In [1]:

```
import warnings
import pandas as pd
import matplotlib.pyplot as plt
import pygwalker as pyg
import seaborn as sns
import numpy as np
```

In [2]:

```
df = pd.read_csv('loan.csv')
warnings.filterwarnings('ignore')
```

```
C:\Users\nitin\AppData\Local\Temp\ipykernel_22264\1345965906.py:1: DtypeWa
rning: Columns (47) have mixed types. Specify dtype option on import or se
t low_memory=False.
  df = pd.read_csv('loan.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installn
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	16
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	5
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	8
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	33
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	6

5 rows × 111 columns

In [4]:

```
df.columns
```

Out[4]:

```
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)
```

In [5]:

```
df.describe()
```

Out[5]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	installm
count	3.971700e+04	3.971700e+04	39717.000000	39717.000000	39717.000000	39717.000(
mean	6.831319e+05	8.504636e+05	11219.443815	10947.713196	10397.448868	324.561(
std	2.106941e+05	2.656783e+05	7456.670694	7187.238670	7128.450439	208.874(
min	5.473400e+04	7.069900e+04	500.000000	500.000000	0.000000	15.690(
25%	5.162210e+05	6.667800e+05	5500.000000	5400.000000	5000.000000	167.020(
50%	6.656650e+05	8.508120e+05	10000.000000	9600.000000	8975.000000	280.220(
75%	8.377550e+05	1.047339e+06	15000.000000	15000.000000	14400.000000	430.780(
max	1.077501e+06	1.314167e+06	35000.000000	35000.000000	35000.000000	1305.190(

8 rows × 87 columns

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
id                                0
member_id                         0
loan_amnt                         0
funded_amnt                       0
funded_amnt_inv                   0
                                 ...
tax_liens                        39
tot_hi_cred_lim                39717
total_bal_ex_mort              39717
total_bc_limit                 39717
total_il_high_credit_limit     39717
Length: 111, dtype: int64
```

In [ ]:

```

```

In [7]:

```
df.shape
```

Out[7]:

```
(39717, 111)
```

In [8]:

```
df.shape[0]
```

Out[8]:

```
39717
```

In [9]:

```
df_null_percentage = df.apply(lambda x: round(x.isnull().mean()* 100, 2))
df_null_percentage[df_null_percentage>0.0]
```

Out[9]:

```
emp_title                      6.19
emp_length                     2.71
desc                          32.58
title                          0.03
mths_since_last_delinq        64.66
                              ...
tax_liens                      0.10
tot_hi_cred_lim              100.00
total_bal_ex_mort            100.00
total_bc_limit               100.00
total_il_high_credit_limit   100.00
Length: 68, dtype: float64
```

In [10]:

```
df_null_above_thresold = df_null_percentage[df_null_percentage>25.0]
null_above_thresold_columns = df_null_above_thresold.index
f"{len(null_above_thresold_columns)} columns out of {len(df.columns)} columns to be drop
```

Out[10]:

```
'58 columns out of 111 columns to be dropped from the dataframe df'
```

In [11]:

```
df.drop(null_above_thresold_columns, axis=1, inplace= True)
```

In [12]:

```
df
```

Out[12]:

id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	gı
77501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	
77430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	
77175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	
76863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	
75358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	
...	...	...	...	...	...	...	...	
92187	92174	2500	2500	1075.0	36 months	8.07%	78.42	
90665	90607	8500	8500	875.0	36 months	10.28%	275.38	
90395	90390	5000	5000	1325.0	36 months	8.07%	156.84	
90376	89243	5000	5000	650.0	36 months	7.43%	155.38	
87023	86999	7500	7500	800.0	36 months	13.75%	255.43	

s × 53 columns

In [13]:

```
#lets analyse categorical data
for column in df.columns:
    if df[column].dtype == 'object':
        print(column)
```

```
term
int_rate
grade
sub_grade
emp_title
emp_length
home_ownership
verification_status
issue_d
loan_status
pymnt_plan
url
purpose
title
zip_code
addr_state
earliest_cr_line
revol_util
initial_list_status
last_pymnt_d
last_credit_pull_d
application_type
```

In [14]:

```
df['term']
```

Out[14]:

```
0         36 months
1         60 months
2         36 months
3         36 months
4         60 months
           ...
39712     36 months
39713     36 months
39714     36 months
39715     36 months
39716     36 months
Name: term, Length: 39717, dtype: object
```

In [15]:

```
df.term.unique()
```

Out[15]:

```
array([' 36 months', ' 60 months'], dtype=object)
```

In [16]:

```
df.nunique().sort_values(ascending=True)
```

Out[16]:

```
tax_liens                        1
delinq_amnt                      1
chargeoff_within_12_mths         1
acc_now_delinq                   1
application_type                 1
policy_code                      1
collections_12_mths_ex_med       1
initial_list_status              1
pymnt_plan                       1
term                             2
pub_rec_bankruptcies             3
verification_status              3
loan_status                      3
pub_rec                          5
home_ownership                   5
grade                            7
inq_last_6mths                   9
delinq_2yrs                     11
emp_length                      11
purpose                         14
sub_grade                       35
open_acc                        40
addr_state                      50
issue_d                         55
total_acc                       82
last_pymnt_d                   101
last_credit_pull_d             106
int_rate                       371
earliest_cr_line               526
zip_code                       823
loan_amnt                      885
funded_amnt                   1041
revol_util                    1089
out_prncp                     1137
out_prncp_inv                 1138
total_rec_late_fee            1356
collection_recovery_fee       2616
dti                           2868
recoveries                    4040
annual_inc                    5318
total_rec_prncp               7976
funded_amnt_inv               8205
installment                  15383
title                        19615
revol_bal                    21711
emp_title                    28820
last_pymnt_amnt              34930
total_rec_int                35148
total_pymnt_inv              37518
total_pymnt                  37850
url                          39717
member_id                    39717
id                           39717
dtype: int64
```

In [17]:

```
columns_to_sort = df.columns[df.nunique() == 1]

# Sort the columns with a unique value of 1
df.drop(columns_to_sort, axis=1, inplace=True)
```

In [18]:

```
df
```

Out[18]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	ins
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	
...	...	...	...	...	...	...	...	
39712	92187	92174	2500	2500	1075.0	36 months	8.07%	
39713	90665	90607	8500	8500	875.0	36 months	10.28%	
39714	90395	90390	5000	5000	1325.0	36 months	8.07%	
39715	90376	89243	5000	5000	650.0	36 months	7.43%	
39716	87023	86999	7500	7500	800.0	36 months	13.75%	

39717 rows × 44 columns

In [19]:

```
df.drop(['member_id', 'url', 'title', 'emp_title'], axis=1, inplace=True)
```

There are few columns which are mainly applicable to a person who has either taken loan or completed the loan. Since the main focus is study those factors which are significant and important for the loan approval, so we will remove these below columns.

- total_rec_int : Interest received to date
- total_rec_prncp : Principal received to date
- total_rec_late_fee : Late fees received to date
- recoveries : post charge off gross recovery
- collection_recovery_fee : post charge off collection fee

- last_credit_pull_d : The most recent month LC pulled credit for this loan
- last_pymnt_d : Last month payment was received
- out_prncp : Remaining outstanding principal for total amount funded
- out_prncp_inv : Remaining outstanding principal for portion of total amount funded by investors

In [20]:

```
# Drop columns which are not important from the loan lending point of view.

columns_to_drop = ['total_rec_int', 'total_rec_prncp', 'total_rec_late_fee', 'recoveries
                   'collection_recovery_fee','last_credit_pull_d', 'last_pymnt_d',  'out
df.drop(columns_to_drop, axis=1, inplace=True)
```

In [21]:

```
df
```

Out[21]:

	id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	gr
0	1077501	5000	5000	4975.0	36 months	10.65%	162.87	
1	1077430	2500	2500	2500.0	60 months	15.27%	59.83	
2	1077175	2400	2400	2400.0	36 months	15.96%	84.33	
3	1076863	10000	10000	10000.0	36 months	13.49%	339.31	
4	1075358	3000	3000	3000.0	60 months	12.69%	67.79	
...	...	...	...	...	...	...	...	
39712	92187	2500	2500	1075.0	36 months	8.07%	78.42	
39713	90665	8500	8500	875.0	36 months	10.28%	275.38	
39714	90395	5000	5000	1325.0	36 months	8.07%	156.84	
39715	90376	5000	5000	650.0	36 months	7.43%	155.38	
39716	87023	7500	7500	800.0	36 months	13.75%	255.43	

39717 rows × 31 columns

Explore percentage of null values in the final dataframe

In [22]:

```
# Calculate percentage null values.

df_null_percentage = df.apply(lambda x: round(x.isnull().mean()* 100, 2))
df_null_percentage[df_null_percentage>0.0]
```

Out[22]:

```
emp_length              2.71
revol_util              0.13
pub_rec_bankruptcies    1.75
dtype: float64
```

In [23]:

```
# Since, this percentage is very small, so we will remove these null values in the row f

# Dropping null values.

df.dropna(subset=['emp_length', 'revol_util', 'pub_rec_bankruptcies'], inplace=True)

# Finding percentage of null or missing values (TO verify!).

null_perc_after_na_drop = round(100*(df.isnull().sum()/len(df.index)), 2)
null_perc_after_na_drop[null_perc_after_na_drop > 0 ]
```

Out[23]:

```
Series([], dtype: float64)
```

Hence, the dataframe is now treated with nan's and irrelavent columns.

In [24]:

```
# dataframe info after above steps:

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37898 entries, 0 to 39680
Data columns (total 31 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    37898 non-null  int64  
 1   loan_amnt             37898 non-null  int64  
 2   funded_amnt           37898 non-null  int64  
 3   funded_amnt_inv       37898 non-null  float64
 4   term                  37898 non-null  object 
 5   int_rate              37898 non-null  object 
 6   installment           37898 non-null  float64
 7   grade                 37898 non-null  object 
 8   sub_grade             37898 non-null  object 
 9   emp_length            37898 non-null  object 
 10  home_ownership        37898 non-null  object 
 11  annual_inc            37898 non-null  float64
 12  verification_status   37898 non-null  object 
 13  issue_d               37898 non-null  object 
 14  loan_status           37898 non-null  object 
 15  purpose               37898 non-null  object 
 16  zip_code              37898 non-null  object 
 17  addr_state            37898 non-null  object 
 18  dti                   37898 non-null  float64
 19  delinq_2yrs           37898 non-null  int64  
 20  earliest_cr_line      37898 non-null  object 
 21  inq_last_6mths        37898 non-null  int64  
 22  open_acc              37898 non-null  int64  
 23  pub_rec               37898 non-null  int64  
 24  revol_bal             37898 non-null  int64  
 25  revol_util            37898 non-null  object 
 26  total_acc             37898 non-null  int64  
 27  total_pymnt           37898 non-null  float64
 28  total_pymnt_inv       37898 non-null  float64
 29  last_pymnt_amnt       37898 non-null  float64
 30  pub_rec_bankruptcies  37898 non-null  float64
dtypes: float64(8), int64(9), object(14)
memory usage: 9.3+ MB
```

Let us treat now some important object datatype

In [25]:

```
# Generally, employment length may play an important role to decide the loan application

df.emp_length.unique()
```

Out[25]:

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

In [26]:

```
# Since, we don't have numeric values to this columns, so we will extract first number f

# Also, we have employees have less than one year of experience (< 1 year). Let's first

emp_number = (f"Application with < 1 year of empoyement are : {len(df[df['emp_length'] =
              f"Application with 1 year of empoyement are : {len(df[df['emp_length'] ==

print(emp_number)
```

```
Application with < 1 year of empoyement are : 4404
Application with 1 year of empoyement are : 3142
```

In [27]:

```
# Now, lets extract years from emp_length for the rest of rows

df['emp_length']=df['emp_length'].str.extract('(\d+)')
df.head()
```

Out[27]:

	id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade
0	1077501	5000	5000	4975.0	36 months	10.65%	162.87	B
1	1077430	2500	2500	2500.0	60 months	15.27%	59.83	C
2	1077175	2400	2400	2400.0	36 months	15.96%	84.33	C
3	1076863	10000	10000	10000.0	36 months	13.49%	339.31	C
4	1075358	3000	3000	3000.0	60 months	12.69%	67.79	B

5 rows × 31 columns

In [28]:

```
# It may be relavent to treat such employee with either 1 or 0 year of exp.
# But since, we have such number greater than emp of 1 yr of exp, so this may create som
# So, we will replace <1 yr of exp to 0 year of exp for ease of our study.

# Replace, '< 1 year' with 0:

df['emp_length'].replace('< 1 year', 0, inplace=True)
```

In [29]:

```
# Let's convert now int_rate and

# Lets remove % symbol from intrest rate column so that it can be used in calculations
df['int_rate'] = df['int_rate'].str.rstrip('%')

# Lets remove % symbol from revol_util column so that it can be used in calculations
df['revol_util'] = df['revol_util'].str.rstrip('%')

df.head()
```

Out[29]:

	id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade
0	1077501	5000	5000	4975.0	36 months	10.65	162.87	B
1	1077430	2500	2500	2500.0	60 months	15.27	59.83	C
2	1077175	2400	2400	2400.0	36 months	15.96	84.33	C
3	1076863	10000	10000	10000.0	36 months	13.49	339.31	C
4	1075358	3000	3000	3000.0	60 months	12.69	67.79	B

5 rows × 31 columns

In [30]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37898 entries, 0 to 39680
Data columns (total 31 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    37898 non-null  int64  
 1   loan_amnt             37898 non-null  int64  
 2   funded_amnt           37898 non-null  int64  
 3   funded_amnt_inv       37898 non-null  float64
 4   term                  37898 non-null  object 
 5   int_rate              37898 non-null  object 
 6   installment           37898 non-null  float64
 7   grade                 37898 non-null  object 
 8   sub_grade             37898 non-null  object 
 9   emp_length            37898 non-null  object 
 10  home_ownership        37898 non-null  object 
 11  annual_inc            37898 non-null  float64
 12  verification_status   37898 non-null  object 
 13  issue_d               37898 non-null  object 
 14  loan_status           37898 non-null  object 
 15  purpose               37898 non-null  object 
 16  zip_code              37898 non-null  object 
 17  addr_state            37898 non-null  object 
 18  dti                   37898 non-null  float64
 19  delinq_2yrs           37898 non-null  int64  
 20  earliest_cr_line      37898 non-null  object 
 21  inq_last_6mths        37898 non-null  int64  
 22  open_acc              37898 non-null  int64  
 23  pub_rec               37898 non-null  int64  
 24  revol_bal             37898 non-null  int64  
 25  revol_util            37898 non-null  object 
 26  total_acc             37898 non-null  int64  
 27  total_pymnt           37898 non-null  float64
 28  total_pymnt_inv       37898 non-null  float64
 29  last_pymnt_amnt       37898 non-null  float64
 30  pub_rec_bankruptcies  37898 non-null  float64
dtypes: float64(8), int64(9), object(14)
memory usage: 9.3+ MB
```

In [31]:

```
# Now, we will convert int_rate, emp_length and revo_util into numeric type from object (

df['int_rate'] = df['int_rate'].apply(pd.to_numeric)
df['revol_util'] = df['revol_util'].apply(pd.to_numeric)
df['emp_length'] = df['emp_length'].apply(pd.to_numeric)
```

In [32]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37898 entries, 0 to 39680
Data columns (total 31 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    37898 non-null  int64  
 1   loan_amnt             37898 non-null  int64  
 2   funded_amnt           37898 non-null  int64  
 3   funded_amnt_inv       37898 non-null  float64
 4   term                  37898 non-null  object 
 5   int_rate              37898 non-null  float64
 6   installment           37898 non-null  float64
 7   grade                 37898 non-null  object 
 8   sub_grade             37898 non-null  object 
 9   emp_length            37898 non-null  int64  
 10  home_ownership        37898 non-null  object 
 11  annual_inc            37898 non-null  float64
 12  verification_status   37898 non-null  object 
 13  issue_d               37898 non-null  object 
 14  loan_status           37898 non-null  object 
 15  purpose               37898 non-null  object 
 16  zip_code              37898 non-null  object 
 17  addr_state            37898 non-null  object 
 18  dti                   37898 non-null  float64
 19  delinq_2yrs           37898 non-null  int64  
 20  earliest_cr_line      37898 non-null  object 
 21  inq_last_6mths        37898 non-null  int64  
 22  open_acc              37898 non-null  int64  
 23  pub_rec               37898 non-null  int64  
 24  revol_bal             37898 non-null  int64  
 25  revol_util            37898 non-null  float64
 26  total_acc             37898 non-null  int64  
 27  total_pymnt           37898 non-null  float64
 28  total_pymnt_inv       37898 non-null  float64
 29  last_pymnt_amnt       37898 non-null  float64
 30  pub_rec_bankruptcies  37898 non-null  float64
dtypes: float64(10), int64(10), object(11)
memory usage: 9.3+ MB
```

Let's convert issue_d into month and year to have a deep dive into loan issue

In [33]:

```
df['issue_d'] = pd.to_datetime(df['issue_d'], format='%b-%y')
df['issue_year'] = df['issue_d'].dt.year
df['issue_month'] = df['issue_d'].dt.month
df.head()
```

Out[33]:

	id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade
0	1077501	5000	5000	4975.0	36 months	10.65	162.87	B
1	1077430	2500	2500	2500.0	60 months	15.27	59.83	C
2	1077175	2400	2400	2400.0	36 months	15.96	84.33	C
3	1076863	10000	10000	10000.0	36 months	13.49	339.31	C
4	1075358	3000	3000	3000.0	60 months	12.69	67.79	B

5 rows × 33 columns

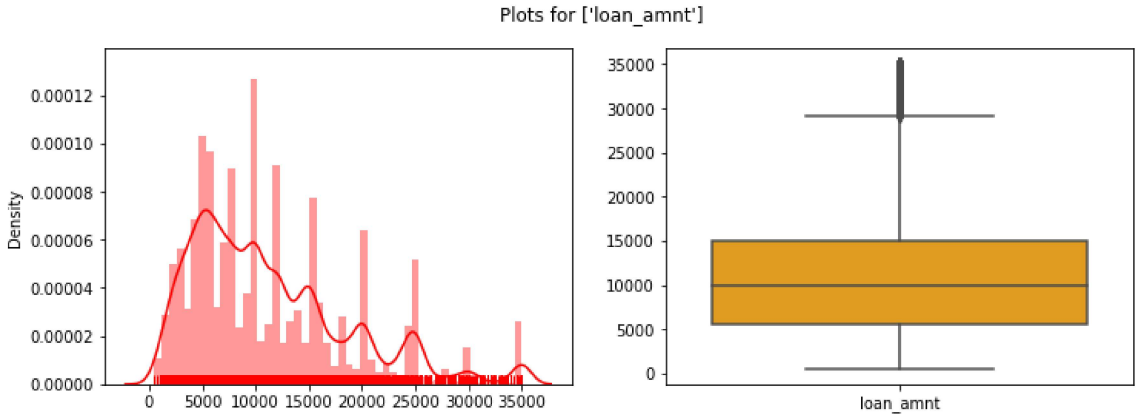
Univariate analysis

In [34]:

```
# create a funtion which plot box plots
def draw_boxplot(dataframe, variable):
    plt.figure(figsize=(12,4))
    plt.subplot(1,2,1)
    sns.distplot(a=dataframe[variable], rug=True, color='red')
    plt.subplot(1,2,2)
    sns.boxplot(data=dataframe[variable], color='orange')
    plt.suptitle(f'Plots for {variable}')
    plt.show()
```

In [35]:

```
# loan amount box plot

columns_to_be_plotted = ['loan_amnt']
draw_boxplot(df, columns_to_be_plotted )
df['loan_amnt'].describe()
```



Out[35]:

```
count    37898.000000
mean     11324.333738
std       7478.589882
min        500.000000
25%       5600.000000
50%      10000.000000
75%      15000.000000
max      35000.000000
Name: loan_amnt, dtype: float64
```

In [36]:

```
# since, we see outliers, so we will treat them

df['loan_amnt'].describe(percentiles=[0.05,0.1,0.25,0.5,0.75,0.9,0.95,0.99])
```

Out[36]:

```
count    37898.000000
mean     11324.333738
std       7478.589882
min        500.000000
5%        2400.000000
10%       3250.000000
25%       5600.000000
50%      10000.000000
75%      15000.000000
90%      22400.000000
95%      25000.000000
99%      35000.000000
max      35000.000000
Name: loan_amnt, dtype: float64
```

Observation

In [37]:

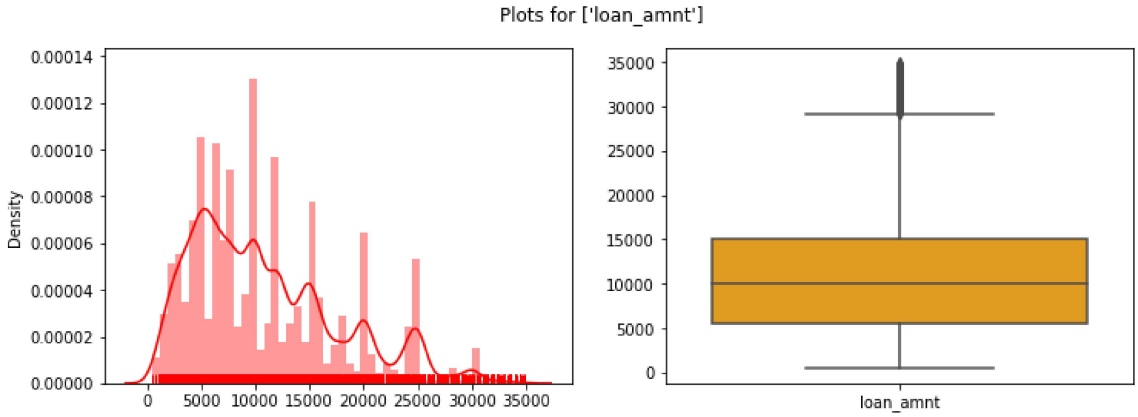
```
# outlier treatment

df = df[df["loan_amnt"] < df["loan_amnt"].quantile(0.99)]
df['loan_amnt'].describe()
```

Out[37]:

```
count    37229.000000
mean     10898.885278
std       6832.299611
min        500.000000
25%       5500.000000
50%      10000.000000
75%      15000.000000
max      34800.000000
Name: loan_amnt, dtype: float64
```

In [38]:

```
# loan amount box plot after outlier treatment

columns_to_be_plotted = ['loan_amnt']
draw_boxplot(df, columns_to_be_plotted )
df['loan_amnt'].describe()
```

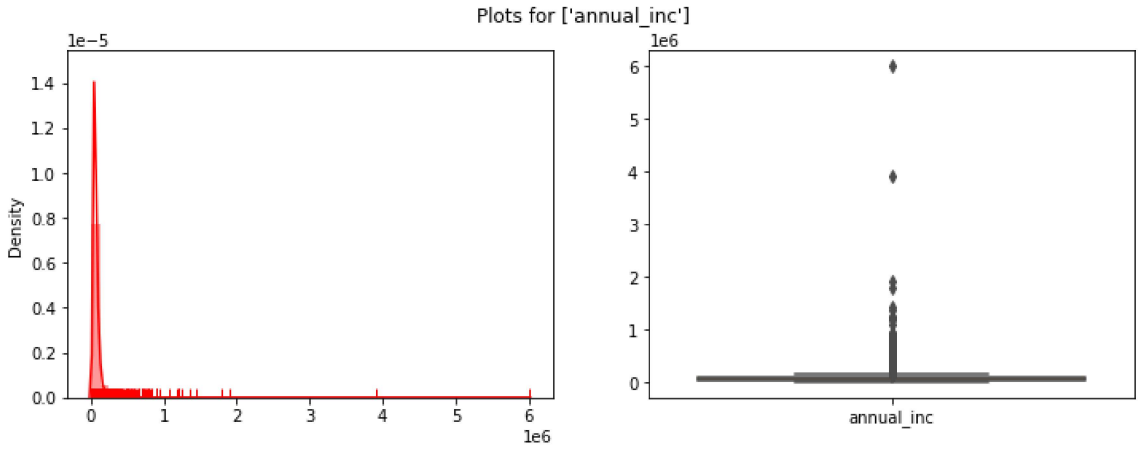


Out[38]:

```
count    37229.000000
mean     10898.885278
std       6832.299611
min        500.000000
25%       5500.000000
50%      10000.000000
75%      15000.000000
max      34800.000000
Name: loan_amnt, dtype: float64
```

In [39]:

```
# Annual income box plot

columns_to_be_plotted = ['annual_inc']
draw_boxplot(df, columns_to_be_plotted )
df['annual_inc'].describe()
```



Out[39]:

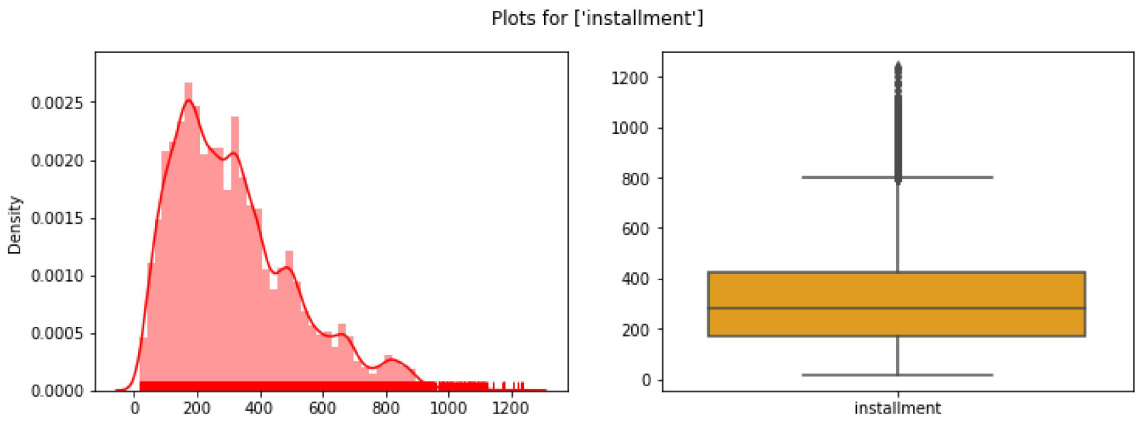
```
count    3.722900e+04
mean     6.853547e+04
std      6.277006e+04
min      4.000000e+03
25%      4.100000e+04
50%      5.900000e+04
75%      8.199600e+04
max      6.000000e+06
Name: annual_inc, dtype: float64
```

Observation:

- Around 361 people are having more the 235000 annual income. and 99% of people are having incomes below 235000.

In [40]:

```
# installment box plot

columns_to_be_plotted = ['installment']
draw_boxplot(df, columns_to_be_plotted )
df['installment'].describe()
df['installment'].describe(percentiles=[0.05,0.1,0.25,0.5,0.75,0.9,0.95,0.99])
```



Out[40]:

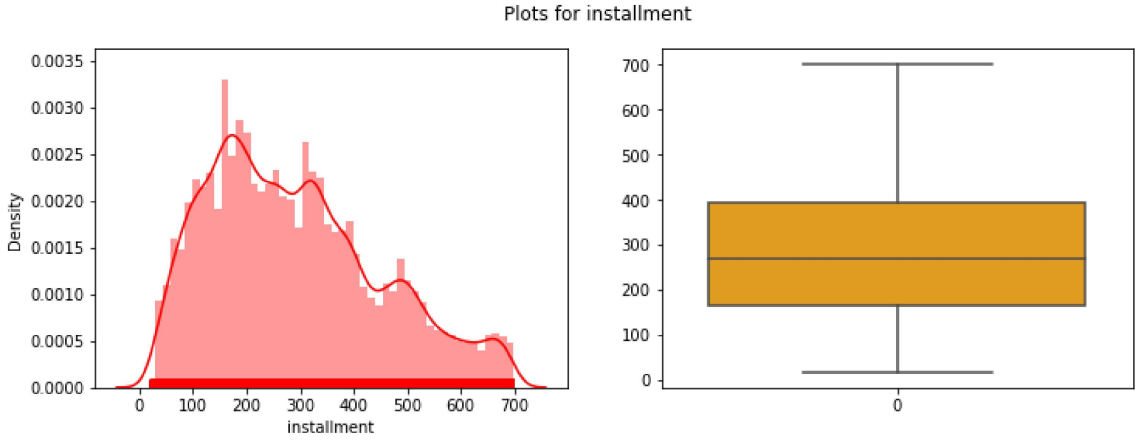
```
count    37229.000000
mean       316.926484
std        195.007127
min         16.080000
5%          72.880000
10%        100.400000
25%        167.340000
50%        278.830000
75%        420.070000
90%        595.370000
95%        699.730000
99%        873.760000
max       1236.740000
Name: installment, dtype: float64
```

Observation

- We see that most of the data having loan installment around 271.

