Loan status analysis based on bivariate and multivariate analysis

Problem statement : Analyse Loan data, and predict the relation of the loan_status with the other columns, like intrest rate, home ownership type and recommend the way to reduce the risk in loan approvement.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
sns.set_style("darkgrid")
sns.set_theme(style="ticks", color_codes=True)
cm = sns.light_palette("red", as_cmap=True)
```

This how we load the load csv

In [2]:
```
df = pd.read_csv('loan.csv', low_memory=False)
df.head()
```

Out[2]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	num_tl_90g_dpd_24m	nu
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	...	NaN	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	...	NaN	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	C5	...	NaN	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	...	NaN	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	...	NaN	

5 rows × 111 columns

Data cleaning

Checking datatype of the dataset

In [3]:
```
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
```

As there are 111 columns lets try to find how many of them can be deleted

In [4]:
```
df.isnull().sum() / df.count()
```

Out[4]:
```
id                              0.000000
member_id                       0.000000
loan_amnt                       0.000000
funded_amnt                     0.000000
funded_amnt_inv                 0.000000
                                  ...
tax_liens                       0.000983
tot_hi_cred_lim                      inf
total_bal_ex_mort                    inf
total_bc_limit                       inf
total_il_high_credit_limit           inf
Length: 111, dtype: float64
```

Delete the columns having null values more than 40%

In [293]:
```
result = df.isnull().sum() / df.count() > 0.50
for pair in zip(result.index, list(result)):
    index, isAlmostNull = pair
    if isAlmostNull:
        del df[index]

df.head()
df.describe()
```

Out[293]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	installment	annual_inc	dti	delinq_2yrs	inq_la
count	3.971700e+04	3.971700e+04	39717.000000	39717.000000	39717.000000	39717.000000	3.971700e+04	39717.000000	39717.000000	397
mean	6.831319e+05	8.504636e+05	11219.443815	10947.713196	10397.448868	324.561922	6.896893e+04	13.315130	0.146512	
std	2.106941e+05	2.656783e+05	7456.670694	7187.238670	7128.450439	208.874874	6.379377e+04	6.678594	0.491812	
min	5.473400e+04	7.069900e+04	500.000000	500.000000	0.000000	15.690000	4.000000e+03	0.000000	0.000000	
25%	5.162210e+05	6.667800e+05	5500.000000	5400.000000	5000.000000	167.020000	4.040400e+04	8.170000	0.000000	
50%	6.656650e+05	8.508120e+05	10000.000000	9600.000000	8975.000000	280.220000	5.900000e+04	13.400000	0.000000	
75%	8.377550e+05	1.047339e+06	15000.000000	15000.000000	14400.000000	430.780000	8.230000e+04	18.600000	0.000000	
max	1.077501e+06	1.314167e+06	35000.000000	35000.000000	35000.000000	1305.190000	6.000000e+06	29.990000	11.000000	

8 rows × 31 columns

We can see now there are only 53 columns left when we deleted all the columns having mostly null values

In [294]:
```
df.info()
```

```
 36  total_pymnt                 39717 non-null  float64
 37  total_pymnt_inv             39717 non-null  float64
 38  total_rec_prncp             39717 non-null  float64
 39  total_rec_int               39717 non-null  float64
 40  total_rec_late_fee          39717 non-null  float64
 41  recoveries                  39717 non-null  float64
 42  collection_recovery_fee     39717 non-null  float64
 43  last_pymnt_d                39646 non-null  object
 44  last_pymnt_amnt             39717 non-null  float64
 45  last_credit_pull_d          39715 non-null  object
 46  collections_12_mths_ex_med  39661 non-null  float64
 47  policy_code                 39717 non-null  int64
 48  application_type            39717 non-null  object
 49  acc_now_delinq              39717 non-null  int64
 50  chargeoff_within_12_mths    39661 non-null  float64
 51  delinq_amnt                 39717 non-null  int64
 52  pub_rec_bankruptcies        39020 non-null  float64
 53  tax_liens                   39678 non-null  float64
dtypes: float64(18), int64(13), object(23)
memory usage: 16.4+ MB
```

Lets add correct data type of int_rate

In [295]:
```python
import re

def cleanIntRate(intRate):
    newIntRate = re.sub(r'%+', '', str(intRate))
    if len(newIntRate):
        return float(newIntRate)

    return 0

df['int_rate'] = df['int_rate'].apply(cleanIntRate)
print(df['int_rate'].head(10))
df['int_rate'].describe()
```

```
0    10.65
1    15.27
2    15.96
3    13.49
4    12.69
5     7.90
6    15.96
7    18.64
8    21.28
9    12.69
Name: int_rate, dtype: float64
```

Out[295]:
```
count    39717.000000
mean        12.021177
std          3.724825
min          5.420000
25%          9.250000
50%         11.860000
75%         14.590000
max         24.590000
Name: int_rate, dtype: float64
```

Fill null values in int_rate with the most occuring value

In [296]:
```
df['int_rate'].mode()
```

Out[296]:
```
0    10.99
Name: int_rate, dtype: float64
```

In [297]:
```
df['int_rate'].fillna(df['int_rate'].mode(), inplace=True)
```

Creating the categorical data out of the int_rate column

In [298]:
```
np.quantile(df['int_rate'], 0.25)
```

Out[298]:
```
9.25
```

In [299]:
```
np.quantile(df['int_rate'], 0.50)
```

Out[299]:
```
11.86
```

In [300]:
```
np.quantile(df['int_rate'], 0.75)
```

Out[300]:
```
14.59
```

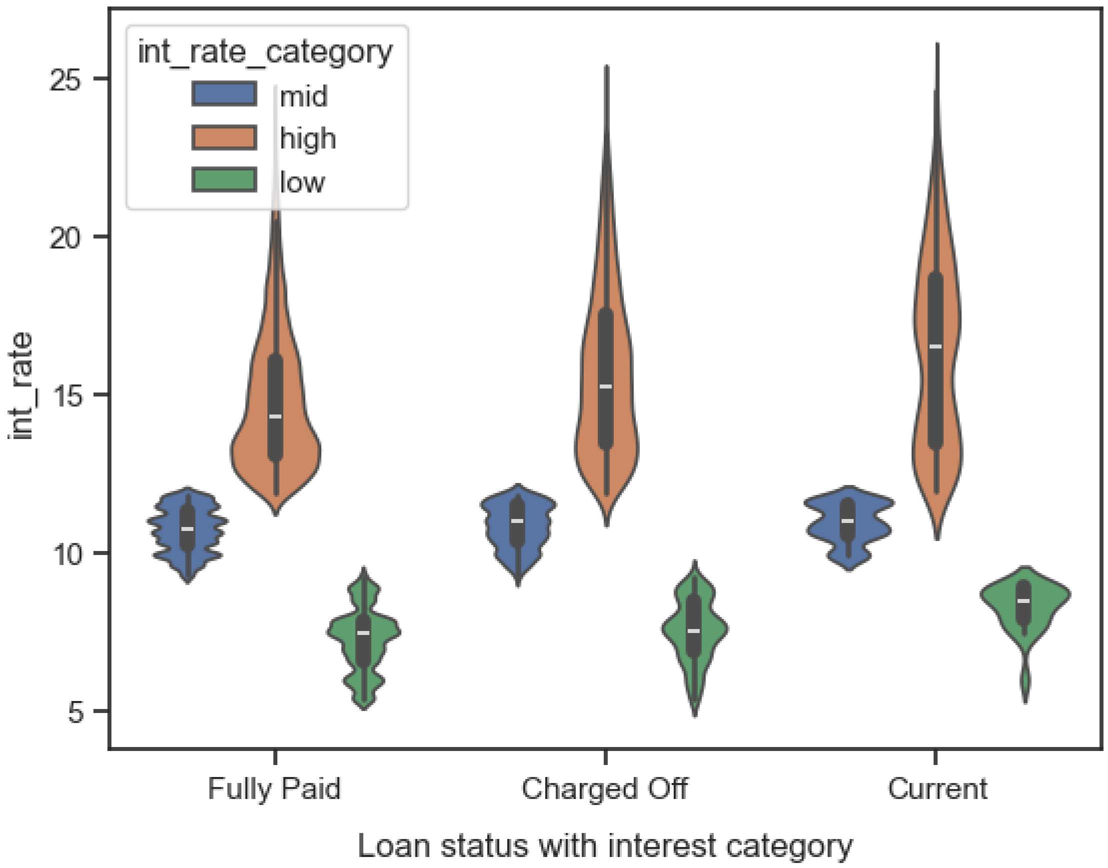
In [301]:
```
def createCategorical(int_rate):
    if int_rate <= 9.25:
        return 'low'
    elif int_rate <= 11.86:
        return 'mid'
    else:
        return 'high'

df['int_rate_category'] = df['int_rate'].apply(createCategorical)
df['int_rate_category'].head(10)
```

Out[301]:
```
0     mid
1    high
2    high
3    high
4    high
5     low
6    high
7    high
8    high
9    high
Name: int_rate_category, dtype: object
```

Uivariate and Multivariate analysis for the `int_rate_category` and `int_rate`

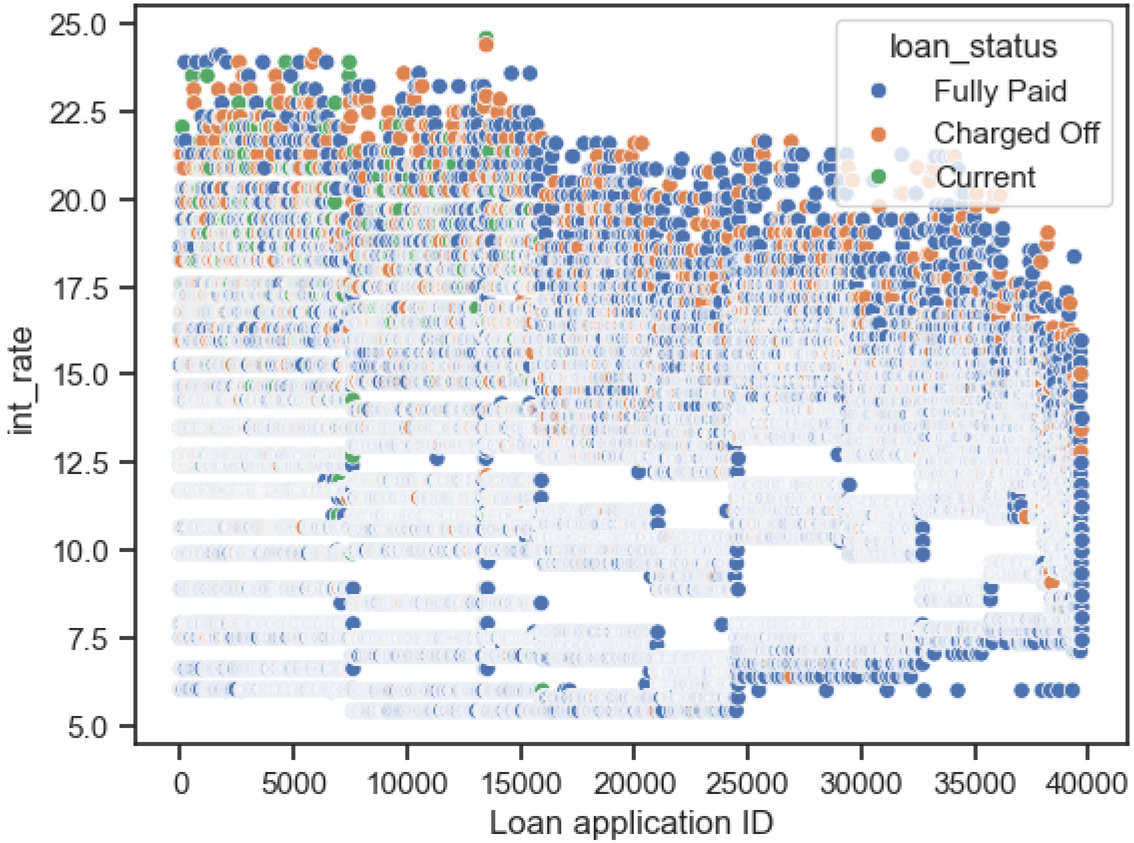
In [302]:
```
plot = sns.violinplot(df, y='int_rate', x='loan_status', hue=df['int_rate_category'])
plot.set_xlabel('Loan status with interest category', labelpad=10)
```

Out[302]: Text(0.5, 0, 'Loan status with interest category')



In [303]:
```
plot = sns.scatterplot(df, y='int_rate', x=df.index, hue=df['loan_status'])
plot.set_xlabel('Loan application ID')
```

Out[303]:
```
Text(0.5, 0, 'Loan application ID')
```

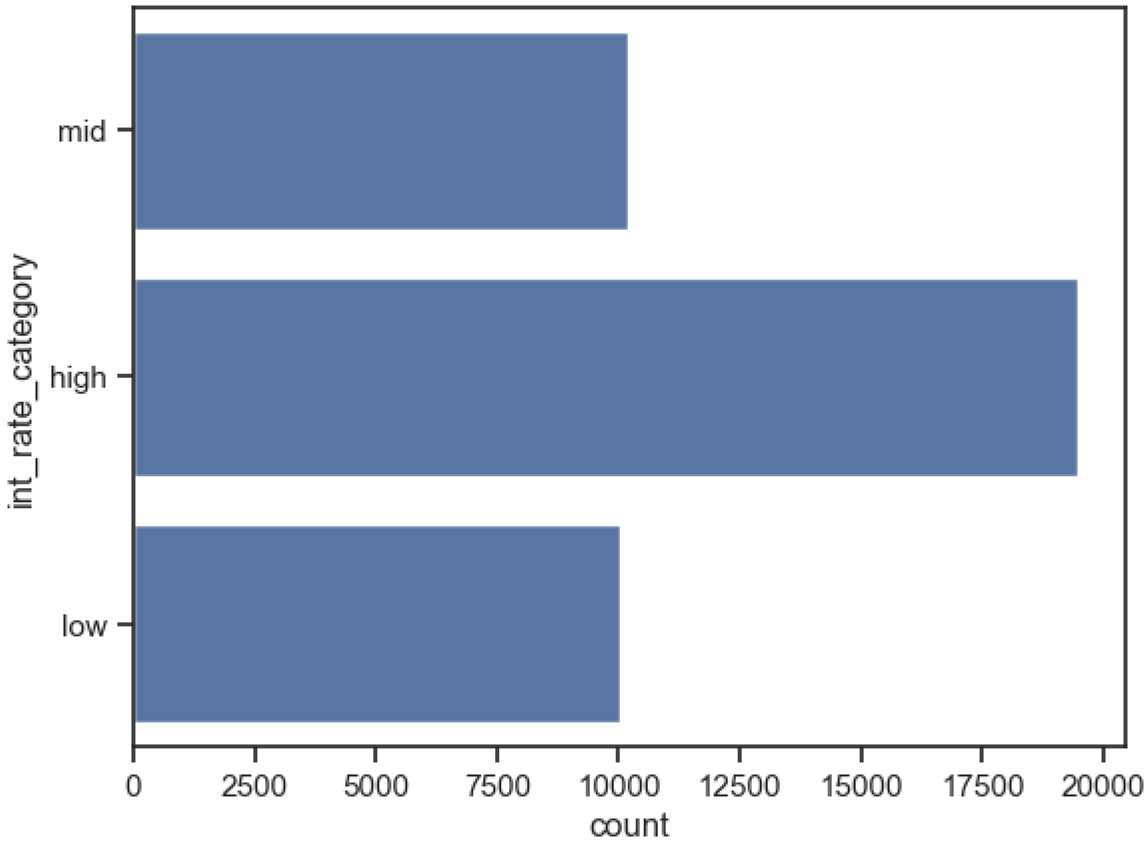


In [304]:
```
df['int_rate_category'].value_counts()
```

Out[304]:
```
high    19482
mid     10208
low     10027
Name: int_rate_category, dtype: int64
```

In [305]:
```
sns.countplot(df['int_rate_category'])
```

Out[305]: <AxesSubplot:xlabel='count', ylabel='int_rate_category'>



Observations

- Loan users tends to default when interest rate is higher.
