predicting-loan-defaults

November 26, 2023

Financial institutions incur significant losses due to the default of vehicle loans. This has led to the tightening up of vehicle loan underwriting and increased vehicle loan rejection rates. The need for a better credit risk scoring model is also raised by these institutions. This warrants a study to estimate the determinants of vehicle loan default.

There is 1 dataset data that have 41 attributes. You are required to determine and examine factors that affected the ratio of vehicle loan defaulters. Also, use the findings to create a model to predict the potential defaulter

[2]:
```
# Import libraries necessary for this project
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings(action='ignore') #to ignore warnings
pd.set_option('display.max_columns', None) #to display all columns
pd.set_option('display.max_rows',None) #to display all rows
```

[4]:
```
df = pd.read_csv("loan_CSV.csv") # Loading dataset
```

[5]:
```
df.dropna(axis=1, how='all', inplace= True) # to drop columns with all null␣
 ↪values
df = df.loc[:, df.nunique().values!=1] # to take only those rows and columns␣
 ↪which are not havinig unique values
```

[6]:
```
#As we do not have to deal with those whose loan_status is Current i.e active,␣
 ↪so take only those whose loan status is not current
df = df[df.loan_status!='Current']
```

[7]:
```
df.head()
```

[7]:
```
        id  member_id  loan_amnt  funded_amnt_inv        term int_rate  \
0  1077501    1296599       5000           4975.0   36 months   10.65%
1  1077430    1314167       2500           2500.0   60 months   15.27%
2  1077175    1313524       2400           2400.0   36 months   15.96%
3  1076863    1277178      10000          10000.0   36 months   13.49%
5  1075269    1311441       5000           5000.0   36 months    7.90%

   installment grade sub_grade             emp_title emp_length  \
```

```
0       162.87     B        B2                    NaN  10+ years
1        59.83     C        C4                  Ryder   < 1 year
2        84.33     C        C5                    NaN  10+ years
3       339.31     C        C1   AIR RESOURCES BOARD  10+ years
5       156.46     A        A4  Veolia Transportaton    3 years

  home_ownership  annual_inc verification_status issue_d  loan_status  \
0           RENT     24000.0            Verified  Dec-11   Fully Paid
1           RENT     30000.0     Source Verified  Dec-11  Charged Off
2           RENT     12252.0        Not Verified  Dec-11   Fully Paid
3           RENT     49200.0     Source Verified  Dec-11   Fully Paid
5           RENT     36000.0     Source Verified  Dec-11   Fully Paid

          purpose                                   title  revol_bal revol_util
0     credit_card                                Computer      13648     83.70%
1             car                                    bike       1687      9.40%
2  small_business                    real estate business       2956     98.50%
3           other                                personel       5598        21%
5         wedding  My wedding loan I promise to pay back       7963     28.30%
```

[8]:
```
##Data Exploration
```

[9]:
```
df.shape
```

[9]:
```
(38577, 20)
```

[10]:
```
#to Check null and unique values in parallel
null_unique = pd.DataFrame()
null_unique['nulls'] = pd.Series(df.isnull().sum())
null_unique['unique'] = pd.Series(df.nunique())
null_unique
```

[10]:
```
                      nulls  unique
id                        0   38577
member_id                 0   38577
loan_amnt                 0     870
funded_amnt_inv           0    8050
term                      0       2
int_rate                  0     370
installment               0   15022
grade                   156       7
sub_grade                 0      35
emp_title              2386   28027
emp_length             1033      11
home_ownership            0       5
annual_inc                0    5215
verification_status       0       3
```

```
issue_d                   0      55
loan_status               0       2
purpose                   0      14
title                    11   19297
revol_bal                 0   21275
revol_util               50    1088
```

[11]:
```
round((df.isnull().sum()/df.shape[0])*100,2)
```

[11]:
```
id                      0.00
member_id               0.00
loan_amnt               0.00
funded_amnt_inv         0.00
term                    0.00
int_rate                0.00
installment             0.00
grade                   0.40
sub_grade               0.00
emp_title               6.19
emp_length              2.68
home_ownership          0.00
annual_inc              0.00
verification_status     0.00
issue_d                 0.00
loan_status             0.00
purpose                 0.00
title                   0.03
revol_bal               0.00
revol_util              0.13
dtype: float64
```

[12]:
```
df[df.duplicated()]
```

[12]:
```
Empty DataFrame
Columns: [id, member_id, loan_amnt, funded_amnt_inv, term, int_rate,
installment, grade, sub_grade, emp_title, emp_length, home_ownership,
annual_inc, verification_status, issue_d, loan_status, purpose, title,
revol_bal, revol_util]
Index: []
```

[13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 38577 entries, 0 to 39716
Data columns (total 20 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
```

