LoanTap Classification Project

1.Define Problem Statement and perform Exploratory Data Analysis

1.1.Definition of problem

The Goal of this project is to predict/make decisoion to give loan to customers to mitigate the risk of defaults

In [3]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency, f_oneway, ttest_ind,spearmanr
from scipy.stats import chi2, shapiro, boxcox, ttest_rel
from scipy.stats import chisquare, kruskal
import statsmodels.api as sm
import scipy.stats as stats
```

In [4]:

```
df=pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/003/549/original/logistic_regression.csv?1651045921')
```

In [5]:

```
df.shape
```

Out[5]:

```
(396030, 27)
```

In [6]:

```
df.head()
```

Out[6]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	op
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	...	
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	...	
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	...	
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	...	
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	...	

5 rows × 27 columns

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  object 
 2   int_rate              396030 non-null  float64
 3   installment           396030 non-null  float64
 4   grade                 396030 non-null  object 
 5   sub_grade             396030 non-null  object 
 6   emp_title             373103 non-null  object 
 7   emp_length            377729 non-null  object 
 8   home_ownership        396030 non-null  object 
 9   annual_inc            396030 non-null  float64
 10  verification_status   396030 non-null  object 
 11  issue_d               396030 non-null  object 
 12  loan_status           396030 non-null  object 
 13  purpose               396030 non-null  object 
 14  title                 394275 non-null  object 
 15  dti                   396030 non-null  float64
 16  earliest_cr_line      396030 non-null  object 
 17  open_acc              396030 non-null  float64
 18  pub_rec               396030 non-null  float64
 19  revol_bal             396030 non-null  float64
 20  revol_util            395754 non-null  float64
 21  total_acc             396030 non-null  float64
 22  initial_list_status   396030 non-null  object 
 23  application_type      396030 non-null  object 
 24  mort_acc              358235 non-null  float64
 25  pub_rec_bankruptcies  395495 non-null  float64
 26  address               396030 non-null  object 
dtypes: float64(12), object(15)
memory usage: 81.6+ MB
```

2.1.Duplicate Value Check

In [8]:

```
df.duplicated().sum()
```

Out[8]:

0

In [9]:

```
df.describe()
```

Out[9]:

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_b
count	396030.000000	396030.000000	396030.000000	3.960300e+05	396030.000000	396030.000000	396030.000000	3.960300e+(
mean	14113.888089	13.639400	431.849698	7.420318e+04	17.379514	11.311153	0.178191	1.584454e+(
std	8357.441341	4.472157	250.727790	6.163762e+04	18.019092	5.137649	0.530671	2.059184e+(
min	500.000000	5.320000	16.080000	0.000000e+00	0.000000	0.000000	0.000000	0.000000e+(
25%	8000.000000	10.490000	250.330000	4.500000e+04	11.280000	8.000000	0.000000	6.025000e+(
50%	12000.000000	13.330000	375.430000	6.400000e+04	16.910000	10.000000	0.000000	1.118100e+(
75%	20000.000000	16.490000	567.300000	9.000000e+04	22.980000	14.000000	0.000000	1.962000e+(
max	40000.000000	30.990000	1533.810000	8.706582e+06	9999.000000	90.000000	86.000000	1.743266e+(

1.2.Conversion of categorical attributes to 'category'

In [10]:

```
columns=df.select_dtypes(include=['object']).columns
for column in columns:
    print(f'column name :{column}')
    print(df[column].value_counts(ascending=True))
    print('..............................................................................
.............................')
    print('..............................................................................
.............................')
```

```
column name :term
 60 months     94025
 36 months    302005
Name: term, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :grade
G      3054
F     11772
E     31488
D     63524
A     64187
C    105987
B    116018
Name: grade, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :sub_grade
G5      316
G4      374
G3      552
G2      754
G1     1058
F5     1397
F4     1787
F3     2286
F2     2766
F1     3536
E5     4572
E4     5361
E3     6207
E2     7431
E1     7917
A2     9567
D5     9700
A1     9729
A3    10576
D4    11657
D3    12223
D2    13951
A4    15789
D1    15993
C5    18244
A5    18526
B1    19182
C4    20280
C3    21221
B5    22085
B2    22495
C2    22580
C1    23662
B4    25601
B3    26655
Name: sub_grade, dtype: int64
```

```
Name: sub_grade, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :emp_title
analytic33                                 1
Odin fashion                               1
Aztec Animal Clinic                        1
MCCS                                       1
Hand Surgery & Rehabilitation Center       1
                                        ...
Supervisor                              1830
RN                                      1846
Registered Nurse                        1856
Manager                                 4250
Teacher                                 4389
Name: emp_title, Length: 173105, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :emp_length
9 years       15314
8 years       19168
7 years       20819
6 years       20841
4 years       23952
1 year        25882
5 years       26495
3 years       31665
< 1 year      31725
2 years       35827
10+ years    126041
Name: emp_length, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :home_ownership
ANY              3
NONE            31
OTHER          112
OWN          37746
RENT        159790
MORTGAGE    198348
Name: home_ownership, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :verification_status
Not Verified       125082
Source Verified    131385
Verified           139563
Name: verification_status, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :issue_d
Jun-2007        1
Sep-2007       15
Nov-2007       22
Sep-2008       25
Jul-2007       26
            ...
Nov-2013    10496
Dec-2013    10618
Jan-2015    11705
Jul-2014    12609
Oct-2014    14846
Name: issue_d, Length: 115, dtype: int64
```

```
Name: issue_d, Length: 115, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :loan_status
Charged Off     77673
Fully Paid     318357
Name: loan_status, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :purpose
educational              257
renewable_energy         329
wedding                 1812
house                   2201
vacation                2452
moving                  2854
medical                 4196
car                     4697
small_business          5701
major_purchase          8790
other                  21185
home_improvement       24030
credit_card            83019
debt_consolidation    234507
Name: purpose, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :title
Appliances Falling Apart         1
Wedding1                         1
Finance Relief                   1
kevins loan                      1
cridit payoff                    1
                             ...
Debt Consolidation           11608
Other                        12930
Home improvement             15264
Credit card refinancing      51487
Debt consolidation          152472
Name: title, Length: 48817, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :earliest_cr_line
Aug-1959       1
Nov-1957       1
Jul-1958       1
Apr-1960       1
Dec-1950       1
            ...
Nov-2000    2736
Aug-2001    2884
Oct-2001    2896
Aug-2000    2935
Oct-2000    3017
Name: earliest_cr_line, Length: 684, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :initial_list_status
w    157964
f    238066
Name: initial_list_status, dtype: int64
.........................................................................................
..................
```

```
..................
.........................................................................................
..................
column name :application_type
DIRECT_PAY        286
JOINT             425
INDIVIDUAL     395319
Name: application_type, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
column name :address
0174 Michelle Gateway\r\nMendozaberg, OK 22690         1
01246 Carrie Passage\r\nNew Kyle, ND 11650             1
16658 Perez Key\r\nCampbellside, AL 70466              1
110 Gomez Flat Apt. 561\r\nJosephchester, RI 70466     1
27963 Jessica Lodge Suite 200\r\nDunntown, CT 70466    1
                                                      ..
USNS Johnson\r\nFPO AP 48052                           7
USS Smith\r\nFPO AP 70466                              8
USS Johnson\r\nFPO AE 48052                            8
USNS Johnson\r\nFPO AE 05113                           8
USCGC Smith\r\nFPO AE 70466                            8
Name: address, Length: 393700, dtype: int64
.........................................................................................
..................
.........................................................................................
..................
```

Treatment of column 'term'

In [11]:

```
df['term']=df['term'].str.split(' ',expand=True)[1]
```

In [12]:

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

Out[12]:

```
36    302005
60     94025
Name: term, dtype: int64
```

In [13]:

```
df.rename(columns = {'term':'term (months)'}, inplace = True)
```

In [14]:

```
df.head()
```

Out[14]:

	loan_amnt	term (months)	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	c
0	10000.0	36	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	...	
1	8000.0	36	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	...	
2	15600.0	36	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	...	
3	7200.0	36	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	...	

Advocate

	loan_amnt	term (months)	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	c
4	24375.0	60	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	...	

5 rows × 27 columns

Treatment of column 'address'

In [15]:

```
df['address']=df['address'].str[-8:-6]
```

In [16]:

```
df['address'].value_counts()
```

Out[16]:

```
AP    14308
AE    14157
AA    13919
NJ     7091
WI     7081
LA     7068
NV     7038
AK     7034
MA     7022
VA     7022
VT     7005
NY     7004
MS     7003
TX     7000
SC     6973
ME     6972
AR     6969
OH     6969
GA     6967
ID     6958
IN     6958
KS     6945
WV     6944
RI     6940
MO     6939
IL     6934
WY     6933
NE     6927
HI     6927
IA     6926
FL     6921
AZ     6918
CO     6914
OK     6911
CT     6904
MN     6904
NC     6901
OR     6898
CA     6898
AL     6898
MD     6896
WA     6895
UT     6887
SD     6887
MT     6883
DE     6874
TN     6869
ND     6858
MI     6854
DC     6842
```

```
NM      6842
PA      6825
NH      6818
KY      6800
Name: address, dtype: int64
```

Treatment of column 'emp_length'

In [17]:

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

In [18]:

```
df['emp_length'].replace('10+ years','10 years', inplace=True)
```

In [19]:

```
df['emp_length']=df['emp_length'].str.split(' ',expand=True)[0]
```

In [20]:

```
df.rename(columns = {'emp_length':'emp_length (years)'}, inplace = True)
```

In [21]:

```
df.head()
```

Out[21]:

	loan_amnt	term (months)	int_rate	installment	grade	sub_grade	emp_title	emp_length (years)	home_ownership	annual_inc	...	c
0	10000.0	36	11.44	329.48	B	B4	Marketing	10	RENT	117000.0	...	
1	8000.0	36	11.99	265.68	B	B5	Credit analyst	4	MORTGAGE	65000.0	...	
2	15600.0	36	10.49	506.97	B	B3	Statistician	0	RENT	43057.0	...	
3	7200.0	36	6.49	220.65	A	A2	Client Advocate	6	RENT	54000.0	...	
4	24375.0	60	17.27	609.33	C	C5	Destiny Management Inc.	9	MORTGAGE	55000.0	...	