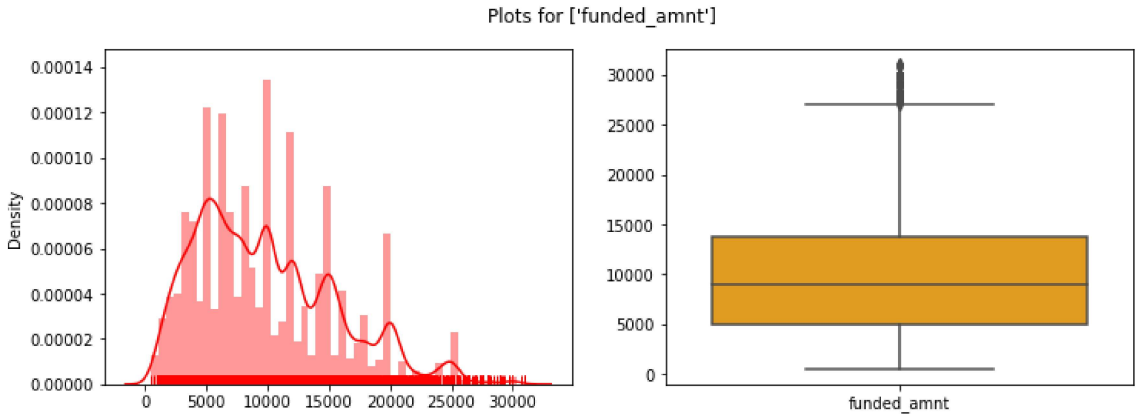
In [41]:

```
# After the outlier removal

df = df[df["installment"] < df["installment"].quantile(0.95)]
df['installment'].describe()
draw_boxplot(df, 'installment')
```



In [42]:

```
# box plot for funded_amnt

columns_to_be_plotted = ['funded_amnt']
draw_boxplot(df, columns_to_be_plotted )
df['funded_amnt'].describe()
```



Out[42]:

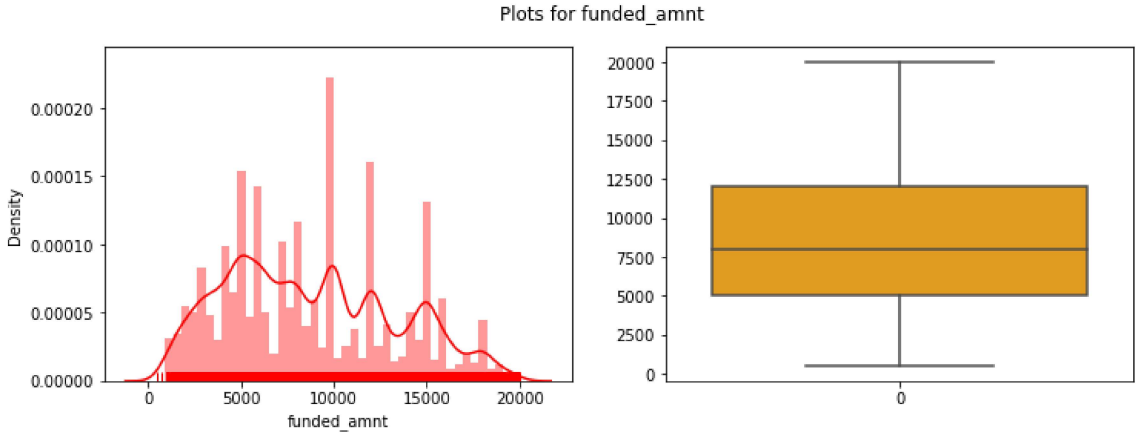
```
count    35365.000000
mean      9863.948113
std       5748.879513
min        500.000000
25%       5000.000000
50%       9000.000000
75%      13800.000000
max      31000.000000
Name: funded_amnt, dtype: float64
```

Observation

- We see that higher % of the data is having funded amount between 5000 to 15000.
- This may be due to high approval rate of the loan application.

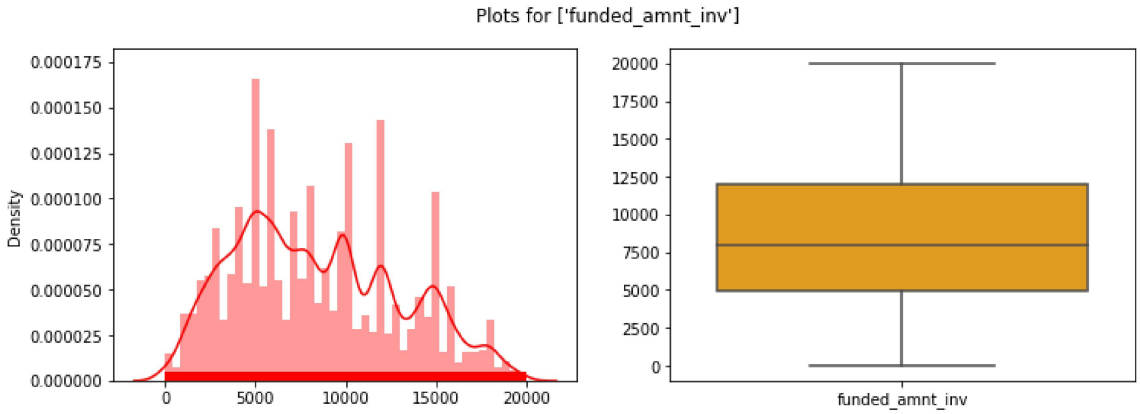
In [43]:

```
# box plot for funded_amnt after the outlier removal

df = df[df['funded_amnt'] < df['funded_amnt'].quantile(0.95)]
df['funded_amnt'].describe()
draw_boxplot(df, 'funded_amnt')
```



In [44]:

```
# box plot for funded_amnt_inv

columns_to_be_plotted = ['funded_amnt_inv']
draw_boxplot(df, columns_to_be_plotted )
df['funded_amnt_inv'].describe()
```

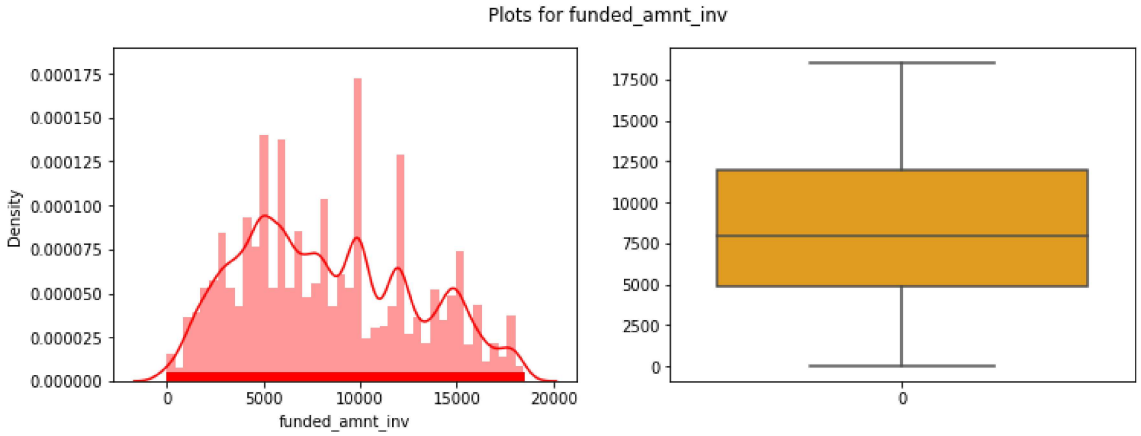


Out[44]:

```
count    32468.000000
mean      8432.740513
std       4545.919599
min          0.000000
25%       4950.000000
50%       7975.000000
75%      11975.000000
max      19950.000000
Name: funded_amnt_inv, dtype: float64
```

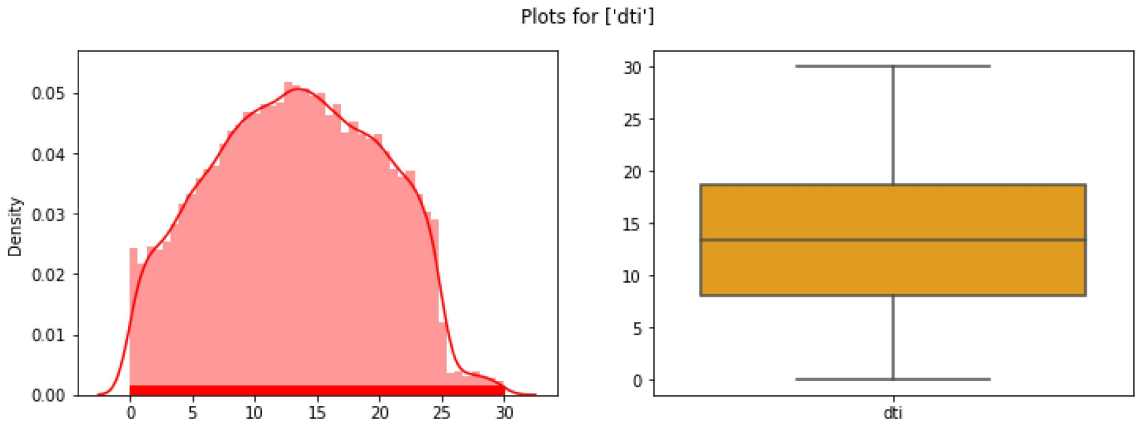
In [45]:

```
# box plot for funded_amnt_inv after outlier removal

df = df[df['funded_amnt_inv'] < df['funded_amnt_inv'].quantile(0.99)]
df['funded_amnt_inv'].describe()
draw_boxplot(df, 'funded_amnt_inv')
```



In [46]:

```
# box plot for dti

columns_to_be_plotted = ['dti']
draw_boxplot(df, columns_to_be_plotted)
df['dti'].describe()
```



Out[46]:

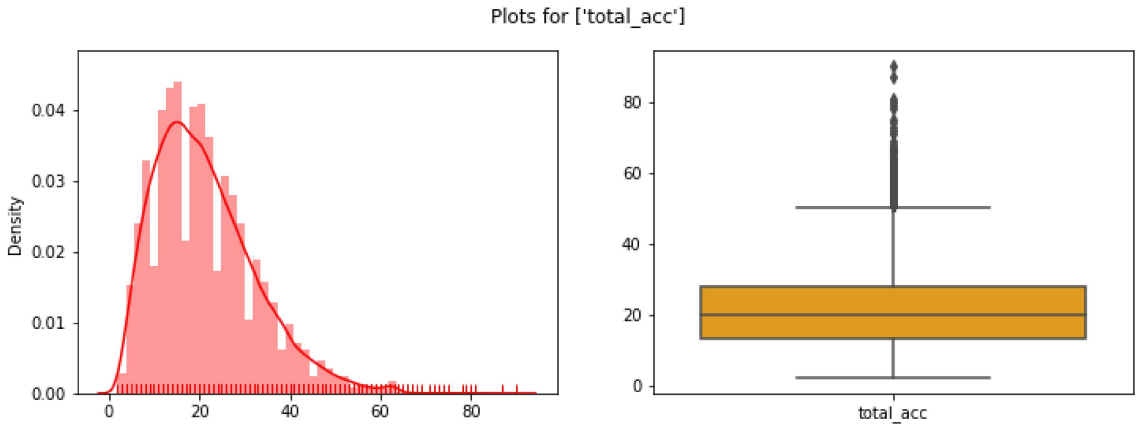
```
count    32133.000000
mean        13.281958
std          6.698356
min          0.000000
25%          8.080000
50%         13.340000
75%         18.610000
max         29.990000
Name: dti, dtype: float64
```

Observation

- There are no outlier which mean the loans are given to barrower's who have Debt to Income ration less than 30.

In [47]:

```
# box plot for total_acc

columns_to_be_plotted = ['total_acc']
draw_boxplot(df, columns_to_be_plotted)
df['total_acc'].describe()
```



Out[47]:

```
count    32133.000000
mean        21.336134
std         11.188688
min          2.000000
25%         13.000000
50%         20.000000
75%         28.000000
max         90.000000
Name: total_acc, dtype: float64
```

In [48]:

```
df['total_acc'].describe(percentiles=[0.05,0.1,0.25,0.5,0.75,0.9,0.95,0.99])
```

Out[48]:

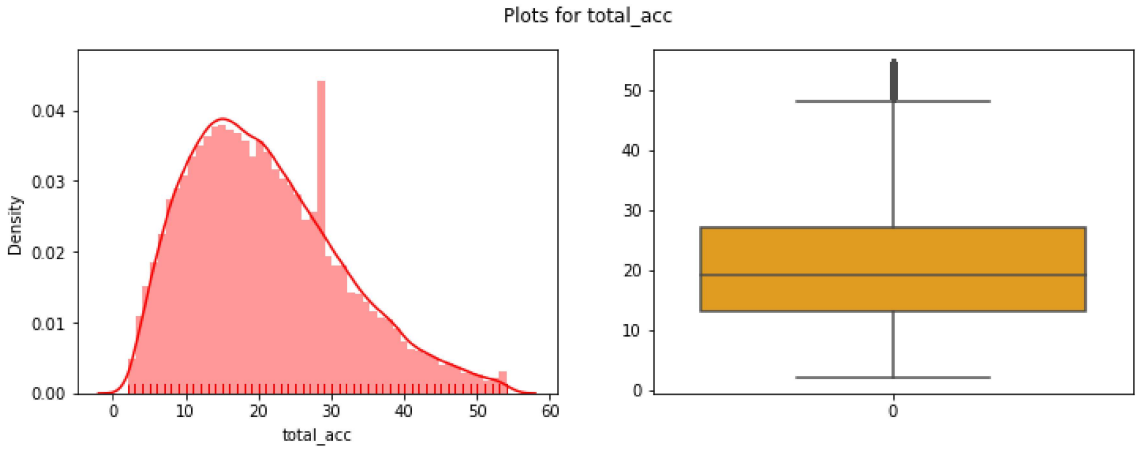
```
count    32133.000000
mean        21.336134
std         11.188688
min          2.000000
5%           6.000000
10%          8.000000
25%         13.000000
50%         20.000000
75%         28.000000
90%         37.000000
95%         42.000000
99%         55.000000
max         90.000000
Name: total_acc, dtype: float64
```

