

Importing Libraries

```
In [1]: ► import numpy as np  
      import pandas as pd  
  
      import matplotlib.pyplot as plt  
      %matplotlib inline  
      import seaborn as sns  
  
      import warnings  
      warnings.filterwarnings('ignore')
```

```
In [2]: ► plt.rcParams['figure.figsize']=(10,5)  
      plt.style.use('ggplot')
```

```
In [3]: ► pd.options.display.max_rows = 4000  
      pd.options.display.max_columns = 1000
```

Loading the data

```
In [4]: ┌─ data_dict=pd.read_csv('C:/Users/HP/Desktop/Data Science/Relevel/Python/Class pdf/Relevel 28 Oct/columns_descriptio  
data_dict.head(100)
```

Out[4]:

	0	Table	Row	Description	Special
0	1	application_data	SK_ID_CURR	ID of loan in our sample	NaN
1	2	application_data	TARGET	Target variable (1 - client with payment diffi...	NaN
2	5	application_data	NAME_CONTRACT_TYPE	Identification if loan is cash or revolving	NaN
3	6	application_data	CODE_GENDER	Gender of the client	NaN
4	7	application_data	FLAG_OWN_CAR	Flag if the client owns a car	NaN
5	8	application_data	FLAG_OWN_REALTY	Flag if client owns a house or flat	NaN
6	9	application_data	CNT_CHILDREN	Number of children the client has	NaN
7	10	application_data	AMT_INCOME_TOTAL	Income of the client	NaN
8	11	application_data	AMT_CREDIT	Credit amount of the loan	NaN
9	12	application_data	AMT_ANNUITY	Loan annuity	NaN
10	13	application_data	AMT_GOODS_PRICE	For consumer loans it is the price of the good...	NaN
11	14	application_data	NAME_TYPE_SUITE	Who was accompanying client when he was applying...	NaN
12	15	application_data	NAME_INCOME_TYPE	Clients income type (businessman, working, mat...	NaN
13	16	application_data	NAME_EDUCATION_TYPE	Level of highest education the client achieved	NaN
14	17	application_data	NAME_FAMILY_STATUS	Family status of the client	NaN
15	18	application_data	NAME_HOUSING_TYPE	What is the housing situation of the client (renter, owner, etc.)	NaN
16	19	application_data	REGION_POPULATION_RELATIVE	Normalized population of region where client lives	normalized
17	20	application_data	DAYS_BIRTH	Client's age in days at the time of application	time only relative to the application
18	21	application_data	DAYS_EMPLOYED	How many days before the application the person has been employed	time only relative to the application

Unnamed: 0	Table	Row	Description	Special
19	22 application_data	DAYS_REGISTRATION	How many days before the application did client apply	time only relative to the application
20	23 application_data	DAYS_ID_PUBLISH	How many days before the application did client publish	time only relative to the application
21	24 application_data	OWN_CAR_AGE	Age of client's car	NaN
22	25 application_data	FLAG_MOBIL	Did client provide mobile phone (1=YES, 0=NO)	NaN
23	26 application_data	FLAG_EMP_PHONE	Did client provide work phone (1=YES, 0=NO)	NaN
24	27 application_data	FLAG_WORK_PHONE	Did client provide home phone (1=YES, 0=NO)	NaN
25	28 application_data	FLAG_CONT_MOBILE	Was mobile phone reachable (1=YES, 0=NO)	NaN
26	29 application_data	FLAG_PHONE	Did client provide home phone (1=YES, 0=NO)	NaN
27	30 application_data	FLAG_EMAIL	Did client provide email (1=YES, 0=NO)	NaN
28	31 application_data	OCCUPATION_TYPE	What kind of occupation does the client have	NaN
29	32 application_data	CNT_FAM_MEMBERS	How many family members does client have	NaN
30	33 application_data	REGION_RATING_CLIENT	Our rating of the region where client lives (1=LOW, 5=HIGH)	NaN
31	34 application_data	REGION_RATING_CLIENT_W_CITY	Our rating of the region where client lives with respect to city (1=LOW, 5=HIGH)	NaN
32	35 application_data	WEEKDAY_APPR_PROCESS_START	On which day of the week did the client apply	NaN
			...	
33	36 application_data	HOUR_APPR_PROCESS_START	Approximately at what hour did the client apply	rounded
34	37 application_data	REG_REGION_NOT_LIVE_REGION	Flag if client's permanent address does not match their region	NaN
35	38 application_data	REG_REGION_NOT_WORK_REGION	Flag if client's permanent address does not match their work region	NaN

Unnamed:		0	Table	Row	Description	Special
36	39	application_data	LIVE_REGION_NOT_WORK_REGION		Flag if client's contact address does not match...	NaN
37	40	application_data	REG_CITY_NOT_LIVE_CITY		Flag if client's permanent address does not match...	NaN
38	41	application_data	REG_CITY_NOT_WORK_CITY		Flag if client's permanent address does not match...	NaN
39	42	application_data	LIVE_CITY_NOT_WORK_CITY		Flag if client's contact address does not match...	NaN
40	43	application_data	ORGANIZATION_TYPE		Type of organization where client works	NaN
41	44	application_data	EXT_SOURCE_1		Normalized score from external data source	normalized
42	45	application_data	EXT_SOURCE_2		Normalized score from external data source	normalized
43	46	application_data	EXT_SOURCE_3		Normalized score from external data source	normalized
44	47	application_data	APARTMENTS_AVG		Normalized information about building where th...	normalized
45	48	application_data	BASEMENTAREA_AVG		Normalized information about building where th...	normalized
46	49	application_data	YEARS_BEGINEXPLUATATION_AVG		Normalized information about building where th...	normalized
47	50	application_data	YEARS_BUILD_AVG		Normalized information about building where th...	normalized
48	51	application_data	COMMONAREA_AVG		Normalized information about building where th...	normalized
49	52	application_data	ELEVATORS_AVG		Normalized information about building where th...	normalized
50	53	application_data	ENTRANCES_AVG		Normalized information about building where th...	normalized
51	54	application_data	FLOORSMAX_AVG		Normalized information about building where th...	normalized

Unnamed:		0	Table	Row	Description	Special
52	55	application_data		FLOORSMIN_AVG	Normalized information about building where th...	normalized
53	56	application_data		LANDAREA_AVG	Normalized information about building where th...	normalized
54	57	application_data		LIVINGAPARTMENTS_AVG	Normalized information about building where th...	normalized
55	58	application_data		LIVINGAREA_AVG	Normalized information about building where th...	normalized
56	59	application_data		NONLIVINGAPARTMENTS_AVG	Normalized information about building where th...	normalized
57	60	application_data		NONLIVINGAREA_AVG	Normalized information about building where th...	normalized
58	61	application_data		APARTMENTS_MODE	Normalized information about building where th...	normalized
59	62	application_data		BASEMENTAREA_MODE	Normalized information about building where th...	normalized
60	63	application_data	YEARS_BEGINEXPLUATATION_MODE		Normalized information about building where th...	normalized
61	64	application_data		YEARS_BUILD_MODE	Normalized information about building where th...	normalized
62	65	application_data		COMMONAREA_MODE	Normalized information about building where th...	normalized
63	66	application_data		ELEVATORS_MODE	Normalized information about building where th...	normalized
64	67	application_data		ENTRANCES_MODE	Normalized information about building where th...	normalized
65	68	application_data		FLOORSMAX_MODE	Normalized information about building where th...	normalized

Unnamed:		0	Table	Row	Description	Special
66	69	application_data		FLOORSMIN_MODE	Normalized information about building where th...	normalized
67	70	application_data		LANDAREA_MODE	Normalized information about building where th...	normalized
68	71	application_data		LIVINGAPARTMENTS_MODE	Normalized information about building where th...	normalized
69	72	application_data		LIVINGAREA_MODE	Normalized information about building where th...	normalized
70	73	application_data		NONLIVINGAPARTMENTS_MODE	Normalized information about building where th...	normalized
71	74	application_data		NONLIVINGAREA_MODE	Normalized information about building where th...	normalized
72	75	application_data		APARTMENTS_MEDI	Normalized information about building where th...	normalized
73	76	application_data		BASEMENTAREA_MEDI	Normalized information about building where th...	normalized
74	77	application_data	YEARS_BEGINEXPLUATATION_MEDI		Normalized information about building where th...	normalized
75	78	application_data		YEARS_BUILD_MEDI	Normalized information about building where th...	normalized
76	79	application_data		COMMONAREA_MEDI	Normalized information about building where th...	normalized
77	80	application_data		ELEVATORS_MEDI	Normalized information about building where th...	normalized
78	81	application_data		ENTRANCES_MEDI	Normalized information about building where th...	normalized
79	82	application_data		FLOORSMAX_MEDI	Normalized information about building where th...	normalized

Unnamed: 0	Table	Row	Description	Special
80	83 application_data	FLOORSMIN_MEDI	Normalized information about building where th...	normalized
81	84 application_data	LANDAREA_MEDI	Normalized information about building where th...	normalized
82	85 application_data	LIVINGAPARTMENTS_MEDI	Normalized information about building where th...	normalized
83	86 application_data	LIVINGAREA_MEDI	Normalized information about building where th...	normalized
84	87 application_data	NONLIVINGAPARTMENTS_MEDI	Normalized information about building where th...	normalized
85	88 application_data	NONLIVINGAREA_MEDI	Normalized information about building where th...	normalized
86	89 application_data	FONDKAPREMONT_MODE	Normalized information about building where th...	normalized
87	90 application_data	HOUSETYPE_MODE	Normalized information about building where th...	normalized
88	91 application_data	TOTALAREA_MODE	Normalized information about building where th...	normalized
89	92 application_data	WALLSMATERIAL_MODE	Normalized information about building where th...	normalized
90	93 application_data	EMERGENCYSTATE_MODE	Normalized information about building where th...	normalized
91	94 application_data	OBS_30_CNT_SOCIAL_CIRCLE	How many observation of client's social surrou...	NaN
92	95 application_data	DEF_30_CNT_SOCIAL_CIRCLE	How many observation of client's social surrou...	NaN
93	96 application_data	OBS_60_CNT_SOCIAL_CIRCLE	How many observation of client's social surrou...	NaN

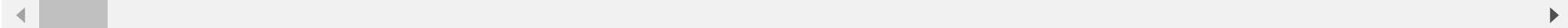
Unnamed:

0	Table	Row	Description	Special
94	97 application_data	DEF_60_CNT_SOCIAL_CIRCLE	How many observation of client's social surrou...	NaN
95	98 application_data	DAYS_LAST_PHONE_CHANGE	How many days before application did client ch...	NaN
96	99 application_data	FLAG_DOCUMENT_2	Did client provide document 2	NaN
97	100 application_data	FLAG_DOCUMENT_3	Did client provide document 3	NaN
98	101 application_data	FLAG_DOCUMENT_4	Did client provide document 4	NaN
99	102 application_data	FLAG_DOCUMENT_5	Did client provide document 5	NaN

In [5]: ► credit_data=pd.read_csv('C:/Users/HP/Desktop/Data Science/Relevel/Python/Class pdf/Relevel 28 Oct/application_data.csv') credit_data.head(10)

Out[5]:

SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN
0	100002	1	Cash loans	M	N	Y
1	100003	0	Cash loans	F	N	N
2	100004	0	Revolving loans	M	Y	Y
3	100006	0	Cash loans	F	N	Y
4	100007	0	Cash loans	M	N	Y
5	100008	0	Cash loans	M	N	Y
6	100009	0	Cash loans	F	Y	Y
7	100010	0	Cash loans	M	Y	Y
8	100011	0	Cash loans	F	N	Y
9	100012	0	Revolving loans	M	N	Y



Data Wrangling

Inspecting the data

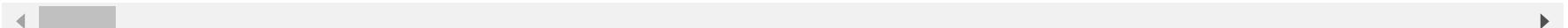
```
In [6]: └─ # checking the shape of the data  
      credit_data.shape
```

Out[6]: (307511, 122)

```
In [7]: └─ # checking 5 point summary  
      credit_data.describe()
```

Out[7]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	R
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	307499.000000	3.072330e+05	
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	27108.573909	5.383962e+05	
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	14493.737315	3.694465e+05	
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1615.500000	4.050000e+04	
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16524.000000	2.385000e+05	
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	24903.000000	4.500000e+05	
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.000000	6.795000e+05	
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	258025.500000	4.050000e+06	



```
In [8]: ┆ null_perc=credit_data.isna().sum()*100/len(credit_data)
      null_perc.sort_values(ascending=False)
```

Out[8]:

COMMONAREA_MEDI	69.872297
COMMONAREA_AVG	69.872297
COMMONAREA_MODE	69.872297
NONLIVINGAPARTMENTS_MODE	69.432963
NONLIVINGAPARTMENTS_AVG	69.432963
NONLIVINGAPARTMENTS_MEDI	69.432963
FONDKAPREMONT_MODE	68.386172
LIVINGAPARTMENTS_MODE	68.354953
LIVINGAPARTMENTS_AVG	68.354953
LIVINGAPARTMENTS_MEDI	68.354953
FLOORSMIN_AVG	67.848630
FLOORSMIN_MODE	67.848630
FLOORSMIN_MEDI	67.848630
YEARS_BUILD_MEDI	66.497784
YEARS_BUILD_MODE	66.497784
YEARS_BUILD_AVG	66.497784
OWN_CAR_AGE	65.990810
LANDAREA_MEDI	59.376738
LANDAREA_MODE	59.376738
LANDAREA_AVG	59.376738
BASEMENTAREA_MEDI	58.515956
BASEMENTAREA_AVG	58.515956
BASEMENTAREA_MODE	58.515956
EXT_SOURCE_1	56.381073
NONLIVINGAREA_MODE	55.179164
NONLIVINGAREA_AVG	55.179164
NONLIVINGAREA_MEDI	55.179164
ELEVATORS_MEDI	53.295980
ELEVATORS_AVG	53.295980
ELEVATORS_MODE	53.295980
WALLSMATERIAL_MODE	50.840783
APARTMENTS_MEDI	50.749729
APARTMENTS_AVG	50.749729
APARTMENTS_MODE	50.749729
ENTRANCES_MEDI	50.348768
ENTRANCES_AVG	50.348768
ENTRANCES_MODE	50.348768
LIVINGAREA_AVG	50.193326
LIVINGAREA_MODE	50.193326
LIVINGAREA_MEDI	50.193326
HOUSETYPE_MODE	50.176091

FLOORSMAX_MODE	49.760822
FLOORSMAX_MEDI	49.760822
FLOORSMAX_AVG	49.760822
YEARS_BEGINEXPLUATATION_MODE	48.781019
YEARS_BEGINEXPLUATATION_MEDI	48.781019
YEARS_BEGINEXPLUATATION_AVG	48.781019
TOTALAREA_MODE	48.268517
EMERGENCYSTATE_MODE	47.398304
OCCUPATION_TYPE	31.345545
EXT_SOURCE_3	19.825307
AMT_REQ_CREDIT_BUREAU_HOUR	13.501631
AMT_REQ_CREDIT_BUREAU_DAY	13.501631
AMT_REQ_CREDIT_BUREAU_WEEK	13.501631
AMT_REQ_CREDIT_BUREAU_MON	13.501631
AMT_REQ_CREDIT_BUREAU_QRT	13.501631
AMT_REQ_CREDIT_BUREAU_YEAR	13.501631
NAME_TYPE_SUITE	0.420148
OBS_30_CNT_SOCIAL_CIRCLE	0.332021
DEF_30_CNT_SOCIAL_CIRCLE	0.332021
OBS_60_CNT_SOCIAL_CIRCLE	0.332021
DEF_60_CNT_SOCIAL_CIRCLE	0.332021
EXT_SOURCE_2	0.214626
AMT_GOODS_PRICE	0.090403
AMT_ANNUITY	0.003902
CNT_FAM_MEMBERS	0.000650
DAYS_LAST_PHONE_CHANGE	0.000325
CNT_CHILDREN	0.000000
FLAG_DOCUMENT_8	0.000000
NAME_CONTRACT_TYPE	0.000000
CODE_GENDER	0.000000
FLAG_OWN_CAR	0.000000
FLAG_DOCUMENT_2	0.000000
FLAG_DOCUMENT_3	0.000000
FLAG_DOCUMENT_4	0.000000
FLAG_DOCUMENT_5	0.000000
FLAG_DOCUMENT_6	0.000000
FLAG_DOCUMENT_7	0.000000
FLAG_DOCUMENT_9	0.000000
FLAG_DOCUMENT_21	0.000000
FLAG_DOCUMENT_10	0.000000
FLAG_DOCUMENT_11	0.000000
FLAG_OWN_REALTY	0.000000

```
FLAG_DOCUMENT_13           0.000000
FLAG_DOCUMENT_14           0.000000
FLAG_DOCUMENT_15           0.000000
FLAG_DOCUMENT_16           0.000000
FLAG_DOCUMENT_17           0.000000
FLAG_DOCUMENT_18           0.000000
FLAG_DOCUMENT_19           0.000000
FLAG_DOCUMENT_20           0.000000
FLAG_DOCUMENT_12           0.000000
AMT_CREDIT                 0.000000
AMT_INCOME_TOTAL           0.000000
FLAG_PHONE                 0.000000
LIVE_CITY_NOT_WORK_CITY     0.000000
REG_CITY_NOT_WORK_CITY      0.000000
TARGET                      0.000000
REG_CITY_NOT_LIVE_CITY      0.000000
LIVE_REGION_NOT_WORK_REGION 0.000000
REG_REGION_NOT_WORK_REGION   0.000000
REG_REGION_NOT_LIVE_REGION   0.000000
HOUR_APPR_PROCESS_START     0.000000
WEEKDAY_APPR_PROCESS_START   0.000000
REGION_RATING_CLIENT_W_CITY 0.000000
REGION_RATING_CLIENT         0.000000
FLAG_EMAIL                  0.000000
FLAG_CONT_MOBILE             0.000000
ORGANIZATION_TYPE            0.000000
FLAG_WORK_PHONE              0.000000
FLAG_EMP_PHONE                0.000000
FLAG_MOBIL                   0.000000
DAYS_ID_PUBLISH              0.000000
DAYS_REGISTRATION             0.000000
DAYS_EMPLOYED                  0.000000
DAYS_BIRTH                     0.000000
REGION_POPULATION_RELATIVE    0.000000
NAME_HOUSING_TYPE              0.000000
NAME_FAMILY_STATUS               0.000000
NAME_EDUCATION_TYPE              0.000000
NAME_INCOME_TYPE                  0.000000
SK_ID_CURR                     0.000000
dtype: float64
```

```
In [9]: # filter top 60 columns with max null values  
null_perc.sort_values(ascending=False).head(60)
```

Out[9]:

COMMONAREA_MEDI	69.872297
COMMONAREA_AVG	69.872297
COMMONAREA_MODE	69.872297
NONLIVINGAPARTMENTS_MODE	69.432963
NONLIVINGAPARTMENTS_AVG	69.432963
NONLIVINGAPARTMENTS_MEDI	69.432963
FONDKAPREMONT_MODE	68.386172
LIVINGAPARTMENTS_MODE	68.354953
LIVINGAPARTMENTS_AVG	68.354953
LIVINGAPARTMENTS_MEDI	68.354953
FLOORSMIN_AVG	67.848630
FLOORSMIN_MODE	67.848630
FLOORSMIN_MEDI	67.848630
YEARS_BUILD_MEDI	66.497784
YEARS_BUILD_MODE	66.497784
YEARS_BUILD_AVG	66.497784
OWN_CAR_AGE	65.990810
LANDAREA_MEDI	59.376738
LANDAREA_MODE	59.376738
LANDAREA_AVG	59.376738
BASEMENTAREA_MEDI	58.515956
BASEMENTAREA_AVG	58.515956
BASEMENTAREA_MODE	58.515956
EXT_SOURCE_1	56.381073
NONLIVINGAREA_MODE	55.179164
NONLIVINGAREA_AVG	55.179164
NONLIVINGAREA_MEDI	55.179164
ELEVATORS_MEDI	53.295980
ELEVATORS_AVG	53.295980
ELEVATORS_MODE	53.295980
WALLSMATERIAL_MODE	50.840783
APARTMENTS_MEDI	50.749729
APARTMENTS_AVG	50.749729
APARTMENTS_MODE	50.749729
ENTRANCES_MEDI	50.348768
ENTRANCES_AVG	50.348768
ENTRANCES_MODE	50.348768
LIVINGAREA_AVG	50.193326
LIVINGAREA_MODE	50.193326
LIVINGAREA_MEDI	50.193326
HOUSETYPE_MODE	50.176091

```
FLOORSMAX_MODE           49.760822
FLOORSMAX_MEDI            49.760822
FLOORSMAX_AVG             49.760822
YEARS_BEGINEXPLUATATION_MODE 48.781019
YEARS_BEGINEXPLUATATION_MEDI 48.781019
YEARS_BEGINEXPLUATATION_AVG 48.781019
TOTALAREA_MODE              48.268517
EMERGENCYSTATE_MODE         47.398304
OCCUPATION_TYPE             31.345545
EXT_SOURCE_3                 19.825307
AMT_REQ_CREDIT_BUREAU_HOUR 13.501631
AMT_REQ_CREDIT_BUREAU_DAY   13.501631
AMT_REQ_CREDIT_BUREAU_WEEK 13.501631
AMT_REQ_CREDIT_BUREAU_MON   13.501631
AMT_REQ_CREDIT_BUREAU_QRT   13.501631
AMT_REQ_CREDIT_BUREAU_YEAR 13.501631
NAME_TYPE_SUITE               0.420148
OBS_30_CNT_SOCIAL_CIRCLE    0.332021
DEF_30_CNT_SOCIAL_CIRCLE     0.332021
dtype: float64
```

