```
## importing all the necessery libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

## importing the warnings to avoid unnecessary
import warnings
warnings.filterwarnings("ignore")

## loading the dataset
df=pd.read_csv('loan.csv')
df.head()
```

```
        id  member_id  loan_amnt  funded_amnt  funded_amnt_inv        
term  \
0  1077501    1296599       5000         5000           4975.0   36 
months   
1  1077430    1314167       2500         2500           2500.0   60 
months   
2  1077175    1313524       2400         2400           2400.0   36 
months   
3  1076863    1277178      10000        10000          10000.0   36 
months   
4  1075358    1311748       3000         3000           3000.0   60 
months   

  int_rate  installment grade sub_grade  ... num_tl_90g_dpd_24m  \
0   10.65%       162.87     B        B2  ...                NaN   
1   15.27%        59.83     C        C4  ...                NaN   
2   15.96%        84.33     C        C5  ...                NaN   
3   13.49%       339.31     C        C1  ...                NaN   
4   12.69%        67.79     B        B5  ...                NaN   

  num_tl_op_past_12m pct_tl_nvr_dlq  percent_bc_gt_75  
pub_rec_bankruptcies  \
0                NaN            NaN               NaN                   
0.0   
1                NaN            NaN               NaN                   
0.0   
2                NaN            NaN               NaN                   
0.0   
3                NaN            NaN               NaN                   
0.0   
4                NaN            NaN               NaN                   
0.0   

  tax_liens tot_hi_cred_lim total_bal_ex_mort total_bc_limit  \
0       0.0             NaN               NaN            NaN   
```

```
1       0.0             NaN                 NaN              NaN
2       0.0             NaN                 NaN              NaN
3       0.0             NaN                 NaN              NaN
4       0.0             NaN                 NaN              NaN

  total_il_high_credit_limit
0                         NaN
1                         NaN
2                         NaN
3                         NaN
4                         NaN

[5 rows x 111 columns]
```

```
## get the details of  the dataframe
df.shape
```

```
(39717, 111)
```

so the data is having 111 columns and 39717 rows!!

```
## some statistical description about the column
df.describe()
```

```
                 id     member_id     loan_amnt   funded_amnt  \
count  3.971700e+04  3.971700e+04  39717.000000  39717.000000
mean   6.831319e+05  8.504636e+05  11219.443815  10947.713196
std    2.106941e+05  2.656783e+05   7456.670694   7187.238670
min    5.473400e+04  7.069900e+04    500.000000    500.000000
25%    5.162210e+05  6.667800e+05   5500.000000   5400.000000
50%    6.656650e+05  8.508120e+05  10000.000000   9600.000000
75%    8.377550e+05  1.047339e+06  15000.000000  15000.000000
max    1.077501e+06  1.314167e+06  35000.000000  35000.000000

       funded_amnt_inv   installment    annual_inc           dti  \
count     39717.000000  39717.000000  3.971700e+04  39717.000000
mean      10397.448868    324.561922  6.896893e+04     13.315130
std        7128.450439    208.874874  6.379377e+04      6.678594
min           0.000000     15.690000  4.000000e+03      0.000000
25%        5000.000000    167.020000  4.040400e+04      8.170000
50%        8975.000000    280.220000  5.900000e+04     13.400000
75%       14400.000000    430.780000  8.230000e+04     18.600000
max       35000.000000   1305.190000  6.000000e+06     29.990000

        delinq_2yrs  inq_last_6mths  ...  num_tl_90g_dpd_24m  \
count  39717.000000    39717.000000  ...                 0.0
mean       0.146512        0.869200  ...                 NaN
std        0.491812        1.070219  ...                 NaN
min        0.000000        0.000000  ...                 NaN
25%        0.000000        0.000000  ...                 NaN
50%        0.000000        1.000000  ...                 NaN
```

```
75%        0.000000        1.000000  ...                 NaN
max       11.000000        8.000000  ...                 NaN

       num_tl_op_past_12m  pct_tl_nvr_dlq  percent_bc_gt_75  \
count                 0.0             0.0               0.0
mean                  NaN             NaN               NaN
std                   NaN             NaN               NaN
min                   NaN             NaN               NaN
25%                   NaN             NaN               NaN
50%                   NaN             NaN               NaN
75%                   NaN             NaN               NaN
max                   NaN             NaN               NaN

       pub_rec_bankruptcies  tax_liens  tot_hi_cred_lim
total_bal_ex_mort  \
count          39020.000000    39678.0              0.0
0.0
mean               0.043260        0.0              NaN
NaN
std                0.204324        0.0              NaN
NaN
min                0.000000        0.0              NaN
NaN
25%                0.000000        0.0              NaN
NaN
50%                0.000000        0.0              NaN
NaN
75%                0.000000        0.0              NaN
NaN
max                2.000000        0.0              NaN
NaN

       total_bc_limit  total_il_high_credit_limit
count             0.0                         0.0
mean              NaN                         NaN
std               NaN                         NaN
min               NaN                         NaN
25%               NaN                         NaN
50%               NaN                         NaN
75%               NaN                         NaN
max               NaN                         NaN

[8 rows x 87 columns]
```

```
## get the names of all the columns
df.columns
```

```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt',
'funded_amnt_inv',
       'term', 'int_rate', 'installment', 'grade', 'sub_grade',
```

```
       ...
       'num_tl_90g_dpd_24m', 'num_tl_op_past_12m', 'pct_tl_nvr_dlq',
       'percent_bc_gt_75', 'pub_rec_bankruptcies', 'tax_liens',
       'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit',
       'total_il_high_credit_limit'],
      dtype='object', length=111)
```

```
#Check the datatypes of all the columns of the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Columns: 111 entries, id to total_il_high_credit_limit
dtypes: float64(74), int64(13), object(24)
memory usage: 33.6+ MB
```

List of Columns & NA counts where NA values

```
na_values=df.isnull().sum()
## checking all the columns where the null values are greater than 30 percent
na_col=na_values[na_values.values>0.30*len(df)]
na_col
```

```
desc                              12940
mths_since_last_delinq            25682
mths_since_last_record            36931
next_pymnt_d                      38577
mths_since_last_major_derog       39717
annual_inc_joint                  39717
dti_joint                         39717
verification_status_joint         39717
tot_coll_amt                      39717
tot_cur_bal                       39717
open_acc_6m                       39717
open_il_6m                        39717
open_il_12m                       39717
open_il_24m                       39717
mths_since_rcnt_il                39717
total_bal_il                      39717
il_util                           39717
open_rv_12m                       39717
open_rv_24m                       39717
max_bal_bc                        39717
all_util                          39717
total_rev_hi_lim                  39717
inq_fi                            39717
total_cu_tl                       39717
inq_last_12m                      39717
acc_open_past_24mths              39717
avg_cur_bal                       39717
bc_open_to_buy                    39717
```

```
bc_util                            39717
mo_sin_old_il_acct                 39717
mo_sin_old_rev_tl_op               39717
mo_sin_rcnt_rev_tl_op              39717
mo_sin_rcnt_tl                     39717
mort_acc                           39717
mths_since_recent_bc               39717
mths_since_recent_bc_dlq           39717
mths_since_recent_inq              39717
mths_since_recent_revol_delinq     39717
num_accts_ever_120_pd              39717
num_actv_bc_tl                     39717
num_actv_rev_tl                    39717
num_bc_sats                        39717
num_bc_tl                          39717
num_il_tl                          39717
num_op_rev_tl                      39717
num_rev_accts                      39717
num_rev_tl_bal_gt_0                39717
num_sats                           39717
num_tl_120dpd_2m                   39717
num_tl_30dpd                       39717
num_tl_90g_dpd_24m                 39717
num_tl_op_past_12m                 39717
pct_tl_nvr_dlq                     39717
percent_bc_gt_75                   39717
tot_hi_cred_lim                    39717
total_bal_ex_mort                  39717
total_bc_limit                     39717
total_il_high_credit_limit         39717
dtype: int64
```

```
##  printing down all the columns with more than 30 percent of the
null values
na_col.index
```

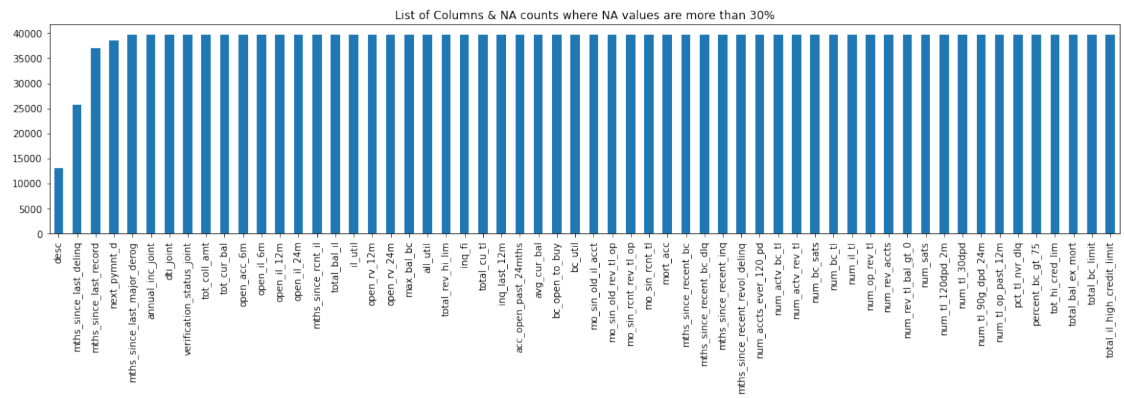
```
Index(['desc', 'mths_since_last_delinq', 'mths_since_last_record',
       'next_pymnt_d', 'mths_since_last_major_derog',
'annual_inc_joint',
       'dti_joint', 'verification_status_joint', 'tot_coll_amt',
'tot_cur_bal',
       'open_acc_6m', 'open_il_6m', 'open_il_12m', 'open_il_24m',
       'mths_since_rcnt_il', 'total_bal_il', 'il_util', 'open_rv_12m',
       'open_rv_24m', 'max_bal_bc', 'all_util', 'total_rev_hi_lim',
'inq_fi',
       'total_cu_tl', 'inq_last_12m', 'acc_open_past_24mths',
'avg_cur_bal',
       'bc_open_to_buy', 'bc_util', 'mo_sin_old_il_acct',
       'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op',
'mo_sin_rcnt_tl',
       'mort_acc', 'mths_since_recent_bc', 'mths_since_recent_bc_dlq',
```

```
       'mths_since_recent_inq', 'mths_since_recent_revol_delinq',
       'num_accts_ever_120_pd', 'num_actv_bc_tl', 'num_actv_rev_tl',
       'num_bc_sats', 'num_bc_tl', 'num_il_tl', 'num_op_rev_tl',
       'num_rev_accts', 'num_rev_tl_bal_gt_0', 'num_sats',
'num_tl_120dpd_2m',
       'num_tl_30dpd', 'num_tl_90g_dpd_24m', 'num_tl_op_past_12m',
       'pct_tl_nvr_dlq', 'percent_bc_gt_75', 'tot_hi_cred_lim',
       'total_bal_ex_mort', 'total_bc_limit',
'total_il_high_credit_limit'],
      dtype='object')
```

```
print("the number of columns which are having more than 30 percent nan
values are ",len(na_col))
```

```
the number of columns which are having more than 30 percent nan values
are  58
```

```
plt.figure(figsize=(20,4))
na_col.plot(kind='bar')
plt.title('List of Columns & NA counts where NA values are more than
30%')
plt.show()
```



