Part_I_exploration_template

February 17, 2023

1 Part I - (Dataset Exploration Title)

1.1 by (Ruqayyah)

1.2 Introduction

Introduce the dataset

This data set contains 113,937 loans with 81 variables on each loan, including loan amount, borrower rate (or interest rate),current loan status, borrower income, and many others.

Some quetions can be asked for exploration. 1. What factors affect a loan's outcome status? 2. What affects the borrower's APR or interest rate? 3. Are there differences between loans depending on how large the original loan amount was?

1.3 Preliminary Wrangling

```
In [1]: # import all packages and set plots to be embedded inline
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns

        %matplotlib inline
```

Load in your dataset and describe its properties through the questions below. Try and motivate your exploration goals through this section.

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

```
Out[2]:                 ListingKey  ListingNumber            ListingCreationDate  \
        0  1021339766868145413AB3B         193129  2007-08-26 19:09:29.263000000
        1  10273602499503308B223C1        1209647  2014-02-27 08:28:07.900000000
        2  0EE9337825851032864889A          81716  2007-01-05 15:00:47.090000000
        3  0EF5356002482715299901A         658116  2012-10-22 11:02:35.010000000
        4  0F023589499656230C5E3E2         909464  2013-09-14 18:38:39.097000000

          CreditGrade  Term LoanStatus           ClosedDate  BorrowerAPR  \
```

```
0          C    36  Completed  2009-08-14 00:00:00      0.16516
1        NaN    36    Current                  NaN      0.12016
2         HR    36  Completed  2009-12-17 00:00:00      0.28269
3        NaN    36    Current                  NaN      0.12528
4        NaN    36    Current                  NaN      0.24614

   BorrowerRate  LenderYield    ...     LP_ServiceFees  LP_CollectionFees  \
0        0.1580       0.1380    ...            -133.18                0.0
1        0.0920       0.0820    ...               0.00                0.0
2        0.2750       0.2400    ...             -24.20                0.0
3        0.0974       0.0874    ...            -108.01                0.0
4        0.2085       0.1985    ...             -60.27                0.0

   LP_GrossPrincipalLoss  LP_NetPrincipalLoss LP_NonPrincipalRecoverypayments  \
0                    0.0                  0.0                             0.0
1                    0.0                  0.0                             0.0
2                    0.0                  0.0                             0.0
3                    0.0                  0.0                             0.0
4                    0.0                  0.0                             0.0

   PercentFunded  Recommendations InvestmentFromFriendsCount  \
0            1.0                0                          0
1            1.0                0                          0
2            1.0                0                          0
3            1.0                0                          0
4            1.0                0                          0

  InvestmentFromFriendsAmount Investors
0                         0.0       258
1                         0.0         1
2                         0.0        41
3                         0.0       158
4                         0.0        20

[5 rows x 81 columns]
```

In [3]:
```
data.describe()
```

Out[3]:
```
       ListingNumber           Term    BorrowerAPR   BorrowerRate  \
count   1.139370e+05  113937.000000  113912.000000  113937.000000
mean    6.278857e+05      40.830248       0.218828       0.192764
std     3.280762e+05      10.436212       0.080364       0.074818
min     4.000000e+00      12.000000       0.006530       0.000000
25%     4.009190e+05      36.000000       0.156290       0.134000
50%     6.005540e+05      36.000000       0.209760       0.184000
75%     8.926340e+05      36.000000       0.283810       0.250000
max     1.255725e+06      60.000000       0.512290       0.497500
```

	LenderYield	EstimatedEffectiveYield	EstimatedLoss	EstimatedReturn	\
count	113937.000000	84853.000000	84853.000000	84853.000000	
mean	0.182701	0.168661	0.080306	0.096068	
std	0.074516	0.068467	0.046764	0.030403	
min	-0.010000	-0.182700	0.004900	-0.182700	
25%	0.124200	0.115670	0.042400	0.074080	
50%	0.173000	0.161500	0.072400	0.091700	
75%	0.240000	0.224300	0.112000	0.116600	
max	0.492500	0.319900	0.366000	0.283700	

	ProsperRating (numeric)	ProsperScore	...	LP_ServiceFees	\
count	84853.000000	84853.000000	...	113937.000000	
mean	4.072243	5.950067	...	-54.725641	
std	1.673227	2.376501	...	60.675425	
min	1.000000	1.000000	...	-664.870000	
25%	3.000000	4.000000	...	-73.180000	
50%	4.000000	6.000000	...	-34.440000	
75%	5.000000	8.000000	...	-13.920000	
max	7.000000	11.000000	...	32.060000	

	LP_CollectionFees	LP_GrossPrincipalLoss	LP_NetPrincipalLoss	\
count	113937.000000	113937.000000	113937.000000	
mean	-14.242698	700.446342	681.420499	
std	109.232758	2388.513831	2357.167068	
min	-9274.750000	-94.200000	-954.550000	
25%	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	
max	0.000000	25000.000000	25000.000000	

	LP_NonPrincipalRecoverypayments	PercentFunded	Recommendations	\
count	113937.000000	113937.000000	113937.000000	
mean	25.142686	0.998584	0.048027	
std	275.657937	0.017919	0.332353	
min	0.000000	0.700000	0.000000	
25%	0.000000	1.000000	0.000000	
50%	0.000000	1.000000	0.000000	
75%	0.000000	1.000000	0.000000	
max	21117.900000	1.012500	39.000000	

	InvestmentFromFriendsCount	InvestmentFromFriendsAmount	Investors
count	113937.000000	113937.000000	113937.000000
mean	0.023460	16.550751	80.475228
std	0.232412	294.545422	103.239020
min	0.000000	0.000000	1.000000
25%	0.000000	0.000000	2.000000
50%	0.000000	0.000000	44.000000
75%	0.000000	0.000000	115.000000

```
max                   33.000000                25000.000000    1189.000000

[8 rows x 61 columns]
```

61 out of the original 81 columns of the original dataset are numeric columns.

```
In [4]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 81 columns):
ListingKey                             113937 non-null object
ListingNumber                          113937 non-null int64
ListingCreationDate                    113937 non-null object
CreditGrade                            28953 non-null object
Term                                   113937 non-null int64
LoanStatus                             113937 non-null object
ClosedDate                             55089 non-null object
BorrowerAPR                            113912 non-null float64
BorrowerRate                           113937 non-null float64
LenderYield                            113937 non-null float64
EstimatedEffectiveYield                84853 non-null float64
EstimatedLoss                          84853 non-null float64
EstimatedReturn                        84853 non-null float64
ProsperRating (numeric)                84853 non-null float64
ProsperRating (Alpha)                  84853 non-null object
ProsperScore                           84853 non-null float64
ListingCategory (numeric)              113937 non-null int64
BorrowerState                          108422 non-null object
Occupation                             110349 non-null object
EmploymentStatus                       111682 non-null object
EmploymentStatusDuration               106312 non-null float64
IsBorrowerHomeowner                    113937 non-null bool
CurrentlyInGroup                       113937 non-null bool
GroupKey                               13341 non-null object
DateCreditPulled                       113937 non-null object
CreditScoreRangeLower                  113346 non-null float64
CreditScoreRangeUpper                  113346 non-null float64
FirstRecordedCreditLine                113240 non-null object
CurrentCreditLines                     106333 non-null float64
OpenCreditLines                        106333 non-null float64
TotalCreditLinespast7years             113240 non-null float64
OpenRevolvingAccounts                  113937 non-null int64
OpenRevolvingMonthlyPayment            113937 non-null float64
InquiriesLast6Months                   113240 non-null float64
TotalInquiries                         112778 non-null float64
CurrentDelinquencies                   113240 non-null float64
AmountDelinquent                       106315 non-null float64
```

```
DelinquenciesLast7Years                 112947 non-null float64
PublicRecordsLast10Years                113240 non-null float64
PublicRecordsLast12Months               106333 non-null float64
RevolvingCreditBalance                  106333 non-null float64
BankcardUtilization                     106333 non-null float64
AvailableBankcardCredit                 106393 non-null float64
TotalTrades                             106393 non-null float64
TradesNeverDelinquent (percentage)      106393 non-null float64
TradesOpenedLast6Months                 106393 non-null float64
DebtToIncomeRatio                       105383 non-null float64
IncomeRange                             113937 non-null object
IncomeVerifiable                        113937 non-null bool
StatedMonthlyIncome                     113937 non-null float64
LoanKey                                 113937 non-null object
TotalProsperLoans                       22085 non-null float64
TotalProsperPaymentsBilled              22085 non-null float64
OnTimeProsperPayments                   22085 non-null float64
ProsperPaymentsLessThanOneMonthLate     22085 non-null float64
ProsperPaymentsOneMonthPlusLate         22085 non-null float64
ProsperPrincipalBorrowed                22085 non-null float64
ProsperPrincipalOutstanding             22085 non-null float64
ScorexChangeAtTimeOfListing             18928 non-null float64
LoanCurrentDaysDelinquent               113937 non-null int64
LoanFirstDefaultedCycleNumber           16952 non-null float64
LoanMonthsSinceOrigination              113937 non-null int64
LoanNumber                              113937 non-null int64
LoanOriginalAmount                      113937 non-null int64
LoanOriginationDate                     113937 non-null object
LoanOriginationQuarter                  113937 non-null object
MemberKey                               113937 non-null object
MonthlyLoanPayment                      113937 non-null float64
LP_CustomerPayments                     113937 non-null float64
LP_CustomerPrincipalPayments            113937 non-null float64
LP_InterestandFees                      113937 non-null float64
LP_ServiceFees                          113937 non-null float64
LP_CollectionFees                       113937 non-null float64
LP_GrossPrincipalLoss                   113937 non-null float64
LP_NetPrincipalLoss                     113937 non-null float64
LP_NonPrincipalRecoverypayments         113937 non-null float64
PercentFunded                           113937 non-null float64
Recommendations                         113937 non-null int64
InvestmentFromFriendsCount              113937 non-null int64
InvestmentFromFriendsAmount             113937 non-null float64
Investors                               113937 non-null int64
dtypes: bool(3), float64(50), int64(11), object(17)
memory usage: 68.1+ MB
```