- `Charged off` status rows in the dataset have the higer median (`int_rate`) than the other loan status data.

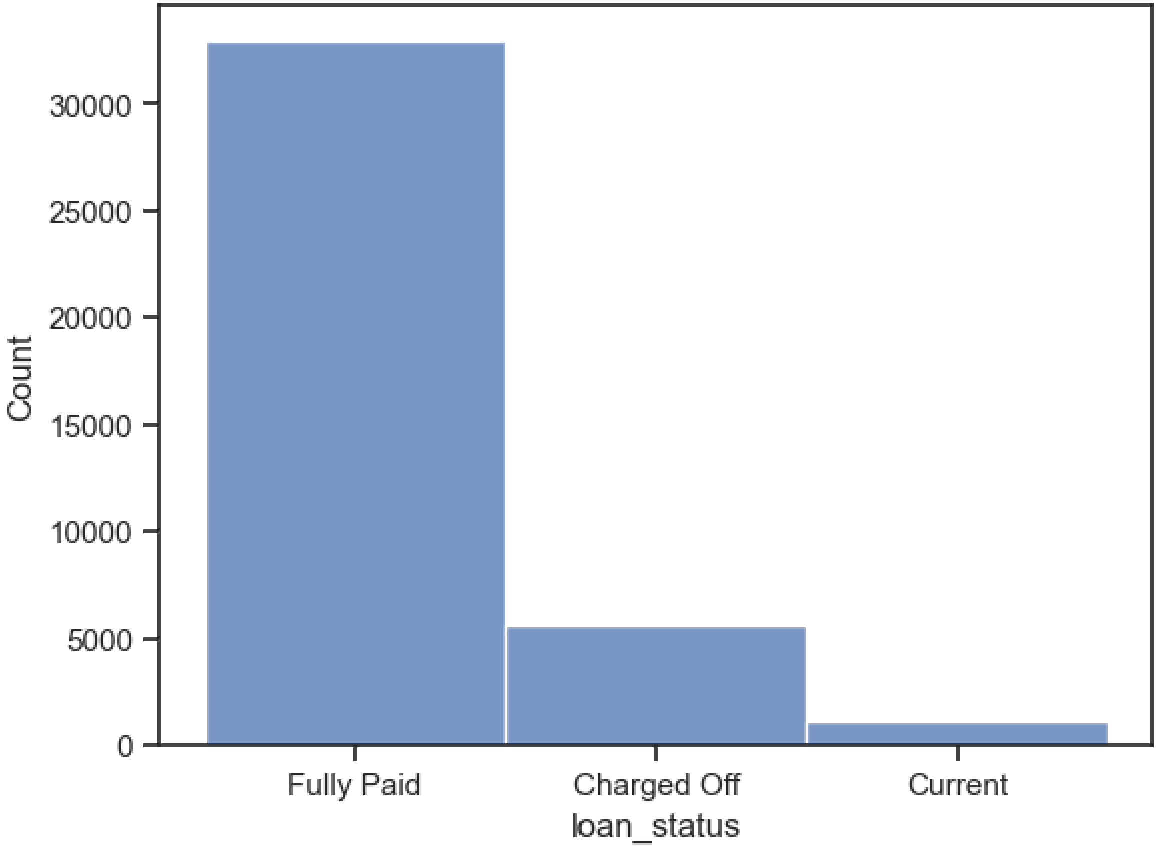
Uivariate and Bivariabte analysis for the `loan_status` with the other columns.

In [306]:
```
df['loan_status'].value_counts()
```

Out[306]:
```
Fully Paid     32950
Charged Off     5627
Current         1140
Name: loan_status, dtype: int64
```

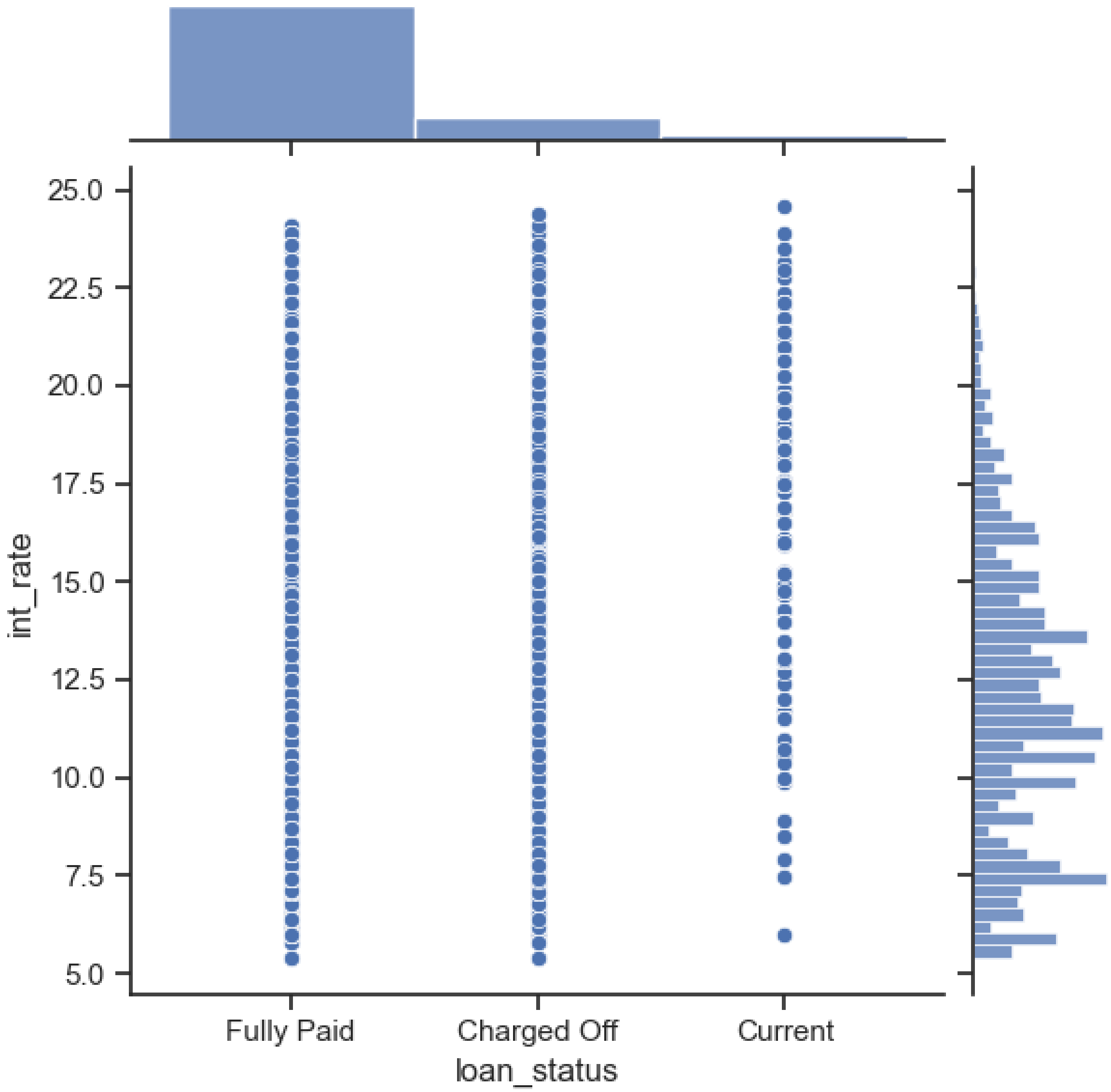
In [307]:
```
sns.histplot(df['loan_status'])
```

Out[307]:
```
<AxesSubplot:xlabel='loan_status', ylabel='Count'>
```



In [308]:
```
sns.jointplot(df, x='loan_status', y='int_rate')
```

Out[308]: <seaborn.axisgrid.JointGrid at 0x7fbc27fcf520>



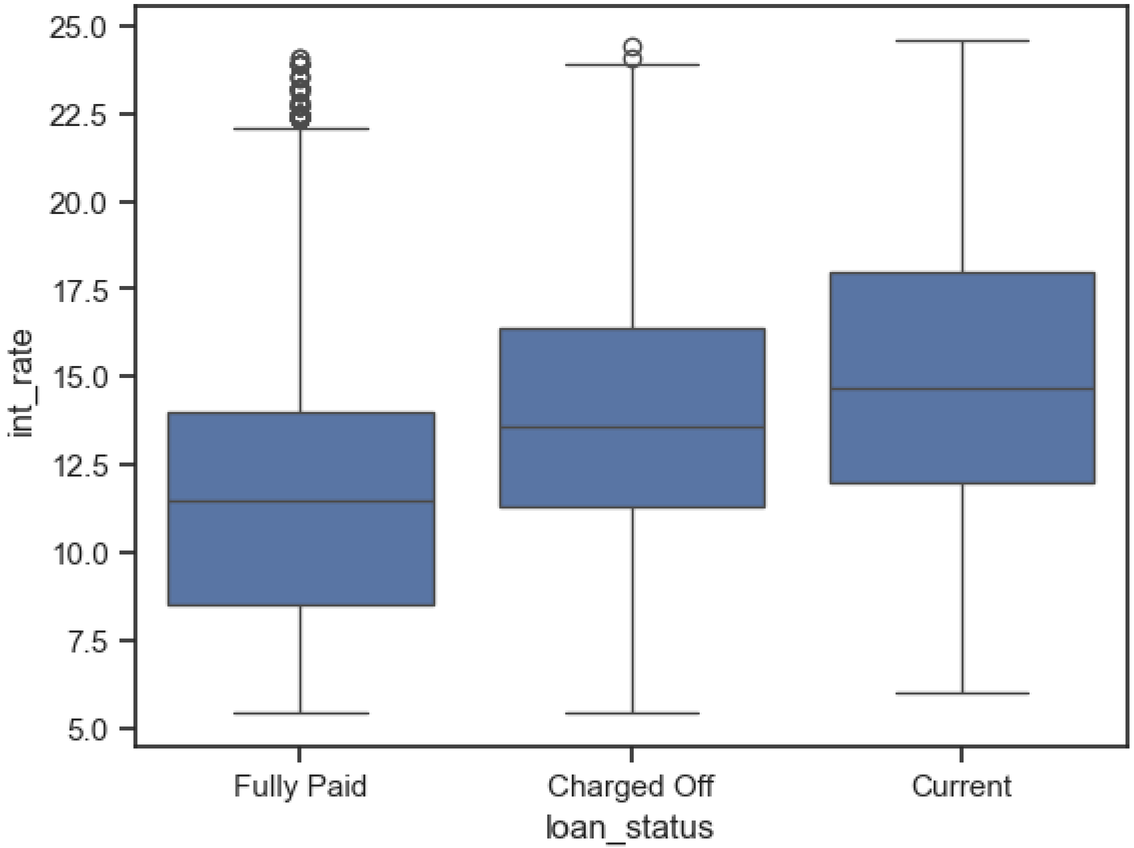
In [309]:
```
df.pivot_table(values=['int_rate'], index=['loan_status'], aggfunc=np.median)
```

Out[309]:

	int_rate
loan_status	
Charged Off	13.61
Current	14.65
Fully Paid	11.49

In [310]:
```
sns.boxplot(x=df.loan_status, y=df.int_rate)
```

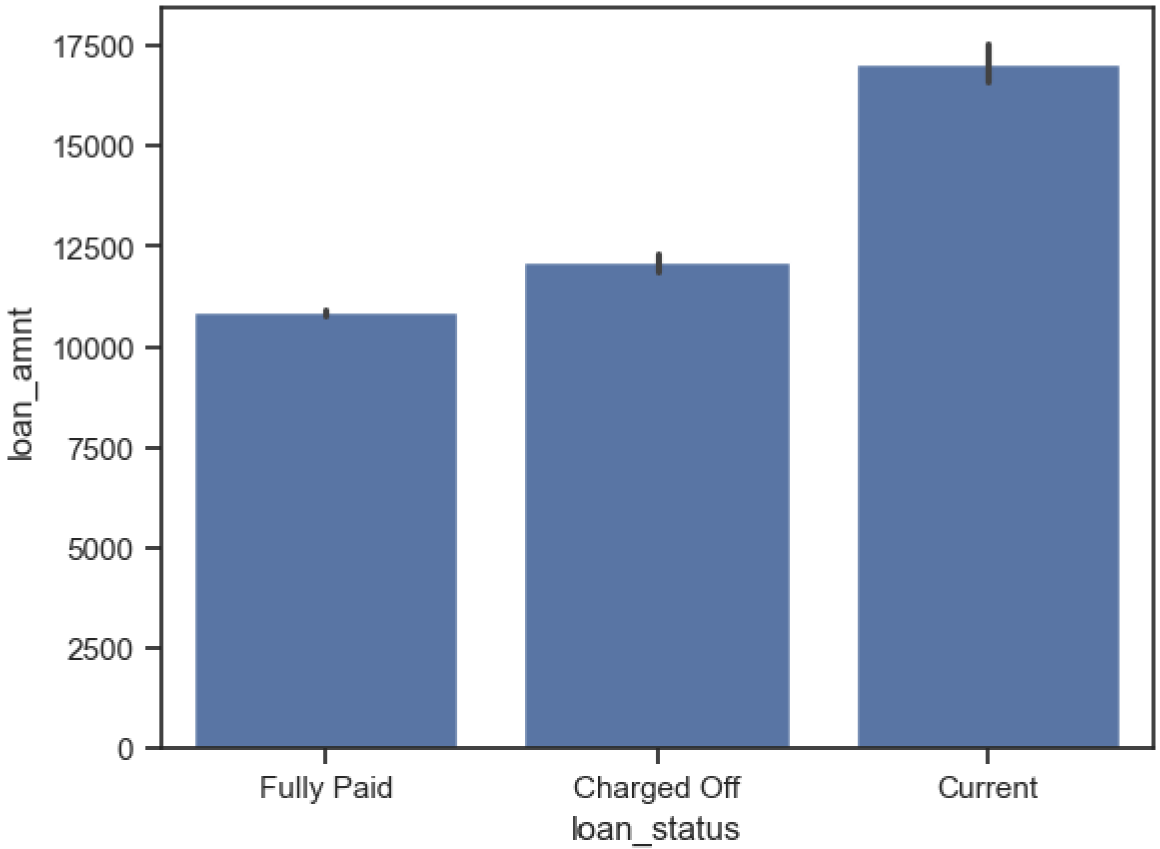
Out[310]: <AxesSubplot:xlabel='loan_status', ylabel='int_rate'>



`loan_status` and `loan_amount` lets do the analysis on this.

