GOAL: Analyse lending data to identify the variables that are strong indicators of default

GROUP 8

- Rohit Rohit - 0773987
- Dimple Annapareddy – 0789185
- Ninad Patil - 0779680
- Dhruvi Patel - 0788987
- Sri Kartik Kotni - 0774337

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

In [2]:
```
data= pd.read_csv('loan.csv')
data
```

```
C:\Users\dhruv\anaconda3\envs\Finance\lib\site-packages\IPython\core\interactivesh
ell.py:3457: DtypeWarning: Columns (47) have mixed types.Specify dtype option on i
mport or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[2]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	install
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	1
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	3
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	
...	...	...	...	...	...	...	...	
39712	92187	92174	2500	2500	1075.0	36 months	8.07%	
39713	90665	90607	8500	8500	875.0	36 months	10.28%	2
39714	90395	90390	5000	5000	1325.0	36 months	8.07%	1
39715	90376	89243	5000	5000	650.0	36 months	7.43%	1
39716	87023	86999	7500	7500	800.0	36 months	13.75%	2

39717 rows × 111 columns

In [3]:
```
data.isnull().sum().head(100)
```

Out[3]:
```
id                          0
member_id                   0
loan_amnt                   0
funded_amnt                 0
funded_amnt_inv             0
                          ...
num_op_rev_tl           39717
num_rev_accts           39717
num_rev_tl_bal_gt_0     39717
num_sats                39717
num_tl_120dpd_2m        39717
Length: 100, dtype: int64
```

In [4]:
```
Blank_columns = data.columns[100*(data.isnull().sum()/len(data.index)) == 100]
data.drop(columns = Blank_columns, inplace = True)
data.shape
```

Out[4]:
```
(39717, 57)
```

In [5]:
```
col = [column for column in data.columns if len(data[column].value_counts()) <2 ]

print(data.acc_now_delinq.value_counts())  # the number of accounts on which the borro
```

```
print('\n', data.application_type.value_counts()) #Individual application or a joint a

print('\n', data.initial_list_status.value_counts())

print('\n', data.policy_code.value_counts())

print( '\n', data.pymnt_plan.value_counts())

print( '\n', data.collections_12_mths_ex_med.value_counts())

print( '\n', data.chargeoff_within_12_mths.value_counts())

print( '\n', data.tax_liens.value_counts())

data.drop(columns=col, inplace=True)
```

```
0    39717
Name: acc_now_delinq, dtype: int64

 INDIVIDUAL    39717
Name: application_type, dtype: int64

 f    39717
Name: initial_list_status, dtype: int64

 1    39717
Name: policy_code, dtype: int64

 n    39717
Name: pymnt_plan, dtype: int64

 0.0    39661
Name: collections_12_mths_ex_med, dtype: int64

 0.0    39661
Name: chargeoff_within_12_mths, dtype: int64

 0.0    39678
Name: tax_liens, dtype: int64
```

Sanity Checks

In [6]:
```
#Check IDs
print(data.id.nunique())

print(data.addr_state.nunique())

print(data.term.value_counts())

print(data.loan_amnt.describe())
```

```
39717
50
 36 months    29096
 60 months    10621
Name: term, dtype: int64
count    39717.000000
mean     11219.443815
std       7456.670694
min        500.000000
25%       5500.000000
50%      10000.000000
75%      15000.000000
max      35000.000000
Name: loan_amnt, dtype: float64
```

In [7]:
```
ignore_columns = ['url', 'member_id', 'emp_title', 'zip_code', 'title', 'desc','out_pr
data.drop(columns=ignore_columns, inplace=True)
```

In [8]:
```
# % of missing values
missing_data = 100*(data.isnull().sum()/len(data.index))
print(missing_data)

# Lets handle these case by case in the following cells
```

```
id                          0.000000
loan_amnt                   0.000000
funded_amnt                 0.000000
term                        0.000000
int_rate                    0.000000
installment                 0.000000
grade                       0.000000
sub_grade                   0.000000
emp_length                  2.706650
home_ownership              0.000000
annual_inc                  0.000000
verification_status         0.000000
issue_d                     0.000000
loan_status                 0.000000
purpose                     0.000000
addr_state                  0.000000
dti                         0.000000
delinq_2yrs                 0.000000
earliest_cr_line            0.000000
inq_last_6mths              0.000000
mths_since_last_delinq     64.662487
mths_since_last_record     92.985372
open_acc                    0.000000
pub_rec                     0.000000
revol_bal                   0.000000
revol_util                  0.125891
total_acc                   0.000000
last_credit_pull_d          0.005036
pub_rec_bankruptcies        1.754916
dtype: float64
```

In [9]:
```
data.drop(columns=['mths_since_last_record', 'mths_since_last_delinq'], inplace=True)
```

In [10]:
```
print(data.pub_rec_bankruptcies.describe())
```

```
data.pub_rec_bankruptcies.fillna(0, inplace=True)
```

```
count    39020.000000
mean         0.043260
std          0.204324
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          2.000000
Name: pub_rec_bankruptcies, dtype: float64
```

In [11]:
```
print(data.emp_length.value_counts(), data.emp_length.describe())
```

```
10+ years    8879
< 1 year     4583
2 years      4388
3 years      4095
4 years      3436
5 years      3282
1 year       3240
6 years      2229
7 years      1773
8 years      1479
9 years      1258
Name: emp_length, dtype: int64 count         38642
unique           11
top       10+ years
freq           8879
Name: emp_length, dtype: object
```

In [12]:
```
print('\n', data.revol_util.value_counts(), data.revol_util.describe())
```

```
 0%        977
0.20%      63
63%        62
40.70%     58
66.70%     58
          ...
25.74%      1
47.36%      1
24.65%      1
10.61%      1
7.28%       1
Name: revol_util, Length: 1089, dtype: int64 count     39667
unique     1089
top          0%
freq        977
Name: revol_util, dtype: object
```

In [13]:
```
print('\n', data.last_credit_pull_d.value_counts(), data.last_credit_pull_d.describe()
```

```
 May-16    10308
Apr-16     2547
Mar-16     1123
Feb-13      843
Feb-16      736
           ...
May-08        1
Jun-08        1
Jul-08        1
May-07        1
Jul-07        1
Name: last_credit_pull_d, Length: 106, dtype: int64 count      39715
unique       106
top       May-16
freq       10308
Name: last_credit_pull_d, dtype: object
```

Datatype Transformation

In [14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Data columns (total 27 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    39717 non-null  int64  
 1   loan_amnt             39717 non-null  int64  
 2   funded_amnt           39717 non-null  int64  
 3   term                  39717 non-null  object 
 4   int_rate              39717 non-null  object 
 5   installment           39717 non-null  float64
 6   grade                 39717 non-null  object 
 7   sub_grade             39717 non-null  object 
 8   emp_length            38642 non-null  object 
 9   home_ownership        39717 non-null  object 
 10  annual_inc            39717 non-null  float64
 11  verification_status   39717 non-null  object 
 12  issue_d               39717 non-null  object 
 13  loan_status           39717 non-null  object 
 14  purpose               39717 non-null  object 
 15  addr_state            39717 non-null  object 
 16  dti                   39717 non-null  float64
 17  delinq_2yrs           39717 non-null  int64  
 18  earliest_cr_line      39717 non-null  object 
 19  inq_last_6mths        39717 non-null  int64  
 20  open_acc              39717 non-null  int64  
 21  pub_rec               39717 non-null  int64  
 22  revol_bal             39717 non-null  int64  
 23  revol_util            39667 non-null  object 
 24  total_acc             39717 non-null  int64  
 25  last_credit_pull_d    39715 non-null  object 
 26  pub_rec_bankruptcies  39717 non-null  float64
dtypes: float64(4), int64(9), object(14)
memory usage: 8.2+ MB
```