Obeservation

- From the above, we see that most of the application have credit line between 15 to 20

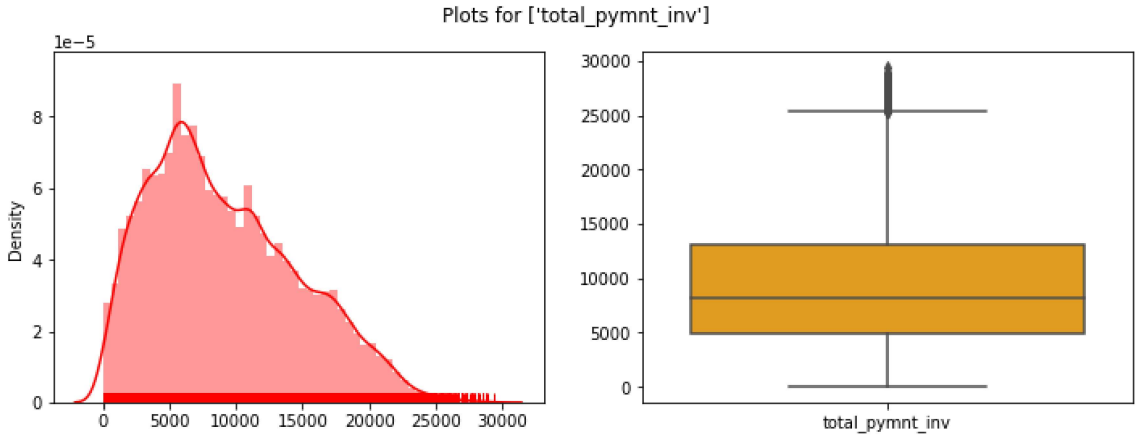
In [49]:

```
# box plot for total_acc after oultlier treatment
df = df[df["total_acc"] < df["total_acc"].quantile(0.99)]
df['total_acc'].describe()
draw_boxplot(df, "total_acc")
```



In [50]:

```
# box plot for total_pymnt_inv
columns_to_be_plotted = ['total_pymnt_inv']
draw_boxplot(df, columns_to_be_plotted)
df['total_pymnt_inv'].describe()
```



Out[50]:

```
count    31806.000000
mean      9173.211764
std       5583.124067
min          0.000000
25%       4829.232500
50%       8233.295000
75%      13039.575000
max      29339.680000
Name: total_pymnt_inv, dtype: float64
```

In [51]:

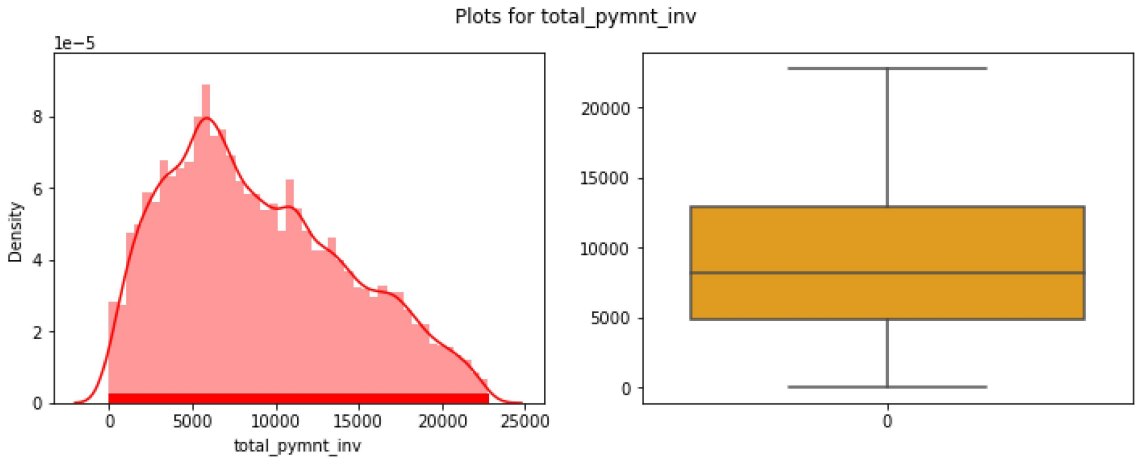
```
df['total_pymnt_inv'].describe(percentiles=[0.05,0.1,0.25,0.5,0.75,0.9,0.95,0.99])
```

Out[51]:

```
count    31806.000000
mean      9173.211764
std       5583.124067
min          0.000000
5%        1471.275000
10%       2436.150000
25%       4829.232500
50%       8233.295000
75%      13039.575000
90%      17345.845000
95%      19506.867500
99%      22748.272000
max      29339.680000
Name: total_pymnt_inv, dtype: float64
```

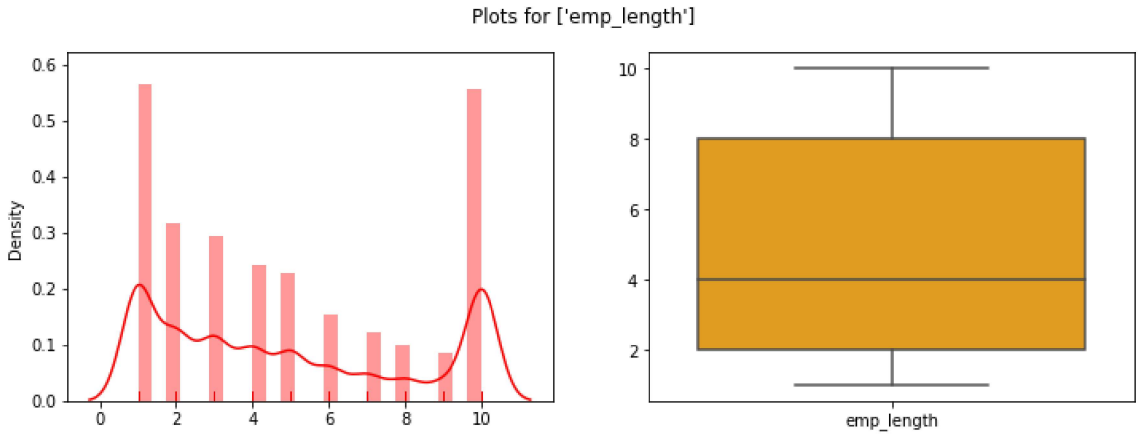
In [52]:

```
#box plot for total_pymnt_inv after outlier removal
df = df[df["total_pymnt_inv"] < df["total_pymnt_inv"].quantile(0.99)]
df['total_pymnt_inv'].describe()
draw_boxplot(df, "total_pymnt_inv")
```



In [53]:

```
# box plot for total_pymnt_inv
columns_to_be_plotted = ['emp_length']
draw_boxplot(df, columns_to_be_plotted)
df['emp_length'].describe()
```



Out[53]:

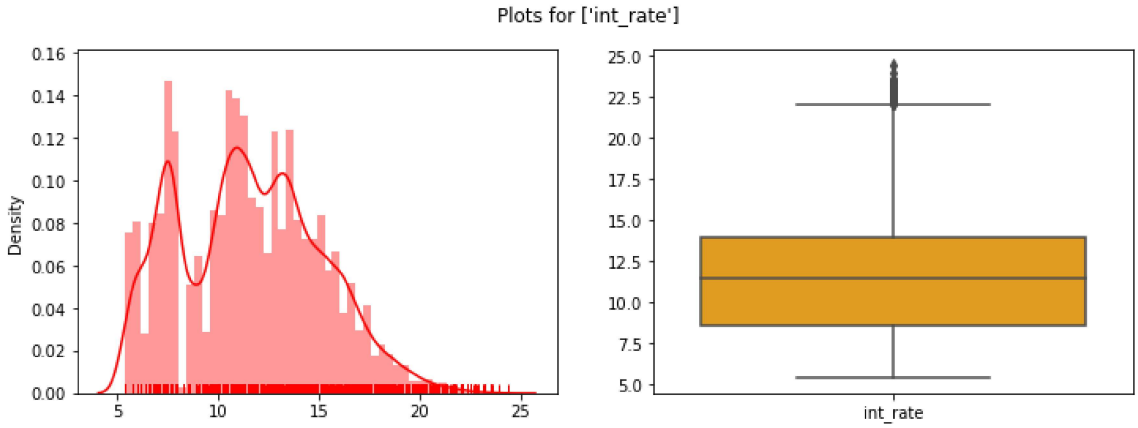
```
count    31487.000000
mean         4.921269
std          3.362265
min          1.000000
25%          2.000000
50%          4.000000
75%          8.000000
max         10.000000
Name: emp_length, dtype: float64
```

Obeservation

- From the above, we see that most of the loan application has employment of more than 2 year and less than 10 years.
- Some people took loan with some stable employement length (more than a year)

In [54]:

```
# box plot for interest rate
columns_to_be_plotted = ['int_rate']
draw_boxplot(df, columns_to_be_plotted)
df['int_rate'].describe()
```



Out[54]:

```
count    31487.000000
mean        11.582327
std          3.519794
min          5.420000
25%          8.590000
50%         11.490000
75%         13.990000
max         24.400000
Name: int_rate, dtype: float64
```

Obeservation

- From the above, we see that most of the interest rates lies between 9% to 14.5%.
- Some people took loan at higher rates of interest i.e., 22.5%

Analysis for categorical data

In [55]:

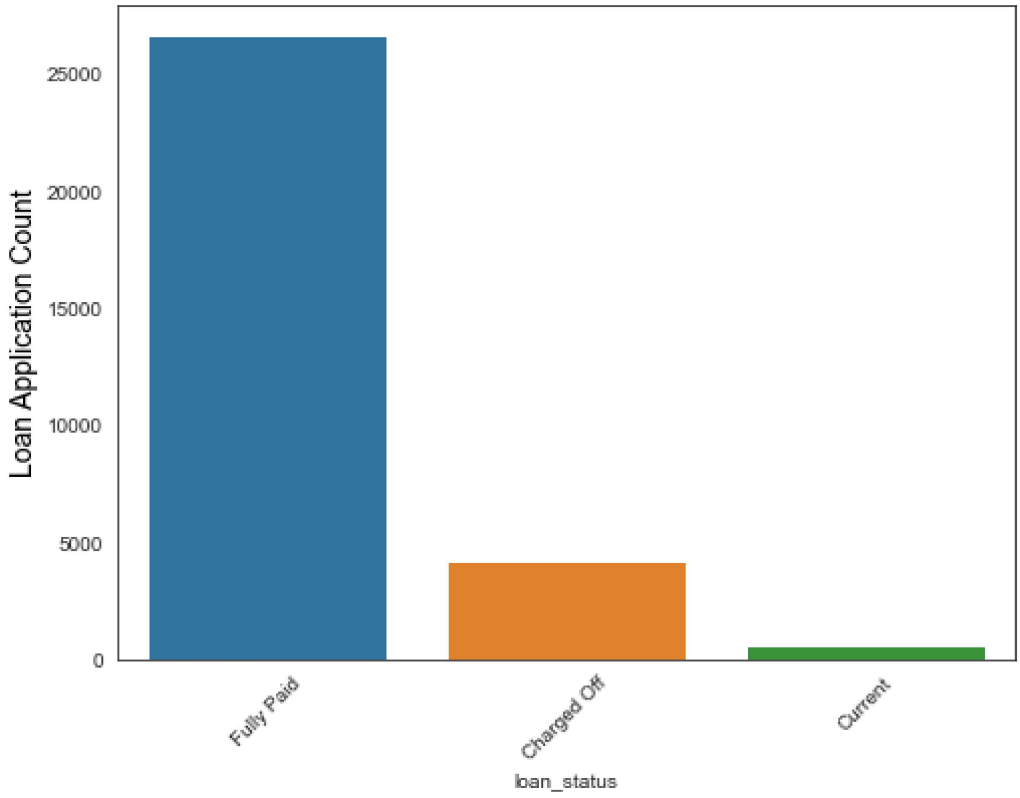
```
# Lets define a function that plots bar plot
```

In [56]:

```
def draw_bar_plot(dataframe, column_name):
    plt.figure(figsize=(8,6))
    sns.set_style("white")
    plt.xticks(rotation = 45)
    ax = sns.countplot(x= f'{column_name}',data= dataframe)
    ax.set_ylabel('Loan Application Count',fontsize=14,color = 'black')
    s=dataframe[column_name].value_counts()
```

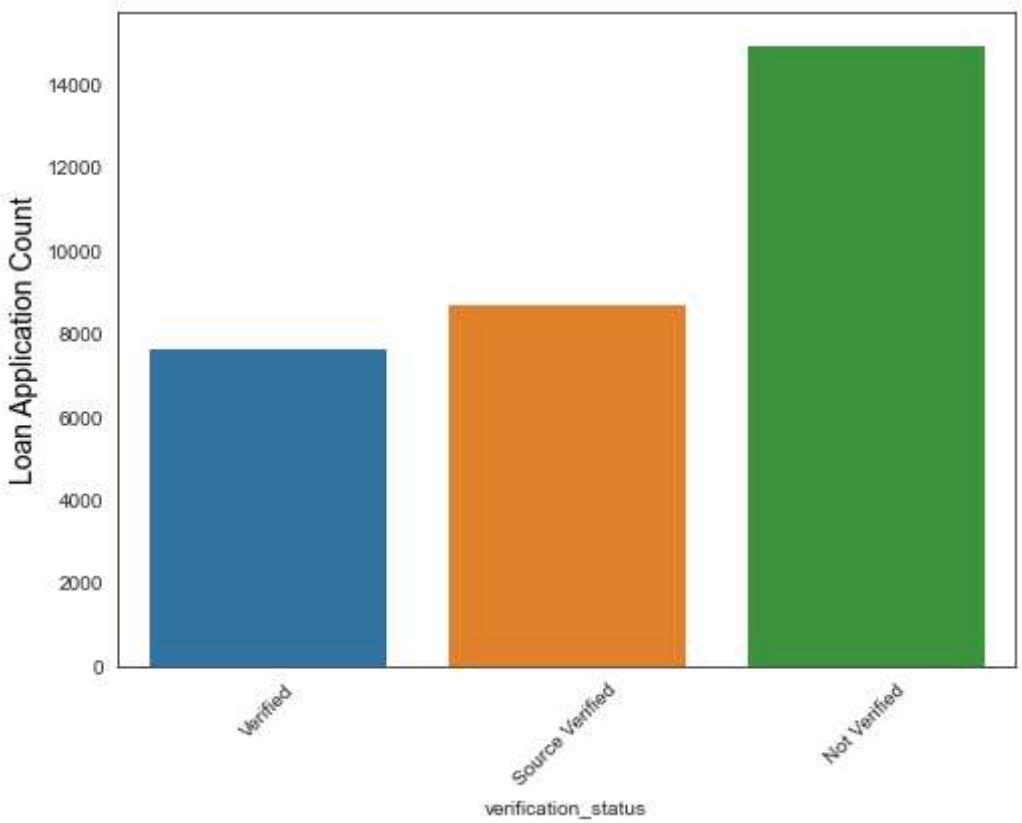
In [57]:

```
# Bar plot for Loan application vs loan status
draw_bar_plot(df, 'loan_status')
```



