Inspecting the datasets

```
import numpy as np, pandas as pd, matplotlib.pyplot as plt, seaborn as
sns
import warnings
warnings.filterwarnings("ignore")

#Importing the data table
#previous dataset contains all the information about whether the
client has any payment difficulties
#data dataset contains all the information whether the previous
application has been  Approved, Cancelled, Refused or Unused.

previous=pd.read_csv("previous_application.csv")
data=pd.read_csv("application_data.csv")

#Checking the datasets
previous.head(10)
```

```
   SK_ID_PREV  SK_ID_CURR NAME_CONTRACT_TYPE  AMT_ANNUITY
AMT_APPLICATION  \
0     2030495      271877     Consumer loans     1730.430
17145.0
1     2802425      108129         Cash loans    25188.615
607500.0
2     2523466      122040         Cash loans    15060.735
112500.0
3     2819243      176158         Cash loans    47041.335
450000.0
4     1784265      202054         Cash loans    31924.395
337500.0
5     1383531      199383         Cash loans    23703.930
315000.0
6     2315218      175704         Cash loans          NaN
0.0
7     1656711      296299         Cash loans          NaN
0.0
8     2367563      342292         Cash loans          NaN
0.0
9     2579447      334349         Cash loans          NaN
0.0

   AMT_CREDIT  AMT_DOWN_PAYMENT  AMT_GOODS_PRICE
WEEKDAY_APPR_PROCESS_START  \
0     17145.0               0.0          17145.0
SATURDAY
1    679671.0               NaN         607500.0
THURSDAY
2    136444.5               NaN         112500.0
```

```
TUESDAY
3    470790.0              NaN         450000.0
MONDAY
4    404055.0              NaN         337500.0
THURSDAY
5    340573.5              NaN         315000.0
SATURDAY
6         0.0              NaN              NaN
TUESDAY
7         0.0              NaN              NaN
MONDAY
8         0.0              NaN              NaN
MONDAY
9         0.0              NaN              NaN
SATURDAY

   HOUR_APPR_PROCESS_START  ... NAME_SELLER_INDUSTRY  CNT_PAYMENT  \
0                       15  ...         Connectivity         12.0
1                       11  ...                  XNA         36.0
2                       11  ...                  XNA         12.0
3                        7  ...                  XNA         12.0
4                        9  ...                  XNA         24.0
5                        8  ...                  XNA         18.0
6                       11  ...                  XNA          NaN
7                        7  ...                  XNA          NaN
8                       15  ...                  XNA          NaN
9                       15  ...                  XNA          NaN

   NAME_YIELD_GROUP       PRODUCT_COMBINATION  DAYS_FIRST_DRAWING  \
0            middle  POS mobile with interest            365243.0
1        low_action          Cash X-Sell: low            365243.0
2              high         Cash X-Sell: high            365243.0
3            middle       Cash X-Sell: middle            365243.0
4              high         Cash Street: high                 NaN
5        low_normal          Cash X-Sell: low            365243.0
6               XNA                      Cash                 NaN
7               XNA                      Cash                 NaN
8               XNA                      Cash                 NaN
9               XNA                      Cash                 NaN

  DAYS_FIRST_DUE DAYS_LAST_DUE_1ST_VERSION  DAYS_LAST_DUE
DAYS_TERMINATION  \
0          -42.0                     300.0          -42.0            -
37.0
1         -134.0                     916.0       365243.0
365243.0
2         -271.0                      59.0       365243.0
365243.0
3         -482.0                    -152.0         -182.0           -
177.0
```

```
4            NaN                        NaN            NaN
NaN
5         -654.0                     -144.0         -144.0             -
137.0
6            NaN                        NaN            NaN
NaN
7            NaN                        NaN            NaN
NaN
8            NaN                        NaN            NaN
NaN
9            NaN                        NaN            NaN
NaN

  NFLAG_INSURED_ON_APPROVAL
0                        0.0
1                        1.0
2                        1.0
3                        1.0
4                        NaN
5                        1.0
6                        NaN
7                        NaN
8                        NaN
9                        NaN

[10 rows x 37 columns]
```

```
#Checking the datasets
data.head(10)
```

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

  FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT
AMT_ANNUITY  \
0               Y             0          202500.0    406597.5
24700.5
1               N             0          270000.0   1293502.5
35698.5
2               Y             0           67500.0    135000.0
6750.0
```

```
3                Y              0           135000.0    312682.5
29686.5
4                Y              0           121500.0    513000.0
21865.5
5                Y              0            99000.0    490495.5
27517.5
6                Y              1           171000.0   1560726.0
41301.0
7                Y              0           360000.0   1530000.0
42075.0
8                Y              0           112500.0   1019610.0
33826.5
9                Y              0           135000.0    405000.0
20250.0

   ...  FLAG_DOCUMENT_18 FLAG_DOCUMENT_19 FLAG_DOCUMENT_20
FLAG_DOCUMENT_21  \
0  ...                 0                0                0
0
1  ...                 0                0                0
0
2  ...                 0                0                0
0
3  ...                 0                0                0
0
4  ...                 0                0                0
0
5  ...                 0                0                0
0
6  ...                 0                0                0
0
7  ...                 0                0                0
0
8  ...                 0                0                0
0
9  ...                 0                0                0
0

  AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_DAY  \
0                        0.0                       0.0
1                        0.0                       0.0
2                        0.0                       0.0
3                        NaN                       NaN
4                        0.0                       0.0
5                        0.0                       0.0
6                        0.0                       0.0
7                        0.0                       0.0
8                        0.0                       0.0
9                        NaN                       NaN
```

```
   AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
0                         0.0                        0.0
1                         0.0                        0.0
2                         0.0                        0.0
3                         NaN                        NaN
4                         0.0                        0.0
5                         0.0                        0.0
6                         0.0                        1.0
7                         0.0                        0.0
8                         0.0                        0.0
9                         NaN                        NaN

   AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
0                        0.0                         1.0
1                        0.0                         0.0
2                        0.0                         0.0
3                        NaN                         NaN
4                        0.0                         0.0
5                        1.0                         1.0
6                        1.0                         2.0
7                        0.0                         0.0
8                        0.0                         1.0
9                        NaN                         NaN

[10 rows x 122 columns]
```

```
#Fetching information about the nature of dataset
previous.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   SK_ID_PREV                   1670214 non-null  int64
 1   SK_ID_CURR                   1670214 non-null  int64
 2   NAME_CONTRACT_TYPE           1670214 non-null  object
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_DOWN_PAYMENT             774370 non-null   float64
 7   AMT_GOODS_PRICE              1284699 non-null  float64
 8   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object
 9   HOUR_APPR_PROCESS_START      1670214 non-null  int64
 10  FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object
```

```
 16  NAME_CONTRACT_STATUS         1670214 non-null  object 
 17  DAYS_DECISION                1670214 non-null  int64  
 18  NAME_PAYMENT_TYPE            1670214 non-null  object 
 19  CODE_REJECT_REASON           1670214 non-null  object 
 20  NAME_TYPE_SUITE              849809 non-null   object 
 21  NAME_CLIENT_TYPE             1670214 non-null  object 
 22  NAME_GOODS_CATEGORY          1670214 non-null  object 
 23  NAME_PORTFOLIO               1670214 non-null  object 
 24  NAME_PRODUCT_TYPE            1670214 non-null  object 
 25  CHANNEL_TYPE                 1670214 non-null  object 
 26  SELLERPLACE_AREA             1670214 non-null  int64  
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object 
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object 
 30  PRODUCT_COMBINATION          1669868 non-null  object 
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
```

```
data.describe()
```

```
          SK_ID_CURR         TARGET   CNT_CHILDREN  
AMT_INCOME_TOTAL  \
count  307511.000000  307511.000000  307511.000000      3.075110e+05   

mean   278180.518577       0.080729       0.417052      1.687979e+05   

std    102790.175348       0.272419       0.722121      2.371231e+05   

min    100002.000000       0.000000       0.000000      2.565000e+04   

25%    189145.500000       0.000000       0.000000      1.125000e+05   

50%    278202.000000       0.000000       0.000000      1.471500e+05   

75%    367142.500000       0.000000       1.000000      2.025000e+05   

max    456255.000000       1.000000      19.000000      1.170000e+08   


         AMT_CREDIT    AMT_ANNUITY  AMT_GOODS_PRICE  \
count  3.075110e+05  307499.000000     3.072330e+05   
mean   5.990260e+05   27108.573909     5.383962e+05   
std    4.024908e+05   14493.737315     3.694465e+05   
min    4.500000e+04    1615.500000     4.050000e+04   
```

```
25%    2.700000e+05   16524.000000     2.385000e+05
50%    5.135310e+05   24903.000000     4.500000e+05
75%    8.086500e+05   34596.000000     6.795000e+05
max    4.050000e+06  258025.500000     4.050000e+06

       REGION_POPULATION_RELATIVE     DAYS_BIRTH
DAYS_EMPLOYED  ...  \
count               307511.000000  307511.000000  307511.000000  ...

mean                     0.020868  -16036.995067   63815.045904  ...

std                      0.013831    4363.988632  141275.766519  ...

min                      0.000290  -25229.000000  -17912.000000  ...

25%                      0.010006  -19682.000000   -2760.000000  ...

50%                      0.018850  -15750.000000   -1213.000000  ...

75%                      0.028663  -12413.000000    -289.000000  ...

max                      0.072508   -7489.000000  365243.000000  ...


       FLAG_DOCUMENT_18  FLAG_DOCUMENT_19  FLAG_DOCUMENT_20
FLAG_DOCUMENT_21  \
count     307511.000000     307511.000000     307511.000000
307511.000000
mean           0.008130          0.000595          0.000507
0.000335
std            0.089798          0.024387          0.022518
0.018299
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%            0.000000          0.000000          0.000000
0.000000
max            1.000000          1.000000          1.000000
1.000000

       AMT_REQ_CREDIT_BUREAU_HOUR  AMT_REQ_CREDIT_BUREAU_DAY  \
count               265992.000000              265992.000000
mean                     0.006402                   0.007000
std                      0.083849                   0.110757
min                      0.000000                   0.000000
25%                      0.000000                   0.000000
```

```
50%                         0.000000                      0.000000
75%                         0.000000                      0.000000
max                         4.000000                      9.000000

       AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
count               265992.000000              265992.000000
mean                     0.034362                   0.267395
std                      0.204685                   0.916002
min                      0.000000                   0.000000
25%                      0.000000                   0.000000
50%                      0.000000                   0.000000
75%                      0.000000                   0.000000
max                      8.000000                  27.000000

       AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
count              265992.000000               265992.000000
mean                    0.265474                    1.899974
std                     0.794056                    1.869295
min                     0.000000                    0.000000
25%                     0.000000                    0.000000
50%                     0.000000                    1.000000
75%                     0.000000                    3.000000
max                   261.000000                   25.000000

[8 rows x 106 columns]
```

```
#Finding the null vaulues in data

null=data.isnull().sum()/len(data)*100
null.sort_values(ascending=False).head(60)
```

```
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
```

Cleaning the dataset

From the above observation, we can see that there is a sharp increase in missing data values from 19% to 31% to 47%. Hence, it seems reasonable to inspect the data which has data missing more than 31%

```
#Finding columns whose percentage of null values in columns exceed 32%
null_count= data.isnull().sum()
l=len(data)
null_count= null_count[null_count.values>(0.32*l)]
null_count
```

```
OWN_CAR_AGE                      202929
EXT_SOURCE_1                     173378
APARTMENTS_AVG                   156061
BASEMENTAREA_AVG                 179943
YEARS_BEGINEXPLUATATION_AVG      150007
YEARS_BUILD_AVG                  204488
COMMONAREA_AVG                   214865
ELEVATORS_AVG                    163891
ENTRANCES_AVG                    154828
FLOORSMAX_AVG                    153020
FLOORSMIN_AVG                    208642
LANDAREA_AVG                     182590
LIVINGAPARTMENTS_AVG             210199
LIVINGAREA_AVG                   154350
NONLIVINGAPARTMENTS_AVG          213514
NONLIVINGAREA_AVG                169682
APARTMENTS_MODE                  156061
BASEMENTAREA_MODE                179943
YEARS_BEGINEXPLUATATION_MODE     150007
YEARS_BUILD_MODE                 204488
COMMONAREA_MODE                  214865
ELEVATORS_MODE                   163891
ENTRANCES_MODE                   154828
FLOORSMAX_MODE                   153020
FLOORSMIN_MODE                   208642
LANDAREA_MODE                    182590
LIVINGAPARTMENTS_MODE            210199
LIVINGAREA_MODE                  154350
NONLIVINGAPARTMENTS_MODE         213514
NONLIVINGAREA_MODE               169682
APARTMENTS_MEDI                  156061
BASEMENTAREA_MEDI                179943
YEARS_BEGINEXPLUATATION_MEDI     150007
YEARS_BUILD_MEDI                 204488
COMMONAREA_MEDI                  214865
ELEVATORS_MEDI                   163891
ENTRANCES_MEDI                   154828
FLOORSMAX_MEDI                   153020
FLOORSMIN_MEDI                   208642
```

```
LANDAREA_MEDI                    182590
LIVINGAPARTMENTS_MEDI            210199
LIVINGAREA_MEDI                  154350
NONLIVINGAPARTMENTS_MEDI         213514
NONLIVINGAREA_MEDI               169682
FONDKAPREMONT_MODE               210295
HOUSETYPE_MODE                   154297
TOTALAREA_MODE                   148431
WALLSMATERIAL_MODE               156341
EMERGENCYSTATE_MODE              145755
dtype: int64
```

```
#Finding number of null columns with the given condition

len(null_count)
```

```
49
```

```
#Removing all these columns having null values > 32%; cdata stands for
cleaned dataset.

cdata= data.drop(data.columns[data.apply(lambda col:
col.isnull().sum() > (0.32*l))], axis=1)
cdata.shape
```

```
(307511, 73)
```

We can see that the number of columns have been reduced to 73. considering 112 columns, it is apparent that the 49 columns have been dropped.

