Lending Club Case Study

Problem Statement

- This company is the largest online loan marketplace, facilitating personal loans, business loans, and financing of medical procedures. Borrowers can easily access lower interest rate loans through a fast online interface.
- Like most other lending companies, lending loans to ‘risky’ applicants is the largest source of financial loss (called credit loss). Credit loss is the amount of money lost by the lender when the borrower refuses to pay or runs away with the money owed. In other words, borrowers who default cause the largest amount of loss to the lenders. In this case, the customers labelled as 'charged-off' are the 'defaulters'.
- If one is able to identify these risky loan applicants, then such loans can be reduced thereby cutting down the amount of credit loss.

Abstract

Credit risk is something peer-to-peer (P2P) lending investors must carefully consider when making informed investment decisions; it is the risk of default as a result of borrowers failing to make required payments, leading to loss of principal and interest.In this project, we build predicting modelhistorical loan data that help investors quantify credit risks using numpy and pandas libraries predicting whether a given loan will be fully paid or not.

Import packages are used

In [80]:

```
import pandas as pd
import numpy as np
import seaborn as sea
import warnings
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', 60)
pd.set_option('display.max_rows', 100)
pd.set_option('display.float_format', lambda x: '%.3f' % x)
#ignore warning
def ignore_warn(*args, **kwargs):
    pass

# ignore annoying warning if any
warnings.warn = ignore_warn
```

loading the loan data

In [81]:

```
loan = pd.read_csv('Loan.csv')
loan.shape
```

Out[81]:

```
(39717, 111)
```

In [82]:

```
# Show the first five rows
loan.head()
```

Out[82]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment
0	1077501	1296599	5000	5000	4975.000	36 months	10.65%	162.870
1	1077430	1314167	2500	2500	2500.000	60 months	15.27%	59.830
2	1077175	1313524	2400	2400	2400.000	36 months	15.96%	84.330
3	1076863	1277178	10000	10000	10000.000	36 months	13.49%	339.310
4	1075358	1311748	3000	3000	3000.000	60 months	12.69%	67.790

5 rows × 111 columns

In [83]:

```
#Shape from the original dataset
print('Number of rows    :',loan.shape[0])
print('Number of columns:',loan.shape[1])
```

```
Number of rows    : 39717
Number of columns: 111
```

In [84]:

```
loan.isnull().sum()
```

Out[84]:

```
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
```

As per data observation more columns contains all nulls values so that these columns data isn ot useful our analysis so removing

In [85]:

```
loan.dropna(axis=1,how="all",inplace=True)
loan.head()
```

Out[85]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installmen
0	1077501	1296599	5000	5000	4975.000	36 months	10.65%	162.87
1	1077430	1314167	2500	2500	2500.000	60 months	15.27%	59.83
2	1077175	1313524	2400	2400	2400.000	36 months	15.96%	84.33
3	1076863	1277178	10000	10000	10000.000	36 months	13.49%	339.31
4	1075358	1311748	3000	3000	3000.000	60 months	12.69%	67.79

Some columns contains single value;so removing those columns.

In [86]:

```
loan.drop(['pymnt_plan', "initial_list_status",'collections_12_mths_ex_med','polic
loan.head()
```

Out[86]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installmen
0	1077501	1296599	5000	5000	4975.000	36 months	10.65%	162.87
1	1077430	1314167	2500	2500	2500.000	60 months	15.27%	59.83
2	1077175	1313524	2400	2400	2400.000	36 months	15.96%	84.33
3	1076863	1277178	10000	10000	10000.000	36 months	13.49%	339.31
4	1075358	1311748	3000	3000	3000.000	60 months	12.69%	67.79

In [87]:

```
loan.shape
```

Out[87]:

```
(39717, 48)
```

We have around 48 columns out of which some correspond to the post approval of loan

- We are analyzing the user details and the driving factors of loan defaulting before approving loan.
- So we can safely remove the columns / variables corresponding to that scenario.
- Also there are some columns such as "id", "member_id", "url", "title", "emp_title", "zip_code", "last_credit_pull_d", "addr_state".
- The above features or columns doesnt contribute to the loan defaulting in any way due to irrelevant information. So removing them.
- "desc" has description (text data) which we cannot do anythhing about for now. So removing the column.
- "out_prncp_inv" , "total_pymnt_inv " are useful for investors but not contributing to the loan defaulting analysis. So removing them.
- "funded_amnt" is not needed because we only need info as to how much is funded in actual. As we have "funded_amnt_inv" , we can remove the earlier column.

```
In [88]: loan.drop(["id", "member_id", "url", "title", "emp_title", "zip_code", "last_credi
```

```
In [89]: loan.shape
```

```
Out[89]: (39717, 21)
```

```
In [90]: loan.columns
```

```
Out[90]: Index(['loan_amnt', 'funded_amnt_inv', 'term', 'int_rate', 'installment',
                'grade', 'sub_grade', 'emp_length', 'home_ownership', 'annual_inc',
                'verification_status', 'issue_d', 'loan_status', 'purpose', 'dti',
                'earliest_cr_line', 'inq_last_6mths', 'open_acc', 'pub_rec',
                'revol_util', 'total_acc'],
               dtype='object')
```

- we can ignore the current loans because may be need look for fully paid and charged off status. Hence, removing the current loan status records

```
In [91]: loan = loan[loan.loan_status != "Current"]
         loan.loan_status.unique()
```

```
Out[91]: array(['Fully Paid', 'Charged Off'], dtype=object)
```

Find the columns which contains missing values

```
In [92]: (loan.isna().sum()/len(loan.index))*100
```

```
Out[92]: loan_amnt              0.000
         funded_amnt_inv        0.000
         term                   0.000
         int_rate               0.000
         installment            0.000
         grade                  0.000
         sub_grade              0.000
         emp_length             2.678
         home_ownership         0.000
         annual_inc             0.000
         verification_status    0.000
         issue_d                0.000
         loan_status            0.000
         purpose                0.000
         dti                    0.000
         earliest_cr_line       0.000
         inq_last_6mths         0.000
         open_acc               0.000
         pub_rec                0.000
         revol_util             0.130
         total_acc              0.000
         dtype: float64
```

Handling Missing values

- As per above data, missing values are "emp_length", "revol_util".
- So before doing that, lets see what kind of data each column has.

```
In [93]: loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 38577 entries, 0 to 39716
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   loan_amnt             38577 non-null  int64  
 1   funded_amnt_inv       38577 non-null  float64
 2   term                  38577 non-null  object 
 3   int_rate              38577 non-null  object 
 4   installment           38577 non-null  float64
 5   grade                 38577 non-null  object 
 6   sub_grade             38577 non-null  object 
 7   emp_length            37544 non-null  object 
 8   home_ownership        38577 non-null  object 
 9   annual_inc            38577 non-null  float64
 10  verification_status   38577 non-null  object 
 11  issue_d               38577 non-null  object 
 12  loan_status           38577 non-null  object 
 13  purpose               38577 non-null  object 
 14  dti                   38577 non-null  float64
 15  earliest_cr_line      38577 non-null  object 
 16  inq_last_6mths        38577 non-null  int64  
 17  open_acc              38577 non-null  int64  
 18  pub_rec               38577 non-null  int64  
 19  revol_util            38527 non-null  object 
 20  total_acc             38577 non-null  int64  
dtypes: float64(4), int64(5), object(12)
memory usage: 6.5+ MB
```

In [94]:

```
loan.emp_length.fillna(loan.emp_length.mode()[0], inplace = True)
loan.emp_length.isna().sum()
```

Out[94]:

```
0
```

In [95]:

```
loan.dropna(axis = 0, subset = ['revol_util'] , inplace = True)
loan.revol_util.isna().sum()
```

Out[95]:

```
0
```

Standardizing the data

- "revol_util" column although described as an object column, it has continous values.
- So we need to standardize the data in this column
- "int_rate" is one such column.
- "emp_length" --> { (< 1 year) is assumed as 0 and 10+ years is assumed as 10 }
- Although the datatype of "term" is arguable to be an integer, there are only two values in the whole column and it might as well be declared a categorical variable.

In [96]:

```
loan.revol_util = pd.to_numeric(loan.revol_util.apply(lambda x : x.split('%')[0]))
```

In [97]:

```
loan.int_rate = pd.to_numeric(loan.int_rate.apply(lambda x : x.split('%')[0]))
```

In [98]:

```
loan.emp_length = pd.to_numeric(loan.emp_length.apply(lambda x: 0 if "<" in x else
```

In [99]:

```
loan.head()
```

Out[99]:

	loan_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_length	hc
0	5000	4975.000	36 months	10.650	162.870	B	B2	10	
1	2500	2500.000	60 months	15.270	59.830	C	C4	0	
2	2400	2400.000	36 months	15.960	84.330	C	C5	10	
3	10000	10000.000	36 months	13.490	339.310	C	C1	10	
5	5000	5000.000	36 months	7.900	156.460	A	A4	3	

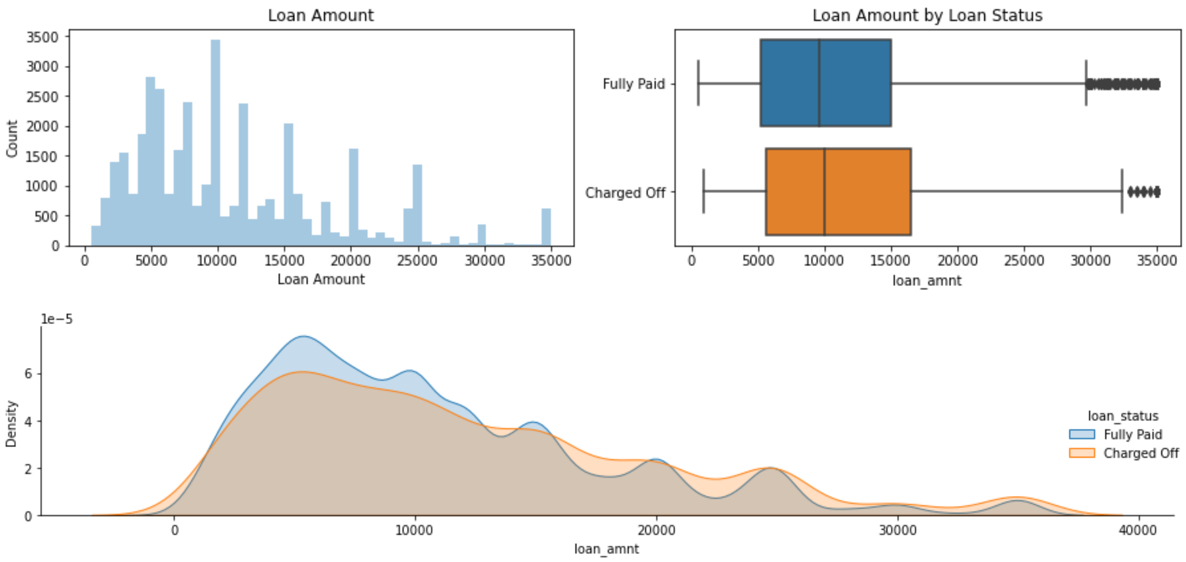
Outlier identification

In [104…

```
# make general plots to examine each feature
def plot_var(col_name, full_name, continuous):
    """
    Visualize a variable with/without faceting on the loan status.
    - col_name is the variable name in the dataframe
    - full_name is the full variable name
    - continuous is True for continuous variables
    """
    fig, (ax1, ax2) = plt.subplots(1, 2, sharex=False, figsize=(15,3))
    # plot1: counts distribution of the variable

    if continuous:
        sea.distplot(loan.loc[loan[col_name].notnull(), col_name], kde=False, ax=a
    else:
        sea.countplot(loan[col_name], order=sorted(loan[col_name].unique()), color
    ax1.set_xlabel(full_name)
    ax1.set_ylabel('Count')
    ax1.set_title(full_name)


    # plot2: bar plot of the variable grouped by loan_status
    if continuous:
        sea.boxplot(x=col_name, y='loan_status', data=loan, ax=ax2)
        ax2.set_ylabel('')
        ax2.set_title(full_name + ' by Loan Status')
    else:
        Charged_Off_rates = loan.groupby(col_name)['loan_status'].value_counts(nor
        sea.barplot(x=Charged_Off_rates.index, y=Charged_Off_rates.values, color='
        ax2.set_ylabel('Fraction of Loans Charged Off')
        ax2.set_title('Charged Off Rate by ' + full_name)
        ax2.set_xlabel(full_name)

    # plot3: kde plot of the variable gropued by loan_status
    if continuous:
        facet = sea.FacetGrid(loan, hue = 'loan_status', size=3, aspect=4)
        facet.map(sea.kdeplot, col_name, shade=True)
        #facet.set(xlim=(df[col_name].min(), df[col_name].max()))
        facet.add_legend()
    else:
        fig = plt.figure(figsize=(12,3))
        sea.countplot(x=col_name, hue='loan_status', data=loan, order=sorted(loan[
    plt.tight_layout()
```

In [105…

```
plot_var('loan_amnt', 'Loan Amount', continuous=True)
```



In [106…

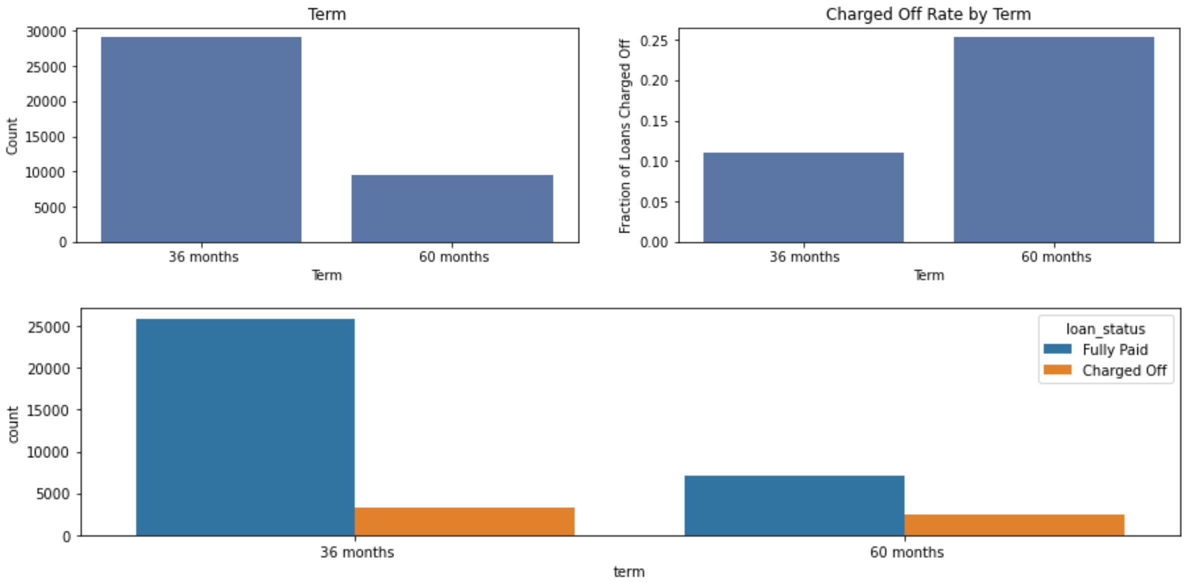
```
loan['term'].value_counts(dropna=False)
```