Insights: So we can see from the above plot that there are 58 columns in the dataset where all the values are NA.

As we can see there are 887379 rows & 74 columns in the dataset, it will be very difficult to look at each column one by one & find the NA or missing values. So let's find out all columns where missing values are more than certain percentage, let's say 30%. We will remove those columns as it is not feasable to impute missing values for those columns.

```
## taking all the column names in the list and removing or dropping
all the columns
columns_with_nanvalues=['desc', 'mths_since_last_delinq',
'mths_since_last_record','next_pymnt_d',
'mths_since_last_major_derog', 'annual_inc_joint', 'dti_joint',
'verification_status_joint', 'tot_coll_amt',
'tot_cur_bal','open_acc_6m', 'open_il_6m', 'open_il_12m',
'open_il_24m','mths_since_rcnt_il', 'total_bal_il', 'il_util',
```

```
'open_rv_12m','open_rv_24m', 'max_bal_bc', 'all_util',
'total_rev_hi_lim', 'inq_fi','total_cu_tl', 'inq_last_12m',
'acc_open_past_24mths', 'avg_cur_bal','bc_open_to_buy', 'bc_util',
'mo_sin_old_il_acct','mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op',
'mo_sin_rcnt_tl','mort_acc', 'mths_since_recent_bc',
'mths_since_recent_bc_dlq','mths_since_recent_inq',
'mths_since_recent_revol_delinq','num_accts_ever_120_pd',
'num_actv_bc_tl', 'num_actv_rev_tl','num_bc_sats', 'num_bc_tl',
'num_il_tl', 'num_op_rev_tl','num_rev_accts', 'num_rev_tl_bal_gt_0',
'num_sats', 'num_tl_120dpd_2m','num_tl_30dpd', 'num_tl_90g_dpd_24m',
'num_tl_op_past_12m','pct_tl_nvr_dlq', 'percent_bc_gt_75',
'tot_hi_cred_lim','total_bal_ex_mort', 'total_bc_limit',
'total_il_high_credit_limit']
new_df=df.drop(columns=columns_with_nanvalues,axis=1)

new_df.head()
```

```
        id  member_id  loan_amnt  funded_amnt  funded_amnt_inv
term  \
0  1077501    1296599       5000         5000           4975.0   36
months
1  1077430    1314167       2500         2500           2500.0   60
months
2  1077175    1313524       2400         2400           2400.0   36
months
3  1076863    1277178      10000        10000          10000.0   36
months
4  1075358    1311748       3000         3000           3000.0   60
months

  int_rate  installment grade sub_grade  ... last_pymnt_amnt  \
0   10.65%       162.87     B        B2  ...          171.62
1   15.27%        59.83     C        C4  ...          119.66
2   15.96%        84.33     C        C5  ...          649.91
3   13.49%       339.31     C        C1  ...          357.48
4   12.69%        67.79     B        B5  ...           67.79

  last_credit_pull_d collections_12_mths_ex_med  policy_code
application_type  \
0             May-16                        0.0            1
INDIVIDUAL
1             Sep-13                        0.0            1
INDIVIDUAL
2             May-16                        0.0            1
INDIVIDUAL
3             Apr-16                        0.0            1
INDIVIDUAL
4             May-16                        0.0            1
INDIVIDUAL
```

```
  acc_now_delinq chargeoff_within_12_mths delinq_amnt
pub_rec_bankruptcies  \
0              0                       0.0            0
0.0
1              0                       0.0            0
0.0
2              0                       0.0            0
0.0
3              0                       0.0            0
0.0
4              0                       0.0            0
0.0

  tax_liens
0       0.0
1       0.0
2       0.0
3       0.0
4       0.0

[5 rows x 53 columns]
```

```
## printing the old dataframe AND THE new dataframe
print("the shape of old dataframe is",df.shape)
print("After dropping the columns with more than 30 percent is")
print("the shape of new dataframe is", new_df.shape)
```

```
the shape of old dataframe is (39717, 111)
After dropping the columns with more than 30 percent is
the shape of new dataframe is (39717, 53)
```

```
## now that we have 53 columns lets look at the column names
new_df.columns
```

```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt',
'funded_amnt_inv',
       'term', 'int_rate', 'installment', 'grade', 'sub_grade',
'emp_title',
       'emp_length', 'home_ownership', 'annual_inc',
'verification_status',
       'issue_d', 'loan_status', 'pymnt_plan', 'url', 'purpose',
'title',
       'zip_code', 'addr_state', 'dti', 'delinq_2yrs',
'earliest_cr_line',
       'inq_last_6mths', 'open_acc', 'pub_rec', 'revol_bal',
'revol_util',
       'total_acc', 'initial_list_status', 'out_prncp',
'out_prncp_inv',
       'total_pymnt', 'total_pymnt_inv', 'total_rec_prncp',
'total_rec_int',
       'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
```

```
       'last_pymnt_d', 'last_pymnt_amnt', 'last_credit_pull_d',
       'collections_12_mths_ex_med', 'policy_code',
'application_type',
       'acc_now_delinq', 'chargeoff_within_12_mths', 'delinq_amnt',
       'pub_rec_bankruptcies', 'tax_liens'],
      dtype='object')
```

Remove irrelevant columns. Till now we have removed the columns based on the count & statistics. Now let's look at each column from business perspective if that is required or not for our analysis such as Unique ID's, URL. As last 2 digits of zip code is masked 'xx', we can remove that as well.

```
irrelevant_columns=['id','member_id','zip_code','url']
new_df.drop(columns=irrelevant_columns,axis=1,inplace=True)
```

```
## lets check the dataframe columns now
new_df.shape
```

```
(39717, 49)
```

```
new_df.columns
```

```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term',
'int_rate',
       'installment', 'grade', 'sub_grade', 'emp_title', 'emp_length',
       'home_ownership', 'annual_inc', 'verification_status',
'issue_d',
       'loan_status', 'pymnt_plan', 'purpose', 'title', 'addr_state',
'dti',
       'delinq_2yrs', 'earliest_cr_line', 'inq_last_6mths',
'open_acc',
       'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'initial_list_status', 'out_prncp', 'out_prncp_inv',
'total_pymnt',
       'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
       'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
       'last_pymnt_d', 'last_pymnt_amnt', 'last_credit_pull_d',
       'collections_12_mths_ex_med', 'policy_code',
'application_type',
       'acc_now_delinq', 'chargeoff_within_12_mths', 'delinq_amnt',
       'pub_rec_bankruptcies', 'tax_liens'],
      dtype='object')
```

Remove columns where number of unique value is only 1. Let's look at no of unique values for each column.We will remove all columns where number of unique value is only 1 because that will not make any sense in the analysis

```
unique=new_df.nunique()
unique_columns=unique[unique.values==1]
unique_columns.index
```

```
Index(['pymnt_plan', 'initial_list_status',
'collections_12_mths_ex_med',
       'policy_code', 'application_type', 'acc_now_delinq',
       'chargeoff_within_12_mths', 'delinq_amnt', 'tax_liens'],
      dtype='object')
```

```
## again removing the columns which is containing the unique val;ue
only 1
unique_value_columns=['pymnt_plan', 'initial_list_status',
'collections_12_mths_ex_med',
       'policy_code', 'application_type', 'acc_now_delinq',
       'chargeoff_within_12_mths', 'delinq_amnt', 'tax_liens']
new_df.drop(columns=unique_value_columns,axis=1,inplace=True)
```

```
new_df.shape
```

```
(39717, 40)
```