```
#Checking % of null values after dropping the columns

null_new = cdata.isnull().sum()/len(cdata)*100
null_new.sort_values(ascending = False).head(60)
```

```
OCCUPATION_TYPE                31.345545
EXT_SOURCE_3                   19.825307
AMT_REQ_CREDIT_BUREAU_YEAR     13.501631
AMT_REQ_CREDIT_BUREAU_QRT      13.501631
AMT_REQ_CREDIT_BUREAU_MON      13.501631
AMT_REQ_CREDIT_BUREAU_WEEK     13.501631
AMT_REQ_CREDIT_BUREAU_DAY      13.501631
AMT_REQ_CREDIT_BUREAU_HOUR     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
FLAG_DOCUMENT_17               0.000000
FLAG_DOCUMENT_18               0.000000
FLAG_DOCUMENT_21               0.000000
FLAG_DOCUMENT_20               0.000000
FLAG_DOCUMENT_19               0.000000
FLAG_DOCUMENT_2                0.000000
FLAG_DOCUMENT_3                0.000000
FLAG_DOCUMENT_4                0.000000
FLAG_DOCUMENT_5                0.000000
FLAG_DOCUMENT_16               0.000000
FLAG_DOCUMENT_6                0.000000
FLAG_DOCUMENT_7                0.000000
FLAG_DOCUMENT_8                0.000000
FLAG_DOCUMENT_9                0.000000
FLAG_DOCUMENT_10               0.000000
FLAG_DOCUMENT_11               0.000000
ORGANIZATION_TYPE              0.000000
FLAG_DOCUMENT_13               0.000000
FLAG_DOCUMENT_14               0.000000
FLAG_DOCUMENT_15               0.000000
FLAG_DOCUMENT_12               0.000000
SK_ID_CURR                     0.000000
LIVE_CITY_NOT_WORK_CITY        0.000000
DAYS_REGISTRATION              0.000000
NAME_CONTRACT_TYPE             0.000000
CODE_GENDER                    0.000000
FLAG_OWN_CAR                   0.000000
FLAG_OWN_REALTY                0.000000
CNT_CHILDREN                   0.000000
AMT_INCOME_TOTAL               0.000000
AMT_CREDIT                     0.000000
NAME_INCOME_TYPE               0.000000
NAME_EDUCATION_TYPE            0.000000
NAME_FAMILY_STATUS             0.000000
NAME_HOUSING_TYPE              0.000000
REGION_POPULATION_RELATIVE     0.000000
DAYS_BIRTH                     0.000000
DAYS_EMPLOYED                  0.000000
DAYS_ID_PUBLISH                0.000000
REG_CITY_NOT_WORK_CITY         0.000000
FLAG_MOBIL                     0.000000
FLAG_EMP_PHONE                 0.000000
dtype: float64

#Now is the time for imputation of the values which are missing at
small scale.

#Let's start with OCCUPATION_TYPE, having the largest percentage of
```

```
missing data now.

data.OCCUPATION_TYPE.value_counts(normalize= True)*100

Laborers                 26.139636
Sales staff              15.205570
Core staff               13.058924
Managers                 10.122679
Drivers                   8.811576
High skill tech staff     5.390299
Accountants               4.648067
Medicine staff            4.043672
Security staff            3.183498
Cooking staff             2.816408
Cleaning staff            2.203960
Private service staff     1.256158
Low-skill Laborers        0.991379
Waiters/barmen staff      0.638499
Secretaries               0.618132
Realty agents             0.355722
HR staff                  0.266673
IT staff                  0.249147
Name: OCCUPATION_TYPE, dtype: float64

#Because the variable is categorical in nature, the type of
distribution makes it idle to impute the missing values with the mode
of the dataset

data.OCCUPATION_TYPE.mode()

#Hence, the OCCUPATION_TYPE IS IMPUTED

0    Laborers
dtype: object

#Then comes EXT_SOURCE_3

data.EXT_SOURCE_3.value_counts().head(50)

0.746300    1460
0.713631    1315
0.694093    1276
0.670652    1191
0.652897    1154
0.581484    1141
0.689479    1138
0.595456    1136
0.554947    1132
0.621226    1109
0.657784    1092
0.607557    1067
```

```
0.643026    1066
0.450747    1064
0.626304    1054
0.673830    1030
0.651260    1029
0.511892    1026
0.706205     992
0.553165     984
0.593718     978
0.634706     969
0.740799     961
0.681706     959
0.565608     956
0.728141     953
0.771362     947
0.576209     943
0.586740     942
0.656158     931
0.631355     929
0.484851     922
0.709189     919
0.665855     912
0.684828     911
0.538863     911
0.617826     907
0.591977     904
0.513694     895
0.683269     895
0.579727     895
0.000527     886
0.733815     880
0.619528     879
0.501075     879
0.508287     877
0.754406     874
0.712155     867
0.832785     865
0.558507     864
Name: EXT_SOURCE_3, dtype: int64
```

```
#It looks ideal to impute the missing values of this set by mean.

data.EXT_SOURCE_3.median()
```

```
0.5352762504724826
```

```
#Now comes the turn of AMT_REQ_CREDIT_BUREAU_***, 6 columns which have
same percentage of missing values.
#Inspecting on them,

data.AMT_REQ_CREDIT_BUREAU_YEAR.value_counts()
```

```
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
```

```
#Given the nature of dataset, we shall impute the missing data by mode
of this set which seems to be the most apt and obvious imputation.
Similar is the case with all AMT_REQ_CREDIT_BUREAU_*** columns

data.AMT_REQ_CREDIT_BUREAU_YEAR.mode()
data.AMT_REQ_CREDIT_BUREAU_QRT.mode()
data.AMT_REQ_CREDIT_BUREAU_MON.mode()
data.AMT_REQ_CREDIT_BUREAU_WEEK.mode()
data.AMT_REQ_CREDIT_BUREAU_DAY.mode()
data.AMT_REQ_CREDIT_BUREAU_HOUR.mode()
```

```
0    0.0
dtype: float64
```

```
#Inspecting the type of data distribution in NAME_TYPE_SUITS
data.NAME_TYPE_SUITE.value_counts()
```

```
Unaccompanied      248526
Family              40149
Spouse, partner     11370
Children             3267
Other_B              1770
Other_A               866
```

```
Group of people       271
Name: NAME_TYPE_SUITE, dtype: int64
```

```
#It is to be imputed with the mode here, i.e., 'Unaccompanied'
data.NAME_TYPE_SUITE.mode()
```

```
0    Unaccompanied
dtype: object
```

```
#Inspecting OBS_30_CNT_SOCIAL_CIRCLE
data.OBS_30_CNT_SOCIAL_CIRCLE.value_counts()
```

```
0.0      163910
1.0       48783
2.0       29808
3.0       20322
4.0       14143
5.0        9553
6.0        6453
7.0        4390
8.0        2967
9.0        2003
10.0       1376
11.0        852
12.0        652
13.0        411
14.0        258
15.0        166
16.0        133
17.0         88
18.0         46
19.0         44
20.0         30
21.0         29
22.0         22
23.0         15
25.0         11
24.0         11
27.0          5
26.0          3
30.0          2
28.0          1
29.0          1
47.0          1
348.0         1
Name: OBS_30_CNT_SOCIAL_CIRCLE, dtype: int64
```

```
#We inspect that 0 is the mode, and seeing the distribution, it is apt
to replace the data by mode.
data.OBS_30_CNT_SOCIAL_CIRCLE.mode()
#Similar goes for the rest of SOCIAL_CIRCLE, to impute all of it with
```

```
mode
data.DEF_30_CNT_SOCIAL_CIRCLE.mode()
data.OBS_60_CNT_SOCIAL_CIRCLE.mode()
data.DEF_60_CNT_SOCIAL_CIRCLE.mode()
```

```
0    0.0
dtype: float64
```

```
#Inspecting EXT_SOURCE_2
data.EXT_SOURCE_2.value_counts()
```

```
0.285898    721
0.262258    417
0.265256    343
0.159679    322
0.265312    306
           ...
0.004725      1
0.257313      1
0.282030      1
0.181540      1
0.267834      1
Name: EXT_SOURCE_2, Length: 119831, dtype: int64
```

```
#It seems ideal to impute the missing values with the median of this
set.
data.EXT_SOURCE_2.median()
```

```
0.5659614260608526
```

```
#Inspecting AMT_GOODS_PRICE
data.AMT_GOODS_PRICE.value_counts()
```

```
450000.0     26022
225000.0     25282
675000.0     24962
900000.0     15416
270000.0     11428
             ...
1265751.0        1
503266.5         1
810778.5         1
666090.0         1
743863.5         1
Name: AMT_GOODS_PRICE, Length: 1002, dtype: int64
```

```
#Seeing the spread, the median seems to be the ideal imputation value
for the missing values in the above column
data.AMT_GOODS_PRICE.median()
```

```
450000.0
```

```
#Next comes the AMT_ANNUITY column. Analysing it,
data.AMT_ANNUITY.value_counts()
```

```
9000.0      6385
13500.0     5514
6750.0      2279
10125.0     2035
37800.0     1602
            ...
79902.0        1
106969.5       1
60885.0        1
59661.0        1
77809.5        1
Name: AMT_ANNUITY, Length: 13672, dtype: int64
```

```
#Imputing it by median,
data.AMT_ANNUITY.median()
```

```
24903.0
```

```
#Inspecting CNT_FAM_MEMBERS,
data.CNT_FAM_MEMBERS.value_counts()
```

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

```
#Imputing the missing values with mode seems to be the most apt
choice.
data.CNT_FAM_MEMBERS.mode()
```

```
0    2.0
dtype: float64
```

```
#Finally, inspecting the column DAYS_LAST_PHONE_CHANGE
data.DAYS_LAST_PHONE_CHANGE.value_counts()
```

```
 0.0       37672
-1.0        2812
-2.0        2318
-3.0        1763
-4.0        1285
           ...
-4051.0        1
-3593.0        1
-3622.0        1
-3570.0        1
-3538.0        1
Name: DAYS_LAST_PHONE_CHANGE, Length: 3773, dtype: int64
```

```
#Once again, imputing the missing values with mode seems to be the
most apt choice,
data.DAYS_LAST_PHONE_CHANGE.mode()
```

```
0    0.0
dtype: float64
```

After filling in the missing values and doing all the basic cleaning, the next target is to get the dataset rid of any errors that might have crept in

```
#Let us check CODE_GENDER for any errors
data.CODE_GENDER.value_counts()
```

```
F      202448
M      105059
XNA         4
Name: CODE_GENDER, dtype: int64
```

```
#Here we see, XNA is an error as gender mostly is either male or
female. We need to replace it by the mode of the dataset, which is F.
data.loc[data.CODE_GENDER == 'XNA','CODE_GENDER'] = 'F'
```

```
#Checking the dataset ORGANIZATION_TYPE,
data.ORGANIZATION_TYPE.value_counts()
```

