loan case study:random forest

In [2]:

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

Loading and Cleaning Data

In [3]:

```
loan= pd.read_csv('E:/301/loan.csv', delimiter = ',', encoding =  'ISO-8859-1')
loan
```

```
c:\users\hari\appdata\local\programs\python\python38\lib\site-
packages\IPython\core\interactiveshell.py:3062: DtypeWarning: Columns (47) have mixed
types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Out[3]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	num_tl_90
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	...	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	...	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	C5	...	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	...	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	...	
...	...	...	...	...	...	...	...	...	...	...	...	
39712	92187	92174	2500	2500	1075.0	36 months	8.07%	78.42	A	A4	...	
39713	90665	90607	8500	8500	875.0	36 months	10.28%	275.38	C	C1	...	
39714	90395	90390	5000	5000	1325.0	36 months	8.07%	156.84	A	A4	...	
39715	90376	89243	5000	5000	650.0	36 months	7.43%	155.38	A	A2	...	
39716	87023	86999	7500	7500	800.0	36 months	13.75%	255.43	E	E2	...	

39717 rows × 111 columns

Inspect the dataframe

In [4]:

```
loan.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 [5]:

```
loan.shape
```

Out[5]:

```
(39717, 111)
```

In [6]:

```
loan.head()
```

Out[6]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	num_tl_90g_d
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	...	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	...	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	C5	...	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	...	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	...	

5 rows × 111 columns

In [7]:

```
loan.tail()
```

Out[7]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	num_tl_90g_
39712	92187	92174	2500	2500	1075.0	36 months	8.07%	78.42	A	A4	...	
39713	90665	90607	8500	8500	875.0	36 months	10.28%	275.38	C	C1	...	
39714	90395	90390	5000	5000	1325.0	36 months	8.07%	156.84	A	A4	...	
39715	90376	89243	5000	5000	650.0	36 months	7.43%	155.38	A	A2	...	
39716	87023	86999	7500	7500	800.0	36 months	13.75%	255.43	E	E2	...	

5 rows × 111 columns

In [8]:

```
loan.describe()
```

Out[8]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	installment	annual_inc	dti	delinq_
count	3.971700e+04	3.971700e+04	39717.000000	39717.000000	39717.000000	39717.000000	3.971700e+04	39717.000000	39717.00
mean	6.831319e+05	8.504636e+05	11219.443815	10947.713196	10397.448868	324.561922	6.896893e+04	13.315130	0.14
std	2.106941e+05	2.656783e+05	7456.670694	7187.238670	7128.450439	208.874874	6.379377e+04	6.678594	0.49
min	5.473400e+04	7.069900e+04	500.000000	500.000000	0.000000	15.690000	4.000000e+03	0.000000	0.00
25%	5.162210e+05	6.667800e+05	5500.000000	5400.000000	5000.000000	167.020000	4.040400e+04	8.170000	0.00
50%	6.656650e+05	8.508120e+05	10000.000000	9600.000000	8975.000000	280.220000	5.900000e+04	13.400000	0.00
75%	8.377550e+05	1.047339e+06	15000.000000	15000.000000	14400.000000	430.780000	8.230000e+04	18.600000	0.00

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	installment	annual_inc	dti	delinq
max	1.077501e+06	1.314167e+06	35000.000000	35000.000000	35000.000000	1305.190000	6.000000e+06	29.990000	11.00

8 rows × 87 columns

Treating Missing Values in Column

In [9]:

```
round(100*(loan.isnull().sum(axis=0))/len(loan.index),2)
```

Out[9]:

```
id                              0.0
member_id                       0.0
loan_amnt                       0.0
funded_amnt                     0.0
funded_amnt_inv                 0.0
                               ...
tax_liens                       0.1
tot_hi_cred_lim               100.0
total_bal_ex_mort             100.0
total_bc_limit                100.0
total_il_high_credit_limit    100.0
Length: 111, dtype: float64
```

In [10]:

```
loan.isnull().sum(axis=0)
```

Out[10]:

```
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 [11]:

```
loan=loan.drop(['delinq_2yrs','earliest_cr_line','inq_last_6mths','open_acc','pub_rec','revol_bal'
,'revol_util','total_acc','out_prncp',

'out_prncp_inv','total_pymnt_inv','total_rec_prncp','total_rec_int','total_rec_late_fee','recoverie
s',
               'collection_recovery_fee','last_pymnt_d','last_pymnt_amnt','next_pymnt_d','last_crec
it_pull_d','application_type'], axis=1)
```

In [12]:

```
loan.isnull().sum(axis=0)
```

Out[12]:

```
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: 90, dtype: int64
```

In [13]:

```
loan= loan.loc[:,round(100*(loan.isnull().sum()/len(loan.index)),2) < 80]
```

In [14]:

```
round(100*(loan.isnull().sum(axis=0))/len(loan.index),2)
```

Out[14]:

```
id                            0.00
member_id                     0.00
loan_amnt                     0.00
funded_amnt                   0.00
funded_amnt_inv               0.00
term                          0.00
int_rate                      0.00
installment                   0.00
grade                         0.00
sub_grade                     0.00
emp_title                     6.19
emp_length                    2.71
home_ownership                0.00
annual_inc                    0.00
verification_status           0.00
issue_d                       0.00
loan_status                   0.00
pymnt_plan                    0.00
url                           0.00
desc                         32.58
purpose                       0.00
title                         0.03
zip_code                      0.00
addr_state                    0.00
dti                           0.00
mths_since_last_delinq       64.66
initial_list_status           0.00
total_pymnt                   0.00
collections_12_mths_ex_med    0.14
policy_code                   0.00
acc_now_delinq                0.00
chargeoff_within_12_mths      0.14
delinq_amnt                   0.00
pub_rec_bankruptcies          1.75
tax_liens                     0.10
dtype: float64
```

In [15]:

```
loan =
loan.drop(['member_id','id','acc_now_delinq','chargeoff_within_12_mths','pymnt_plan','initial_list_
status','delinq_amnt','pub_rec_bankruptcies','tax_liens','collections_12_mths_ex_med','policy_code'
,'url','emp_title','zip_code','addr_state','title','desc'], axis=1)
```

In [16]:

```
# Verifying Null value percentage in each column
round(100*(loan.isnull().sum(axis=0))/len(loan.index),2)
```

Out[16]:

```
loan_amnt                0.00
funded_amnt              0.00
funded_amnt_inv          0.00
term                     0.00
int_rate                 0.00
installment              0.00
grade                    0.00
sub_grade                0.00
emp_length               2.71
```

```
emp_length                2.71
home_ownership            0.00
annual_inc                0.00
verification_status       0.00
issue_d                   0.00
loan_status               0.00
purpose                   0.00
dti                       0.00
mths_since_last_delinq   64.66
total_pymnt               0.00
dtype: float64
```

Treating Missing Values in Row

In [17]:

```
loan = loan[~pd.isnull(loan['mths_since_last_delinq'])]
loan = loan[~pd.isnull(loan['emp_length'])]
```

In [18]:

```
round(100*(loan.isnull().sum(axis=0))/len(loan.index),2)
```

Out[18]:

```
loan_amnt                 0.0
funded_amnt               0.0
funded_amnt_inv           0.0
term                      0.0
int_rate                  0.0
installment               0.0
grade                     0.0
sub_grade                 0.0
emp_length                0.0
home_ownership            0.0
annual_inc                0.0
verification_status       0.0
issue_d                   0.0
loan_status               0.0
purpose                   0.0
dti                       0.0
mths_since_last_delinq    0.0
total_pymnt               0.0
dtype: float64
```

In [19]:

```
loan
```

Out[19]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_length	home_ownership	annu
3	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	10+ years	RENT	4
4	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	1 year	RENT	8
16	10000	10000	10000.0	36 months	15.27%	347.98	C	C4	4 years	RENT	4
18	6000	6000	6000.0	36 months	11.71%	198.46	B	B3	1 year	MORTGAGE	8
27	5000	5000	5000.0	60 months	16.77%	123.65	D	D2	2 years	RENT	5
...	...	...	...	...	...	...	...	...	...	...	
39712	2500	2500	1075.0	36 months	8.07%	78.42	A	A4	4 years	MORTGAGE	11
39713	8500	8500	875.0	36 months	10.28%	275.38	C	C1	3 years	RENT	1
39714	5000	5000	1025.0	36	8.07%	156.84	A	A4	1 year	MORTGAGE	10

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_length	home_ownership	annu
39714	5000	5000	1325.0	months	8.07%	156.84	A	A4	< 1 year	MORTGAGE	1(
39715	5000	5000	650.0	36 months	7.43%	155.38	A	A2	< 1 year	MORTGAGE	2(
39716	7500	7500	800.0	36 months	13.75%	255.43	E	E2	< 1 year	OWN	2

13690 rows × 18 columns

Rectifying the values

In [20]:

```
loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13690 entries, 3 to 39716
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   loan_amnt               13690 non-null  int64  
 1   funded_amnt             13690 non-null  int64  
 2   funded_amnt_inv         13690 non-null  float64
 3   term                    13690 non-null  object 
 4   int_rate                13690 non-null  object 
 5   installment             13690 non-null  float64
 6   grade                   13690 non-null  object 
 7   sub_grade               13690 non-null  object 
 8   emp_length              13690 non-null  object 
 9   home_ownership          13690 non-null  object 
 10  annual_inc              13690 non-null  float64
 11  verification_status     13690 non-null  object 
 12  issue_d                 13690 non-null  object 
 13  loan_status             13690 non-null  object 
 14  purpose                 13690 non-null  object 
 15  dti                     13690 non-null  float64
 16  mths_since_last_delinq  13690 non-null  float64
 17  total_pymnt             13690 non-null  float64
dtypes: float64(6), int64(2), object(10)
memory usage: 2.0+ MB
```

In [21]:

```
loan['int_rate'] = loan['int_rate'].str.replace('%', '')
loan['int_rate']=loan['int_rate'].astype(float)
loan['int_rate']=loan['int_rate'].apply(lambda x : x/100)
loan
```

Out[21]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_length	home_ownership	annu
3	10000	10000	10000.0	36 months	0.1349	339.31	C	C1	10+ years	RENT	4
4	3000	3000	3000.0	60 months	0.1269	67.79	B	B5	1 year	RENT	8
16	10000	10000	10000.0	36 months	0.1527	347.98	C	C4	4 years	RENT	4
18	6000	6000	6000.0	36 months	0.1171	198.46	B	B3	1 year	MORTGAGE	8
27	5000	5000	5000.0	60 months	0.1677	123.65	D	D2	2 years	RENT	5
...	...	...	...	...	...	...	...	...	...	...	
39712	2500	2500	1075.0	36 months	0.0807	78.42	A	A4	4 years	MORTGAGE	11
39713	8500	8500	875.0	36 months	0.1028	275.38	C	C1	3 years	RENT	1
39714	5000	5000	1325.0	36 months	0.0807	156.84	A	A4	< 1 year	MORTGAGE	1(

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_length	home_ownership	annu
39715	5000	5000	650.0	36 months	0.0743	155.38	A	A2	< 1 year	MORTGAGE	20
39716	7500	7500	800.0	36 months	0.1375	255.43	E	E2	< 1 year	OWN	2

13690 rows × 18 columns

In [22]:

```
loan['term'] = loan['term'].apply(lambda x : x[:3])
loan['term']=loan['term'].astype(int)
loan['emp_length'] = loan['emp_length'].str.replace('years','')
loan['emp_length'] = loan['emp_length'].str.replace('year','')

import datetime
from datetime import datetime
loan['issue_d'] = loan['issue_d'].apply(lambda x: datetime.strptime(x, '%b-%y'))

loan['emp_length']= loan['emp_length'].str.strip()
loan.loc[loan['emp_length']=='< 1','emp_length']=0
loan.loc[loan['emp_length']=='10+','emp_length']=10
loan['emp_length']=loan['emp_length'].astype(int)

loan
```

Out[22]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_length	home_ownership	annua
3	10000	10000	10000.0	36	0.1349	339.31	C	C1	10	RENT	492
4	3000	3000	3000.0	60	0.1269	67.79	B	B5	1	RENT	800
16	10000	10000	10000.0	36	0.1527	347.98	C	C4	4	RENT	420
18	6000	6000	6000.0	36	0.1171	198.46	B	B3	1	MORTGAGE	840
27	5000	5000	5000.0	60	0.1677	123.65	D	D2	2	RENT	500
...	...	...	...	...	...	...	...	...	...	...	
39712	2500	2500	1075.0	36	0.0807	78.42	A	A4	4	MORTGAGE	1100
39713	8500	8500	875.0	36	0.1028	275.38	C	C1	3	RENT	180
39714	5000	5000	1325.0	36	0.0807	156.84	A	A4	0	MORTGAGE	1000
39715	5000	5000	650.0	36	0.0743	155.38	A	A2	0	MORTGAGE	2000
39716	7500	7500	800.0	36	0.1375	255.43	E	E2	0	OWN	220

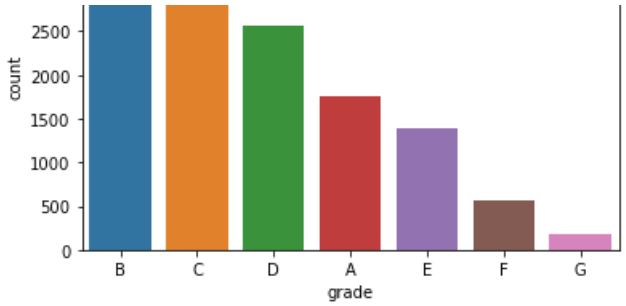
13690 rows × 18 columns

Exploratory Data Analysis

Univariate Analysis

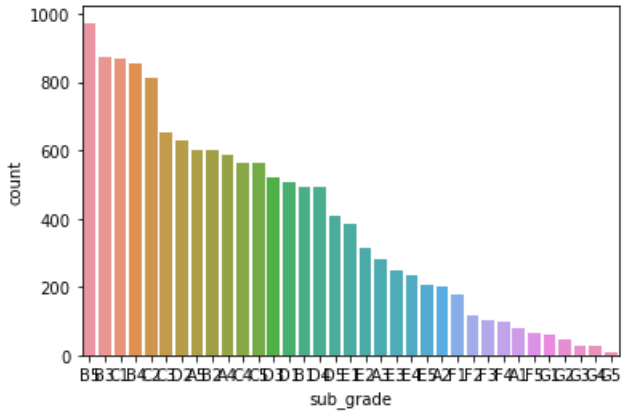
In [23]:

```
sns.countplot(loan['grade'],order = loan['grade'].value_counts().index)
plt.show()
```



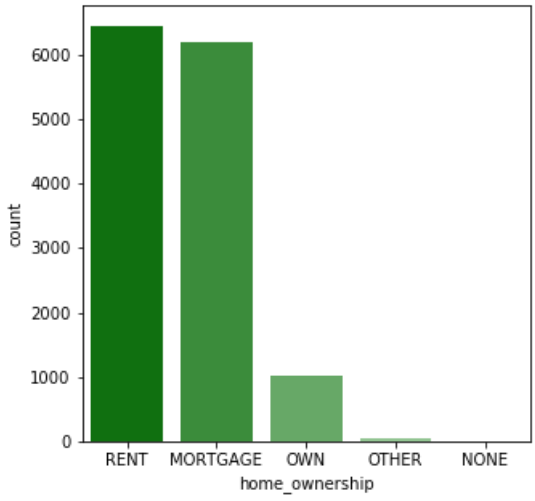
In [24]:

```
sns.countplot(loan['sub_grade'],order = loan['sub_grade'].value_counts().index)
plt.show()
```



In [25]:

```
plt.figure(figsize=(5,5))
sns.countplot(loan['home_ownership'],palette = sns.light_palette("green", reverse= True),order =
loan['home_ownership'].value_counts().index)
plt.show()
```

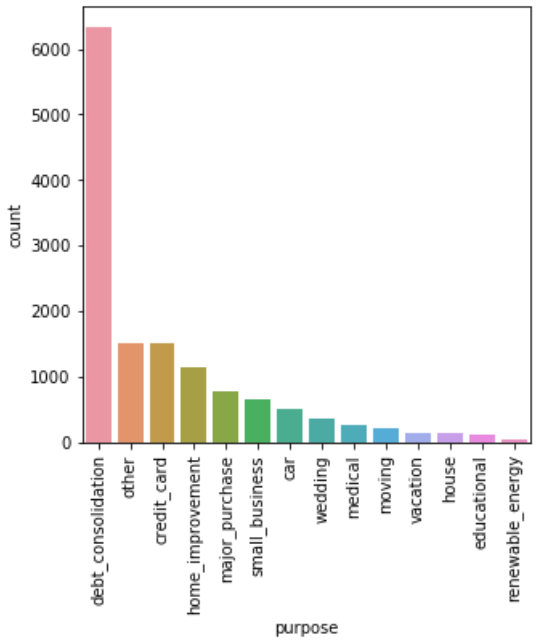


In [26]:

```
plt.figure(figsize=(5,5))
sns.countplot(loan['verification_status'],palette = sns.light_palette("green", reverse=
True),order = loan['verification_status'].value_counts().index)
plt.show()
```

In [27]:

```
plt.figure(figsize=(5,5))
purpose = sns.countplot(loan['purpose'],order = loan['purpose'].value_counts().index)
purpose.tick_params(axis='x', rotation=90)
plt.show()
```



Selected Data Understanding

In [28]:

```
loan['loan_amnt'].describe()
```

Out[28]:

```
count    13690.000000
mean     10623.414901
std       7194.919321
min        500.000000
25%       5000.000000
50%       9000.000000
75%      15000.000000
max      35000.000000
Name: loan_amnt, dtype: float64
```

In [29]:

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

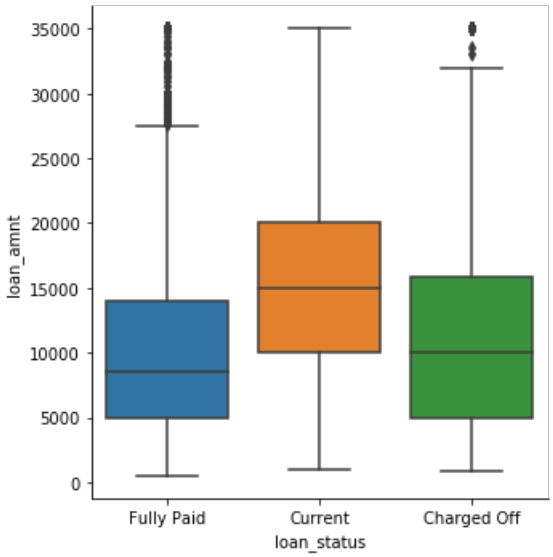
Out[29]:

```
count    13690.000000
```

```
mean     10383.088020
std       6914.353228
min        500.000000
25%       5000.000000
50%       9000.000000
75%      14400.000000
max      35000.000000
Name: funded_amnt, dtype: float64
```

In [30]:

```
sns.catplot(y="loan_amnt", x="loan_status", kind="box", data=loan)
plt.show()
```



In [31]:

```
loan.groupby('loan_status')['loan_amnt'].describe()
```

Out[31]:

loan_status	count	mean	std	min	25%	50%	75%	max
Charged Off	2060.0	11572.330097	7770.780091	900.0	5000.0	10000.0	15881.25	35000.0
Current	352.0	15806.463068	8492.250686	1000.0	10000.0	15000.0	20000.00	35000.0
Fully Paid	11278.0	10288.320181	6960.872469	500.0	5000.0	8500.0	14000.00	35000.0

In [32]:

```
loan['term'].describe()
```

Out[32]:

```
count    13690.000000
mean        42.263842
std         10.540617
min         36.000000
25%         36.000000
50%         36.000000
75%         60.000000
max         60.000000
Name: term, dtype: float64
```

In [33]:

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

Out[33]:

```
36    10117
60     3573
Name: term, dtype: int64
```

In [34]:

```
loan.groupby('term')['loan_status'].value_counts(normalize=True)
```

Out[34]:

```
term  loan_status
36    Fully Paid     0.879905
      Charged Off    0.120095
60    Fully Paid     0.664987
      Charged Off    0.236496
      Current        0.098517
Name: loan_status, dtype: float64
```

In [35]:

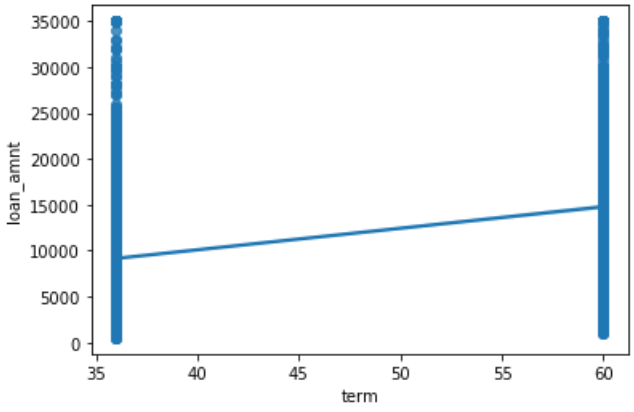
```
#Calculation the percentage for terms only for the charged of loans
print('Charged Off')
print(loan.groupby('term')['loan_status'].value_counts(normalize=True).loc[:,'Charged Off'])

print('Fully Paid')
print(loan.groupby('term')['loan_status'].value_counts(normalize=True).loc[:,'Fully Paid'])
```

```
Charged Off
term
36    0.120095
60    0.236496
Name: loan_status, dtype: float64
Fully Paid
term
36    0.879905
60    0.664987
Name: loan_status, dtype: float64
```

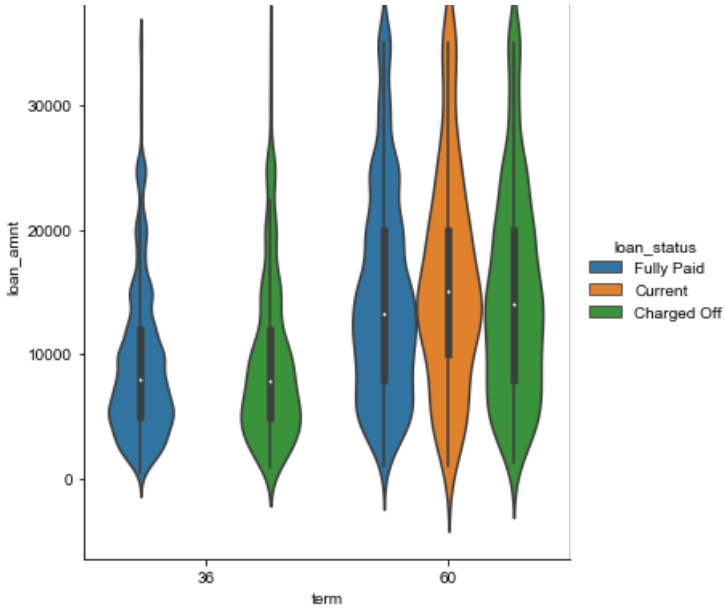
In [36]:

```
sns.regplot(x=loan["term"], y=loan["loan_amnt"])
plt.show()
```



In [37]:

```
sns.catplot(x="term", y="loan_amnt", hue="loan_status",kind="violin", data=loan,height=6, aspect=.9
)
sns.set(rc={'figure.figsize':(11.7,8.27)})
plt.show()
```



In [38]:

```
loan['grade'].unique()
```

Out[38]:

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

In [39]:

```
loan['grade'].value_counts()
```

Out[39]:

```
B    3796
C    3463
D    2555
A    1754
E    1389
F     559
G     174
Name: grade, dtype: int64
```

In [40]:

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

Out[40]:

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

In [41]:

```
loan['sub_grade'].value_counts()
```

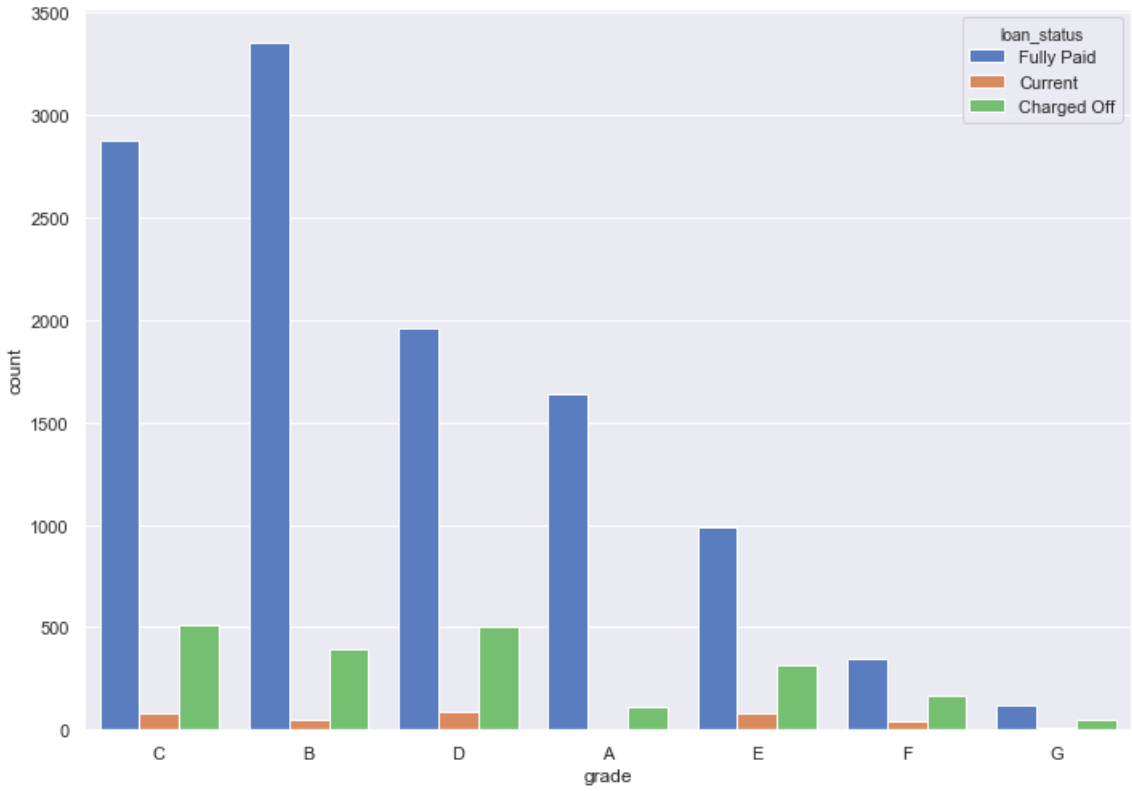
Out[41]:

```
B5    973
B3    874
C1    870
B4    853
C2    813
C3    653
D2    629
A5    603
B2    602
```

```
B2     602
A4     587
C4     564
C5     563
D3     522
D1     506
B1     494
D4     491
D5     407
E1     386
E2     315
A3     282
E3     248
E4     236
E5     204
A2     202
F1     179
F2     117
F3     102
F4      96
A1      80
F5      65
G1      59
G2      45
G3      30
G4      29
G5      11
Name: sub_grade, dtype: int64
```

In [42]:

```
sns.countplot(x='grade',hue="loan_status", data=loan,palette="muted")
sns.set(rc={'figure.figsize':(11.7,8.27)})
plt.show()
```



In [43]:

```
sns.countplot(x='sub_grade',hue="loan_status", data=loan,palette="muted")
sns.set(rc={'figure.figsize':(11.7,8.27)})
plt.show()
```

In [44]:

```
loan.groupby('grade')['loan_amnt'].describe()
```

Out[44]:

grade	count	mean	std	min	25%	50%	75%	max
A	1754.0	7540.692702	4544.241539	500.0	4500.0	6500.0	10000.0	35000.0
B	3796.0	9233.515543	6194.951440	500.0	4800.0	8000.0	12000.0	35000.0
C	3463.0	9454.822408	6075.623153	500.0	5000.0	8000.0	12375.0	35000.0
D	2555.0	11449.716243	6996.147430	1000.0	6000.0	10000.0	15000.0	35000.0
E	1389.0	15107.847372	8661.591115	1000.0	8000.0	14000.0	20000.0	35000.0
F	559.0	18869.364937	8870.562245	1400.0	12000.0	18800.0	25000.0	35000.0
G	174.0	20855.747126	8088.233802	1600.0	16000.0	21375.0	25000.0	35000.0

In [45]:

```
loan.groupby('grade')['int_rate'].describe()
```

Out[45]:

grade	count	mean	std	min	25%	50%	75%	max
A	1754.0	0.075754	0.009329	0.0542	0.0699	0.0751	0.0800	0.0963
B	3796.0	0.110140	0.009305	0.0600	0.1037	0.1099	0.1171	0.1269
C	3463.0	0.134674	0.010173	0.0600	0.1299	0.1349	0.1399	0.1611
D	2555.0	0.155980	0.012851	0.0600	0.1484	0.1562	0.1632	0.1825
E	1389.0	0.176733	0.013989	0.0600	0.1669	0.1756	0.1879	0.2099
F	559.0	0.196728	0.015269	0.1501	0.1854	0.1941	0.2089	0.2294
G	174.0	0.213143	0.013383	0.1734	0.2016	0.2090	0.2248	0.2411

In [46]:

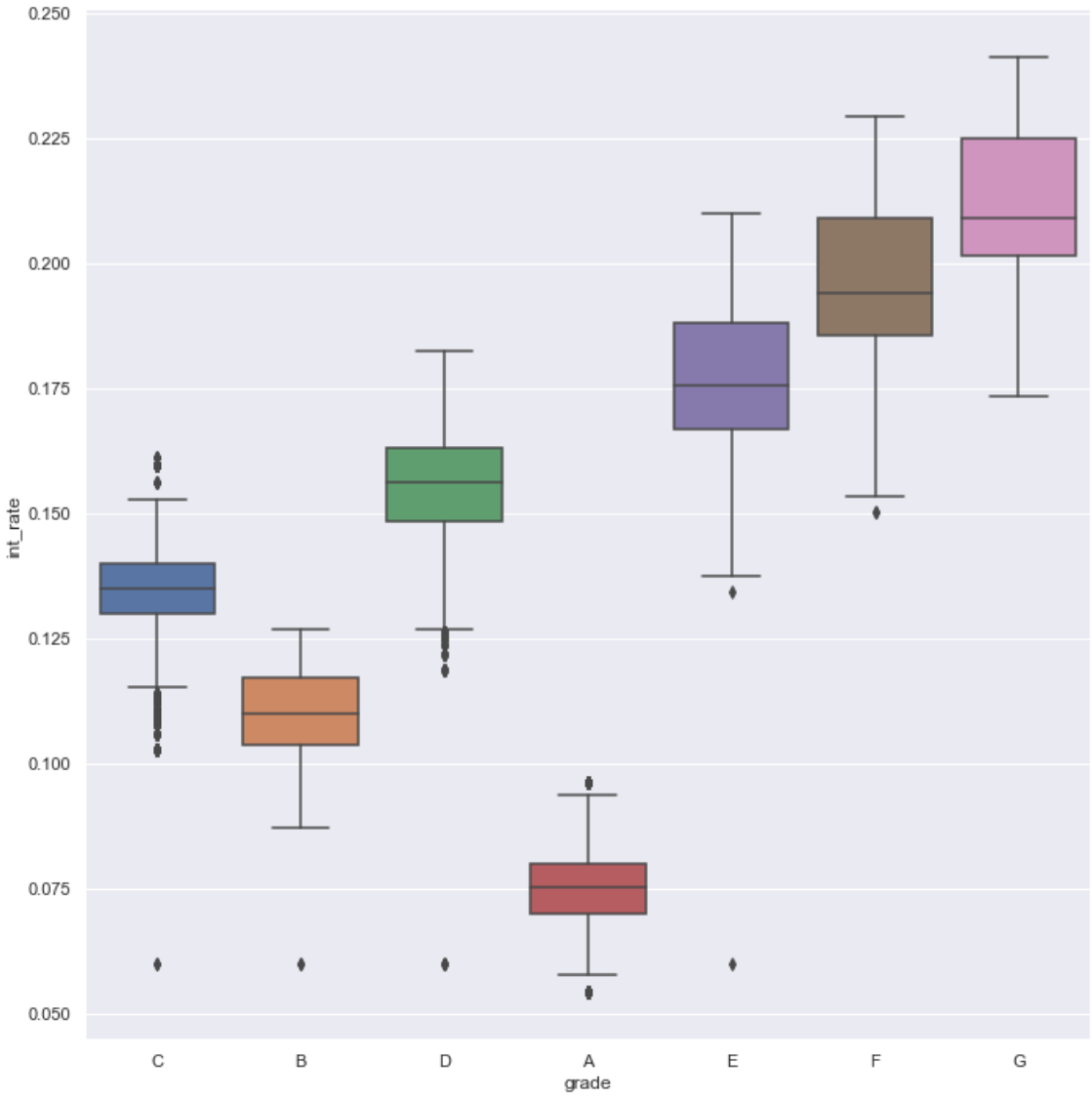
```
loan.groupby('grade')['int_rate'].describe()
```

Out[46]:

grade	count	mean	std	min	25%	50%	75%	max
A	1754.0	0.075754	0.009329	0.0542	0.0699	0.0751	0.0800	0.0963
B	3796.0	0.110140	0.009305	0.0600	0.1037	0.1099	0.1171	0.1269
C	3463.0	0.134674	0.010173	0.0600	0.1299	0.1349	0.1399	0.1611
D	2555.0	0.155980	0.012851	0.0600	0.1484	0.1562	0.1632	0.1825
E	1389.0	0.176733	0.013989	0.0600	0.1669	0.1756	0.1879	0.2099
F	559.0	0.196728	0.015269	0.1501	0.1854	0.1941	0.2089	0.2294
G	174.0	0.213143	0.013383	0.1734	0.2016	0.2090	0.2248	0.2411

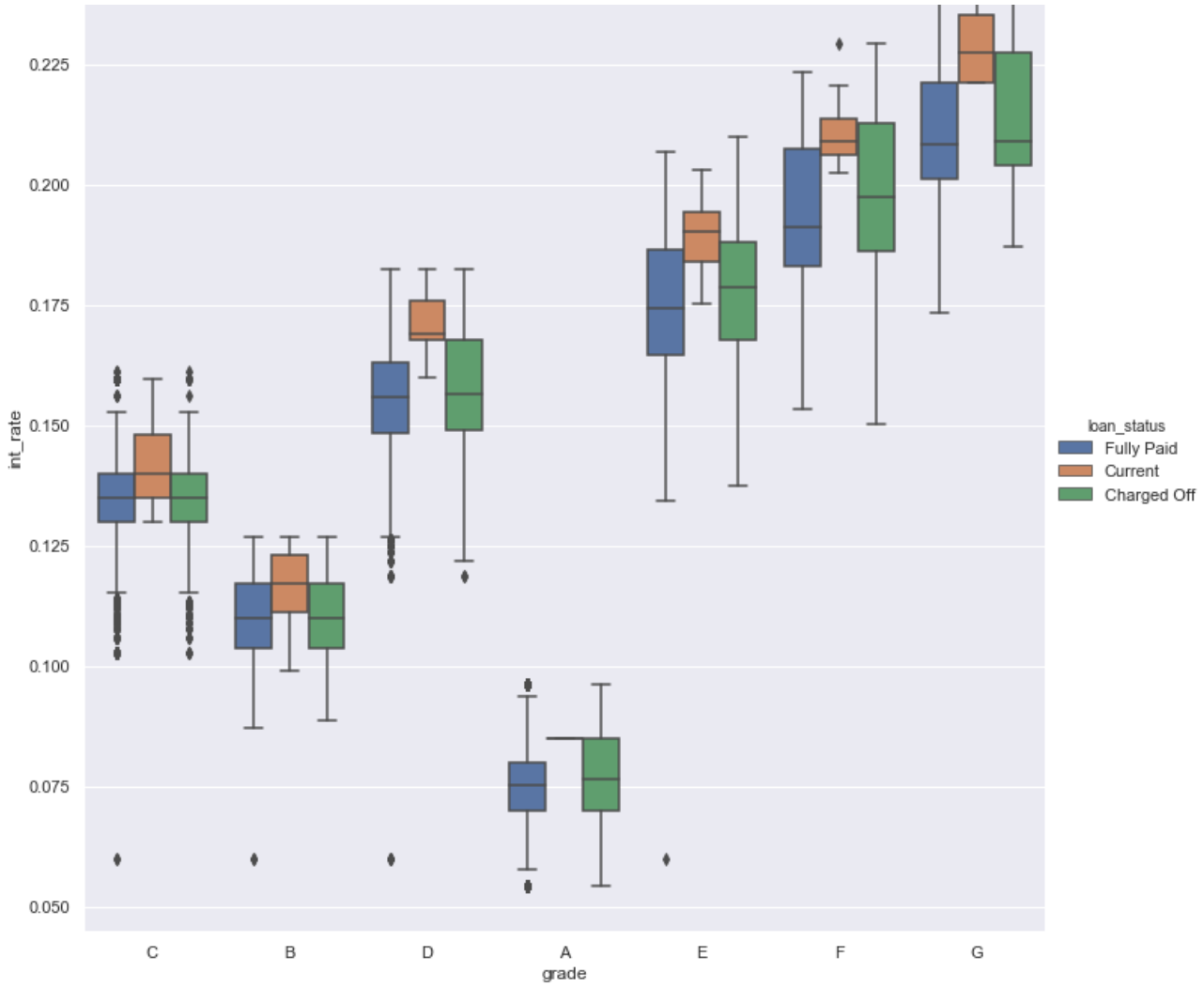
In [47]:

```
sns.catplot(x="grade", y="int_rate", kind="box", data=loan,height=10, aspect=1)
plt.show()
```



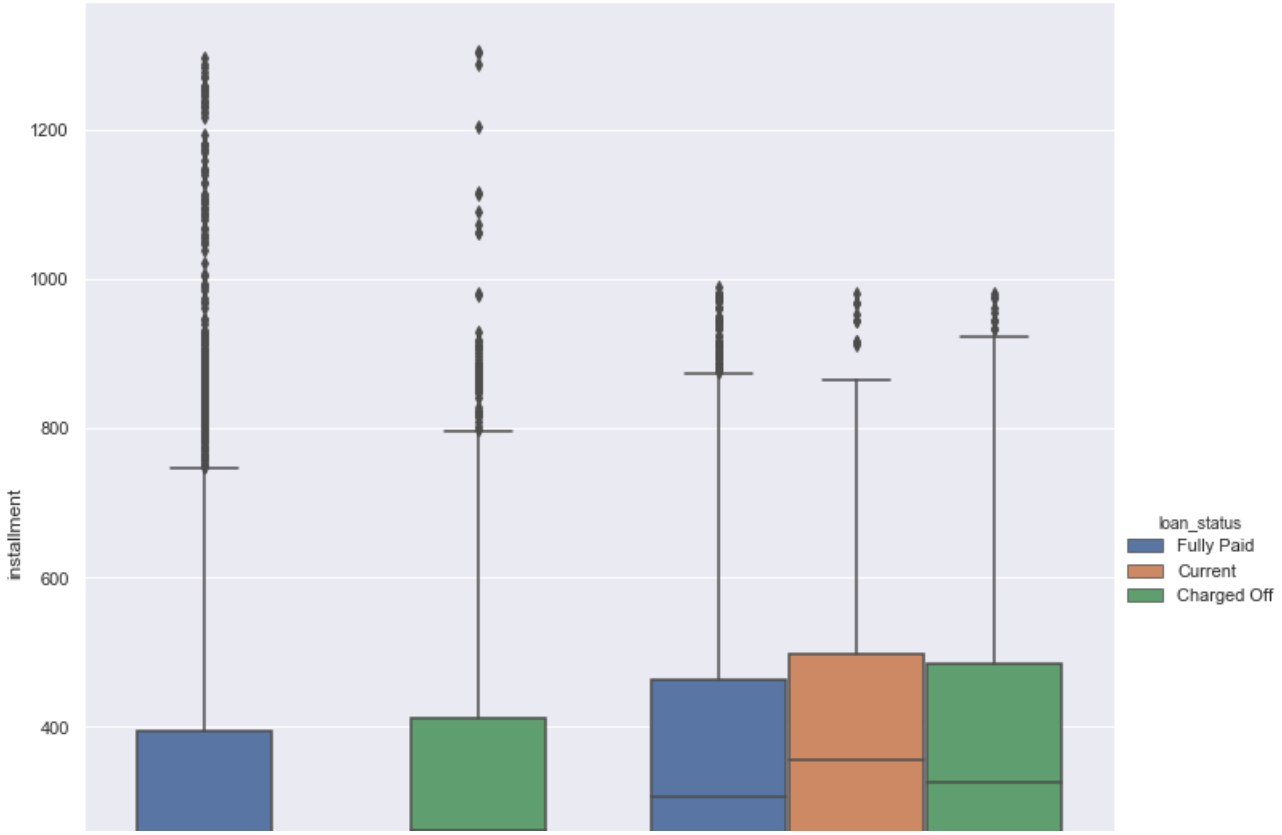
In [48]:

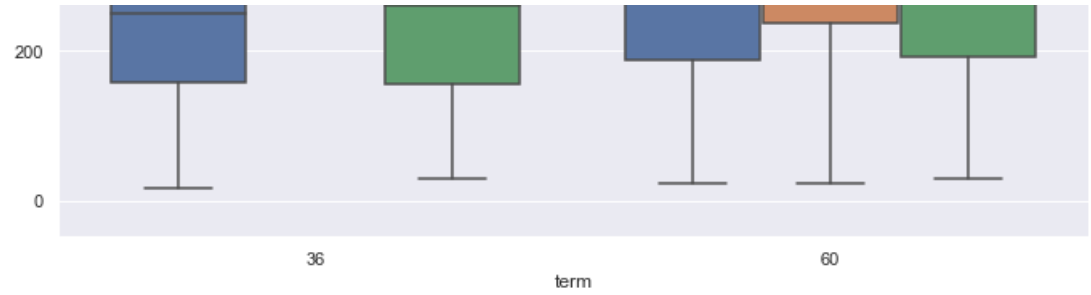
```
sns.catplot(x="grade", y="int_rate",hue="loan_status", kind="box", data=loan,height=10, aspect=1)
plt.show()
```



In [52]:

```
sns.catplot(x="term", y="installment",hue="loan_status", kind="box", data=loan,height=10, aspect=1)
plt.show()
```





In [53]:

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

Out[53]:

```
array([10,  1,  4,  2,  6,  5,  3,  0,  7,  8,  9])
```

In [54]:

```
loan['emp_length'].value_counts()
```

Out[54]:

```
10    3336
0     1513
2     1499
3     1431
4     1229
5     1128
1     1122
6      783
7      664
8      559
9      426
Name: emp_length, dtype: int64
```

In [55]:

```
loan['emp_length'].describe()
```

Out[55]:

```
count    13690.000000
mean         5.111833
std          3.572770
min          0.000000
25%          2.000000
50%          5.000000
75%          9.000000
max         10.000000
Name: emp_length, dtype: float64
```

In [56]:

```
loanChargedOff=loan[loan['loan_status']=='Charged Off']
```

In [57]:

```
loanPaid=loan[loan['loan_status']=='Fully Paid']
```

In [58]:

```
loanPaid['emp_length'].value_counts()
```

Out[58]:

```
10    2663
```

```
2      1264
0      1251
3      1192
4      1019
5       931
1       926
6       671
7       540
8       465
9       356
Name: emp_length, dtype: int64
```

In [59]:

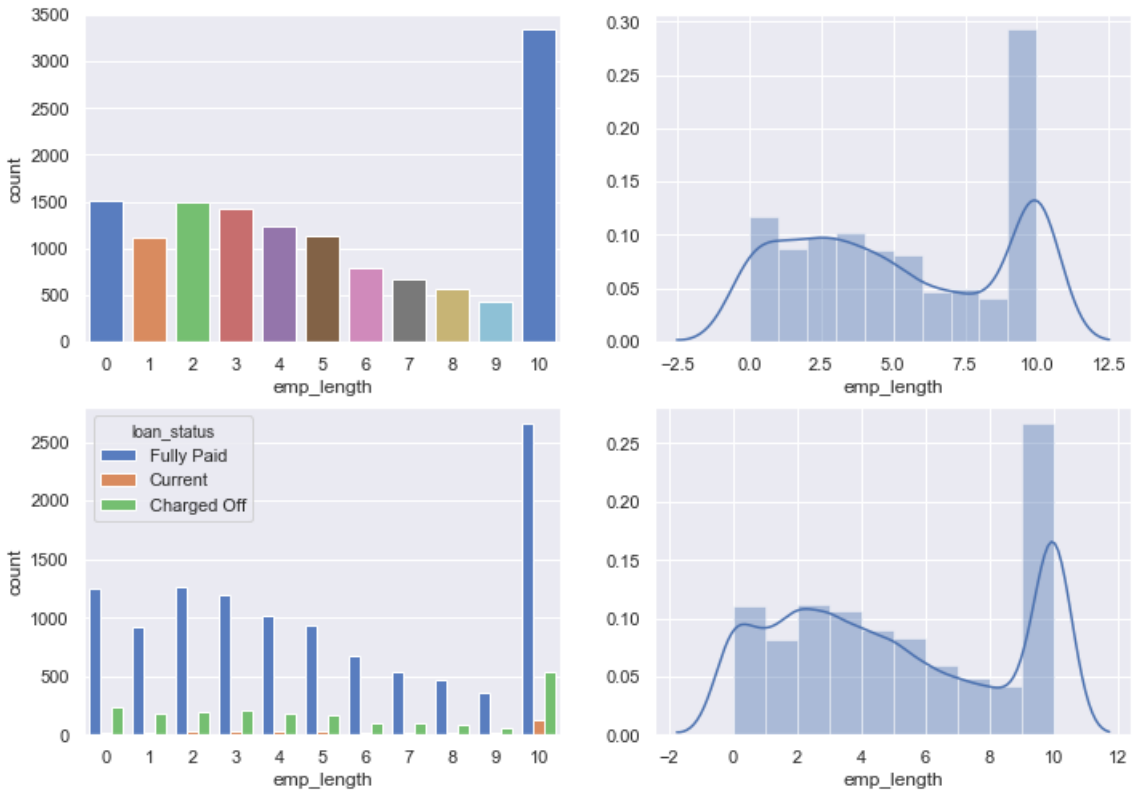
```
loanChargedOff['emp_length'].value_counts()
```

Out[59]:

```
10     543
0      241
3      212
2      201
1      180
4      175
5      168
7      103
6       95
8       83
9       59
Name: emp_length, dtype: int64
```

In [60]:

```
f, axes = plt.subplots(2, 2)
sns.countplot(x='emp_length', data=loan,palette="muted",ax=axes[0,0])
sns.distplot(loanChargedOff['emp_length'],bins=10,ax=axes[0,1])
sns.countplot(x='emp_length',hue='loan_status', data=loan,palette="muted",ax=axes[1,0])
sns.distplot(loanPaid['emp_length'],bins=10,ax=axes[1,1])
plt.show()
```



In [61]:

```
loan.groupby('emp_length')['loan_status'].value_counts(normalize=True)
```

Out[61]:

```
emp_length  loan_status
0           Fully Paid     0.826834
            Charged Off    0.159286
            Current        0.013880
1           Fully Paid     0.825312
            Charged Off    0.160428
            Current        0.014260
2           Fully Paid     0.843229
            Charged Off    0.134089
            Current        0.022682
3           Fully Paid     0.832984
            Charged Off    0.148148
            Current        0.018868
4           Fully Paid     0.829129
            Charged Off    0.142392
            Current        0.028478
5           Fully Paid     0.825355
            Charged Off    0.148936
            Current        0.025709
6           Fully Paid     0.856960
            Charged Off    0.121328
            Current        0.021711
7           Fully Paid     0.813253
            Charged Off    0.155120
            Current        0.031627
8           Fully Paid     0.831843
            Charged Off    0.148479
            Current        0.019678
9           Fully Paid     0.835681
            Charged Off    0.138498
            Current        0.025822
10          Fully Paid     0.798261
            Charged Off    0.162770
            Current        0.038969
Name: loan_status, dtype: float64
```

In [62]:

```
loan['verification_status'].unique()
```

Out[62]:

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

In [63]:

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

Out[63]:

```
Not Verified       6022
Verified           4202
Source Verified    3466
Name: verification_status, dtype: int64
```

In [64]:

```
loan.groupby('verification_status')['loan_amnt'].describe()
```

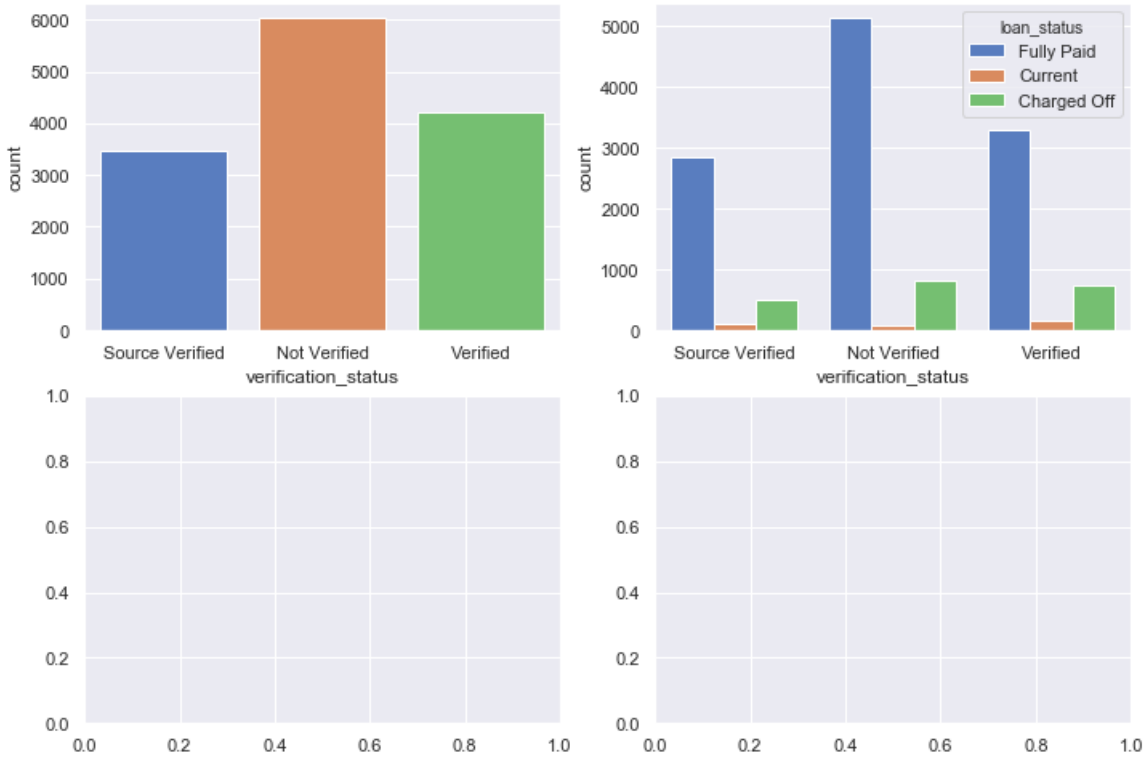
Out[64]:

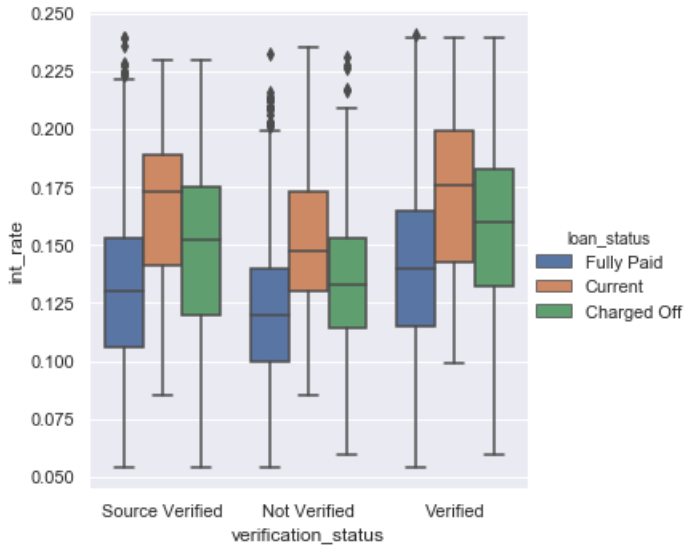
	count	mean	std	min	25%	50%	75%	max
verification_status								
Not Verified	6022.0	8274.680339	5111.011012	500.0	4762.5	7200.0	11000.00	35000.0
Source Verified	3466.0	9576.334391	6446.993370	1000.0	5000.0	8000.0	12093.75	35000.0
Verified	4202.0	14853.129462	8381.787523	1000.0	8000.0	14000.0	20000.00	35000.0

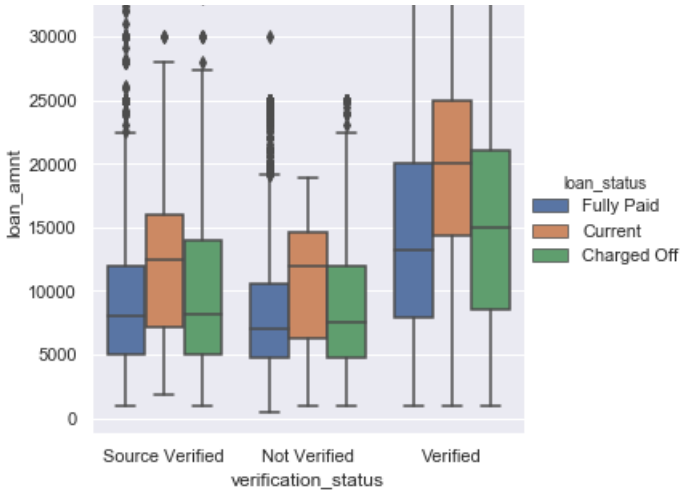
In [65]:

```
f, axes = plt.subplots(2, 2)
sns.countplot(x='verification_status', data=loan,palette="muted",ax=axes[0,0])
sns.countplot(x='verification_status',hue='loan_status', data=loan,palette="muted",ax=axes[0,1])
sns.catplot(x="verification_status", y="int_rate",hue="loan_status", kind="box", data=loan,ax=axes[
1,0])
sns.catplot(x="verification_status", y="loan_amnt",hue="loan_status", kind="box", data=loan,ax=axes
[1,1])
plt.show()
```

```
c:\users\hari\appdata\local\programs\python\python38\lib\site-
packages\seaborn\categorical.py:3720: UserWarning: catplot is a figure-level function and does not
accept target axes. You may wish to try boxplot
  warnings.warn(msg, UserWarning)
c:\users\hari\appdata\local\programs\python\python38\lib\site-
packages\seaborn\categorical.py:3720: UserWarning: catplot is a figure-level function and does not
accept target axes. You may wish to try boxplot
  warnings.warn(msg, UserWarning)
```







In [66]:

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

Out[66]:

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

In [67]:

```
loan['purpose'].value_counts().sort_values()
```

Out[67]:

```
renewable_energy         35
educational             127
house                   136
vacation                142
moving                  212
medical                 256
wedding                 358
car                     504
small_business          665
major_purchase          772
home_improvement       1133
credit_card            1505
other                  1511
debt_consolidation     6334
Name: purpose, dtype: int64
```

In [68]:

```
loan.groupby('purpose')['loan_amnt'].median().sort_values()
```

Out[68]:

```
purpose
vacation                4000
moving                  4800
educational             5000
renewable_energy        5550
car                     5775
major_purchase          6000
other                   6000
medical                 6275
wedding                 8000
home_improvement        8975
credit_card             9800
debt_consolidation     10000
house                  10000
small_business         11200
```