In [311]:
```
sns.barplot(y=df.loan_amnt, x=df.loan_status)
```

Out[311]:
```
<AxesSubplot:xlabel='loan_status', ylabel='loan_amnt'>
```



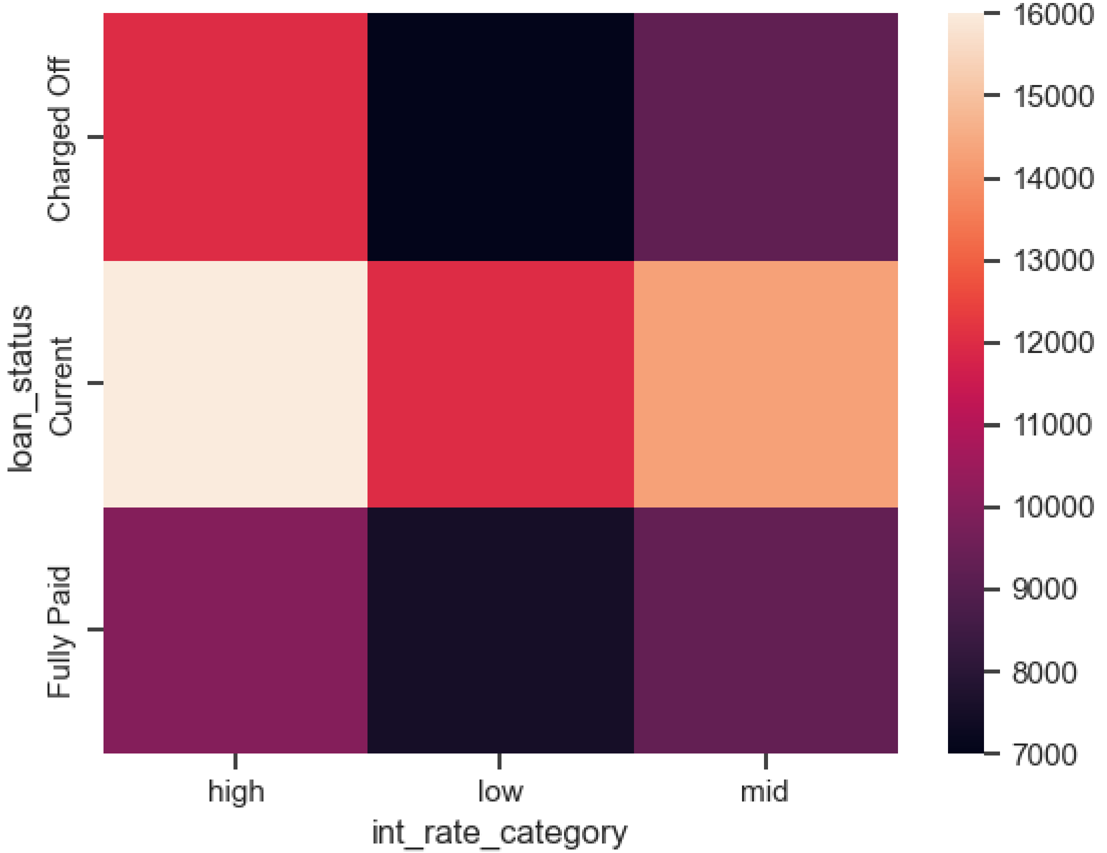
In [312]:
```
df.pivot_table(index='loan_status', columns="int_rate_category", values="loan_amnt", aggfunc='median')
```

Out[312]:

int_rate_category	high	low	mid
loan_status			
Charged Off	12000	7000	9225
Current	16000	12000	14300
Fully Paid	10000	7500	9300

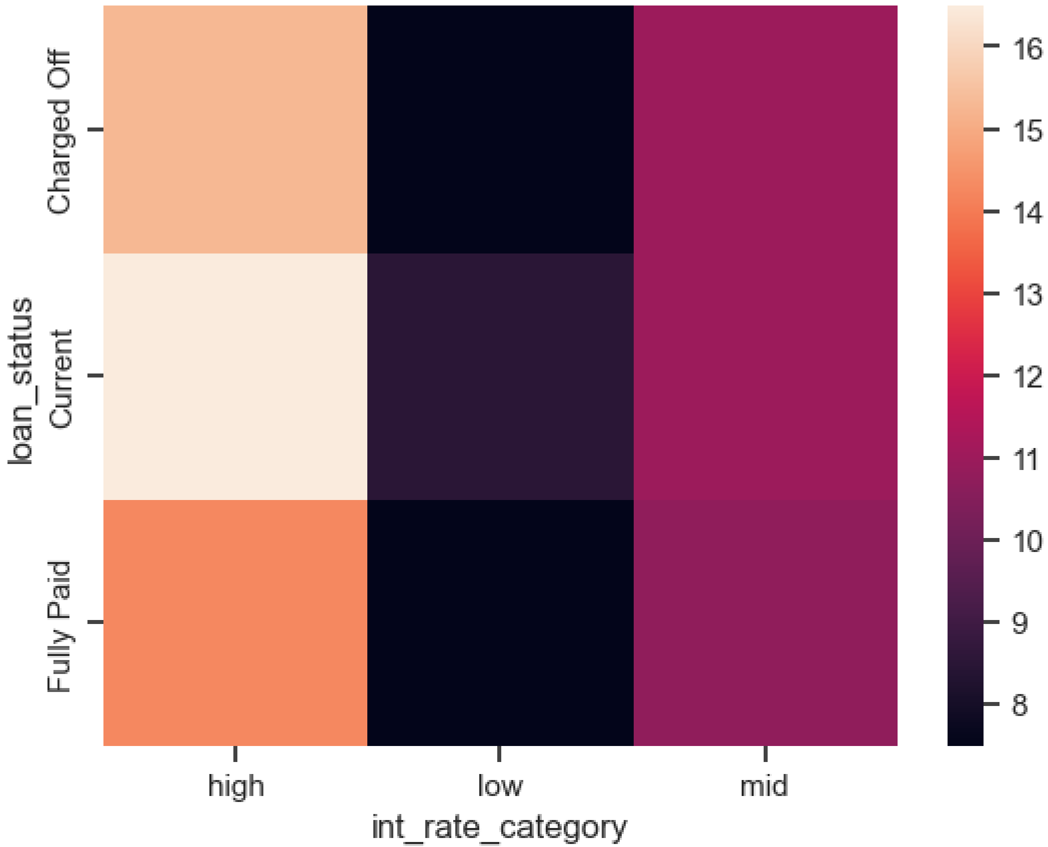
In [313]:
```
sns.heatmap(df.pivot_table(index=['loan_status'], columns=['int_rate_category'], values="loan_amnt", aggfunc=np.mediar
```

Out[313]: <AxesSubplot:xlabel='int_rate_category', ylabel='loan_status'>



In [314]:
```
sns.heatmap(df.pivot_table(index=['loan_status'], columns=['int_rate_category'], values="int_rate", aggfunc=np.median)
```

Out[314]: <AxesSubplot:xlabel='int_rate_category', ylabel='loan_status'>

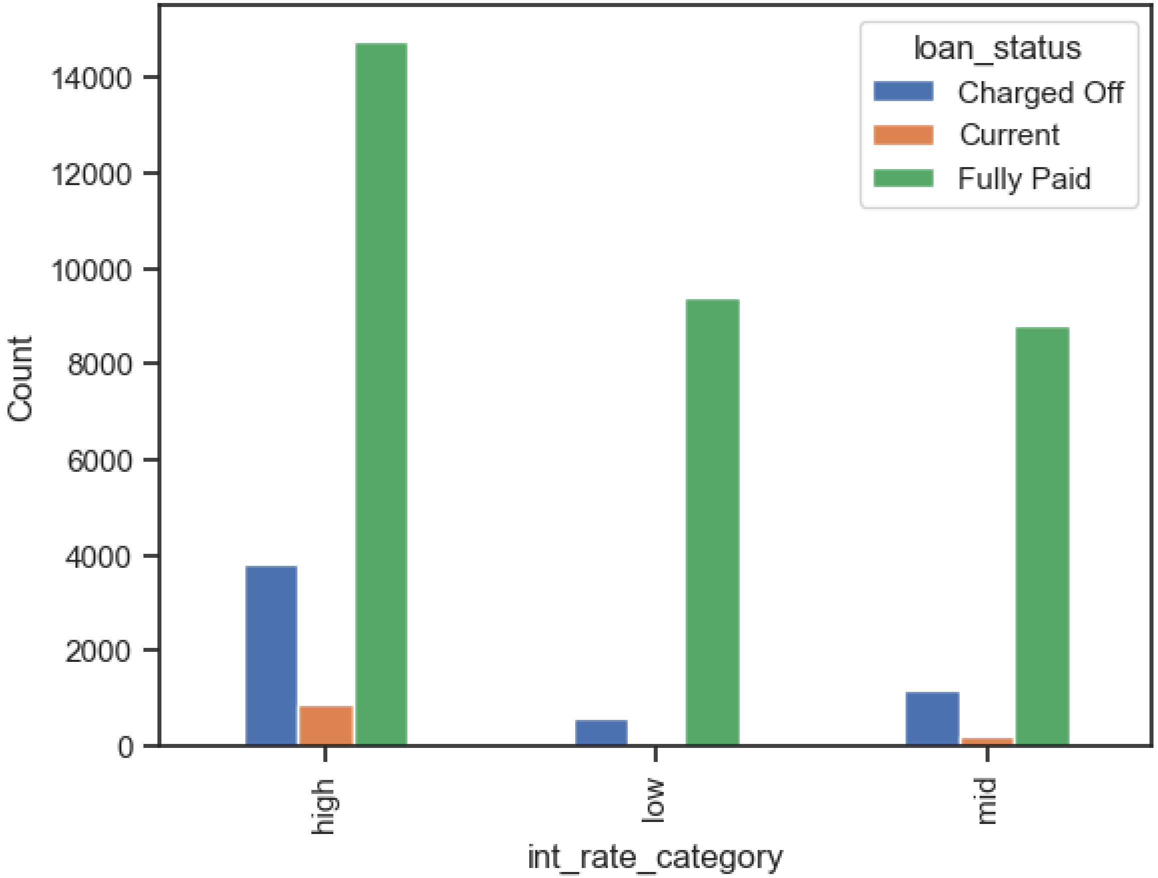


In [315]:
```
df.groupby(['int_rate_category', 'loan_status']).size()
```

Out[315]:
```
int_rate_category  loan_status
high               Charged Off     3835
                   Current          881
                   Fully Paid     14766
low                Charged Off      600
                   Current           42
                   Fully Paid      9385
mid                Charged Off     1192
                   Current          217
                   Fully Paid      8799
dtype: int64
```

In [316]:
```
plot = df.groupby(['int_rate_category', 'loan_status']).size().unstack().plot.bar()
plot.set_ylabel('Count')
```

Out[316]: Text(0, 0.5, 'Count')



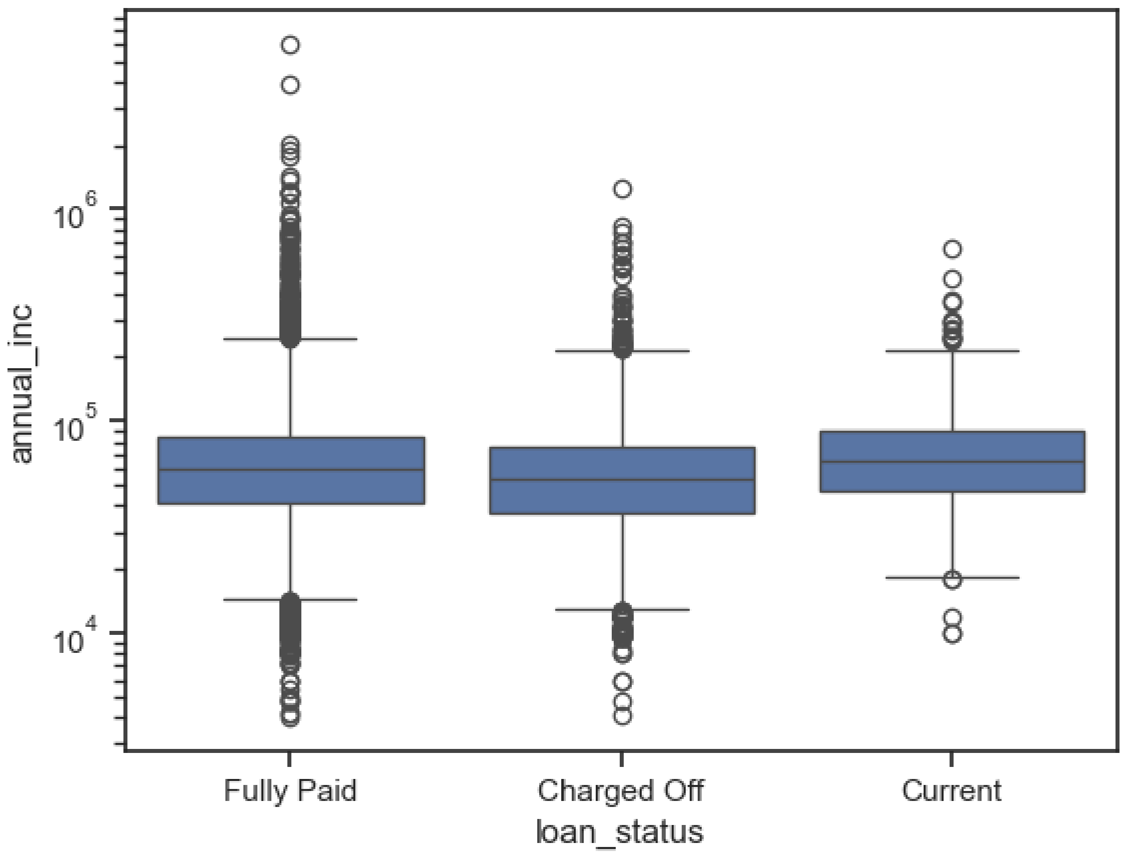
Observations

- When intrest rate is high, there are more number of people who default on loan
- Peple tend to take more amount, when interest rate is higher.

