0	1	0	0

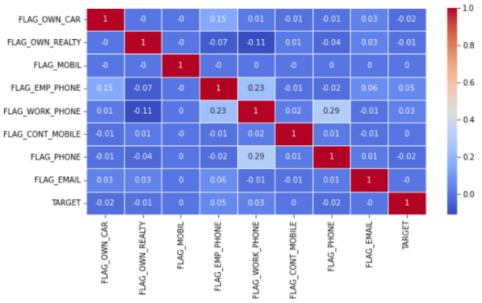
In [25]:
```
#replacing y and with 0 and 1
flag_binary["FLAG_OWN_CAR"]=flag_binary["FLAG_OWN_CAR"].replace({"N":0,'Y':1})
flag_binary["FLAG_OWN_REALTY"]=flag_binary["FLAG_OWN_REALTY"].replace({"N":0,'Y':1})
```

```
C:\Users\SANJEE~1\AppData\Local\Temp/ipykernel_33732/388380337.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  flag_binary["FLAG_OWN_CAR"]=flag_binary["FLAG_OWN_CAR"].replace({"N":0,'Y':1})
C:\Users\SANJEE~1\AppData\Local\Temp/ipykernel_33732/388380337.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  flag_binary["FLAG_OWN_REALTY"]=flag_binary["FLAG_OWN_REALTY"].replace({"N":0,'Y':1})
```

In [26]:
```
#so for correation as for selective flags using heatmap
flag_comp=round(flag_binary.corr(),2)
mplot.figure(figsize=(10,5))
sea.heatmap(flag_comp,cmap='coolwarm',linewidth=0.6,annot=True)
```

Out[26]: <Figure size 720x360 with 0 Axes>

Out[26]: <AxesSubplot:>



In [27]:
```
# as per Observation these are not required for analysis of data as they do not factor the results so we remove them
app_ext_removal=app_external_removal.drop(labels=['FLAG_OWN_CAR','FLAG_OWN_REALTY','FLAG_MOBIL','FLAG_EMP_PHONE','FLAG_WORK_PHONE',
                                                  'FLAG_CONT_MOBILE', 'FLAG_PHONE','FLAG_EMAIL','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'],axis=1)
app_ext_removal.shape
app_ext_removal.head()
```

Out[27]: (307511, 43)

Out[27]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE
0	100002	1	Cash loans	M	0	202500.0	406597.5	24700.5	351000.0
1	100003	0	Cash loans	F	0	270000.0	1293502.5	35698.5	1129500.0
2	100004	0	Revolving loans	M	0	67500.0	135000.0	6750.0	135000.0
3	100006	0	Cash loans	F	0	135000.0	312682.5	29686.5	297000.0

4	100007	0	Cash loans	M	0	121500.0	513000.0	21865.5	513000.0

In [28]:
```
starting_index = app.columns.get_loc('OBS_30_CNT_SOCIAL_CIRCLE')
ending_index = app.columns.get_loc('DEF_60_CNT_SOCIAL_CIRCLE')

social_circle_df = app.iloc[:, starting_index:ending_index+1]
social_circle_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 4 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   OBS_30_CNT_SOCIAL_CIRCLE  306490 non-null  float64
 1   DEF_30_CNT_SOCIAL_CIRCLE  306490 non-null  float64
 2   OBS_60_CNT_SOCIAL_CIRCLE  306490 non-null  float64
 3   DEF_60_CNT_SOCIAL_CIRCLE  306490 non-null  float64
dtypes: float64(4)
memory usage: 9.4 MB
```

In [29]:
```
social_circle_df.describe()
```

Out[29]:

	OBS_30_CNT_SOCIAL_CIRCLE	DEF_30_CNT_SOCIAL_CIRCLE	OBS_60_CNT_SOCIAL_CIRCLE	DEF_60_CNT_SOCIAL_CIRCLE
count	306490.000000	306490.000000	306490.000000	306490.000000
mean	1.422245	0.143421	1.405292	0.100049
std	2.400989	0.446698	2.379803	0.362291
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000
75%	2.000000	0.000000	2.000000	0.000000
max	348.000000	34.000000	344.000000	24.000000

In [31]:
```
#both the columns are highly correlated and identical columns
fig=mplot.subplots(figsize=(7,7))

for i, j in enumerate(['DEF_60_CNT_SOCIAL_CIRCLE','OBS_60_CNT_SOCIAL_CIRCLE']):
    mplot.subplot(2, 2, i+1)
    mplot.subplots_adjust(hspace = 1.0)
    sea.countplot(j, data=Re_payer)
    mplot.title("Repayers")
    mplot.xticks(rotation=90)
    mplot.tight_layout()
```

Out[31]: <AxesSubplot:>

```
E:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will re
sult in an error or misinterpretation.
  warnings.warn(
```

Out[31]: <AxesSubplot:xlabel='DEF_60_CNT_SOCIAL_CIRCLE', ylabel='count'>

Out[31]: Text(0.5, 1.0, 'Repayers')

Out[31]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
 [Text(0, 0, '0.0'),
  Text(1, 0, '1.0'),
  Text(2, 0, '2.0'),
  Text(3, 0, '3.0'),
  Text(4, 0, '4.0'),
  Text(5, 0, '5.0'),
```

In [30]:
```
sea.heatmap(social_circle_df.corr(), annot=True)
```

Out[30]: <AxesSubplot:>


```
  Text(6, 0, '6.0'),
  Text(7, 0, '7.0'),
  Text(8, 0, '24.0')])
```

Out[31]: <AxesSubplot:>

```
E:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will re
sult in an error or misinterpretation.
  warnings.warn(
```

Out[31]: <AxesSubplot:xlabel='OBS_60_CNT_SOCIAL_CIRCLE', ylabel='count'>

Out[31]: Text(0.5, 1.0, 'Repayers')

Out[31]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]),
 [Text(0, 0, '0.0'),
  Text(1, 0, '1.0'),
  Text(2, 0, '2.0'),
  Text(3, 0, '3.0'),
  Text(4, 0, '4.0'),
  Text(5, 0, '5.0'),
```

```
  Text(6, 0, '6.0'),
  Text(7, 0, '7.0'),
  Text(8, 0, '8.0'),
  Text(9, 0, '9.0'),
  Text(10, 0, '10.0'),
  Text(11, 0, '11.0'),
  Text(12, 0, '12.0'),
  Text(13, 0, '13.0'),
```

Out[32]:
```
Text(0.5, 1.0, 'defaulters')
```

Out[32]:
```
(array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, '0.0'),
  Text(1, 0, '1.0'),
  Text(2, 0, '2.0'),
  Text(3, 0, '3.0'),
  Text(4, 0, '4.0'),
  Text(5, 0, '5.0')])
```

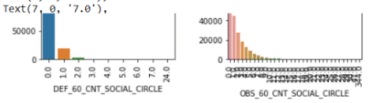
Out[32]:
```
<AxesSubplot:>
```

```
E:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will re
sult in an error or misinterpretation.
  warnings.warn(
```

Out[32]:
```
<AxesSubplot:xlabel='OBS_60_CNT_SOCIAL_CIRCLE', ylabel='count'>
```

Out[32]:
```
Text(0.5, 1.0, 'defaulters')
```

Out[32]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24]),
 [Text(0, 0, '0.0'),
  Text(1, 0, '1.0'),
  Text(2, 0, '2.0'),
  Text(3, 0, '3.0'),
  Text(4, 0, '4.0'),
  Text(5, 0, '5.0'),
  Text(6, 0, '6.0'),
  Text(7, 0, '7.0'),
```



In [32]:
```
fig=mplot.subplots(figsize=(7,7))

for i, j in enumerate(['DEF_60_CNT_SOCIAL_CIRCLE','OBS_60_CNT_SOCIAL_CIRCLE']):
    mplot.subplot(2, 2, i+1)
    mplot.subplots_adjust(hspace = 1.0)
    sea.countplot(j, data=default)
    mplot.title("defaulters")
    mplot.xticks(rotation=90)
    mplot.tight_layout()

    #For defaulter and non-defaulter 'DEF_60_CNT_SOCIAL_CIRCLE','OBS_60_CNT_SOCIAL_CIRCLE' features show similar trend.
    #so it does not constitute as a factor
```

Out[32]:
```
<AxesSubplot:>
```

```
E:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will re
sult in an error or misinterpretation.
  warnings.warn(
```

Out[32]:
```
<AxesSubplot:xlabel='DEF_60_CNT_SOCIAL_CIRCLE', ylabel='count'>
```

```
 15  DAYS_BIRTH                  307511 non-null  int64
 16  DAYS_EMPLOYED               307511 non-null  int64
 17  DAYS_REGISTRATION           307511 non-null  float64
 18  DAYS_ID_PUBLISH             307511 non-null  int64
 19  OCCUPATION_TYPE             211120 non-null  object
 20  CNT_FAM_MEMBERS             307509 non-null  float64
 21  REGION_RATING_CLIENT        307511 non-null  int64
 22  REGION_RATING_CLIENT_W_CITY 307511 non-null  int64
 23  WEEKDAY_APPR_PROCESS_START  307511 non-null  object
 24  HOUR_APPR_PROCESS_START     307511 non-null  int64
 25  REG_REGION_NOT_LIVE_REGION  307511 non-null  int64
 26  REG_REGION_NOT_WORK_REGION  307511 non-null  int64
 27  LIVE_REGION_NOT_WORK_REGION 307511 non-null  int64
 28  REG_CITY_NOT_LIVE_CITY      307511 non-null  int64
 29  REG_CITY_NOT_WORK_CITY      307511 non-null  int64
 30  LIVE_CITY_NOT_WORK_CITY     307511 non-null  int64
 31  ORGANIZATION_TYPE           307511 non-null  object
 32  OBS_30_CNT_SOCIAL_CIRCLE    306490 non-null  float64
 33  DEF_30_CNT_SOCIAL_CIRCLE    306490 non-null  float64
 34  OBS_60_CNT_SOCIAL_CIRCLE    306490 non-null  float64
 35  DEF_60_CNT_SOCIAL_CIRCLE    306490 non-null  float64
 36  DAYS_LAST_PHONE_CHANGE      307510 non-null  float64
 37  AMT_REQ_CREDIT_BUREAU_HOUR  265992 non-null  float64
 38  AMT_REQ_CREDIT_BUREAU_DAY   265992 non-null  float64
 39  AMT_REQ_CREDIT_BUREAU_WEEK  265992 non-null  float64
 40  AMT_REQ_CREDIT_BUREAU_MON   265992 non-null  float64
 41  AMT_REQ_CREDIT_BUREAU_QRT   265992 non-null  float64
 42  AMT_REQ_CREDIT_BUREAU_YEAR  265992 non-null  float64
dtypes: float64(18), int64(15), object(10)
memory usage: 100.9+ MB
```

In [34]:
```
# we need to understand the columns have null values and according according to the relevance of the column we
#remove the null values
app_ext_removal.isnull().sum()/app_ext_removal.shape[0]
```