```
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
```

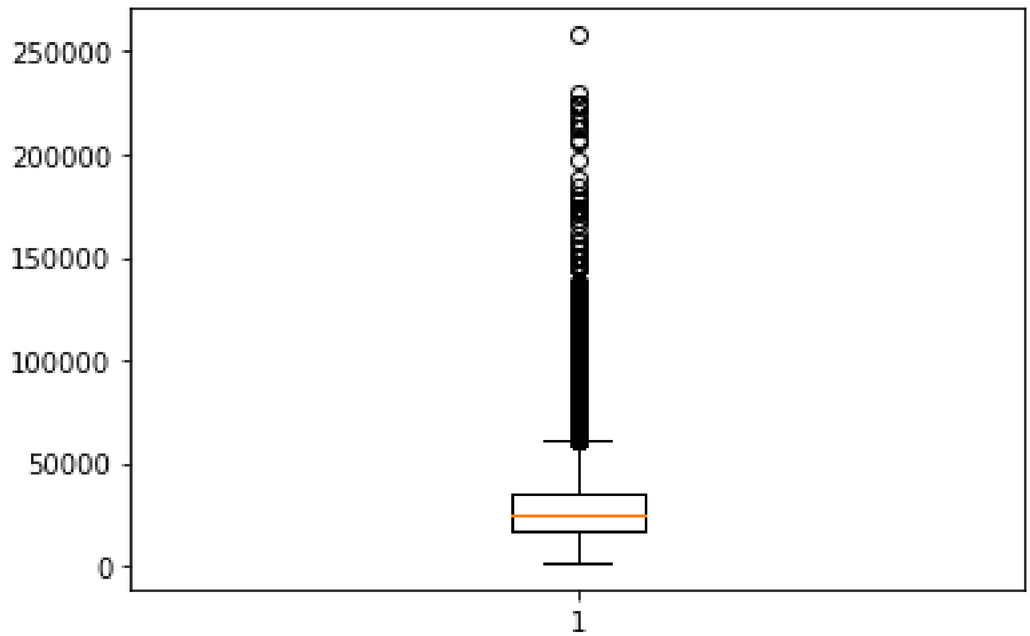
```
Industry: type 9          3368
Industry: type 3          3278
Security                  3247
Housing                   2958
Industry: type 11         2704
Military                  2634
Bank                      2507
Agriculture               2454
Police                    2341
Transport: type 2         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
```

```
#Replacing the XNA with np.nan,
data=data.replace('XNA',np.nan)
```

Analysing the dataset

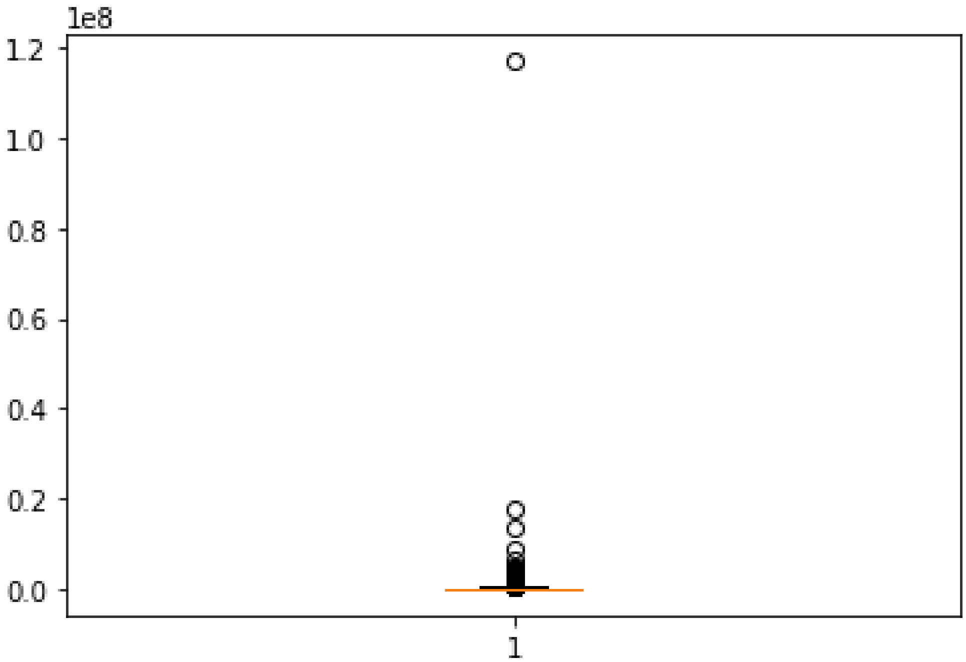
We shall do univariate analysis with the columns first which can be quantitively measured

```
#Let us start with the AMT_ANNUITY

amt_ann=data['AMT_ANNUITY']
amt_annf = amt_ann[~np.isnan(amt_ann)]
plt.boxplot(amt_annf)
plt.show()
```

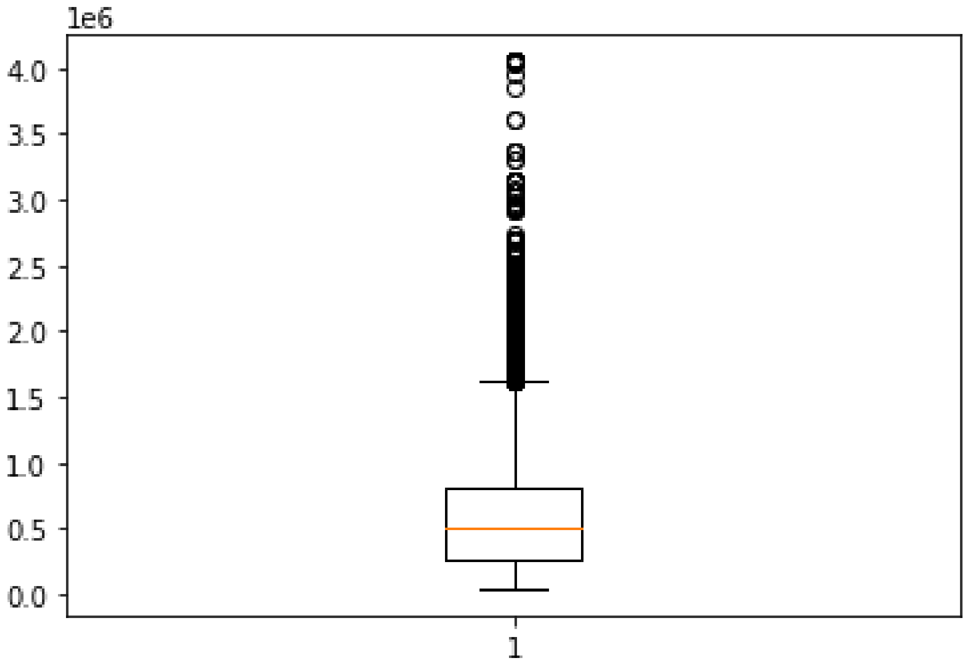



```
#Finding the outlier by inter-quartile range

q1 = data['AMT_ANNUITY'].quantile(0.25)
q3 = data['AMT_ANNUITY'].quantile(0.75)

iqr = q3-q1
outlier=(q3 + 1.5 * iqr)
print(outlier)

#Hence, any value of AMT_ANNUITY is an outlier if it is more than Rs.
61704.

61704.0

#Let us analyse the AMT_APPLICATION

amt_it=data['AMT_INCOME_TOTAL']
amt_itf = amt_it[~np.isnan(amt_it)]
plt.boxplot(amt_itf)
plt.show()
```

```
#There is one outlier in this dataset
```




```
#Let us analyse the AMT_CREDIT

amt_c=data['AMT_CREDIT']
amt_cf = amt_c[~np.isnan(amt_c)]
plt.boxplot(amt_cf)
plt.show()
```



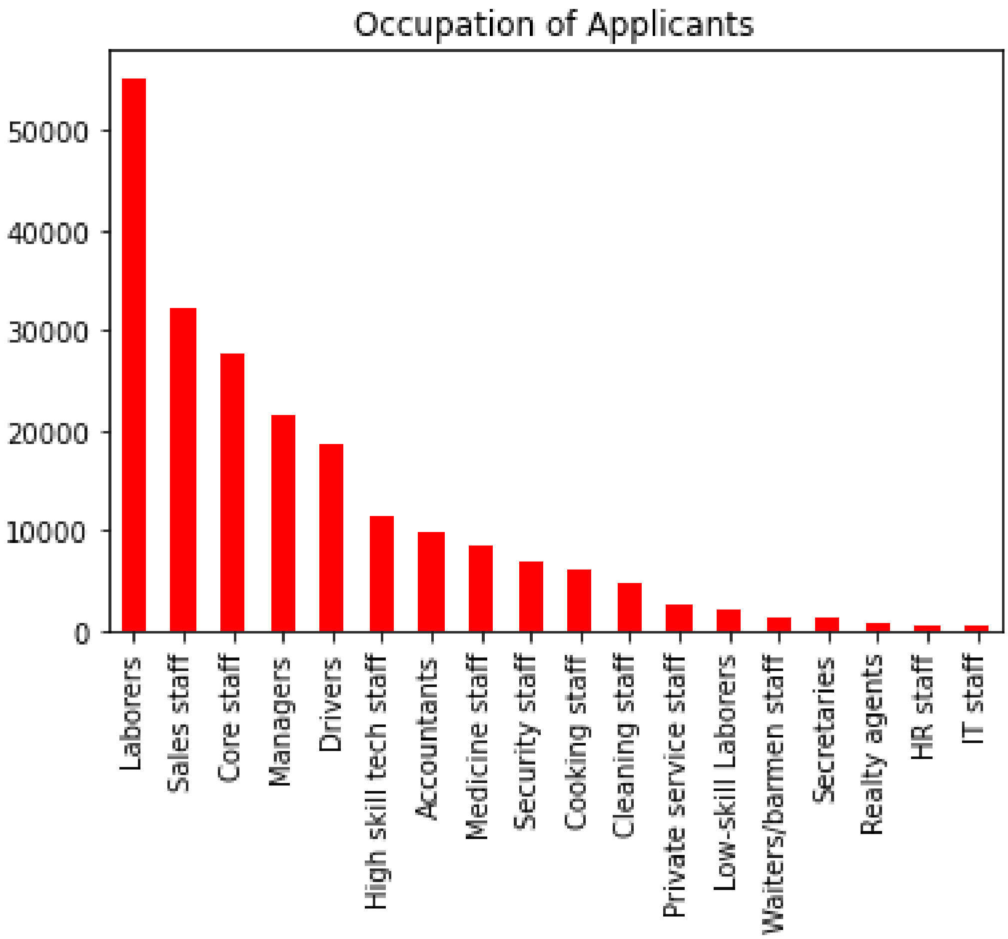

```
#Finding the outlier by inter-quartile range

qq1 = data['AMT_CREDIT'].quantile(0.25)
qq3 = data['AMT_CREDIT'].quantile(0.75)

iqqr = qq3-qq1
outlier=(qq3 + 1.5 * iqqr)
print(outlier)

#A credit amount more than Rs. 1616625 is an outlier for this column
```

```
1616625.0
```

Analysing distribution type of categorical variables

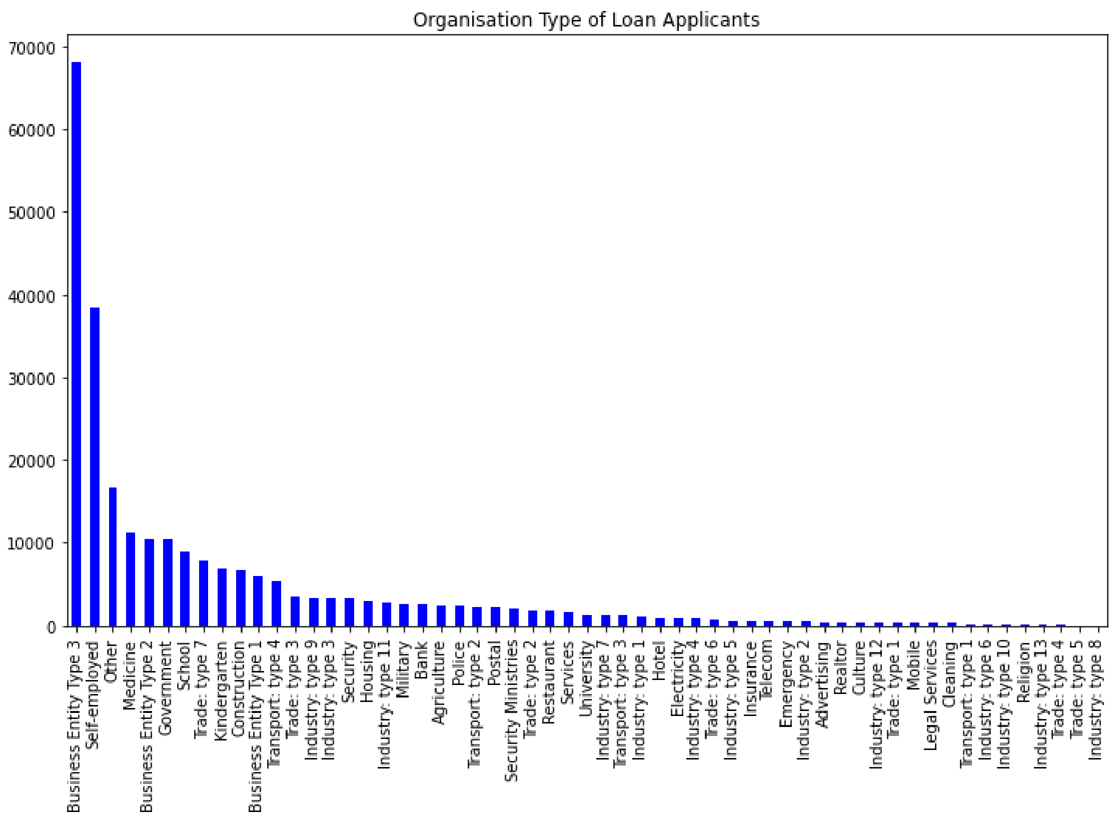
```
# Distribution of OCCUPATION_TYPE
occ = data['OCCUPATION_TYPE'].value_counts()
occ.plot.bar(x = 'Occupation', y = "Count", title = 'Occupation of 
Applicants', color = 'red')