Bivariate with the `annual_inc` and `loan_status`

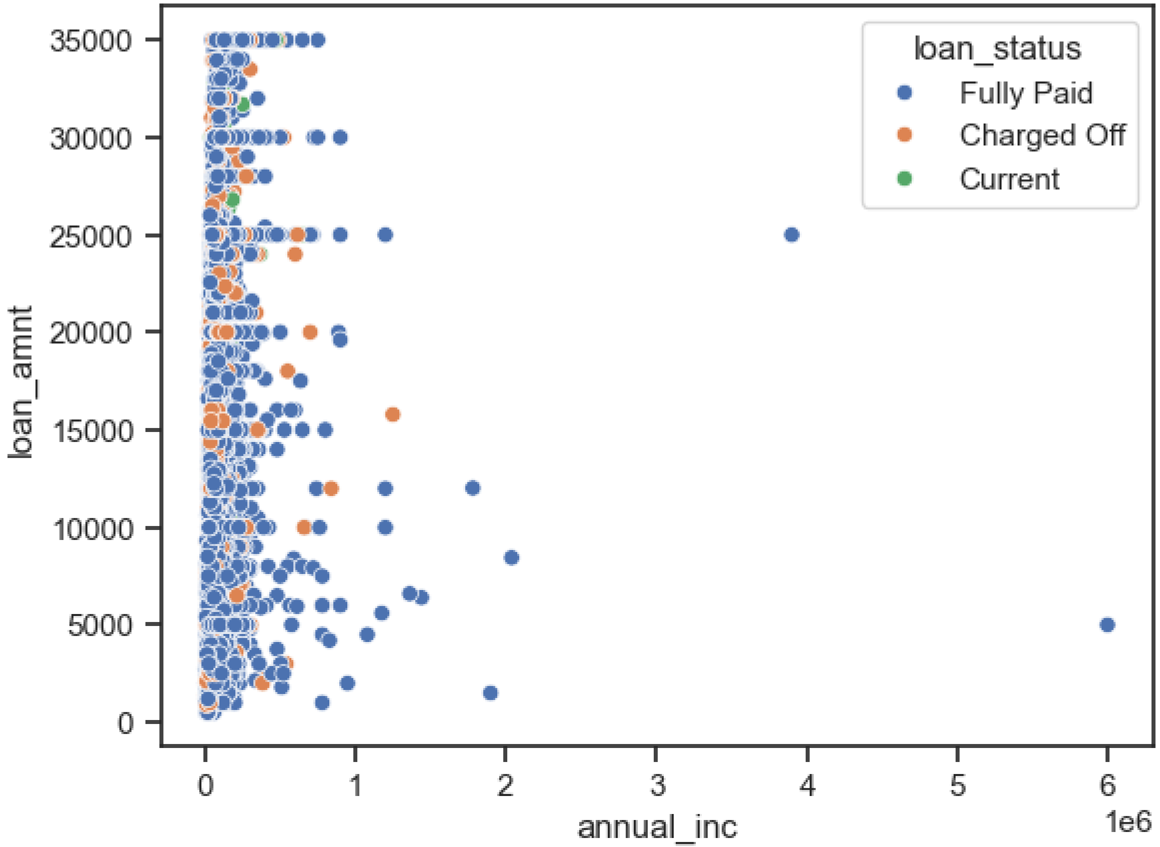
In [317]:
```
sns.boxplot(df, x='loan_status', y='annual_inc', log_scale=True)
```

Out[317]:
```
<AxesSubplot:xlabel='loan_status', ylabel='annual_inc'>
```



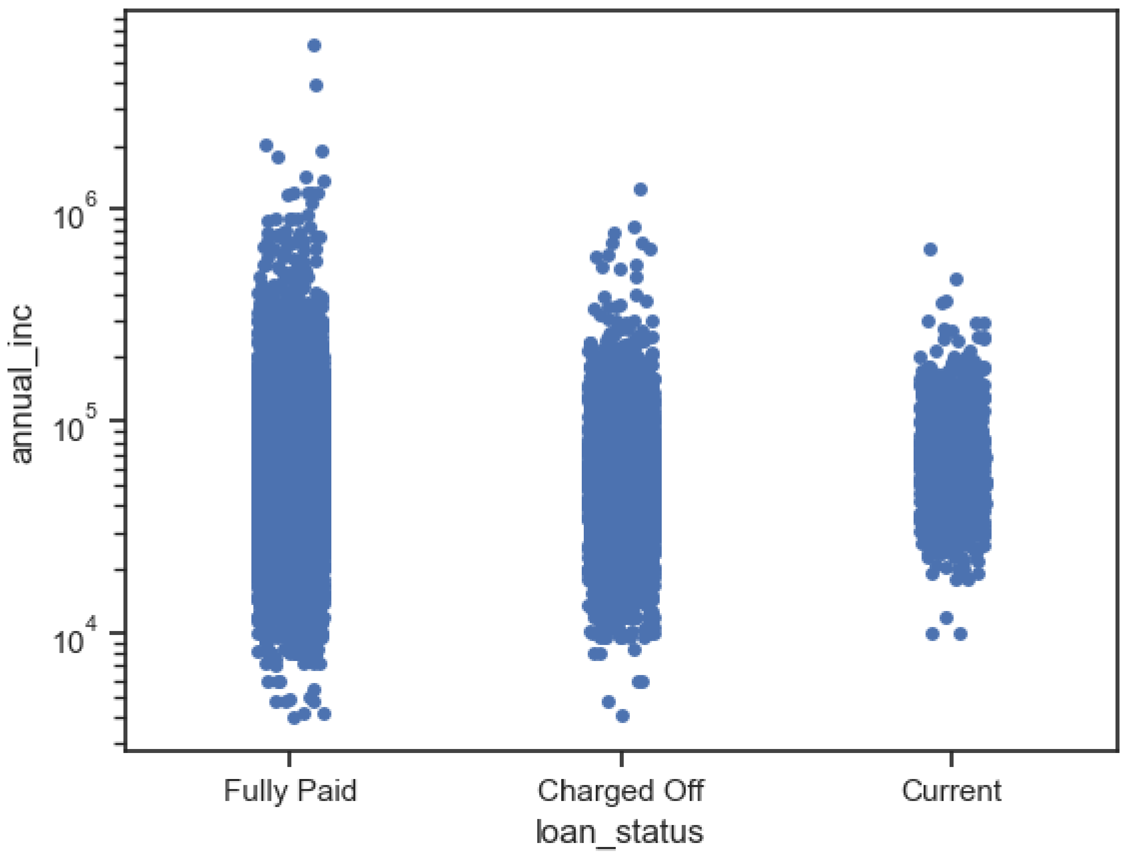
In [318]:
```
sns.scatterplot(df, x='annual_inc', y='loan_amnt', hue=df['loan_status'])
```

Out[318]: <AxesSubplot:xlabel='annual_inc', ylabel='loan_amnt'>



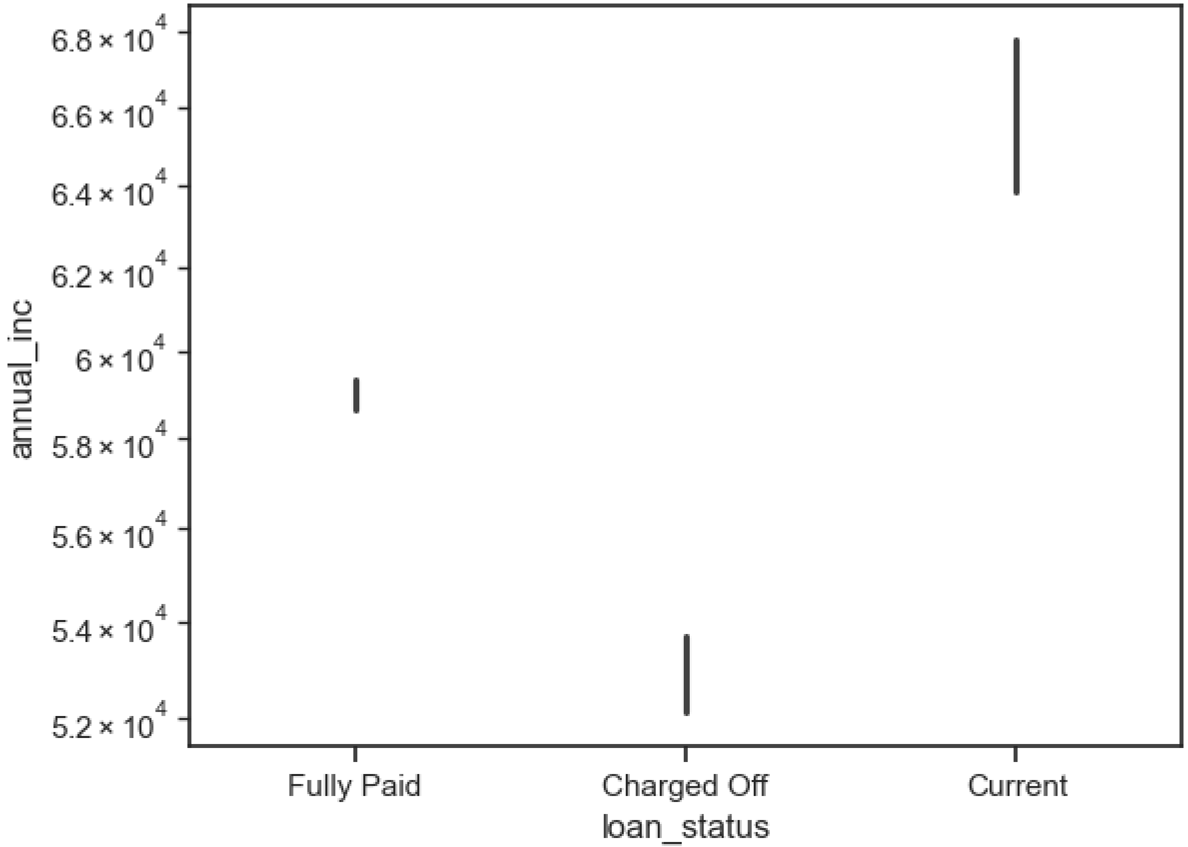
In [319]:
```
sns.stripplot(df, x='loan_status', y='annual_inc', log_scale=True)
```

Out[319]: <AxesSubplot:xlabel='loan_status', ylabel='annual_inc'>



In [320]:
```
sns.barplot(df, x='loan_status', y='annual_inc', log_scale=True)
```

Out[320]: <AxesSubplot:xlabel='loan_status', ylabel='annual_inc'>



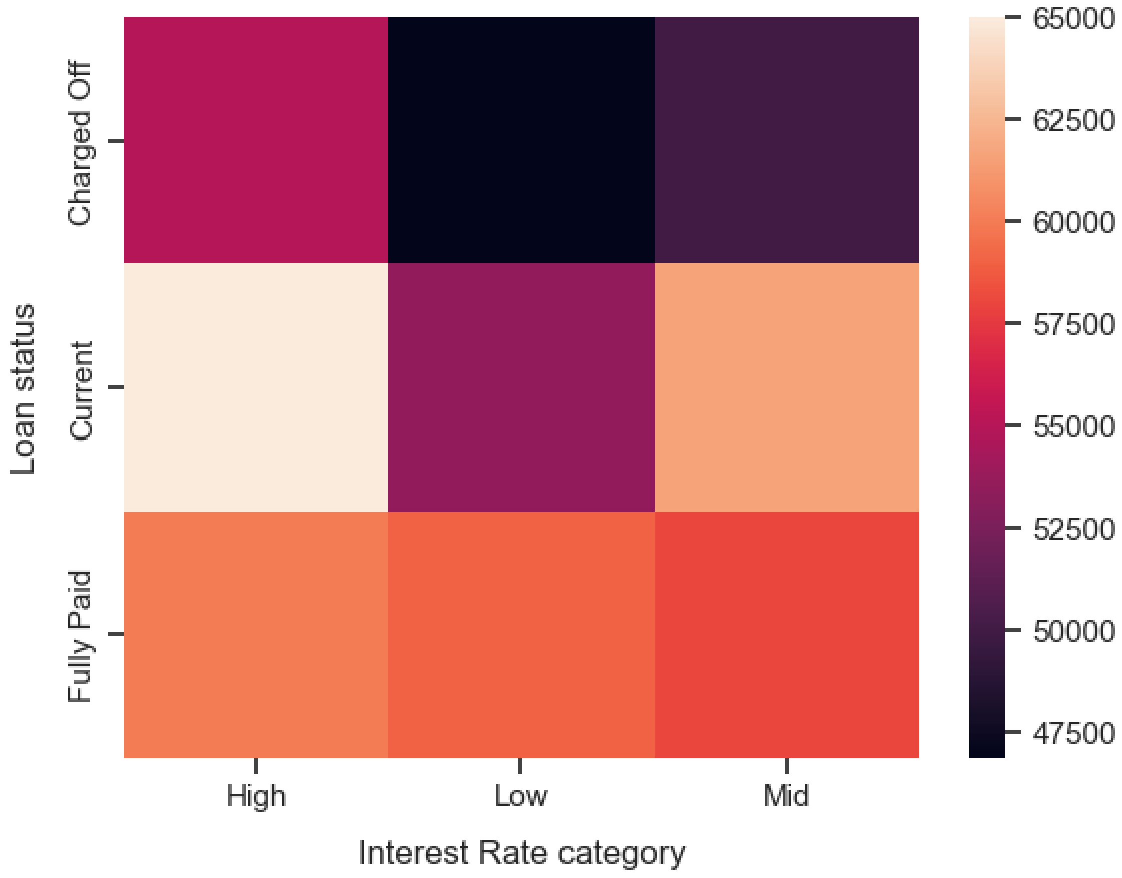
In [321]:
```
loan_status_int_annual_inc_table = df.pivot_table(index=['loan_status', 'int_rate_category'], values='annual_inc', agg
loan_status_pivot_table.style.background_gradient(cmap=cm)
```

Out[321]:

		member_id
home_ownership	**loan_status**	
MORTGAGE	**Charged Off**	2327
	Current	638
	Fully Paid	14694
NONE	**Fully Paid**	3
OTHER	**Charged Off**	18
	Fully Paid	80
OWN	**Charged Off**	443
	Current	83
	Fully Paid	2532
RENT	**Charged Off**	2839
	Current	419
	Fully Paid	15641

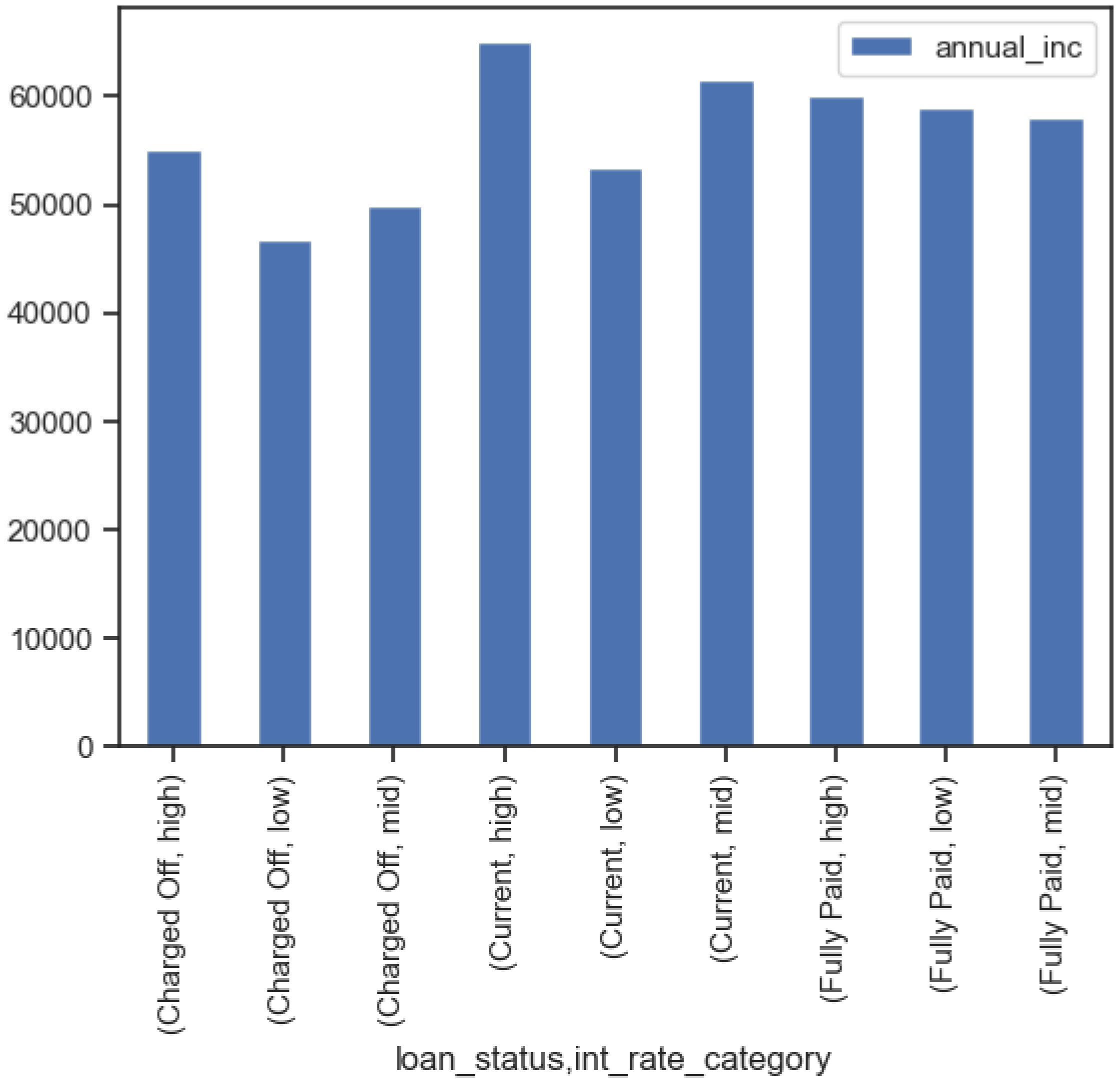
In [322]:
```
plot_axis = sns.heatmap(loan_status_int_annual_inc_table.unstack(), xticklabels=['High', 'Low', 'Mid'])
plot_axis.set_xlabel('Interest Rate category', labelpad=10)
plot_axis.set_ylabel('Loan status', labelpad=10)
```

Out[322]:
```
Text(47.24999999999999, 0.5, 'Loan status')
```



In [323]:
```
loan_status_int_annual_inc_table.plot.bar()
```

Out[323]: <AxesSubplot:xlabel='loan_status,int_rate_category'>



Observations

- When applicatants takes loan on high interest and having lower than median annual inc compared to the people who takes high interest loan and paids fully.