In our case the columns which contains more than 45% null values will be discarded

Data Cleaning

Identifying and Removing null values > 45%

```
In [10]: ┆ null_cols=credit_data.isna().sum().sort_values(ascending=False)
null_cols=null_cols[null_cols.values>(.45*len(credit_data))]

no=len(null_cols)
print("There are "+ str(no) + " columns with more than 45% NULLs")

null_cols
```

There are 49 columns with more than 45% NULLs

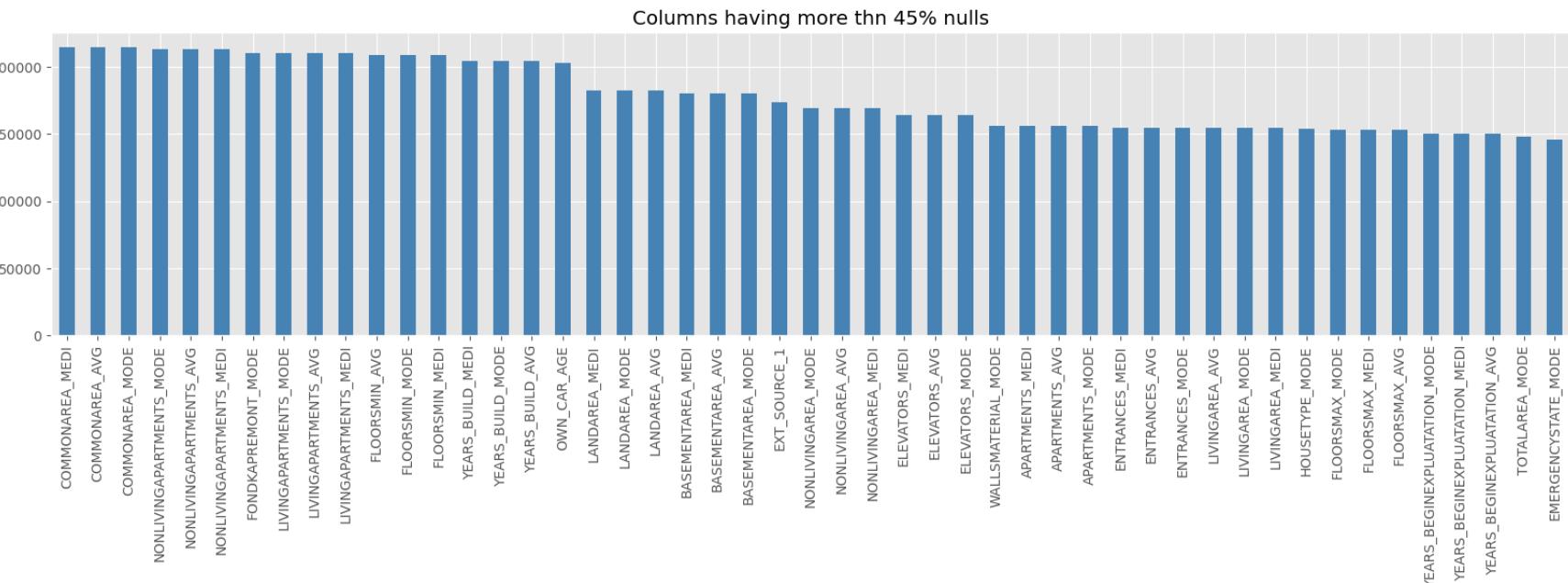
Out[10]:

COMMONAREA_MEDI	214865
COMMONAREA_AVG	214865
COMMONAREA_MODE	214865
NONLIVINGAPARTMENTS_MODE	213514
NONLIVINGAPARTMENTS_AVG	213514
NONLIVINGAPARTMENTS_MEDI	213514
FONDKAPREMONT_MODE	210295
LIVINGAPARTMENTS_MODE	210199
LIVINGAPARTMENTS_AVG	210199
LIVINGAPARTMENTS_MEDI	210199
FLOORSMIN_AVG	208642
FLOORSMIN_MODE	208642
FLOORSMIN_MEDI	208642
YEARS_BUILD_MEDI	204488
YEARS_BUILD_MODE	204488
YEARS_BUILD_AVG	204488
OWN_CAR_AGE	202929
LANDAREA_MEDI	182590
LANDAREA_MODE	182590
LANDAREA_AVG	182590
BASEMENTAREA_MEDI	179943
BASEMENTAREA_AVG	179943
BASEMENTAREA_MODE	179943
EXT_SOURCE_1	173378
NONLIVINGAREA_MODE	169682
NONLIVINGAREA_AVG	169682
NONLIVINGAREA_MEDI	169682
ELEVATORS_MEDI	163891
ELEVATORS_AVG	163891
ELEVATORS_MODE	163891
WALLSMATERIAL_MODE	156341
APARTMENTS_MEDI	156061
APARTMENTS_AVG	156061
APARTMENTS_MODE	156061
ENTRANCES_MEDI	154828
ENTRANCES_AVG	154828
ENTRANCES_MODE	154828
LIVINGAREA_AVG	154350
LIVINGAREA_MODE	154350
LIVINGAREA_MEDI	154350
HOUSETYPE_MODE	154297

```
FLOORSMAX_MODE           153020
FLOORSMAX_MEDI          153020
FLOORSMAX_AVG           153020
YEARS_BEGINEXPLUATATION_MODE 150007
YEARS_BEGINEXPLUATATION_MEDI 150007
YEARS_BEGINEXPLUATATION_AVG 150007
TOTALAREA_MODE          148431
EMERGENCYSTATE_MODE     145755
dtype: int64
```

In [11]: # Let's visually look at the columns with NULLs>45% and their NULL counts

```
plt.figure(figsize=(20,4))
null_cols.plot(kind='bar', color="steelblue")
plt.title('Columns having more than 45% nulls')
plt.show()
```



```
In [12]: # Removal of Null columns > 45%  
  
def remove_null_cols(data):  
    perc=0.45  
    df=data.copy()  
    shape_before = df.shape  
    remove_cols=(df.isna().sum()/len(df))  
    remove_cols=list(remove_cols[remove_cols.values>=perc].index)  
    df.drop(labels=remove_cols, axis=1, inplace=True)  
    print("Number of Columns dropped\t: ",len(remove_cols))  
    print("\nOld dataset rows,columns",shape_before,"\\nNew dataset rows,columns",df.shape)  
    return df
```

```
In [13]: credit_data_1=remove_null_cols(credit_data)
```

Number of Columns dropped : 49

Old dataset rows,columns (307511, 122)
New dataset rows,columns (307511, 73)

Imputing Missing Data

The below listed columns can be categorized into a group of columns with similar significance as they all represent number of queries made to the Credit Bureau

AMT_REQ_CREDIT_BUREAU_YEAR

AMT_REQ_CREDIT_BUREAU_MON

AMT_REQ_CREDIT_BUREAU_WEEK

AMT_REQ_CREDIT_BUREAU_DAY

AMT_REQ_CREDIT_BUREAU_HOUR

AMT REQ CREDIT BUREAU QRT

In [14]: # Checking value counts for AMT_REQ_CREDIT_BUREAU_YEAR

```
credit_data_1.AMT_REQ_CREDIT_BUREAU_YEAR.value_counts()  
  
# We see that there are 71k 0s
```

Out[14]:

0.0	71801
1.0	63405
2.0	50192
3.0	33628
4.0	20714
5.0	12052
6.0	6967
7.0	3869
8.0	2127
9.0	1096
11.0	31
12.0	30
10.0	22
13.0	19
14.0	10
17.0	7
15.0	6
19.0	4
18.0	4
16.0	3
25.0	1
23.0	1
22.0	1
21.0	1
20.0	1

Name: AMT_REQ_CREDIT_BUREAU_YEAR, dtype: int64

```
In [15]: credit_data_1.AMT_REQ_CREDIT_BUREAU_YEAR.value_counts(normalize=True)*100  
# so around 71K or approx 27% of the column contains 0
```

```
Out[15]: 0.0    26.993669  
1.0    23.837183  
2.0    18.869740  
3.0    12.642485  
4.0    7.787452  
5.0    4.530963  
6.0    2.619252  
7.0    1.454555  
8.0    0.799648  
9.0    0.412042  
11.0   0.011654  
12.0   0.011279  
10.0   0.008271  
13.0   0.007143  
14.0   0.003760  
17.0   0.002632  
15.0   0.002256  
19.0   0.001504  
18.0   0.001504  
16.0   0.001128  
25.0   0.000376  
23.0   0.000376  
22.0   0.000376  
21.0   0.000376  
20.0   0.000376  
Name: AMT_REQ_CREDIT_BUREAU_YEAR, dtype: float64
```

In [16]: # here used mode function to find out the most occured value in each column

```
print(credit_data_1.AMT_REQ_CREDIT_BUREAU_YEAR.mode())
print(credit_data_1.AMT_REQ_CREDIT_BUREAU_MON.mode())
print(credit_data_1.AMT_REQ_CREDIT_BUREAU_WEEK.mode())
print(credit_data_1.AMT_REQ_CREDIT_BUREAU_DAY.mode())
print(credit_data_1.AMT_REQ_CREDIT_BUREAU_HOUR.mode())
print(credit_data_1.AMT_REQ_CREDIT_BUREAU_QRT.mode())
```

```
0    0.0
Name: AMT_REQ_CREDIT_BUREAU_YEAR, dtype: float64
0    0.0
Name: AMT_REQ_CREDIT_BUREAU_MON, dtype: float64
0    0.0
Name: AMT_REQ_CREDIT_BUREAU_WEEK, dtype: float64
0    0.0
Name: AMT_REQ_CREDIT_BUREAU_DAY, dtype: float64
0    0.0
Name: AMT_REQ_CREDIT_BUREAU_HOUR, dtype: float64
0    0.0
Name: AMT_REQ_CREDIT_BUREAU_QRT, dtype: float64
```

In [17]: credit_data_2=credit_data_1.copy()

In [18]: # Imputing null with 0s

```
impute_list = ['AMT_REQ_CREDIT_BUREAU_YEAR', 'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_WEEK',
               'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_QRT']

for i in impute_list:
    credit_data_2[i] = credit_data_1[i].fillna(credit_data_1[i].mode()[0])
```

In [19]: # Verifying count of NULLs after imputation

```
print(credit_data_2['AMT_REQ_CREDIT_BUREAU_YEAR'].isnull().sum())
print(credit_data_2['AMT_REQ_CREDIT_BUREAU_MON'].isnull().sum())
print(credit_data_2['AMT_REQ_CREDIT_BUREAU_WEEK'].isnull().sum())
print(credit_data_2['AMT_REQ_CREDIT_BUREAU_DAY'].isnull().sum())
print(credit_data_2['AMT_REQ_CREDIT_BUREAU_HOUR'].isnull().sum())
print(credit_data_2['AMT_REQ_CREDIT_BUREAU_QRT'].isnull().sum())
```

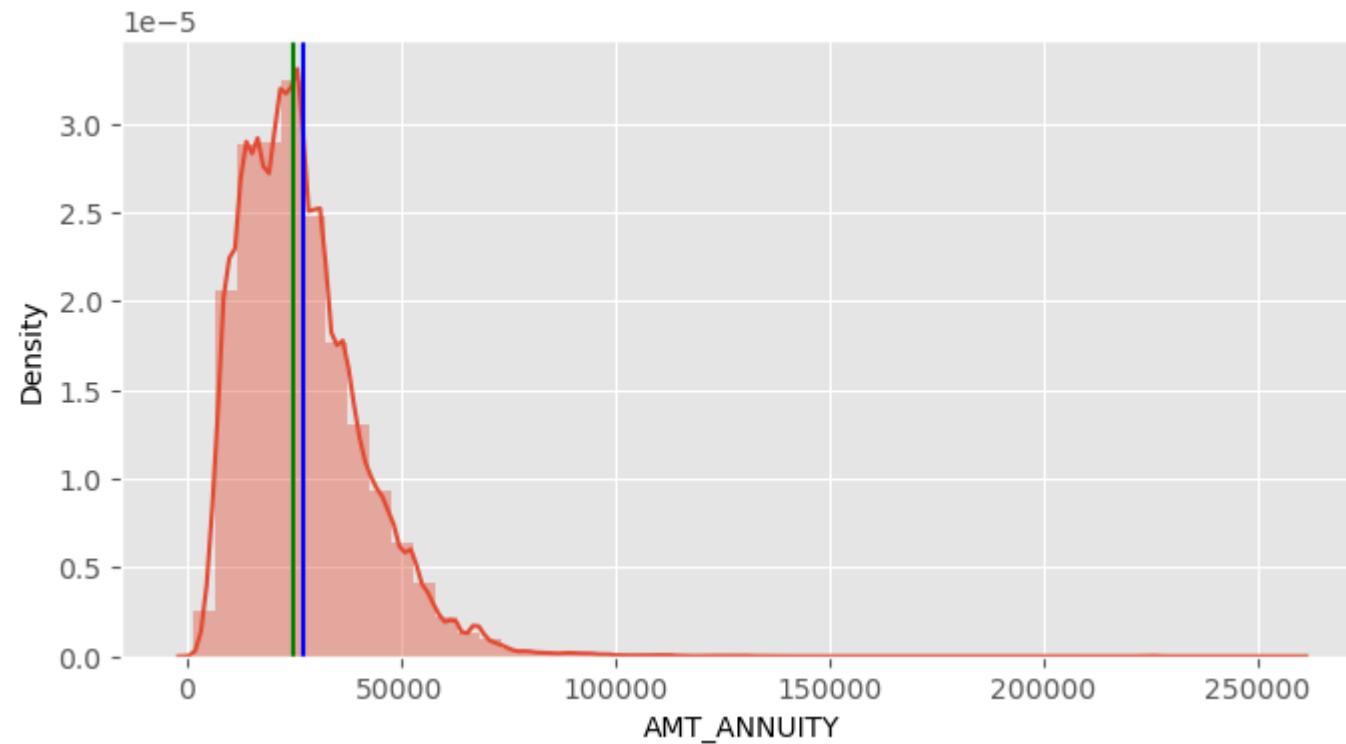
```
0
0
0
0
0
0
```

AMT_Annuity

In [20]: credit_data_1.AMT_ANNUITY.describe()

```
Out[20]: 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 [21]: plt.figure(figsize=(8,4))
sns.distplot(credit_data_1.AMT_ANNUITY)
plt.axvline(credit_data_1.AMT_ANNUITY.mean(),color='blue')
plt.axvline(credit_data_1.AMT_ANNUITY.median(),color='green')
plt.show()
```



```
In [22]: credit_data_1.AMT_ANNUITY.skew()
```

Out[22]: 1.5797773638612507

A skewness value greater than 1 or less than -1 indicates a highly skewed distribution. A value between 0.5 and 1 or -0.5 and -1 is moderately skewed. A value between -0.5 and 0.5 indicates that the distribution is fairly symmetrical.

Because we see a huge skewness, we will fill the missing value by median.

```
In [23]: credit_data_1.AMT_ANNUITY.median()
```

```
Out[23]: 24903.0
```

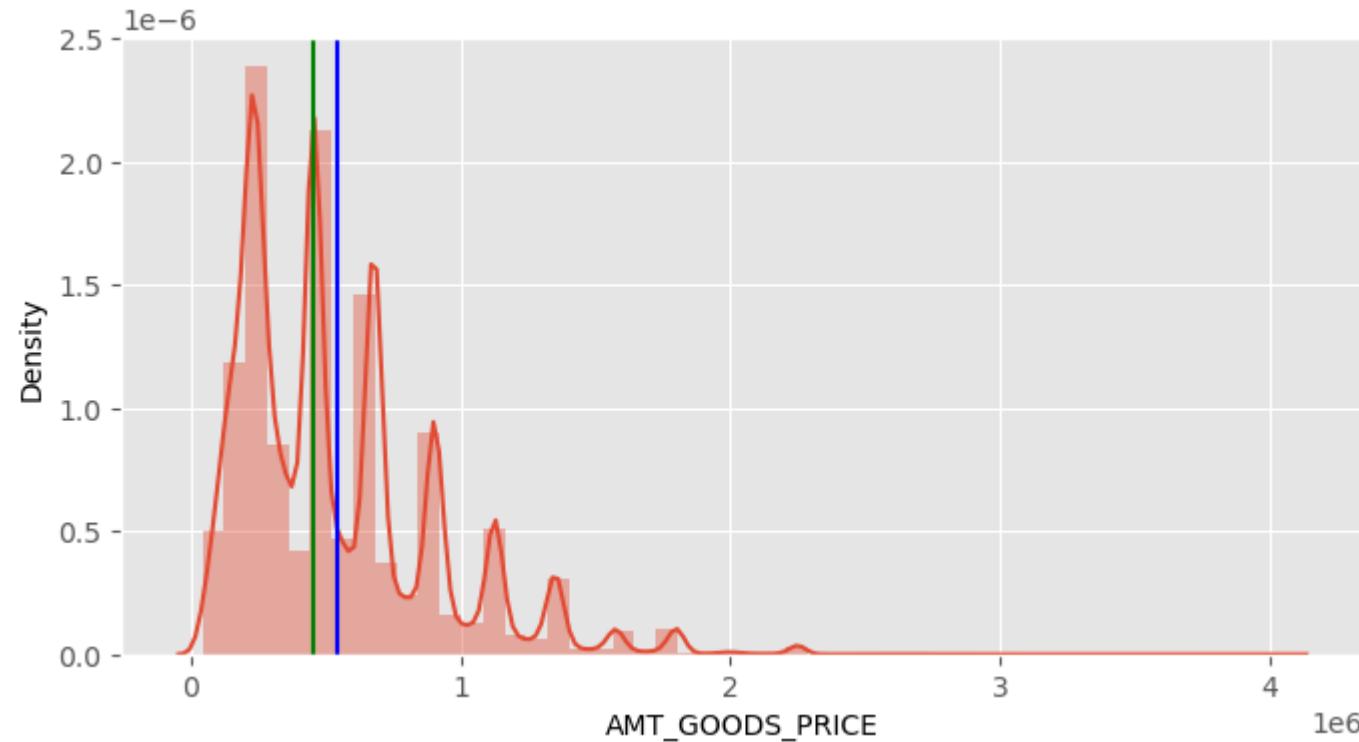
```
In [24]: # Imputing NULLs with Median
```

```
credit_data_2['AMT_ANNUITY'] = credit_data_1['AMT_ANNUITY'].fillna(credit_data_1.AMT_ANNUITY.median())
credit_data_2['AMT_ANNUITY'].isnull().sum()
```

```
Out[24]: 0
```

AMT_GOODS_PRICE

```
In [25]: plt.figure(figsize=(8,4))
sns.distplot(credit_data_1.AMT_GOODS_PRICE)
plt.axvline(credit_data_1.AMT_GOODS_PRICE.mean(),color='blue')
plt.axvline(credit_data_1.AMT_GOODS_PRICE.median(),color='green')
plt.show()
```



```
In [26]: credit_data_1.AMT_GOODS_PRICE.skew()
```

Out[26]: 1.3490003414747445

hence here also we have to impute median

In [27]: # Imputing NULLs with Median

```
credit_data_2['AMT_GOODS_PRICE'] = credit_data_1['AMT_GOODS_PRICE'].fillna(credit_data_1.AMT_GOODS_PRICE.median())
```

In [28]: # Verifying count of NULLs to be 0

```
credit_data_2['AMT_GOODS_PRICE'].isnull().sum()
```

Out[28]: 0

Fixing Erroneous Data

As seen already with the help of describe function, we know that we need to treat -ve values in days columns.