#It can ba asessed that the people who apply for loan the most are 
occupied as laborers, sales staffs, core staffs and managers.
```

```
<AxesSubplot:title={'center':'Occupation of Applicants'}>
```




```
# Distribution of ORGANIZATION_TYPE
plt.rcParams['figure.figsize'] = [12, 7]
occ = data['ORGANIZATION_TYPE'].value_counts()
occ.plot.bar(x = 'Organisation', y = "Count", title = 'Organisation
Type of Loan Applicants', color = 'blue')

#It can ba asessed that the organisations who apply for loan the most
are from Business entity type 3, self employed and Other.
```

```
<AxesSubplot:title={'center':'Organisation Type of Loan Applicants'}>
```

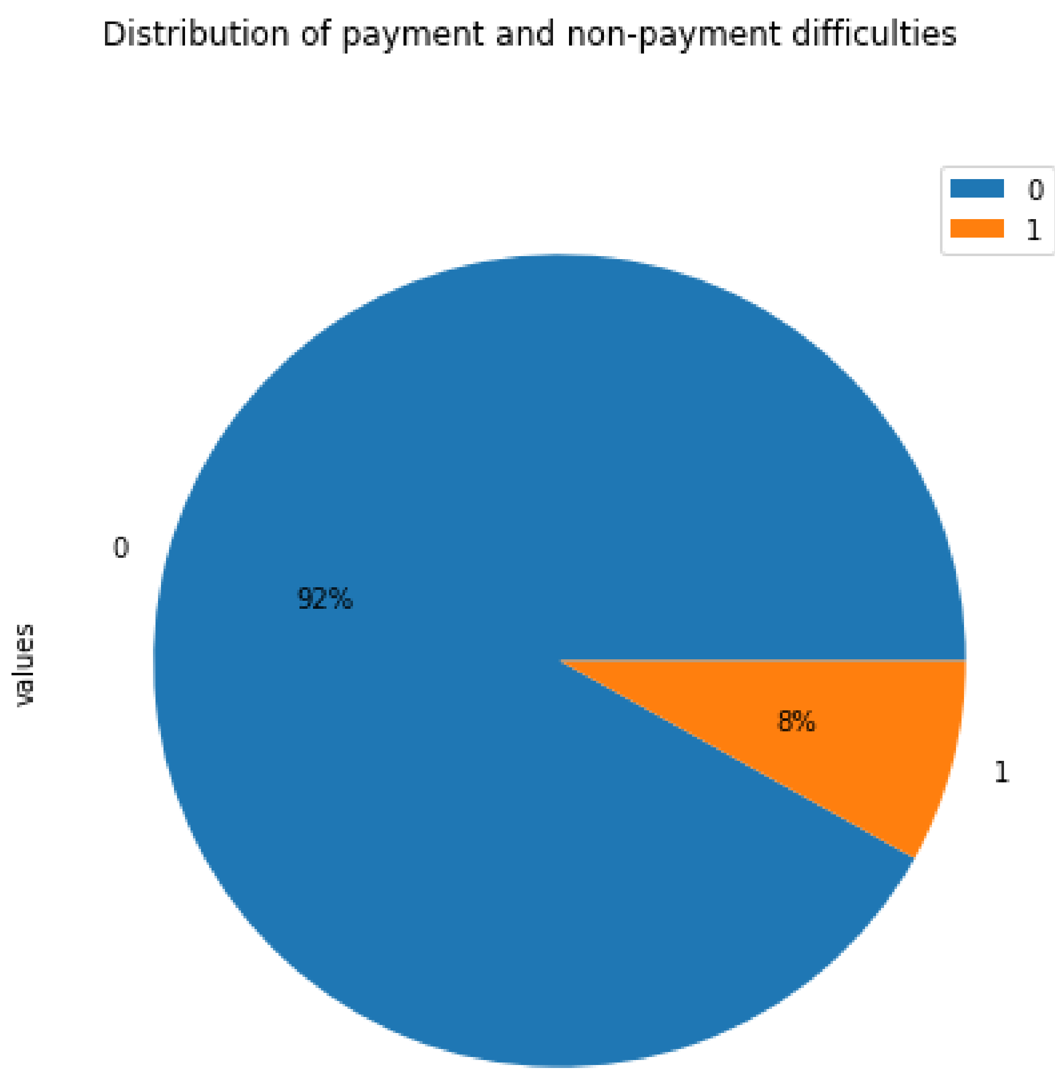


It can be seen that laborers and business entity type 3 people are the people who applies for most of the loan

Now for univariate analysis, we shall break down the data for difficulties in loan application using the 'target' variable into two parts- one having payment difficulties and the other category being of non-payment difficulties.

```
#Checking the percentage distribution of target variable

t = data["TARGET"].value_counts()
df = pd.DataFrame({'values': t.values})
df.plot.pie( title='Distribution of payment and non-payment difficulties', subplots=True, autopct='%1.0f%%')
```

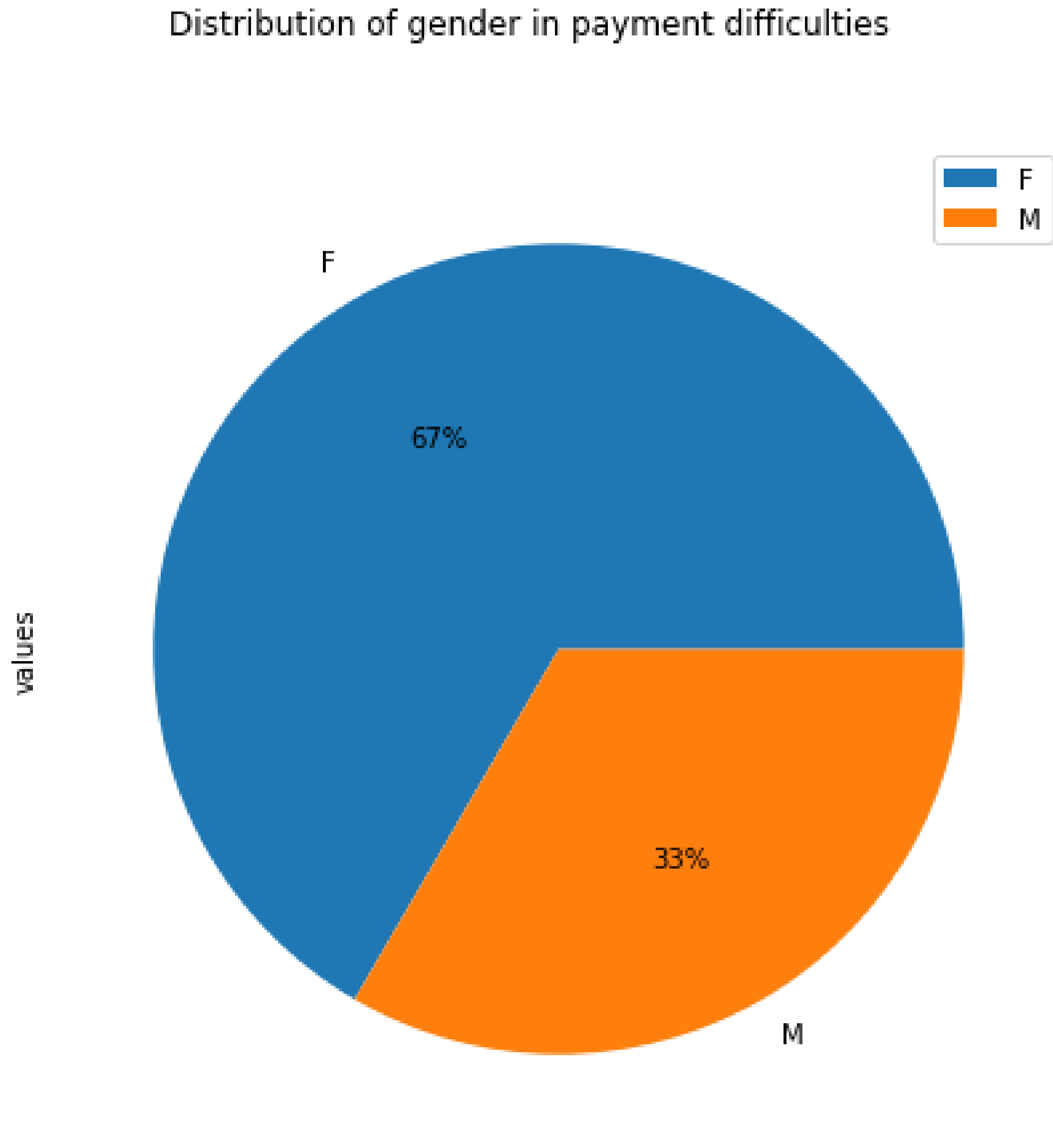
```
array([<AxesSubplot:ylabel='values'>], dtype=object)
```

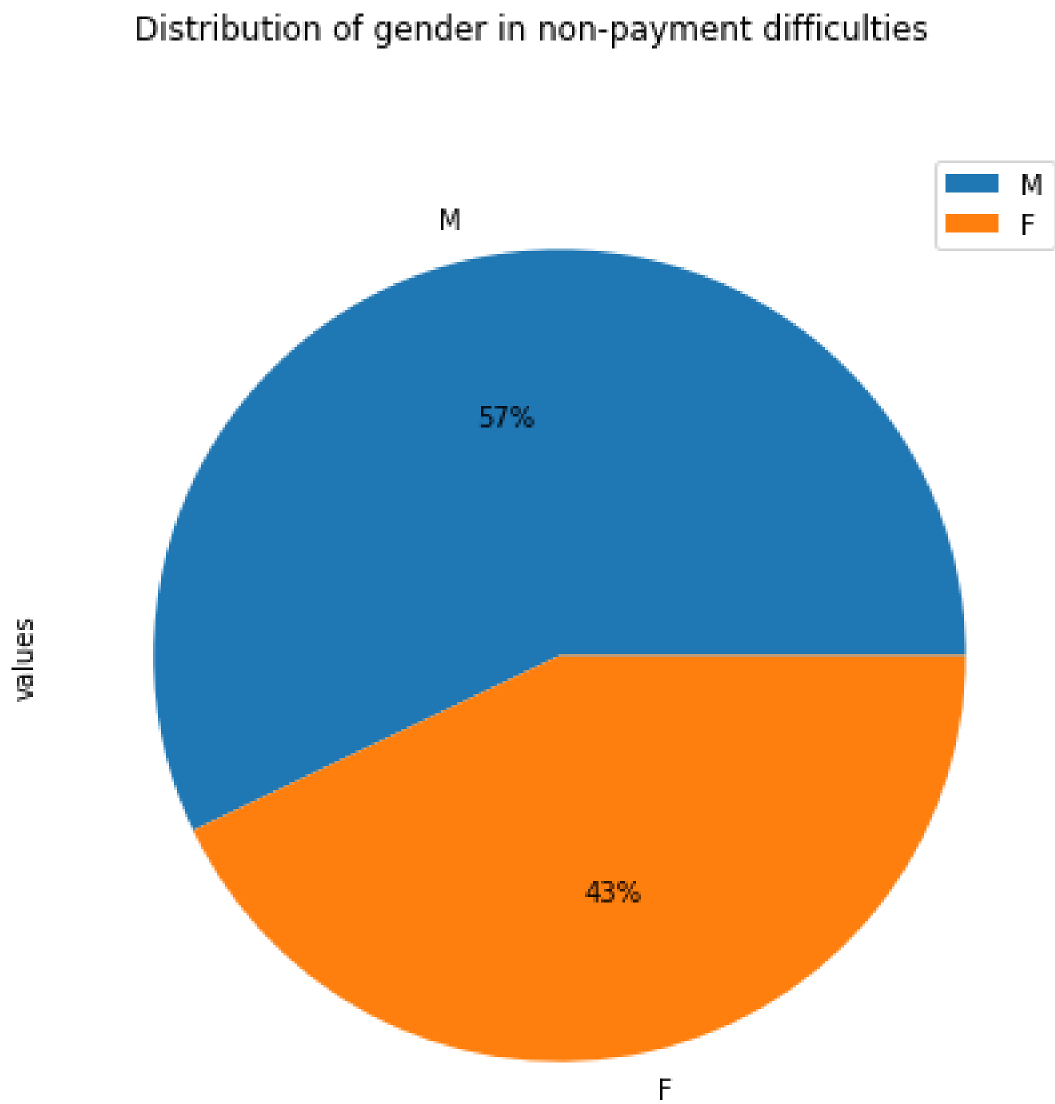



