Loan Dataset

Description

Patients with Liver disease hSolving this case study will give you an idea about how real business problems are solved using EDA. In this case study, apart from applying the techniques you have learnt in EDA, you will also develop a basic understanding of risk analytics in banking and financial services and understand how data is used to minimise the risk of losing money while lending to customers.ave been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles and drugs. This dataset was used to evaluate prediction algorithms in an effort to reduce burden on doctors.

Content

You work for a consumer finance company which specialises in lending various types of loans to urban customers. When the company receives a loan application, the company has to make a decision for loan approval based on the applicant's profile. Two types of risks are associated with the bank's decision:

```
If the applicant is likely to repay the loan, then not approving the loan results i
n a loss of business to the company
If the applicant is not likely to repay the loan, i.e. he/she is likely to default,
then approving the loan may lead to a financial loss for the company.
The data given below contains the information about past loan applicants and whethe
r they 'defaulted' or not. The aim is to identify patterns which indicate if a pers
on is likely to default, which may be used for taking actions such as denying the l
oan, reducing the amount of loan, lending (to risky applicants) at a higher interes
t rate, etc.
```

In this case study, you will use EDA to understand how consumer attributes and loan attributes influence the tendency of default. When a person applies for a loan, there are two types of decisions that could be taken by the company:

1. Loan accepted: If the company approves the loan, there are 3 possible scenarios described below: o Fully paid: Applicant has fully paid the loan (the principal and the interest rate) o Current: Applicant is in the process of paying the instalments, i.e. the tenure of the loan is not yet completed. These candidates are not labelled as 'defaulted'. o Charged-off: Applicant has not paid the instalments in due time for a long period of time, i.e. he/she has defaulted on the loan
2. Loan rejected: The company had rejected the loan (because the candidate does not meet their requirements etc.). Since the loan was rejected, there is no transactional history of those applicants with the company and so this data is not available with the company (and thus in this dataset)

Data Description

Any patient whose age exceeded 89 is listed as being of age "90".

Columns:

```
Age of the patient
Gender of the patient
Total Bilirubin
Direct Bilirubin
Alkaline Phosphotase
Alamine Aminotransferase
Aspartate Aminotransferase
Total Protiens
Albumin
Albumin and Globulin Ratio
Dataset: field used to split the data into two sets (patient with liver disease, or
no disease)
```

Business Objectives

This company is the largest online loan marketplace, facilitating personal loans, business loans, and financing of medical procedures. Borrowers can easily access lower interest rate loans through a fast online interface.

Like most other lending companies, lending loans to ‘risky’ applicants is the largest source of financial loss (called credit loss). The credit loss is the amount of money lost by the lender when the borrower refuses to pay or runs away with the money owed. In other words, borrowers who default cause the largest amount of loss to the lenders. In this case, the customers labelled as 'charged-off' are the 'defaulters'.

If one is able to identify these risky loan applicants, then such loans can be reduced thereby cutting down the amount of credit loss. Identification of such applicants using EDA is the aim of this case study.

In other words, the company wants to understand the driving factors (or driver variables) behind loan default, i.e. the variables which are strong indicators of default. The company can utilise this knowledge for its portfolio and risk assessment.

To develop your understanding of the domain, you are advised to independently research a little about risk analytics (understanding the types of variables and their significance should be enough).

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

loading data

In [2]:
```
loan_df = pd.read_csv("loan.csv")
```

```
C:\Users\rohith\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:
3063: DtypeWarning: Columns (47) have mixed types.Specify dtype option on imp
ort or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:
```
loan_df.head()
```

Out[3]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79

5 rows × 111 columns

In [4]:
```
print("Shape of the Dataset is {} Rows and {} Columns" .format(len(loan_df), l
en(loan_df.columns)))
```