```
REG_REGION_NOT_LIVE_REGION      0.000000
REG_REGION_NOT_WORK_REGION      0.000000
LIVE_REGION_NOT_WORK_REGION     0.000000
REG_CITY_NOT_LIVE_CITY          0.000000
REG_CITY_NOT_WORK_CITY          0.000000
LIVE_CITY_NOT_WORK_CITY         0.000000
ORGANIZATION_TYPE               0.000000
OBS_30_CNT_SOCIAL_CIRCLE        0.003320
DEF_30_CNT_SOCIAL_CIRCLE        0.003320
OBS_60_CNT_SOCIAL_CIRCLE        0.003320
DEF_60_CNT_SOCIAL_CIRCLE        0.003320
DAYS_LAST_PHONE_CHANGE          0.000003
AMT_REQ_CREDIT_BUREAU_HOUR      0.135016
AMT_REQ_CREDIT_BUREAU_DAY       0.135016
AMT_REQ_CREDIT_BUREAU_WEEK      0.135016
AMT_REQ_CREDIT_BUREAU_MON       0.135016
AMT_REQ_CREDIT_BUREAU_QRT       0.135016
```

```
AMT_REQ_CREDIT_BUREAU_YEAR     0.135016
dtype: float64
```

In [35]:
```
#AMT_ANNUITY : Loan annuity
#as observed above the datatype os float or we can also use
#Amt_ann_dtype=app_ext_removal['AMT_ANNUITY'].dtypes
#print(Amt_ann_dtype)
```

```
 15  DAYS_BIRTH                  307511 non-null  int64
 16  DAYS_EMPLOYED               307511 non-null  int64
 17  DAYS_REGISTRATION           307511 non-null  float64
 18  DAYS_ID_PUBLISH             307511 non-null  int64
 19  OCCUPATION_TYPE             211120 non-null  object
 20  CNT_FAM_MEMBERS             307509 non-null  float64
 21  REGION_RATING_CLIENT        307511 non-null  int64
 22  REGION_RATING_CLIENT_W_CITY 307511 non-null  int64
 23  WEEKDAY_APPR_PROCESS_START  307511 non-null  object
 24  HOUR_APPR_PROCESS_START     307511 non-null  int64
 25  REG_REGION_NOT_LIVE_REGION  307511 non-null  int64
 26  REG_REGION_NOT_WORK_REGION  307511 non-null  int64
 27  LIVE_REGION_NOT_WORK_REGION 307511 non-null  int64
 28  REG_CITY_NOT_LIVE_CITY      307511 non-null  int64
 29  REG_CITY_NOT_WORK_CITY      307511 non-null  int64
 30  LIVE_CITY_NOT_WORK_CITY     307511 non-null  int64
 31  ORGANIZATION_TYPE           307511 non-null  object
 32  OBS_30_CNT_SOCIAL_CIRCLE    306490 non-null  float64
 33  DEF_30_CNT_SOCIAL_CIRCLE    306490 non-null  float64
 34  OBS_60_CNT_SOCIAL_CIRCLE    306490 non-null  float64
 35  DEF_60_CNT_SOCIAL_CIRCLE    306490 non-null  float64
 36  DAYS_LAST_PHONE_CHANGE      307510 non-null  float64
 37  AMT_REQ_CREDIT_BUREAU_HOUR  265992 non-null  float64
 38  AMT_REQ_CREDIT_BUREAU_DAY   265992 non-null  float64
 39  AMT_REQ_CREDIT_BUREAU_WEEK  265992 non-null  float64
 40  AMT_REQ_CREDIT_BUREAU_MON   265992 non-null  float64
 41  AMT_REQ_CREDIT_BUREAU_QRT   265992 non-null  float64
 42  AMT_REQ_CREDIT_BUREAU_YEAR  265992 non-null  float64
dtypes: float64(18), int64(15), object(10)
memory usage: 100.9+ MB
AMT_REQ_CREDIT_BUREAU_YEAR     0.135016
dtype: float64
```

In [37]:
```
#AMT_GOODS_PRICE:For consumer loans it is the price of the goods for which the loan is given
app_ext_removal['AMT_GOODS_PRICE'].describe()
```

Out[37]:
```
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 [38]:
```
#since the above data is difficult for analysis we will seprately define min max and median
app_ext_removal['AMT_GOODS_PRICE'].agg(['min','max','median','mean'])
```

Out[38]:
```
min       4.050000e+04
max       4.050000e+06
median    4.500000e+05
mean      5.383962e+05
Name: AMT_GOODS_PRICE, dtype: float64
```

In [39]:
```
#full value of mean
app_ext_removal['AMT_GOODS_PRICE'].mean()
```

Out[39]: 538396.2074288895

Out[49]:
```
CNT_FAM_MEMBERS
15.0         1
13.0         1
11.0         1
20.0         2
14.0         2
12.0         2
16.0         2
10.0         3
9.0          6
8.0         20
7.0         81
6.0        408
5.0       3478
4.0      24697
3.0      52601
1.0      67847
2.0     158357
dtype: int64
```

In [50]:
```
app_ext_removal['CNT_FAM_MEMBERS'].mode()
```

Out[50]:
```
0    2.0
dtype: float64
```

In [51]:
```
app_ext_removal['CNT_FAM_MEMBERS']=app_ext_removal['CNT_FAM_MEMBERS'].fillna((app_ext_removal['CNT_FAM_MEMBERS'].mode()[0]))
```

```
WEEKDAY_APPR_PROCESS_START     0.000000
HOUR_APPR_PROCESS_START        0.000000
REG_REGION_NOT_LIVE_REGION     0.000000
REG_REGION_NOT_WORK_REGION     0.000000
LIVE_REGION_NOT_WORK_REGION    0.000000
REG_CITY_NOT_LIVE_CITY         0.000000
REG_CITY_NOT_WORK_CITY         0.000000
LIVE_CITY_NOT_WORK_CITY        0.000000
ORGANIZATION_TYPE              0.000000
OBS_30_CNT_SOCIAL_CIRCLE       0.003320
DEF_30_CNT_SOCIAL_CIRCLE       0.003320
OBS_60_CNT_SOCIAL_CIRCLE       0.003320
DEF_60_CNT_SOCIAL_CIRCLE       0.003320
DAYS_LAST_PHONE_CHANGE         0.000003
AMT_REQ_CREDIT_BUREAU_HOUR     0.135016
AMT_REQ_CREDIT_BUREAU_DAY      0.135016
AMT_REQ_CREDIT_BUREAU_WEEK     0.135016
AMT_REQ_CREDIT_BUREAU_MON      0.135016
AMT_REQ_CREDIT_BUREAU_QRT      0.135016
AMT_REQ_CREDIT_BUREAU_YEAR     0.135016
dtype: float64
```

In [41]:
```
#NAME_TYPE_SUITE:Who was accompanying client when he was applying for the loan
```

```
app_ext_removal.groupby(['NAME_TYPE_SUITE']).size().sort_values()
```

Out[41]:
```
NAME_TYPE_SUITE
Group of people       271
Other_A               866
```

```
#since data is float we will use statistics
app_ext_removal['AMT_ANNUITY'].describe()
```