now that we have a columns with 40 features it is now useful to do some analysis and the manupilation by fropping the irrelevant column names

```
new_df.head()
```

```
   loan_amnt  funded_amnt  funded_amnt_inv        term int_rate
installment  \
0       5000         5000           4975.0   36 months   10.65%
162.87
1       2500         2500           2500.0   60 months   15.27%
59.83
2       2400         2400           2400.0   36 months   15.96%
84.33
3      10000        10000          10000.0   36 months   13.49%
339.31
4       3000         3000           3000.0   60 months   12.69%
67.79

  grade sub_grade                 emp_title emp_length  ...
total_pymnt_inv  \
0     B        B2                       NaN  10+ years  ...
5833.84
1     C        C4                     Ryder   < 1 year  ...
1008.71
2     C        C5                       NaN  10+ years  ...
3005.67
3     C        C1       AIR RESOURCES BOARD  10+ years  ...
12231.89
4     B        B5  University Medical Group     1 year  ...
3513.33

   total_rec_prncp total_rec_int total_rec_late_fee recoveries  \
0          5000.00        863.16               0.00       0.00
1           456.46        435.17               0.00     117.08
```

```
2          2400.00        605.67                 0.00       0.00
3         10000.00       2214.92                16.97       0.00
4          2475.94       1037.39                 0.00       0.00

  collection_recovery_fee last_pymnt_d last_pymnt_amnt
last_credit_pull_d  \
0                    0.00       Jan-15          171.62
May-16
1                    1.11       Apr-13          119.66
Sep-13
2                    0.00       Jun-14          649.91
May-16
3                    0.00       Jan-15          357.48
Apr-16
4                    0.00       May-16           67.79
May-16

   pub_rec_bankruptcies
0                   0.0
1                   0.0
2                   0.0
3                   0.0
4                   0.0

[5 rows x 40 columns]
```

```
new_df.columns
```

```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term',
'int_rate',
       'installment', 'grade', 'sub_grade', 'emp_title', 'emp_length',
       'home_ownership', 'annual_inc', 'verification_status',
'issue_d',
       'loan_status', 'purpose', 'title', 'addr_state', 'dti',
'delinq_2yrs',
       'earliest_cr_line', 'inq_last_6mths', 'open_acc', 'pub_rec',
       'revol_bal', 'revol_util', 'total_acc', 'out_prncp',
'out_prncp_inv',
       'total_pymnt', 'total_pymnt_inv', 'total_rec_prncp',
'total_rec_int',
       'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
       'last_pymnt_d', 'last_pymnt_amnt', 'last_credit_pull_d',
       'pub_rec_bankruptcies'],
      dtype='object')
```

```
## we will be now categorizing the data into the numerical and the
categorical data
numerical_features=[features for features in new_df if
new_df[features].dtype!='o']
categorical_features=[features for features in new_df if
```

```
new_df[features].dtype=='o']
print("the numerical_features are",numerical_features)
print("")
print("the categorical features are ",categorical_features)
```

```
the numerical_features are ['loan_amnt', 'funded_amnt',
'funded_amnt_inv', 'term', 'int_rate', 'installment', 'grade',
'sub_grade', 'emp_title', 'emp_length', 'home_ownership',
'annual_inc', 'verification_status', 'issue_d', 'loan_status',
'purpose', 'title', 'addr_state', 'dti', 'delinq_2yrs',
'earliest_cr_line', 'inq_last_6mths', 'open_acc', 'pub_rec',
'revol_bal', 'revol_util', 'total_acc', 'out_prncp', 'out_prncp_inv',
'total_pymnt', 'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
'last_pymnt_d', 'last_pymnt_amnt', 'last_credit_pull_d',
'pub_rec_bankruptcies']

the categorical features are  []
```

lets now start exploring the feature one by one

```
## we will check now the null values in the features
new_df.isnull().sum()
```

```
loan_amnt                  0
funded_amnt                0
funded_amnt_inv            0
term                       0
int_rate                   0
installment                0
grade                      0
sub_grade                  0
emp_title               2459
emp_length              1075
home_ownership             0
annual_inc                 0
verification_status        0
issue_d                    0
loan_status                0
purpose                    0
title                     11
addr_state                 0
dti                        0
delinq_2yrs                0
earliest_cr_line           0
inq_last_6mths             0
open_acc                   0
pub_rec                    0
revol_bal                  0
revol_util                50
total_acc                  0
```

```
out_prncp                    0
out_prncp_inv                0
total_pymnt                  0
total_pymnt_inv              0
total_rec_prncp              0
total_rec_int                0
total_rec_late_fee           0
recoveries                   0
collection_recovery_fee      0
last_pymnt_d                71
last_pymnt_amnt              0
last_credit_pull_d           2
pub_rec_bankruptcies       697
dtype: int64
```

```
## lets explore the title feature of the new_df
new_df['title'].value_counts()
## removing the title column since this is not necesary for the
analysis
new_df.drop(columns=['title','emp_title'],axis=1,inplace=True)
new_df.shape
```

```
(39717, 38)
```

```
new_df.columns
```

```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term',
'int_rate',
       'installment', 'grade', 'sub_grade', 'emp_length',
'home_ownership',
       'annual_inc', 'verification_status', 'issue_d', 'loan_status',
       'purpose', 'addr_state', 'dti', 'delinq_2yrs',
'earliest_cr_line',
       'inq_last_6mths', 'open_acc', 'pub_rec', 'revol_bal',
'revol_util',
       'total_acc', 'out_prncp', 'out_prncp_inv', 'total_pymnt',
       'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
       'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
       'last_pymnt_d', 'last_pymnt_amnt', 'last_credit_pull_d',
       'pub_rec_bankruptcies'],
      dtype='object')
```

Data Handling and Cleaning

The first few steps involve making sure that there are no **missing values** or **incorrect data types** before we proceed to the analysis stage. These aforementioned problems are handled as follows:

- For Missing Values: Some common techniques to treat this issue are
  - Dropping the rows containing the missing values
  - Imputing the missing values

