In [424]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import seaborn
```

In [540]:
```
df = pd.read_csv('logistic_regression.csv?1651045921')
df.shape
```

Out[540]: (396030, 27)

In [531]:
```
df.head()
```

Out[531]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_acc	pub_rec	revol_
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	...	16.0	0.0	3636
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	...	17.0	0.0	2013
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	...	13.0	0.0	1198
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	...	6.0	0.0	547
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	...	13.0	0.0	2458

5 rows × 27 columns

Define Problem Statement and perform Exploratory Data Analysis

LoanTap is an online platform committed to delivering customized loan products to millennials. They innovate in an otherwise dull loan segment, to deliver instant, flexible loans on consumer friendly terms to salaried professionals and businessmen.

The data science team at LoanTap is building an underwriting layer to determine the creditworthiness of MSMEs as well as individuals.

In [144]:
```
df.shape
```

Out[144]: (396030, 27)

In [147]:
```
df.dtypes
```

Out[147]:
```
loan_amnt              float64
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               float64
pub_rec                float64
revol_bal              float64
revol_util             float64
total_acc              float64
initial_list_status     object
application_type        object
mort_acc               float64
pub_rec_bankruptcies   float64
address                 object
dtype: object
```

In [148]:
```
df.isnull().sum()
```

Out[148]:
```
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                  1755
dti                       0
earliest_cr_line          0
open_acc                  0
pub_rec                   0
revol_bal                 0
revol_util              276
total_acc                 0
initial_list_status       0
application_type          0
mort_acc              37795
pub_rec_bankruptcies    535
address                   0
dtype: int64
```

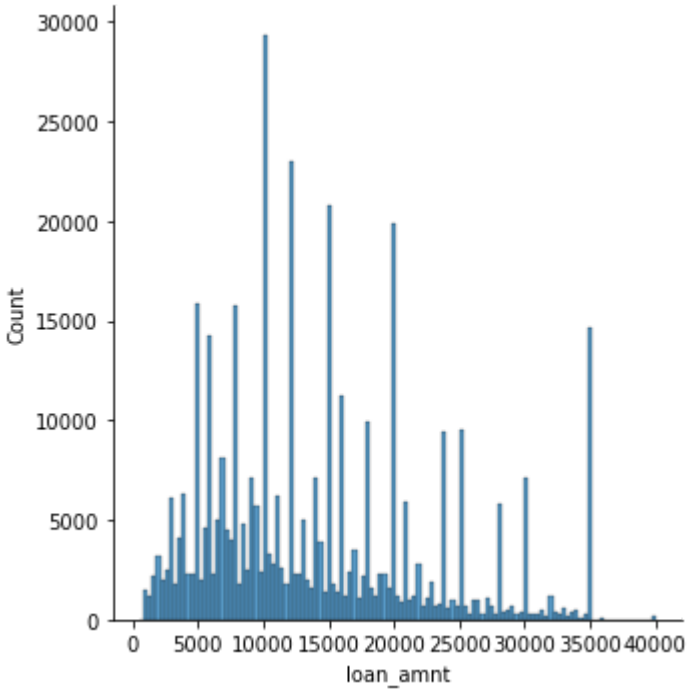
In [149]:
```
df.describe()
```

Out[149]:

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util	
count	396030.000000	396030.000000	396030.000000	3.960300e+05	396030.000000	396030.000000	396030.000000	3.960300e+05	395754.000000	3960
mean	14113.888089	13.639400	431.849698	7.420318e+04	17.379514	11.311153	0.178191	1.584454e+04	53.791749	
std	8357.441341	4.472157	250.727790	6.163762e+04	18.019092	5.137649	0.530671	2.059184e+04	24.452193	
min	500.000000	5.320000	16.080000	0.000000e+00	0.000000	0.000000	0.000000	0.000000e+00	0.000000	
25%	8000.000000	10.490000	250.330000	4.500000e+04	11.280000	8.000000	0.000000	6.025000e+03	35.800000	
50%	12000.000000	13.330000	375.430000	6.400000e+04	16.910000	10.000000	0.000000	1.118100e+04	54.800000	
75%	20000.000000	16.490000	567.300000	9.000000e+04	22.980000	14.000000	0.000000	1.962000e+04	72.900000	
max	40000.000000	30.990000	1533.810000	8.706582e+06	9999.000000	90.000000	86.000000	1.743266e+06	892.300000	1

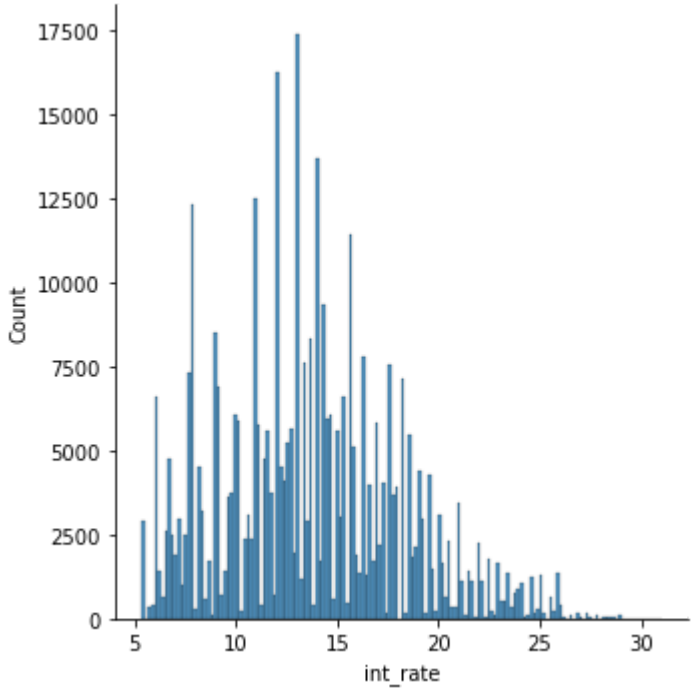
In [160]:
```
seaborn.displot(df['loan_amnt'])
```

Out[160]:
```
<seaborn.axisgrid.FacetGrid at 0x7fd41b45c3a0>
```



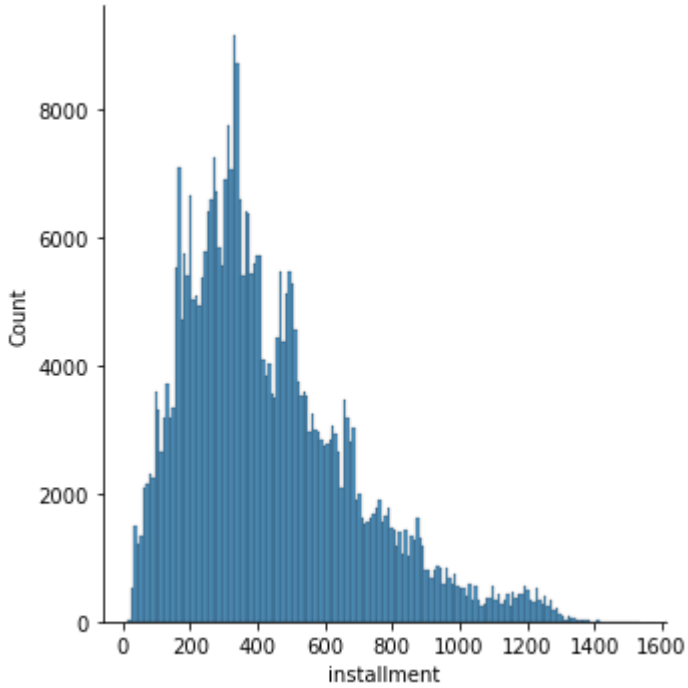
In [161]:
```
seaborn.displot(df['int_rate'])
```

Out[161]:
```
<seaborn.axisgrid.FacetGrid at 0x7fd45faad0a0>
```



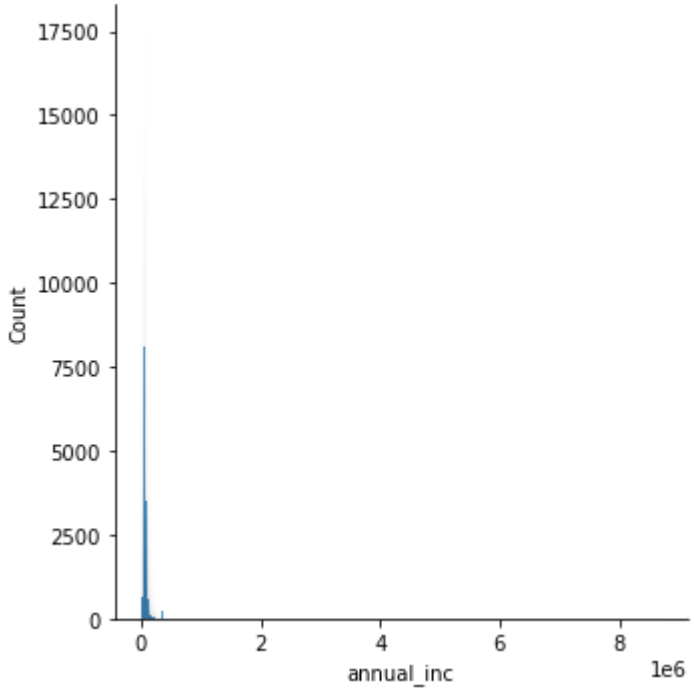
In [162]:
```
seaborn.displot(df['installment'])
```

Out[162]:
```
<seaborn.axisgrid.FacetGrid at 0x7fd420b96370>
```



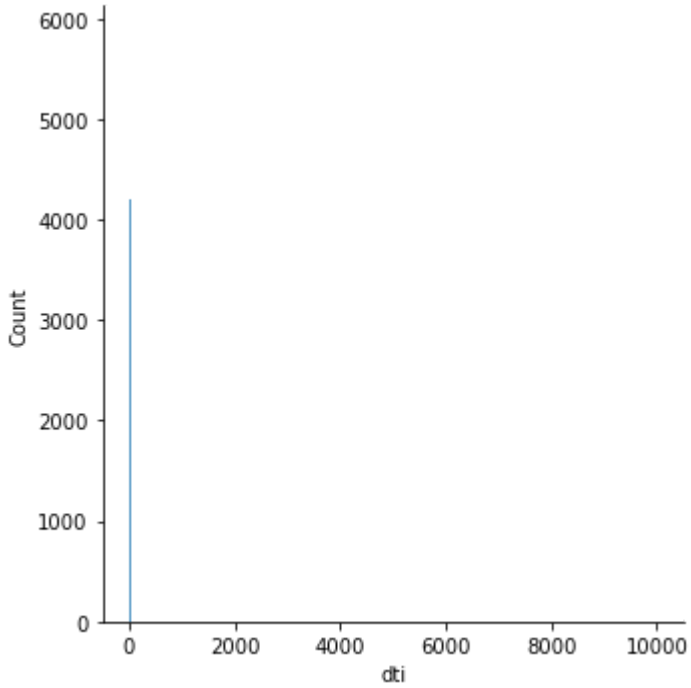
In [187]:
```
seaborn.displot(df['annual_inc'])
```

Out[187]:
```
<seaborn.axisgrid.FacetGrid at 0x7fd40f5a8670>
```



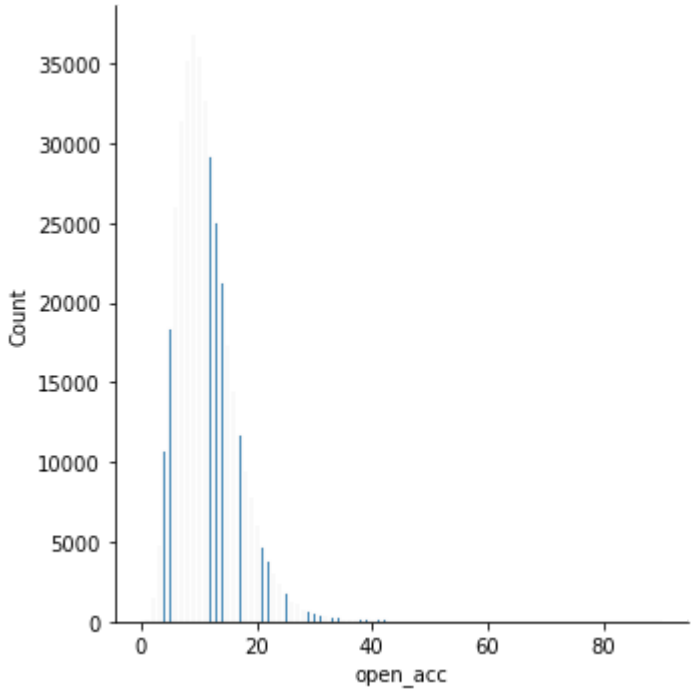
In [188]:
```
seaborn.displot(df['dti'])
```

Out[188]: <seaborn.axisgrid.FacetGrid at 0x7fd413e72df0>



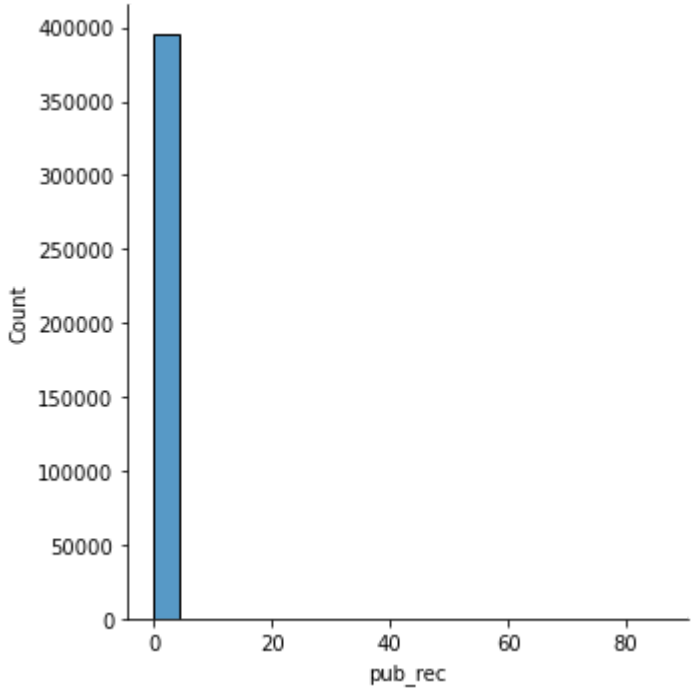
In [189]:
```
seaborn.displot(df['open_acc'])
```

Out[189]:
```
<seaborn.axisgrid.FacetGrid at 0x7fd4171b5490>
```



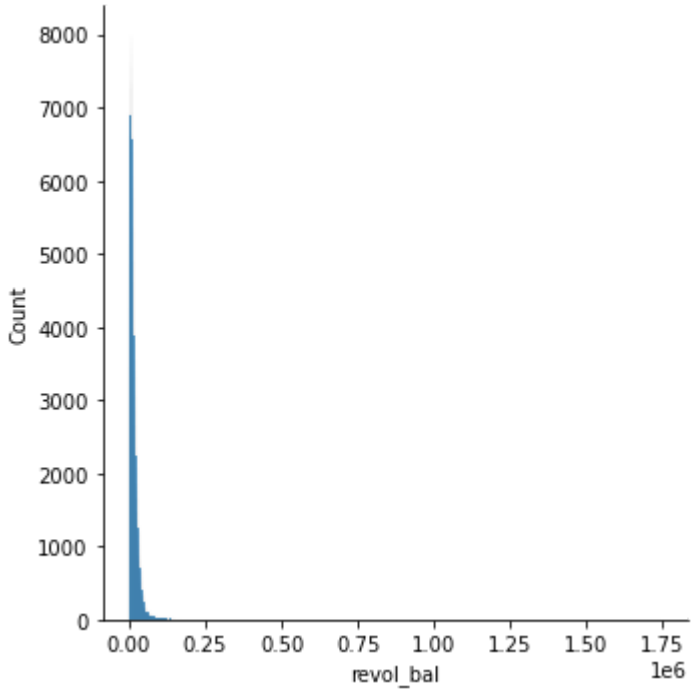
In [190]:
```
seaborn.displot(df['pub_rec'])
```

Out[190]: <seaborn.axisgrid.FacetGrid at 0x7fd41837c280>



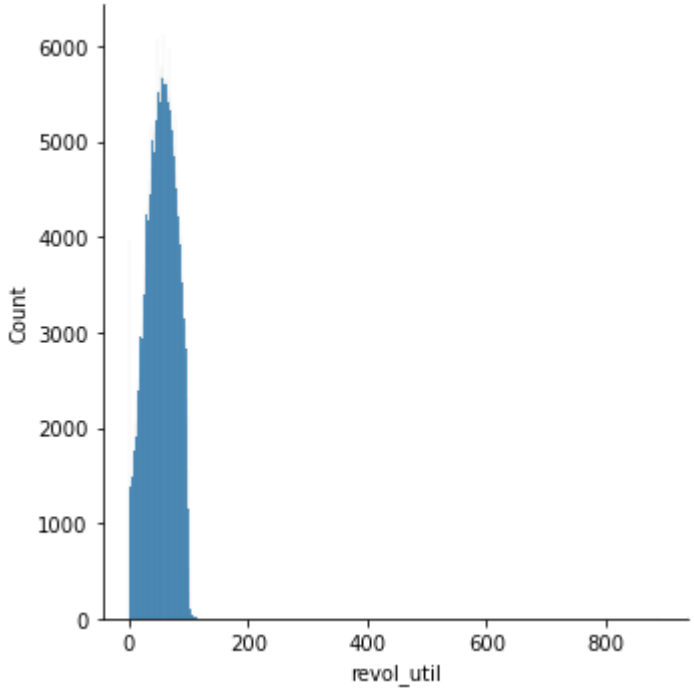
In [191]:
```
seaborn.displot(df['revol_bal'])
```

Out[191]: <seaborn.axisgrid.FacetGrid at 0x7fd4110c6940>



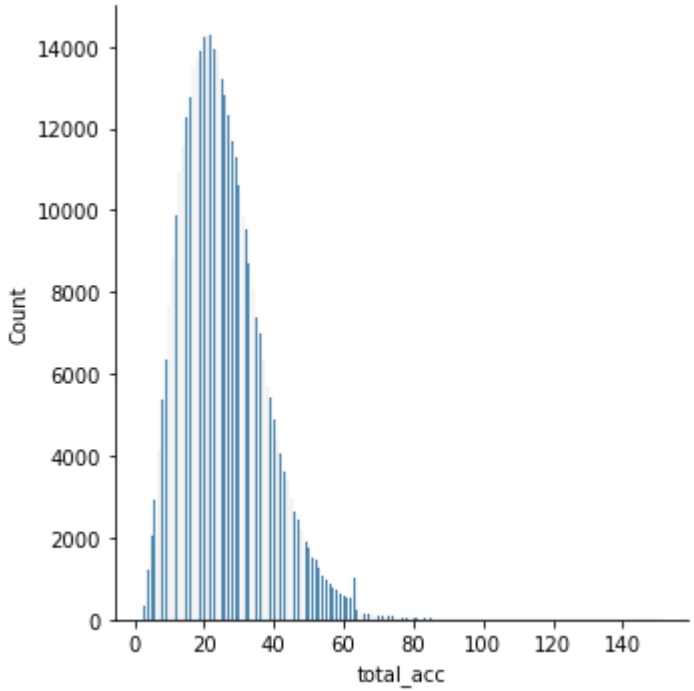
In [192]:
```
seaborn.displot(df['revol_util'])
```

Out[192]: <seaborn.axisgrid.FacetGrid at 0x7fd3f72c6cd0>



In [193]:
```
seaborn.displot(df['total_acc'])
```

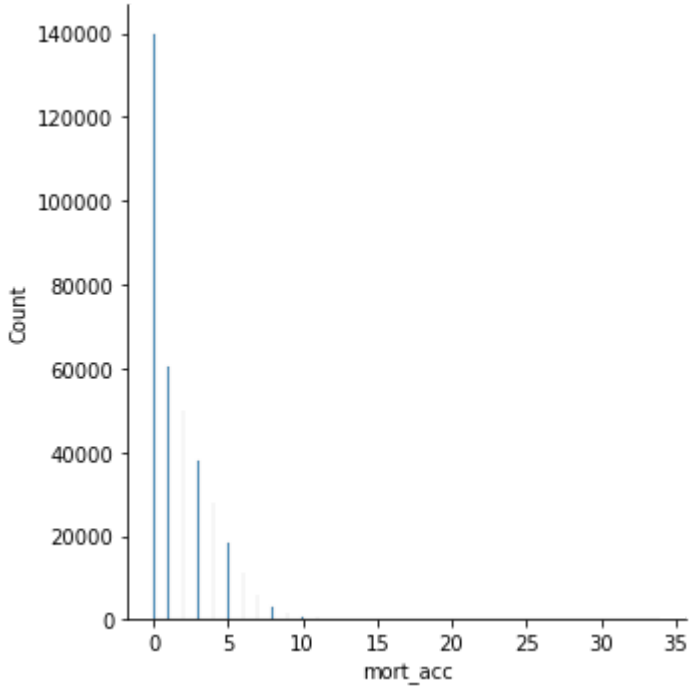
Out[193]: <seaborn.axisgrid.FacetGrid at 0x7fd3f4d360a0>



In [251]:
```
fig = plt.figure(figsize = (15, 10))
seaborn.displot(df['mort_acc'], legend=True)
```

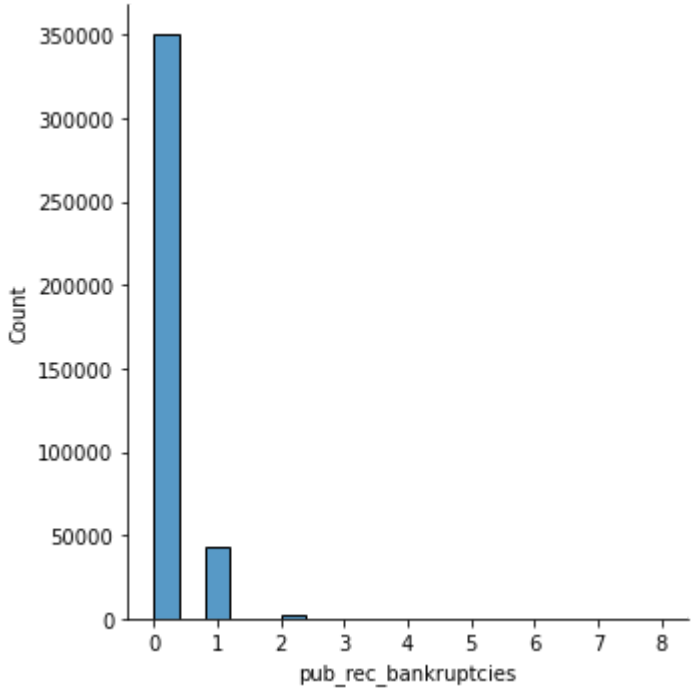
Out[251]:
```
<seaborn.axisgrid.FacetGrid at 0x7fd3e252d9a0>