```
 0   id                    38577 non-null  int64
 1   member_id             38577 non-null  int64
 2   loan_amnt             38577 non-null  int64
 3   funded_amnt_inv       38577 non-null  float64
 4   term                  38577 non-null  object
 5   int_rate              38577 non-null  object
 6   installment           38577 non-null  float64
 7   grade                 38421 non-null  object
 8   sub_grade             38577 non-null  object
 9   emp_title             36191 non-null  object
 10  emp_length            37544 non-null  object
 11  home_ownership        38577 non-null  object
 12  annual_inc            38577 non-null  float64
 13  verification_status   38577 non-null  object
 14  issue_d               38577 non-null  object
 15  loan_status           38577 non-null  object
 16  purpose               38577 non-null  object
 17  title                 38566 non-null  object
 18  revol_bal             38577 non-null  int64
 19  revol_util            38527 non-null  object
dtypes: float64(3), int64(4), object(13)
memory usage: 6.2+ MB
```

[14]:
```
df.describe()
```

[14]:

	id	member_id	loan_amnt	funded_amnt_inv	\
count	3.857700e+04	3.857700e+04	38577.000000	38577.000000	
mean	6.763787e+05	8.422843e+05	11047.025430	10222.481123	
std	2.092639e+05	2.644519e+05	7348.441646	7022.720644	
min	5.473400e+04	7.069900e+04	500.000000	0.000000	
25%	5.120330e+05	6.611310e+05	5300.000000	5000.000000	
50%	6.564230e+05	8.392920e+05	9600.000000	8733.440000	
75%	8.291460e+05	1.037336e+06	15000.000000	14000.000000	
max	1.077501e+06	1.314167e+06	35000.000000	35000.000000	

	installment	annual_inc	revol_bal
count	38577.000000	3.857700e+04	38577.000000
mean	322.466318	6.877797e+04	13289.489826
std	208.639215	6.421868e+04	15866.492241
min	15.690000	4.000000e+03	0.000000
25%	165.740000	4.000000e+04	3650.000000
50%	277.860000	5.886800e+04	8762.000000
75%	425.550000	8.200000e+04	16912.000000
max	1305.190000	6.000000e+06	149588.000000

[15]:
```
#Data Cleaning
```

[16]:
```
df.columns
```

[16]:
```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt_inv', 'term', 'int_rate',
       'installment', 'grade', 'sub_grade', 'emp_title', 'emp_length',
       'home_ownership', 'annual_inc', 'verification_status', 'issue_d',
       'loan_status', 'purpose', 'title', 'revol_bal', 'revol_util'],
      dtype='object')
```

[17]:
```
remove_col = ['id', 'member_id','emp_title',␣
 ↪'title','term','installment','sub_grade','issue_d','revol_bal','revol_util']
df.drop(axis=1,labels=remove_col,inplace=True)
```

[18]:
```
df.head()
```

[18]:

	loan_amnt	funded_amnt_inv	int_rate	grade	emp_length	home_ownership	\
0	5000	4975.0	10.65%	B	10+ years	RENT	
1	2500	2500.0	15.27%	C	< 1 year	RENT	
2	2400	2400.0	15.96%	C	10+ years	RENT	
3	10000	10000.0	13.49%	C	10+ years	RENT	
5	5000	5000.0	7.90%	A	3 years	RENT	

	annual_inc	verification_status	loan_status	purpose
0	24000.0	Verified	Fully Paid	credit_card
1	30000.0	Source Verified	Charged Off	car
2	12252.0	Not Verified	Fully Paid	small_business
3	49200.0	Source Verified	Fully Paid	other
5	36000.0	Source Verified	Fully Paid	wedding

[19]:
```
#to print null and unique values in parallel
null_unique = pd.DataFrame()
null_unique['nulls'] = pd.Series(df.isnull().sum())
null_unique['unique'] = pd.Series(df.nunique())
print(null_unique)
print(df.shape)
```

```
                     nulls  unique
loan_amnt                0     870
funded_amnt_inv          0    8050
int_rate                 0     370
grade                  156       7
emp_length            1033      11
home_ownership           0       5
annual_inc               0    5215
verification_status      0       3
loan_status              0       2
purpose                  0      14
(38577, 10)
```

[21]:
```
## Data Analysis
```

[22]:
```
df['loan_status'].value_counts()
```

[22]:
```
Fully Paid     32950
Charged Off     5627
Name: loan_status, dtype: int64
```

[23]:
```
df.columns
```

[23]:
```
Index(['loan_amnt', 'funded_amnt_inv', 'int_rate', 'grade', 'emp_length',
       'home_ownership', 'annual_inc', 'verification_status', 'loan_status',
       'purpose'],
      dtype='object')
```

[24]:
```
df['verification_status'].value_counts()
```

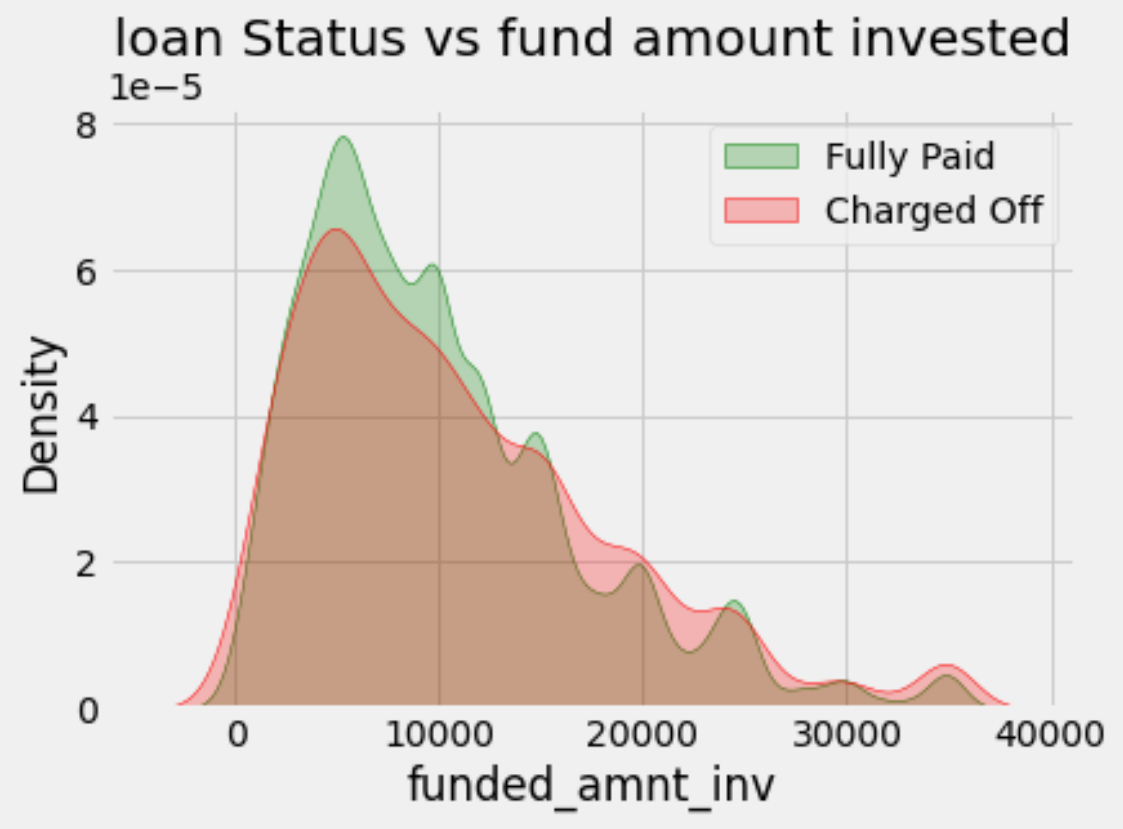
[24]:
```
Not Verified       16694
Verified           12206
Source Verified     9677
Name: verification_status, dtype: int64
```

[25]:
```
#Bivariate Analysis
```

[26]:
```
# function to check behaviour of different columns with loan status