    - Keep the missing values if they don't affect the analysis
- Incorrect Data Types:
    - Clean certain values
    - Clean and convert an entire column

```
new_df['term'].value_counts()
```

```
 36 months    29096
 60 months    10621
Name: term, dtype: int64
```

```
## removing the months from the term features
def manupliate_terms(value):
    return int(value.replace("months",""))
manupliate_terms("36 months")
## calling the new  df term feature
new_df['term']=new_df['term'].apply(manupliate_terms)
```

```
new_df['term'].value_counts()
```

```
36    29096
60    10621
Name: term, dtype: int64
```

```
## exploring the int_rate feature now
new_df['int_rate'].value_counts()
```

```
## creating a function whcih will remmove all the percent signs
def manupilate_int_rate(value):
    return float(value.replace("%",""))
manupilate_int_rate("10.99%")
## applying it in the feature
new_df['int_rate']=new_df['int_rate'].apply(manupilate_int_rate)
```

```
new_df['int_rate'].value_counts()
```

```
10.99    956
13.49    826
11.49    825
7.51     787
7.88     725
        ...
18.36      1
16.96      1
16.15      1
16.01      1
17.44      1
Name: int_rate, Length: 371, dtype: int64
```

```
new_df['emp_length'].value_counts()
new_df['emp_length'].unique()
```

```
array(['10+ years', '< 1 year', '1 year', '3 years', '8 years', '9 
years',
       '4 years', '5 years', '6 years', '2 years', '7 years', nan],
      dtype=object)
```

```
## lets explore the employee  length and replace the nan  value with 
the self employed
new_df['emp_length']=new_df['emp_length'].fillna('0')
```

```
new_df['emp_length'].unique()
```

```
array(['10+ years', '< 1 year', '1 year', '3 years', '8 years', '9 
years',
       '4 years', '5 years', '6 years', '2 years', '7 years', '0'],
      dtype=object)
```

```
new_df['sub_grade'].unique()
```

```
array(['B2', 'C4', 'C5', 'C1', 'B5', 'A4', 'E1', 'F2', 'C3', 'B1', 
'D1',
       'A1', 'B3', 'B4', 'C2', 'D2', 'A3', 'A5', 'D5', 'A2', 'E4', 
'D3',
       'D4', 'F3', 'E3', 'F4', 'F1', 'E5', 'G4', 'E2', 'G3', 'G2', 
'G1',
       'F5', 'G5'], dtype=object)
```

```
new_df.isnull().sum()
```

```
loan_amnt                    0
funded_amnt                  0
funded_amnt_inv              0
term                         0
int_rate                     0
installment                  0
grade                        0
sub_grade                    0
emp_length                   0
home_ownership               0
annual_inc                   0
verification_status          0
issue_d                      0
loan_status                  0
purpose                      0
addr_state                   0
dti                          0
delinq_2yrs                  0
earliest_cr_line             0
inq_last_6mths               0
open_acc                     0
pub_rec                      0
revol_bal                    0
revol_util                  50
```

```
total_acc                    0
out_prncp                    0
out_prncp_inv                0
total_pymnt                  0
total_pymnt_inv              0
total_rec_prncp              0
total_rec_int                0
total_rec_late_fee           0
recoveries                   0
collection_recovery_fee      0
last_pymnt_d                71
last_pymnt_amnt              0
last_credit_pull_d           2
pub_rec_bankruptcies       697
dtype: int64
```

```
new_df['home_ownership'].value_counts()
```

```
RENT        18899
MORTGAGE    17659
OWN          3058
OTHER          98
NONE            3
Name: home_ownership, dtype: int64
```

```
new_df['loan_status'].value_counts()
```

```
Fully Paid     32950
Charged Off     5627
Current         1140
Name: loan_status, dtype: int64
```

```
new_df['purpose'].unique()
```

```
array(['credit_card', 'car', 'small_business', 'other', 'wedding',
       'debt_consolidation', 'home_improvement', 'major_purchase',
       'medical', 'moving', 'vacation', 'house', 'renewable_energy',
       'educational'], dtype=object)
```

```
new_df['addr_state'].unique()
```

```
array(['AZ', 'GA', 'IL', 'CA', 'OR', 'NC', 'TX', 'VA', 'MO', 'CT',
'UT',
       'FL', 'NY', 'PA', 'MN', 'NJ', 'KY', 'OH', 'SC', 'RI', 'LA',
'MA',
       'WA', 'WI', 'AL', 'CO', 'KS', 'NV', 'AK', 'MD', 'WV', 'VT',
'MI',
       'DC', 'SD', 'NH', 'AR', 'NM', 'MT', 'HI', 'WY', 'OK', 'DE',
'MS',
       'TN', 'IA', 'NE', 'ID', 'IN', 'ME'], dtype=object)
```

```
new_df['loan_amnt'].unique()
```

```
array([ 5000,  2500,  2400, 10000,  3000,  7000,  5600,  5375,  6500,
       12000,  9000,  1000,  3600,  6000,  9200, 20250, 21000, 15000,
        4000,  8500,  4375, 31825, 12400, 10800, 12500,  9600,  4400,
       14000, 11000, 25600, 16000,  7100, 13000, 17500, 17675,  8000,
        3500, 16425,  8200, 20975,  6400, 14400,  7250, 18000, 35000,
       11800,  4500, 10500, 15300, 20000,  6200,  7200,  9500, 18825,
       24000,  2100,  5500, 26800, 25000, 19750, 13650, 28000, 10625,
        8850,  6375, 11100,  4200,  8875, 13500, 21600,  8450, 13475,
       22000,  7325,  7750, 13350, 22475,  8400, 13250,  7350, 11500,
       29500,  2000, 11625, 15075,  5300,  8650,  7400, 24250, 26000,
        1500, 19600,  4225, 16500, 15600, 14125, 13200, 12300,  1400,
        3200, 11875,  1800, 23200,  4800,  7300, 10400,  6600, 30000,
        4475,  6300,  8250,  9875, 21500,  7800,  9750, 15550, 17000,
        7500,  5800,  8050,  5400,  4125,  9800, 15700,  9900,  6250,
       10200, 23000, 25975, 21250, 33425,  8125, 18800, 19200, 12875,
        2625, 11300,  4100, 18225, 18500, 16800,  2200, 14050, 16100,
       10525, 19775, 14500, 11700,  4150, 12375,  1700, 22250, 11200,
       22500, 15900,  3150, 18550,  8575,  7700, 24500, 22200, 21400,
        9400, 22400,  5825,  7650, 20675, 27050, 20500, 12800, 27575,
        7600, 29000,  9575, 14575,  7125, 10700, 10375,  3050, 27000,
       28625, 14100, 20050, 24925, 13600, 26400,  7150, 32000, 15500,
       17475,  2250, 17050,  3250, 22750,  1200,  5900, 12600,  6750,
       17250, 19075, 17200, 13225, 11775, 16400, 10075,  9350,  8075,
       15625, 20125,  8300,  2425,  6950,  5350,  5875,  9450, 19000,
       20400, 21650, 20300,  2300, 24575,  5850,  4750,  5275,  9175,
       34475, 10050, 19400, 18200,  8800, 34000, 19500,  5200, 11900,
       29100, 25850,  3300, 12200, 22575,  7175, 18250, 16750, 12950,
        6350, 14750,  6625,  6900, 18650,  9250, 22800, 27300, 12250,
        4350, 21200,  2700,  6025,  3825,  5325, 14150,  1600,  2800,
       18975,  2575,  5450,  3800,  2125, 14650, 11250, 31000,  6075,
        8475,  3625, 31300,  4250, 12650, 27600, 13150,  4300, 10275,
       23600,  7875, 14550,  9925, 15850,  1325,  6325, 29700, 15200,
       28100, 15250,  6800, 11325, 13975, 13800,  3100,  3975, 25450,
        3575, 33600, 23700, 28200,  6475, 27700, 17375, 15800, 17625,
       16675,  5250, 22950,  1950,  4650, 10250,  6100,  8325,  4850,
        9425, 12700, 25475, 14850, 14300, 33000,  5150, 21625,  3775,
       21575, 16250,  8375, 18725, 11125,  3525, 19800,  9300, 19125,
        5575,  1450, 12900, 10150, 20450, 23500, 16600,  1300,  6925,
       14675, 11550, 17400,  1100,  3400, 12775,  5050, 12100, 26375,
        6975, 26300,  3125, 23325, 11600,  5100, 10175, 18400, 30750,
       16550,  5650, 16450, 18950,  3650, 33950, 10125, 16775,  5700,
       20200, 10600,  3725, 19425, 25900, 23800,  4025,  2600,  8900,
       10900, 17600, 14825,  7925, 14950,  6700,  8600,  1925, 30500,
        4900, 15575,  3175, 14800, 32275,  5750, 14600, 25200,  6550,
       30400, 22900,  6850,  4600, 11425, 16950, 29850, 10675,  6650,
       10775, 17325, 27250,  3700,  6450, 20800, 13575, 29275,  4725,
       24800, 15750, 17100, 15875, 10925,  4950, 10575,  2850, 32875,
       21100, 11050, 20375,  9325,  9375,  7475, 22125, 27525, 25500,
       17750,  8675,  7450, 24625, 17900, 12075,  6725, 24400,  5225,
       14075, 17175,  9475,  9975, 20900, 12150, 17725, 15350,  4925,
```

```
 4550, 18750, 15125, 10950, 12475,  2750,  4625, 12175,  7575,
23525, 12350, 17950,  9525,  8975, 11975, 12850, 19850, 21850,
 4425, 32250,  2550, 11400, 21725, 23100, 13700,  9950, 21750,
13750, 12025, 23400, 14975, 19700, 27500,  3900, 14725, 17800,
 5175, 15025, 29550, 23850, 31500,  9100, 27400, 23675,  9825,
16200, 11650, 18875, 29175,  3950,  2050, 19950, 12750, 24375,
 2875, 25875, 16275, 10300, 17450,  3450,  1825, 13100, 23275,
 8700,  3675,  8150, 23975,  3350,  7075,  8625, 31800, 26200,
34675, 11025,  7850, 14175,  9150, 19925, 14275, 25400, 17825,
16875, 21800, 14475, 14225, 10225, 10650, 12725, 31400,  1550,
31700, 31200,  1875, 16300, 12550, 11725, 22600, 26500,  6225,
 4450,  3875, 13275, 34525, 31025,  6775, 19450,  2900,  2450,
27200, 21300,  4700,  7425, 19575, 31150, 19100, 30100, 24600,
32350,  1900, 29300,  2350, 15950, 13300,  2975, 28250,  8100,
28600,  6425,  4050, 23450, 32400, 13675, 21350,  9050,  2675,
 5025,  5950, 12625, 29800,  1750, 10825, 24700, 13125,  6125,
26850, 28800,  7275,  6825, 14775, 10975, 20950,  3850, 28500,
31325, 11750, 15825,  7525,  3550,  7950, 13400,  3375,  1250,
29600, 22350,  1850, 17850, 17875,  7550,  6175, 30800, 21125,
30225,  3750, 10025, 14350,  7775, 33500, 18900,  8025,  5125,
13775,  3075, 29900, 11525,  5550,  5975, 32500, 22100, 25300,
14700,  3325,  5075,  5625, 27175, 11575, 16325, 24200, 15050,
 5425, 17700, 12450, 19725, 19550,  3025, 22875, 23075, 15450,
10750,  4325,  3275,  8175, 20700,  1775,  4775,  8225,  4575,
15775, 19475, 14200, 21225, 17225, 12425,  7900, 14525,  2650,
 8275, 13325, 30600,  6275,  4075,  1625,  1275, 13075, 23750,
24650, 14250,  8825,  5775,  8350, 19150,  9725, 18575,  8725,
16050, 26250, 16075,  6150,  8750, 11075, 10875, 16350,  2275,
 3925, 11375,  4275, 18325,  9650,  2725, 10425,  6575,  2075,
13175,  9550, 12675, 15425, 18300, 18600,  5525, 10550, 22325,
15175, 12225, 12525, 28750, 15650, 11450, 23350,  1525, 31725,
13625, 32775, 20600,  8550, 15975,  9775, 13425,  1050,  2950,
12925, 29375, 12325,  9075,  1350, 21700, 15400,  4975, 11275,
 7725,  9225,  2325, 13725,  8775, 19250, 14900, 34800, 17300,
 9700,  2150, 10100, 10350,  2825, 17975,  1650, 15275,  7975,
 2925,  2525,  2225,  5725, 23425,  4875,  2475,  3425, 16700,
 2775, 13050, 34200,  5925, 26025, 16225,  9275, 11350, 21450,
10850,  7225,  1425,  5475, 19300,  7050, 24175, 12050,  1225,
13850, 32525, 17075,  1375,  1675, 18275,  9125, 33250, 16525,
11850, 22300,  2375,  7675,  8525, 31050,  4525,  7025, 14625,
13375,  4675, 25375, 24975, 12825, 18150, 18050,  9850, 14875,
17425, 16725, 13550,  9625, 15150, 19875,  1475, 22650, 17150,
 6875,  7375,  5675,  7625,  6525,  3225,  6675,  1075, 15675,
17275, 11475, 12975, 15325,  1125,  8950, 11675, 12275,  3475,
21425, 18125, 23050, 11175, 10450, 21825, 10475, 20150, 24750,
13900,  4175, 24100, 17925, 24150, 19975, 19900, 13950, 12125,
11225, 23475, 19650, 13450, 10725,  1150, 20475, 17525,   500,
  725, 23575,   700,   950, 19275,   900,   750, 17350,   800,
10325, 13025, 22550], dtype=int64)
```

```
new_df['issue_d'].value_counts()
```

```
Dec-11    2260
Nov-11    2223
Oct-11    2114
Sep-11    2063
Aug-11    1928
Jul-11    1870
Jun-11    1827
May-11    1689
Apr-11    1562
Mar-11    1443
Jan-11    1380
Feb-11    1297
Dec-10    1267
Oct-10    1132
Nov-10    1121
Jul-10    1119
Sep-10    1086
Aug-10    1078
Jun-10    1029
May-10     920
Apr-10     827
Mar-10     737
Feb-10     627
Nov-09     602
Dec-09     598
Jan-10     589
Oct-09     545
Sep-09     449
Aug-09     408
Jul-09     374
Jun-09     356
May-09     319
Apr-09     290
Mar-09     276
Feb-09     260
Jan-09     239
Mar-08     236
Dec-08     223
Nov-08     184
Feb-08     174
Jan-08     171
Apr-08     155
Oct-08      96
Dec-07      85
Jul-08      83
May-08      71
Aug-08      71
Jun-08      66
```

```
Oct-07      47
Nov-07      37
Aug-07      33
Sep-08      32
Jul-07      30
Sep-07      18
Jun-07       1
Name: issue_d, dtype: int64
```

```
new_df['verification_status'].value_counts()
```

```
Not Verified       16921
Verified           12809
Source Verified     9987
Name: verification_status, dtype: int64
```