Out[106]:

```
 36 months    29049
 60 months     9478
Name: term, dtype: int64
```

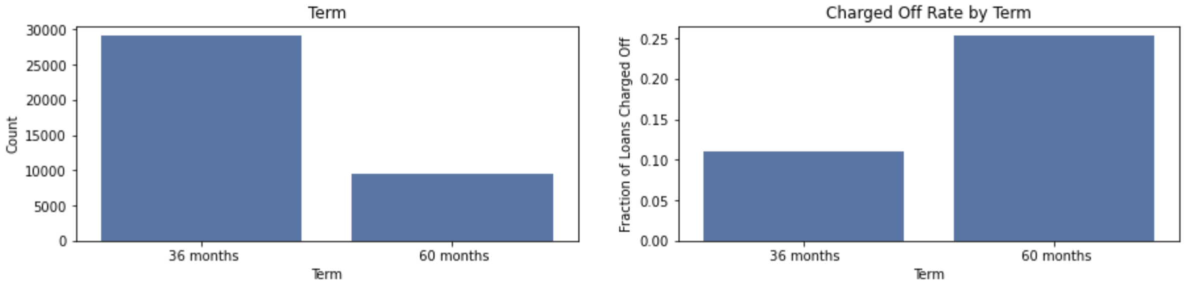
In [108…

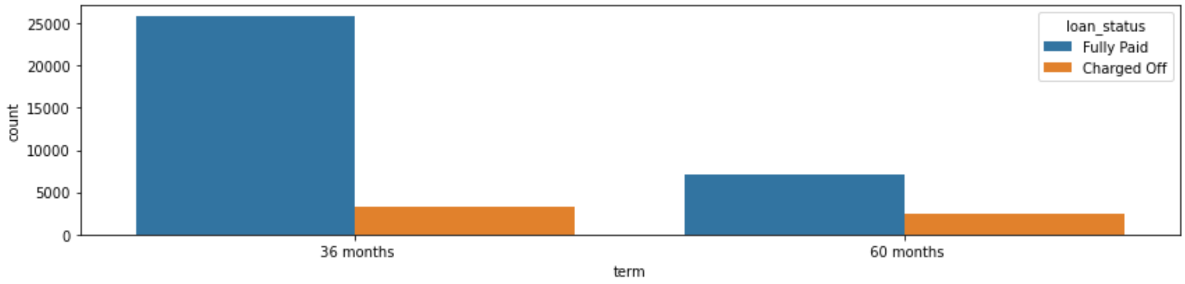
```
#loan['term'] = loan['term'].apply(lambda s: np.int8(s.split()[0]))
plot_var('term', 'Term', continuous=False)
```



In [109…

```
plot_var('term', 'Term', continuous=False)
```





Clearly indincating the presence of outliers.

- So, Removing them.
- Let's see the quantile info and take an appropriate action.
- The values after 95 percentile seems to be disconected from the general distribution and also there is huge increase in the value for small quantile variation.
- So, considering threshold for removing outliers as 0.95

In [110…

```
quantile_info = loan.annual_inc.quantile([0.5, 0.75,0.90, 0.95, 0.97,0.98, 0.99])
quantile_info
```

Out[110]:

```
0.500    59000.000
0.750    82000.000
0.900   115000.000
0.950   140004.000
0.970   165000.000
0.980   187000.000
0.990   234000.000
Name: annual_inc, dtype: float64
```

In [111…

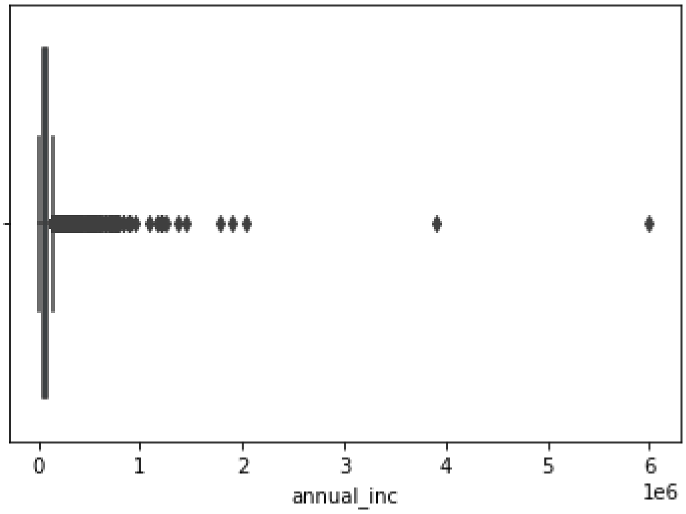
```
per_95_annual_inc = loan['annual_inc'].quantile(0.95)
loan_data = loan[loan.annual_inc <= per_95_annual_inc]
```

In [116…

```
#plot_var('annual_inc', 'Annual income', continuous=True)
sea.boxplot(loan.annual_inc)
```

Out[116]:

```
<AxesSubplot:xlabel='annual_inc'>
```



Now the "annual_inc" data looks good and proceeding next.

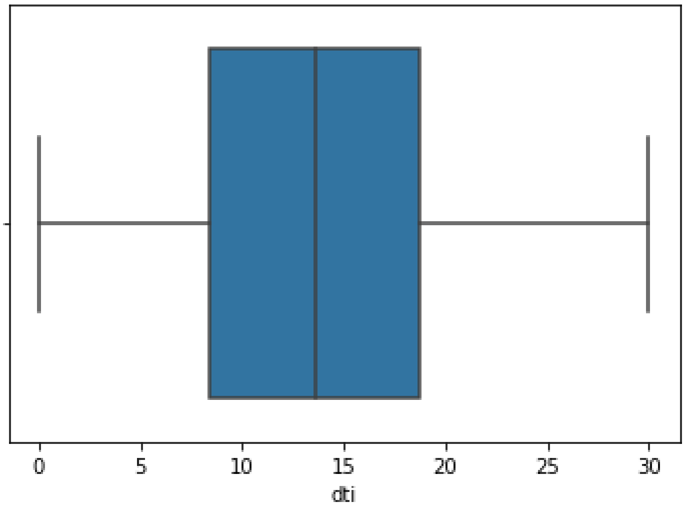
Let's analyze other numerical variables which could possibly have outliers. dti loan_amnt funded_amnt_inv

In [117…

```
sea.boxplot(loan_data.dti)
```

Out[117]:

```
<AxesSubplot:xlabel='dti'>
```

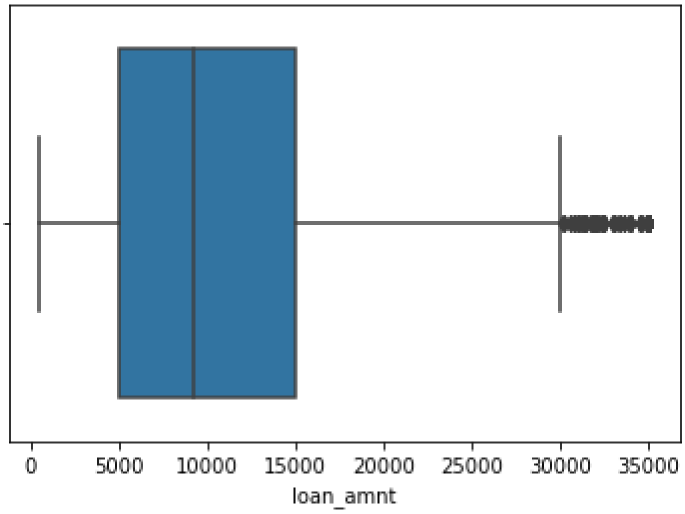


In [118…

```
sea.boxplot(loan_data.loan_amnt)
```

Out[118]:

```
<AxesSubplot:xlabel='loan_amnt'>
```



In [119…

```
loan.loan_amnt.quantile([0.75,0.90,0.95,0.97,0.975, 0.98, 0.99, 1.0])
```

Out[119]:

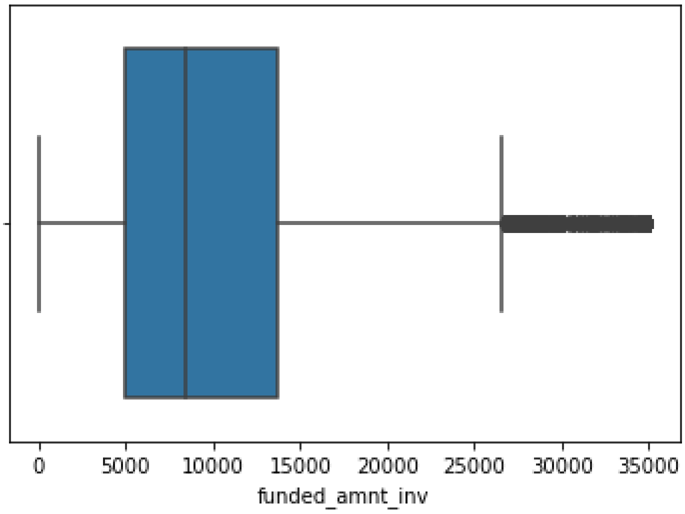
```
0.750   15000.000
0.900   21600.000
0.950   25000.000
0.970   28000.000
0.975   30000.000
0.980   30000.000
0.990   35000.000
1.000   35000.000
Name: loan_amnt, dtype: float64
```

In [120…

```
sea.boxplot(loan_data.funded_amnt_inv)
```

Out[120]:

```
<AxesSubplot:xlabel='funded_amnt_inv'>
```



```
In [121…  loan.funded_amnt_inv.quantile([0.5,0.75,0.90,0.95,0.97,0.975, 0.98,0.985, 0.99, 1.0
```

```
Out[121]: 0.500     8750.000
          0.750    14000.000
          0.900    19975.000
          0.950    24506.582
          0.970    25828.061
          0.975    27975.000
          0.980    29890.415
          0.985    30000.000
          0.990    34721.583
          1.000    35000.000
          Name: funded_amnt_inv, dtype: float64
```

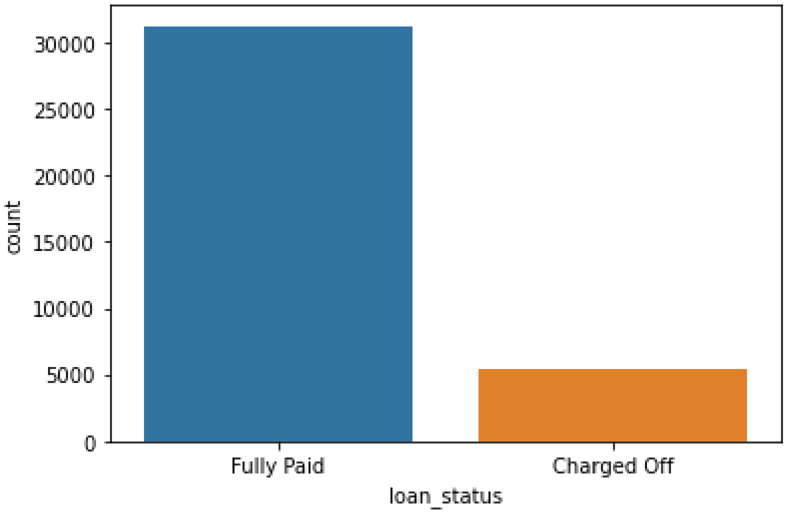
```
In [ ]:
```

Visualizing Categorical Data

- Already have grade column, extracting only subgrade (int level value) from the sub_grade variable
- Analyzing and visualizing only the defaulter data. So subsetting the data while plotting only for 'Charged Off' loan_status for below plots

```
In [122…  sea.countplot(x = 'loan_status', data = loan_data)
```

```
Out[122]: <AxesSubplot:xlabel='loan_status', ylabel='count'>
```

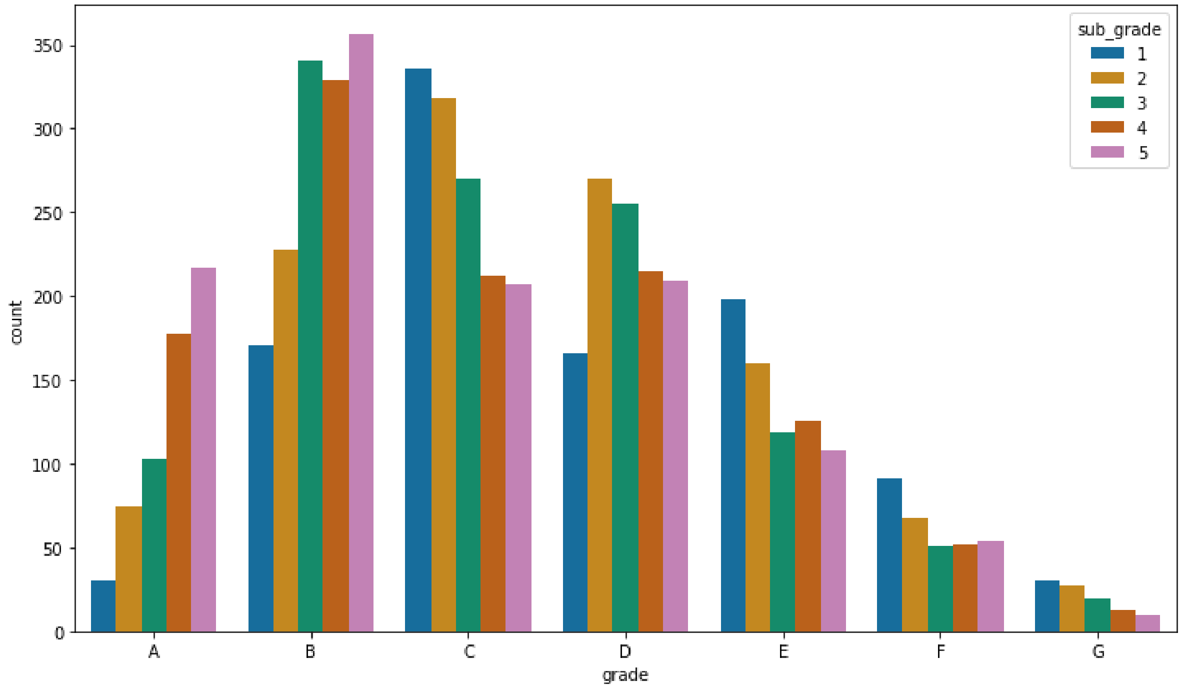


In [123…
```
loan.sub_grade = pd.to_numeric(loan.sub_grade.apply(lambda x : x[-1]))
loan.sub_grade.head()
```