<Figure size 1080x720 with 0 Axes>
```



In [198]:
```
seaborn.displot(df['pub_rec_bankruptcies'], legend=True)
```

Out[198]:
```
<seaborn.axisgrid.FacetGrid at 0x7fd3dad2c5e0>
```



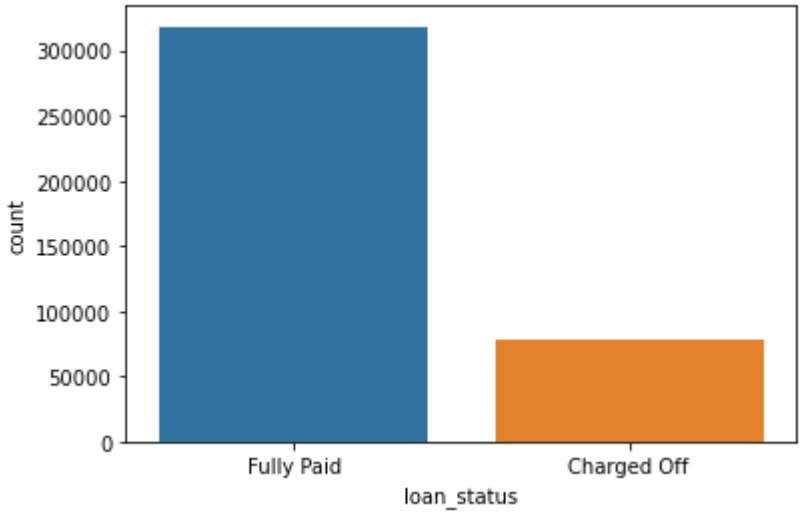
In [203]:
```
df['loan_status'].value_counts()/len(df['loan_status'])
```

Out[203]:
```
Fully Paid     0.803871
Charged Off    0.196129
Name: loan_status, dtype: float64
```

In [201]:
```
seaborn.countplot(df['loan_status'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

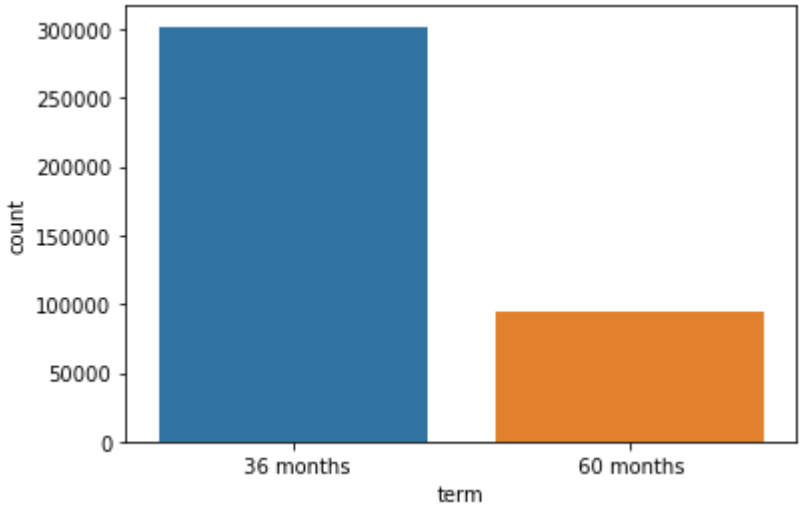
Out[201]: <AxesSubplot:xlabel='loan_status', ylabel='count'>



In [204]:
```
seaborn.countplot(df['term'])
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

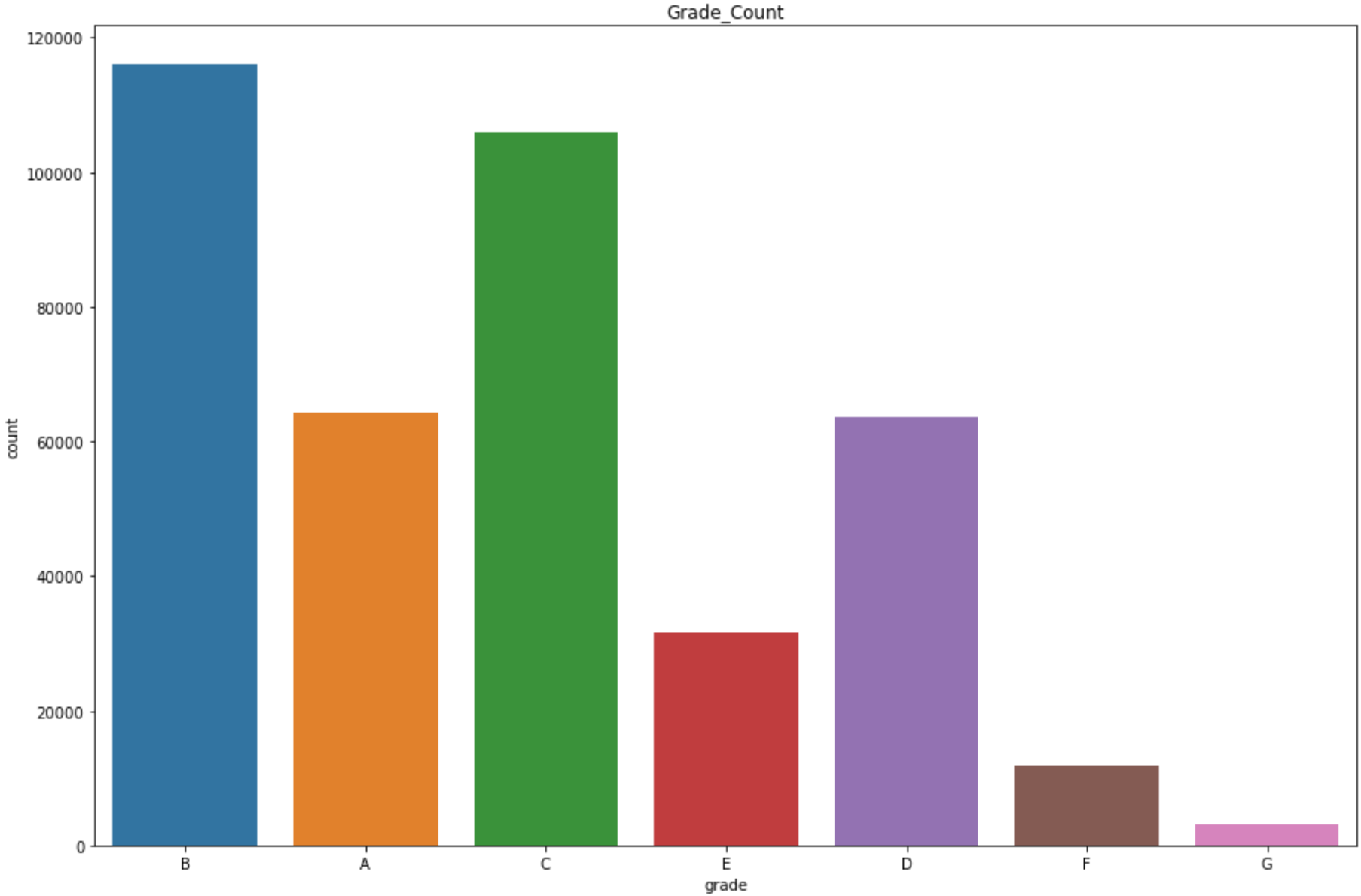
Out[204]:
```
<AxesSubplot:xlabel='term', ylabel='count'>
```



In [254]:
```
fig = plt.figure(figsize = (15, 10))
seaborn.countplot(df['grade'])
plt.title("Grade_Count")
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

