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

import importlib

from importlib import reload

plt=reload(plt)

data=pd.read_csv('logistic_regression.csv')

data.head()

   loan_amnt        term  int_rate  installment grade sub_grade  \
0      10000   36 months     11.44       329.48     B        B4
1       8000   36 months     11.99       265.68     B        B5
2      15600   36 months     10.49       506.97     B        B3
3       7200   36 months      6.49       220.65     A        A2
4      24375   60 months     17.27       609.33     C        C5

                 emp_title emp_length home_ownership  annual_inc  ...
\
0                Marketing  10+ years           RENT    117000.0  ...

1          Credit analyst     4 years       MORTGAGE     65000.0  ...

2             Statistician   < 1 year           RENT     43057.0  ...

3          Client Advocate    6 years           RENT     54000.0  ...

4  Destiny Management Inc.    9 years       MORTGAGE     55000.0  ...


  open_acc pub_rec revol_bal revol_util total_acc  initial_list_status
\
0       16       0     36369       41.8        25                    w

1       17       0     20131       53.3        27                    f

2       13       0     11987       92.2        26                    f

3        6       0      5472       21.5        13                    f

4       13       0     24584       69.8        43                    f


  application_type  mort_acc  pub_rec_bankruptcies  \
0       INDIVIDUAL       0.0                   0.0
1       INDIVIDUAL       3.0                   0.0
2       INDIVIDUAL       0.0                   0.0
```

```
3       INDIVIDUAL       0.0                     0.0
4       INDIVIDUAL       1.0                     0.0

                                             address
0     0174 Michelle Gateway\nMendozaberg, OK 22690
1  1076 Carney Fort Apt. 347\nLoganmouth, SD 05113
2  87025 Mark Dale Apt. 269\nNew Sabrina, WV 05113
3            823 Reid Ford\nDelacruzside, MA 00813
4             679 Luna Roads\nGreggshire, VA 11650

[5 rows x 27 columns]
```

```
data.shape
```

```
(396030, 27)
```

```
data.columns
```

```
Index(['loan_amnt', 'term', 'int_rate', 'installment', 'grade',
'sub_grade',
       'emp_title', 'emp_length', 'home_ownership', 'annual_inc',
       'verification_status', 'issue_d', 'loan_status', 'purpose',
'title',
       'dti', 'earliest_cr_line', 'open_acc', 'pub_rec', 'revol_bal',
       'revol_util', 'total_acc', 'initial_list_status',
'application_type',
       'mort_acc', 'pub_rec_bankruptcies', 'address'],
      dtype='object')
```

```
data.dtypes
```

```
loan_amnt                 int64
term                     object
int_rate                float64
installment             float64
grade                    object
sub_grade                object
emp_title                object
emp_length               object
home_ownership           object
annual_inc              float64
verification_status      object
issue_d                  object
loan_status              object
purpose                  object
title                    object
dti                     float64
earliest_cr_line         object
open_acc                  int64
pub_rec                   int64
revol_bal                 int64
```

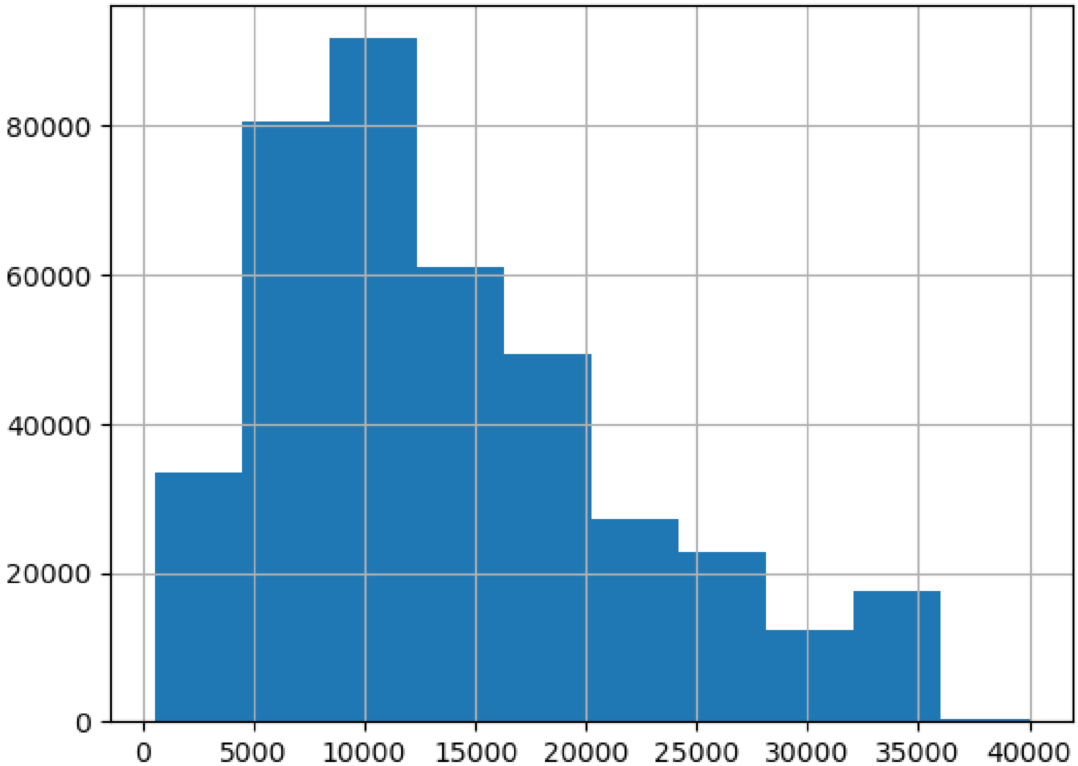
```
revol_util                float64
total_acc                   int64
initial_list_status        object
application_type           object
mort_acc                  float64
pub_rec_bankruptcies      float64
address                    object
dtype: object
```

```python
data.isnull().sum()
```

```
loan_amnt                   0
term                        0
int_rate                    0
installment                 0
grade                       0
sub_grade                   0
emp_title               22927
emp_length              18301
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
loan_status                 0
purpose                     0
title                    1756
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
```

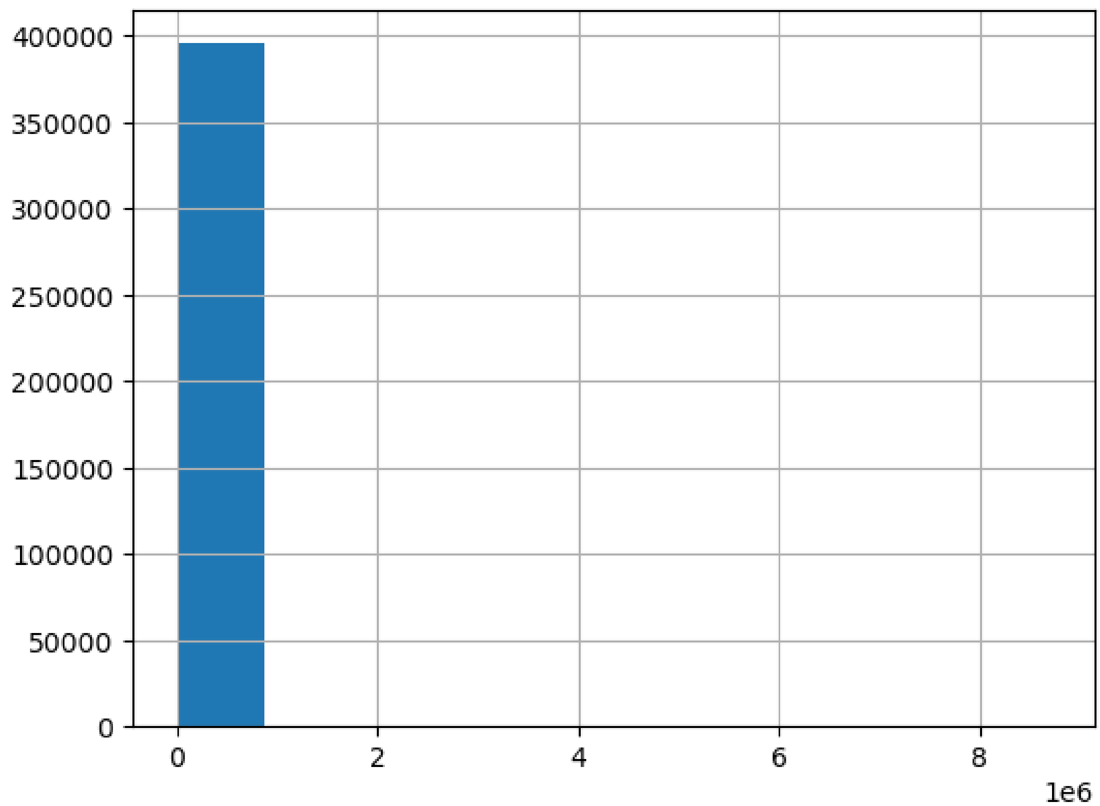
```python
data.loan_amnt.hist()
```

```
<Axes: >
```




```
data.annual_inc.hist() #This shows the presence of a outliers
```

```
<Axes: >
```




```
data.annual_inc.describe()
```

```
count    3.960300e+05
mean     7.420318e+04
std      6.163762e+04
min      0.000000e+00
25%      4.500000e+04
50%      6.400000e+04
75%      9.000000e+04
max      8.706582e+06
Name: annual_inc, dtype: float64
```

```
data.term.value_counts()
```

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

```
data.term=data.term.str.split(expand=True)[0]

data.term.value_counts()
```

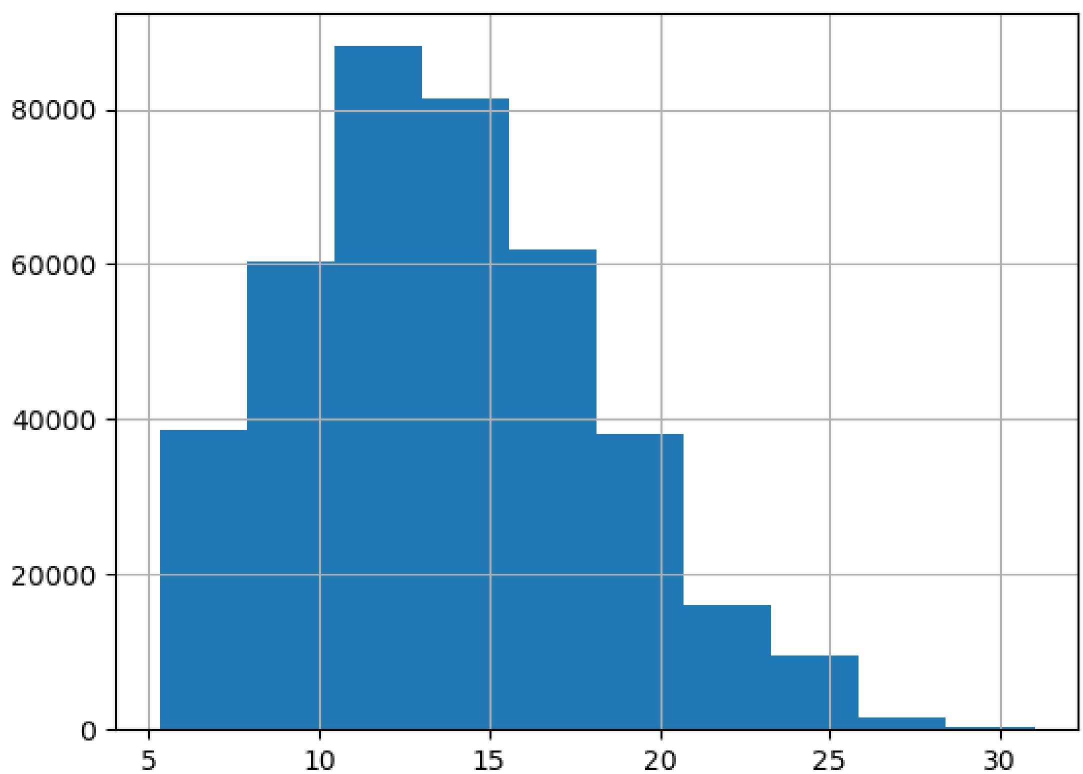
```
term
36    302005
60     94025
Name: count, dtype: int64
```

```
data.int_rate.describe()
```

```
count    396030.000000
mean         13.639400
std           4.472157
min           5.320000
25%          10.490000
50%          13.330000
75%          16.490000
max          30.990000
Name: int_rate, dtype: float64
```

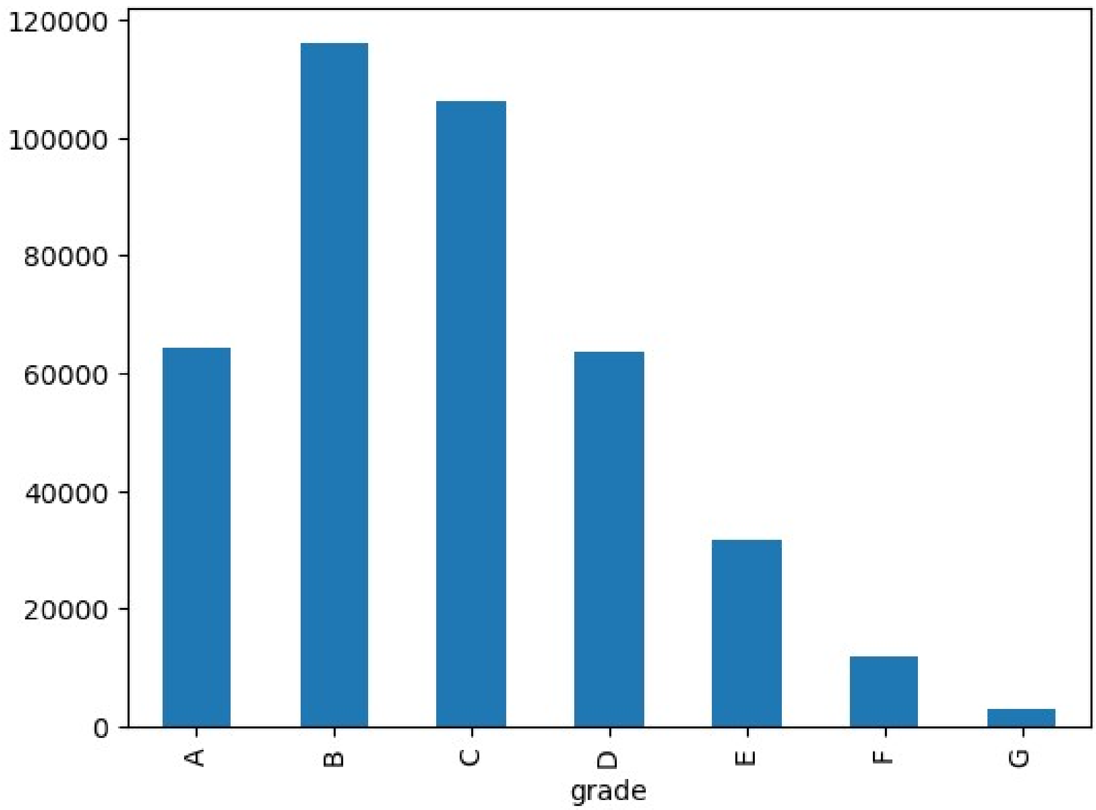
```
data.int_rate.hist()
```

```
<Axes: >
```




```
data.grade.value_counts().sort_index().plot(kind='bar')
```

```
<Axes: xlabel='grade'>
```




```
data.grade.value_counts()

grade
B    116018
C    105987
A     64187
D     63524
E     31488
F     11772
G      3054
Name: count, dtype: int64

sorted(data.sub_grade.unique()) #assuming A1 is better than F5 because this is a grade and ordinal

['A1',
 'A2',
 'A3',
 'A4',
 'A5',
 'B1',
 'B2',
 'B3',
```

```
 'B4',
 'B5',
 'C1',
 'C2',
 'C3',
 'C4',
 'C5',
 'D1',
 'D2',
 'D3',
 'D4',
 'D5',
 'E1',
 'E2',
 'E3',
 'E4',
 'E5',
 'F1',
 'F2',
 'F3',
 'F4',
 'F5',
 'G1',
 'G2',
 'G3',
 'G4',
 'G5']
```

```
data.emp_title.value_counts()
```

```
emp_title
Teacher                          4389
Manager                          4250
Registered Nurse                 1856
RN                               1846
Supervisor                       1830
                                 ...
sikorsky                            1
Postman                             1
McCarthy & Holthus, LLC             1
jp flooring                         1
Gracon Services, Inc                1
Name: count, Length: 173103, dtype: int64
```

```
data[data.emp_title.isnull()]
```

```
        loan_amnt term  int_rate  installment grade sub_grade
emp_title  \
```

```
35               5375   36     13.11        181.39     B        B4
NaN
36               3250   36     16.78        115.52     C        C5
NaN
40              35000   60     16.99        869.66     D        D1
NaN
49              15000   36      7.89        469.29     A        A5
NaN
58              10000   36     17.56        359.33     D        D1
NaN
...               ...  ...       ...           ...   ...       ...        ..
.
395946          35000   60     16.20        854.86     C        C4
NaN
395963           7000   36     20.20        260.86     E        E3
NaN
395988          35000   60     15.59        843.53     D        D1
NaN
395999          11125   36     24.11        437.11     F        F2
NaN
396015           4000   36      9.16        127.50     B        B2
NaN

       emp_length home_ownership  annual_inc  ... open_acc pub_rec
revol_bal  \
35            NaN           RENT    34000.00  ...        9       1
14998
36            NaN           RENT    22500.00  ...        7       0
7587
40        4 years       MORTGAGE   130000.00  ...       10       0
34130
49            NaN       MORTGAGE    90000.00  ...        7       0
8205
58            NaN       MORTGAGE    32000.00  ...        6       0
11615
...           ...            ...         ...  ...      ...     ...
...
395946        NaN       MORTGAGE    84000.00  ...        7       0
4241
395963        NaN            OWN    32964.00  ...       24       1
3236
395988        NaN            OWN   102396.00  ...       15       0
31665
395999        NaN       MORTGAGE    31789.88  ...        8       0
22385
396015        NaN       MORTGAGE    57400.00  ...       12       0
3134

       revol_util total_acc  initial_list_status application_type
```

```
mort_acc  \
35          88.7        20                    f        INDIVIDUAL
5.0
36          54.6         7                    f        INDIVIDUAL
0.0
40          53.8        27                    f        INDIVIDUAL
10.0
49          93.2        18                    w        INDIVIDUAL
6.0
58          82.4         7                    w        INDIVIDUAL
0.0
...          ...       ...                  ...               ...
...
395946      18.8        21                    w        INDIVIDUAL
5.0
395963       9.7        44                    w        INDIVIDUAL
0.0
395988      32.4        33                    w        INDIVIDUAL
1.0
395999      81.0        24                    w        INDIVIDUAL
4.0
396015       5.8        27                    w        INDIVIDUAL
5.0

        pub_rec_bankruptcies  \
35                       1.0
36                       0.0
40                       0.0
49                       0.0
58                       0.0
...                      ...
395946                   0.0
395963                   1.0
395988                   0.0
395999                   0.0
396015                   0.0

                                                  address
35           23617 Michael Viaduct\nWest John, MS 05113
36             361 Erica Forest\nLake Mariaton, TN 30723
40      8268 Reed Gardens Suite 996\nEast Johnmouth, N...
49        84009 David Stream\nSouth Nicolehaven, IL 05113
58                965 Spencer Courts\nPacetown, AZ 00813
...                                                   ...
395946            2645 Wayne Street\nMarymouth, HI 22690
395963  8339 Daniel Forges Suite 273\nPort Oscarmouth,...
395988              114 Sonya Pass\nCarlamouth, SD 00813
395999       1314 Bridget Terrace\nRebeccashire, NE 30723
396015                   Unit 4067 Box 2110\nDPO AA 05113
```

```
[22927 rows x 27 columns]

data[data.emp_length.isnull()]

        loan_amnt term  int_rate  installment grade sub_grade
emp_title  \
35           5375   36     13.11       181.39     B        B4
NaN
36           3250   36     16.78       115.52     C        C5
NaN
49          15000   36      7.89       469.29     A        A5
NaN
58          10000   36     17.56       359.33     D        D1
NaN
91          30225   60     18.24       771.47     D        D5
NaN
...           ...  ...       ...          ...   ...       ...        ..
.
395946      35000   60     16.20       854.86     C        C4
NaN
395963       7000   36     20.20       260.86     E        E3
NaN
395988      35000   60     15.59       843.53     D        D1
NaN
395999      11125   36     24.11       437.11     F        F2
NaN
396015       4000   36      9.16       127.50     B        B2
NaN

       emp_length home_ownership  annual_inc  ... open_acc pub_rec
revol_bal  \
35            NaN           RENT    34000.00  ...        9       1
14998
36            NaN           RENT    22500.00  ...        7       0
7587
49            NaN       MORTGAGE    90000.00  ...        7       0
8205
58            NaN       MORTGAGE    32000.00  ...        6       0
11615
91            NaN       MORTGAGE    65800.00  ...       11       0
14390
...           ...            ...         ...  ...      ...     ...
...
395946        NaN       MORTGAGE    84000.00  ...        7       0
4241
395963        NaN            OWN    32964.00  ...       24       1
3236
395988        NaN            OWN   102396.00  ...       15       0
31665
```

```
395999        NaN       MORTGAGE    31789.88  ...        8       0
22385
396015        NaN       MORTGAGE    57400.00  ...       12       0
3134

       revol_util total_acc  initial_list_status application_type
mort_acc  \
35           88.7        20                    f       INDIVIDUAL
5.0
36           54.6         7                    f       INDIVIDUAL
0.0
49           93.2        18                    w       INDIVIDUAL
6.0
58           82.4         7                    w       INDIVIDUAL
0.0
91           69.5        31                    w       INDIVIDUAL
1.0
...           ...       ...                  ...              ...
...
395946       18.8        21                    w       INDIVIDUAL
5.0
395963        9.7        44                    w       INDIVIDUAL
0.0
395988       32.4        33                    w       INDIVIDUAL
1.0
395999       81.0        24                    w       INDIVIDUAL
4.0
396015        5.8        27                    w       INDIVIDUAL
5.0

        pub_rec_bankruptcies  \
35                       1.0
36                       0.0
49                       0.0
58                       0.0
91                       0.0
...                      ...
395946                   0.0
395963                   1.0
395988                   0.0
395999                   0.0
396015                   0.0

                                                      address
35            23617 Michael Viaduct\nWest John, MS 05113
36             361 Erica Forest\nLake Mariaton, TN 30723
49       84009 David Stream\nSouth Nicolehaven, IL 05113
58                965 Spencer Courts\nPacetown, AZ 00813
91                 493 Michael Route\nHillfurt, AZ 70466
...                                                   ...
```

```
395946              2645 Wayne Street\nMarymouth, HI 22690
395963  8339 Daniel Forges Suite 273\nPort Oscarmouth,...
395988                114 Sonya Pass\nCarlamouth, SD 00813
395999       1314 Bridget Terrace\nRebeccashire, NE 30723
396015                    Unit 4067 Box 2110\nDPO AA 05113

[18301 rows x 27 columns]
```

```python
data.home_ownership.value_counts()
```

```
home_ownership
MORTGAGE    198348
RENT        159790
OWN          37746
OTHER          112
NONE            31
ANY              3
Name: count, dtype: int64
```

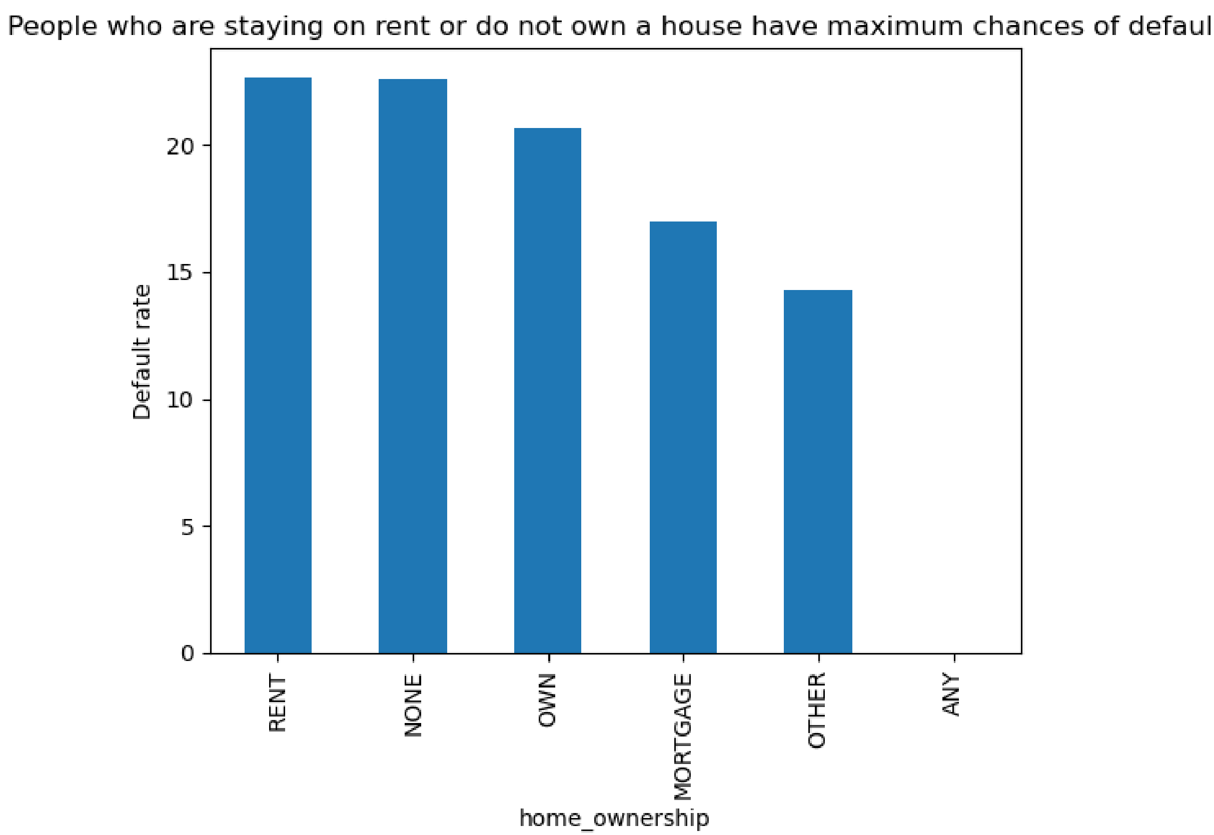
```python
data.groupby(['home_ownership','loan_status'])['loan_status'].size()
```

```
home_ownership  loan_status
ANY             Fully Paid          3
MORTGAGE        Charged Off     33632
                Fully Paid     164716
NONE            Charged Off         7
                Fully Paid         24
OTHER           Charged Off        16
                Fully Paid         96
OWN             Charged Off      7806
                Fully Paid      29940
RENT            Charged Off     36212
                Fully Paid     123578
Name: loan_status, dtype: int64
```

```python
ho_fp=data[data['loan_status']=='Fully
Paid'].groupby('home_ownership')['loan_status'].count()

ho_fp
```

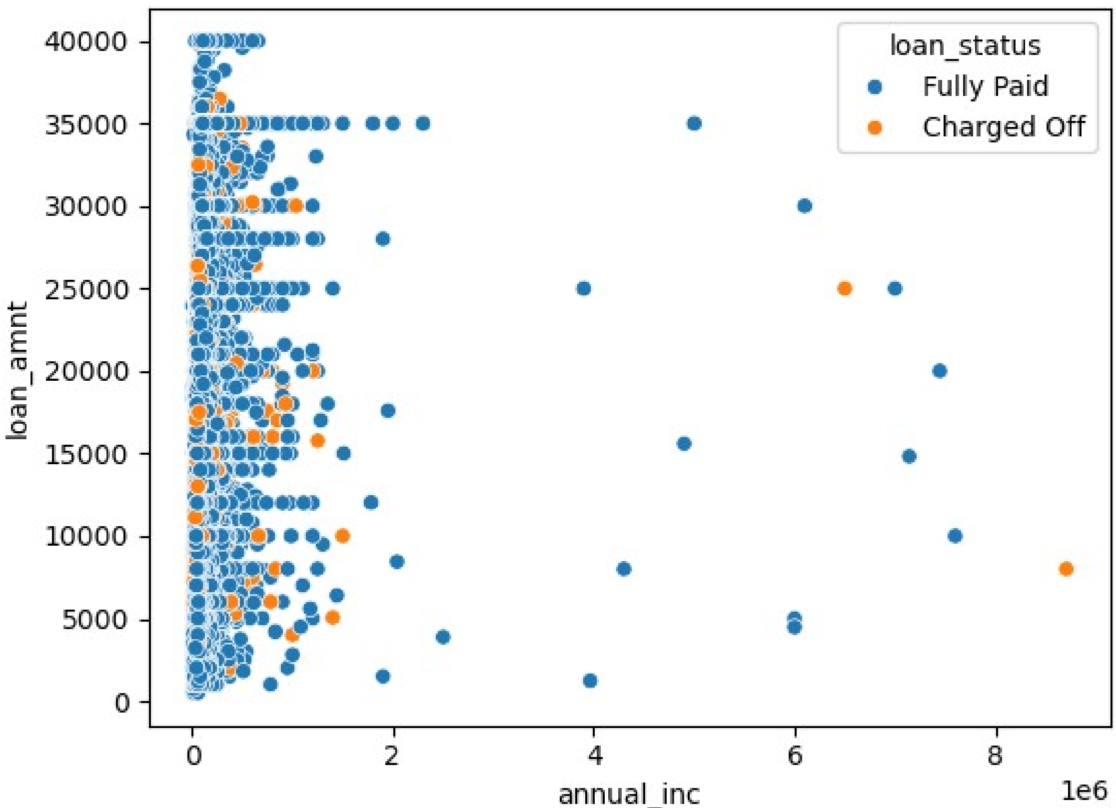
```
home_ownership
ANY              3
MORTGAGE    164716
NONE            24
OTHER           96
OWN          29940
```