5 rows × 27 columns

2.2.Missing value treatment

In [22]:

```
df.isna().sum()
```

Out[22]:

```
loan_amnt                 0
term (months)             0
int_rate                  0
installment               0
grade                     0
sub_grade                 0
emp_title             22927
emp_length (years)    18301
home_ownership            0
annual_inc                0
verification_status       0
issue_d                   0
loan_status               0
```

```
purpose                       0
title                      1755
dti                           0
earliest_cr_line              0
open_acc                      0
pub_rec                       0
revol_bal                     0
revol_util                  276
total_acc                     0
initial_list_status           0
application_type              0
mort_acc                  37795
pub_rec_bankruptcies        535
address                       0
dtype: int64
```

Droping rows with null value for 'pub_rec_bankruptcies', 'revol_util' & 'title' as they are less than 1% of the data

In [23]:

```
df = df.dropna(axis=0, subset=['pub_rec_bankruptcies','revol_util','title'])
```

In [24]:

```
df.isna().sum()
```

Out[24]:

```
loan_amnt                     0
term (months)                 0
int_rate                      0
installment                   0
grade                         0
sub_grade                     0
emp_title                 22668
emp_length (years)        18076
home_ownership                0
annual_inc                    0
verification_status           0
issue_d                       0
loan_status                   0
purpose                       0
title                         0
dti                           0
earliest_cr_line              0
open_acc                      0
pub_rec                       0
revol_bal                     0
revol_util                    0
total_acc                     0
initial_list_status           0
application_type              0
mort_acc                  37195
pub_rec_bankruptcies          0
address                       0
dtype: int64
```

Imputing 'mort_acc' by linear interpolation

In [25]:

```
df['mort_acc'] = df['mort_acc'].interpolate(method = 'linear',    limit_direction = 'for
ward', axis = 0)
```

In [26]:

```
df.isna().sum()
```

Out[26]:

```
loan_amnt                  0
term (months)              0
int_rate                   0
installment                0
grade                      0
sub_grade                  0
emp_title              22668
emp_length (years)     18076
home_ownership             0
annual_inc                 0
verification_status        0
issue_d                    0
loan_status                0
purpose                    0
title                      0
dti                        0
earliest_cr_line           0
open_acc                   0
pub_rec                    0
revol_bal                  0
revol_util                 0
total_acc                  0
initial_list_status        0
application_type           0
mort_acc                   0
pub_rec_bankruptcies       0
address                    0
dtype: int64
```

Imputing 'emp_length (years)' & 'emp_title' by forward fill by making dataFrame in order by 'loan_amnt','annual_inc', 'installment','grade','sub_grade'

In [27]:

```
df = df.sort_values(by = ['loan_amnt','annual_inc', 'installment','grade','sub_grade'])
```

In [28]:

```
df.fillna(method="ffill",inplace=True)
```

In [29]:

```
df.isna().sum()
```

Out[29]:

```
loan_amnt              0
term (months)          0
int_rate               0
installment            0
grade                  0
sub_grade              0
emp_title              0
emp_length (years)     0
home_ownership         0
annual_inc             0
verification_status    0
issue_d                0
loan_status            0
purpose                0
title                  0
dti                    0
earliest_cr_line       0
open_acc               0
pub_rec                0
revol_bal              0
revol_util             0
total_acc              0
initial_list_status    0
application_type       0
```

```
mort_acc                0
pub_rec_bankruptcies    0
address                 0
dtype: int64
```

In [30]:

```
# shuffle the DataFrame rows
df = df.sample(frac = 1)
```

In [31]:

```
df.reset_index(drop=True)
```

Out[31]:

	loan_amnt	term (months)	int_rate	installment	grade	sub_grade	emp_title	emp_length (years)	home_ownership	annual_in
0	5000.0	36	15.61	174.83	D	D1	Instructor	2	RENT	26000.(
1	12575.0	36	13.33	425.71	C	C3	Computer Network Engineer	10	RENT	127900.(
2	5125.0	36	16.78	182.16	C	C5	Administrative Assistant	10	MORTGAGE	44316.(
3	8000.0	36	14.09	273.78	B	B5	Indiana University Health	4	RENT	105000.(
4	9600.0	36	10.99	314.25	B	B3	General Manager	10	MORTGAGE	82940.(
...	...	...	...	...	...	...	...	...	...	..
393460	8825.0	36	15.61	308.57	D	D1	Psych Tech	10	MORTGAGE	26000.(
393461	1000.0	36	11.99	33.21	B	B3	Designer	4	OWN	30000.(
393462	6000.0	36	12.69	201.27	C	C2	Vice President, Human Resources	10	MORTGAGE	115000.(
393463	18000.0	36	12.99	606.41	B	B5	Sr. Account Mgr	10	MORTGAGE	170000.(
393464	18000.0	36	11.14	590.50	B	B2	Del Mar College	6	MORTGAGE	73000.(

393465 rows × 27 columns

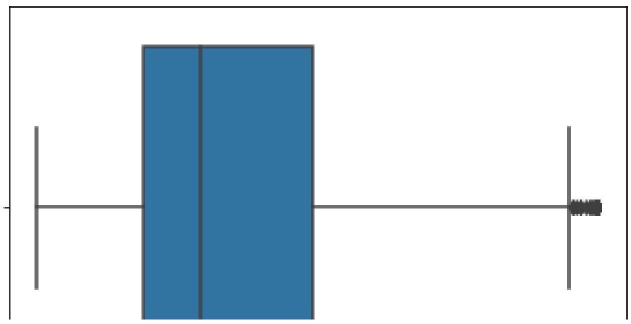
1.3.Univariate Analysis

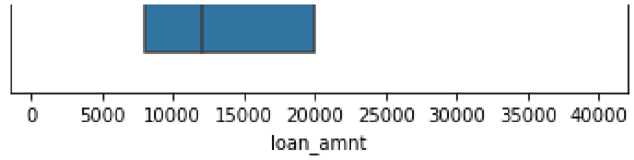
In [32]:

```
sns.boxplot(x=df['loan_amnt'])
```

Out[32]:

<AxesSubplot:xlabel='loan_amnt'>





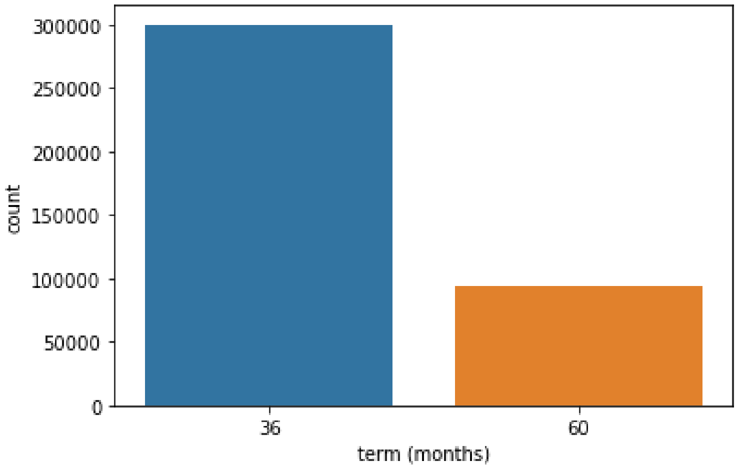
Median Loan amount is 12000 and distribution is right skewed

In [33]:

```
sns.countplot(x=df['term (months)'])
```

Out[33]:

```
<AxesSubplot:xlabel='term (months)', ylabel='count'>
```



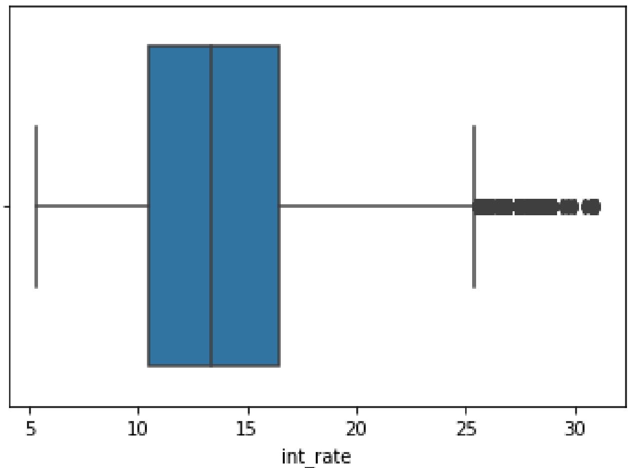
Around 75% of loans are having 3 Years of term

In [34]:

```
sns.boxplot(x=df['int_rate'])
```

Out[34]:

```
<AxesSubplot:xlabel='int_rate'>
```



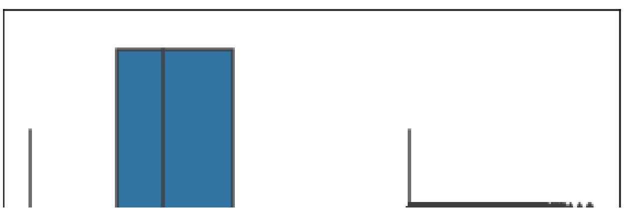
Median interest rate is close to 13.5 and Distribution is rigt skewed

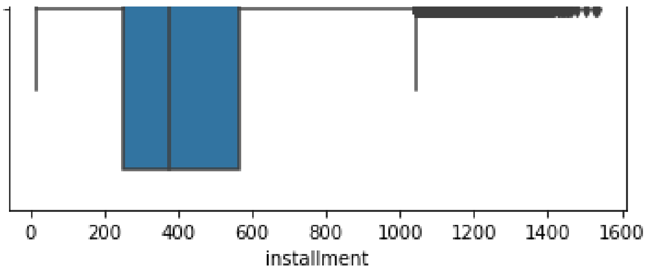
In [35]:

```
sns.boxplot(x=df['installment'])
```