Bivariate analysis with the `loan_status` and `home_ownership`

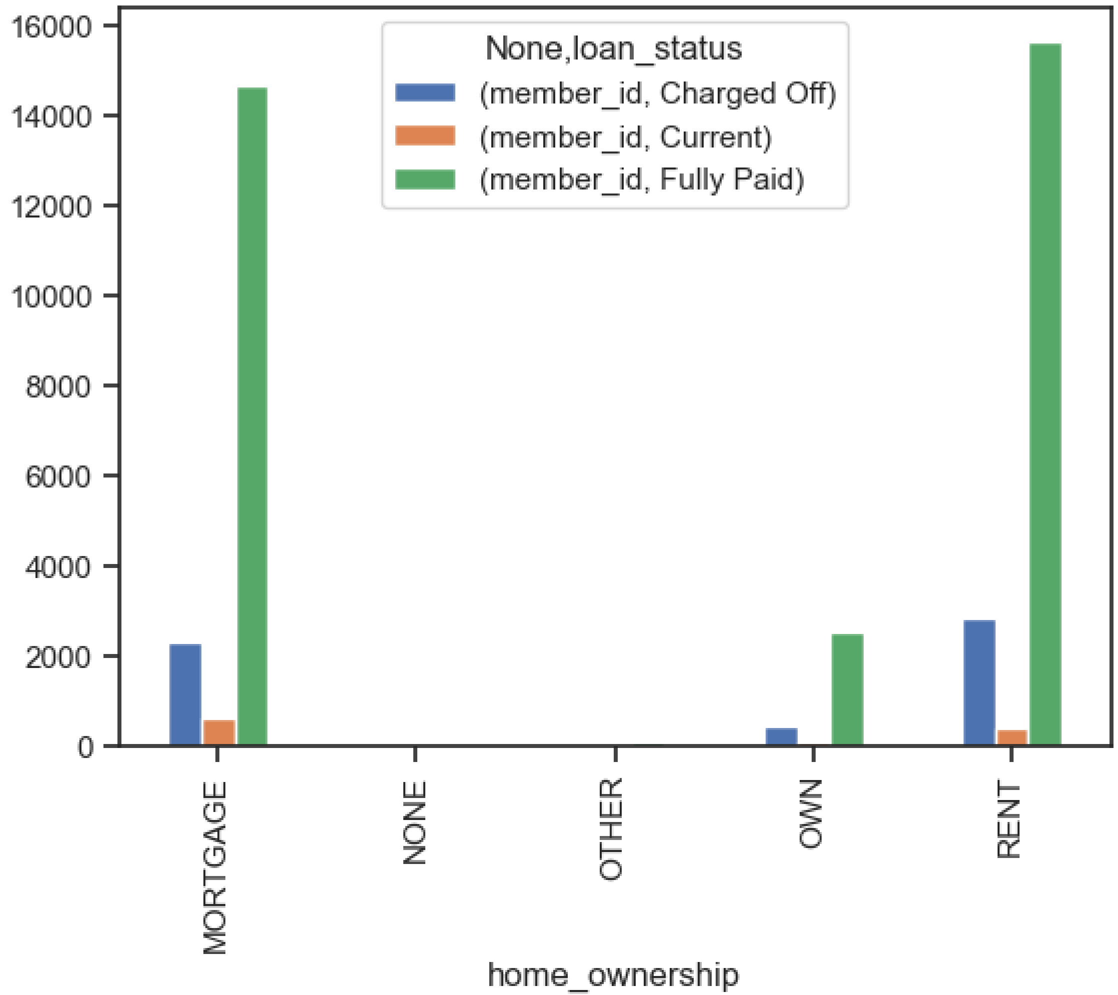
In [324]:
```
loan_status_pivot_table = df.pivot_table(index=['home_ownership', 'loan_status'], values="member_id", aggfunc='count')
loan_status_pivot_table.style.background_gradient(cmap=cm)
```

Out[324]:

home_ownership	loan_status	member_id
MORTGAGE	Charged Off	2327
	Current	638
	Fully Paid	14694
NONE	Fully Paid	3
OTHER	Charged Off	18
	Fully Paid	80
OWN	Charged Off	443
	Current	83
	Fully Paid	2532
RENT	Charged Off	2839
	Current	419
	Fully Paid	15641

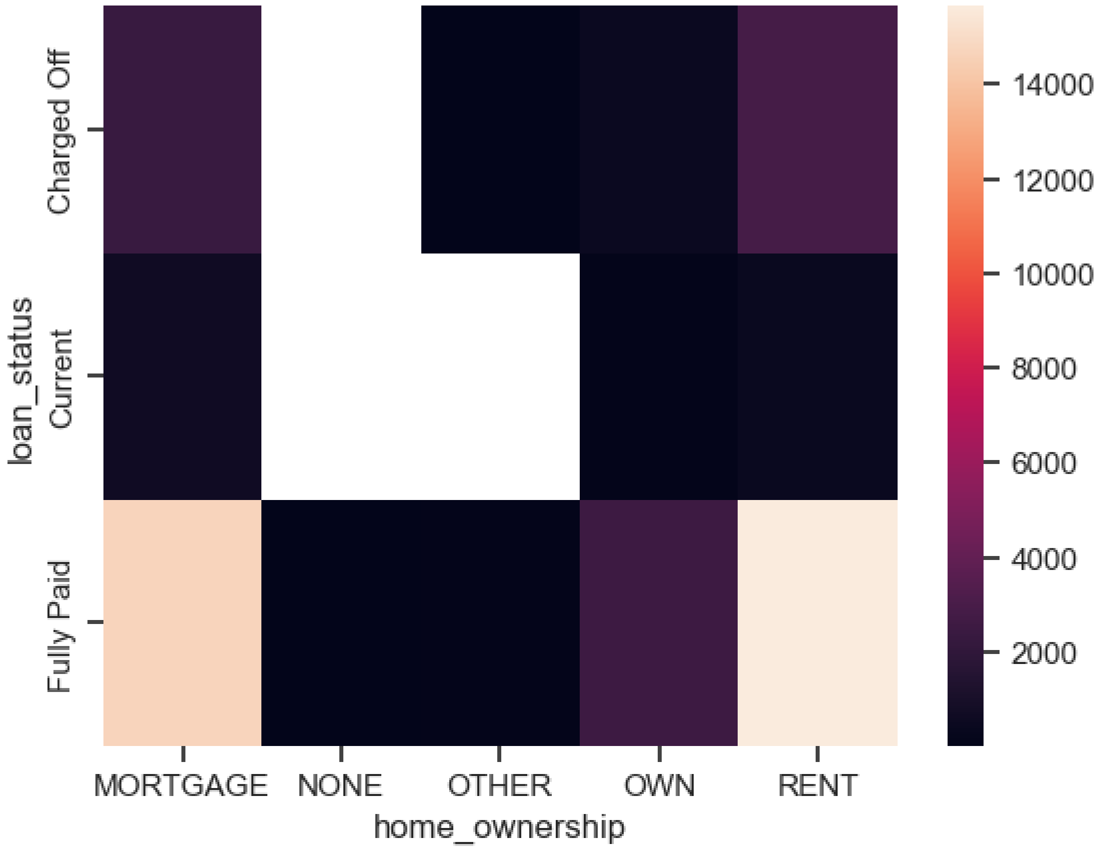
In [325]:
```
loan_status_pivot_table.unstack().plot.bar()
```

Out[325]: <AxesSubplot:xlabel='home_ownership'>



In [326]:
```
sns.heatmap(df.pivot_table(index=['loan_status'], columns=['home_ownership'], values="member_id", aggfunc='count'))
```

Out[326]: <AxesSubplot:xlabel='home_ownership', ylabel='loan_status'>



In [327]:
```
people_on_rent = df[df['home_ownership'].isin(('RENT', 'MORTGAGE'))]
people_on_rent.groupby(['loan_status', 'int_rate_category']).size()
```

Out[327]:
```
loan_status  int_rate_category
Charged Off  high                  3528
             low                    546
             mid                   1092
Current      high                   823
             low                     38
             mid                    196
Fully Paid   high                 13672
             low                   8565
             mid                   8098
dtype: int64
```

Observations

- People having house ownership equal to `RENT` tend to default on loan
- People having house ownership euqal to `RENT` and `MORTAGE` tend to take loan on `high` interest rate and tend to default more.

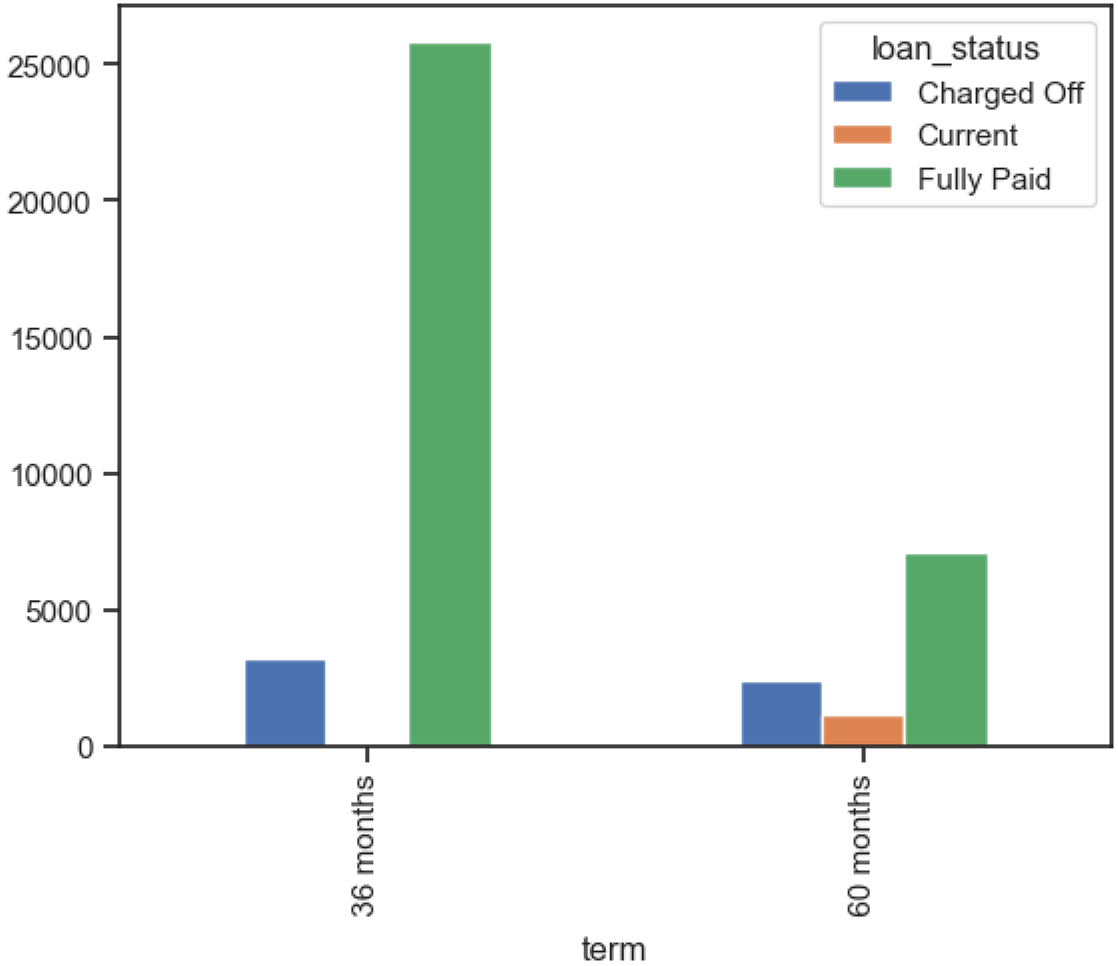
Bivariate analysis with the `loan_status` and installment `term`

In [328]:
```
df.groupby(['term', 'loan_status']).size()
```

Out[328]:
```
term        loan_status
 36 months  Charged Off     3227
            Fully Paid     25869
 60 months  Charged Off     2400