def proportion_plot(data, feature, title):
    #creating list of unique values of feature of dataset
    values_list = data[feature].unique().to_list()
    values_list.sort()
    value_prop = {}
    for value in values_list:
        prop = len(data[(data[feature] == value) & (data['loan_status'] ==␣
 ↪'Charged Off')].index) / len(data[(data[feature] == value)
                                                                          ␣
 ↪                 & (data['loan_status'] == 'Fully Paid')].index) * 100
        value_prop[value] = (round(prop,2))

    sns.set_theme(style='whitegrid')
    ax = sns.barplot(x = list(value_prop.keys()), y = list(value_prop.
 ↪values()), palette='colorblind')
    ax.bar_label(ax.containers[0])
    plt.xlabel(feature)
    plt.ylabel('percent value')
    plt.xticks(rotation = 45)
    plt.title('Defaulters proportion vs ' + title)
    #plt.show()
```

[27]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[28]:
```
sns.kdeplot(data = df[df['loan_status'] == 'Fully Paid'], x =␣
 ↪'funded_amnt_inv', shade=True, color='green')
sns.kdeplot(data = df[df['loan_status'] == 'Charged Off'], x =␣
 ↪'funded_amnt_inv', shade = True, color='red')
plt.legend(['Fully Paid', 'Charged Off'])
plt.title('loan Status vs fund amount invested')
plt.show()
```



[29]:
```
#checking datatypes of all columns
datatp = pd.DataFrame()
datatp['Original DataTypes'] = pd.Series(df.dtypes)
datatp.transpose()
```

[29]:

	loan_amnt	funded_amnt_inv	int_rate	grade	emp_length	\
Original DataTypes	int64	float64	object	object	object	

	home_ownership	annual_inc	verification_status	loan_status	\
Original DataTypes	object	float64	object	object	

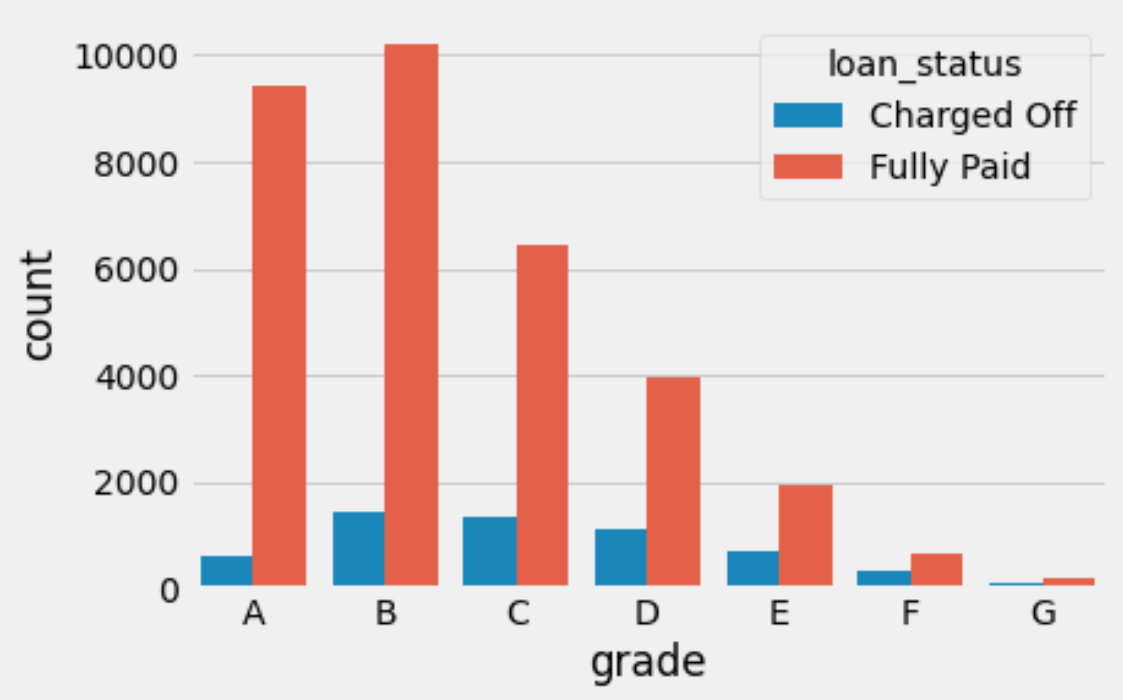
```
                        purpose
Original DataTypes       object
```

[30]:
```
#Altering datatypes
category_col = ['grade', 'home_ownership', 'verification_status', 'loan_status',
 'purpose']
df[category_col] = df[category_col].astype('category')
datatp['Altered Datatypes'] = pd.Series(df.dtypes)
datatp.transpose()
```

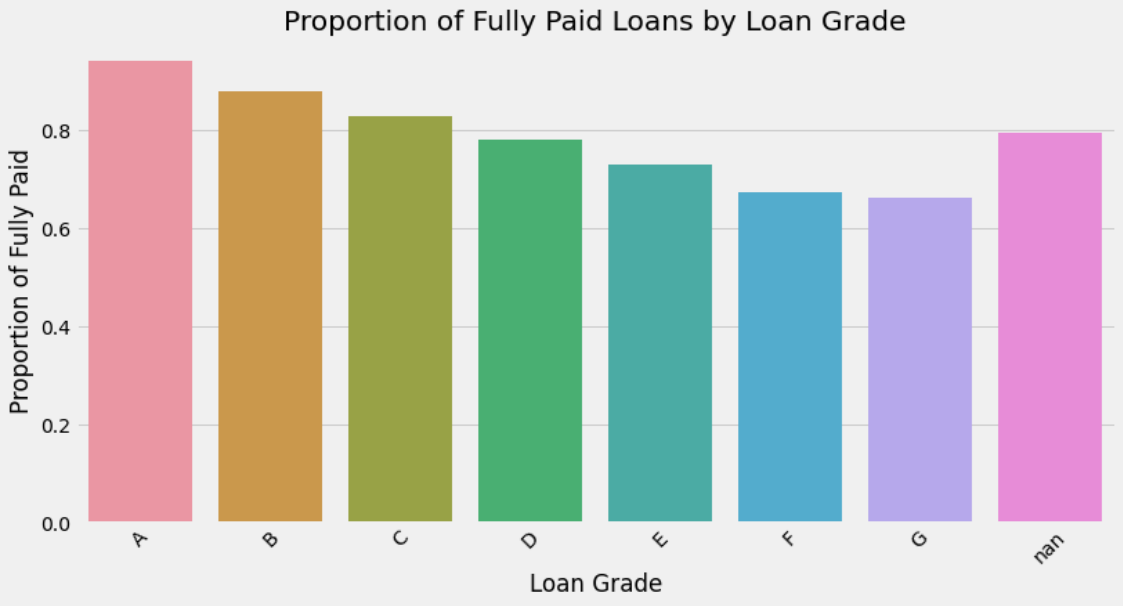
[30]:
```
                    loan_amnt funded_amnt_inv int_rate     grade emp_length  \
Original DataTypes      int64         float64   object    object     object
Altered Datatypes       int64         float64   object  category     object