Out[123]:
```
0    2
1    4
2    5
3    1
5    4
Name: sub_grade, dtype: int64
```

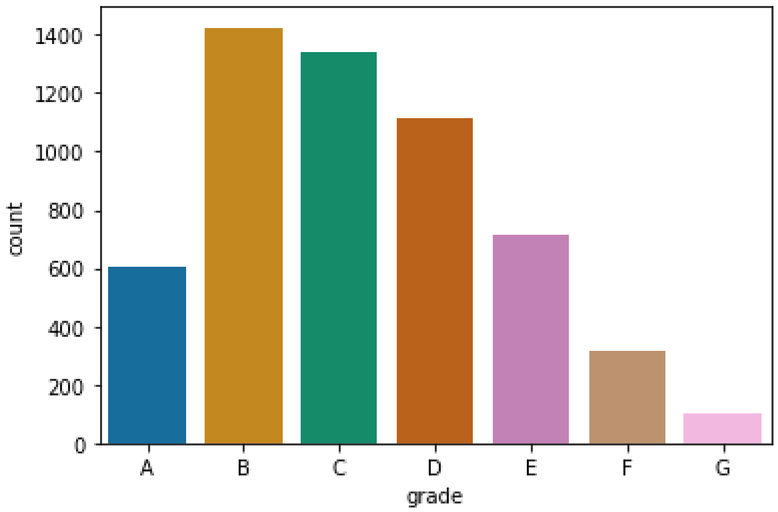
In [124…
```
fig, ax = plt.subplots(figsize=(12,7))
sea.set_palette('colorblind')
sea.countplot(x = 'grade', order = ['A', 'B', 'C', 'D', 'E', 'F', 'G'] , hue = 'sul
```

Out[124]:
```
<AxesSubplot:xlabel='grade', ylabel='count'>
```



In [126…
```
sea.countplot(x = 'grade', data = loan[loan.loan_status == 'Charged Off'], order =
```

Out[126]:
```
<AxesSubplot:xlabel='grade', ylabel='count'>
```



Analyzing home_ownership

In [127…
```
#checking unique values for home_ownership
loan['home_ownership'].unique()
```

Out[127]:

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

In [135…

```
loan[loan.home_ownership== 'NONE']
```

Out[135]:

	loan_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_lengtl
39318	10000	1228.060	36 months	7.750	312.220	A	3	5
39659	4000	1925.000	36 months	9.640	128.410	B	4	0
39660	2800	1625.000	36 months	8.700	88.650	B	1	0

There are only 3 records with 'NONE' value in the data. So replacing the value with 'OTHER'

In [136…

```
#replacing 'NONE' with 'OTHERS'
loan['home_ownership'].replace(to_replace = ['NONE'],value='OTHER',inplace = True)
```

In [137…

```
#checking unique values for home_ownership again
loan['home_ownership'].unique()
```

Out[137]:

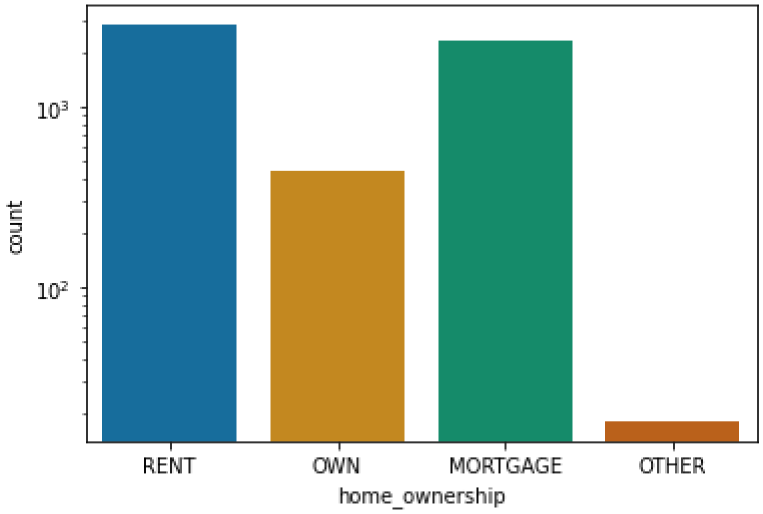
```
array(['RENT', 'OWN', 'MORTGAGE', 'OTHER'], dtype=object)
```

In [138…

```
fig, ax = plt.subplots(figsize = (6,4))
ax.set(yscale = 'log')
sea.countplot(x='home_ownership', data=loan[loan['loan_status']=='Charged Off'])
```

Out[138]:

```
<AxesSubplot:xlabel='home_ownership', ylabel='count'>
```



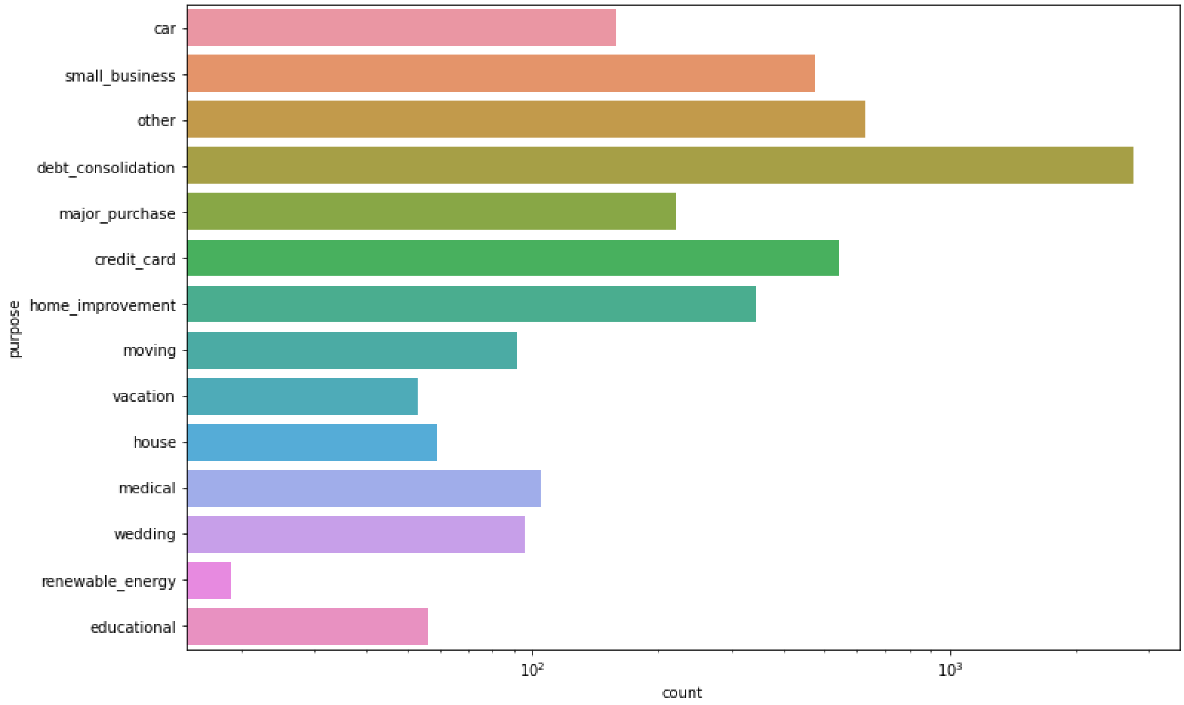
Analyzing purpose

In [139…

```
fig, ax = plt.subplots(figsize = (12,8))
ax.set(xscale = 'log')
sea.countplot(y ='purpose', data=loan[loan.loan_status == 'Charged Off'])
```

Out[139]:

```
<AxesSubplot:xlabel='count', ylabel='purpose'>
```



Creating bins for some numerical variable to make them categorical

In [140…

```
#creating bins for int_rate,open_acc,revol_util,total_acc
loan['int_rate_groups'] = pd.cut(loan['int_rate'], bins=5,precision =0,labels=['5%
loan['open_acc_groups'] = pd.cut(loan['open_acc'],bins = 5,precision =0,labels=['2
loan['revol_util_groups'] = pd.cut(loan['revol_util'], bins=5,precision =0,labels=
loan['total_acc_groups'] = pd.cut(loan['total_acc'], bins=5,precision =0,labels=['
loan['annual_inc_groups'] = pd.cut(loan['annual_inc'], bins=5,precision =0,labels =
```

In [141…

```
# Viewing new bins created
loan.head()
```

Out[141]:

	loan_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_length	ho
0	5000	4975.000	36 months	10.650	162.870	B	2	10	
1	2500	2500.000	60 months	15.270	59.830	C	4	0	
2	2400	2400.000	36 months	15.960	84.330	C	5	10	
3	10000	10000.000	36 months	13.490	339.310	C	1	10	
5	5000	5000.000	36 months	7.900	156.460	A	4	3	

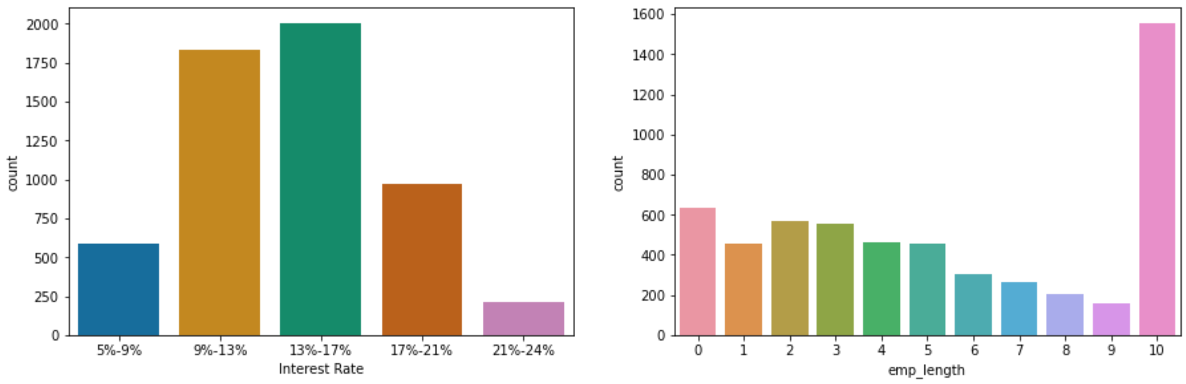
Analyzing interest rate wrt the interest rate bins created

In [142…

```
fig, ax = plt.subplots(figsize = (15,10))
plt.subplot(221)
sea.countplot(x='int_rate_groups', data=loan[loan.loan_status == 'Charged Off'])
plt.xlabel('Interest Rate')
plt.subplot(222)
sea.countplot(x='emp_length', data=loan[loan.loan_status == 'Charged Off'])
```

Out[142]:

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



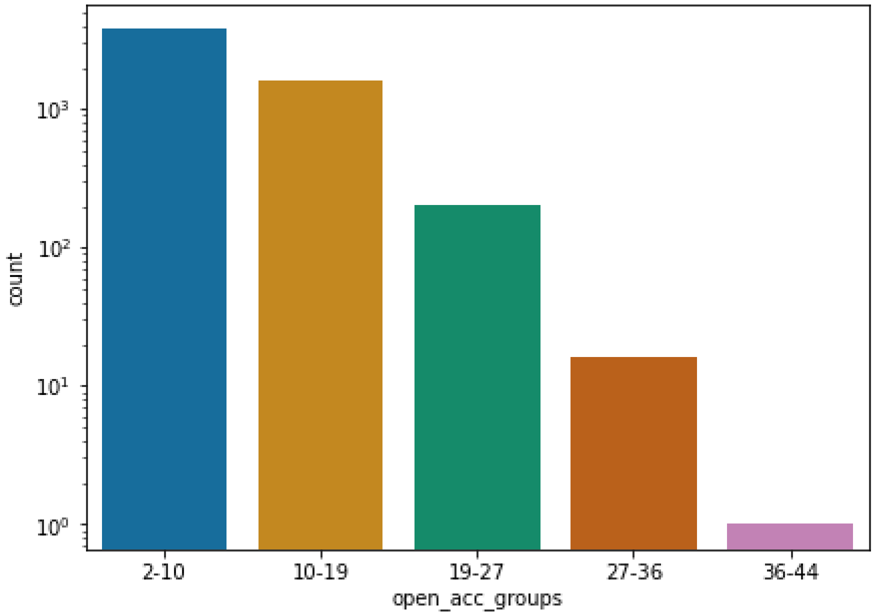
Similarly analyzing open_acc,revol_util,total_acc,annual_inc

In [143…

```
fig, ax = plt.subplots(figsize = (7,5))
ax.set_yscale('log')
sea.countplot(x='open_acc_groups', data=loan[loan.loan_status == 'Charged Off'])
```