```
Name: loan_amnt, dtype: int64
```

In [69]:

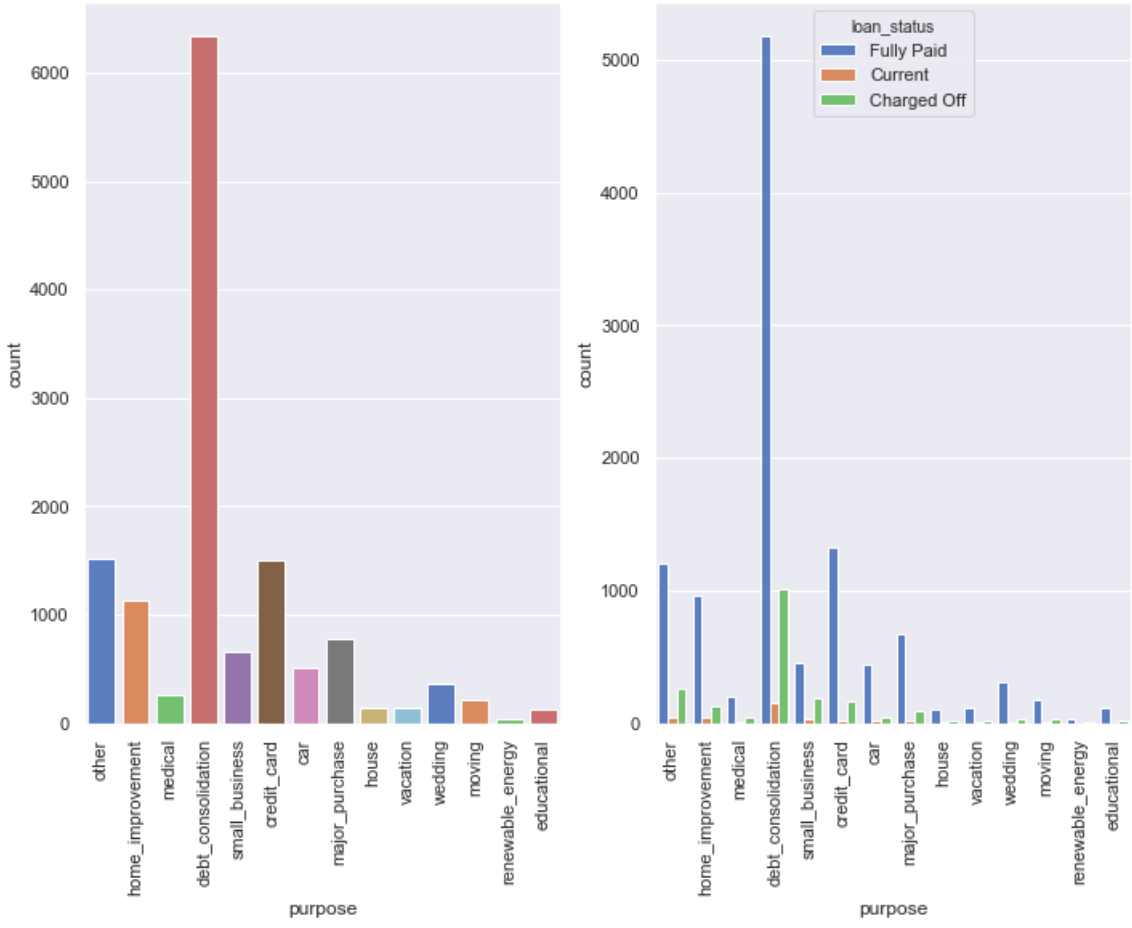
```
loan.groupby('purpose')['loan_status'].value_counts(normalize=True).loc[:,'Charged
Off'].sort_values()
```

Out[69]:

```
purpose
wedding                0.094972
car                    0.095238
credit_card            0.104983
major_purchase         0.111399
home_improvement       0.117387
educational            0.118110
moving                 0.136792
renewable_energy       0.142857
debt_consolidation     0.159141
house                  0.161765
vacation               0.169014
medical                0.175781
other                  0.176042
small_business         0.281203
Name: loan_status, dtype: float64
```

In [70]:

```
f, axes = plt.subplots(1, 2)
g=sns.countplot(x='purpose', data=loan,palette="muted",ax=axes[0])
g.tick_params(axis='x',rotation=90)
s=sns.countplot(x='purpose',hue='loan_status', data=loan,palette="muted",ax=axes[1])
s.tick_params(axis='x',rotation=90)
plt.show()
```



In [71]:

```
loan['dti'].describe()
```

Out[71]:

```
count    13690.000000
mean        12.862677
std          6.482152
min          0.000000
25%          7.882500
50%         12.935000
75%         17.960000
max         29.990000
Name: dti, dtype: float64
```

In [72]:

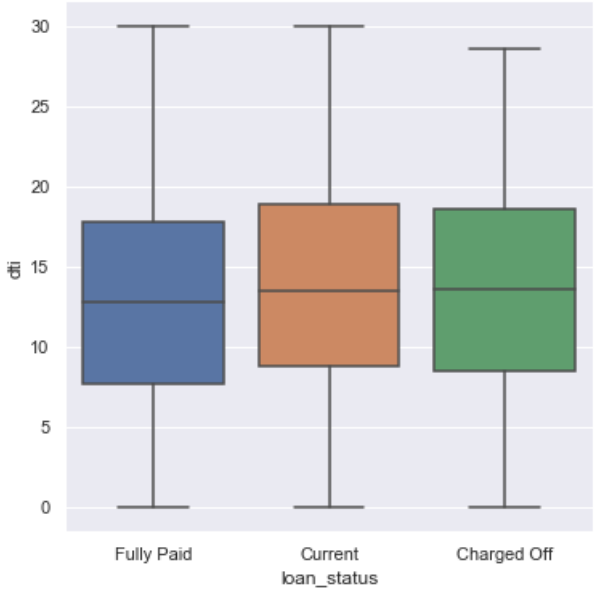
```
loan.groupby('loan_status')['dti'].describe()
```

Out[72]:

loan_status	count	mean	std	min	25%	50%	75%	max
Charged Off	2060.0	13.440631	6.508464	0.0	8.560	13.64	18.6050	28.58
Current	352.0	13.673324	6.642116	0.0	8.785	13.52	18.8800	29.95
Fully Paid	11278.0	12.731809	6.465240	0.0	7.760	12.78	17.7875	29.99

In [73]:

```
dti = sns.catplot(x="loan_status", y="dti", kind="box",data=loan,height=10, aspect=1)
dti.fig.set_size_inches(5,5)
plt.show()
```



In [74]:

```
loan['int_rate'].describe()
```

Out[74]:

```
count    13690.000000
mean         0.132097
std          0.035051
min          0.054200
25%          0.107500
50%          0.131600
75%          0.156200
max          0.241100
```

```
Name: int_rate, dtype: float64
```

In [75]:

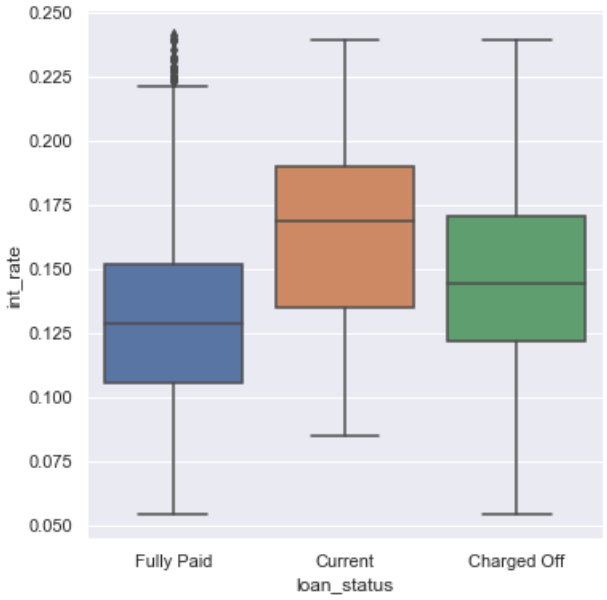
```
loan.groupby('loan_status')['int_rate'].describe()
```

Out[75]:

loan_status	count	mean	std	min	25%	50%	75%	max
Charged Off	2060.0	0.146054	0.034837	0.0542	0.1218	0.1446	0.1706	0.2391
Current	352.0	0.166380	0.032826	0.0849	0.1349	0.1689	0.1903	0.2391
Fully Paid	11278.0	0.128478	0.033924	0.0542	0.1059	0.1285	0.1521	0.2411

In [76]:

```
interest = sns.catplot(x="loan_status", y="int_rate", kind="box",data=loan,height=10, aspect=1)
interest.fig.set_size_inches(5,5)
plt.show()
```



In [ ]:

```
```