```
RENT          123578
Name: loan_status, dtype: int64

ho_tls=data.groupby('home_ownership')['loan_status'].count()

ho_tls

home_ownership
ANY              3
MORTGAGE    198348
NONE            31
OTHER          112
OWN          37746
RENT        159790
Name: loan_status, dtype: int64

perc_default_by_ho=(1-(ho_fp/ho_tls))*100

perc_default_by_ho.sort_values(ascending=False).plot(kind='bar')
plt.title('People who are staying on rent or do not own a house have
maximum chances of default')
plt.ylabel('Default rate')
plt.show()
```



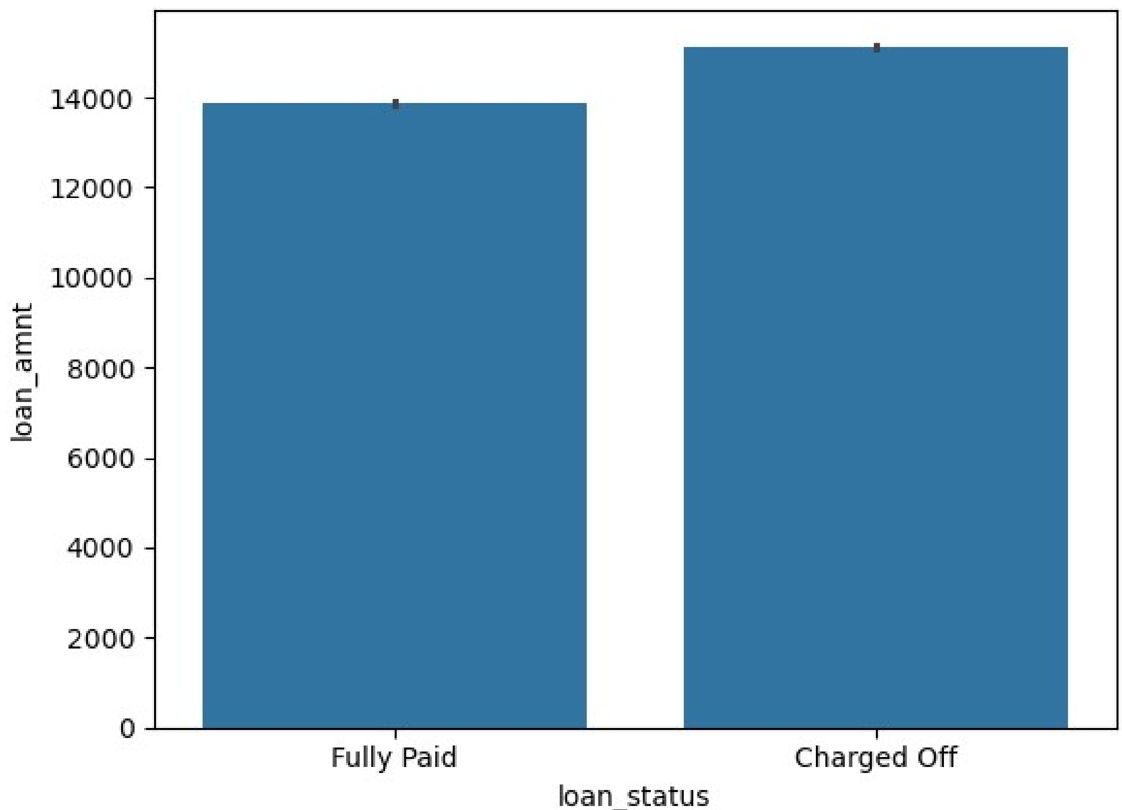
```python
sns.scatterplot(x='annual_inc', y='loan_amnt', hue='loan_status',
data=data) #The loans are more for people who do not have great
incomes
#So there is an opportunity to target more high value customers for
loans
```

```
<Axes: xlabel='annual_inc', ylabel='loan_amnt'>
```

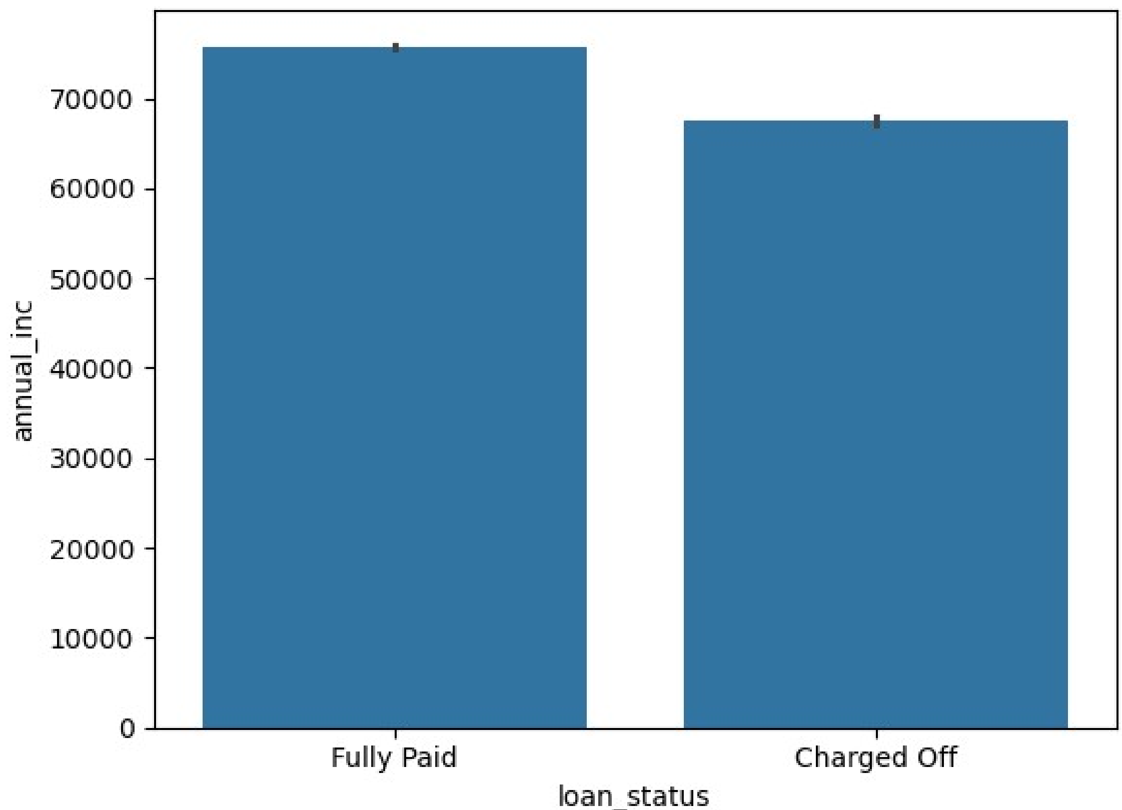



```python
sns.barplot(x='loan_status', y='loan_amnt', data=data) #people who
have charged off took higher loans on average
```

```
<Axes: xlabel='loan_status', ylabel='loan_amnt'>
```




```
sns.barplot(y='annual_inc',x='loan_status', data=data) #People who
charged off had lower incomes on average

<Axes: xlabel='loan_status', ylabel='annual_inc'>
```

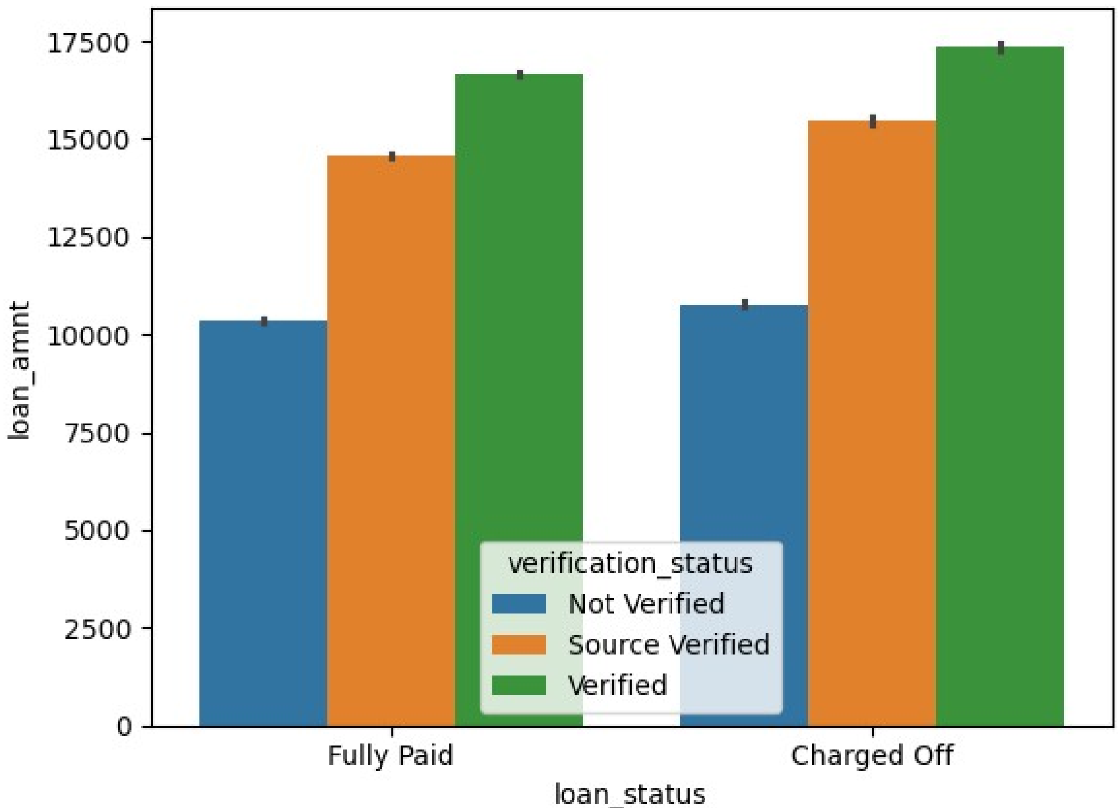



```
data.verification_status.value_counts()
```

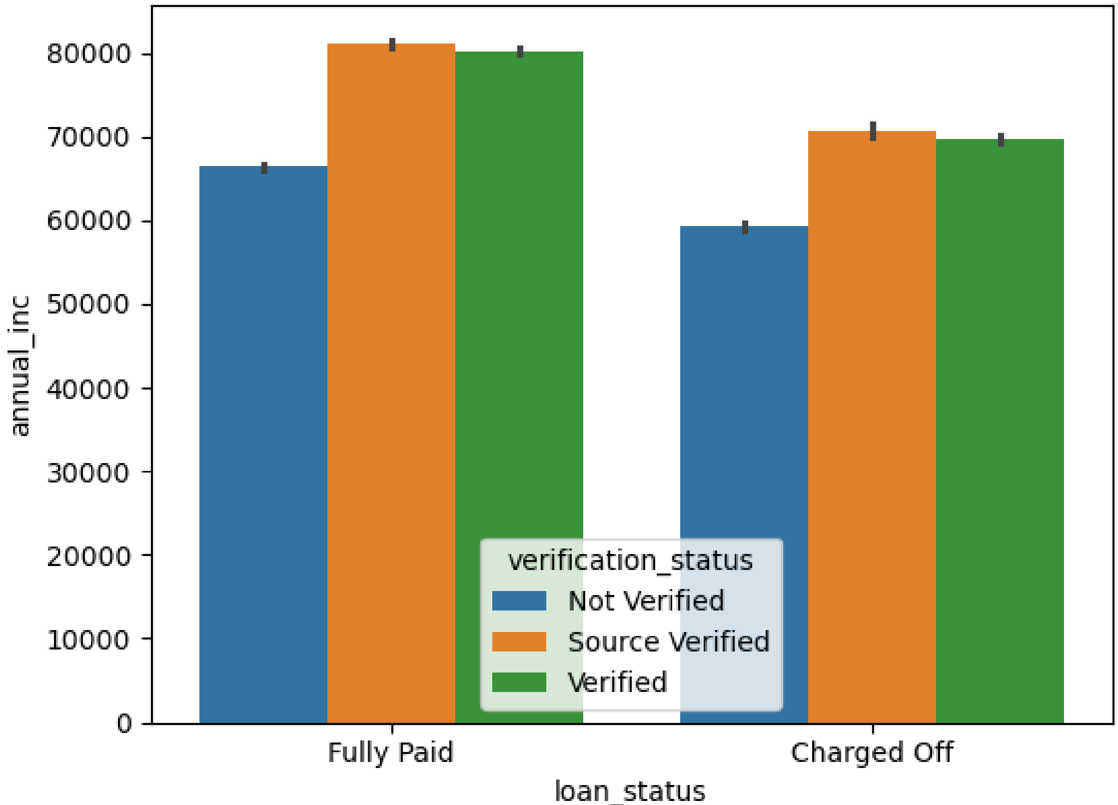
```
verification_status
Verified           139563
Source Verified    131385
Not Verified       125082
Name: count, dtype: int64
```

```
sns.barplot(x='loan_status', y='loan_amnt', hue='verification_status',
data=data) #verified people get higher loans
#People who took ran away got higher loans than those who paid up on
time
```

```
<Axes: xlabel='loan_status', ylabel='loan_amnt'>
```

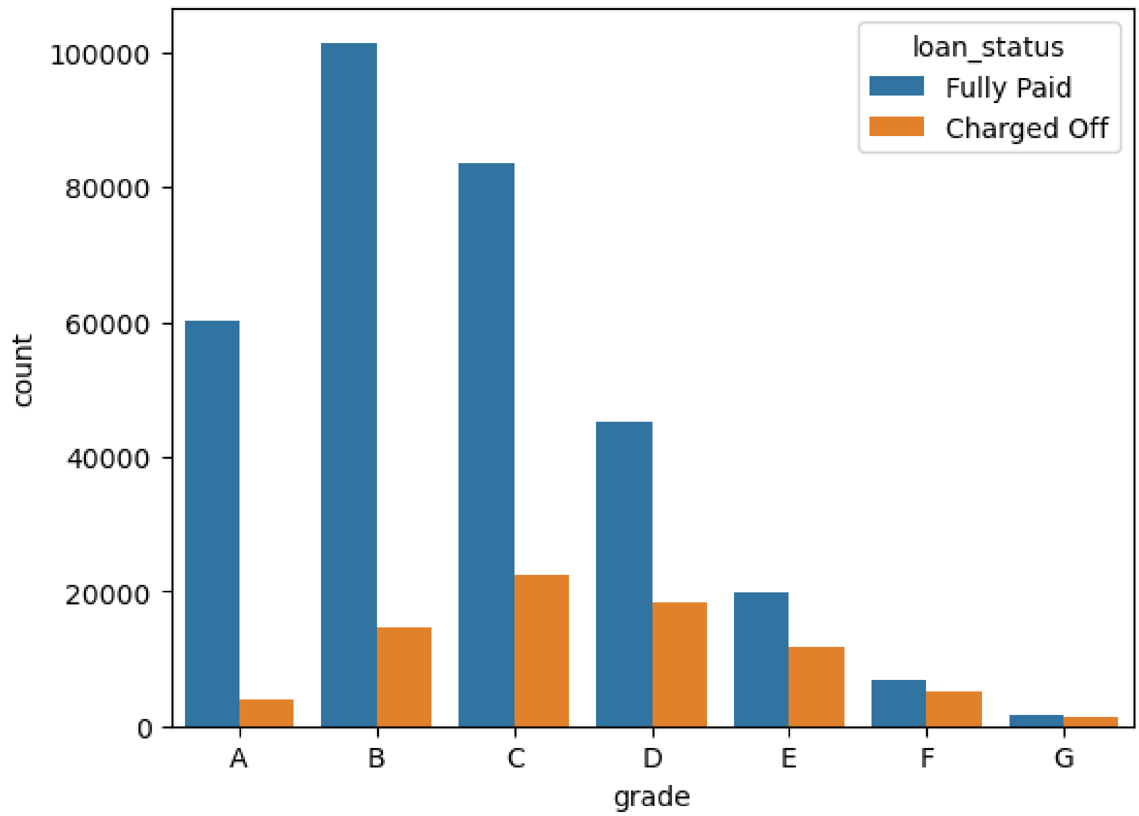



```
sns.barplot(x='loan_status', y='annual_inc',
hue='verification_status', data=data) #People who paid up had higher
incomes than people who did not

<Axes: xlabel='loan_status', ylabel='annual_inc'>
```



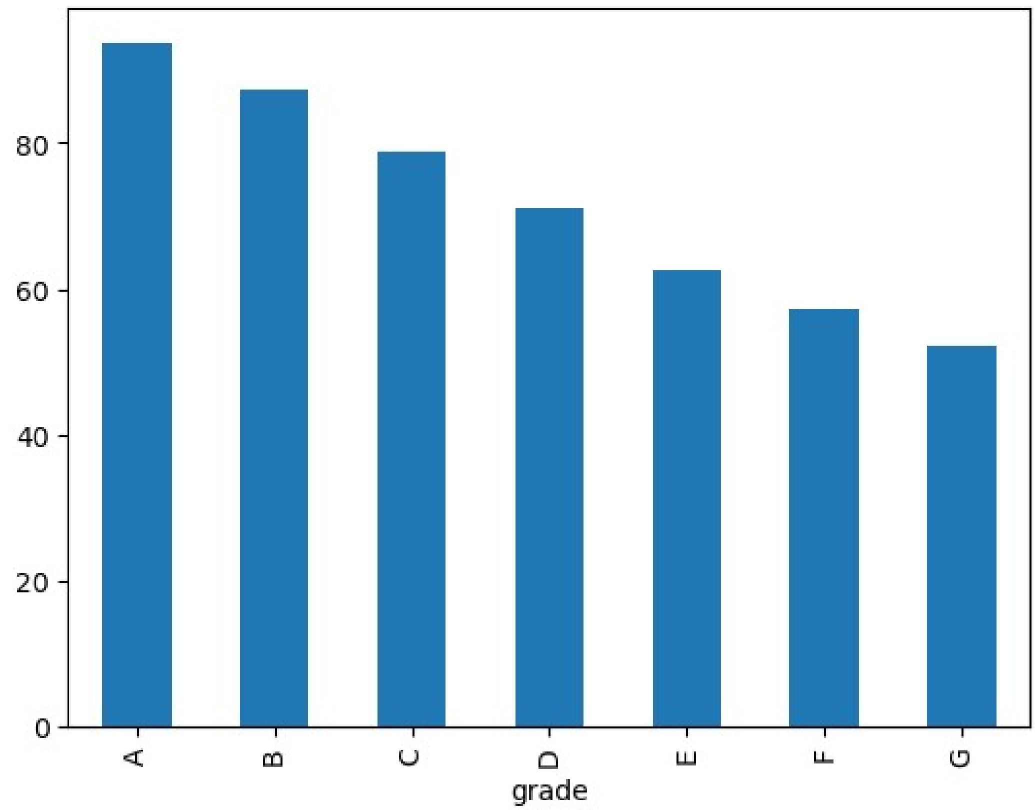

```
sns.countplot(x='grade', hue='loan_status',
order=['A','B','C','D','E','F','G'],data=data) #Since this depends on
the count of the data, I am going to take a %
```

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



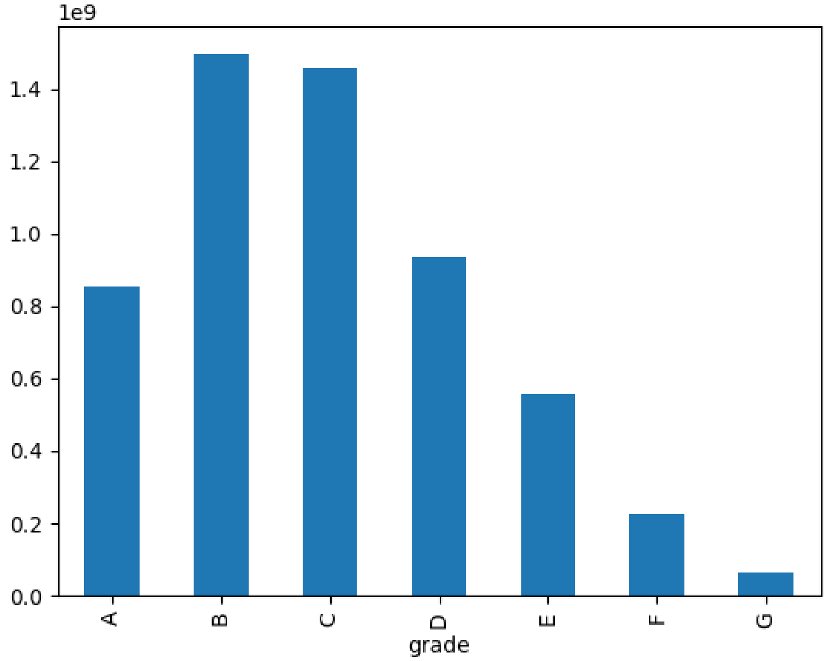

```
perc_paid_by_grade=100*data[data['loan_status']=='Fully
Paid'].groupby('grade')['loan_status'].count()/data.groupby('grade')
['loan_status'].count()

perc_paid_by_grade.plot(kind='bar') #clearly, having an "A" grade
ensures that more people who take the loans pay up
```

```
<Axes: xlabel='grade'>
```



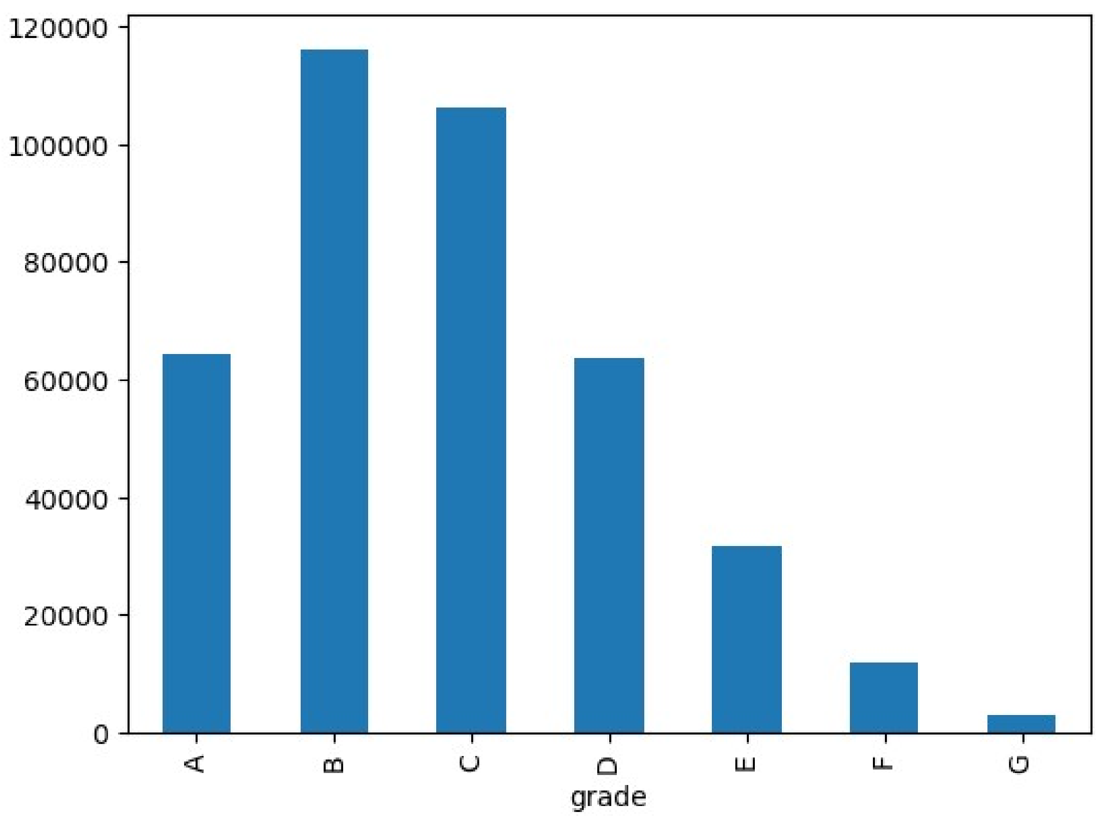

```
data.groupby('grade')['loan_amnt'].sum().plot(kind='bar')
plt.title('The share of total loan amount is more for B & C categories
compared to A which is more safer')
plt.show()
#More loans are going to b and c grade people when compared to a grade
people and this increases the risk
#Therefore, there is an opportunity for the company to reroute most of
its loan amount to a grade customers.
```

The share of total loan amount is more for B & C categories compared to A which is more safer



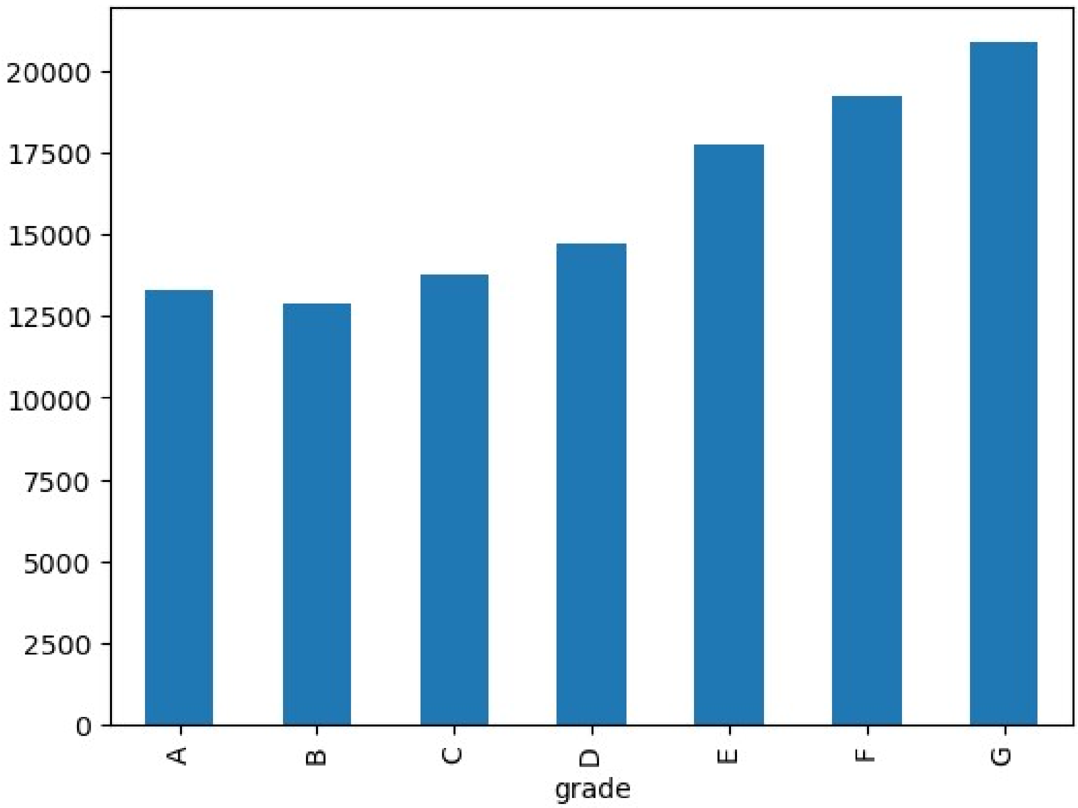

```python
data.grade.value_counts().sort_index().plot(kind='bar')
```

```
<Axes: xlabel='grade'>
```



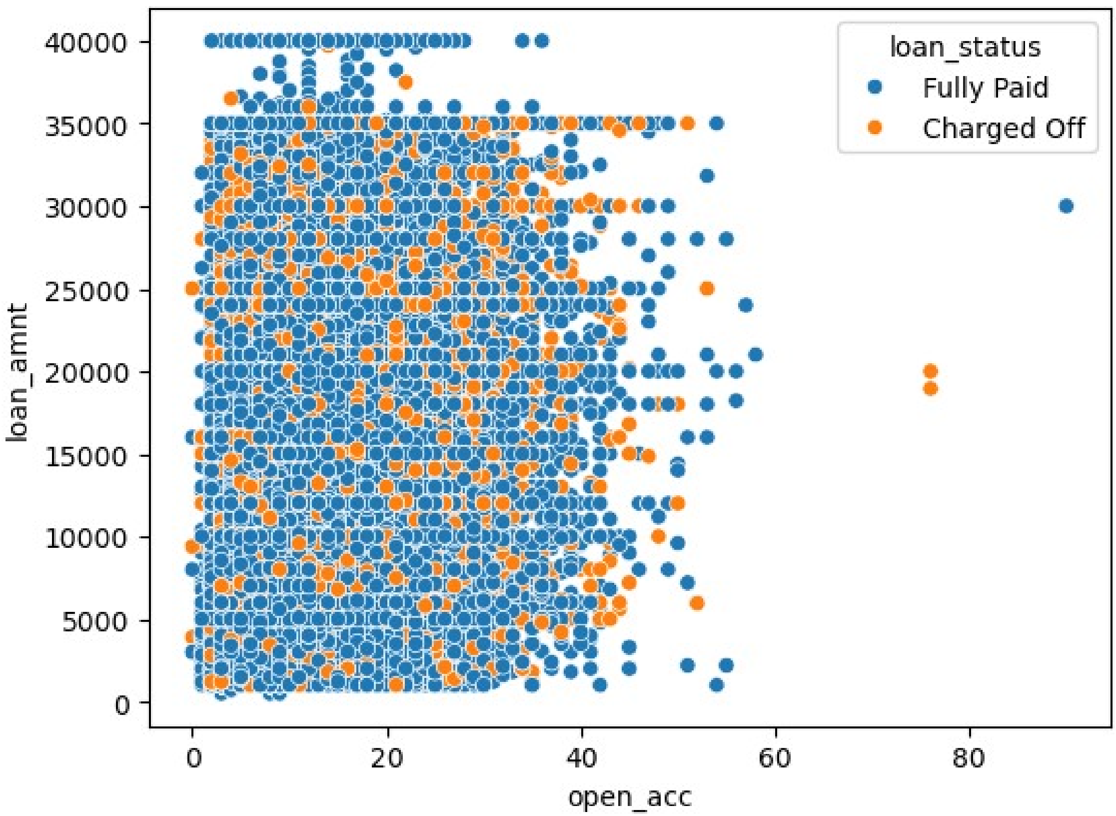
```
data.groupby('grade')['loan_amnt'].mean().plot(kind='bar')
#This shows that on an average a higher loan is being provided to
those who have the highest risk of default.
#This means the company should lower the risk of losing money, instead
providing the highest loans to grade a customers
```

```
<Axes: xlabel='grade'>
```



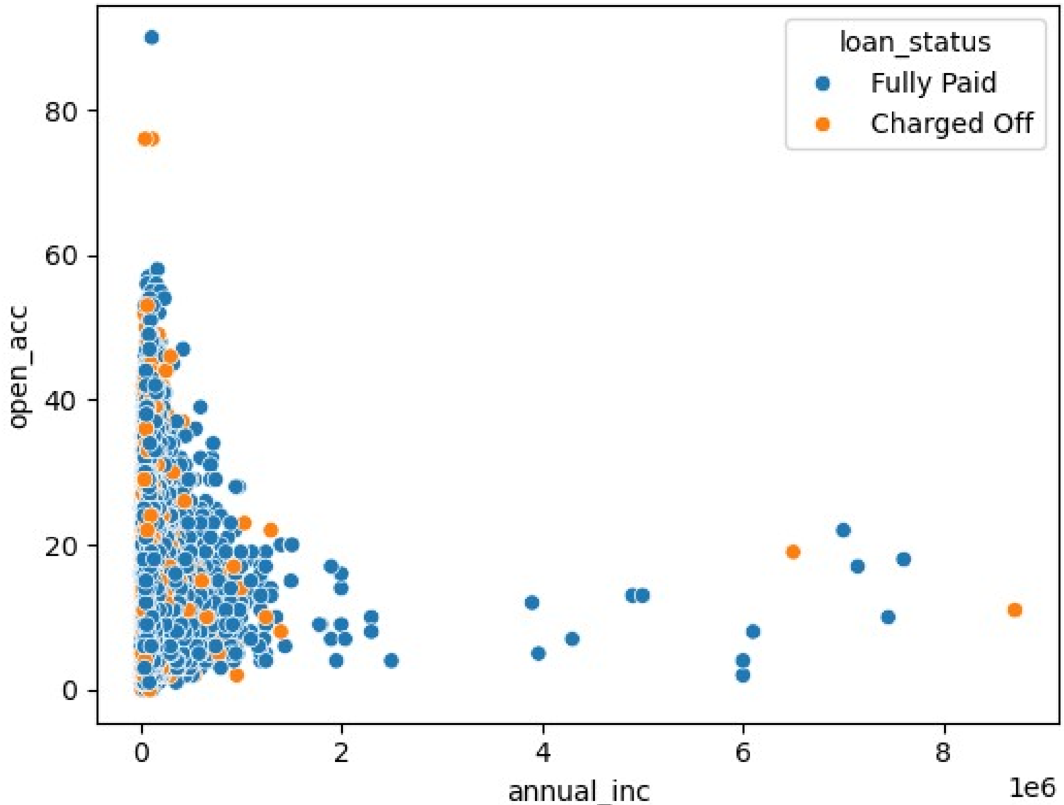

```
sns.scatterplot(x='open_acc', y='loan_amnt', hue='loan_status',
data=data)
```

```
<Axes: xlabel='open_acc', ylabel='loan_amnt'>
```




```
sns.scatterplot(x='annual_inc', y='open_acc', hue='loan_status',
data=data)
```

```
<Axes: xlabel='annual_inc', ylabel='open_acc'>
```




```
data.initial_list_status.value_counts()
```

```
initial_list_status
f    238066
w    157964
Name: count, dtype: int64
```

```
data.loan_status.value_counts()
```

```
loan_status
Fully Paid     318357
Charged Off     77673
Name: count, dtype: int64
```

```
data.groupby(['initial_list_status','loan_status'])
['loan_status'].count()
```

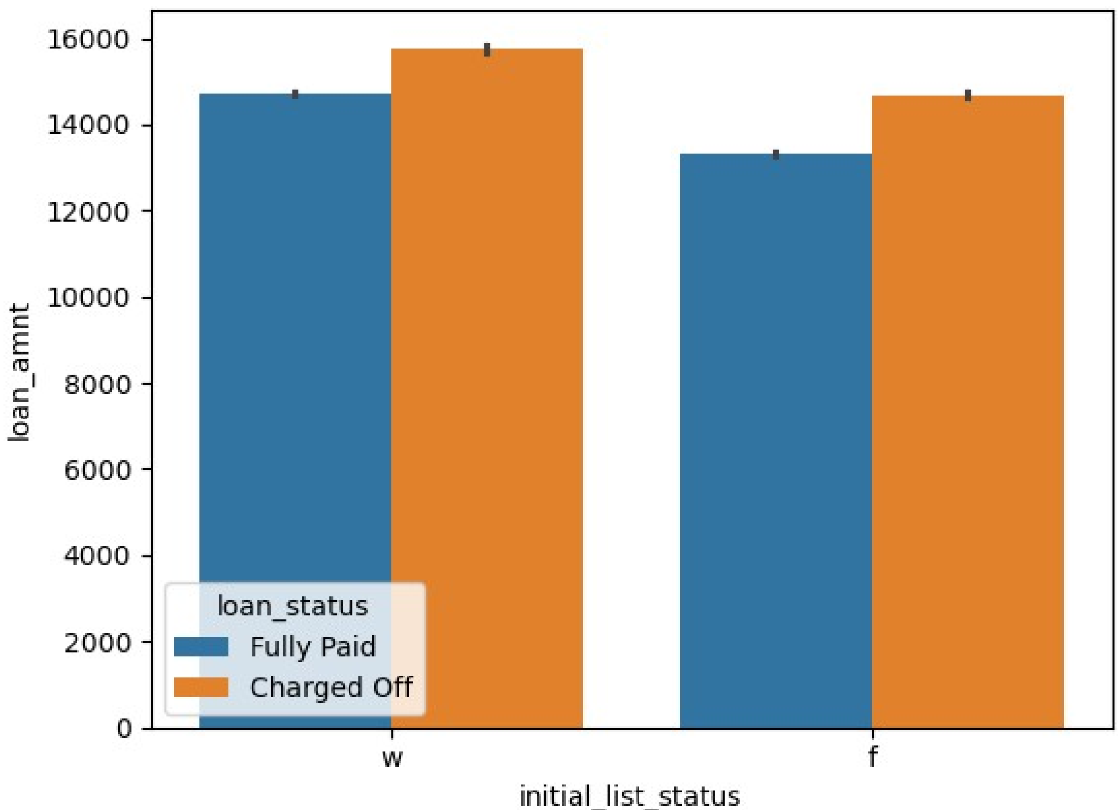
```
initial_list_status  loan_status
f                    Charged Off     45961
                     Fully Paid     192105
w                    Charged Off     31712
                     Fully Paid     126252
Name: loan_status, dtype: int64
```

```
init_status_default_risk=(data[data['loan_status']=='Charged
Off'].groupby('initial_list_status')['initial_list_status'].count())/
data.groupby('initial_list_status')['initial_list_status'].count()

init_status_default_risk #There is not much of a difference here
```

```
initial_list_status
f    0.193060
w    0.200755
Name: initial_list_status, dtype: float64
```

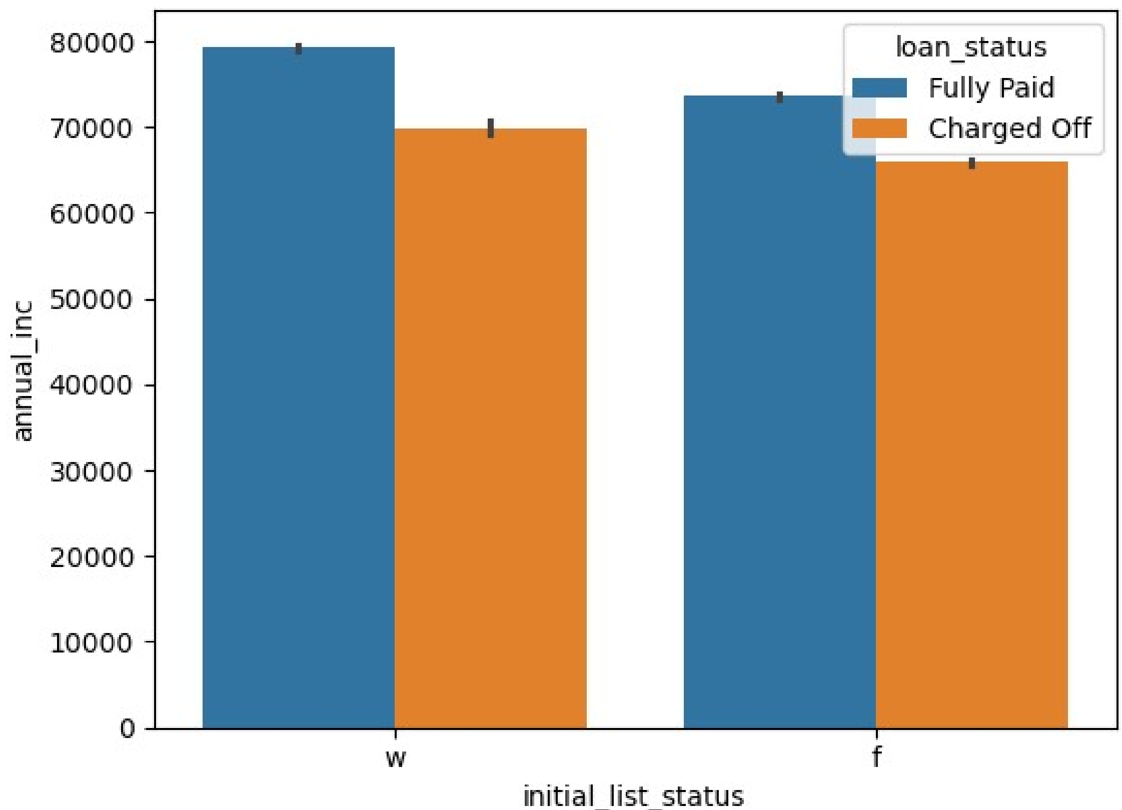
```
sns.barplot(x='initial_list_status', y='loan_amnt', hue='loan_status',
data=data)
```

```
<Axes: xlabel='initial_list_status', ylabel='loan_amnt'>
```

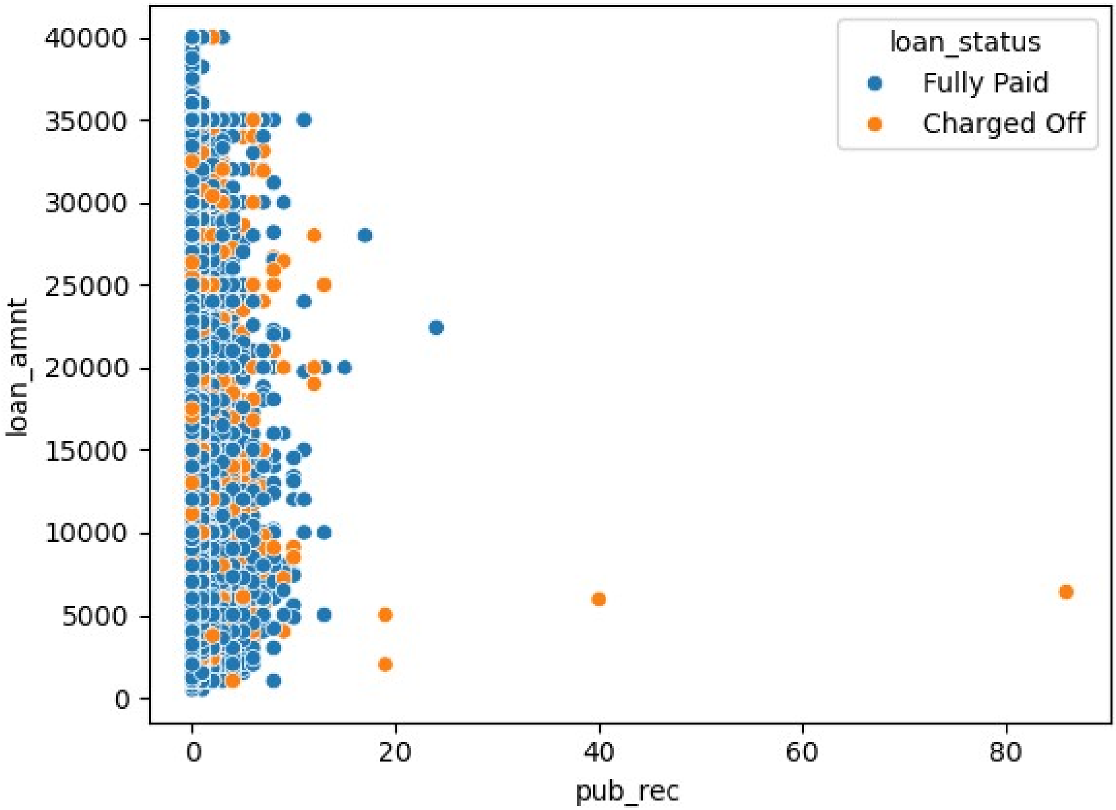



```
sns.barplot(x='initial_list_status', y='annual_inc',
hue='loan_status', data=data)
```

```
<Axes: xlabel='initial_list_status', ylabel='annual_inc'>
```

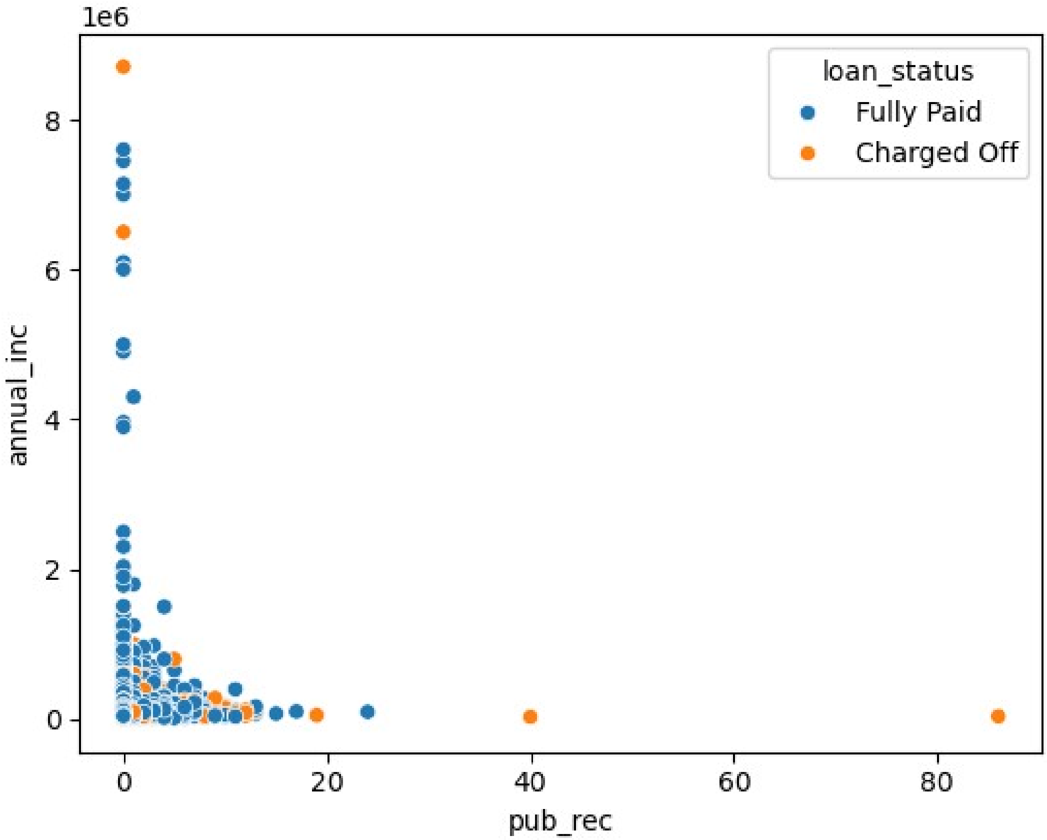



```
sns.scatterplot(x='pub_rec', y='loan_amnt', hue='loan_status',
data=data)

<Axes: xlabel='pub_rec', ylabel='loan_amnt'>
```




```
sns.scatterplot(x='pub_rec', y='annual_inc', hue='loan_status',
data=data)
```

```
<Axes: xlabel='pub_rec', ylabel='annual_inc'>
```




```
data.purpose.value_counts()
#this shows that the top 2 categories are debt_consolidation and credit cards, both of which indicate that one loan is funding the recovery of other loans
#This means the company is at high risk from these categories, and should move to give loans in categories where there is security available
```

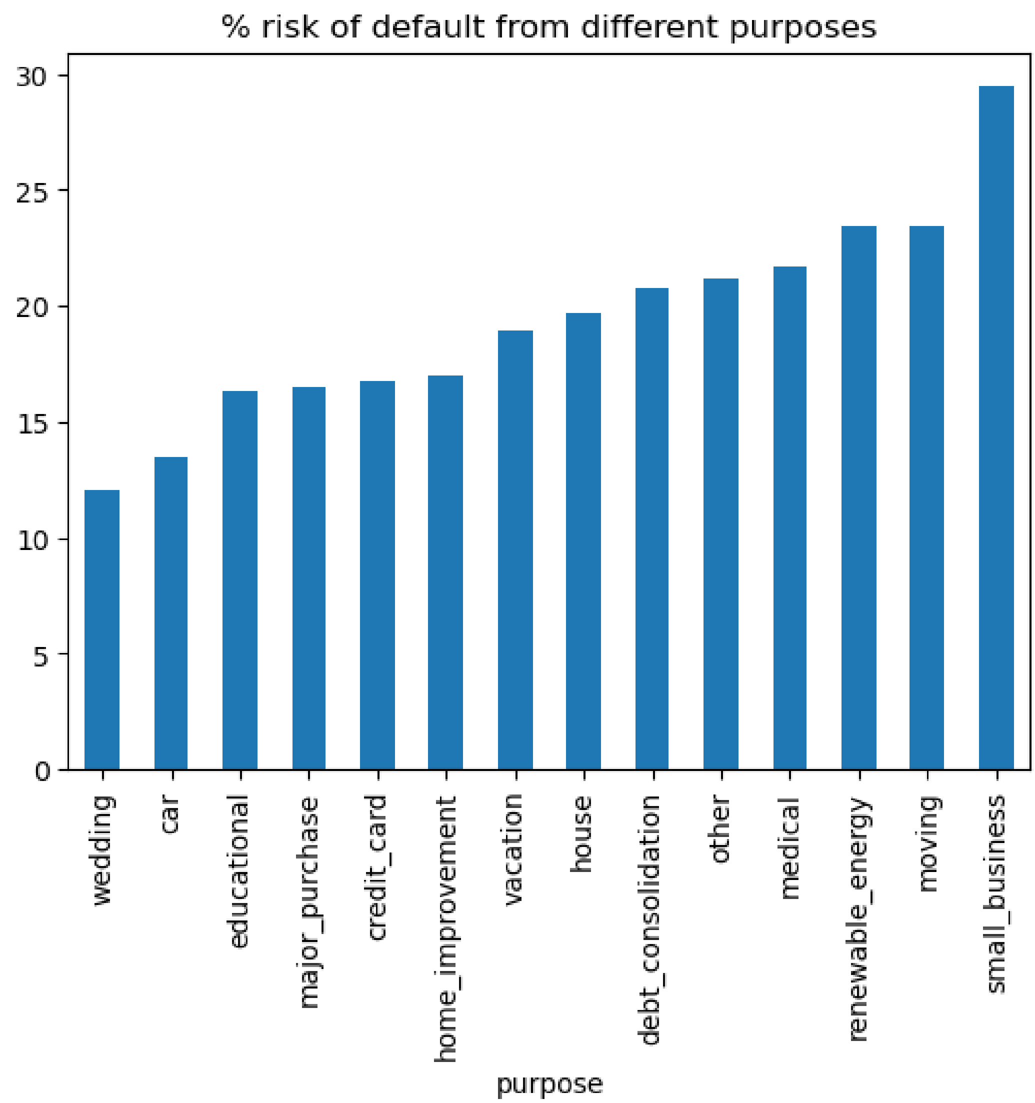
```
purpose
debt_consolidation    234507
credit_card            83019
home_improvement       24030
other                  21185
major_purchase          8790
small_business          5701
car                     4697
medical                 4196
moving                  2854
vacation                2452
house                   2201
wedding                 1812
renewable_energy         329
```

```
educational                257
Name: count, dtype: int64
```

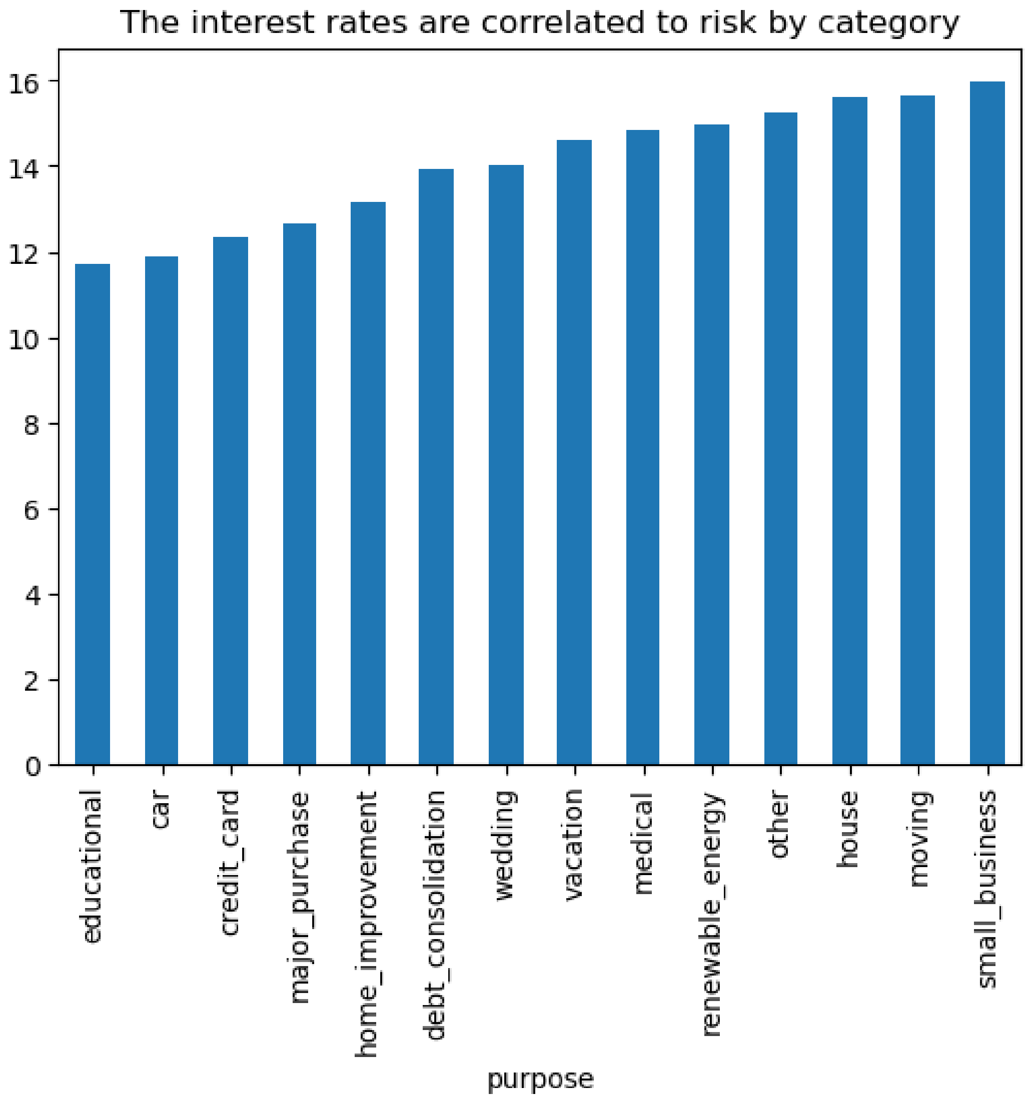
```
data.groupby('purpose')
['loan_amnt'].sum().sort_values(ascending=False) #this confirms the
above finding, the loan should be in areas where there is security,
not to repay other risky loans
```

```
purpose
debt_consolidation    3489116875
credit_card           1202306225
home_improvement       339160650
other                  203980925
major_purchase          96065225
small_business          87722650
car                     38615950
medical                 37574525
house                   33884825
moving                  22473500
wedding                 18540025
vacation                15252050
renewable_energy         3076750
educational              1752925
Name: loan_amnt, dtype: int64
```

```
(data[data['loan_status']=='Charged Off'].groupby('purpose')
['purpose'].count()*100/data.groupby('purpose')
['purpose'].count()).sort_values().plot(kind='bar')
plt.title('% risk of default from different purposes')
plt.show()
```




```
#From the graph above, we see that wedding and educational loans are
low risk, yet the company gives fewer loans to them

#Now, is the interest rate correlated with risk?
data.groupby('purpose')
['int_rate'].mean().sort_values().plot(kind='bar')
plt.title("The interest rates are correlated to risk by category")
plt.show()
```




```
#This shows that the bank gives high interest rates to risky categories, and penalizes innocent borrowers to account for those who are defaulting
#The most likely reason is because they do not maintain data at an individual level and adjust it regularly based on each customer's behaviour
#Therefore in this case, I recommend that the bank should implement a blockchain solution so that they can identify who is likely to default rather than just which category is at high risk using aggregate data

data.application_type.value_counts()
```

```
application_type
INDIVIDUAL    395319
JOINT            425
```

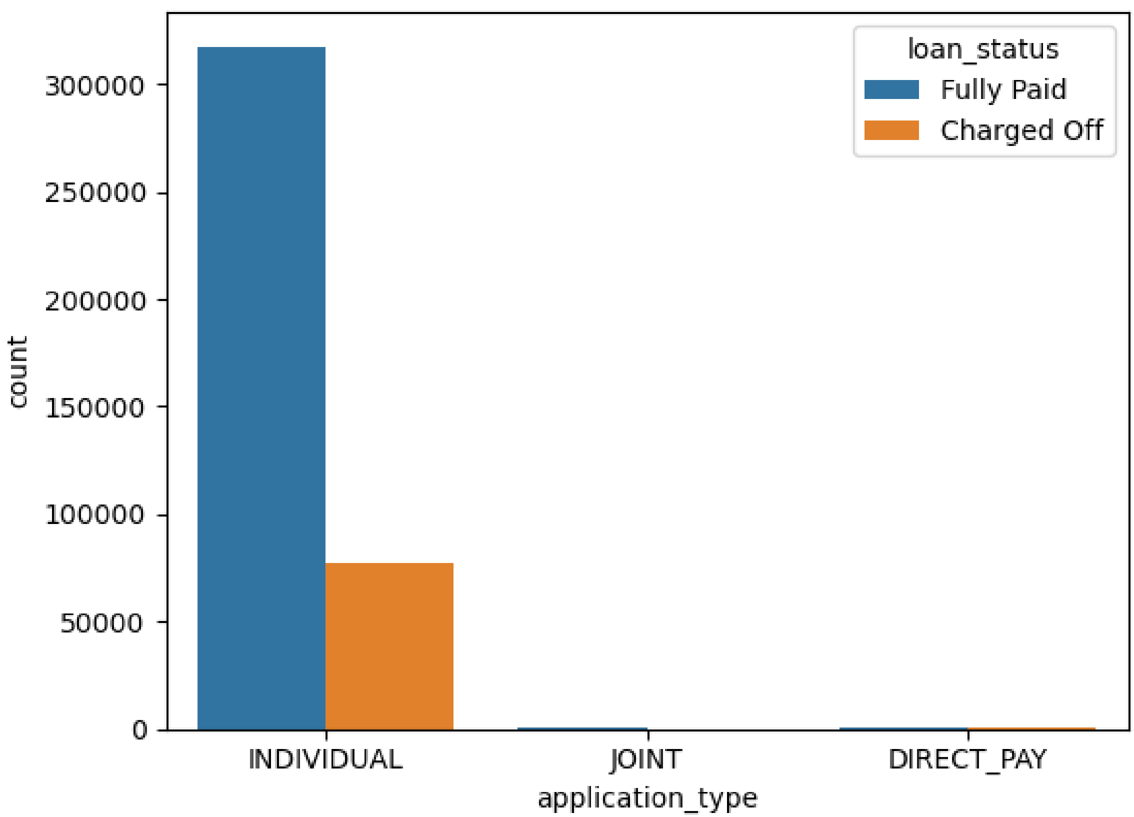
```
DIRECT_PAY       286
Name: count, dtype: int64
```

```
data.groupby('application_type')['loan_status'].value_counts() #This shows that we should look only at individual borrowers only
```

```
application_type  loan_status
DIRECT_PAY        Fully Paid        184
                  Charged Off       102
INDIVIDUAL        Fully Paid     317802
                  Charged Off     77517
JOINT             Fully Paid        371
                  Charged Off        54
Name: count, dtype: int64
```

```
sns.countplot(x='application_type', hue='loan_status', data=data)
```

```
<Axes: xlabel='application_type', ylabel='count'>
```



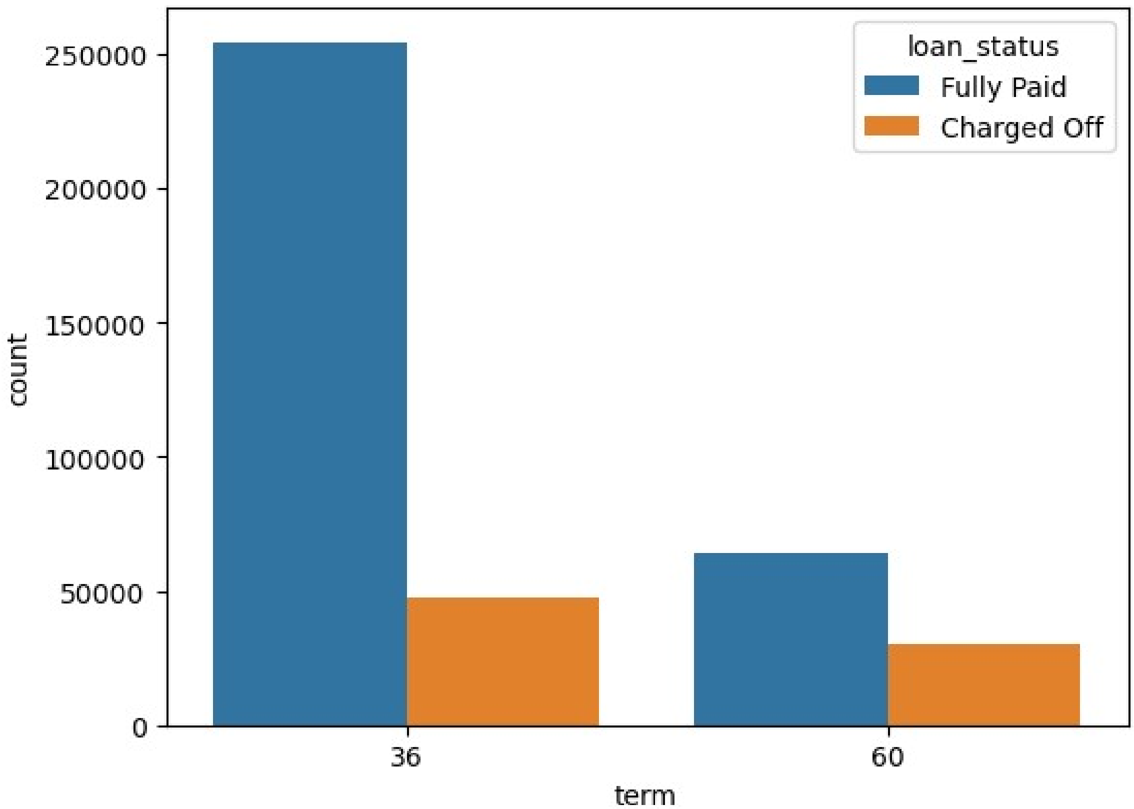

```
data.issue_d
```

```
0         Jan-15
1         Jan-15
2         Jan-15
```

```
3         Nov-14
4         Apr-13
            ...
396025    Oct-15
396026    Feb-15
396027    Oct-13
396028    Aug-12
396029    Jun-10
Name: issue_d, Length: 396030, dtype: object
```

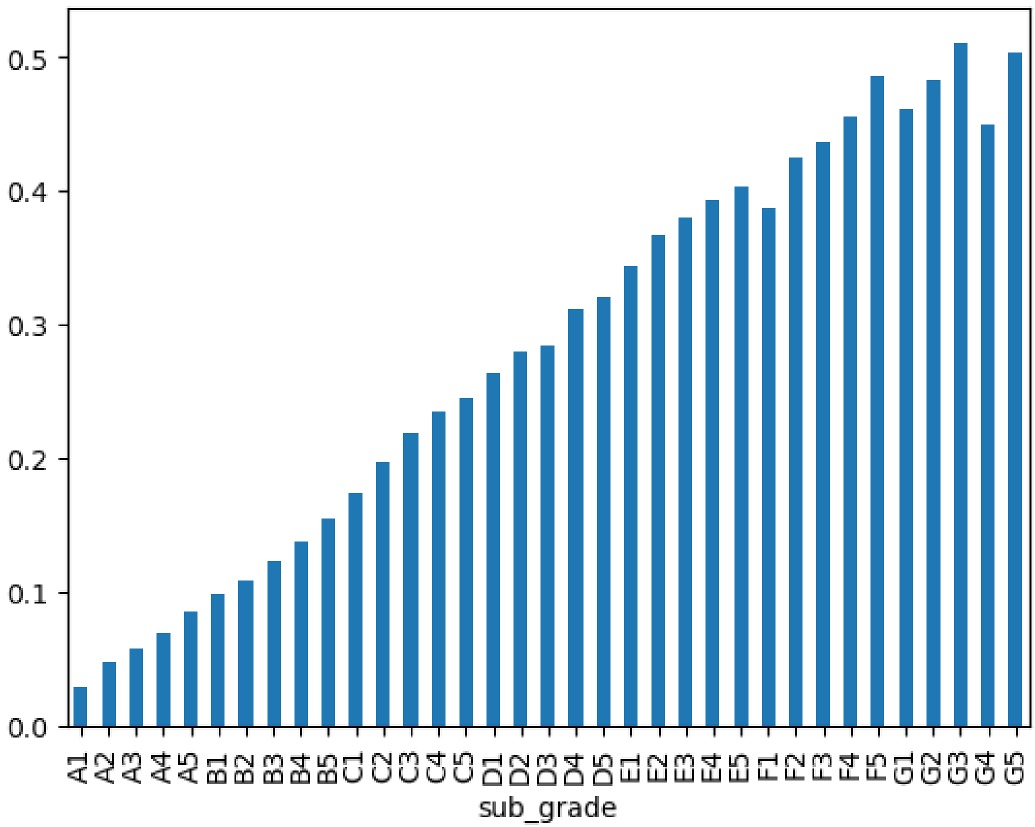
```
sns.countplot(x='term', hue='loan_status', data=data)
```