In [29]: # Confirming that all DAYS fields have -ve values

```
print(credit_data['DAYS_BIRTH'].unique())
print(credit_data['DAYS_EMPLOYED'].unique())
print(credit_data['DAYS_REGISTRATION'].unique())
print(credit_data['DAYS_ID_PUBLISH'].unique())
print(credit_data['DAYS_LAST_PHONE_CHANGE'].unique())
```

```
[ -9461 -16765 -19046 ... -7951 -7857 -25061]
[ -637 -1188 -225 ... -12971 -11084 -8694]
[ -3648. -1186. -4260. ... -16396. -14558. -14798.]
[-2120 -291 -2531 ... -6194 -5854 -6211]
[-1134. -828. -815. ... -3988. -3899. -3538.]
```

In [30]: ┌ # Preparing the List of columns to be treated

```
erroneous_cols = [cols for cols in credit_data_2 if cols.startswith('DAYS')]
erroneous_cols
```

Out[30]: ['DAYS_BIRTH',
'DAYS_EMPLOYED',
'DAYS_REGISTRATION',
'DAYS_ID_PUBLISH',
'DAYS_LAST_PHONE_CHANGE']

In [31]: ┌ # Changing the column values with Absolute values using abs function

```
credit_data_2[erroneous_cols] = np.abs(credit_data_2[erroneous_cols])
```

In [32]: ┌ # Verifying absence of -ve values in data

```
credit_data_2.describe()
```

Out[32]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	R
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	307511.000000	3.075110e+05	
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	27108.487841	5.383163e+05	
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	14493.461065	3.692890e+05	
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1615.500000	4.050000e+04	
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16524.000000	2.385000e+05	
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	24903.000000	4.500000e+05	
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.000000	6.795000e+05	
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	258025.500000	4.050000e+06	

Replacing XNAs for CODE_GENDER

In [33]: credit_data_2['CODE_GENDER'].value_counts()

Out[33]:

F	202448
M	105059
XNA	4

Name: CODE_GENDER, dtype: int64

In [34]: # Replacing XNAs with F

```
credit_data_2.loc[credit_data_2.CODE_GENDER == 'XNA', 'CODE_GENDER'] = 'F'  
credit_data_2.CODE_GENDER.value_counts()
```

Out[34]:

F	202452
M	105059

Name: CODE_GENDER, dtype: int64

Replacing XNAs for ORGANIZATION_TYPE

In [35]: # Checking value counts for ORGANIZATION_TYPE
credit_data_2.ORGANIZATION_TYPE.value_counts()

Out[35]:

Business Entity Type 3	67992
XNA	55374
Self-employed	38412
Other	16683
Medicine	11193
Business Entity Type 2	10553
Government	10404
School	8893
Trade: type 7	7831
Kindergarten	6880
Construction	6721
Business Entity Type 1	5984
Transport: type 4	5398
Trade: type 3	3492
Industry: type 9	3368
Industry: type 3	3278
Security	3247
Housing	2958
Industry: type 11	2704
Military	2634
Bank	2507
Agriculture	2454
Police	2341
Transport: type 2	2204
Postal	2157
Security Ministries	1974
Trade: type 2	1900
Restaurant	1811
Services	1575
University	1327
Industry: type 7	1307
Transport: type 3	1187
Industry: type 1	1039
Hotel	966
Electricity	950
Industry: type 4	877
Trade: type 6	631
Industry: type 5	599
Insurance	597
Telecom	577
Emergency	560

```
Industry: type 2           458
Advertising                 429
Realtor                      396
Culture                       379
Industry: type 12            369
Trade: type 1                  348
Mobile                        317
Legal Services                  305
Cleaning                      260
Transport: type 1              201
Industry: type 6                  112
Industry: type 10                109
Religion                      85
Industry: type 13                67
Trade: type 4                  64
Trade: type 5                  49
Industry: type 8                  24
Name: ORGANIZATION_TYPE, dtype: int64
```

In [36]: ┌ # Replacing XNAs with Nulls

```
credit_data_2['ORGANIZATION_TYPE'] = credit_data_1['ORGANIZATION_TYPE'].replace('XNA',np.NaN)
```

In [37]: # Checking value counts for credit_data_2
credit_data_2.ORGANIZATION_TYPE.value_counts()

Out[37]:

Business Entity Type 3	67992
Self-employed	38412
Other	16683
Medicine	11193
Business Entity Type 2	10553
Government	10404
School	8893
Trade: type 7	7831
Kindergarten	6880
Construction	6721
Business Entity Type 1	5984
Transport: type 4	5398
Trade: type 3	3492
Industry: type 9	3368
Industry: type 3	3278
Security	3247
Housing	2958
Industry: type 11	2704
Military	2634
Bank	2507
Agriculture	2454
Police	2341
Transport: type 2	2204
Postal	2157
Security Ministries	1974
Trade: type 2	1900
Restaurant	1811
Services	1575
University	1327
Industry: type 7	1307
Transport: type 3	1187
Industry: type 1	1039
Hotel	966
Electricity	950
Industry: type 4	877
Trade: type 6	631
Industry: type 5	599
Insurance	597
Telecom	577
Emergency	560
Industry: type 2	458

```
Advertising           429
Realtor              396
Culture              379
Industry: type 12    369
Trade: type 1        348
Mobile               317
Legal Services        305
Cleaning              260
Transport: type 1    201
Industry: type 6     112
Industry: type 10    109
Religion              85
Industry: type 13    67
Trade: type 4         64
Trade: type 5         49
Industry: type 8      24
Name: ORGANIZATION_TYPE, dtype: int64
```

Adding new columns by Binning Continuous Variables

```
In [38]: credit_data_2['AMT_INCOME_TOTAL'].describe()
```

```
Out[38]: count    3.075110e+05
mean      1.687979e+05
std       2.371231e+05
min       2.565000e+04
25%      1.125000e+05
50%      1.471500e+05
75%      2.025000e+05
max      1.170000e+08
Name: AMT_INCOME_TOTAL, dtype: float64
```

```
In [39]: # Using pd.qcut function to bin AMT_INCOME_TOTAL into 5 categories  
  
credit_data_2['AMT_INCOME_RANGE'] = pd.qcut(credit_data_2.AMT_INCOME_TOTAL,  
                                         q=[0, 0.2, 0.5, 0.8, 0.95, 1],  
                                         labels=['VERY_LOW', 'LOW', "MEDIUM", 'HIGH', 'VERY_HIGH'])  
credit_data_2['AMT_INCOME_RANGE'].head(7)
```

```
Out[39]: 0      MEDIUM  
1      HIGH  
2    VERY_LOW  
3      LOW  
4      LOW  
5    VERY_LOW  
6      MEDIUM  
Name: AMT_INCOME_RANGE, dtype: category  
Categories (5, object): ['VERY_LOW' < 'LOW' < 'MEDIUM' < 'HIGH' < 'VERY_HIGH']
```

```
In [40]: credit_data_2['AMT_INCOME_RANGE'].value_counts()
```

```
Out[40]: MEDIUM      106633  
LOW          90089  
VERY_LOW     63671  
HIGH         33083  
VERY_HIGH    14035  
Name: AMT_INCOME_RANGE, dtype: int64
```

Binning AMT_CREDIT

```
In [41]: # Using pd.qcut function to bin AMT_CREDIT_RANGE into 5 categories  
  
credit_data_2['AMT_CREDIT_RANGE'] = pd.qcut(credit_data_2.AMT_CREDIT, q=[0, 0.2, 0.5, 0.8, 0.95, 1],  
                                         labels=['VERY_LOW', 'LOW', "MEDIUM", 'HIGH', 'VERY_HIGH'])  
credit_data_2['AMT_CREDIT_RANGE'].head(7)
```

```
Out[41]: 0      LOW  
1      HIGH  
2    VERY_LOW  
3      LOW  
4      LOW  
5      LOW  
6  VERY_HIGH  
Name: AMT_CREDIT_RANGE, dtype: category  
Categories (5, object): ['VERY_LOW' < 'LOW' < 'MEDIUM' < 'HIGH' < 'VERY_HIGH']
```

```
In [42]: credit_data_2['AMT_CREDIT_RANGE'].value_counts()
```

```
Out[42]: MEDIUM      94750  
LOW        88924  
VERY_LOW    64925  
HIGH        44878  
VERY_HIGH   14034  
Name: AMT_CREDIT_RANGE, dtype: int64
```

Binning DAYS_BIRTH

```
In [43]: credit_data_2['DAYS_BIRTH']= (credit_data_2['DAYS_BIRTH']/365).astype(int)  
credit_data_2['DAYS_BIRTH'].unique()
```

```
Out[43]: array([25, 45, 52, 54, 46, 37, 51, 55, 39, 27, 36, 38, 23, 35, 26, 48, 31,  
 50, 40, 30, 68, 43, 28, 41, 32, 33, 47, 57, 65, 44, 64, 21, 59, 49,  
 56, 62, 53, 42, 29, 67, 63, 61, 58, 60, 34, 22, 24, 66, 69, 20])
```

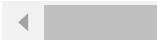
In [44]: # Using pd.cut function to bin DAYS_BIRTH into 5 categories

```
credit_data_2['DAYS_BIRTH_BINS']=pd.cut(credit_data_2['DAYS_BIRTH'],
                                         bins=[19,25,35,60,100],
                                         labels=['Very_Young', 'Young', 'Middle_Age', 'Senior_Citizen'])
```

In [45]: credit_data_2.head()

Out[45]:

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



In [46]: # Checking value counts for DAYS_BIRTH_BINS

```
credit_data_2['DAYS_BIRTH_BINS'].value_counts()
```

Out[46]:

Middle_Age	185900
Young	75925
Senior_Citizen	29368
Very_Young	16318
Name: DAYS_BIRTH_BINS, dtype:	int64

Splitting data based on TARGET

```
In [47]: ┏ credit_data_2.TARGET.value_counts()
```

```
Out[47]: 0    282686  
1    24825  
Name: TARGET, dtype: int64
```

```
In [48]: ┏ # Splitting data as per TARGET into deafulter and non-deafulter datasets
```

```
defaulter = credit_data_2[credit_data_2.TARGET==1]  
non_defaulter = credit_data_2[credit_data_2.TARGET==0]
```

```
In [49]: ┏ print(" Defaulter data shape - " + str(defaulter.shape) )  
print(" Non-Defaulter data shape - " + str(non_defaulter.shape) )
```

```
Defaulter data shape - (24825, 76)  
Non-Defaulter data shape - (282686, 76)
```

```
In [50]: ┏ # Checking % of data split as per TARGET
```

```
print(" Defaulter data % - " + str(round(len(defaulter)*100/len(credit_data_2),2) ))  
print(" Non-Defaulter data % - " + str(round(len(non_defaulter)*100/len(credit_data_2),2) ))
```

```
Defaulter data % - 8.07  
Non-Defaulter data % - 91.93
```

Univariate Analysis

```
In [51]: credit_data_2["OCCUPATION_TYPE"].value_counts()
```

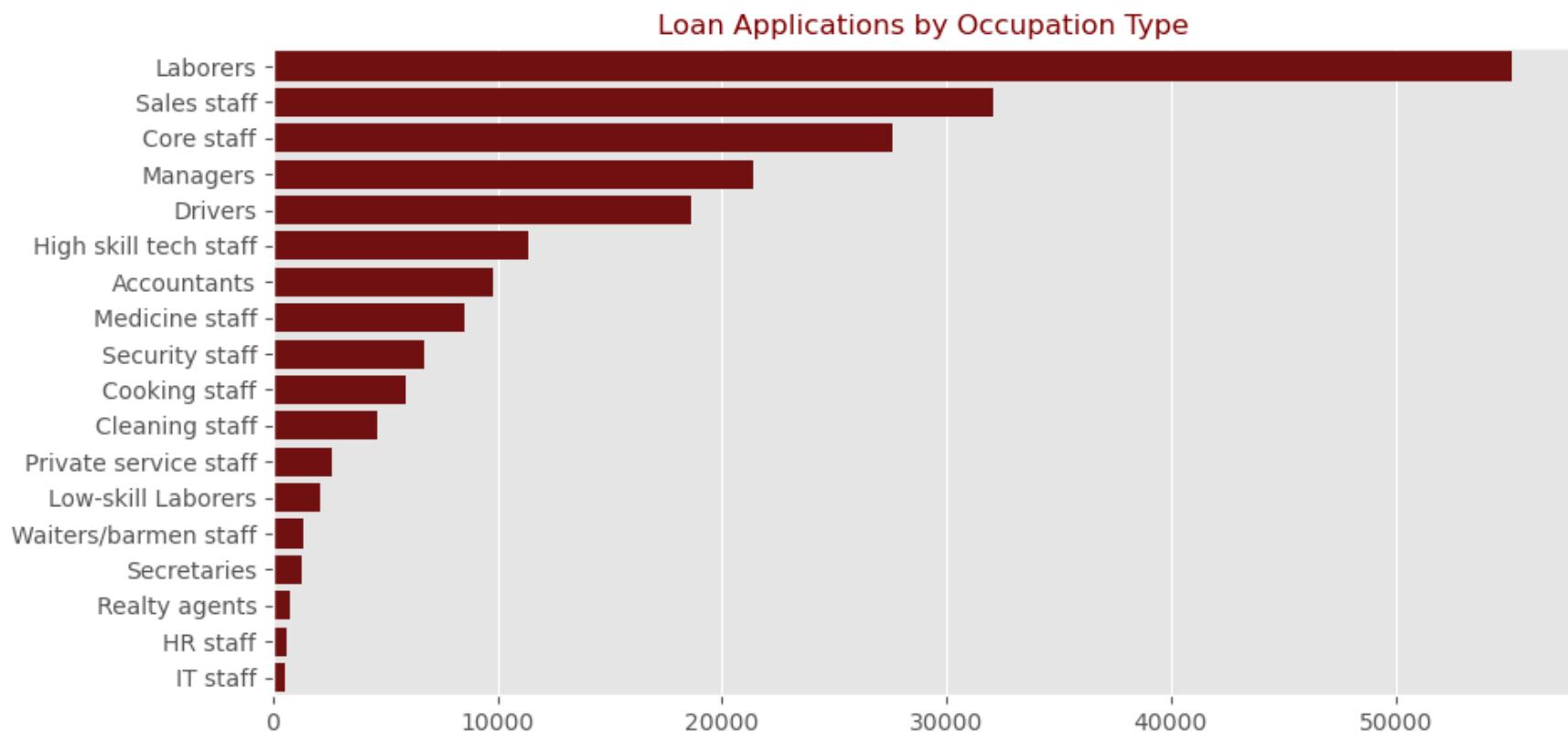
```
Out[51]:
```

Laborers	55186
Sales staff	32102
Core staff	27570
Managers	21371
Drivers	18603
High skill tech staff	11380
Accountants	9813
Medicine staff	8537
Security staff	6721
Cooking staff	5946
Cleaning staff	4653
Private service staff	2652
Low-skill Laborers	2093
Waiters/barmen staff	1348
Secretaries	1305
Realty agents	751
HR staff	563
IT staff	526

```
Name: OCCUPATION_TYPE, dtype: int64
```

In [52]: # Distribution of 'OCCUPATION_TYPE'

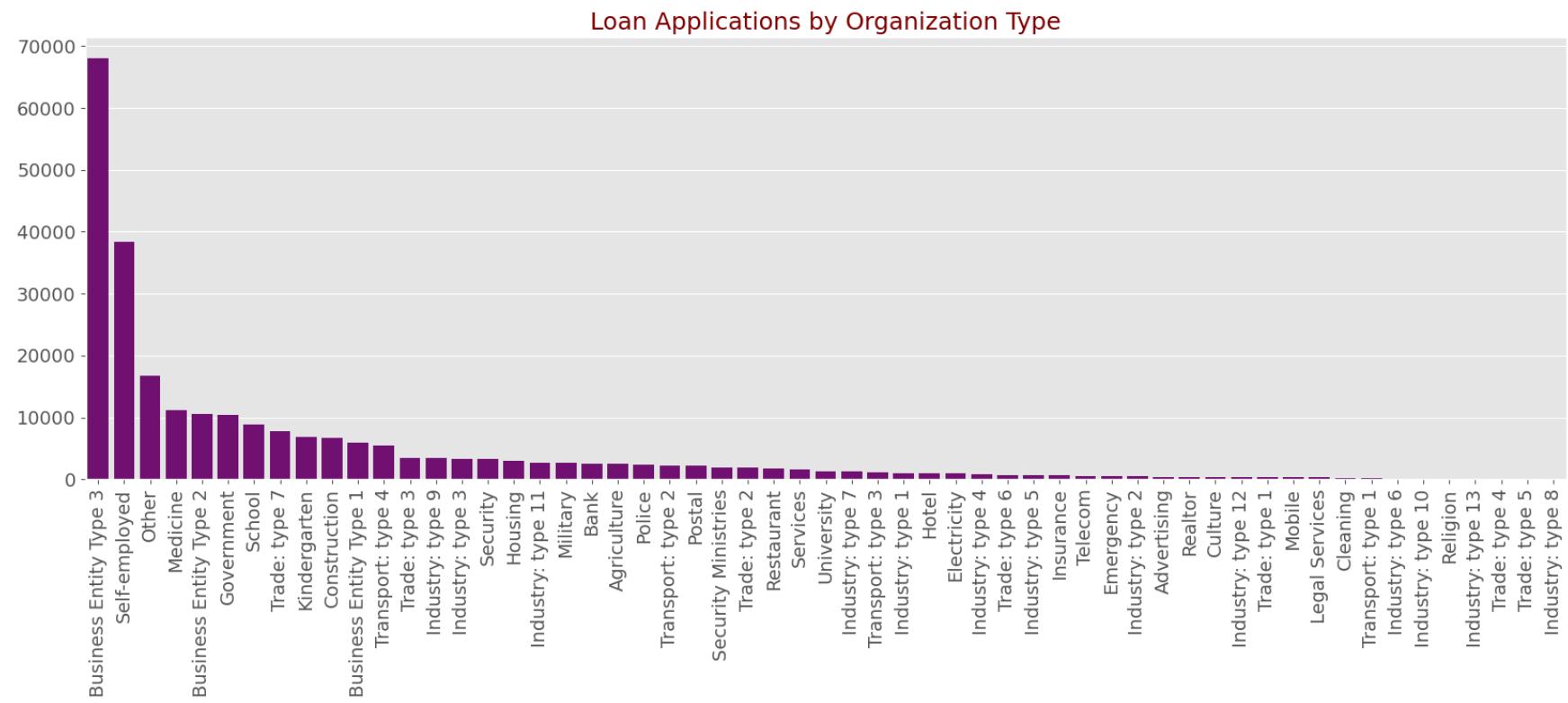
```
temp = credit_data_2["OCCUPATION_TYPE"].value_counts()
sns.barplot(y=temp.index, x = temp.values, color = 'maroon')
plt.xticks(size = 10)
plt.yticks( size = 10)
plt.title('Loan Applications by Occupation Type', size=12,color = 'maroon')
plt.show()
```



We can infer that most of the applications come for Labourers, Sales Staff and Core Staff.

In [53]: # Distribution of 'Organization Type'

```
plt.figure(figsize=(20,6))
temp = credit_data_2["ORGANIZATION_TYPE"].value_counts()
sns.barplot(x=temp.index, y = temp.values, color = 'purple')
plt.xticks(rotation=90, size = 14)
plt.yticks(size = 14)
plt.title('Loan Applications by Organization Type', size=18,color = 'maroon')
plt.show()
```



It is observed that majority of the applicants belong to Business Entity Type 3 and Self Employed.