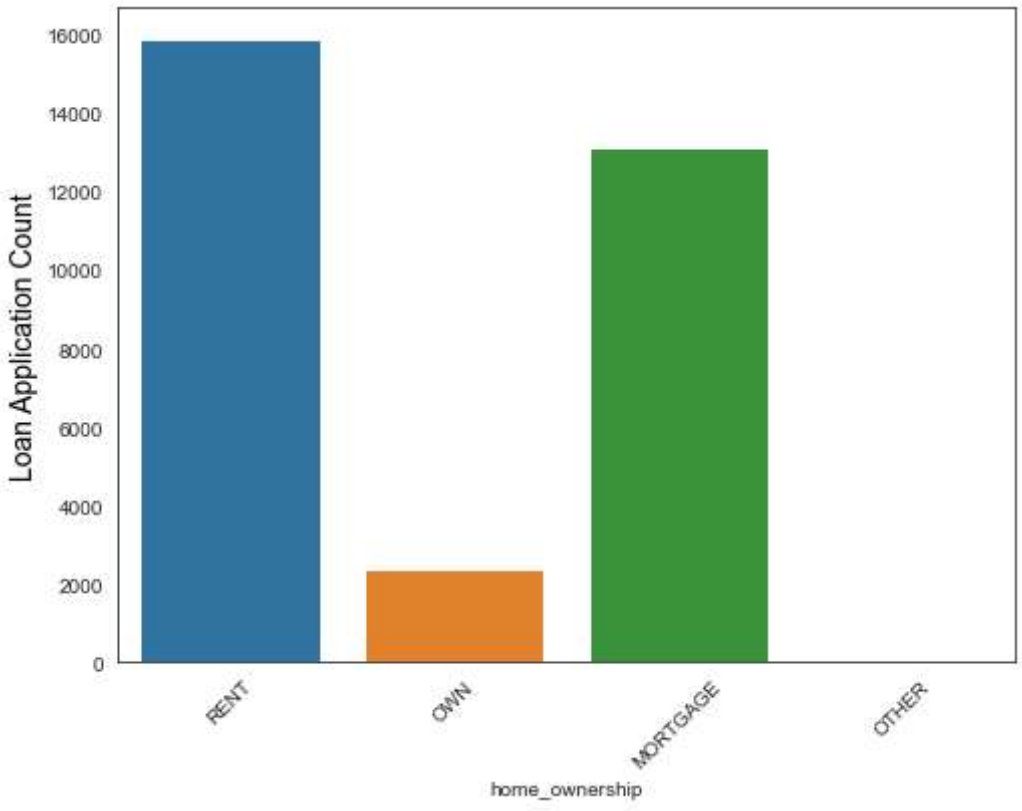
In [58]:

```
# bar plot for varification status count
draw_bar_plot(df, 'verification_status')
```



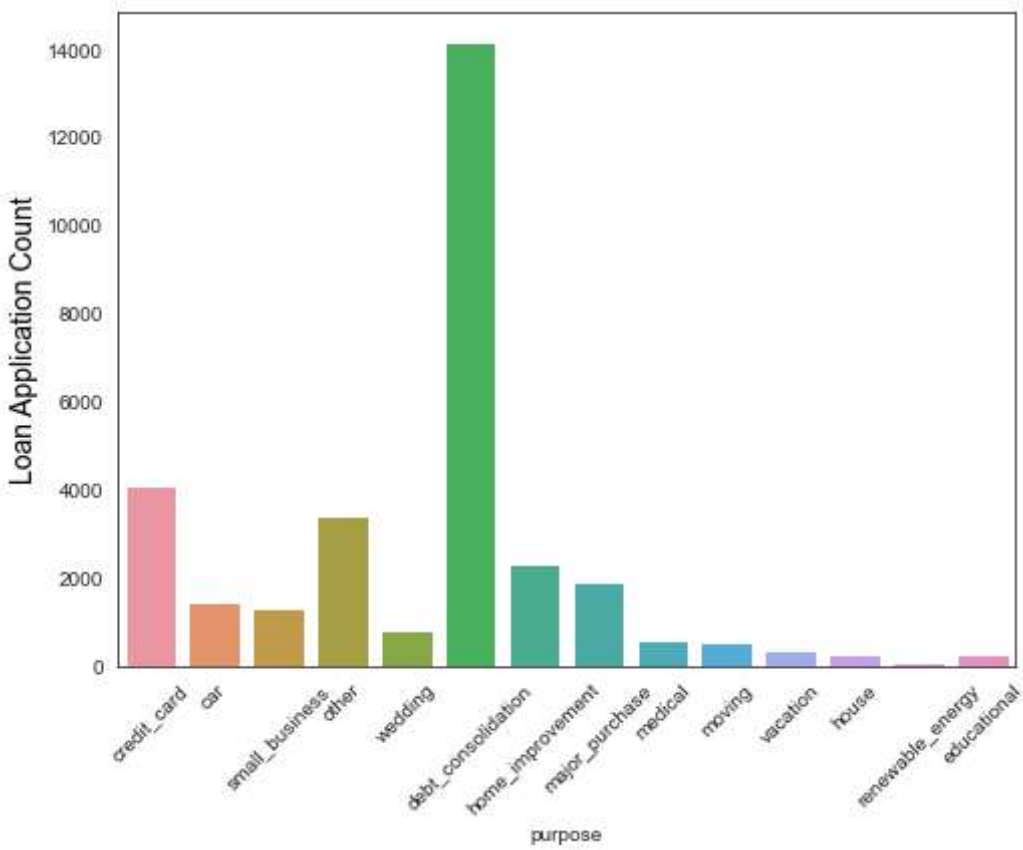
In [59]:

```
# bar plot for home ownership count
draw_bar_plot(df, 'home_ownership')
```



In [60]:

```
# bar plot for purpose count
draw_bar_plot(df, 'purpose')
```



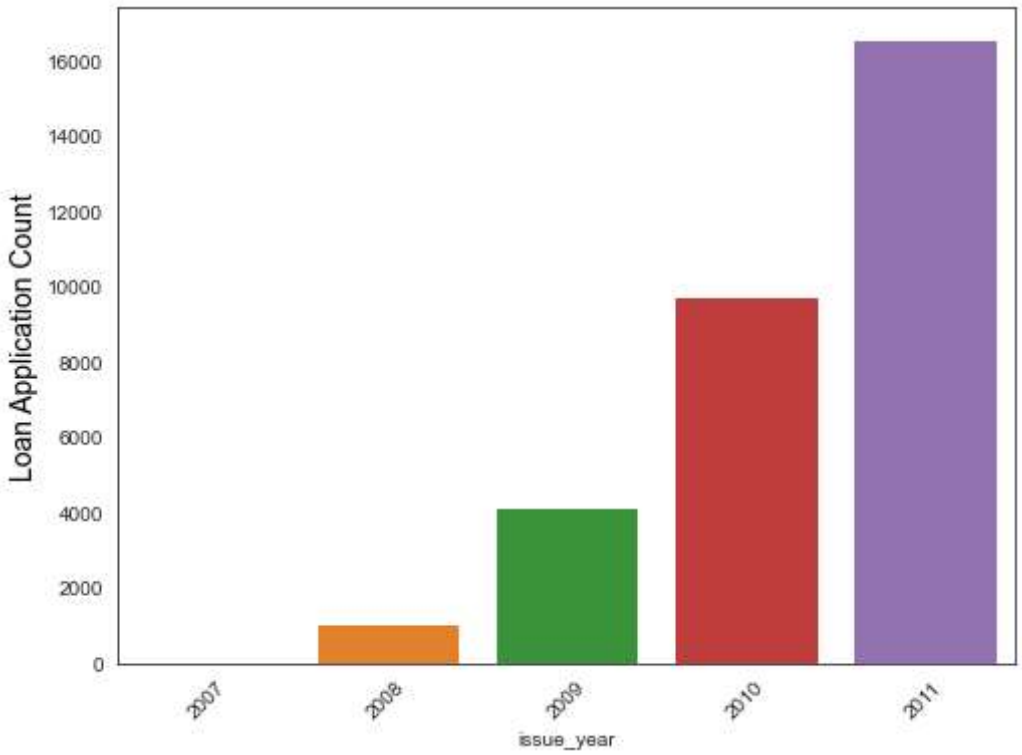
Observation:

- Most of the loan application are completed.
- Most of the apllication have not varified their application.
- Most of the application have home ownership as rented.
- Most of the loan applicant are taking loan for the purpose of debt consolidation and least for the renewable energy.

Number of application over the years

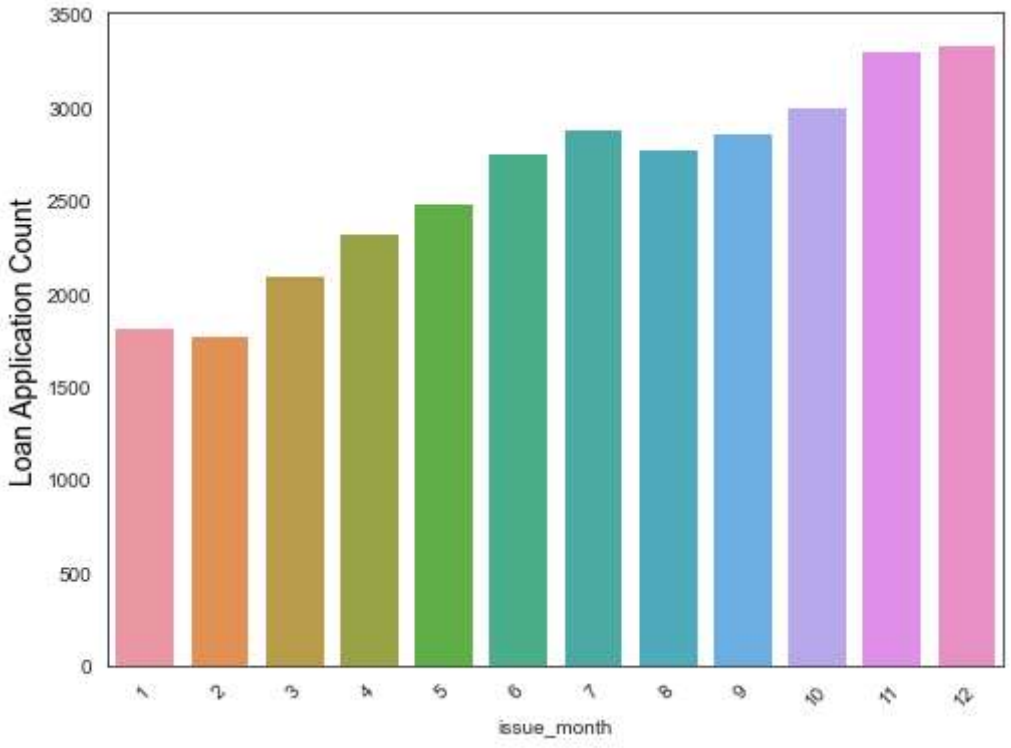
In [61]:

```
#bar plot for loan application count over the years
draw_bar_plot(df, 'issue_year')
```



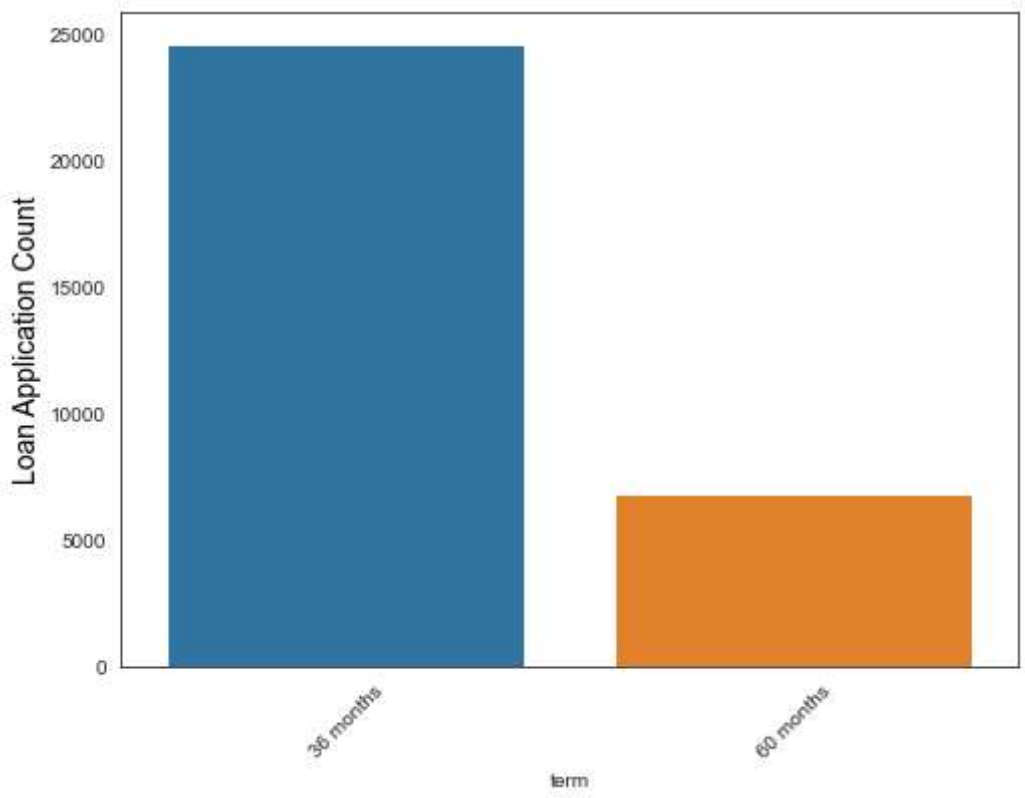
In [62]:

```
# bar plot for application count w.r.t monts
draw_bar_plot(df, 'issue_month')
```