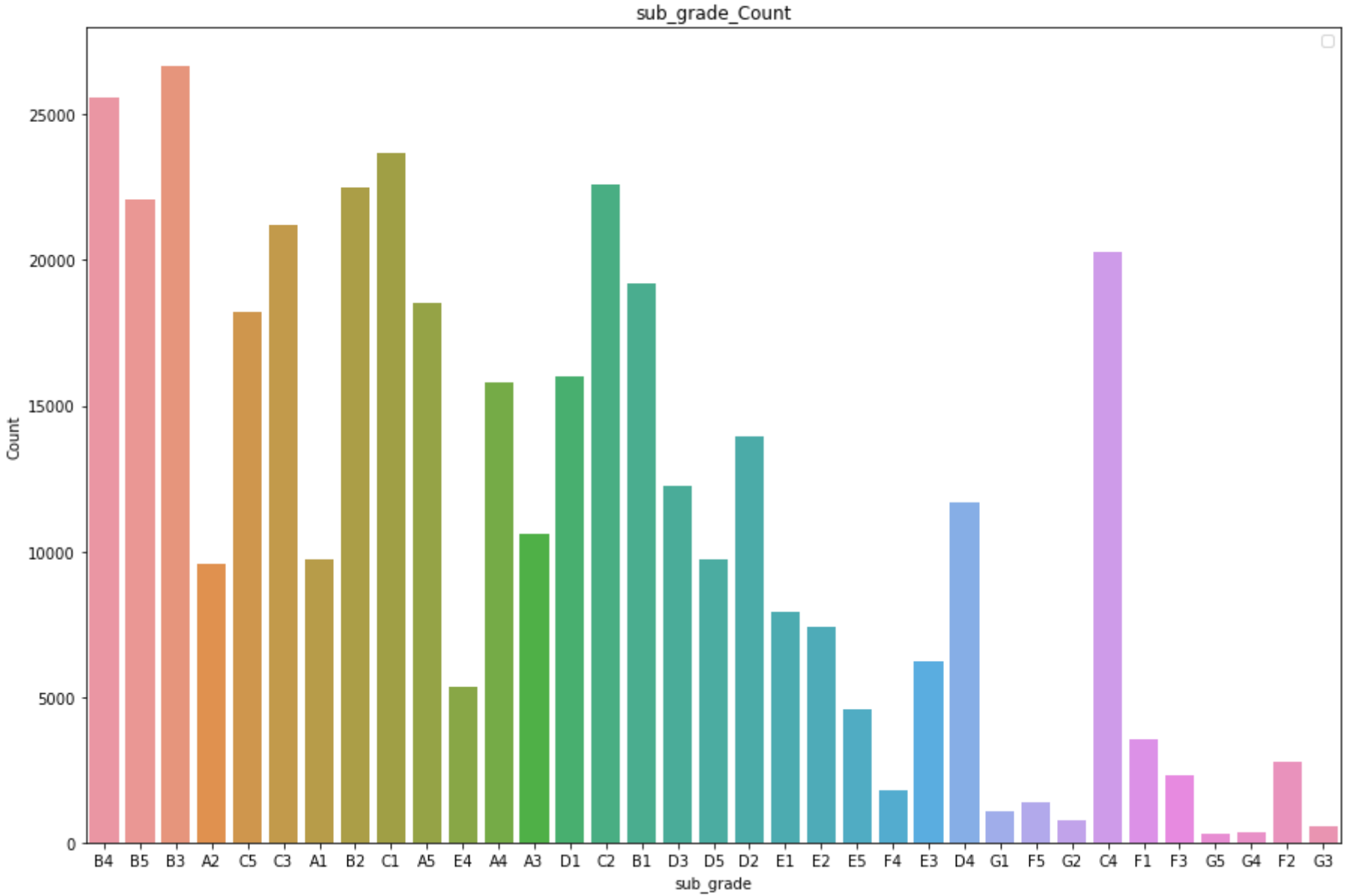
Out[254]:
```
Text(0.5, 1.0, 'Grade_Count')
```


Grade_Count
120000
100000
80000
60000
40000
20000
0
count
B
A
C
E
D
F
G
grade

In [248]:
```
import matplotlib.pyplot as plt

fig = plt.figure(figsize = (15, 10))
seaborn.countplot(df['sub_grade'])
plt.xlabel("sub_grade")
plt.ylabel("Count")
plt.title("sub_grade_Count")
plt.legend()
plt.show()
```

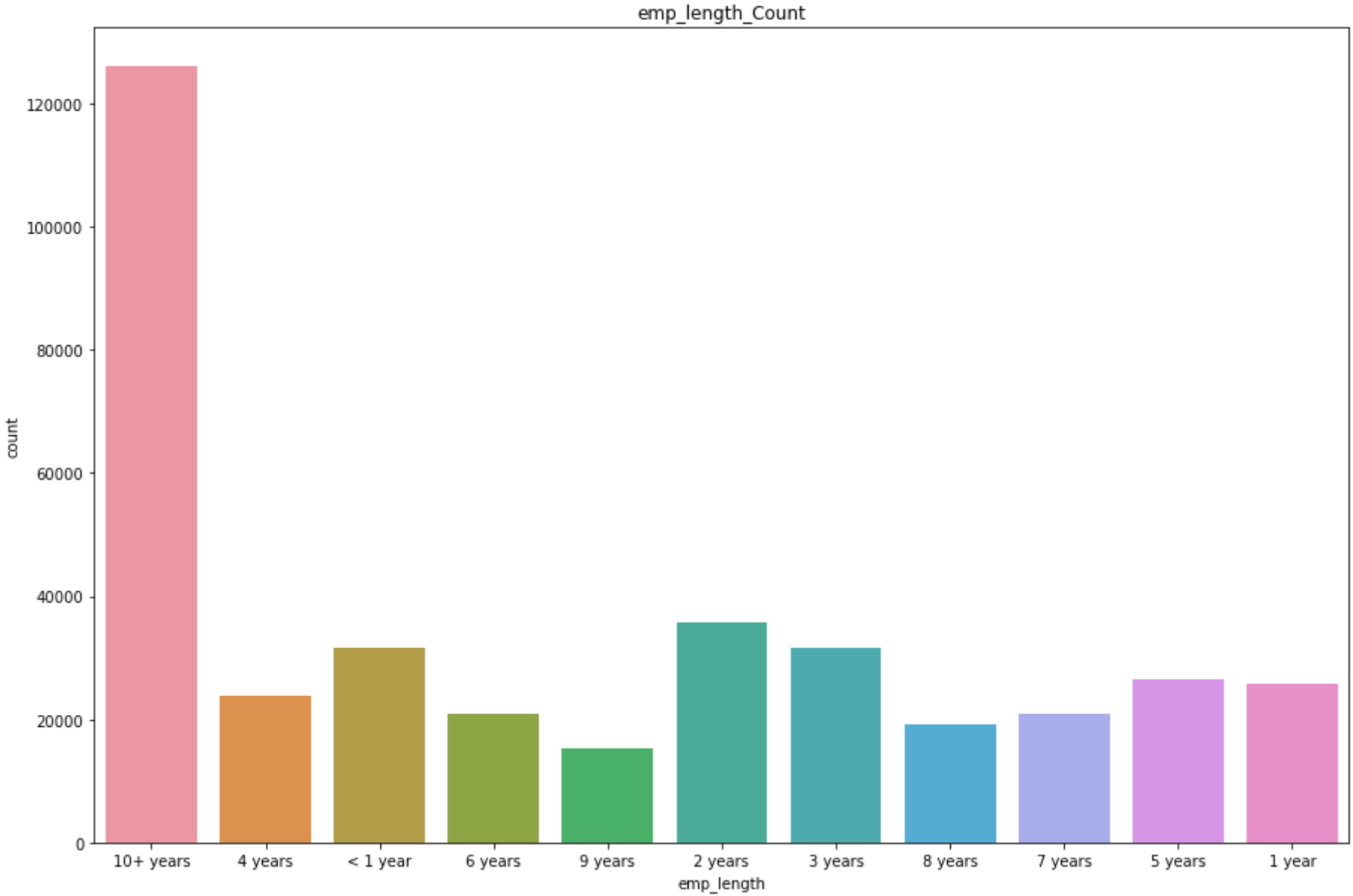
```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ig
nored when legend() is called with no argument.
```


sub_grade_Count
Count
25000
20000
15000
10000
5000
0
B4 B5 B3 A2 C5 C3 A1 B2 C1 A5 E4 A4 A3 D1 C2 B1 D3 D5 D2 E1 E2 E5 F4 E3 D4 G1 F5 G2 C4 F1 F3 G5 G4 F2 G3
sub_grade

In [256]:
```
fig = plt.figure(figsize = (15, 10))
seaborn.countplot(df['emp_length'])
plt.title("emp_length_Count")
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

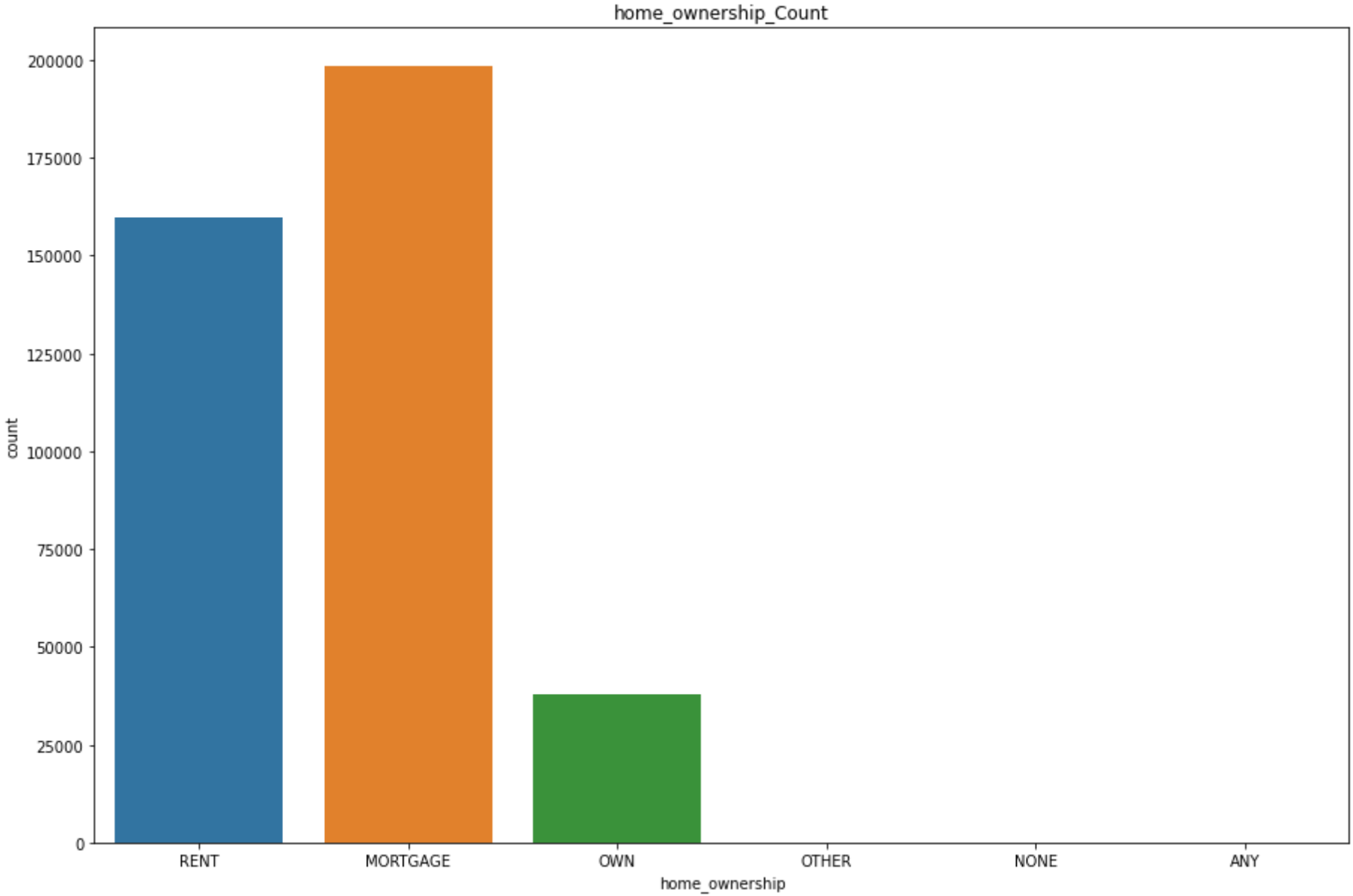
Out[256]:
```
Text(0.5, 1.0, 'emp_length_Count')
```


emp_length_Count
120000
100000
80000
60000
40000
20000
0
count
10+ years
4 years
< 1 year
6 years
9 years
2 years
3 years
8 years
7 years
5 years
1 year
emp_length

In [257]:
```
fig = plt.figure(figsize = (15, 10))
seaborn.countplot(df['home_ownership'])
plt.title("home_ownership_Count")
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