In [5]: data.isnull().sum()

```
Out[5]: ListingKey                                   0
        ListingNumber                                0
        ListingCreationDate                          0
        CreditGrade                              84984
        Term                                         0
        LoanStatus                                   0
        ClosedDate                               58848
        BorrowerAPR                                 25
        BorrowerRate                                 0
        LenderYield                                  0
        EstimatedEffectiveYield                  29084
        EstimatedLoss                            29084
        EstimatedReturn                          29084
        ProsperRating (numeric)                  29084
        ProsperRating (Alpha)                    29084
        ProsperScore                             29084
        ListingCategory (numeric)                    0
        BorrowerState                             5515
        Occupation                                3588
        EmploymentStatus                          2255
        EmploymentStatusDuration                  7625
        IsBorrowerHomeowner                          0
        CurrentlyInGroup                             0
        GroupKey                                100596
        DateCreditPulled                             0
        CreditScoreRangeLower                      591
        CreditScoreRangeUpper                      591
        FirstRecordedCreditLine                    697
        CurrentCreditLines                        7604
        OpenCreditLines                           7604
                                                 ...
        TotalProsperLoans                        91852
        TotalProsperPaymentsBilled               91852
        OnTimeProsperPayments                    91852
        ProsperPaymentsLessThanOneMonthLate      91852
        ProsperPaymentsOneMonthPlusLate          91852
        ProsperPrincipalBorrowed                 91852
        ProsperPrincipalOutstanding              91852
        ScorexChangeAtTimeOfListing              95009
        LoanCurrentDaysDelinquent                    0
        LoanFirstDefaultedCycleNumber            96985
        LoanMonthsSinceOrigination                   0
        LoanNumber                                   0
        LoanOriginalAmount                           0
        LoanOriginationDate                          0
        LoanOriginationQuarter                       0
```

```
MemberKey                                  0
MonthlyLoanPayment                         0
LP_CustomerPayments                        0
LP_CustomerPrincipalPayments               0
LP_InterestandFees                         0
LP_ServiceFees                             0
LP_CollectionFees                          0
LP_GrossPrincipalLoss                      0
LP_NetPrincipalLoss                        0
LP_NonPrincipalRecoverypayments            0
PercentFunded                              0
Recommendations                            0
InvestmentFromFriendsCount                 0
InvestmentFromFriendsAmount                0
Investors                                  0
Length: 81, dtype: int64
```

We can see that some columns have a considerably large amount of missing data. Hence these columns would not be useful in our analysis as it would not be a good represntative of the dataset.

Keeping this in mind, we select those columns that might be of vital essence to our analysis.

In [6]:
```
target_columns =  [
    'Term', 'LoanStatus', 'BorrowerRate', 'ListingCategory (numeric)', 'EmploymentStatus
    'DelinquenciesLast7Years', 'StatedMonthlyIncome', 'TotalProsperLoans', 'LoanOriginal
    'LoanOriginationDate', 'Recommendations', 'Investors', 'ProsperRating (Alpha)'
]
```

In [7]:
```
selected_df = data[target_columns]
selected_df.head()
```

Out[7]:
```
   Term LoanStatus  BorrowerRate  ListingCategory (numeric) EmploymentStatus  \
0    36  Completed        0.1580                          0    Self-employed
1    36    Current        0.0920                          2         Employed
2    36  Completed        0.2750                          0    Not available
3    36    Current        0.0974                         16         Employed
4    36    Current        0.2085                          2         Employed

   DelinquenciesLast7Years  StatedMonthlyIncome  TotalProsperLoans  \
0                      4.0          3083.333333                NaN
1                      0.0          6125.000000                NaN
2                      0.0          2083.333333                NaN
3                     14.0          2875.000000                NaN
4                      0.0          9583.333333                1.0

   LoanOriginalAmount  LoanOriginationDate  Recommendations  Investors  \
0                9425  2007-09-12 00:00:00                0        258
1               10000  2014-03-03 00:00:00                0          1
2                3001  2007-01-17 00:00:00                0         41
3               10000  2012-11-01 00:00:00                0        158
```

```
4              15000  2013-09-20 00:00:00                0         20

  ProsperRating (Alpha)
0                   NaN
1                     A
2                   NaN
3                     A
4                     D
```

```
In [8]: selected_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 13 columns):
Term                         113937 non-null int64
LoanStatus                   113937 non-null object
BorrowerRate                 113937 non-null float64
ListingCategory (numeric)    113937 non-null int64
EmploymentStatus             111682 non-null object
DelinquenciesLast7Years      112947 non-null float64
StatedMonthlyIncome          113937 non-null float64
TotalProsperLoans            22085 non-null float64
LoanOriginalAmount           113937 non-null int64
LoanOriginationDate          113937 non-null object
Recommendations              113937 non-null int64
Investors                    113937 non-null int64
ProsperRating (Alpha)        84853 non-null object
dtypes: float64(4), int64(5), object(4)
memory usage: 11.3+ MB
```

```
In [9]: #converting the column with date to the datetime type
        selected_df['LoanOriginationDate'] = pd.to_datetime(selected_df['LoanOriginationDate'])
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
```

```
In [10]: selected_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 13 columns):
Term                         113937 non-null int64
LoanStatus                   113937 non-null object
```

```
BorrowerRate                 113937 non-null float64
ListingCategory (numeric)    113937 non-null int64
EmploymentStatus             111682 non-null object
DelinquenciesLast7Years      112947 non-null float64
StatedMonthlyIncome          113937 non-null float64
TotalProsperLoans            22085 non-null float64
LoanOriginalAmount           113937 non-null int64
LoanOriginationDate          113937 non-null datetime64[ns]
Recommendations              113937 non-null int64
Investors                    113937 non-null int64
ProsperRating (Alpha)        84853 non-null object
dtypes: datetime64[ns](1), float64(4), int64(5), object(3)
memory usage: 11.3+ MB
```

1.3.1 What is the structure of your dataset?

The original dataset has 81 columns and 113937 rows.

1.3.2 What is/are the main feature(s) of interest in your dataset?

Some columns like employmet status of the borrower or if they are home owners would have benefitted the analysis but upon investigation, thse columns have too many missing figures. Loan amount, Borrower rate, Loan Status are features of interest.

1.3.3 What features in the dataset do you think will help support your investigation into your feature(s) of interest?

Recomendations, Employment Status, Stated monthly income.