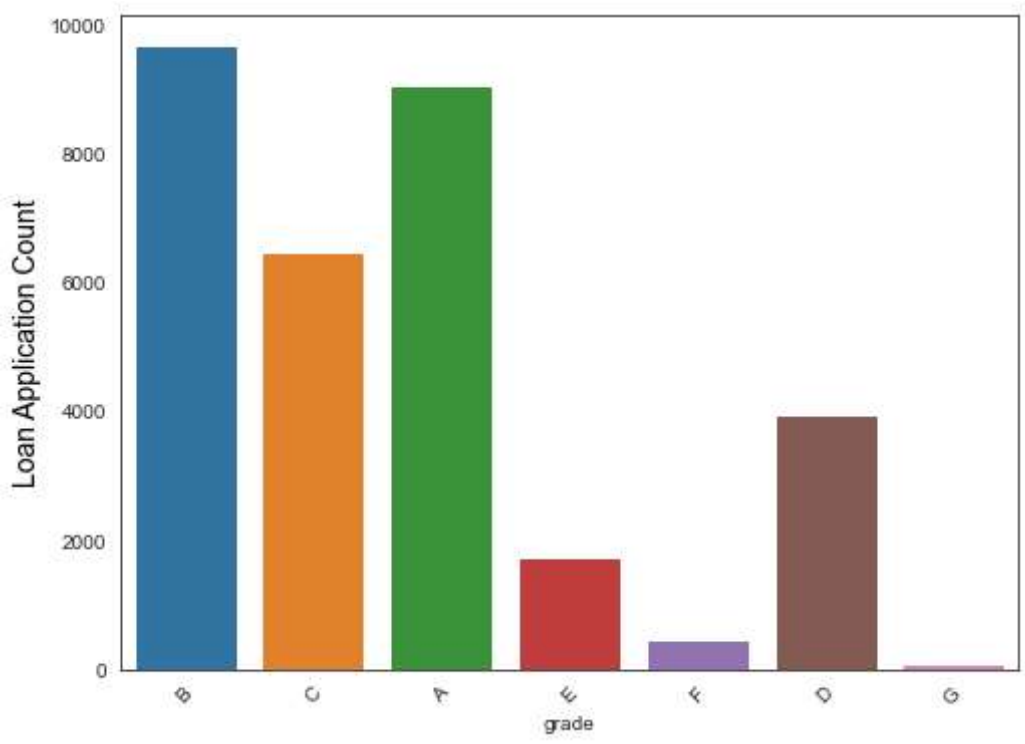
In [63]:

```
# bar plot for application based on term duration
draw_bar_plot(df, 'term')
```



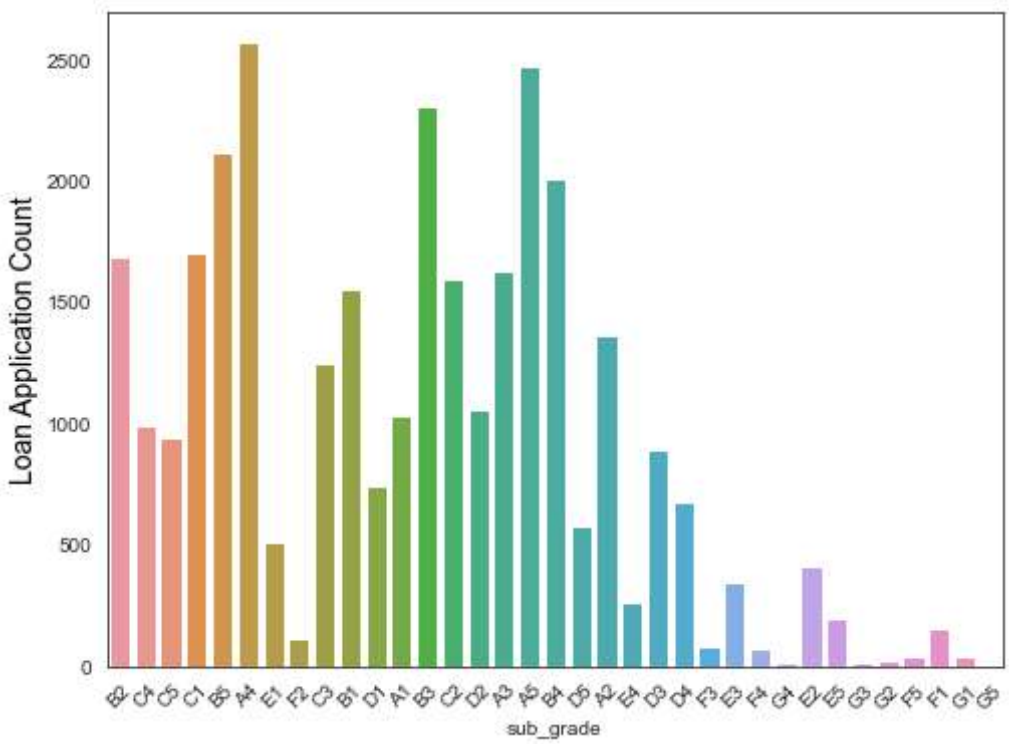
In [64]:

```
# bar plot for the application based on grade
draw_bar_plot(df, 'grade')
```



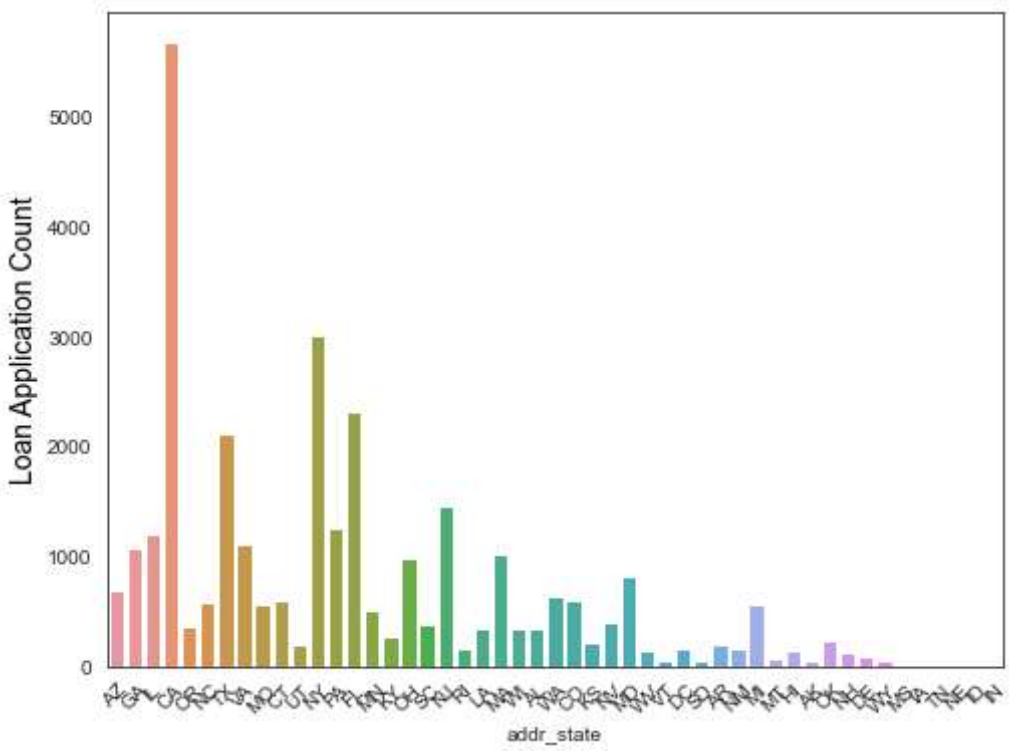
In [65]:

```
# bar plot for applicant count based on sub grade
draw_bar_plot(df, 'sub_grade')
```



In [66]:

```
# bar plot for applicants counts based on states
draw_bar_plot(df, 'addr_state')
```



Observation:

- In recent years, the loan application have been increased

- In the month of oct, nov and dec most of the loans are being issued.
- This is due to decrease in the interest rate
- There are more applicant who took 36 months duration loan as compared to 60 months.
- Most of the applicants belongs to grade A and B
- For subgrade same as most of the applicants belong to Grade A and B.
- Most of the applicants are from NY and CA.

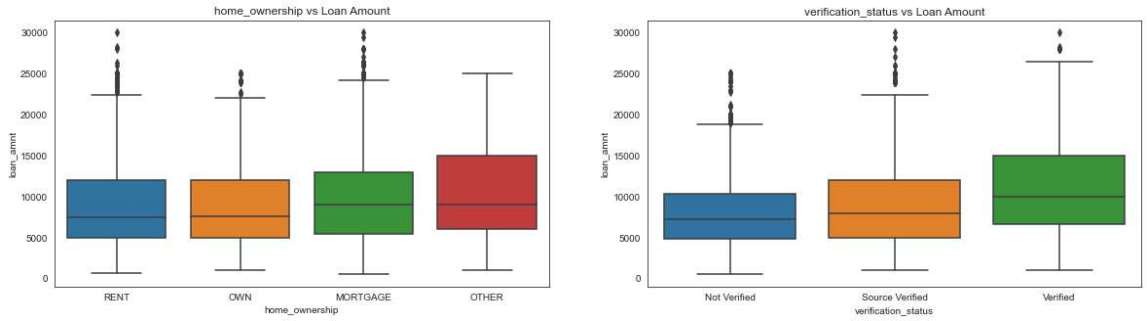
Segmented Analysis

In [67]:

```
# home_ownership vs Loan Amount

plt.figure(figsize=(20,5))
plt.subplot(121)
sns.boxplot(x='home_ownership', y=df.loan_amnt, data=df)
plt.title('home_ownership vs Loan Amount')
plt.subplot(122)
plt.title('verification_status-Loan Amount')
verification_status_ord = df.verification_status.unique()
verification_status_ord.sort()
sns.boxplot(x='verification_status', y=df.loan_amnt, order = verification_status_ord, da
```

Out[67]:

```
<AxesSubplot:title={'center':'verification_status vs Loan Amount'}, xlabel
='verification_status', ylabel='loan_amnt'>
```



Observations:

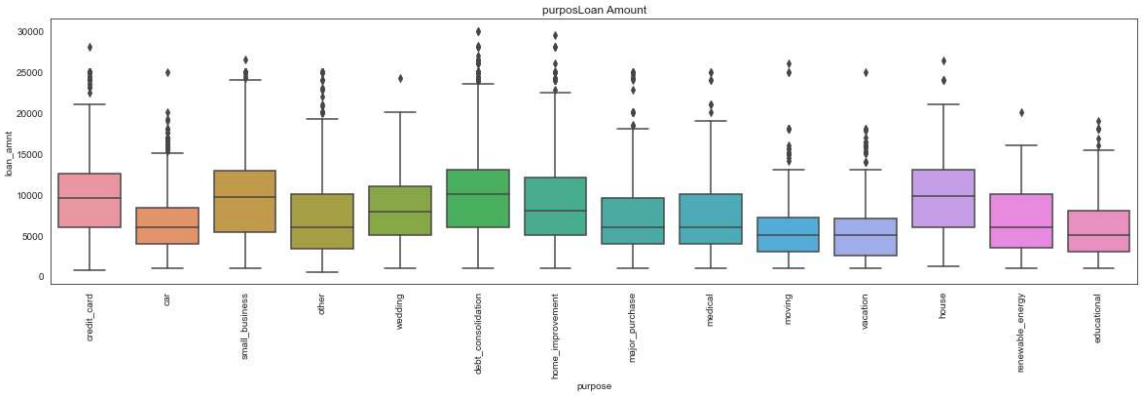
- Most of the loan applicants are having MORTGAGE homeownership.
- Most of the borrower have sourced verified

In [70]:

```
# purpose vs loan amount
plt.figure(figsize=(20,5))
sns.boxplot(x='purpose', y=df.loan_amnt, data=df)
plt.xticks(rotation=90)
plt.title('purposLoan Amount')
```

Out[70]:

```
Text(0.5, 1.0, 'purposLoan Amount')
```



Observations:

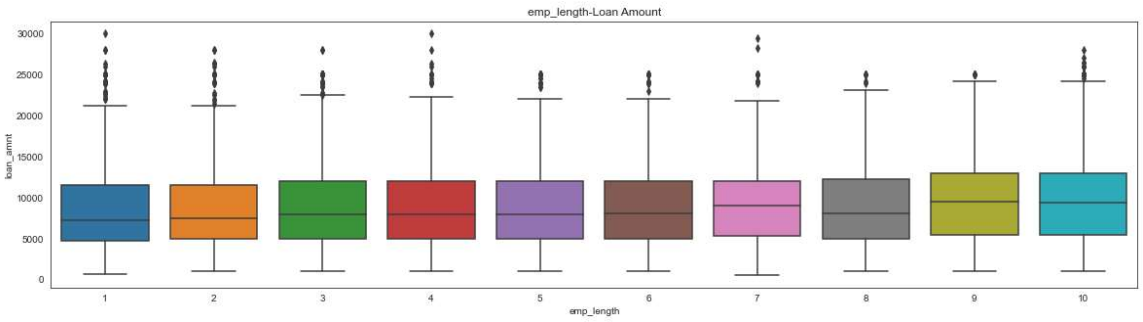
- Most of the loan amount is for Small bussiness

In [71]:

```
# emp_length vs amount
plt.figure(figsize=(20,5))
sns.boxplot(x='emp_length', y=df.loan_amnt, data=df)
plt.title('emp_length-Loan Amount')
```

Out[71]:

```
Text(0.5, 1.0, 'emp_length-Loan Amount')
```

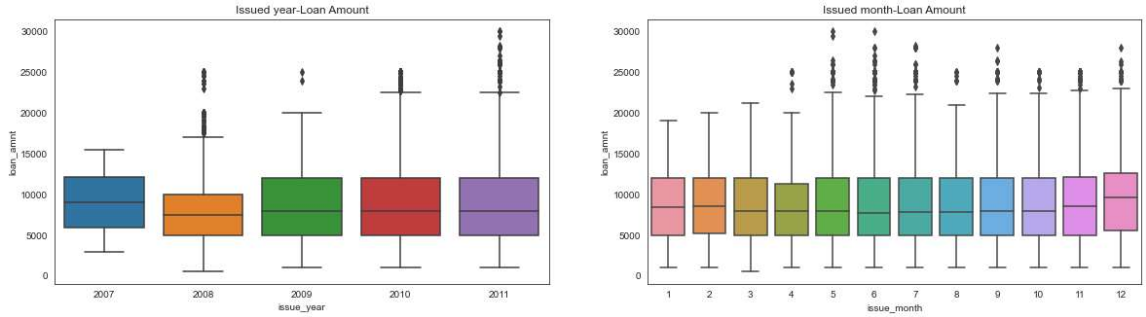


Observations:

- More applicants who got loan approved are of 10yrs, so less age is the factor to get loan rejected.