In [15]:
```
data.int_rate.head()
```

Out[15]:
```
0    10.65%
1    15.27%
2    15.96%
3    13.49%
4    12.69%
Name: int_rate, dtype: object
```

In [16]:
```
data.int_rate = data.int_rate.str[:-1].astype('float')
print(data.int_rate)
```

```
0        10.65
1        15.27
2        15.96
3        13.49
4        12.69
         ...
39712     8.07
39713    10.28
39714     8.07
39715     7.43
39716    13.75
Name: int_rate, Length: 39717, dtype: float64
```

In [17]:
```
data.revol_util = data.revol_util.str[:-1].astype('float')
```

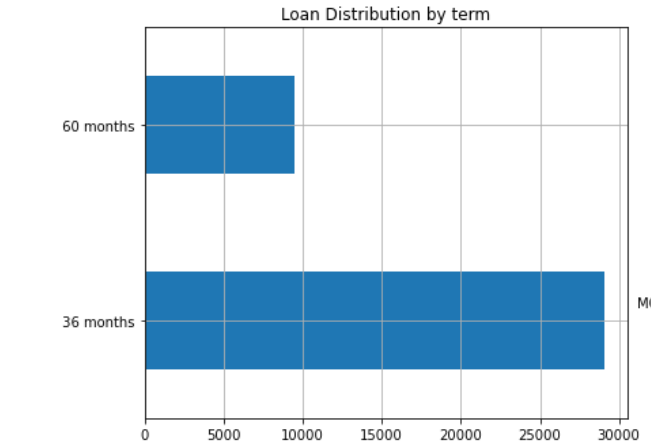
In [18]:
```
def to_datetime(x):
    (month, year)=x.strip().split('-')
    year = int(year)
    if year > 20:
        year = 1900+year
    else:
        year = 2000+ year
    x = '-'.join(['01',month,str(year)])
    return datetime.datetime.strptime(x,'%d-%b-%Y')

def to_month(x):
    x = to_datetime(x)
    return x.month

def to_year(x):
    x = to_datetime(x)
    return x.year

data['earliest_cr_line_year'] = data.earliest_cr_line.apply(to_year)
data['earliest_cr_line_month'] = data.earliest_cr_line.apply(to_month)

data['issue_d_year'] = data.issue_d.apply(to_year)
data['issue_d_month'] = data.issue_d.apply(to_month)

data.drop(columns=['earliest_cr_line', 'issue_d'], inplace=True)
```

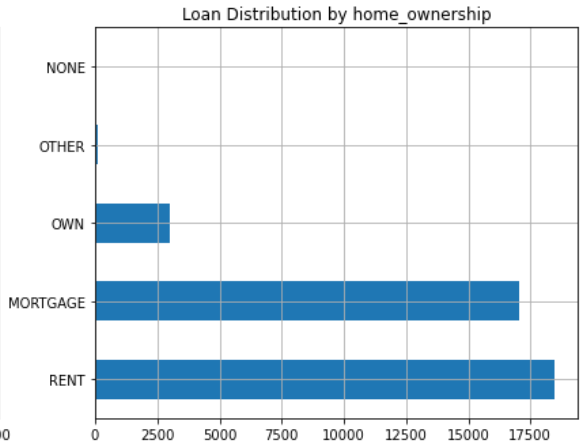
Filtering Data

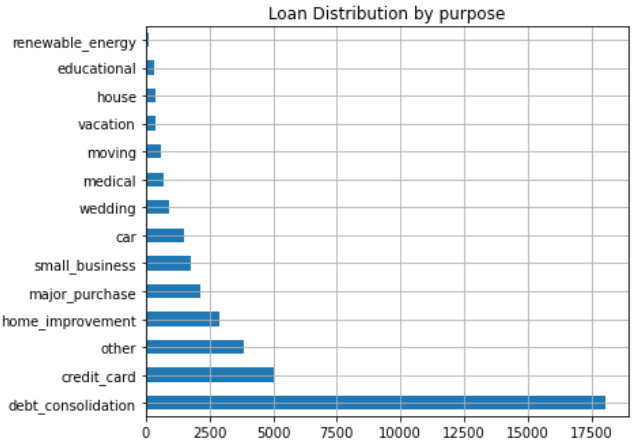
In [19]:
```
# filter out those account details, in which the loan status is current since our
# target is analyze factors that indicate default.
data = data[data.loan_status != 'Current']
```

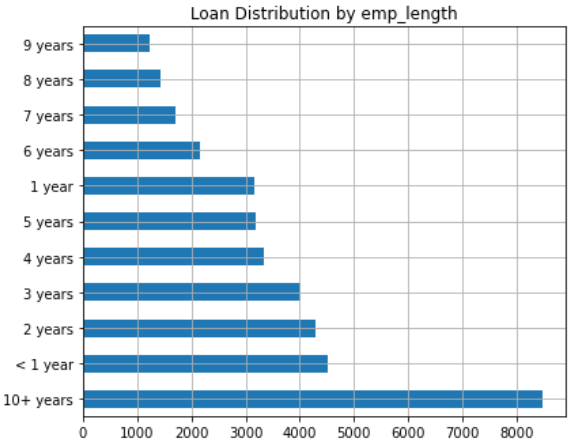
Univariate Analysis

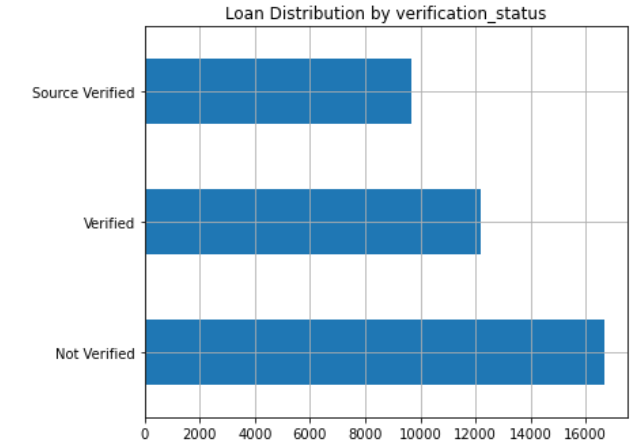
In [20]:
```
# Lets checkout the catagorical variables
uni_cols=['term', 'home_ownership','purpose', 'emp_length','verification_status', 'gra
ncol=2
nrow=int(round(len(uni_cols)/2,0))

# make a list of all dataframes
df_list = [data[x] for x in uni_cols]

fig, axes = plt.subplots(nrow, ncol,figsize=(7*ncol, 6 * nrow))
# plot counter
count=0
for r in range(nrow):
    for c in range(ncol):
        if count < len(uni_cols):
            df_list[count].value_counts().plot(ax=axes[r,c], kind='barh',grid=True, ti
        count= count+1

plt.show()
```