Derived matrice

We will now derive some new columns based on our business understanding that will be helpful in our analysis.

```
## we will be calculating the salary loan  income ratio
new_df['loan_income_ratio']=new_df['loan_amnt']/new_df['annual_inc']

## we will be deriving the  month on which the loan was funded
def derive_month(value):
    return value.split('-')[0]
new_df['loan_issue_month']=new_df['issue_d'].apply(derive_month)

def derive_month(value):
    return value.split('-')[1]
new_df['loan_issue_year']=new_df['issue_d'].apply(derive_month)
```

we will be creating the bins for the loan amount, interest rate and the annual income

```
## creating the derived matrics for the loan amount
bins = [0,5000, 10000,20000,25000,30000,40000]
slot = ['0-5000', '5000-10000', '10000-15000', '15000-20000', '20000-
25000','25000 and above']

new_df['loan_amount_range'] = pd.cut(new_df['loan_amnt'],
bins,labels=slot,ordered=False)

bins = [0, 7.5, 10, 12.5, 15,20]
slot = ['0-7.5', '7.5-10', '10-12.5', '12.5-15', '15 and above']
new_df['int_rate_range'] = pd.cut(new_df['int_rate'], bins,
labels=slot)

bins = [0, 25000, 50000, 75000, 100000,1000000]
slot = ['0-25000', '25000-50000', '50000-75000', '75000-100000',
'100000 and above']
new_df['annual_inc_range'] = pd.cut(new_df['annual_inc'], bins,
labels=slot)
```

```
new_df.head()

   loan_amnt  funded_amnt  funded_amnt_inv  term  int_rate
installment grade  \
0       5000         5000           4975.0    36     10.65
162.87     B
1       2500         2500           2500.0    60     15.27
59.83     C
2       2400         2400           2400.0    36     15.96
84.33     C
3      10000        10000          10000.0    36     13.49
339.31     C
4       3000         3000           3000.0    60     12.69
67.79     B

  sub_grade emp_length home_ownership  ...  last_pymnt_d
last_pymnt_amnt  \
0        B2  10+ years           RENT  ...        Jan-15
171.62
1        C4   < 1 year           RENT  ...        Apr-13
119.66
2        C5  10+ years           RENT  ...        Jun-14
649.91
3        C1  10+ years           RENT  ...        Jan-15
357.48
4        B5     1 year           RENT  ...        May-16
67.79

  last_credit_pull_d pub_rec_bankruptcies loan_income_ratio
loan_issue_month  \
0             May-16                  0.0          0.208333
Dec
1             Sep-13                  0.0          0.083333
Dec
2             May-16                  0.0          0.195886
Dec
3             Apr-16                  0.0          0.203252
Dec
4             May-16                  0.0          0.037500
Dec

   loan_issue_year  loan_amount_range int_rate_range  annual_inc_range

0               11             0-5000        10-12.5           0-25000

1               11             0-5000   15 and above       25000-50000

2               11             0-5000   15 and above           0-25000

3               11         5000-10000        12.5-15       25000-50000
```

```
4                11              0-5000        12.5-15      75000-100000


[5 rows x 44 columns]
```

```python
## now dropping the unused columns
unused_columns=['int_rate','issue_d']
new_df.drop(columns=unused_columns,axis=1,inplace=True)

new_df.head()
```

```
   loan_amnt  funded_amnt  funded_amnt_inv  term  installment grade 
sub_grade  \
0       5000         5000           4975.0    36       162.87     B   
B2   
1       2500         2500           2500.0    60        59.83     C   
C4   
2       2400         2400           2400.0    36        84.33     C   
C5   
3      10000        10000          10000.0    36       339.31     C   
C1   
4       3000         3000           3000.0    60        67.79     B   
B5   

  emp_length home_ownership  annual_inc  ... last_pymnt_d 
last_pymnt_amnt  \
0  10+ years           RENT     24000.0  ...       Jan-15          
171.62   
1   < 1 year           RENT     30000.0  ...       Apr-13          
119.66   
2  10+ years           RENT     12252.0  ...       Jun-14          
649.91   
3  10+ years           RENT     49200.0  ...       Jan-15          
357.48   
4     1 year           RENT     80000.0  ...       May-16           
67.79   

  last_credit_pull_d pub_rec_bankruptcies  loan_income_ratio  \
0             May-16                  0.0           0.208333   
1             Sep-13                  0.0           0.083333   
2             May-16                  0.0           0.195886   
3             Apr-16                  0.0           0.203252   
4             May-16                  0.0           0.037500   

   loan_issue_month loan_issue_year  loan_amount_range  int_rate_range 
\
0               Dec              11             0-5000         10-12.5   

1               Dec              11             0-5000    15 and above   
```

```
2                Dec                11              0-5000    15 and above

3                Dec                11          5000-10000         12.5-15

4                Dec                11              0-5000         12.5-15


   annual_inc_range
0           0-25000
1       25000-50000
2           0-25000
3       25000-50000
4      75000-100000

[5 rows x 42 columns]
```

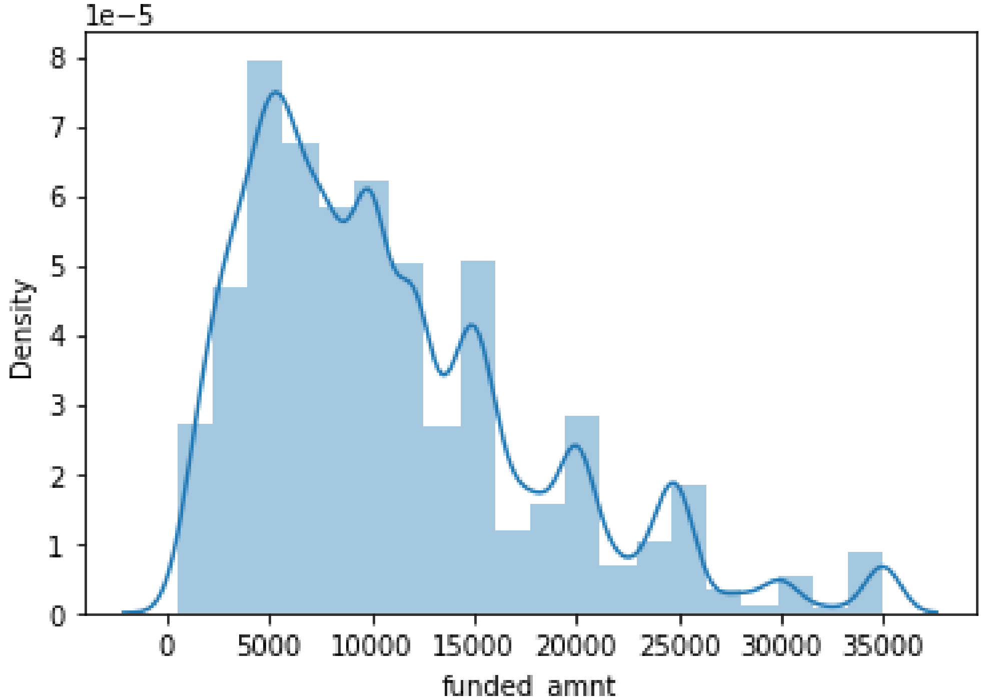
```
new_df.columns
```

```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term',
'installment',
       'grade', 'sub_grade', 'emp_length', 'home_ownership',
'annual_inc',
       'verification_status', 'loan_status', 'purpose', 'addr_state',
'dti',
       'delinq_2yrs', 'earliest_cr_line', 'inq_last_6mths',
'open_acc',
       'pub_rec', 'revol_bal', 'revol_util', 'total_acc', 'out_prncp',
       'out_prncp_inv', 'total_pymnt', 'total_pymnt_inv',
'total_rec_prncp',
       'total_rec_int', 'total_rec_late_fee', 'recoveries',
       'collection_recovery_fee', 'last_pymnt_d', 'last_pymnt_amnt',
       'last_credit_pull_d', 'pub_rec_bankruptcies',
'loan_income_ratio',
       'loan_issue_month', 'loan_issue_year', 'loan_amount_range',
       'int_rate_range', 'annual_inc_range'],
      dtype='object')
```

univarient analysis -

for the univarient analysis we will be doing the plot representation for the continous and the categorical varibales.

```
sns.distplot(new_df['funded_amnt'],bins=20)
plt.show()
```



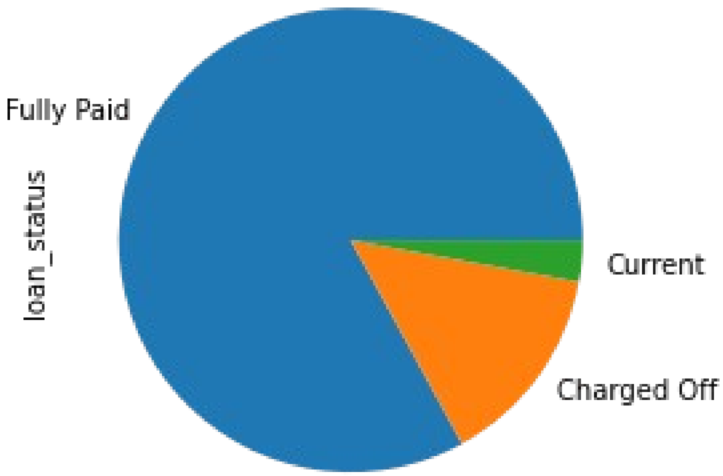

```
new_df['funded_amnt'].describe()
```

```
count    39717.000000
mean     10947.713196
std       7187.238670
min        500.000000
25%       5400.000000
50%       9600.000000
75%      15000.000000
max      35000.000000
Name: funded_amnt, dtype: float64
```

from the above distribution plot we can see that The total amount committed to that loan at that point in time.is more between 5000 to 10000