```
<Axes: xlabel='term', ylabel='count'>
```



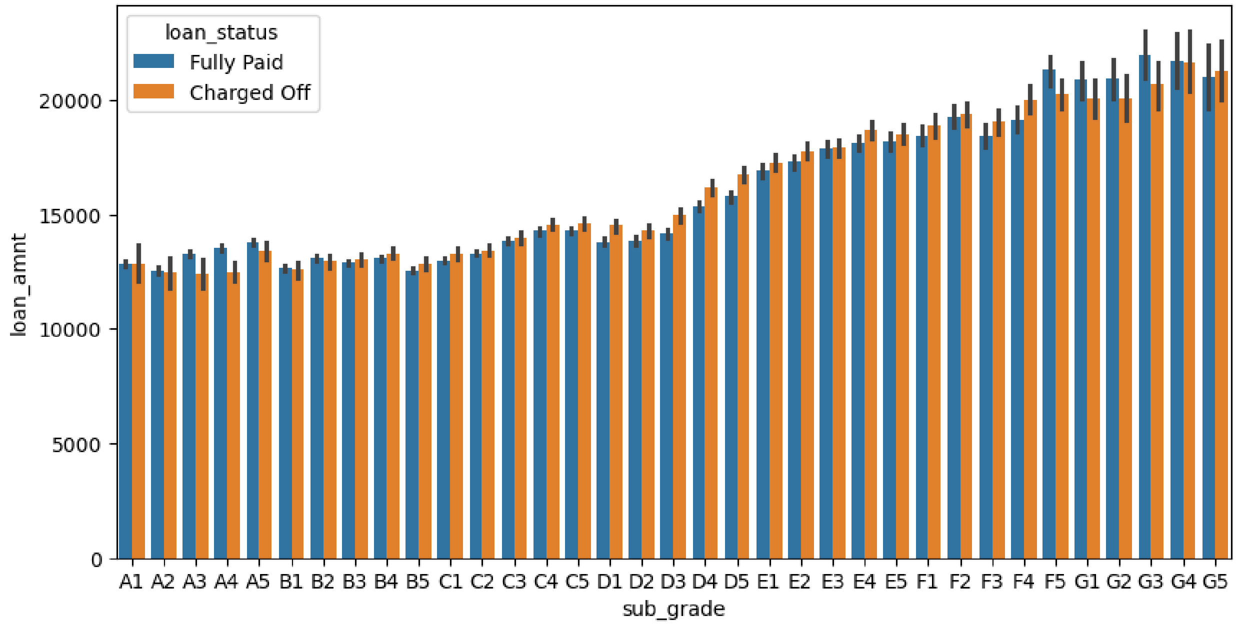

```
((data[data['loan_status']=='Charged Off'].groupby('sub_grade')
['loan_status'].count())/(data.groupby('sub_grade')
['sub_grade'].count())).plot(kind='bar')
```

```
<Axes: xlabel='sub_grade'>
```

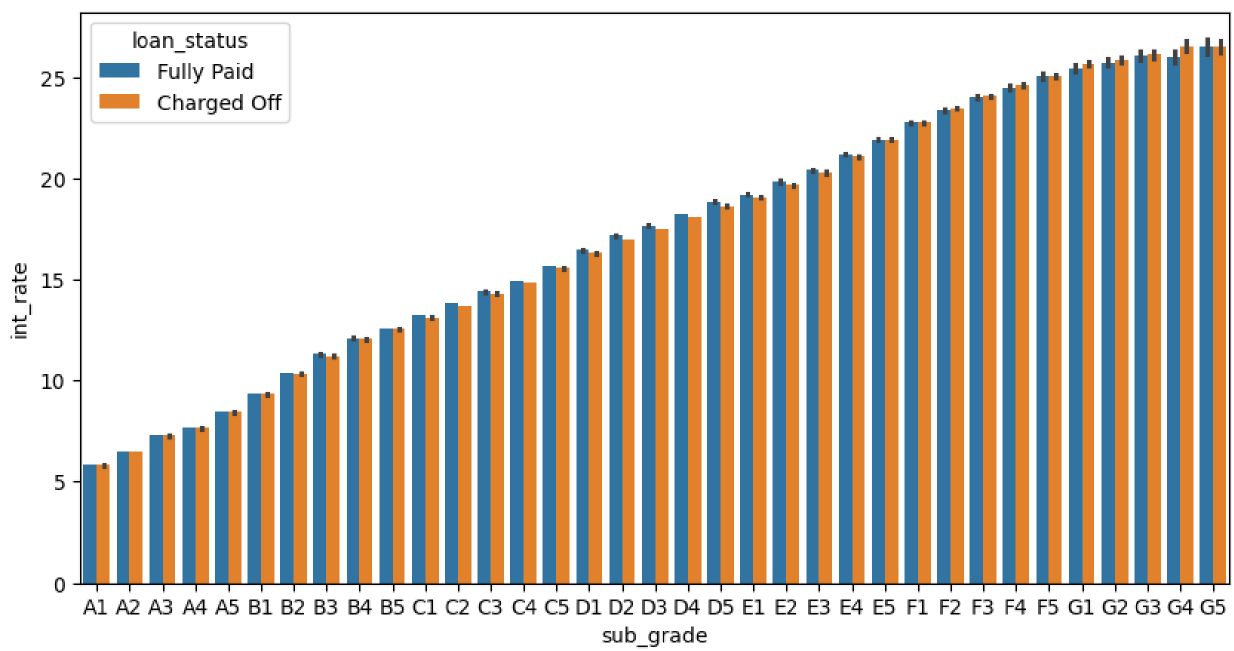



```
plt.figure(figsize=(10,5))
sns.barplot(x='sub_grade', y='loan_amnt', hue='loan_status', data=data,order=['A1','A2','A3','A4','A5','B1','B2','B3','B4','B5','C1','C2','C3','C4','C5','D1','D2','D3','D4','D5','E1','E2','E3','E4','E5','F1','F2','F3','F4','F5','G1','G2','G3','G4','G5'])
```

```
<Axes: xlabel='sub_grade', ylabel='loan_amnt'>
```




```
plt.figure(figsize=(10,5))
sns.barplot(x='sub_grade', y='int_rate', hue='loan_status',
data=data,order=['A1','A2','A3','A4','A5','B1','B2','B3','B4','B5','C1
','C2','C3','C4','C5','D1','D2','D3','D4','D5','E1','E2','E3','E4','E5
','F1','F2','F3','F4','F5','G1','G2','G3','G4','G5'])

<Axes: xlabel='sub_grade', ylabel='int_rate'>
```



```
#The 2 graphs above show that:
#a) the bank is putting more money in risky categories of borrowers
#b) the interest rates for fully paid borrowers and those charging off
are same in each category, which means the innocent borrowers are
being penalized for those who run away
#Recommendation:
#1. The bank should put more money into its group A & B customers
#2. For groups C & D where the default rate is high, the interest rate
should be tailored according to the borrowers recent history even if
it is not with the bank.
#This will be possible by collaborating with other banks and using a
blockchain solution to identify behaviours that imply default, even
though they are not financial related.
#So for example a customer's pattern of recent buying transactions in
a retail store or in automobile purchases, might indicate that he will
carry that same behaviour into his banking transactions.

data.isnull().sum()
```

```
loan_amnt                   0
term                        0
int_rate                    0
installment                 0
grade                       0
sub_grade                   0
emp_title               22927
emp_length              18301
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
loan_status                 0
purpose                     0
title                    1756
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
```

```
data.title.unique() #This is similar to the purpose column and can be
deleted
```

```
array(['Vacation', 'Debt consolidation', 'Credit card
refinancing', ...,
       'Credit buster ', 'Loanforpayoff', 'Toxic Debt Payoff'],
      dtype=object)
```

```
data.title.value_counts()
```

```
title
Debt consolidation            152472
Credit card refinancing        51487
Home improvement               15264
Other                          12930
Debt Consolidation             11608
                               ...
PayOffHighIntCreditCards           1
Heat my home                       1
Graduation/Travel Expenses         1
Daughter's Wedding Bill            1
Toxic Debt Payoff                  1
Name: count, Length: 48804, dtype: int64
```

```
data.emp_title.unique() #This does not provide any ideas on loan
repayment, so I can delete
```

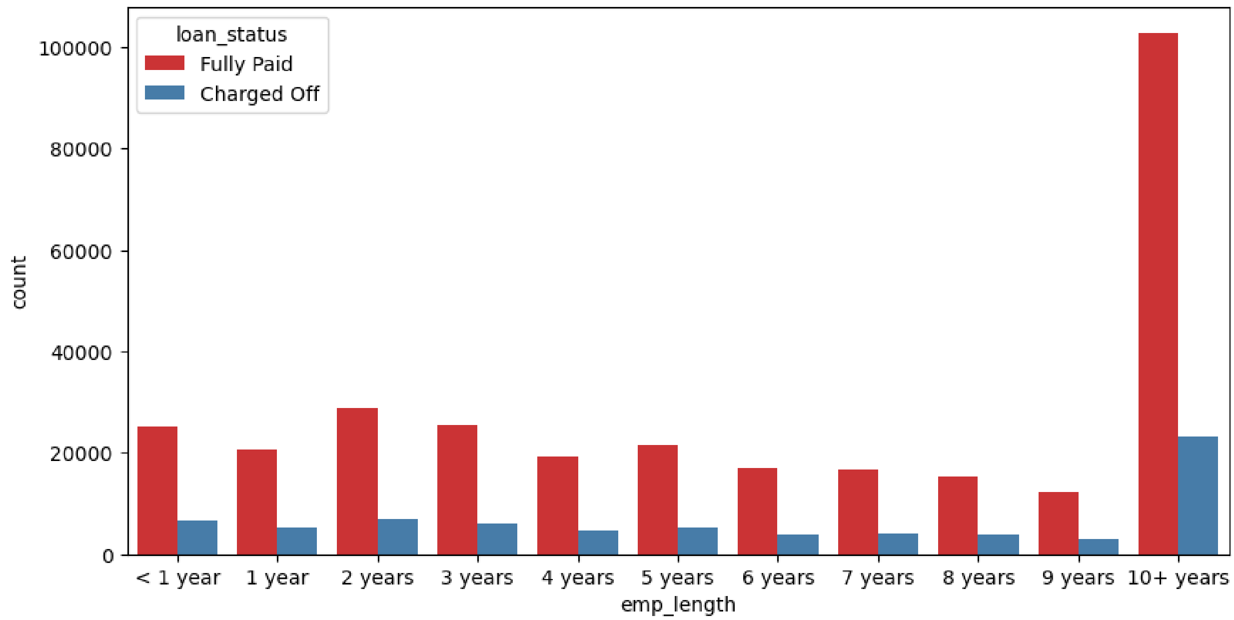
```
array(['Marketing', 'Credit analyst ', 'Statistician', ...,
       "Michael's Arts & Crafts", 'licensed bankere',
       'Gracon Services, Inc'], dtype=object)
```

```
data.emp_length.value_counts()
```

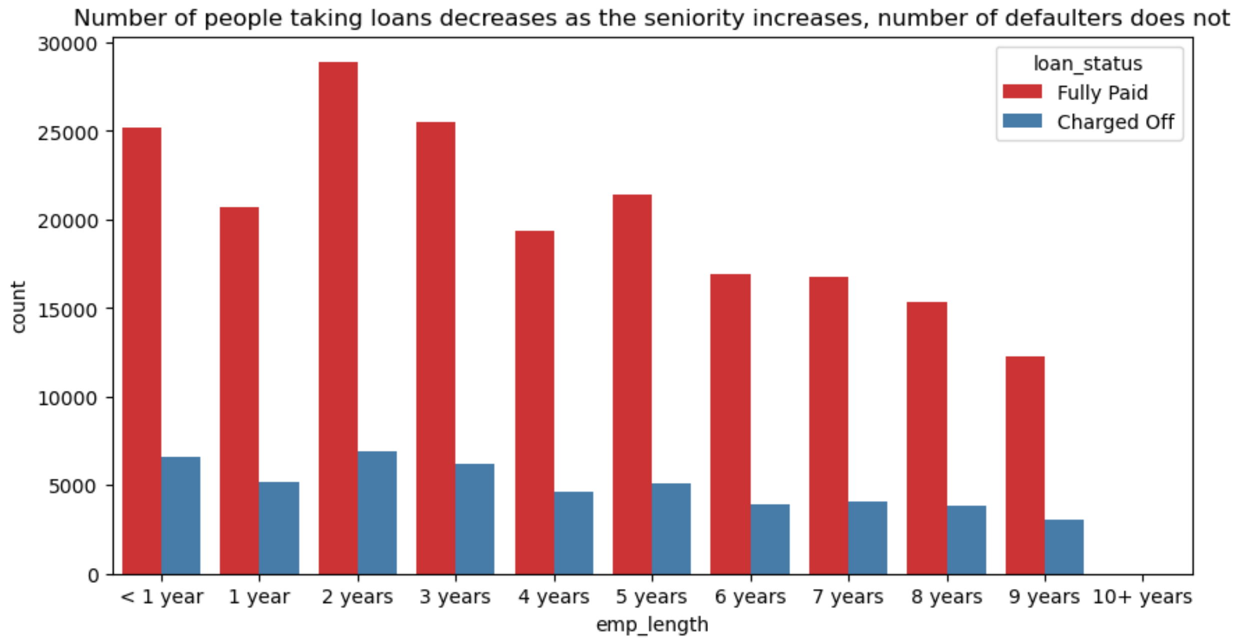
```
emp_length
10+ years    126041
2 years       35827
< 1 year      31725
3 years       31665
5 years       26495
1 year        25882
4 years       23952
6 years       20841
7 years       20819
8 years       19168
9 years       15314
Name: count, dtype: int64
```

```
plt.figure(figsize=(10,5))
sns.countplot(hue='loan_status', x='emp_length', order=['< 1 year', '1
year', '2 years','3 years', '4 years', '5 years', '6 years', '7
years', '8 years','9 years', '10+ years'],data=data, palette='Set1')
```

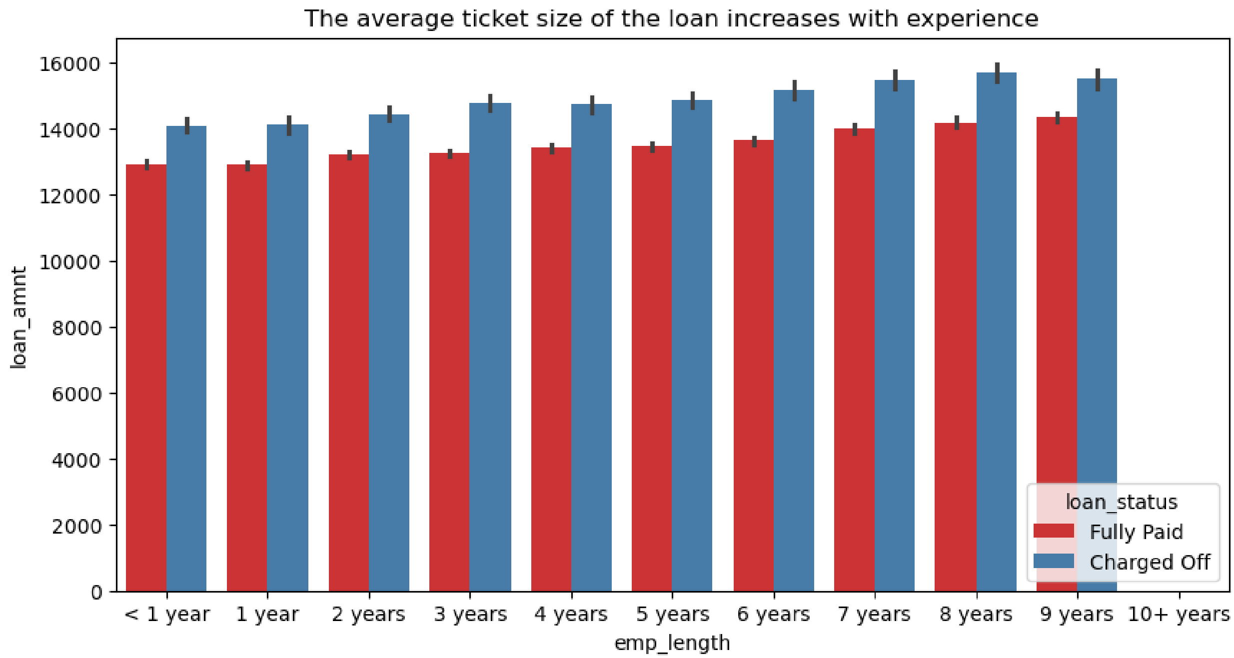
```
<Axes: xlabel='emp_length', ylabel='count'>
```




```
el_new=data[data['emp_length']!='10+ years']

plt.figure(figsize=(10,5))
sns.countplot(x='emp_length', hue='loan_status',order=['< 1 year', '1 year', '2 years','3 years', '4 years', '5 years', '6 years', '7 years', '8 years','9 years', '10+ years'],data=el_new, palette='Set1')
plt.title('Number of people taking loans decreases as the seniority increases, number of defaulters does not')
plt.show()
```

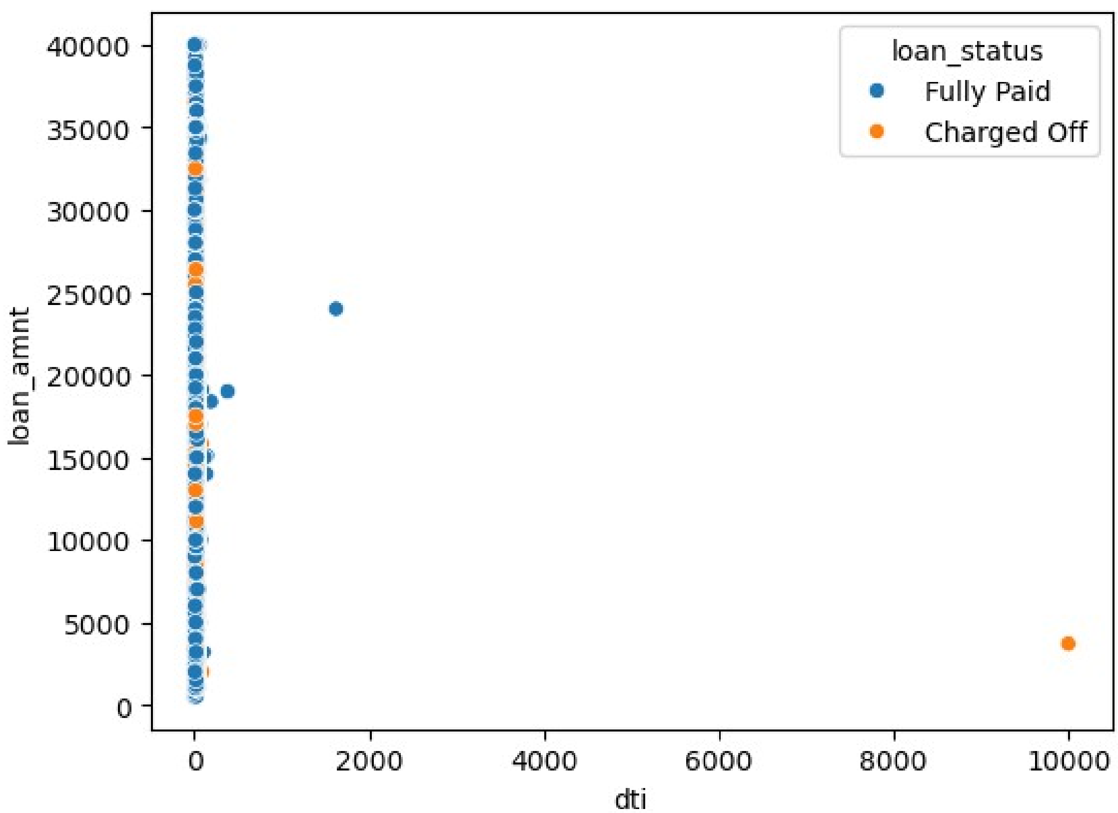



```
#The graph shows that as people spend more time in the company, the number of proportion of people paying the loan back is reducing, as long as it is <=10 years
#So my recommendation is to have a higher interest rate for people who are longer serving

plt.figure(figsize=(10,5))
sns.barplot(x='emp_length', y='loan_amnt', hue='loan_status',order=['< 1 year', '1 year', '2 years','3 years', '4 years', '5 years', '6 years', '7 years', '8 years','9 years', '10+ years'],data=el_new, palette='Set1')
plt.title('The average ticket size of the loan increases with experience')
plt.show()
```



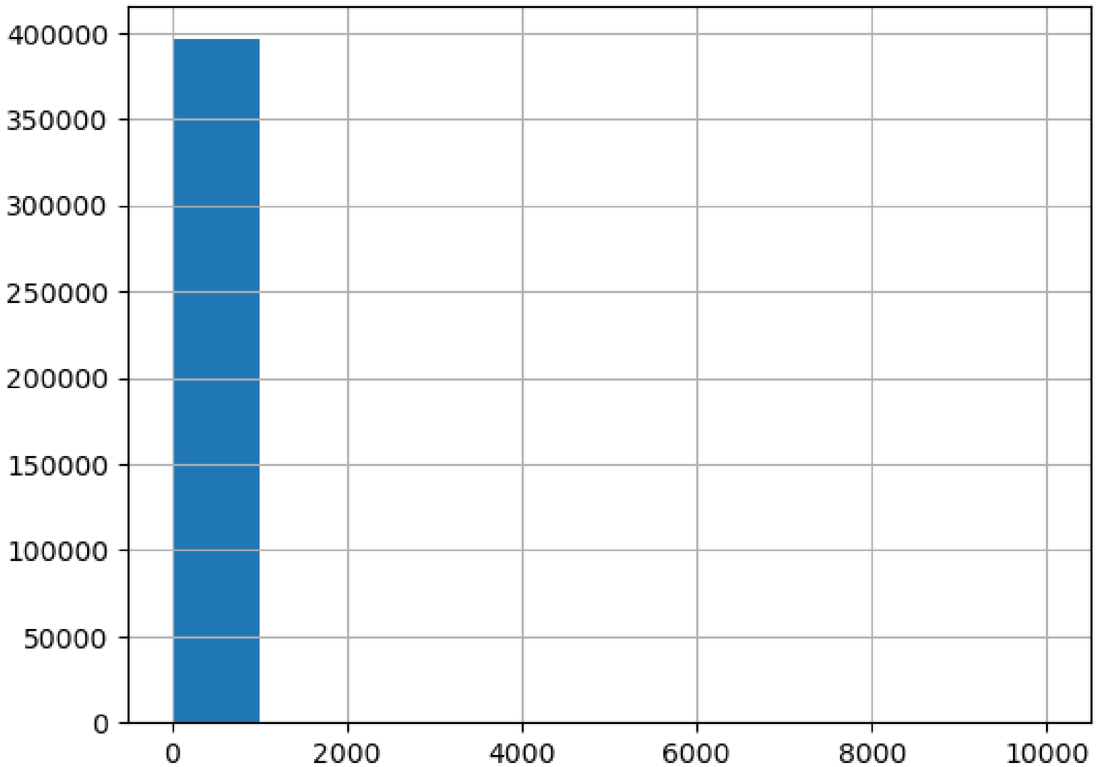

```
sns.scatterplot(x='dti',y='loan_amnt', hue='loan_status', data=data)
```

```
<Axes: xlabel='dti', ylabel='loan_amnt'>
```



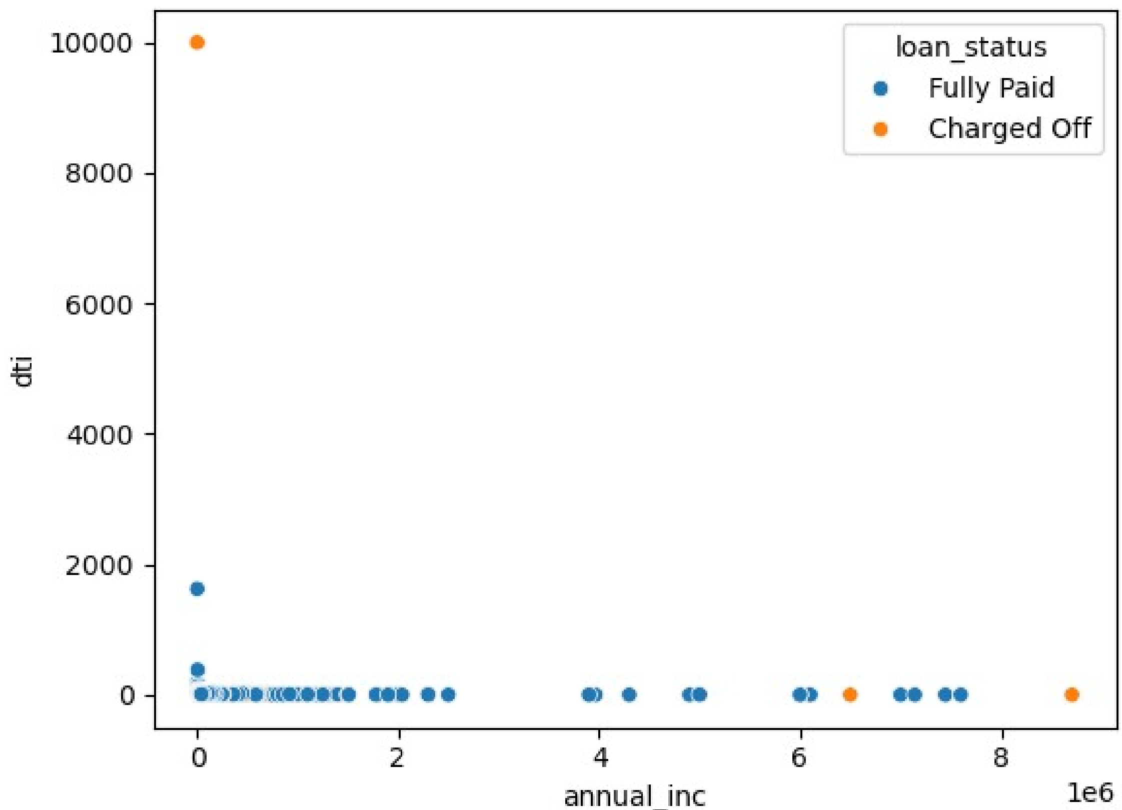
```
data.dti.hist()
```

```
<Axes: >
```



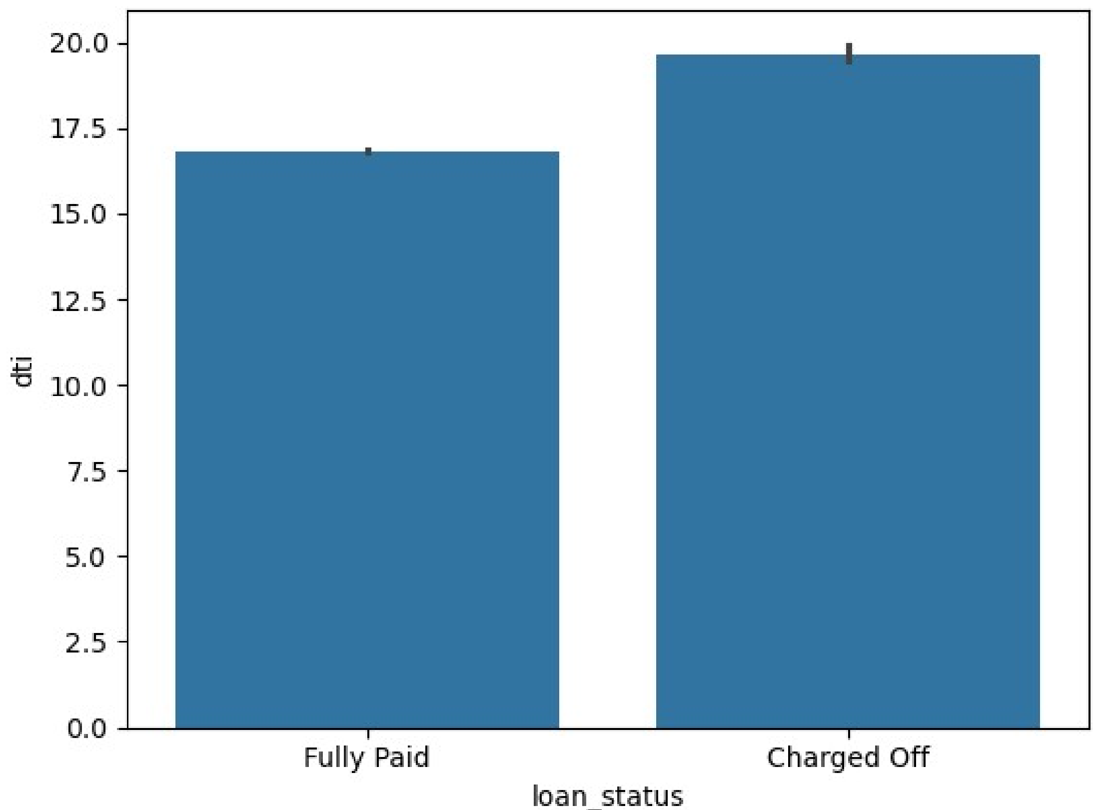

```
sns.scatterplot(x='annual_inc', y='dti', hue='loan_status', data=data)
```

```
<Axes: xlabel='annual_inc', ylabel='dti'>
```



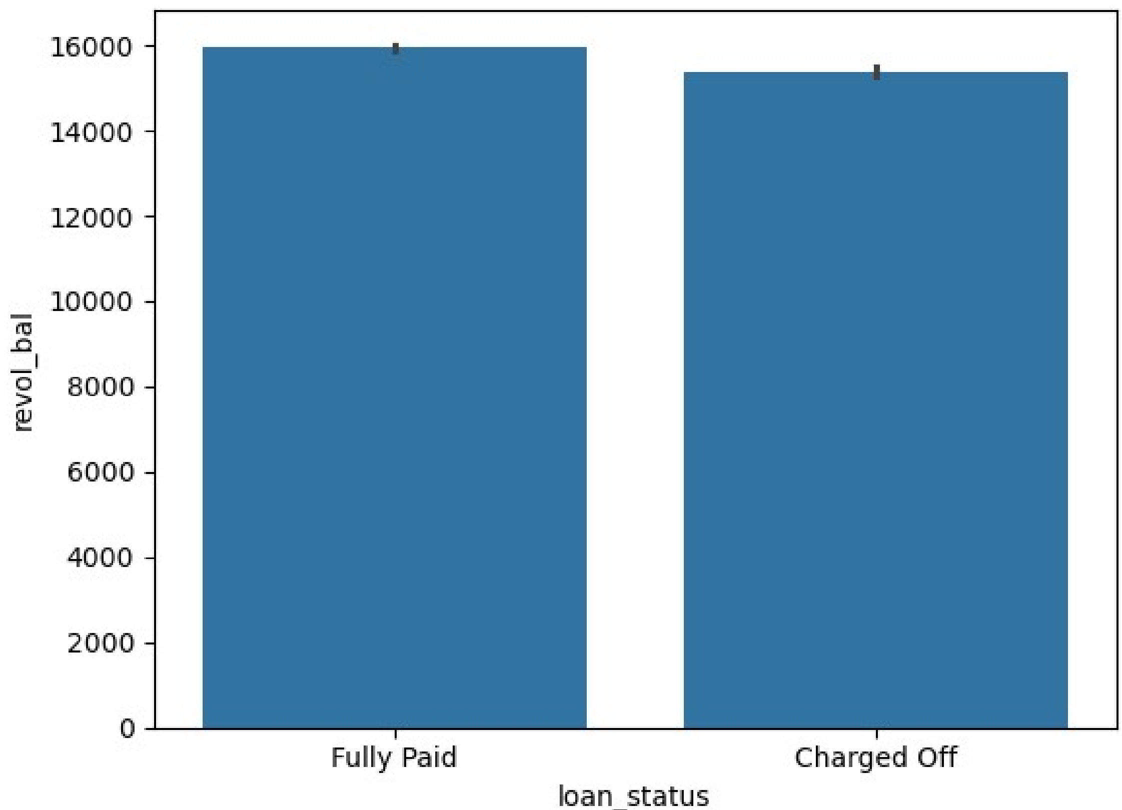