            Current         1140
            Fully Paid      7081
dtype: int64
```

In [329]:
```
df.groupby(['term', 'loan_status']).size().unstack().plot.bar()
```

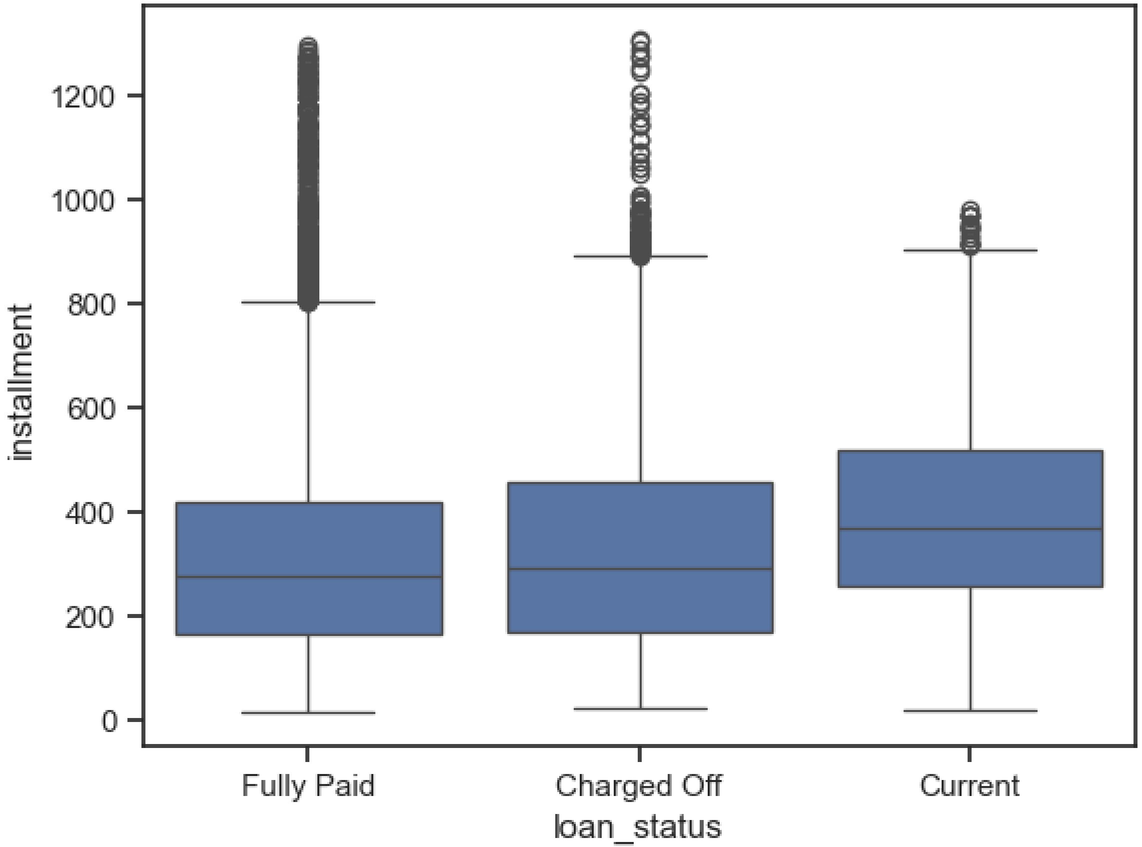
Out[329]: <AxesSubplot:xlabel='term'>



Bivariate analysis with the `loan_status` and `installment`

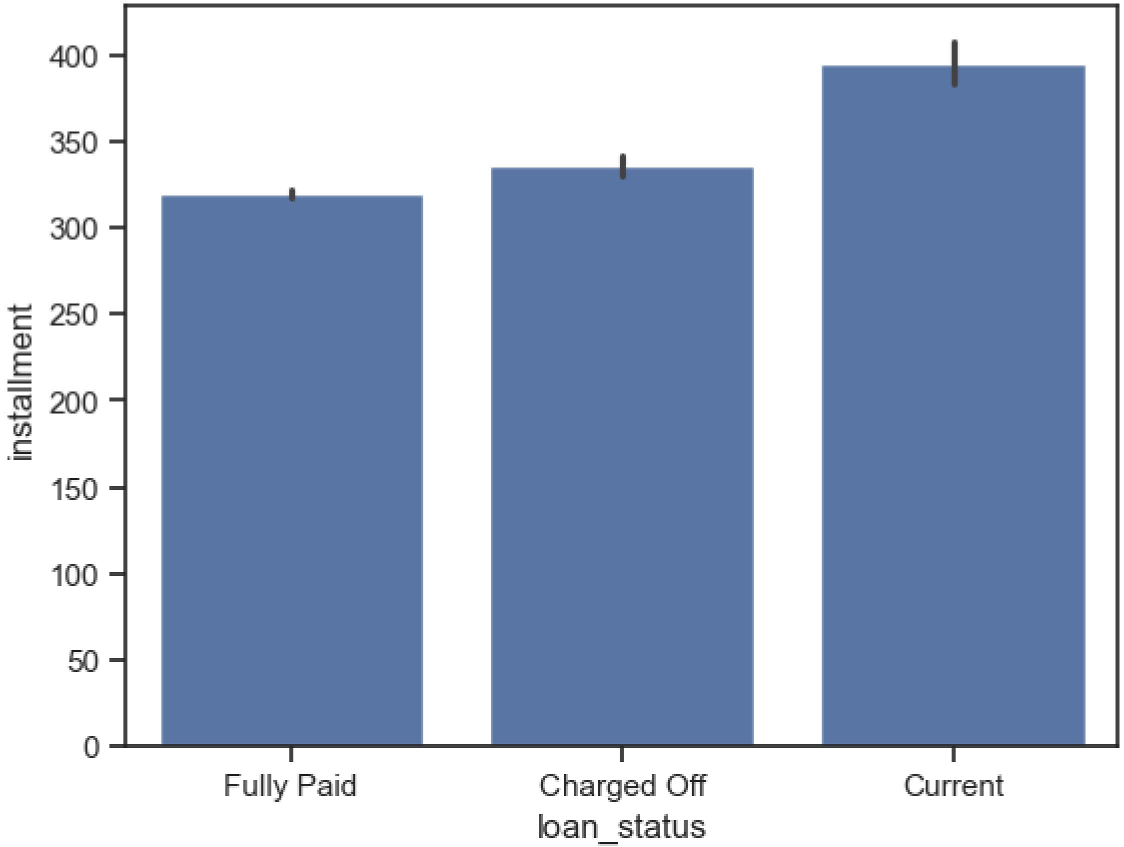
In [330]:
```
sns.boxplot(df, x='loan_status', y='installment', log_scale=False)
```

Out[330]:
```
<AxesSubplot:xlabel='loan_status', ylabel='installment'>
```



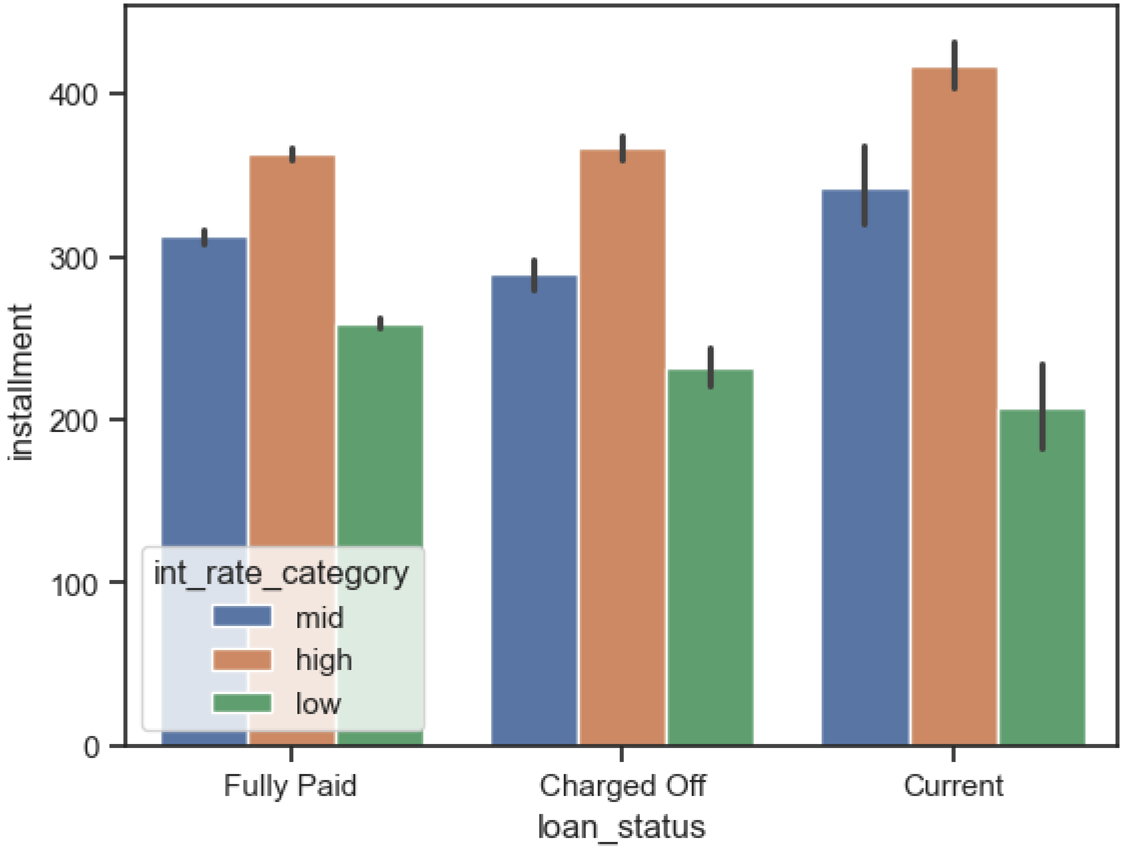
In [331]:
```
sns.barplot(df, x='loan_status', y='installment')
```

Out[331]:
```
<AxesSubplot:xlabel='loan_status', ylabel='installment'>
```



In [332]:
```
sns.barplot(df, x='loan_status', y='installment', hue=df['int_rate_category'])
```

Out[332]: <AxesSubplot:xlabel='loan_status', ylabel='installment'>



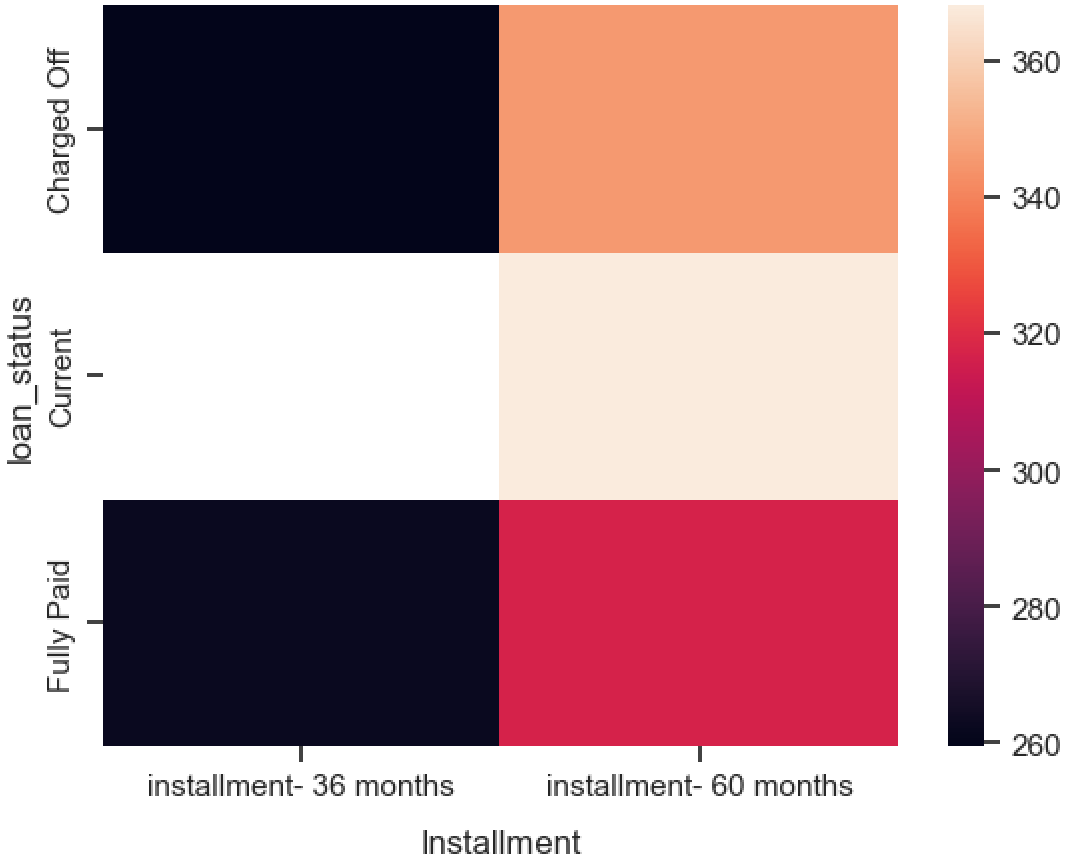
In [333]:
```
term_installment_loan_status_table = df.pivot_table(index=['loan_status'], columns=['term'], values=['installment'], a
term_installment_loan_status_table
```

Out[333]:

	installment	
term	**36 months**	**60 months**
loan_status		
Charged Off	259.57	345.59
Current	NaN	368.19
Fully Paid	262.21	317.04

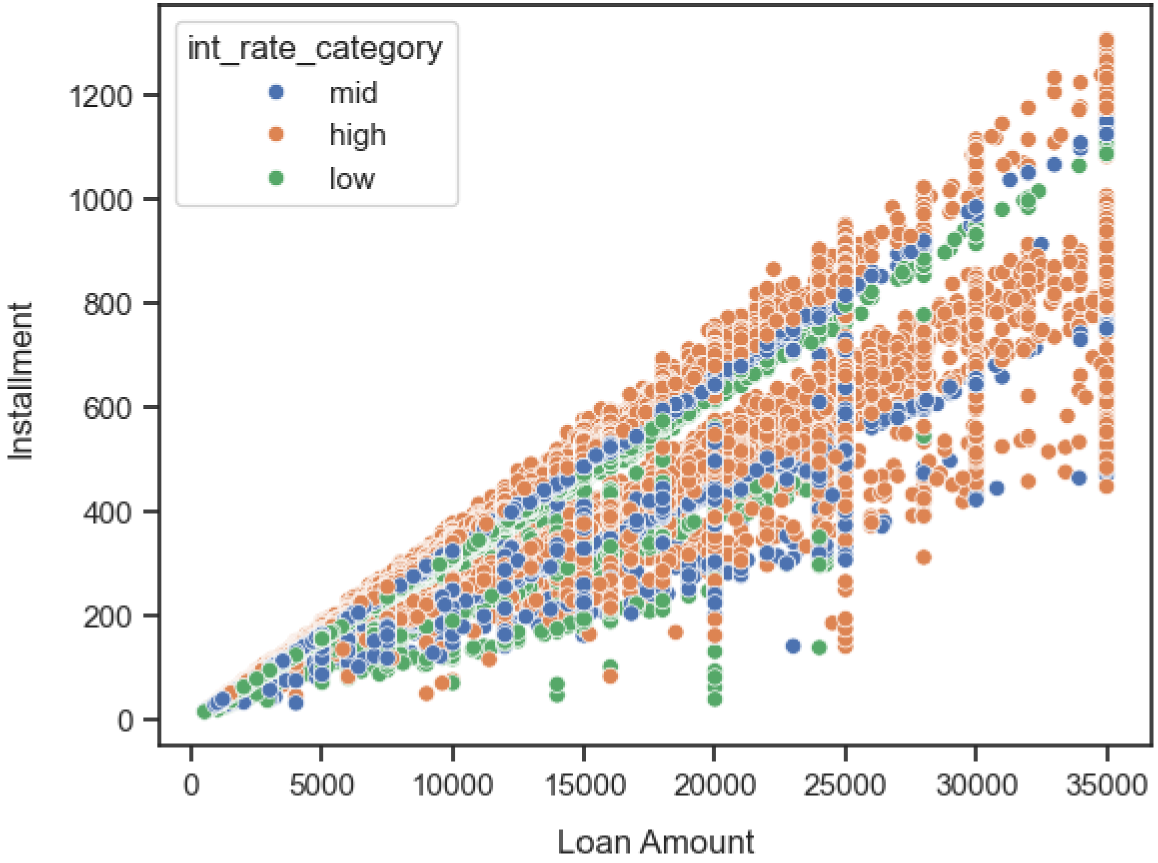
In [334]:
```
ax = sns.heatmap(term_installment_loan_status_table)
ax.set_xlabel('Installment', labelpad=10)
```

Out[334]:
```
Text(0.5, 20.049999999999997, 'Installment')
```

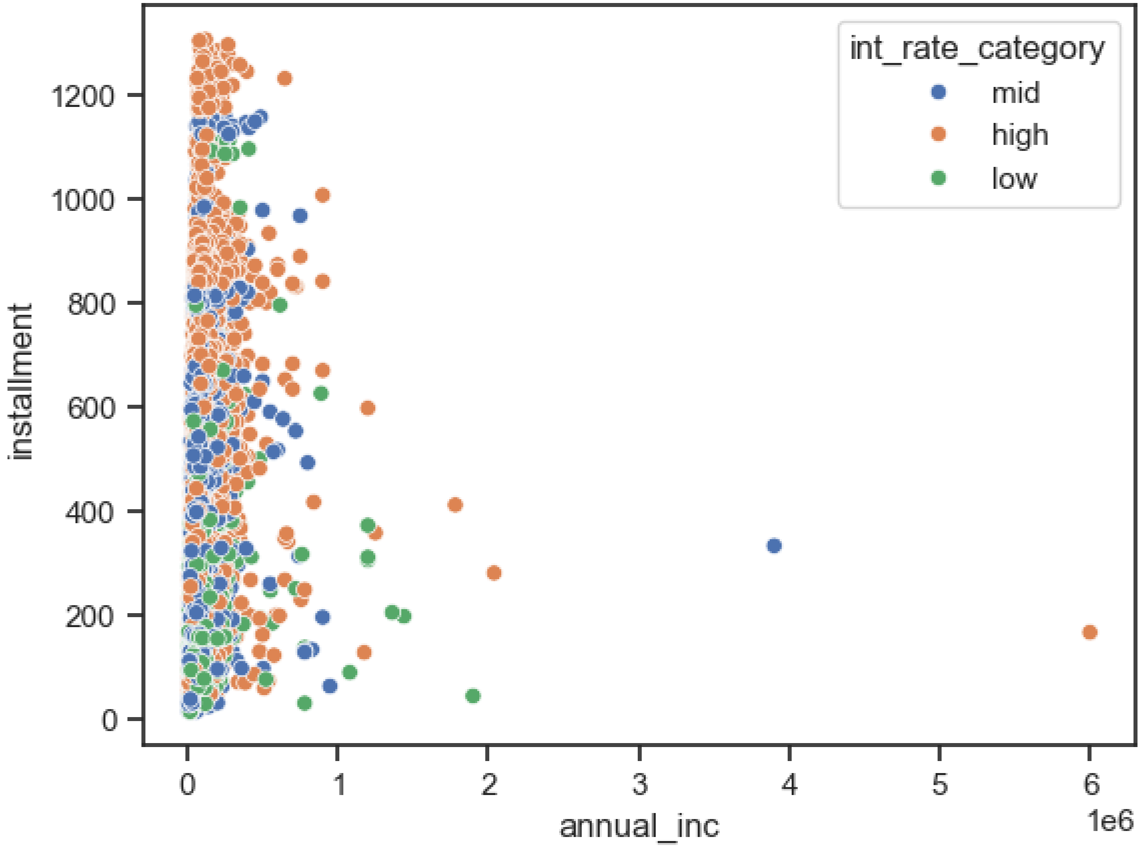


In [335]:
```
higher_term_charged_off = df[(df['term'] == ' 60 months') & df['loan_status'].isin(['Charged Off'])]
plot = sns.scatterplot(df, x='loan_amnt', y='installment', hue=df['int_rate_category'])
plot.set_xlabel('Loan Amount', labelpad=10)
plot.set_ylabel('Installment', labelpad=10)
```

Out[335]:
```
Text(0, 0.5, 'Installment')
```



In [336]:
```
plot = sns.scatterplot(df, x='annual_inc', y='installment', hue=df['int_rate_category'])
```



Observations

- When people take loan on `60 months` term and higher installment amount tends to default more.