Out[143]:

```
<AxesSubplot:xlabel='open_acc_groups', ylabel='count'>
```

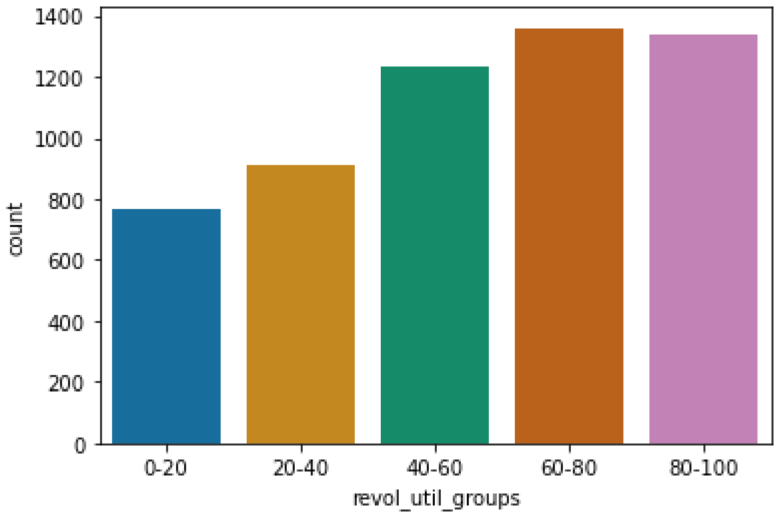


In [144…

```
sns.countplot(x='revol_util_groups', data=loan[loan.loan_status == 'Charged Off'])
```

Out[144]:

```
<AxesSubplot:xlabel='revol_util_groups', ylabel='count'>
```

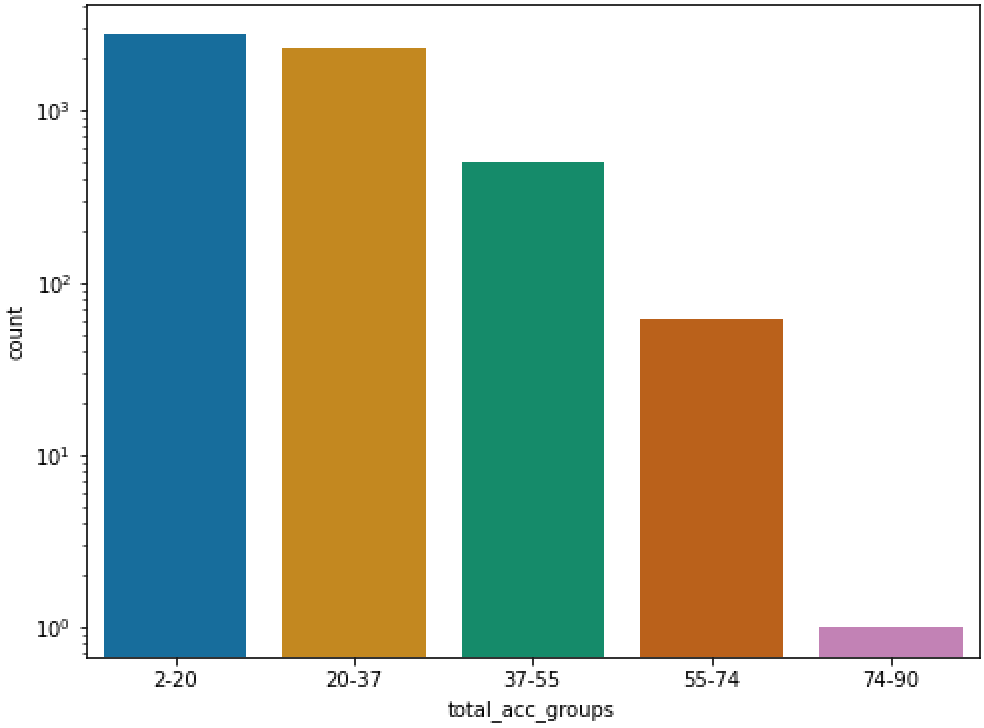


In [145…

```
fig, ax = plt.subplots(figsize = (8,6))
ax.set_yscale('log')
sea.countplot(x='total_acc_groups', data=loan[loan.loan_status == 'Charged Off'])
```

Out[145]:

```
<AxesSubplot:xlabel='total_acc_groups', ylabel='count'>
```

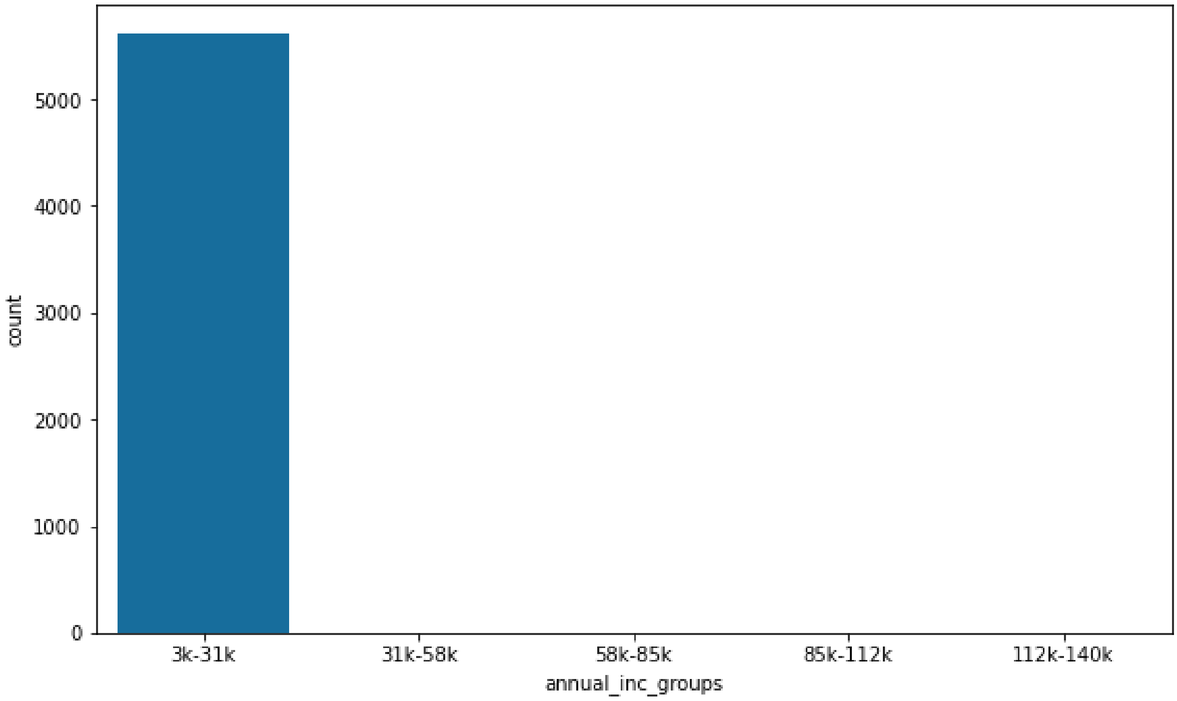


In [146…

```
fig, ax = plt.subplots(figsize = (10,6))
sea.countplot(x='annual_inc_groups', data=loan[loan.loan_status == 'Charged Off'])
```

Out[146]:

```
<AxesSubplot:xlabel='annual_inc_groups', ylabel='count'>
```

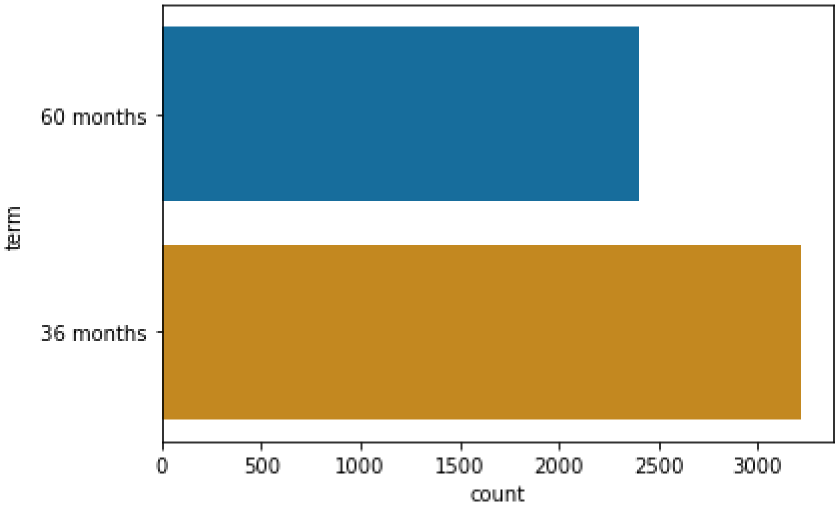


In [147…

```
sea.countplot(y='term', data=loan[loan['loan_status']=='Charged Off'])
```

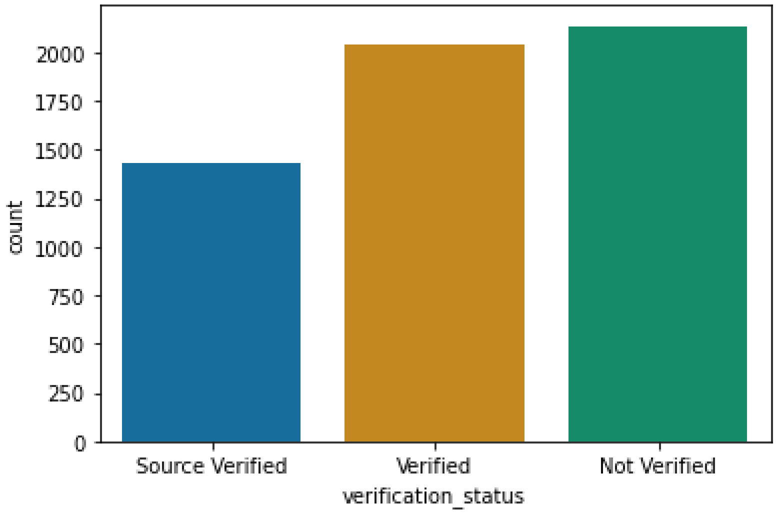
Out[147]:

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



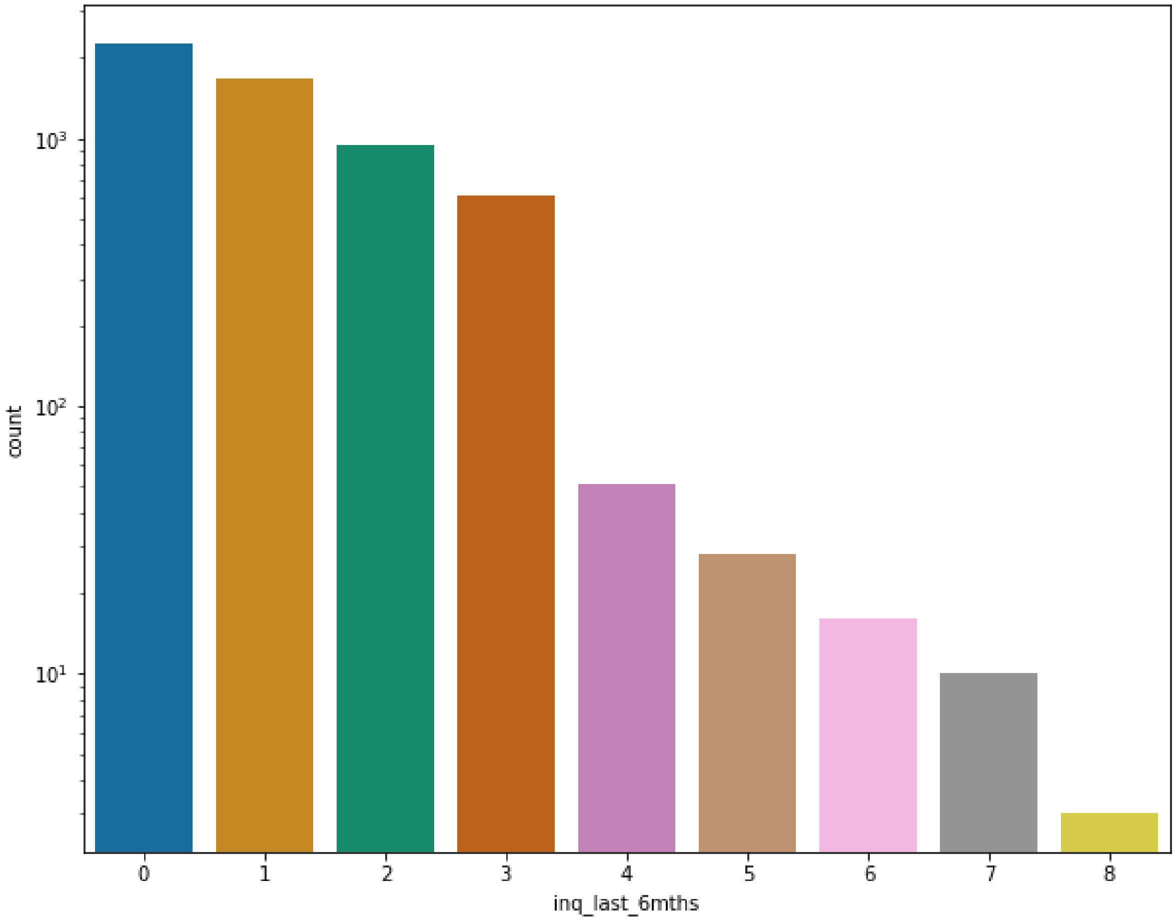
```
In [148…  sea.countplot(x='verification_status', data=loan[loan['loan_status']=='Charged Off
```

```
Out[148]: <AxesSubplot:xlabel='verification_status', ylabel='count'>
```



```
In [149…  fig,ax = plt.subplots(figsize = (10,8))
          ax.set_yscale('log')
          sea.countplot(x='inq_last_6mths', data=loan[loan['loan_status']=='Charged Off'])
```

```
Out[149]: <AxesSubplot:xlabel='inq_last_6mths', ylabel='count'>
```



Analyzing by issued month and year

In [150…

```
## Extracting month and year
df_month_year = loan['issue_d'].str.partition("-", True)
loan['issue_month']=df_month_year[0]
loan['issue_year']='20' + df_month_year[2]
```

In [151…

```
loan.head()
```

Out[151]:

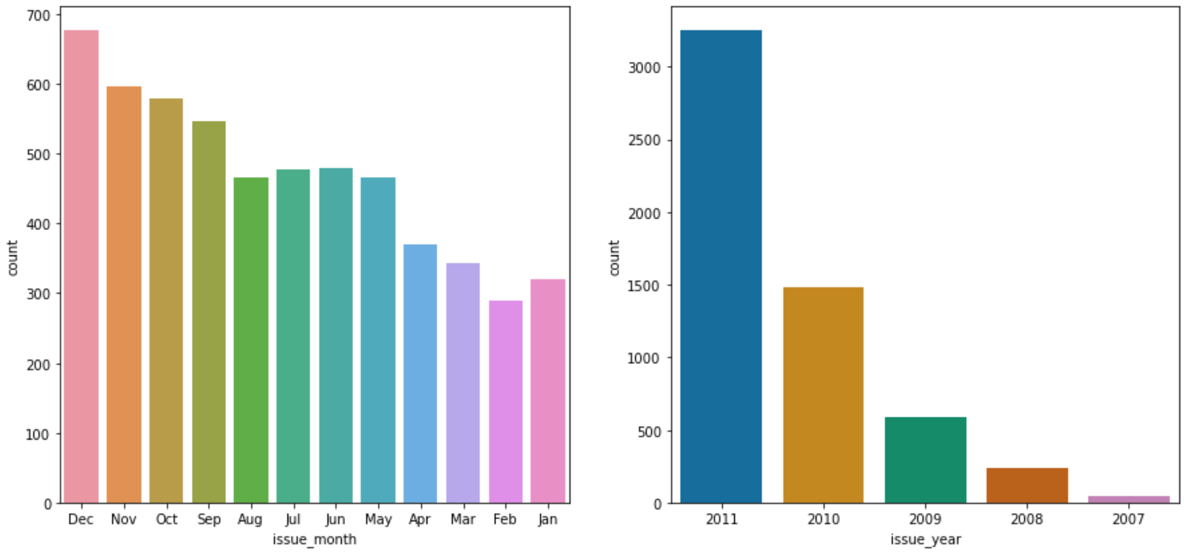
	loan_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_length	hc
0	5000	4975.000	36 months	10.650	162.870	B	2	10	
1	2500	2500.000	60 months	15.270	59.830	C	4	0	
2	2400	2400.000	36 months	15.960	84.330	C	5	10	
3	10000	10000.000	36 months	13.490	339.310	C	1	10	
5	5000	5000.000	36 months	7.900	156.460	A	4	3	

In [152…

```
plt.figure(figsize=(15,15))
plt.subplot(221)
sea.countplot(x='issue_month', data=loan[loan['loan_status']=='Charged Off'])
plt.subplot(222)
sea.countplot(x='issue_year', data=loan[loan['loan_status']=='Charged Off'])
```

Out[152]:

```
<AxesSubplot:xlabel='issue_year', ylabel='count'>
```



Maximum number of defaults occured when the loan was sanctioned/issued in Dec. Loan issued in the year 2011 were also as compared to other years

Analyzing installment,dti, loan_amnt

In [153…

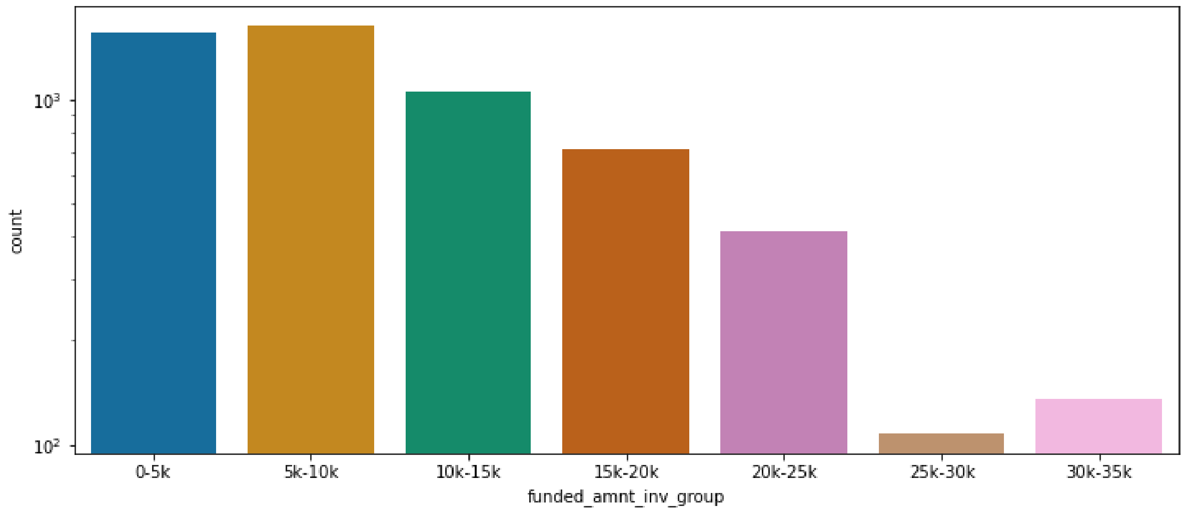
```
loan['installment_groups'] = pd.cut(loan['installment'], bins=10,precision =0,label
loan['funded_amnt_inv_group'] = pd.cut(loan['funded_amnt_inv'], bins=7,labels=['0-5
loan['loan_amnt_groups'] = pd.cut(loan['loan_amnt'], bins=7,precision =0,labels=['(
loan['dti_groups'] = pd.cut(loan['dti'], bins=5,precision =0,labels=['0-6','6-12',
```

In [154…

```
fig,ax = plt.subplots(figsize = (12,5))
ax.set_yscale('log')
sea.countplot(x='funded_amnt_inv_group', data=loan[loan['loan_status']=='Charged O
```

Out[154]:

```
<AxesSubplot:xlabel='funded_amnt_inv_group', ylabel='count'>
```

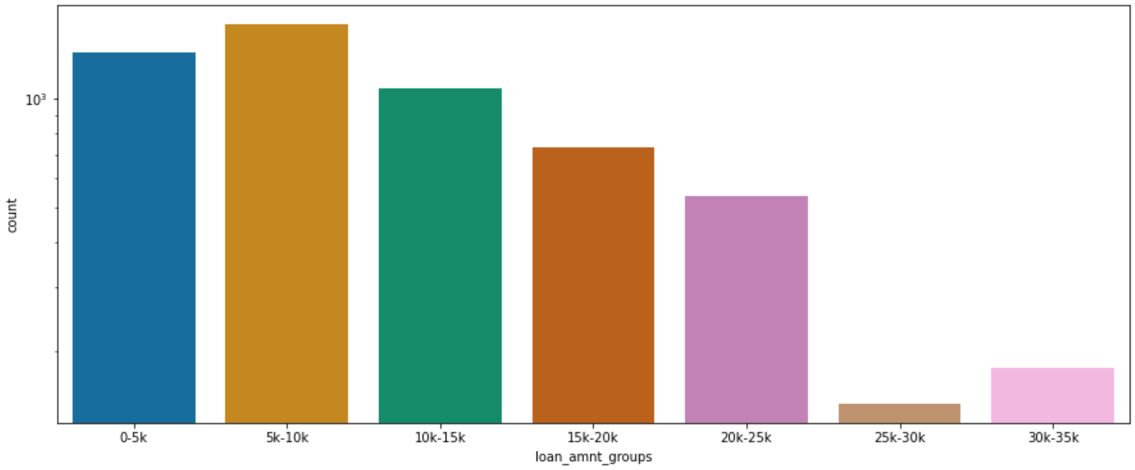


In [155…

```
fig,ax = plt.subplots(figsize = (15,6))
ax.set_yscale('log')
sea.countplot(x='loan_amnt_groups', data=loan[loan['loan_status']=='Charged Off'])
```

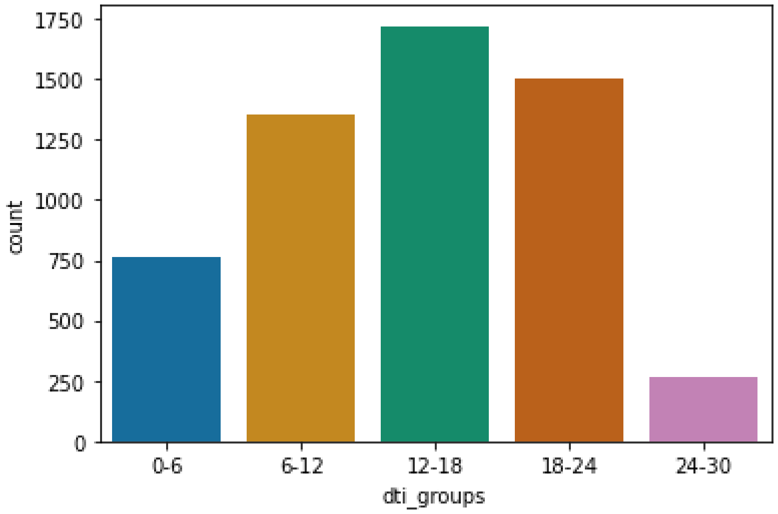
Out[155]:

```
<AxesSubplot:xlabel='loan_amnt_groups', ylabel='count'>
```



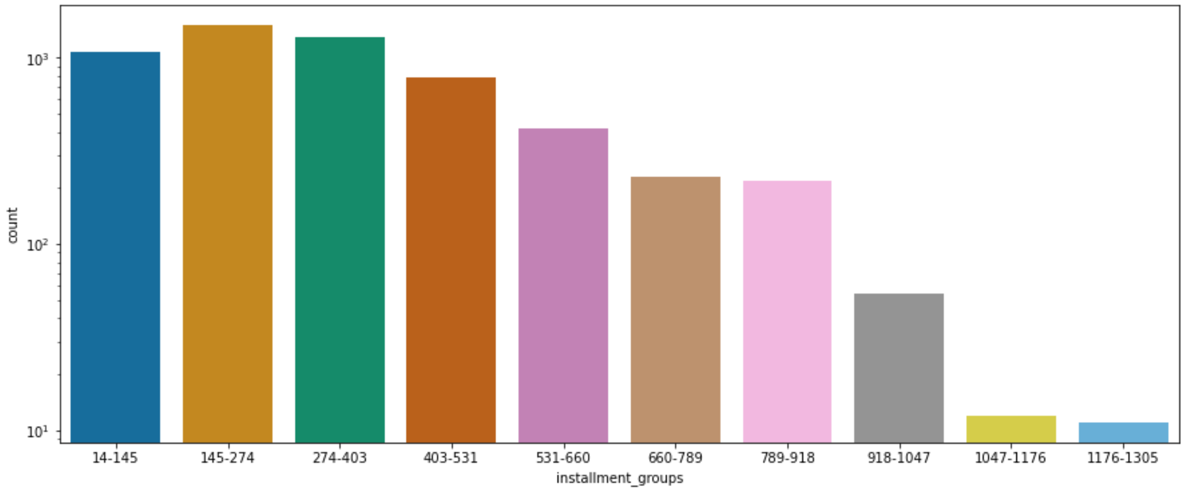
```
In [156…  sea.countplot(x='dti_groups', data=loan[loan['loan_status']=='Charged Off'])
```

```
Out[156]: <AxesSubplot:xlabel='dti_groups', ylabel='count'>
```



```
In [157…  fig,ax = plt.subplots(figsize = (15,6))
          ax.set_yscale('log')
          sea.countplot(x='installment_groups', data=loan[loan['loan_status']=='Charged Off'
```

```
Out[157]: <AxesSubplot:xlabel='installment_groups', ylabel='count'>
```



Observations

- The above analysis with respect to the charged off loans for each variable suggests the following. There is a more probability of defaulting when :

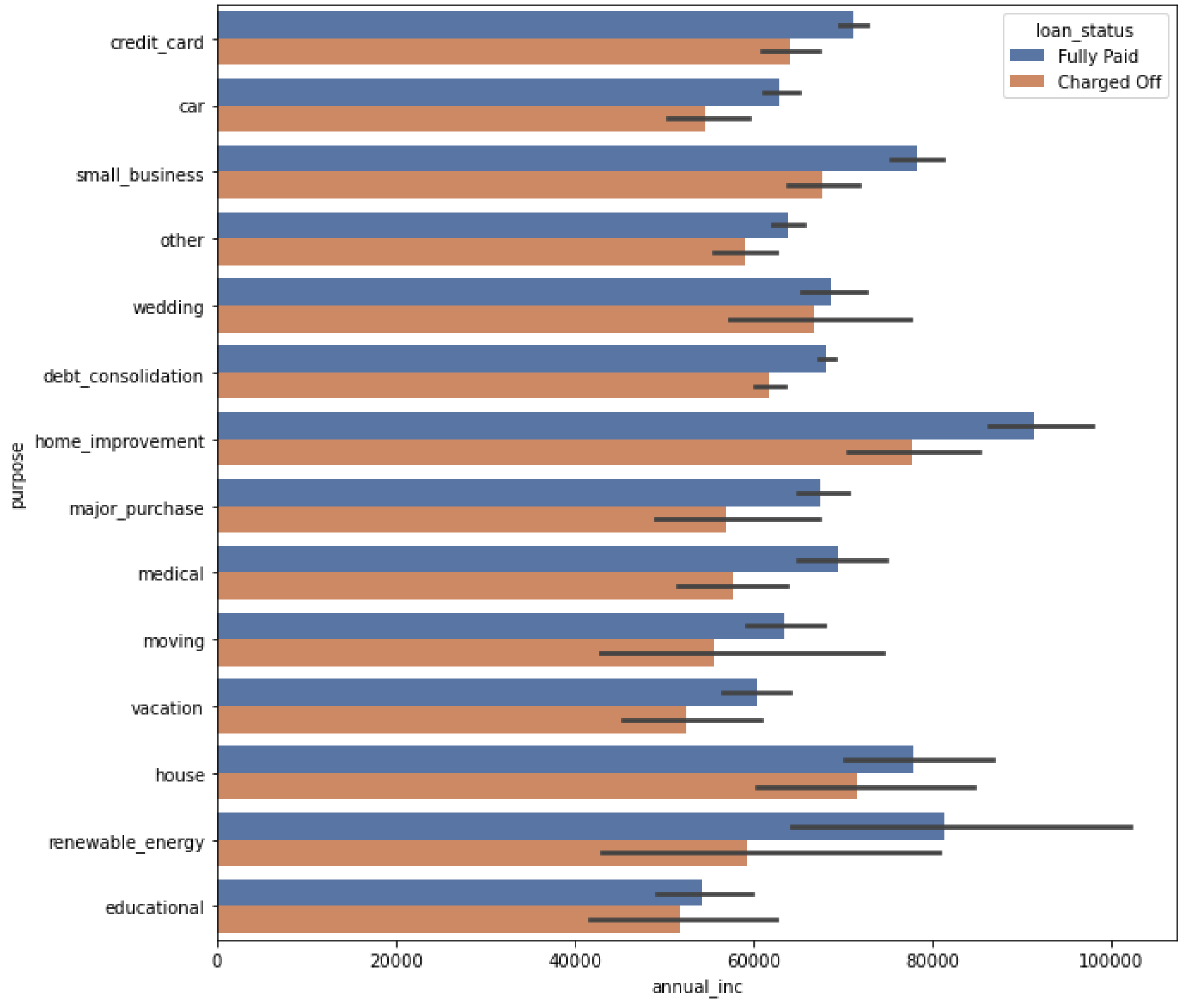
- Applicants having house_ownership as 'RENT'
- Applicants who use the loan to clear other debts
- Applicants who receive interest at the rate of 13-17%
- Applicants who have an income of range 31201 - 58402
- Applicants who have 20-37 open_acc
- Applicants with employement length of 10
- When funded amount by investor is between 5000-10000
- Loan amount is between 5429 - 10357
- Dti is between 12-18
- When monthly installments are between 145-274
- Term of 36 months
- When the loan status is Not verified
- When the no of enquiries in last 6 months is 0
- When the number of derogatory public records is 0
- When the purpose is 'debt_consolidation'
- Grade is 'B'
- And a total grade of 'B5' level.