```
Shape of the Dataset is 39717 Rows and 111 Columns
```

In [5]:
```
# Total number of columns in the dataset
loan_df.columns
```

Out[5]:
```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
       'term', 'int_rate', 'installment', 'grade', 'sub_grade',
       ...
       'num_tl_90g_dpd_24m', 'num_tl_op_past_12m', 'pct_tl_nvr_dlq',
       'percent_bc_gt_75', 'pub_rec_bankruptcies', 'tax_liens',
       'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit',
       'total_il_high_credit_limit'],
      dtype='object', length=111)
```

In [6]:
```
# Information about the dataset
loan_df.info()
```

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

In [7]:
```
# To know more about the dataset
loan_df.describe()
```

Out[7]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	installment
count	3.971700e+04	3.971700e+04	39717.000000	39717.000000	39717.000000	39717.000000
mean	6.831319e+05	8.504636e+05	11219.443815	10947.713196	10397.448868	324.561922
std	2.106941e+05	2.656783e+05	7456.670694	7187.238670	7128.450439	208.874874
min	5.473400e+04	7.069900e+04	500.000000	500.000000	0.000000	15.690000
25%	5.162210e+05	6.667800e+05	5500.000000	5400.000000	5000.000000	167.020000
50%	6.656650e+05	8.508120e+05	10000.000000	9600.000000	8975.000000	280.220000
75%	8.377550e+05	1.047339e+06	15000.000000	15000.000000	14400.000000	430.780000
max	1.077501e+06	1.314167e+06	35000.000000	35000.000000	35000.000000	1305.190000

8 rows × 87 columns

In [8]:
```
# Checking if there is some null values or not
loan_df.isnull()
```

Out[8]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
39712	False	False	False	False	False	False	False	False
39713	False	False	False	False	False	False	False	False
39714	False	False	False	False	False	False	False	False
39715	False	False	False	False	False	False	False	False
39716	False	False	False	False	False	False	False	False

39717 rows × 111 columns

In [9]:
```
# Checking if there is some null values or not
loan_df.isnull().sum()
```

Out[9]:
```
id                                0
member_id                         0
loan_amnt                         0
funded_amnt                       0