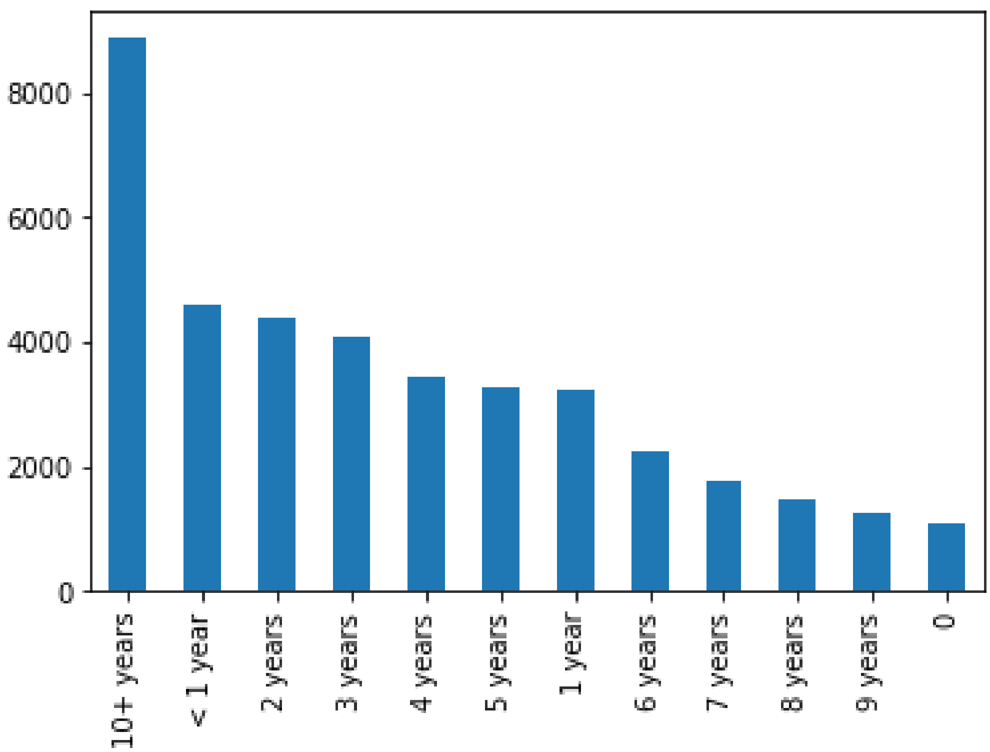
```
##  plotting the loan status for the analysis how many people have actually paid the loadn
new_df['loan_status'].value_counts().plot.pie()
```

```
<AxesSubplot:ylabel='loan_status'>
```



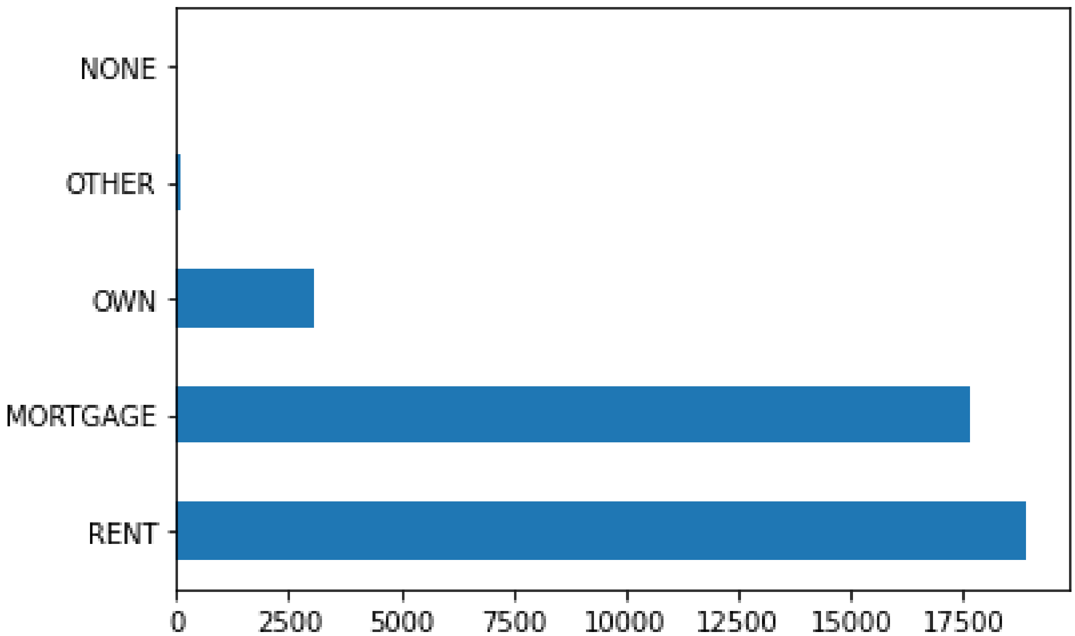
we can see that the most of the people have fully paid the loan and very less are going on with current loan status!!

```
## plotting the bar chart for the  employeee length
new_df['emp_length'].value_counts().plot.bar()
plt.show()
```



we can see that the most of the people who are having the employeee experinece more than 10 years are havingor needing the loan!!!

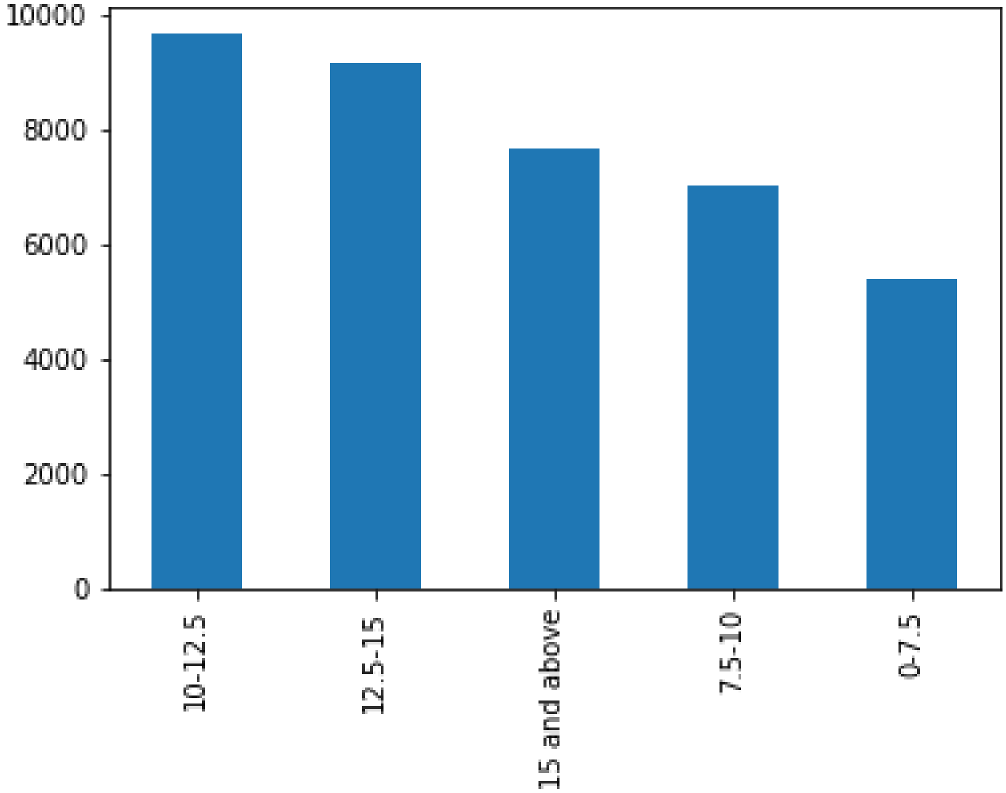
```
new_df['home_ownership'].value_counts().plot.barh()
plt.show()
```



we can see from the graph that the people mostly having the rent home ownership which might be the loan taking reasons !!

```
new_df['int_rate_range'].value_counts().plot.bar()
```

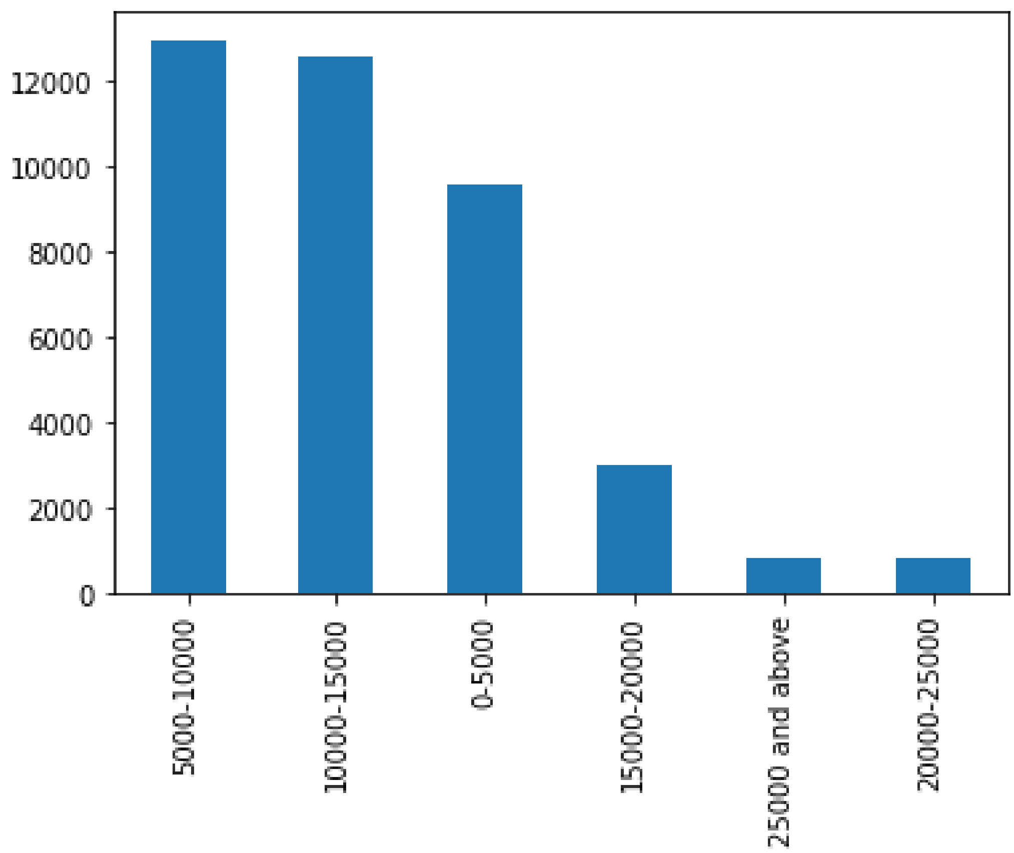
```
<AxesSubplot:>
```



people are getting the loan mostly between the interest rate 10-12.5 percent!!

```
new_df['loan_amount_range'].value_counts().plot.bar()
```