Also there is a very interesting observation from the date issued. The late months of an year indicated the high possibility of defaulting.

1. Annual income vs loan purpose

In [158…

```
plt.figure(figsize=(10,10))
sea.barplot(data =loan,x='annual_inc', y='purpose', hue ='loan_status',palette="de
plt.show()
```

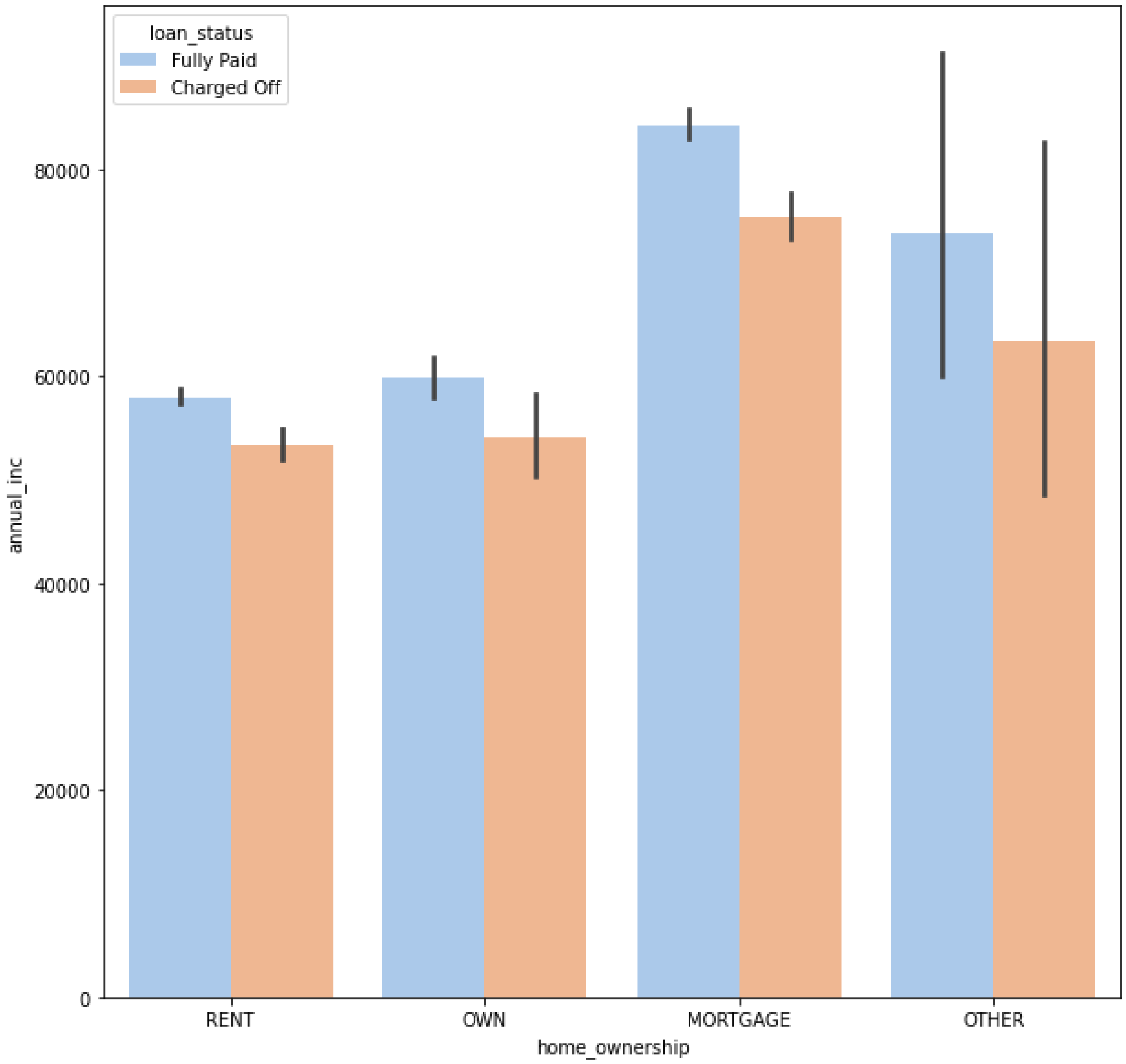


Though the number of loans applied and defaulted are the highest in number for "debt_consolation", the annual income of those who applied isn't the highest.

Applicants with higher salary mostly applied loans for "home_improvment", "house", "renewable_energy" and "small_businesses"

2.Annual income vs home ownership

```
In [159…
plt.figure(figsize=(10,10))
sea.barplot(data =loan,x='home_ownership', y='annual_inc', hue ='loan_status',pale
plt.show()
```



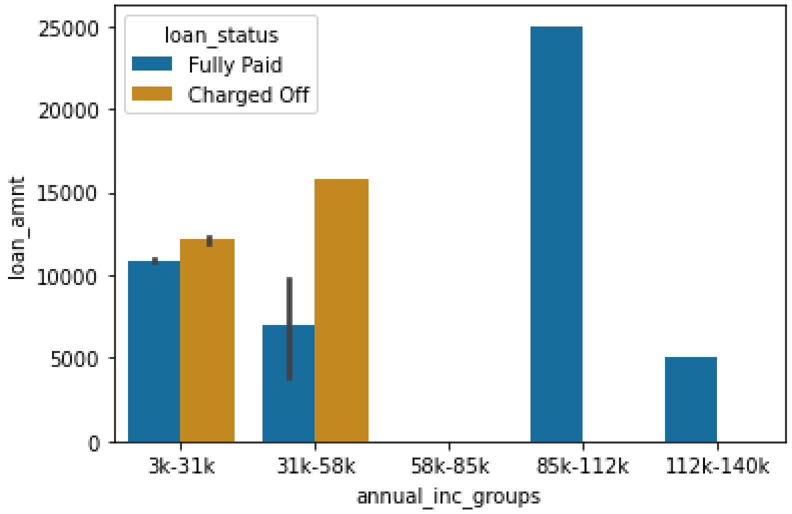
Annual Income vs Loan amount

In [162…

```
sea.barplot(x = "annual_inc_groups", y = "loan_amnt", hue = 'loan_status', data = 
```

Out[162]:

```
<AxesSubplot:xlabel='annual_inc_groups', ylabel='loan_amnt'>
```

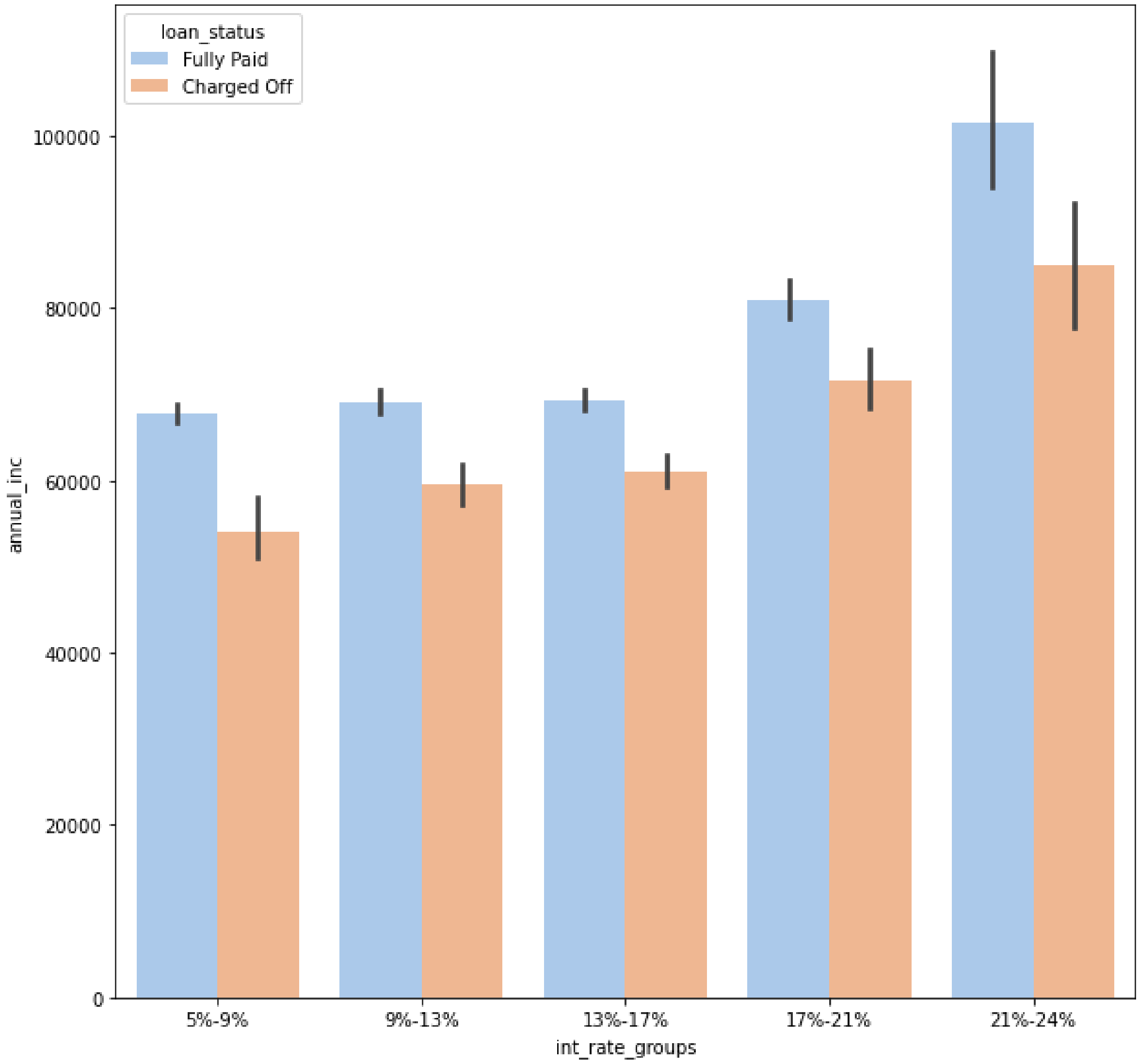


Across all the income groups, the loan_amount is higher for people who defaulted.

3. Annual income vs int_rate

In [163…

```
plt.figure(figsize=(10,10))
sea.barplot(data =loan,x='int_rate_groups', y='annual_inc', hue ='loan_status',pal
plt.show()
```