```
#Now we give different variable names for the two quantities

t0=data.loc[data.TARGET == 0]
t1=data.loc[data.TARGET == 1]

#t0 and t1 represents target variable with value 0 and 1 respectively;
0 represents payment difficulties and 1 represents non-payment
difficulties

#We find out gender distribution statistics for both payment and non-
payment related difficulties

#ag0 represents analysis of gender distribution for target variable 0.
```

```
ag0 = t0["CODE_GENDER"].value_counts()
df1 = pd.DataFrame({'values': ag0.values})
df1.plot.pie(labels=t0.CODE_GENDER,title='Distribution of gender in
payment difficulties', subplots=True, autopct='%1.0f%%')

#ag1 represents analysis of gender distribution for target variable 1.

ag1 = t1["CODE_GENDER"].value_counts()
df2 = pd.DataFrame({'values': ag1.values})
df2.plot.pie(labels=t1.CODE_GENDER,title='Distribution of gender in
non-payment difficulties', subplots=True, autopct='%1.0f%%')
```

```
array([<AxesSubplot:ylabel='values'>], dtype=object)
```

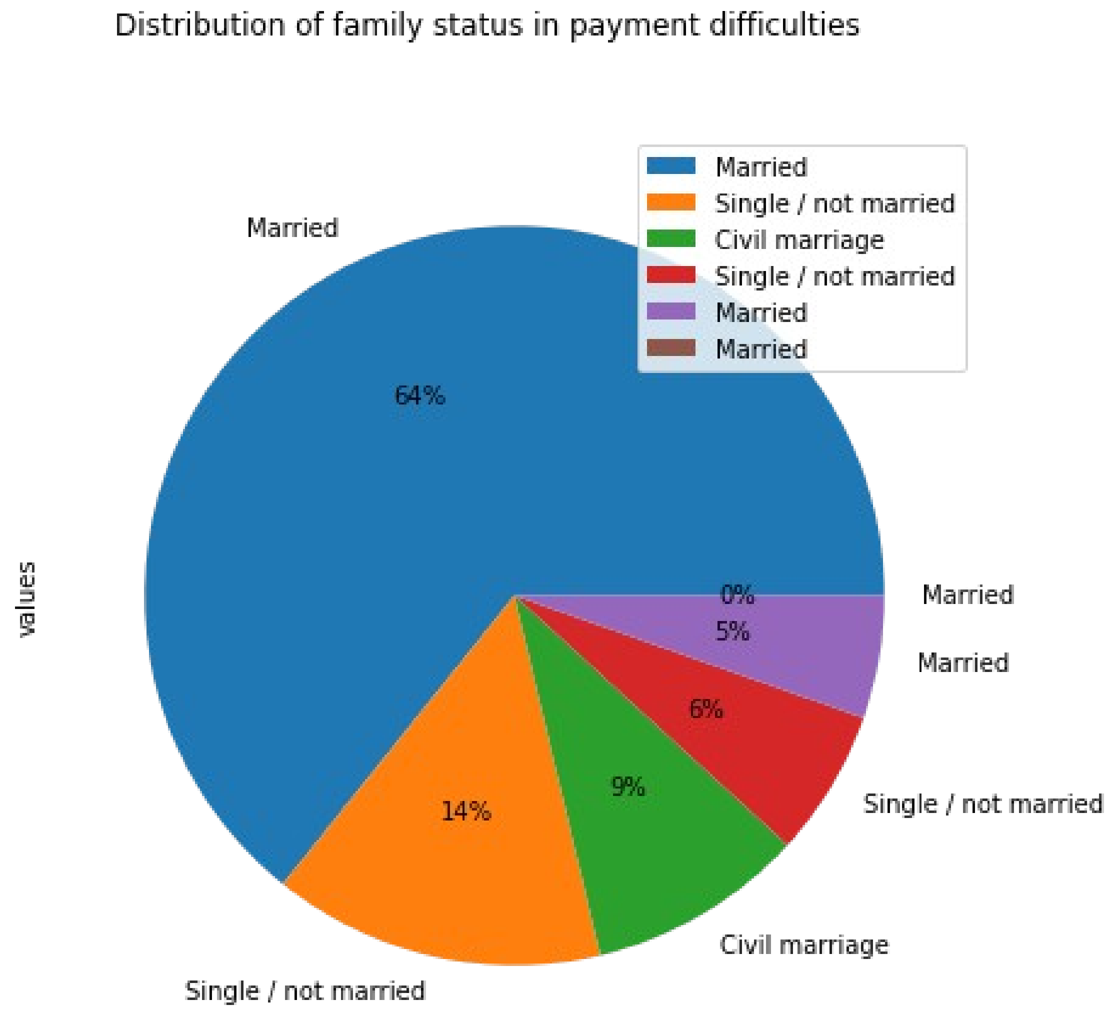


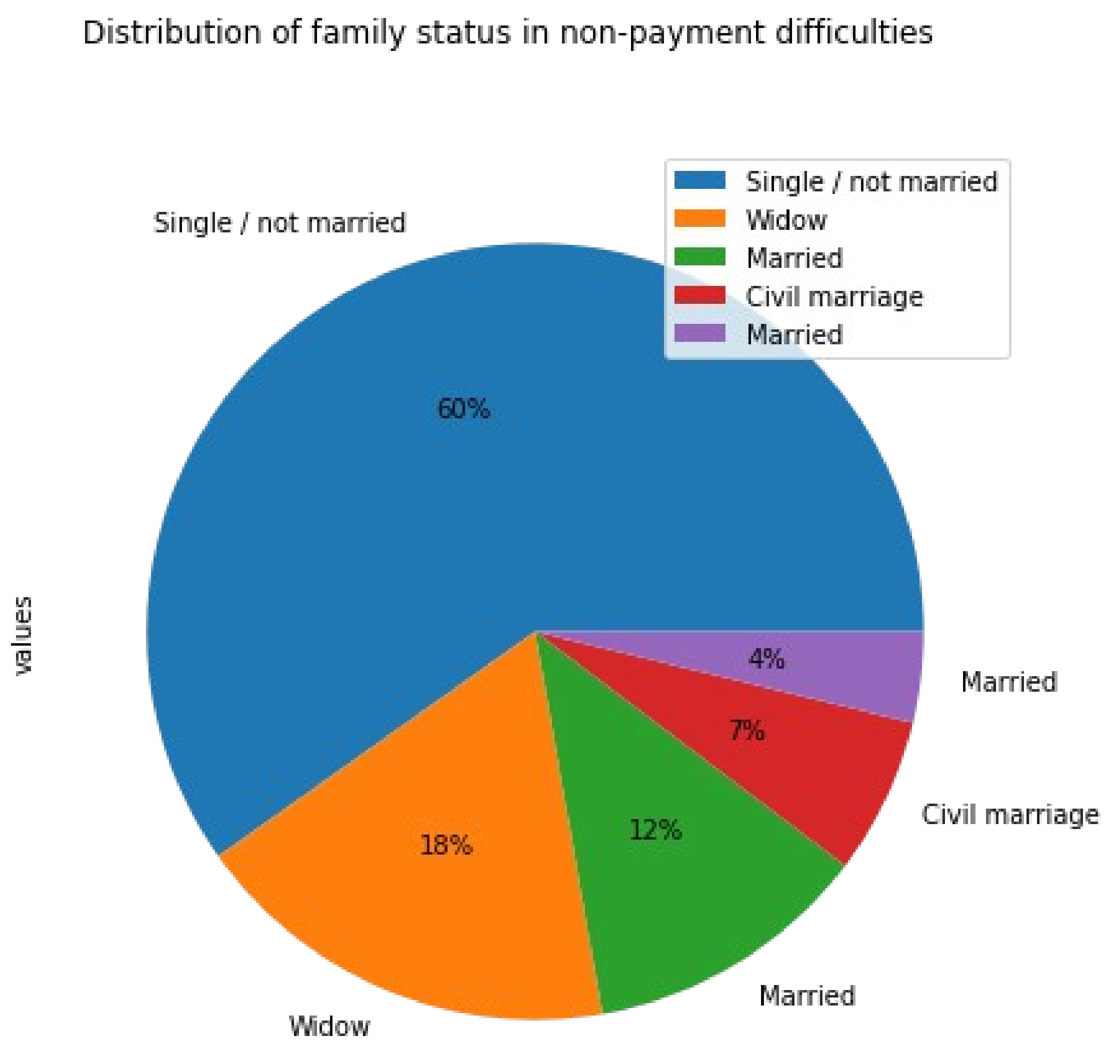



```
#We find the distribution of family status for payment and non-payment
difficulties

#af0 represents analysis of family status for target variable 0.

af0 = t0["NAME_FAMILY_STATUS"].value_counts()
df3 = pd.DataFrame({'values': af0.values})
df3.plot.pie(labels=t0.NAME_FAMILY_STATUS,title='Distribution of
family status in payment difficulties', subplots=True, autopct='%1.0f%
%')

#af1 represents analysis of family status for target variable 1.