Out[35]:
```
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 [36]:
```
#median and mean of this column are habing relatitively similar values
app_ext_removal['AMT_ANNUITY']=app_ext_removal['AMT_ANNUITY'].fillna((app_ext_removal['AMT_ANNUITY'].mean()))
app_ext_removal.isnull().sum()/app_ext_removal.shape[0]
```

Out[36]:
```
SK_ID_CURR                    0.000000
TARGET                        0.000000
NAME_CONTRACT_TYPE            0.000000
CODE_GENDER                   0.000000
CNT_CHILDREN                  0.000000
AMT_INCOME_TOTAL              0.000000
AMT_CREDIT                    0.000000
AMT_ANNUITY                   0.000000
AMT_GOODS_PRICE               0.000904
NAME_TYPE_SUITE               0.004201
NAME_INCOME_TYPE              0.000000
NAME_EDUCATION_TYPE           0.000000
                                  ...
NAME_HOUSING_TYPE                    6
NAME_FAMILY_STATUS                   6
WEEKDAY_APPR_PROCESS_START           7
NAME_TYPE_SUITE                      7
NAME_INCOME_TYPE                     8
AMT_REQ_CREDIT_BUREAU_DAY            9
DEF_60_CNT_SOCIAL_CIRCLE             9
AMT_REQ_CREDIT_BUREAU_WEEK           9
DEF_30_CNT_SOCIAL_CIRCLE            10
AMT_REQ_CREDIT_BUREAU_QRT           11
CNT_CHILDREN                        15
CNT_FAM_MEMBERS                     17
OCCUPATION_TYPE                     18
HOUR_APPR_PROCESS_START             24
AMT_REQ_CREDIT_BUREAU_MON           24
AMT_REQ_CREDIT_BUREAU_YEAR          25
OBS_30_CNT_SOCIAL_CIRCLE            33
OBS_60_CNT_SOCIAL_CIRCLE            33
ORGANIZATION_TYPE                   58
REGION_POPULATION_RELATIVE          81
AMT_GOODS_PRICE                   1002
AMT_INCOME_TOTAL                  2548
DAYS_LAST_PHONE_CHANGE            3773
AMT_CREDIT                        5603
DAYS_ID_PUBLISH                   6168
DAYS_EMPLOYED                    12574
AMT_ANNUITY                      13673
DAYS_REGISTRATION                15688
DAYS_BIRTH                       17460
```

In [57]:
```
for col in amt_req_col:
    app_ext_removal[col]=app_ext_removal[col].fillna((app_ext_removal[col].median()))
```

In [58]:
```
app_ext_removal.nunique().sort_values()
# through this we understand in which columns Outliers can exist
#therefore by this columns to be observed are
#AMT_GOODS_PRICE                1002
#AMT_INCOME_TOTAL               2548
#AMT_CREDIT                     5603
#DAYS_EMPLOYED                 12574
#AMT_ANNUITY                   13673
#DAYS_BIRTH                    17460
```

Out[58]:
```
LIVE_REGION_NOT_WORK_REGION          2
TARGET                               2
NAME_CONTRACT_TYPE                   2
REG_REGION_NOT_LIVE_REGION           2
REG_CITY_NOT_LIVE_CITY               2
REG_CITY_NOT_WORK_CITY               2
LIVE_CITY_NOT_WORK_CITY              2
REG_REGION_NOT_WORK_REGION           2
REGION_RATING_CLIENT_W_CITY          3
REGION_RATING_CLIENT                 3
CODE_GENDER                          3
NAME_EDUCATION_TYPE                  5
AMT_REQ_CREDIT_BUREAU_HOUR           5
```

In [63]:
```
app_ext_removal.groupby(['AMT_GOODS_PRICE_RANGE']).size()
```

Out[63]:
```
AMT_GOODS_PRICE_RANGE
0-100K         8709
100k-200K     32956
200K-300K     62761
300K-400K     21219
400K-500K     57251
500K-600K     13117
600K-700K     40024
700K-800K      8110
800K-900K     21484
Above 900K    41880
dtype: int64
```

In [64]:
```
len(app_ext_removal.columns)
```

Out[64]:
```
44
```

In [65]:
```
#for AMT_INCOME_TOTAL
app_ext_removal['AMT_INCOME_TOTAL'].quantile([0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.99])
```

Out[65]:
```
0.10     81000.0
0.20     99000.0
0.30    112500.0
0.40    135000.0
0.50    147150.0
```

In [57]:
```
for col in amt_req_col:
    app_ext_removal[col]=app_ext_removal[col].fillna((app_ext_removal[col].median()))
```

In [58]:
```
app_ext_removal.nunique().sort_values()
# through this we understand in which Columns Outliers can exist
#therefore by this columns to be observed are
#AMT_GOODS_PRICE                  1002
#AMT_INCOME_TOTAL                 2548
#AMT_CREDIT                       5603
#DAYS_EMPLOYED                   12574
#AMT_ANNUITY                     13673
#DAYS_BIRTH                      17460
```

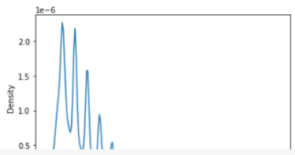
Out[58]:
```
LIVE_REGION_NOT_WORK_REGION         2
TARGET                              2
NAME_CONTRACT_TYPE                  2
REG_REGION_NOT_LIVE_REGION          2
REG_CITY_NOT_LIVE_CITY              2
REG_CITY_NOT_WORK_CITY              2
LIVE_CITY_NOT_WORK_CITY             2
REG_REGION_NOT_WORK_REGION          2
REGION_RATING_CLIENT_W_CITY         3
REGION_RATING_CLIENT                3
CODE_GENDER                         3
NAME_EDUCATION_TYPE                 5
AMT_REQ_CREDIT_BUREAU_HOUR          5
NAME_HOUSING_TYPE                   6
NAME_FAMILY_STATUS                  6
WEEKDAY_APPR_PROCESS_START          7
NAME_TYPE_SUITE                     7
NAME_INCOME_TYPE                    8
AMT_REQ_CREDIT_BUREAU_DAY           9
DEF_60_CNT_SOCIAL_CIRCLE            9
AMT_REQ_CREDIT_BUREAU_WEEK          9
DEF_30_CNT_SOCIAL_CIRCLE           10
AMT_REQ_CREDIT_BUREAU_QRT          11
CNT_CHILDREN                       15
CNT_FAM_MEMBERS                    17
OCCUPATION_TYPE                    18
HOUR_APPR_PROCESS_START            24
AMT_REQ_CREDIT_BUREAU_MON          24
AMT_REQ_CREDIT_BUREAU_YEAR         25
OBS_30_CNT_SOCIAL_CIRCLE           33
OBS_60_CNT_SOCIAL_CIRCLE           33
ORGANIZATION_TYPE                  58
REGION_POPULATION_RELATIVE         81
AMT_GOODS_PRICE                  1002
AMT_INCOME_TOTAL                 2548
DAYS_LAST_PHONE_CHANGE           3773
AMT_CREDIT                       5603
DAYS_ID_PUBLISH                  6168
DAYS_EMPLOYED                   12574
AMT_ANNUITY                     13673
DAYS_REGISTRATION               15688
DAYS_BIRTH                      17460
SK_ID_CURR                     307511
dtype: int64
```

In [59]:
```
#AMT_GOODS_PRICE range
#app_ext_removal['AMT_GOODS_PRICE'].describe()
app_ext_removal['AMT_GOODS_PRICE'].agg(['min','max','median'])
#as observed the data has outliers so we plot the data to understand the trend
```

Out[59]:
```
min         40500.0
max       4050000.0
median     450000.0
Name: AMT_GOODS_PRICE, dtype: float64
```

In [60]:
```
sea.kdeplot(data=app_ext_removal,x='AMT_GOODS_PRICE')
```

Out[60]:
```
<AxesSubplot:xlabel='AMT_GOODS_PRICE', ylabel='Density'>
```



In [63]:
```
app_ext_removal.groupby(['AMT_GOODS_PRICE_RANGE']).size()
```

Out[63]:
```
AMT_GOODS_PRICE_RANGE
0-100K          8709
100k-200K      32956
200K-300K      62761
300K-400K      21219
400K-500K      57251
500K-600K      13117
600K-700K      40024
700K-800K       8110
800K-900K      21484
Above 900K     41880
dtype: int64
```

In [64]:
```
len(app_ext_removal.columns)
```

Out[64]:
```
44
```

In [65]:
```
#for AMT_INCOME_TOTAL
app_ext_removal['AMT_INCOME_TOTAL'].quantile([0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.99])
```

Out[65]:
```
0.10     81000.0
```

```
0.20     99000.0
0.30    112500.0
0.40    135000.0
0.50    147150.0
0.60    162000.0
0.70    180000.0
0.80    225000.0
0.90    270000.0
0.99    472500.0
Name: AMT_INCOME_TOTAL, dtype: float64
```

In [66]:
```
app_ext_removal['AMT_INCOME_TOTAL'].max()
```

Out[66]:
```
117000000.0
```

In [67]:
```
bins = [0,100000,150000,200000,250000,300000,350000,400000,117000000]
ranges = ['0-100K','100K-150K','150K-200K','200K-250K','250K-300K','300K-350K','350K-400K'
          ,'Above 400K']

app_ext_removal['AMT_INCOME_TOTAL_RANGE'] = pan.cut(app_ext_removal['AMT_INCOME_TOTAL'],bins,labels=ranges)
```

In [68]:
```
app_ext_removal.groupby(['AMT_INCOME_TOTAL_RANGE']).size()
```

Out[68]:
```
AMT_INCOME_TOTAL_RANGE
0-100K        63698
100K-150K     91591
150K-200K     64307
200K-250K     48137
250K-300K     17039
300K-350K      8874
350K-400K      5802
Above 400K     8063
dtype: int64
```

In [69]:
```
#for AMT_CREDIT
app_ext_removal['AMT_CREDIT'].quantile([0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.99])
```

Out[69]:
```
0.10     180000.0
0.20     254700.0
0.30     306306.0
0.40     432000.0
0.50     513531.0
0.60     604152.0
0.70     755190.0
0.80     900000.0
0.90    1133748.0
0.99    1854000.0
Name: AMT_CREDIT, dtype: float64
```

In [70]:
```
app_ext_removal['AMT_CREDIT'].max()
```

Out[70]:
```
4050000.0
```

In [71]:
```
bins = [0,200000,400000,600000,800000,900000,1000000,2000000,3000000,4050000]
ranges = ['0-200K','200K-400K','400K-600K','600K-800K','800K-900K','900K-1M','1M-2M','2M-3M','Above 3M']

app_ext_removal['AMT_CREDIT_RANGE'] = pan.cut(app_ext_removal['AMT_CREDIT'],bins,labels=ranges)
```

In [72]:
```
app_ext_removal.groupby(['AMT_CREDIT_RANGE']).size()
```

Out[72]:
```
AMT_CREDIT_RANGE
0-200K       36144
200K-400K    81151
400K-600K    66270
600K-800K    43242
800K-900K    21792
900K-1M       8927
1M-2M        47956
2M-3M         1997
Above 3M        32
dtype: int64
```

In [73]:
```
# for AMT_ANNUITy
app_ext_removal['AMT_ANNUITY'].quantile([0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.99])
```

Out[73]:
```
0.10    11074.5
0.20    14701.5
0.30    18189.0
0.40    21870.0
0.50    24903.0
0.60    28062.0
0.70    32004.0
0.80    37516.5
0.90    45954.0
0.99    70006.5
Name: AMT_ANNUITY, dtype: float64
```

In [74]:
```
app_ext_removal['AMT_ANNUITY'].max()
```

```
#len(num_all_dt)
#len(num_dt)
```

In [90]:
```
#we will make 2 subset , in which one will have all the defaulter and the other repayers as the data is imbalance
#as shown
num_data= app_ext_removal[num_dt]
num_data.groupby(['TARGET']).size()
```

Out[90]:
```
TARGET
0    282686
1     24825
dtype: int64
```

In [91]:
```
num_data.groupby(['TARGET']).size()/num_data.shape[0]*100
defaulters=num_data[num_data["TARGET"]==1].drop(["TARGET"],axis=1)
#defaulters.head()
repayers=num_data[num_data["TARGET"]==0].drop(["TARGET"],axis=1)
#repayers.head()
```

Out[91]:
```
TARGET
0    91.927118
1     8.072882
dtype: float64
```

In [92]:
```
# the data is in binary ( 0 and 1 )so we convert it in
```

```
#boolean (False and True)
```

AMT_INCOME_TOTAL	NaN	NaN	NaN	0.038131	0.046421	0.037591
AMT_CREDIT	NaN	NaN	NaN	NaN	0.752195	0.982783
AMT_ANNUITY	NaN	NaN	NaN	NaN	NaN	0.752295
AMT_GOODS_PRICE	NaN	NaN	NaN	NaN	NaN	NaN
REGION_POPULATION_RELATIVE	NaN	NaN	NaN	NaN	NaN	NaN
DAYS_BIRTH	NaN	NaN	NaN	NaN	NaN	NaN
DAYS_EMPLOYED	NaN	NaN	NaN	NaN	NaN	NaN
DAYS_REGISTRATION	NaN	NaN	NaN	NaN	NaN	NaN
DAYS_ID_PUBLISH	NaN	NaN	NaN	NaN	NaN	NaN
CNT_FAM_MEMBERS	NaN	NaN	NaN	NaN	NaN	NaN
REGION_RATING_CLIENT	NaN	NaN	NaN	NaN	NaN	NaN
REGION_RATING_CLIENT_W_CITY	NaN	NaN	NaN	NaN	NaN	NaN
HOUR_APPR_PROCESS_START	NaN	NaN	NaN	NaN	NaN	NaN
REG_REGION_NOT_LIVE_REGION	NaN	NaN	NaN	NaN	NaN	NaN
REG_REGION_NOT_WORK_REGION	NaN	NaN	NaN	NaN	NaN	NaN
LIVE_REGION_NOT_WORK_REGION	NaN	NaN	NaN	NaN	NaN	NaN
REG_CITY_NOT_LIVE_CITY	NaN	NaN	NaN	NaN	NaN	NaN
REG_CITY_NOT_WORK_CITY	NaN	NaN	NaN	NaN	NaN	NaN
LIVE_CITY_NOT_WORK_CITY	NaN	NaN	NaN	NaN	NaN	NaN