In [72]:

```
#Issue time vs loan amount

plt.figure(figsize=(20,5))
plt.subplot(121)
sns.boxplot(x=df.issue_year, y=df.loan_amnt, data=df)
plt.title('Issued year-Loan Amount')
plt.subplot(122)
sns.boxplot(x=df.issue_month, y=df.loan_amnt, data=df)
plt.title('Issued month-Loan Amount')
plt.show()
```



Observations:

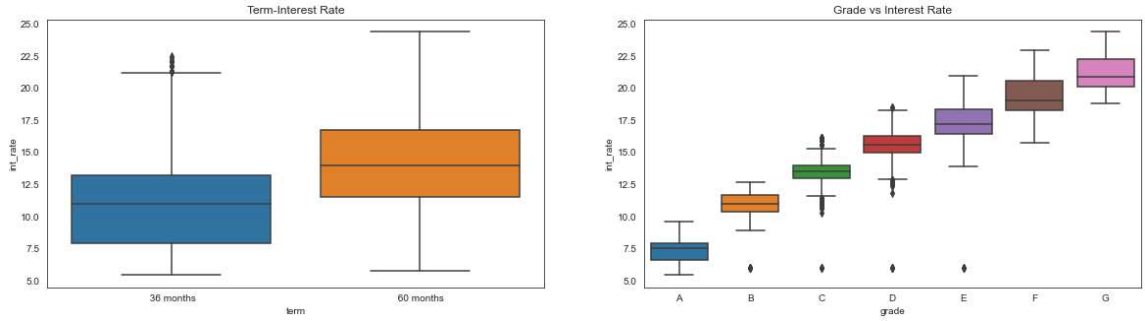
- More amount of loan is disbursed in year 2011 and in the last three months of the year.

In [73]:

```
# Term vs interest rate
plt.figure(figsize=(20,5))
plt.subplot(121)
sns.boxplot(x='term', y=df.int_rate, data=df)
plt.title('Term-Interest Rate')
plt.subplot(122)
plt.title('Grade vs Interest Rate')
grade_ord = df.grade.unique()
grade_ord.sort()
sns.boxplot(x='grade', y=df.int_rate, order = grade_ord, data=df)
```

Out[73]:

```
<AxesSubplot:title={'center':'Grade vs Interest Rate'}, xlabel='grade', yl
abel='int_rate'>
```



Observation:

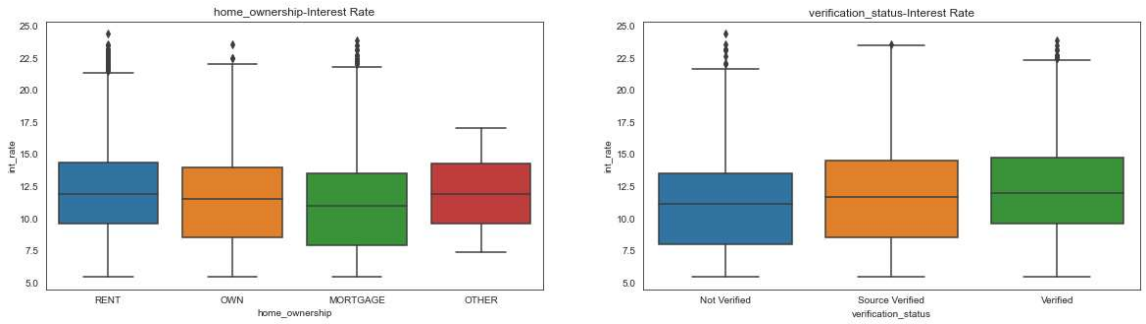
- The most interest rate are from 60 months tenure and from Grade G

In [74]:

```
# home_ownership vs Interest Rate
plt.figure(figsize=(20,5))
plt.subplot(121)
sns.boxplot(x='home_ownership', y=df.int_rate, data=df)
plt.title('home_ownership-Interest Rate')
plt.subplot(122)
plt.title('verification_status-Interest Rate')
verification_status_ord = df.verification_status.unique()
verification_status_ord.sort()
sns.boxplot(x='verification_status', y=df.int_rate, order = verification_status_ord, dat
```

Out[74]:

```
<AxesSubplot:title={'center':'verification_status-Interest Rate'}, xlabel
='verification_status', ylabel='int_rate'>
```



Observations:

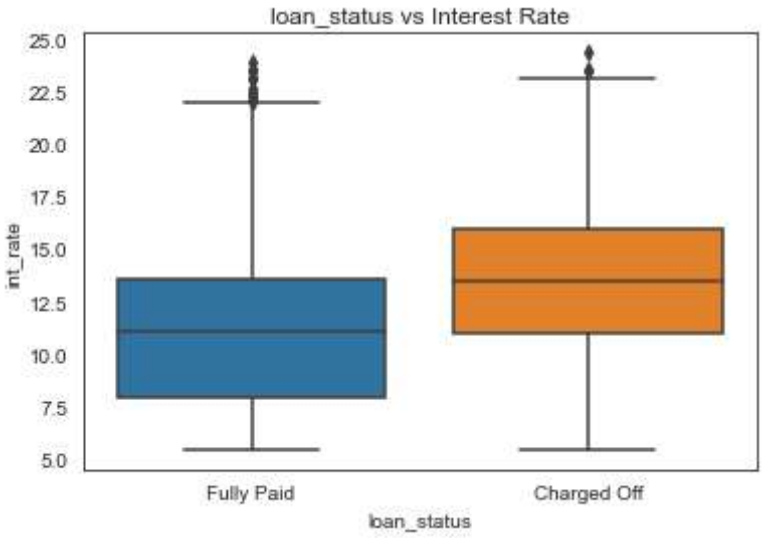
- Home ownership having MORTGAGE is having less rate of interest.

In [75]:

```
data = df[df.loan_status != 'Current']
sns.boxplot(x='loan_status', y=data.int_rate, data=data)
plt.title('loan_status vs Interest Rate')
```

Out[75]:

```
Text(0.5, 1.0, 'loan_status vs Interest Rate')
```



Observations:

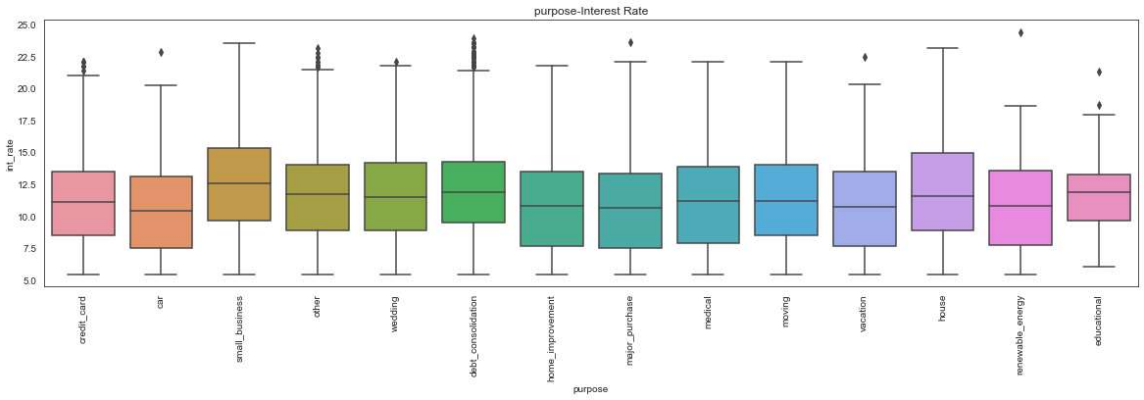
- People having more rate of interest are getting defaulted.

In [76]:

```
# purpose vs Interest Rate
plt.figure(figsize=(20,5))
sns.boxplot(x='purpose', y=df.int_rate, data=df)
plt.xticks(rotation=90)
plt.title('purpose-Interest Rate')
```

Out[76]:

```
Text(0.5, 1.0, 'purpose-Interest Rate')
```



Observations:

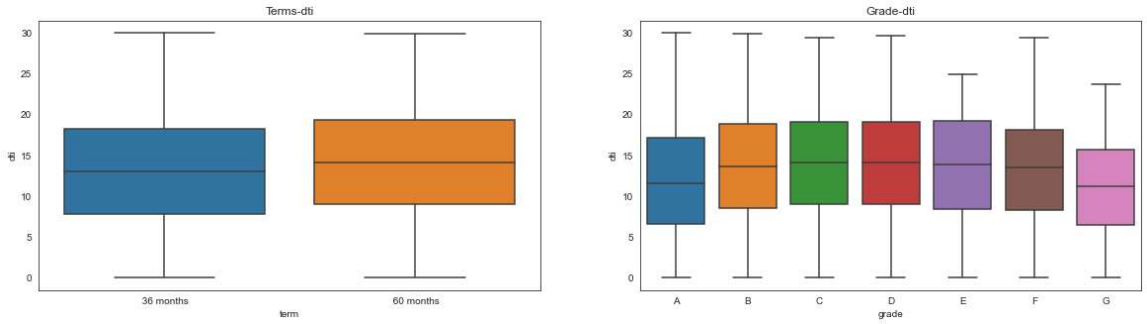
- Small Business, Debt Consolidation and House loans are given with more interest rates comparatively

In [79]:

```
#Terms vs dti
plt.figure(figsize=(20,5))
plt.subplot(121)
sns.boxplot(x='term', y=df.dti, data=df)
plt.title('Terms-dti')
plt.subplot(122)
plt.title('Grade-dti')
grade_ord = df.grade.unique()
grade_ord.sort()
sns.boxplot(x='grade', y=df.dti, order = grade_ord, data=df)
```

Out[79]:

```
<AxesSubplot:title={'center':'Grade-dti'}, xlabel='grade', ylabel='dti'>
```



Observation:

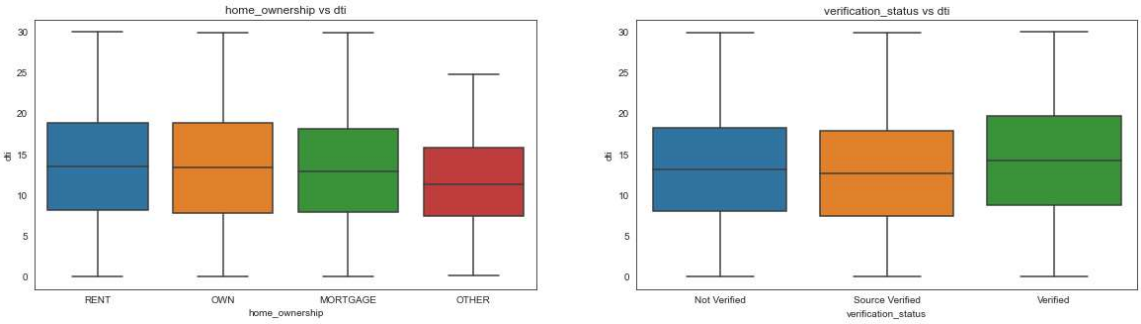
- DTI is high 60 months tenure
- A Grade barrowers is having low DTI.

In [80]:

```
# verification_status vs dti
plt.figure(figsize=(20,5))
plt.subplot(121)
sns.boxplot(x='home_ownership', y=df.dti, data=df)
plt.title('home_ownership-dti')
plt.subplot(122)
plt.title('verification_status-dti')
verification_status_ord = df.verification_status.unique()
verification_status_ord.sort()
sns.boxplot(x='verification_status', y=df.dti, order = verification_status_ord, data=df)
```

Out[80]:

```
<AxesSubplot:title={'center':'verification_status vs dti'}, xlabel='verifi
cation_status', ylabel='dti'>
```



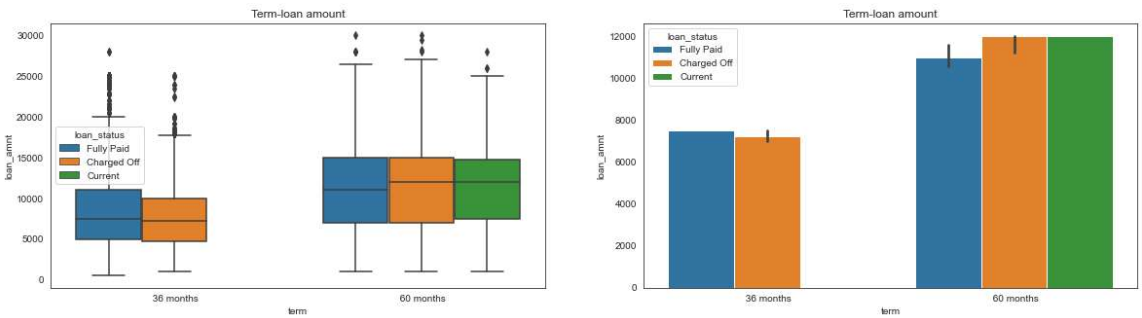
Observations:

- Other home ownership has less DTI due to other people have mortgage and home loans.