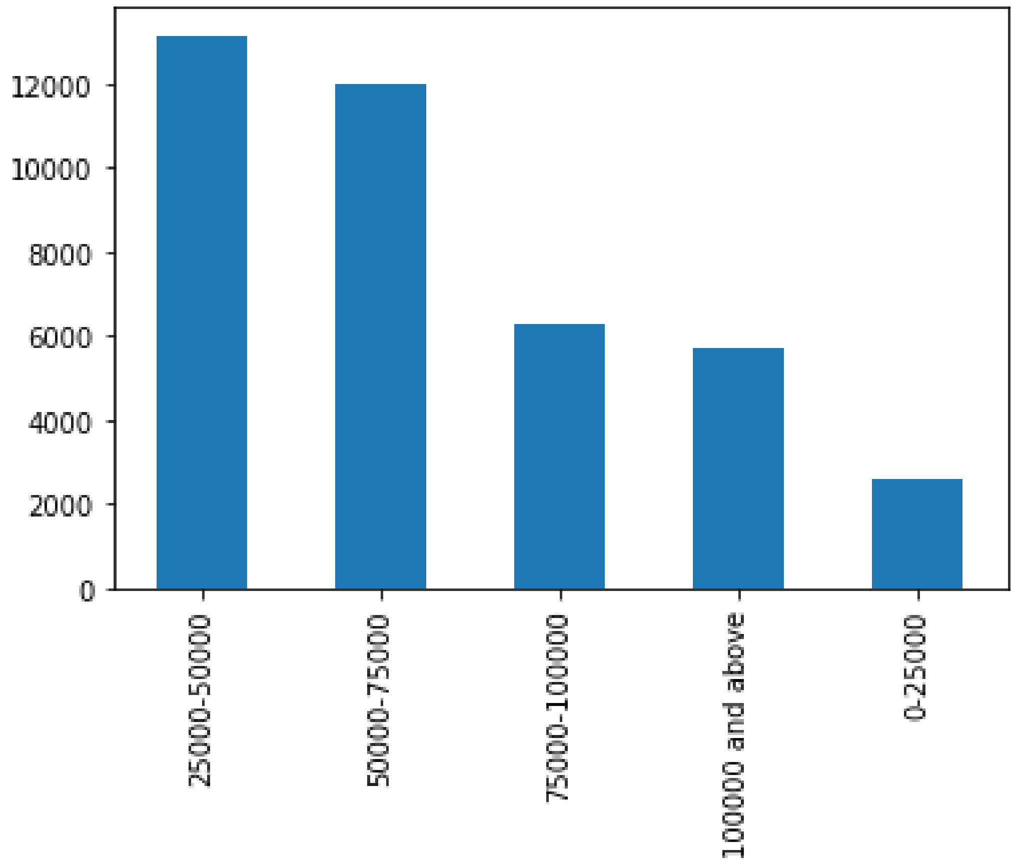
```
<AxesSubplot:>
```



most of the people are taking the loan mostly in the range of 5k to 10k and very less number of people are taking it above 25k!!

```
new_df['annual_inc_range'].value_counts().plot.bar()
```

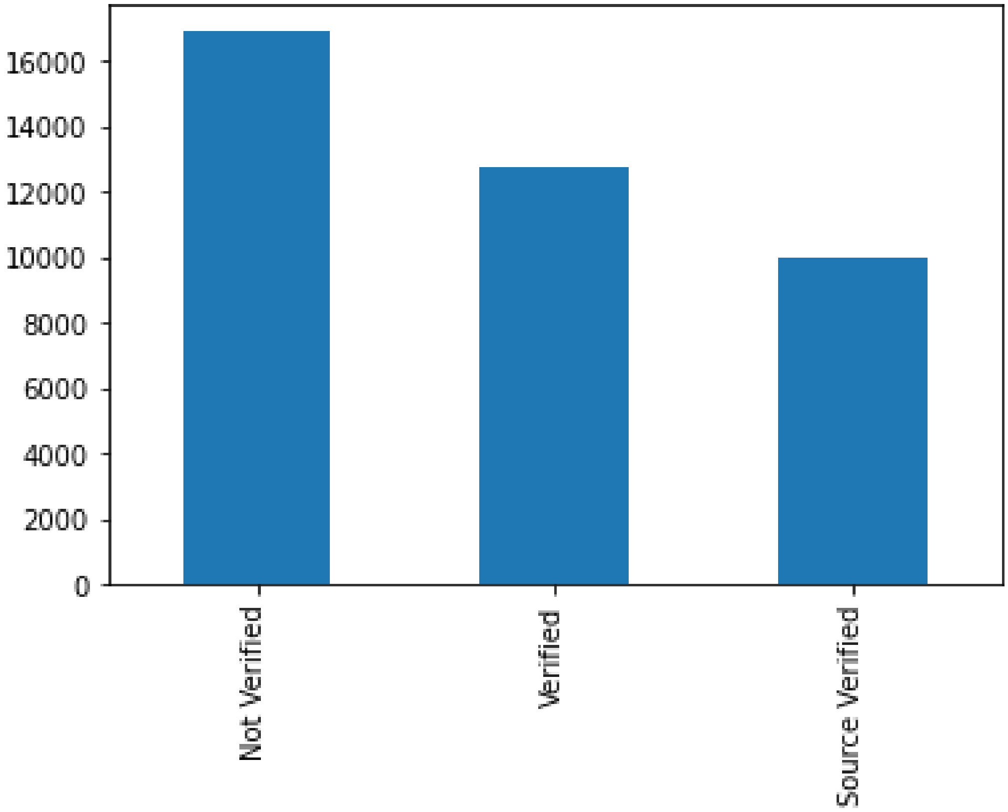
```
<AxesSubplot:>
```



So we can clearly see that most of the people are having the salary in the range 25k and 50k and very least is below 25k!!

```
new_df['verification_status'].value_counts().plot.bar()
```