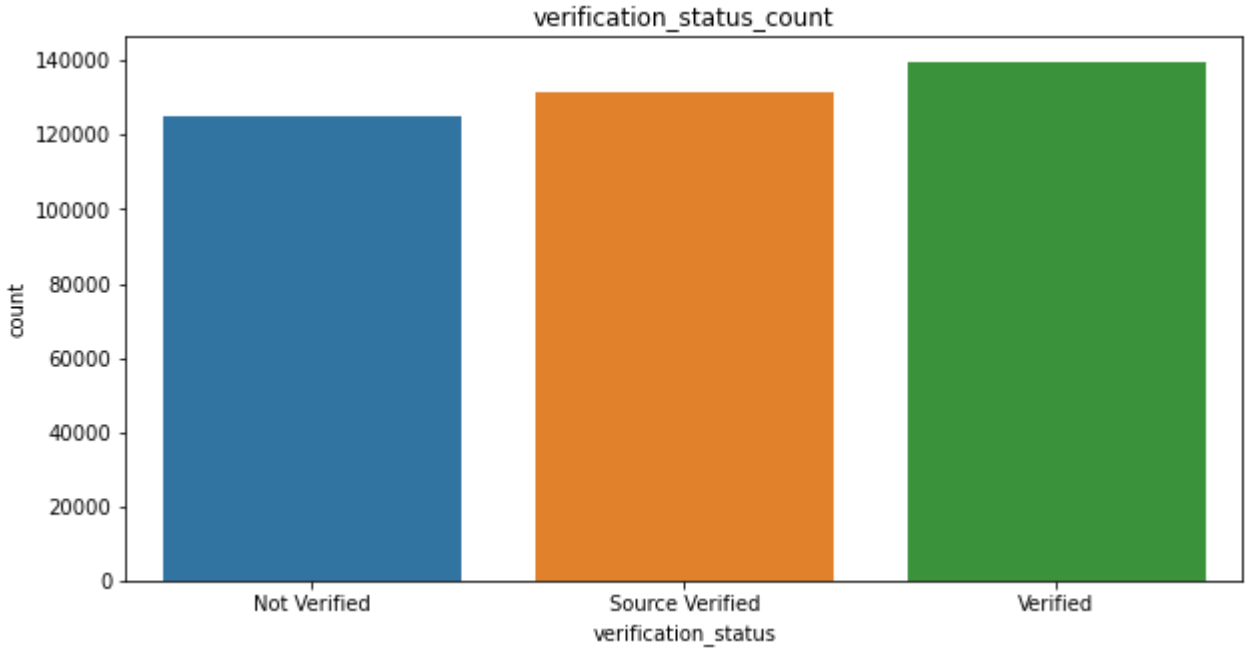
Out[257]:
```
Text(0.5, 1.0, 'home_ownership_Count')
```


home_ownership_Count
200000
175000
150000
125000
100000
75000
50000
25000
0
count
RENT
MORTGAGE
OWN
OTHER
NONE
ANY
home_ownership

In [259]:
```
fig = plt.figure(figsize = (10, 5))
seaborn.countplot(df['verification_status'])
plt.title("verification_status_count")
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

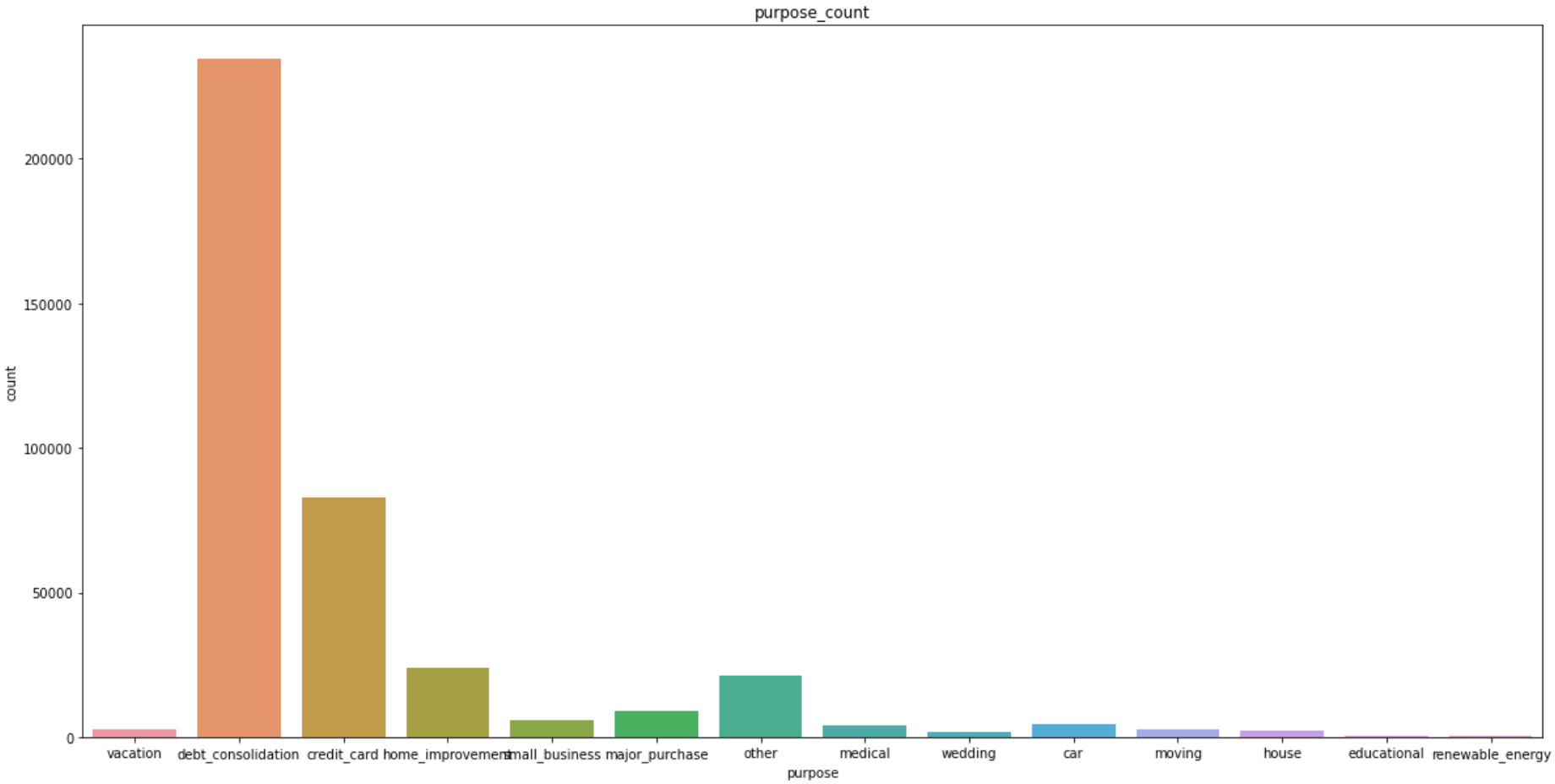
Out[259]:
```
Text(0.5, 1.0, 'verification_status_count')
```



In [266]:
```
fig = plt.figure(figsize = (20, 10))
seaborn.countplot(df['purpose'])
plt.title("purpose_count")
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[266]: Text(0.5, 1.0, 'purpose_count')



In [272]:
```
df['title'].nunique()
```

Out[272]: 48817

In [279]:
```
df['earliest_cr_line'].nunique()
```

Out[279]: 684

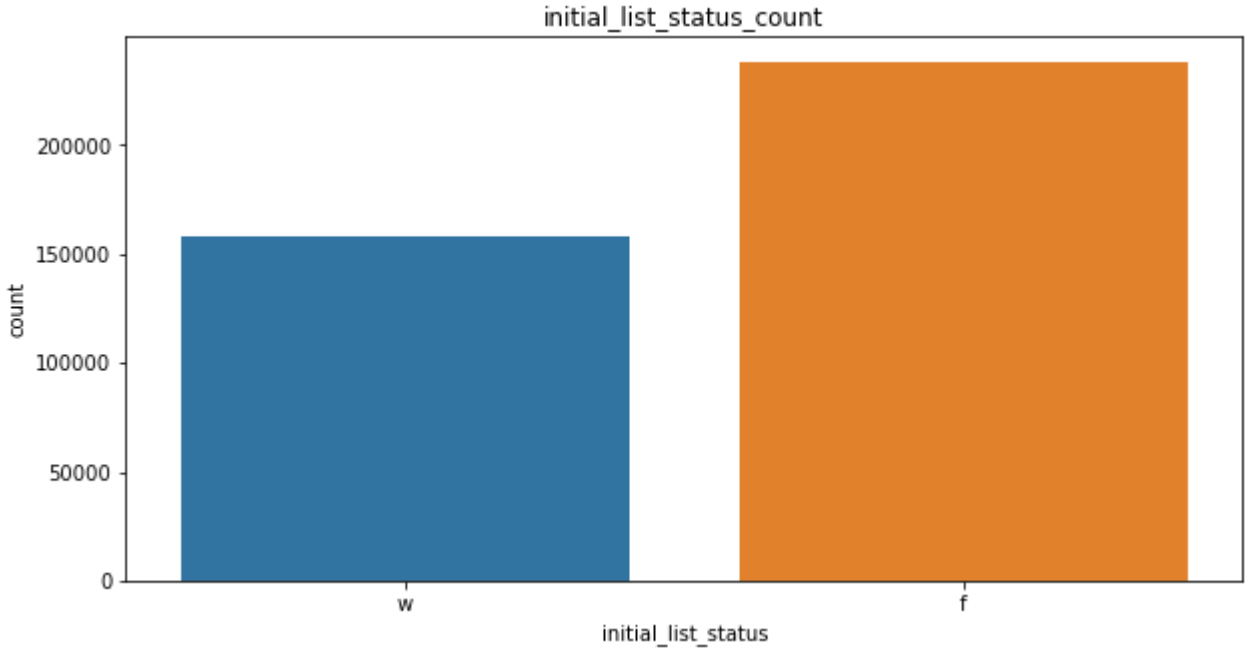
In [ ]:
```

```

In [276]:
```
fig = plt.figure(figsize = (10, 5))
seaborn.countplot(df['initial_list_status'])
plt.title("initial_list_status_count")
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

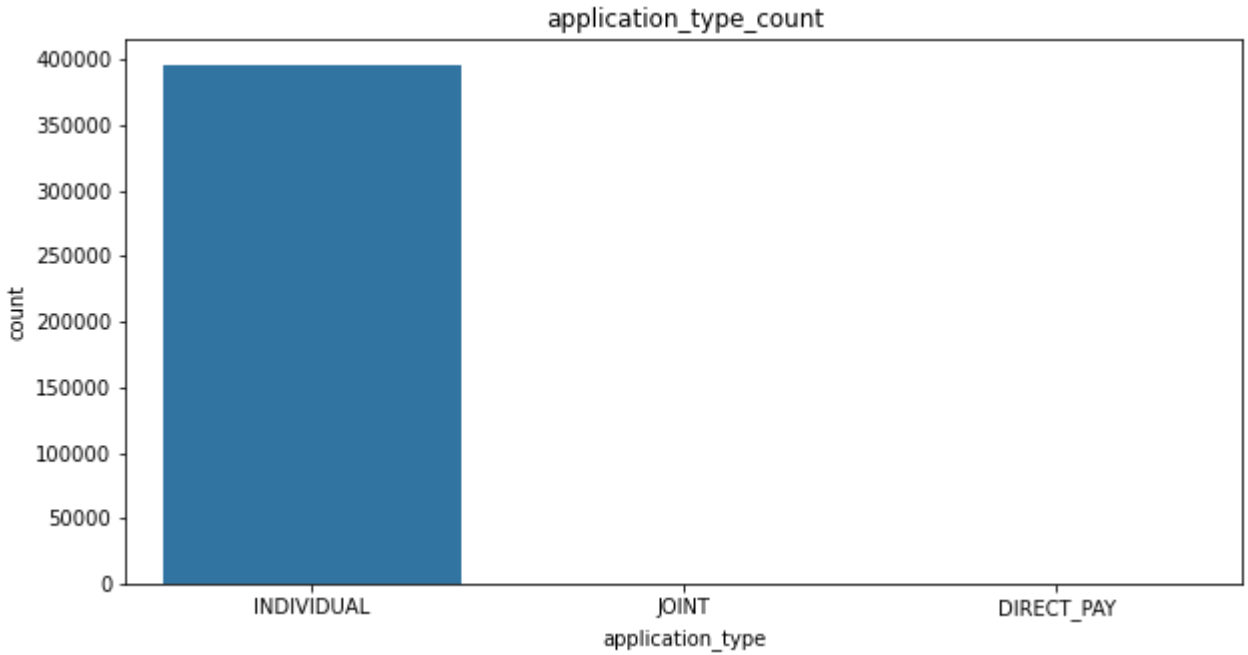
Out[276]:
```
Text(0.5, 1.0, 'initial_list_status_count')
```



In [281]:
```
fig = plt.figure(figsize = (10, 5))
seaborn.countplot(df['application_type'])
plt.title("application_type_count")
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[281]: Text(0.5, 1.0, 'application_type_count')



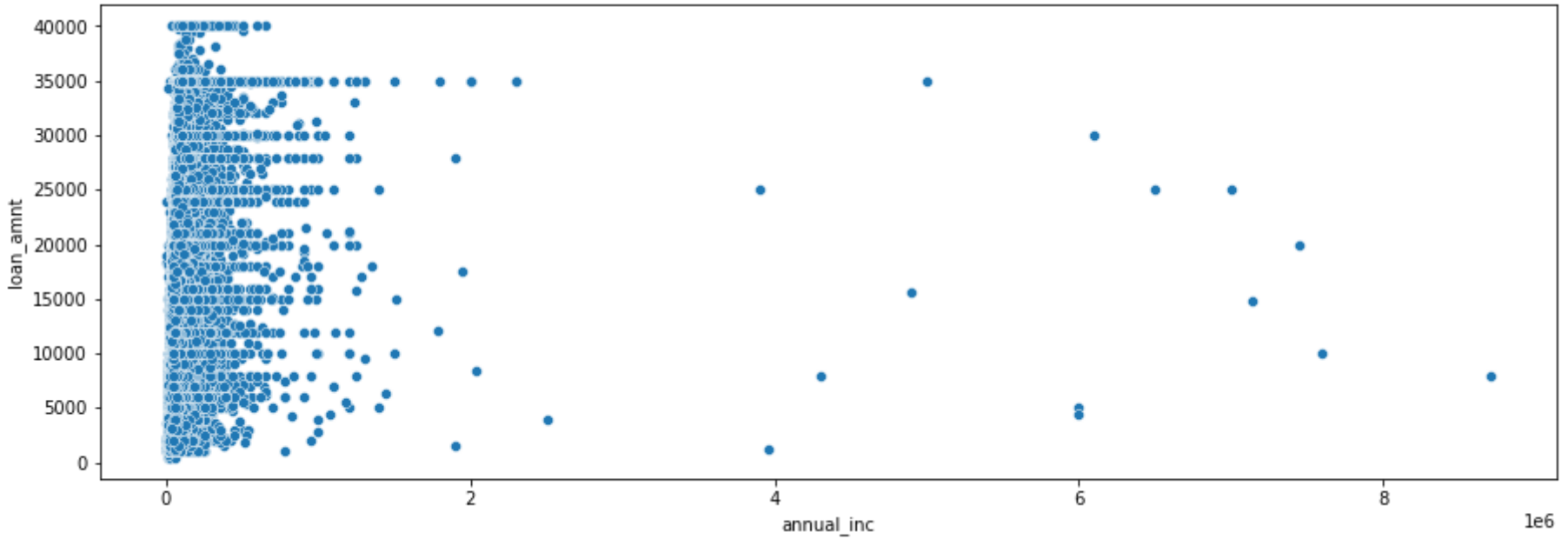
In [ ]:

In [284]:
```
df.columns
```

Out[284]:
```
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')
```

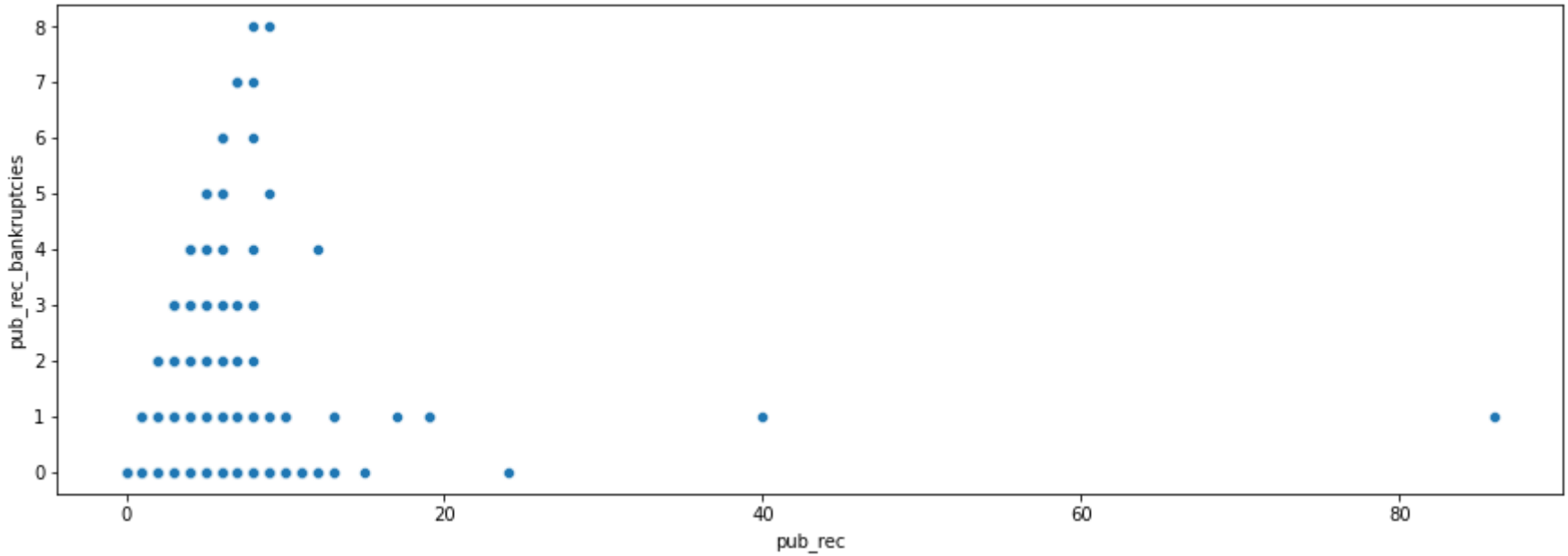
In [297]:
```
fig = plt.figure(figsize = (15, 5))
sns.scatterplot(data=df, y='loan_amnt', x='annual_inc')
```

Out[297]:
```
<AxesSubplot:xlabel='annual_inc', ylabel='loan_amnt'>
```



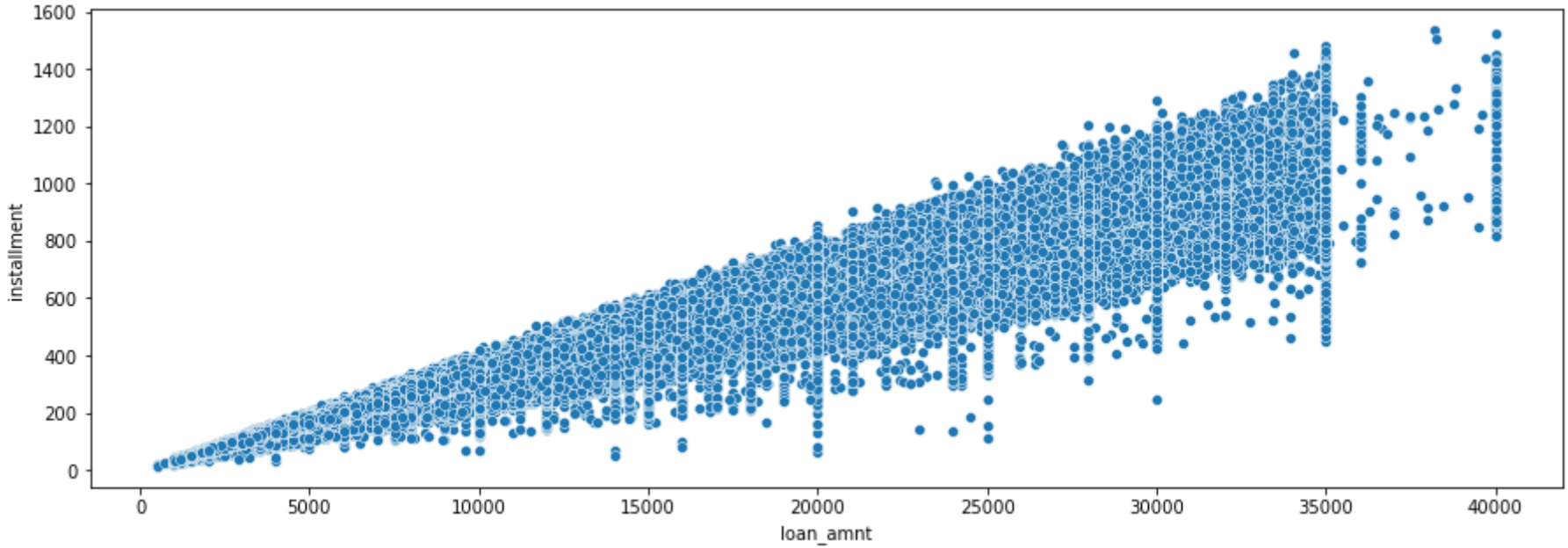
In [303]:
```
fig = plt.figure(figsize = (15, 5))
sns.scatterplot(data=df, x='pub_rec', y='pub_rec_bankruptcies')
```

Out[303]:
```
<AxesSubplot:xlabel='pub_rec', ylabel='pub_rec_bankruptcies'>
```



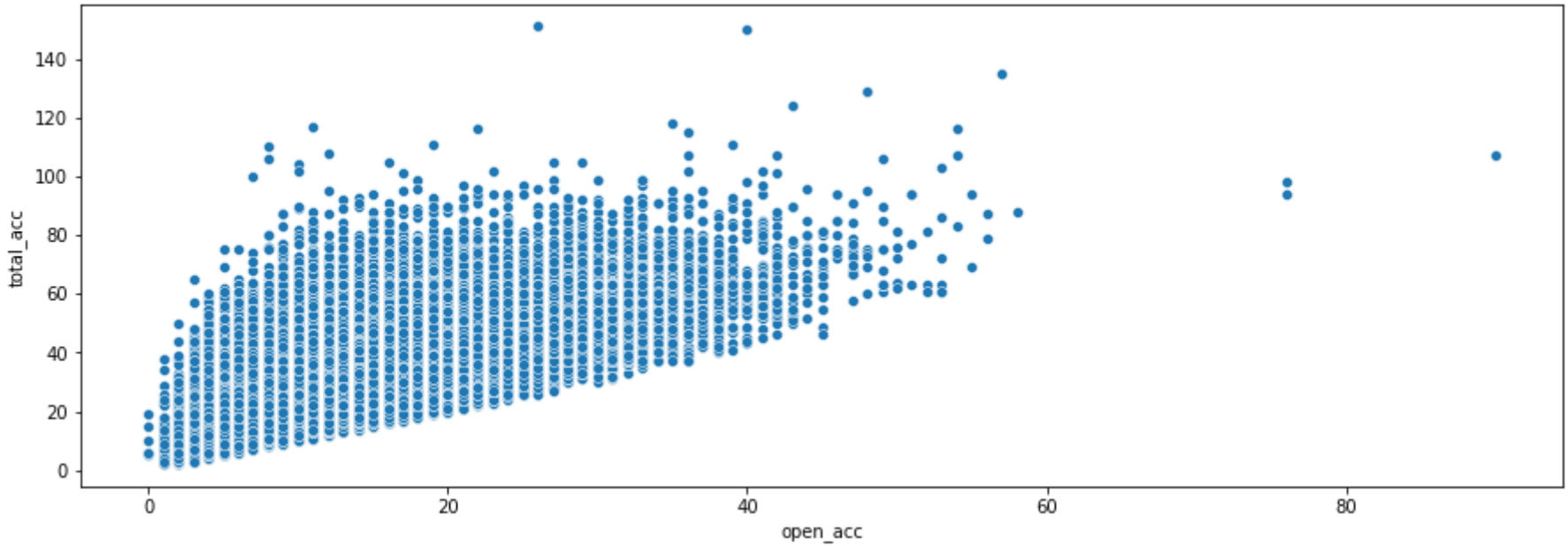
In [300]:
```
fig = plt.figure(figsize = (15, 5))
sns.scatterplot(data=df, x='loan_amnt', y='installment')
```

Out[300]: <AxesSubplot:xlabel='loan_amnt', ylabel='installment'>



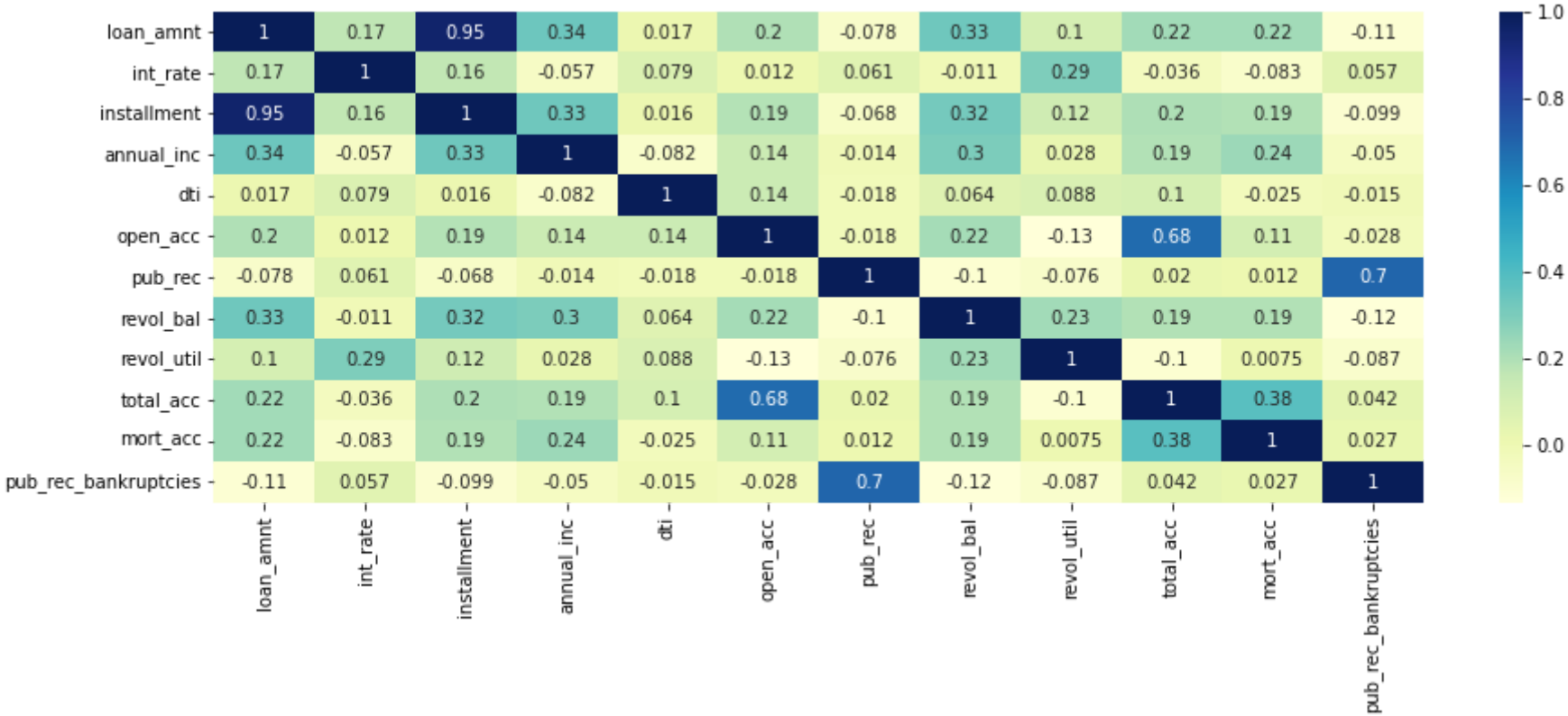
In [302]:
```
fig = plt.figure(figsize = (15, 5))
sns.scatterplot(data=df, x='open_acc', y='total_acc')
```