Comparison of Gender Applicants Distribution among Defaulters and Non-Defaulters

```
In [55]: # colors = sns.color_palette('tab10')[0:5]

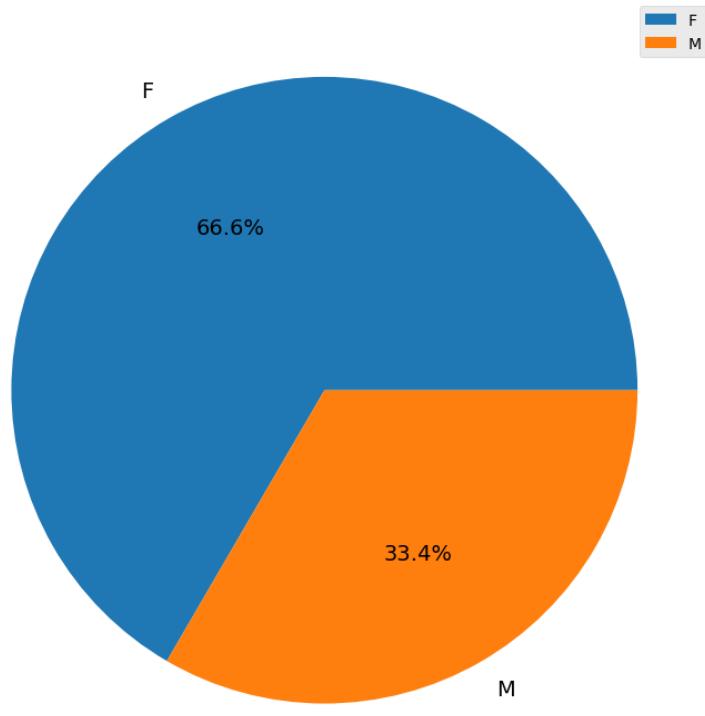
fig, ax=plt.subplots(nrows =1,ncols=2,figsize=(20,12))

data1=non_defaulter['CODE_GENDER'].value_counts()
ax[0].pie(data1.values, labels=data1.index.to_list(), colors = colors, autopct='%.1f%%',textprops={'fontsize': 14}
ax[0].set_title('Applicants by CODE_GENDER', size=18,color = '#291038')
ax[0].legend()

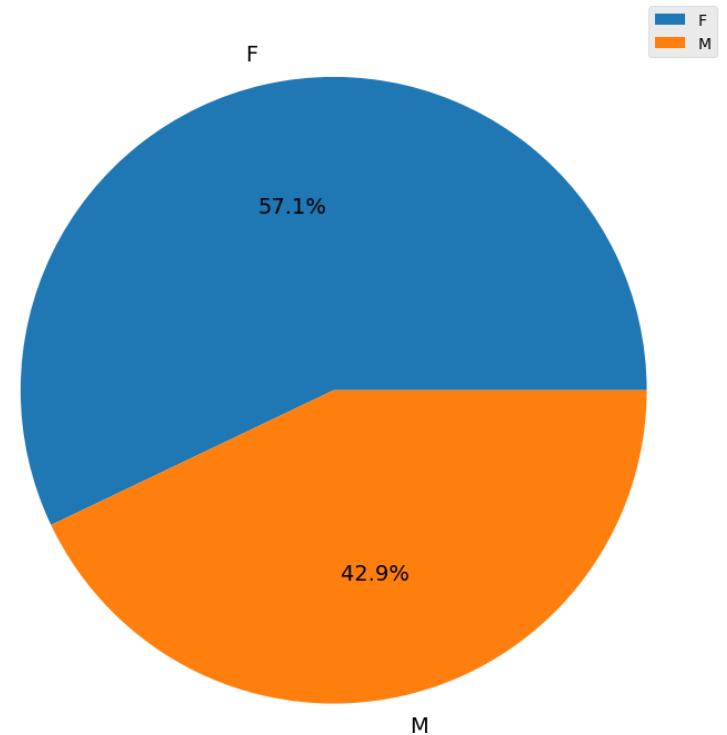
data2=defaulter['CODE_GENDER'].value_counts()
ax[1].pie(data2.values, labels=data2.index.to_list(), colors = colors, autopct='%.1f%%',textprops={'fontsize': 14}
ax[1].set_title('Defaulters by CODE_GENDER', size=18,color = '#291038')
ax[1].legend()

plt.show()
```

Applicants by CODE_GENDER



Defaulters by CODE_GENDER

**Insights -**

- There is majority of Female loan applicants.
- More Men default loans as compared to Women, since the % split has increased further for Men in case of Defaulter distribution.

In [56]: # Function for univariate comparison

```
def univariate_comparison(col,hue=None):
    colors = sns.color_palette('tab10')[0:5]

    fig, ax=plt.subplots(nrows =1,ncols=2,figsize=(25,20))

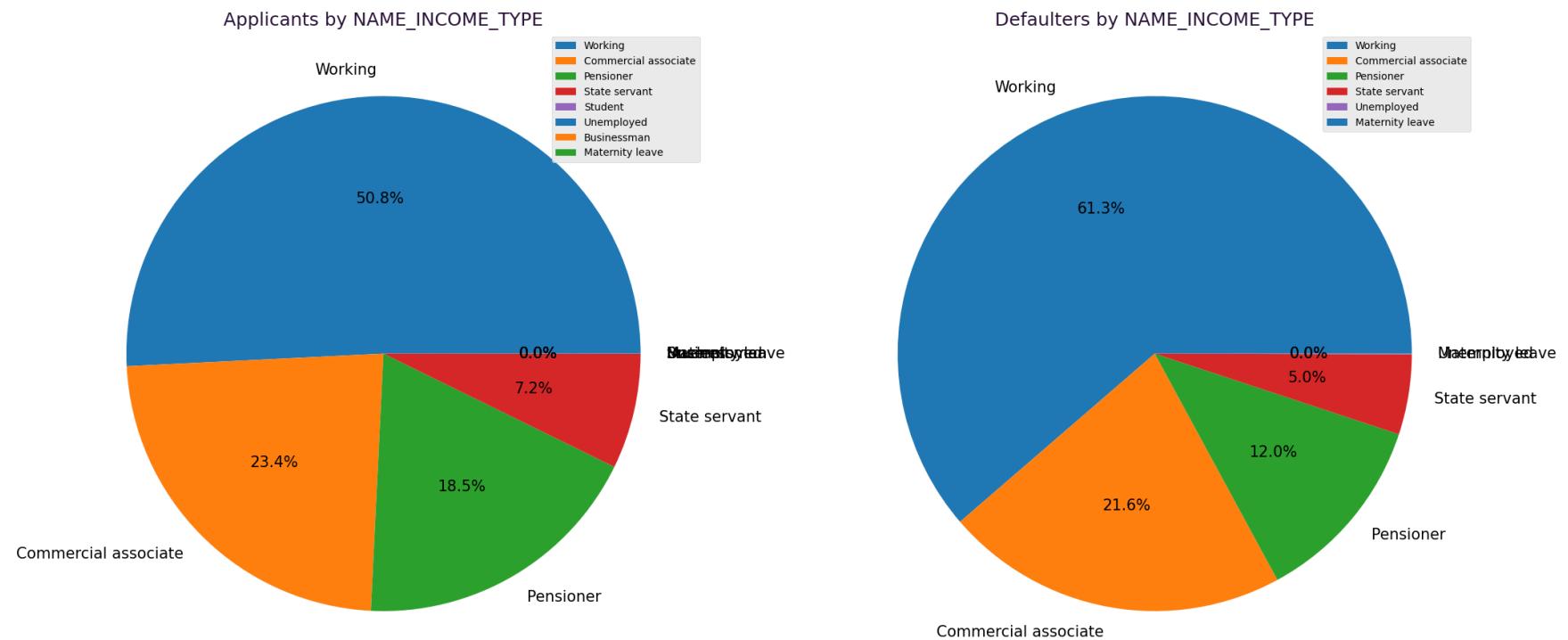
    data1=non_defaulter[col].value_counts()
    ax[0].pie(data1.values, labels=data1.index.to_list(), colors = colors, autopct='%.0.1f%%',textprops={'fontsize':18,'fontweight':'bold','color': '#291038'})
    ax[0].set_title('Applicants by '+col, size=18,color = '#291038')
    ax[0].legend()

    data2=defaultter[col].value_counts()
    ax[1].pie(data2.values, labels=data2.index.to_list(), colors = colors, autopct='%.0.1f%%',textprops={'fontsize':18,'fontweight':'bold','color': '#291038'})
    ax[1].set_title('Defaulters by '+col, size=18,color = '#291038')
    ax[1].legend()

    plt.show()
```

Comparison of Income Type Distribution among Defaulters and Non Defaulters

```
In [57]: ┌─ univariate_comparison('NAME_INCOME_TYPE')
```

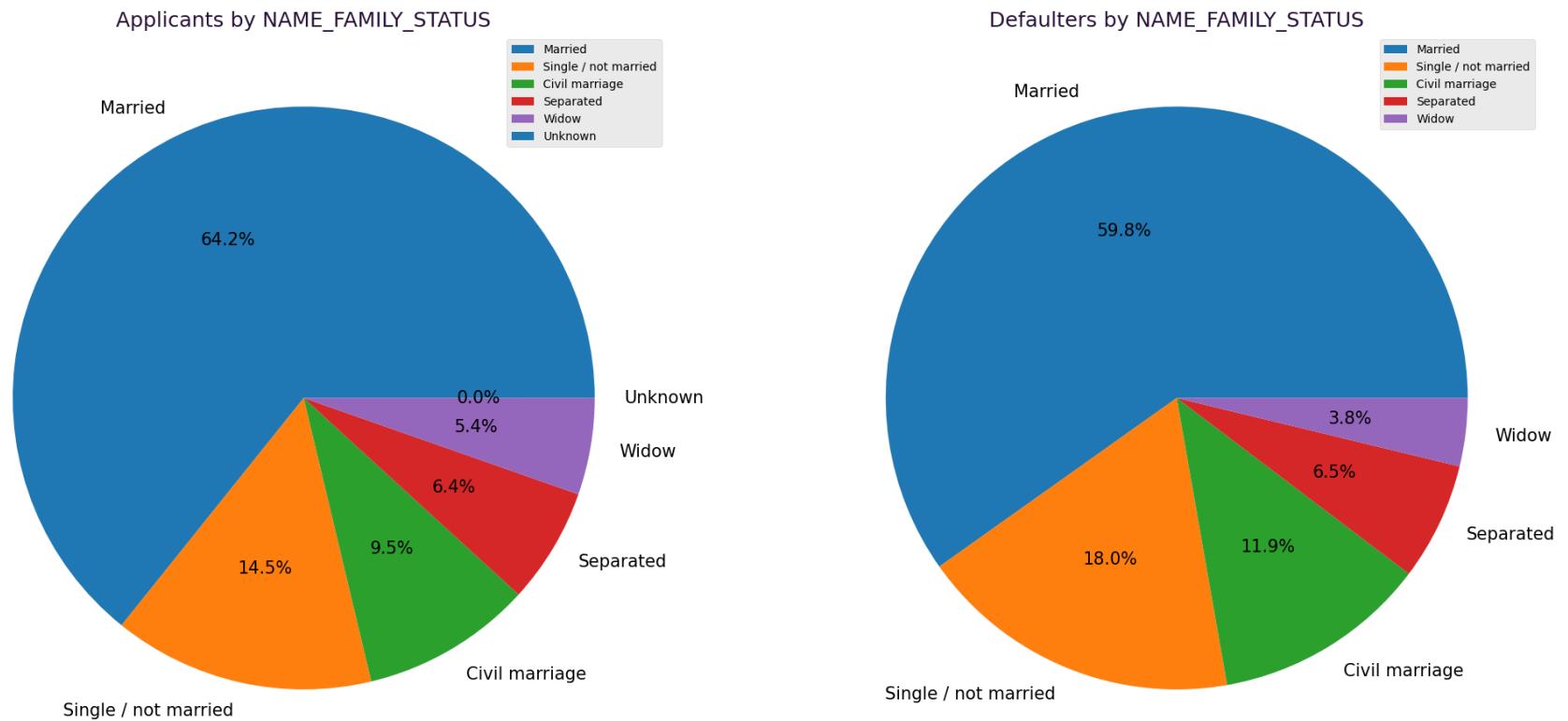


Insights -

- Almost half of the Loan applications come from Working professionals.
- Working professionals contribute more than expected to loan defaults. The % split has increased from 51% to 61%

Comparison of Family Status Distribution among Defaulters and Non Defaulters

```
In [58]: ┌─ univariate_comparison('NAME_FAMILY_STATUS')
```



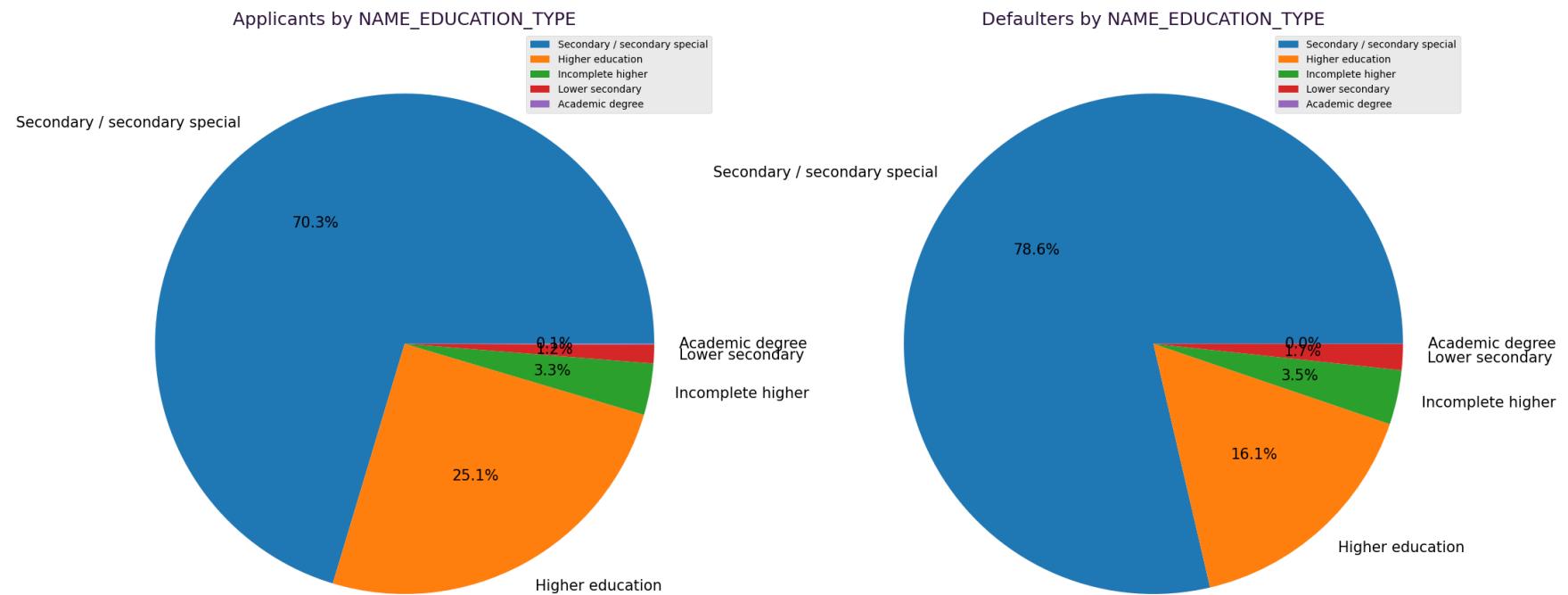
65 % of the Loan applicants are married.

Married professionals contribute to lesser in loan default rates.

Whereas Single professionals contribute slightly more for loan default rates as percentage increases from 14% to 18%

Comparison of Education Type Distribution among Defaulters and Non Defaulters

```
In [59]: ┌─ univariate_comparison('NAME_EDUCATION_TYPE')
```

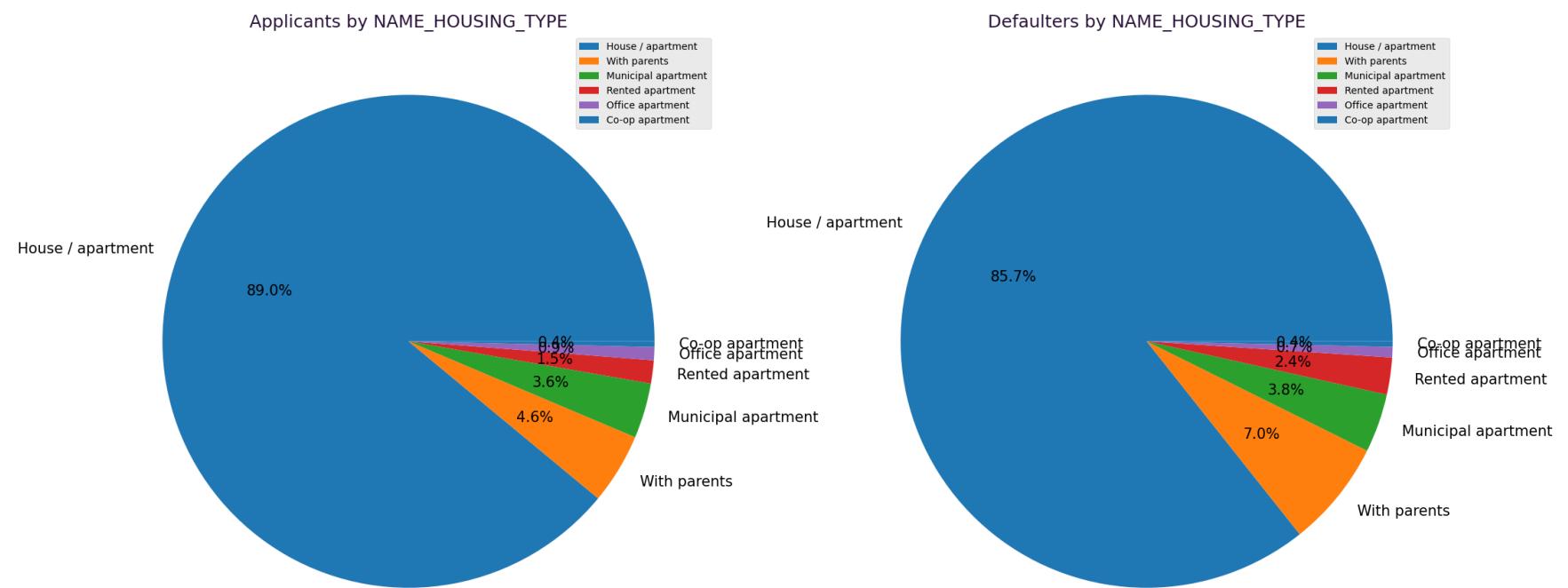


Insights-

- More than 2/3rds of Loan applicants have highest education as Secondary.
- Secondary Education class contribute majorly (more than expected too) for loan defaults.
- There is a considerable decrease in % split for loan defaults by people with higher education. (from 25% to 16%)

Comparison of Housing Type Distribution among Defaulters and Non Defaulters

```
In [60]: ┌─ univariate_comparison('NAME_HOUSING_TYPE')
```

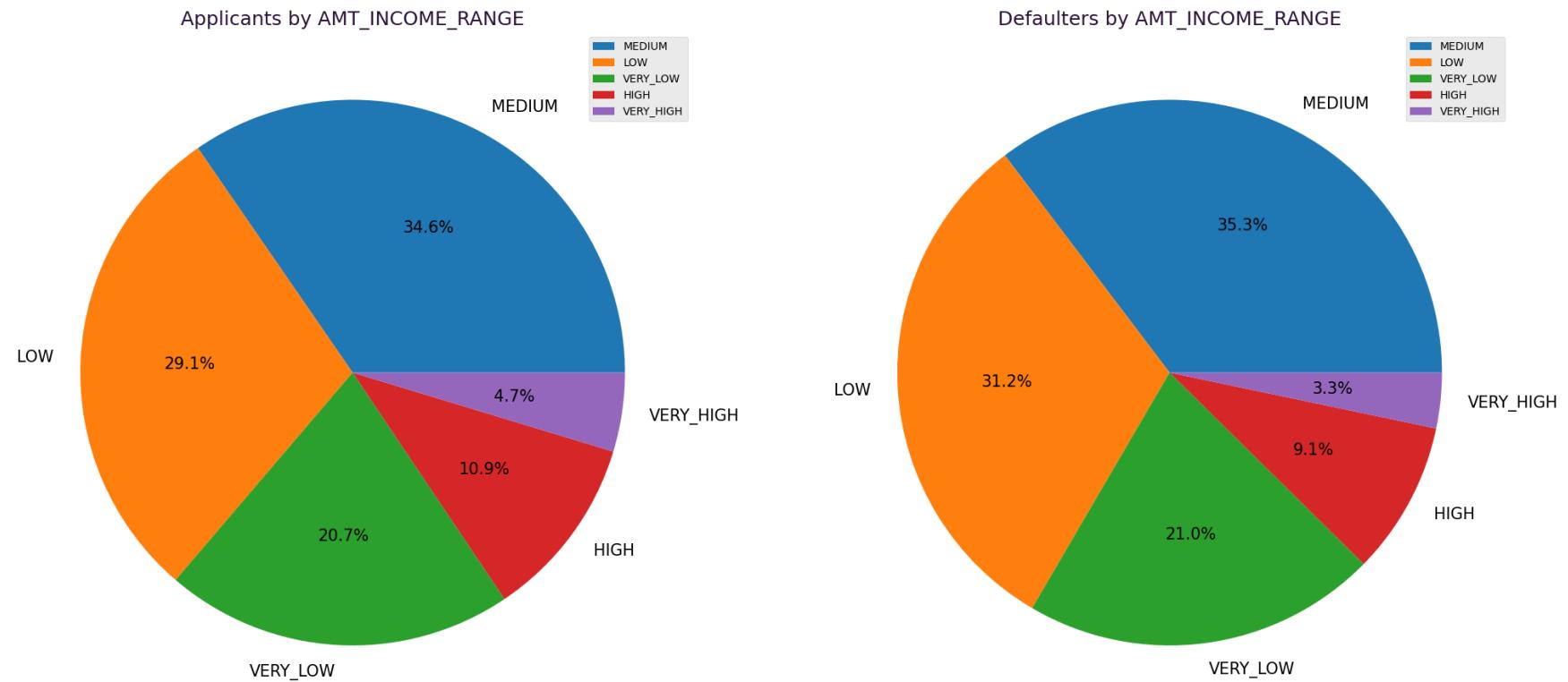


Almost 90% of Loan applicants have their own home.