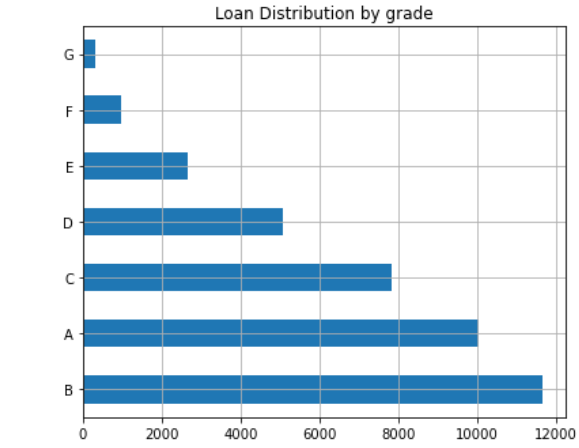










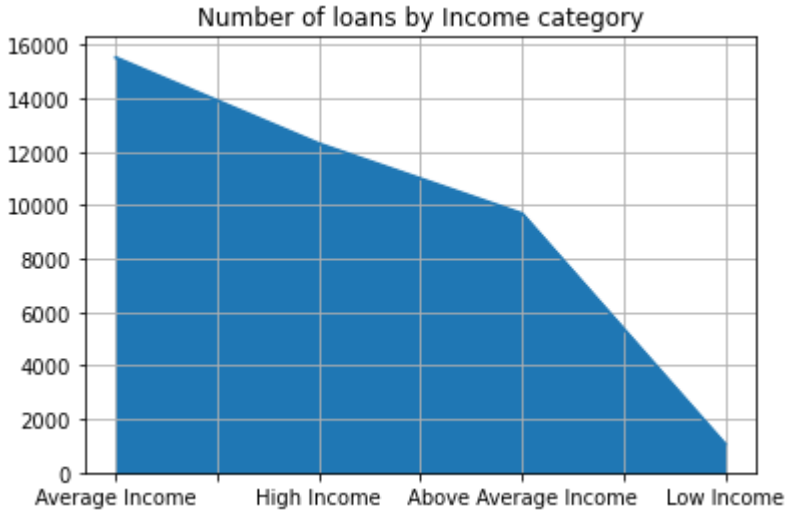


Univariate Analysis of quantitative Variable

In [21]:
```
def bin_annual_salary(sal):
    sal = sal / 1000
    if sal < 20:
        return 'Low Income'
    elif sal < 53:
        return 'Average Income'
    elif sal < 75:
        return 'Above Average Income'
    else:
        return 'High Income'

data['income_category'] = data.annual_inc.apply(bin_annual_salary)
```

```
data.income_category.value_counts().plot.area(grid=True, title='Number of loans by Inc
plt.show()
```

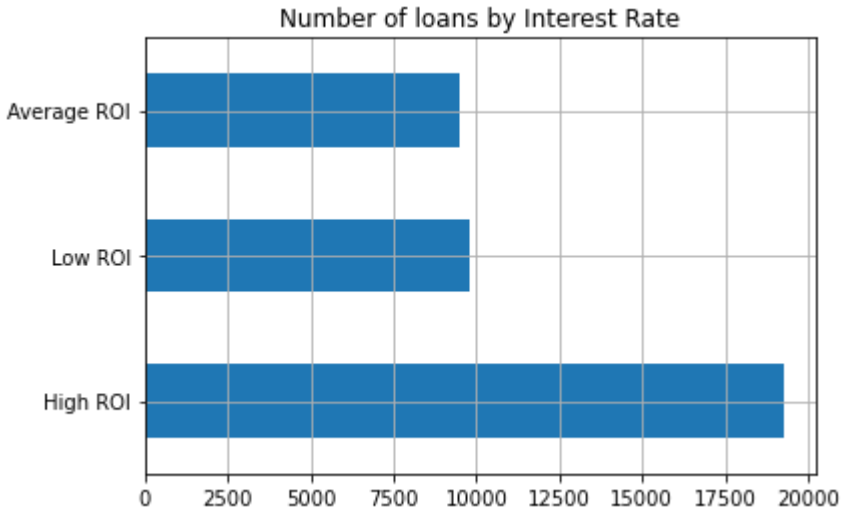
```
C:\Users\dhruv\anaconda3\envs\Finance\lib\site-packages\ipykernel_launcher.py:12: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  if sys.path[0] == '':
```



Univariate Analysis of numerical variable

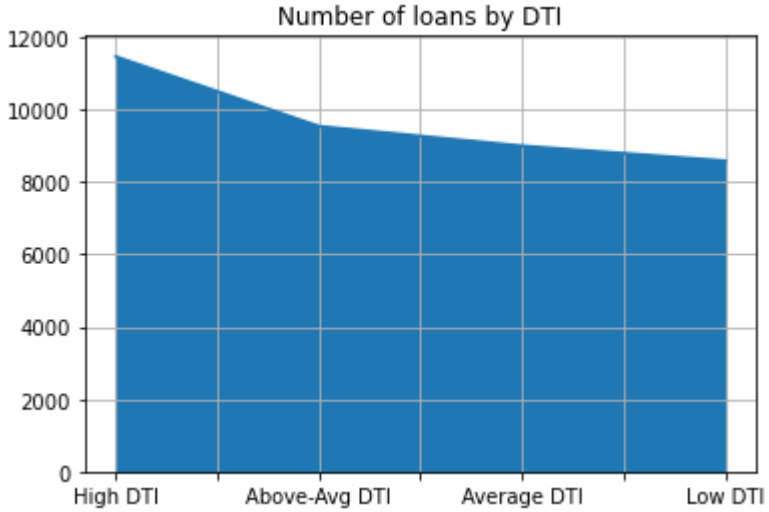
In [22]:
```
data['int_rate_category'] = pd.qcut(data.int_rate, q=[0,0.25,0.50,1],labels=['Low ROI'

data['int_rate_category'].value_counts().plot(kind='barh', grid=True, title='Number of
plt.show()
```

```
C:\Users\dhruv\anaconda3\envs\Finance\lib\site-packages\ipykernel_launcher.py:1: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```



In [23]:
```
def bin_dti(value):
    value = round(value,0)
    if value < 8:
        return 'Low DTI'
    elif value < 13:
        return 'Average DTI'
    elif value < 18:
        return 'Above-Avg DTI'
    else:
        return 'High DTI'

data['dti_category'] = data.dti.apply(bin_dti)
data['dti_category'].value_counts().plot(kind='area', grid=True, title='Number of loar
plt.show()
```

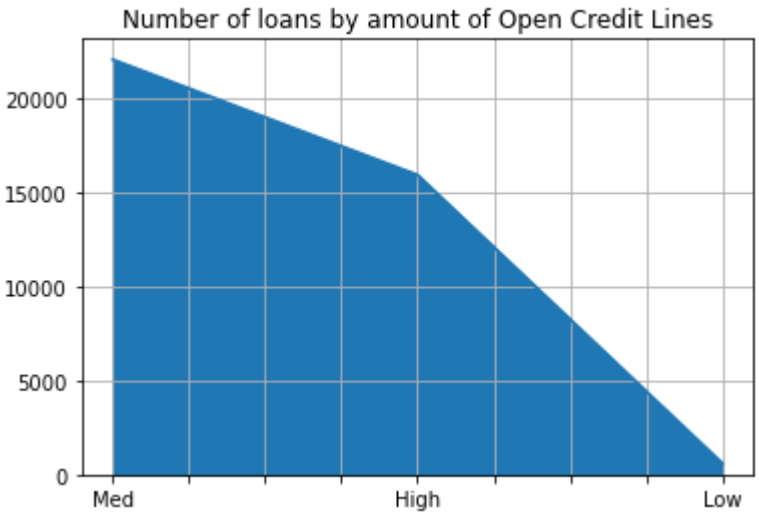
```
C:\Users\dhruv\anaconda3\envs\Finance\lib\site-packages\ipykernel_launcher.py:12: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  if sys.path[0] == '':
```



In [24]:
```
def bin_openacc(value):
    if value <= 2:
```