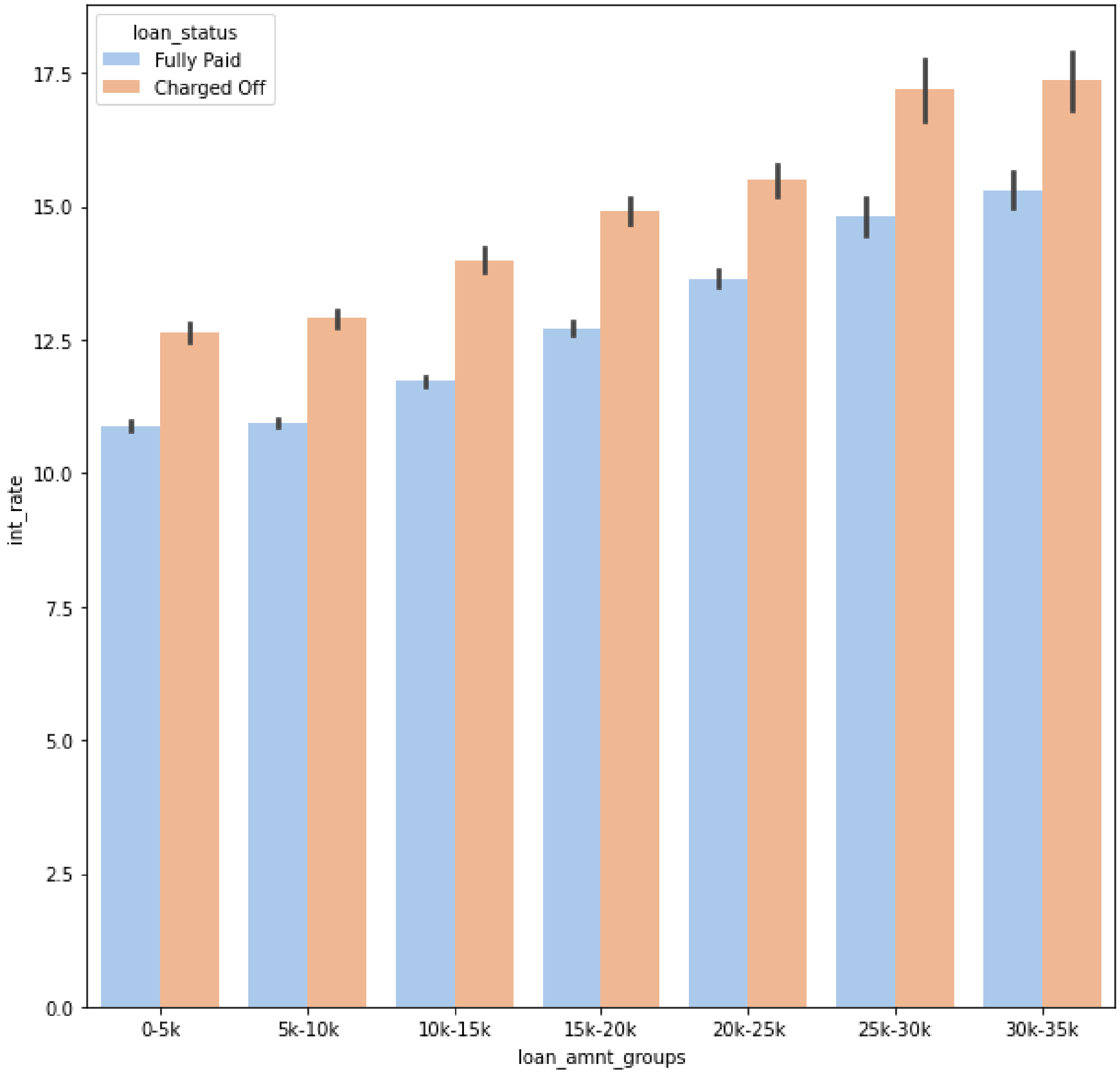


Analysing loan_amount with other columns for more insights

1.Loan Amount vs Interest Rate

In [165…

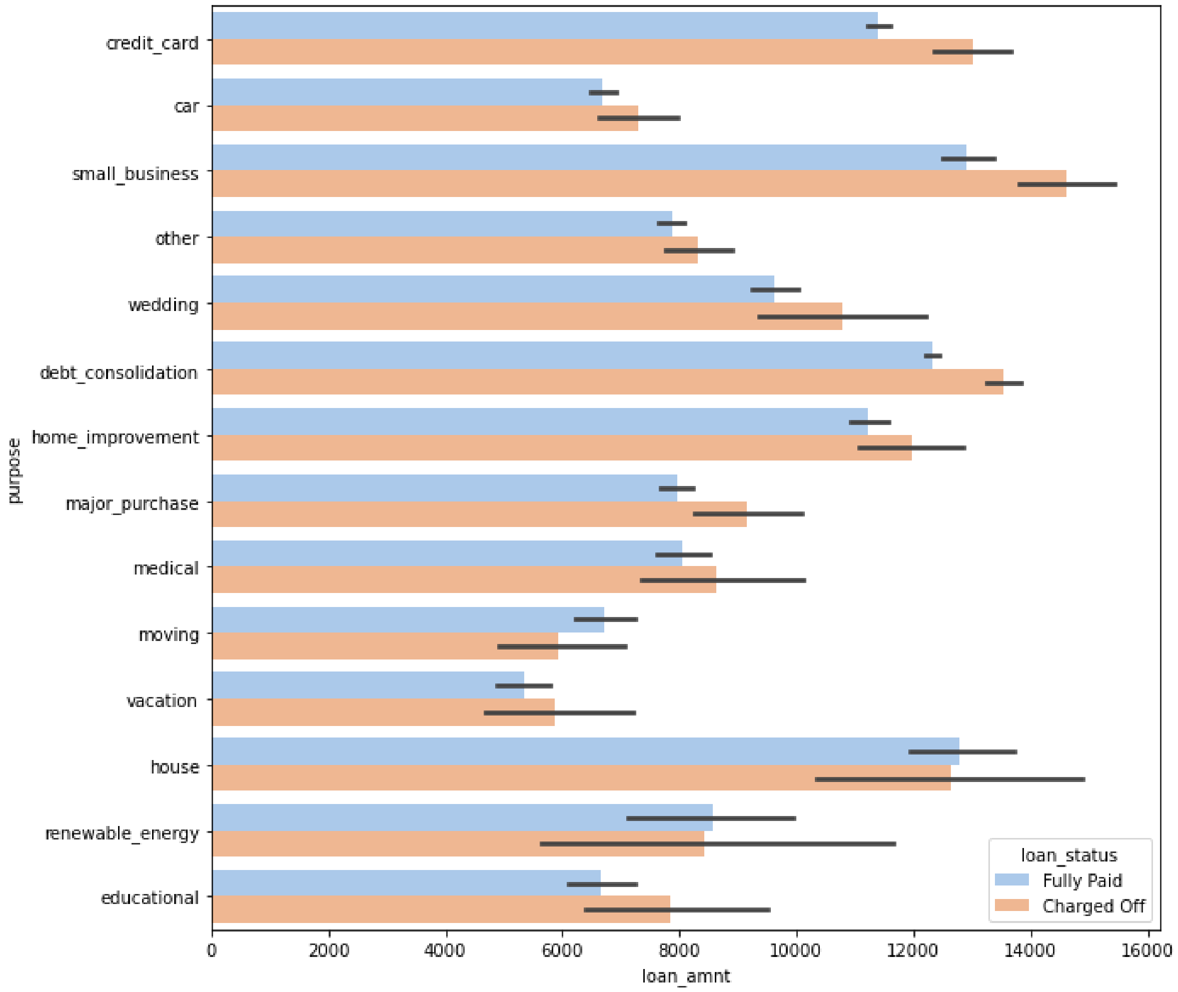
```
plt.figure(figsize=(10,10))
sea.barplot(data =loan,x='loan_amnt_groups', y='int_rate', hue ='loan_status',pale
plt.show()
```



2.Loan vs Loan purpose

In [166…

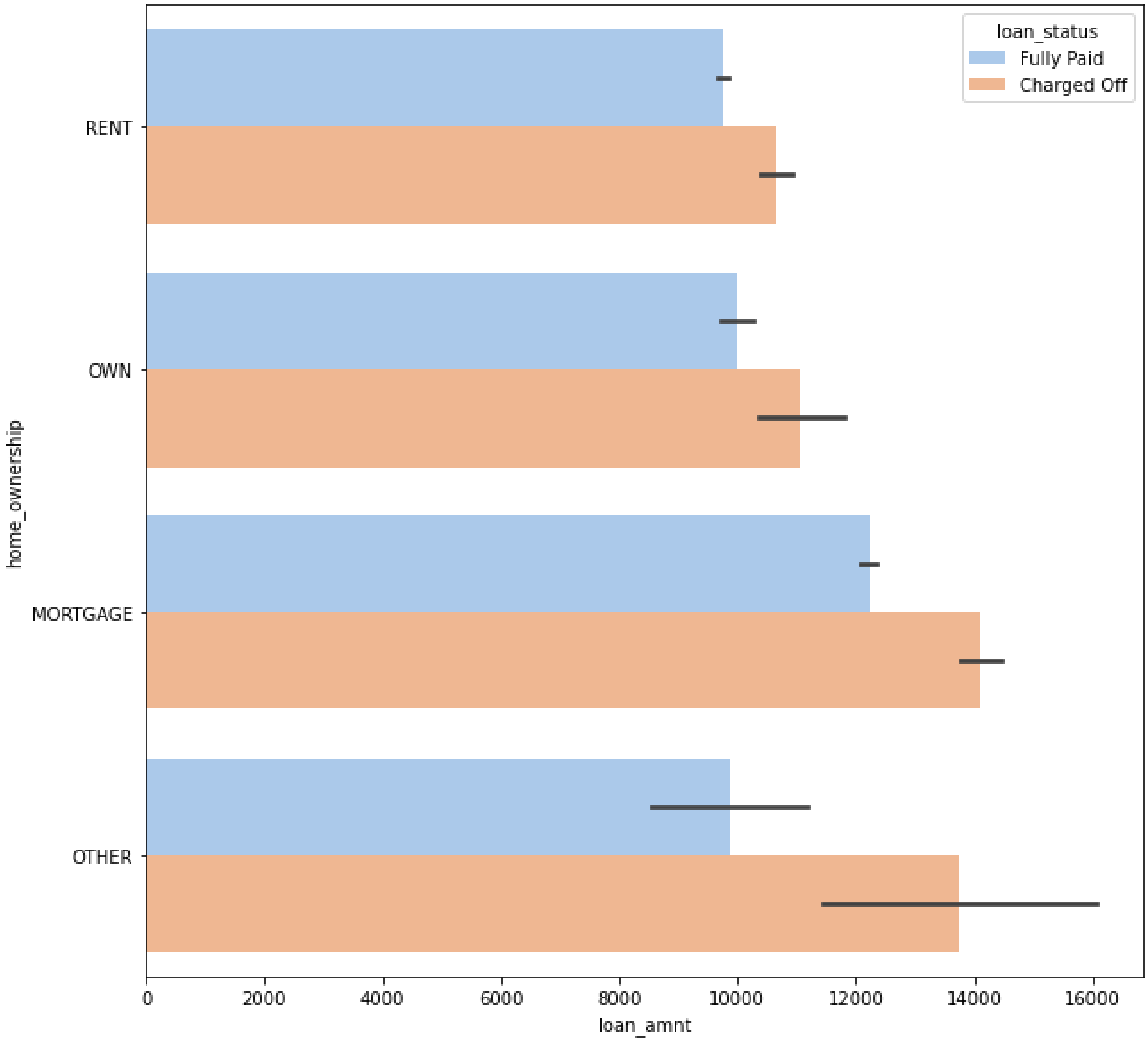
```
plt.figure(figsize=(10,10))
sea.barplot(data =loan,x='loan_amnt', y='purpose', hue ='loan_status',palette="pas
plt.show()
```



3.Loan vs House Ownership

In [167…

```
plt.figure(figsize=(10,10))
sea.barplot(data =loan,x='loan_amnt', y='home_ownership', hue ='loan_status',palet
plt.show()
```



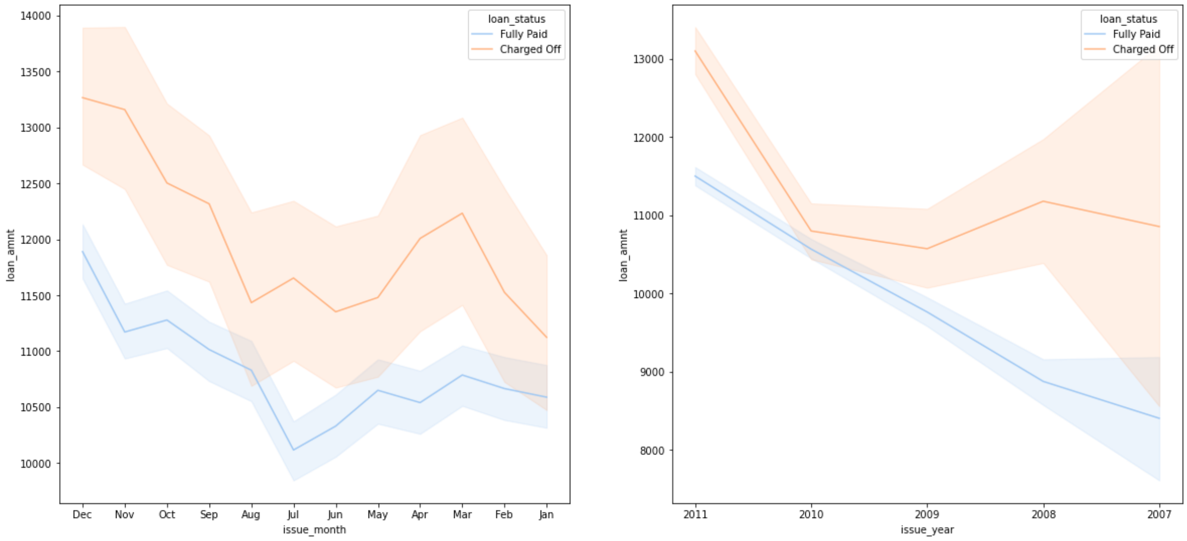
4.Loan amount vs month issued and year issued

In [168…

```
plt.figure(figsize=(20,20))
plt.subplot(221)
sea.lineplot(data =loan,y='loan_amnt', x='issue_month', hue ='loan_status',palette=
plt.subplot(222)
sea.lineplot(data =loan,y='loan_amnt', x='issue_year', hue ='loan_status',palette='
```

Out[168]:

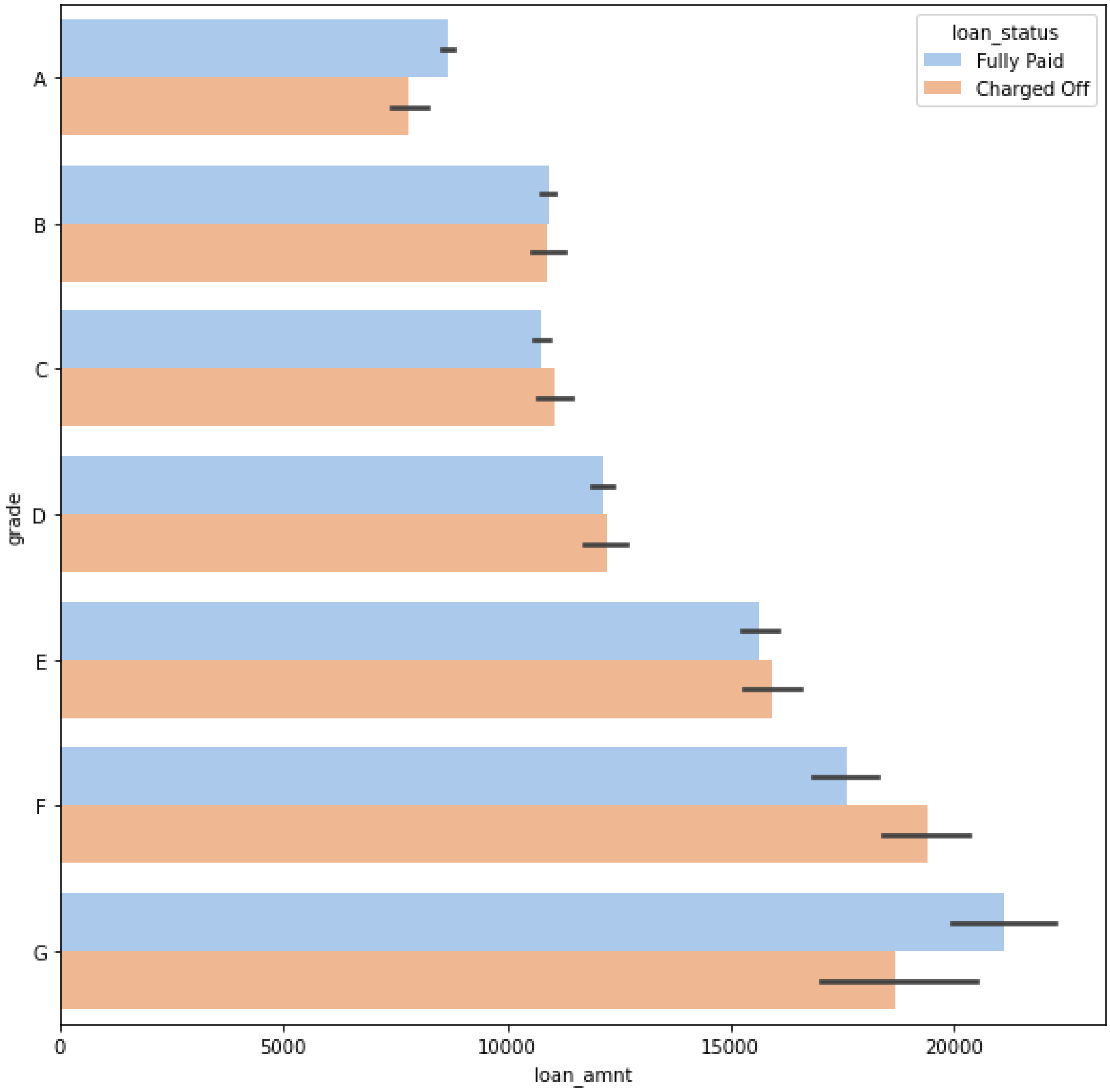
```
<AxesSubplot:xlabel='issue_year', ylabel='loan_amnt'>
```