af1 = t1["NAME_FAMILY_STATUS"].value_counts()
```

```
df4 = pd.DataFrame({'values': af1.values})
df4.plot.pie(labels=t1.NAME_FAMILY_STATUS,title='Distribution of
family status in non-payment difficulties', subplots=True,
autopct='%1.0f%%')
```

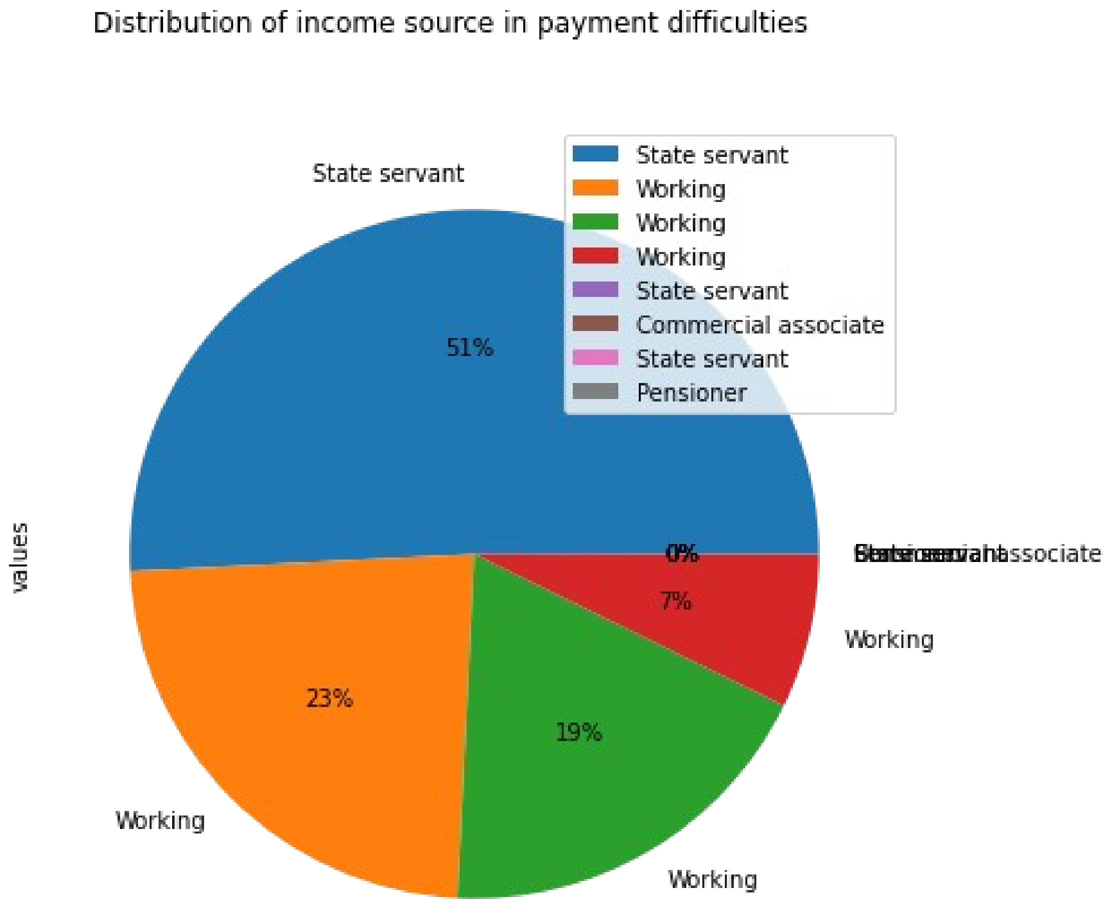
```
array([<AxesSubplot:ylabel='values'>], dtype=object)
```

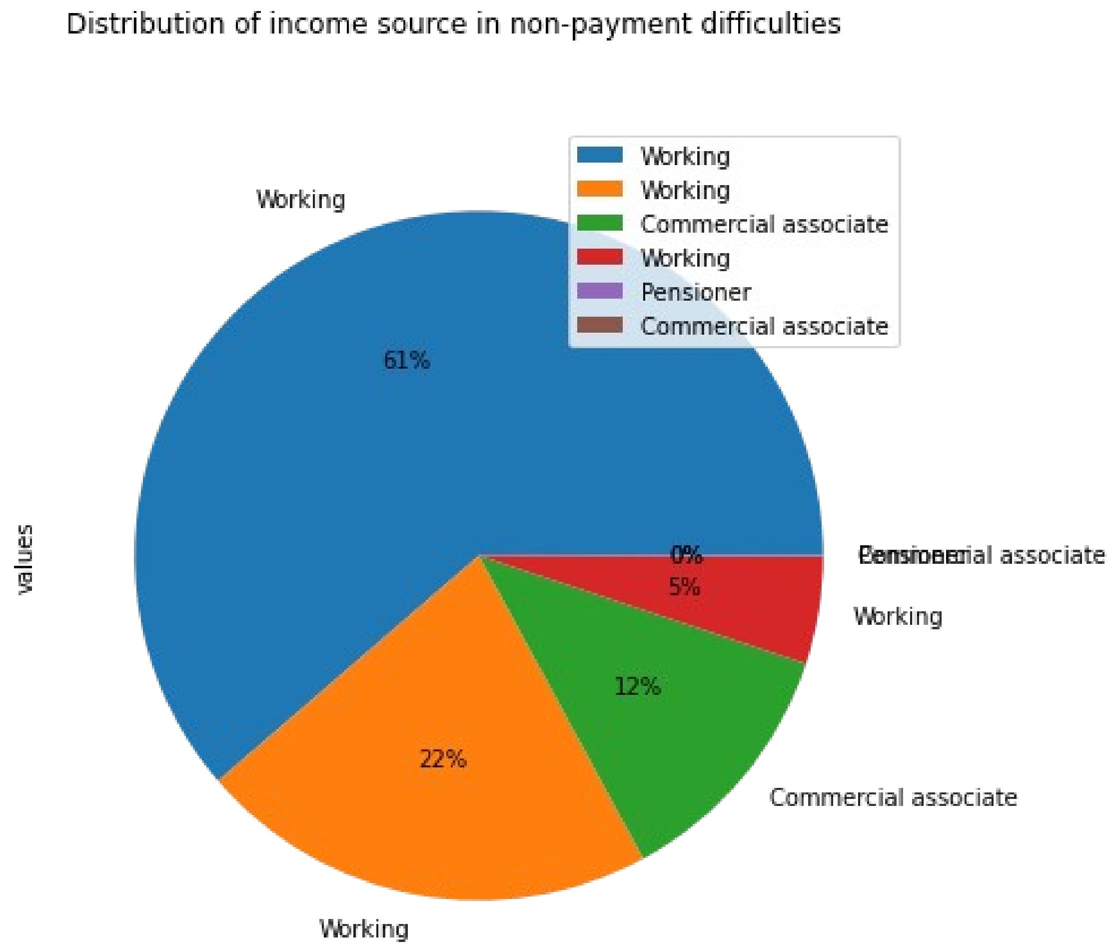





```
#We find the distribution of income source for payment and non-payment difficulties

#ai0 represents analysis of income source for target variable 0.

ai0 = t0["NAME_INCOME_TYPE"].value_counts()
df5 = pd.DataFrame({'values': ai0.values})
df5.plot.pie(labels=t0.NAME_INCOME_TYPE,title='Distribution of income source in payment difficulties', subplots=True, autopct='%1.0f%%')

#ai1 represents analysis of income source for target variable 1.

ai1 = t1["NAME_INCOME_TYPE"].value_counts()
df6 = pd.DataFrame({'values': ai1.values})
df6.plot.pie(labels=t1.NAME_INCOME_TYPE,title='Distribution of income source in non-payment difficulties', subplots=True, autopct='%1.0f%%')
```

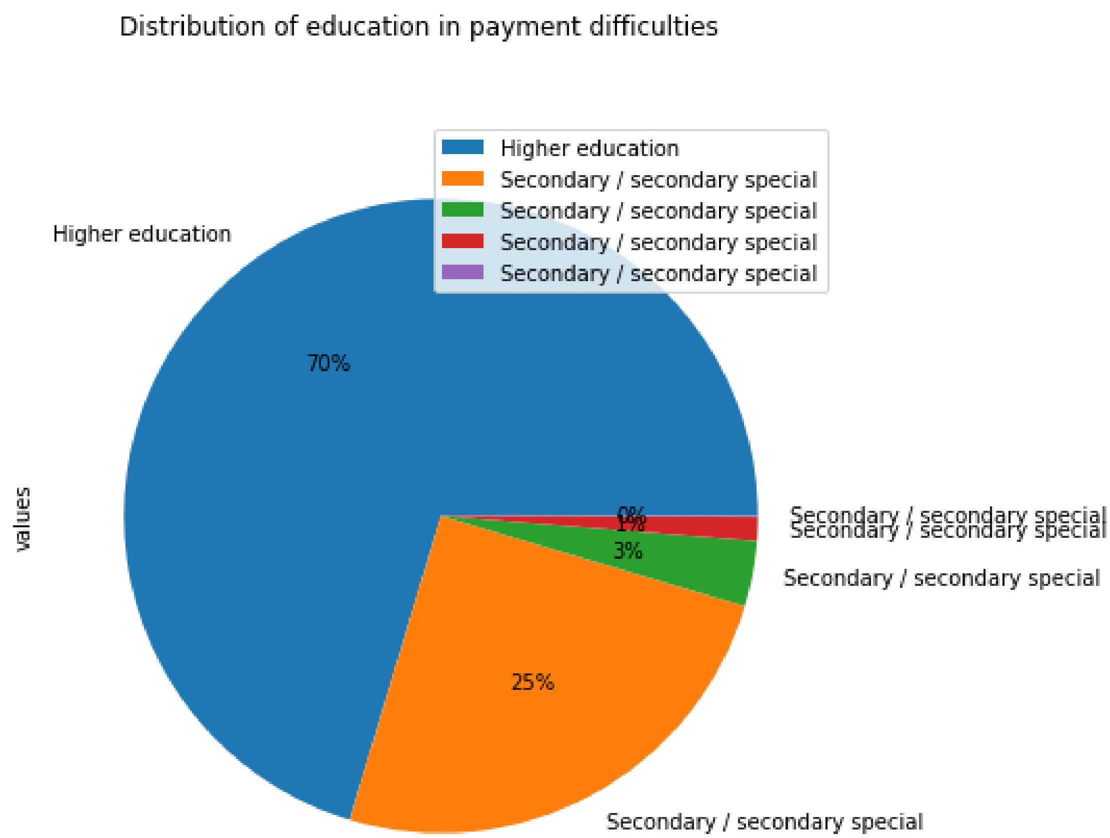
```
array([<AxesSubplot:ylabel='values'>], dtype=object)
```



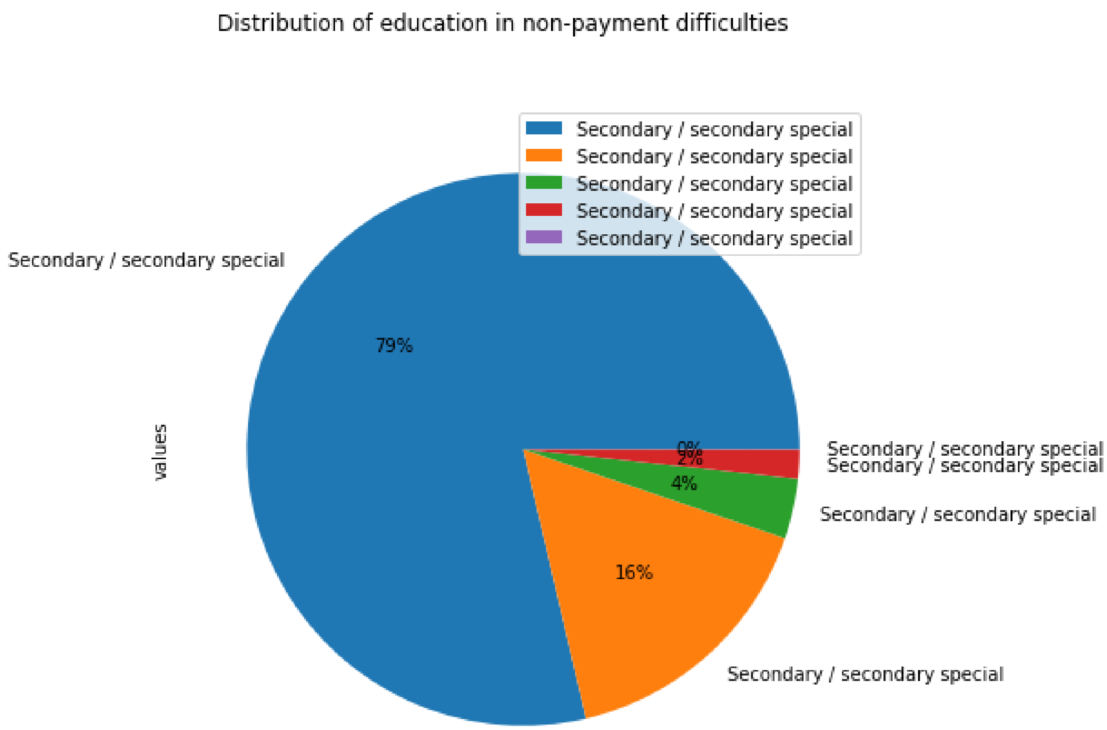



```
#Now we analyse the educational background of applicants having loan difficulties

#al0 represents analysis of educational background for target variable 0.

al0 = t0["NAME_EDUCATION_TYPE"].value_counts()
df7 = pd.DataFrame({'values': al0.values})
df7.plot.pie(labels=t0.NAME_EDUCATION_TYPE,title='Distribution of education in payment difficulties', subplots=True, autopct='%1.0f%%')

#ai1 represents analysis of income source for target variable 1.

al1 = t1["NAME_EDUCATION_TYPE"].value_counts()
df8 = pd.DataFrame({'values': al1.values})
df8.plot.pie(labels=t1.NAME_EDUCATION_TYPE,title='Distribution of education in non-payment difficulties', subplots=True, autopct='%1.0f%%')
```