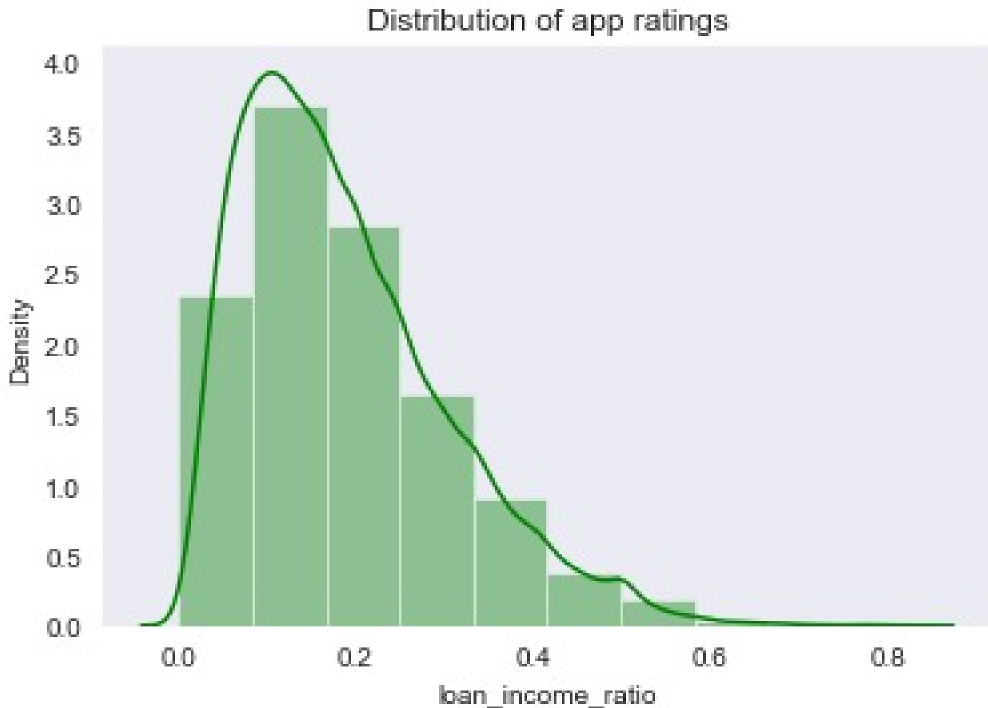
```
<AxesSubplot:>
```



most of the people's income are not verified by the LC

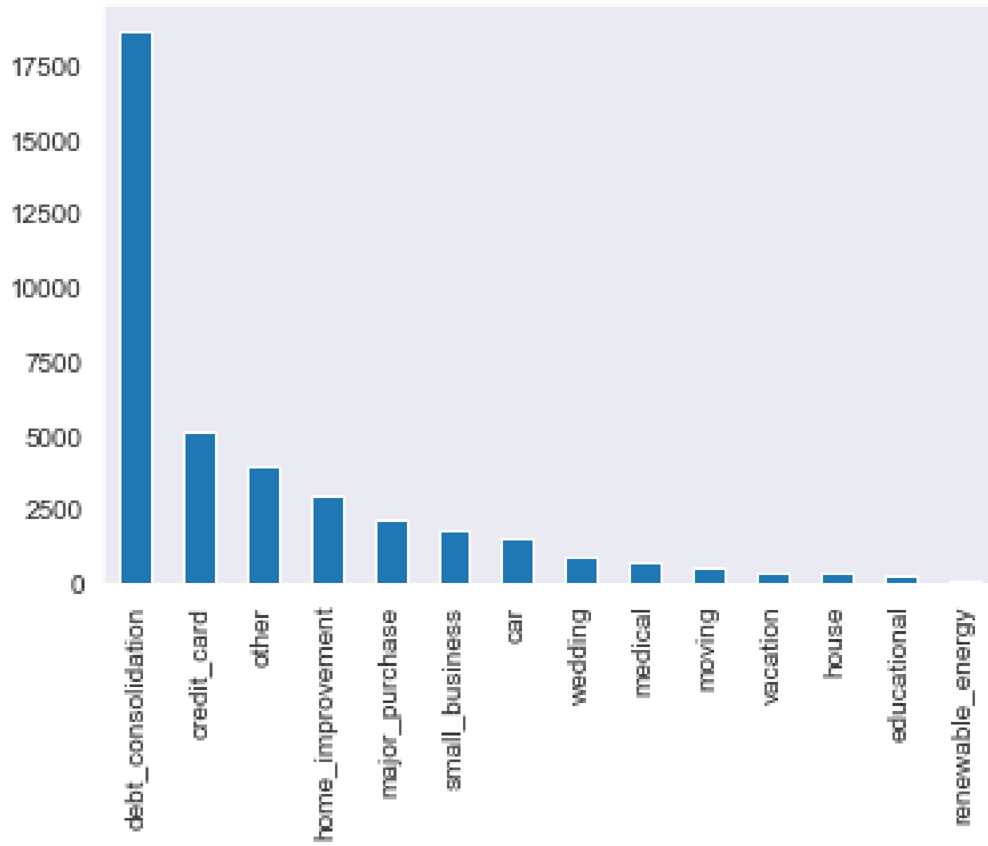
```
## let us see what is the income loan ration people fall into

sns.set_style("dark")
sns.distplot(new_df['loan_income_ratio'], bins=10, color="g")
plt.title("Distribution of app ratings", fontsize=12)
plt.show()
```



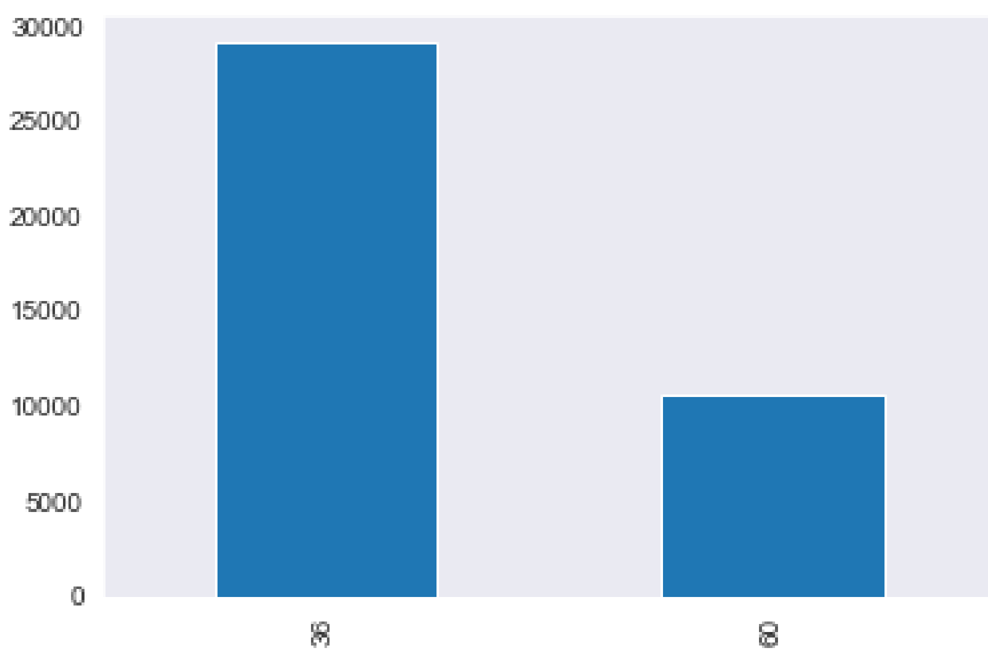
most of the people are falling into the range of 0 to 0.2 specially 0.1 is the maximum people ratio of loan is to income!!!

```
new_df['purpose'].value_counts().plot.bar()
plt.show()
```




```
new_df['term'].value_counts().plot.bar()
```

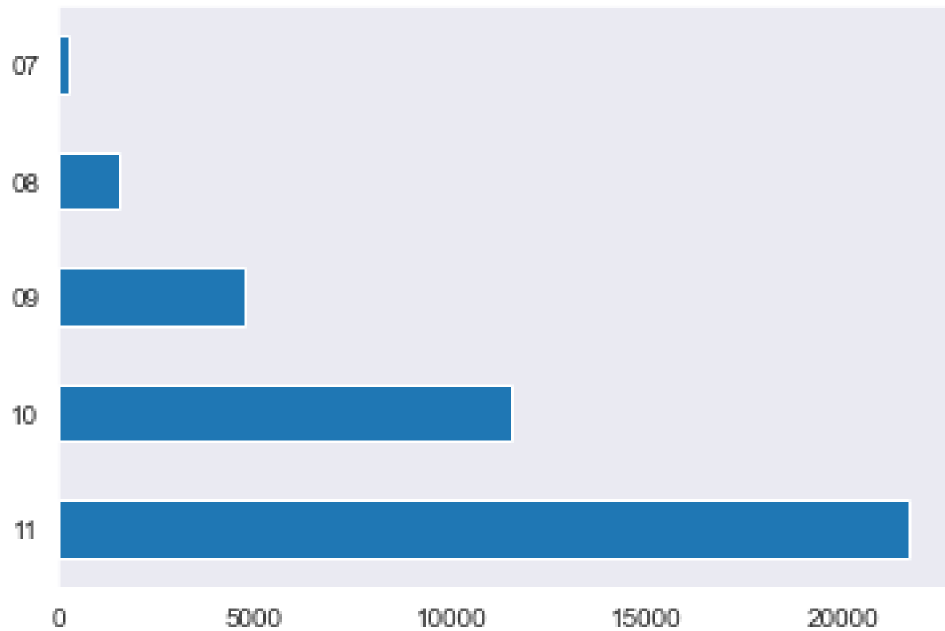
```
<AxesSubplot:>
```



we can observe that most of the people are taking the loan or replaying the loan for the 36 months!!!

```
## we will look into the year the loan is increaing
new_df['loan_issue_year'].value_counts().plot.barh()
```

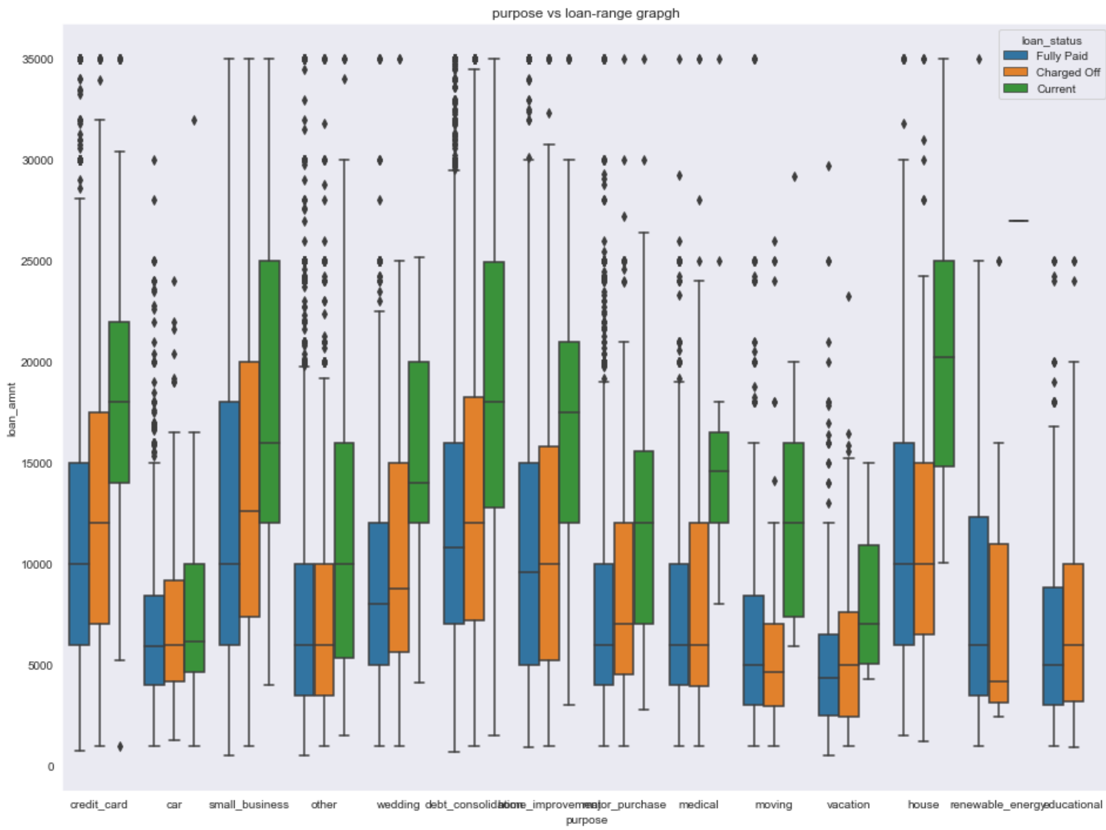
```
<AxesSubplot:>
```



most of the loan is taken in the 2011 and it gradually increased from 2007 to 2011!!!

Bivarient Analysis

```
## we will look what is the most purplose for the loan requirement
plt.figure(figsize=(16,12))
sns.boxplot(data=new_df,x='purpose',y='loan_amnt',hue='loan_status')
plt.title('purpose vs loan-range grapgh')
plt.show()
```




```
## creating a correlation matrix now and we will see which features
are most corelated
rs = np.random.RandomState(0)

corr = new_df.corr()
corr.style.background_gradient(cmap='coolwarm')
```

```
<pandas.io.formats.style.Styler at 0x289a6c90ee0>
```