5.Loan amount vs Grade

In [169…

```
plt.figure(figsize=(10,10))
sea.barplot(data =loan,x='loan_amnt', y='grade', hue ='loan_status',palette="paste
plt.show()
```

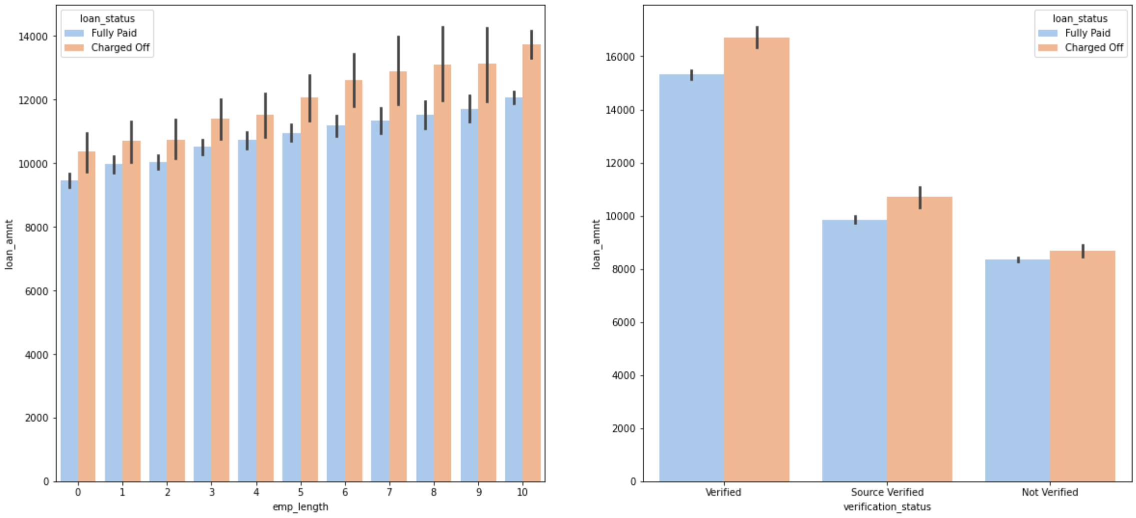


In [170…

```
plt.figure(figsize=(20,20))
plt.subplot(221)
sea.barplot(data =loan,y='loan_amnt', x='emp_length', hue ='loan_status',palette="
plt.subplot(222)
sea.barplot(data =loan,y='loan_amnt', x='verification_status', hue ='loan_status',
```

Out[170]:

```
<AxesSubplot:xlabel='verification_status', ylabel='loan_amnt'>
```



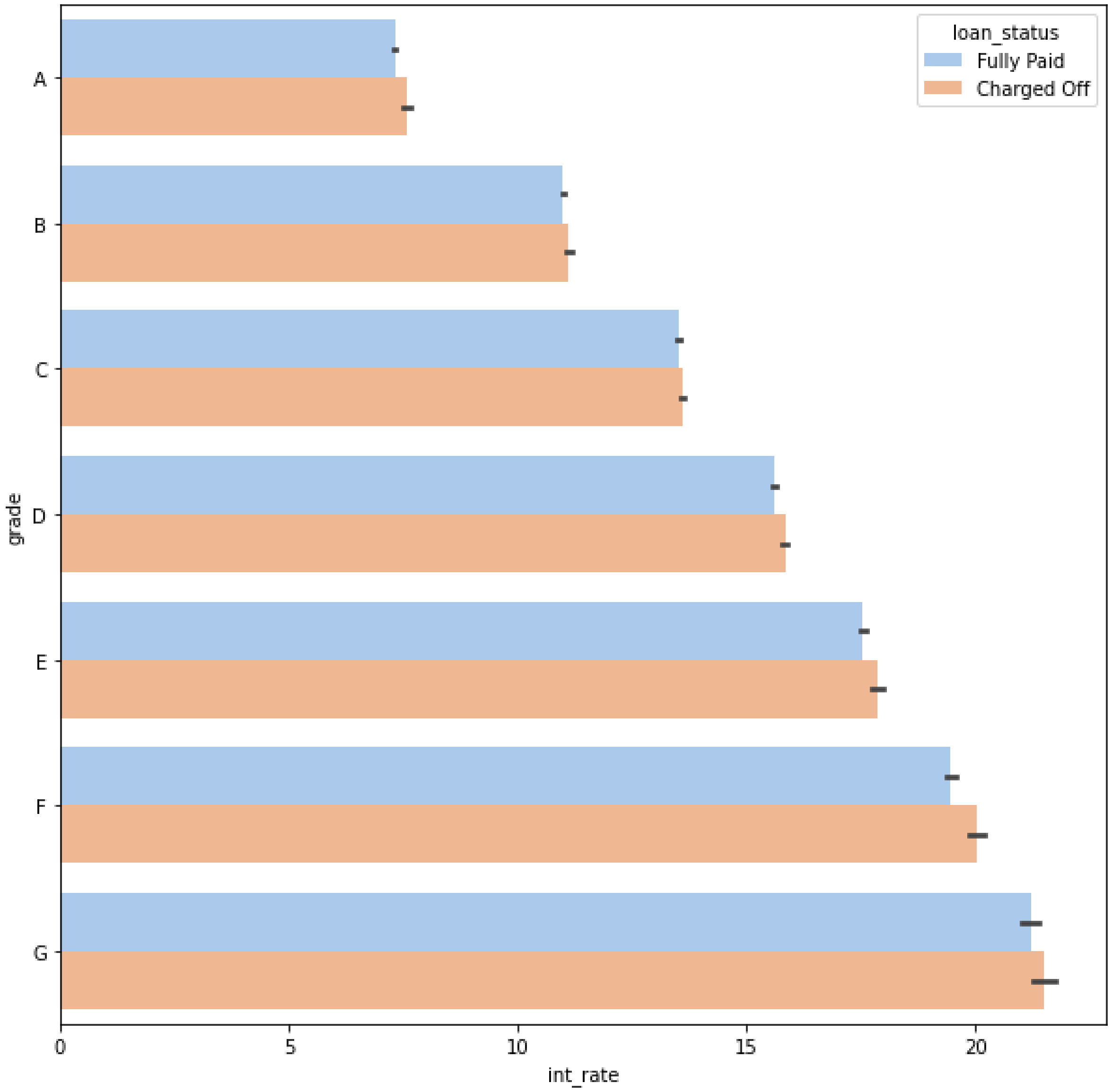
Employees with longer working history got the loan approved for a higher amount.

- Looking at the verification status data, verified loan applications tend to have higher loan amount. Which might indicate that the firms are first verifying the loans with higher values.

grade vs interest rate

In [171…

```
plt.figure(figsize=(10,10))
sea.barplot(data =loan,x='int_rate', y='grade', hue ='loan_status',palette="pastel
plt.show()
```

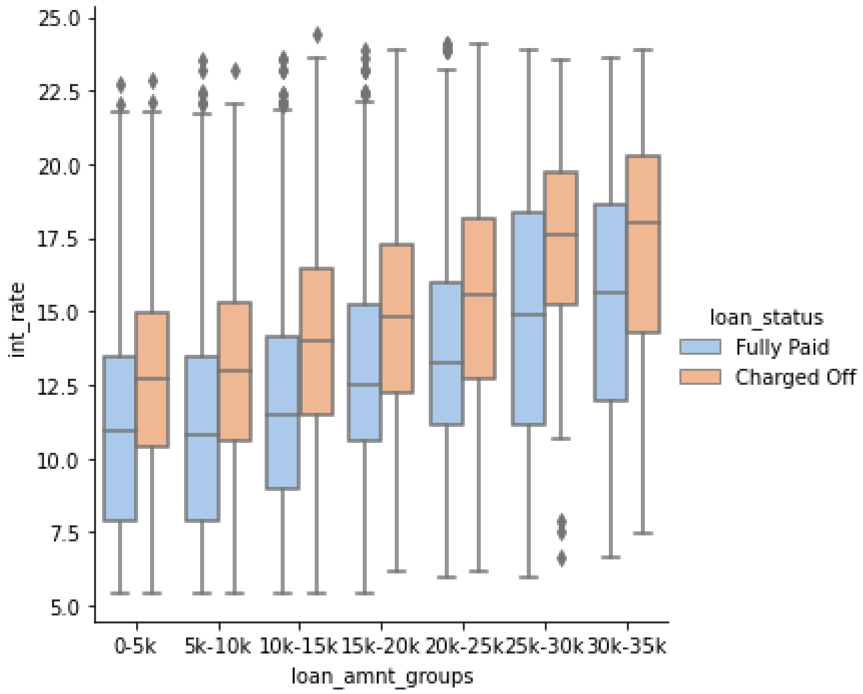


In [172…

```
plt.tight_layout()
sea.catplot(data =loan,y ='int_rate', x ='loan_amnt_groups', hue ='loan_status',pa
```

Out[172]:

```
<seaborn.axisgrid.FacetGrid at 0x1775b4d0df0>

<Figure size 432x288 with 0 Axes>
```

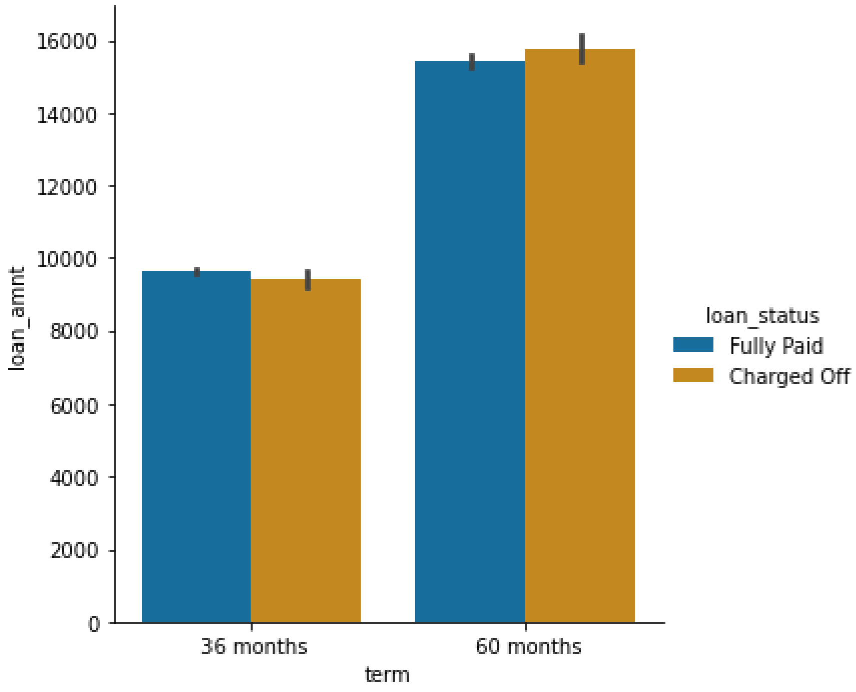


The interest rate for charged off loans is pretty high than that of fully paid loans in all the loan_amount groups.

- This can be a pretty strong driving factor for loan defaulting.

```
In [173…  sea.catplot(x = 'term', y = 'loan_amnt', data = loan,hue = 'loan_status', kind = '
```

```
Out[173]: <seaborn.axisgrid.FacetGrid at 0x1775b4761f0>
```



Applicants who applied and defaulted have no significant difference in loan_amounts.

- Which means that applicants applying for long term has applied for more loan.

Observations

The above analysis with respect to the charged off loans. There is a more probability of defaulting when :

- Applicants taking loan for 'home improvement' and have income of 60k -70k
- Applicants whose home ownership is 'MORTGAGE and have income of 60-70k
- Applicants who receive interest at the rate of 21-24% and have an income of 70k-80k
- Applicants who have taken a loan in the range 30k - 35k and are charged interest rate of 15-17.5 %
- Applicants who have taken a loan for small business and the loan amount is greater than 14k
- Applicants whose home ownership is 'MORTGAGE and have loan of 14-16k
- When grade is F and loan amount is between 15k-20k
- When employment length is 10yrs and loan amount is 12k-14k
- When the loan is verified and loan amount is above 16k
- For grade G and interest rate above 20%

Reference

- Machine Learning With Python book
- https://www.lendingclub.com
- https://pandas.pydata.org/
- https://numpy.org/
- https://seaborn.pydata.org/

```
In [ ]:
```