funded_amnt_inv                   0
                              ...
tax_liens                        39
tot_hi_cred_lim               39717
total_bal_ex_mort             39717
total_bc_limit                39717
total_il_high_credit_limit    39717
Length: 111, dtype: int64
```

In [10]:
```
loan_df['loan_amnt'].unique()
```

Out[10]:
```
array([ 5000,  2500,  2400, 10000,  3000,  7000,  5600,  5375,  6500,
       12000,  9000,  1000,  3600,  6000,  9200, 20250, 21000, 15000,
        4000,  8500,  4375, 31825, 12400, 10800, 12500,  9600,  4400,
       14000, 11000, 25600, 16000,  7100, 13000, 17500, 17675,  8000,
        3500, 16425,  8200, 20975,  6400, 14400,  7250, 18000, 35000,
       11800,  4500, 10500, 15300, 20000,  6200,  7200,  9500, 18825,
       24000,  2100,  5500, 26800, 25000, 19750, 13650, 28000, 10625,
        8850,  6375, 11100,  4200,  8875, 13500, 21600,  8450, 13475,
       22000,  7325,  7750, 13350, 22475,  8400, 13250,  7350, 11500,
       29500,  2000, 11625, 15075,  5300,  8650,  7400, 24250, 26000,
        1500, 19600,  4225, 16500, 15600, 14125, 13200, 12300,  1400,
        3200, 11875,  1800, 23200,  4800,  7300, 10400,  6600, 30000,
        4475,  6300,  8250,  9875, 21500,  7800,  9750, 15550, 17000,
        7500,  5800,  8050,  5400,  4125,  9800, 15700,  9900,  6250,
       10200, 23000, 25975, 21250, 33425,  8125, 18800, 19200, 12875,
        2625, 11300,  4100, 18225, 18500, 16800,  2200, 14050, 16100,
       10525, 19775, 14500, 11700,  4150, 12375,  1700, 22250, 11200,
       22500, 15900,  3150, 18550,  8575,  7700, 24500, 22200, 21400,
        9400, 22400,  5825,  7650, 20675, 27050, 20500, 12800, 27575,
        7600, 29000,  9575, 14575,  7125, 10700, 10375,  3050, 27000,
       28625, 14100, 20050, 24925, 13600, 26400,  7150, 32000, 15500,
       17475,  2250, 17050,  3250, 22750,  1200,  5900, 12600,  6750,
       17250, 19075, 17200, 13225, 11775, 16400, 10075,  9350,  8075,
       15625, 20125,  8300,  2425,  6950,  5350,  5875,  9450, 19000,
       20400, 21650, 20300,  2300, 24575,  5850,  4750,  5275,  9175,
       34475, 10050, 19400, 18200,  8800, 34000, 19500,  5200, 11900,
