Part_I_exploration_template

August 22, 2022

0.1 PROSPERLOAN DATASET EXPLORATION

0.2 by Josephine Seyram Agyeman

0.3 Introduction

This document explores a dataset containing various variables like term, recommendations, investors, original loan amount, occcupation, among others on approximately 110,000 loan borrowers.

0.4 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 sb

        %matplotlib inline
```

```
In [2]: # Loading dataset into pandas dataframe
        loan = pd.read_csv('prosperLoanData.csv')
```

```
In [3]: #Getting familiar with the dataset
        print(loan.shape)
```

```
(113937, 81)
```

```
In [4]: loan.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]: loan.head()
```

```
Out[5]:                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 [6]: print(loan.dtypes)
```

```
ListingKey                              object
ListingNumber                            int64
ListingCreationDate                     object
CreditGrade                             object
Term                                     int64
LoanStatus                              object
ClosedDate                              object
BorrowerAPR                            float64
BorrowerRate                           float64
LenderYield                            float64
EstimatedEffectiveYield                float64
EstimatedLoss                          float64
EstimatedReturn                        float64
ProsperRating (numeric)                float64
ProsperRating (Alpha)                   object
ProsperScore                           float64
```

```
ListingCategory (numeric)                  int64
BorrowerState                             object
Occupation                                object
EmploymentStatus                          object
EmploymentStatusDuration                 float64
IsBorrowerHomeowner                         bool
CurrentlyInGroup                            bool
GroupKey                                  object
DateCreditPulled                          object
CreditScoreRangeLower                    float64
CreditScoreRangeUpper                    float64
FirstRecordedCreditLine                   object
CurrentCreditLines                       float64
OpenCreditLines                          float64
                                          ...
TotalProsperLoans                        float64
TotalProsperPaymentsBilled               float64
OnTimeProsperPayments                    float64
ProsperPaymentsLessThanOneMonthLate      float64
ProsperPaymentsOneMonthPlusLate          float64
ProsperPrincipalBorrowed                 float64
ProsperPrincipalOutstanding              float64
ScorexChangeAtTimeOfListing              float64
LoanCurrentDaysDelinquent                  int64
LoanFirstDefaultedCycleNumber            float64
LoanMonthsSinceOrigination                 int64
LoanNumber                                 int64
LoanOriginalAmount                         int64
LoanOriginationDate                       object
LoanOriginationQuarter                    object
MemberKey                                 object
MonthlyLoanPayment                       float64
LP_CustomerPayments                      float64
LP_CustomerPrincipalPayments             float64
LP_InterestandFees                       float64
LP_ServiceFees                           float64
LP_CollectionFees                        float64
LP_GrossPrincipalLoss                    float64
LP_NetPrincipalLoss                      float64
LP_NonPrincipalRecoverypayments          float64
PercentFunded                            float64
Recommendations                            int64
InvestmentFromFriendsCount                 int64
InvestmentFromFriendsAmount              float64
Investors                                  int64
Length: 81, dtype: object
```

In [7]: loan.describe()

Out[7]:

	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]
```

There are 81 columns and not all will be explored

In [8]:
```
#Selecting columns that will be explored

loan = loan.loc[:, ['Term', 'LoanStatus', 'BorrowerState', 'Occupation', 'EmploymentStat
                  'Recommendations', 'MonthlyLoanPayment', 'Investors']]
loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 9 columns):
Term                  113937 non-null int64
LoanStatus            113937 non-null object
BorrowerState         108422 non-null object
Occupation            110349 non-null object
EmploymentStatus      111682 non-null object
LoanOriginalAmount    113937 non-null int64
Recommendations       113937 non-null int64
MonthlyLoanPayment    113937 non-null float64
Investors             113937 non-null int64
dtypes: float64(1), int64(4), object(4)
memory usage: 7.8+ MB
```

In [9]:
```
#Changing the data types of some columns to catergories

loan['Occupation'] = loan['Occupation'].astype('category')
loan['BorrowerState'] = loan['BorrowerState'].astype('category')
loan['LoanStatus'] = loan['LoanStatus'].astype('category')
loan['EmploymentStatus'] = loan['EmploymentStatus'].astype('category')
loan['Term'] = loan['Term'].astype('category')
loan['Recommendations'] = loan['Recommendations'].astype('category')
loan['Investors'] = loan['Investors'].astype('category')
```

```
In [10]: loan['Occupation'].value_counts()
```

```
Out[10]: Other                                  28617
         Professional                           13628
         Computer Programmer                     4478
         Executive                               4311
         Teacher                                 3759
         Administrative Assistant                3688
         Analyst                                 3602
         Sales - Commission                      3446
         Accountant/CPA                          3233
         Clerical                                3164
         Sales - Retail                          2797
         Skilled Labor                           2746
         Retail Management                       2602
         Nurse (RN)                              2489
         Construction                            1790
         Truck Driver                            1675
         Laborer                                 1595
         Police Officer/Correction Officer       1578
         Civil Service                           1457
         Engineer - Mechanical                   1406
         Military Enlisted                       1272
         Food Service Management                 1239
         Engineer - Electrical                   1125
         Food Service                            1123
         Medical Technician                      1117
         Attorney                                1046
         Tradesman - Mechanic                     951
         Social Worker                            741
         Postal Service                           627
         Professor                                557
                                                ...
         Scientist                                372
         Military Officer                         346
         Bus Driver                               316
         Principal                                312
         Teacher's Aide                           276
         Pharmacist                               257
         Student - College Graduate Student       245
         Landscaping                              236
         Engineer - Chemical                      225
         Investor                                 214
         Architect                                213
         Pilot - Private/Commercial               199
         Clergy                                   196
         Student - College Senior                 188
         Car Dealer                               180
```

```
Chemist                                 145
Psychologist                            145
Biologist                               125
Religious                               124
Flight Attendant                        123
Tradesman - Carpenter                   120
Homemaker                               120
Student - College Junior                112
Tradesman - Plumber                     102
Student - College Sophomore              69
Dentist                                  68
Student - College Freshman               41
Student - Community College              28
Judge                                    22
Student - Technical School               16
Name: Occupation, Length: 67, dtype: int64
```

```
In [11]: print(loan.shape)
```

```
(113937, 9)
```

```
In [12]: print(loan.dtypes)
```

```
Term                      category
LoanStatus                category
BorrowerState             category
Occupation                category
EmploymentStatus          category
LoanOriginalAmount           int64
Recommendations           category
MonthlyLoanPayment         float64
Investors                 category
dtype: object
```

0.4.1 What is the structure of your dataset?

The dataset is made up of 113937 rows and 9 columns. Most of the variables are categoric. The LoanOriginalAmount and MonthlyLoanPayment are the only numeric variables.

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

I'm interested in figuring out which features are best for recieving a huge loan original amount in the dataset. I am particulary interested the Term, Original Loan Amount, Recommendations, Employment Status, Investors and any other variable that might be of importance

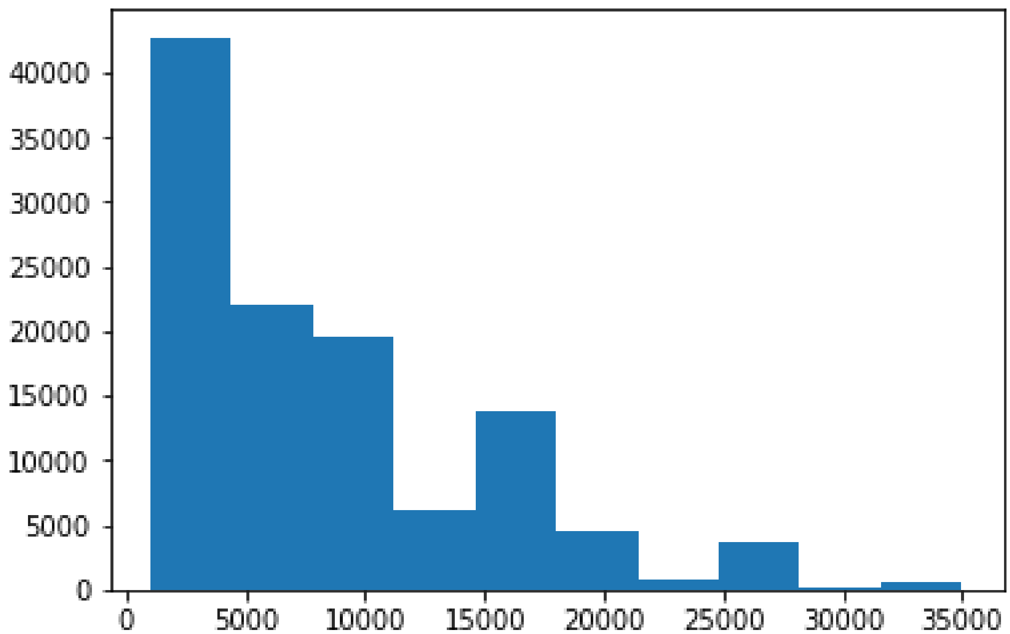
0.4.3 What features in the dataset do you think will help support your investigation into your feature(s) of interest?

I expect that Employment Status and Recommendations will have the strongest effect on each loan amount: the higher number of recommendations a person has, the higher loan amount they receive. Also, people working will receive higher loans. I think the Term and Investors will also have an effect on the loan amount.

0.5 Univariate Exploration

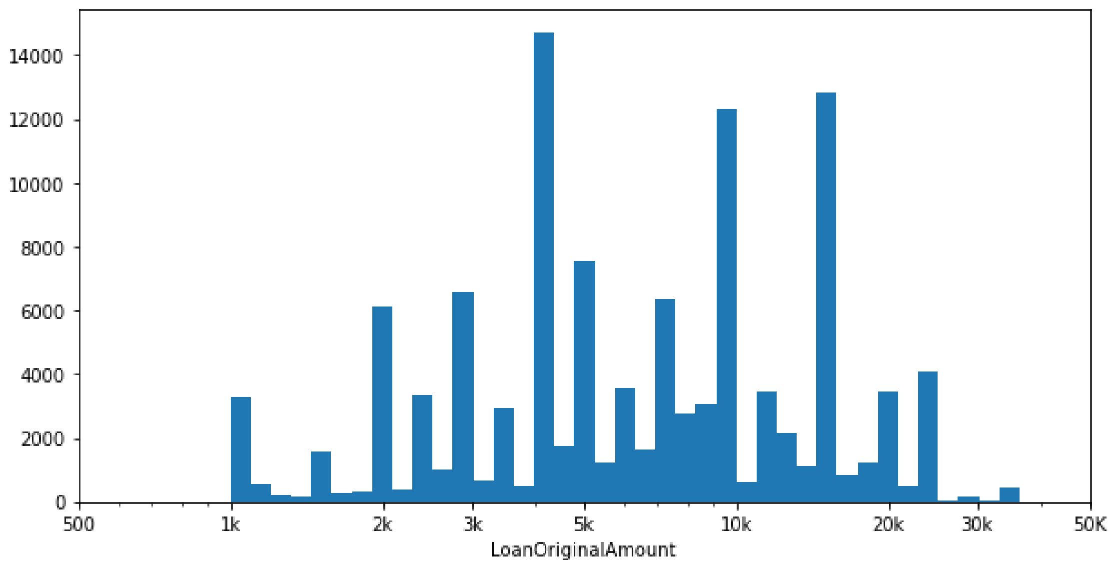
Taking note of the main variable, loan original amount

In [13]:
```
#Plotting loan original amount using a hist graph
plt.hist(data = loan, x= 'LoanOriginalAmount');
```



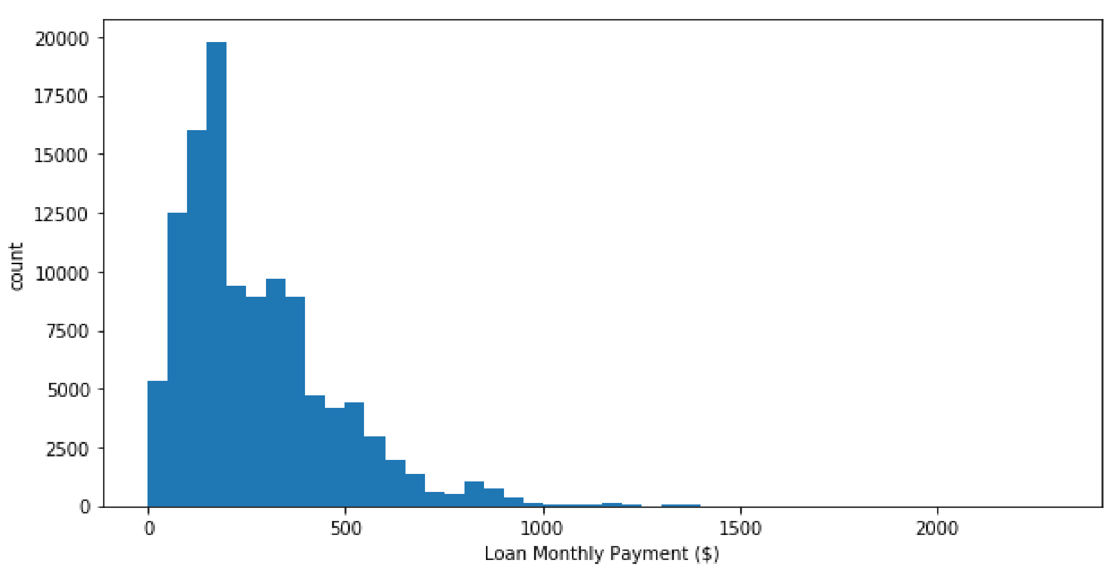
The graph has a long tail. Plotting it on a log scale to see it clearer

In [14]:
```
log_binsize = 0.040
bins = 10 ** np.arange(3, np.log10(loan['LoanOriginalAmount'].max())+log_binsize, log_b

plt.figure(figsize=[10, 5])
plt.hist(data = loan, x = 'LoanOriginalAmount', bins = bins)
plt.xscale('log')
plt.xticks([500, 1e3, 2e3,3e3, 5e3, 1e4, 2e4, 3e4, 5e4], ['500', '1k', '2k', '3k', '5k'
plt.xlabel('LoanOriginalAmount')
plt.show()
```

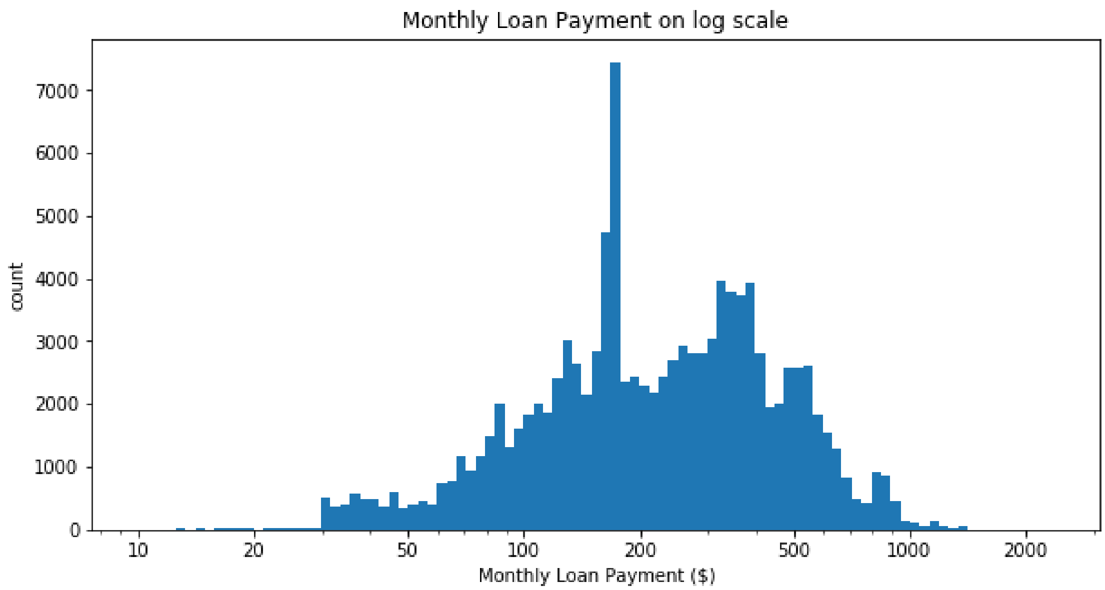


The graph looks multimodal on the log scale with a number of peaks. The first significiant one between 3,000 and 5,000 and the second one just before 10,000. The third peak is between 10,000 and 20,000

```
In [15]: #Plotting MonthlyLoanPayment
         binsize = 50
         bins = np.arange(0, loan['MonthlyLoanPayment'].max()+binsize, binsize)

         plt.figure(figsize=[10, 5])
         plt.hist(data = loan, x = 'MonthlyLoanPayment', bins = bins)
         plt.xlabel('Loan Monthly Payment ($)')
         plt.ylabel('count')
         plt.show()
```



In [16]:
```
#Plotting MonthlyLoanPayemnt on a log scale
log_binsize = 0.025
bins = 10 ** np.arange(1, np.log10(loan['MonthlyLoanPayment'].max())+log_binsize, log_b

plt.figure(figsize=[10, 5])
plt.hist(data = loan, x = 'MonthlyLoanPayment', bins = bins)
plt.xscale('log')
plt.xticks([10, 20, 50, 100, 200, 500, 1e3, 2e3], ['10', '20','50', '100', '200', '500'
plt.xlabel('Monthly Loan Payment ($)')
plt.ylabel('count')
plt.title('Monthly Loan Payment on log scale')
plt.show()
```



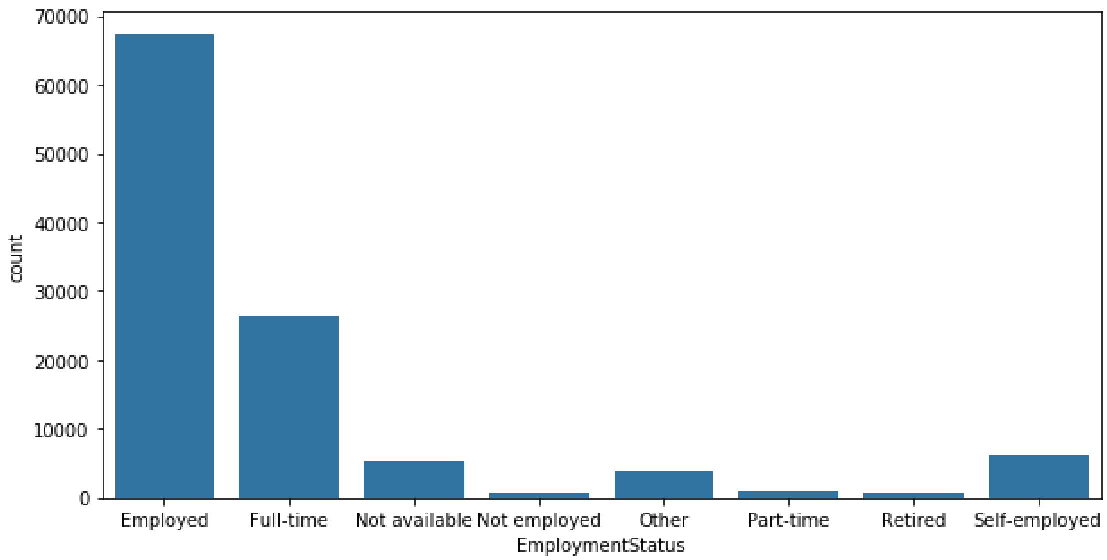
The log scale graph of Monthly Loan Payment has the highest peak between 100 and 200 ($)

In [17]:
```
#Plotting graphs for other variables of interest
#(Employment Status, Investors, Term, Recommendations, Occupation, Borrower State, Loan

sorted_counts = loan['EmploymentStatus'].value_counts()
sorted_counts
```

Out[17]:
```
Employed         67322
Full-time        26355
Self-employed     6134
Not available     5347
Other             3806
```

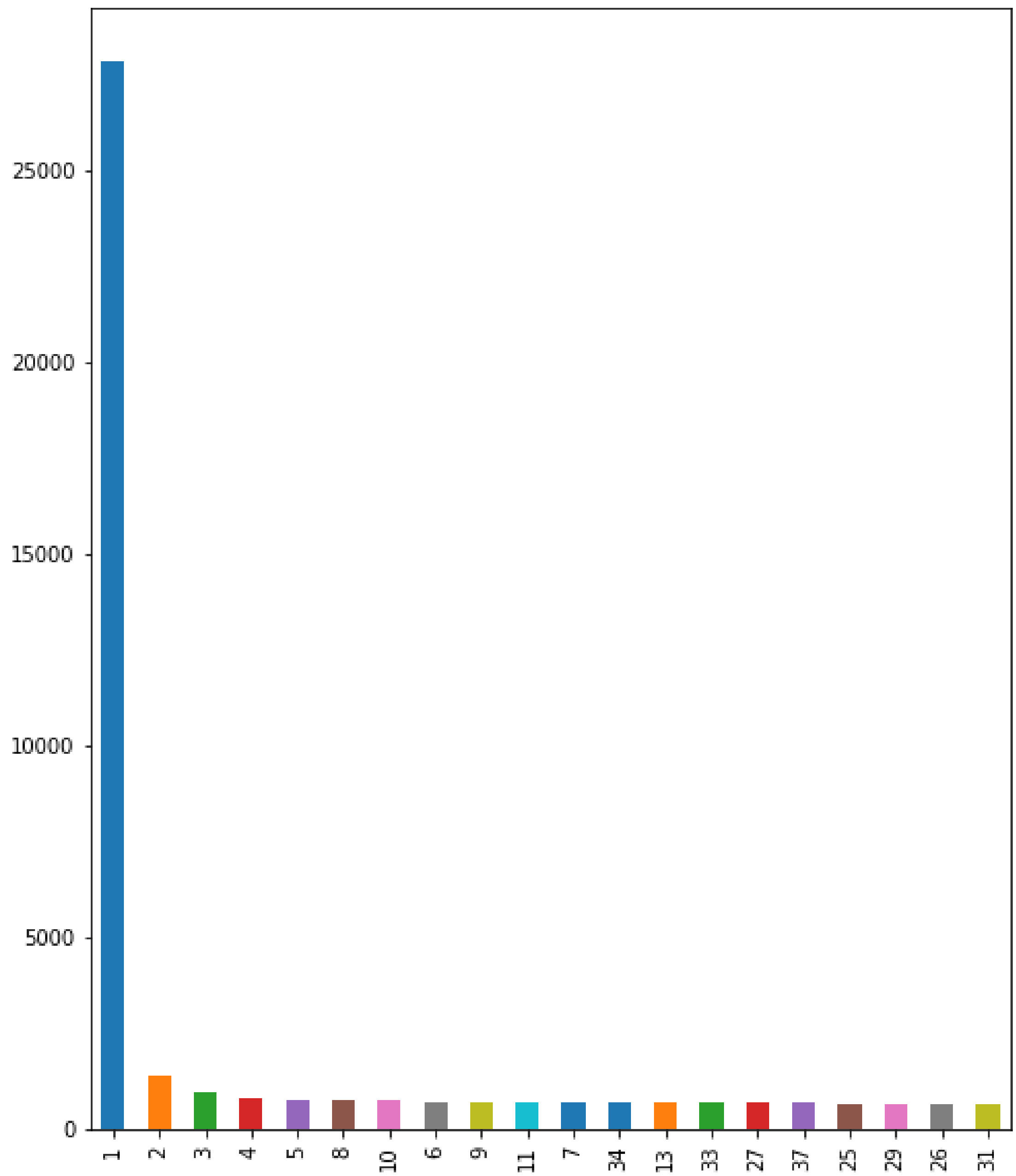
```
Part-time          1088
Not employed        835
Retired             795
Name: EmploymentStatus, dtype: int64
```

In [18]:
```
plt.figure(figsize=[10,5])
default_color = sb.color_palette()[0]
sb.countplot(data = loan, x = 'EmploymentStatus', color = default_color);
```



From the bar plot, most borrowers are employed

In [19]:
```
#Investors Graph
investors = loan['Investors'].value_counts().head(20)
plt.figure(figsize=[8,10])
investors.plot(kind='bar');
```

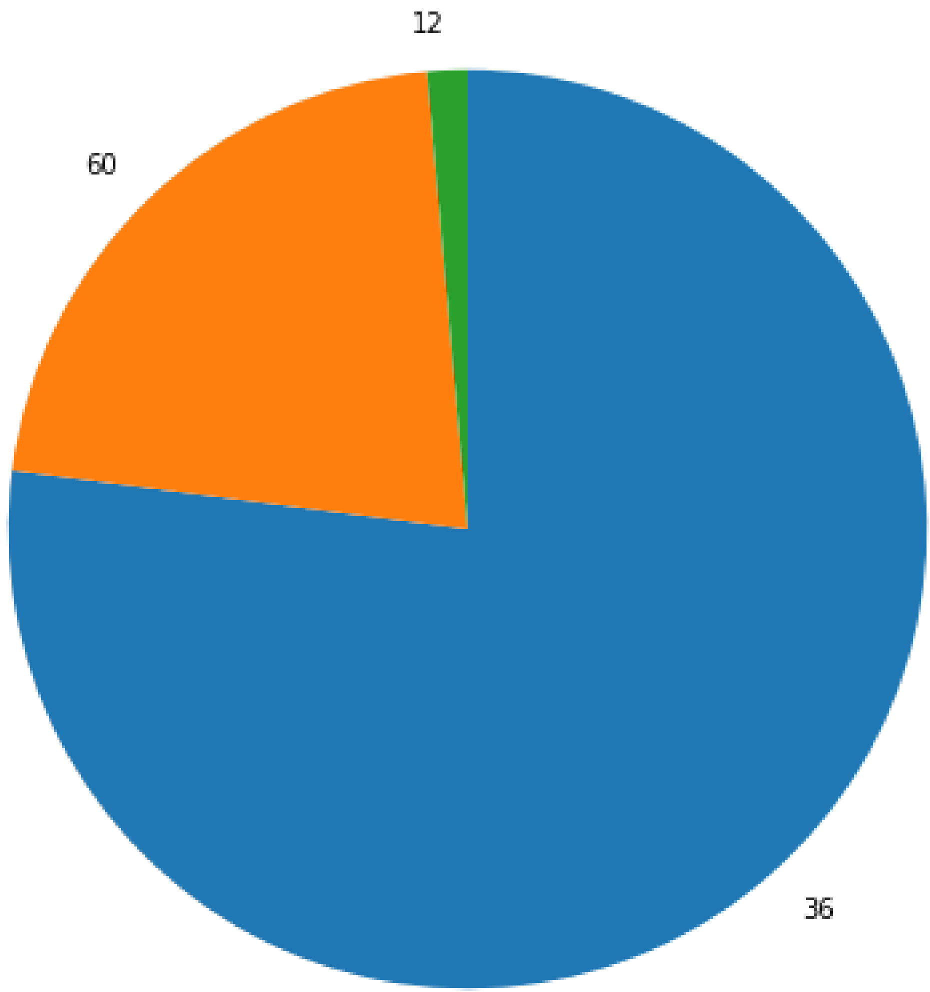


From the bar graph above, the first 20 loans were funded by one investor

```
In [20]: #Term Graph
         loan['Term'].value_counts()

Out[20]: 36    87778
         60    24545
         12     1614
         Name: Term, dtype: int64
```

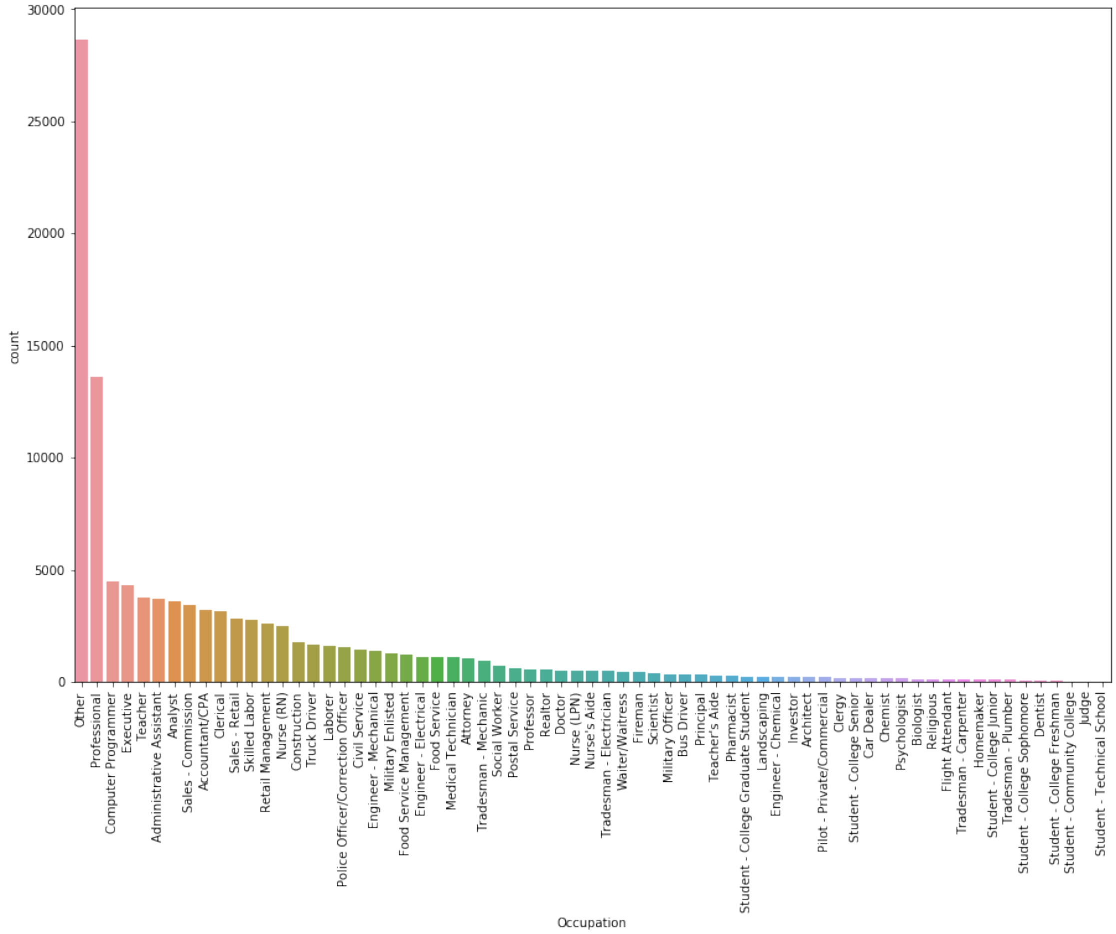
The Terms for the loans were 12, 36 and 60 months.

```
In [21]: plt.figure(figsize=[7,7])
         sorted_counts = loan['Term'].value_counts()
         plt.pie(sorted_counts, labels = sorted_counts.index,startangle = 90, counterclock=False
         plt.axis('square');
```

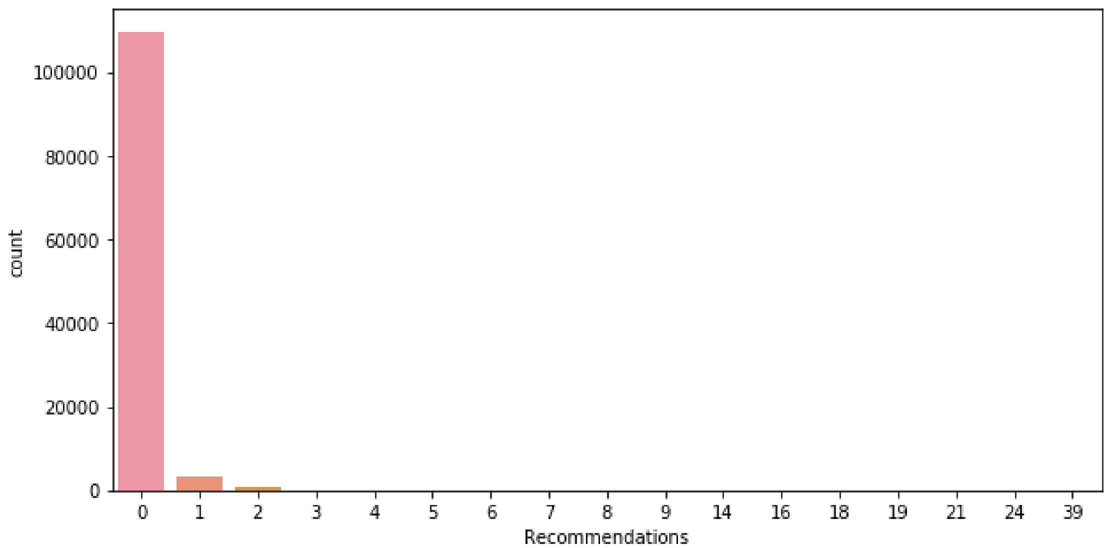


Most of the loans given out were for the 36 months term

```
In [22]: #Occupation
         cat_order = loan['Occupation'].value_counts().index
         plt.figure(figsize=[15, 10])
         sb.countplot(data=loan,x='Occupation', order=cat_order);
         plt.xticks(rotation=90);
```

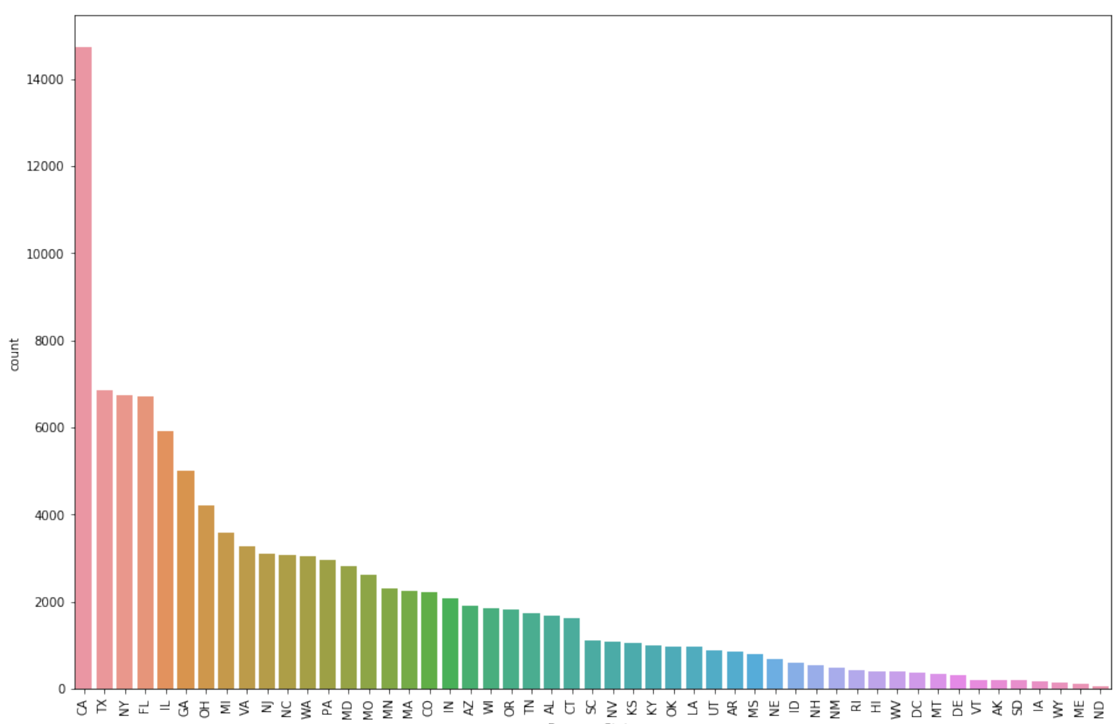


In [23]:
```
#Recommemdations
plt.figure(figsize=[10, 5])
sb.countplot(data=loan,x='Recommendations');
```



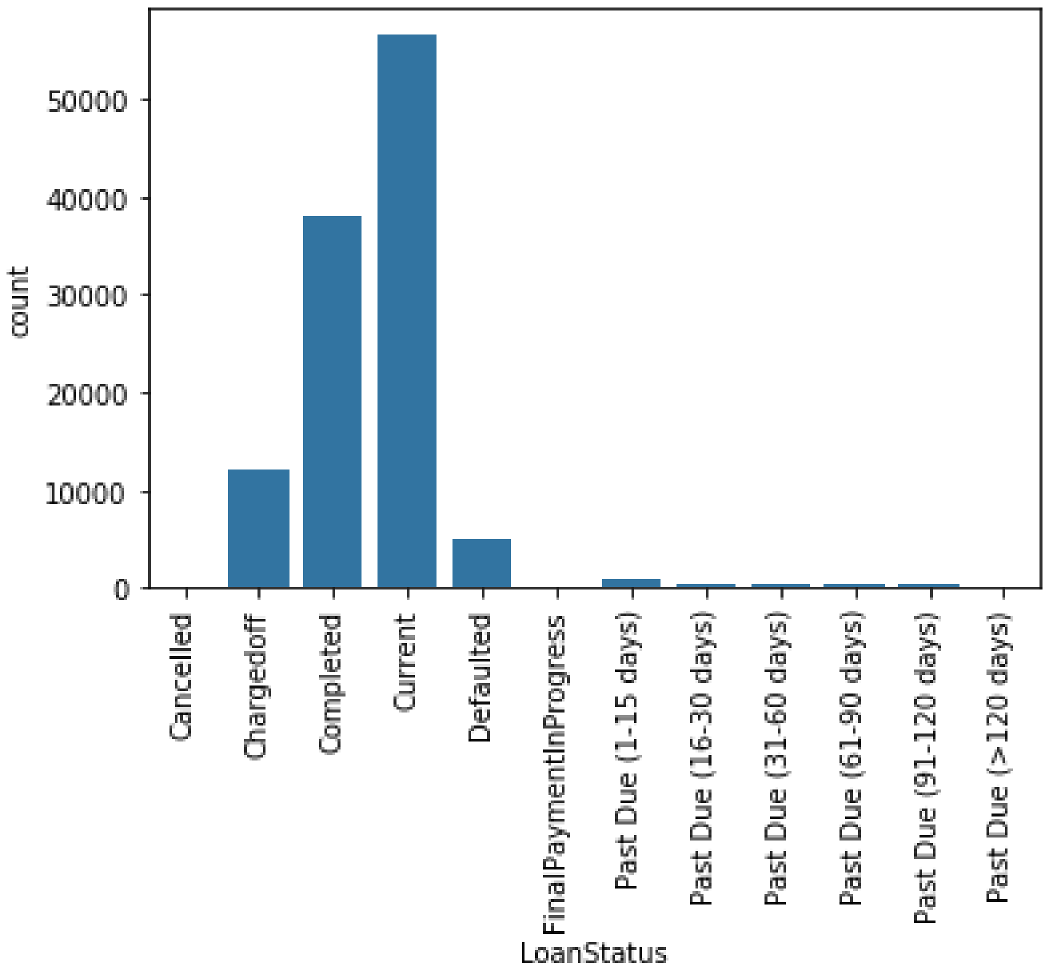
Most of the loans given out had no Recommendations

```
In [24]: #Borrower State
         cat_order = loan['BorrowerState'].value_counts().index
         plt.figure(figsize=[15, 10])
         sb.countplot(data=loan,x='BorrowerState', order=cat_order);
         plt.xticks(rotation=90);
```



Califorina (CA) is the state that had most borrowers. It is followed by Texas (TX) and New York (NY). With North Dakota (ND) as the state with the least borrowers

```
In [25]: #Loan Status
         base_color=sb.color_palette()[0]
         sb.countplot(data=loan, x='LoanStatus', color=base_color);
         plt.xticks(rotation=90);
```



Most of the loan are current (in existence)

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

The LoanOriginalAmount variable had a long tail when plotted on a histogram chart. It was then plotted on a log scale and had three significant peaks. The first significiant one between 3,000 and 5,000 and the second one just before 10,000. The third peak is between 10,000 and 20,000

0.5.2 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?

Most of the variables investigated looked like int datatype from the onset but are actually categories so they were changed using the .astype function. The MonthlyLoanPayment variable had a long tail; it was plotted on a log scale to get a clearer view

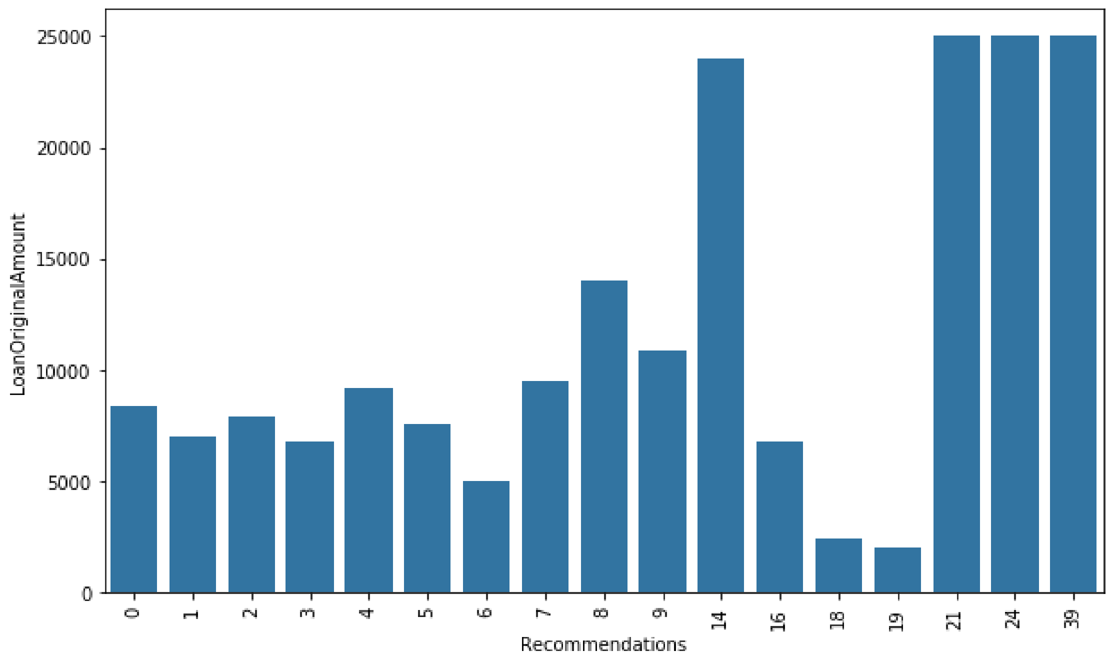
0.6 Bivariate Exploration

Two variables will be plotted against each other and conclusions will be drawn

```
In [26]: #Findng the mean of the LoanOriginalAmount and the number of Recommendations
         data = loan[['Recommendations','LoanOriginalAmount']]
         data = data.groupby('Recommendations')['LoanOriginalAmount'].mean()
         data
```

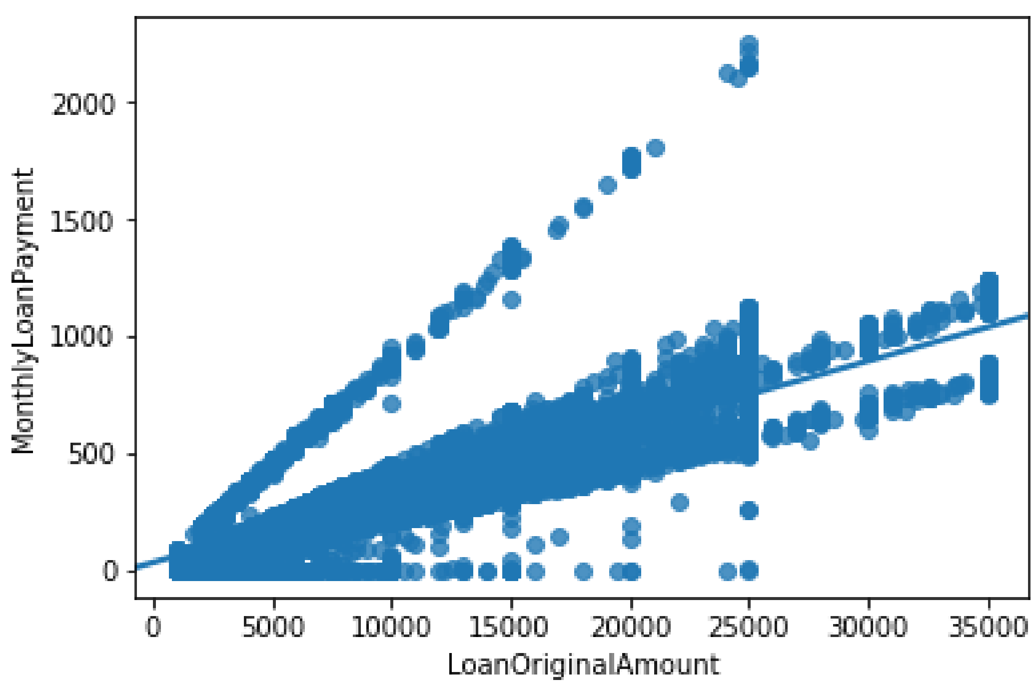
```
Out[26]: Recommendations
         0       8383.879073
         1       6969.317975
         2       7912.802817
         3       6762.370370
         4       9194.038462
         5       7536.428571
         6       5025.000000
         7       9466.600000
         8      14000.000000
         9      10830.000000
         14     24000.000000
         16      6750.000000
         18      2430.000000
         19      2000.000000
         21     25000.000000
         24     25000.000000
         39     25000.000000
         Name: LoanOriginalAmount, dtype: float64
```

```
In [27]: #Plotting Recommendations against LoanOriginalAmount
         plt.figure(figsize = [10, 6])
         sb.barplot(data=loan,x='Recommendations',y='LoanOriginalAmount',color=base_color,errwid
         plt.xticks(rotation=90);
```



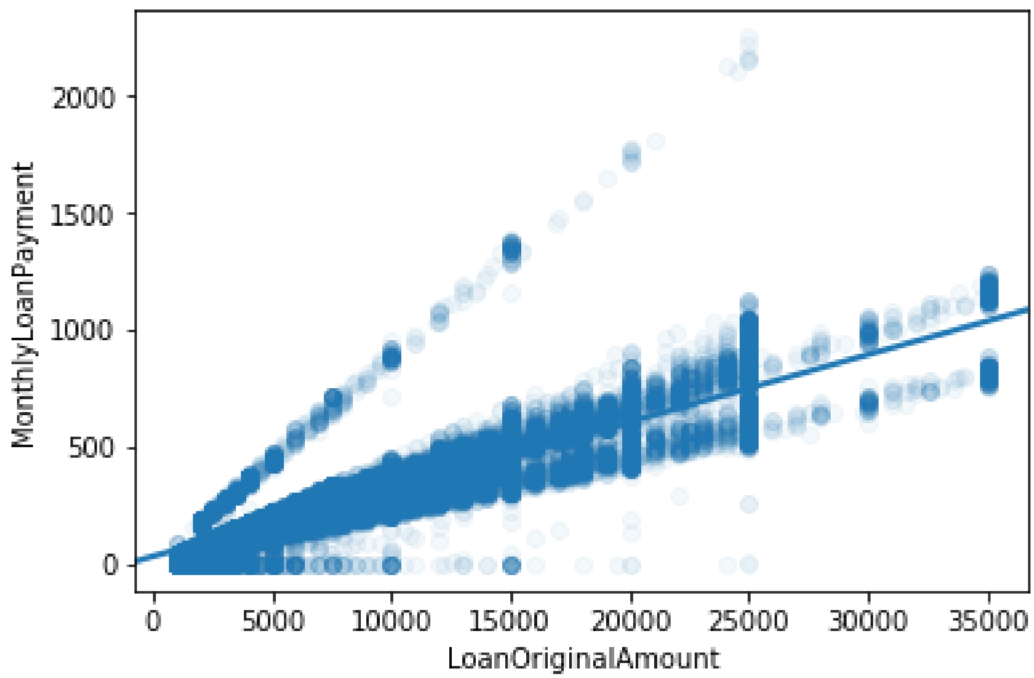
On the average, borrowers with 21, 24 and 39 recommenders had the highest loan amount which was around 25,000. Those with 14 recommenders were closely behind; receiving approximately more than 24,000

```
In [28]: #LoanOriginalAmount vs MonthlyLoanPayment
         sb.regplot(data=loan, x='LoanOriginalAmount', y='MonthlyLoanPayment');
```



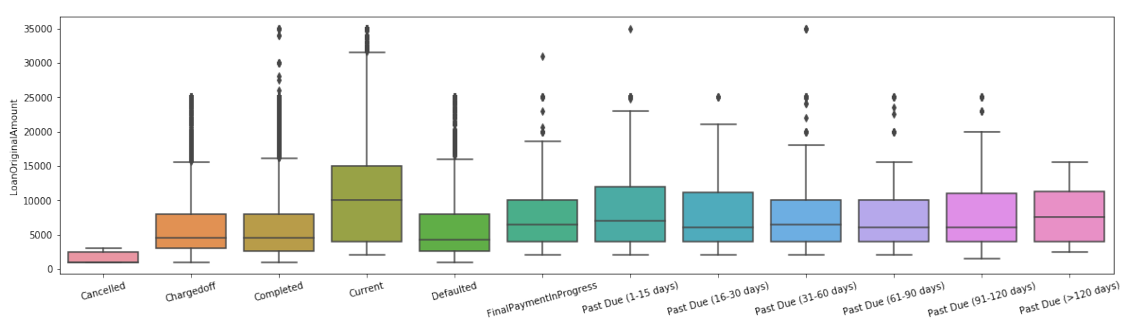
The points are overlapping. Let's use jitter and transparency to get a good view

```
In [29]: sb.regplot(data=loan, x='LoanOriginalAmount', y='MonthlyLoanPayment', truncate=False, x
```

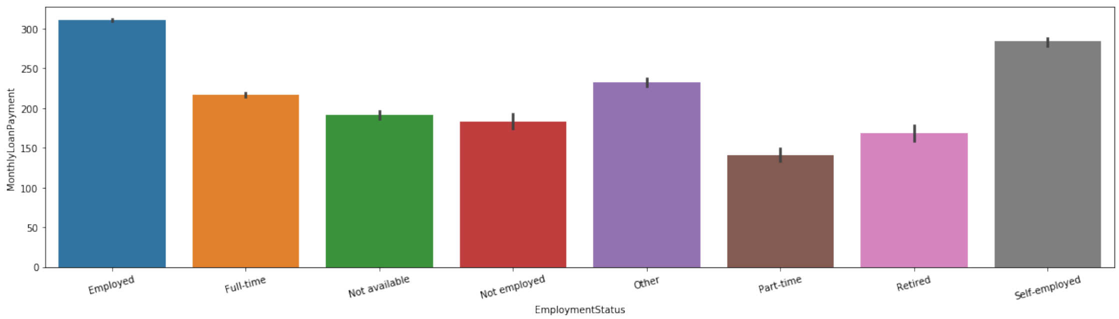


The scatter plot shows a positive correlation between LoanOriginalAmount and Monthly-LoanAmount

```
In [30]: #LoanOriginalAmount vs LoanStatus
         plt.figure(figsize = [20, 5])
         sb.boxplot(data=loan, y='LoanOriginalAmount', x='LoanStatus');
         plt.xticks(rotation=15);
```

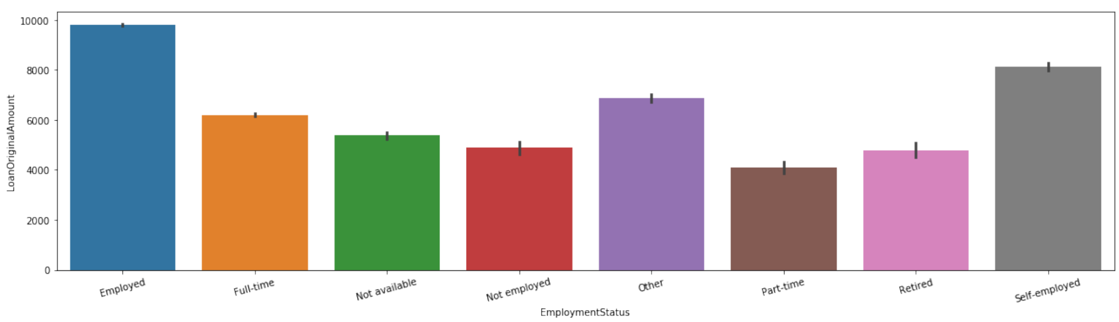


```
In [31]: #MonthlyLoanPayment vs EmploymentStatus
         plt.figure(figsize = [20, 5])
         sb.barplot(data=loan,y='MonthlyLoanPayment',x='EmploymentStatus');
         plt.xticks(rotation=15);
```



Borrowers who are employed and self employed payed most of thier loans monthly.

```
In [32]: #LoanOriginalAmount vs EmploymentStatus
         plt.figure(figsize = [20, 5])
         sb.barplot(data=loan,y='LoanOriginalAmount',x='EmploymentStatus');
         plt.xticks(rotation=15);
```



Borrowers who are employed and self employed received huge loan amounts.

0.6.1 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?

From the relation between Recommenders and LoanOriginalAmount, those with higher recommenders received huge loan amounts. However, there is an exception since those with 14 recommenders received huger amounts than those with 16, 18 and 19 recommenders. Those who are employed and self-employed also received huge amounts of loan and there was an exception there too. The Not Available and Retired employment statutes received more loans than those working part time. The inital expection of borrowers with high recommendations and the working class receiving more loans has been disproved.

0.6.2 Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

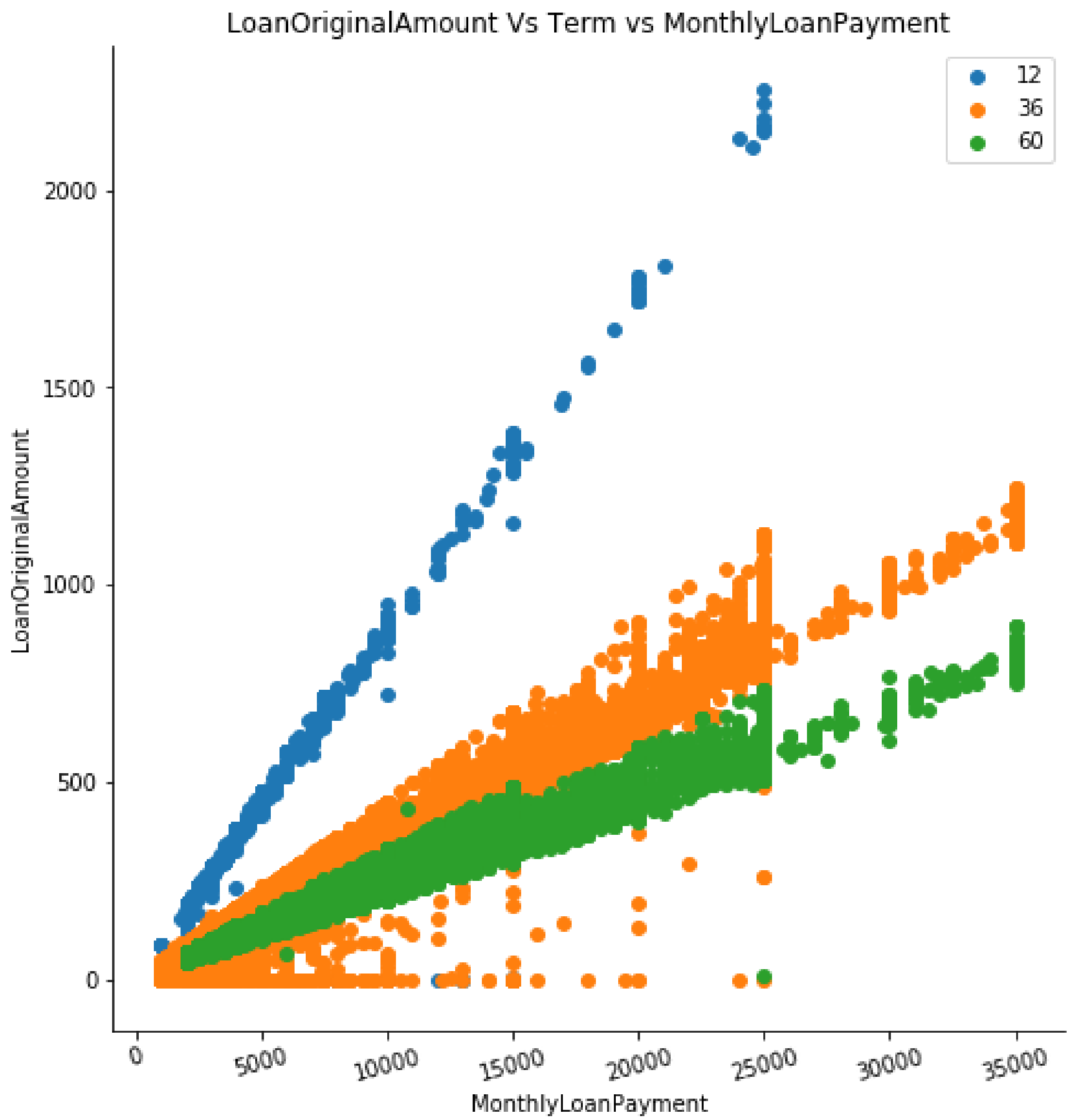
The relation between Employment Status and MonthlyLoanPayment is an interesting one. The Employed and Self Employed workers payed most the loans monthly. However, the Retired payed more loans than those working part-time

0.7 Multivariate Exploration

Investigating three or more variables at once to draw further insights

```
In [33]: #LoanOriginalAmount vs LoanStatus vs EmploymentStatus
         plt.figure(figsize=[8,5]);
         g=sb.FacetGrid(data=loan, hue="Term", size= 7)
         g.map(plt.scatter, 'LoanOriginalAmount', 'MonthlyLoanPayment');
         plt.title('LoanOriginalAmount Vs Term vs MonthlyLoanPayment')
         plt.xlabel('MonthlyLoanPayment')
         plt.ylabel('LoanOriginalAmount');
         plt.xticks(rotation = 15)
         plt.legend();
```

```
<matplotlib.figure.Figure at 0x7effd122e588>
```



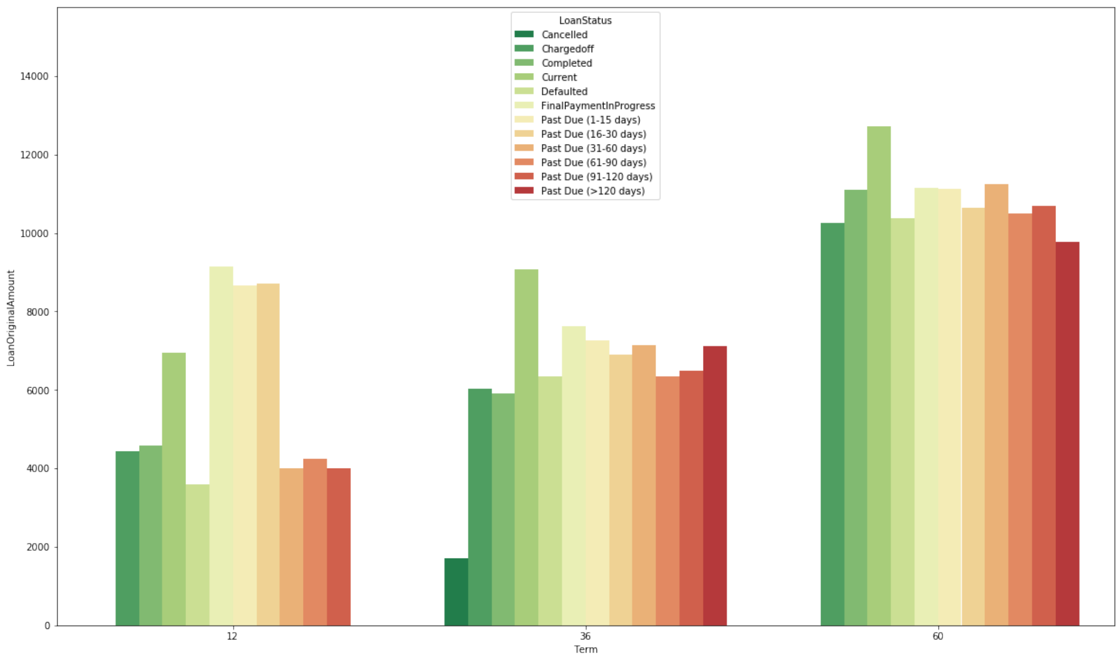
There is a positive correlation between LoanOriginalAmount, MonthlyLoanPayment and Term

```
In [34]: loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 9 columns):
Term                   113937 non-null category
LoanStatus             113937 non-null category
BorrowerState          108422 non-null category
Occupation             110349 non-null category
EmploymentStatus       111682 non-null category
```

```
LoanOriginalAmount    113937 non-null int64
Recommendations       113937 non-null category
MonthlyLoanPayment    113937 non-null float64
Investors             113937 non-null category
dtypes: category(7), float64(1), int64(1)
memory usage: 2.6 MB
```

```
In [35]: #Term vs LoanOrignalAmount vs LoanStatus
         plt.figure(figsize=[20,12])
         sb.barplot(x="Term", y="LoanOriginalAmount", hue='LoanStatus', data=loan, errwidth=Fals
```



Loans with longer terms generally have the greatest mean LoanoriginalAmount. For each loan status across the term, is an increasing caterogy

0.7.1 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?

Taking into consideration Term as a variable of interest, from the barplot, borrowers received huge loans, approximately 13,000 with the highest term of sixty (60). Most of those borrowers too had the 'completed' loan status. Borrowers on the sixty term received huge loan original amounts and have completed payment.

0.7.2 Were there any interesting or surprising interactions between features?

I think it interesting that most borrowers with the longest term (Sixty months) have completed their loan or are in the final payment stage. There is also a positive correlation between LoanOriginalAmount, MonthlyLoanPayment and Term. Regardless of the Term or OriginalLoanAmount, borrowers were paying their loans monthly.

```
In [ ]:
```