Housing type doesn't play a significant role in determining whether there will be a loan defaulter.

Comparison of Income Range Distribution among Defaulters and Non Defaulters

```
In [61]: ┌─ univariate_comparison('AMT_INCOME_RANGE')
```

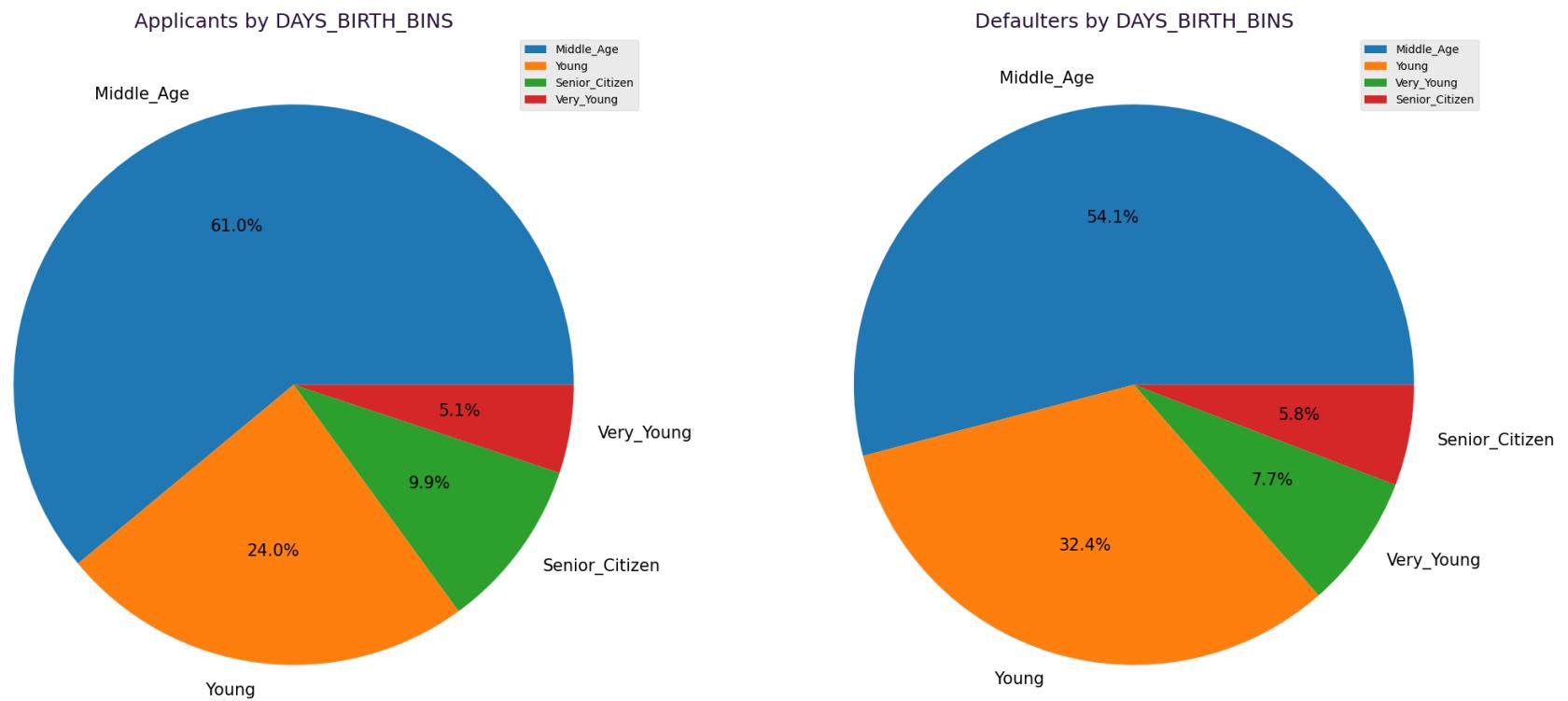


Insights-

- Here also, the % split is more or less unchanged for Defaulters. It suggests that Income doesn't play a significant role in loan defaults. Although, further drilldown analysis (later done in this notebook) would tell us a different story.

Comparison of Age Distribution among Defaulters and Non Defaulters

```
In [62]: ⏷ univariate_comparison('DAYS_BIRTH_BINS')
```

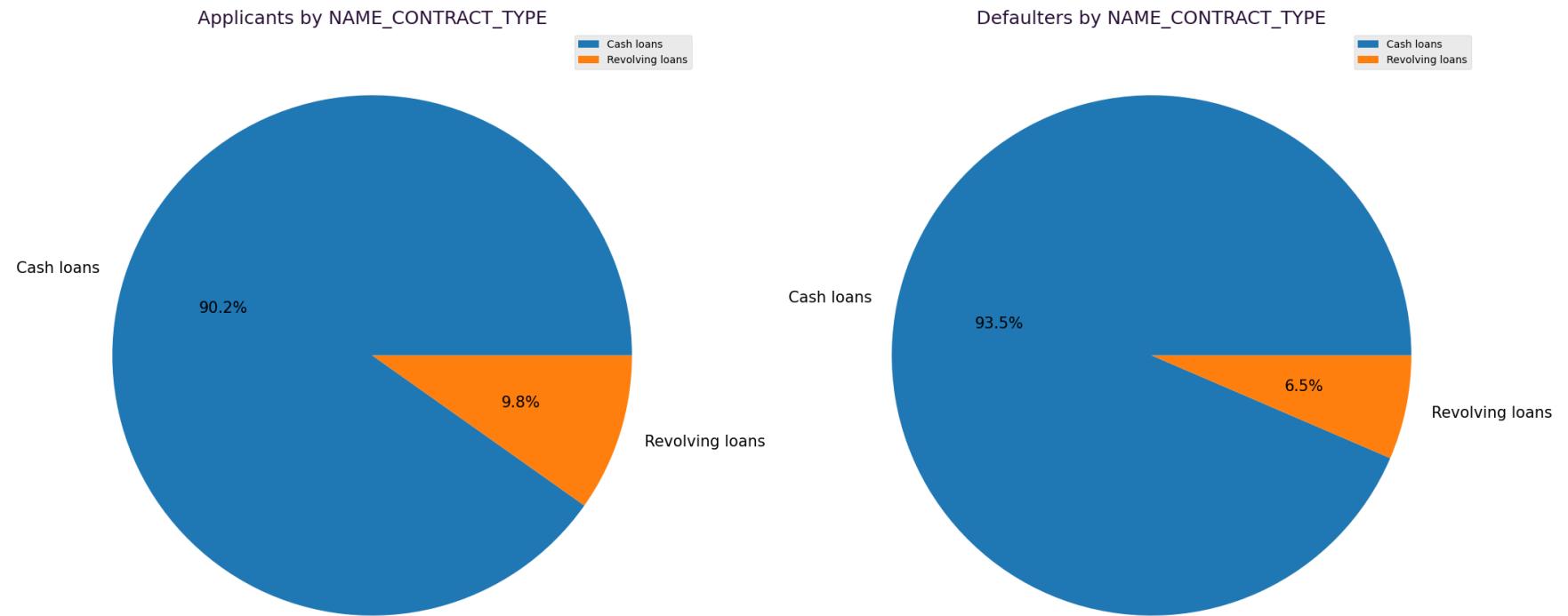


Insights -

- There is a significant shift in % split for Middle Age and Young applicants.
- Middle Aged applicants are contributing lesser to loan defaults
- Young applicants are more expected to default on a loan since there is a change in % split from 24% to 32%

Comparison of Loan Type Distribution among Defaulters and Non Defaulters

In [63]: ┌ univariate_comparison('NAME_CONTRACT_TYPE')

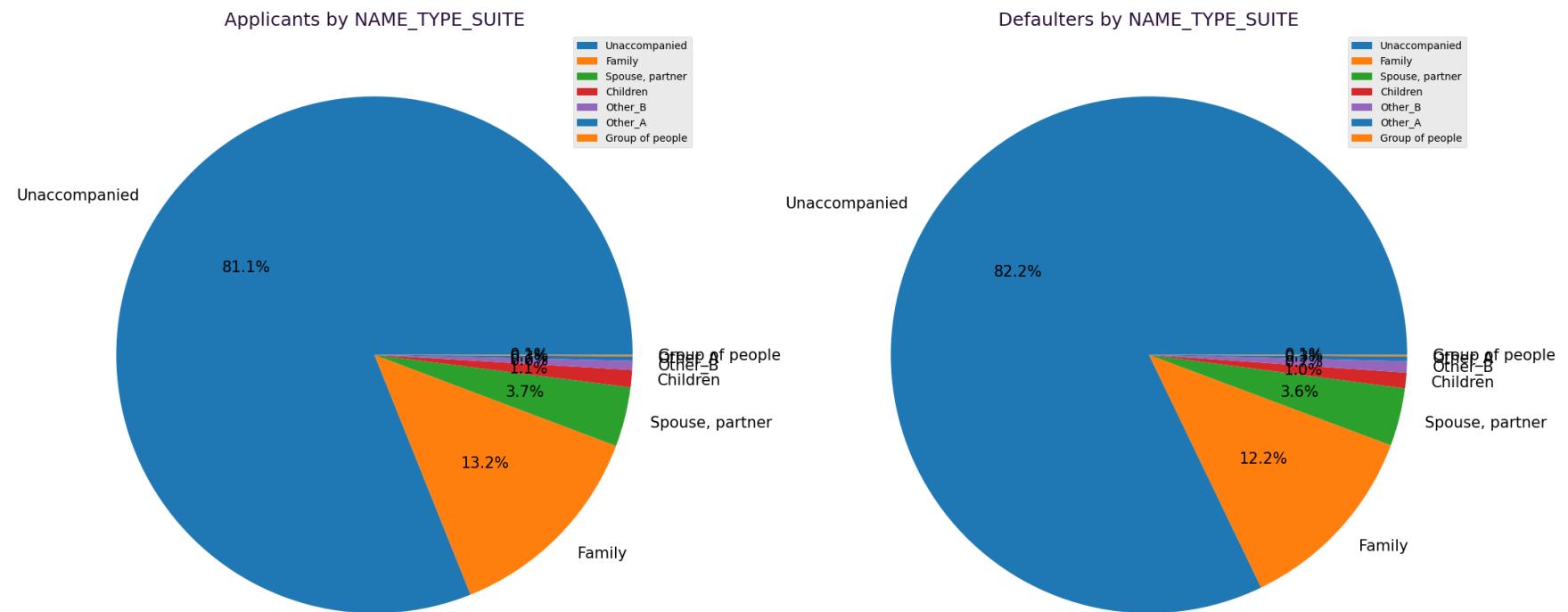


Insights-

- Cash loans are slightly more likely to be defaulted than revolving loans.

Comparison of Accompany Type Distribution among Defaulters and Non Defaulters

```
In [64]: ┌─ univariate_comparison('NAME_TYPE_SUITE')
```



Insights-

- Majority of loans are applied by single occupants
- This parameter doesn't have any impact on loan defaults as the % split is unchanged in both cases.

Univariate Analysis of Quantitative Variables

In [65]: # Defining function for Univariate Analysis of Quantitative Variables

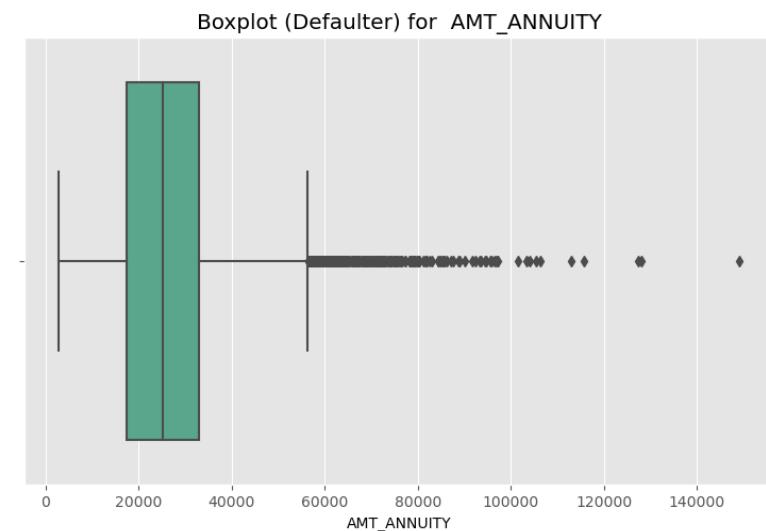
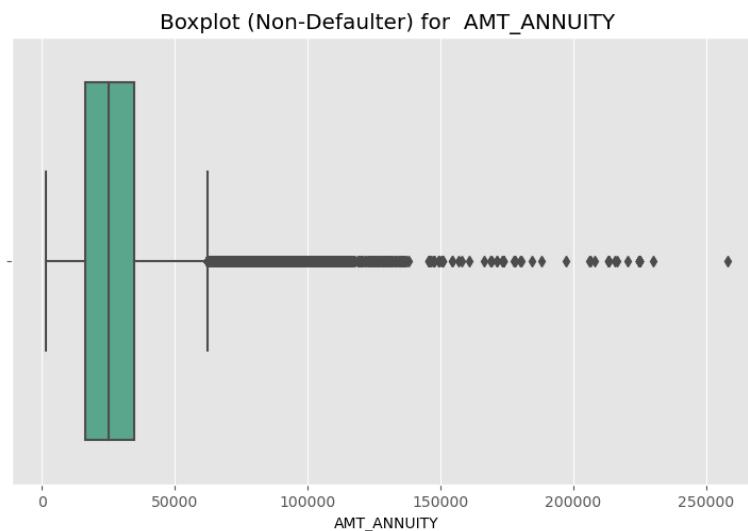
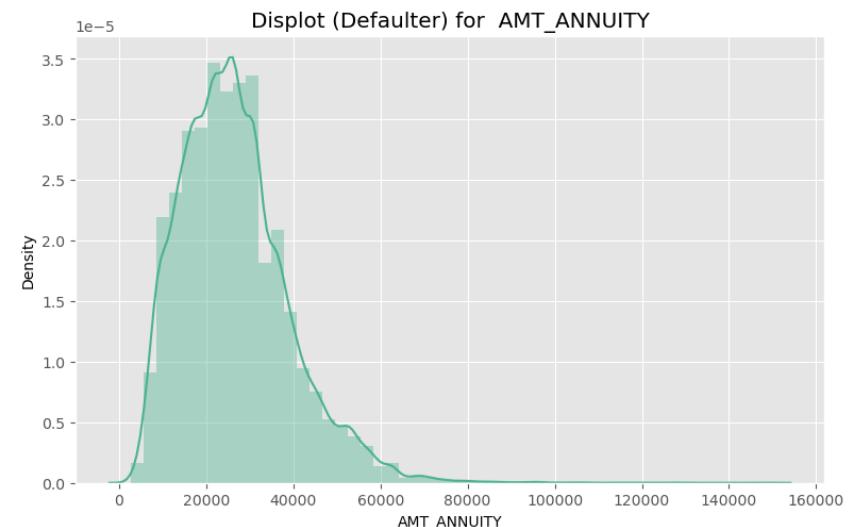
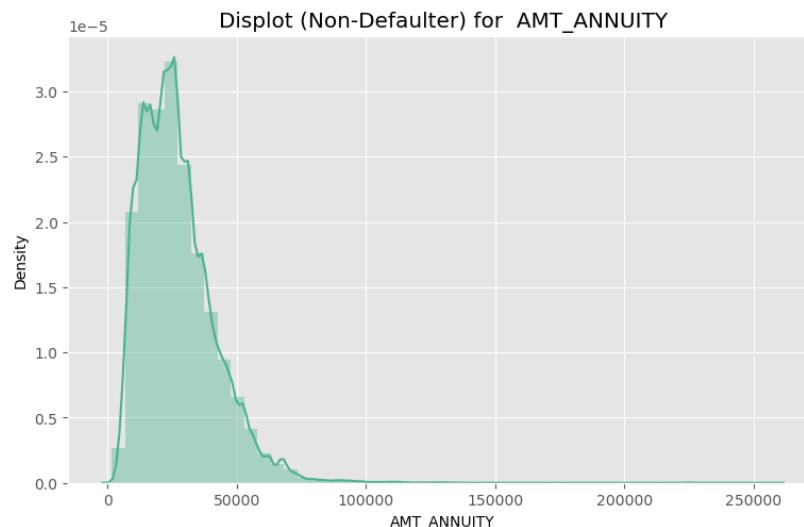
```
def univariate_comparison_quant(col,hue=None):  
  
    fig, axes=plt.subplots(nrows =2,ncols=2,figsize=(20,12))  
    axes[0,0].set_title("Displot (Non-Defaulter) for " + col )  
    sns.distplot(non_defaulter[~non_defaulter[col].isna()][col],ax=axes[0,0], color="#4CB391")  
  
    axes[0,1].set_title("Displot (Defaulter) for " + col )  
    sns.distplot(defaulter[~defaulter[col].isna()][col],ax=axes[0,1], color="#4CB391")  
  
    axes[1,0].set_title("Boxplot (Non-Defaulter) for " + col )  
    sns.boxplot(x=non_defaulter[~non_defaulter[col].isna()][col],ax=axes[1,0], color="#4CB391")  
  
    axes[1,1].set_title("Boxplot (Defaulter) for " + col )  
    sns.boxplot(x=defaulter[~defaulter[col].isna()][col],ax=axes[1,1], orient='h',color="#4CB391")  
  
    plt.show()
```

In [66]: defaulter['AMT_ANNUITY']

```
Out[66]: 0      24700.5  
26     27076.5  
40     35028.0  
42     16258.5  
81     14593.5  
      ...  
307448   32746.5  
307475   46809.0  
307481   19975.5  
307489   23089.5  
307509   20205.0  
Name: AMT_ANNUITY, Length: 24825, dtype: float64
```

In [67]: # Univariate Analysis for Annuity Amount

```
univariate_comparison_quant('AMT_ANNUITY')
```