       29100, 25850,  3300, 12200, 22575,  7175, 18250, 16750, 12950,
        6350, 14750,  6625,  6900, 18650,  9250, 22800, 27300, 12250,
        4350, 21200,  2700,  6025,  3825,  5325, 14150,  1600,  2800,
       18975,  2575,  5450,  3800,  2125, 14650, 11250, 31000,  6075,
        8475,  3625, 31300,  4250, 12650, 27600, 13150,  4300, 10275,
       23600,  7875, 14550,  9925, 15850,  1325,  6325, 29700, 15200,
       28100, 15250,  6800, 11325, 13975, 13800,  3100,  3975, 25450,
        3575, 33600, 23700, 28200,  6475, 27700, 17375, 15800, 17625,
       16675,  5250, 22950,  1950,  4650, 10250,  6100,  8325,  4850,
        9425, 12700, 25475, 14850, 14300, 33000,  5150, 21625,  3775,
       21575, 16250,  8375, 18725, 11125,  3525, 19800,  9300, 19125,
        5575,  1450, 12900, 10150, 20450, 23500, 16600,  1300,  6925,
       14675, 11550, 17400,  1100,  3400, 12775,  5050, 12100, 26375,
        6975, 26300,  3125, 23325, 11600,  5100, 10175, 18400, 30750,
       16550,  5650, 16450, 18950,  3650, 33950, 10125, 16775,  5700,
       20200, 10600,  3725, 19425, 25900, 23800,  4025,  2600,  8900,
       10900, 17600, 14825,  7925, 14950,  6700,  8600,  1925, 30500,
        4900, 15575,  3175, 14800, 32275,  5750, 14600, 25200,  6550,
       30400, 22900,  6850,  4600, 11425, 16950, 29850, 10675,  6650,
       10775, 17325, 27250,  3700,  6450, 20800, 13575, 29275,  4725,
       24800, 15750, 17100, 15875, 10925,  4950, 10575,  2850, 32875,
       21100, 11050, 20375,  9325,  9375,  7475, 22125, 27525, 25500,
       17750,  8675,  7450, 24625, 17900, 12075,  6725, 24400,  5225,
       14075, 17175,  9475,  9975, 20900, 12150, 17725, 15350,  4925,
        4550, 18750, 15125, 10950, 12475,  2750,  4625, 12175,  7575,
       23525, 12350, 17950,  9525,  8975, 11975, 12850, 19850, 21850,
        4425, 32250,  2550, 11400, 21725, 23100, 13700,  9950, 21750,
       13750, 12025, 23400, 14975, 19700, 27500,  3900, 14725, 17800,
        5175, 15025, 29550, 23850, 31500,  9100, 27400, 23675,  9825,
       16200, 11650, 18875, 29175,  3950,  2050, 19950, 12750, 24375,