```
array([<AxesSubplot:ylabel='values'>], dtype=object)
```

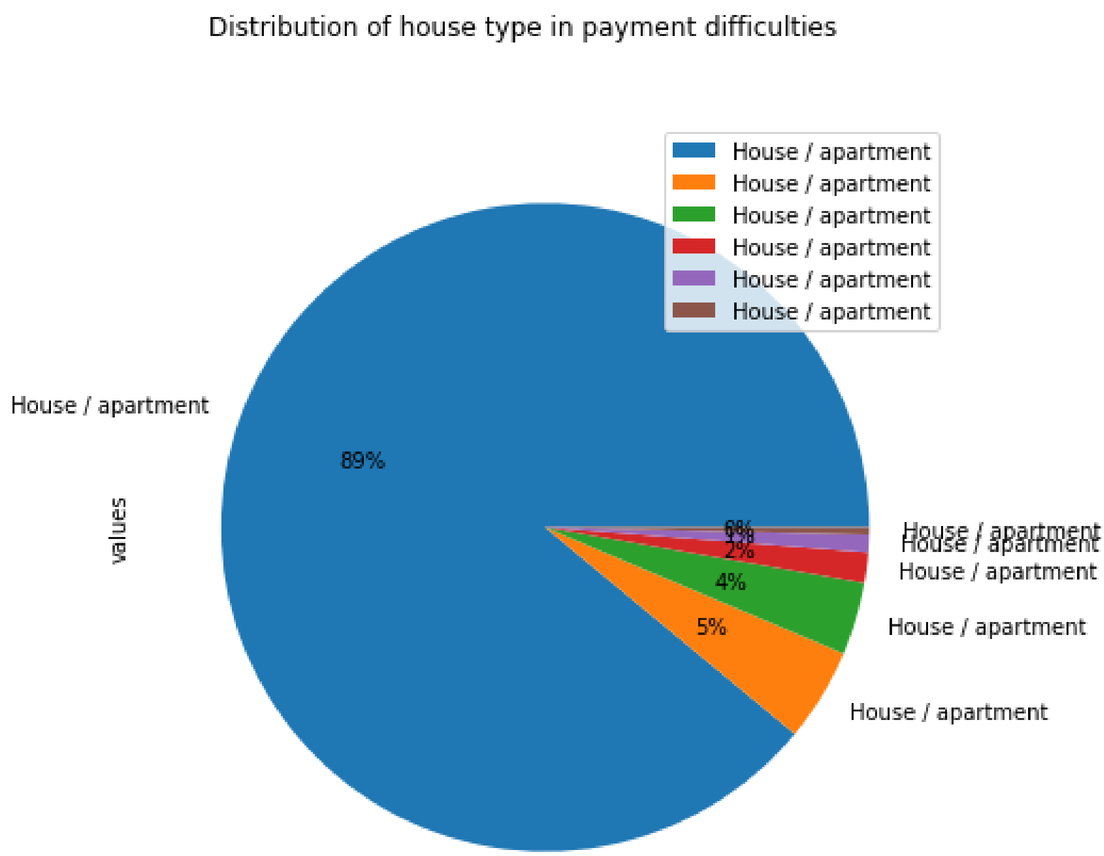


Distribution of education in payment difficulties
Higher education
Secondary / secondary special
Secondary / secondary special
Secondary / secondary special
Secondary / secondary special
Higher education
70%
values
0%
1%
3%
25%
Secondary / secondary special
Secondary / secondary special
Secondary / secondary special
Secondary / secondary special

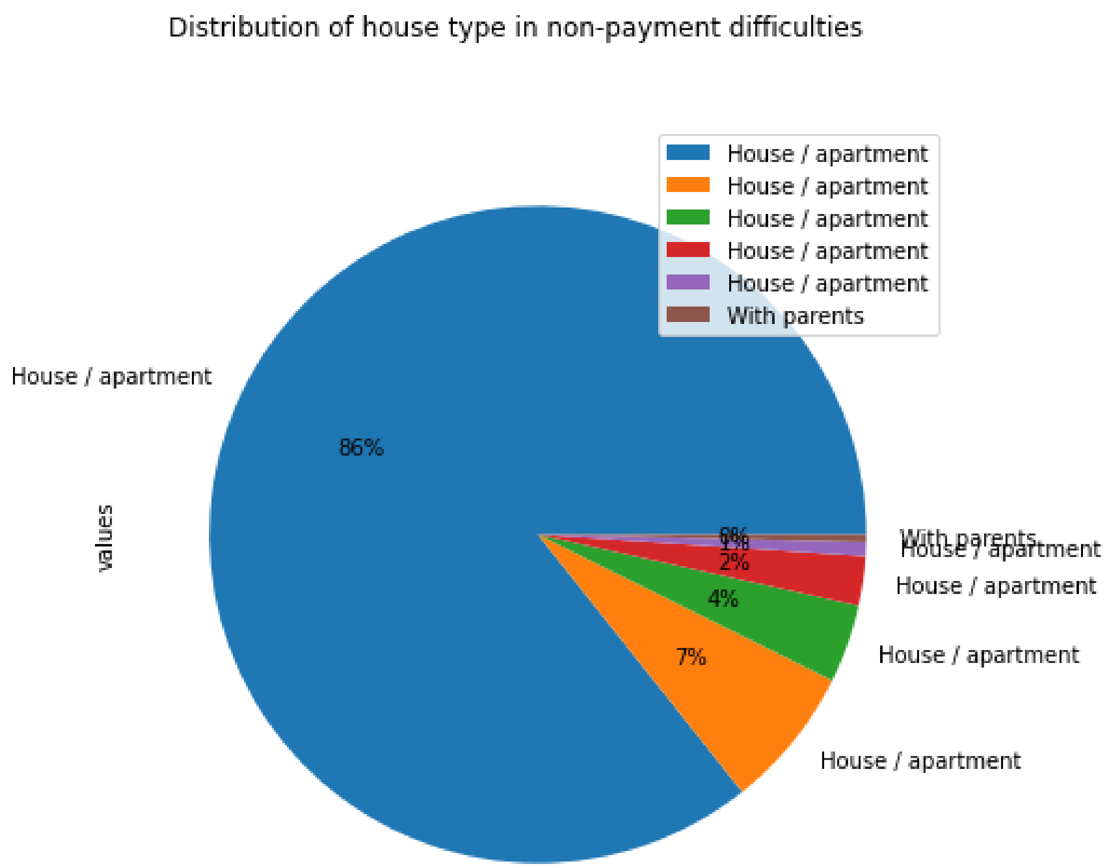



```
#Now we analyse the type of house of applicants having loan
difficulties

#ah0 represents analysis of house type for target variable 0.

ah0 = t0["NAME_HOUSING_TYPE"].value_counts()
df9 = pd.DataFrame({'values': ah0.values})
df9.plot.pie(labels=t0.NAME_HOUSING_TYPE,title='Distribution of house
type in payment difficulties', subplots=True, autopct='%1.0f%%')

#ah1 represents analysis of house type for target variable 1.

ah1 = t1["NAME_HOUSING_TYPE"].value_counts()
df9 = pd.DataFrame({'values': ah1.values})
df9.plot.pie(labels=t1.NAME_HOUSING_TYPE,title='Distribution of house
type in non-payment difficulties', subplots=True, autopct='%1.0f%%')
```

```
array([<AxesSubplot:ylabel='values'>], dtype=object)
```



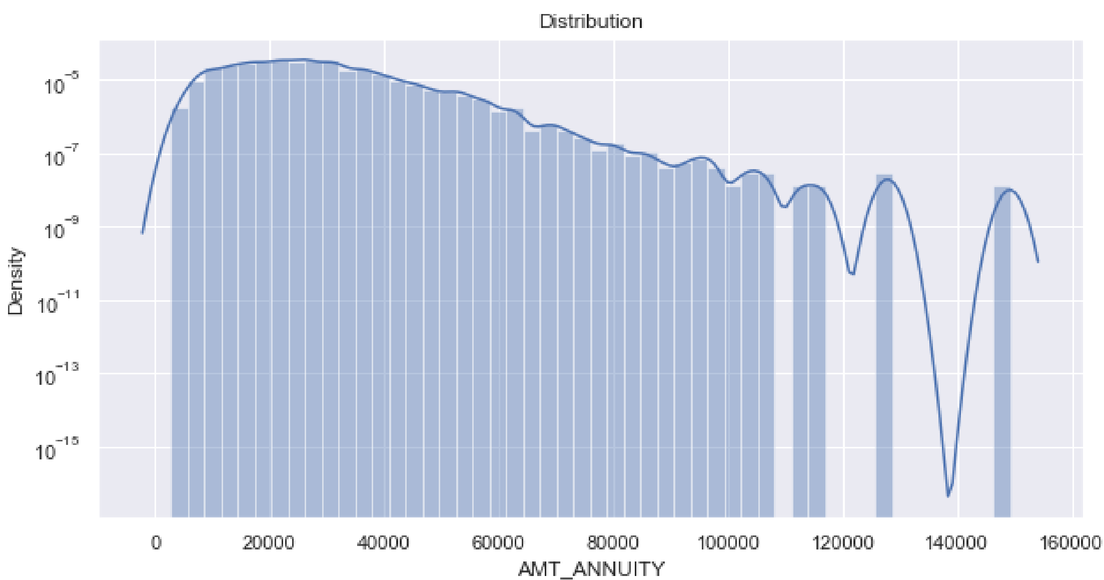
Distribution of house type in payment difficulties
House / apartment
House / apartment
House / apartment
House / apartment
House / apartment
House / apartment
values
House / apartment
89%
5%
4%
2%
House / apartment
House / apartment
House / apartment
House / apartment
House / apartment



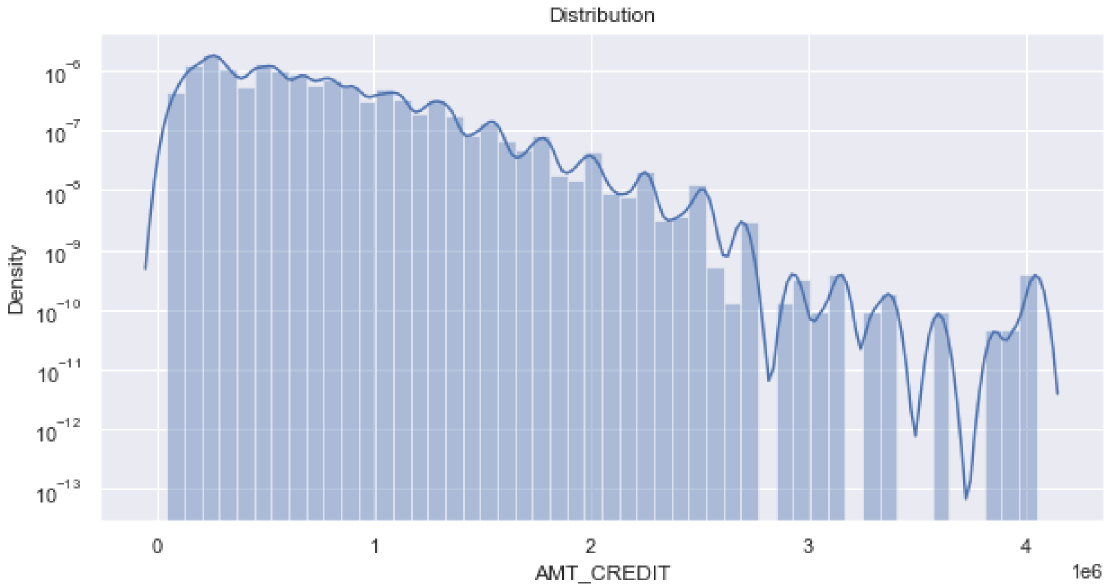
Univariate analysis on the basis of target variable

```
#Defining a function that would work for all the target variables

def dist(df,col,hue =None):
    fig, ax=plt.subplots(figsize=(10,5))
    ax.set_title("Distribution")
    sns.distplot(df[~df[col].isna()][col])
    plt.yscale('log')
    plt.show()

# Distribution for 'AMT_ANNUITY' for target variable 0
dist(df=t0,col='AMT_ANNUITY')
```

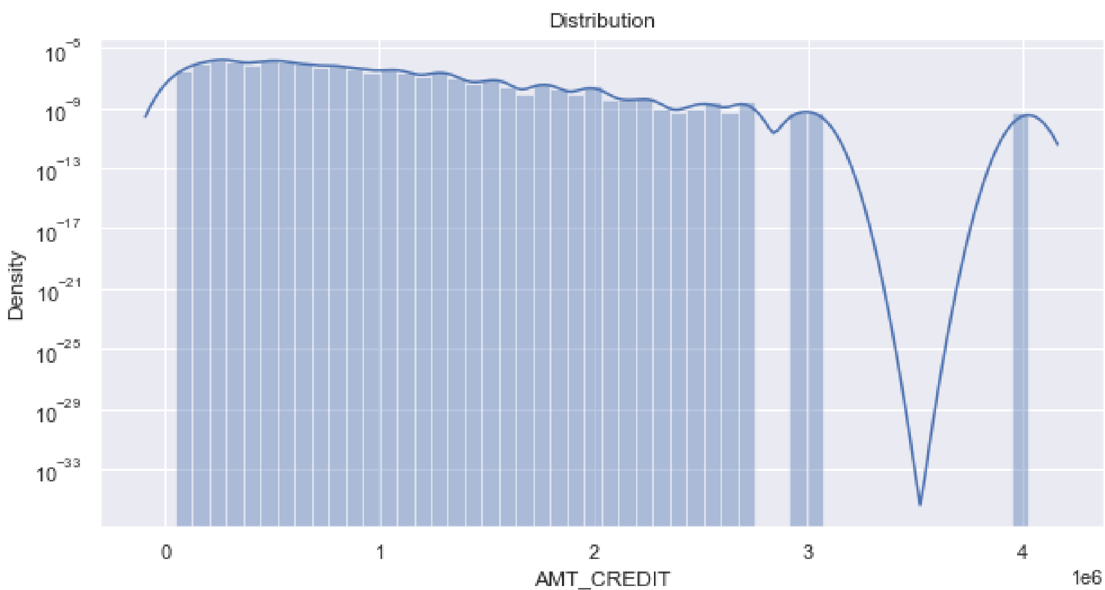