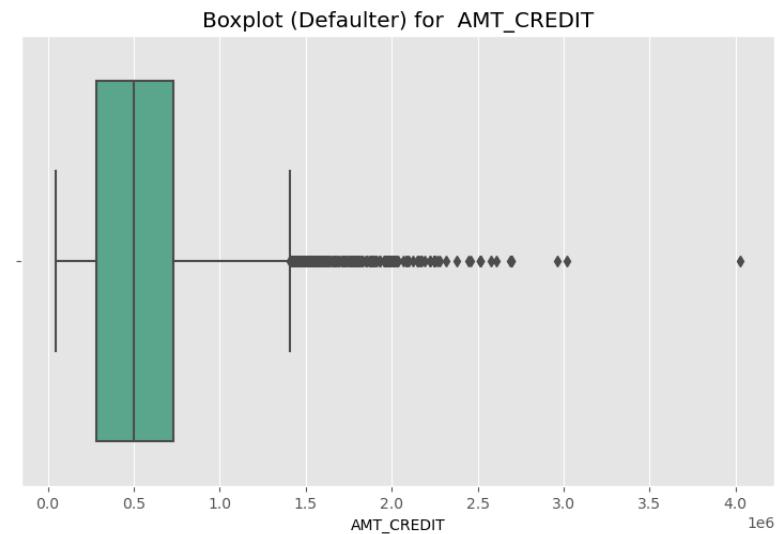
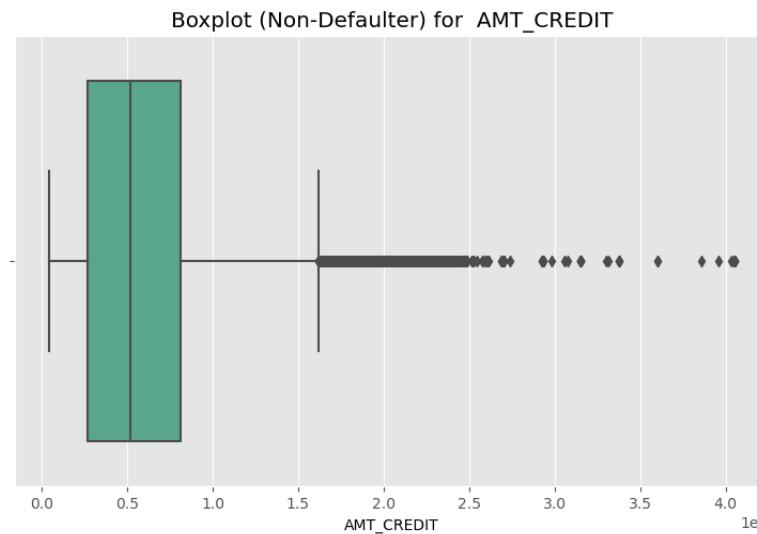
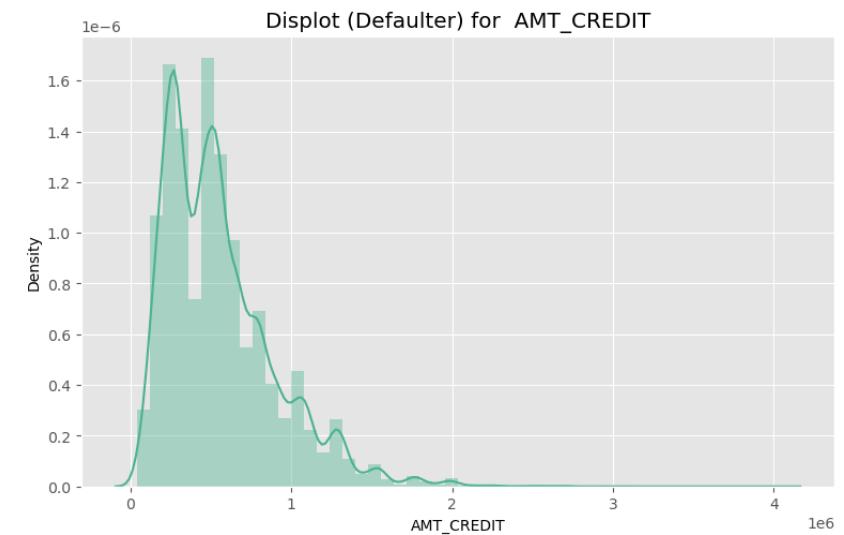
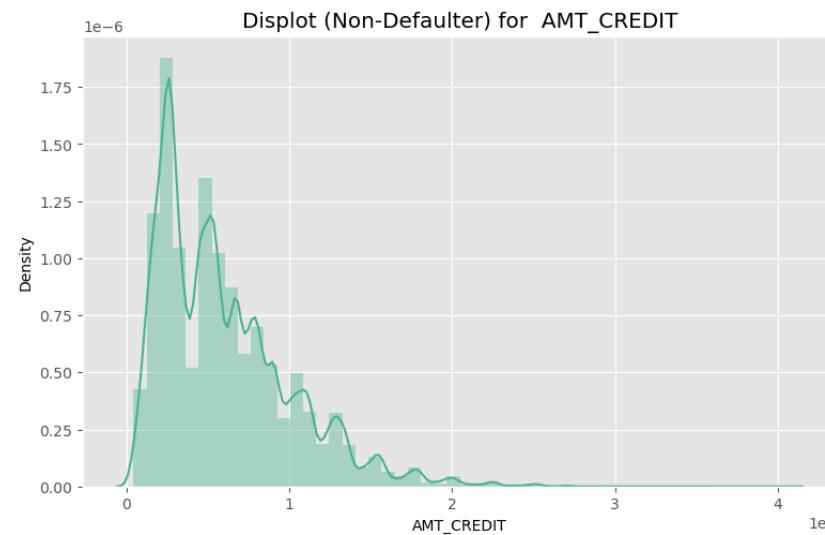


Insights -

- Applicants with lower Annuity Amount are slightly more likely to default on a loan.
- Majority of Loan applicants come from 1st quartile of Annuity data (Low salary people)

In [68]: # Univariate Analysis for Loan Amount

```
univariate_comparison_quant('AMT_CREDIT')
```

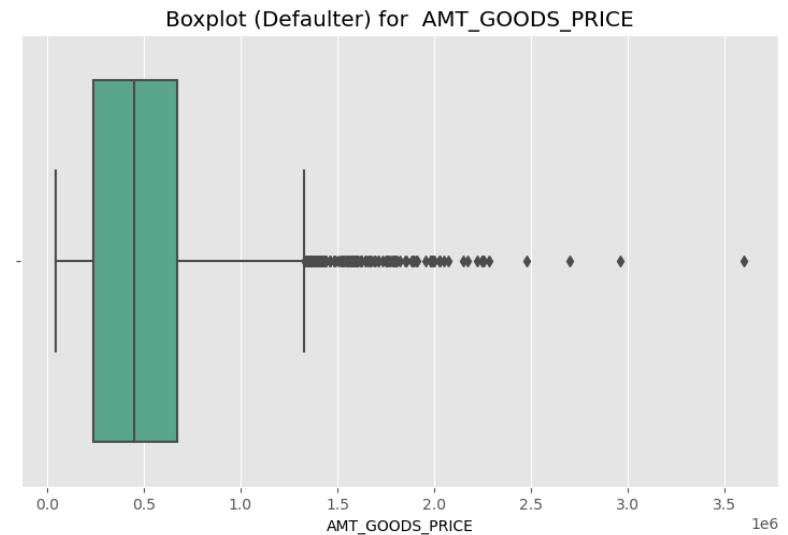
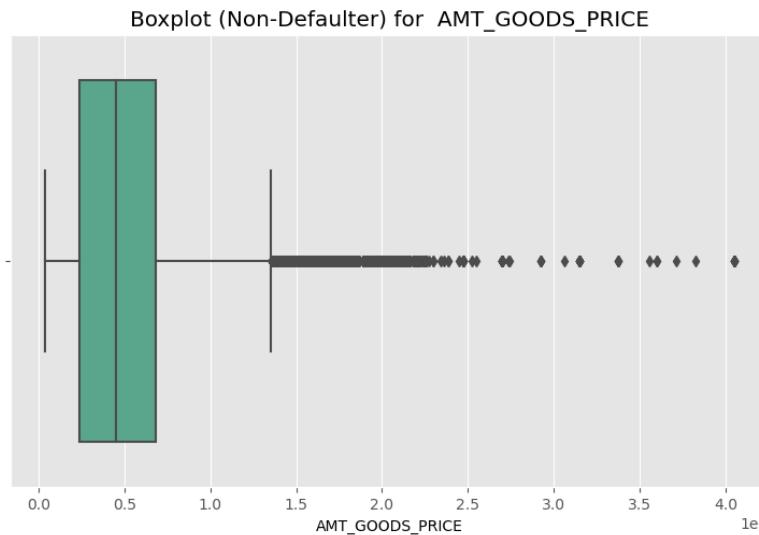
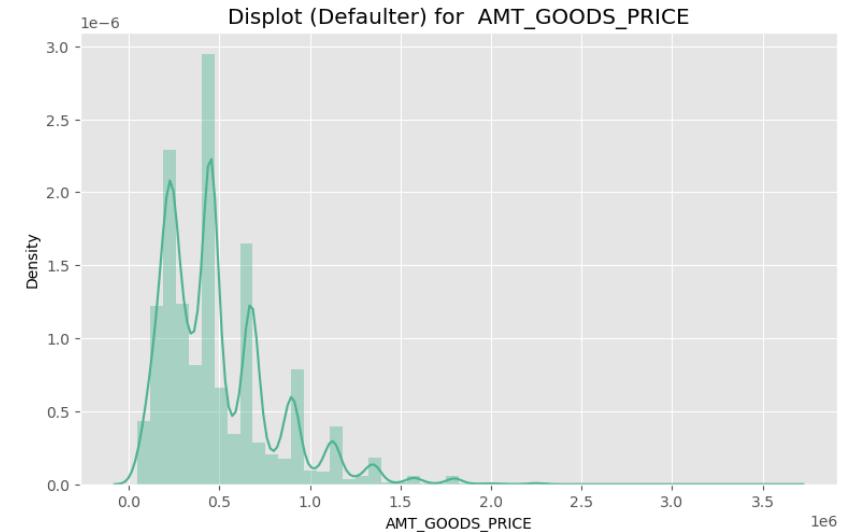
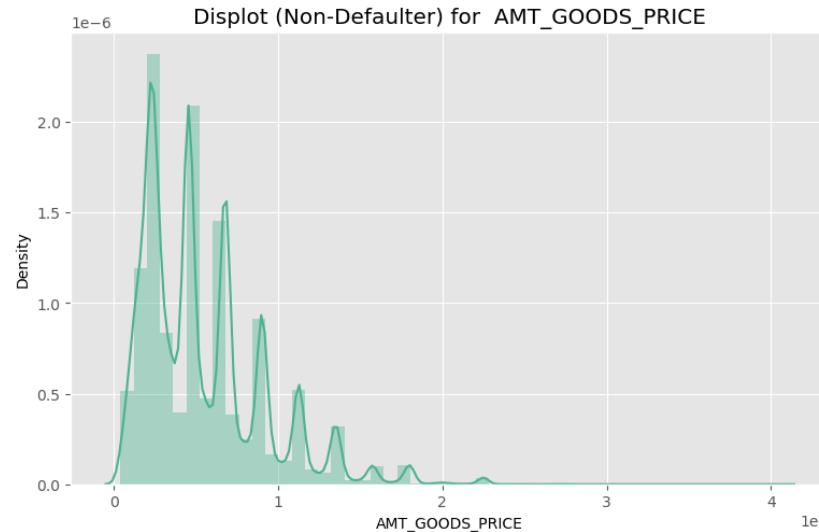


Insights-

- We can infer that Loan Amount doesn't correlate with Loan defaults since the Loan Amount has the exact quartile boundaries in two cases.

In [69]: # Univariate Analysis for Goods Price Amount

```
univariate_comparison_quant(col='AMT_GOODS_PRICE')
```



Insights-

- The distribution are almost unchanged for Defaulters and Non Defaulters, hence we can say that Goods Price doesn't impact the chance of a loan default.

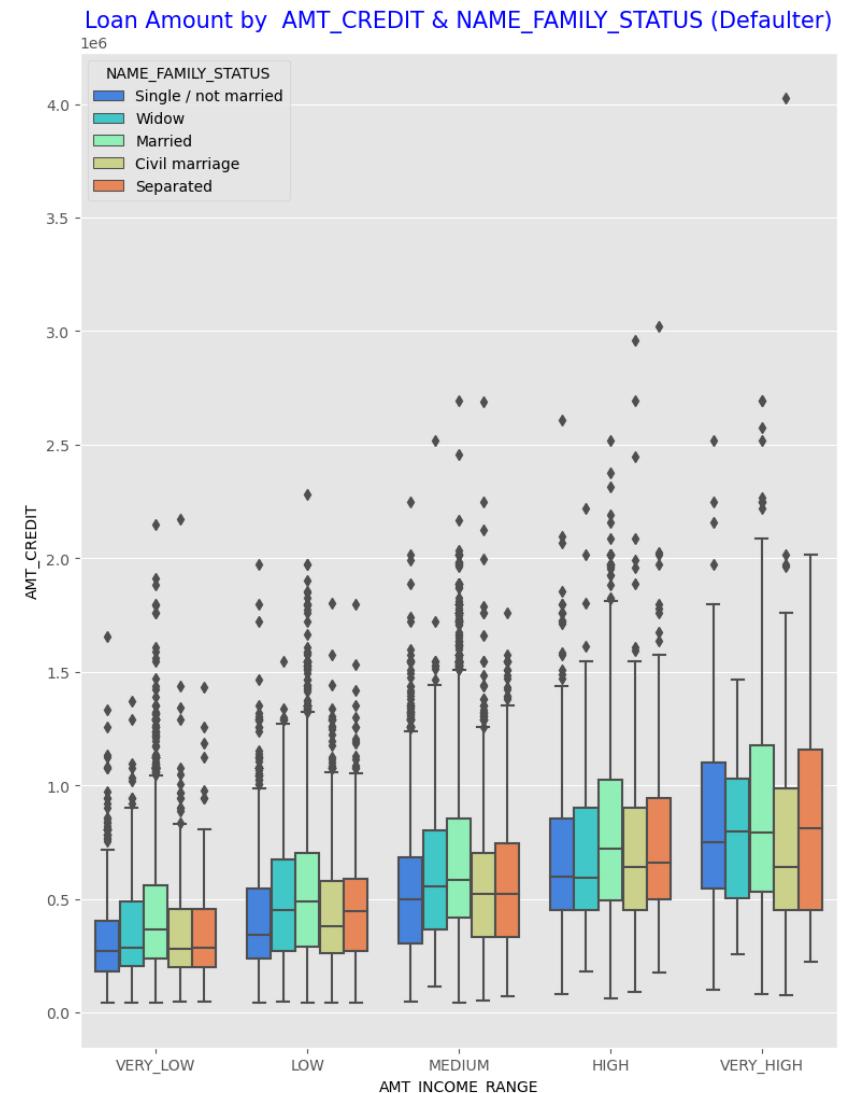
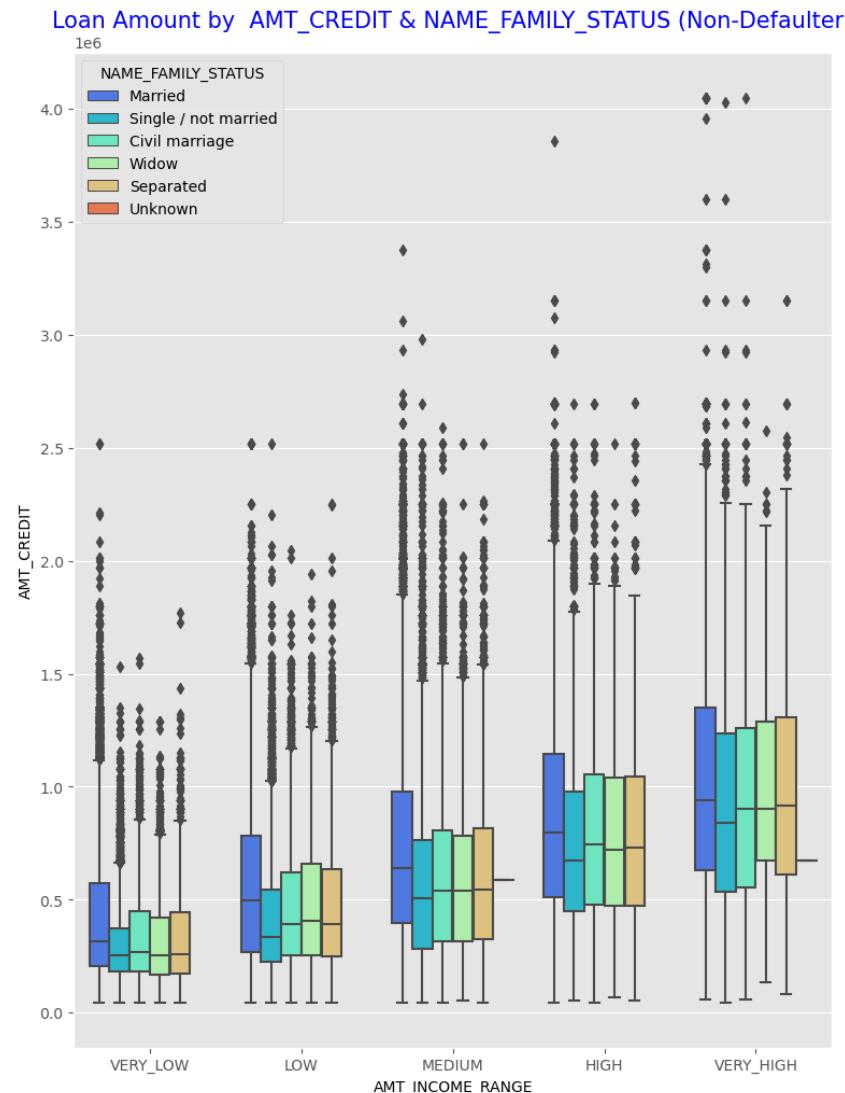
Bivariate & Multivariate Analysis

In [70]: ► # Function for Multivariate analysis

```
def multivariate(col1,col2,col3=None):  
  
    fig, axes=plt.subplots(nrows =1,ncols=2,figsize=(20,12))  
  
    axes[0].set_title('Loan Amount by ' + col2 + ' & ' + col3 + ' (Non-Defaulter)', size=15,color = 'blue')  
    sns.boxplot(data=non_defaulter,x=col1, y=col2,palette = 'rainbow', hue= col3,ax=axes[0])  
  
    axes[1].set_title('Loan Amount by ' + col2 + ' & ' + col3 + ' (Defaulter)', size=15,color = 'blue')  
    sns.boxplot(data=defaultter,x=col1, y=col2,palette = 'rainbow', hue= col3,ax=axes[1])
```

In [71]: # Analysis of AMT_INCOME_RANGE, AMT_CREDIT & NAME_FAMILY_STATUS

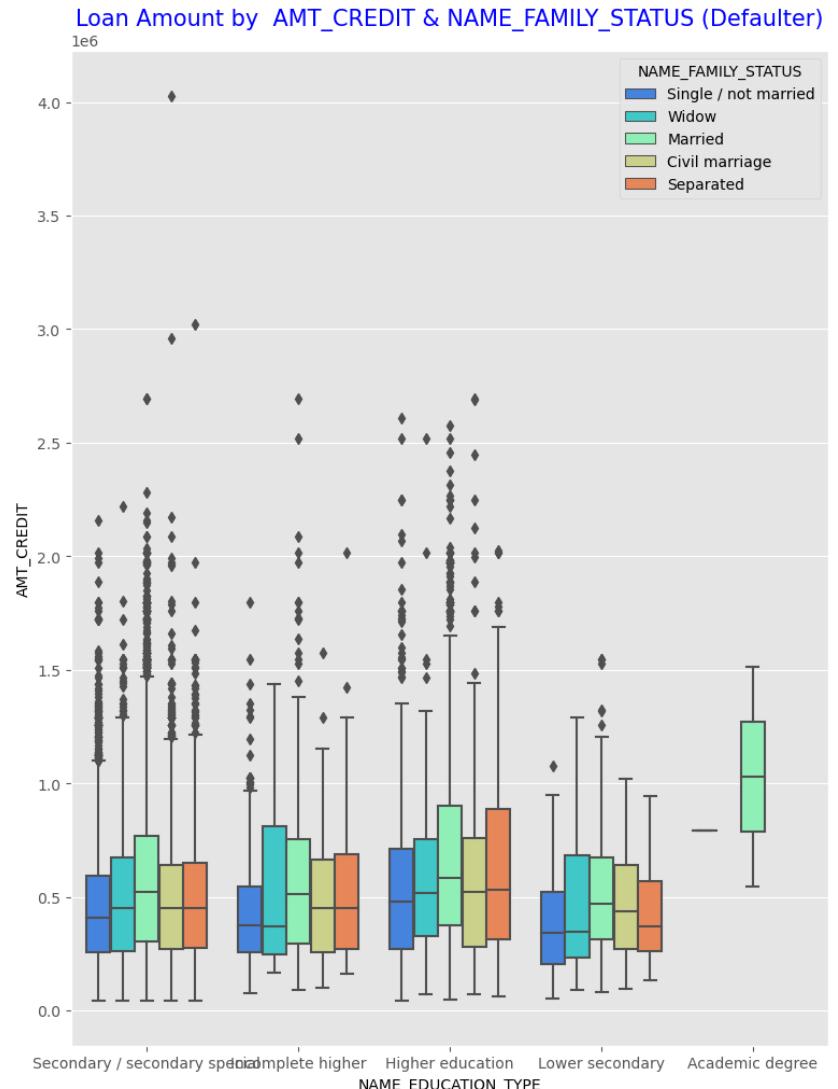
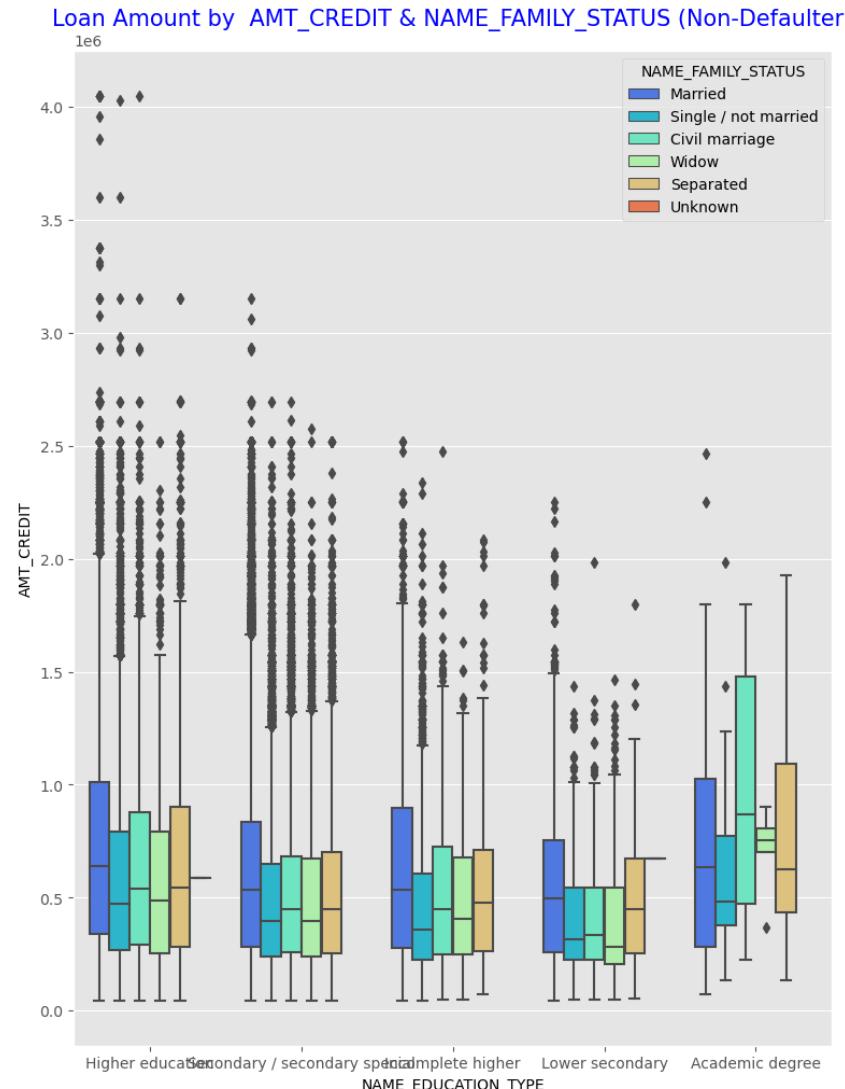
```
multivariate('AMT_INCOME_RANGE', 'AMT_CREDIT', 'NAME_FAMILY_STATUS')
```



Insights-

- With increase in Income range, the loan amount increases proportionally.
- On family status axis, we observe that Married applicants have higher loan amount than others.