```
0.60    17220.0
0.70    18885.0
0.80    20474.0
0.81    20641.0
0.85    21316.0
0.90    22181.0
0.95    23204.0
0.99    24419.0
Name: DAYS_BIRTH, dtype: float64
```

In [84]:
```
app_ext_removal['DAYS_BIRTH'].min()
```

Out[84]:
```
7489
```

In [85]:
```
#univariate analysis: univariate analysis is just a condition or subset that your data falls into, think of it as a "category."
#We create Category based on Datatypes
app_ext_removal.dtypes.value_counts()
```

Out[85]:
```
float64     18
int64       15
object      10
category     1
category     1
category     1
category     1
category     1
dtype: int64
```

In [86]:
```
#object datatype
object_dt = app_ext_removal.select_dtypes(include=['object']).columns
object_dt
```

Out[86]:
```
Index(['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'NAME_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE', 'WEEKDAY_APPR_PROCESS_START',
       'ORGANIZATION_TYPE'],
      dtype='object')
```

In [87]:
```
mplot.figure(figsize=(30,60))


for i, var in enumerate(object_dt):

    nc_pct = app_ext_removal[[var,'TARGET']].groupby([var], as_index=False).mean().sort_values(by='TARGET',ascending=False)
    nc_pct['PCT'] = nc_pct['TARGET']*100


    mplot.subplot(10,2,i+i+1)#row,column and indexing position
    mplot.subplots_adjust(wspace=0.1,hspace=1)#for spacing
    sea.countplot(data=app_ext_removal,x=var,hue='TARGET')
    mplot.xticks(rotation=90)

    mplot.subplot(10,2,i+i+2)
    sea.barplot(data=nc_pct,x=var,y='PCT',palette='coolwarm')
    mplot.xticks(rotation=90)
```

Out[87]:
```
<AxesSubplot:xlabel='NAME_EDUCATION_TYPE', ylabel='PCT'>
```

Out[87]:
```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'Lower secondary'),
  Text(1, 0, 'Secondary / secondary special'),
  Text(2, 0, 'Incomplete higher'),
  Text(3, 0, 'Higher education'),
  Text(4, 0, 'Academic degree')])
```

Out[87]:
```
<AxesSubplot:>
```

Out[87]:
```
<AxesSubplot:xlabel='NAME_FAMILY_STATUS', ylabel='count'>
```

Out[87]:
```
(array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, 'Single / not married'),
  Text(1, 0, 'Married'),
  Text(2, 0, 'Civil marriage'),
  Text(3, 0, 'Widow'),
  Text(4, 0, 'Separated'),
  Text(5, 0, 'Unknown')])
```

In [88]:
```
# NAME_CONTRACT_TYPE: here we can observe that the poeple who took cash loan has higher no of defaulters, but as we observed abov
#people who took cash loan are compartively much larger than revolving loans so
#we can infer that people who are cash loans are less likely to be defaulter


#Code Gender: Female have taken more loans than male and are less likely to be defaulter , so they are more preffered to
#be given loan
```

```
#NAME_TYPE_SUITE: At the time of taking loan Unacompained people have taken the most loans and default rate is highter
#compared to the others
#len(num_all_dt)
#len(num_dt)
```

In [90]:
```
#we will make 2 subset , in which one will have all the defaulter and the other repayers as the data is imbalance
#as shown
num_data= app_ext_removal[num_dt]
num_data.groupby(['TARGET']).size()
```

Out[90]:
```
TARGET
0    282686
1     24825
dtype: int64
```

In [91]:
```
num_data.groupby(['TARGET']).size()/num_data.shape[0]*100
defaulters=num_data[num_data["TARGET"]==1].drop(["TARGET"],axis=1)
#defaulters.head()
repayers=num_data[num_data["TARGET"]==0].drop(["TARGET"],axis=1)
#repayers.head()
```

Out[91]:
```
TARGET
0    91.927118
1     8.072882
dtype: float64
```

In [92]:
```
# the data is in binary ( 0 and 1 )so we convert it in
#boolean (False and True)
```

```
0-200K       36144
200K-400K    81151
400K-600K    66270
600K-800K    43242
800K-900K    21792
900K-1M       8927
1M-2M        47956
2M-3M         1997
Above 3M        32
dtype: int64
```

In [73]:
```
# for AMT_ANNUITY
app_ext_removal['AMT_ANNUITY'].quantile([0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.99])
```

Out[73]:
```
0.10    11074.5
0.20    14701.5
0.30    18189.0
0.40    21870.0
0.50    24903.0
0.60    28062.0
0.70    32004.0
0.80    37516.5
0.90    45954.0
0.99    70006.5
Name: AMT_ANNUITY, dtype: float64
```

In [74]:
```
app_ext_removal['AMT_ANNUITY'].max()
```


```
#len(num_all_dt)
#len(num_dt)
```

In [90]:
```
#we will make 2 subset , in which one will have all the defaulter and the other repayers as the data is imbalance
#as shown
num_data= app_ext_removal[num_dt]
num_data.groupby(['TARGET']).size()
```

Out[90]:
```
TARGET
0    282686
1     24825
dtype: int64
```

In [91]:
```
num_data.groupby(['TARGET']).size()/num_data.shape[0]*100
defaulters=num_data[num_data["TARGET"]==1].drop(["TARGET"],axis=1)
#defaulters.head()
repayers=num_data[num_data["TARGET"]==0].drop(["TARGET"],axis=1)
#repayers.head()
```

Out[91]:
```
TARGET
0    91.927118
1     8.072882
dtype: float64
```

In [92]:
```
# the data is in binary ( 0 and 1 )so we convert it in
#boolean (False and True)
```


```
amount_var = ['AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','AMT_GOODS_PRICE']
```

In [101]:
```
#here we understand how Amount column affects default and repayers
mplot.figure(figsize=(10,5))

for i, col in enumerate(amount_var):
    mplot.subplot(2,2,i+1)
    sea.kdeplot(data=num_data,x=col,hue='TARGET')
    mplot.subplots_adjust(wspace=0.5,hspace=0.5)
```

Out[101]: <Figure size 720x360 with 0 Axes>

Out[101]: <AxesSubplot:>

Out[101]: <AxesSubplot:xlabel='AMT_INCOME_TOTAL', ylabel='Density'>

Out[101]: <AxesSubplot:>

Out[101]: <AxesSubplot:xlabel='AMT_CREDIT', ylabel='Density'>

Out[101]: <AxesSubplot:>

Out[101]: <AxesSubplot:xlabel='AMT_ANNUITY', ylabel='Density'>

Out[101]: <AxesSubplot:>

Out[101]: <AxesSubplot:xlabel='AMT_GOODS_PRICE', ylabel='Density'>

OBS_30_CNT_SOCIAL_CIRCLE	NaN	NaN	NaN	NaN	NaN	NaN
DEF_30_CNT_SOCIAL_CIRCLE	NaN	NaN	NaN	NaN	NaN	NaN
OBS_60_CNT_SOCIAL_CIRCLE	NaN	NaN	NaN	NaN	NaN	NaN
DEF_60_CNT_SOCIAL_CIRCLE	NaN	NaN	NaN	NaN	NaN	NaN
DAYS_LAST_PHONE_CHANGE	NaN	NaN	NaN	NaN	NaN	NaN
AMT_REQ_CREDIT_BUREAU_HOUR	NaN	NaN	NaN	NaN	NaN	NaN
AMT_REQ_CREDIT_BUREAU_DAY	NaN	NaN	NaN	NaN	NaN	NaN
AMT_REQ_CREDIT_BUREAU_WEEK	NaN	NaN	NaN	NaN	NaN	NaN
AMT_REQ_CREDIT_BUREAU_MON	NaN	NaN	NaN	NaN	NaN	NaN
AMT_REQ_CREDIT_BUREAU_QRT	NaN	NaN	NaN	NaN	NaN	NaN
AMT_REQ_CREDIT_BUREAU_YEAR	NaN	NaN	NaN	NaN	NaN	NaN

In [95]:
```
#we need to drop the nan values

defaulters_corr.where(np.triu(np.ones(defaulters_corr.shape),k=1).astype(np.bool)).unstack()
```

```
C:\Users\SANJEE~1\AppData\Local\Temp/ipykernel_33732/1689656856.py:3: DeprecationWarning: `np.bool` is a deprecated alias for t
he builtin `bool`. To silence this warning, use `bool` by itself. Doing this will not modify any behavior and is safe. If you s
pecifically wanted the numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  defaulters_corr.where(np.triu(np.ones(defaulters_corr.shape),k=1).astype(np.bool)).unstack()
```

Out[95]:
```
SK_ID_CURR                  SK_ID_CURR                  NaN
0.60    17220.0
0.70    18885.0
0.80    20474.0
0.81    20641.0
0.85    21316.0
0.90    22181.0
0.95    23204.0
0.99    24419.0
Name: DAYS_BIRTH, dtype: float64
```

In [84]:
```
app_ext_removal['DAYS_BIRTH'].min()
```

Out[84]:
```
7489
```

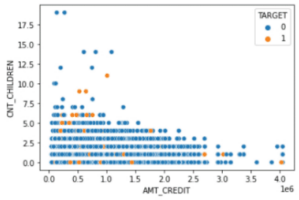
In [85]:
```
#univariate analysis: univariate analysis is just a condition or subset that your data falls into, think of it as a "category."
#We create Category based on Datatypes
app_ext_removal.dtypes.value_counts()
```

Out[85]:
```
float64     18
int64       15
object      10
category     1
category     1
category     1
category     1
category     1
dtype: int64
```

In [86]:
```
#object datatype
```

In [106]:
```
sea.scatterplot(data=num_data,x='AMT_CREDIT',y='CNT_CHILDREN',hue='TARGET')
```

Out[106]:
```
<AxesSubplot:xlabel='AMT_CREDIT', ylabel='CNT_CHILDREN'>
```



In [107]:
```
#if we consider no of children or no of family members, we observe
# a similar trend therefore we can infer that people having chidren/family member
# between 1 to 5 should be more preferable to be given loan
```

Out[87]:
```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'Lower secondary'),
  Text(1, 0, 'Secondary / secondary special'),
  Text(2, 0, 'Incomplete higher'),
  Text(3, 0, 'Higher education'),
  Text(4, 0, 'Academic degree')])
```

Out[87]:
```
<AxesSubplot:>
```

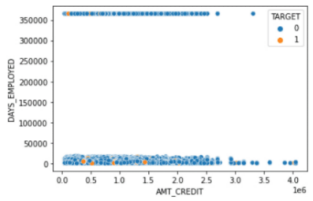
Out[87]:
```
<AxesSubplot:xlabel='NAME_FAMILY_STATUS', ylabel='count'>
```

Out[87]:
```
(array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, 'Single / not married'),
  Text(1, 0, 'Married'),
  Text(2, 0, 'Civil marriage'),
  Text(3, 0, 'Widow'),
  Text(4, 0, 'Separated'),
  Text(5, 0, 'Unknown')])
```

In [88]:
```
# NAME_CONTRACT_TYPE: here we can observe that the poeple who took cash loan has higher no of defaulters, but as we observed abov
#people who took cash loan are compartively much larger than revolving loans so
#we can infer that people who are cash loans are less likely to be defaulter


#Code Gender: Female have taken more loans than male and are less likely to be defaulter , so they are more preffered to
#be given loan