- People take loan on `60 months` term and higher interest rate and high installment payment and tend default more.

Bivariate Analysis of `loan_status` and `purpose`

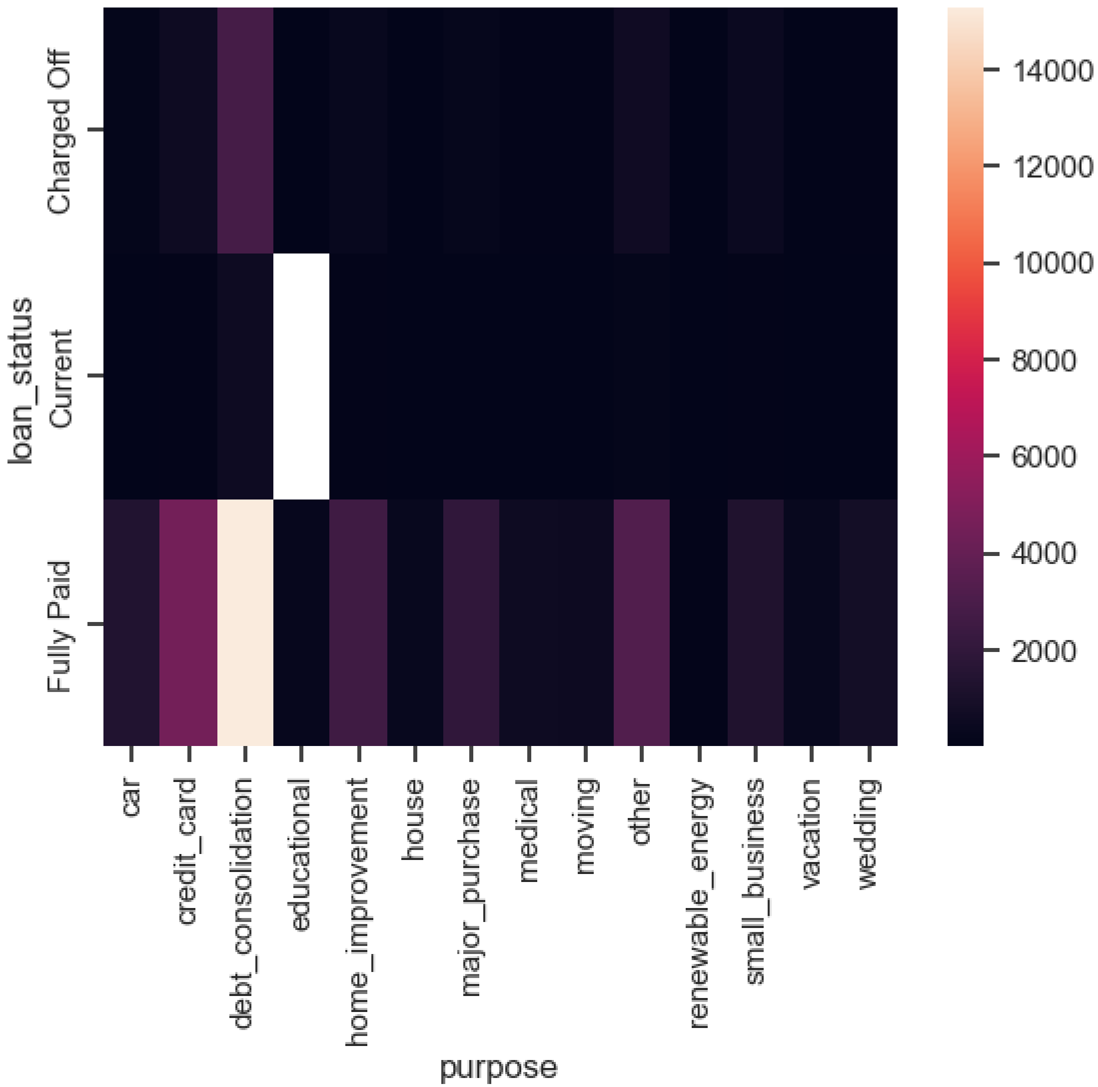
In [337]:
```
loan_status_purpose_table = df.pivot_table(index=['loan_status'], columns=['purpose'], values='member_id', aggfunc='co
loan_status_purpose_table.unstack()
loan_status_purpose_table
```

Out[337]:

purpose	car	credit_card	debt_consolidation	educational	home_improvement	house	major_purchase	medical	moving	other	renewable
loan_status											
Charged Off	160.0	542.0	2767.0	56.0	347.0	59.0	222.0	106.0	92.0	633.0	
Current	50.0	103.0	586.0	NaN	101.0	14.0	37.0	12.0	7.0	128.0	
Fully Paid	1339.0	4485.0	15288.0	269.0	2528.0	308.0	1928.0	575.0	484.0	3232.0	

In [338]:
```
sns.heatmap(loan_status_purpose_table)
```

Out[338]: <AxesSubplot:xlabel='purpose', ylabel='loan_status'>



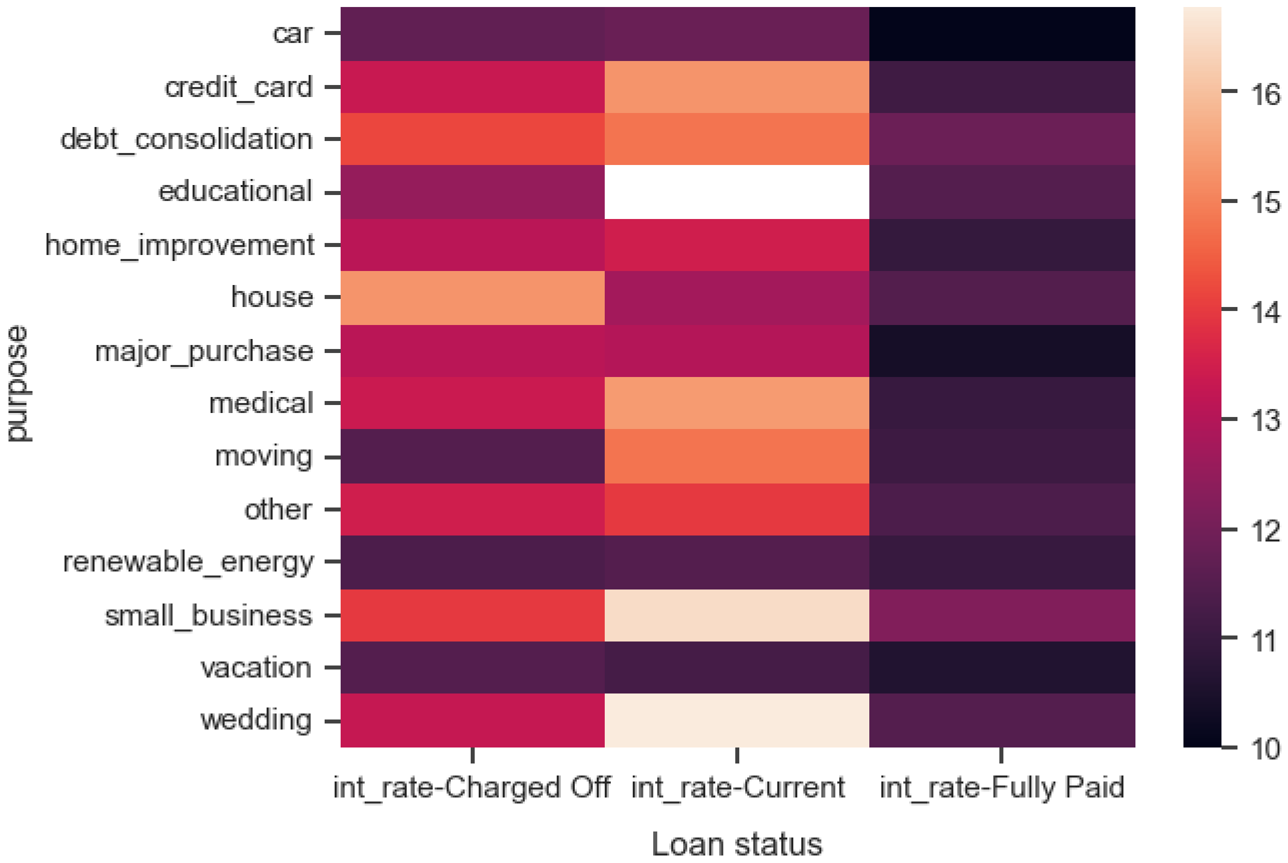
`loan_status` , `purpose` , and `int_rate` analysis

In [339]:
```
table = df.pivot_table(index=['purpose', 'loan_status'], values=['int_rate'], aggfunc=np.median)
plot = sns.heatmap(table.unstack())

plot.set_xlabel('Loan status', labelpad=10)
table
```

Out[339]:

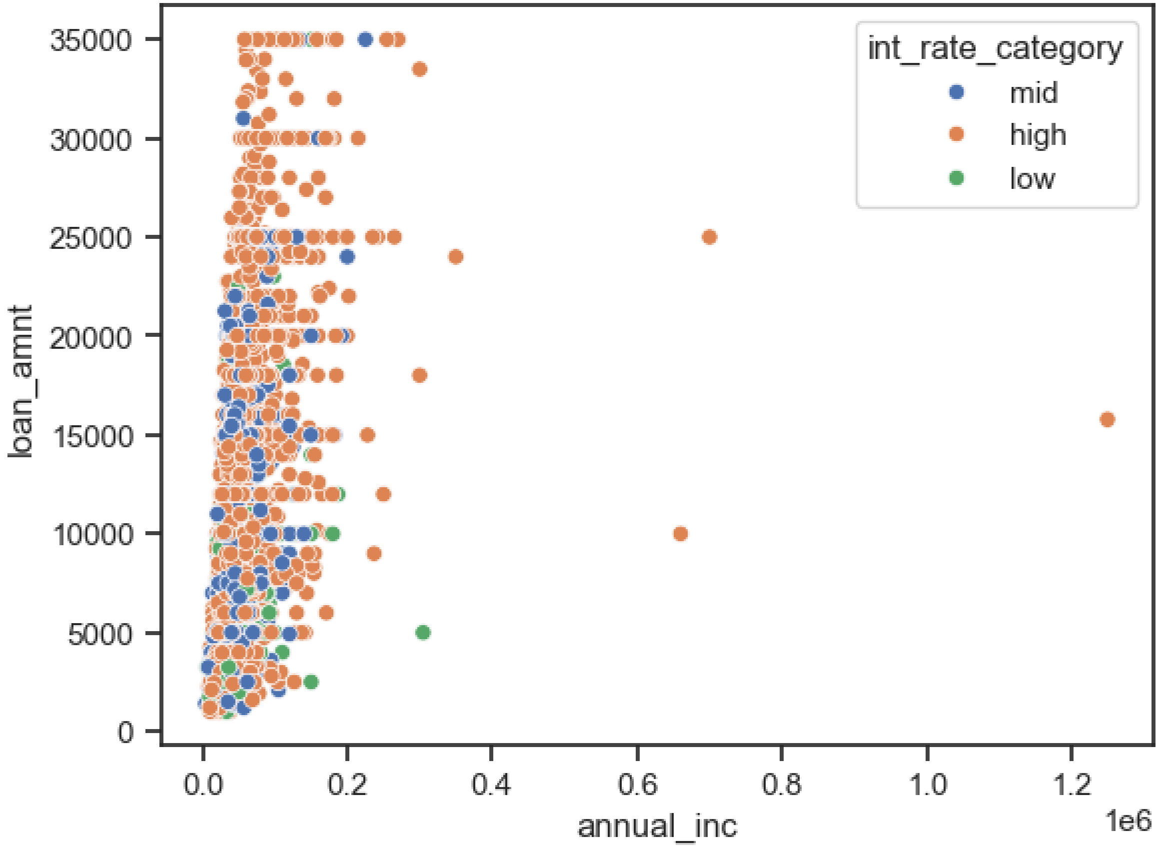
purpose	loan_status	int_rate
car	Charged Off	11.710
	Current	11.850
	Fully Paid	10.000
credit_card	Charged Off	13.350
	Current	15.270
	Fully Paid	11.120
debt_consolidation	Charged Off	14.170
	Current	14.790
	Fully Paid	11.860
educational	Charged Off	12.530
	Fully Paid	11.490
home_improvement	Charged Off	13.110
	Current	13.490
	Fully Paid	10.950
house	Charged Off	15.270
	Current	12.740
	Fully Paid	11.475
major_purchase	Charged Off	13.110
	Current	12.990
	Fully Paid	10.380
medical	Charged Off	13.360
	Current	15.380
	Fully Paid	10.990

		int_rate
purpose	**loan_status**	
moving	**Charged Off**	11.490
	Current	14.790
	Fully Paid	11.110
other	**Charged Off**	13.490
	Current	13.990
	Fully Paid	11.360
renewable_energy	**Charged Off**	11.360
	Current	11.490
	Fully Paid	10.990
small_business	**Charged Off**	13.990
	Current	16.490
	Fully Paid	12.210
vacation	**Charged Off**	11.490
	Current	11.240
	Fully Paid	10.590
wedding	**Charged Off**	13.290
	Current	16.770
	Fully Paid	11.490


car
credit_card
debt_consolidation
educational
home_improvement
house
major_purchase
medical
moving
other
renewable_energy
small_business
vacation
wedding
purpose
int_rate-Charged Off
int_rate-Current
int_rate-Fully Paid
Loan status
16
15
14
13
12
11
10

In [340]:
```
charged_off_debt_consolidation = df[(df['loan_status'].isin(['Charged Off'])) & df['purpose'].isin(['debt_consolidatic
sns.scatterplot(charged_off_debt_consolidation, x='annual_inc', y='loan_amnt', hue=df['int_rate_category'])
```

Out[340]: <AxesSubplot:xlabel='annual_inc', ylabel='loan_amnt'>



Observations

- People default more when purpose is `debt_consolidation` and tend to take loan on higher int_rate, even when income is on lower side.

Bivariate analysis for `verification_status` and `loan_status`

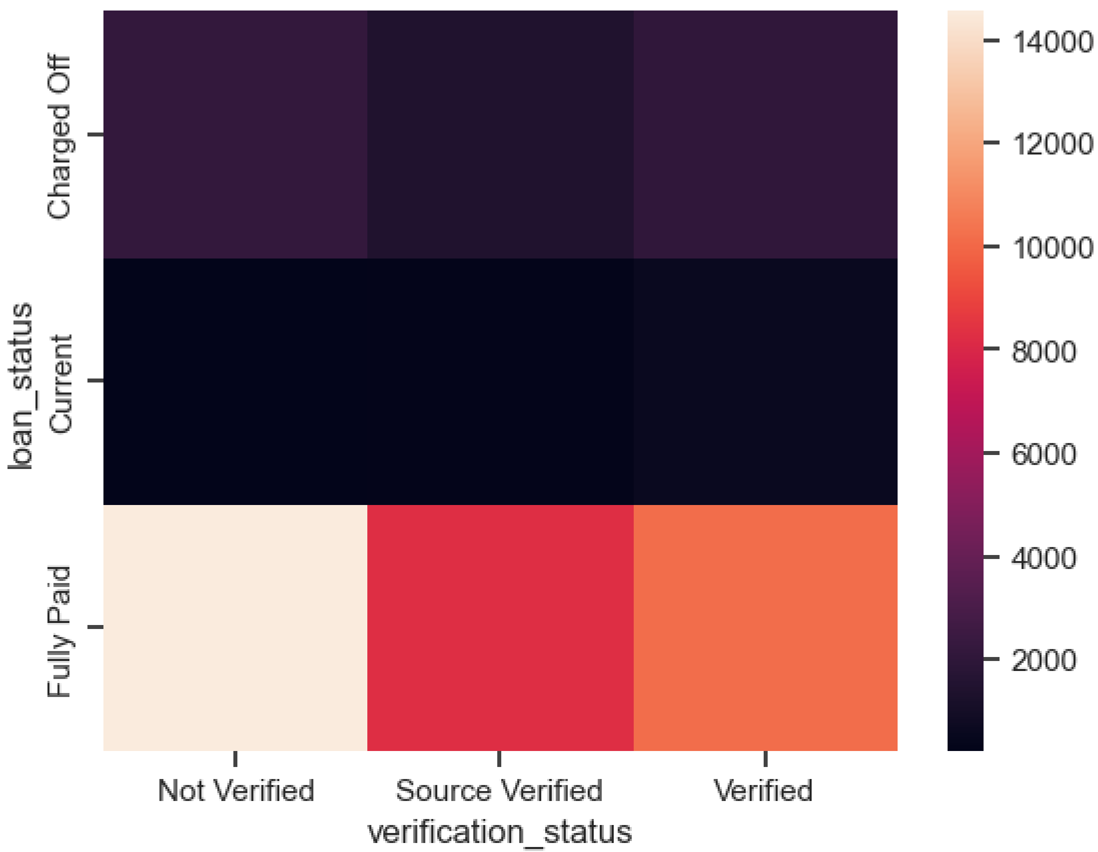
In [341]:
```
verification_and_loan_status_table = df.pivot_table(index=['loan_status'], values='member_id', columns='verification_s
verification_and_loan_status_table
```

Out[341]:

verification_status	Not Verified	Source Verified	Verified
loan_status			
Charged Off	2142	1434	2051
Current	227	310	603
Fully Paid	14552	8243	10155

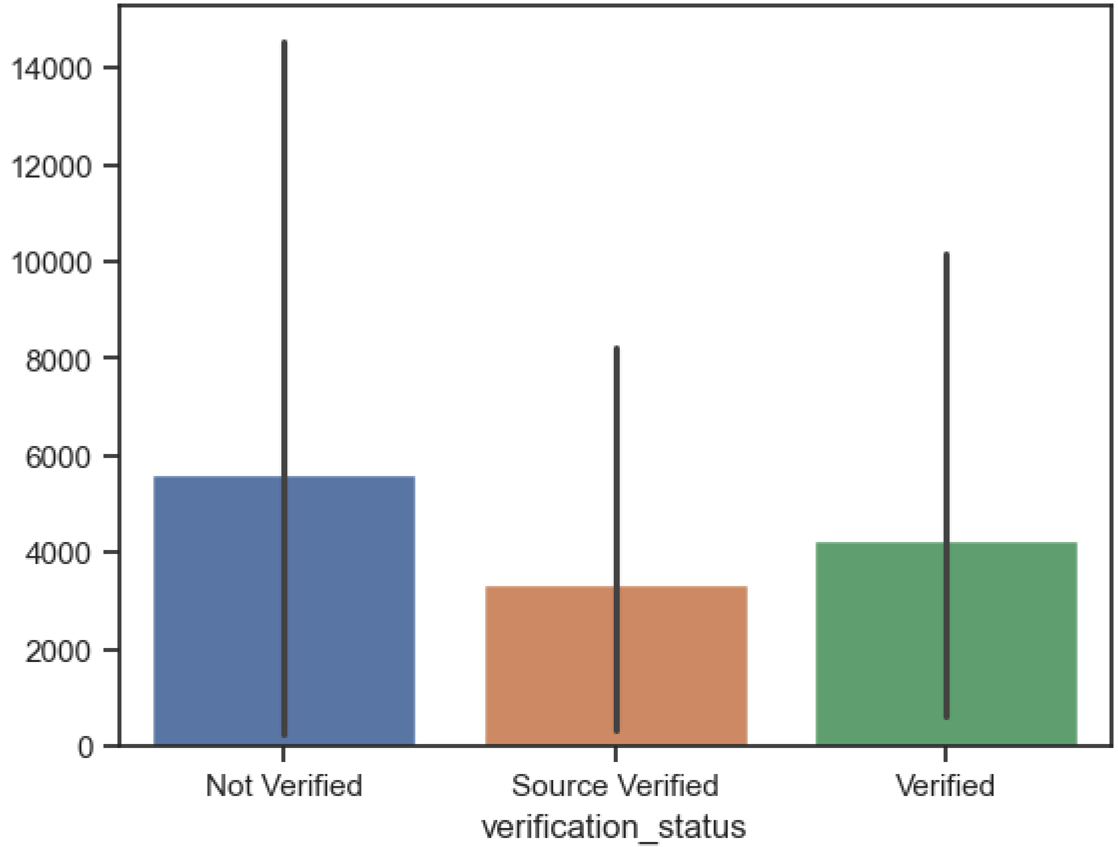
In [342]:
```
sns.heatmap(verification_and_loan_status_table)
```

Out[342]:
```
<AxesSubplot:xlabel='verification_status', ylabel='loan_status'>
```



In [343]:
```
sns.barplot(verification_and_loan_status_table)
```

Out[343]: <AxesSubplot:xlabel='verification_status'>



Multi Variate analysis

Of `loan_status` with other columns like `installment` , `term` , `annual_inc` , `home_ownership` , `int_rate` , `purpose` , `loan_amount` , `int_rate_category` , `verification_status`

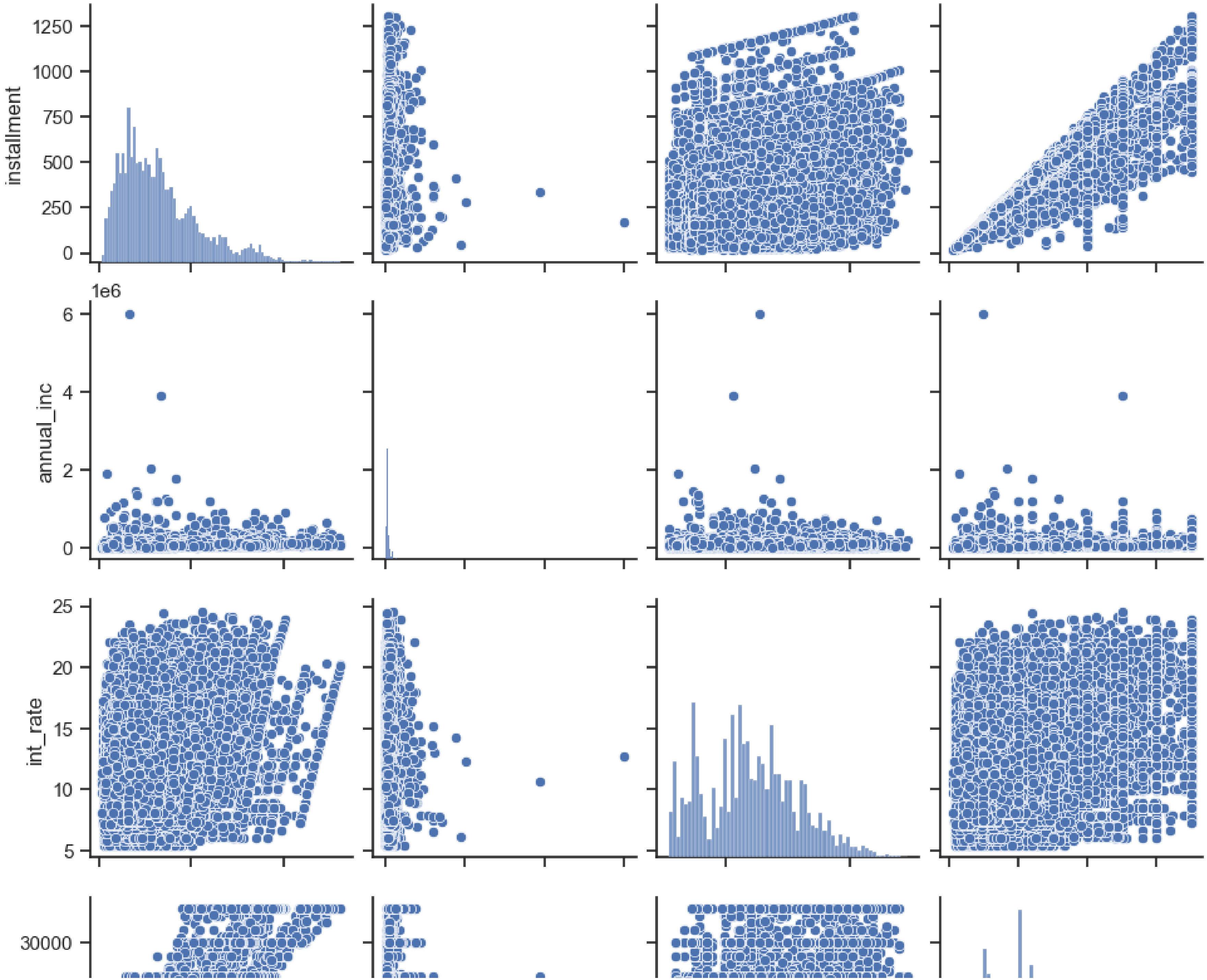
In [344]:
```
df_subset = df[['loan_status', 'installment', 'term', 'annual_inc', 'home_ownership', 'int_rate', 'purpose', 'loan_amn
df_subset.head()
```

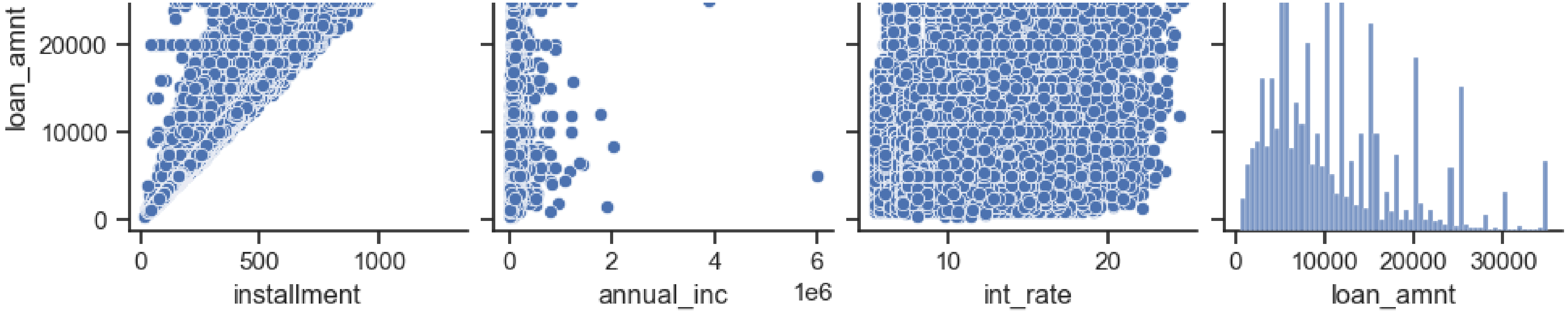
Out[344]:

	loan_status	installment	term	annual_inc	home_ownership	int_rate	purpose	loan_amnt	int_rate_category	verification_status
0	Fully Paid	162.87	36 months	24000.0	RENT	10.65	credit_card	5000	mid	Verified
1	Charged Off	59.83	60 months	30000.0	RENT	15.27	car	2500	high	Source Verified
2	Fully Paid	84.33	36 months	12252.0	RENT	15.96	small_business	2400	high	Not Verified
3	Fully Paid	339.31	36 months	49200.0	RENT	13.49	other	10000	high	Source Verified
4	Current	67.79	60 months	80000.0	RENT	12.69	other	3000	high	Source Verified

In [345]:
```
sns.pairplot(df_subset)
```

Out[345]:
```
<seaborn.axisgrid.PairGrid at 0x7fbc14eaf460>
```


installment
1250
1000
750
500
250
0
1e6
annual_inc
6
4
2
0
int_rate
25
20
15
10
5
30000



loan_status , int_category , purpose , home_ownership

In [346]:
```
table = df.pivot_table(index=['loan_status', 'purpose', 'int_rate_category', 'term', 'home_ownership'], values=['annua
```

In [347]:
```
table.style.background_gradient(cmap=cm)
```

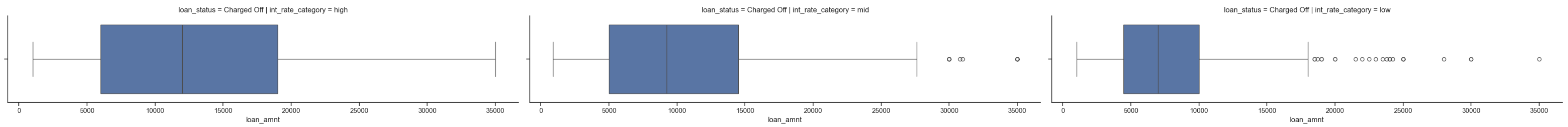
Out[347]:

					annual_inc	loan_amnt
loan_status	**purpose**	**int_rate_category**	**term**	**home_ownership**		
				MORTGAGE	54902.520000	6000.000000
			36 months	**OTHER**	37800.000000	10000.000000
				OWN	57000.000000	9000.000000
		high		**RENT**	45314.000000	5000.000000
				MORTGAGE	57000.000000	5600.000000
			60 months	**OWN**	38400.000000	3600.000000
				RENT	52500.000000	6000.000000
				MORTGAGE	67500.000000	5150.000000
	car		**36 months**	**OWN**	30900.000000	4775.000000
		low		**RENT**	39000.000000	6600.000000
			60 months	**MORTGAGE**	39000.000000	9800.000000

In [348]:
```
plot = sns.FacetGrid(df[df['loan_status'] == 'Charged Off'], row='loan_status', col='int_rate_category', height=3, asp
plot.map(sns.boxplot, "loan_amnt")
```

```
/Users/shaifali.jangra/opt/anaconda3/lib/python3.9/site-packages/seaborn/axisgrid.py:718: UserWarning: Using the boxp
lot function without specifying `order` is likely to produce an incorrect plot.
  warnings.warn(warning)
```

Out[348]: <seaborn.axisgrid.FacetGrid at 0x7fbc1bd29880>



Recommendations

1. Do more background check when when purpose is `debt_consolidation` and ready to take loan on higher int_rate, even when income is on lower side.

    - When purpose is `debt_consolidation` , people do full payment when interest rate is lower.
    - When purpose is `debt_consolidation` , people do better, full payment rate is higher when `loan amount` and `annual_inc` is lower.
2. When `income is low` , and people ready to take on `high` interest and high installment amount, they tend to default more, do more background check.
3. When people living on `RENT` tend to default more as they are ready to take loan on `high` interest rate, and `high` installment amount, and have low `annual_inc` .

    - Pople living on `RENT` do better when interate rate is `low` and installment amount is median of the current dataset.