                    home_ownership annual_inc verification_status loan_status  \
Original DataTypes          object    float64               object      object
Altered Datatypes         category    float64            category    category

                      purpose
Original DataTypes     object
Altered Datatypes    category
```

[31]:
```
df['int_rate'] = pd.to_numeric(df['int_rate'], errors='coerce')
```

[32]:
```
sns.countplot(data = df, x = 'grade', hue = 'loan_status')
plt.show()
```

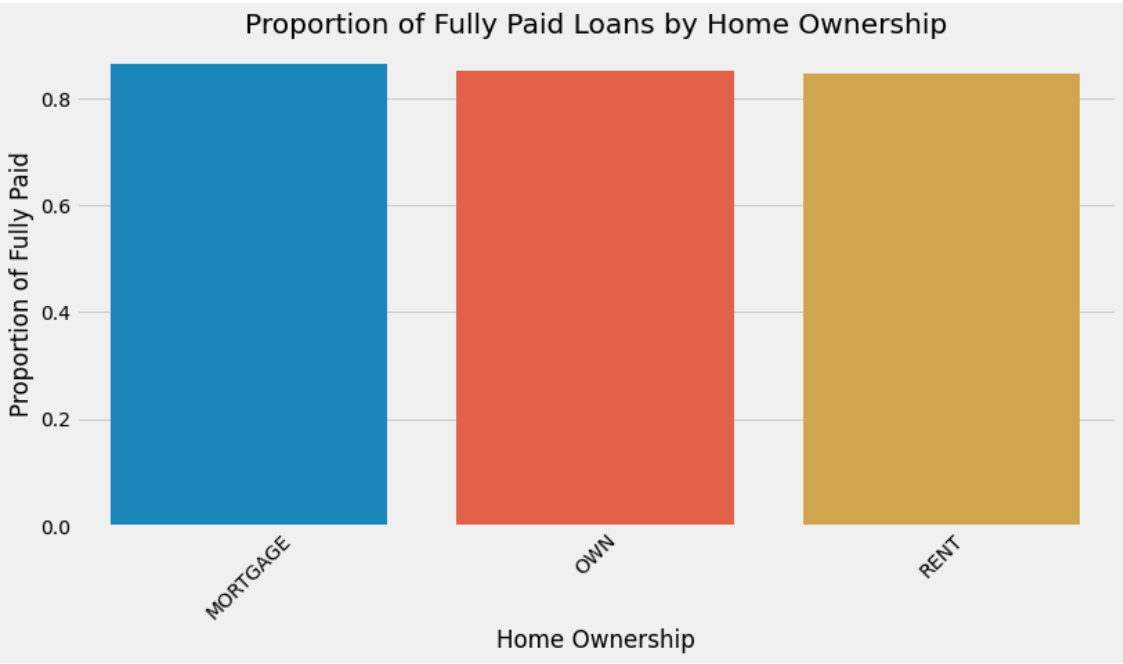


[33]:
```
import seaborn as sns
import matplotlib.pyplot as plt

def proportion_plot(data, feature, title):
    # Check if the feature column contains numeric values and convert them to␣
 ↪strings
    if data[feature].dtype != 'object':
        data[feature] = data[feature].astype(str)

    # Creating a list of unique values of the feature in the dataset
    values_list = data[feature].unique().tolist()
    values_list.sort()

    value_prop = {}
    for value in values_list:
        value_count = data[data[feature] == value]['loan_status'].value_counts()
        total_count = len(data[data[feature] == value])
        prop = value_count / total_count
        value_prop[value] = prop

    # Create a bar plot
    plt.figure(figsize=(12, 6))
    sns.barplot(x=values_list, y=[value_prop[value]['Fully Paid'] for value in␣
 ↪values_list])
    plt.xlabel(title)
    plt.ylabel('Proportion of Fully Paid')
    plt.title(f'Proportion of Fully Paid Loans by {title}')
    plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
    plt.show()