        2875, 25875, 16275, 10300, 17450,  3450,  1825, 13100, 23275,
```

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

In [11]:
```
loan_df['int_rate'].unique()
```

Out[11]:
```
array(['10.65%', '15.27%', '15.96%', '13.49%', '12.69%', '7.90%',
       '18.64%', '21.28%', '14.65%', '9.91%', '16.29%', '6.03%', '11.71%',
       '12.42%', '14.27%', '16.77%', '7.51%', '8.90%', '18.25%', '6.62%',
       '19.91%', '17.27%', '17.58%', '21.67%', '19.42%', '22.06%',
       '20.89%', '20.30%', '23.91%', '19.03%', '23.52%', '23.13%',
       '22.74%', '22.35%', '24.11%', '6.00%', '22.11%', '7.49%', '11.99%',
       '5.99%', '10.99%', '9.99%', '18.79%', '11.49%', '8.49%', '15.99%',
       '16.49%', '6.99%', '12.99%', '15.23%', '14.79%', '5.42%', '10.59%',
       '17.49%', '15.62%', '21.36%', '19.29%', '13.99%', '18.39%',
       '16.89%', '17.99%', '20.62%', '20.99%', '22.85%', '19.69%',
       '20.25%', '23.22%', '21.74%', '22.48%', '23.59%', '12.62%',
       '18.07%', '11.63%', '7.91%', '7.42%', '11.14%', '20.20%', '12.12%',
       '19.39%', '16.11%', '17.54%', '22.64%', '13.84%', '16.59%',
       '17.19%', '12.87%', '20.69%', '9.67%', '21.82%', '19.79%',
       '18.49%', '22.94%', '24.59%', '24.40%', '21.48%', '14.82%',
       '14.17%', '7.29%', '17.88%', '20.11%', '16.02%', '17.51%',
       '13.43%', '14.91%', '13.06%', '15.28%', '15.65%', '17.14%',
       '11.11%', '10.37%', '16.40%', '7.66%', '10.00%', '18.62%',
       '10.74%', '5.79%', '6.92%', '9.63%', '14.54%', '12.68%', '19.36%',
       '13.80%', '18.99%', '21.59%', '20.85%', '21.22%', '19.74%',
       '20.48%', '6.91%', '12.23%', '12.61%', '10.36%', '6.17%', '6.54%',
       '9.25%', '16.69%', '15.95%', '8.88%', '13.35%', '9.62%', '16.32%',
       '12.98%', '14.83%', '13.72%', '14.09%', '14.46%', '20.03%',
       '17.80%', '15.20%', '15.57%', '18.54%', '19.66%', '17.06%',
       '18.17%', '17.43%', '20.40%', '20.77%', '18.91%', '21.14%',
       '17.44%', '13.23%', '7.88%', '11.12%', '13.61%', '10.38%',
       '17.56%', '17.93%', '15.58%', '13.98%', '14.84%', '15.21%',
       '6.76%', '6.39%', '11.86%', '7.14%', '14.35%', '16.82%', '10.75%',