```
        return "Low"
    elif value <= 9:
        return "Med"
    else:
        return 'High'

data['open_acc_category'] = data.open_acc.apply(bin_openacc)
data['open_acc_category'].value_counts().plot(kind='area', grid=True, title='Number of
plt.show()
```

```
C:\Users\dhruv\anaconda3\envs\Finance\lib\site-packages\ipykernel_launcher.py:9: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  if __name__ == '__main__':
```



In [25]:
```
data.revol_util.describe()
```

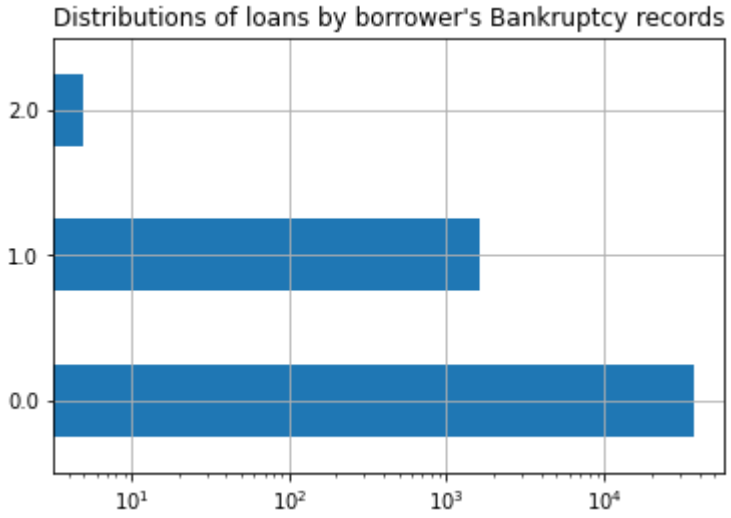
Out[25]:
```
count    38527.000000
mean        48.702777
std         28.364741
min          0.000000
25%         25.200000
50%         49.100000
75%         72.300000
max         99.900000
Name: revol_util, dtype: float64
```

In [26]:
```
# Segment quatitative variable revol_util
data['revol_util_category'] = pd.qcut(data.revol_util, q=[0,0.50,1],labels=['Avg(Revol
```

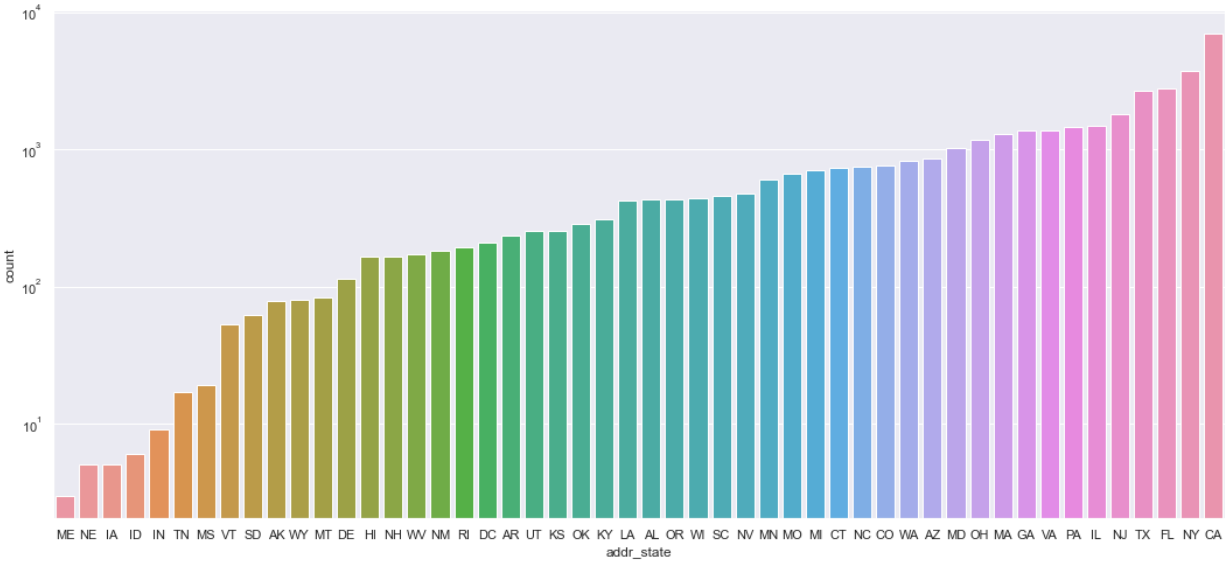
```
C:\Users\dhruv\anaconda3\envs\Finance\lib\site-packages\ipykernel_launcher.py:2: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
```

In [27]:
```
# Lets visualize the number of loans against number of public bankruptcies
```