In [72]: # Analysis of NAME_EDUCATION_TYPE, AMT_CREDIT & NAME_FAMILY_STATUS
multivariate('NAME_EDUCATION_TYPE', 'AMT_CREDIT', 'NAME_FAMILY_STATUS')



Insights-

- Higher the education, lesser is the likelihood of a loan default
- Among different family status, married ones have the highest likelihood of loan default

Drilldown Analysis

Here we'll look for % defaulters within different classes in a particular variable.

In [115]: ⏷ credit_data_2.head()

Out[115]:

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

In [87]: # Defining function for drilldown analysis

```
def perc_defaulters(col):

    fig, axes=plt.subplots(nrows =1,ncols=2,figsize=(20,10))

    total = credit_data_2[[col, 'TARGET']].groupby(col).count()
    defaulter_1 = defaulter[[col, 'TARGET']].groupby(col).count()
    perc = defaulter_1*100/total

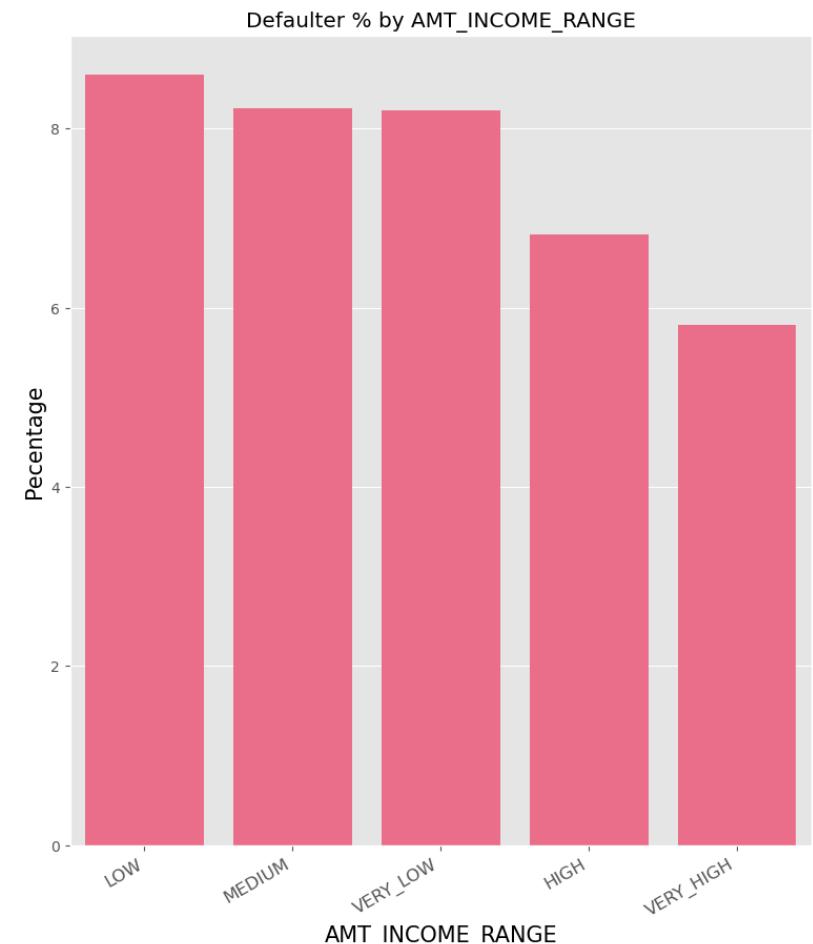
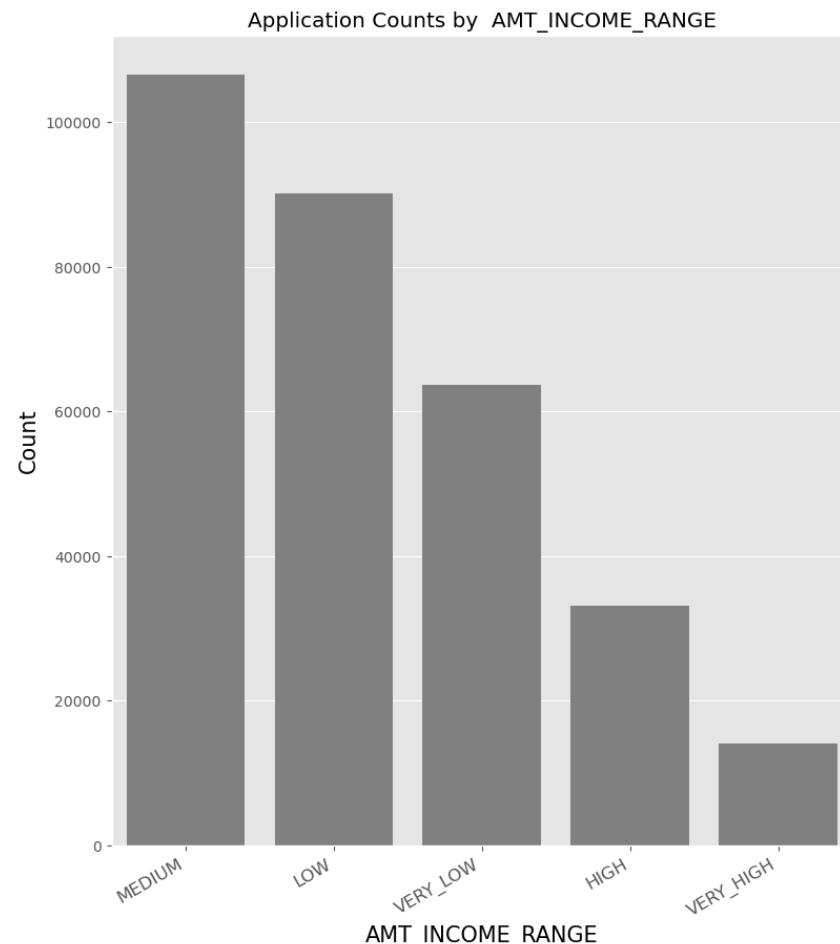
    axes[0].set_title("Application Counts by " + col )
    sns.barplot(x=total.index,y=total.TARGET,color='grey',order=total.sort_values('TARGET',ascending=False).index,
    axes[0].set_xticklabels(total.sort_values('TARGET',ascending=False).index,rotation=30, ha='right',size=12)
    axes[0].set_xlabel(col,size=15)
    axes[0].set_ylabel("Count",size=15)

    axes[1].set_title("Defaulter % by " + col )
    sns.barplot(x=perc.index,y=perc.TARGET,color='#ff597d',order=perc.sort_values('TARGET',ascending=False).index,
    axes[1].set_xticklabels(perc.sort_values('TARGET',ascending=False).index,rotation=30, ha='right',size=12)
    axes[1].set_xlabel(col,size=15)
    axes[1].set_ylabel('Percentage',size=15)

    plt.show()
```

In [88]: # Drilldown analysis of AMT_INCOME_RANGE

```
perc_defaulters('AMT_INCOME_RANGE')
```

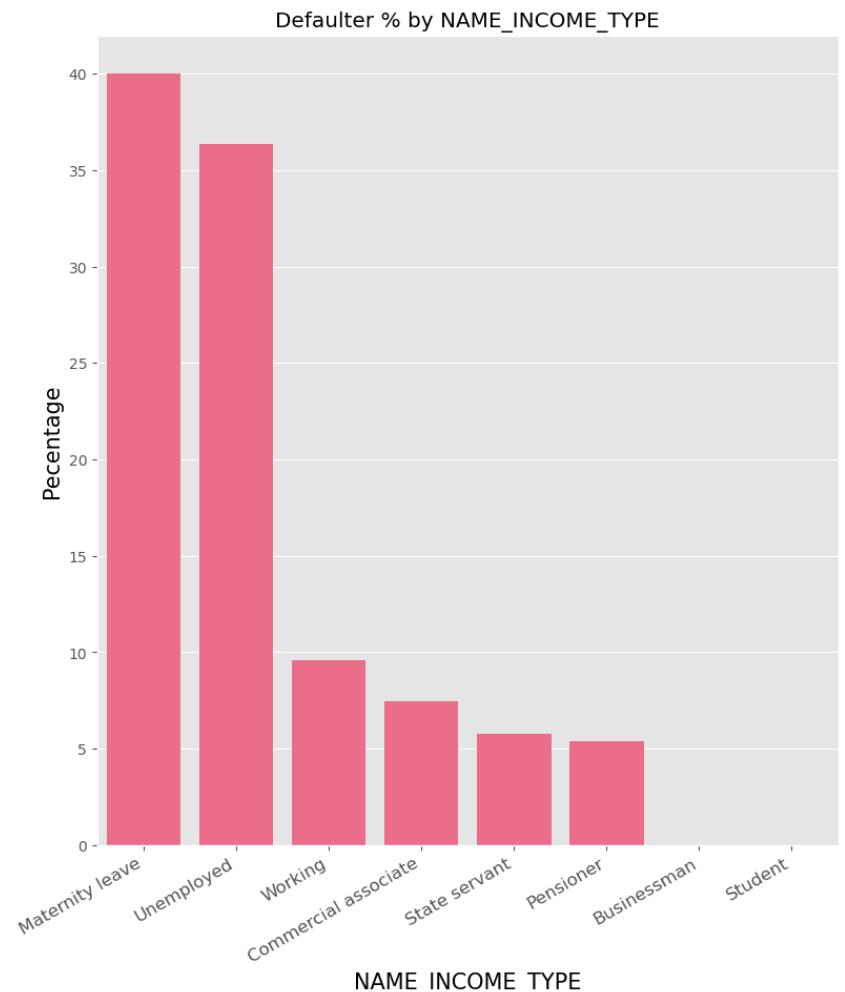
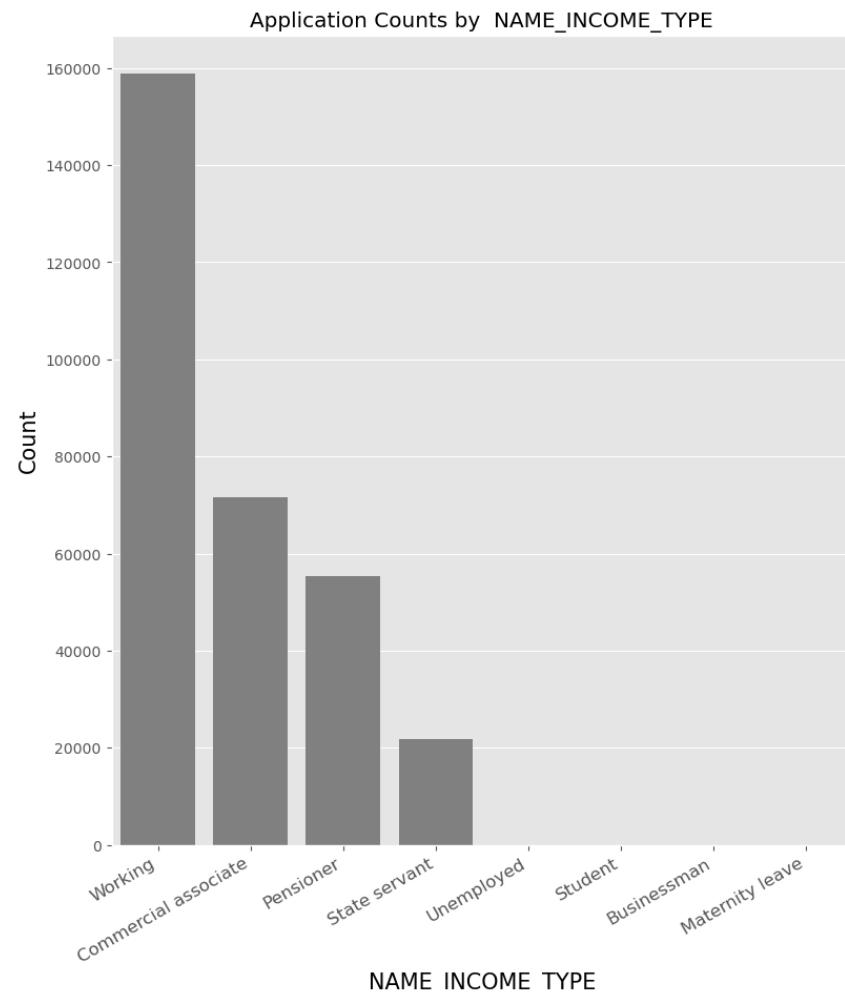


Insights-

- Medium income range professionals have maximum applications in the data
- Low Income range have maximum % of loan defaults
- As the Income range increases, loan default probability decreases

In [89]: # Drilldown analysis of NAME_INCOME_TYPE

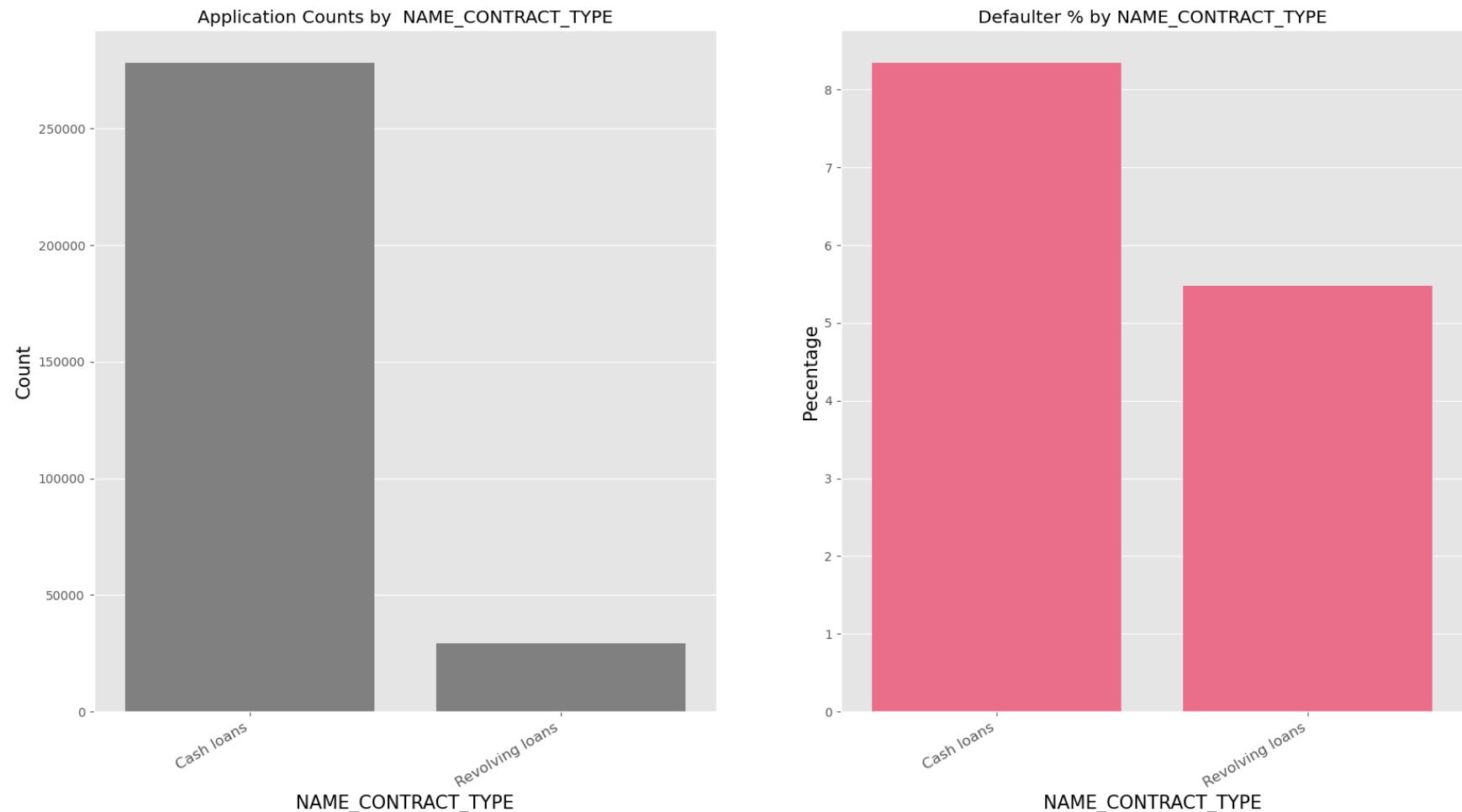
```
perc_defaulters('NAME_INCOME_TYPE')
```



Insights-

- Applicants on Maternity leave have a whopping 40% loan default rate
- The second to the list are Unemployed applicants with 35% loan defaults

In [90]: # Drilldown analysis of NAME_CONTRACT_TYPE
perc_defaulters('NAME_CONTRACT_TYPE')

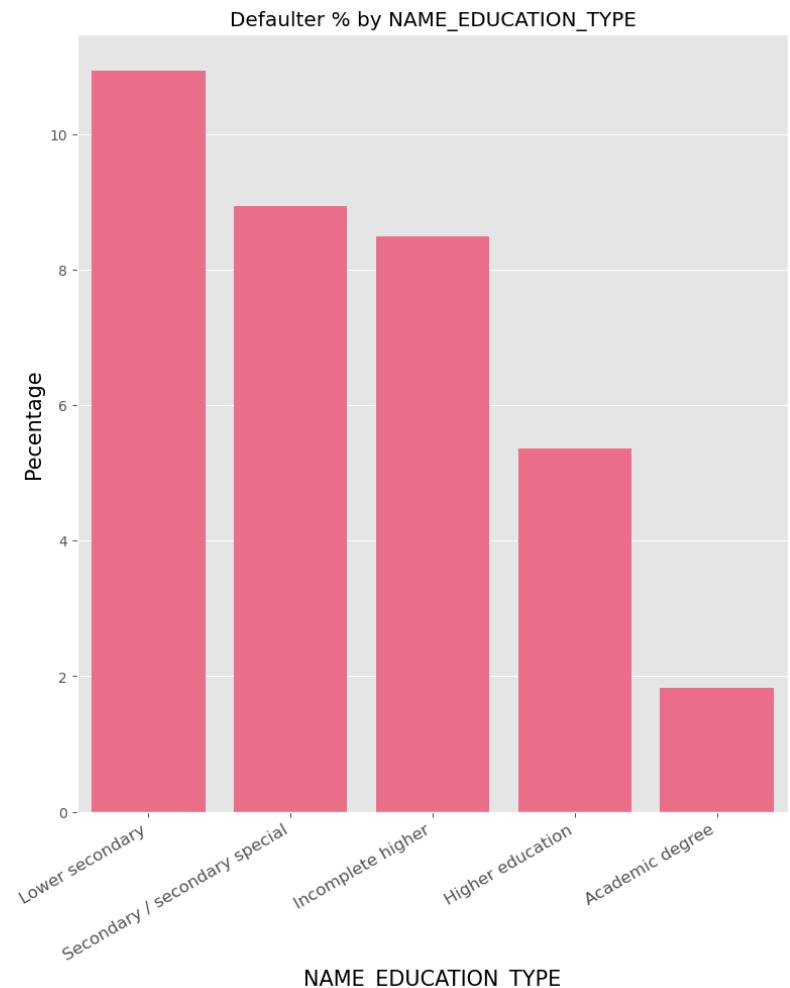
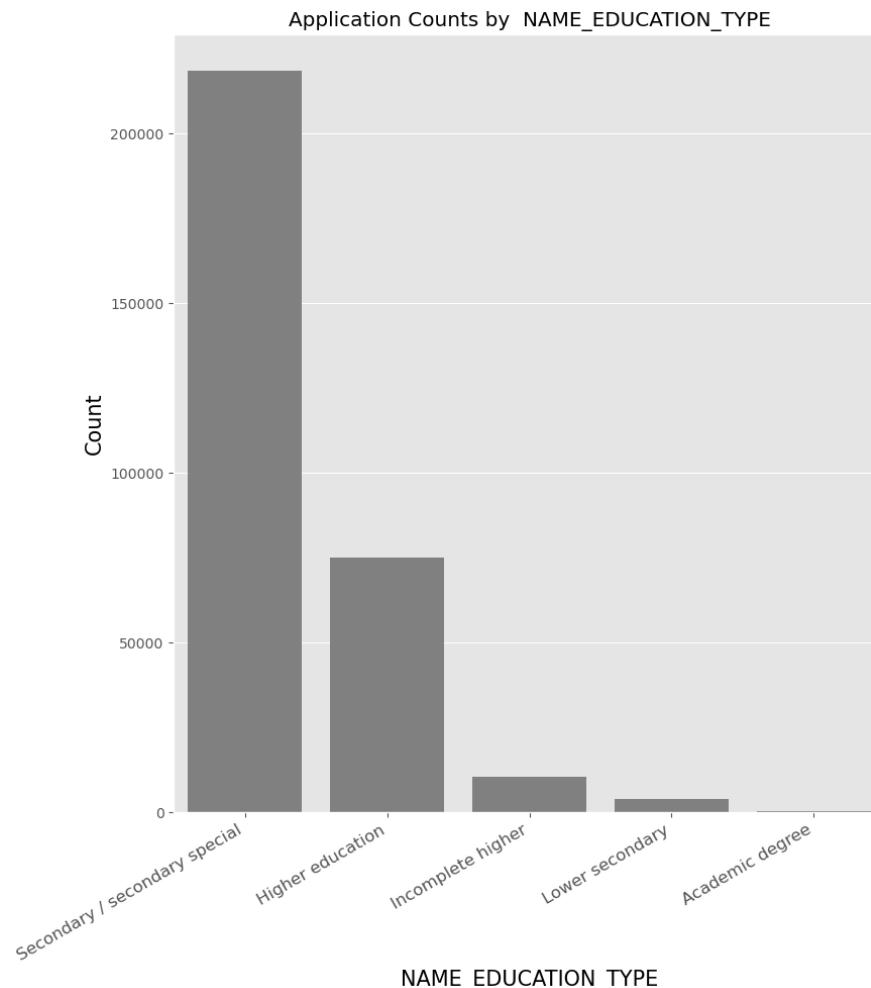


Insights-

- Majority of the loans are cash loans. Cash loans also have almost double probability of a loan default than revolving loans.