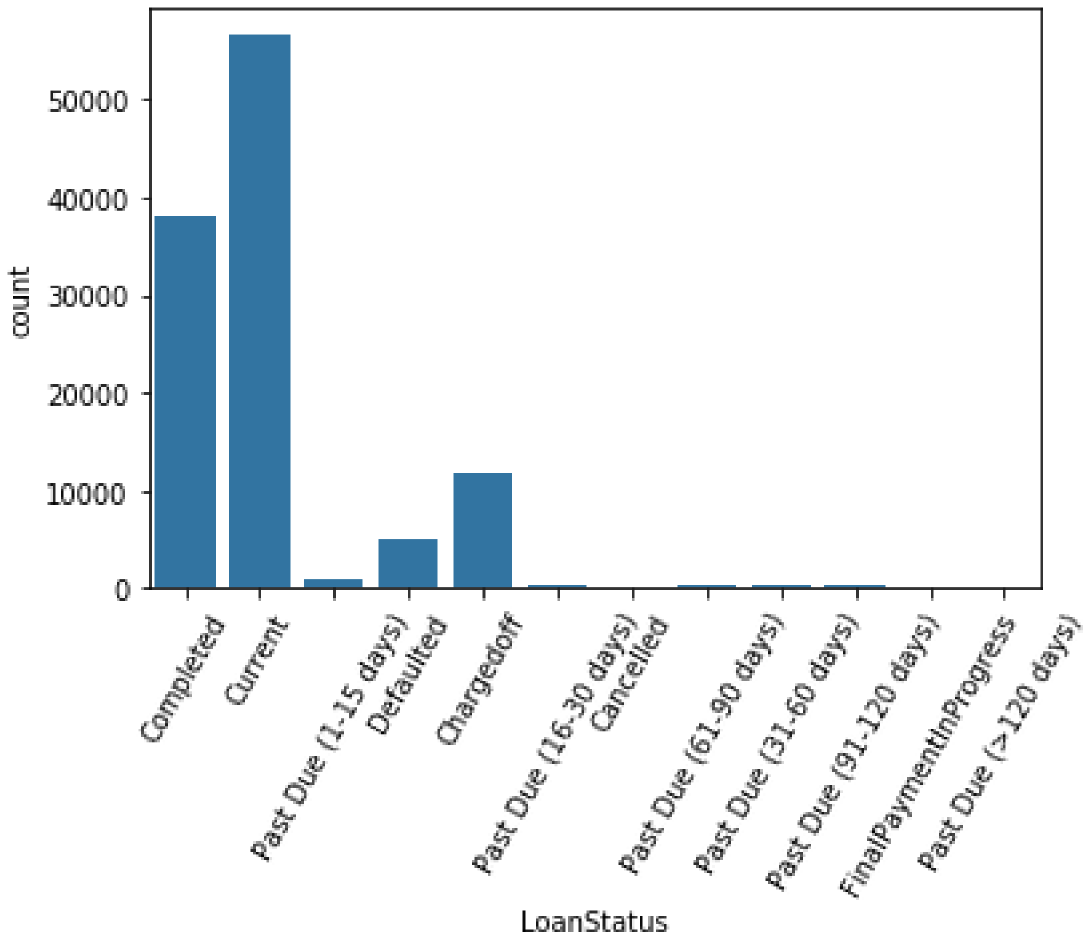
1.4 Univariate Exploration

In this section, investigate distributions of individual variables. If you see unusual points or outliers, take a deeper look to clean things up and prepare yourself to look at relationships between variables.

1.4.1 Visualization 1

Count of Loan Status What is the status on the Loan given out?

```
In [11]: # setting color
         base_color = sns.color_palette()[0]
         plt.xticks(rotation=60)
         sns.countplot(data = selected_df, x = 'LoanStatus', color = base_color);
```

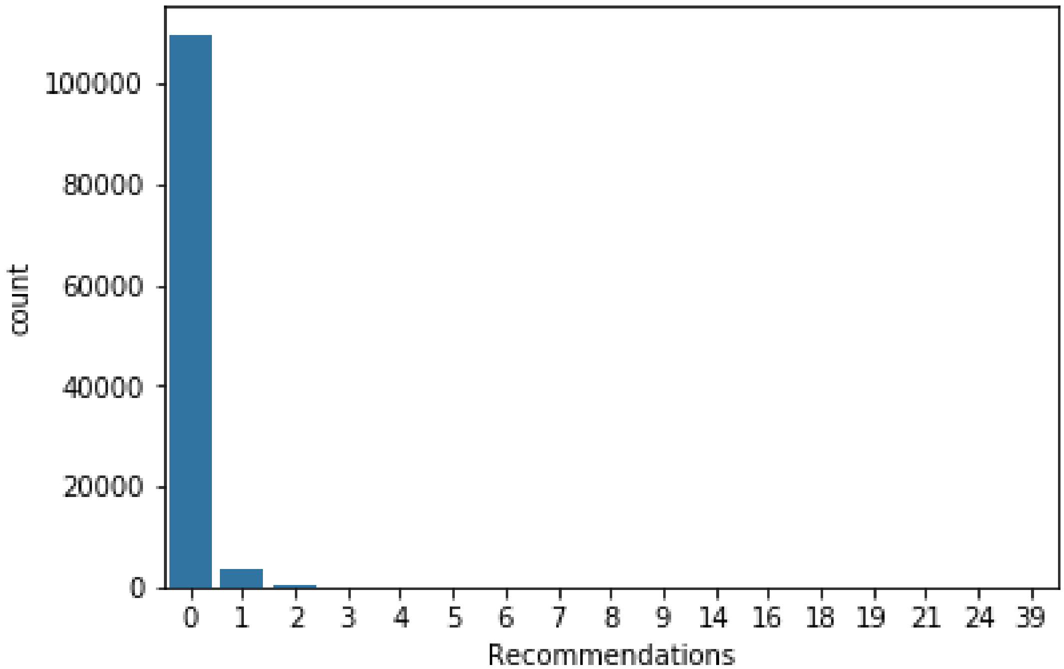


We see that overall the number of Loan that are still active borowers at the time of the data being collected is the highest with quite a high amount also having completed the duration.

1.4.2 Visualization 2

Recommendation What is the signifance of recommendation on the Loans?

```
In [12]: sns.countplot(data = selected_df, x = 'Recommendations', color = base_color);
```

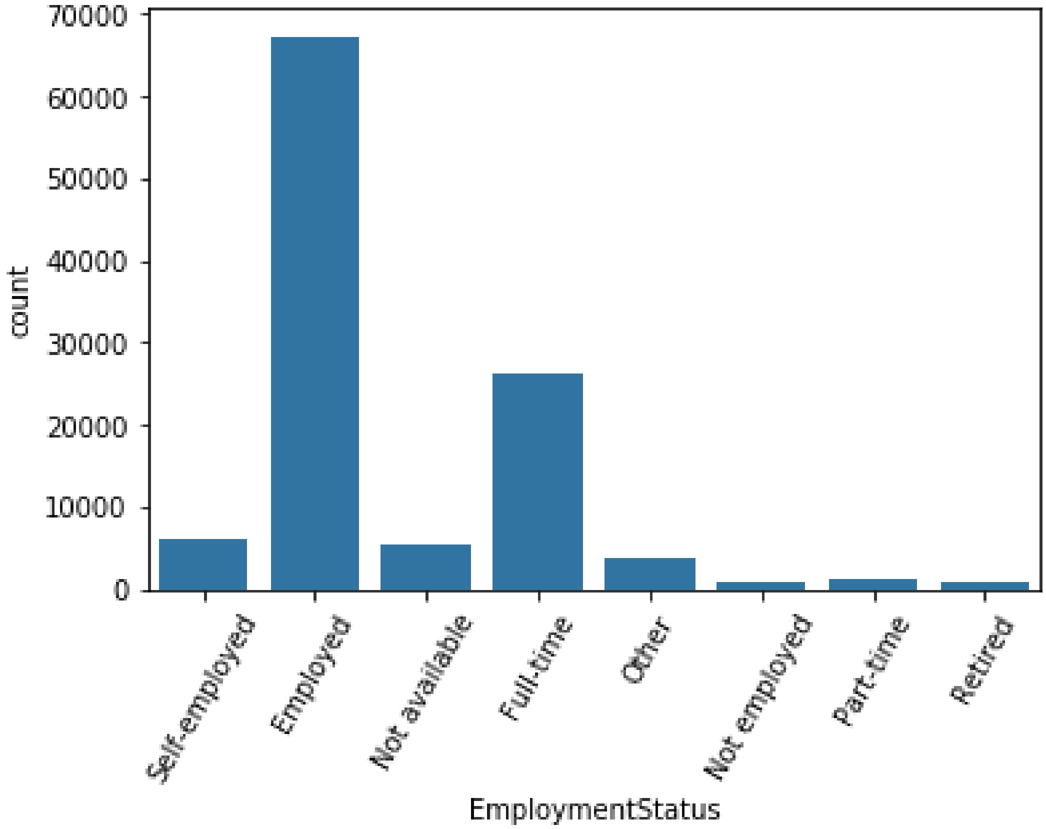


Since majority of the loans allocated have a reommendation of 0 (no recommendation), we can then say that the likelihood of being given a loan has nothing to do with whether ypu present a recommender or not.

1.4.3 Visualization 3

Employment Status What is the most common employment status of the people with the loans?

```
In [13]: sns.countplot(data = selected_df, x = 'EmploymentStatus', color = base_color);
         plt.xticks(rotation = 60);
```

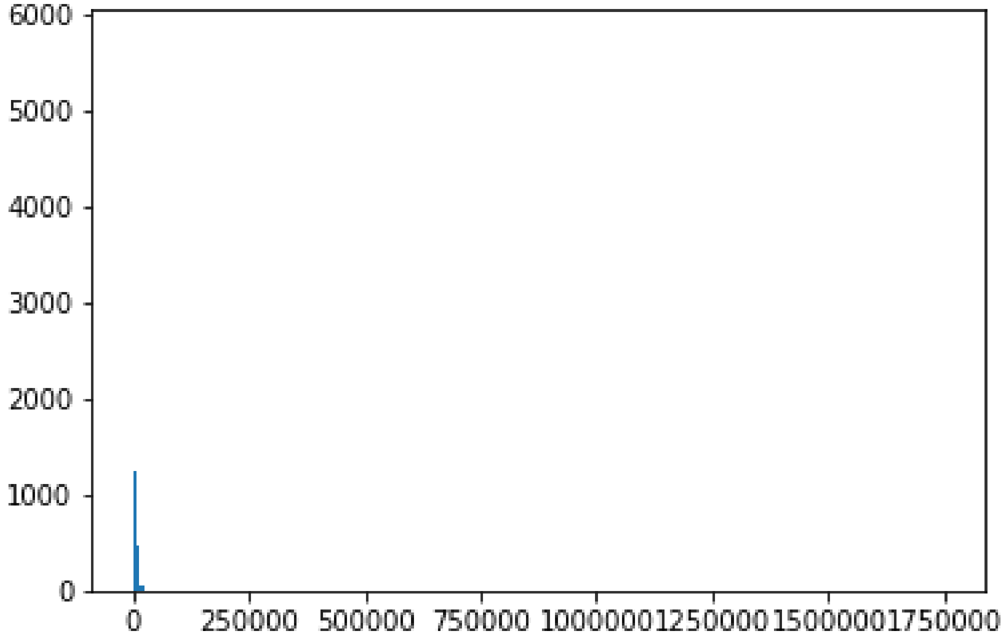


We see that Employed people make for a large percent of the people at the time of listing and people who work full time. People who are retired are. very minute amount of the people constituted as borrowers.

1.4.4 Visualization 4

Stated Monthly Income Our is the monthly income of the borrowers like?

```
In [14]: plt.hist(data=selected_df, x='StatedMonthlyIncome', bins=10000);
```

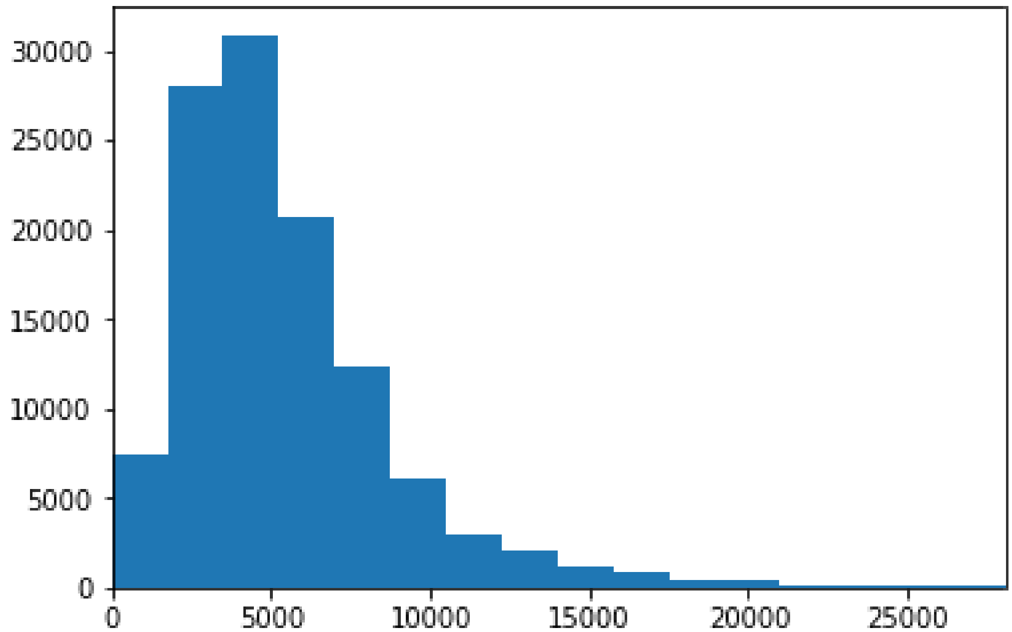


This is highly skewed

In [15]:
```
income_std = selected_df['StatedMonthlyIncome'].std()
income_mean = selected_df['StatedMonthlyIncome'].mean()
boundary = income_mean + income_std * 3
len(selected_df[selected_df['StatedMonthlyIncome'] >= boundary])
```

Out[15]: 428

In [16]:
```
## Zooming in

plt.hist(data=selected_df, x='StatedMonthlyIncome', bins=1000);
plt.xlim(0, boundary);
```

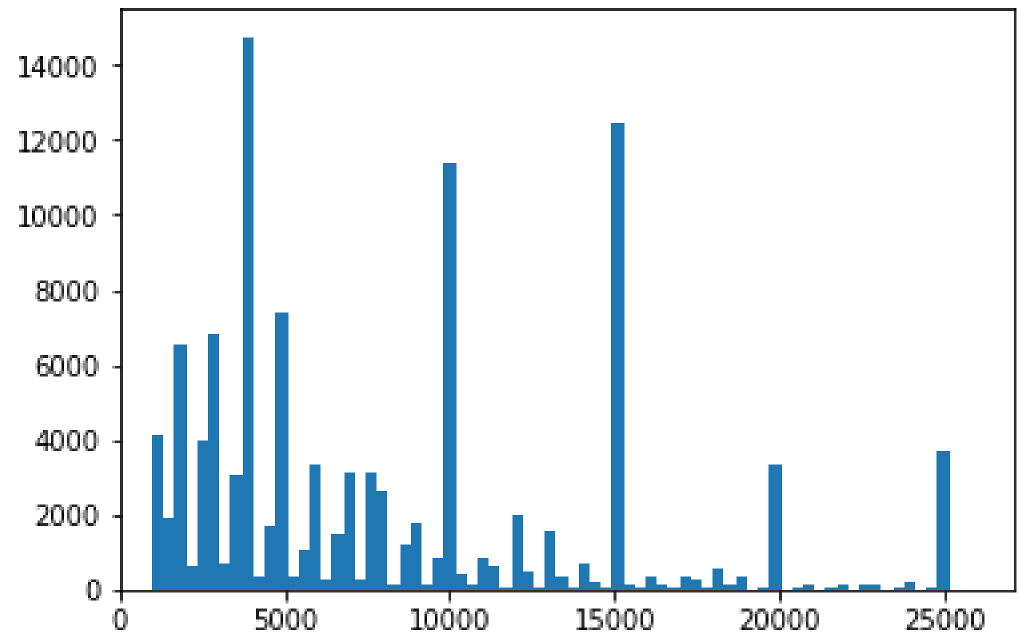


After zooming, we see that the most common monthly income bracket of people is 5000

1.4.5 Visualization 5

Original Loan Amount What are the most common amount given as loans?

```
In [17]: loan_std = selected_df['LoanOriginalAmount'].std()
         loan_mean = selected_df['LoanOriginalAmount'].mean()
         boundary = loan_mean + loan_std * 3

         plt.hist(data=selected_df, x='LoanOriginalAmount', bins=100);
         plt.xlim(0, boundary);
```



We see that the distribution is skewed to the right. The most common amount given as loan is about 4000 and closer to that is 15000 and then 10000

1.4.6 Discuss the distribution(s) of your variable(s) of interest. Were there any unusual points? Did you need to perform any transformations?

The distribution of the monthly icome was highly skewed and nothing of great imoortance could be told from the original plot. The plot was then zoomed into the plot using the boundary gotten form the mean and standard deviation. Most of the borrowers are employed and all other categories as small part of borrowers and most of the loans in the data set are actually current loans.

1.4.7 Of the features you investigated, were there any unusual distributions? Did you perform any operations on the data to tidy, adjust, or change the form of the data? If so, why did you do this?

Recommendation gave almost no other additional information

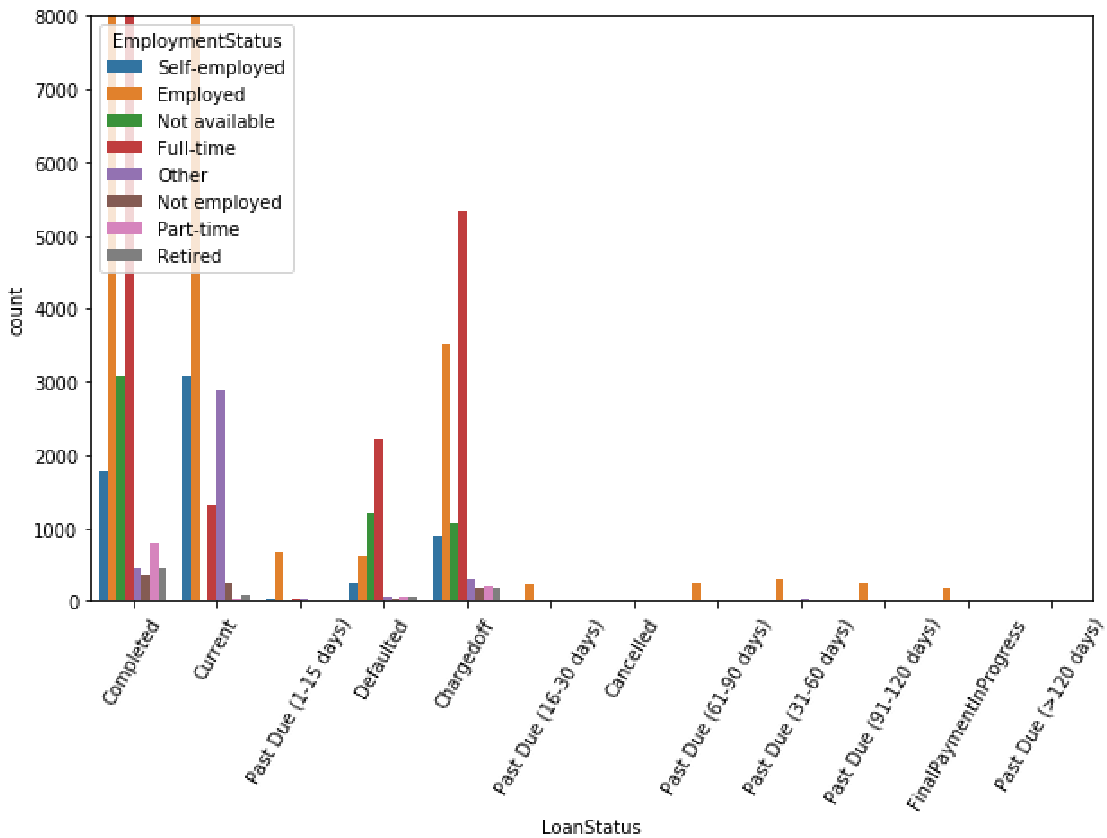
1.5 Bivariate Exploration

In this section, investigate relationships between pairs of variables in your data. Make sure the variables that you cover here have been introduced in some fashion in the previous section (univariate exploration).

1.5.1 Visualization 6

Loan Status and Employment Status

```
In [18]: plt.figure(figsize=[10, 6])
         sns.countplot(data = selected_df, x = 'LoanStatus', hue = 'EmploymentStatus');
         plt.xticks(rotation = 60);
         plt.ylim(0, 8000);
```



For the people who have completed their loans, we see that those employed make up the majority of them. Likweise, majority of the people currently with loans are also employed.

1.5.2 Visualization 7

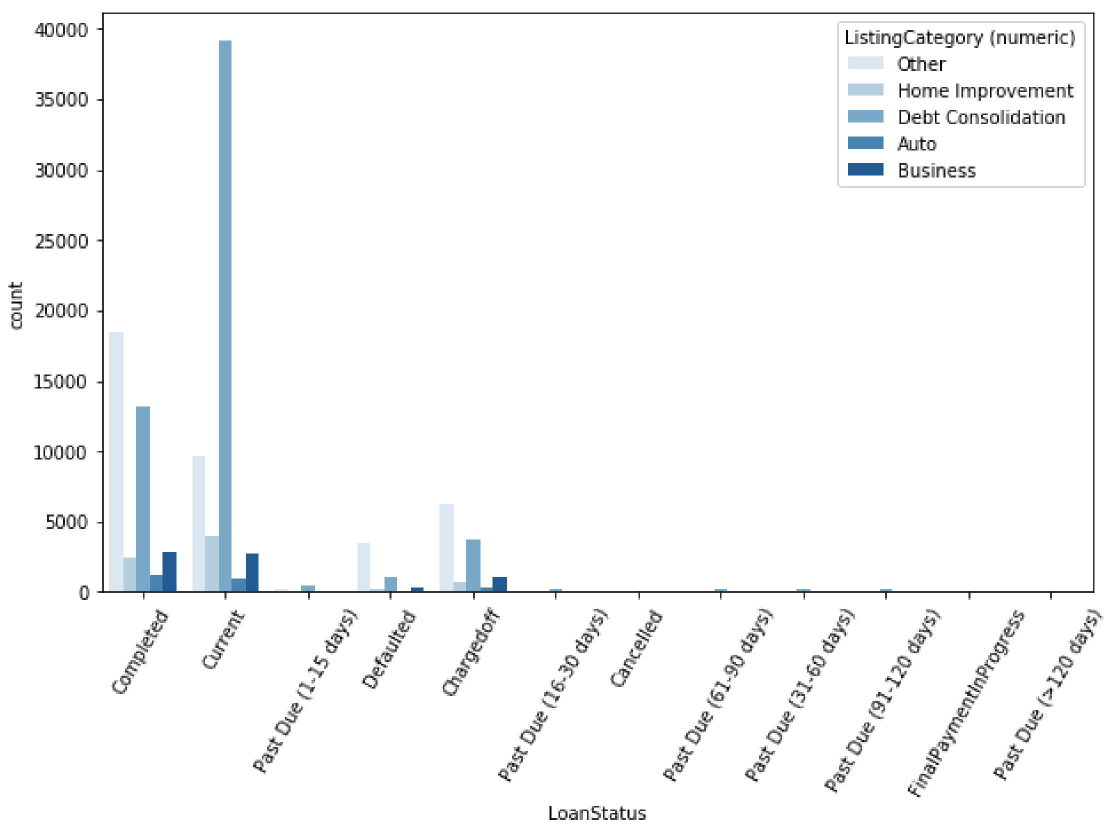
Loan Status and Listing Categories

```
In [19]: categories = {1: 'Debt Consolidation', 2: 'Home Improvement', 3: 'Business', 6: 'Auto',
         def reduce_categorie(row):
             loan_category = row['ListingCategory (numeric)']
             if  loan_category in categories:
                 return categories[loan_category]
             else:
                 return categories[7]

         selected_df['ListingCategory (numeric)'] = selected_df.apply(reduce_categorie, axis=1)
         selected_df['ListingCategory (numeric)'].value_counts()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:9: SettingWithCopyWarning:
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  if __name__ == '__main__':
```

```
Out[19]: Debt Consolidation    58308
         Other                 38435
         Home Improvement       7433
         Business               7189
         Auto                   2572
         Name: ListingCategory (numeric), dtype: int64
```

```
In [20]: plt.figure(figsize=[10, 6])
         sns.countplot(data = selected_df, x = 'LoanStatus', hue = 'ListingCategory (numeric)',
         plt.xticks(rotation = 60);
```



We can say that the majority of borrowers currently borrowed the money for debt consolidation. And majority of those that have completed their loans borrowed for something other than that.

1.5.3 Visualization 8

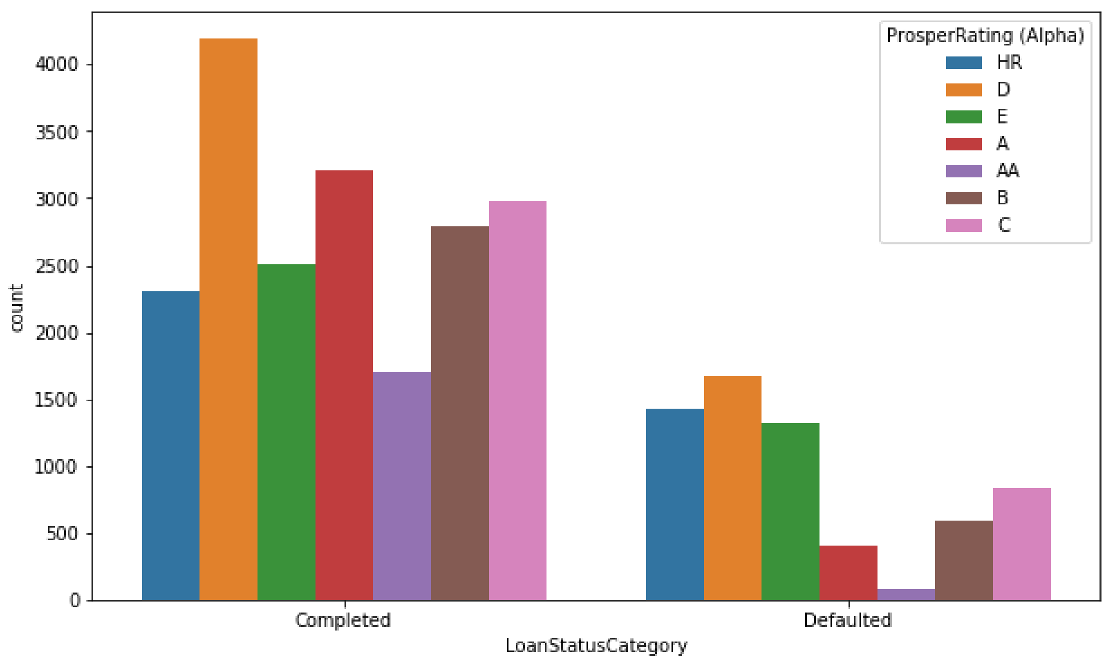
Loan Status vs Rating listed

In [21]:
```
condition = (selected_df['LoanStatus'] == 'Completed') | (selected_df['LoanStatus'] ==
                            (selected_df['LoanStatus'] == 'Chargedoff')
selected_df = selected_df[condition]

def change_to_defaulted(row):
    if row['LoanStatus'] == 'Chargedoff':
        return 'Defaulted'
    else:
        return row['LoanStatus']

selected_df['LoanStatusCategory'] = selected_df.apply(change_to_defaulted, axis=1)
selected_df['LoanStatusCategory'].value_counts()
```

Out[21]:
```
Completed    38074
Defaulted    17010
Name: LoanStatusCategory, dtype: int64
```

In [22]:
```
plt.figure(figsize=[10, 6])
sns.countplot(data = selected_df, x = 'LoanStatusCategory', hue = 'ProsperRating (Alpha
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0dd60c7b8>
```

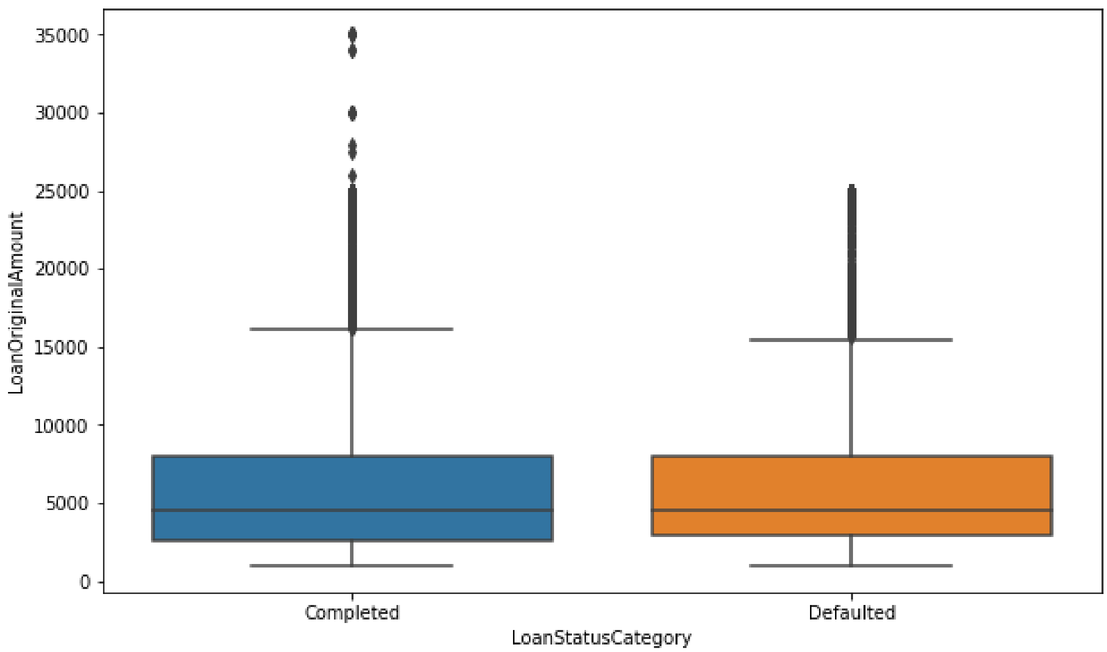


Majority of completed loans are in category D followed by those in category A Majority of defaulted loans are likewise in D followed by those in category HR.

1.5.4 Visualization 9

Loan Status Category vs Loan Amount

```
In [23]: plt.figure(figsize=[10, 6])
         sns.boxplot(data = selected_df, x = 'LoanStatusCategory', y = 'LoanOriginalAmount');
```



1.5.5 Visualization 10

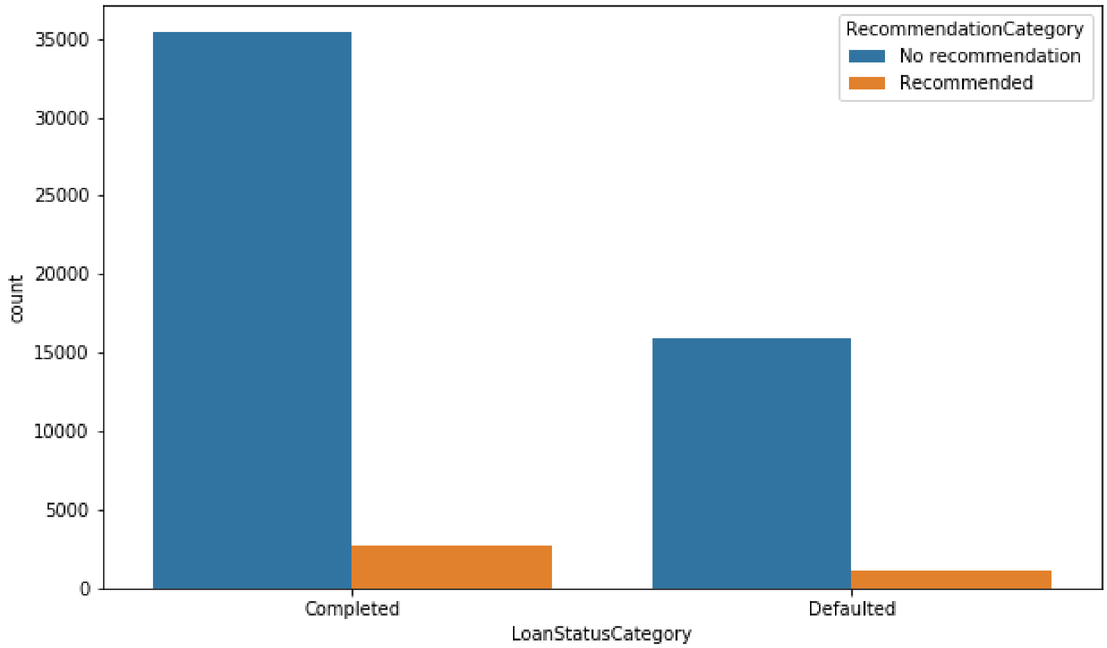
Loan Status vs recommendation

```
In [24]: def change_recommendation(row):
             if row['Recommendations'] == 0:
                 return 'No recommendation'
             else:
                 return 'Recommended'

         selected_df['RecommendationCategory'] = selected_df.apply(change_recommendation, axis=1
         selected_df['RecommendationCategory'].value_counts()
```

```
Out[24]: No recommendation    51281
         Recommended           3803
         Name: RecommendationCategory, dtype: int64
```

```
In [25]: plt.figure(figsize=[10, 6])
         sns.countplot(data = selected_df, x = 'LoanStatusCategory', hue = 'RecommendationCatego
```

```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe0dd7fe710>
```



The majority of all loan categories are without recommendation

1.5.6 Talk about some of the relationships you observed in this part of the investigation. How did the feature(s) of interest vary with other features in the dataset?

Loan Status and recoomendation and not so connected Majority of completed loans are by employed people and those with some full time kind of work.

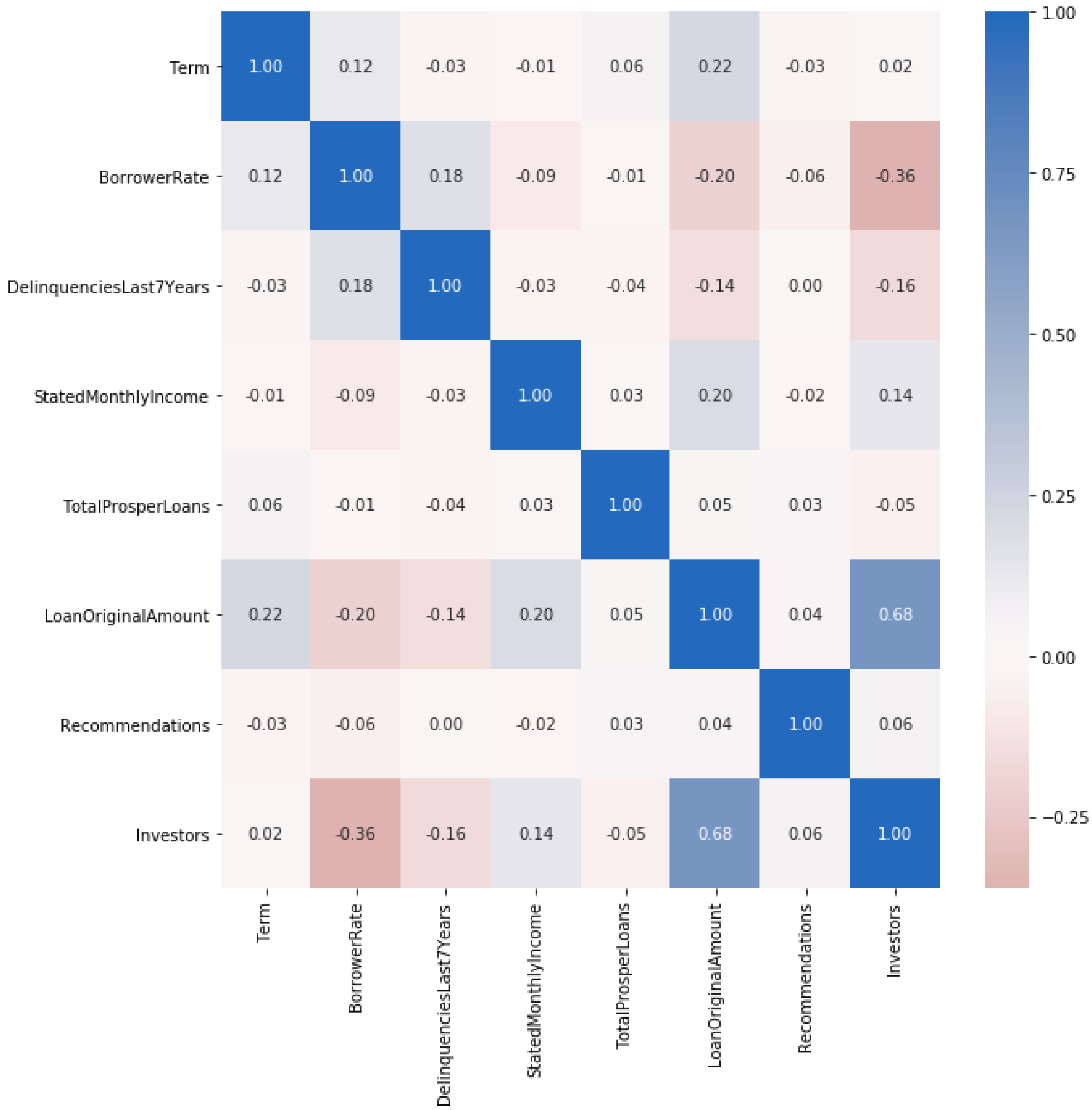
1.6 Multivariate Exploration

Create plots of three or more variables to investigate your data even further. Make sure that your investigations are justified, and follow from your work in the previous sections.

1.6.1 Visulaization11

1.6.2 Correlation betweeen the data set

```
In [26]: plt.figure(figsize=[10, 10])
         sns.heatmap(selected_df.corr(), annot = True, fmt = '.2f', cmap = 'vlag_r', center = 0)

         plt.show();
```

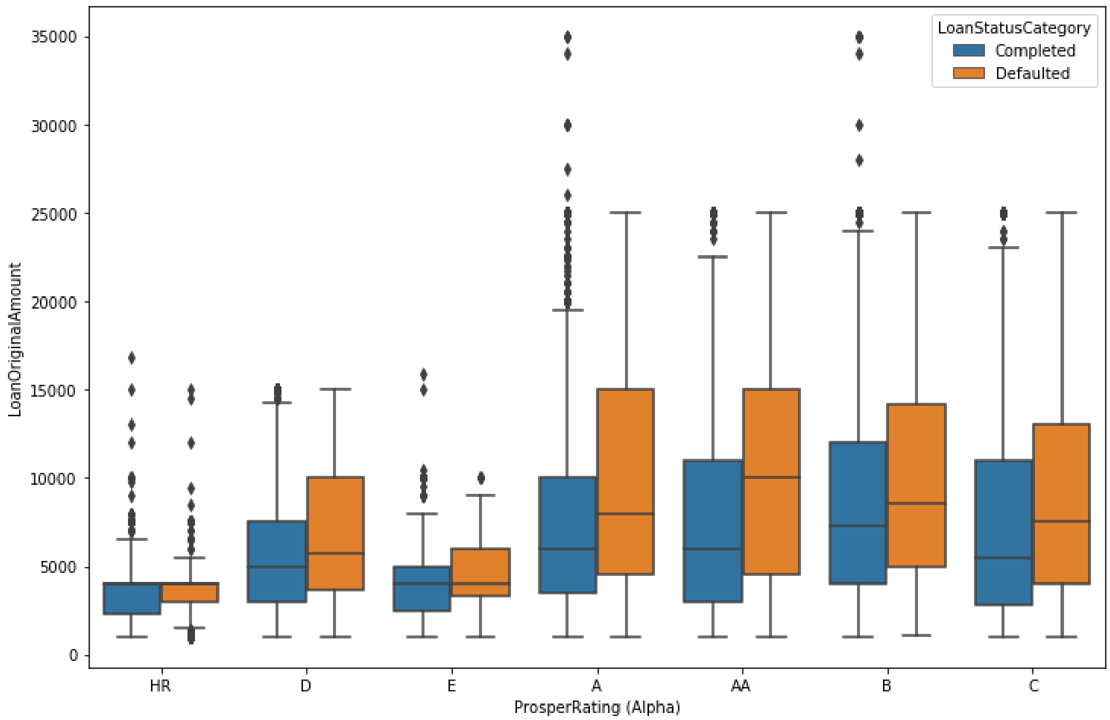


Looking at the correlation matrix plot for the entore datset, there isn't so much to tell. However, we can see there is a stronger relationship between original loan amount and no of Investors.

1.6.3 Visualization 12

Loan Status vs Loan Amount vs Rating

```
In [27]: plt.figure(figsize = [12, 8])
         sns.boxplot(data=selected_df, x='ProsperRating (Alpha)', y='LoanOriginalAmount', hue='L
```

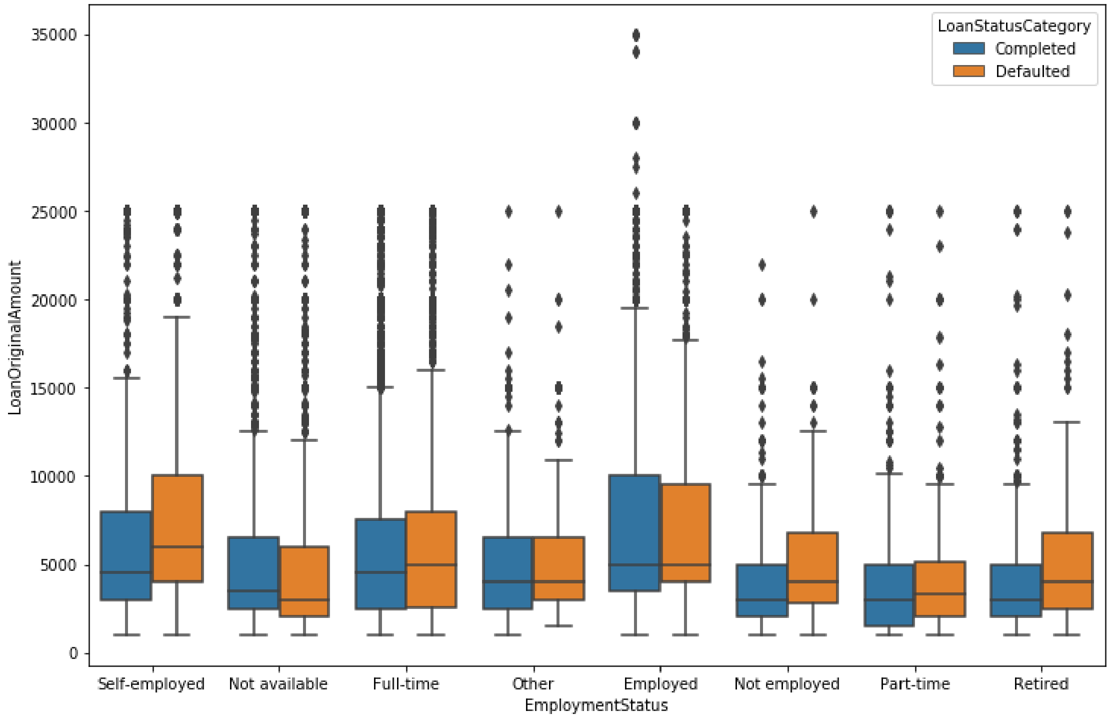


The defaulters make up amjority of thr category A, B and AA

1.6.4 Visualization 13

Loan Status vs Loan Amount vs Employment Status

```
In [28]: plt.figure(figsize = [12, 8])
         sns.boxplot(data=selected_df, x='EmploymentStatus', y='LoanOriginalAmount', hue='LoanSt
```



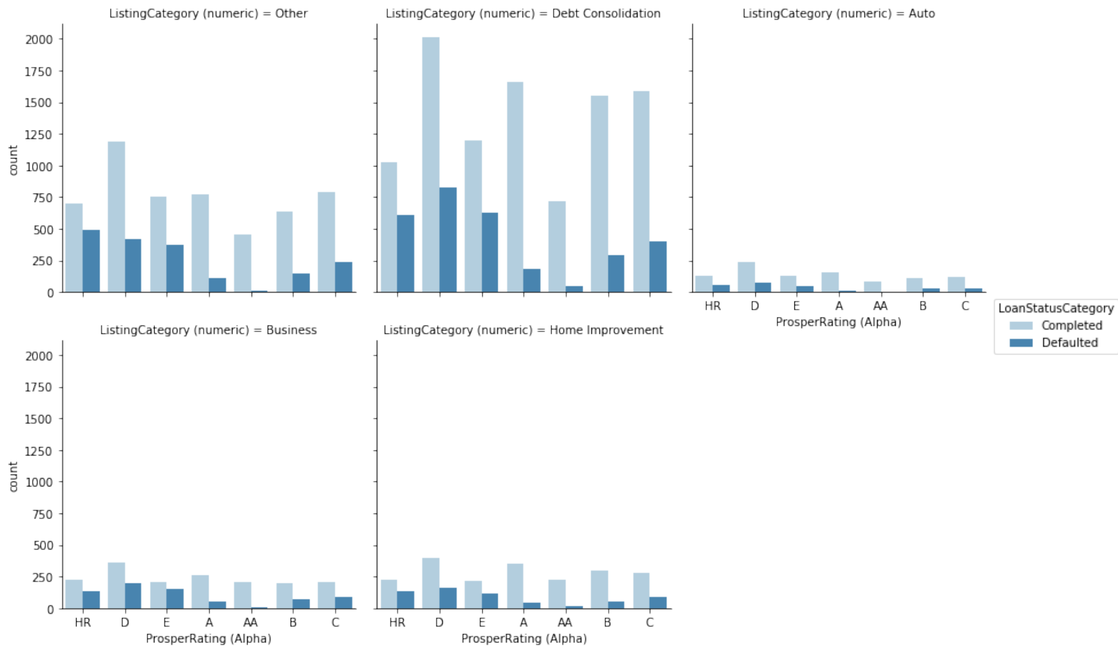
For Self-employed people, majority of them are defaulters of loan while employed poeple majorly complete their loans.

1.6.5 Visualization 14

Loan Status, Listing Category and Alpha Rating.

```
In [29]: plt.figure(figsize = [10, 6])
         sns.factorplot(x = 'ProsperRating (Alpha)', hue = 'LoanStatusCategory', col = 'ListingC
                        data = selected_df, kind = 'count', palette = 'Blues', col_wrap = 3);
         plt.show()
```

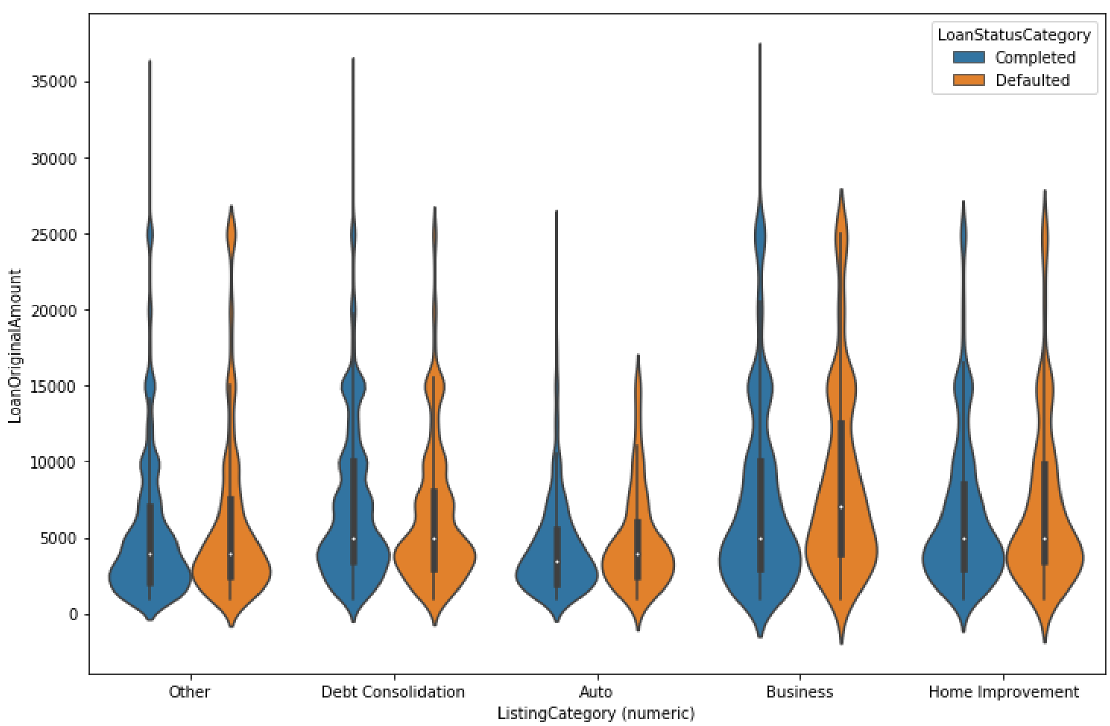
```
<matplotlib.figure.Figure at 0x7fe0dd86c630>
```



1.6.6 Visualization 15

Loan Status, Original Amount and Listing Category.

```
In [30]: plt.figure(figsize = [12, 8])
         sns.violinplot(data=selected_df, x='ListingCategory (numeric)', y='LoanOriginalAmount',
```



1.6.7 Talk about some of the relationships you observed in this part of the investigation. Were there features that strengthened each other in terms of looking at your feature(s) of interest?

Mosst of the loans were taken for debt consolidation. Employed people payback mostly Defaulters are mostly self employed.

1.6.8 Were there any interesting or surprising interactions between features?

Defaulters do no t make up the higher amount of loans given.

1.7 Conclusions

During the data exploration, we found many data columns that were unusable due to their large amount of missing values. However, we were able to pick about 13 columns that were vitals to this analysis. Univariate visulaization were done for single features and then they were compared with other important features in bivariate visualization. The relationship on a wider range like correlation was viwed in the multivariate visualization.