       '14.72%', '16.45%', '18.67%', '20.53%', '19.41%', '20.16%',
       '21.27%', '18.30%', '19.04%', '20.90%', '21.64%', '12.73%',
       '10.25%', '13.11%', '10.62%', '13.48%', '14.59%', '16.07%',
       '15.70%', '9.88%', '11.36%', '15.33%', '13.85%', '14.96%',
       '14.22%', '7.74%', '13.22%', '13.57%', '8.59%', '17.04%', '14.61%',
       '8.94%', '12.18%', '11.83%', '11.48%', '16.35%', '13.92%',
       '15.31%', '14.26%', '19.13%', '12.53%', '16.70%', '16.00%',
       '17.39%', '18.09%', '7.40%', '18.43%', '17.74%', '7.05%', '20.52%',
       '20.86%', '19.47%', '18.78%', '21.21%', '19.82%', '20.17%',
       '13.16%', '8.00%', '13.47%', '12.21%', '16.63%', '9.32%', '12.84%',
       '11.26%', '15.68%', '15.37%', '10.95%', '11.89%', '14.11%',
       '13.79%', '7.68%', '11.58%', '7.37%', '16.95%', '15.05%', '18.53%',
       '14.74%', '14.42%', '18.21%', '17.26%', '18.84%', '17.90%',
       '19.16%', '13.67%', '9.38%', '12.72%', '13.36%', '11.46%',
       '10.51%', '9.07%', '13.04%', '11.78%', '12.41%', '10.83%',
       '12.09%', '17.46%', '14.30%', '17.15%', '15.25%', '10.20%',
       '15.88%', '14.93%', '16.20%', '18.72%', '14.62%', '8.32%',
       '14.12%', '10.96%', '10.33%', '10.01%', '12.86%', '11.28%',
       '11.59%', '8.63%', '12.54%', '12.22%', '11.91%', '15.38%',
       '16.96%', '13.17%', '9.70%', '16.33%', '14.75%', '15.07%',
       '16.01%', '10.71%', '10.64%', '9.76%', '11.34%', '10.39%',
       '13.87%', '11.03%', '11.66%', '13.24%', '10.08%', '9.45%',
       '13.55%', '12.29%', '11.97%', '12.92%', '15.45%', '14.50%',
       '14.18%', '15.13%', '16.08%', '15.76%', '17.03%', '17.34%',
       '16.71%', '9.83%', '13.62%', '10.46%', '9.51%', '9.20%', '13.30%',
       '10.78%', '7.75%', '8.38%', '12.36%', '12.67%', '11.72%', '13.93%',
       '8.07%', '7.43%', '12.04%', '14.25%', '14.88%', '11.41%', '11.09%',
       '10.14%', '16.15%', '15.83%', '7.12%', '18.36%', '9.64%', '9.96%',
       '11.22%', '9.01%', '9.33%', '11.54%', '12.17%', '12.80%', '14.38%',
```

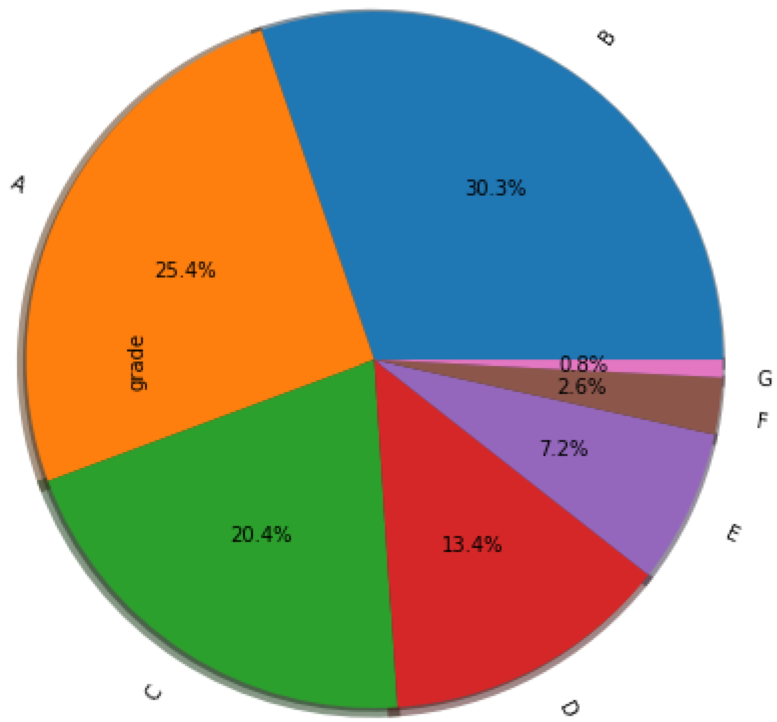
```
'13.75%', '14.70%', '12.49%', '14.07%', '10.91%', '13.12%',