```
sns.barplot(x='loan_status', y='dti', data=data)
```

```
<Axes: xlabel='loan_status', ylabel='dti'>
```



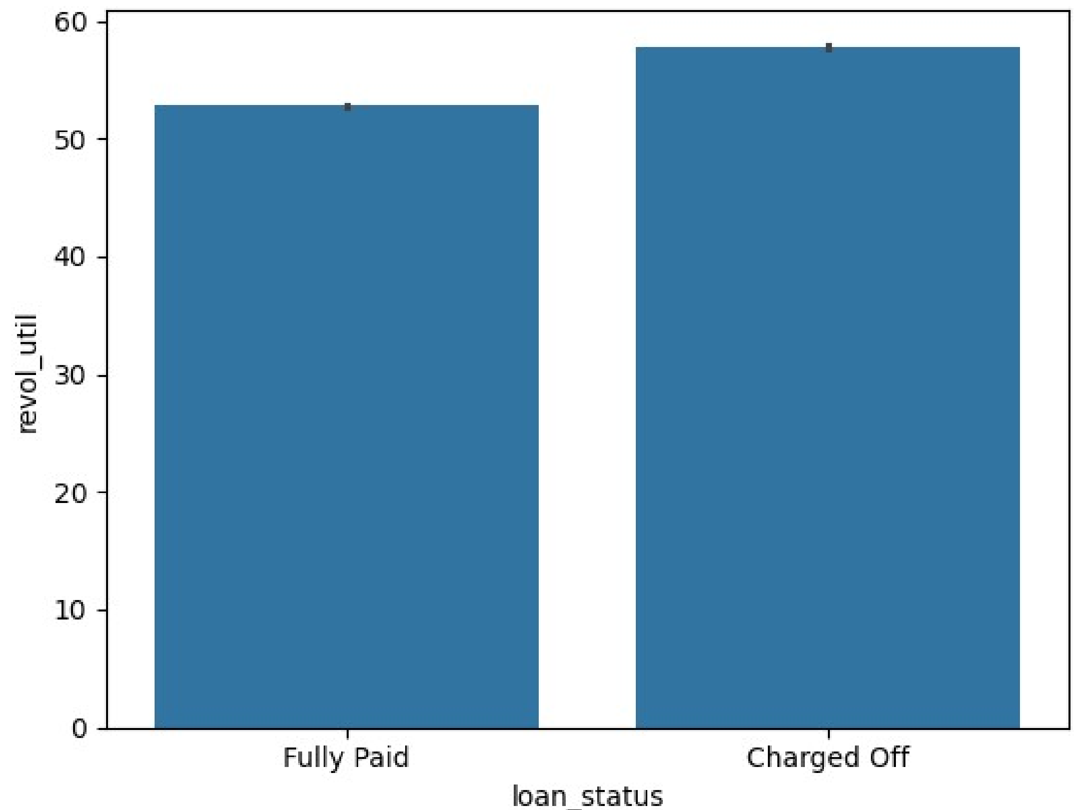

```
sns.barplot(x='loan_status',y='revol_bal', data=data) #This is too
small a difference so the feature can be removed
```

```
<Axes: xlabel='loan_status', ylabel='revol_bal'>
```



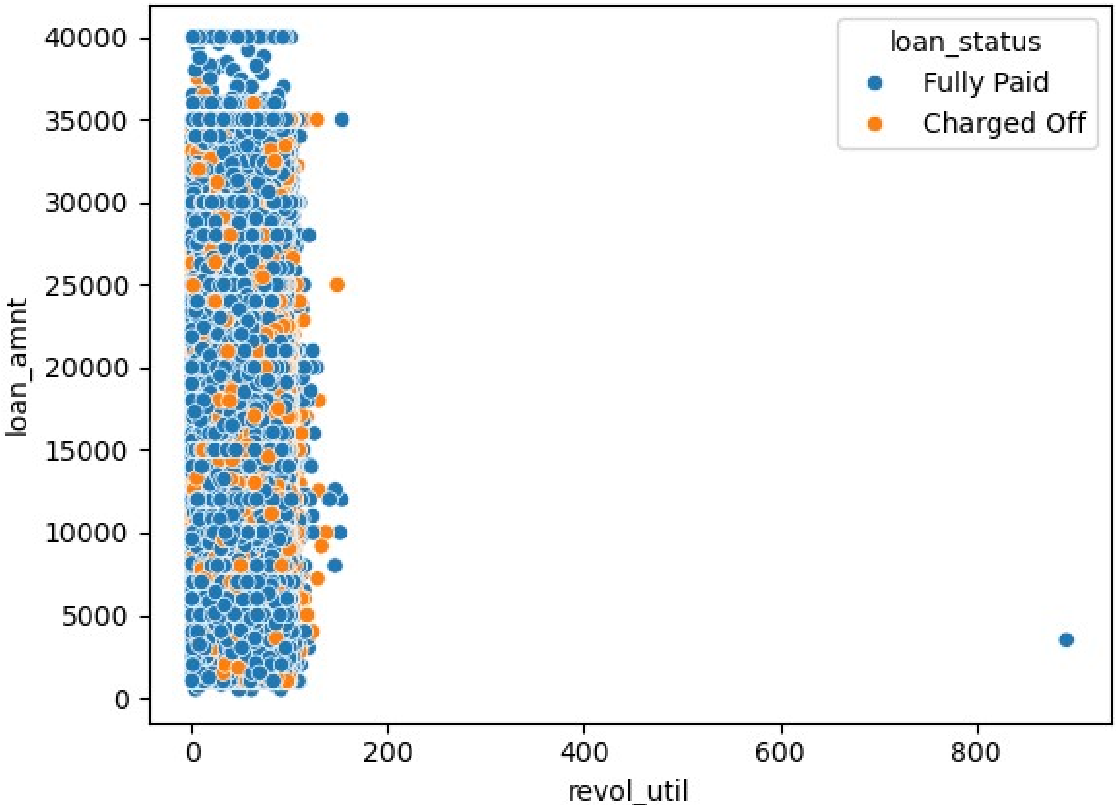

```
sns.barplot(x='loan_status', y='revol_util', data=data) #This is a %
figure so it makes sense to keep it, the 5% or so difference is
significant
```

```
<Axes: xlabel='loan_status', ylabel='revol_util'>
```

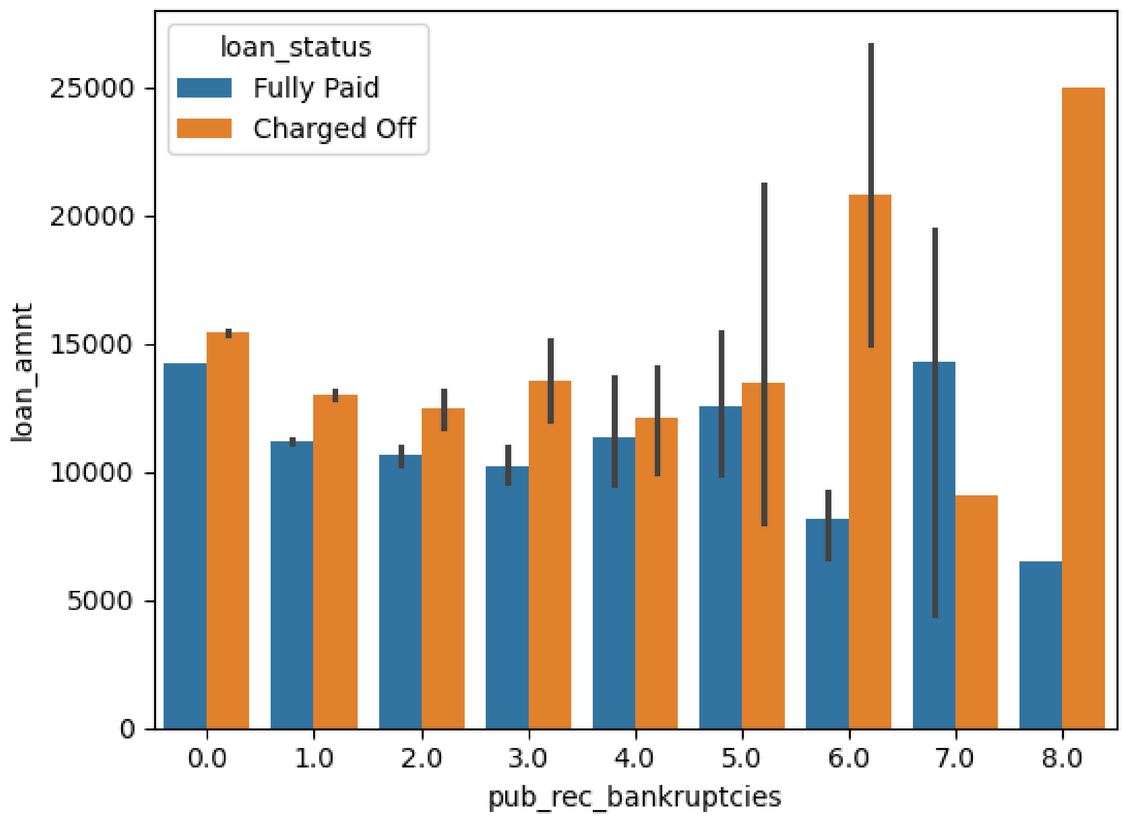



```
sns.scatterplot(x='revol_util', y='loan_amnt', hue='loan_status',
data=data) #There appears to be no correlation between the revolving
utilization and the loan status or loan amount, so we can remove it
```

```
<Axes: xlabel='revol_util', ylabel='loan_amnt'>
```

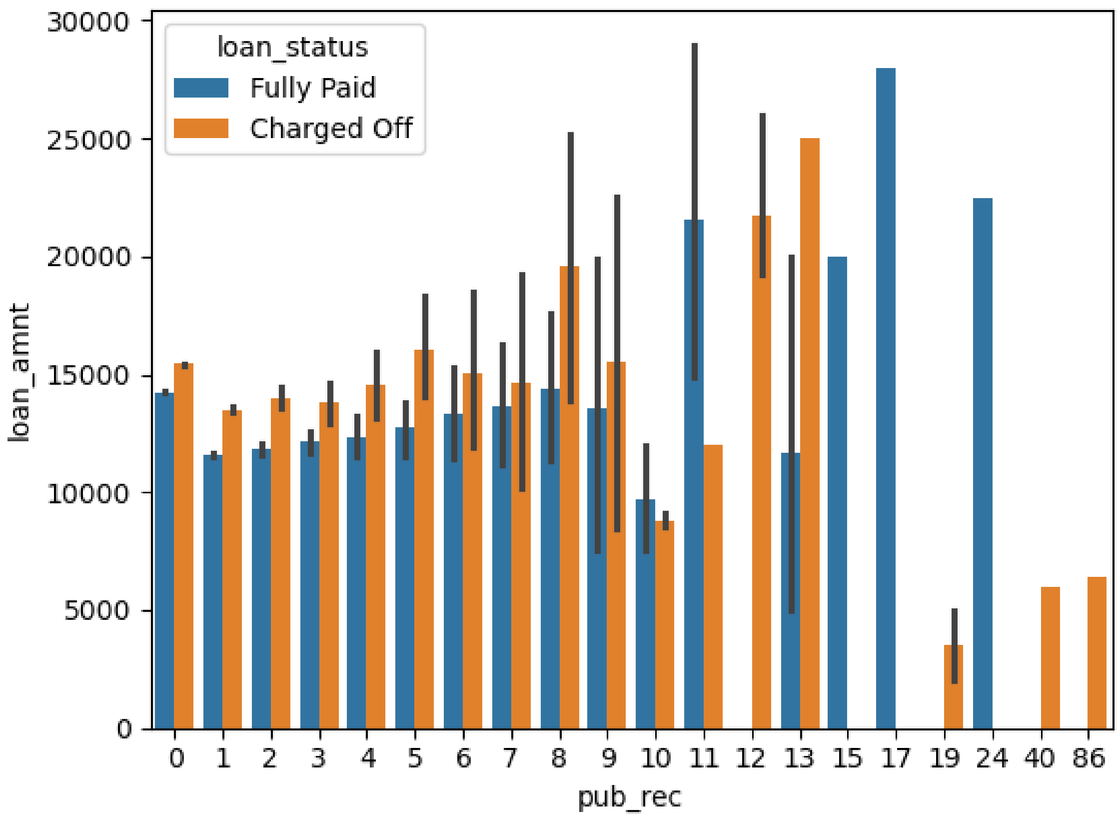



```
sns.barplot(x='pub_rec_bankruptcies', y='loan_amnt',
hue='loan_status', data=data)
#So in any case, the money loaned to people who defect is more than
that loaned to people who repay
#Also if the number of bankruptcies is above 5, the bank is giving
even more money to the defaulters relative to the people who are
paying

<Axes: xlabel='pub_rec_bankruptcies', ylabel='loan_amnt'>
```



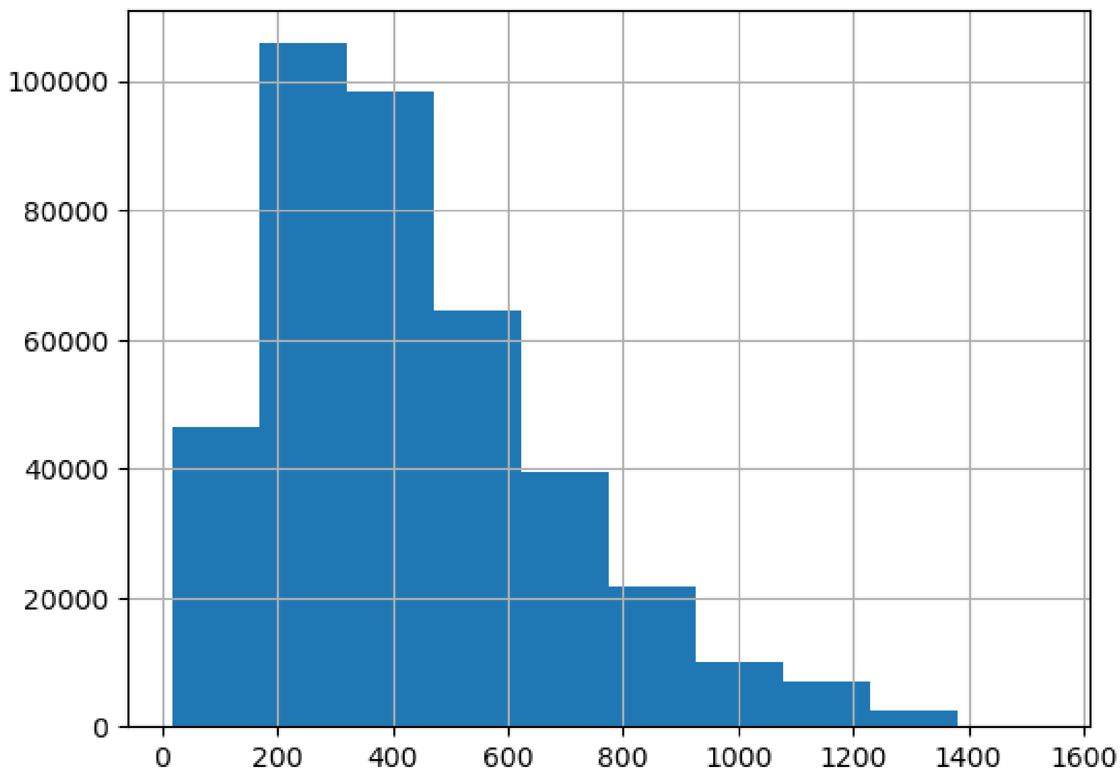

```
sns.barplot(x='pub_rec', y='loan_amnt', hue='loan_status', data=data)
```

```
<Axes: xlabel='pub_rec', ylabel='loan_amnt'>
```



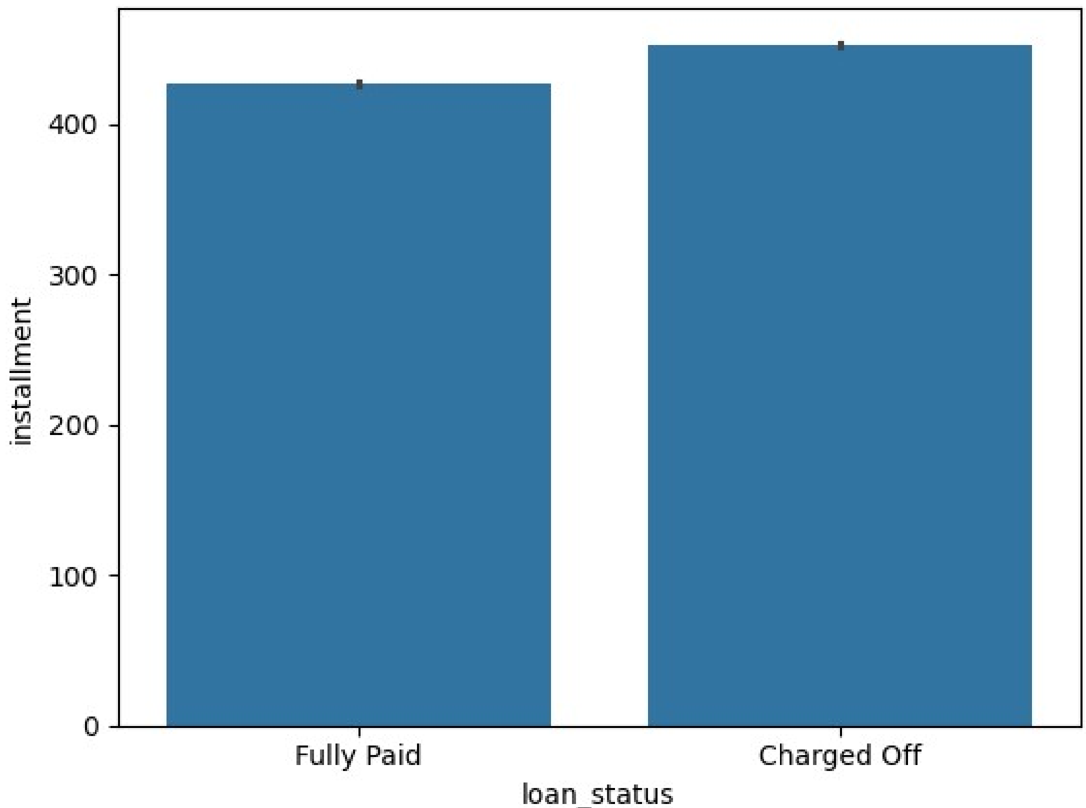

```
data.installment.hist()
```

```
<Axes: >
```



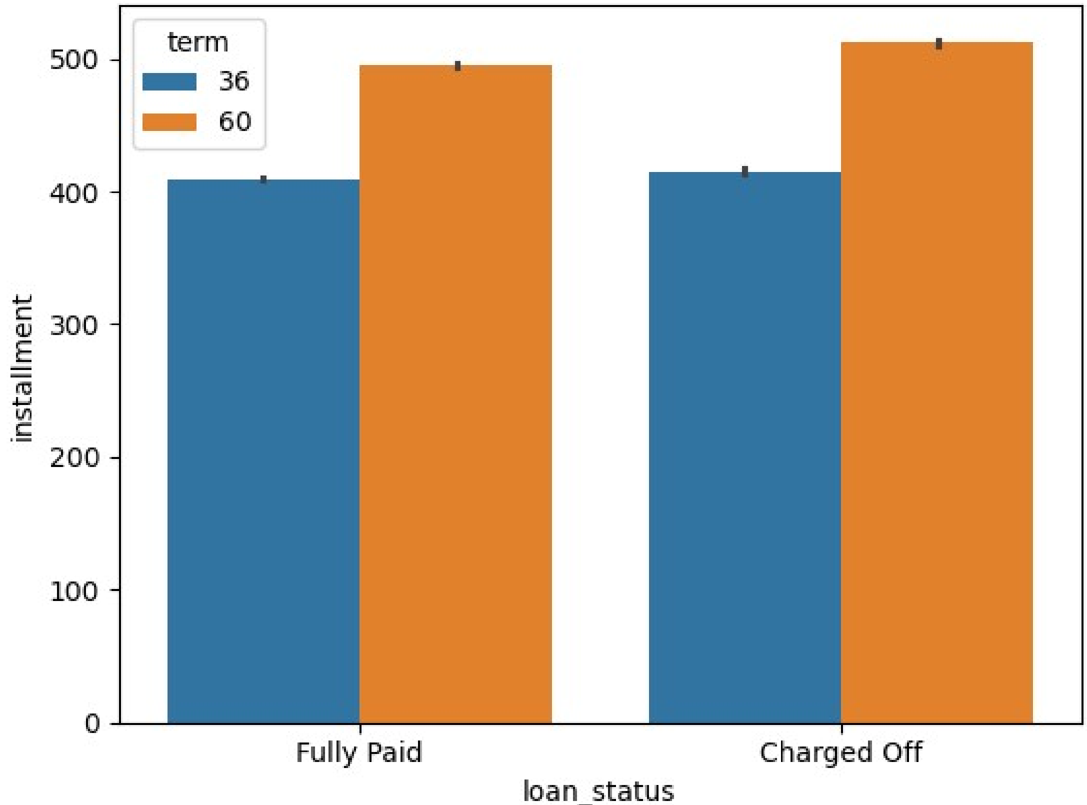

```
sns.barplot(x='loan_status', y='installment', data=data)
```

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




```
sns.barplot(hue='term', y='installment', x='loan_status', data=data)
```

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




```
features_to_drop=['emp_title', 'issue_d', 'loan_status',
'title','earliest_cr_line', 'open_acc',
'pub_rec_bankruptcies','pub_rec', 'revol_util', 'revol_bal',
'total_acc', 'installment','mort_acc','address', 'grade']



data.emp_length.isnull().sum()/data.shape[0] #only 4.6% of the records
are missing the employee length column, so I will drop the missing
rows

0.046211145620281294

data.drop(data[data.emp_length.isnull()].index, inplace=True)

data=data[data.emp_length!='10+ years']

x=data.drop(features_to_drop, axis=1)

x.shape

(251688, 12)

x.isnull().sum()
```

```
loan_amnt               0
term                    0
int_rate                0
sub_grade               0
emp_length              0
home_ownership          0
annual_inc              0
verification_status     0
purpose                 0
dti                     0
initial_list_status     0
application_type        0
dtype: int64
```

```
#x[x.emp_length.notnull()]['loan_amnt'].describe()
```

```
y=data['loan_status']
```

```
y
```

```
1          Fully Paid
2          Fully Paid
3          Fully Paid
4         Charged Off
6          Fully Paid
             ...
396017     Fully Paid
396022     Fully Paid
396024     Fully Paid
396025     Fully Paid
396026     Fully Paid
Name: loan_status, Length: 251688, dtype: object
```

```
y.replace(['Fully Paid','Charged Off'],[0,1], inplace=True)
```

```
/var/folders/c5/jzdz65t53qj6w2j6h9rj5sr80000gn/T/
ipykernel_8360/4151647945.py:1: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To
retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior,
set `pd.set_option('future.no_silent_downcasting', True)`
  y.replace(['Fully Paid','Charged Off'],[0,1], inplace=True)
```

```
x.columns
```

```
Index(['loan_amnt', 'term', 'int_rate', 'sub_grade', 'emp_length',
       'home_ownership', 'annual_inc', 'verification_status',
'purpose', 'dti',
```

```
       'initial_list_status', 'application_type'],
      dtype='object')
```

```
x.dtypes
```

```
loan_amnt                 int64
term                     object
int_rate                float64
sub_grade                object
emp_length               object
home_ownership           object
annual_inc              float64
verification_status      object
purpose                  object
dti                     float64
initial_list_status      object
application_type         object
dtype: object
```

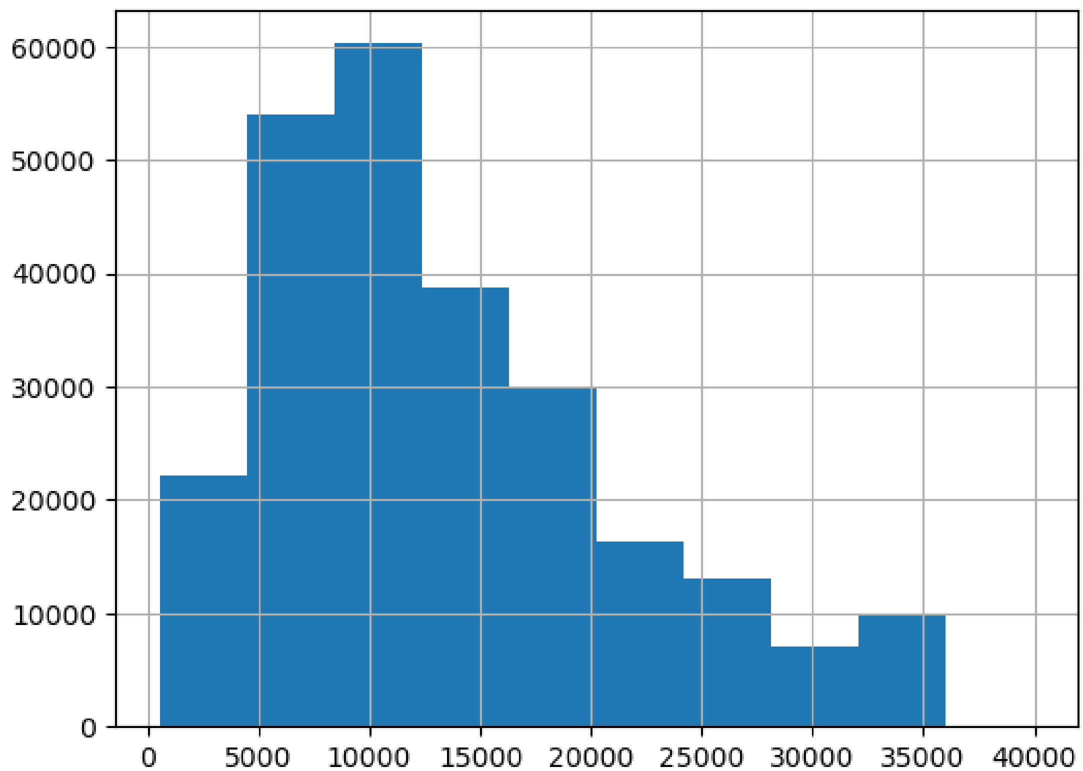
```
#Cleaning the data
```

```
x.loan_amnt.describe()
```

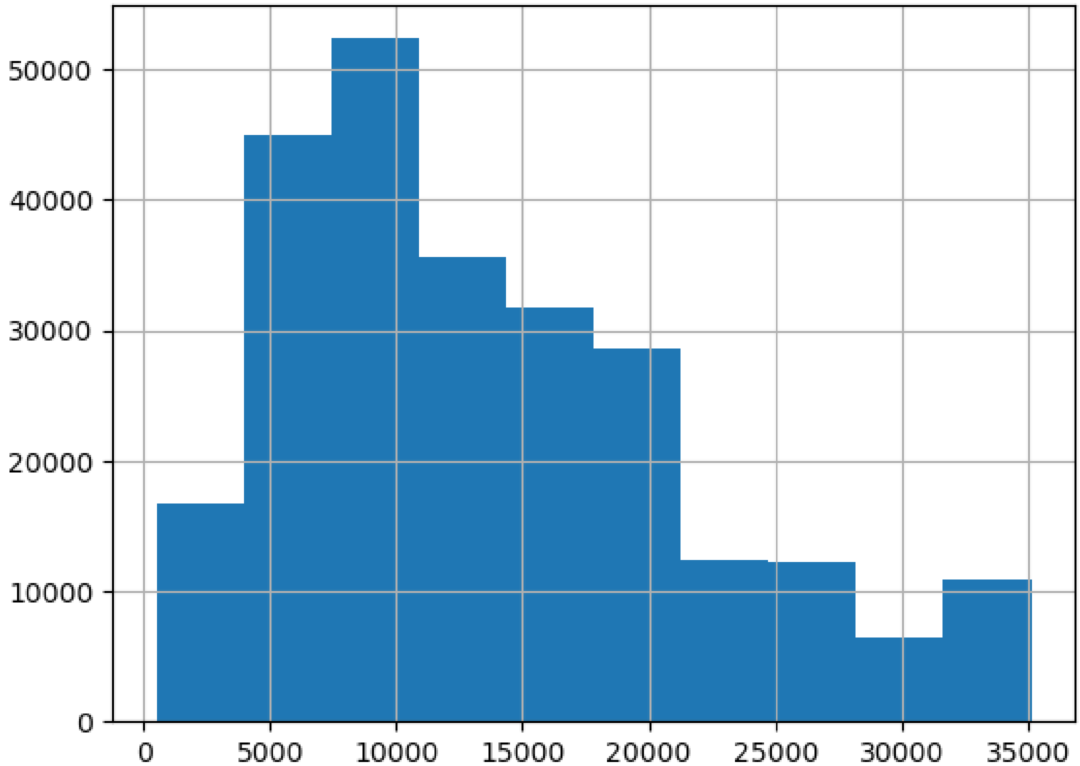
```
count    251688.000000
mean      13703.230885
std        8183.898226
min         500.000000
25%        7500.000000
50%       12000.000000
75%       18550.000000
max       40000.000000
Name: loan_amnt, dtype: float64
```