```
# Distribution for 'AMT_ANNUITY' for target variable 1
dist(df=t1,col='AMT_ANNUITY')
```




```
# Distribution for 'AMT_CREDIT' for target variable 0
dist(df=t0,col='AMT_CREDIT')
```




```
# Distribution for 'AMT_CREDIT' for target variable 1
dist(df=t1,col='AMT_CREDIT')
```



Data analysis for Previous data

```
# Replacing the negative values by positive values of name starting
with 'DAYS'

new_col = [col for col in previous if col.startswith('DAYS')]
previous[new_col]= abs(previous[filter_col])
previous.head()

   SK_ID_PREV  SK_ID_CURR NAME_CONTRACT_TYPE  AMT_ANNUITY
AMT_APPLICATION  \
0     2030495      271877     Consumer loans     1730.430
```

```
17145.0
1     2802425      108129         Cash loans    25188.615
607500.0
2     2523466      122040         Cash loans    15060.735
112500.0
3     2819243      176158         Cash loans    47041.335
450000.0
4     1784265      202054         Cash loans    31924.395
337500.0

   AMT_CREDIT  AMT_DOWN_PAYMENT  AMT_GOODS_PRICE
WEEKDAY_APPR_PROCESS_START  \
0     17145.0               0.0          17145.0
SATURDAY
1    679671.0               NaN         607500.0
THURSDAY
2    136444.5               NaN         112500.0
TUESDAY
3    470790.0               NaN         450000.0
MONDAY
4    404055.0               NaN         337500.0
THURSDAY

   HOUR_APPR_PROCESS_START  ... NAME_SELLER_INDUSTRY  CNT_PAYMENT  \
0                       15  ...         Connectivity         12.0
1                       11  ...                  NaN         36.0
2                       11  ...                  NaN         12.0
3                        7  ...                  NaN         12.0
4                        9  ...                  NaN         24.0

   NAME_YIELD_GROUP       PRODUCT_COMBINATION  DAYS_FIRST_DRAWING  \
0            middle  POS mobile with interest            365243.0
1        low_action          Cash X-Sell: low            365243.0
2              high         Cash X-Sell: high            365243.0
3            middle       Cash X-Sell: middle            365243.0
4              high         Cash Street: high                 NaN

  DAYS_FIRST_DUE DAYS_LAST_DUE_1ST_VERSION  DAYS_LAST_DUE
DAYS_TERMINATION  \
0           42.0                     300.0           42.0
37.0
1          134.0                     916.0       365243.0
365243.0
2          271.0                      59.0       365243.0
365243.0
3          482.0                     152.0          182.0
177.0
4            NaN                       NaN            NaN
NaN
```

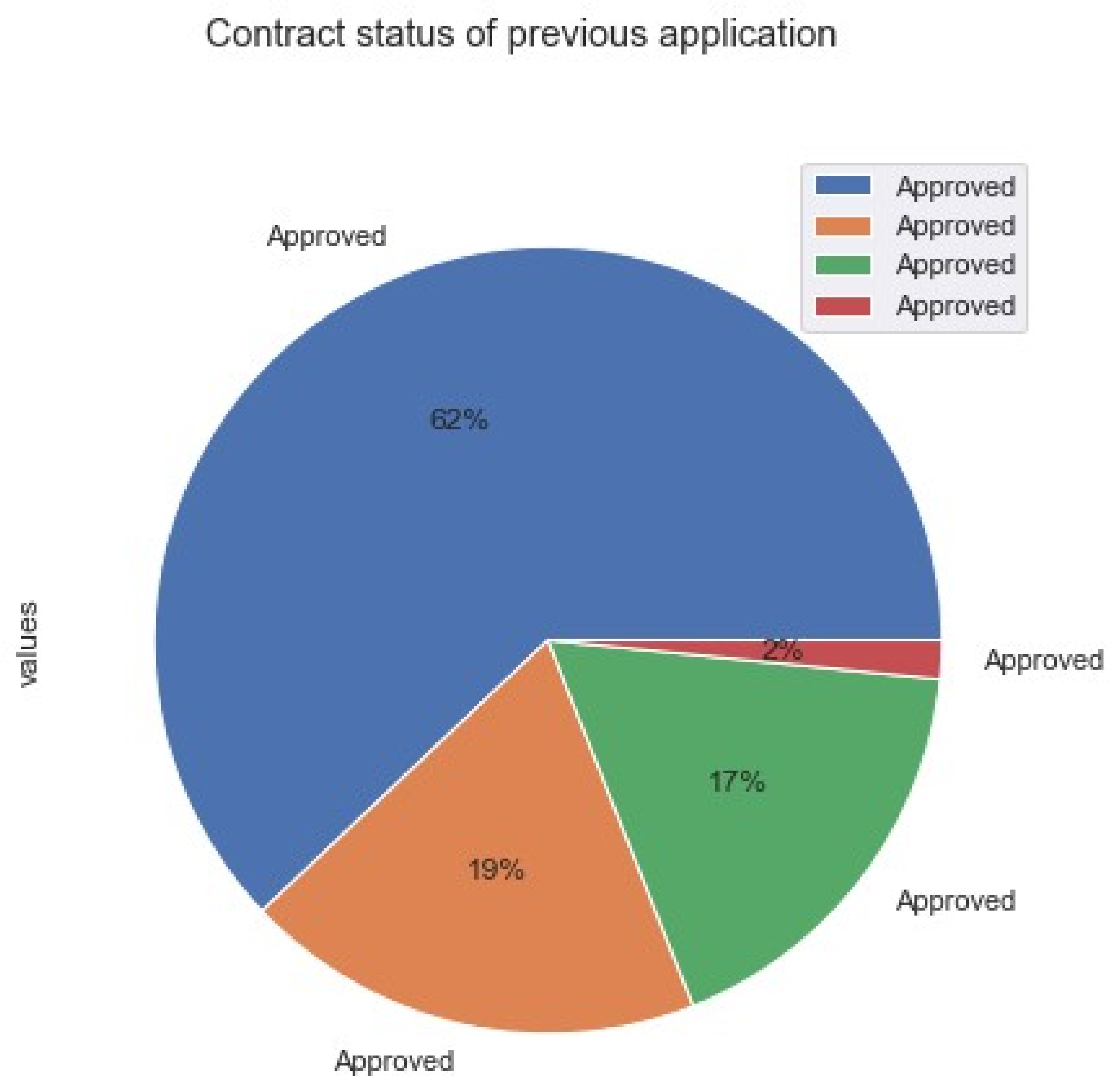
```
  NFLAG_INSURED_ON_APPROVAL
0                        0.0
1                        1.0
2                        1.0
3                        1.0
4                        NaN

[5 rows x 37 columns]
```

```
#Replace XAP & XNA values by np.NaN

previous=previous.replace('XNA', np.NaN)
previous=previous.replace('XAP', np.NaN)

previous['NAME_CONTRACT_STATUS'].value_counts()
```

```
Approved        1036781
Canceled         316319
Refused          290678
Unused offer      26436
Name: NAME_CONTRACT_STATUS, dtype: int64
```

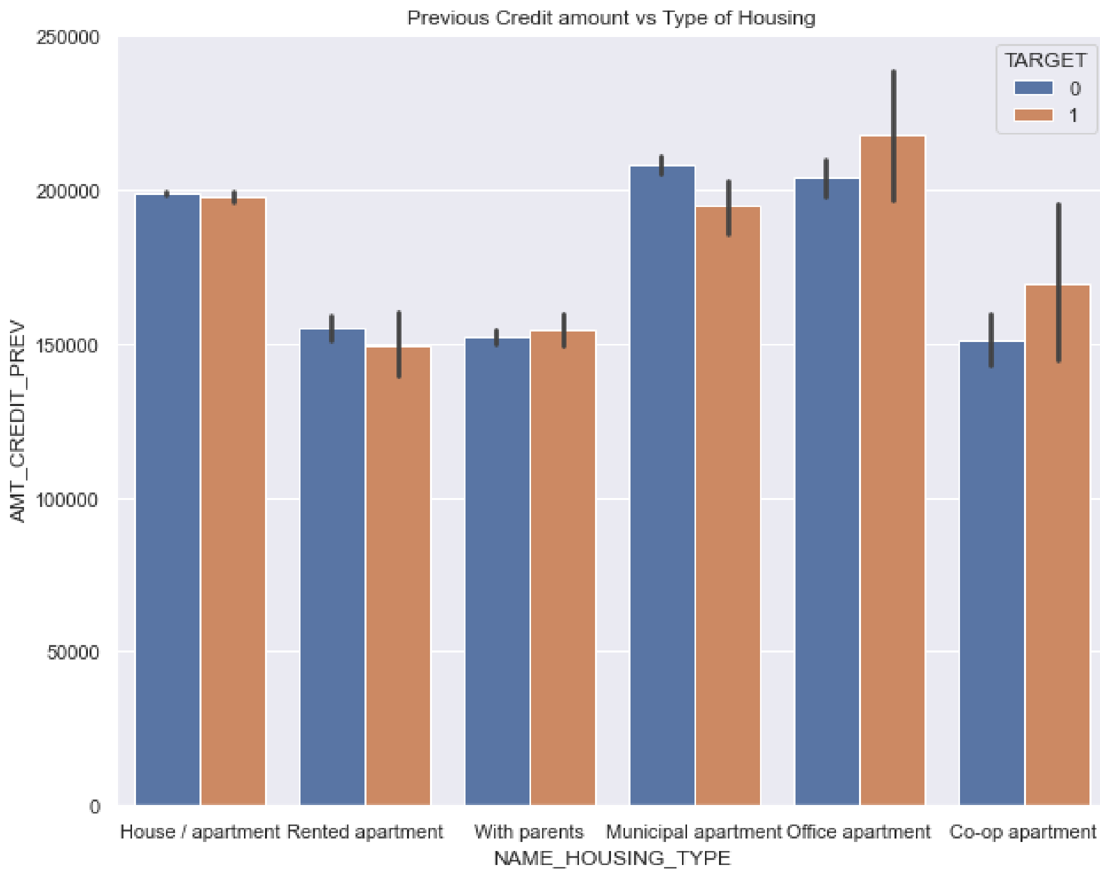
```
# Contract status of previous application

con = previous["NAME_CONTRACT_STATUS"].value_counts()
df = pd.DataFrame({'values': con.values})
df.plot.pie(labels=previous['NAME_CONTRACT_STATUS'],title='Contract
status of previous application', subplots=True, autopct='%1.0f%%')
```

```
array([<AxesSubplot:ylabel='values'>], dtype=object)
```




```
#Merging two tables

mergeddf=pd.merge(left=data,right=previous,how='inner',on='SK_ID_CURR'
,suffixes='_x')
mergedfinal = new_df.rename({'NAME_CONTRACT_TYPE_' :
'NAME_CONTRACT_TYPE','AMT_CREDIT_':'AMT_CREDIT','AMT_ANNUITY_':'AMT_AN
NUITY',
                         'WEEKDAY_APPR_PROCESS_START_' :
'WEEKDAY_APPR_PROCESS_START',

'HOUR_APPR_PROCESS_START_':'HOUR_APPR_PROCESS_START','NAME_CONTRACT_TY
PEx':'NAME_CONTRACT_TYPE_PREV',

'AMT_CREDITx':'AMT_CREDIT_PREV','AMT_ANNUITYx':'AMT_ANNUITY_PREV',
```

```
'WEEKDAY_APPR_PROCESS_STARTx':'WEEKDAY_APPR_PROCESS_START_PREV',

'HOUR_APPR_PROCESS_STARTx':'HOUR_APPR_PROCESS_START_PREV'}, axis=1)
# Box plotting for Credit amount prev vs Housing type

plt.figure(figsize=(10,8))

sns.barplot(data =mergedfinal,
y='AMT_CREDIT_PREV',hue='TARGET',x='NAME_HOUSING_TYPE')
plt.title('Previous Credit amount vs Type of Housing')
plt.show()
```



INSIGHTS AND CONCLUSION

- For the probability of successful payments to be high, banks should focus on Businessmen and Students and should avoid the Working people as they have the maximum amount of defaults.
- Banks should focus on the apartment types other than Co-ap apartment and Office apartment