'10.28%', '8.70%', '14.67%', '15.01%'], dtype=object)
```

Data Analysis and Visualization

In [12]:
```
loan_df['grade'].unique()
```

Out[12]:
```
array(['B', 'C', 'A', 'E', 'F', 'D', 'G'], dtype=object)
```

In [13]:
```
(loan_df["grade"].value_counts()).plot.pie(autopct="%.1f%%", shadow=True,rotat
elabels=True, wedgeprops={'linewidth': 6}, radius=2)
plt.show()
```



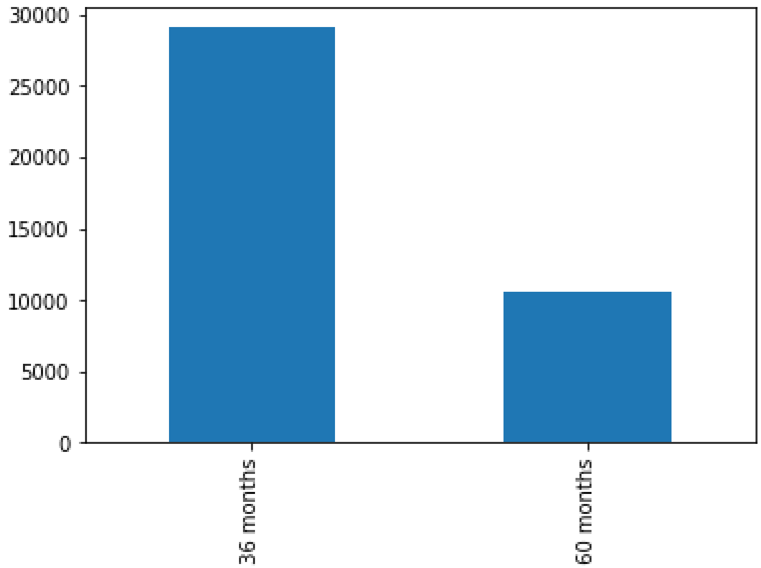
In [14]:
```
loan_df['term'].unique()
```

Out[14]:
```
array([' 36 months', ' 60 months'], dtype=object)
```

In [15]:
```
loan_df.term.value_counts().plot(kind='bar')
```

Out[15]:

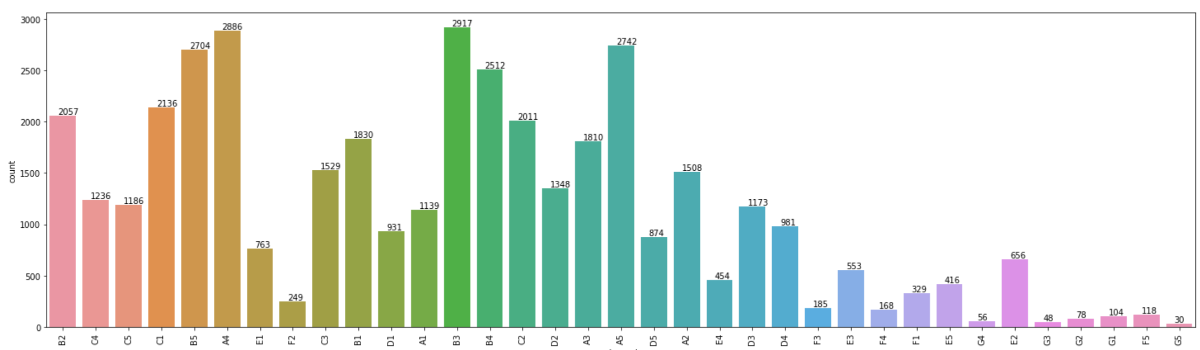
```
<matplotlib.axes._subplots.AxesSubplot at 0x304e264108>
```



In [16]:
```
loan_df['sub_grade'].unique()
```

Out[16]:
```
array(['B2', 'C4', 'C5', 'C1', 'B5', 'A4', 'E1', 'F2', 'C3', 'B1', 'D1',
       'A1', 'B3', 'B4', 'C2', 'D2', 'A3', 'A5', 'D5', 'A2', 'E4', 'D3',
       'D4', 'F3', 'E3', 'F4', 'F1', 'E5', 'G4', 'E2', 'G3', 'G2', 'G1',
       'F5', 'G5'], dtype=object)
```

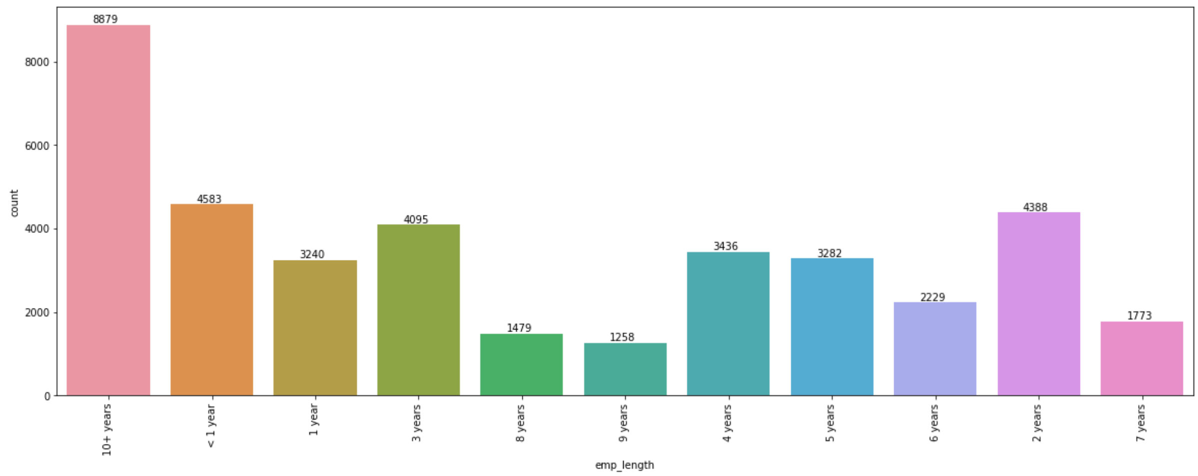
In [17]:
```
plt.figure(figsize = (25,7))
ax=sns.countplot(x = 'sub_grade', data = loan_df)
plt.xticks(rotation = 90)
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va='b
ottom', color= 'black')
```



In [18]:
```
loan_df['emp_length'].unique()
```

Out[18]:
```
array(['10+ years', '< 1 year', '1 year', '3 years', '8 years', '9 years',
       '4 years', '5 years', '6 years', '2 years', '7 years', nan],
      dtype=object)
```

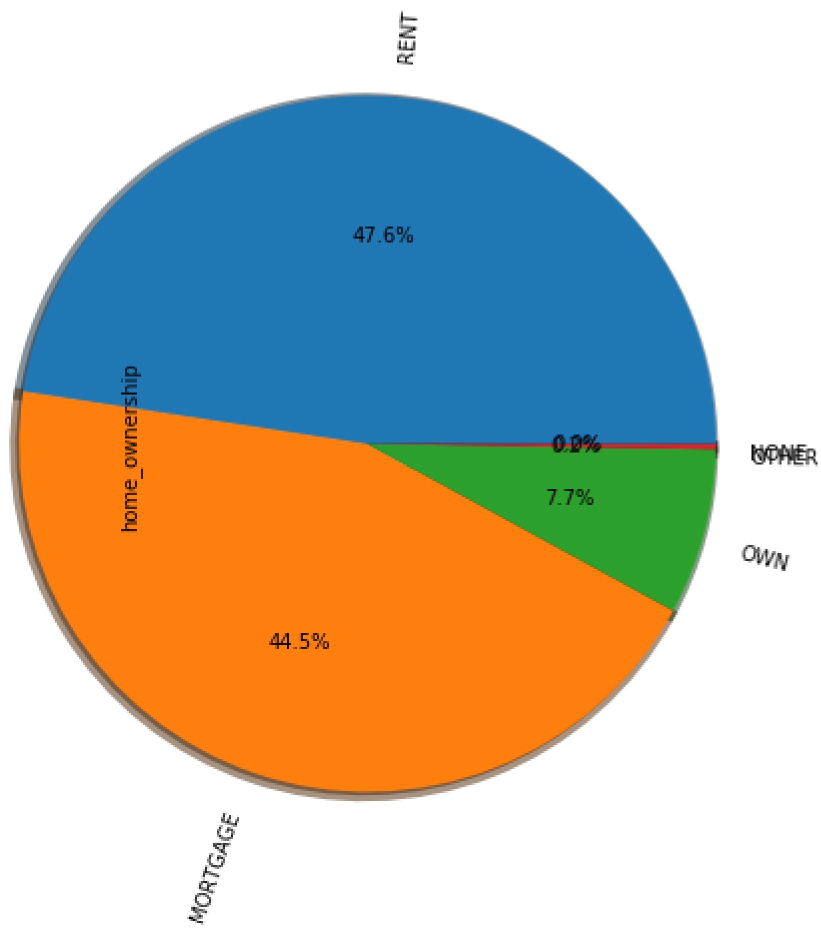
In [19]:
```
plt.figure(figsize = (20,7))
ax=sns.countplot(x = 'emp_length', data = loan_df)
plt.xticks(rotation = 90)
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va='b
ottom', color= 'black')
```



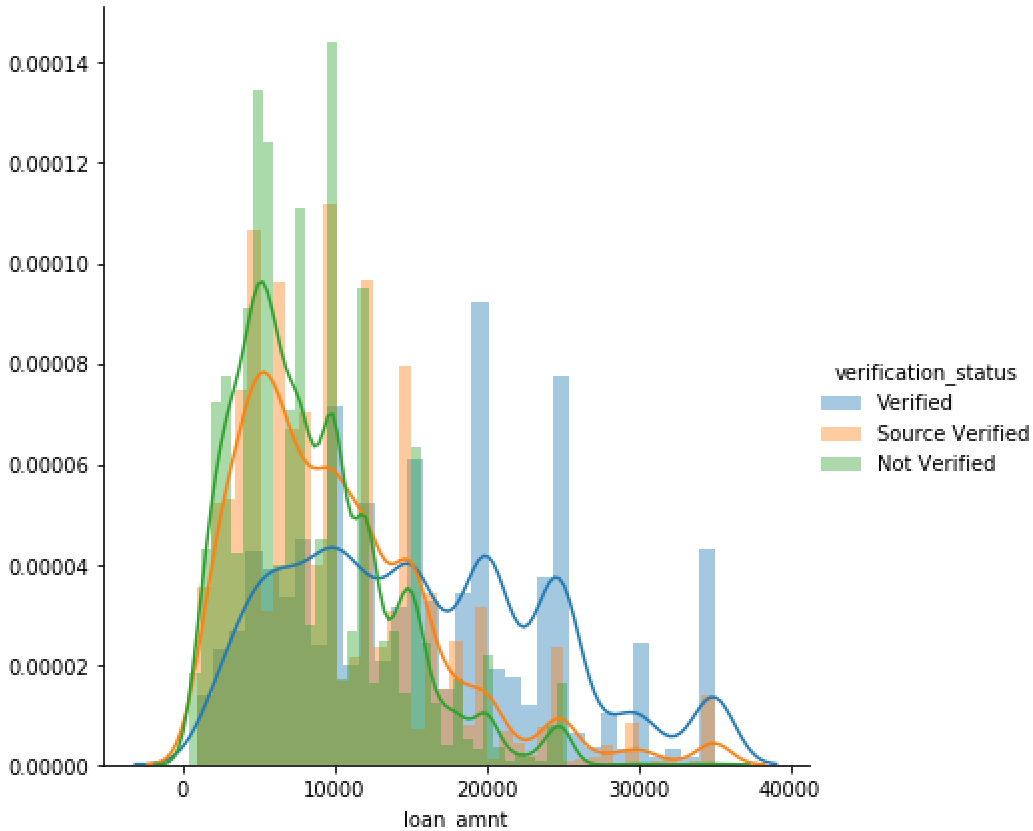
In [20]:
```
loan_df['home_ownership'].unique()
```

Out[20]:
```
array(['RENT', 'OWN', 'MORTGAGE', 'OTHER', 'NONE'], dtype=object)
```

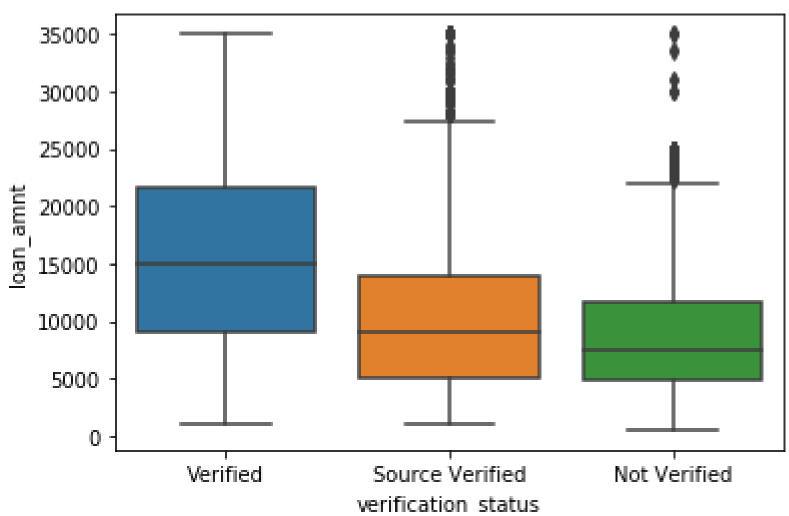
In [21]:
```
(loan_df["home_ownership"].value_counts()).plot.pie(autopct="%.1f%%", shadow=True,rotatelabels=True, wedgeprops={'linewidth': 6}, radius=2)
plt.show()
```