```
x.loan_amnt.hist()
```

```
<Axes: >
```




```
iqr_la=x.loan_amnt.quantile(.75)-x.loan_amnt.quantile(.25)

uql=x.loan_amnt.quantile(.75)+(1.5*iqr_la)

uql

35125.0

x.loan_amnt=[uql if i>uql else i for i in x.loan_amnt]

x.loan_amnt.hist()

<Axes: >
```



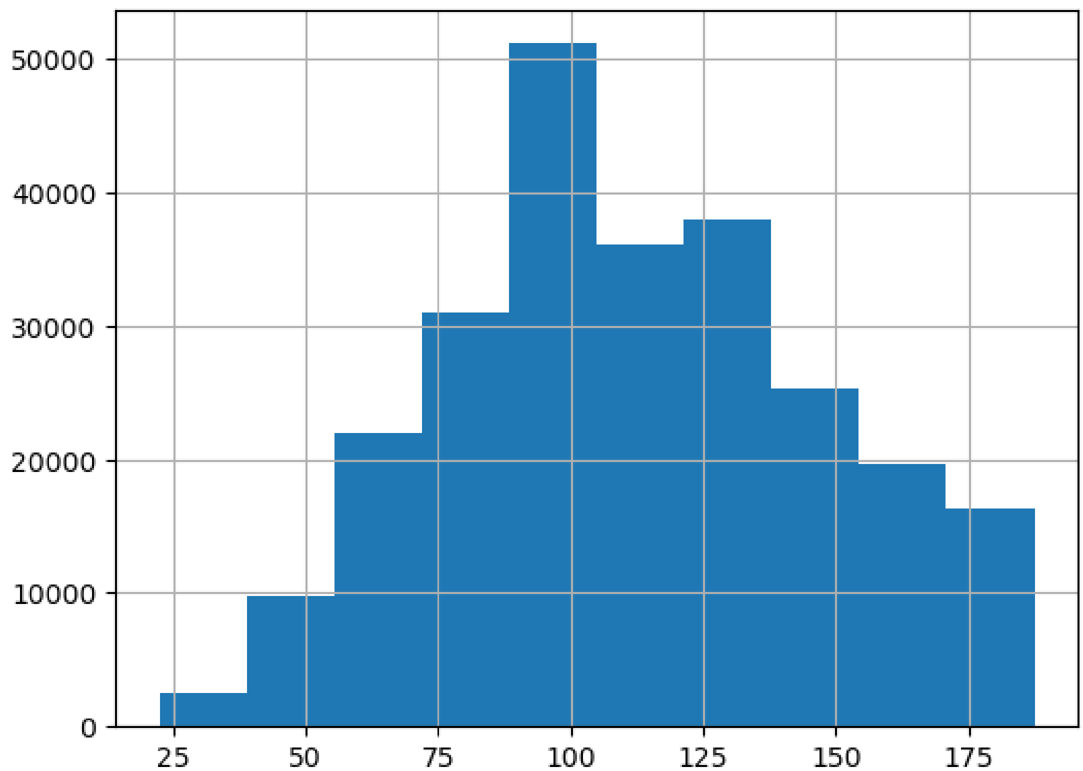

```python
import scipy
from scipy.stats import shapiro

shapiro(x.loan_amnt) #This shows the data is not normal
```

```
/opt/anaconda3/lib/python3.12/site-packages/scipy/stats/
_axis_nan_policy.py:531: UserWarning: scipy.stats.shapiro: For N >
5000, computed p-value may not be accurate. Current N is 251688.
  res = hypotest_fun_out(*samples, **kwds)
```

```
ShapiroResult(statistic=0.9335602104856013,
pvalue=1.4959779862908053e-119)
```

```python
x.loan_amnt=np.sqrt(x.loan_amnt)

x.loan_amnt.hist() #This looks normal now
```

```
<Axes: >
```

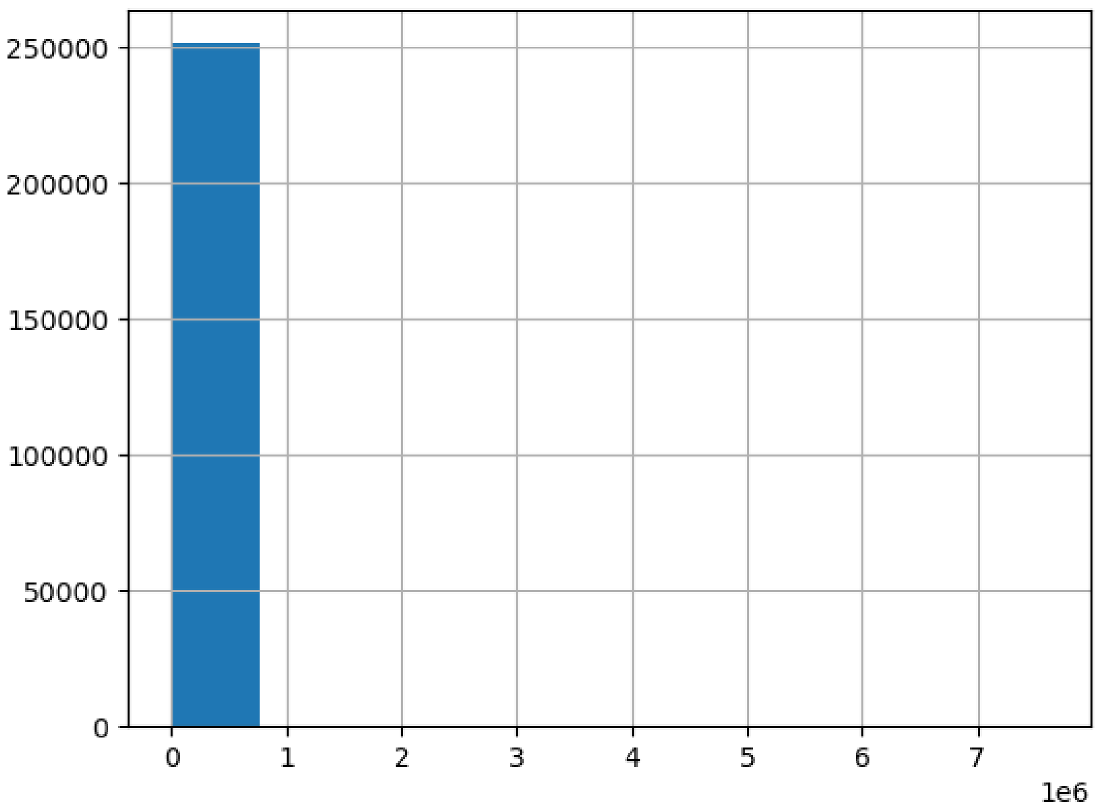



```
x.annual_inc.describe()
```

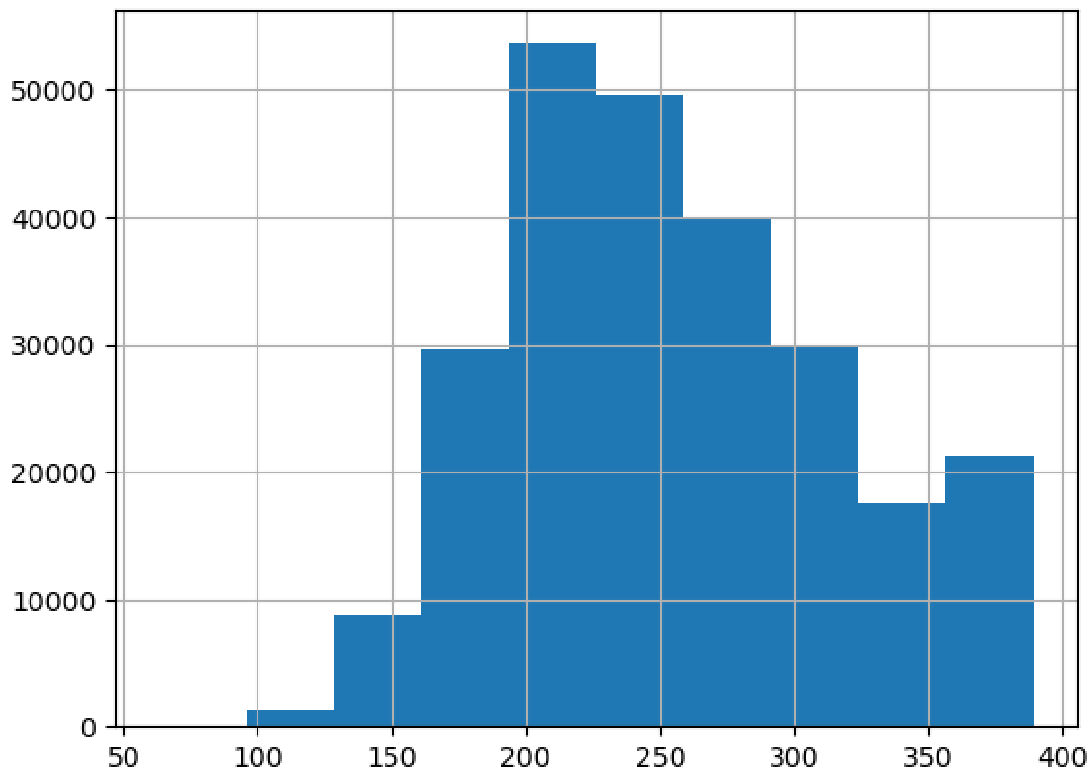
```
count    2.516880e+05
mean     7.246300e+04
std      5.657586e+04
min      4.000000e+03
25%      4.400000e+04
50%      6.000000e+04
75%      8.700000e+04
max      7.600000e+06
Name: annual_inc, dtype: float64
```

```
x.annual_inc.hist()
```

```
<Axes: >
```




```
ann_inc_iqr=x.annual_inc.quantile(.75)-x.annual_inc.quantile(.25)

uql_ann_inc=x.annual_inc.quantile(.75)+(1.5*ann_inc_iqr)

x.annual_inc=[uql_ann_inc if i>=uql_ann_inc else i for i in
x.annual_inc]

x.annual_inc=np.sqrt(x.annual_inc) #This looks more normal to me than
earlier

x.annual_inc.hist()

<Axes: >
```




```
sorted(x.sub_grade.unique())
```

```
['A1',
 'A2',
 'A3',
 'A4',
 'A5',
 'B1',
 'B2',
 'B3',
 'B4',
 'B5',
 'C1',
 'C2',
 'C3',
 'C4',
 'C5',
 'D1',
 'D2',
 'D3',
 'D4',
 'D5',
 'E1',
 'E2',
 'E3',
```

```
 'E4',
 'E5',
 'F1',
 'F2',
 'F3',
 'F4',
 'F5',
 'G1',
 'G2',
 'G3',
 'G4',
 'G5']
```

```
x.sub_grade.replace(['A1','A2','A3','A4','A5','B1','B2','B3','B4','B5'
,'C1','C2','C3','C4','C5','D1','D2','D3','D4','D5','E1','E2','E3','E4'
,'E5','F1','F2','F3','F4','F5','G1','G2','G3','G4','G5'],list(range(35
)), inplace=True)
```

```
/var/folders/c5/jzdz65t53qj6w2j6h9rj5sr80000gn/T/
ipykernel_8360/2111346709.py:1: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.



x.sub_grade.replace(['A1','A2','A3','A4','A5','B1','B2','B3','B4','B5'
,'C1','C2','C3','C4','C5','D1','D2','D3','D4','D5','E1','E2','E3','E4'
,'E5','F1','F2','F3','F4','F5','G1','G2','G3','G4','G5'],list(range(35
)), inplace=True)
/var/folders/c5/jzdz65t53qj6w2j6h9rj5sr80000gn/T/ipykernel_8360/211134
6709.py:1: FutureWarning: Downcasting behavior in `replace` is
deprecated and will be removed in a future version. To retain the old
behavior, explicitly call `result.infer_objects(copy=False)`. To opt-
in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
```

```
x.sub_grade.replace(['A1','A2','A3','A4','A5','B1','B2','B3','B4','B5'
,'C1','C2','C3','C4','C5','D1','D2','D3','D4','D5','E1','E2','E3','E4'
,'E5','F1','F2','F3','F4','F5','G1','G2','G3','G4','G5'],list(range(35
)), inplace=True)
```

```
x.sub_grade
```

```
1          9
2          7
3          1
4         14
6          0
          ..
396017     7
396022    10
396024     8
396025     8
396026    10
Name: sub_grade, Length: 251688, dtype: int64
```

```
x.home_ownership.value_counts()
```

```
home_ownership
RENT        117718
MORTGAGE    111973
OWN          21893
OTHER           83
NONE            18
ANY              3
Name: count, dtype: int64
```

```
#I am clubbing the categories "ANY" with "OTHER" in home ownership as
otherwise the categories will be too small
x.home_ownership.replace('ANY','OTHER', inplace=True)
```

```
/var/folders/c5/jzdz65t53qj6w2j6h9rj5sr80000gn/T/
ipykernel_8360/336726384.py:2: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  x.home_ownership.replace('ANY','OTHER', inplace=True)
```

```
x.home_ownership.value_counts()
```

```
home_ownership
RENT        117718
MORTGAGE    111973
OWN          21893
OTHER           86
```

```
NONE             18
Name: count, dtype: int64
```

```
dummies_ho=pd.get_dummies(x.home_ownership, drop_first=True)
```

```
x=pd.concat([x.drop('home_ownership', axis=1), dummies_ho], axis=1)
```

```
x.verification_status.unique()
#since there is an order here, source verified is better than not
verified and verified is better than source verified, I will replace
it as 0,1,2
```

```
array(['Not Verified', 'Source Verified', 'Verified'], dtype=object)
```

```
x.verification_status.replace(['Not Verified', 'Source Verified',
'Verified'],[0,1,2], inplace=True)
```

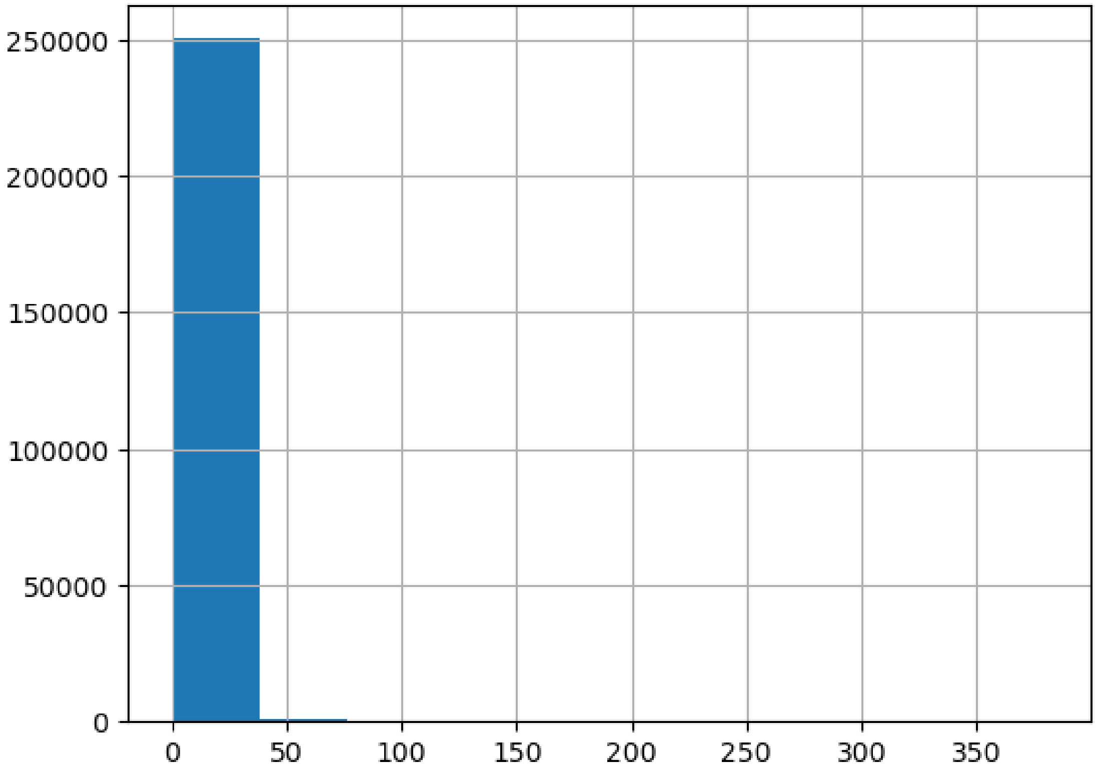
```
/var/folders/c5/jzdz65t53qj6w2j6h9rj5sr80000gn/T/
ipykernel_8360/1240797381.py:1: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  x.verification_status.replace(['Not Verified', 'Source Verified',
'Verified'],[0,1,2], inplace=True)
/var/folders/c5/jzdz65t53qj6w2j6h9rj5sr80000gn/T/ipykernel_8360/124079
7381.py:1: FutureWarning: Downcasting behavior in `replace` is
deprecated and will be removed in a future version. To retain the old
behavior, explicitly call `result.infer_objects(copy=False)`. To opt-
in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  x.verification_status.replace(['Not Verified', 'Source Verified',
'Verified'],[0,1,2], inplace=True)
```

```
dummies_purpose=pd.get_dummies(x.purpose, drop_first=True)
```

```
x=pd.concat([x.drop('purpose', axis=1), dummies_purpose], axis=1)
```

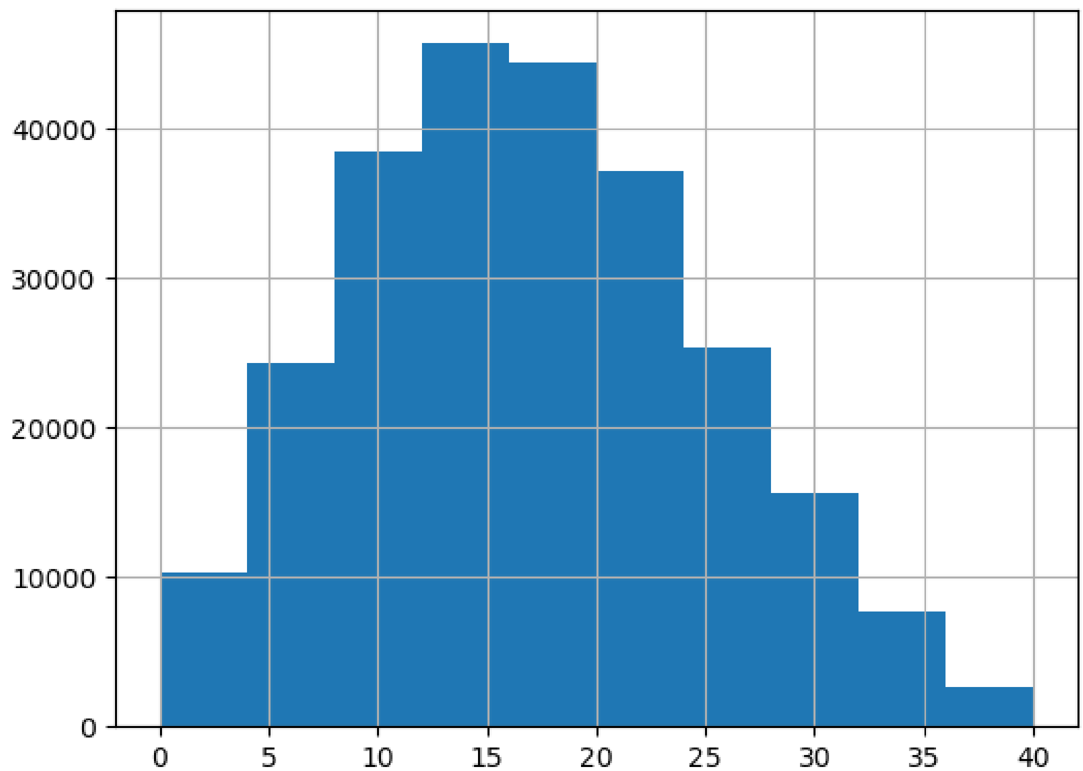
```
x.dti.hist()
```

```
<Axes: >
```




```
iqr_dti=x.dti.quantile(.75)-x.dti.quantile(.25)
uql_dti=x.dti.quantile(.75)+(1.5*iqr_dti)

x.dti=[uql_dti if i>=uql_dti else i for i in x.dti]

x.dti.hist()
```

```
<Axes: >
```




```
x.initial_list_status.replace(['w','f'],[0,1], inplace=True)

/var/folders/c5/jzdz65t53qj6w2j6h9rj5sr80000gn/T/
ipykernel_8360/1420288000.py:1: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  x.initial_list_status.replace(['w','f'],[0,1], inplace=True)
/var/folders/c5/jzdz65t53qj6w2j6h9rj5sr80000gn/T/ipykernel_8360/142028
8000.py:1: FutureWarning: Downcasting behavior in `replace` is
deprecated and will be removed in a future version. To retain the old
behavior, explicitly call `result.infer_objects(copy=False)`. To opt-
in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  x.initial_list_status.replace(['w','f'],[0,1], inplace=True)

x.initial_list_status.unique()
```

```
array([1, 0])
```

```
x.application_type.value_counts()
```

```
application_type
INDIVIDUAL    251285
JOINT            226
DIRECT_PAY       177
Name: count, dtype: int64
```

```
dummies_app_type=pd.get_dummies(x.application_type, drop_first=True)
```

```
x=pd.concat([x.drop('application_type', axis=1), dummies_app_type],
axis=1)
```

```
x.emp_length.unique()
```

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

```
x.emp_length.value_counts()
```

```
emp_length
2 years     35827
< 1 year    31725
3 years     31665
5 years     26495
1 year      25882
4 years     23952
6 years     20841
7 years     20819
8 years     19168
9 years     15314
Name: count, dtype: int64
```

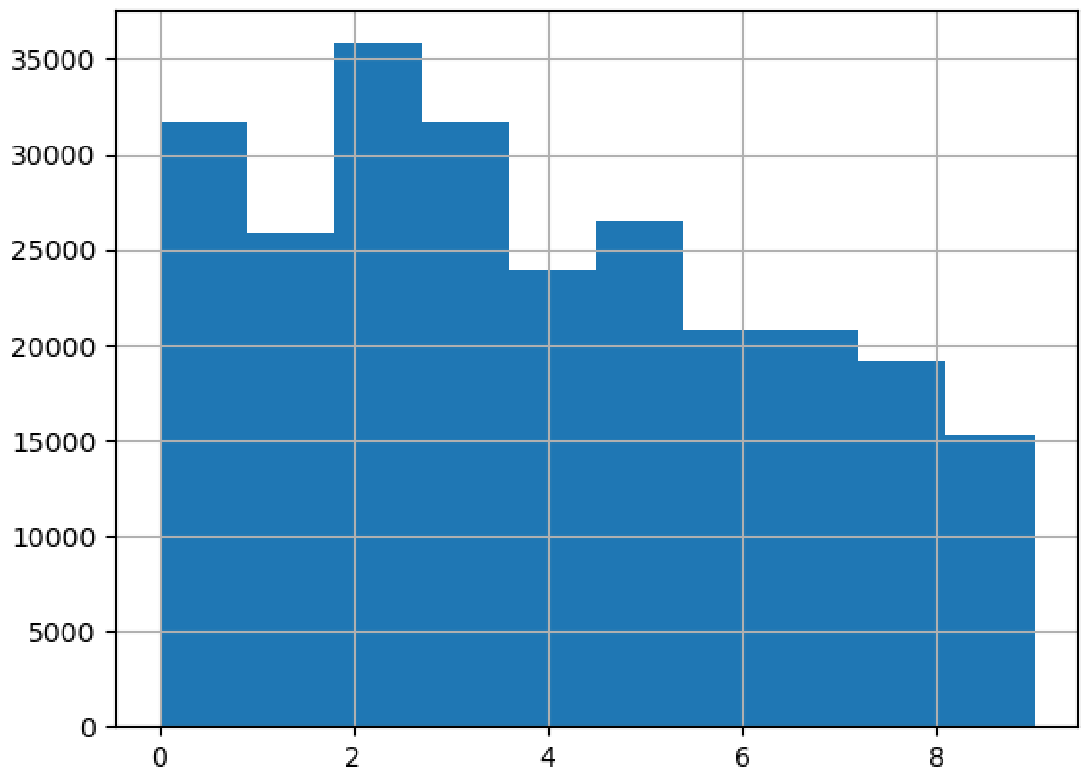
```
x.replace('< 1 year',"0", inplace=True)
```

```
x.emp_length=x.emp_length.str.split(expand=True)[0].astype('int64')
```

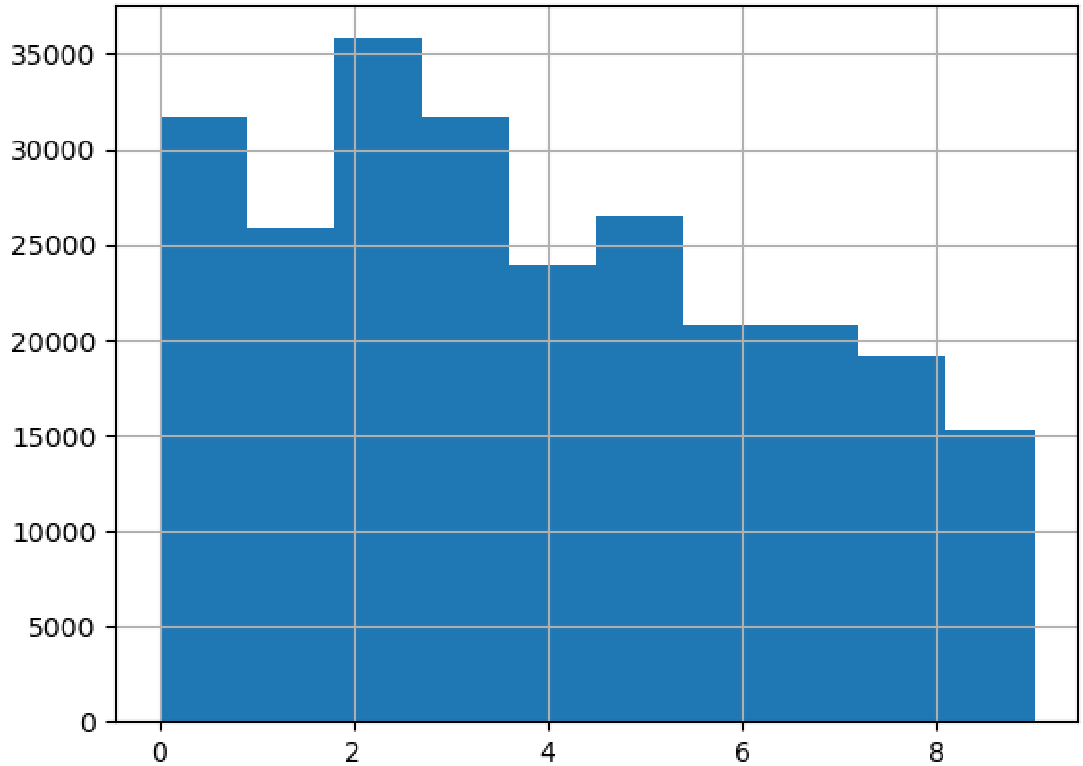
```
x.emp_length
```

```
1         4
2         0
3         6
4         9
6         2
         ..
396017    8
396022    1
396024    5
396025    2
```

```
396026    5
Name: emp_length, Length: 251688, dtype: int64

x.emp_length.hist()

<Axes: >
```



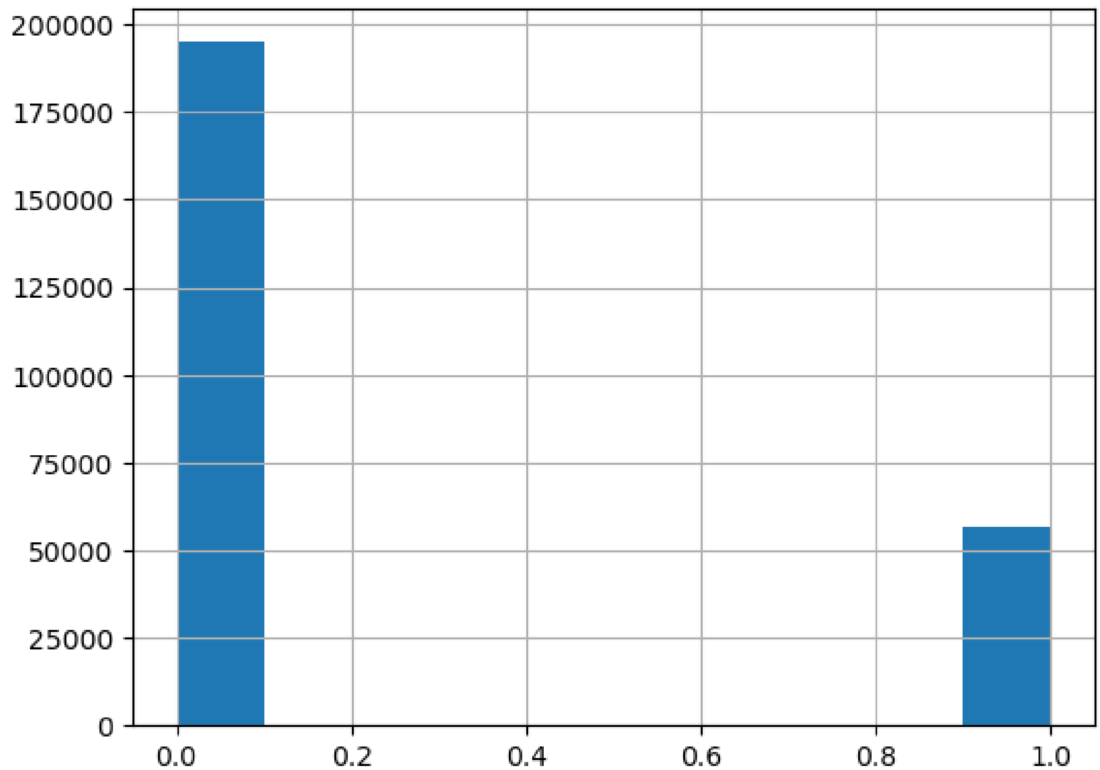

```
iqr_el=x.emp_length.quantile(.75)-x.emp_length.quantile(.25)

ul=x.emp_length.quantile(.75)+(1.5*iqr_el)

ll=x.emp_length.quantile(.25)-(1.5*iqr_el)

ul

12.0

ll

-4.0

x.emp_length=[ul if i>=ul else i for i in x.emp_length]

x.emp_length.hist() #This still does not look normal enough and I have
let go after trying multiple transforms

<Axes: >
```




```
import sklearn
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split



import statsmodels
from statsmodels.stats.outliers_influence import
variance_inflation_factor as vif

x.shape

(251688, 28)



x.term=x.term.str.split(expand=True)[0].replace(['36','60'],
[0,1]).astype('int64')

/var/folders/c5/jzdz65t53qj6w2j6h9rj5sr80000gn/T/
ipykernel_8360/2232850954.py:1: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To
retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior,
set `pd.set_option('future.no_silent_downcasting', True)`
```