Out[302]: <AxesSubplot:xlabel='open_acc', ylabel='total_acc'>



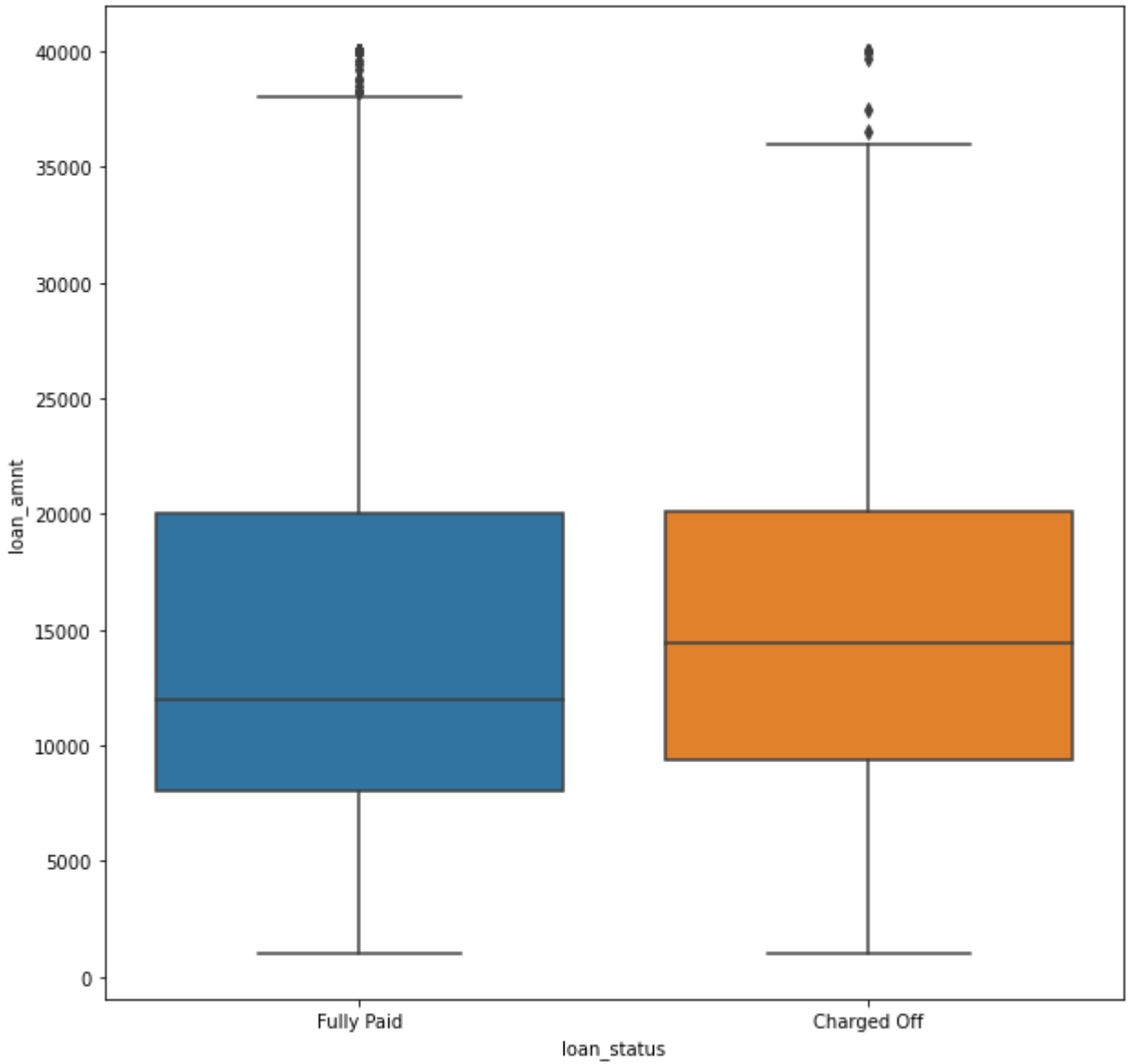
In [292]:
```
fig = plt.figure(figsize = (15, 5))
sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)
```

Out[292]: <AxesSubplot:>



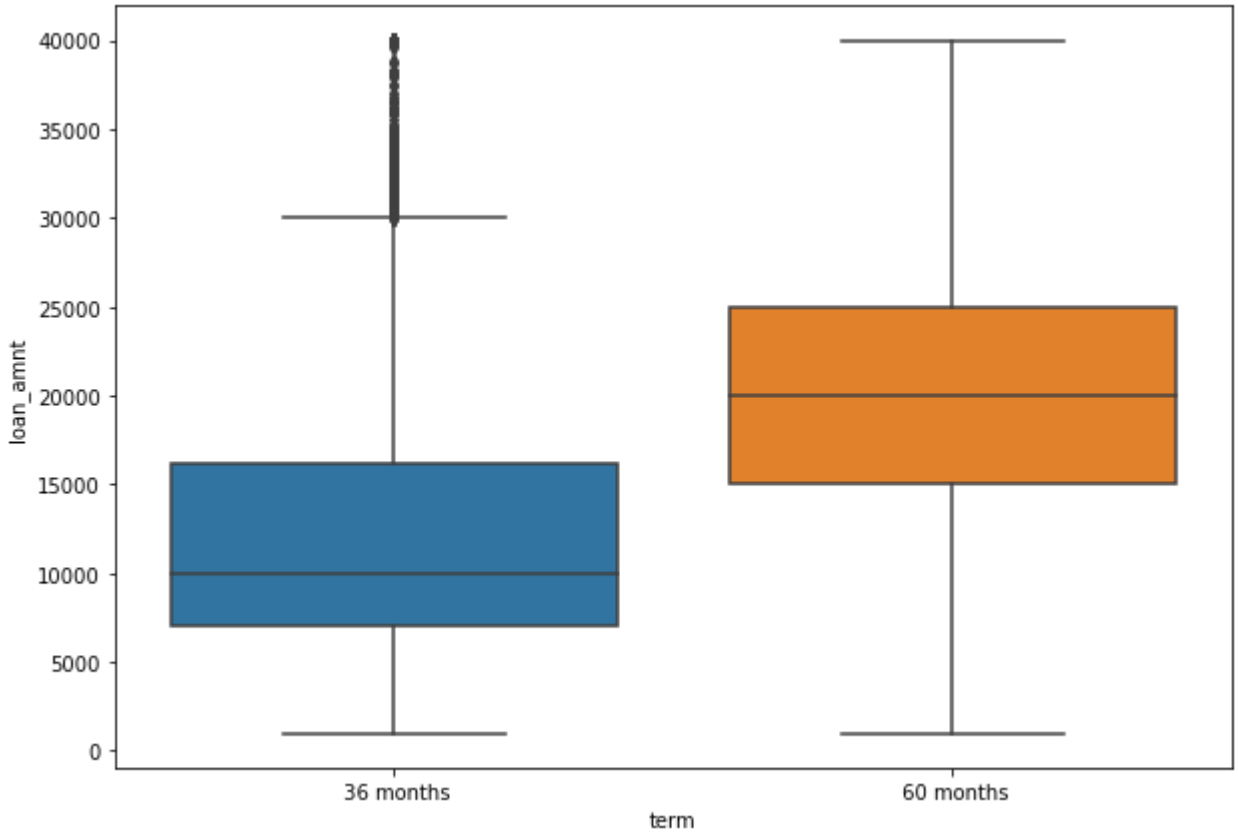
In [325]:
```
fig = plt.figure(figsize = (10, 10))
seaborn.boxplot(y=df['loan_amnt'], x=df['loan_status'], orient='v')
```

Out[325]: <AxesSubplot:xlabel='loan_status', ylabel='loan_amnt'>



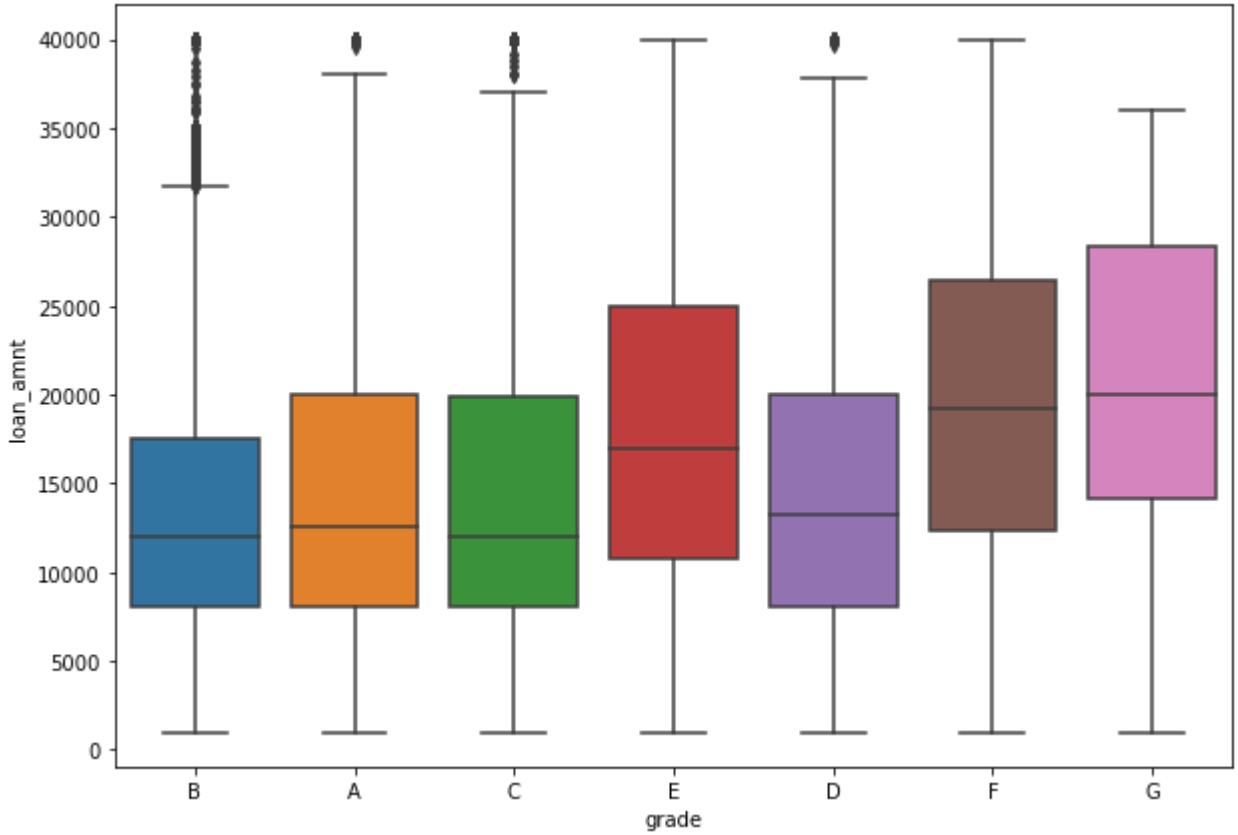
In [327]:
```
fig = plt.figure(figsize = (10, 7))
seaborn.boxplot(y=df['loan_amnt'], x=df['term'], orient='v')
```

Out[327]: <AxesSubplot:xlabel='term', ylabel='loan_amnt'>



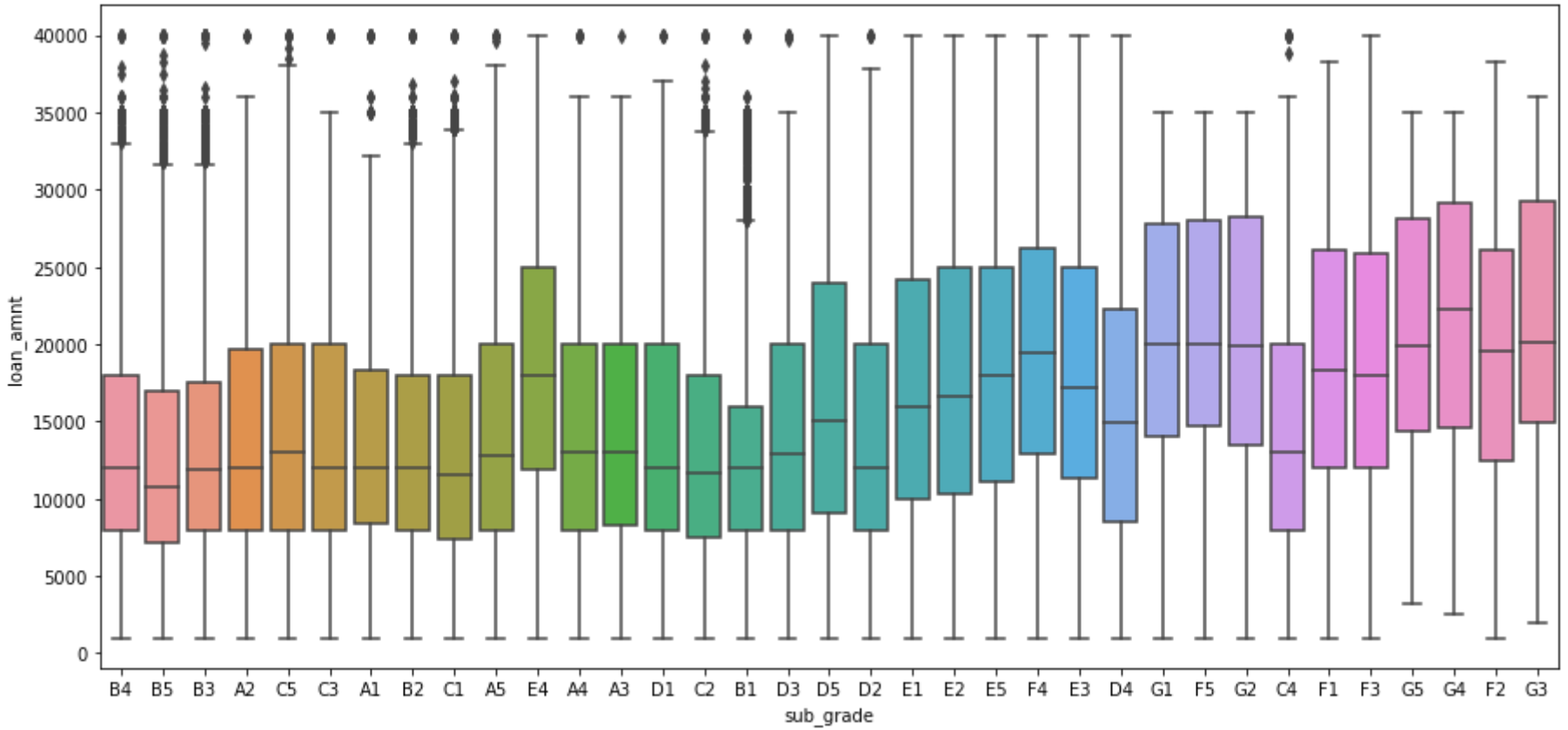
In [336]:
```
fig = plt.figure(figsize = (10, 7))
seaborn.boxplot(y=df['loan_amnt'], x=df['grade'], orient='v')
```