```
data['pub_rec_bankruptcies'].value_counts().plot(kind='barh', logx=True, grid=True, ti
plt.show()
```



In [28]:
```
# Lets see the distribution of loans by borrower's (address) state
sns.set(style="darkgrid")
df = data[['id','addr_state','loan_status']]
[['id','addr_state','loan_status']]
plt.figure(figsize=(18,8))
ax = sns.countplot(x=df['addr_state'], log=True, order=df['addr_state'].value_counts(a
plt.show()
```



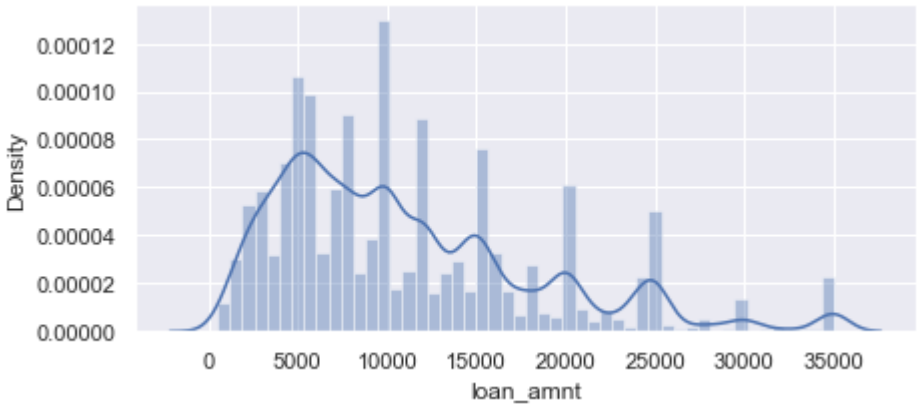
In [29]:
```
df.head()
```

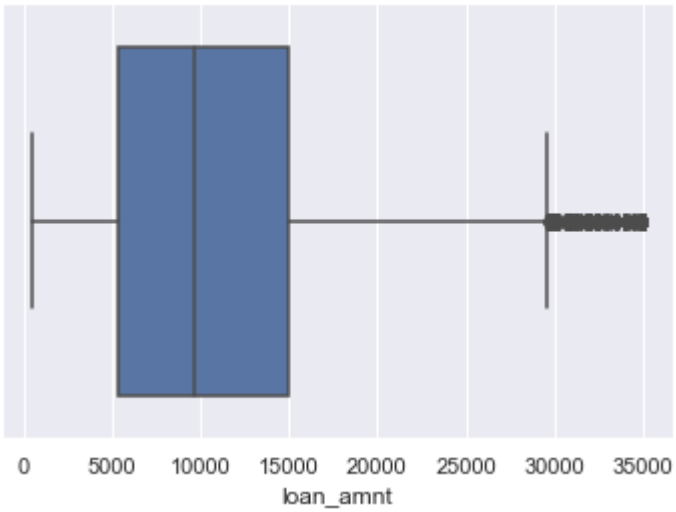
Out[29]:

	id	addr_state	loan_status
0	1077501	AZ	Fully Paid
1	1077430	GA	Charged Off
2	1077175	IL	Fully Paid
3	1076863	CA	Fully Paid
5	1075269	AZ	Fully Paid

In [30]:
```
# analyse loan amount
f, ax = plt.subplots(figsize=(7, 3))
ax = sns.distplot(data['loan_amnt'])
plt.show();
ax = sns.boxplot(x=data['loan_amnt'])
plt.show();
```

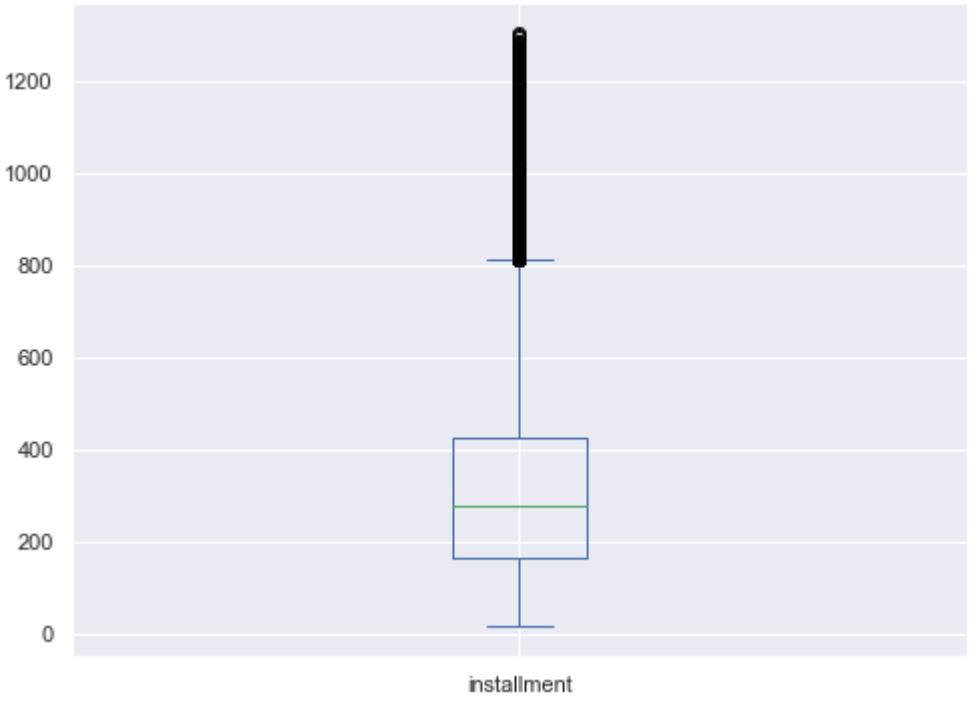
```
C:\Users\dhruv\anaconda3\envs\Finance\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



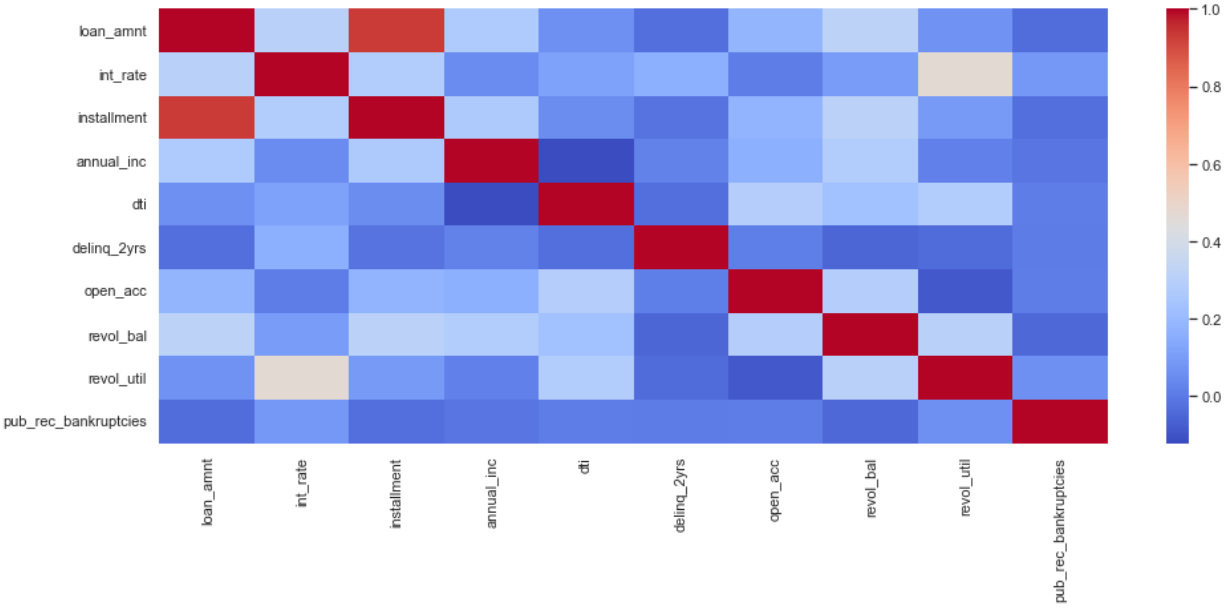


In [31]:
```
# distribution of installment amount
print(data['installment'].describe())
data['installment'].plot(kind='box', figsize=(8,6), grid=True)
plt.show()
```

```
count    38577.000000
mean       322.466318
std        208.639215
min         15.690000
25%        165.740000
50%        277.860000
75%        425.550000
max       1305.190000
Name: installment, dtype: float64
```