# Example usage
proportion_plot(df, 'grade', 'Loan Grade')
plt.savefig('grade_vs_loan_status')
```




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

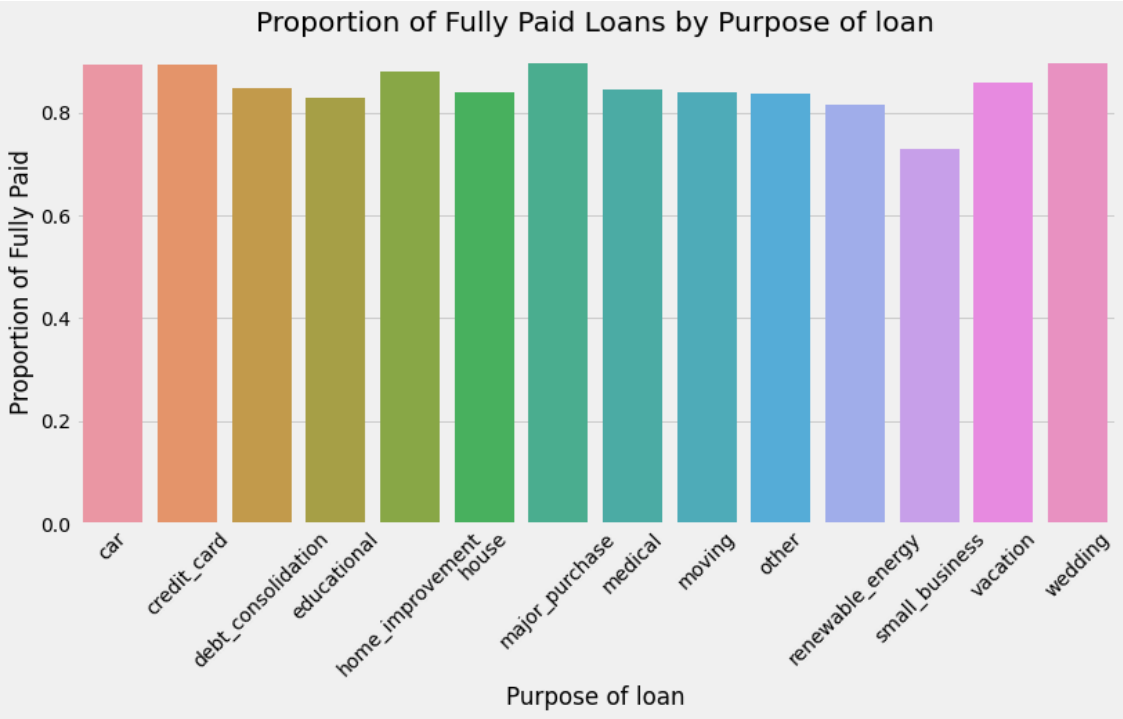
[34]:
```
proportion_plot(df[df['home_ownership'].isin(['MORTGAGE', 'RENT', 'OWN'])],␣
↪'home_ownership', 'Home Ownership')
plt.savefig('home_ownership')
```



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

[35]:
```
plt.figure(figsize=(4,2))
proportion_plot(df, 'purpose', 'Purpose of loan')
plt.savefig('purpose')
```

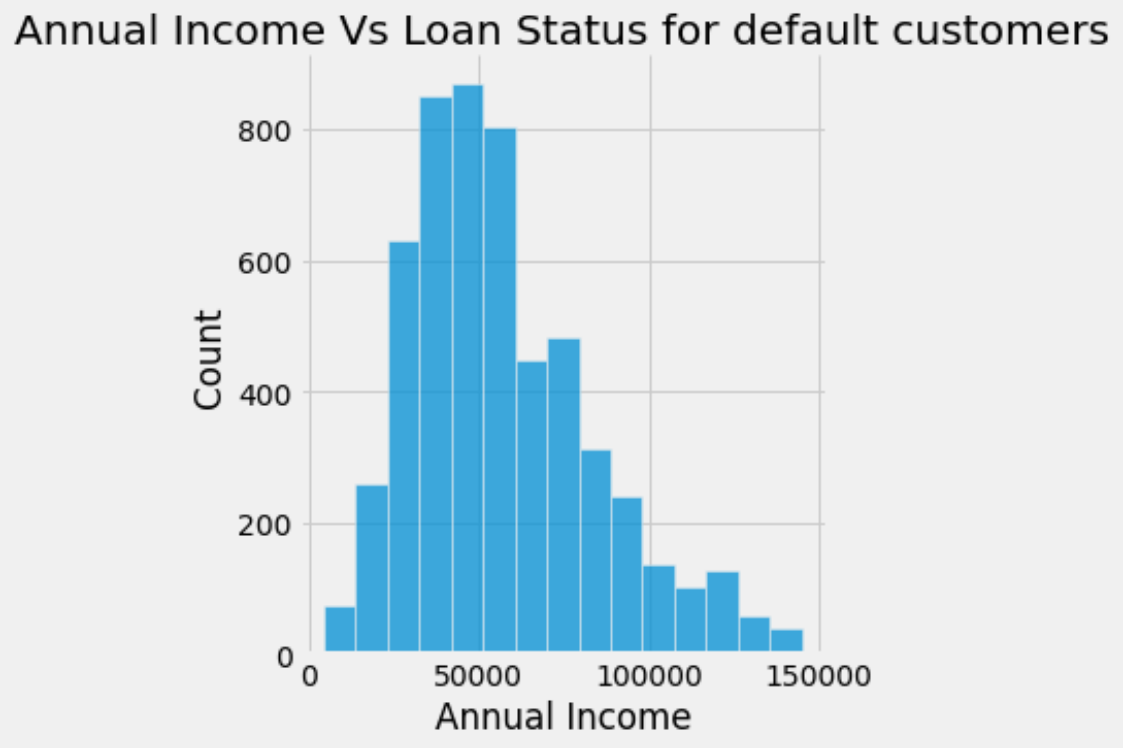
```
<Figure size 288x144 with 0 Axes>
```

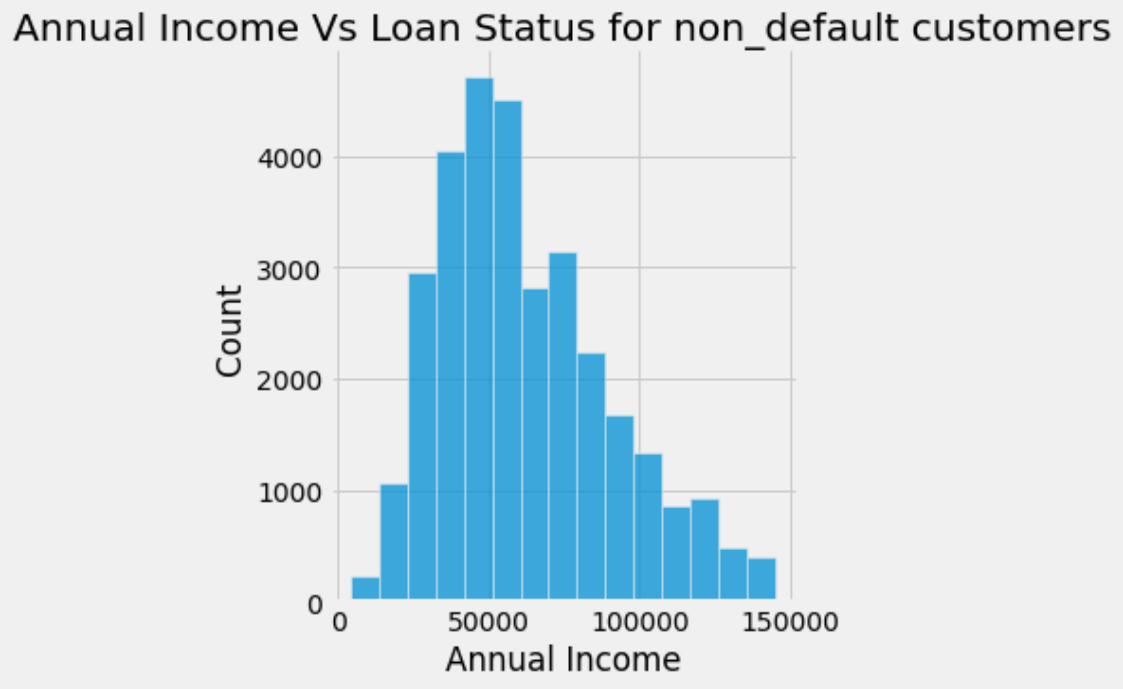