Out[336]: <AxesSubplot:xlabel='grade', ylabel='loan_amnt'>



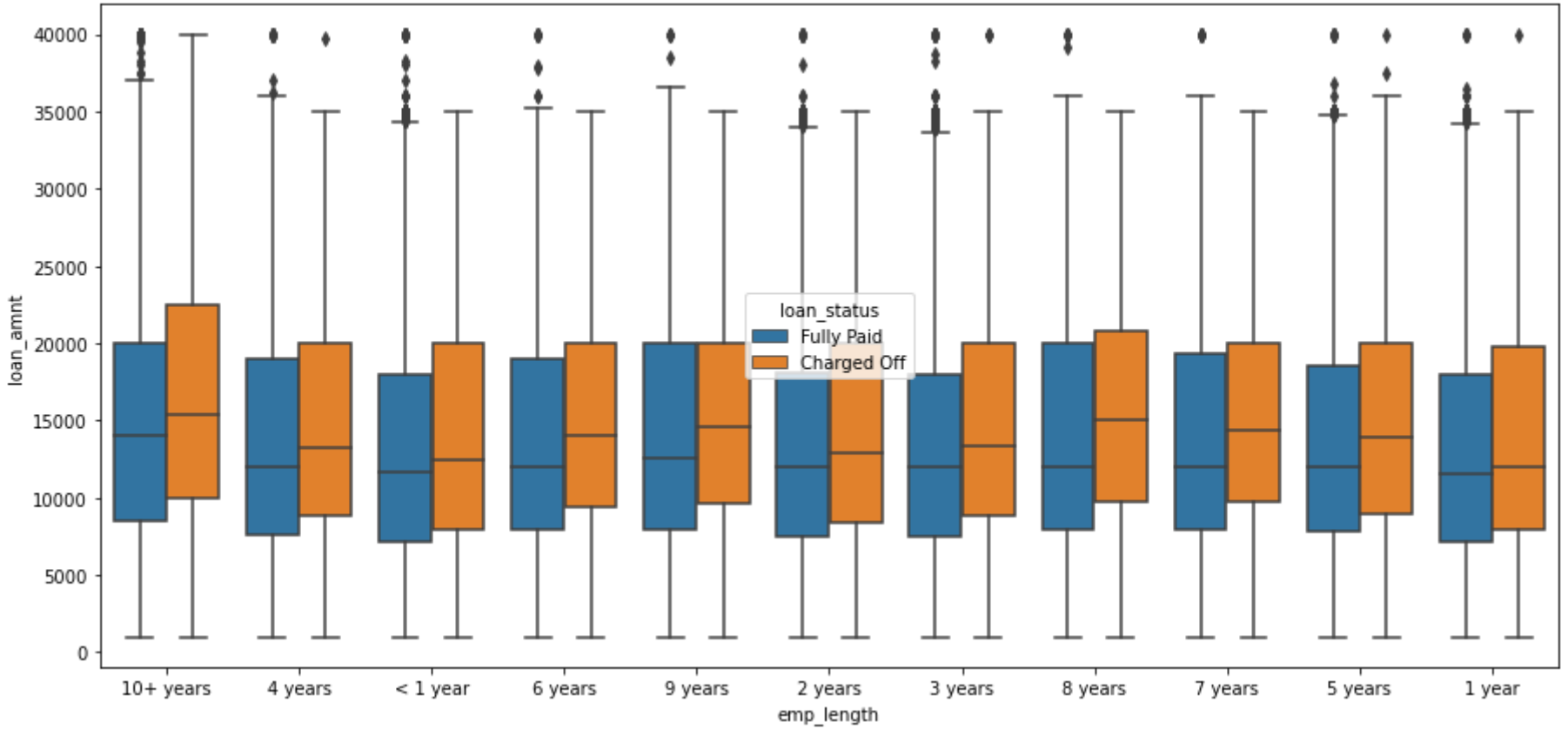
In [337]:
```
fig = plt.figure(figsize = (15, 7))
seaborn.boxplot(y=df['loan_amnt'], x=df['sub_grade'], orient='v')
```

Out[337]: <AxesSubplot:xlabel='sub_grade', ylabel='loan_amnt'>



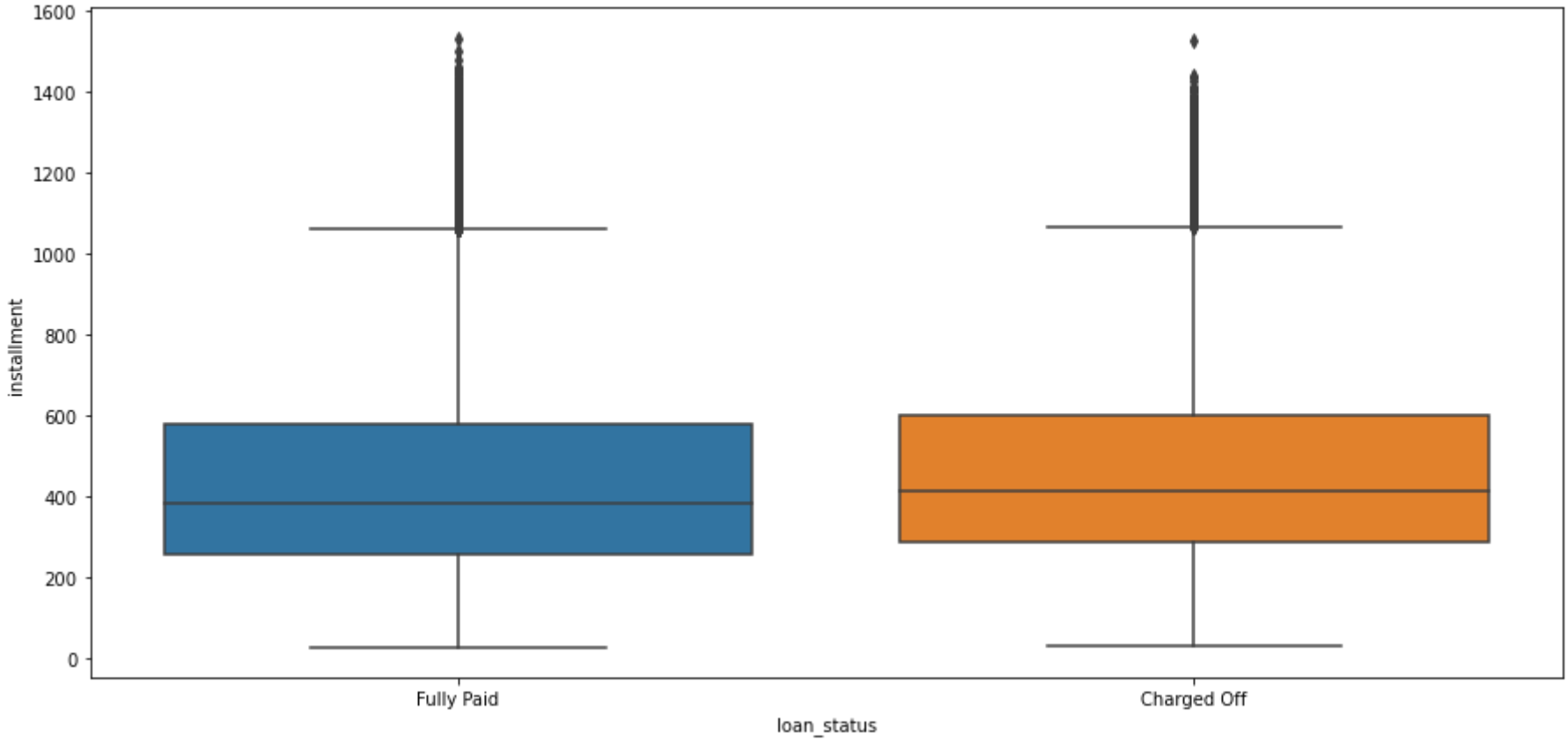
In [333]:
```
fig = plt.figure(figsize = (15, 7))
seaborn.boxplot(y=df['loan_amnt'], x=df['emp_length'],hue=df['loan_status'], orient='v')
```

Out[333]: <AxesSubplot:xlabel='emp_length', ylabel='loan_amnt'>



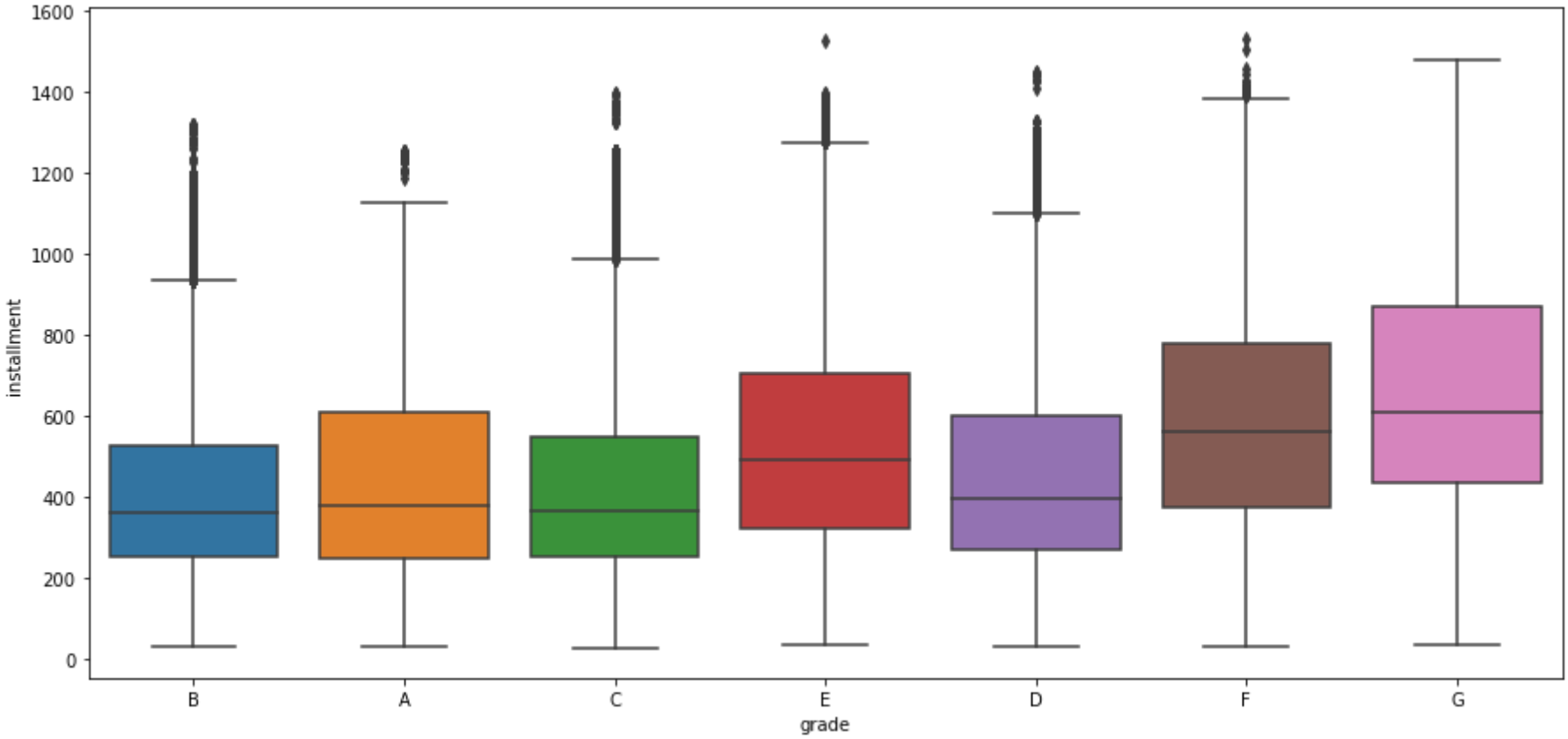
In [341]:
```
fig = plt.figure(figsize = (15, 7))
seaborn.boxplot(y=df['installment'], x=df['loan_status'], orient='v')
```

Out[341]: <AxesSubplot:xlabel='loan_status', ylabel='installment'>



In [342]:
```
fig = plt.figure(figsize = (15, 7))
seaborn.boxplot(y=df['installment'], x=df['grade'], orient='v')
```

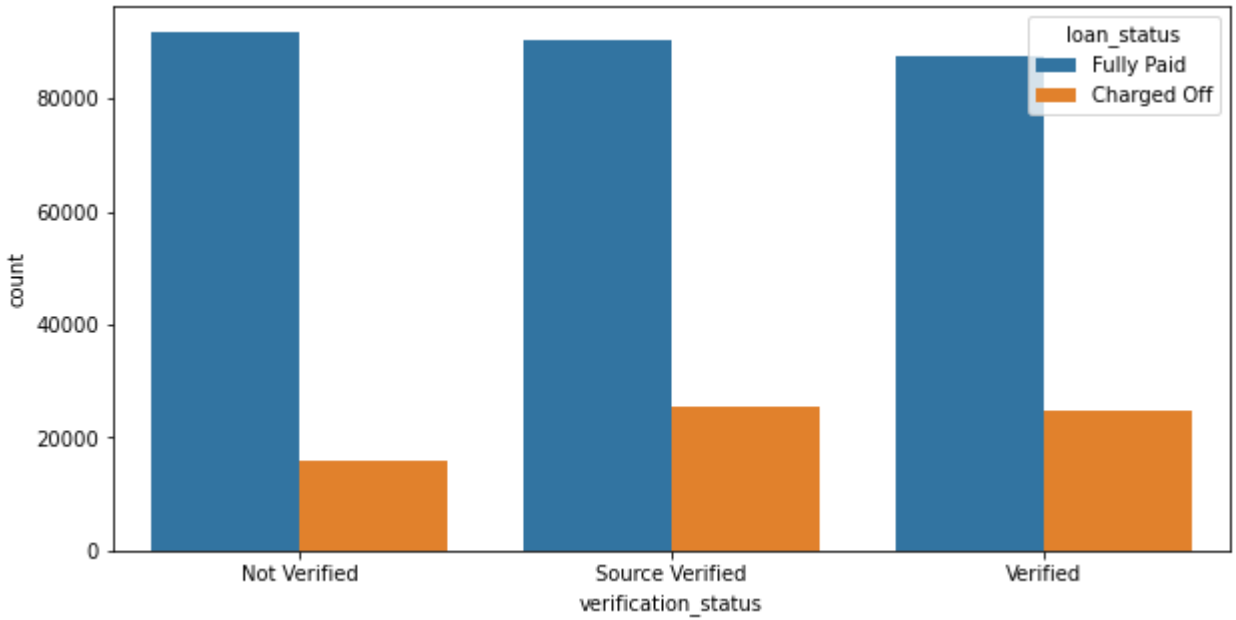
Out[342]: <AxesSubplot:xlabel='grade', ylabel='installment'>



In [346]:
```
fig = plt.figure(figsize = (10, 5))
seaborn.countplot(df['verification_status'], hue=df['loan_status'])
```