Out[35]:

```
<AxesSubplot:xlabel='installment'>
```





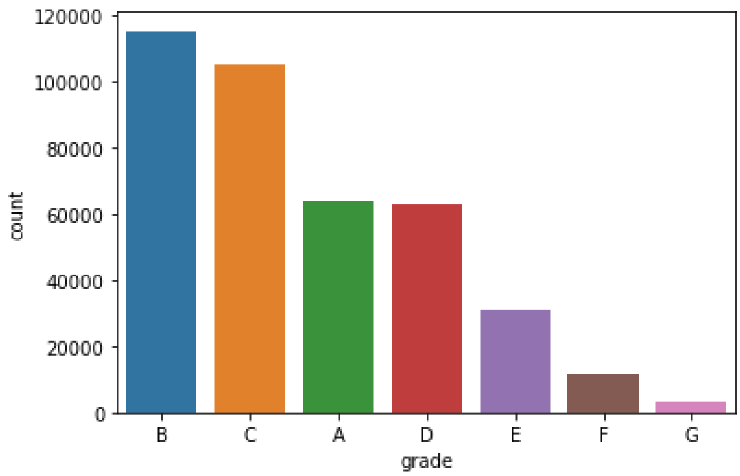
Monthly median installment amount is close to 350 and Distribution is right skewed

In [36]:

```
sns.countplot(x=df['grade'],order=df['grade'].value_counts().iloc[:7].index)
```

Out[36]:

```
<AxesSubplot:xlabel='grade', ylabel='count'>
```



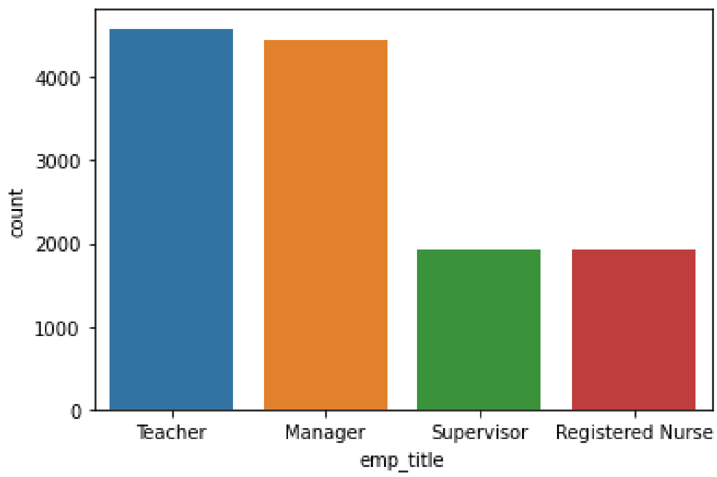
Maximum loan are graded as B category

In [37]:

```
sns.countplot(x=df['emp_title'],order=df['emp_title'].value_counts().iloc[:4].index)
```

Out[37]:

```
<AxesSubplot:xlabel='emp_title', ylabel='count'>
```



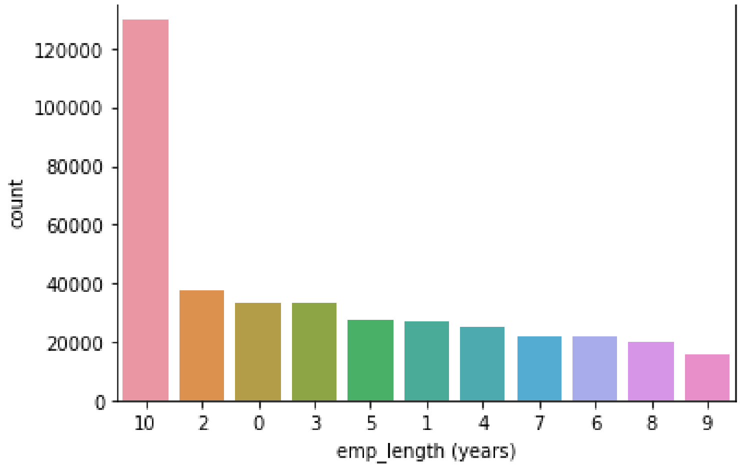
Teacher, Manager and Supervisor are amonst the top 3 employee title

In [38]:

```
sns.countplot(x=df['emp_length (years)'],order=df['emp_length (years)'].value_counts().i
loc[:11].index)
```

Out[38]:

```
<AxesSubplot:xlabel='emp_length (years)', ylabel='count'>
```



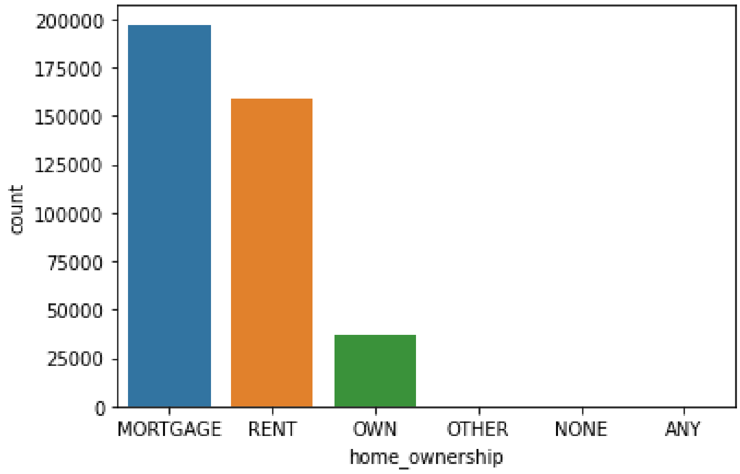
Around 30% of Loans are given to employee with 10+years working lenght

In [39]:

```
sns.countplot(x=df['home_ownership'],order=df['home_ownership'].value_counts().iloc[:6].
index)
```

Out[39]:

```
<AxesSubplot:xlabel='home_ownership', ylabel='count'>
```

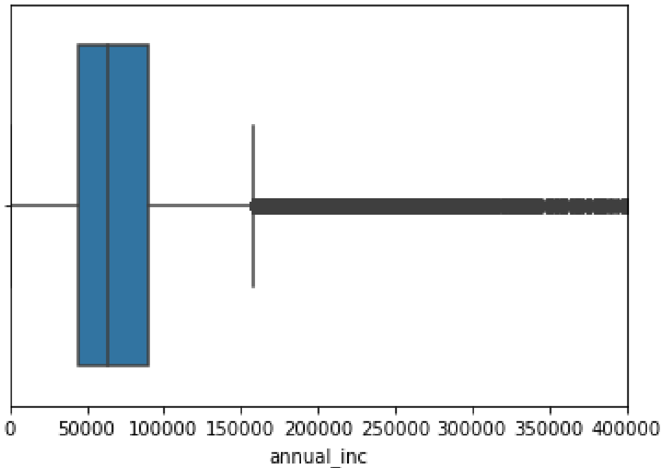


Around 50% of home ownership is MORTGAGE

In [40]:

```
ax=sns.boxplot(x=df['annual_inc'])

plt.xlim([0, 400000])
plt.show()
```



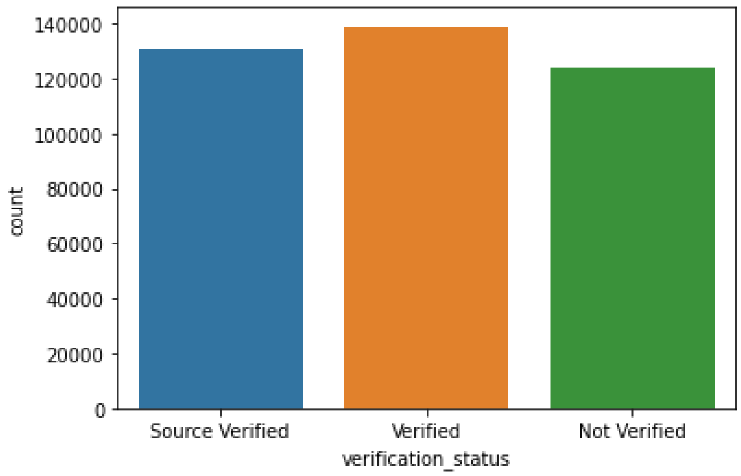
Mean annual income is around 65000

In [41]:

```
sns.countplot(x=df['verification_status'])
```

Out[41]:

<AxesSubplot:xlabel='verification_status', ylabel='count'>



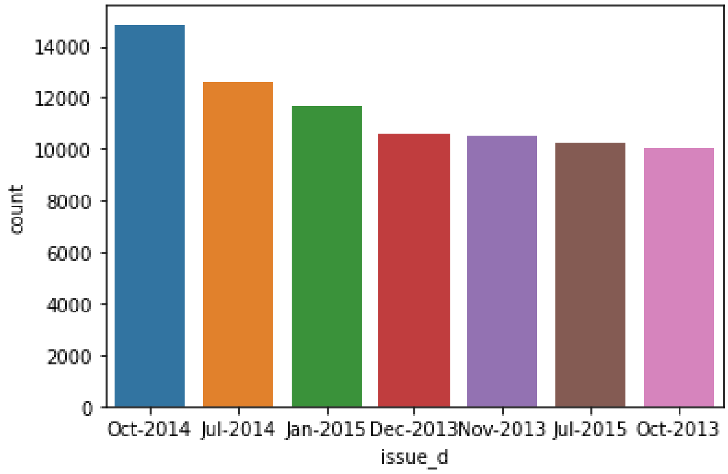
Not much difference in count of verification status

In [42]:

```
sns.countplot(x=df['issue_d'],order=df['issue_d'].value_counts().iloc[:7].index)
```

Out[42]:

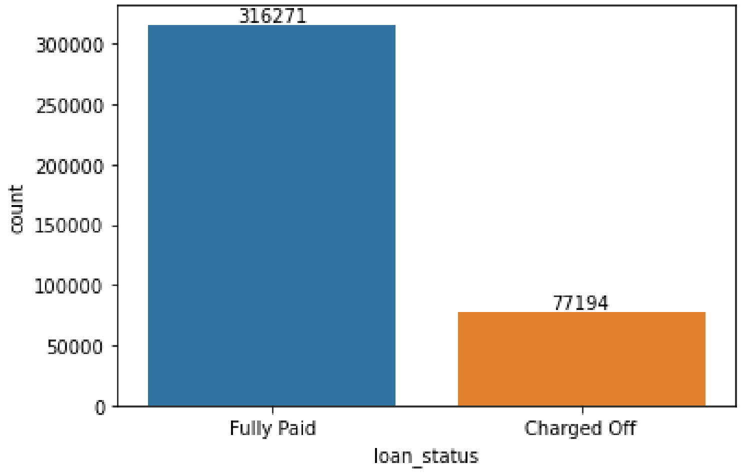
<AxesSubplot:xlabel='issue_d', ylabel='count'>