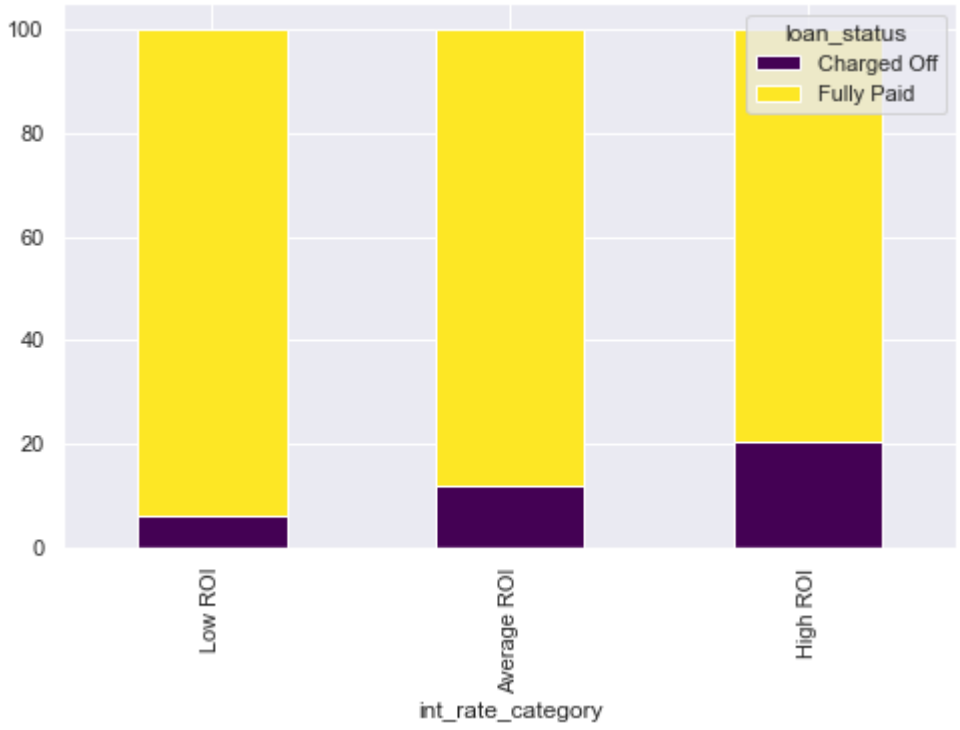
In [32]:
```
# correlation map
col_for_correlation = ['loan_amnt','int_rate', 'installment','annual_inc','dti', 'deli
plt.figure(figsize=(16,6))
corr = data[col_for_correlation].corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values,cmap = 'coolwarm')
plt.show()
```



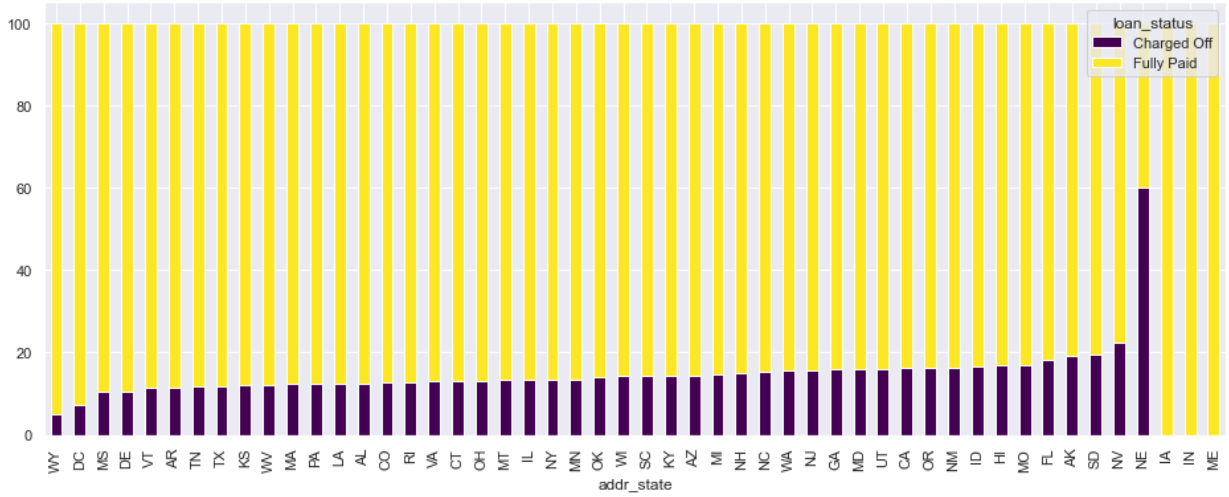
In [33]:
```
# plots the percentage of loans fully paid and charged off as a stacked chart for a gi
def custom_stacked_chart_by_loanstatues(column, size=(8,5), chartkind='bar',theme='vir
    pivot_df = data.pivot_table(index=column, columns='loan_status', values='id',aggfu
    pivot_df = pivot_df.div( pivot_df.iloc[:,-1], axis=0 ).mul(100, axis=0)
    pivot_df.drop(columns='Total', inplace=True)
    pivot_df = pivot_df[0:-1]
    pivot_df.sort_values('Charged Off', inplace=True)
```

```
    pivot_df.plot(kind=chartkind, stacked=True, grid=True, figsize=size, colormap=them
    plt.show()
```

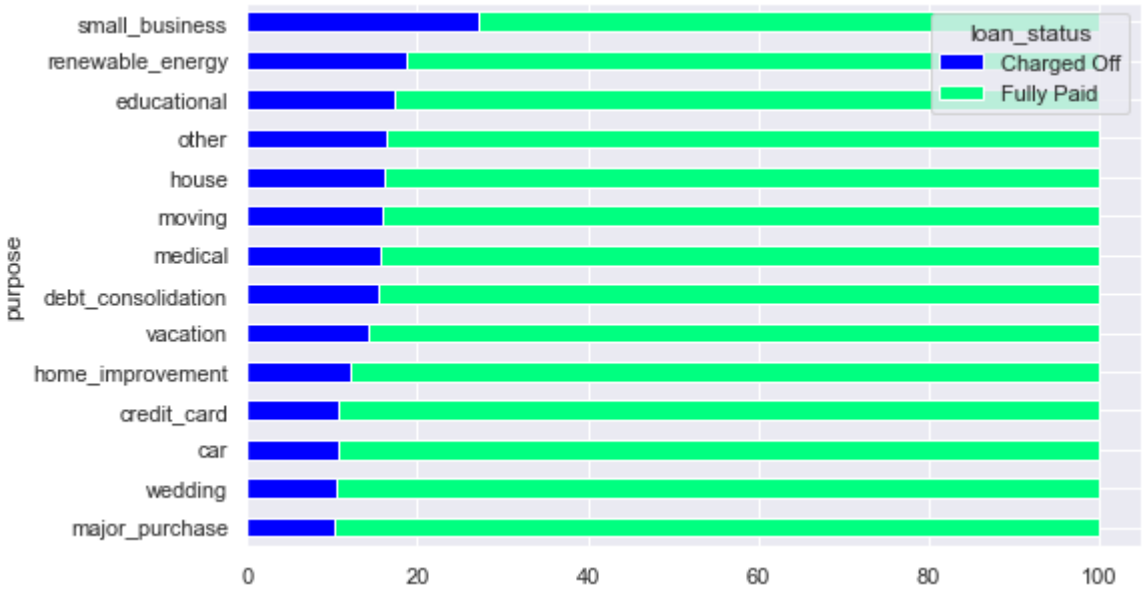
In [34]:
```
# checkout effect of int_rate_category on loan status

custom_stacked_chart_by_loanstatues('int_rate_category')
```