#NAME_TYPE_SUITE: At the time of taking loan Unacompained people have taken the most loans and default rate is highter
#compared to the others
```

In [108]:
```
sea.scatterplot(data=num_data,x='AMT_CREDIT',y='DAYS_EMPLOYED',hue='TARGET')
```

Out[108]:
```
<AxesSubplot:xlabel='AMT_CREDIT', ylabel='DAYS_EMPLOYED'>
```



In [109]:
```
# As we observe that people working less than 50000 days and earning more than
#2 million or people working more than 350000 and earning more than 1 million
# are more likely to get a loan
```

In [110]:
```
amount_var = num_data[['AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','AMT_GOODS_PRICE','TARGET']]
```

In [111]:
```
sea.pairplot(data=amount_var,hue='TARGET')
```

```
#len(num_all_dt)
#len(num_dt)
```

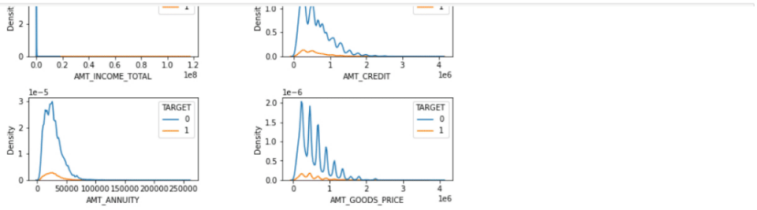
In [90]:
```
#we will make 2 subset , in which one will have all the defaulter and the other repayers as the data is imbalance
#as shown
num_data= app_ext_removal[num_dt]
num_data.groupby(['TARGET']).size()
```

Out[90]:
```
TARGET
0    282686
1     24825
dtype: int64
```

In [91]:
```
num_data.groupby(['TARGET']).size()/num_data.shape[0]*100
defaulters=num_data[num_data["TARGET"]==1].drop(["TARGET"],axis=1)
#defaulters.head()
repayers=num_data[num_data["TARGET"]==0].drop(["TARGET"],axis=1)
#repayers.head()
```

Out[91]:
```
TARGET
0    91.927118
1     8.072882
dtype: float64
```

In [92]:
```
# the data is in binary ( 0 and 1 )so we convert it in
#boolean (False and True)
```



In [102]:
```
#univariate numeric variables analysis
# as we can see the trend does not have any trends that are unsual to concude that it factors
# in terms of Defaulters or repayers
# AMT_INCOME_TOTAL: customers have income between range of 0 to 1 million
#AMT_CREDIT: Loan given for credit amount ranges from 0 to 1 million
#AMT_ANNUITY: Costomers paying annuity ranges from  0 to 50k
#AMT_GOODS_PRICE : Loan given for goods price ranges from 0 to 1 million
```

In [103]:
```
num_data.head()
```

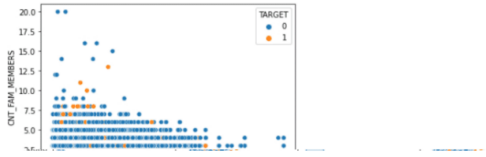
Out[103]:

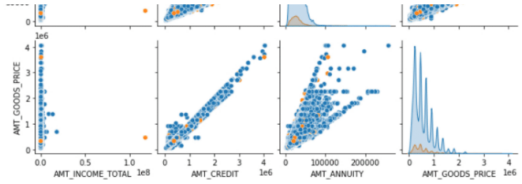
	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_POPULATION_RELATIVE	DAYS_E
0	100002	1	0	202500.0	406597.5	24700.5	351000.0	0.018801	
1	100003	0	0	270000.0	1293502.5	35698.5	1129500.0	0.003541	
2	100004	0	0	67500.0	135000.0	6750.0	135000.0	0.010032	
3	100006	0	0	135000.0	312682.5	29686.5	297000.0	0.008019	
4	100007	0	0	121500.0	513000.0	21865.5	513000.0	0.028663	

In [104]:
```
# bivariate analysis : is a statistical method examining how two different things are related.
#The bivariate analysis aims to determine if there is a statistical link
#between the two variables
```

In [105]:
```
sea.scatterplot(data=num_data,x='AMT_CREDIT',y='CNT_FAM_MEMBERS',hue='TARGET')
```

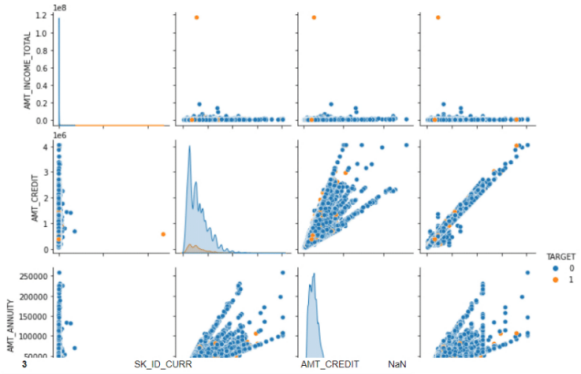
Out[105]:
```
<AxesSubplot:xlabel='AMT_CREDIT', ylabel='CNT_FAM_MEMBERS'>
```





In [112]:
```
#we can infer that amount credit and amount good price are  in linear correation
# as we see the denisty of defaulter decreases as the amount increases
#so we can infer people with higher loan amont are less likely to be defaulters
# a trend can be observed in amount annuity and amount credit,amount good price
#and amount annunity
#people that pay annunity in range 100k-2million are highly likely of getting a loan
#
#people earning less than million are more likely to opt for loan
#among those people opting for loan less than 1.5 million are more likely
#to be defaulters

#this gives us the target popluation of people earning less than 1million
#in need on loan more than 1.5 million
```

Out[111]: <seaborn.axisgrid.PairGrid at 0x1e99ce45dc0>



3	SK_ID_CURR	AMT_CREDIT	NaN
4	SK_ID_CURR	AMT_ANNUITY	NaN
...	...	...	...
1019	AMT_REQ_CREDIT_BUREAU_YEAR	AMT_REQ_CREDIT_BUREAU_DAY	0.007970
1020	AMT_REQ_CREDIT_BUREAU_YEAR	AMT_REQ_CREDIT_BUREAU_WEEK	0.031736
1021	AMT_REQ_CREDIT_BUREAU_YEAR	AMT_REQ_CREDIT_BUREAU_MON	0.024771
1022	AMT_REQ_CREDIT_BUREAU_YEAR	AMT_REQ_CREDIT_BUREAU_QRT	0.133739
1023	AMT_REQ_CREDIT_BUREAU_YEAR	AMT_REQ_CREDIT_BUREAU_YEAR	NaN

1024 rows × 3 columns

In [97]:
```
#it has negative values too,so by using absolute value
defaulter_corr_new['corr']=abs(defaulter_corr_new['corr'])
# and finding top 10 correations
defaulter_corr_new.dropna(subset=['corr']).sort_values(by=['corr'],ascending=False).head(10)
```

Out[97]:

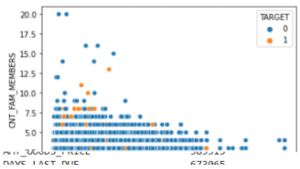
	var1	var2	corr
757	OBS_60_CNT_SOCIAL_CIRCLE	OBS_30_CNT_SOCIAL_CIRCLE	0.998269
163	AMT_GOODS_PRICE	AMT_CREDIT	0.982783
428	REGION_RATING_CLIENT_W_CITY	REGION_RATING_CLIENT	0.956637
353	CNT_FAM_MEMBERS	CNT_CHILDREN	0.885484
790	DEF_60_CNT_SOCIAL_CIRCLE	DEF_30_CNT_SOCIAL_CIRCLE	0.868994

0	100002	1	0	202500.0	406597.5	24700.5	351000.0	0.018801
1	100003	0	0	270000.0	1293502.5	35698.5	1129500.0	0.003541
2	100004	0	0	67500.0	135000.0	6750.0	135000.0	0.010032
3	100006	0	0	135000.0	312682.5	29686.5	297000.0	0.008019
4	100007	0	0	121500.0	513000.0	21865.5	513000.0	0.028663

In [104]:
```
# bivariate analysis : is a statistical method examining how two different things are related.
#The bivariate analysis aims to determine if there is a statistical link
#between the two variables
```

In [105]:
```
sea.scatterplot(data=num_data,x='AMT_CREDIT',y='CNT_FAM_MEMBERS',hue='TARGET')
```

Out[105]: <AxesSubplot:xlabel='AMT_CREDIT', ylabel='CNT_FAM_MEMBERS'>



```
DAYS_LAST_DUE_1ST_VERSION       673065
DAYS_FIRST_DUE                  673065
DAYS_FIRST_DRAWING              673065
NFLAG_INSURED_ON_APPROVAL       673065
DAYS_TERMINATION                673065
NAME_TYPE_SUITE                 820405
AMT_DOWN_PAYMENT                895844
RATE_DOWN_PAYMENT               895844
RATE_INTEREST_PRIMARY          1664263
RATE_INTEREST_PRIVILEGED       1664263
dtype: int64
```

In [117]:
```
null_count = pan.DataFrame(prev_app.isnull().sum().sort_values(ascending=False)/prev_app.shape[0]*100).reset_index().rename(colum
null_count
```

Out[117]:

	column	missing_percentage
0	RATE_INTEREST_PRIVILEGED	99.643698
1	RATE_INTEREST_PRIMARY	99.643698
2	AMT_DOWN_PAYMENT	53.636480
3	RATE_DOWN_PAYMENT	53.636480
4	NAME_TYPE_SUITE	49.119754
5	NFLAG_INSURED_ON_APPROVAL	40.298129