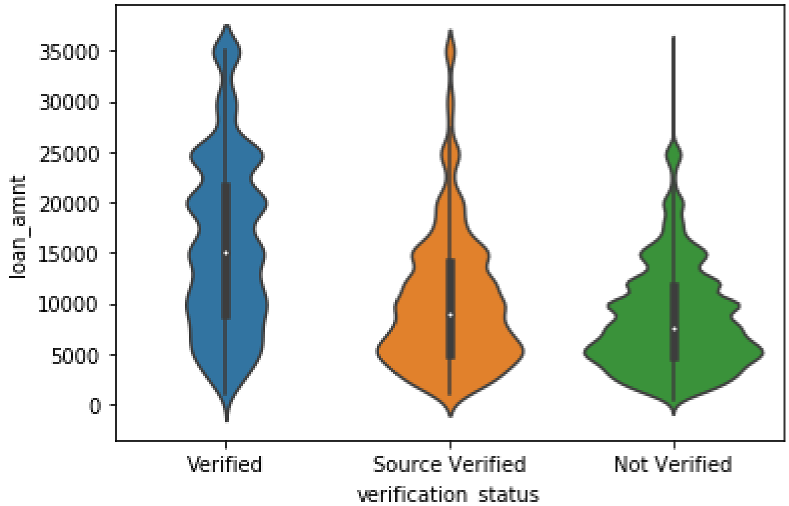
In [22]:
```
sns.FacetGrid(loan_df, hue='verification_status', height=6).map(sns.distplot,
'loan_amnt').add_legend()
plt.show()
```



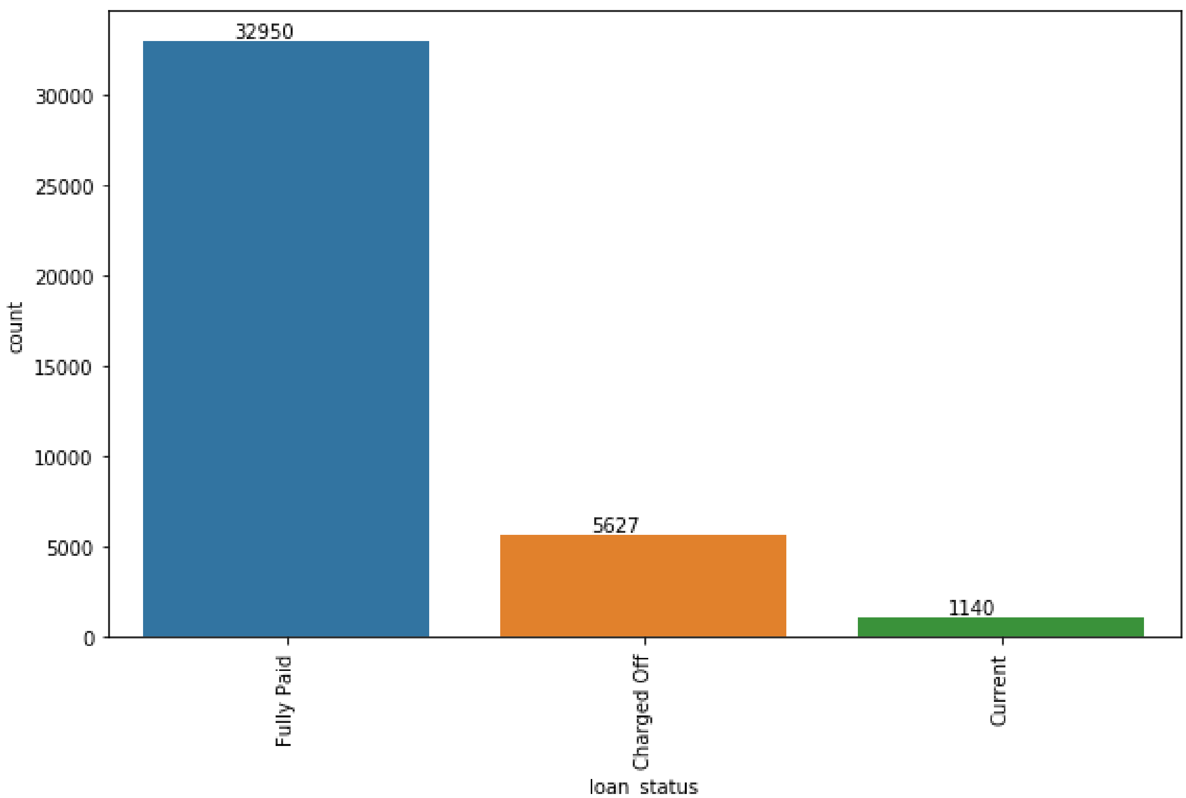
In [23]:
```
sns.boxplot(x = 'verification_status', y = 'loan_amnt', data = loan_df)
plt.show()
```



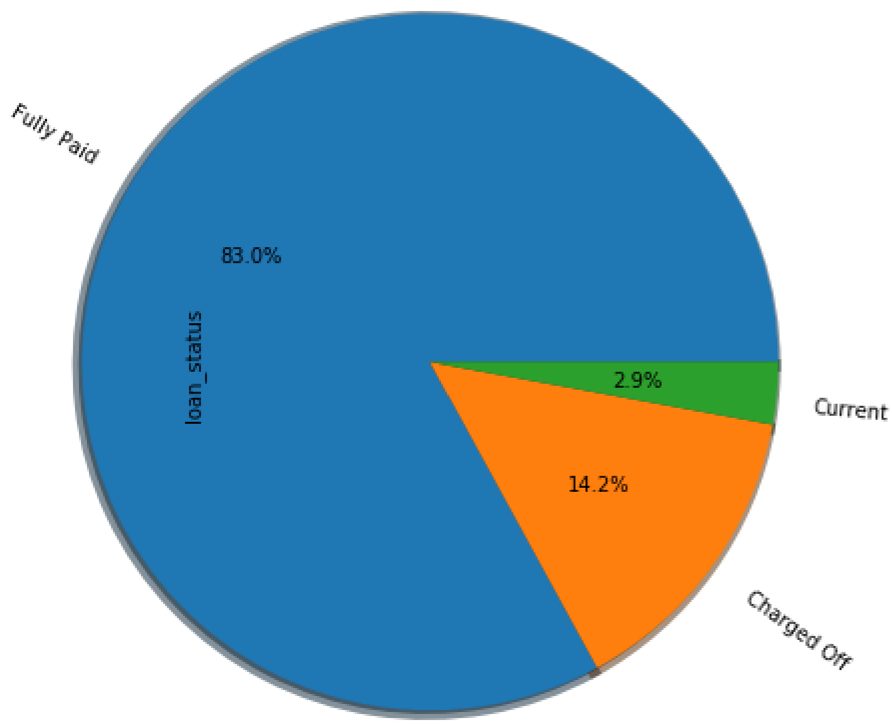
In [24]:
```
sns.violinplot(x = 'verification_status', y = 'loan_amnt', data = loan_df)
plt.show()
```



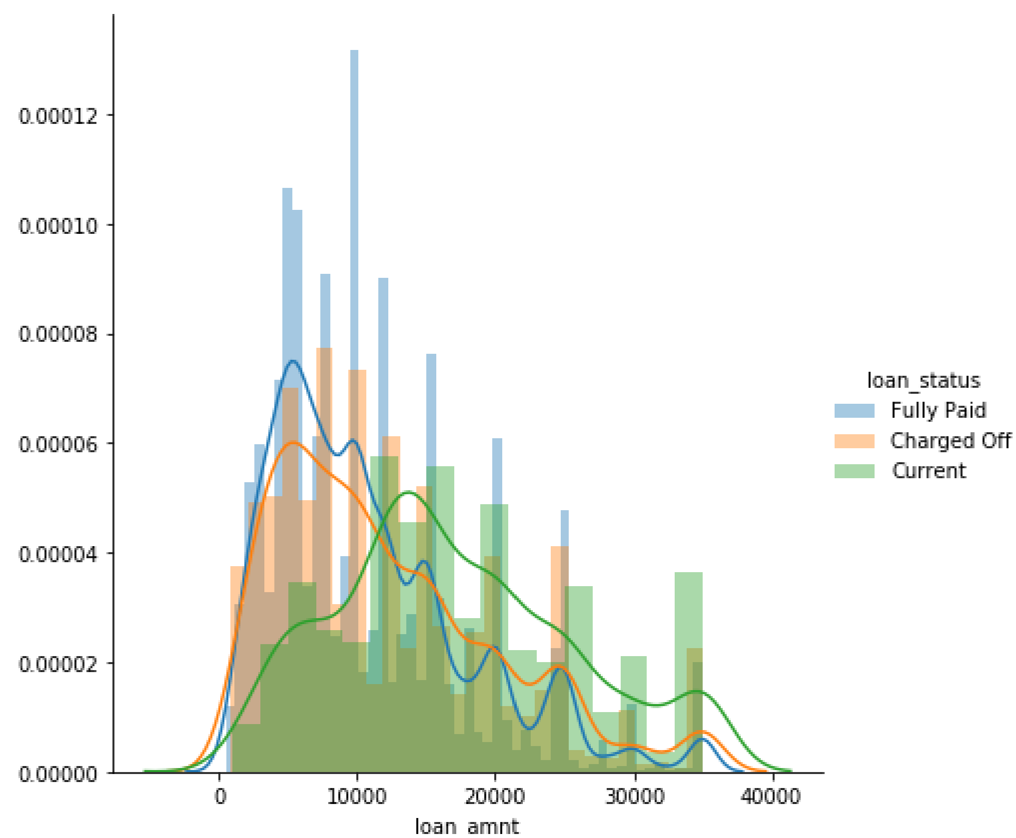
In [25]:
```
plt.figure(figsize = (10,6))
ax=sns.countplot(x = 'loan_status', data = loan_df)
plt.xticks(rotation = 90)
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va='b
ottom', color= 'black')
```



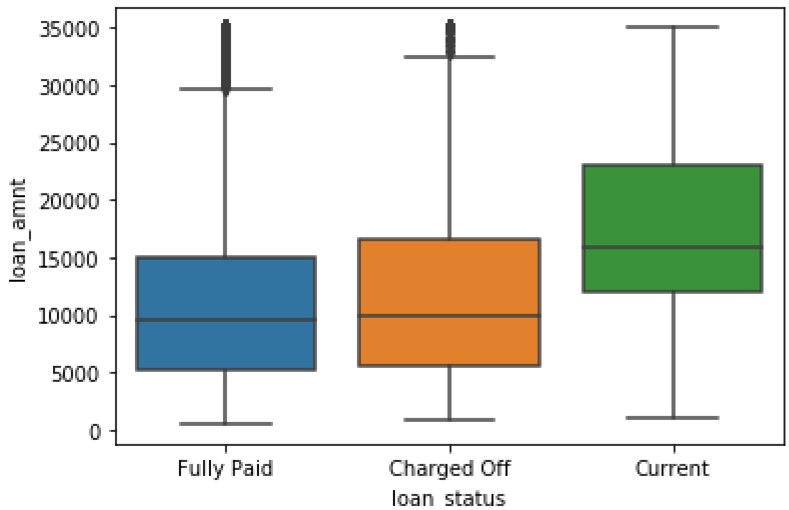
In [26]:
```
(loan_df["loan_status"].value_counts()).plot.pie(autopct="%.1f%%", shadow=True
,rotatelabels=True, wedgeprops={'linewidth': 6}, radius=2)
plt.show()
```



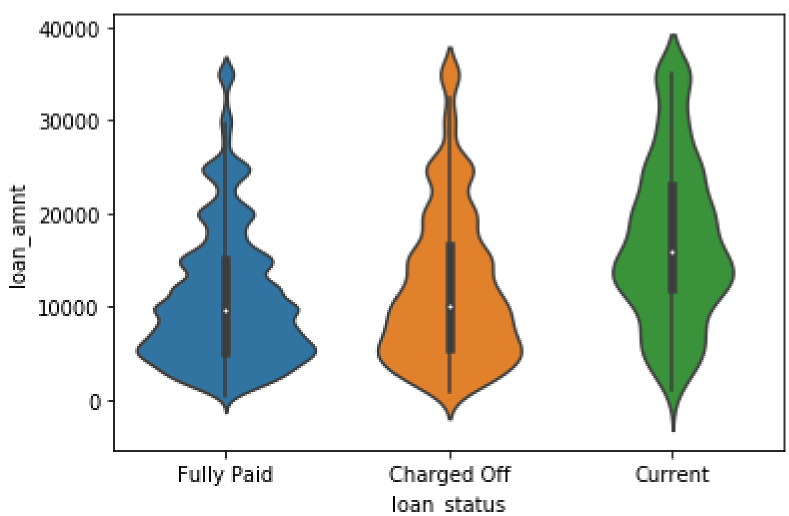
In [27]:
```
sns.FacetGrid(loan_df, hue='loan_status', height=6).map(sns.distplot, 'loan_am
nt').add_legend()
plt.show()
```



In [28]:
```
sns.boxplot(x = 'loan_status', y = 'loan_amnt', data = loan_df)
plt.show()
```



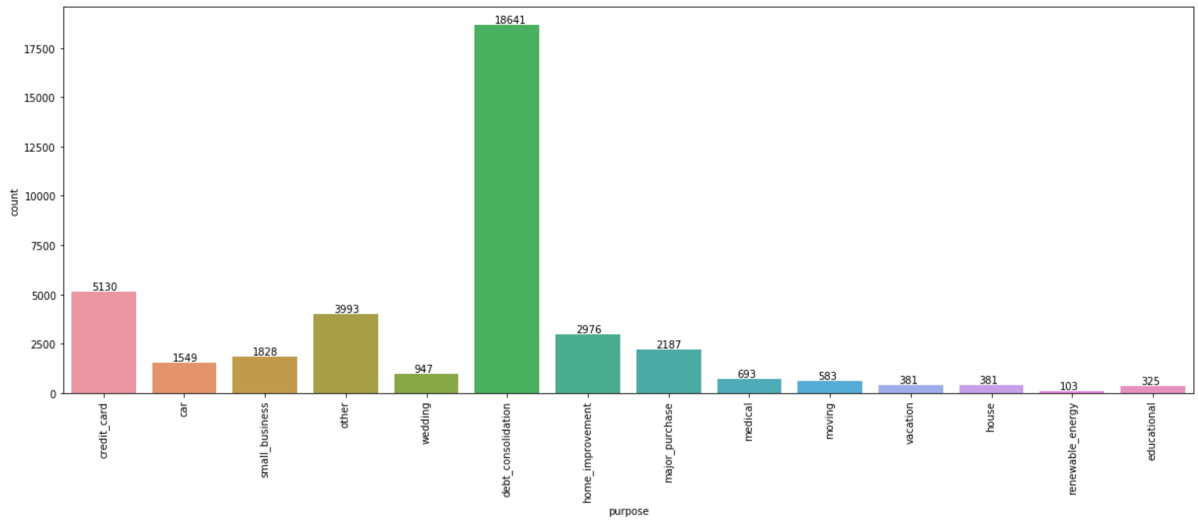
In [29]:
```
sns.violinplot(x = 'loan_status', y = 'loan_amnt', data = loan_df)
plt.show()
```



In [30]:
```
loan_df['purpose'].unique()
```

Out[30]:
```
array(['credit_card', 'car', 'small_business', 'other', 'wedding',
       'debt_consolidation', 'home_improvement', 'major_purchase',
       'medical', 'moving', 'vacation', 'house', 'renewable_energy',
       'educational'], dtype=object)
```

In [31]:
```
plt.figure(figsize = (20,7))
ax=sns.countplot(x = 'purpose', data = loan_df)
plt.xticks(rotation = 90)
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va='bottom', color= 'black')
```



In [32]:
```
loan_df['title'].unique()
```

Out[32]:
```
array(['Computer', 'bike', 'real estate business', ...,
       'Retiring credit card debt', 'MBA Loan Consolidation', 'JAL Loan'],
      dtype=object)
```

In [33]:
```
sns.catplot(x = 'loan_status', y = 'loan_amnt', data = loan_df)
plt.show()
```