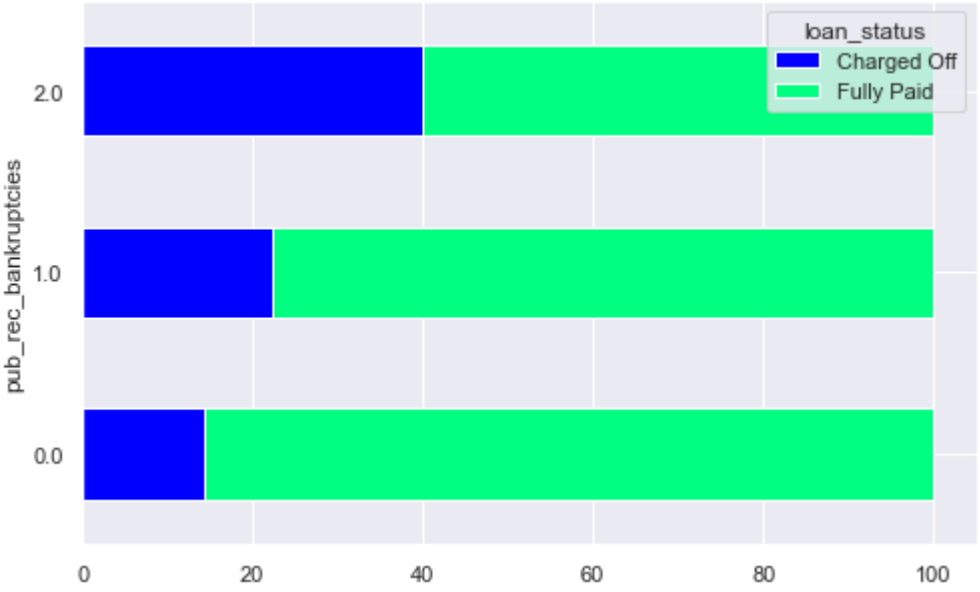
In [35]:
```
custom_stacked_chart_by_loanstatues('addr_state', size=(16,6))
```



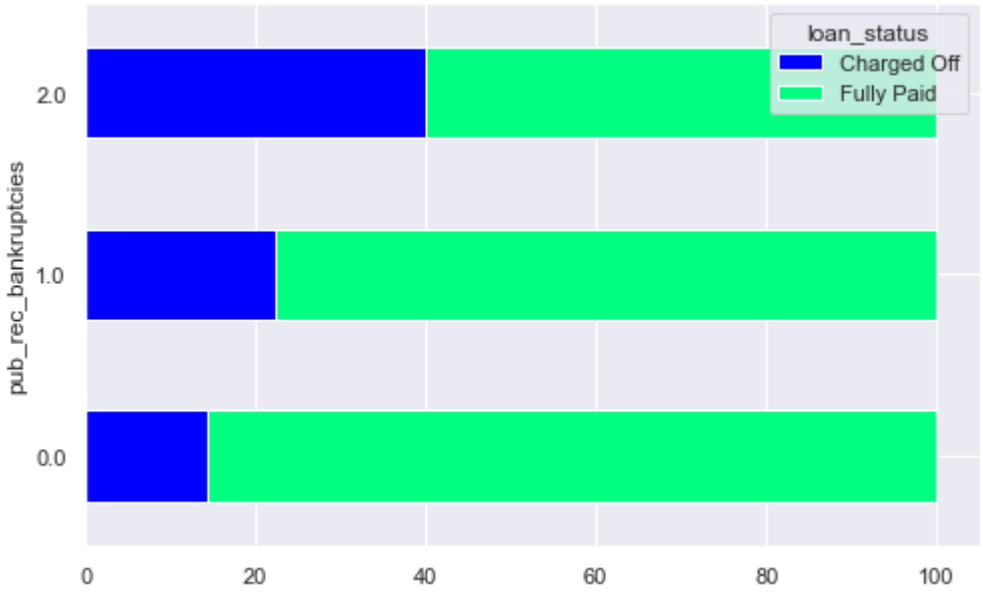
In [36]:
```
custom_stacked_chart_by_loanstatues('purpose', chartkind='barh', theme='winter')
```



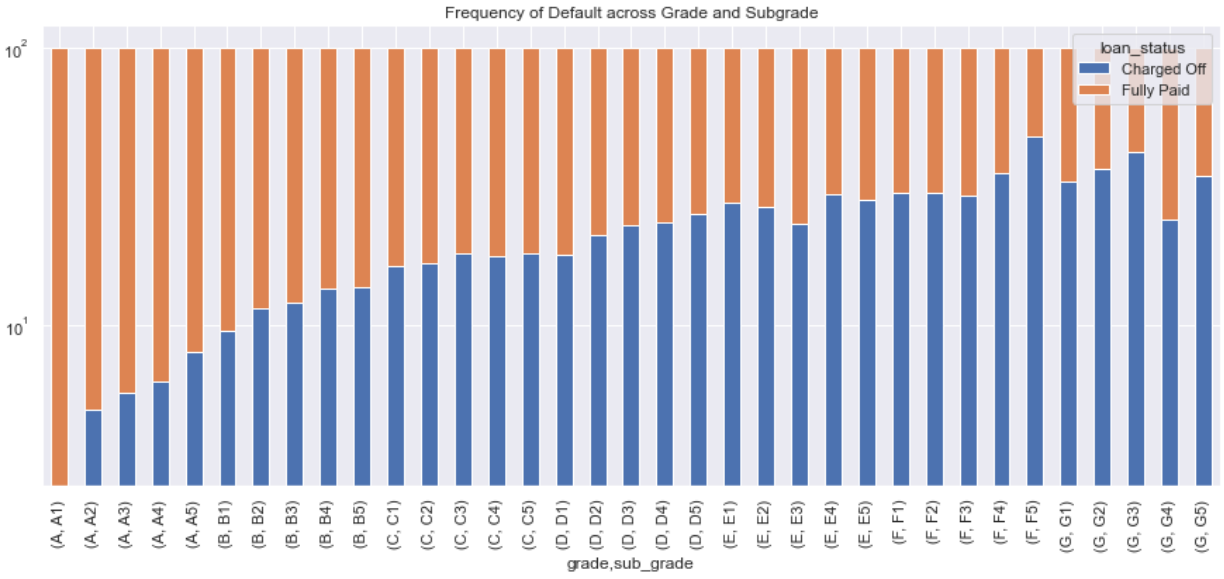
In [37]:
```
custom_stacked_chart_by_loanstatues('pub_rec_bankruptcies', chartkind='barh', theme='w
```