```
amount_var = ['AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','AMT_GOODS_PRICE']
```

In [101]:
```
#here we understand how Amount column affects default and repayers
mplot.figure(figsize=(10,5))

for i, col in enumerate(amount_var):
    mplot.subplot(2,2,i+1)
    sea.kdeplot(data=num_data,x=col,hue='TARGET')
    mplot.subplots_adjust(wspace=0.5,hspace=0.5)
```

Out[101]: <Figure size 720x360 with 0 Axes>

Out[101]: <AxesSubplot:>

Out[101]: <AxesSubplot:xlabel='AMT_INCOME_TOTAL', ylabel='Density'>

Out[101]: <AxesSubplot:>

Out[101]: <AxesSubplot:xlabel='AMT_CREDIT', ylabel='Density'>

Out[101]: <AxesSubplot:>

Out[101]: <AxesSubplot:xlabel='AMT_ANNUITY', ylabel='Density'>

Out[101]: <AxesSubplot:>

Out[101]: <AxesSubplot:xlabel='AMT_GOODS_PRICE', ylabel='Density'>

6	DAYS_TERMINATION	40.298129
7	DAYS_LAST_DUE	40.298129
8	DAYS_LAST_DUE_1ST_VERSION	40.298129
9	DAYS_FIRST_DUE	40.298129
10	DAYS_FIRST_DRAWING	40.298129
11	AMT_GOODS_PRICE	23.081773
12	AMT_ANNUITY	22.286665
13	CNT_PAYMENT	22.286366
14	PRODUCT_COMBINATION	0.020716
15	AMT_CREDIT	0.000060
16	NAME_YIELD_GROUP	0.000000
17	NAME_PORTFOLIO	0.000000
18	NAME_SELLER_INDUSTRY	0.000000
19	SELLERPLACE_AREA	0.000000
20	CHANNEL_TYPE	0.000000
21	NAME_PRODUCT_TYPE	0.000000
22	SK_ID_PREV	0.000000
23	NAME_GOODS_CATEGORY	0.000000
24	NAME_CLIENT_TYPE	0.000000
25	CODE_REJECT_REASON	0.000000
26	SK_ID_CURR	0.000000
27	DAYS_DECISION	0.000000
28	NAME_CONTRACT_STATUS	0.000000
29	NAME_CASH_LOAN_PURPOSE	0.000000
30	NFLAG_LAST_APPL_IN_DAY	0.000000
31	FLAG_LAST_APPL_PER_CONTRACT	0.000000
32	HOUR_APPR_PROCESS_START	0.000000
33	WEEKDAY_APPR_PROCESS_START	0.000000
34	AMT_APPLICATION	0.000000
35	NAME_CONTRACT_TYPE	0.000000
36	NAME_PAYMENT_TYPE	0.000000

In [118]:
```
#we are creating excel to understand columns and remove those column
#which will not be required for analysis
null_count.to_excel(r"C:\Users\Sanjeev Kumar Singh\Desktop\Aayushi\projects\Trainity\Bank Loan Case Study\missing_info_prev_app.x
```

In [119]:
```
len(prev_app.columns)
```

Out[119]: 37

In [120]:
```
#as in prev dataset we removed all the percentage above 40
```

25	CODE_REJECT_REASON	0.000000
26	SK_ID_CURR	0.000000

27	DAYS_DECISION	0.000000
28	NAME_CONTRACT_STATUS	0.000000
29	NAME_CASH_LOAN_PURPOSE	0.000000
30	NFLAG_LAST_APPL_IN_DAY	0.000000
31	FLAG_LAST_APPL_PER_CONTRACT	0.000000
32	HOUR_APPR_PROCESS_START	0.000000
33	WEEKDAY_APPR_PROCESS_START	0.000000
34	AMT_APPLICATION	0.000000
35	NAME_CONTRACT_TYPE	0.000000
36	NAME_PAYMENT_TYPE	0.000000

In [118]:
```
#we are creating excel to understand columns and remove those column
#which will not be required for analysis
null_count.to_excel(r"C:\Users\Sanjeev Kumar Singh\Desktop\Aayushi\projects\Trainity\Bank Loan Case Study\missing_info_prev_app.x
```

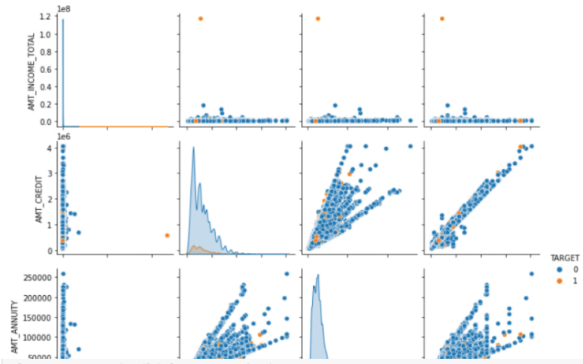
In [119]:
```
len(prev_app.columns)
```

Out[119]: 37

In [120]:
```
#as in prev_dataset_we_removed all the percentage above 40
```

6	DAYS_TERMINATION	40.298129
7	DAYS_LAST_DUE	40.298129
8	DAYS_LAST_DUE_1ST_VERSION	40.298129
9	DAYS_FIRST_DUE	40.298129
10	DAYS_FIRST_DRAWING	40.298129
11	AMT_GOODS_PRICE	23.081773
12	AMT_ANNUITY	22.286665
13	CNT_PAYMENT	22.286366
14	PRODUCT_COMBINATION	0.020716
15	AMT_CREDIT	0.000060
16	NAME_YIELD_GROUP	0.000000
17	NAME_PORTFOLIO	0.000000
18	NAME_SELLER_INDUSTRY	0.000000
19	SELLERPLACE_AREA	0.000000
20	CHANNEL_TYPE	0.000000
21	NAME_PRODUCT_TYPE	0.000000
22	SK_ID_PREV	0.000000
23	NAME_GOODS_CATEGORY	0.000000
24	NAME_CLIENT_TYPE	0.000000

Out[111]: <seaborn.axisgrid.PairGrid at 0x1e99ce45dc0>



In [123]:
```
col_rem = prev_app.drop(labels=new_rem,axis=1)


len(col_rem.columns)
col_rem.columns
```

Out[123]: 22

Out[123]:
```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_ANNUITY',
       'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_GOODS_PRICE',
       'NAME_CASH_LOAN_PURPOSE', 'NAME_CONTRACT_STATUS', 'DAYS_DECISION',
       'NAME_PAYMENT_TYPE', 'CODE_REJECT_REASON', 'NAME_CLIENT_TYPE',
       'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO', 'NAME_PRODUCT_TYPE',
       'CHANNEL_TYPE', 'SELLERPLACE_AREA', 'NAME_SELLER_INDUSTRY',
       'CNT_PAYMENT', 'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION'],
      dtype='object')
```

In [124]:
```
col_rem.head()
```

Out[124]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_GOODS_PRICE	NAME_CASH_LOAN_PURPOSE
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	17145.0	XAP
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	607500.0	XNA
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	112500.0	XNA
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	450000.0	XNA

```
sea.kdeplot(data=col_rem,x=col)
mplot.subplots_adjust(wspace=0.5,hspace=0.5)
```

Out[131]: <Figure size 720x360 with 0 Axes>

Out[131]: <AxesSubplot:>

Out[131]: <AxesSubplot:xlabel='AMT_GOODS_PRICE', ylabel='Density'>

Out[131]: <AxesSubplot:>

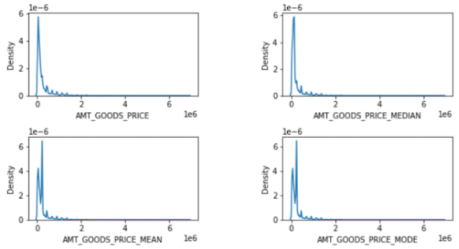
Out[131]: <AxesSubplot:xlabel='AMT_GOODS_PRICE_MEDIAN', ylabel='Density'>

Out[131]: <AxesSubplot:>

Out[131]: <AxesSubplot:xlabel='AMT_GOODS_PRICE_MEAN', ylabel='Density'>

Out[131]: <AxesSubplot:>

Out[131]: <AxesSubplot:xlabel='AMT_GOODS_PRICE_MODE', ylabel='Density'>



In [132]:
```
#as we can observe AMT_GOODS_PRICE_MEDIAN is similar to AMT_GOODS_PRICE
col_rem['AMT_GOODS_PRICE'] = col_rem['AMT_GOODS_PRICE'].fillna(col_rem['AMT_GOODS_PRICE'].median())
```

In [133]:
```
col_rem.isnull().sum().sort_values(ascending=False)/col_rem.shape[0]*100
```

Out[133]:
```
AMT_ANNUITY               22.286665
CNT_PAYMENT               22.286366
PRODUCT_COMBINATION        0.020716
AMT_CREDIT                 0.000060
SK_ID_PREV                 0.000000
NAME_PORTFOLIO             0.000000
AMT_GOODS_PRICE_MEDIAN     0.000000
AMT_GOODS_PRICE_MEAN       0.000000
NAME_YIELD_GROUP           0.000000
NAME_SELLER_INDUSTRY       0.000000
SELLERPLACE_AREA           0.000000
CHANNEL_TYPE               0.000000
NAME_PRODUCT_TYPE          0.000000
NAME_CLIENT_TYPE           0.000000
NAME_GOODS_CATEGORY        0.000000
SK_ID_CURR                 0.000000
CODE_REJECT_REASON         0.000000
NAME_PAYMENT_TYPE          0.000000
DAYS_DECISION              0.000000
NAME_CONTRACT_STATUS       0.000000
NAME_CASH_LOAN_PURPOSE     0.000000
AMT_GOODS_PRICE            0.000000
AMT_APPLICATION            0.000000
NAME_CONTRACT_TYPE         0.000000
AMT_GOODS_PRICE_MODE       0.000000
dtype: float64
```

In [134]:
```
# droping extra amount columns of men median mode that we created
col_rem = col_rem.drop(labels=['AMT_GOODS_PRICE_MEDIAN','AMT_GOODS_PRICE_MEAN','AMT_GOODS_PRICE_MODE'],axis=1)
```

In [135]:
```
#for AMT_ANNUITY
col_rem['AMT_ANNUITY'].agg(func=['mean','median','max'])
```

Out[135]:
```
mean       15955.120659
median     11250.000000
max       418058.145000
Name: AMT_ANNUITY, dtype: float64
```

In [136]:
```
#mean and median has near to values so either can be used, for easier analysis median is preferred
col_rem['AMT_ANNUITY'] =col_rem['AMT_ANNUITY'].fillna(col_rem['AMT_ANNUITY'].median())
```

```
DAYS_LAST_DUE                673065
DAYS_LAST_DUE_1ST_VERSION    673065
DAYS_FIRST_DUE               673065
DAYS_FIRST_DRAWING           673065
NFLAG_INSURED_ON_APPROVAL    673065
DAYS_TERMINATION             673065
NAME_TYPE_SUITE              820405
AMT_DOWN_PAYMENT             895844
RATE_DOWN_PAYMENT            895844
RATE_INTEREST_PRIMARY       1664263
RATE_INTEREST_PRIVILEGED    1664263
dtype: int64
```

In [117]:
```
null_count = pan.DataFrame(prev_app.isnull().sum().sort_values(ascending=False)/prev_app.shape[0]*100).reset_index().rename(colum
null_count
```

Out[117]:

	column	missing_percentage
0	RATE_INTEREST_PRIVILEGED	99.643698
1	RATE_INTEREST_PRIMARY	99.643698
2	AMT_DOWN_PAYMENT	53.636480
3	RATE_DOWN_PAYMENT	53.636480
4	NAME_TYPE_SUITE	49.119754
5	NFLAG_INSURED_ON_APPROVAL	40.298129