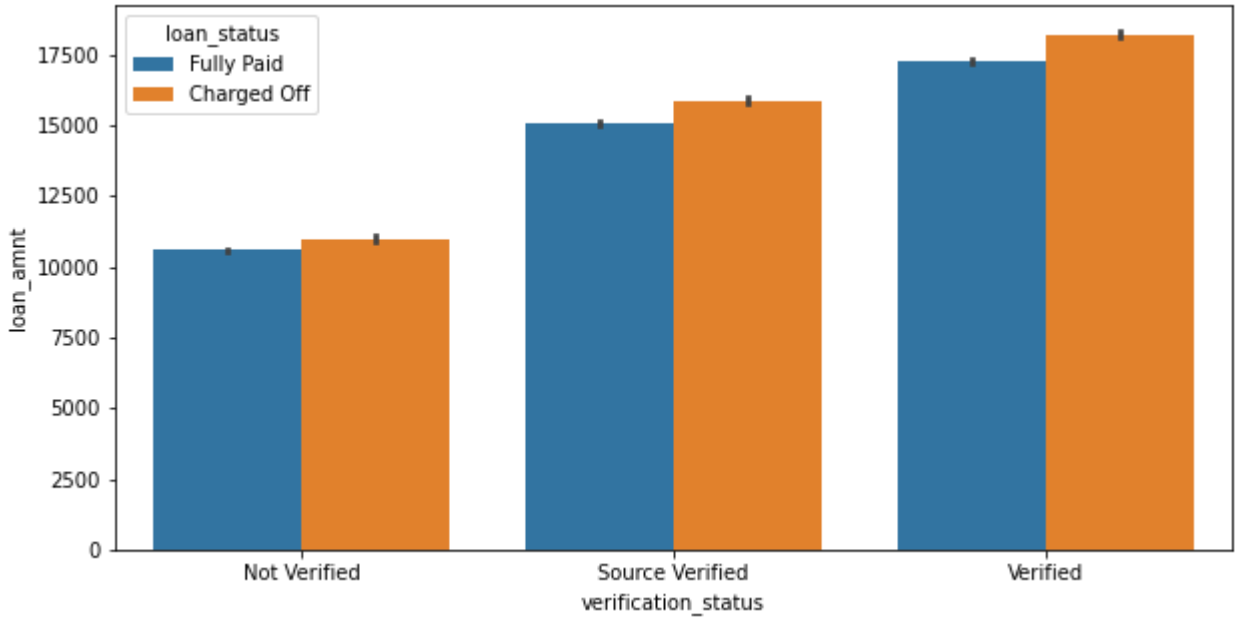
```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[346]: <AxesSubplot:xlabel='verification_status', ylabel='count'>



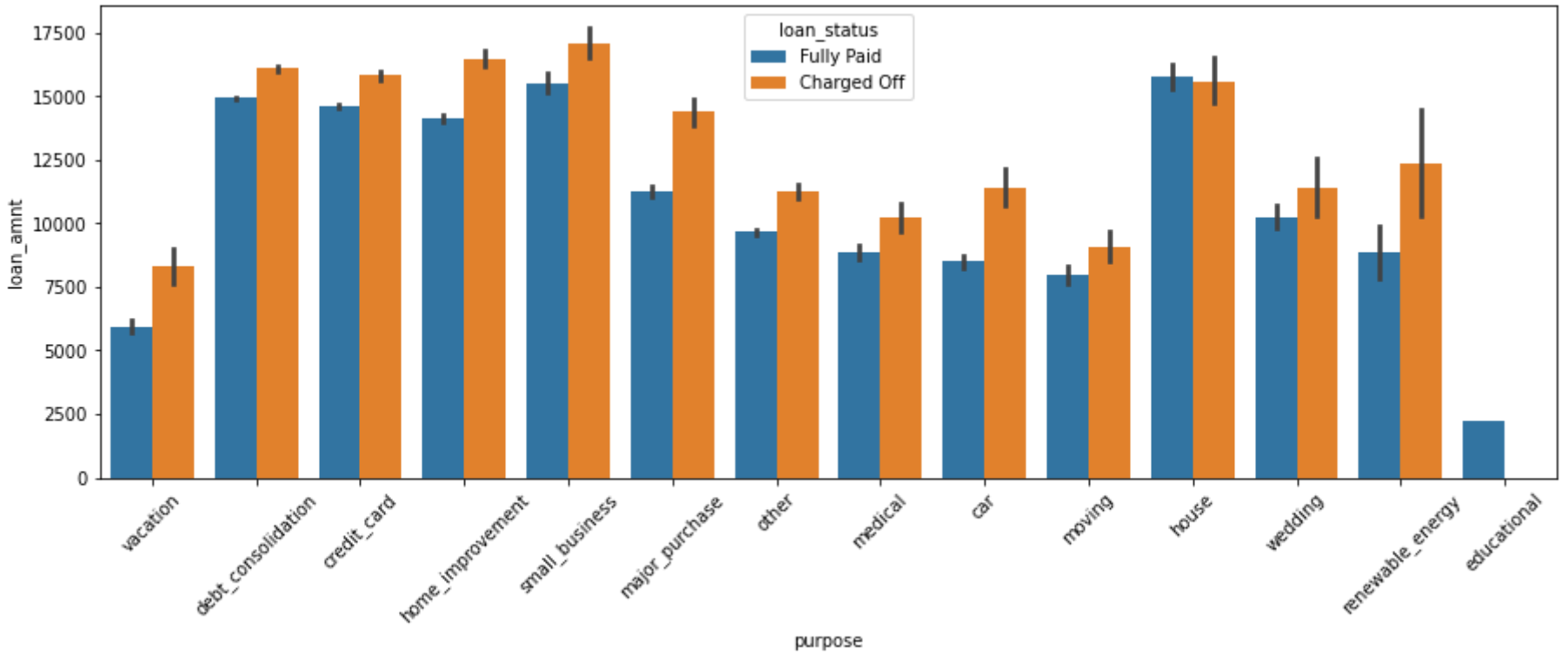
In [351]:
```
fig = plt.figure(figsize = (10, 5))
seaborn.barplot(x=df['verification_status'],y=df['loan_amnt'] ,hue=df['loan_status'])
```

Out[351]: <AxesSubplot:xlabel='verification_status', ylabel='loan_amnt'>



In [354]:
```
fig = plt.figure(figsize = (15, 5))
seaborn.barplot(x=df['purpose'],y=df['loan_amnt'] ,hue=df['loan_status'])
plt.xticks(rotation = 45)
```

Out[354]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13]),
 [Text(0, 0, 'vacation'),
  Text(1, 0, 'debt_consolidation'),
  Text(2, 0, 'credit_card'),
  Text(3, 0, 'home_improvement'),
  Text(4, 0, 'small_business'),
  Text(5, 0, 'major_purchase'),
  Text(6, 0, 'other'),
  Text(7, 0, 'medical'),
  Text(8, 0, 'car'),
  Text(9, 0, 'moving'),
  Text(10, 0, 'house'),
  Text(11, 0, 'wedding'),
  Text(12, 0, 'renewable_energy'),
  Text(13, 0, 'educational')])
```



Data Preprocessing

In [472]:
```
df.drop(labels = ["address", 'issue_d','earliest_cr_line'], axis = 1, inplace = True)
```

In [473]:
```
df.dropna(inplace=True)
```

In [474]:
```
df.isnull().sum()
```

Out[474]:
```
loan_amnt               0
term                    0
int_rate                0
installment             0
grade                   0
sub_grade               0
emp_title               0
emp_length              0
home_ownership          0
annual_inc              0
verification_status     0
loan_status             0
purpose                 0
title                   0
dti                     0
open_acc                0
pub_rec                 0
revol_bal               0
revol_util              0
total_acc               0
initial_list_status     0
application_type        0
mort_acc                0
pub_rec_bankruptcies    0
dtype: int64
```

In [430]:
```
df['title'].nunique()
```

Out[430]:
```
32187
```

In [475]:
```
for i in range(len(df.columns)):

    if (df[df.columns[i]].dtypes)=='float64':

        Q1 = np.percentile(df[df.columns[i]], 25,
                   interpolation = 'midpoint')

        Q3 = np.percentile(df[df.columns[i]], 75,
                   interpolation = 'midpoint')
        IQR = Q3 - Q1


        # Upper bound
        df[df.columns[i]] = (df[df.columns[i]] <= (Q3+1.5*IQR))
        # Lower bound
        df[df.columns[i]] = (df[df.columns[i]] >= (Q1-1.5*IQR))
```

In [ ]:
```
```

In [432]:
```
a
```

Out[432]:
```
['term',
 'grade',
 'sub_grade',
 'emp_title',
 'emp_length',
 'home_ownership',
 'verification_status',
 'issue_d',
 'loan_status',
 'purpose',
 'title',
 'earliest_cr_line',
 'initial_list_status',
 'application_type',
 'address']
```

In [433]:
```
df.initial_list_status.value_counts()
```

Out[433]:
```
f    187555
w    148313
Name: initial_list_status, dtype: int64
```

In [434]:
```
drop - earliest_cr_line, issue_d, address
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [434], in <cell line: 1>()
----> 1 drop - earliest_cr_line, issue_d, address

NameError: name 'drop' is not defined
```

In [ ]:
```

```

In [506]:
```
df.head()
```

Out[506]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_acc	pub_rec	revol_
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	...	16.0	0.0	3636
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	...	17.0	0.0	2013
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	...	13.0	0.0	1198
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	...	6.0	0.0	547
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	...	13.0	0.0	2458

5 rows × 27 columns

In [ ]:
```
```

In [417]:
```
df.columns
```

Out[417]:
```
Index(['loan_amnt', 'term', 'int_rate', 'installment', 'grade', 'sub_grade',
       'emp_title', 'emp_length', 'home_ownership', 'annual_inc',
       'verification_status', 'loan_status', 'purpose', 'title', 'dti',
       'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'initial_list_status', 'application_type', 'mort_acc',
       'pub_rec_bankruptcies'],
      dtype='object')
```

In [419]:
```
df.dtypes
```

Out[419]:
```
loan_amnt               bool
term                  object
int_rate                bool
installment             bool
grade                 object
sub_grade             object
emp_title             object
emp_length            object
home_ownership        object
annual_inc              bool
verification_status   object
loan_status           object
purpose               object
title                 object
dti                     bool
open_acc                bool
pub_rec                 bool
revol_bal               bool
revol_util              bool
total_acc               bool
initial_list_status   object
application_type      object
mort_acc                bool
pub_rec_bankruptcies    bool
dtype: object
```

In [476]:
```
kk=pd.DataFrame()
```

In [497]:
```
ıt', 'annual_inc','dti','open_acc', 'pub_rec','revol_bal','total_acc','mort_acc','pub_rec_bankruptcies']].copy()
```

In [479]:
```
kk['term']=le.fit_transform(df['term'])
kk['grade']=le.fit_transform(df['grade'])
kk['sub_grade']=le.fit_transform(df['sub_grade'])
kk['emp_title']=le.fit_transform(df['emp_title'])
kk['emp_length']=le.fit_transform(df['emp_length'])
kk['home_ownership']=le.fit_transform(df['home_ownership'])
kk['verification_status']=le.fit_transform(df['verification_status'])
kk['loan_status']=le.fit_transform(df['loan_status'])
kk['purpose']=le.fit_transform(df['purpose'])
kk['title']=le.fit_transform(df['title'])
kk['initial_list_status']=le.fit_transform(df['initial_list_status'])
kk['application_type']=le.fit_transform(df['application_type'])
```

In [548]:
```
kk['loan_amnt']=df['loan_amnt']
kk['int_rate']=df['int_rate']
kk['installment']=df['installment']
kk['annual_inc']=df['annual_inc']
kk['dti']=df['dti']
kk['open_acc']=df['open_acc']
kk['pub_rec']=df['pub_rec']
kk['revol_bal']=df['revol_bal']
kk['total_acc']=df['total_acc']
kk['mort_acc']=df['mort_acc']
kk['pub_rec_bankruptcies']=df['pub_rec_bankruptcies']
```

In [549]:
```
kk.head()
```

Out[549]:

us	loan_status	purpose	title	...	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	total_acc	mort_acc	pub_rec_bankruptcies
0	1	12	24082	...	11.44	329.48	117000.0	26.24	16.0	0.0	36369.0	25.0	0.0	0.0
0	1	2	8936	...	11.99	265.68	65000.0	22.05	17.0	0.0	20131.0	27.0	3.0	0.0
1	1	1	6850	...	10.49	506.97	43057.0	12.79	13.0	0.0	11987.0	26.0	0.0	0.0
0	1	1	6850	...	6.49	220.65	54000.0	2.60	6.0	0.0	5472.0	13.0	0.0	0.0
2	0	1	6111	...	17.27	609.33	55000.0	33.95	13.0	0.0	24584.0	43.0	1.0	0.0

In [551]:
```
X = kk.drop(['loan_status'], axis=1)

y = kk['loan_status']
```

In [552]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

In [553]:
```
# train a logistic regression model on the training set
from sklearn.linear_model import LogisticRegression


# instantiate the model
logreg = LogisticRegression(solver='liblinear', random_state=0)


# fit the model
logreg.fit(X_train, y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [553], in <cell line: 10>()
      6 logreg = LogisticRegression(solver='liblinear', random_state=0)
      9 # fit the model
---> 10 logreg.fit(X_train, y_train)

File ~/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:1508, in LogisticRegressio
n.fit(self, X, y, sample_weight)
   1505 else:
   1506     _dtype = [np.float64, np.float32]
-> 1508 X, y = self._validate_data(
   1509     X,
   1510     y,
   1511     accept_sparse="csr",
   1512     dtype=_dtype,
   1513     order="C",
   1514     accept_large_sparse=solver not in ["liblinear", "sag", "saga"],
   1515 )
   1516 check_classification_targets(y)
   1517 self.classes_ = np.unique(y)

File ~/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:581, in BaseEstimator._validate_data(self,
 X, y, reset, validate_separately, **check_params)
    579         y = check_array(y, **check_y_params)
    580     else:
--> 581         X, y = check_X_y(X, y, **check_params)
    582     out = X, y
    584 if not no_val_X and check_params.get("ensure_2d", True):
```

```
File ~/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py:964, in check_X_y(X, y, accept_s
parse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_m
in_samples, ensure_min_features, y_numeric, estimator)
    961 if y is None:
    962     raise ValueError("y cannot be None")
--> 964 X = check_array(
    965     X,
    966     accept_sparse=accept_sparse,
    967     accept_large_sparse=accept_large_sparse,
    968     dtype=dtype,
    969     order=order,
    970     copy=copy,
    971     force_all_finite=force_all_finite,
    972     ensure_2d=ensure_2d,
    973     allow_nd=allow_nd,
    974     ensure_min_samples=ensure_min_samples,
    975     ensure_min_features=ensure_min_features,
    976     estimator=estimator,
    977 )
    979 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric)
    981 check_consistent_length(X, y)

File ~/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py:800, in check_array(array, accep
t_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples,
ensure_min_features, estimator)
    794         raise ValueError(
    795             "Found array with dim %d. %s expected <= 2."
    796             % (array.ndim, estimator_name)
    797         )
    799     if force_all_finite:
--> 800         _assert_all_finite(array, allow_nan=force_all_finite == "allow-nan")
    802 if ensure_min_samples > 0:
    803     n_samples = _num_samples(array)

File ~/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py:114, in _assert_all_finite(X, al
low_nan, msg_dtype)
    107     if (
    108         allow_nan
    109         and np.isinf(X).any()
    110         or not allow_nan
    111         and not np.isfinite(X).all()
    112     ):
```

```
    113         type_err = "infinity" if allow_nan else "NaN, infinity"
--> 114         raise ValueError(
    115             msg_err.format(
    116                 type_err, msg_dtype if msg_dtype is not None else X.dtype
    117             )
    118         )
    119 # for object dtype data, we only check for NaNs (GH-13254)
    120 elif X.dtype == np.dtype("object") and not allow_nan:

ValueError: Input contains NaN, infinity or a value too large for dtype('float64').
```