Part1_Loans_Exploration

November 30, 2022

1 Part I - (Prosper Loan Data Exploration)

1.1 by (Innocent Mukoki)

1.2 Introduction

The Prosper Loan Data is a dataset which contains 113937 loans and each loan has 81 variables. Some of the variables in the dataset are loan amount, income range, loan status, borrower rate and listing category.

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 sb

%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]:
```
# load in the dataset into a pandas dataframe, print statistics
loan_data = pd.read_csv('prosperLoanData.csv')
loan_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 [3]: # high-level overview of data shape and composition
        print(loan_data.shape)
        print(loan_data.dtypes)
        print(loan_data.head(10))
```

```
(113937, 81)
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
              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
5  0F05359734824199381F61D        1074836  2013-12-14 08:26:37.093000000
6  0F0A3576754255009D63151         750899  2013-04-12 09:52:56.147000000
7  0F1035772717087366F9EA7         768193  2013-05-05 06:49:27.493000000
8  0F043596202561788EA13D5        1023355  2013-12-02 10:43:39.117000000
9  0F043596202561788EA13D5        1023355  2013-12-02 10:43:39.117000000

  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
5         NaN    60    Current                  NaN      0.15425
6         NaN    36    Current                  NaN      0.31032
7         NaN    36    Current                  NaN      0.23939
8         NaN    36    Current                  NaN      0.07620
9         NaN    36    Current                  NaN      0.07620

   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
5        0.1314       0.1214    ...             -25.33                0.0
6        0.2712       0.2612    ...             -22.95                0.0
7        0.2019       0.1919    ...             -69.21                0.0
8        0.0629       0.0529    ...             -16.77                0.0
9        0.0629       0.0529    ...             -16.77                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
5                    0.0                  0.0                             0.0
6                    0.0                  0.0                             0.0
7                    0.0                  0.0                             0.0
8                    0.0                  0.0                             0.0
9                    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
5            1.0                0                          0
6            1.0                0                          0
7            1.0                0                          0
8            1.0                0                          0
9            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                         0.0         1
6                         0.0         1
7                         0.0         1
8                         0.0         1
9                         0.0         1

[10 rows x 81 columns]
```

In [4]:
```
# Converting the loan origination date column to datetime type
loan_data['LoanOriginationDate'] = pd.to_datetime(loan_data['LoanOriginationDate'])
```

In [5]:
```
# Creating a listing category column with the categories being strings
list_category={0:'Not Available', 1:'Debt Consolidation', 2:'Home Improvement', 3:'Busin
 6:'Auto', 7:'Other', 8:'Baby&Adoption', 9:'Boat', 10:'Cosmetic Procedure', 11:'Engageme
 13:'Household Expenses', 14:'Large Purchases', 15:'Medical/Dental', 16:'Motorcycle', 17
 20:'Wedding Loans'}

categories = []
for i in range(len(loan_data)):
    categories.append(list_category[loan_data['ListingCategory (numeric)'][i]])

loan_data['ListingCategory']=categories
```

In [6]:
```
# Creating a column with only quarter
quarters = []
for i in range(len(loan_data)):
    quarters.append(loan_data['LoanOriginationQuarter'][i][0:2])
```

```
loan_data['OriginationQuarter'] = quarters
loan_data['OriginationQuarter']
```

```
Out[6]: 0         Q3
        1         Q1
        2         Q1
        3         Q4
        4         Q3
        5         Q4
        6         Q2
        7         Q2
        8         Q4
        9         Q4
        10        Q2
        11        Q4
        12        Q1
        13        Q3
        14        Q2
        15        Q2
        16        Q3
        17        Q3
        18        Q1
        19        Q4
        20        Q4
        21        Q4
        22        Q1
        23        Q2
        24        Q4
        25        Q4
        26        Q1
        27        Q2
        28        Q4
        29        Q1
                  ..
        113907    Q4
        113908    Q4
        113909    Q3
        113910    Q1
        113911    Q4
        113912    Q4
        113913    Q2
        113914    Q3
        113915    Q3
        113916    Q4
        113917    Q4
        113918    Q2
        113919    Q2
        113920    Q2
```

```
113921    Q4
113922    Q3
113923    Q3
113924    Q4
113925    Q2
113926    Q3
113927    Q2
113928    Q2
113929    Q3
113930    Q3
113931    Q1
113932    Q2
113933    Q4
113934    Q4
113935    Q4
113936    Q1
Name: OriginationQuarter, Length: 113937, dtype: object
```

In [7]:
```python
# Converting loan status, income verifiable, is borrower homeowner and listing category
categories = ['LoanStatus', 'IncomeVerifiable', 'IsBorrowerHomeowner', 'ListingCategory'
for category in categories:
    loan_data[category] = loan_data[category].astype('category')

print(loan_data.dtypes)
```

```
ListingKey                              object
ListingNumber                            int64
ListingCreationDate                     object
CreditGrade                             object
Term                                     int64
LoanStatus                            category
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                   category
CurrentlyInGroup                          bool
```

```
GroupKey                                        object
DateCreditPulled                                object
CreditScoreRangeLower                          float64
CreditScoreRangeUpper                          float64
FirstRecordedCreditLine                         object
CurrentCreditLines                             float64
OpenCreditLines                                float64
                                             ...
OnTimeProsperPayments                          float64
ProsperPaymentsLessThanOneMonthLate            float64
ProsperPaymentsOneMonthPlusLate                float64
ProsperPrincipalBorrowed                       float64
ProsperPrincipalOutstanding                    float64
ScorexChangeAtTimeOfListing                    float64
LoanCurrentDaysDelinquent                        int64
LoanFirstDefaultedCycleNumber                  float64
LoanMonthsSinceOrigination                       int64
LoanNumber                                       int64
LoanOriginalAmount                               int64
LoanOriginationDate                     datetime64[ns]
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
ListingCategory                               category
OriginationQuarter                              object
Length: 83, dtype: object
```

In [8]:
```
# descriptive statistics for numeric variables
print(loan_data.describe())
```

```
       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]
```

1.3.1 What is the structure of your dataset?

There are 113,937 loans in the dataset with 81 variables. Most variables are numeric in nature. The dataset can be found here and the feature documentation can be found here

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

I am interested in finding out the factors that affect the original loan amount.

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

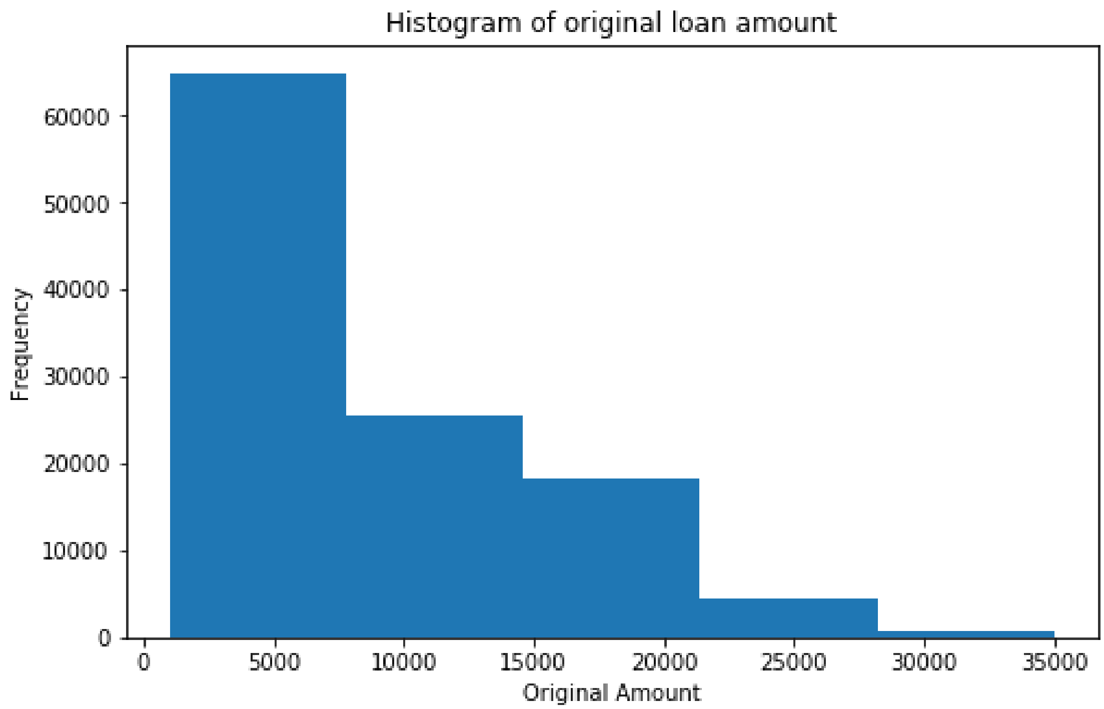
The features in the dataset which I think will help support my investigation into the original loan amount are income range, loan status, monthly income, income verifiable, months since origination, origination quarter, origination date, is borrower homeowner, term, borrow rate, listing category and monthly loan payment.

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.

I'll start by looking at the distribution of the main variable of interest: original amount.

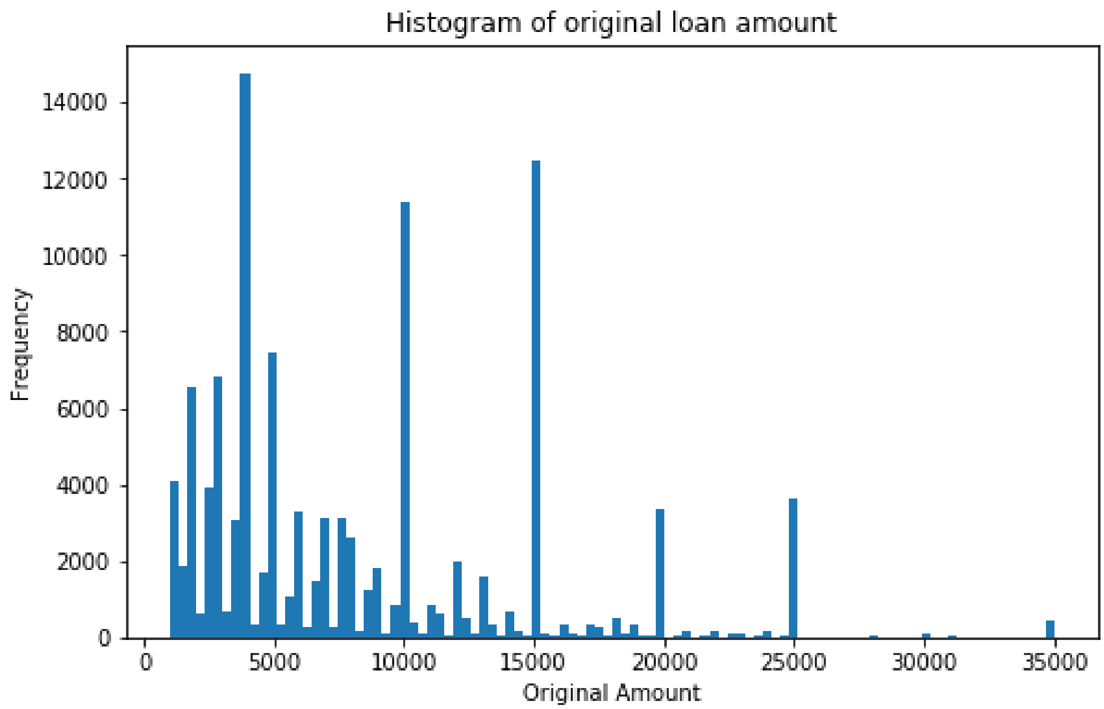
```
In [9]: # Plotting a histogram for original loan amount
        plt.figure(figsize=[8, 5])
        base_color = sb.color_palette()[0]
        loan_data['LoanOriginalAmount'].hist(bins=5, grid=False, color=base_color)

        plt.title('Histogram of original loan amount')
        plt.xlabel('Original Amount')
        plt.ylabel('Frequency')
        plt.show()
```



Most of the loans are small amounts around $5000 and start decreasing as the loan amounts decrease.

Increasing the number of bins

```
In [10]: # Plotting a histogram for original loan amount with more bins
         plt.figure(figsize=[8, 5])
         base_color = sb.color_palette()[0]
         loan_data['LoanOriginalAmount'].hist(bins=100, grid=False)

         plt.title('Histogram of original loan amount')
         plt.xlabel('Original Amount')
         plt.ylabel('Frequency')
         plt.show()
```

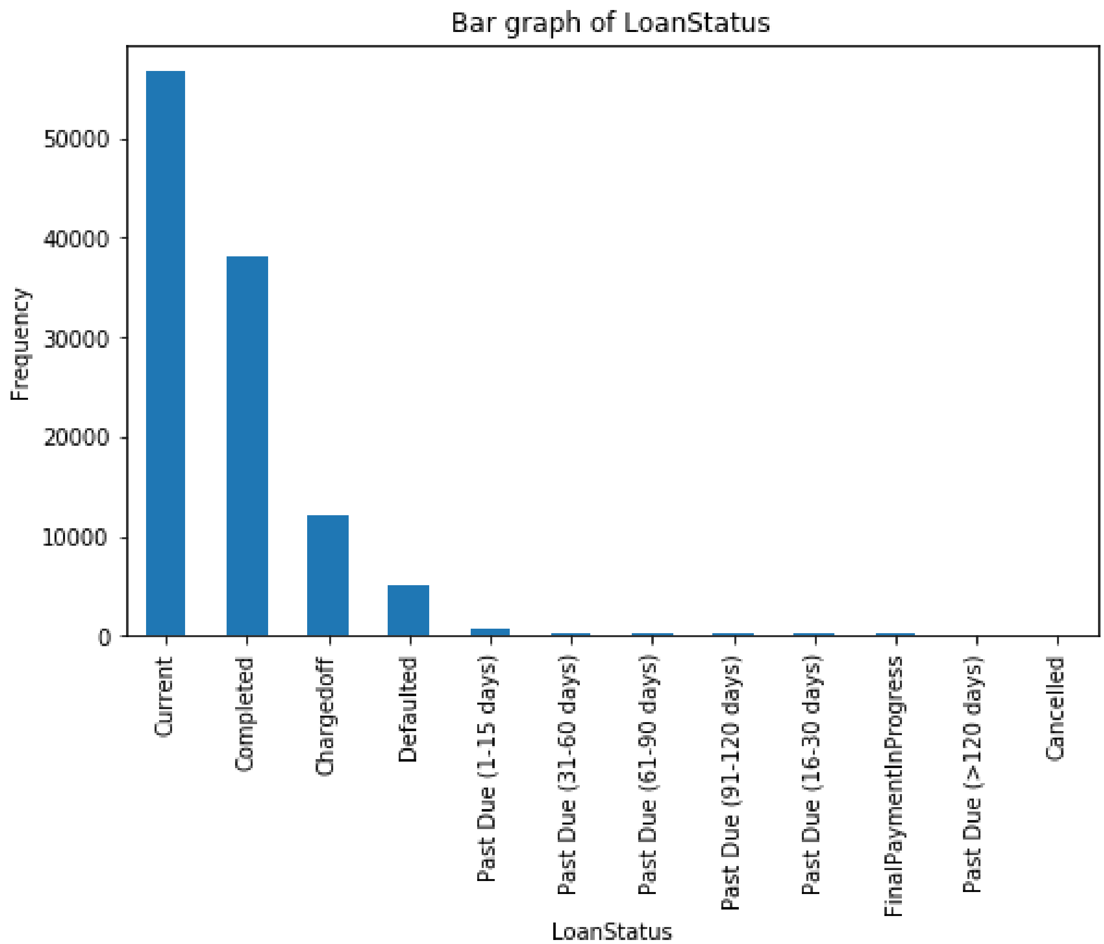


Generally most of the loans have a smaller amount and the number of loans starts decreasing as the original loan amount increases. There are some peaks along the way.

Looking at the distribution of the other variables

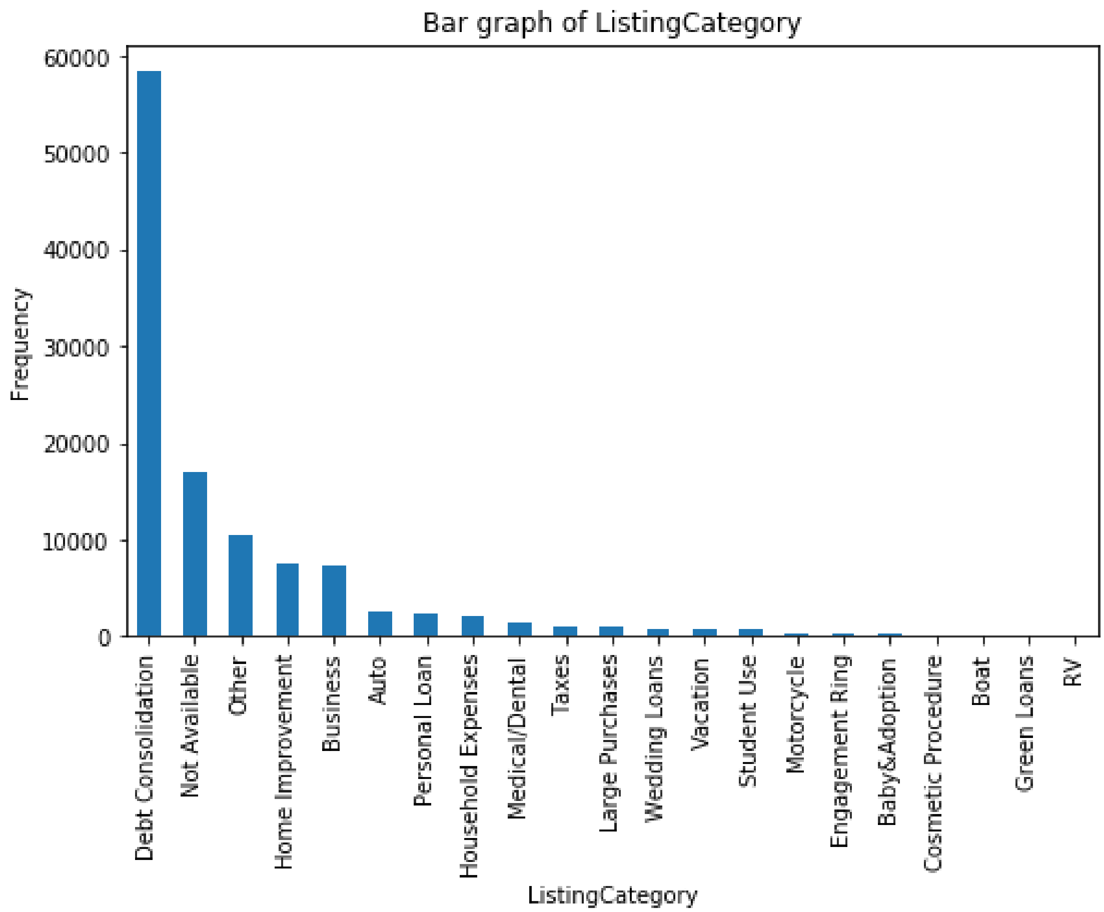
In [11]:
```
# Function to plot bar graphs
def plot_bar(x):
    plt.figure(figsize=[8, 5])
    base_color = sb.color_palette()[0]
    loan_data[x].value_counts().plot(kind='bar', color=base_color)
    plt.title('Bar graph of {}'.format(x))
    plt.xlabel(x)
    plt.ylabel('Frequency')
    plt.xticks(rotation=90)
    plt.show()
```

In [12]:
```
# looking at the distribution of loan status.
plot_bar(x = 'LoanStatus')
```



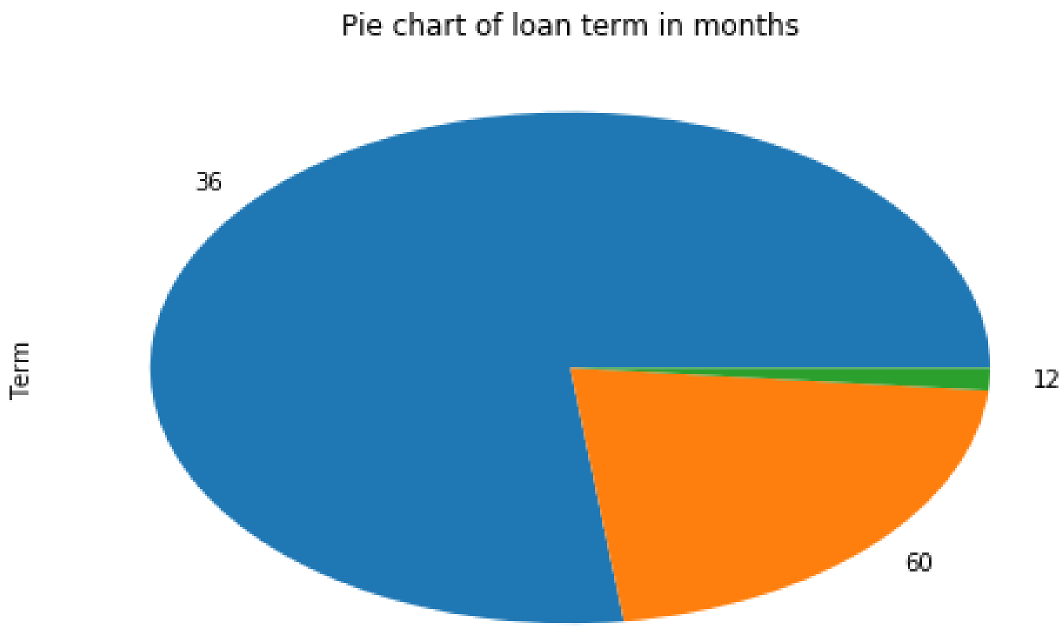
It can be seen from the bar graph that majority of loans are still current followed by completed loans and cancelled loans have the least frequency.

```
In [13]: # looking at the distribution of listing category.
         plot_bar(x = 'ListingCategory')
```



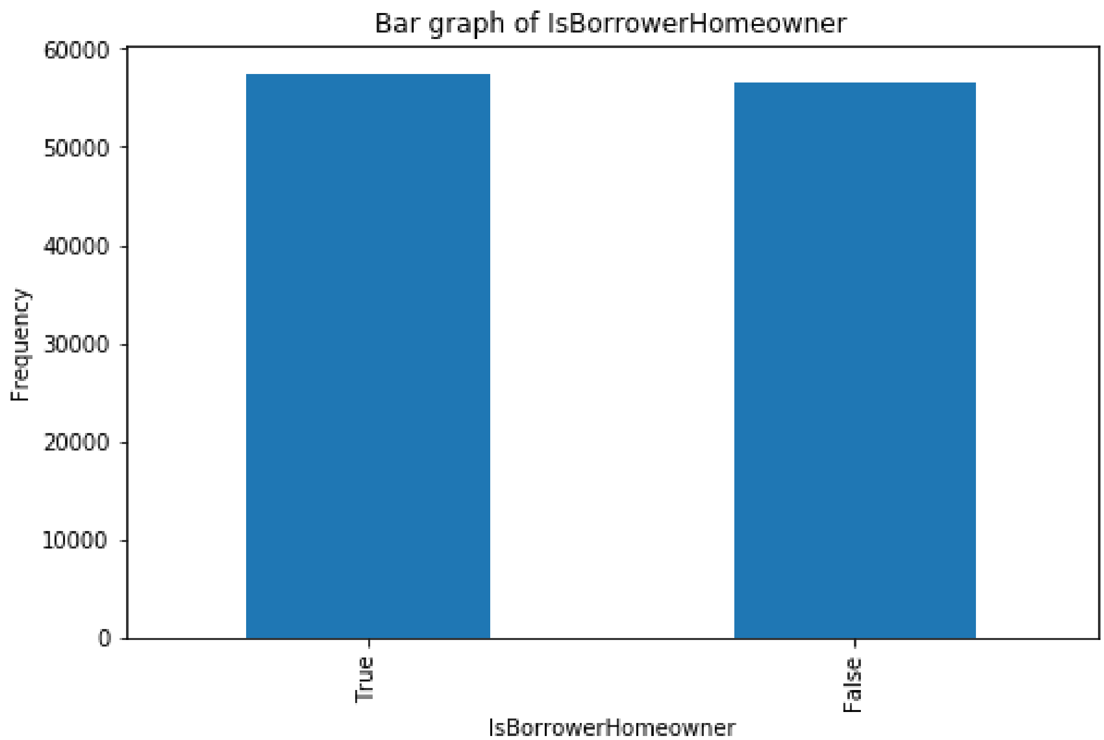
Most people borrowed money to consolidate their loans, followed by not available and the least number of people borrowed for RV. It can be seen that as the number of people decreases the loans become more luxurious.

```
In [14]: # Looking at the distribution of loan term
         loan_term = loan_data['Term'].value_counts()

         plt.figure(figsize=[8, 5])
         loan_term.plot(kind='pie')
         plt.title('Pie chart of loan term in months')
         plt.show()
```



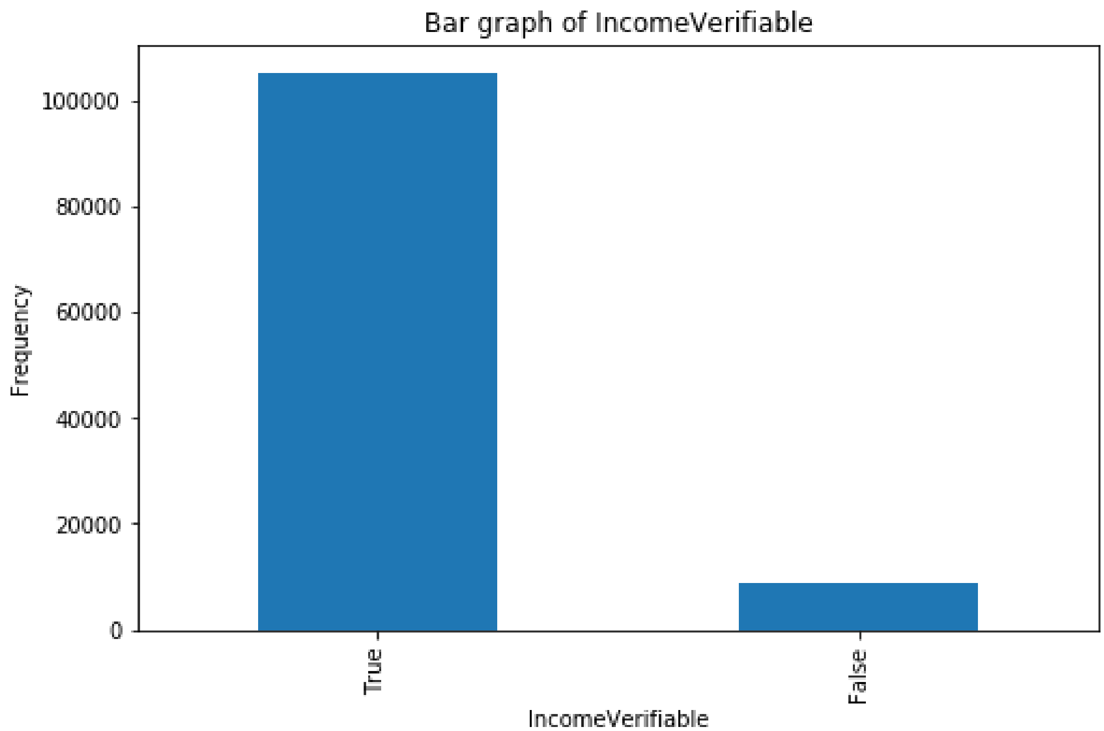
It can be seen from the pie chart that the length of most loans is 36 months, followed by 60 months and lastly 12 months.

```
In [15]: # Looking at the distribution for is borrower homeowner
         plot_bar(x = 'IsBorrowerHomeowner')
```

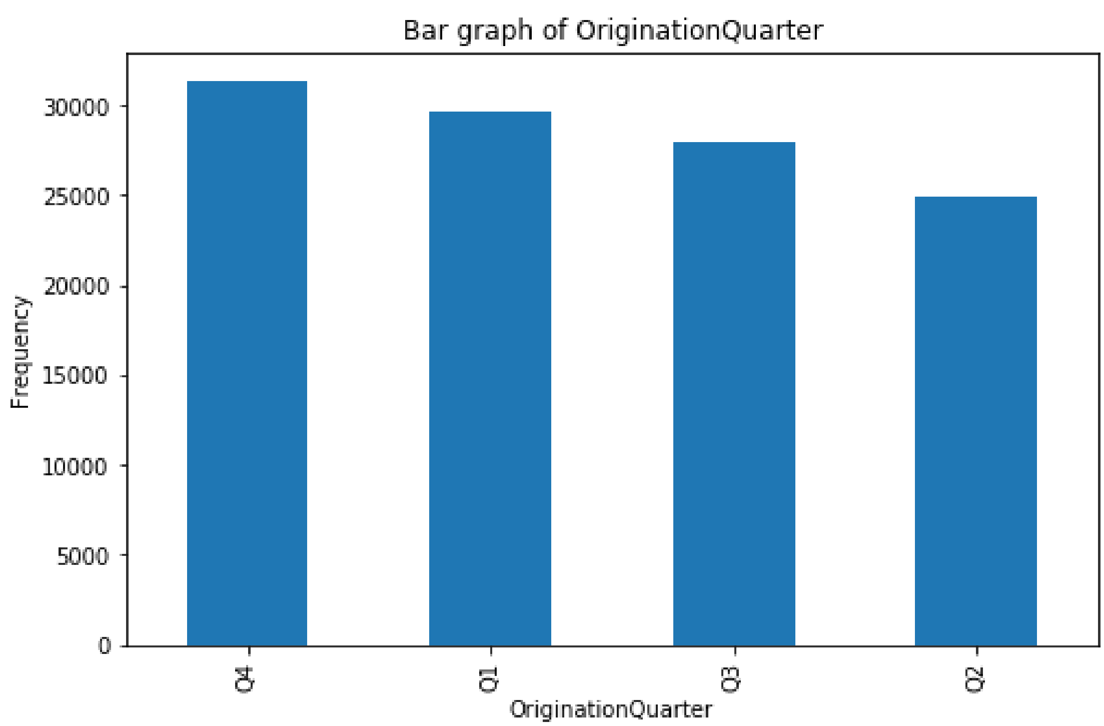


The number of borrowers who are homeowners and who are non-homeowners is equally distributed

```
In [16]: # Looking at the distribution for income verifiable
         plot_bar(x = 'IncomeVerifiable')
```

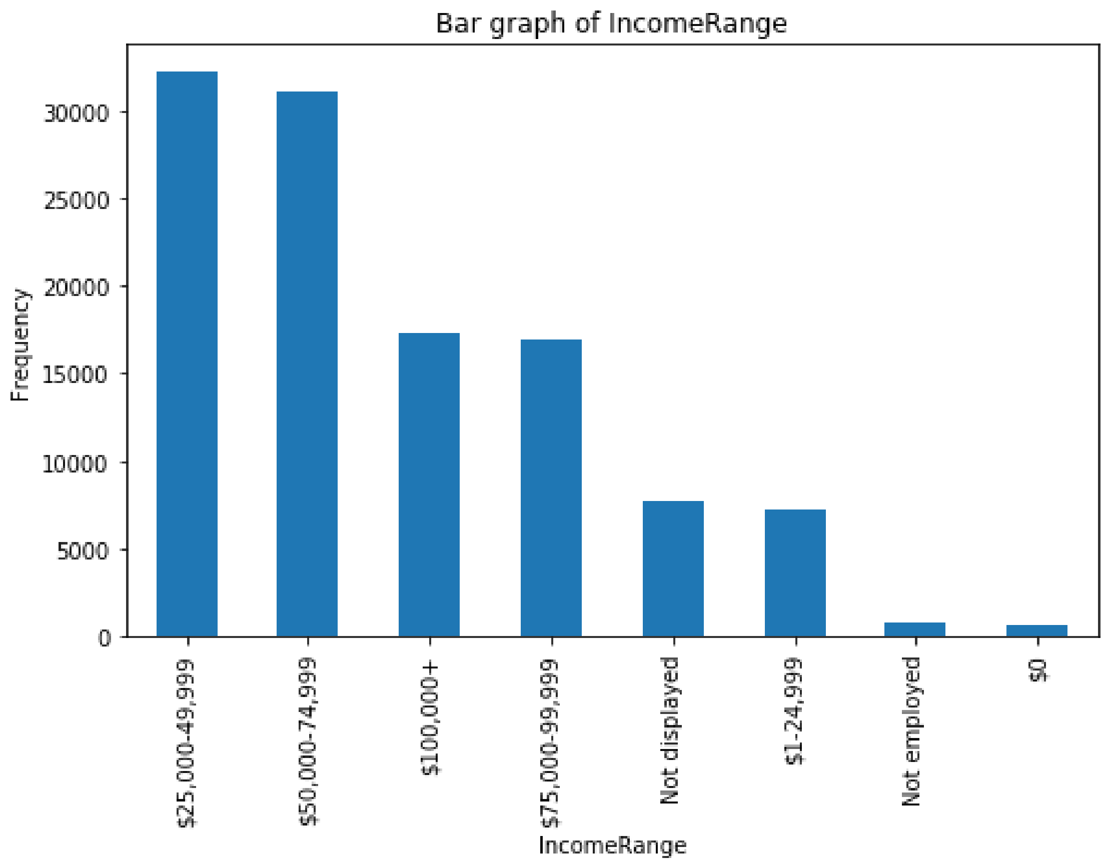


In [17]:
```
# Looking at the distribution for quarter
plot_bar(x = 'OriginationQuarter')
```



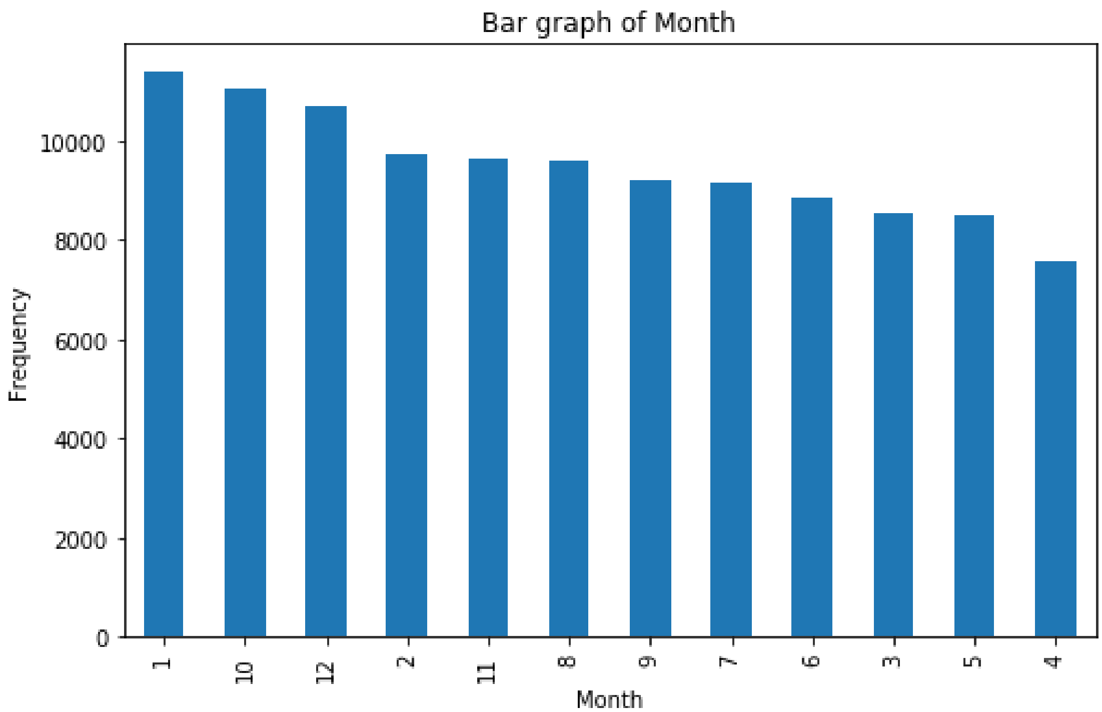
Most people borrow money in the last quarter as compared to all the other quarters and the second quarter has the least number of people who borrow money

```
In [18]: # Looking at the distribution for quarter
         plot_bar(x = 'IncomeRange')
```



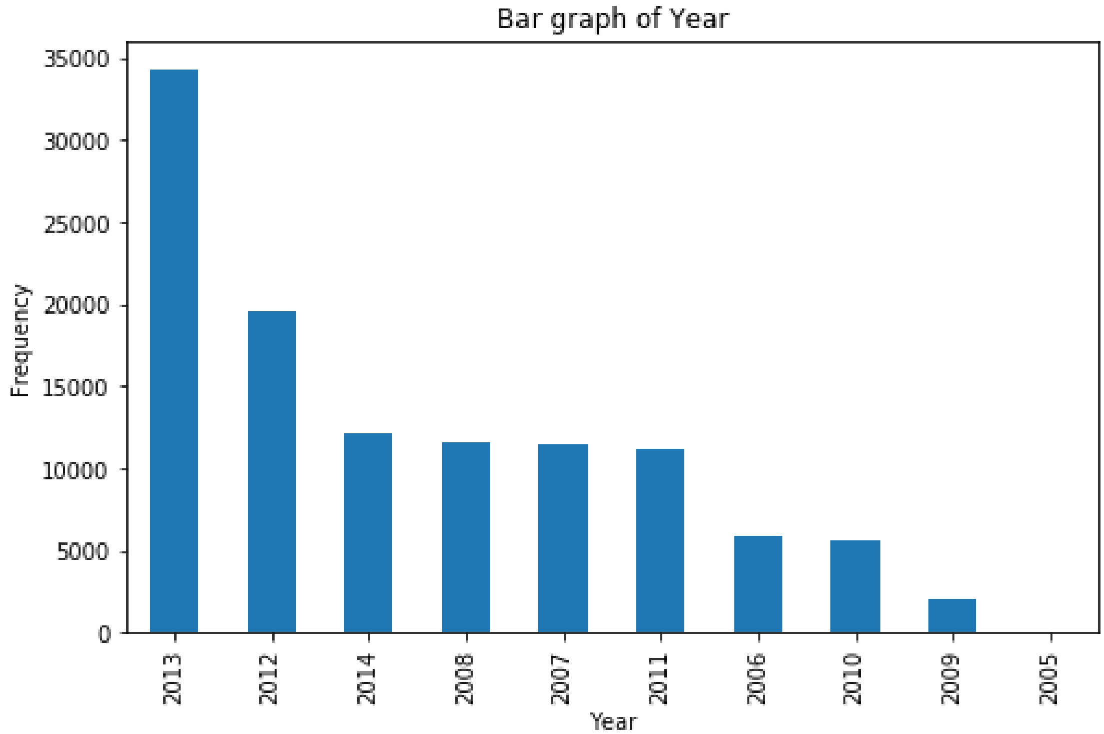
It can be seen that the higher the income a person gets also gives them a better chance of being granted a loan

```
In [19]: # Extracting the month from the date column
         loan_data['Month'] = loan_data['LoanOriginationDate'].dt.month

         # Looking at the distribution for month
         plot_bar(x = 'Month')
```



January is the month when most people borrow money followed by october then december and april is the month with the least number of people who borrow money

```
In [20]: # Extracting the year from the date column
         loan_data['Year'] = loan_data['LoanOriginationDate'].dt.year

         # Looking at the distribution for year
         plot_bar(x = 'Year')
```



2013 has the most number of people who got a loan, followed by 2012 and 2014 respectively. 2005 got the least number number of people who borrowed money

The bar graph above shows that if the income is verifiable the person is more likely to get a loan.

```
In [ ]:
```

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

It was interesting to note that most of the original loan amounts were not very high which makes it easier for the borrower to return the money since when the loan amount becomes bigger it's harder for the borrower to return it. Many loans are issued in the last and first quarter, in the last quarter it is because of the festive season and in the first quarter it is because people would have spent too much money during the festive season and need to borrow money. As your monthly income becomes higher the more likely you are to be given a loan.

1.4.2 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?

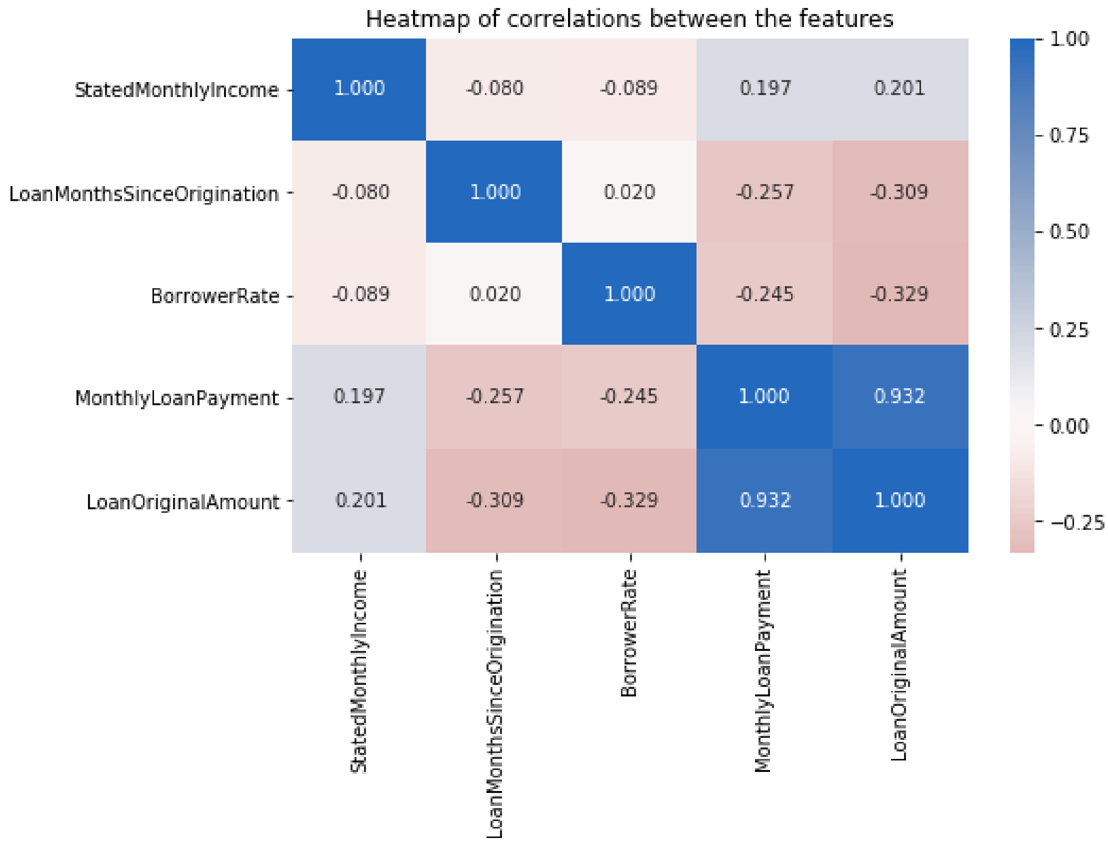
It was quite unsual that the term with the highest number of loans was 36 months since it would be more logical to give out more short term loans like a term of 12 months. These short term loans ensure that the money is returned in no time and this money can be loaned out to other people. It was also surprising to note that the number of homeowners and non-homeowners who got a loan was roughly the same, it would have been more sensible for people who are not homeowners to be more than homeowners. The loan origination date feature was converted from object type to datetime and features like is borrower homeowner, listing category, income verifiable and loan status were converted to categorical type. Other features such as month and year were created to aid with the exploration.

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).

To start off with, I want to look at the pairwise correlations present between features in the data.

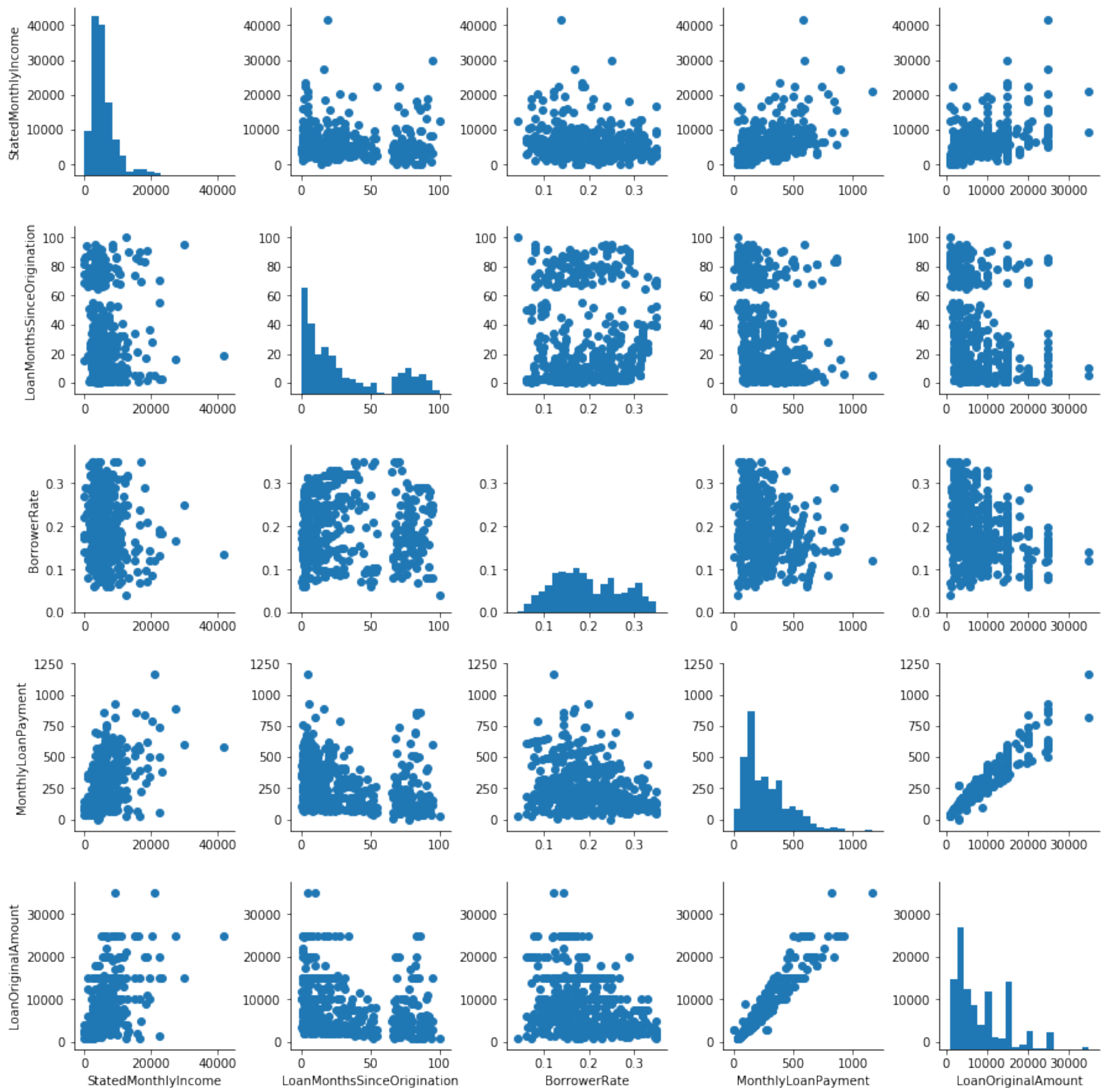
```
In [21]: num_vars = ['StatedMonthlyIncome', 'LoanMonthsSinceOrigination', 'BorrowerRate', 'Month
                     'LoanOriginalAmount']

         # correlation plot
         plt.figure(figsize = [8, 5])
         sb.heatmap(loan_data[num_vars].corr(), annot = True, fmt = '.3f',
                    cmap = 'vlag_r', center = 0)
         plt.title('Heatmap of correlations between the features')
         plt.show()
```



Monthly loan payment is very highly correlated with the original loan amount which makes sense since if the original loan amount is high then the money that is paid back has to be high as well. The other variables such as borrower rate, loan months since origination and stated monthly income have a weak correlation with the variable of interest which is original loan amount.

Checking the pairwise relationship between the variables using a scatter plot

```
In [22]: # plot matrix: sample 500 loans so that plots are clearer and they render faster
         loan_sample = loan_data.sample(n=500, replace=False)
         g = sb.PairGrid(data = loan_sample, vars = num_vars)
         g = g.map_diag(plt.hist, bins = 20);
         g.map_offdiag(plt.scatter)
         plt.show();
```



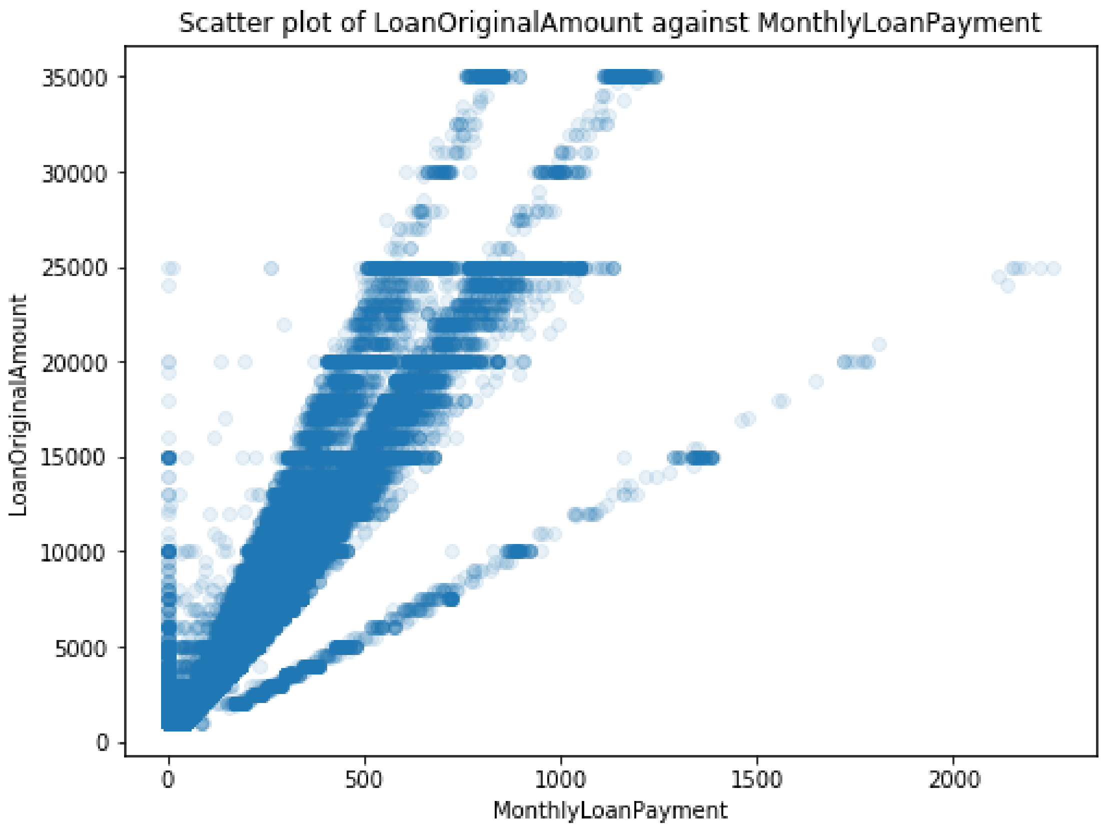
In the plots shown above the only two variables which have a strong relationship between them are original loan amount and monthly loan payment whose relationship looks linear hence the reason why the correlation between the two variables was high. All the other variables show a week relationship between each other hence the low correlation between the variables.

Checking the relationship between monthly loan payment and original loan amount since the correlation between the variables was high

In [23]:
```
# Function used to plot scatter plots of original loan amount with the other variables
def plot_scatter(x, y):
    plt.figure(figsize = [8, 6])
    plt.scatter(data = loan_data, x = x, y = y, alpha = 1/10)
    plt.title('Scatter plot of {} against {}'.format(y, x))
    plt.xlabel(x)
    plt.ylabel(y)
```

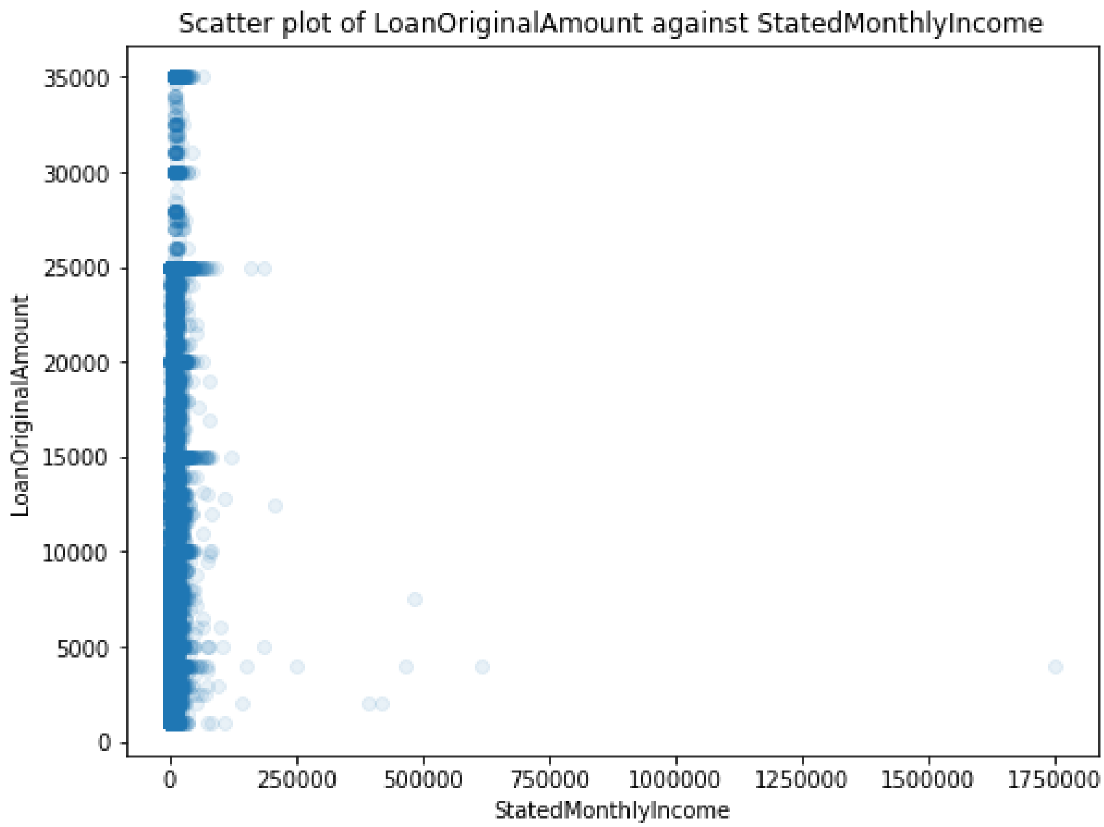
```
plt.show()
```

```
In [24]: # scatter plot of original loan amount vs. monthly loan payment
         plot_scatter(x = 'MonthlyLoanPayment', y = 'LoanOriginalAmount')
```



In general as the monthly loan amount increases the original loan amount also increases
Checking the relationship between stated monthly income and original loan amount

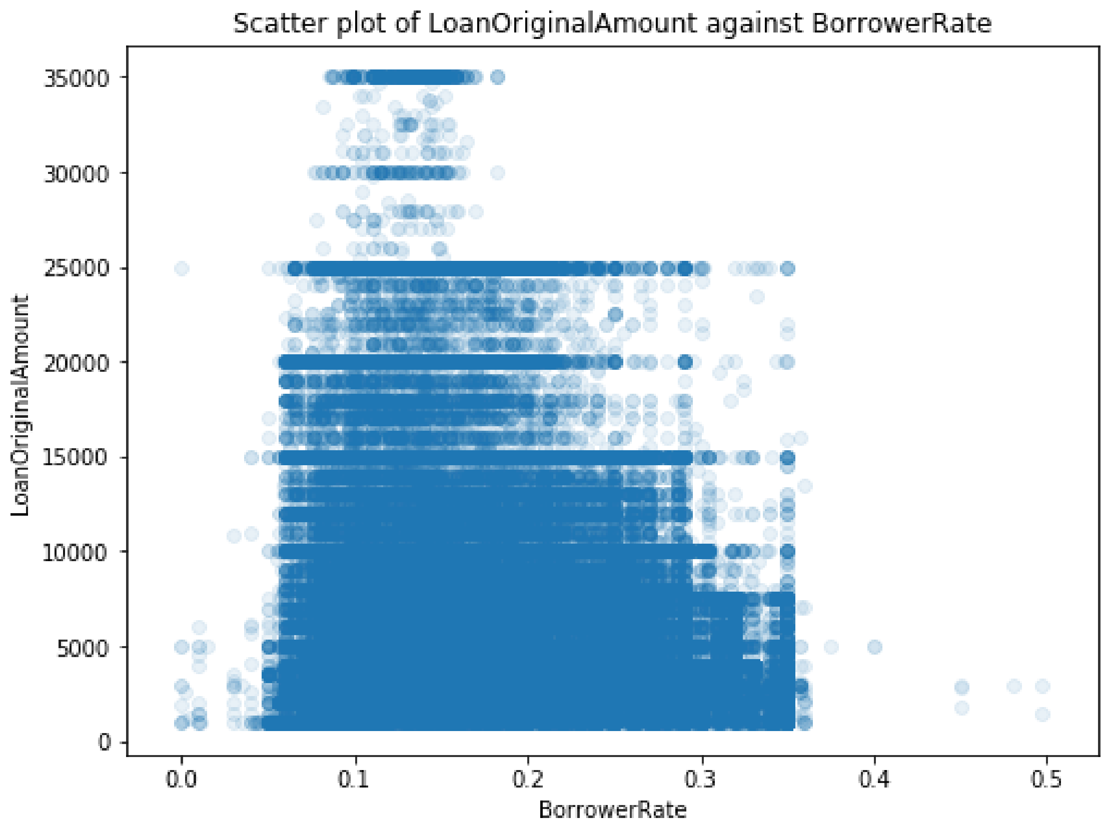
```
In [25]: # scatter plot of original loan amount vs. stated monthly income
         plot_scatter(x = 'StatedMonthlyIncome', y = 'LoanOriginalAmount')
```



People with low stated monthly income tend to be the ones that borrow more as compared to people with high stated monthly income

Checking the relationship between borrower rate and original loan amount

In [26]:
```
# scatter plot of original loan amount vs. borrower rate
plot_scatter(x = 'BorrowerRate', y = 'LoanOriginalAmount')
```

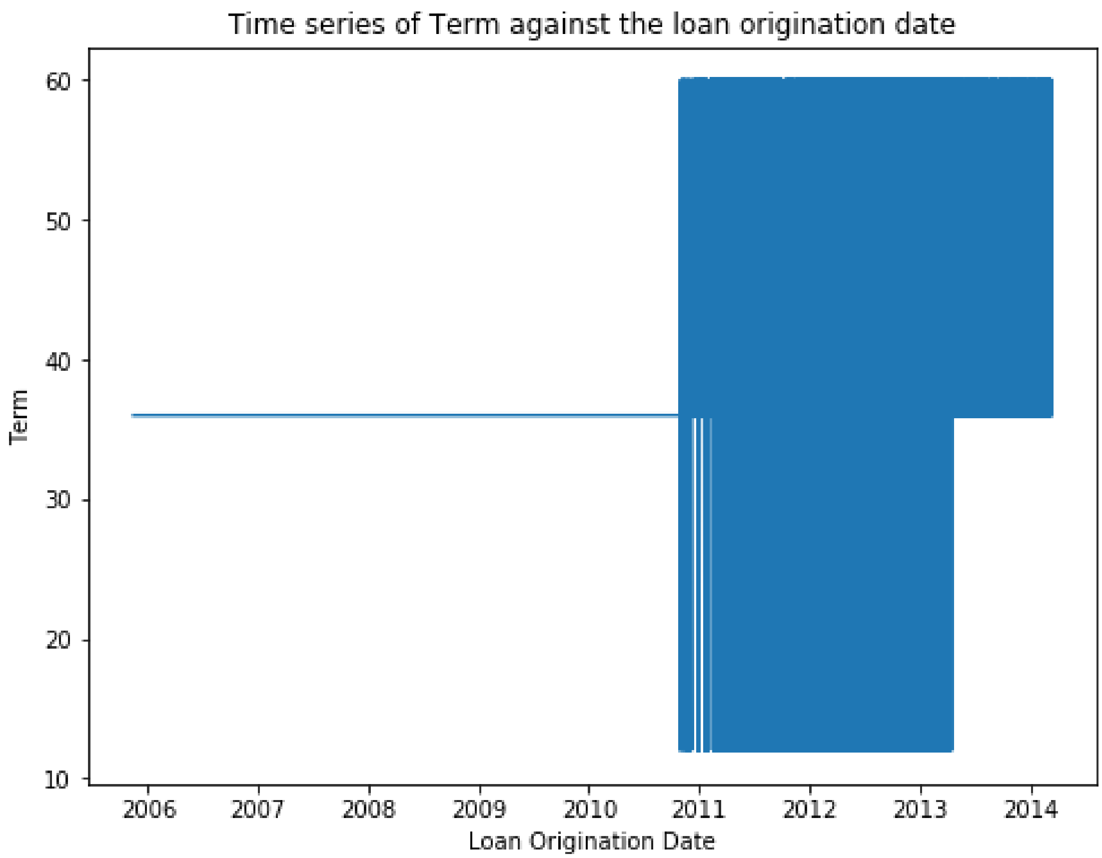


For lower original loan amounts the borrower rate has the greatest variation but as the amount increases the rate tends to be lower. Generally as the original loan amount increases the borrower rate decreases this is why borrower rate and original loan amount have a negative correlation.

Checking how variables changed over time

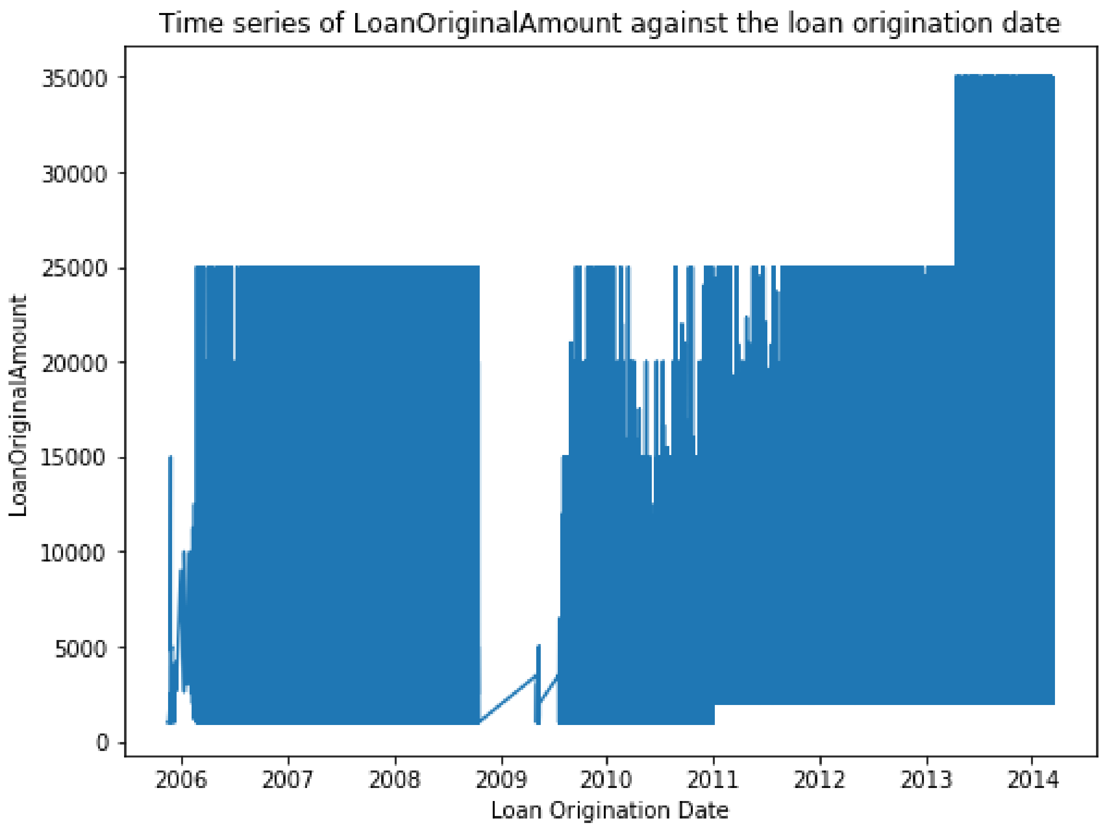
```
In [27]: # Sorting the dataframe by loan origination date
         loan_sort = loan_data.sort_values(by='LoanOriginationDate')

         def time_series(y):
             plt.figure(figsize = [8, 6])
             plt.plot(loan_sort['LoanOriginationDate'], loan_sort[y])
             plt.title('Time series of {} against the loan origination date'.format(y))
             plt.xlabel('Loan Origination Date')
             plt.ylabel(y)
             plt.show()
```

```
In [28]: # Time series for term
         time_series(y = 'Term')
```



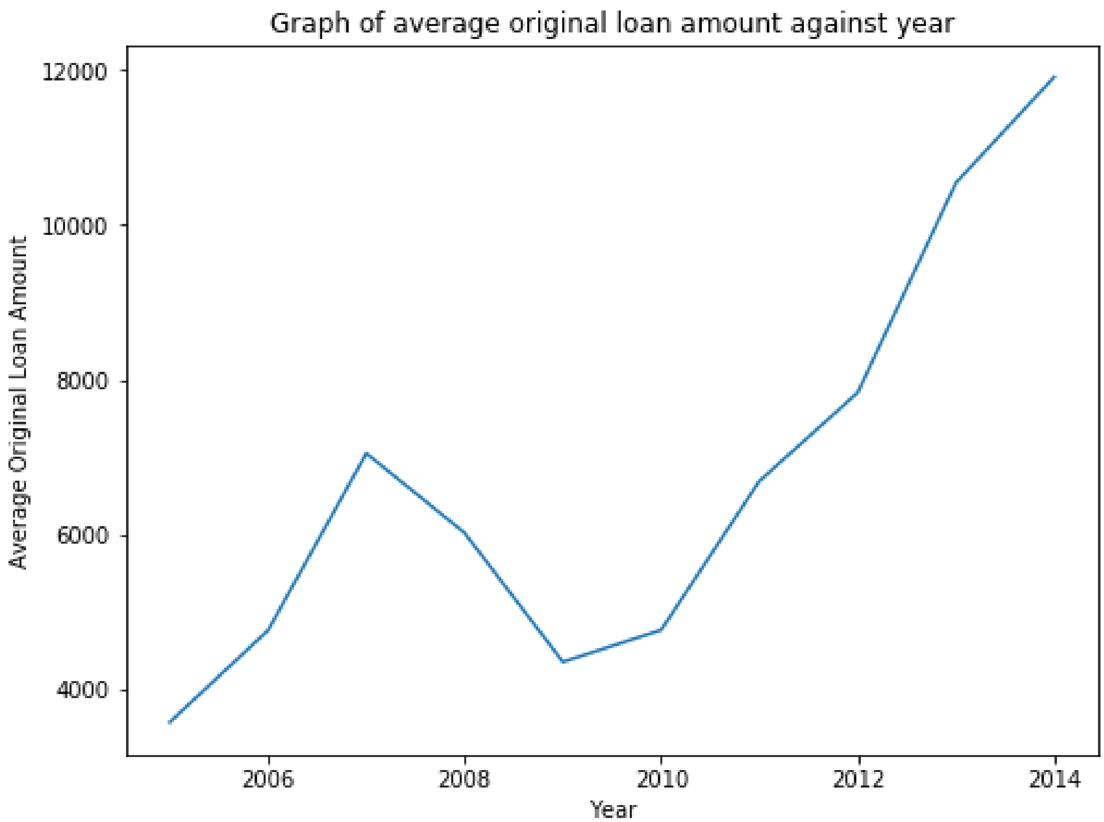
From 2006 to the third quarter of 2010, the loan would last 36 months and from the third quarter of 2010 to the first quarter of 2013 the loan term would vary amongst 12, 36 and 60 months and lastly from the first quarter of 2013 to 2014 the term varied between 36 and 60 months.

```
In [29]: # Time series for original loan amount
         time_series(y='LoanOriginalAmount')
```



There is no pattern that can noticed form this plot as the trendline is just moving up and down Checking further the trend for the average original loan amount per year

In [30]:
```
plt.figure(figsize = [8, 6])
loan_data.groupby('Year').mean()['LoanOriginalAmount'].plot()
plt.title('Graph of average original loan amount against year')
plt.xlabel('Year')
plt.ylabel('Average Original Loan Amount')
plt.show()
```

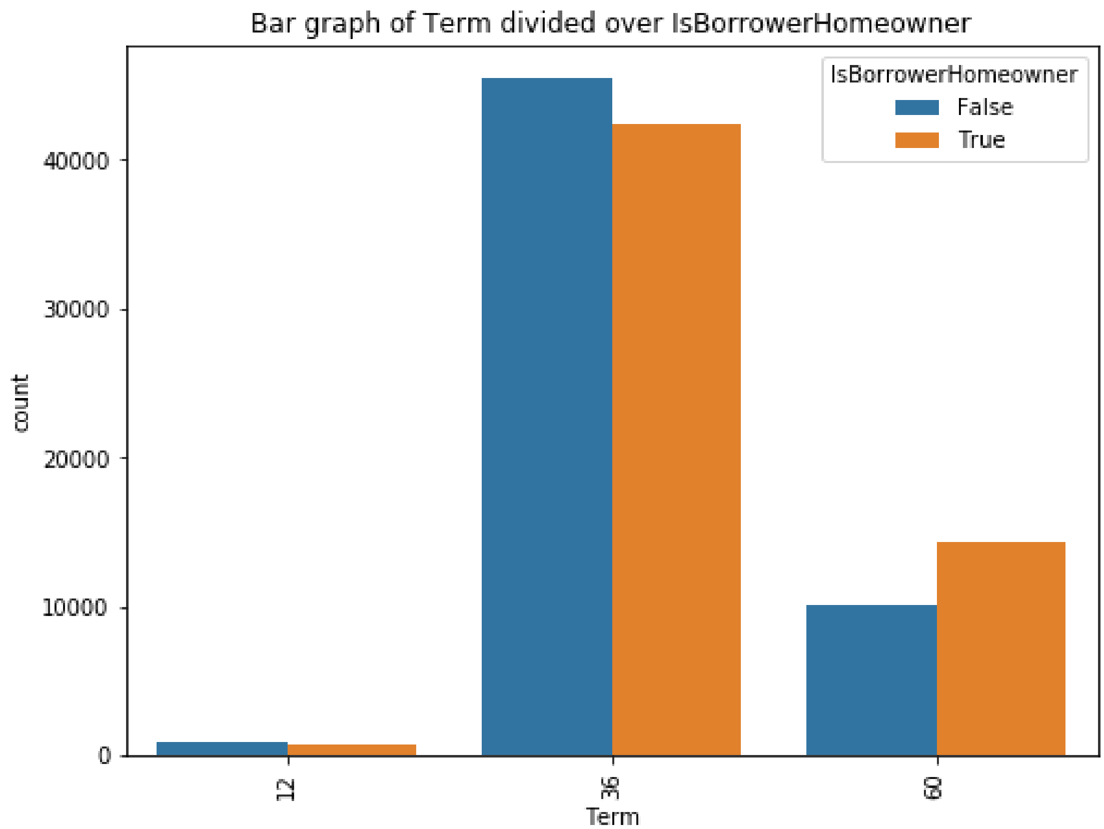


The general trend is that the average original loan amount that is borrowed increases as the years progress

Checking whether being a homeowner helps in getting a long term to pay back the loan

```
In [31]: # Function to plot 2d bar graph
         def plot_bar_2d(x, y):
             plt.figure(figsize = [8, 6])
             sb.countplot(data = loan_data, x = x, hue = y)
             plt.title('Bar graph of {} divided over {}'.format(x,y))
             plt.xticks(rotation=90)
             plt.show()
```

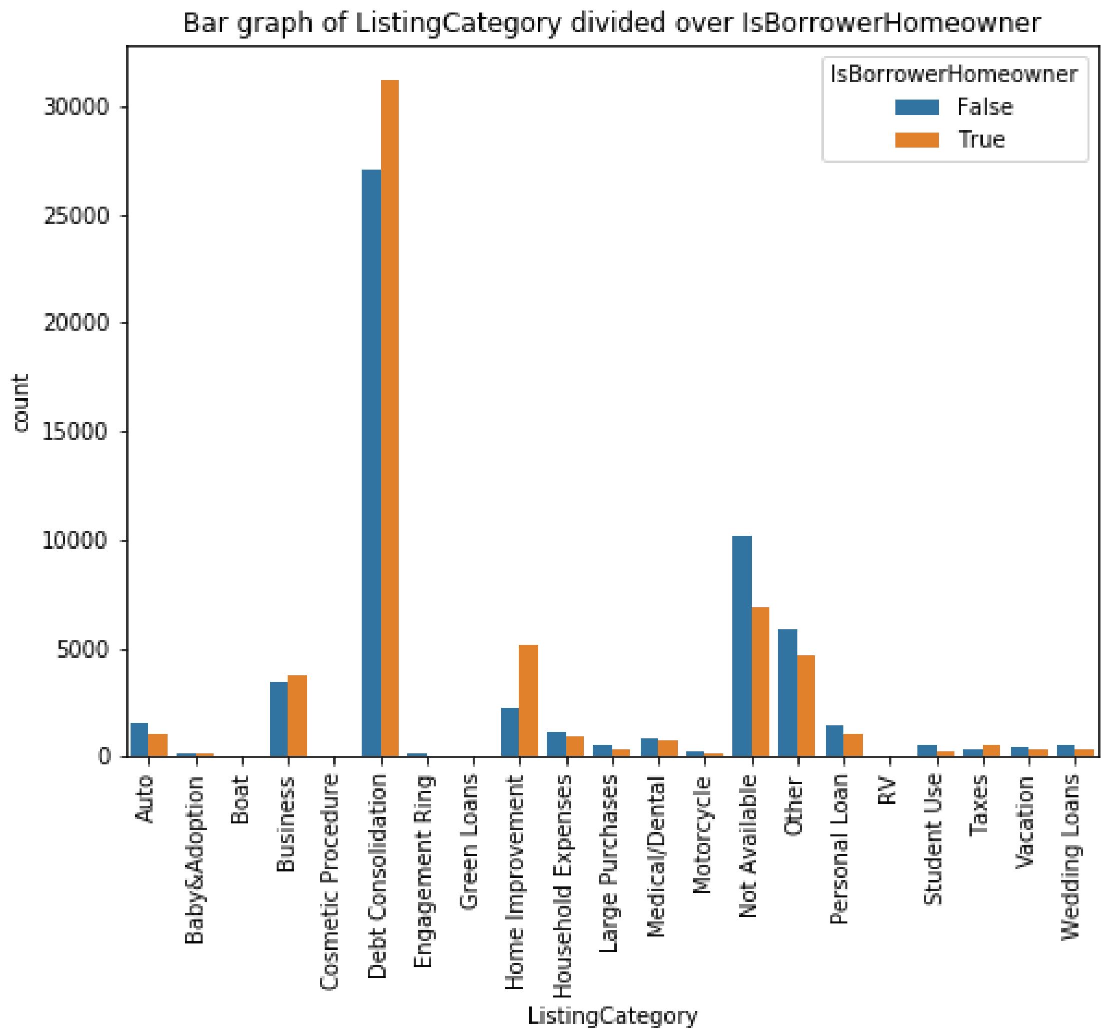
```
In [32]: # Plotting a bar graph of term divided over is borrower owner
         plot_bar_2d(x = 'Term', y = 'IsBorrowerHomeowner')
```



People who are not homeowners tend to be given shorter terms to pay back the loan as compared to people who are homeowners

Checking whether homeowners are the ones that get loans for luxury such as buying a boat

```
In [33]: # Plotting a bar graph of term divided over is borrower owner
         plot_bar_2d(x = 'ListingCategory', y = 'IsBorrowerHomeowner')
```



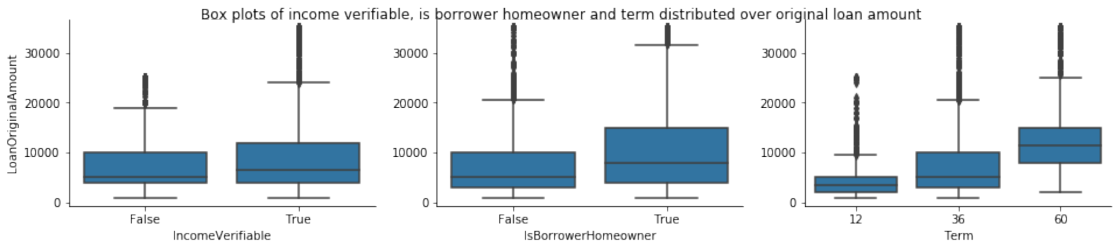
The pattern is not clear since sometimes homeowners are more than in certain listing category and vice-versa

```
In [34]: # Fucntion to create a box plot
         def box_plot(x,y, **kwargs):
             base_color = sb.color_palette()[0]
             sb.boxplot(data = loan_data, x = x, y = y,
                                 color = base_color)
```

Checking the distribution of original loan amount for variables such as income verifiable, is borrower homeowner and term

```
In [35]: categoric_vars = ['IncomeVerifiable', 'IsBorrowerHomeowner', 'Term']

         g = sb.PairGrid(data = loan_data, y_vars = ['LoanOriginalAmount'], x_vars = categoric_v
                         size = 3, aspect = 1.5)
```

```
g.map(box_plot)
plt.suptitle('Box plots of income verifiable, is borrower homeowner and term distribute
plt.tight_layout()
plt.show();
```



As can be seen in the figure above, the original loan amount generally increases if the income is verifiable, the borrower is a homeowner and the term to payback the loan is longer.

```
In [ ]:
```

1.5.1 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?

For the features that are numeric, only original loan amount and monthly loan payment had a strong relationship with each other. This was confirmed by plotting scatter plots of all the features with each other using 500 samples to make it more legible. Other features such as stated monthly income, loan months since origination and borrower rate did not show any pattern between between each other. Only monthly loan payment and original loan amount had a linear relationship. It was noted that the average original loan amount increased as the years progressed. It was also noted that the original loan amount increases as the borrower has a verifiable income, is a homeowner and needs a longer term to pay the loan.

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

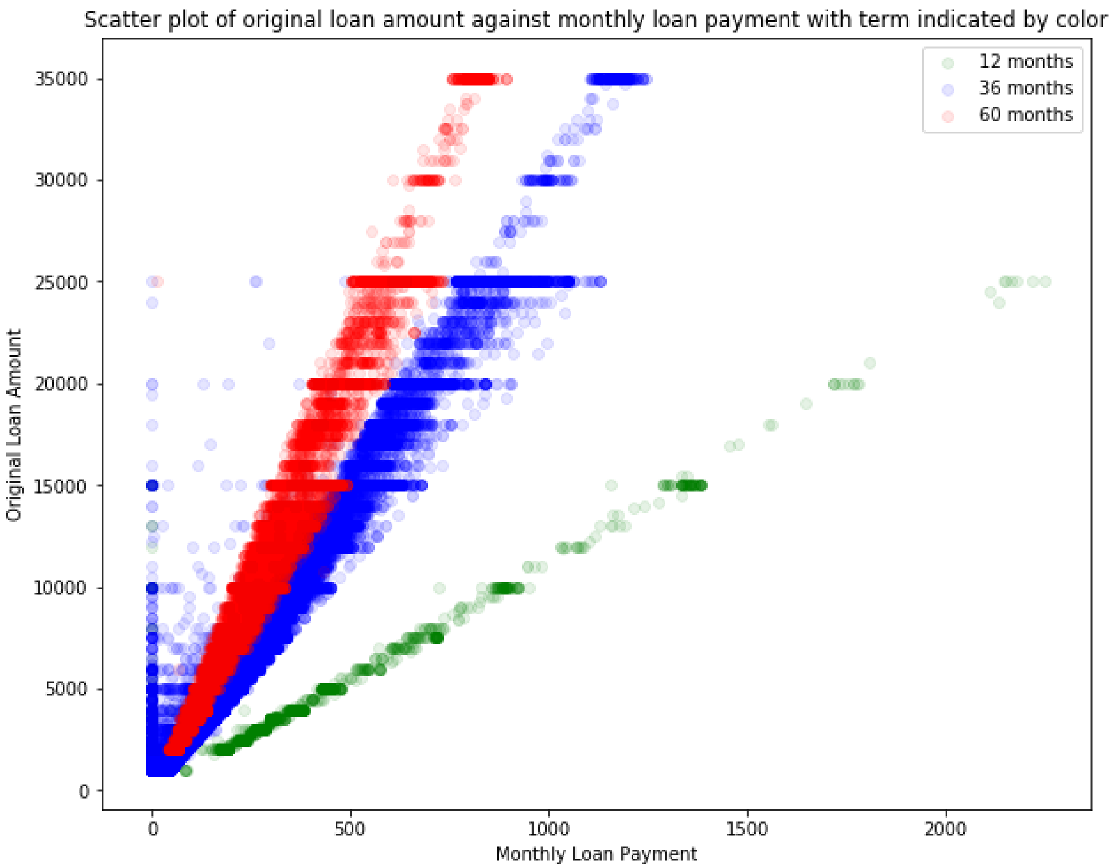
It was noted that the term for the loan was fixed at 36 months from 2006 to the last quarter of 2010 where the term started flactuating amongst 12, 36 and 60 months until the first quarter of 2013 then it flactuated between 36 and 60 months until 2014. It was noted that 36 months is the most common loan term but no relationship could be established between the loan term and whether the borrower is a homeowner. Most of the loans are for debt consolidation followed by not available and RV is the listing category with the least number of loans and also there is no relationship between listing category and whether the borrower is a home owner.

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.

Checking for the relationship between original loan amount, monthly loan payment and term

```
In [36]: term_col = [[12, 'g'], [36, 'b'], [60, 'r']]
         plt.figure(figsize = [10, 8])
         for term, col in term_col:
             df = loan_data[loan_data['Term']==term]
             plt.scatter(df['MonthlyLoanPayment'], df['LoanOriginalAmount'], alpha = 1/10, color
         plt.title('Scatter plot of original loan amount against monthly loan payment with term
         plt.xlabel('Monthly Loan Payment')
         plt.ylabel('Original Loan Amount')
         plt.legend(['12 months', '36 months', '60 months'])
         plt.show()
```

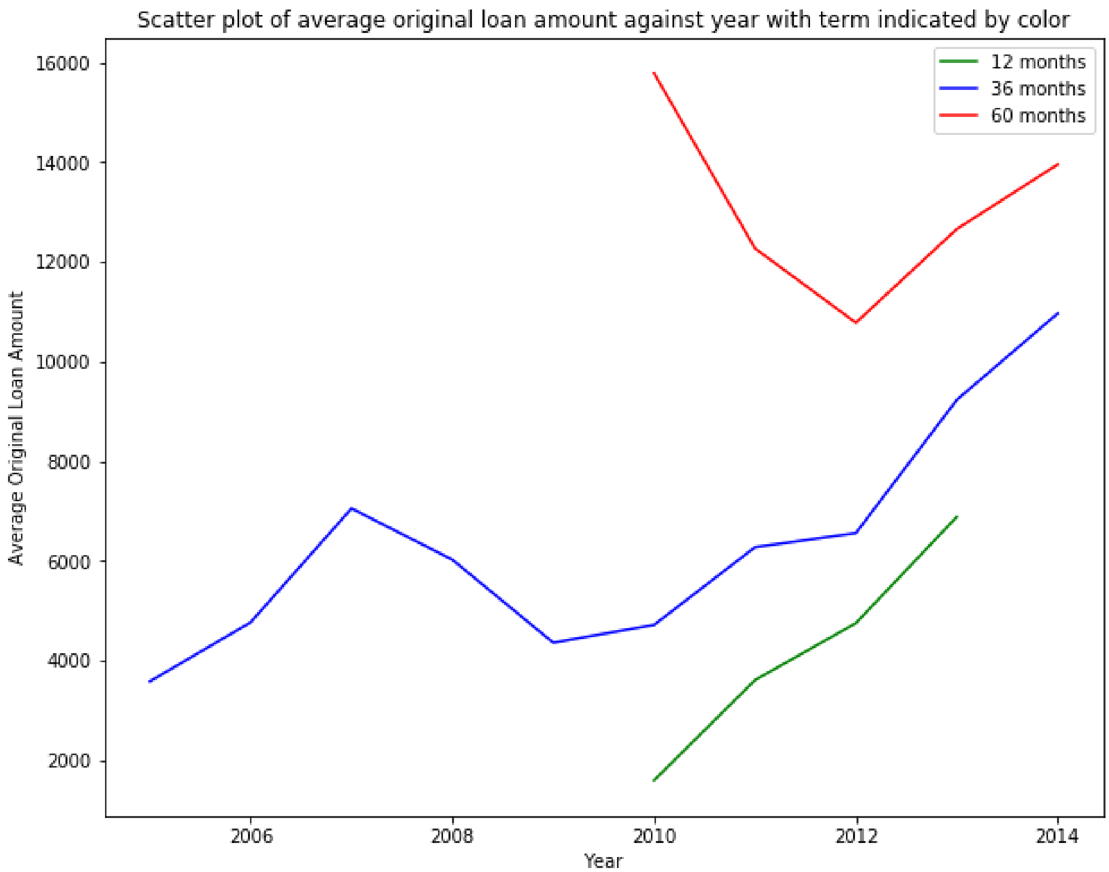


It can be seen in the scatter plot above that the relationship between original loan amount and monthly loan payment is linear and the reason why it looks as if there are 3 lines in the figure is because each of these points clustered close together represent a different term which can be seen from the different colors in the figure.

Checking for the relationship between average original loan amount, year and term

```
In [37]: term_col = [[12, 'g'], [36, 'b'], [60, 'r']]
         plt.figure(figsize = [10, 8])
```

```
for term, col in (term_col):
    plt.plot(loan_data.groupby(['Term','Year']).mean()['LoanOriginalAmount'][term], col
plt.title('Scatter plot of average original loan amount against year with term indicate
plt.xlabel('Year')
plt.ylabel('Average Original Loan Amount')
plt.legend(['12 months', '36 months', '60 months'])
plt.show()
```



In the plot above it can be seen that the general trend for the term of 12 and 36 months is increasing whilst for the term of 60 months it first decreases then it increases. The term of 60 months has the highest average original loan amount, followed by the term of 36 months and lastly the term of 12 months for the years where all 3 terms have values.

1.6.1 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?

Original loan amount and monthly loan payment have a very strong relationship and by conducting mutlivariate exploration it was discovered that the scatter plot can be divided into 3 sections one for the term of 12 months, another one for the term of 36 months and the last one for the term of 60 months. It can also be seen from the gradient of these 3 sections that if the term is longer then the amount paid per month is lower is there is more time to pay. By plotting the average original loan amount against year and separating the plots by term it was discovered that the term of 60 months always has the highest average original loan amount followed by the term of 36 months and lastly the term of 12 months.

1.6.2 Were there any interesting or surprising interactions between features?

It was very interesting to note that the scatter plot of original loan amount and monthly loan payment could be divided into sections by coloring the datapoints using the term feature. It made it easier to understand that a person with a longer term pays less money per month as compared to someone whose term is shorter. It was also interesting to note that the average original loan amount for a longer term is always higher than that for a short term.

1.7 Conclusions

It was discovered that original loan amount and monthly loan payment have a strong relationship. It was further discovered that this relationship is even stronger if only one term is used for each plot since the scatter plot is split into 3 sections using the term represented by different colors. The other numerical features did not have a strong relationship with original loan amount which is the feature of interest as other features. It was also observed that the original loan amount generally increases if the income is verifiable, the borrower is a homeowner and the term to payback the loan is longer.

The first step was to make sure the data is tidy and not dirty and cleaning any unclean data. The second step in the data exploration was to choose the features to use for the exploration and going on further to choose the feature of interest which is original loan amount in this case. The next step was to come up with a question, visualization and observation for each visualization in the univariate exploration, then a summary was written for all the visualizations. This step was repeated for bivariate and multivariate exploration and finally a conclusion was written for the exploration.

```
In [ ]:
```