```
sea.kdeplot(data=col_rem,x=col)
mplot.subplots_adjust(wspace=0.5,hspace=0.5)
```

Out[131]: <Figure size 720x360 with 0 Axes>

Out[131]: <AxesSubplot:>

Out[131]: <AxesSubplot:xlabel='AMT_GOODS_PRICE', ylabel='Density'>

Out[131]: <AxesSubplot:>

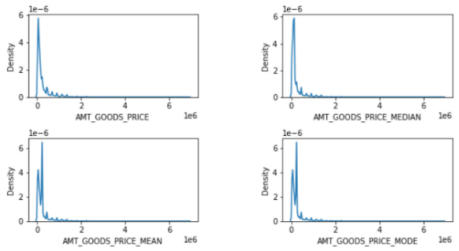
Out[131]: <AxesSubplot:xlabel='AMT_GOODS_PRICE_MEDIAN', ylabel='Density'>

Out[131]: <AxesSubplot:>

Out[131]: <AxesSubplot:xlabel='AMT_GOODS_PRICE_MEAN', ylabel='Density'>

Out[131]: <AxesSubplot:>

Out[131]: <AxesSubplot:xlabel='AMT_GOODS_PRICE_MODE', ylabel='Density'>



In [132]:
```
#as we can observe AMT_GOODS_PRICE_MEDIAN is similar to AMT_GOODS_PRICE
col_rem['AMT_GOODS_PRICE'] = col_rem['AMT_GOODS_PRICE'].fillna(col_rem['AMT_GOODS_PRICE'].median())
```

In [133]:
```
col_rem.isnull().sum().sort_values(ascending=False)/col_rem.shape[0]*100
```

Out[133]:
```
AMT_ANNUITY               22.286665
CNT_PAYMENT               22.286366
PRODUCT_COMBINATION        0.020716
AMT_CREDIT                 0.000060
SK_ID_PREV                 0.000000
NAME_PORTFOLIO             0.000000
AMT_GOODS_PRICE_MEDIAN     0.000000
AMT_GOODS_PRICE_MEAN       0.000000
NAME_YIELD_GROUP           0.000000
NAME_SELLER_INDUSTRY       0.000000
SELLERPLACE_AREA           0.000000
CHANNEL_TYPE               0.000000
NAME_PRODUCT_TYPE          0.000000
```

In [137]:
```
col_rem.isnull().sum().sort_values(ascending=False)/col_rem.shape[0]*100
```

Out[137]:
```
CNT_PAYMENT               22.286366
PRODUCT_COMBINATION        0.020716
AMT_CREDIT                 0.000060
NAME_CLIENT_TYPE           0.000000
NAME_YIELD_GROUP           0.000000
NAME_SELLER_INDUSTRY       0.000000
SELLERPLACE_AREA           0.000000
CHANNEL_TYPE               0.000000
NAME_PRODUCT_TYPE          0.000000
NAME_PORTFOLIO             0.000000
NAME_GOODS_CATEGORY        0.000000
SK_ID_PREV                 0.000000
SK_ID_CURR                 0.000000
NAME_PAYMENT_TYPE          0.000000
DAYS_DECISION              0.000000
NAME_CONTRACT_STATUS       0.000000
NAME_CASH_LOAN_PURPOSE     0.000000
AMT_GOODS_PRICE            0.000000
AMT_APPLICATION            0.000000
AMT_ANNUITY                0.000000
NAME_CONTRACT_TYPE         0.000000
CODE_REJECT_REASON         0.000000
dtype: float64
```

In [138]:
```
#for CNT_PAYMENT
```

In [137]:
```
col_rem.isnull().sum().sort_values(ascending=False)/col_rem.shape[0]*100
```

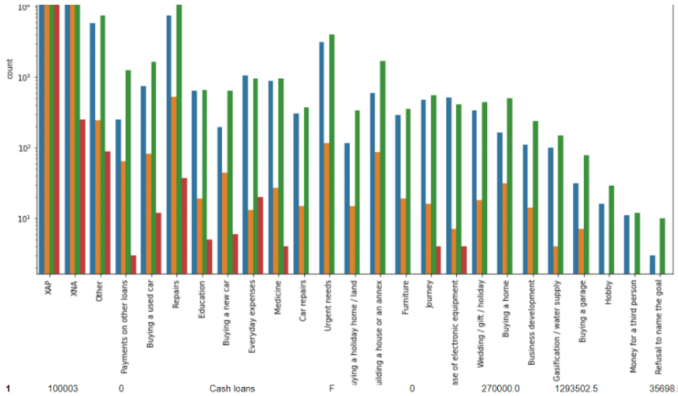
Out[137]:
```
CNT_PAYMENT               22.286366
PRODUCT_COMBINATION        0.020716
AMT_CREDIT                 0.000060
NAME_CLIENT_TYPE           0.000000
NAME_YIELD_GROUP           0.000000
NAME_SELLER_INDUSTRY       0.000000
SELLERPLACE_AREA           0.000000
CHANNEL_TYPE               0.000000
NAME_PRODUCT_TYPE          0.000000
NAME_PORTFOLIO             0.000000
NAME_GOODS_CATEGORY        0.000000
SK_ID_PREV                 0.000000
SK_ID_CURR                 0.000000
NAME_PAYMENT_TYPE          0.000000
DAYS_DECISION              0.000000
NAME_CONTRACT_STATUS       0.000000
NAME_CASH_LOAN_PURPOSE     0.000000
AMT_GOODS_PRICE            0.000000
AMT_APPLICATION            0.000000
AMT_ANNUITY                0.000000
NAME_CONTRACT_TYPE         0.000000
CODE_REJECT_REASON         0.000000
dtype: float64
```

In [138]:
```
#for CNT_PAYMENT
```

4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	337500.0	Repairs

In [125]:
```
col_rem.isnull().sum().sort_values(ascending=False)/col_rem.shape[0]*100
```

```
AMT_GOODS_PRICE              23.081773
AMT_ANNUITY                  22.286665
CNT_PAYMENT                  22.286366
PRODUCT_COMBINATION           0.020716
AMT_CREDIT                    0.000060
NAME_GOODS_CATEGORY           0.000000
NAME_YIELD_GROUP              0.000000
NAME_SELLER_INDUSTRY          0.000000
SELLERPLACE_AREA              0.000000
CHANNEL_TYPE                  0.000000
NAME_PRODUCT_TYPE             0.000000
NAME_PORTFOLIO                0.000000
SK_ID_PREV                    0.000000
NAME_CLIENT_TYPE              0.000000
SK_ID_CURR                    0.000000
NAME_PAYMENT_TYPE             0.000000
DAYS_DECISION                 0.000000
NAME_CONTRACT_STATUS          0.000000
NAME_CASH_LOAN_PURPOSE        0.000000
AMT_APPLICATION               0.000000
NAME_CONTRACT_TYPE            0.000000
CODE_REJECT_REASON            0.000000
dtype: float64
```



1	100003	0	Cash loans	F	0	270000.0	1293502.5	35698.5
2	100003	0	Cash loans	F	0	270000.0	1293502.5	35698.5
3	100003	0	Cash loans	F	0	270000.0	1293502.5	35698.5
4	100004	0	Revolving loans	M	0	67500.0	135000.0	6750.0

In [149]:
```
merged_data.columns
```

Out[149]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE_x', 'CODE_GENDER',
       'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT_x', 'AMT_ANNUITY_x',
       'AMT_GOODS_PRICE_x', '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', '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', 'AMT_GOODS_PRICE_BANGE',
```

```
merged_agg2
```

Out[154]:

	NAME_CONTRACT_STATUS	TARGET	counts_x	counts_y	pct
0	Approved	0	818856	886099	92.41
1	Approved	1	67243	886099	7.59
2	Canceled	0	235641	259441	90.83
3	Canceled	1	23800	259441	9.17
4	Refused	0	215952	245390	88.00
5	Refused	1	29438	245390	12.00
6	Unused offer	0	20892	22771	91.75
7	Unused offer	1	1879	22771	8.25

In [155]:
```
#most of the application that were periously either cancelled or refused
#in current, 80-90 percent of them are repayers
#Offers that were refused previously now have the high portion of defaulter
#despite the high income index
#The loan taken in the name of XNA has high rate of approval and highest cancellation
#but low rate of unused, as compared to XAP which has Highest rate of Approval
#and also has highest no of unused offers
#since the we dont know XAP or XNA stands for if we observe without considering them
#repairs has the highest number of approved and canceled
#others had largest number of unused offers
```

In [156]:
```
# FOR FINAL CONCULSION AND OBSERVATION
# summersizing all the observation we made during our analysis and based on it
#drawing conculsion

#With Respect To bank They should target customers
#Working in accountants,core staff,managers and laboreres
```

```
#Bussiness Entity type 3 , XNA, self employed and others
#having an income below 1 million
#consisting of family members between 2 to 5
#A Indiviual owning his own apartment, married and children no more than 5 are more preferable
#if in factor people having higher education are less likely to be defaulters
#As in for Gender, Female opting for loan has less chances of being a defaulter than men
#Credit amount of the loan must be less than 1 million
#annunity amount depending on the eligibilty must fall in range of 50k
# Customers who are previously cancelled or refused are mostly(in about80 to 90%)

#Low Skill workers and drivers should be avioded
#Transport type 3 has very low loan taken had highest default rate
#total employement days does not affects the defaulters list
#families in civil marriages should be avoided
#the indivisuals that were in high income brakett and previously unused the loan offered
#action or steps should pe taken in that parameter as they are more likely to be defaulters
#for the current dataset
#Applicants living with their parents or in rented apartment have higher rate of default.

# are current repayers, therefore before taking the decision of Refused,cancelled
#or apporved, the bank should consider these parameters as well
```


```
#Bussiness Entity type 3 , XNA, self employed and others
#having an income below 1 million
#consisting of family members between 2 to 5
#A Indiviual owning his own apartment, married and children no more than 5 are more preferable
#if in factor people having higher education are less likely to be defaulters
#As in for Gender, Female opting for loan has less chances of being a defaulter than men
#Credit amount of the loan must be less than 1 million
#annunity amount depending on the eligibilty must fall in range of 50k
# Customers who are previously cancelled or refused are mostly(in about80 to 90%)

#Low Skill workers and drivers should be avioded
#Transport type 3 has very low loan taken had highest default rate
#total employement days does not affects the defaulters list
#families in civil marriages should be avoided
#the indivisuals that were in high income brakett and previously unused the loan offered
#action or steps should pe taken in that parameter as they are more likely to be defaulters
#for the current dataset
#Applicants living with their parents or in rented apartment have higher rate of default.

# are current repayers, therefore before taking the decision of Refused,cancelled
#or apporved, the bank should consider these parameters as well
```


```
col_rem['CNT_PAYMENT'].agg(['mean', 'median', 'max'])
```

Out[138]:
```
mean      16.054082
median    12.000000
max       84.000000
Name: CNT_PAYMENT, dtype: float64
```

In [139]:
```
#as we cannot fill the data based on mean or median as it will very derivate data very much
col_rem[col_rem['CNT_PAYMENT'].isnull()].groupby(['NAME_CONTRACT_STATUS']).size().sort_values(ascending=False)
```

Out[139]:
```
NAME_CONTRACT_STATUS
Canceled        305805
Refused          40897
Unused offer     25524
Approved             4
dtype: int64
```