```
x.term=x.term.str.split(expand=True)[0].replace(['36','60'],
[0,1]).astype('int64')

x.replace('other','OTHER',inplace=True)

x.term.hist()
```

```
<Axes: >
```




```
x.dtypes
```

```
loan_amnt              float64
term                     int64
int_rate               float64
sub_grade                int64
emp_length               int64
annual_inc             float64
verification_status      int64
dti                    float64
initial_list_status      int64
NONE                      bool
OTHER                     bool
OWN                       bool
RENT                      bool
credit_card               bool
```

```
debt_consolidation         bool
educational                bool
home_improvement           bool
house                      bool
major_purchase             bool
medical                    bool
moving                     bool
other                      bool
renewable_energy           bool
small_business             bool
vacation                   bool
wedding                    bool
INDIVIDUAL                 bool
JOINT                      bool
dtype: object
```

```
x.columns
```

```
Index(['loan_amnt', 'term', 'int_rate', 'sub_grade', 'emp_length',
       'annual_inc', 'verification_status', 'dti',
'initial_list_status',
       'NONE', 'OTHER', 'OWN', 'RENT', 'credit_card',
'debt_consolidation',
       'educational', 'home_improvement', 'house', 'major_purchase',
'medical',
       'moving', 'other', 'renewable_energy', 'small_business',
'vacation',
       'wedding', 'INDIVIDUAL', 'JOINT'],
      dtype='object')
```

```
x[['loan_amnt', 'term', 'int_rate', 'sub_grade', 'emp_length',
       'annual_inc', 'verification_status', 'dti',
'initial_list_status',
       'NONE', 'OTHER', 'OWN', 'RENT', 'credit_card',
'debt_consolidation',
       'educational', 'home_improvement', 'house', 'major_purchase',
'medical',
       'moving', 'other', 'renewable_energy', 'small_business',
'vacation',
       'wedding', 'INDIVIDUAL', 'JOINT']]=x[['loan_amnt', 'term',
'int_rate', 'sub_grade', 'emp_length',
       'annual_inc', 'verification_status', 'dti',
'initial_list_status',
       'NONE', 'OTHER', 'OWN', 'RENT', 'credit_card',
'debt_consolidation',
       'educational', 'home_improvement', 'house', 'major_purchase',
'medical',
       'moving', 'other', 'renewable_energy', 'small_business',
'vacation',
       'wedding', 'INDIVIDUAL', 'JOINT']].astype('int64')
```

```
x.dtypes
```

```
loan_amnt              int64
term                   int64
int_rate               int64
sub_grade              int64
emp_length             int64
annual_inc             int64
verification_status    int64
dti                    int64
initial_list_status    int64
NONE                   int64
OTHER                  int64
OWN                    int64
RENT                   int64
credit_card            int64
debt_consolidation     int64
educational            int64
home_improvement       int64
house                  int64
major_purchase         int64
medical                int64
moving                 int64
other                  int64
renewable_energy       int64
small_business         int64
vacation               int64
wedding                int64
INDIVIDUAL             int64
JOINT                  int64
dtype: object
```

```
#Checking for multicollinearity

vif_df=pd.DataFrame()

vif_df['Features']=x.columns

vif_df['vif']=[vif(x.values,i) for i in range(x.shape[1])]

vif_df.sort_values(by='vif',ascending=False)
```

	Features	vif
2	int_rate	159.118540
26	INDIVIDUAL	122.039449
3	sub_grade	66.343499
14	debt_consolidation	43.915574
5	annual_inc	27.842232
0	loan_amnt	20.483719
13	credit_card	16.526390
7	dti	5.957414

16	home_improvement	5.055984
21	other	4.895489
4	emp_length	3.097130
6	verification_status	2.869599
8	initial_list_status	2.749938
18	major_purchase	2.736055
12	RENT	2.347416
23	small_business	2.227005
1	term	1.964752
19	medical	1.757477
20	moving	1.636497
17	house	1.459496
25	wedding	1.437254
24	vacation	1.432592
11	OWN	1.222579
27	JOINT	1.108100
22	renewable_energy	1.064330
15	educational	1.063408
10	OTHER	1.001546
9	NONE	1.000305

```
x_new=x.drop('int_rate', axis=1)

vif_df2=pd.DataFrame()
vif_df2['features']=x_new.columns

vif_df2['vif']=[vif(x_new.values,i) for i in range(x_new.shape[1])]

vif_df2.sort_values(by='vif', ascending=False)
```

	features	vif
25	INDIVIDUAL	100.644484
13	debt_consolidation	43.814229
4	annual_inc	27.804375
0	loan_amnt	20.478800
12	credit_card	16.479834
6	dti	5.955760
2	sub_grade	5.552463
15	home_improvement	5.049818
20	other	4.890019
3	emp_length	3.089352
5	verification_status	2.862165
17	major_purchase	2.734204
7	initial_list_status	2.673364
11	RENT	2.347366
22	small_business	2.226757
1	term	1.948089
18	medical	1.756585
19	moving	1.635659
16	house	1.458538

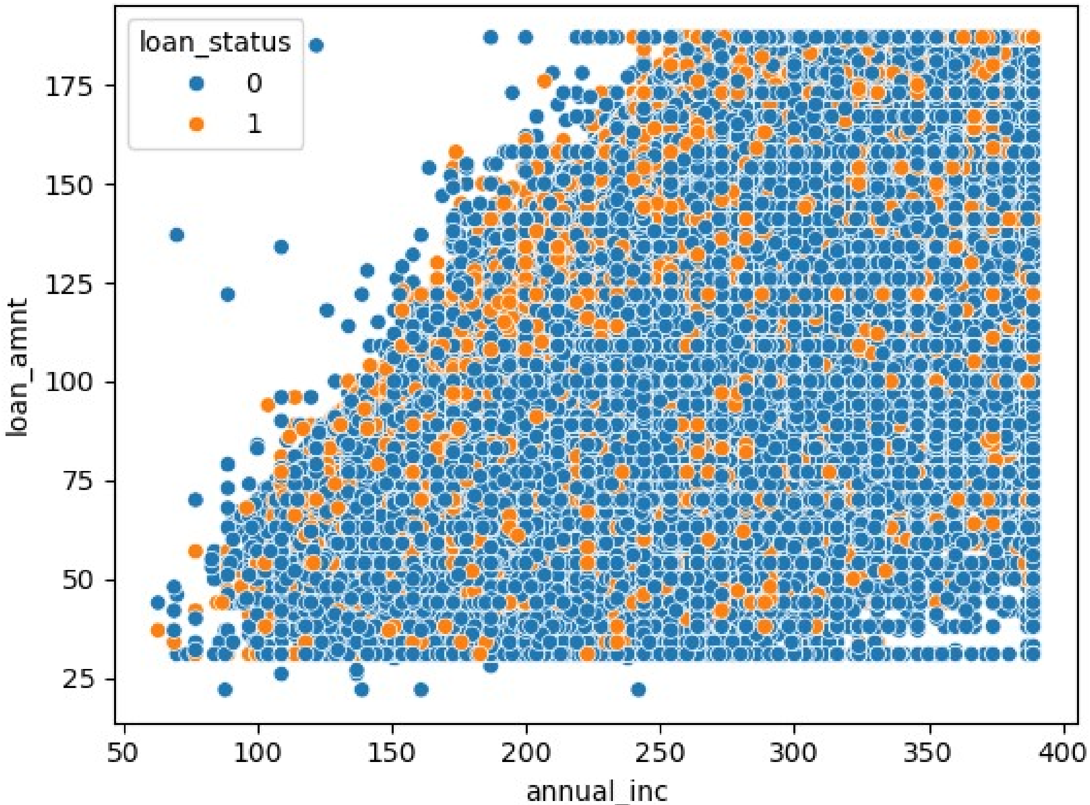
```
24                 wedding    1.436356
23                vacation    1.431907
10                     OWN    1.222479
26                   JOINT    1.090050
21        renewable_energy    1.064305
14             educational    1.063318
9                    OTHER    1.001534
8                     NONE    1.000261
```

```
vif_df3=pd.DataFrame()

x_new_2=x_new.drop('INDIVIDUAL', axis=1)

vif_df3['features']=x_new_2.columns

vif_df3['vif']=[vif(x_new_2.values, i) for i in
range(x_new_2.shape[1])]

vif_df3.sort_values(by='vif', ascending=False)
```

	features	vif
4	annual_inc	23.857701
0	loan_amnt	20.447269
13	debt_consolidation	17.852452
12	credit_card	6.946988
6	dti	5.693950
2	sub_grade	5.491294
3	emp_length	3.048729
5	verification_status	2.859252
7	initial_list_status	2.593314
15	home_improvement	2.520120
20	other	2.467204
11	RENT	2.234134
1	term	1.942312
17	major_purchase	1.620227
22	small_business	1.506574
18	medical	1.288669
19	moving	1.251106
10	OWN	1.204696
16	house	1.188007
24	wedding	1.168347
23	vacation	1.166097
21	renewable_energy	1.024432
14	educational	1.019822
25	JOINT	1.003133
9	OTHER	1.001444
8	NONE	1.000256

```
sns.scatterplot(x=x.annual_inc,y=x.loan_amnt,hue=y, data=data)
```

```
<Axes: xlabel='annual_inc', ylabel='loan_amnt'>
```




```
np.corrcoef(x.annual_inc, x.loan_amnt)

array([[1.        , 0.51401461],
       [0.51401461, 1.        ]])

x_new_3=x_new_2.drop('annual_inc', axis=1)

vif_df_4=pd.DataFrame(columns=['features','vif'])

vif_df_4['features']=x_new_3.columns

vif_df_4['vif']=[vif(x_new_3.values, i) for i in
range(x_new_3.shape[1])]

vif_df_4.sort_values(by='vif', ascending=False) #I will keep the loan
amount column as it is an important variable to determines who pays up
na who runs away

              features        vif
0            loan_amnt  14.192121
12  debt_consolidation  13.206193
5                  dti   5.518560
2            sub_grade   5.465802
11         credit_card   5.234918
3           emp_length   3.030336
4  verification_status   2.858417
```

```
6   initial_list_status   2.592482
10                 RENT   2.212688
14     home_improvement   1.968034
19                other   1.942928
1                  term   1.932355
16       major_purchase   1.389450
21       small_business   1.361115
9                   OWN   1.199809
17              medical   1.176443
18               moving   1.152466
15                house   1.134461
23              wedding   1.114384
22             vacation   1.095230
20     renewable_energy   1.015743
13          educational   1.013799
24                JOINT   1.002837
8                 OTHER   1.001436
7                  NONE   1.000245
```

```
y.value_counts() #This is a little imbalanced
```

```
loan_status
0    202268
1     49420
Name: count, dtype: int64
```

```
import statsmodels.api as sm

from sklearn.linear_model import LogisticRegression

xtrain, xtest, ytrain, ytest=train_test_split(x_new_3, y,
test_size=0.2, random_state=10)

sc=StandardScaler()

xtrain_sc=sc.fit_transform(xtrain)

xtest_sc=sc.transform(xtest)

xtrain_sc=pd.DataFrame(xtrain_sc, columns=xtrain.columns)

xtrain_sc
```

```
        loan_amnt      term  sub_grade  emp_length
verification_status  \
0       -1.382222 -0.539094  -0.920924    1.128624             -
1.200334
1       -0.064616 -0.539094  -0.768967   -0.692137
1.264304
2       -1.181716 -0.539094  -0.313096   -0.327985             -
```

```
1.200334
3       -0.064616 -0.539094  -0.009182    0.036167
0.031985
4       -0.780706 -0.539094  -1.376794    1.492777             -
1.200334
...           ...       ...        ...         ...                   ..
.
201345  -0.809350 -0.539094   0.750602    1.492777
0.031985
201346  -1.181716 -0.539094  -0.313096    1.856929
0.031985
201347  -0.064616 -0.539094  -1.376794   -0.327985
1.264304
201348  -0.465626 -0.539094   0.294731   -0.692137             -
1.200334
201349   2.169585 -0.539094  -0.768967    0.400320
1.264304

             dti  initial_list_status      NONE     OTHER
OWN  ...  \
0      -1.313105            -1.264769 -0.008631 -0.019174
3.244807  ...
1      -0.073361             0.790658 -0.008631 -0.019174 -
0.308185  ...
2      -1.189130            -1.264769 -0.008631 -0.019174 -
0.308185  ...
3      -0.445284             0.790658 -0.008631 -0.019174 -
0.308185  ...
4       0.546511             0.790658 -0.008631 -0.019174 -
0.308185  ...
...          ...                  ...       ...       ...       ...  .
..
201345  0.794460             0.790658 -0.008631 -0.019174 -
0.308185  ...
201346  0.546511             0.790658 -0.008631 -0.019174 -
0.308185  ...
201347 -1.313105             0.790658 -0.008631 -0.019174 -
0.308185  ...
201348  1.290358             0.790658 -0.008631 -0.019174 -
0.308185  ...
201349 -0.197335            -1.264769 -0.008631 -0.019174 -
0.308185  ...

           house  major_purchase   medical    moving     other  \
0      -0.078686       -0.157221 -0.102539 -0.092904 -0.237850
1      -0.078686       -0.157221 -0.102539 -0.092904 -0.237850
2      -0.078686       -0.157221 -0.102539 -0.092904 -0.237850
3      -0.078686       -0.157221 -0.102539 -0.092904 -0.237850
4      -0.078686       -0.157221 -0.102539 -0.092904  4.204328
```

```
...         ...            ...       ...       ...       ...
201345 -0.078686       -0.157221 -0.102539 -0.092904 -0.237850
201346 -0.078686       -0.157221 -0.102539 -0.092904 -0.237850
201347 -0.078686       -0.157221 -0.102539 -0.092904 -0.237850
201348 -0.078686       -0.157221 -0.102539 -0.092904 -0.237850
201349 -0.078686       -0.157221 -0.102539 -0.092904 -0.237850

        renewable_energy  small_business  vacation    wedding
JOINT
0              -0.029578       -0.130842 -0.077008  -0.077722 -
0.029662
1              -0.029578       -0.130842 -0.077008  -0.077722 -
0.029662
2              -0.029578       -0.130842 -0.077008  -0.077722 -
0.029662
3              -0.029578       -0.130842 -0.077008  -0.077722 -
0.029662
4              -0.029578       -0.130842 -0.077008  -0.077722 -
0.029662
...                  ...             ...       ...        ...        ..
.
201345         -0.029578       -0.130842 -0.077008  -0.077722 -
0.029662
201346         -0.029578       -0.130842 -0.077008  12.866336 -
0.029662
201347         -0.029578       -0.130842 -0.077008  -0.077722 -
0.029662
201348         -0.029578       -0.130842 -0.077008  -0.077722 -
0.029662
201349         -0.029578       -0.130842 -0.077008  -0.077722 -
0.029662

[201350 rows x 25 columns]
```

```
xtest_sc=pd.DataFrame(xtest_sc, columns=xtest.columns)

xtest_sc
```

```
       loan_amnt      term  sub_grade  emp_length  verification_status
\
0       2.169585 -0.539094   1.054516   -0.692137             1.264304

1       0.851979  1.854964  -0.465053    1.128624             1.264304

2       0.307751 -0.539094  -0.161139    1.856929             1.264304

3      -0.809350 -0.539094   2.726042    0.400320             1.264304

4      -1.181716 -0.539094  -0.617010    1.856929             0.031985

...          ...       ...        ...         ...                  ...
```

```
50333  -0.866637 -0.539094   0.142775    0.764472            -1.200334

50334  -0.064616  1.854964   0.446688    0.036167             1.264304

50335  -0.809350 -0.539094   0.598645   -1.420442            -1.200334

50336  -0.236478  1.854964   0.142775    0.764472             1.264304

50337  -0.236478 -0.539094  -1.224837   -0.692137             1.264304


            dti  initial_list_status      NONE     OTHER
OWN  ...  \
0     -0.197335             0.790658 -0.008631 -0.019174 -
0.308185  ...
1      0.670486             0.790658 -0.008631 -0.019174 -
0.308185  ...
2      1.786256            -1.264769 -0.008631 -0.019174 -
0.308185  ...
3      0.794460            -1.264769 -0.008631 -0.019174 -
0.308185  ...
4      1.786256            -1.264769 -0.008631 -0.019174 -
0.308185  ...
...         ...                  ...       ...       ...       ...  ..
.
50333  0.174588             0.790658 -0.008631 -0.019174 -
0.308185  ...
50334 -0.197335            -1.264769 -0.008631 -0.019174 -
0.308185  ...
50335 -1.685028             0.790658 -0.008631 -0.019174 -
0.308185  ...
50336 -0.941182            -1.264769 -0.008631 -0.019174
3.244807  ...
50337 -0.197335             0.790658 -0.008631 -0.019174 -
0.308185  ...

           house  major_purchase   medical    moving    other  \
0      -0.078686       -0.157221 -0.102539 -0.092904 -0.23785
1      12.708662       -0.157221 -0.102539 -0.092904 -0.23785
2      -0.078686       -0.157221 -0.102539 -0.092904 -0.23785
3      -0.078686       -0.157221 -0.102539 -0.092904 -0.23785
4      -0.078686       -0.157221 -0.102539 -0.092904 -0.23785
...          ...             ...       ...       ...       ...
50333  -0.078686       -0.157221 -0.102539 -0.092904 -0.23785
50334  -0.078686       -0.157221 -0.102539 -0.092904 -0.23785
50335  -0.078686       -0.157221 -0.102539 -0.092904 -0.23785
50336  -0.078686       -0.157221 -0.102539 -0.092904 -0.23785
50337  -0.078686       -0.157221 -0.102539 -0.092904 -0.23785
```

```
       renewable_energy  small_business  vacation   wedding     JOINT
0             -0.029578       -0.130842 -0.077008 -0.077722 -0.029662
1             -0.029578       -0.130842 -0.077008 -0.077722 -0.029662
2             -0.029578       -0.130842 -0.077008 -0.077722 -0.029662
3             -0.029578       -0.130842 -0.077008 -0.077722 -0.029662
4             -0.029578       -0.130842 -0.077008 -0.077722 -0.029662
...                 ...             ...       ...       ...       ...
50333         -0.029578       -0.130842 -0.077008 -0.077722 -0.029662
50334         -0.029578       -0.130842 -0.077008 -0.077722 -0.029662
50335         -0.029578       -0.130842 -0.077008 -0.077722 -0.029662
50336         -0.029578       -0.130842 -0.077008 -0.077722 -0.029662
50337         -0.029578       -0.130842 -0.077008 -0.077722 -0.029662

[50338 rows x 25 columns]
```

```
xtrain_sc.shape
```

```
(201350, 25)
```

```
ytrain.shape
```

```
(201350,)
```

```
xtrain_sc=sm.add_constant(xtrain_sc)
```

```
model_logit=sm.Logit(list(ytrain), xtrain_sc).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.453414
         Iterations 6
```

```
model_logit.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
======================================================================
========
```

```
Dep. Variable:                      y   No. Observations:
201350
Model:                          Logit   Df Residuals:
201324
Method:                           MLE   Df Model:
25
Date:                Fri, 18 Apr 2025   Pseudo R-squ.:
0.08555
Time:                        19:10:22   Log-Likelihood:
-91295.
converged:                       True   LL-Null:
-99836.
Covariance Type:            nonrobust   LLR p-value:
0.000
======================================================================
=================
                          coef    std err          z      P>|z|
[0.025      0.975]
----------------------------------------------------------------------
-----------------
const                  -1.5605      0.006   -247.090      0.000       -
1.573      -1.548
loan_amnt               0.0279      0.007      4.031      0.000
0.014       0.042
term                    0.1544      0.007     23.375      0.000
0.141       0.167
sub_grade               0.5340      0.007     77.854      0.000
0.521       0.547
emp_length             -0.0076      0.006     -1.272      0.203       -
0.019       0.004
verification_status    -0.0087      0.006     -1.351      0.177       -
0.021       0.004
dti                     0.2370      0.006     39.622      0.000
0.225       0.249
initial_list_status    -0.0161      0.006     -2.700      0.007       -
0.028      -0.004
NONE                    0.0057      0.005      1.078      0.281       -
0.005       0.016
OTHER                  -0.0064      0.007     -0.871      0.384       -
0.021       0.008
OWN                     0.0577      0.006      9.422      0.000
0.046       0.070
RENT                    0.1622      0.007     24.797      0.000
0.149       0.175
credit_card             0.0420      0.025      1.713      0.087       -
0.006       0.090
debt_consolidation      0.0906      0.029      3.117      0.002
0.034       0.148
educational             0.0101      0.006      1.582      0.114       -
```

```
0.002       0.023
home_improvement        0.0424      0.015      2.888      0.004
0.014       0.071
house                  -0.0087      0.007     -1.154      0.248      -
0.023       0.006
major_purchase          0.0165      0.011      1.512      0.131      -
0.005       0.038
medical                 0.0184      0.008      2.254      0.024
0.002       0.034
moving                  0.0176      0.008      2.284      0.022
0.002       0.033
other                   0.0248      0.014      1.740      0.082      -
0.003       0.053
renewable_energy        0.0136      0.006      2.430      0.015
0.003       0.025
small_business          0.0688      0.009      7.517      0.000
0.051       0.087
vacation                0.0071      0.007      0.960      0.337      -
0.007       0.021
wedding                -0.0391      0.009     -4.555      0.000      -
0.056      -0.022
JOINT                  -0.0335      0.007     -4.780      0.000      -
0.047      -0.020
======================================================================
=================
"""

logit_summary=pd.DataFrame(model_logit.summary().tables[1]) #the
tables[0] and tables[1] give us the summary as a df



logit_summary.columns=logit_summary.iloc[0]

logit_summary

0                                coef    std err          z   P>|z|
[0.025  \
0                                coef    std err          z   P>|z|
[0.025
1                   const     -1.5605      0.006   -247.090   0.000
-1.573
2              loan_amnt      0.0279      0.007      4.031   0.000
0.014
3                   term      0.1544      0.007     23.375   0.000
0.141
4              sub_grade      0.5340      0.007     77.854   0.000
0.521
5             emp_length     -0.0076      0.006     -1.272   0.203
-0.019
```

```
6   verification_status     -0.0087       0.006      -1.351    0.177
-0.021
7                   dti      0.2370       0.006      39.622    0.000
0.225
8   initial_list_status     -0.0161       0.006      -2.700    0.007
-0.028
9                  NONE      0.0057       0.005       1.078    0.281
-0.005
10                OTHER     -0.0064       0.007      -0.871    0.384
-0.021
11                  OWN      0.0577       0.006       9.422    0.000
0.046
12                 RENT      0.1622       0.007      24.797    0.000
0.149
13          credit_card      0.0420       0.025       1.713    0.087
-0.006
14   debt_consolidation      0.0906       0.029       3.117    0.002
0.034
15          educational      0.0101       0.006       1.582    0.114
-0.002
16     home_improvement      0.0424       0.015       2.888    0.004
0.014
17                house     -0.0087       0.007      -1.154    0.248
-0.023
18       major_purchase      0.0165       0.011       1.512    0.131
-0.005
19              medical      0.0184       0.008       2.254    0.024
0.002
20               moving      0.0176       0.008       2.284    0.022
0.002
21                other      0.0248       0.014       1.740    0.082
-0.003
22     renewable_energy      0.0136       0.006       2.430    0.015
0.003
23       small_business      0.0688       0.009       7.517    0.000
0.051
24             vacation      0.0071       0.007       0.960    0.337
-0.007
25              wedding     -0.0391       0.009      -4.555    0.000
-0.056
26                JOINT     -0.0335       0.007      -4.780    0.000
-0.047

0      0.975]
0      0.975]
1      -1.548
2       0.042
3       0.167
4       0.547
```

```
5        0.004
6        0.004
7        0.249
8       -0.004
9        0.016
10       0.008
11       0.070
12       0.175
13       0.090
14       0.148
15       0.023
16       0.071
17       0.006
18       0.038
19       0.034
20       0.033
21       0.053
22       0.025
23       0.087
24       0.021
25      -0.022
26      -0.020
```

```
logit_summary=pd.DataFrame(model_logit.summary().tables[1]) #the
values in this are not of float data type, they are of "cell" type

logit_summary.columns=logit_summary.iloc[0]
logit_summary.drop(0, inplace=True)

logit_summary.to_csv('logit_summary.csv')
#this is due to some different data type "cell" of the statsmodels
table, was unable to convert to float directly
#so I am saving it as csv and reading it again

logit_summary=pd.read_csv('logit_summary.csv', index_col=0)

logit_summary.columns=['feature','coef','std err','z','pval',
'lower_crit_val', 'upper_crit_val']

logit_summary[logit_summary.pval>0.05] #getting those features where
we are not confident because the p value shows >0.05
```

```
                feature    coef  std err      z   pval  lower_crit_val
\
5            emp_length -0.0076    0.006 -1.272  0.203          -0.019

6   verification_status -0.0087    0.006 -1.351  0.177          -0.021

9                  NONE  0.0057    0.005  1.078  0.281          -0.005

10                OTHER -0.0064    0.007 -0.871  0.384          -0.021
```

13	credit_card	0.0420	0.025	1.713	0.087	-0.006
15	educational	0.0101	0.006	1.582	0.114	-0.002
17	house	-0.0087	0.007	-1.154	0.248	-0.023
18	major_purchase	0.0165	0.011	1.512	0.131	-0.005
21	other	0.0248	0.014	1.740	0.082	-0.003
24	vacation	0.0071	0.007	0.960	0.337	-0.007

	upper_crit_val
5	0.004
6	0.004
9	0.016
10	0.008
13	0.090
15	0.023
17	0.006
18	0.038
21	0.053
24	0.021

```
#Here it shows that the OTHER feature has a p value that is very high
so I will remove this and repeat

x_new_4=x_new_3.drop('OTHER', axis=1)

vif_df_5=pd.DataFrame(columns=['features','vif'])

vif_df_5.features=x_new_4.columns

vif_df_5.vif=[vif(x_new_4.values, i) for i in range(x_new_4.shape[1])]

vif_df_5.sort_values(by='vif', ascending=False)
```

	features	vif
0	loan_amnt	14.192116
11	debt_consolidation	13.205007
5	dti	5.518438
2	sub_grade	5.465608
10	credit_card	5.234321
3	emp_length	3.030180
4	verification_status	2.858404
6	initial_list_status	2.592180
9	RENT	2.211596
13	home_improvement	1.967961
18	other	1.942700
1	term	1.932223

```
15       major_purchase   1.389370
20       small_business   1.360946
8                   OWN   1.199647
16              medical   1.176360
17               moving   1.152456
14                house   1.134435
22              wedding   1.114360
21             vacation   1.095225
19     renewable_energy   1.015680
12          educational   1.013569
23                JOINT   1.002837
7                  NONE   1.000245
```

```
xtrain_new, xtest_new, ytrain, ytest=train_test_split(x_new_4, y,
test_size=0.2, random_state=10)

xtrain_sc_new=sc.fit_transform(xtrain_new)
xtest_sc_new=sc.transform(xtest_new)

xtrain_sc_new=pd.DataFrame(xtrain_sc_new, columns=xtrain_new.columns)

xtest_sc_new=pd.DataFrame(xtest_sc_new, columns=xtest_new.columns)

xtrain_sc_new=sm.add_constant(xtrain_sc_new)



model_logit_2=sm.Logit(list(ytrain), xtrain_sc_new).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.453416
         Iterations 6
```

```
model_logit_2.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results

======================================================================
========
Dep. Variable:                      y   No. Observations:
201350
Model:                          Logit   Df Residuals:
201325
Method:                           MLE   Df Model:
24
Date:                Fri, 18 Apr 2025   Pseudo R-squ.:
0.08554
Time:                        19:10:47   Log-Likelihood:
-91295.
converged:                       True   LL-Null:
```

```
-99836.
Covariance Type:            nonrobust   LLR p-value:
0.000
======================================================================
=================
                          coef    std err          z      P>|z|
[0.025      0.975]
----------------------------------------------------------------------
-----------------
const                  -1.5604      0.006   -247.092      0.000      -
1.573      -1.548
loan_amnt               0.0279      0.007      4.031      0.000
0.014       0.042
term                    0.1545      0.007     23.385      0.000
0.142       0.167
sub_grade               0.5339      0.007     77.851      0.000
0.520       0.547
emp_length             -0.0075      0.006     -1.265      0.206      -
0.019       0.004
verification_status    -0.0087      0.006     -1.350      0.177      -
0.021       0.004
dti                     0.2370      0.006     39.629      0.000
0.225       0.249
initial_list_status    -0.0161      0.006     -2.707      0.007      -
0.028      -0.004
NONE                    0.0057      0.005      1.079      0.281      -
0.005       0.016
OWN                     0.0578      0.006      9.434      0.000
0.046       0.070
RENT                    0.1624      0.007     24.823      0.000
0.150       0.175
credit_card             0.0420      0.025      1.713      0.087      -
0.006       0.090
debt_consolidation      0.0906      0.029      3.118      0.002
0.034       0.148
educational             0.0101      0.006      1.575      0.115      -
0.002       0.023
home_improvement        0.0425      0.015      2.890      0.004
0.014       0.071
house                  -0.0086      0.007     -1.154      0.249      -
0.023       0.006
major_purchase          0.0165      0.011      1.513      0.130      -
0.005       0.038
medical                 0.0184      0.008      2.249      0.024
0.002       0.034
moving                  0.0176      0.008      2.285      0.022
0.003       0.033
other                   0.0248      0.014      1.740      0.082      -
0.003       0.053
```

```
renewable_energy          0.0136      0.006      2.424      0.015
0.003       0.025
small_business            0.0688      0.009      7.514      0.000
0.051       0.087
vacation                  0.0071      0.007      0.962      0.336      -
0.007       0.022
wedding                  -0.0391      0.009     -4.554      0.000      -
0.056      -0.022
JOINT                    -0.0335      0.007     -4.780      0.000      -
0.047      -0.020
======================================================================
=================
"""
```

```
model_logit_2.summary().tables[1]
```

```
<class 'statsmodels.iolib.table.SimpleTable'>
```

```
logit_summary_2=pd.DataFrame(model_logit_2.summary().tables[1])
```

```
logit_summary_2.columns=logit_summary.columns
logit_summary_2.drop(0, inplace=True)
```

```
logit_summary_2
```

	feature	coef	std err	z	pval \
1	const	-1.5604	0.006	-247.092	0.000
2	loan_amnt	0.0279	0.007	4.031	0.000
3	term	0.1545	0.007	23.385	0.000
4	sub_grade	0.5339	0.007	77.851	0.000
5	emp_length	-0.0075	0.006	-1.265	0.206
6	verification_status	-0.0087	0.006	-1.350	0.177
7	dti	0.2370	0.006	39.629	0.000
8	initial_list_status	-0.0161	0.006	-2.707	0.007
9	NONE	0.0057	0.005	1.079	0.281
10	OWN	0.0578	0.006	9.434	0.000
11	RENT	0.1624	0.007	24.823	0.000
12	credit_card	0.0420	0.025	1.713	0.087
13	debt_consolidation	0.0906	0.029	3.118	0.002
14	educational	0.0101	0.006	1.575	0.115
15	home_improvement	0.0425	0.015	2.890	0.004
16	house	-0.0086	0.007	-1.154	0.249
17	major_purchase	0.0165	0.011	1.513	0.130
18	medical	0.0184	0.008	2.249	0.024
19	moving	0.0176	0.008	2.285	0.022
20	other	0.0248	0.014	1.740	0.082
21	renewable_energy	0.0136	0.006	2.424	0.015
22	small_business	0.0688	0.009	7.514	0.000
23	vacation	0.0071	0.007	0.962	0.336
24	wedding	-0.0391	0.009	-4.554	0.000
25	JOINT	-0.0335	0.007	-4.780	0.000