As per this Dataset, Maximum loan has been issued on oct-2014

In [43]:

```
ax=sns.countplot(x=df['loan_status'])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```



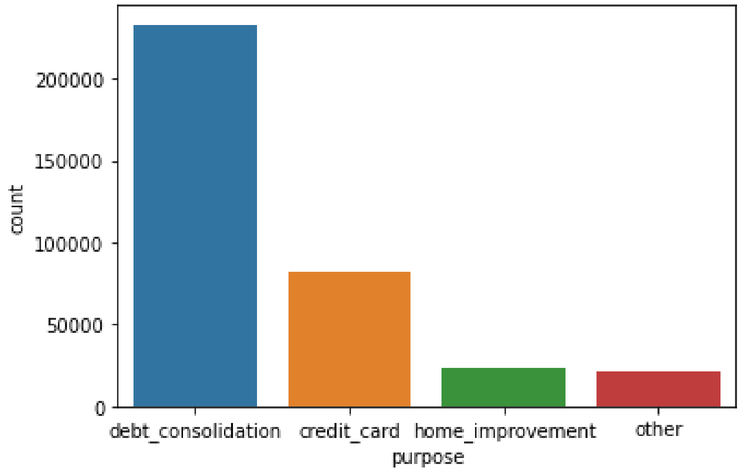
Around 80% of Loan has been Paid fully

In [44]:

```
sns.countplot(x=df['purpose'],order=df['purpose'].value_counts().iloc[:4].index)
```

Out[44]:

<AxesSubplot:xlabel='purpose', ylabel='count'>



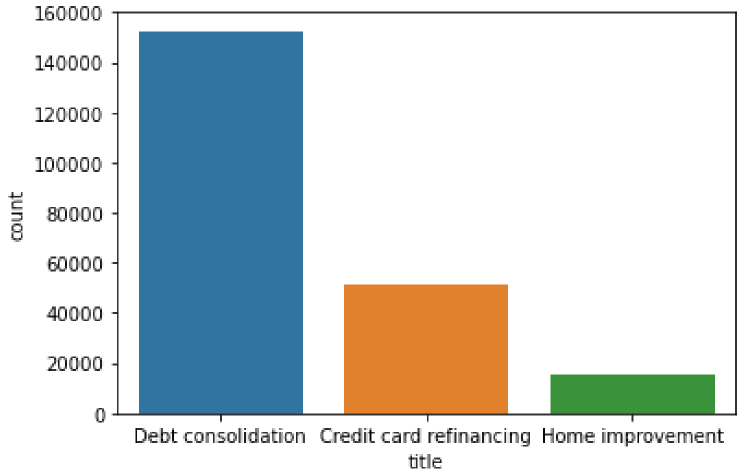
Debt consolidation and Credit card are the major purpose for loans

In [45]:

```
sns.countplot(x=df['title'],order=df['title'].value_counts().iloc[:3].index)
```

Out[45]:

<AxesSubplot:xlabel='title', ylabel='count'>

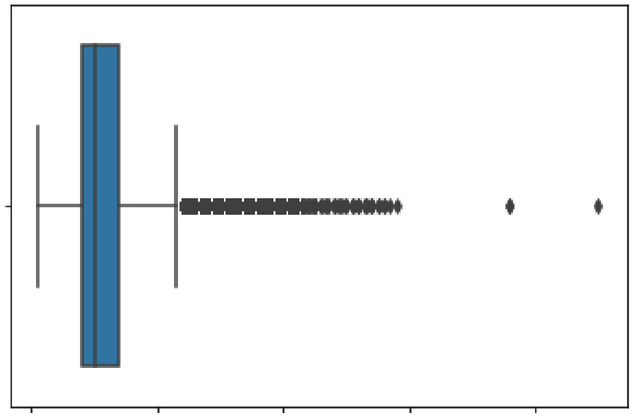


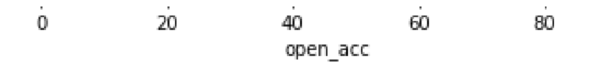
In [46]:

```
sns.boxplot(x=df['open_acc'])
```

Out[46]:

<AxesSubplot:xlabel='open_acc'>



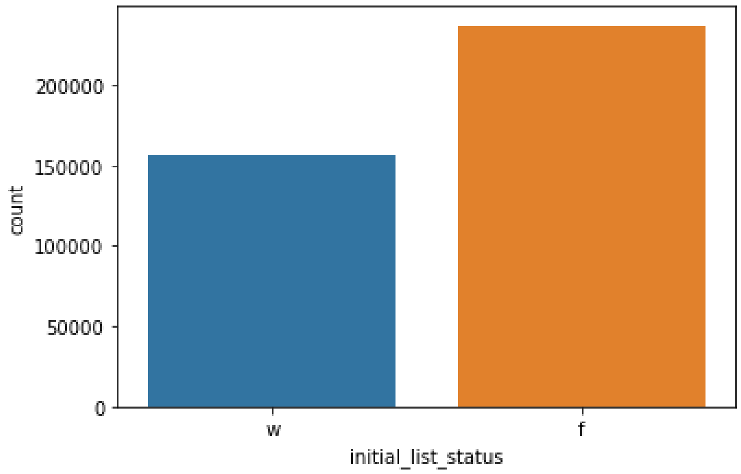


In [47]:

```
sns.countplot(x=df['initial_list_status'])
```

Out[47]:

<AxesSubplot:xlabel='initial_list_status', ylabel='count'>

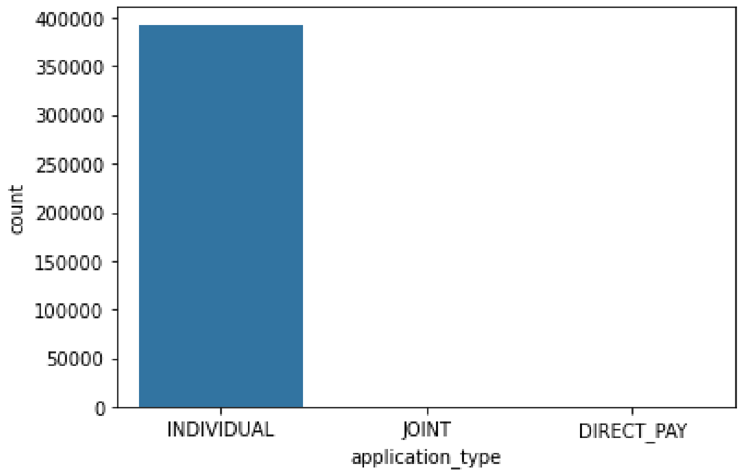


In [48]:

```
sns.countplot(x=df['application_type'])
```

Out[48]:

<AxesSubplot:xlabel='application_type', ylabel='count'>



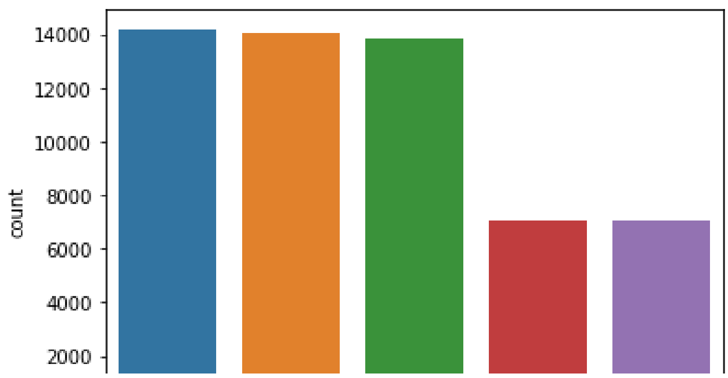
Almost all the loan is falling in Individual Category

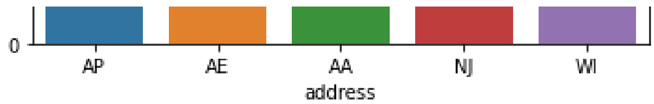
In [49]:

```
sns.countplot(x=df['address'],order=df['address'].value_counts().iloc[:5].index)
```

Out[49]:

<AxesSubplot:xlabel='address', ylabel='count'>





Top 3 city of the borrower are AP,AE & AA

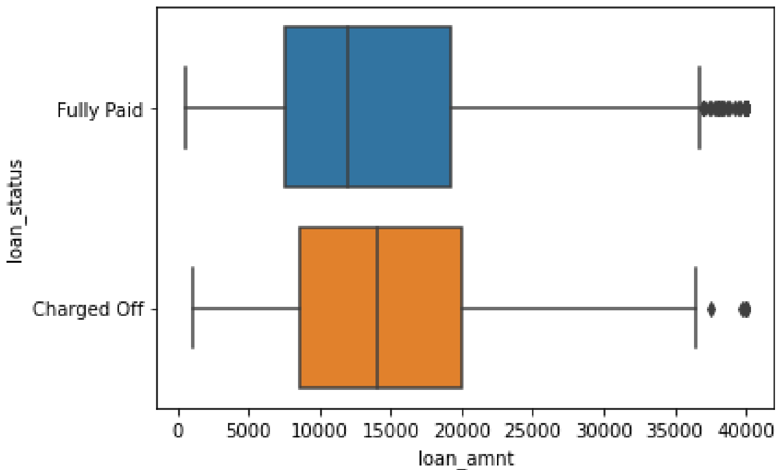
1.4.Bivariate Analysis

In [50]:

```
sns.boxplot(x=df['loan_amnt'], y=df['loan_status'])
```

Out[50]:

<AxesSubplot:xlabel='loan_amnt', ylabel='loan_status'>



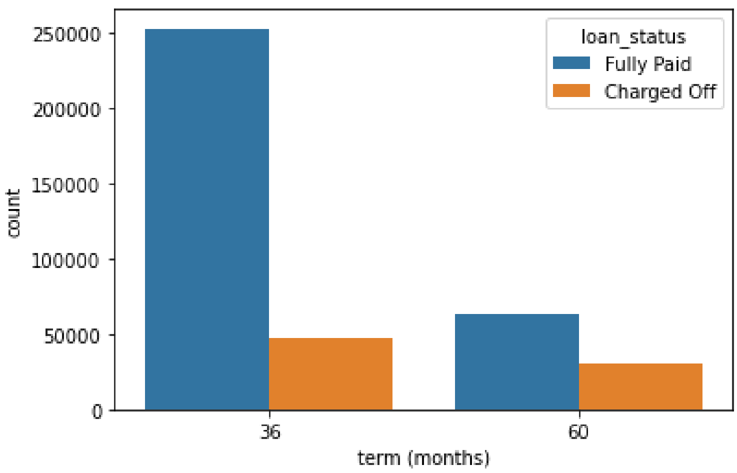
Median Loan_amount of Fully_Paid loans is less tha Charged_off loans

In [51]:

```
sns.countplot(x=df['term (months)'],hue=df['loan_status'])
```

Out[51]:

<AxesSubplot:xlabel='term (months)', ylabel='count'>

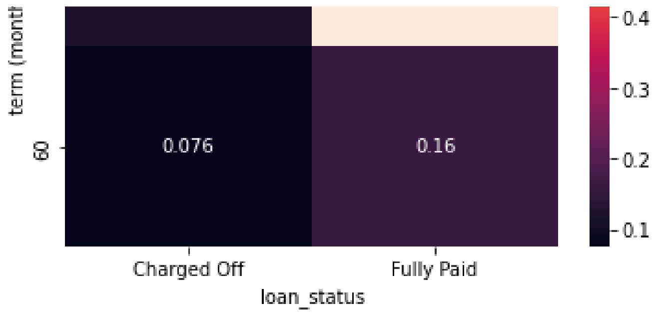


In [52]:

```
sns.heatmap(data=pd.crosstab(df['term (months)'],df['loan_status'],normalize=True),annot
=True)
```

Out[52]:

<AxesSubplot:xlabel='loan_status', ylabel='term (months)'>



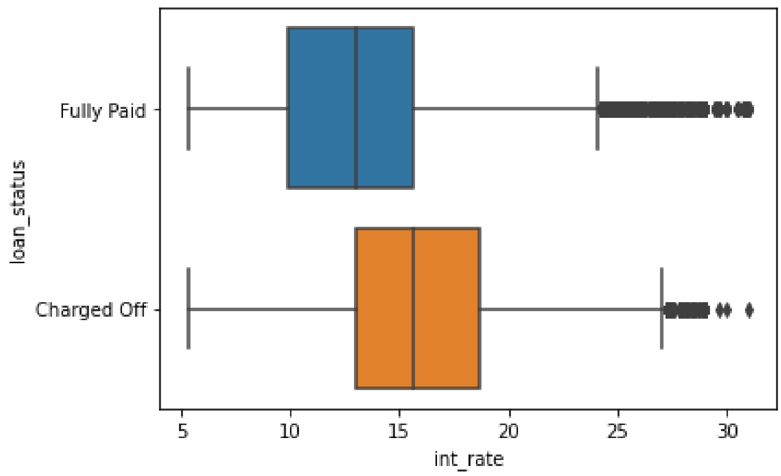
The probability of full_loan_payment when term is 30 months is high as compared to 60 months

In [53]:

```
sns.boxplot(x=df['int_rate'], y=df['loan_status'])
```

Out[53]:

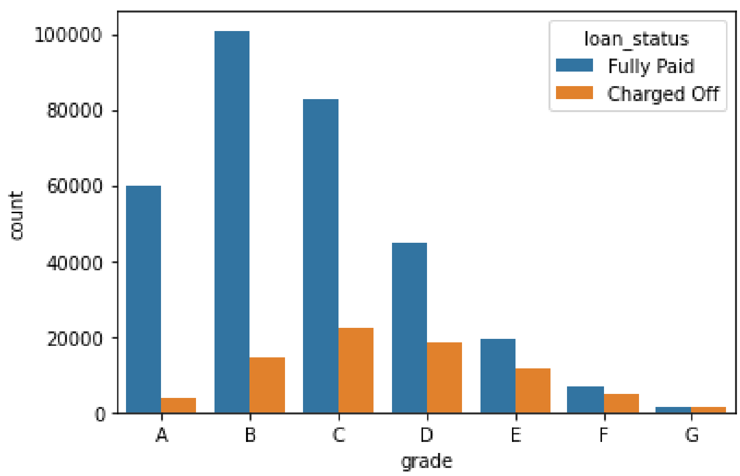
```
<AxesSubplot:xlabel='int_rate', ylabel='loan_status'>
```



Median Interest rate of fully_paid loan is less than Charged_off loans

In [54]:

```
g=sns.countplot(x=df['grade'],hue=df['loan_status'],order=['A','B','C','D','E','F','G'])
```

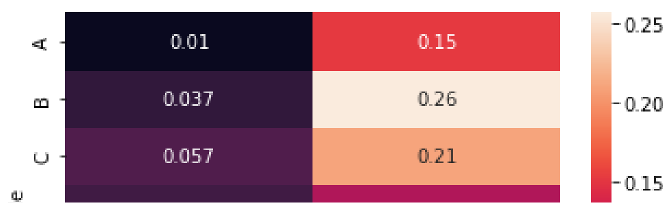


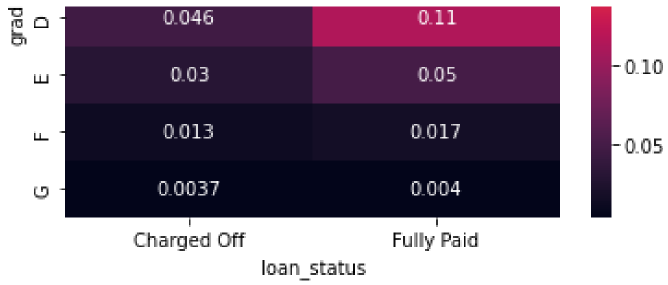
In [55]:

```
sns.heatmap(data=pd.crosstab(df['grade'],df['loan_status'],normalize=True),annot=True)
```

Out[55]:

```
<AxesSubplot:xlabel='loan_status', ylabel='grade'>
```





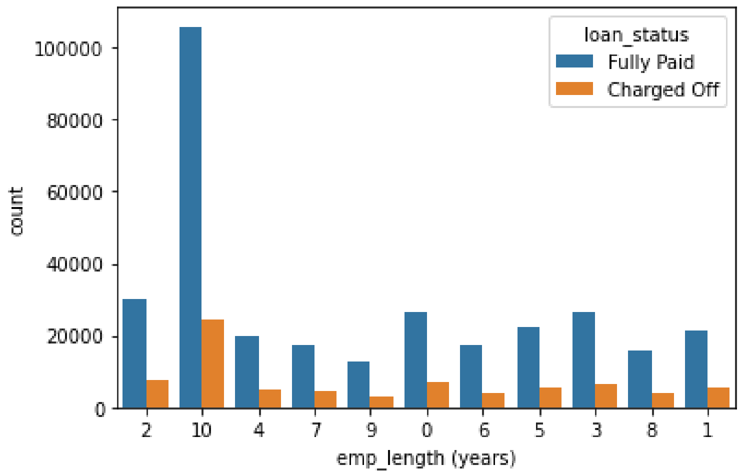
Probability of fully_paid for grade A is 93.75% followed grade B 87.54%

In [56]:

```
sns.countplot(x=df['emp_length (years)'],hue=df['loan_status'])
```

Out[56]:

<AxesSubplot:xlabel='emp_length (years)', ylabel='count'>

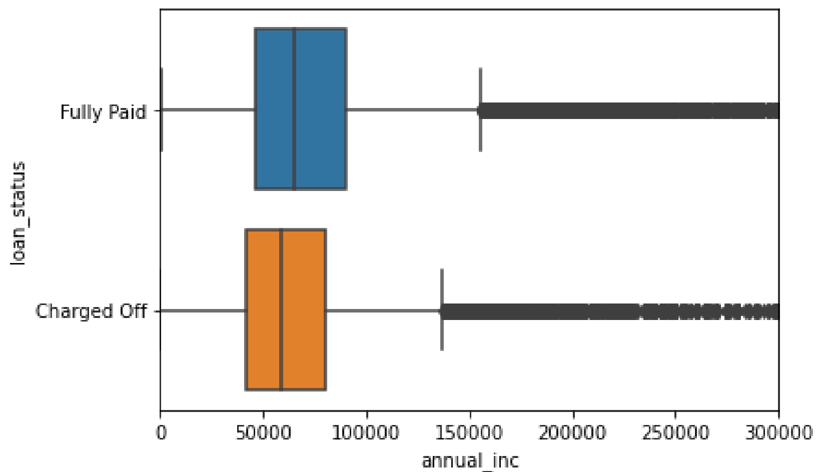


Ratio of fully_paid to charged_off is almost similar for different emp_lenghts

In [57]:

```
ax=sns.boxplot(x=df['annual_inc'], y=df['loan_status'])

plt.xlim([0, 300000])
plt.show()
```



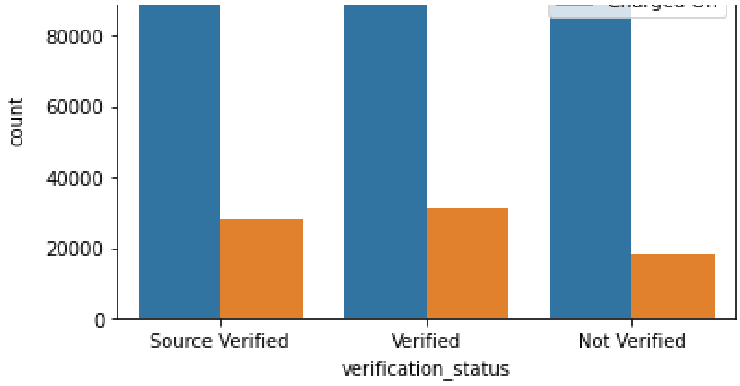
Median Annual income of the people who fully_paid loan is slightly high than people with Charged_off

In [58]:

```
sns.countplot(x=df['verification_status'],hue=df['loan_status'])
```

Out[58]:

<AxesSubplot:xlabel='verification_status', ylabel='count'>



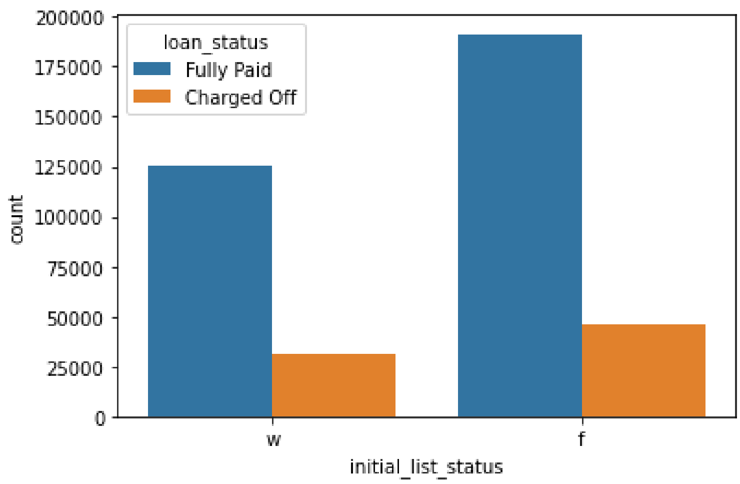
Loan Payment Status is independent with Verification status

In [59]:

```
sns.countplot(x=df['initial_list_status'],hue=df['loan_status'])
```

Out[59]:

```
<AxesSubplot:xlabel='initial_list_status', ylabel='count'>
```



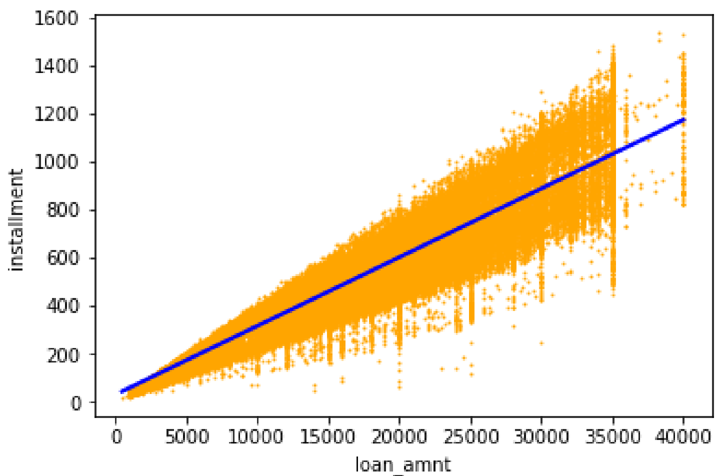
Probability of full repayment is slightly high in case of 'f' tha 'w'

In [60]:

```
sns.regplot(x=df['loan_amnt'],y=df['installment'],fit_reg=True,scatter_kws={"color": "or
ange",'s':1}, line_kws={"color": "blue"})
```

Out[60]:

```
<AxesSubplot:xlabel='loan_amnt', ylabel='installment'>
```

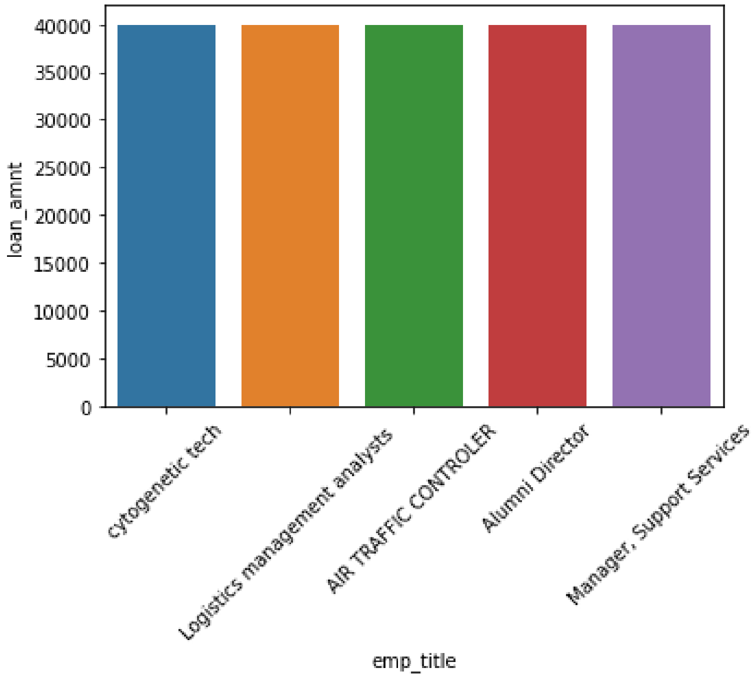


Installment is highly correlated with Loan_amount

In [61]:

```
df_emp_title=df.groupby('emp_title').mean('loan_amnt')
df_emp_title.reset_index(inplace=True)
df_emp_title=df_emp_title[['loan_amnt','emp_title']]
```

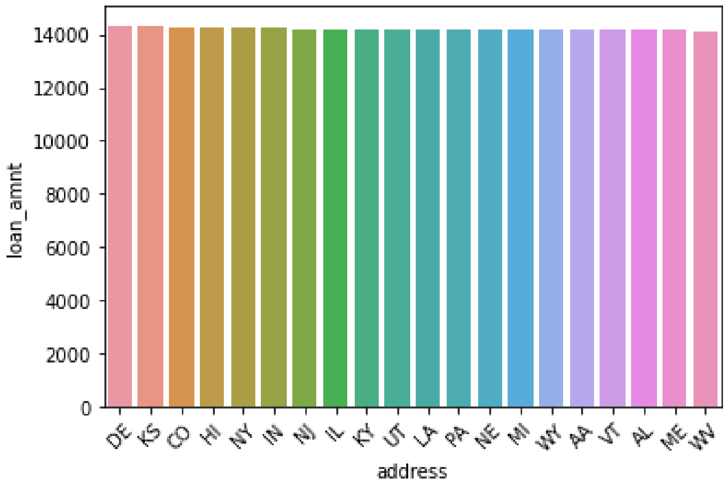
```
df_emp_title=df_emp_title.sort_values(by='loan_amnt',ascending=False)
df_emp_title.reset_index(drop=True,inplace=True)
sns.barplot(data=df_emp_title.iloc[:5,:],x='emp_title',y='loan_amnt')
plt.xticks(rotation=45)
plt.show()
```



cytogenetic tech, Logistic management analtsts are having hogh mean loan_amount

In [62]:

```
df_address=df.groupby('address').mean('loan_amnt')
df_address.reset_index(inplace=True)
df_address=df_address[['loan_amnt','address']]
df_address=df_address.sort_values(by='loan_amnt',ascending=False)
df_address.reset_index(drop=True,inplace=True)
sns.barplot(data=df_address.iloc[:20,:],x='address',y='loan_amnt')
plt.xticks(rotation=45)
plt.show()
```



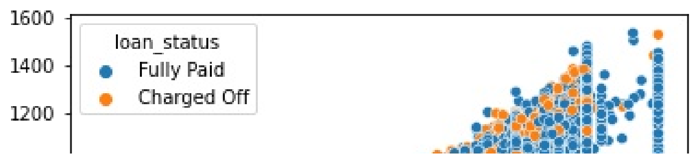
Not much difference in Mean_loan_amount with address

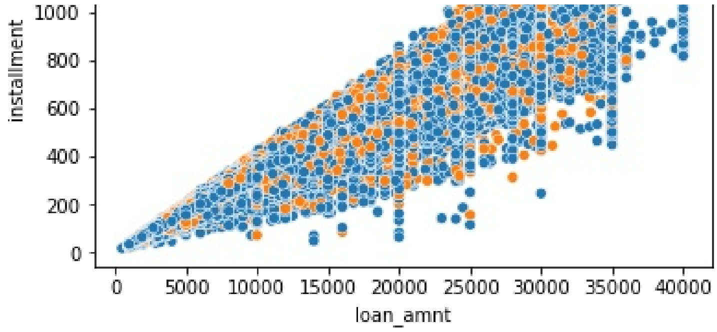
In [63]:

```
sns.scatterplot(x=df['loan_amnt'],y=df['installment'],hue=df['loan_status'])
```

Out[63]:

```
<AxesSubplot:xlabel='loan_amnt', ylabel='installment'>
```





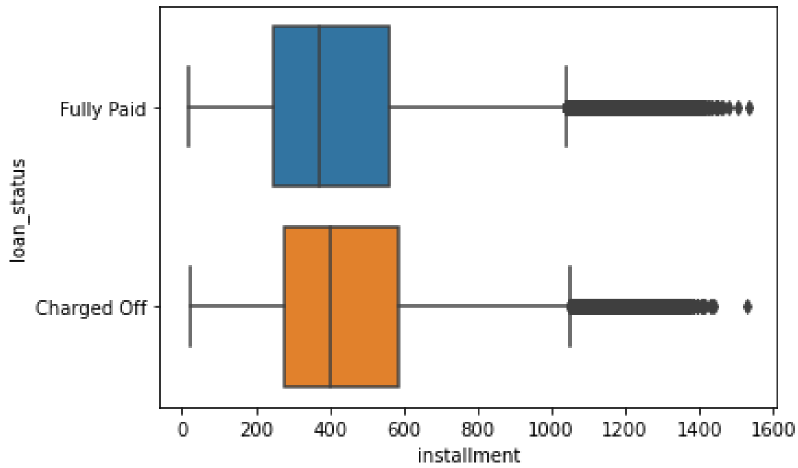
Fully_Paid and Charged_off are having similar trend of installment v loan_amount

In [64]:

```
sns.boxplot(x=df['installment'], y=df['loan_status'])
```

Out[64]:

```
<AxesSubplot:xlabel='installment', ylabel='loan_status'>
```



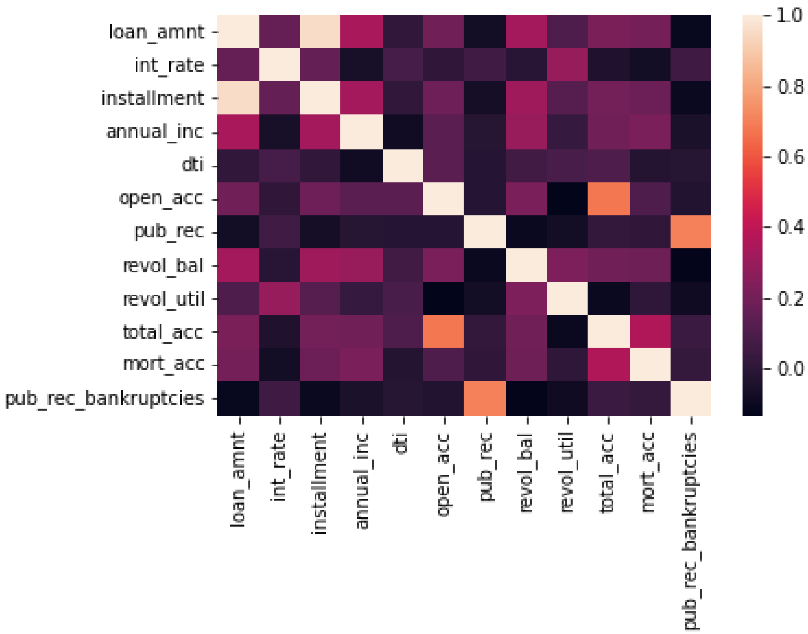
Median of installment for Fully Paid is slightly less than Charged_off

In [65]:

```
sns.heatmap(data=df.corr())
```

Out[65]:

```
<AxesSubplot:>
```



Loan_amt is Highly correlated with installment annual income is correlated with installment and Loan_amt total_acc is correlated with open_acc pub_rec is correlated with pub_rec_bankruptcies

2.3.Outlier treatment

We have already observed in univariate analysis that we have outliers in loan amount, income and intrestWe should be very mindful whenever dealing with outliers. There may be some higher or lower values in the data but at the same time, that could be the identity of the distribution. That's why I am not doing any treatment for outliers

2.4.Feature engineering

In [66]:

```
obj_cols = list(df.select_dtypes(include='object'))
obj_cols
```

Out[66]:

```
['term (months)',
 'grade',
 'sub_grade',
 'emp_title',
 'emp_length (years)',
 'home_ownership',
 'verification_status',
 'issue_d',
 'loan_status',
 'purpose',
 'title',
 'earliest_cr_line',
 'initial_list_status',
 'application_type',
 'address']
```

Converting 'term' and 'emp_length' to integer data type

In [67]:

```
df['term (months)']=df['term (months)'].astype(str).astype(int)
```

In [68]:

```
df['emp_length (years)']=df['emp_length (years)'].astype(str).astype(int)
```

Droping date columns 'issue_d' & 'earliest_cr_line'

In [69]:

```
df.drop(columns=['issue_d'], inplace = True, axis=1)
```

In [70]:

```
df.drop(columns=['earliest_cr_line'], inplace = True, axis=1)
```

Droping 'application_type' as it is having less variance(information) 99.9% are Individual

In [71]:

```
df.drop(columns=['application_type'], inplace = True, axis=1)
```

Converting Traget('loan_status'), Ordinal Categorical to Categorical Numerical & the columns having many unique values ('title','emp_title','address','purpose')

In [72]:

```
col=['grade','sub_grade','loan_status','initial_list_status','address','title','emp_title
','address','purpose']
```

In [73]:

```
for i in col:
    df[i]=df[i].astype('category')
    df[i]=df[i].cat.codes
```

In [74]:

```
obj_col = list(df.select_dtypes(include='object'))
obj_col
```

Out[74]:

```
['home_ownership', 'verification_status']
```

One Hot Encodring for 'home_ownership','verification_status'

In [75]:

```
df=pd.concat((df, pd.get_dummies(df['home_ownership'])),axis=1)
df.drop(columns=['home_ownership'], inplace = True, axis=1)
```

In [76]:

```
df=pd.concat((df, pd.get_dummies(df['verification_status'])),axis=1)
df.drop(columns=['verification_status'], inplace = True, axis=1)
```

In [77]:

```
df.head()
```

Out[77]:

	loan_amnt	term (months)	int_rate	installment	grade	sub_grade	emp_title	emp_length (years)	annual_inc	loan_status	...	addr
70590	5000.0	36	15.61	174.83	3	15	65802	2	26000.0	1	...	
86600	12575.0	36	13.33	425.71	2	12	30284	10	127900.0	0	...	
169233	5125.0	36	16.78	182.16	2	14	4832	10	44316.0	1	...	
125243	8000.0	36	14.09	273.78	1	9	64835	4	105000.0	1	...	
315825	9600.0	36	10.99	314.25	1	7	54989	10	82940.0	1	...	

5 rows × 31 columns

In [78]:

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

Out[78]:

```
1    316271
0     77194
Name: loan_status, dtype: int64
```

1: Fully Paid, 0: Charged Off

2.5.Data preparation for modeling

In [79]:

```
X=df.drop(columns=['loan_status'])
```

In [80]:

```
X.head()
```

Out[80]:

	loan_amnt	term (months)	int_rate	installment	grade	sub_grade	emp_title	emp_length (years)	annual_inc	purpose	...	address
70590	5000.0	36	15.61	174.83	3	15	65802	2	26000.0	9	...	20
86600	12575.0	36	13.33	425.71	2	12	30284	10	127900.0	2	...	8
169233	5125.0	36	16.78	182.16	2	14	4832	10	44316.0	9	...	0
125243	8000.0	36	14.09	273.78	1	9	64835	4	105000.0	1	...	29
315825	9600.0	36	10.99	314.25	1	7	54989	10	82940.0	2	...	49

5 rows × 30 columns

In [81]:

```
Y=df[['loan_status']]
```

In [82]:

```
Y.head()
```

Out[82]:

	loan_status
70590	1
86600	0
169233	1
125243	1
315825	1

3.1.Build the Logistic Regression model and comment on the model statistics

In [83]:

```
from sklearn.model_selection import train_test_split


X_train_val, X_test, y_train_val, y_test = train_test_split(X,Y,test_size=0.2, random_st
ate=42)

X_train, X_val, y_train, y_val = train_test_split(X_train_val,y_train_val,test_size=0.2,
random_state=42)
```

In [84]:

```
print(X_train.shape, y_train.shape)
print(X_val.shape, y_val.shape)
print(X_test.shape, y_test.shape)
```

```
(251817, 30) (251817, 1)
(62955, 30) (62955, 1)
(78693, 30) (78693, 1)
```

In [85]:

```
from sklearn.preprocessing import StandardScaler

st =  StandardScaler()

X_train_scaled = st.fit_transform(X_train.values)
X_val_scaled = st.transform(X_val.values)
X_test_scaled = st.transform(X_test.values)
```

In [86]:

```
y_train  = y_train.values[:,0]
y_val = y_val.values[:,0]
y_test = y_test.values[:,0]
```

In [87]:

```
print(X_train_scaled.shape, y_train.shape)
print(X_val_scaled.shape, y_val.shape)
print(X_test_scaled.shape, y_test.shape)
```

```
(251817, 30) (251817,)
(62955, 30) (62955,)
(78693, 30) (78693,)
```

In [88]:

```
# import the class
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

acc=[]
for i in [0.001, 0.01, 0.1, 1, 10, 100, 1000]:
    # instantiate the model (using the default parameters)
    logreg = LogisticRegression(random_state=16,C=i)

    # fit the model with data
    logreg.fit(X_train_scaled, y_train)

    y_val_pred = logreg.predict(X_val_scaled)

    accuracy_score_val=accuracy_score(y_val, y_val_pred)
    acc.append(accuracy_score_val)
```

In [89]:

```
y_val_pred.shape
```

Out[89]:

```
(62955,)
```

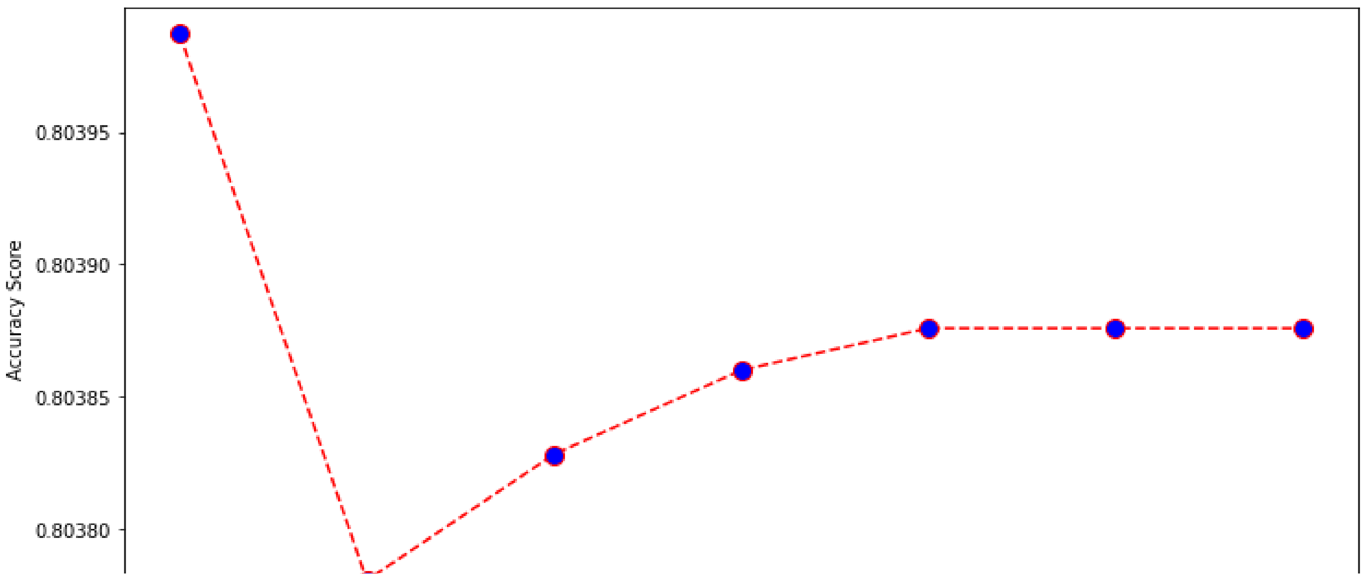
In [90]:

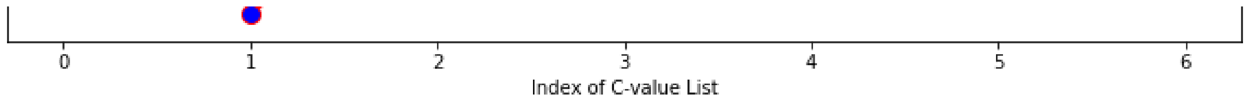
```
plt.figure(figsize=(12, 6))
plt.plot(range(len(acc)), acc, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)

plt.xlabel('Index of C-value List')
plt.ylabel('Accuracy Score')
```

Out[90]:

```
Text(0, 0.5, 'Accuracy Score')
```





AS maximum accuracy is at 1st index. Hence, We Can select HyperParameter C=0.001

In [99]:

```
logreg = LogisticRegression(random_state=16,C=0.001)

# fit the model with data
logreg.fit(X_train_scaled, y_train)

y_test_pred = logreg.predict(X_test_scaled)
```

3.2.Display model coefficients with column names

In [100]:

```
arr=logreg.coef_
arr
```

Out[100]:

```
array([[-0.03509063, -0.17477633,  0.04860077, -0.04057853, -0.13973434,
        -0.36622478, -0.08597907,  0.02357343,  0.19511168, -0.03241274,
         0.01403043, -0.20539446, -0.0995785 , -0.05081877,  0.04992771,
        -0.08436952,  0.08956595, -0.00944975,  0.07265704,  0.0184871 ,
        -0.00113695,  0.01019199,  0.05683804,  0.00463742, -0.00126526,
        -0.00430112, -0.055442  ,  0.04843686, -0.03513717, -0.01246613]])
```

In [101]:

```
logreg.intercept_
```

Out[101]:

```
array([1.57525194])
```

In [102]:

```
abs_weight_ls=[]
col=list(X.columns)
print("Column : Weight")
for i in range(len(col)):
    w=str(round(arr[0][i],2))
    x=col[i]
    print(f"{x} : {w}")
    abs_weight_ls.append([x,abs(float(w))])
```

```
Column : Weight
loan_amnt : -0.04
term (months) : -0.17
int_rate : 0.05
installment : -0.04
grade : -0.14
sub_grade : -0.37
emp_title : -0.09
emp_length (years) : 0.02
annual_inc : 0.2
purpose : -0.03
title : 0.01
dti : -0.21
open_acc : -0.1
pub_rec : -0.05
revol_bal : 0.05
revol_util : -0.08
total_acc : 0.09
initial_list_status : -0.01
mort_acc : 0.07
```

```
pub_rec_bankruptcies : 0.02
address : -0.0
ANY : 0.01
MORTGAGE : 0.06
NONE : 0.0
OTHER : -0.0
OWN : -0.0
RENT : -0.06
Not Verified : 0.05
Source Verified : -0.04
Verified : -0.01
```

In [103]:

```
abs_weight_ls.sort(key=lambda row:row[1],reverse=True)
```

In [107]:

```
abs_weight_ls[:10]
```

Out[107]:

```
[['sub_grade', 0.37],
 ['dti', 0.21],
 ['annual_inc', 0.2],
 ['term (months)', 0.17],
 ['grade', 0.14],
 ['open_acc', 0.1],
 ['emp_title', 0.09],
 ['total_acc', 0.09],
 ['revol_util', 0.08],
 ['mort_acc', 0.07]]
```

sub_grade, dti, annual_inc, term, grade, open_acc, total_acc, emp_title are most contributing Feature

4.1.ROC AUC Curve & comments

In [105]:

```
from sklearn.metrics import plot_roc_curve
```

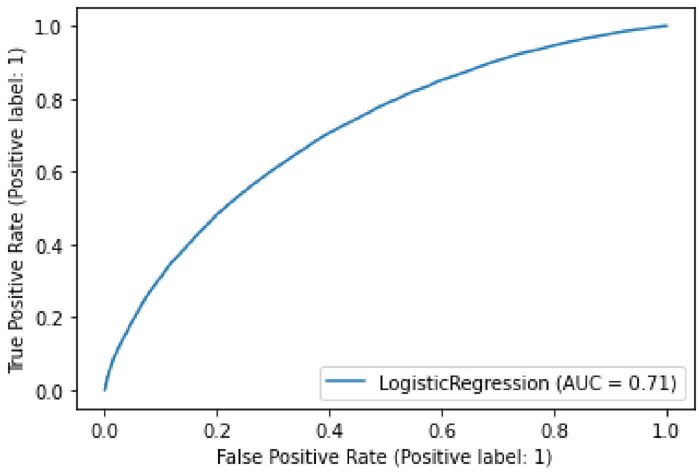
In [106]:

```
plot_roc_curve(logreg,X_test_scaled,y_test)
```

```
C:\Users\rahul.kumar\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureW
arning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metri
c.RocCurveDisplay.from_predictions` or :meth:`sklearn.metric.RocCurveDisplay.from_estimat
or`.
  warnings.warn(msg, category=FutureWarning)
```

Out[106]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x267303f9a90>
```



ROC AUC Score is 0.71, the reason for low roc-auc is imbalance data

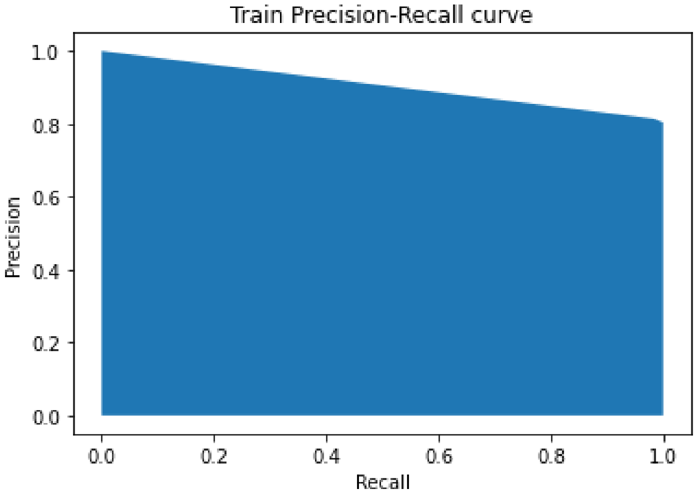
4.2.Precision Recall Curve & comments

In [108]:

```
from sklearn.metrics import precision_recall_curve
```

In [114]:

```
precision, recall, thresholds=precision_recall_curve(y_test,y_test_pred)
plt.fill_between(recall, precision)
plt.ylabel("Precision")
plt.xlabel("Recall")
plt.title("Train Precision-Recall curve");
```



we are getting 0.9 as Precision-Recall Value, which is suggesting good model

4.3.Classification Report (Confusion Matrix etc)

In [116]:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

In [117]:

```
# Calculate the accuracy
accuracy = accuracy_score(y_test, y_test_pred)

# Calculate the precision
precision = precision_score(y_test, y_test_pred)

# Calculate the recall
recall = recall_score(y_test, y_test_pred)

# Calculate the f1 score
f1 = f1_score(y_test, y_test_pred)

# Print the results
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Accuracy: 0.8065012135768111
Precision: 0.8139659545537745
Recall: 0.9842225910995178
F1 Score: 0.8910341274214439
```

In [120]:

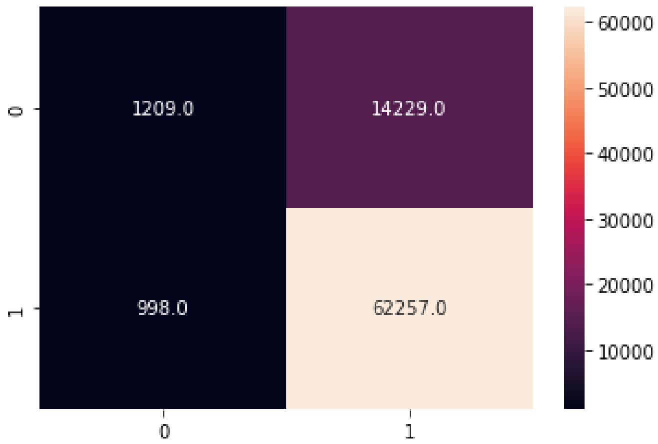
```
from sklearn.metrics import confusion_matrix
```

In [124]:

```python
# Create a confusion matrix
sns.heatmap(confusion_matrix(y_test, y_test_pred),annot=True, fmt=".1f")
```

Out[124]:

<AxesSubplot:>



4.4.1.How can we make sure that our model can detect real defaulters and there are less false positives? This is important as we can lose out on an opportunity to finance more individuals and earn interest on it.

As our recall for class 0 is 90%, we can detect it, if we want further improvement then we need to change the value of threshold of probablity > 0.5 (for true class)

4.4.2.Since NPA (non-performing asset) is a real problem in this industry, it's important we play safe and shouldn't disburse loans to anyone.

No, we should give loans but with minimal risk and not with 0 risk. If we take 0 risk, then no point of doing business in loan sector

4.5.Actionable Insights & Recommendations

1. sub_grade, dti, annual_inc, term, grade, open_acc, total_acc, emp_title are most contributing Feature. 2. Address is not contributing to the outcome. If we remove from feature, we will get slightly better performance time. 3. As the data is imbalance, Upscaling and Downscaling may lead to more accurate results. 4. Other Models like KNN or Decision Tree may improve the accuracy or F1. 5. We need to improve Recall for Class 0(not fully paid) to drive business with minimal risk.1.What percentage of customers have fully paid their Loan Amount? 80% 2.Comment about the correlation between Loan Amount and Installment features? Highly Co-relarelated 3.The majority of people have home ownership as _______? Mortgage 4.People with grades 'A' are more likely to fully pay their loan (T/F)? True 5.Name the top 2 afforded job titles? Teacher & Manager 6.Thinking from a bank's perspective, which metric should our primary focus be on ROC AUC Precision Recall F1 Score? F1 Score 7.How does the gap in precision and recall affect the bank? Recall sould be the most important criteria, spetially for class 0. 8.Which were the features that heavily affected the outcome? sub_grade,dti,annual_inc,term, grade,open_acc,total_acc, emp_title 9.Will the results be affected by geographical location? (Yes/No) No