In [140]:
```
col_rem['CNT_PAYMENT'] = col_rem['CNT_PAYMENT'].fillna(0)
```

In [141]:
```
#for PRODUCT_COMBINATION
col_rem['PRODUCT_COMBINATION'].head()
```

Out[141]:
```
0    POS mobile with interest
1              Cash X-Sell: low
2             Cash X-Sell: high
3           Cash X-Sell: middle
4             Cash Street: high
```

1	100003	0	Cash loans	F	0	270000.0	1293502.5	35698.5
2	100003	0	Cash loans	F	0	270000.0	1293502.5	35698.5
3	100003	0	Cash loans	F	0	270000.0	1293502.5	35698.5
4	100004	0	Revolving loans	M	0	67500.0	135000.0	6750.0

In [149]:
```
merged_data.columns
```

Out[149]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE_x', 'CODE_GENDER',
       'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT_x', 'AMT_ANNUITY_x',
       'AMT_GOODS_PRICE_x', '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', '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', 'AMT_GOODS_PRICE_RANGE',
```

```
merged_agg2
```

Out[154]:

	NAME_CONTRACT_STATUS	TARGET	counts_x	counts_y	pct
0	Approved	0	818856	886099	92.41

1	Approved	1	67243	88009	7.59
2	Canceled	0	235641	259441	90.83
3	Canceled	1	23800	259441	9.17
4	Refused	0	215952	245390	88.00
5	Refused	1	29438	245390	12.00
6	Unused offer	0	20892	22771	91.75
7	Unused offer	1	1879	22771	8.25

In [155]:
```
#most of the application that were periously either cancelled or refused
#in current, 80-90 percent of them are repayers
#Offers that were refused previously now have the high portion of defaulter
#despite the high income index
#The loan taken in the name of XNA has high rate of approval and highest cancellation
#but low rate of unused, as compared to XAP which has Highest rate of Approval
#and also has highest no of unused offers
#since the we dont know XAP or XNA stands for if we observe without considering them
#repairs has the highest number of approved and canceled
#others had largest number of unused offers
```

```
NAME_CLIENT_TYPE            0.000000
NAME_GOODS_CATEGORY         0.000000
SK_ID_CURR                  0.000000
CODE_REJECT_REASON          0.000000
NAME_PAYMENT_TYPE           0.000000
DAYS_DECISION               0.000000
NAME_CONTRACT_STATUS        0.000000
NAME_CASH_LOAN_PURPOSE      0.000000
AMT_GOODS_PRICE             0.000000
AMT_APPLICATION             0.000000
NAME_CONTRACT_TYPE          0.000000
AMT_GOODS_PRICE_MODE        0.000000
dtype: float64
```

In [134]:
```
# droping extra amount columns of men median mode that we created
col_rem = col_rem.drop(labels=['AMT_GOODS_PRICE_MEDIAN','AMT_GOODS_PRICE_MEAN','AMT_GOODS_PRICE_MODE'],axis=1)
```

In [135]:
```
#for AMT_ANNUITY
col_rem['AMT_ANNUITY'].agg(func=['mean','median','max'])
```

Out[135]:
```
mean       15955.120659
median     11250.000000
max       418058.145000
Name: AMT_ANNUITY, dtype: float64
```

In [136]:
```
#mean and median has near to values so either can be used, for easier analysis median is preferred
col_rem['AMT_ANNUITY'] =col_rem['AMT_ANNUITY'].fillna(col_rem['AMT_ANNUITY'].median())
```

In [137]:
```
col_rem.isnull().sum().sort_values(ascending=False)/col_rem.shape[0]*100
```

Out[137]:
```
CNT_PAYMENT               22.286366
PRODUCT_COMBINATION        0.020716
AMT_CREDIT                 0.000060
NAME_CLIENT_TYPE           0.000000
NAME_YIELD_GROUP           0.000000
NAME_SELLER_INDUSTRY       0.000000
SELLERPLACE_AREA           0.000000
CHANNEL_TYPE               0.000000
NAME_PRODUCT_TYPE          0.000000
NAME_PORTFOLIO             0.000000
NAME_GOODS_CATEGORY        0.000000
SK_ID_PREV                 0.000000
SK_ID_CURR                 0.000000
NAME_PAYMENT_TYPE          0.000000
DAYS_DECISION              0.000000
NAME_CONTRACT_STATUS       0.000000
NAME_CASH_LOAN_PURPOSE     0.000000
AMT_GOODS_PRICE            0.000000
AMT_APPLICATION            0.000000
AMT_ANNUITY                0.000000
NAME_CONTRACT_TYPE         0.000000
CODE_REJECT_REASON         0.000000
dtype: float64
```

In [138]:
```
#for CNT_PAYMENT
  Text(8, 0, 'Everyday expenses'),
  Text(9, 0, 'Medicine'),
  Text(10, 0, 'Car repairs'),
  Text(11, 0, 'Urgent needs'),
  Text(12, 0, 'Buying a holiday home / land'),
  Text(13, 0, 'Building a house or an annex'),
  Text(14, 0, 'Furniture'),
  Text(15, 0, 'Journey'),
  Text(16, 0, 'Purchase of electronic equipment'),
  Text(17, 0, 'Wedding / gift / holiday'),
  Text(18, 0, 'Buying a home'),
  Text(19, 0, 'Business development'),
  Text(20, 0, 'Gasification / water supply'),
  Text(21, 0, 'Buying a garage'),
  Text(22, 0, 'Hobby'),
  Text(23, 0, 'Money for a third person'),
  Text(24, 0, 'Refusal to name the goal')])
```


```
col_rem['CNT_PAYMENT'].agg(func=['mean','median','max'])
```

Out[138]:
```
mean       16.054082
median     12.000000
max        84.000000
Name: CNT_PAYMENT, dtype: float64
```

In [139]:
```
#as we cannot fill the data based on means or medians as it will very deviate data very much
```

In [139]:
```
#as we cannot fill the data based on mean or median as it will very deflate data very much
col_rem[col_rem['CNT_PAYMENT'].isnull()].groupby(['NAME_CONTRACT_STATUS']).size().sort_values(ascending=False)
```

Out[139]:
```
NAME_CONTRACT_STATUS
Canceled        305805
Refused          40897
Unused offer     25524
Approved             4
dtype: int64
```

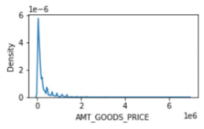
In [140]:
```
col_rem['CNT_PAYMENT'] = col_rem['CNT_PAYMENT'].fillna(0)
```

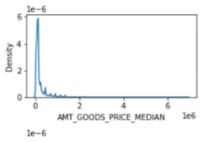
In [141]:
```
#for PRODUCT_COMBINATION
col_rem['PRODUCT_COMBINATION'].head()
```

Out[141]:
```
0    POS mobile with interest
1              Cash X-Sell: low
2             Cash X-Sell: high
3           Cash X-Sell: middle
4             Cash Street: high
```

```
    sea.kdeplot(data=col_rem,x=col)
    mplot.subplots_adjust(wspace=0.5,hspace=0.5)
```

Out[131]:
```
<Figure size 720x360 with 0 Axes>
<AxesSubplot:>
<AxesSubplot:xlabel='AMT_GOODS_PRICE', ylabel='Density'>
<AxesSubplot:>
<AxesSubplot:xlabel='AMT_GOODS_PRICE_MEDIAN', ylabel='Density'>
<AxesSubplot:>
<AxesSubplot:xlabel='AMT_GOODS_PRICE_MEAN', ylabel='Density'>
<AxesSubplot:>
<AxesSubplot:xlabel='AMT_GOODS_PRICE_MODE', ylabel='Density'>
```






```
1e−6
NAME_PORTFOLIO              0.0
NAME_GOODS_CATEGORY         0.0
NAME_CLIENT_TYPE            0.0
CODE_REJECT_REASON          0.0
NAME_PAYMENT_TYPE           0.0
DAYS_DECISION               0.0
NAME_CONTRACT_STATUS        0.0
NAME_CASH_LOAN_PURPOSE      0.0
AMT_GOODS_PRICE             0.0
AMT_CREDIT                  0.0
AMT_APPLICATION             0.0
AMT_ANNUITY                 0.0
NAME_CONTRACT_TYPE          0.0
PRODUCT_COMBINATION         0.0
dtype: float64
```

In [147]:
```
len(col_rem.columns)
```

Out[147]:
```
22
```

In [148]:
```
#merging two dataframe
merged_data = pan.merge(app_ext_removal,col_rem,how='inner',on='SK_ID_CURR')
merged_data.head()
```

Out[148]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE_x	CODE_GENDER	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT_x	AMT_ANNUITY_x	AMT_GOODS_
0	100002	1	Cash loans	M	0	202500.0	406597.5	24700.5	

```
NAME_CLIENT_TYPE            0.000000
NAME_GOODS_CATEGORY         0.000000
SK_ID_CURR                  0.000000
CODE_REJECT_REASON          0.000000
NAME_PAYMENT_TYPE           0.000000
DAYS_DECISION               0.000000
NAME_CONTRACT_STATUS        0.000000
NAME_CASH_LOAN_PURPOSE      0.000000
AMT_GOODS_PRICE             0.000000
AMT_APPLICATION             0.000000
NAME_CONTRACT_TYPE          0.000000
AMT_GOODS_PRICE_MODE        0.000000
dtype: float64
```

In [134]:
```
# droping extra amount columns of men median mode that we created
col_rem = col_rem.drop(labels=['AMT_GOODS_PRICE_MEDIAN','AMT_GOODS_PRICE_MEAN','AMT_GOODS_PRICE_MODE'],axis=1)
```

In [135]:
```
#for AMT_ANNUITY
col_rem['AMT_ANNUITY'].agg(func=['mean','median','max'])
```

Out[135]:
```
mean       15955.120659
median     11250.000000
max       418058.145000
Name: AMT_ANNUITY, dtype: float64
```

In [136]:
```
#mean and median has near to values so either can be used, for easier analysis median is preferred
col_rem['AMT_ANNUITY'] =col_rem['AMT_ANNUITY'].fillna(col_rem['AMT_ANNUITY'].median())
```

In [137]:
```
col_rem.isnull().sum().sort_values(ascending=False)/col_rem.shape[0]*100
```

Out[137]:
```
CNT_PAYMENT               22.286366
PRODUCT_COMBINATION        0.020716
  Text(8, 0, 'Everyday expenses'),
  Text(9, 0, 'Medicine'),
```

```
Text(10, 0, 'Car repairs'),
Text(11, 0, 'Urgent needs'),
Text(12, 0, 'Buying a holiday home / land'),
Text(13, 0, 'Building a house or an annex'),
Text(14, 0, 'Furniture'),
Text(15, 0, 'Journey'),
Text(16, 0, 'Purchase of electronic equipment'),
Text(17, 0, 'Wedding / gift / holiday'),
Text(18, 0, 'Buying a home'),
Text(19, 0, 'Business development'),
Text(20, 0, 'Gasification / water supply'),
Text(21, 0, 'Buying a garage'),
Text(22, 0, 'Hobby'),
Text(23, 0, 'Money for a third person'),
Text(24, 0, 'Refusal to name the goal')])
```