```
   lower_crit_val upper_crit_val
1          -1.573         -1.548
2           0.014          0.042
3           0.142          0.167
4           0.520          0.547
5          -0.019          0.004
6          -0.021          0.004
7           0.225          0.249
8          -0.028         -0.004
9          -0.005          0.016
10          0.046          0.070
11          0.150          0.175
12         -0.006          0.090
13          0.034          0.148
14         -0.002          0.023
15          0.014          0.071
16         -0.023          0.006
17         -0.005          0.038
18          0.002          0.034
19          0.003          0.033
20         -0.003          0.053
21          0.003          0.025
22          0.051          0.087
23         -0.007          0.022
24         -0.056         -0.022
25         -0.047         -0.020
```

```
logit_summary_2.to_csv('logit_summary_2.csv')
```

```
logit_summary_2=pd.read_csv('logit_summary_2.csv', index_col=0)
```

```
logit_summary_2
```

```
                feature    coef  std err        z   pval
lower_crit_val  \
1                 const -1.5604    0.006 -247.092  0.000           -
1.573
2             loan_amnt  0.0279    0.007    4.031  0.000
0.014
3                  term  0.1545    0.007   23.385  0.000
0.142
4             sub_grade  0.5339    0.007   77.851  0.000
0.520
5            emp_length -0.0075    0.006   -1.265  0.206           -
0.019
6   verification_status -0.0087    0.006   -1.350  0.177           -
0.021
7                   dti  0.2370    0.006   39.629  0.000
0.225
```

```
8   initial_list_status -0.0161    0.006   -2.707  0.007           -
0.028
9                  NONE  0.0057    0.005    1.079  0.281           -
0.005
10                  OWN  0.0578    0.006    9.434  0.000            
0.046
11                 RENT  0.1624    0.007   24.823  0.000            
0.150
12          credit_card  0.0420    0.025    1.713  0.087           -
0.006
13   debt_consolidation  0.0906    0.029    3.118  0.002            
0.034
14          educational  0.0101    0.006    1.575  0.115           -
0.002
15     home_improvement  0.0425    0.015    2.890  0.004            
0.014
16                house -0.0086    0.007   -1.154  0.249           -
0.023
17       major_purchase  0.0165    0.011    1.513  0.130           -
0.005
18              medical  0.0184    0.008    2.249  0.024            
0.002
19               moving  0.0176    0.008    2.285  0.022            
0.003
20                other  0.0248    0.014    1.740  0.082           -
0.003
21     renewable_energy  0.0136    0.006    2.424  0.015            
0.003
22       small_business  0.0688    0.009    7.514  0.000            
0.051
23             vacation  0.0071    0.007    0.962  0.336           -
0.007
24              wedding -0.0391    0.009   -4.554  0.000           -
0.056
25                JOINT -0.0335    0.007   -4.780  0.000           -
0.047

    upper_crit_val
1           -1.548
2            0.042
3            0.167
4            0.547
5            0.004
6            0.004
7            0.249
8           -0.004
9            0.016
10           0.070
11           0.175
```

```
12            0.090
13            0.148
14            0.023
15            0.071
16            0.006
17            0.038
18            0.034
19            0.033
20            0.053
21            0.025
22            0.087
23            0.022
24           -0.022
25           -0.020
```

```
logit_summary_2.columns=logit_summary.columns
```

```
logit_summary_2
```

```
                feature    coef  std err        z   pval
lower_crit_val  \
1                 const -1.5604    0.006 -247.092  0.000           -
1.573
2             loan_amnt  0.0279    0.007    4.031  0.000
0.014
3                  term  0.1545    0.007   23.385  0.000
0.142
4             sub_grade  0.5339    0.007   77.851  0.000
0.520
5            emp_length -0.0075    0.006   -1.265  0.206           -
0.019
6   verification_status -0.0087    0.006   -1.350  0.177           -
0.021
7                   dti  0.2370    0.006   39.629  0.000
0.225
8   initial_list_status -0.0161    0.006   -2.707  0.007           -
0.028
9                  NONE  0.0057    0.005    1.079  0.281           -
0.005
10                  OWN  0.0578    0.006    9.434  0.000
0.046
11                 RENT  0.1624    0.007   24.823  0.000
0.150
12          credit_card  0.0420    0.025    1.713  0.087           -
0.006
13   debt_consolidation  0.0906    0.029    3.118  0.002
0.034
14          educational  0.0101    0.006    1.575  0.115           -
0.002
15     home_improvement  0.0425    0.015    2.890  0.004
```

```
0.014
16                house -0.0086    0.007   -1.154  0.249            -
0.023
17       major_purchase  0.0165    0.011    1.513  0.130            -
0.005
18              medical  0.0184    0.008    2.249  0.024
0.002
19               moving  0.0176    0.008    2.285  0.022
0.003
20                other  0.0248    0.014    1.740  0.082            -
0.003
21     renewable_energy  0.0136    0.006    2.424  0.015
0.003
22       small_business  0.0688    0.009    7.514  0.000
0.051
23             vacation  0.0071    0.007    0.962  0.336            -
0.007
24              wedding -0.0391    0.009   -4.554  0.000            -
0.056
25                JOINT -0.0335    0.007   -4.780  0.000            -
0.047

    upper_crit_val
1           -1.548
2            0.042
3            0.167
4            0.547
5            0.004
6            0.004
7            0.249
8           -0.004
9            0.016
10           0.070
11           0.175
12           0.090
13           0.148
14           0.023
15           0.071
16           0.006
17           0.038
18           0.034
19           0.033
20           0.053
21           0.025
22           0.087
23           0.022
24          -0.022
25          -0.020

logit_summary.columns
```

```
Index(['feature', 'coef', 'std err', 'z', 'pval', 'lower_crit_val',
       'upper_crit_val'],
      dtype='object')

logit_summary_2.columns=['feature', 'coef', 'std err', 'z', 'pval',
'lower_crit_val',
       'upper_crit_val']

logit_summary_2[logit_summary_2.pval>0.05]

                feature    coef  std err      z   pval  lower_crit_val  \
5            emp_length -0.0075    0.006 -1.265  0.206          -0.019

6   verification_status -0.0087    0.006 -1.350  0.177          -0.021

9                  NONE  0.0057    0.005  1.079  0.281          -0.005

12          credit_card  0.0420    0.025  1.713  0.087          -0.006

14          educational  0.0101    0.006  1.575  0.115          -0.002

16                house -0.0086    0.007 -1.154  0.249          -0.023

17       major_purchase  0.0165    0.011  1.513  0.130          -0.005

20                other  0.0248    0.014  1.740  0.082          -0.003

23             vacation  0.0071    0.007  0.962  0.336          -0.007


    upper_crit_val
5            0.004
6            0.004
9            0.016
12           0.090
14           0.023
16           0.006
17           0.038
20           0.053
23           0.022

x_new_5=x_new_4.drop('vacation', axis=1)

xtrain_new_2, xtest_new_2, ytrain, ytest=train_test_split(x_new_5, y,
test_size=0.2, random_state=10)

xtrain_sc_new_2=sc.fit_transform(xtrain_new_2)
xtest_sc_new_2=sc.transform(xtest_new_2)

xtrain_sc_new_2=pd.DataFrame(xtrain_sc_new_2,
columns=xtrain_new_2.columns)
```

```
xtest_sc_new_2=pd.DataFrame(xtest_sc_new_2,
columns=xtest_new_2.columns)

xtrain_sc_new_2=sm.add_constant(xtrain_sc_new_2)

model_logit_3=sm.Logit(list(ytrain), xtrain_sc_new_2).fit()

Optimization terminated successfully.
         Current function value: 0.453418
         Iterations 6

model_logit_3.summary()

<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results

======================================================================
========
Dep. Variable:                      y   No. Observations:
201350
Model:                          Logit   Df Residuals:
201326
Method:                           MLE   Df Model:
23
Date:                Fri, 18 Apr 2025   Pseudo R-squ.:
0.08554
Time:                        19:10:50   Log-Likelihood:
-91296.
converged:                       True   LL-Null:
-99836.
Covariance Type:            nonrobust   LLR p-value:
0.000
======================================================================
=================
                          coef    std err          z      P>|z|
[0.025      0.975]
----------------------------------------------------------------------
-----------------
const                  -1.5604      0.006   -247.100      0.000      -
1.573      -1.548
loan_amnt               0.0278      0.007      4.011      0.000
0.014       0.041
term                    0.1543      0.007     23.367      0.000
0.141       0.167
sub_grade               0.5342      0.007     77.968      0.000
0.521       0.548
emp_length             -0.0074      0.006     -1.248      0.212      -
0.019       0.004
verification_status    -0.0086      0.006     -1.339      0.180      -
```

```
0.021       0.004
dti                        0.2371      0.006     39.645      0.000
0.225       0.249
initial_list_status       -0.0162      0.006     -2.718      0.007      -
0.028      -0.005
NONE                       0.0057      0.005      1.079      0.281      -
0.005       0.016
OWN                        0.0578      0.006      9.429      0.000
0.046       0.070
RENT                       0.1624      0.007     24.822      0.000
0.150       0.175
credit_card                0.0285      0.020      1.432      0.152      -
0.011       0.067
debt_consolidation         0.0742      0.023      3.187      0.001
0.029       0.120
educational                0.0091      0.006      1.443      0.149      -
0.003       0.022
home_improvement           0.0349      0.012      2.837      0.005
0.011       0.059
house                     -0.0113      0.007     -1.617      0.106      -
0.025       0.002
major_purchase             0.0114      0.009      1.203      0.229      -
0.007       0.030
medical                    0.0150      0.007      2.042      0.041
0.001       0.029
moving                     0.0145      0.007      2.081      0.037
0.001       0.028
other                      0.0173      0.012      1.462      0.144      -
0.006       0.040
renewable_energy           0.0126      0.006      2.287      0.022
0.002       0.023
small_business             0.0645      0.008      8.121      0.000
0.049       0.080
wedding                   -0.0417      0.008     -5.121      0.000      -
0.058      -0.026
JOINT                     -0.0335      0.007     -4.780      0.000      -
0.047      -0.020
======================================================================
=================
"""

x_new_6=x_new_5.drop(['NONE','emp_length','major_purchase','verificati
on_status', 'educational',
'credit_card','other','medical','moving','renewable_energy'], axis=1)



xtrain_new_3, xtest_new_3, ytrain, ytest=train_test_split(x_new_6, y,
test_size=0.2, random_state=10)
```

```
xtrain_sc_3=sc.fit_transform(xtrain_new_3)
xtest_sc_3=sc.transform(xtest_new_3)

xtrain_sc_3=pd.DataFrame(xtrain_sc_3, columns=xtrain_new_3.columns)
xtest_sc_3=pd.DataFrame(xtest_sc_3, columns=xtest_new_3.columns)

xtrain_sc_3=sm.add_constant(xtrain_sc_3)

model_logit_3=sm.Logit(list(ytrain), xtrain_sc_3).fit()

Optimization terminated successfully.
         Current function value: 0.453457
         Iterations 6

model_logit_3.summary()

<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results

======================================================================
========
Dep. Variable:                      y   No. Observations:
201350
Model:                          Logit   Df Residuals:
201336
Method:                           MLE   Df Model:
13
Date:                Fri, 18 Apr 2025   Pseudo R-squ.:
0.08546
Time:                        19:10:51   Log-Likelihood:
-91304.
converged:                       True   LL-Null:
-99836.
Covariance Type:            nonrobust   LLR p-value:
0.000
======================================================================
=================
                          coef    std err          z      P>|z|
[0.025      0.975]
----------------------------------------------------------------------
-----------------
const                  -1.5602      0.006   -247.113      0.000      -
1.573      -1.548
loan_amnt               0.0246      0.007      3.751      0.000
0.012       0.037
term                    0.1533      0.007     23.265      0.000
0.140       0.166
sub_grade               0.5345      0.007     80.568      0.000
0.522       0.548
dti                     0.2359      0.006     39.856      0.000
```

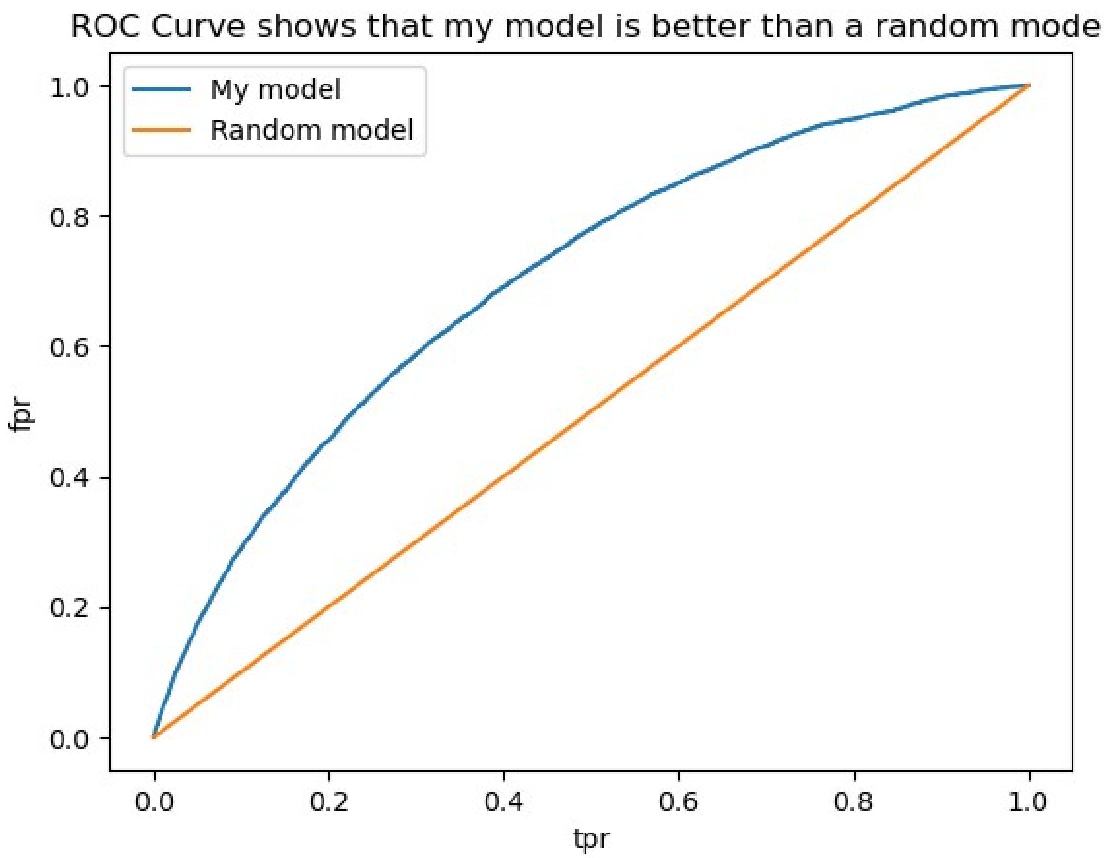
```
0.224       0.248
initial_list_status    -0.0166      0.006     -2.787      0.005      -
0.028      -0.005
OWN                     0.0579      0.006      9.466      0.000
0.046       0.070
RENT                    0.1634      0.006     25.176      0.000
0.151       0.176
debt_consolidation      0.0372      0.007      5.715      0.000
0.024       0.050
home_improvement        0.0176      0.007      2.654      0.008
0.005       0.031
house                  -0.0173      0.006     -2.875      0.004      -
0.029      -0.005
small_business          0.0544      0.005     10.076      0.000
0.044       0.065
wedding                -0.0475      0.007     -6.453      0.000      -
0.062      -0.033
JOINT                  -0.0334      0.007     -4.775      0.000      -
0.047      -0.020
======================================================================
=================
"""

#Now that it is safe to use this, I will test

xtest_sc_3=sm.add_constant(xtest_sc_3)

ypred=model_logit_3.predict(xtest_sc_3)

from sklearn.metrics import classification_report, confusion_matrix,
roc_curve, roc_auc_score

ypred

0        0.288408
1        0.141869
2        0.193979
3        0.546137
4        0.201468
           ...
50333    0.153809
50334    0.229770
50335    0.130264
50336    0.194442
50337    0.101491
Length: 50338, dtype: float64

ypred_bin=[1 if i>=0.5 else 0 for i in ypred] #Putting a threshold of
0.5

print(confusion_matrix(ytest, ypred_bin))
```

```
[[39729   800]
 [ 9046   763]]

print(classification_report(ytest, ypred_bin)) #we see that it is increasing the recall of my 0 class

              precision    recall  f1-score   support

           0       0.81      0.98      0.89     40529
           1       0.49      0.08      0.13      9809

    accuracy                           0.80     50338
   macro avg       0.65      0.53      0.51     50338
weighted avg       0.75      0.80      0.74     50338


#however my aim is to predict those who were charged away-if I get them wrong the bank loses a lot of money

#therefore I want to increase the recall of my class 1 which is "charged away", I want to predict all of them correctly

ypred_bin=[1 if i>0.05 else 0 for i in ypred]

print(classification_report(ytest, ypred_bin))
#this model shows that while my recall of class 1 is now 100%,
#I am likely to lose a large number of good paying customers because I misclassify them as people who are charged off

              precision    recall  f1-score   support

           0       0.97      0.01      0.03     40529
           1       0.20      1.00      0.33      9809

    accuracy                           0.21     50338
   macro avg       0.58      0.51      0.18     50338
weighted avg       0.82      0.21      0.09     50338




fpr, tpr, thresh=roc_curve(ytest, ypred)

fpr

array([0.        , 0.        , 0.        , ..., 0.99962989,
0.99962989,
       1.        ])

tpr

array([0.00000000e+00, 1.01947191e-04, 2.03894383e-04, ...,
       9.99898053e-01, 1.00000000e+00, 1.00000000e+00])
```

```
pd.crosstab(ytest, ypred_bin)
```

col_0	0	1
loan_status		
0	540	39989
1	16	9793

```
#confusion_matrix(ytest, ypred_bin)

plt.plot(fpr,tpr)
plt.plot([0,1],[0,1])
plt.legend(['My model','Random model'])
plt.xlabel('tpr')
plt.ylabel('fpr')
plt.title('ROC Curve shows that my model is better than a random
model')
plt.show()
```




```
roc_auc_score(ytest, ypred)
```

```
0.7027209058659821
```

```
y.value_counts() #This shows that the class which I want to predict is having very few rows
```

```
loan_status
0    202268
1     49420
Name: count, dtype: int64
```

```
#So we need to use SMOTE here to do oversampling
```

```
xtrain_sc_new_3=sc.fit_transform(xtrain_new_3)
```

```
xtest_sc_new_3=sc.transform(xtest_new_3)
```

```
import imblearn
```

```
from imblearn.over_sampling import SMOTE
```

```
smote=SMOTE()
```

```
xtrain_resample, ytrain_resample=smote.fit_resample(xtrain_sc_new_3, ytrain)
```

```
model_logit_sm=sm.Logit(ytrain_resample, xtrain_resample).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.629787
         Iterations 5
```

```
ypred_smote=model_logit_sm.predict(xtest_sc_new_3)
```

```
ypred_smote_bin=[1 if i>=0.5 else 0 for i in ypred_smote]
```

```
print(classification_report(ytest, ypred_smote_bin))
```

```
              precision    recall  f1-score   support

           0       0.89      0.60      0.72     40529
           1       0.30      0.69      0.41      9809

    accuracy                           0.62     50338
   macro avg       0.59      0.65      0.57     50338
weighted avg       0.77      0.62      0.66     50338
```

```
pd.crosstab(ytest, ypred_smote_bin) #this shows our recall for the class 1 has gone up from 8% to 69% by ensuring that our classes are balanced
```

```
col_0            0      1
loan_status
0            24342  16187
1             3014   6795
```

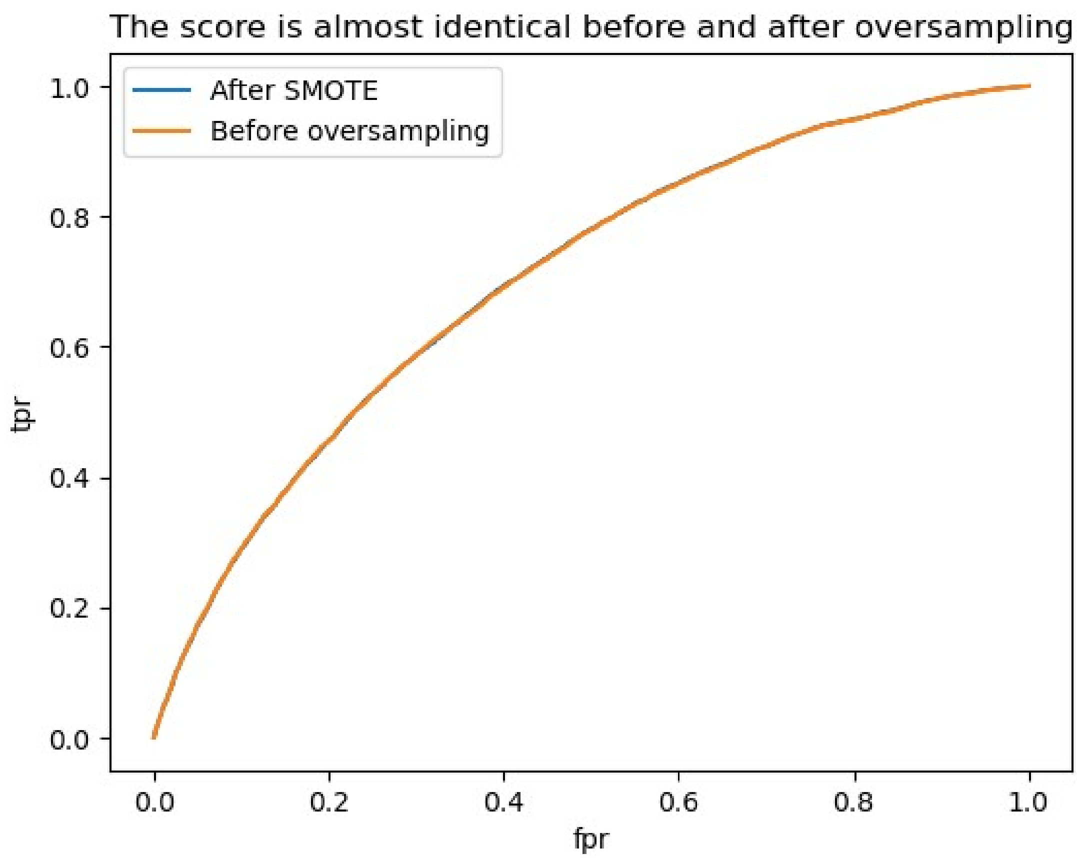
```
#To increase the recall further I am reducing the threshold
ypred_smote_bin=[1 if i>=0.2 else 0 for i in ypred_smote]

print(classification_report(ytest, ypred_smote_bin))
```

```
              precision    recall  f1-score   support

           0       0.97      0.01      0.02     40529
           1       0.20      1.00      0.33      9809

    accuracy                           0.20     50338
   macro avg       0.58      0.51      0.18     50338
weighted avg       0.82      0.20      0.08     50338
```

```
#comparing the scores for the predictions after smote (above) what is
below, we see that the recall has increased for the majority class
with the same threshold

ypred_bin_copy=[1 if i>=0.5 else 0 for i in ypred] #replicating the
older model predictions with 0.5 threshold just for comparison

print(classification_report(ytest, ypred_bin_copy))
```

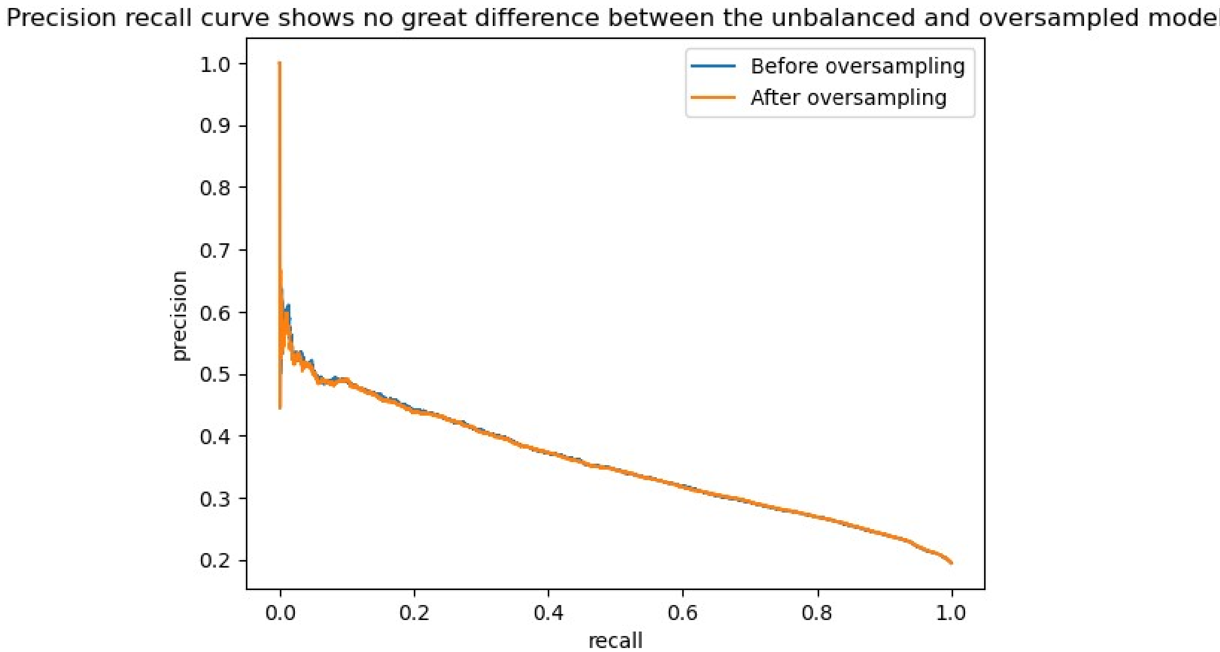
```
              precision    recall  f1-score   support

           0       0.81      0.98      0.89     40529
           1       0.49      0.08      0.13      9809

    accuracy                           0.80     50338
   macro avg       0.65      0.53      0.51     50338
weighted avg       0.75      0.80      0.74     50338
```

```
fpr_sm, tpr_sm, thresh=roc_curve(ytest, ypred_smote)
fpr, tpr, thresh=roc_curve(ytest, ypred)

plt.plot(fpr_sm, tpr_sm)
plt.plot(fpr, tpr)
plt.legend(['After SMOTE', 'Before oversampling'])
plt.title('The score is almost identical before and after
oversampling')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.show()
```




```
roc_auc_score(ytest, ypred_smote) #this is after smote

0.7029267220245609

roc_auc_score(ytest, ypred) #this is before oversampling

0.7027209058659821

#therefore, I see a very small increase in the overall performance of
the model even after oversampling and balancing
#At a threshold of 0.5 however, the increase in recall is nicely
visible because of oversampling.

#trying to see if the difference can better seen using the precision
recall score

from sklearn.metrics import precision_recall_curve

pr, re, thr=precision_recall_curve(ytest, ypred) #This is for the
model without oversampling
pr_sm, re_sm, thr=precision_recall_curve(ytest, ypred_smote) #this is
after smote

plt.plot(re, pr,label='logistic')
plt.plot(re_sm, pr_sm, label='logistic-sm')
```

```
plt.legend(['Before oversampling','After oversampling'])
plt.xlabel('recall')
plt.ylabel('precision')
plt.title('Precision recall curve shows no great difference between
the unbalanced and oversampled model')
plt.show()
```




```
from sklearn.metrics import auc

print(auc(re, pr))

0.35505049832411795

print(auc(re_sm, pr_sm))

0.35423690129110513

#Both these show that my model is actually worse than an average
model.
#This shows something wrong with my data itself since my precision
recall score should be above 0.5 which is for a random model.

#Conclusions and recommendations

#1. More loans are going to b and c grade people when compared to a
grade people and this increases the risk
#Therefore, there is an opportunity for the company to reroute most of
its loan amount to customers with A grade

#2. #People who paid up had higher incomes than people who did not,
```

```
and our data shows that most of the loans go to people with very low
incomes.
#So the company has an opportunity to identify factors other than
income which help predict whether a person pays up, and then use that
to qualify low income groups for a loan

#3.Data that on an average a higher loan is being provided to those in
the G category who have the highest risk of default.
#This means the company should lower the risk of losing money, instead
providing the highest loans to grade a customers and smaller loan to
riskier customers

#4. The bank gives more loans to riskier categories with high
interest: particularly restructuring previous loans or credit cards
#If we want to reduce the risk of default, the company should give
smaller loans at lower interest to people in safer categories,
#such as for education or for household maintenance

#So the company should aim to give loans in safer categories, instead
of giving more loans to a few high risk categories.
#Since weddings, cars, and education are the categories with lowest
default rates,
#the company should give smaller loans to more people with lower
interest rates in these categories

#5 #the company gives high interest rates to risky categories, and
penalizes innocent borrowers to account for those who are defaulting
#The most likely reason is because they do not maintain data at an
individual level and adjust it regularly based on each customer's
behaviour

#Therefore in this case, I recommend that the bank should implement a
blockchain solution
#so that they can identify who is likely to default rather than just
which category is at high risk using aggregate data
#The most common way to do this is to trace the purchase behaviour and
patterns
#from say the retail stores or supermarkets or online shopping for the
same customers by partnering with retail or credit card companies to
analyse behaviour across

#6. The overall number of people taking a loan reduces as their
experience level increases, however the number of people defaulting
remains the same
#Therefore the proportion of defaulting people increases in case of
more experienced people.
#Therefore I recommend increasing interest rates for people who higher
level of experience
#because the average loan ticket is the same as for other groups, yet
default rate is more because fewer people take a loan overall
```

```
# After feature selection and checking for multicollinearity, the most
important features to predict the loan status accurately are:
# a) sub grade
# b) DTI
# c) RENT (Whether the person is taking the loan for paying rent)
# d) Term of the loan in months

#This dataset was unbalanced so using oversampling using SMOTE helped
me improve the recall of my class 1 (the defaulter) from 8% to 69% at
a threshold of 0.5
# however it did not improve the overall performance of the model from
the roc_auc_score

#Since my objective is to maximize the recall of my class 1 (people
who defaulted), I reduced my threshold and it became 100%.
#However in this case I am then getting a very high degree of false
positive errors with a recall of class 0 of only 1% and we can end up
losing genuinely paying customers
#Also as the threshold increases my SMOTE does not have any impact on
the prediction in this dataset,
#the impact is only visible when I keep my threshold at 0.5

#Also for unbalanced data like this, I read that precision recall
curves are better and provide more information.
#In this case the score shows below 0.5 indicating that my model
performs worse than an average model would.
#However my ROC curve shows a 70% performance.
#Therefore, my understanding is that I have not done enough feature
engineering or identified the right feature combination, because very
few of the important features had a good p value of <0.05 in the
summary.

#Therefore, I will request the business to give me other behavioural
data or purchase data from other sources, or maybe data on assets,
#that has not been modified by the company, yet which influences the
outcome, because such variables will not have multicollinearity and be
independent.
```