In [93]: # Drilldown analysis of NAME_EDUCATION_TYPE

```
perc_defaulters('NAME_EDUCATION_TYPE')
```

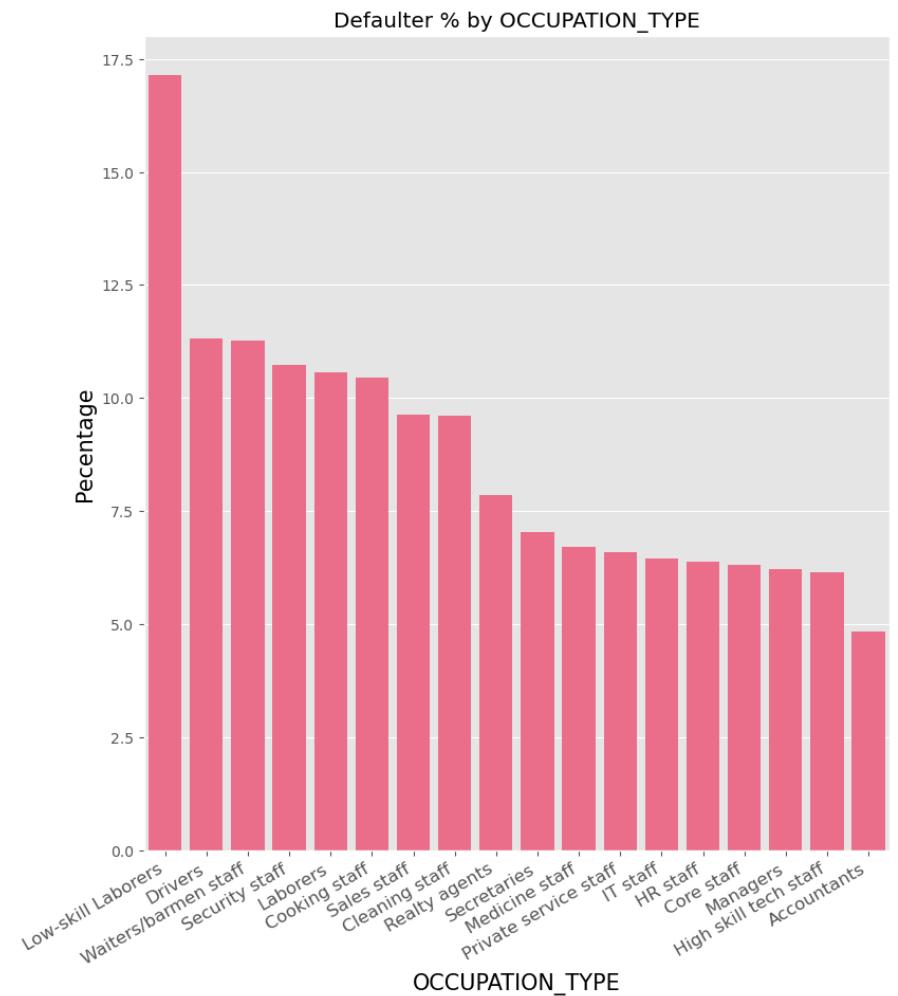
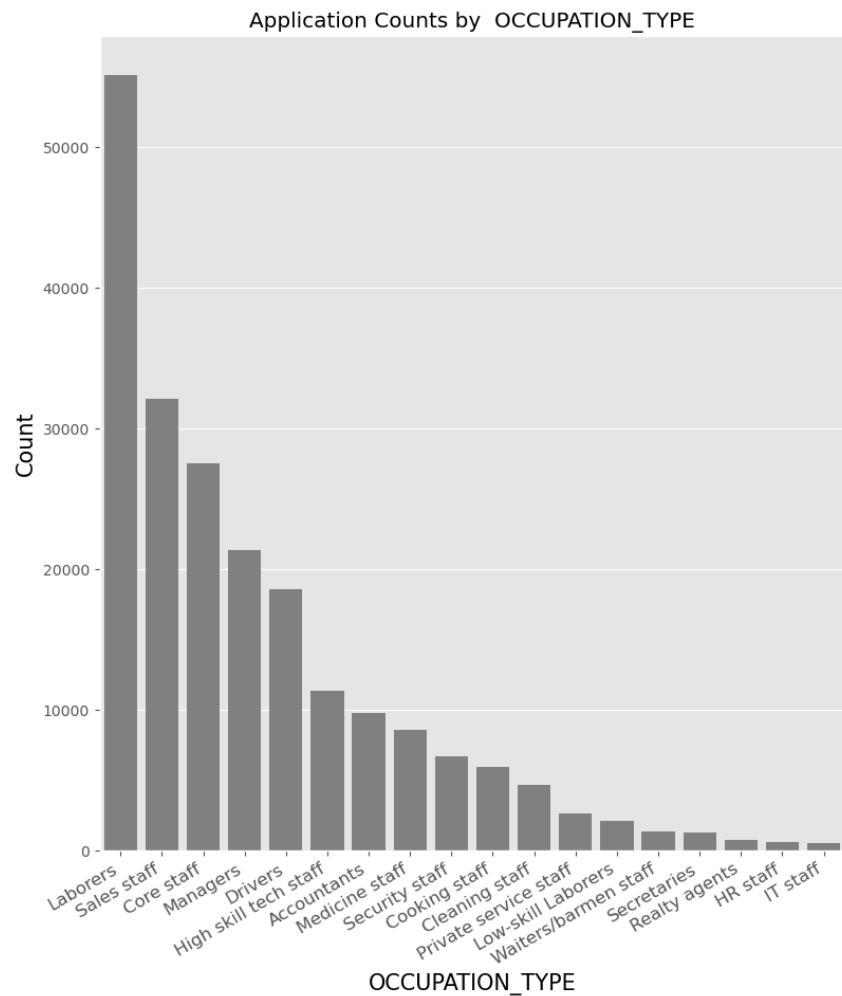


Insights-

- Higher the education of an applicant, lesser the chance of loan default
- Lower secondary applicants have a concerning 11% loan default rate, but the count of applicants is low
- The major concern is of Secondary education applicants. They have highest applicants and a significant 9% loan default rate as well.

In [94]: # Drilldown analysis of OCCUPATION_TYPE

```
perc_defaulters('OCCUPATION_TYPE')
```



Insights-

- Low skill labourers have an alarming 17% loan default rate. The positive here is that they don't have a high applicant count.
- Labourers & Sales staff will be a major area of concern here, with maximum applicants and a significant loan default rate as well.
- Drivers also have an alarming combination of counts and default %.

Pivot table of all loan default %

```
In [95]: perc_defaulters= pd.pivot_table(data=credit_data_2, values='TARGET', index=['CODE_GENDER','AMT_INCOME_RANGE'],
                                         columns=['NAME_EDUCATION_TYPE'], aggfunc=np.mean)
perc_defaulters*100
```

Out[95]:

		NAME_EDUCATION_TYPE	Academic degree	Higher education	Incomplete higher	Lower secondary	Secondary / secondary special
	CODE_GENDER	AMT_INCOME_RANGE					
F		VERY_LOW	0.000000	5.606793	8.639863	8.019324	7.677801
		LOW	0.000000	4.902183	8.007537	11.388889	7.952316
		MEDIUM	0.000000	5.025389	7.843137	9.698276	7.569169
		HIGH	10.526316	4.151552	7.431341	3.896104	7.073552
		VERY_HIGH	7.692308	3.728906	8.225108	6.666667	6.593002
M		VERY_LOW	0.000000	8.041061	12.396694	12.500000	11.806626
		LOW	0.000000	7.330468	9.777778	14.285714	12.369265
		MEDIUM	0.000000	7.008598	9.513024	15.051546	11.346642
		HIGH	0.000000	5.591114	7.462687	8.163265	9.348442
		VERY_HIGH	0.000000	4.407996	7.758621	6.451613	8.993853

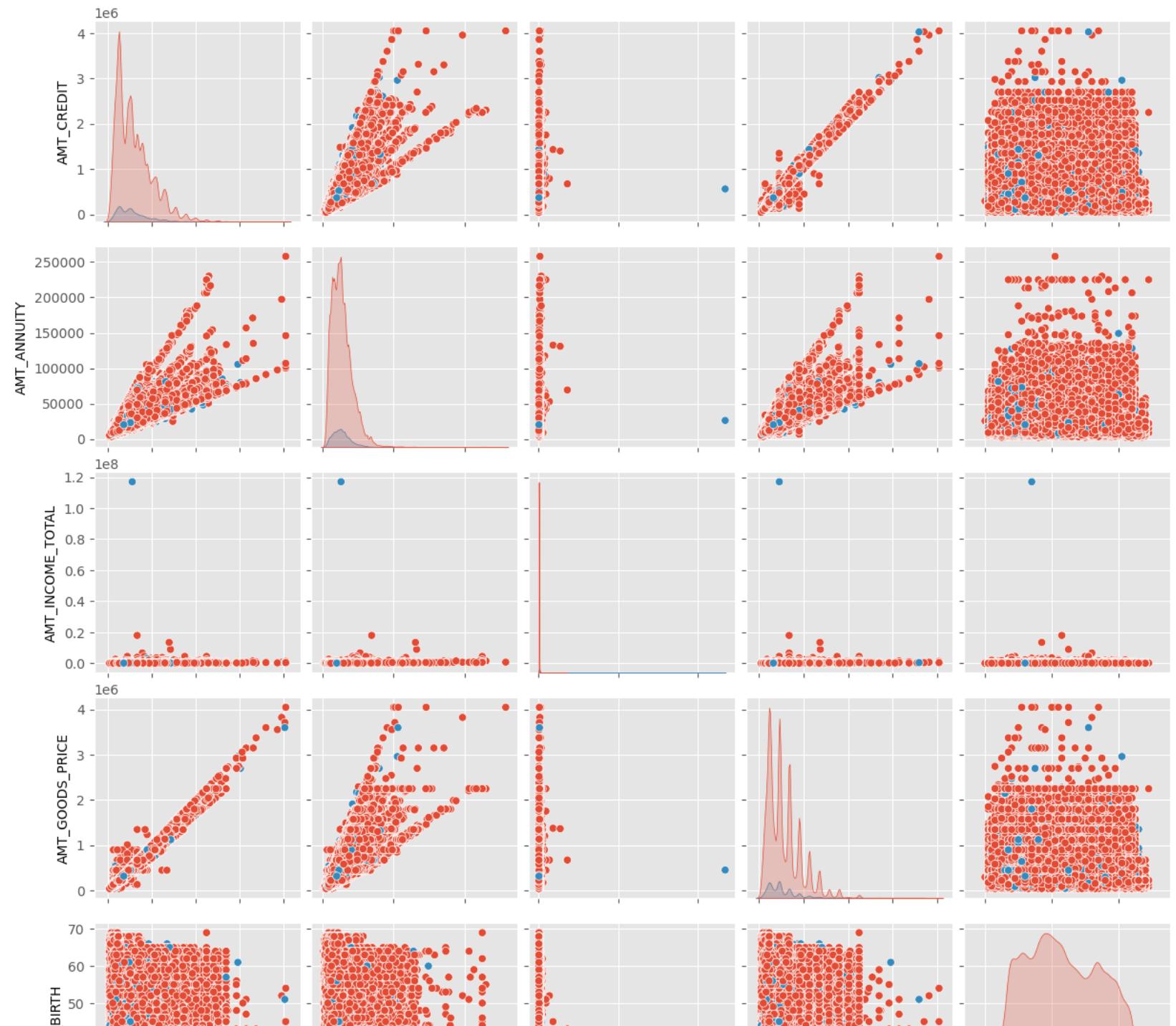
Insights -

Categories with more than 9% default rate -

- Females, High Income, Academic degree
- Male, Very Low income , Incomplete higher
- Male, Low Income , Incomplete higher
- Male, Medium Income , Incomplete higher
- Female, Low Income, Lower Secondary
- Female, Medium Income, Lower Secondary
- Male, Very Low Income, Lower Secondary
- Male, Low Income, Lower Secondary
- Male, Medium Income, Lower Secondary
- Male, {ALL INCOME RANGES} , Secondary

```
In [111]: pairplot_data=credit_data_2[['AMT_CREDIT', 'AMT_ANNUITY', 'AMT_INCOME_TOTAL', 'AMT_GOODS_PRICE', 'DAYS_BIRTH', 'TARGET']].fillna(0)
```

```
In [110]: sns.pairplot(pairplot_data,hue='TARGET',diag_kind='kde')
plt.show()
```

**Insights:**

AMT_CREDIT & AMT_GOODS_PRICE are correlated (With higher-priced goods, the loan amount is higher)

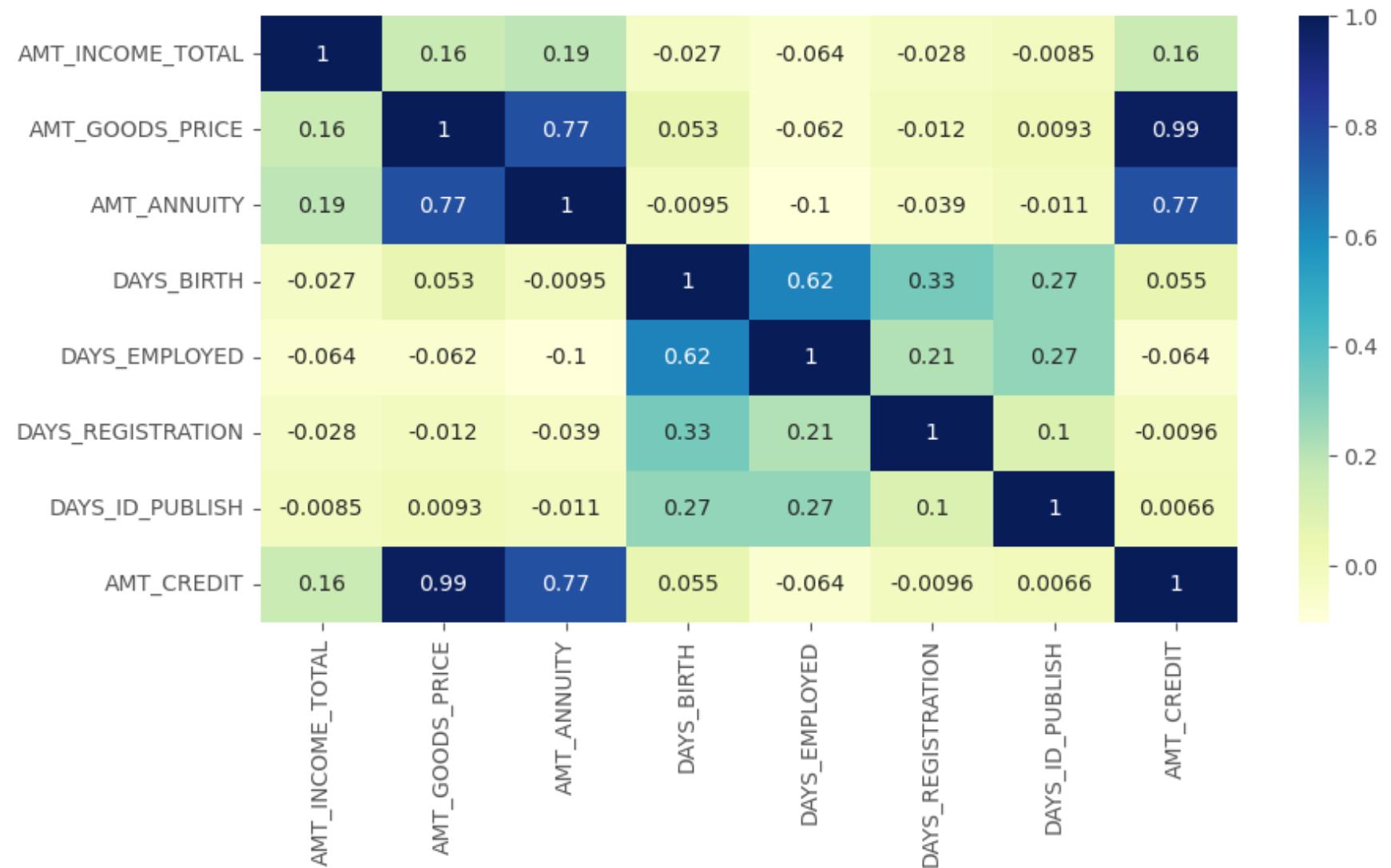
AMT_ANNUITY & AMT_GOODS_PRICE are also correlated (With a higher annuity, expensive goods are purchased)

AMT_ANNUITY & AMT_CREDIT are correlated (Higher the annuity, higher the loan amount)

Concerning TARGET - Loan defaulters (Blue) are younger

```
In [103]: ► heatmap_data=credit_data_2[['AMT_INCOME_TOTAL', 'AMT_GOODS_PRICE', 'AMT_ANNUITY', 'DAYS_BIRTH',  
   'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'AMT_CREDIT']].corr()
```

```
In [107]: sns.heatmap(heatmap_data,cmap='YlGnBu',annot=True)
plt.show()
```



The Heatmap confirms our correlation findings from the pariplot

Final Insights

Following are the driving factors for a loan default -

- Lower the highest education of an applicant, higher the chance of loan default. This is one of the core driving factor in loan defaults.
- Labourers & Sales staff are major area of concern , with maximum applicants and a significant loan default rate. Drivers also have an alarming combination of counts and default %.
- Applicants on Maternity leave have a whopping 40% loan default rate. Unemployed applicants also have 35% loan defaults
- Low Income range have maximum % of loan defaults. As the Income range increases, loan default probability decreases
- Among different family status, married ones have the highest likelihood of loan default
- Applicants with lower Annuity Amount are slightly more likely to default on a loan.
- Young applicants are more expected to default on a loan.
- More Men default loans as compared to Women

Finally, as a conclusion, we recommend being careful while approving loans for applicants with lower education, as that constitutes the majority of the defaults. Alternatively, some policies can be devised to control such loan defaults. All other factors – Labourers class, Low-income range etc. seem to be logically linked with lower education. Hence, fixing that alone should have a chain effect, significantly improving the loan repayments.