Bivariate Analysis

In [83]:

```
#Term vs loan amount
plt.figure(figsize=(20,5))
plt.subplot(121)
sns.boxplot(x='term', y='loan_amnt', hue='loan_status', data=df)
plt.title('Term-loan amount')
plt.subplot(122)
sns.barplot(x='term', y='loan_amnt', hue='loan_status', data=df, estimator=np.median)
plt.title('Term-loan amount')
plt.show()
```



In [84]:

```
#Proportion of values for each category
df = data.groupby(['term', 'loan_status'], as_index=False)['id'].count()
df['proportion'] = df.groupby('term').transform(lambda x: x/x.sum())
sns.barplot(x='term', y='proportion', hue='loan_status', data=df, hue_order = ['Fully Pa

plt.title('Term vs Loan status')
```

Out[84]:

```
Text(0.5, 1.0, 'Term vs Loan status')
```

Observations:

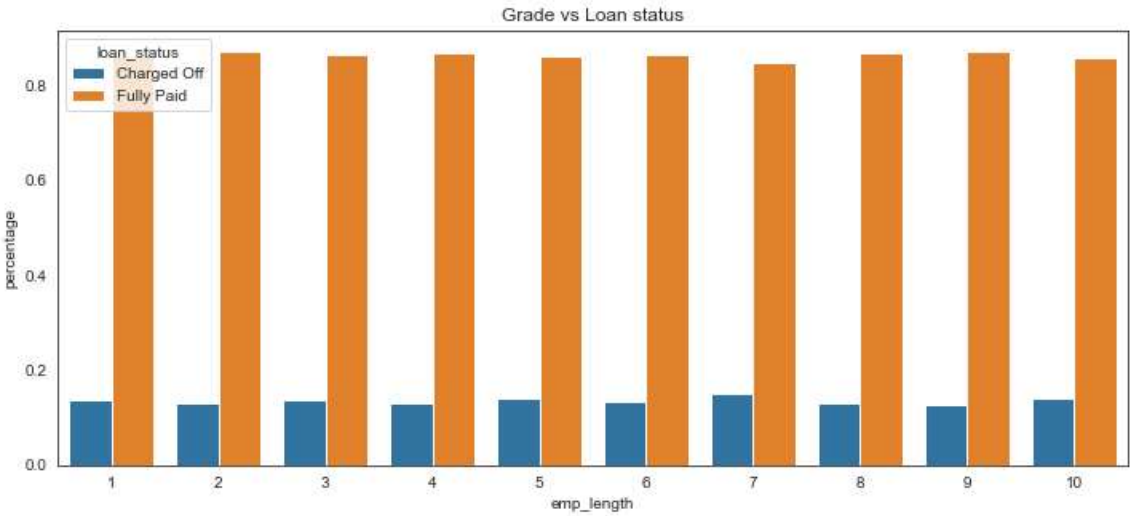
- more proportion of borrowers defaulted loan in 60 months

In [85]:

```
df = data.groupby(['emp_length', 'loan_status'], as_index=False)['id'].count()
df['percentage'] = df.groupby('emp_length').transform(lambda x: x/x.sum())
plt.figure(figsize=(12,5))
sns.barplot(x='emp_length', y='percentage', hue='loan_status', data=df)
plt.title('Grade vs Loan status')
```

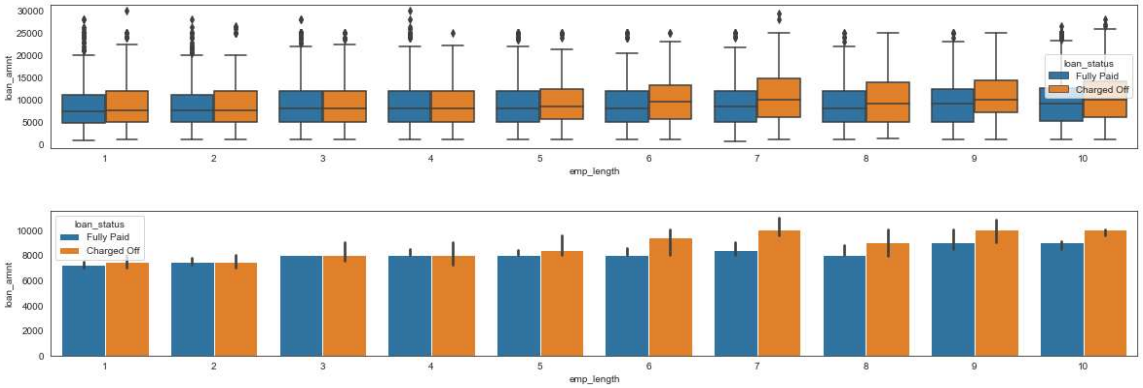
Out[85]:

```
Text(0.5, 1.0, 'Grade vs Loan status')
```



In [86]:

```
plt.figure(figsize=(20,6))
plt.subplot(211)
sns.boxplot(x='emp_length', y='loan_amnt', hue='loan_status', data=data)
plt.figure(figsize=(20,6))
plt.subplot(212)
sns.barplot(x='emp_length', y='loan_amnt', hue='loan_status', data=data, estimator=np.me
plt.show()
```

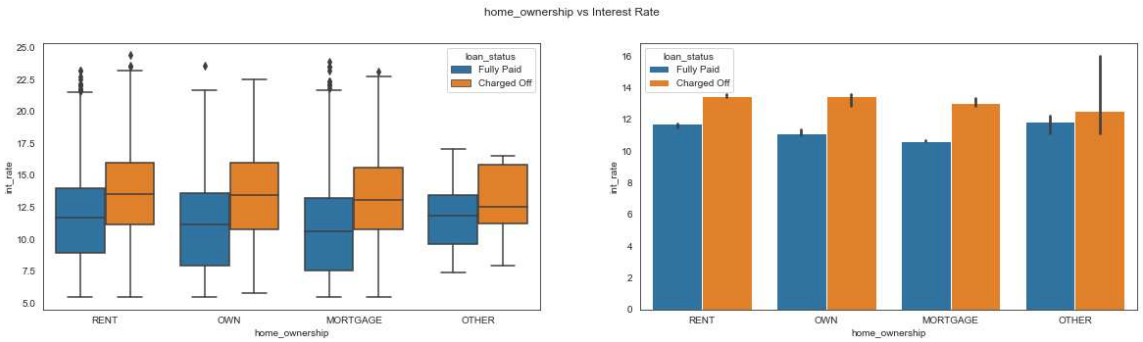


Observation:

- Borrowers with higher employment lengths and took more loan amounts got more default rate.

In [89]:

```
plt.figure(figsize=(20,5))
plt.subplot(121)
sns.boxplot(x='home_ownership', y='int_rate', hue='loan_status', data=data)
plt.subplot(122)
sns.barplot(x='home_ownership', y='int_rate', hue='loan_status', data=data, estimator=np
plt.suptitle('home_ownership vs Interest Rate')
plt.show()
```

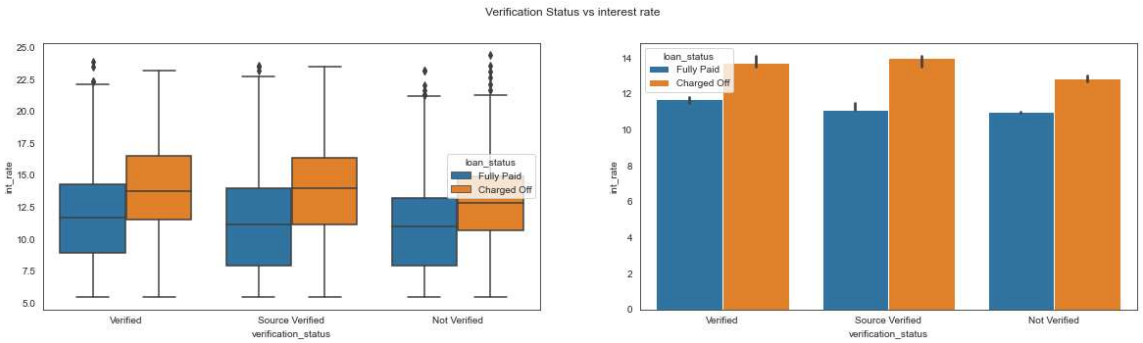


Observations:

- The interest rate iS POSITIVELY correlated with default rate.

In [90]:

```
plt.figure(figsize=(20,5))
plt.subplot(121)
sns.boxplot(x='verification_status', y='int_rate', hue='loan_status', data=data)
plt.subplot(122)
sns.barplot(x='verification_status', y='int_rate', hue='loan_status', data=data, estimat
plt.suptitle('Verification Status vs interest rate')
plt.show()
```

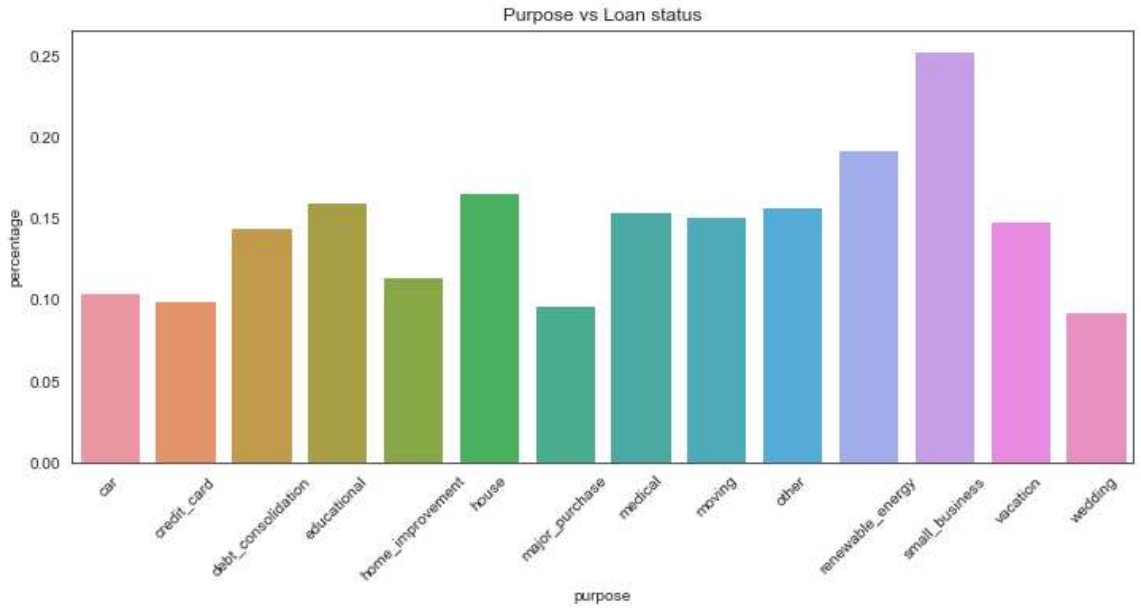


In [94]:

```
df = data.groupby(['purpose', 'loan_status'], as_index=False)['id'].count()
df['percentage'] = df.groupby('purpose').transform(lambda x: x/x.sum())
df = df[df.loan_status == 'Charged Off']
plt.figure(figsize=(12,5))
sns.barplot(x='purpose', y='percentage', data=df)
plt.xticks(rotation=45)
plt.title('Purpose vs Loan status')
```

Out[94]:

```
Text(0.5, 1.0, 'Purpose vs Loan status')
```

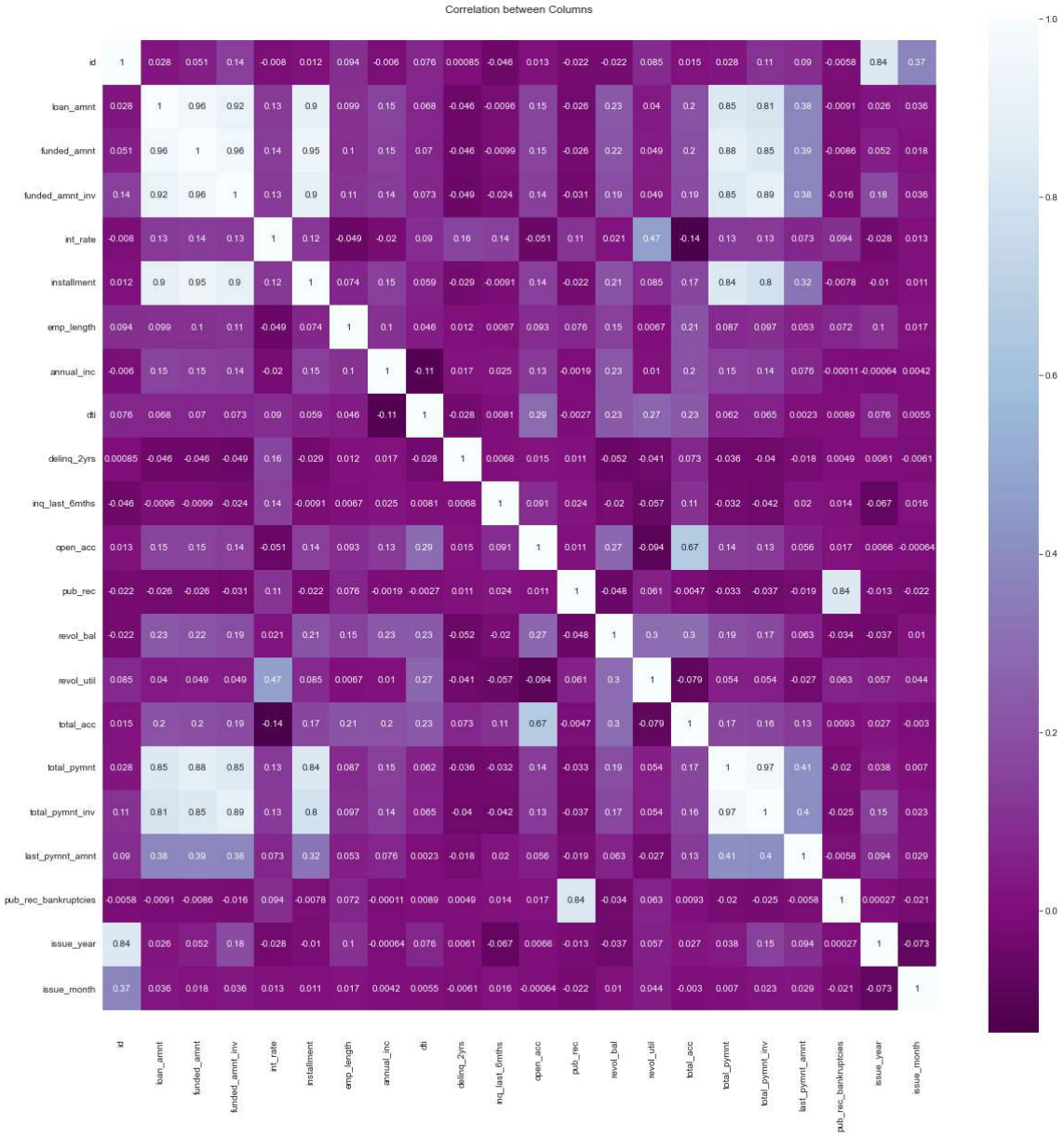


Observations:

- Charged-off are higher for small_business comparitively.

In [95]:

```
#Finding correlation matric
corr_matrix = data.corr()
plt.figure(figsize=(20,20))
#plotting correlatioon matric on a heat map
ax = sns.heatmap(corr_matrix, annot = True, cmap='BuPu_r')
top, bottom = ax.get_ylim()
ax.set_ylim(top+0.5, bottom-0.5)
plt.title("Correlation between Columns")
plt.show()
```



Observations:

- The public derogatory records correlated with public bankruptcies records.
- Interest rates are high for people with high revol utilisation