```
<Figure size 432x288 with 0 Axes>
```

[36]:
```
# Checking distribution of employee salary after removing outliers for each␣
 ↪loan status category

# removing outliers for analysis (above 1.5 times of IQR + 75th percentile␣
 ↪value)
income_iqr = np.percentile(df['annual_inc'], 75) - np.
 ↪percentile(df['annual_inc'], 25)
#calculatin above limit for income
income_limit = np.percentile(df['annual_inc'], 75) + 1.5*income_iqr

df1 = df[df['annual_inc'] <= income_limit]
sns.displot(data = df1[df1['loan_status'] == 'Charged Off'], x = 'annual_inc',␣
 ↪bins=15, )
plt.title('Annual Income Vs Loan Status for default customers')
```

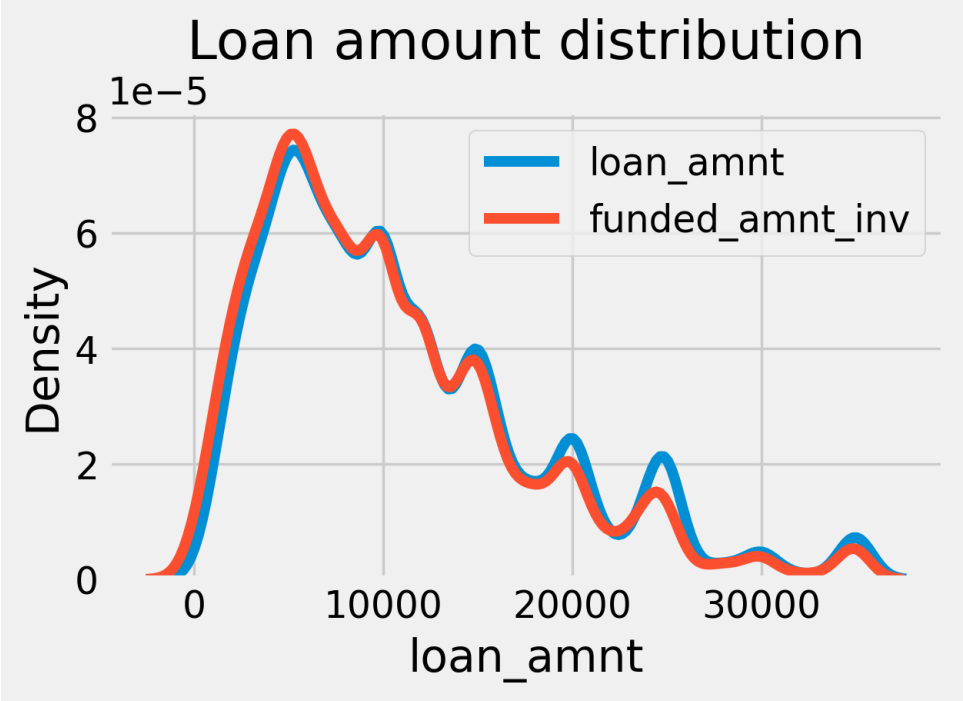
```
plt.xlabel('Annual Income')