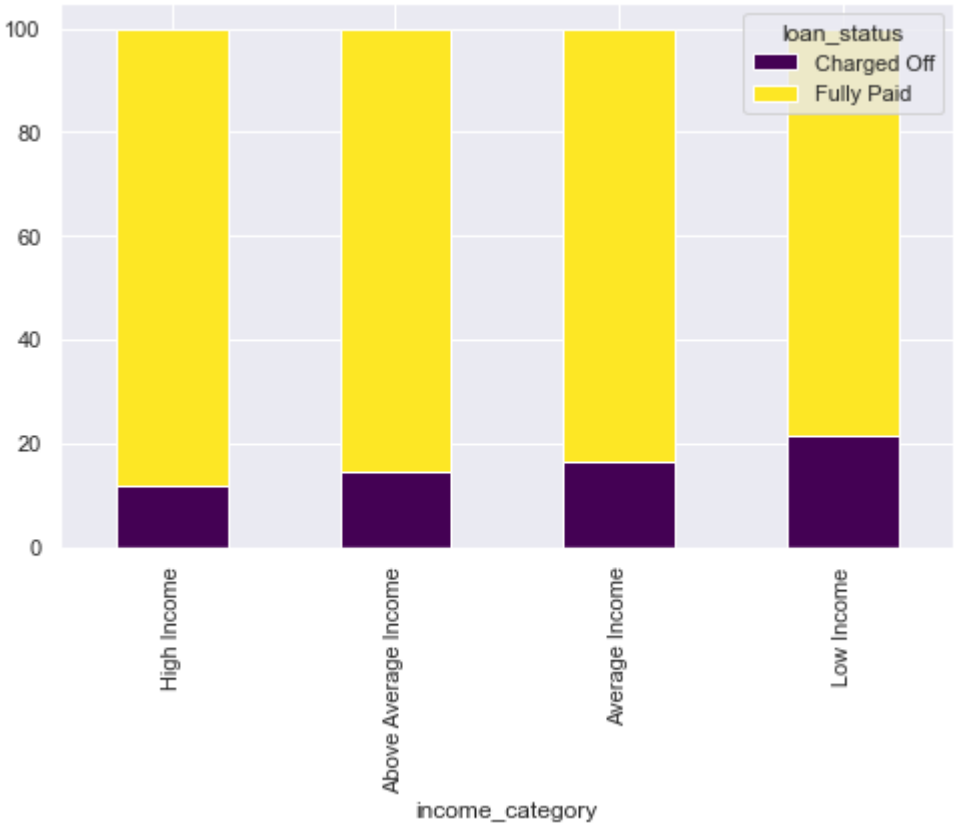
In [38]:
```
custom_stacked_chart_by_loanstatues('pub_rec_bankruptcies', chartkind='barh', theme='w
```



In [39]:
```
pivot_df = data.pivot_table(index=['grade', 'sub_grade'], columns='loan_status', value
pivot_df = pivot_df.div( pivot_df.iloc[:,-1], axis=0 ).mul(100, axis=0)
pivot_df.drop(columns='Total', inplace=True)
pivot_df[0:-1].plot(kind='bar', stacked=True, grid=True, figsize=(15,6), logy=True, ti
plt.show()
```

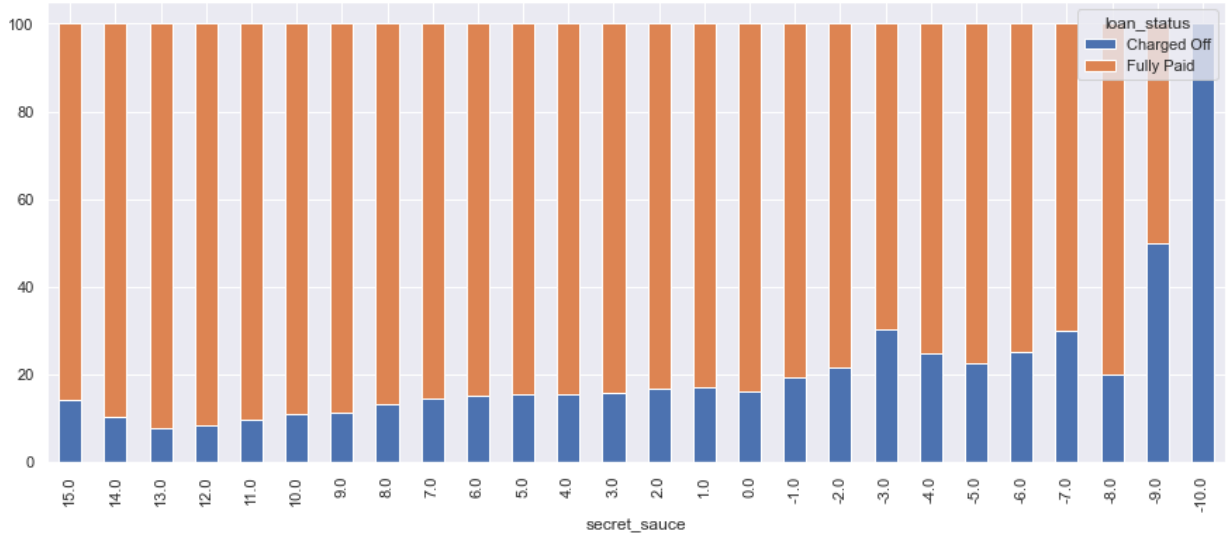


In [40]:
```
#  check the effect of income category on percentage of defaults
custom_stacked_chart_by_loanstatues('income_category')
```



In [41]:
```
custom_df = data[['loan_amnt','term', 'annual_inc','id', 'loan_status','home_ownership

map_home_ownership = {'OWN':2, 'MORTGAGE': 1, 'OTHER':0, 'RENT': 0}

map_income_category = {'Low Income': -1, 'Average Income':1, 'Above Average Income':2,

map_revol_bal = {'Low': -1, 'Average':1, 'Above Average':2, 'High':3}

map_dti = {'Low DTI':1, 'Average DTI':0, 'Above-Avg DTI':-1, 'High DTI':-2}

map_loan_amnt_category = {'Low': 2, 'Average':1, 'Above Average':0, 'High':-1}


custom_df['revol_bal_category'] = pd.qcut(custom_df.revol_bal, q=[0,0.25,0.50,0.75,1],
custom_df['loan_amnt_category'] = pd.qcut(custom_df.loan_amnt, q=[0,0.25,0.50,0.75,1],


def getscore(df):
    score = 0
    # Asset like variables
    score = score + map_home_ownership.get(df.home_ownership,0)
    score = score + round(1.5 * map_revol_bal.get(df.revol_bal_category,0), 0)
    score = score + (2 * map_income_category.get(df.income_category, 0))

    score = score + map_loan_amnt_category.get(df.loan_amnt_category,0)
    score = score + map_dti.get(df.dti_category,0)

    # Penalty like variables
    score = score + (-2 * df.pub_rec_bankruptcies)
    score = score + (-2 * df.pub_rec)
    score = score + (-1 * df.delinq_2yrs)
    return score;
custom_df['secret_sauce'] = custom_df.apply(getscore, axis=1)
```

```
pivot_df = custom_df.pivot_table(index=['secret_sauce'], columns='loan_status', values
pivot_df = pivot_df.div( pivot_df.iloc[:,-1], axis=0 ).mul(100, axis=0)
pivot_df.drop(columns='Total', inplace=True)
pivot_df = pivot_df[0:-1]
pivot_df.sort_values('secret_sauce', ascending=False, inplace=True)
pivot_df.plot(kind='bar', stacked=True, grid=True, figsize=(15,6))
plt.show()
```



Conclusion

After the initial Exploratory Data analysis, we can say that the chances of default increases if:

1. Loan purpose is among Small Business, Other, Renewable Energy and Education
2. Loan records has High ROI (interest rate)
3. If borrowers state is Florida(FL) and Nevada (NV). Other risky states include Nebraska(NE), South Dakota(SD) and Alaska (AK)
4. Worse application Grade and Subgrade
5. Number of Public bankruptcies
6. A negative score on secret_sauce indicates high chances of default.