plt.savefig('income_c_vs_loan_status')
sns.displot(data = df1[df1['loan_status'] == 'Fully Paid'], x = 'annual_inc',␣
 ↪bins=15, )
plt.title('Annual Income Vs Loan Status for non_default customers')
plt.xlabel('Annual Income')
plt.savefig('income_p_vs_loan_status')
plt.show()
```



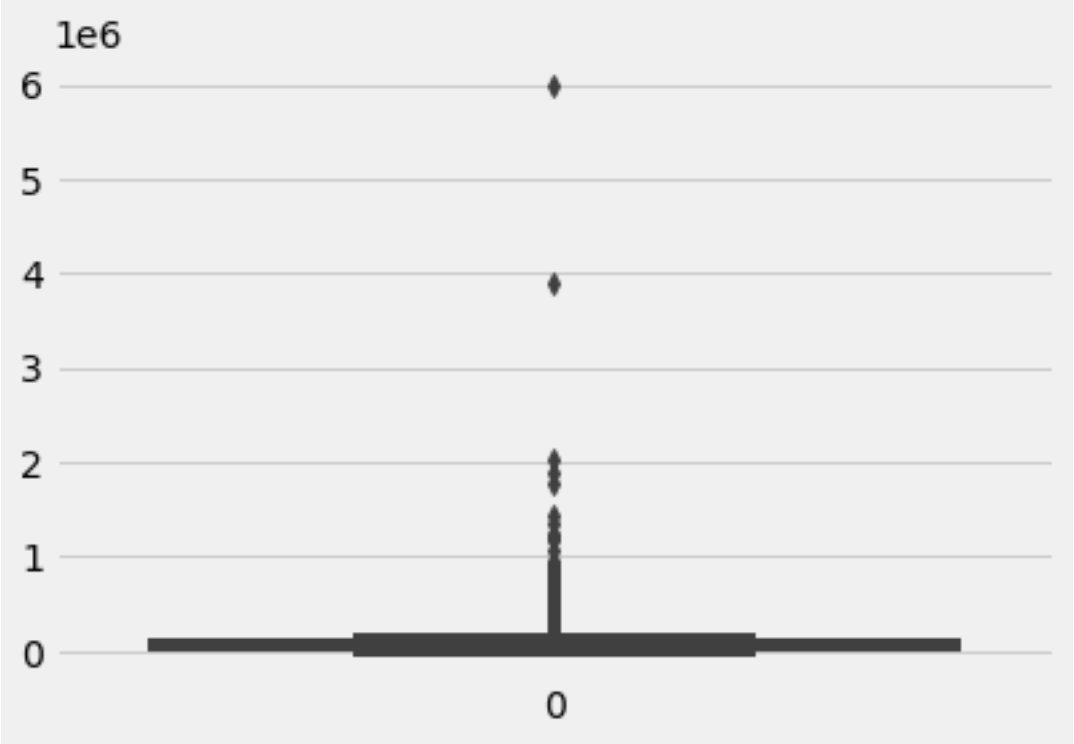


[37]:
```
#Univariate Analysis
```

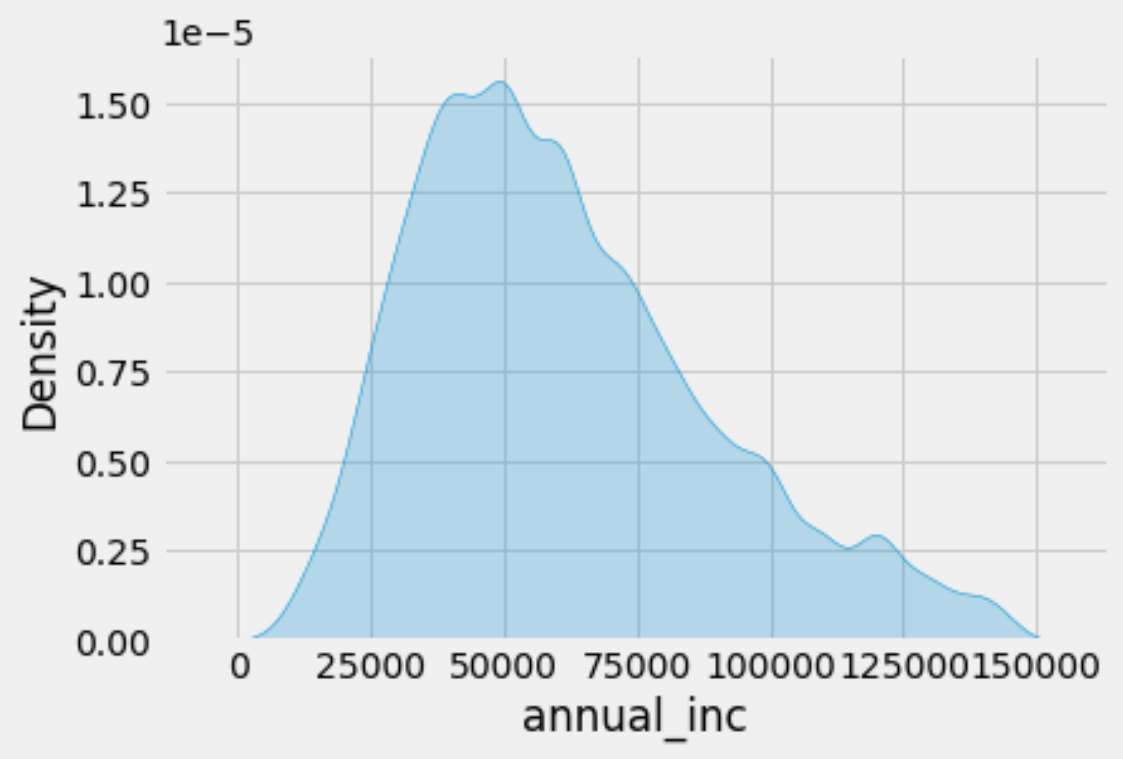
[38]:
```
# checking distribution of loan_amnt, funded_amnt, funded_amnt_inv
funding_variable = ['loan_amnt', 'funded_amnt_inv']
plt.figure(figsize=(5,3), dpi = 250)
for i,j in enumerate(funding_variable):
    mean = df[j].mean()
    median = df[j].median()
    mini = np.min(df[j])
    maxi = np.max(df[j])

    #ax = plt.subplot(1,3,i+1)
    sns.kdeplot(df[j])

plt.title('Loan amount distribution')
plt.legend(funding_variable)
plt.savefig('loan_amt_distribution')
plt.show()
```



[39]:
```
#Checking distribution of annual income of customers
sns.boxplot(df['annual_inc'])
plt.show()
```



[40]:
```
#distribution of income after removing outliers
sns.kdeplot(data = df[df['annual_inc'] <= income_limit], x = 'annual_inc',␣
 ↪shade=True)
plt.show()
```



[41]:
```
df['annual_inc'].describe()
```

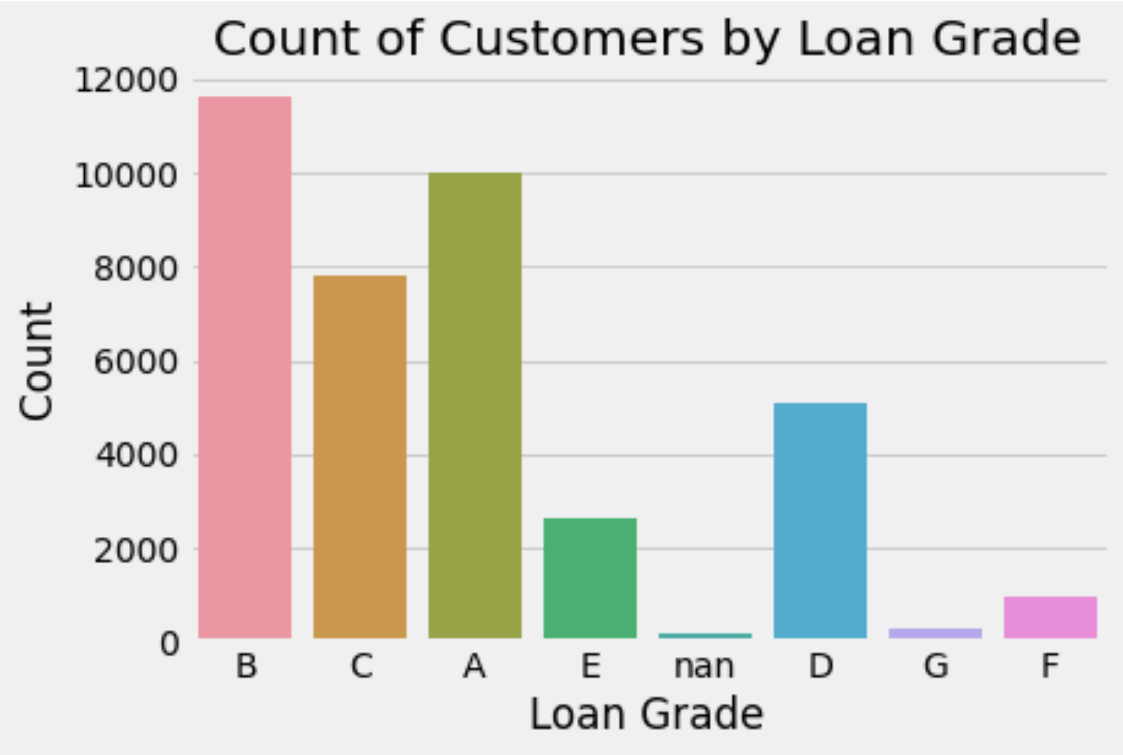
[41]:
```
count    3.857700e+04
mean     6.877797e+04
std      6.421868e+04
min      4.000000e+03
25%      4.000000e+04
50%      5.886800e+04
75%      8.200000e+04
max      6.000000e+06
Name: annual_inc, dtype: float64
```

[42]:
```
df.columns
```

[42]:
```
Index(['loan_amnt', 'funded_amnt_inv', 'int_rate', 'grade', 'emp_length',
       'home_ownership', 'annual_inc', 'verification_status', 'loan_status',
       'purpose'],
      dtype='object')
```

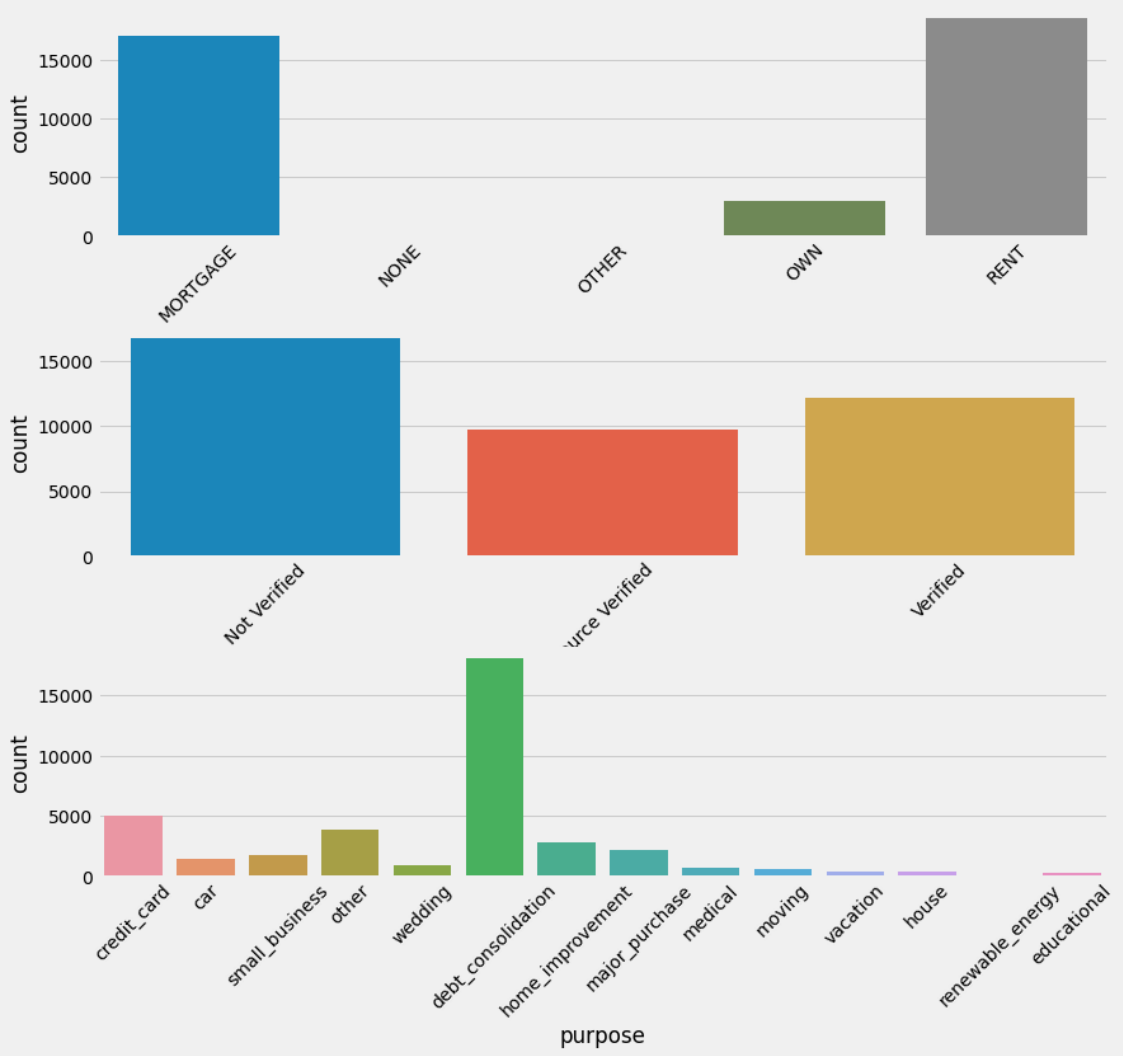
[44]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[45]:
```
# Count of customers based on the rate of interest the loan has been disbursed.
sns.countplot(data=df, x='grade')
plt.xlabel('Loan Grade')
plt.ylabel('Count')
plt.title('Count of Customers by Loan Grade')
plt.show()
```



[46]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Identifying the number of customers based on demography, home ownership,␣
 ↪purpose of loan, and verification status.
# These are the attributes of the customer available before the issuance of the␣
 ↪loan.
attributes_col = ['home_ownership', 'verification_status', 'purpose']
```

```
plt.figure(figsize=(13, 20))
for i, attr in enumerate(attributes_col):
    plt.subplot(6, 1, i + 1)
    sns.countplot(data=df, x=attr)
    plt.xticks(rotation=45)
    plt.subplots_adjust(left=0.1, bottom=0.2, top=1.2, wspace=0.4, hspace=0.4)
    plt.savefig(f'{attr}.png')

plt.show()
```



[ ]: