An Exploratory Analysis of Loan Data from Prosper

by Eyong, Wofai Alfred

Introduction

> Prosper Marketplace, Inc. is a San Francisco, California-based company in the peer-to-peer lending industry. Its subsidiary operates Prosper.com; a website where individuals can either invest in personal loans(peer-to-peer) or request to borrow money. Though founded in 2005 with pioneer status, it has facilitated over 21 billion dollars in loans to over 1.3M people. Prosper provides unsecure loans(Loans which do not require collateral) The prosper loan dataset contains information on several variables representing about 113,937 loans collected from the company. It includes customers who have collected loans, customers who have cleared off pending loans as well as customers who have defaulted on their loan repayment schedules. From the original dataset which contains 113937 rows and 81 columns, i analysed 14 variables.

The 14 parameters analyzed include:

LoanOriginalAmount: The origination amount of the loan.

LenderYield: The Lender yield on the loan. Lender yield is equal to the interest rate on the loan less the servicing fee.

LoanStatus: The current status of the loan: Cancelled, Chargedoff, Completed, Current, Defaulted, FinalPaymentInProgress, PastDue.

IncomeRange: The income range of the borrower at the time the listing was created.

LoanOriginationDate: The date the loan was originated.

CreditScoreRangeLower: The lower value representing the range of the borrower's credit score as provided by a consumer credit rating agency.

CreditScoreRangeUpper: The upper value representing the range of the borrower's credit score as provided by a consumer credit rating agency.

Term: The length of the loan expressed in months.

ProsperRating (numeric): The Prosper Rating assigned at the time the listing was created: 0 - N/A, 1 - HR, 2 - E, 3 - D, 4 - C, 5 - B, 6 - A, 7 - AA. Applicable for loans originated after July 2009. Its rating relevance is arranged in ascending order, with 1 being high risk and 7 being the best score.

ListingCategory: The category of the listing that the borrower selected when posting their listing: 0 - Not Available, 1 - Debt Consolidation, 2 - Home Improvement, 3 - Business, 4 - Personal Loan, 5 - Student Use, 6 - Auto, 7- Other, 8 - Baby&Adoption, 9 - Boat, 10 - Cosmetic Procedure, 11 - Engagement Ring, 12 - Green Loans, 13 - Household Expenses, 14 - Large Purchases, 15 - Medical/Dental, 16 - Motorcycle, 17 - RV, 18 - Taxes, 19 - Vacation, 20 - Wedding Loans

EmploymentStatus: The employment status of the borrower at the time they posted the listing.

IncomeVerifiable: The borrower indicated they have the required documentation to support their income.

IsBorrowerHomeowner: A Borrower will be classified as a homowner if they have a mortgage on their credit profile or provide documentation confirming they are a homeowner.

StatedMonthlyIncome: The monthly income the borrower stated at the time the listing was created.

Preliminary Wrangling

In [2]:
```
# 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
```

In [3]:
```
#Load dataset
loan = pd.read_csv('prosperLoanData.csv')
```

In [4]:
```
#preview the dataset
loan.head()
```

Out[4]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	LoanStatus	ClosedDate	Bo
0	1021339766868145413AB3B	193129	2007-08-26 19:09:29.263000000	C	36	Completed	2009-08-14 00:00:00	
1	1027360249950330B223C1	1209647	2014-02-27 08:28:07.900000000	NaN	36	Current	NaN	
2	0EE9337825851032864889A	81716	2007-01-05 15:00:47.090000000	HR	36	Completed	2009-12-17 00:00:00	
3	0EF5356002482715299901A	658116	2012-10-22 11:02:35.010000000	NaN	36	Current	NaN	
4	0F023589499656230C5E3E2	909464	2013-09-14 18:38:39.097000000	NaN	36	Current	NaN	

5 rows × 81 columns

In [5]:
```
#lets get the shape of the dataset
loan.shape
```

Out[5]:
```
(113937, 81)
```

What is the structure of your dataset?

> The Prosper loan dataset has 113,937 sample entries in this mother dataset with 81 distinct attributes or characteristics. However, i analysed 14 out of the 81 attributes.

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

> My major feature of interest is the Lender Yield. This is because my aim is to identify features of an individual which an investor should consider important when choosing a prosper loan

to fund. These features will also help the Prosper Company minimise the occurence of high risk loans which doesnt benefit investors and the Prosper company as a whole.

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

I am interested in identifying charateristics which will help the Prosper company investors chose the best types of clients for its loan investments, as well as individuals with potential. It is important to invest in customers who do not default on their loan payments for maximum individual profits and an overall company growth. However, as i do not have data on payments already made, i will focus on what is to be expected based on the presence or absense of certain characteristics such as an income, house ownership, employment, credit scores, prosper ratings etc.

Data Accessment

In [6]:
```
#Ensure no columns are hidden
pd.set_option('display.max_columns', None)
```

In [7]:
```
#Access a random sample of the dataset
loan.sample(10)
```

Out[7]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	LoanStatus	ClosedDat
50083	38D734300935780275ED80A	386919	2008-08-22 18:02:07.177000000	C	36	Completed	2011-02-0 00:00:0
11842	95AA3542772728375E2C7B7	570130	2012-03-19 19:42:30.520000000	NaN	36	Current	Na
24776	45253400644927084FB965E	204326	2007-09-20 12:01:06.660000000	B	36	Chargedoff	2009-06-2 00:00:0
11985	05113589569985890E2C096	919013	2013-09-23 10:51:46.290000000	NaN	36	Completed	2014-01-2 00:00:0
20550	C7C93593706499094C4AF17	993358	2013-11-13 17:11:33.760000000	NaN	60	Current	Na
21546	B0C33418744067620E7B4B5	310181	2008-04-14 10:39:12.927000000	C	36	Completed	2010-03-0 00:00:0
52625	56E535559633075564C2D2C	623307	2012-08-12 19:53:00.703000000	NaN	36	Current	Na
65082	6DDD34757937184914CBA63	446709	2010-02-16 10:56:16.427000000	NaN	36	Completed	2011-03-1 00:00:0
46471	5232359639498346207F54E	1082315	2013-12-16 07:34:19.097000000	NaN	60	Current	Na
38416	6A283535233734345469378	547167	2011-12-28 07:19:26.600000000	NaN	60	Chargedoff	2013-11-2 00:00:0

In [8]:
```
#Check column information to identify attributes i want to work on
loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 81 columns):
 #   Column                                Non-Null Count   Dtype
---  ------                                --------------   -----
 0   ListingKey                            113937 non-null  object
 1   ListingNumber                         113937 non-null  int64
 2   ListingCreationDate                   113937 non-null  object
 3   CreditGrade                           28953 non-null   object
 4   Term                                  113937 non-null  int64
 5   LoanStatus                            113937 non-null  object
 6   ClosedDate                            55089 non-null   object
 7   BorrowerAPR                           113912 non-null  float64
 8   BorrowerRate                          113937 non-null  float64
 9   LenderYield                           113937 non-null  float64
 10  EstimatedEffectiveYield               84853 non-null   float64
 11  EstimatedLoss                         84853 non-null   float64
 12  EstimatedReturn                       84853 non-null   float64
 13  ProsperRating (numeric)               84853 non-null   float64
 14  ProsperRating (Alpha)                 84853 non-null   object
 15  ProsperScore                          84853 non-null   float64
 16  ListingCategory (numeric)             113937 non-null  int64
 17  BorrowerState                         108422 non-null  object
 18  Occupation                            110349 non-null  object
 19  EmploymentStatus                      111682 non-null  object
 20  EmploymentStatusDuration              106312 non-null  float64
 21  IsBorrowerHomeowner                   113937 non-null  bool
 22  CurrentlyInGroup                      113937 non-null  bool
 23  GroupKey                              13341 non-null   object
 24  DateCreditPulled                      113937 non-null  object
 25  CreditScoreRangeLower                 113346 non-null  float64
 26  CreditScoreRangeUpper                 113346 non-null  float64
 27  FirstRecordedCreditLine               113240 non-null  object
 28  CurrentCreditLines                    106333 non-null  float64
 29  OpenCreditLines                       106333 non-null  float64
 30  TotalCreditLinespast7years            113240 non-null  float64
 31  OpenRevolvingAccounts                 113937 non-null  int64
 32  OpenRevolvingMonthlyPayment           113937 non-null  float64
 33  InquiriesLast6Months                  113240 non-null  float64
 34  TotalInquiries                        112778 non-null  float64
 35  CurrentDelinquencies                  113240 non-null  float64
 36  AmountDelinquent                      106315 non-null  float64
 37  DelinquenciesLast7Years               112947 non-null  float64
 38  PublicRecordsLast10Years              113240 non-null  float64
 39  PublicRecordsLast12Months             106333 non-null  float64
 40  RevolvingCreditBalance                106333 non-null  float64
 41  BankcardUtilization                   106333 non-null  float64
 42  AvailableBankcardCredit               106393 non-null  float64
 43  TotalTrades                           106393 non-null  float64
 44  TradesNeverDelinquent (percentage)    106393 non-null  float64
 45  TradesOpenedLast6Months               106393 non-null  float64
 46  DebtToIncomeRatio                     105383 non-null  float64
 47  IncomeRange                           113937 non-null  object
 48  IncomeVerifiable                      113937 non-null  bool
 49  StatedMonthlyIncome                   113937 non-null  float64
 50  LoanKey                               113937 non-null  object
 51  TotalProsperLoans                     22085 non-null   float64
 52  TotalProsperPaymentsBilled            22085 non-null   float64
 53  OnTimeProsperPayments                 22085 non-null   float64
 54  ProsperPaymentsLessThanOneMonthLate   22085 non-null   float64
 55  ProsperPaymentsOneMonthPlusLate       22085 non-null   float64
 56  ProsperPrincipalBorrowed              22085 non-null   float64
 57  ProsperPrincipalOutstanding           22085 non-null   float64
 58  ScorexChangeAtTimeOfListing           18928 non-null   float64
 59  LoanCurrentDaysDelinquent             113937 non-null  int64
 60  LoanFirstDefaultedCycleNumber         16952 non-null   float64
```

```
 61  LoanMonthsSinceOrigination          113937 non-null  int64
 62  LoanNumber                          113937 non-null  int64
 63  LoanOriginalAmount                  113937 non-null  int64
 64  LoanOriginationDate                 113937 non-null  object
 65  LoanOriginationQuarter              113937 non-null  object
 66  MemberKey                           113937 non-null  object
 67  MonthlyLoanPayment                  113937 non-null  float64
 68  LP_CustomerPayments                 113937 non-null  float64
 69  LP_CustomerPrincipalPayments        113937 non-null  float64
 70  LP_InterestandFees                  113937 non-null  float64
 71  LP_ServiceFees                      113937 non-null  float64
 72  LP_CollectionFees                   113937 non-null  float64
 73  LP_GrossPrincipalLoss               113937 non-null  float64
 74  LP_NetPrincipalLoss                 113937 non-null  float64
 75  LP_NonPrincipalRecoverypayments     113937 non-null  float64
 76  PercentFunded                       113937 non-null  float64
 77  Recommendations                     113937 non-null  int64
 78  InvestmentFromFriendsCount          113937 non-null  int64
 79  InvestmentFromFriendsAmount         113937 non-null  float64
 80  Investors                           113937 non-null  int64
dtypes: bool(3), float64(50), int64(11), object(17)
memory usage: 68.1+ MB
```

In [9]:
```
#Summary statistics of dataset
loan.describe()
```

Out[9]:

	ListingNumber	Term	BorrowerAPR	BorrowerRate	LenderYield	EstimatedEffectiveYield	Estim
count	1.139370e+05	113937.000000	113912.000000	113937.000000	113937.000000	84853.000000	8485
mean	6.278857e+05	40.830248	0.218828	0.192764	0.182701	0.168661	
std	3.280762e+05	10.436212	0.080364	0.074818	0.074516	0.068467	
min	4.000000e+00	12.000000	0.006530	0.000000	-0.010000	-0.182700	
25%	4.009190e+05	36.000000	0.156290	0.134000	0.124200	0.115670	
50%	6.005540e+05	36.000000	0.209760	0.184000	0.173000	0.161500	
75%	8.926340e+05	36.000000	0.283810	0.250000	0.240000	0.224300	
max	1.255725e+06	60.000000	0.512290	0.497500	0.492500	0.319900	

Data Quality issues

> Datetime objects listed as strings (ListingCreationDate, ClosedDate, DateCreditPulled, FirstRecordedCreditLine, LoanOriginationDate)
>
> Categorical data columns are listed as objects or floats(LoanStatus, ProsperRating (numeric), ProsperRating (Alpha), EmploymentStatus, IncomeRange)
>
> StatedMonthlyIncome column values are floats, making it more difficult for my use case.
>
> Missing values

Data Tidiness issues

> Loan origination date is holding more than one data type (date and time)

Data Cleaning

In [10]:
```
loan_clean = loan.copy()
```

Define

Loan origination date column is holding more than one data type (date and time)

Split Loan origination date column to get date and time

Code

In [11]:
```
#Split column by space dividing date and time
loan_clean[['loan_origination_date','loan_origination_time']] = loan_clean['LoanOriginat
```

Test

In [12]:
```
#Check for the creation of new columns
loan_clean.head(2)
```

Out[12]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	LoanStatus	ClosedDate	Bo
0	10213397668681454 13AB3B	193129	2007-08-26 19:09:29.263000000	C	36	Completed	2009-08-14 00:00:00	
1	10273602499503308B223C1	1209647	2014-02-27 08:28:07.900000000	NaN	36	Current	NaN	

Define

Inaccurate datatype

Convert ListingCreationDate, ClosedDate, DateCreditPulled, FirstRecordedCreditLine, loan_origination_date and LoanOriginationDate columns to datetime datatype

Code

In [13]:
```
#Convert columns in date_columns list to Datetime using a function
date_columns = ['ListingCreationDate', 'ClosedDate', 'DateCreditPulled', 'FirstRecordedC
for cols in date_columns:
    loan_clean[cols] = loan_clean[cols].astype('datetime64')
```

Test

In [14]:
```
#Confirm new data types
loan_clean['ListingCreationDate'].dtype
loan_clean['DateCreditPulled'].dtype
loan_clean['LoanOriginationDate'].dtype
loan_clean['loan_origination_date'].dtype
```

Out[14]:
```
dtype('<M8[ns]')
```

In [15]:
```
np.dtype('datetime64[ns]') == np.dtype('<M8[ns]')
```

Out[15]:
```
True
```

Define

Categorical data columns are listed as objects or floats

Convert LoanStatus, ProsperRating (numeric), ProsperRating (Alpha), EmploymentStatus and IncomeRange columns to categorical data.

In [16]:
```
#Convert rows in cat_columns list to categorical data using a function
cat_columns = ['LoanStatus','ProsperRating (numeric)', 'ProsperRating (Alpha)', 'Employm
for cols in cat_columns:
    loan_clean[cols] = loan_clean[cols].astype('category')
```

Test

In [17]:
```
#Confirm new data type
print(loan_clean['LoanStatus'].dtype)
print(loan_clean['ProsperRating (Alpha)'].dtype)
print(loan_clean['IncomeRange'].dtype)
```

```
category
category
category
```

Define

StatedMonthlyIncome column values are floats

Round up StatedMonthlyIncome column to the nearest whole number

Code

In [18]:
```
#Round up StatedMonthlyIncome column
loan_clean['StatedMonthlyIncome'] = loan_clean['StatedMonthlyIncome'].round()
```

Test

In [19]:
```
#Check values in StatedMonthlyIncome column
loan_clean['StatedMonthlyIncome'].value_counts()
```

Out[19]:
```
4167.0     3530
5000.0     3391
3333.0     2925
3750.0     2430
5417.0     2376
           ...
11392.0       1
17232.0       1
3109.0        1
796.0         1
18756.0       1
Name: StatedMonthlyIncome, Length: 7921, dtype: int64
```

Define

Missing values

Columns with a significant number of missing values as they will not be used for this analysis

Univariate Exploration

Using the "Question-Visualization-Observations" framework, i explore individual data attributes to generate answers to questions, using exploratory data visuals.

In [20]:
```
#Set colour palette and style
sns.set_style('darkgrid')
sns.set(rc={"figure.figsize": (12, 8)})
colors = "#5A9"
set_color = sns.set_palette('Set2')
color='teal'
```

Defining functions to be used for analysis

In [21]:
```
#creating a function to plot histograms
def plot_histogram1(x, title, xlabel, ylabel, lower_limit, bns):
    '''This function plots single histograms'''

    #defining bins
    bins = np.arange(lower_limit, loan_clean[x].max()+bns, bns)

    #plot the histogram
    plt.hist( data = loan_clean, x = x,  bins = bins, color=colors)

    #display graph labels
    plt.xlabel(xlabel, fontsize=16)
    plt.ylabel(ylabel, fontsize=16)
    plt.title(title, fontsize=22)
```

In [22]:
```
#creating a function to plot histograms
def plot_histogram2(subplot, x, title, xlabel, ylabel, lower_limit, bns):
    '''This function plots multiple histograms'''

    #define subplots
    ax = plt.subplot(1,2,subplot)

    #defining bins
    bins = np.arange(lower_limit, loan_clean[x].max()+bns, bns)

    #plot the histogram
    plt.hist( data = loan_clean, x = x,  bins = bins, color=colors)

    #display graph labels
    plt.xlabel(xlabel, fontsize=16)
    plt.ylabel(ylabel, fontsize=16)
    plt.title(title, fontsize=22)
```

In [23]:
```
#creating a function to plot barcharts
def plot_bar1(x, title, xlabel, ylabel):
    '''This function plots single barcharts'''
```

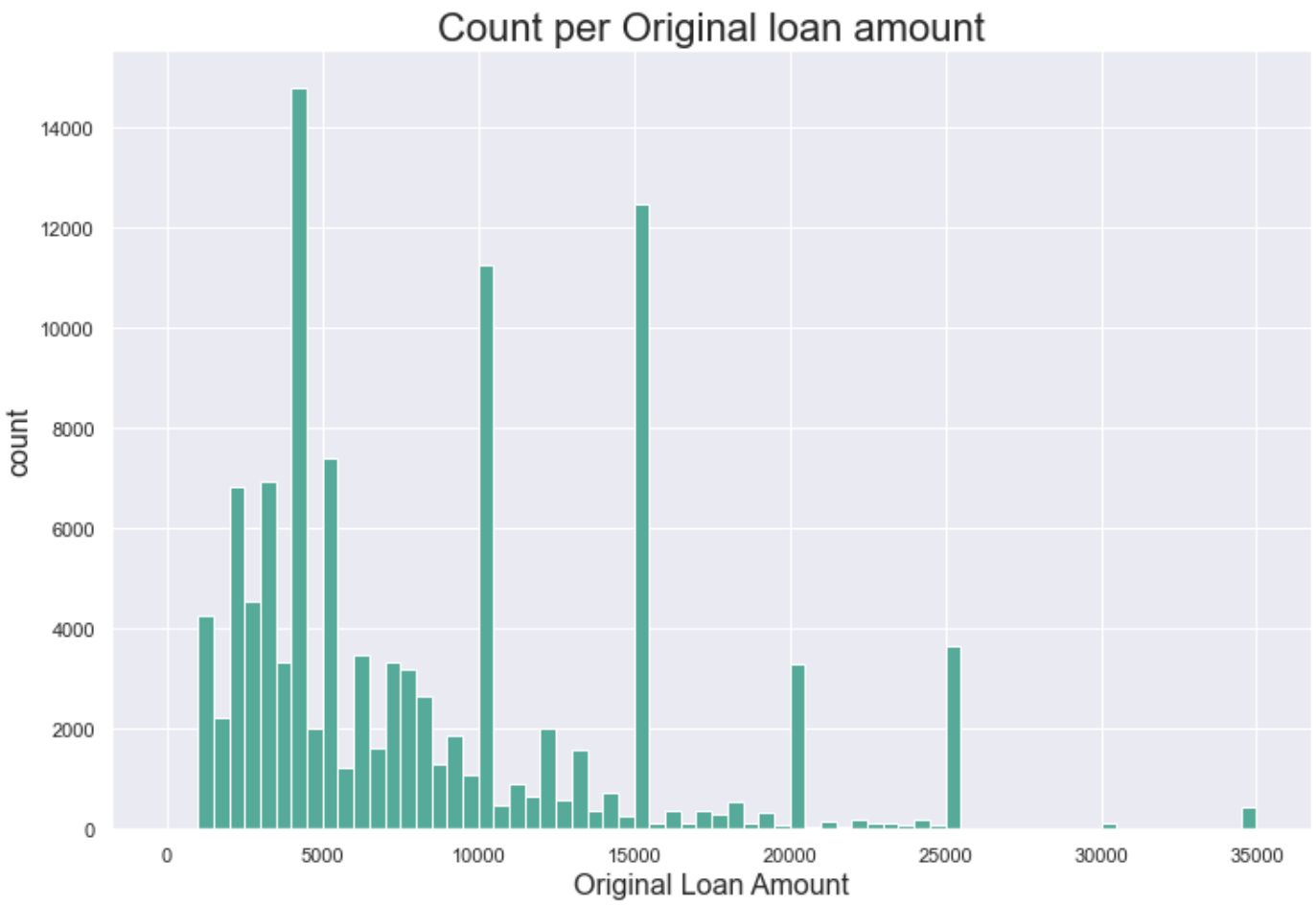
```
    #plot the barchart
    sns.countplot(data=loan_clean, x= x, color =colors)

    #display graph labels
    plt.xlabel(xlabel, fontsize=20)
    plt.ylabel(ylabel, fontsize=20)
    plt.title(title, fontsize=22)
```

In [24]:
```
#creating a function to plot horizontal barcharts
def plot_barh1(y, title, xlabel, ylabel):
    '''This function plots single horizontal barcharts'''


    #plot the barchart
    sns.countplot(data=loan_clean, y= y, order= y.value_counts().index, color =colors)

    #display graph labels
    plt.xlabel(xlabel, fontsize=20)
    plt.ylabel(ylabel, fontsize=20)
    plt.title(title, fontsize=22)
```

In [25]:
```
#creating a function to plot barcharts
def plot_bar2(subplot ,x , title, xlabel, ylabel):
    '''This function plots multiple barcharts'''

    #defining subplot locations
    ax = plt.subplot(1 ,2 ,subplot)

    #plot the barchart
    sns.countplot(data=loan_clean, x= x, color =colors)

    #display graph labels
    plt.xlabel(xlabel, fontsize=20)
    plt.ylabel(ylabel, fontsize=20)
    plt.title(title, fontsize=22)
```

What loan amounts are most requested for?

In [26]:
```
#Histogram showing the distribution of original loans collected
plot_histogram1(x='LoanOriginalAmount', title = 'Count per Original loan amount',xlabel=
```



4000 dollar loans were most requested for. Significant spikes were also observed at the 10000, 15000, 20000 and 25000 dollar loan marks. Lower amounts of monies were requested for compared to the higher values. This graph also shows a decline in the count of individuals who requested for loans above 25000 dollars.

What percentage profit is to be expected as an investor?

In [27]:
```
#Summary statistics on the LenderYield column
loan_clean['LenderYield'].describe()
```

Out[27]:
```
count    113937.000000
mean          0.182701
std           0.074516
min          -0.010000
25%           0.124200
50%           0.173000
75%           0.240000
max           0.492500
Name: LenderYield, dtype: float64
```

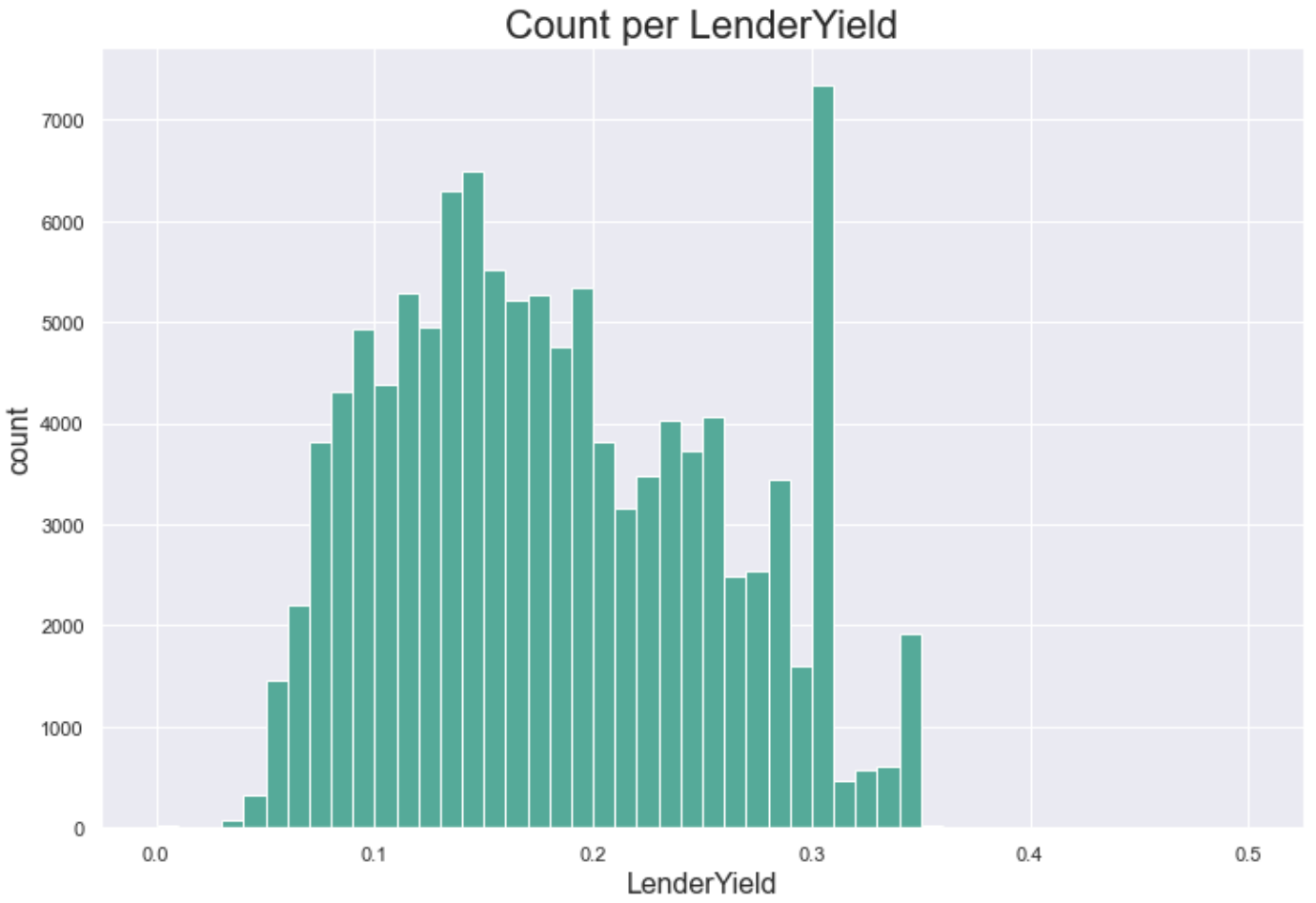
In [28]:
```
loan_clean.query('LenderYield==-0.010000')
```

Out[28]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	LoanStatus	ClosedDa
46874	3F5C3389508503122919326	143562	2007-05-28 06:40:54.803	HR	36	Defaulted	2010-10-
65259	BB4E34191946516388AD563	310792	2008-04-15 05:35:10.670	HR	36	Completed	2011-04-

76858	7ADE33852947773364BFEDD6	115163	2007-03-24 10:03:30.717	C	36	Completed	2010-03-
78920	CBC034185520106668B09260	319969	2008-04-28 11:48:10.400	C	36	Completed	2009-09-
112717	C52F3426964405073574757	369381	2008-07-17 22:44:53.000	A	36	Completed	2010-10-

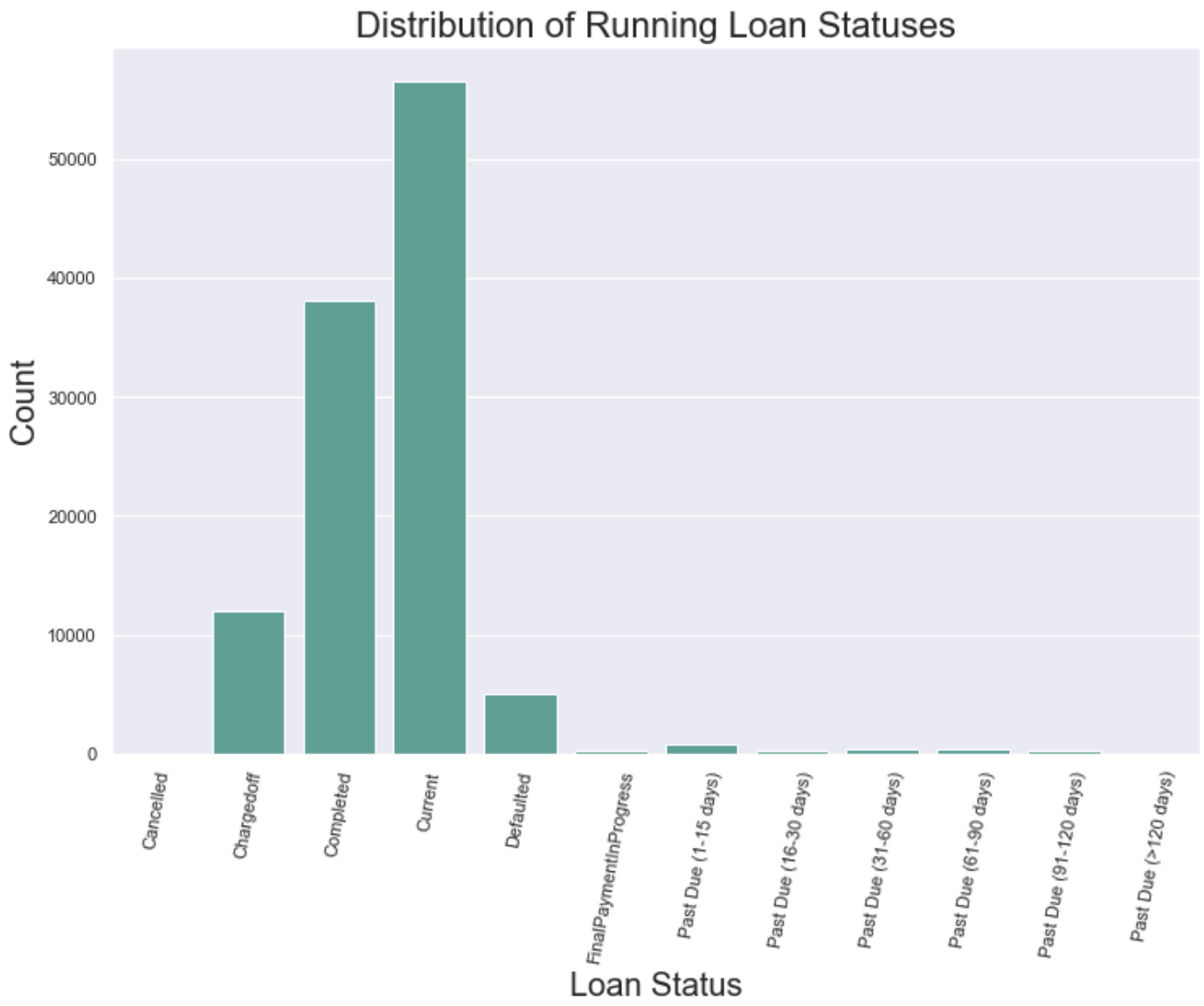
In [29]:
```
#Histogram showing lender yield percentage distribution
plot_histogram1(x='LenderYield', title = 'Count per LenderYield',xlabel= 'LenderYield',
```



The highest possible yield on any loan is 0.492500% on the loaned amount. 0.3% has been most earned by investors, followed closely by 0.15%. On 5 loans, some investors earned -0.010%. This means that as an investor, if the right loans are not selected before investment, one could end up with losses. This business definitely generates more gains than losses for its investors, making it a viable income source.

What is the current state of prosper loans?

In [30]:
```
#Barchart showing the distribution of current prosper loans
plot_bar1(x = 'LoanStatus', title= "Distribution of Running Loan Statuses", xlabel= "Loa
plt.xticks(rotation = 80);
```



Most individuals using this service have loans that are currently running while a significant number of individuals have successfully collected and cleared off their loans. A minimal number of accounts are having difficulties paying back their premiums as agreed.

What is the income range of individuals who have requested these loans?

In [43]:
```
#creating a dictionary of the categories in ordered form
arranged_income = { 'IncomeRange': ['$0', '$1-24,999', '$25,000-49,999', '$50,000-74,999


#changing the data type to category
for col in arranged_income:
    cat_type = pd.api.types.CategoricalDtype(ordered = True, categories = arranged_incom
    loan_clean[col] = loan_clean[col].astype(cat_type)

loan_clean['IncomeRange'].replace('IncomeRange', inplace = True)
```

In [44]:
```
#Barchart showing income range count.
plot_bar1(x = 'IncomeRange', title= "Count per Income Range ", xlabel= "Income Range", y
```

Individuals who earn between 25,000 and 75,000 dollars have most access to prosper loans. Individuals who earn below 25,000 dollars or are unemployed do not receive as many loans as their other counterparts. This may be a significant feature for prosper loan collection.

What year and month had the highest loan requests?

In [34]:
```
#Create new columns for the year, month and weekday of loan origination
loan_clean['loan_created_year'] = loan_clean['loan_origination_date'].dt.year
loan_clean['loan_created_month'] = loan_clean['loan_origination_date'].dt.month_name()


loan_clean['loan_created_day'] = loan_clean['loan_origination_date'].dt.day_name()
loan_clean.head(3)
```

Out[34]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	LoanStatus	ClosedDate	Bo
0	1021339766868145413AB3B	193129	2007-08-26 19:09:29.263	C	36	Completed	2009-08-14	
1	10273602499503308B223C1	1209647	2014-02-27 08:28:07.900	NaN	36	Current	NaT	
2	0EE9337825851032864889A	81716	2007-01-05 15:00:47.090	HR	36	Completed	2009-12-17	

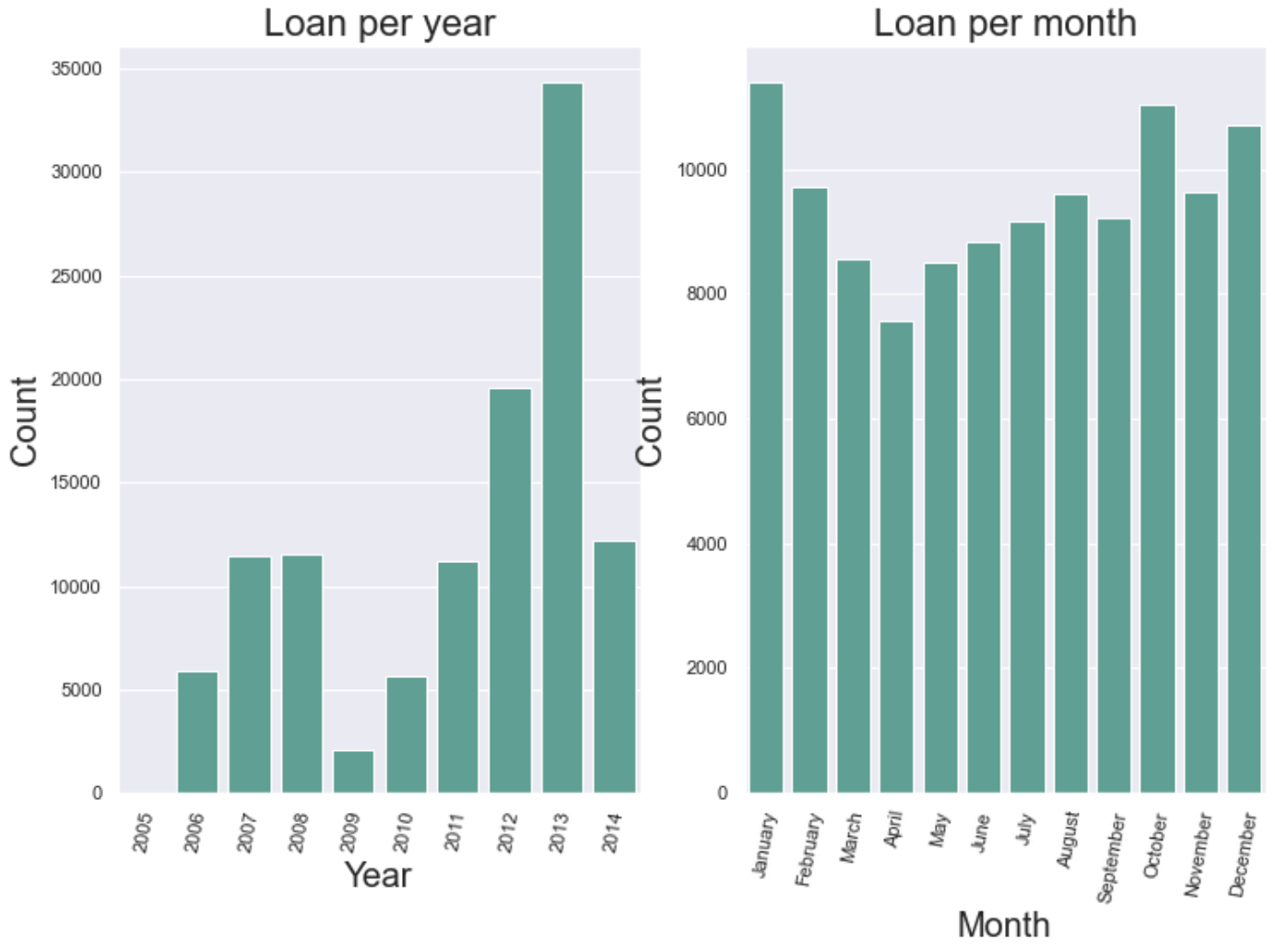
In [35]:
```
loan_clean['loan_created_month'].value_counts()
```

```
January      11395
```

Out[35]:
```
October      11043
December     10708
February      9728
November      9635
August        9592
September     9221
July          9154
June          8847
March         8555
May           8500
April         7559
Name: loan_created_month, dtype: int64
```

In [41]:
```
#creating a dictionary of the categories in ordered form
arranged_col = { 'loan_created_month': ['January', 'February', 'March', 'April', 'May',


#changing the data type to category
for col in arranged_col:
    cat_type = pd.api.types.CategoricalDtype(ordered = True, categories = arranged_col[c
    loan_clean[col] = loan_clean[col].astype(cat_type)

loan_clean['loan_created_month'].replace('loan_created_month', inplace = True)
```

In [42]:
```
#Count plot showing the number of loans received per year and month
plot_bar2(subplot= 1,x= 'loan_created_year' , title= 'Loan per year', xlabel= 'Year', yl
plt.xticks(rotation = 80)
plot_bar2(subplot= 2,x= 'loan_created_month' , title= 'Loan per month', xlabel= 'Month',
plt.xticks(rotation = 80);
```

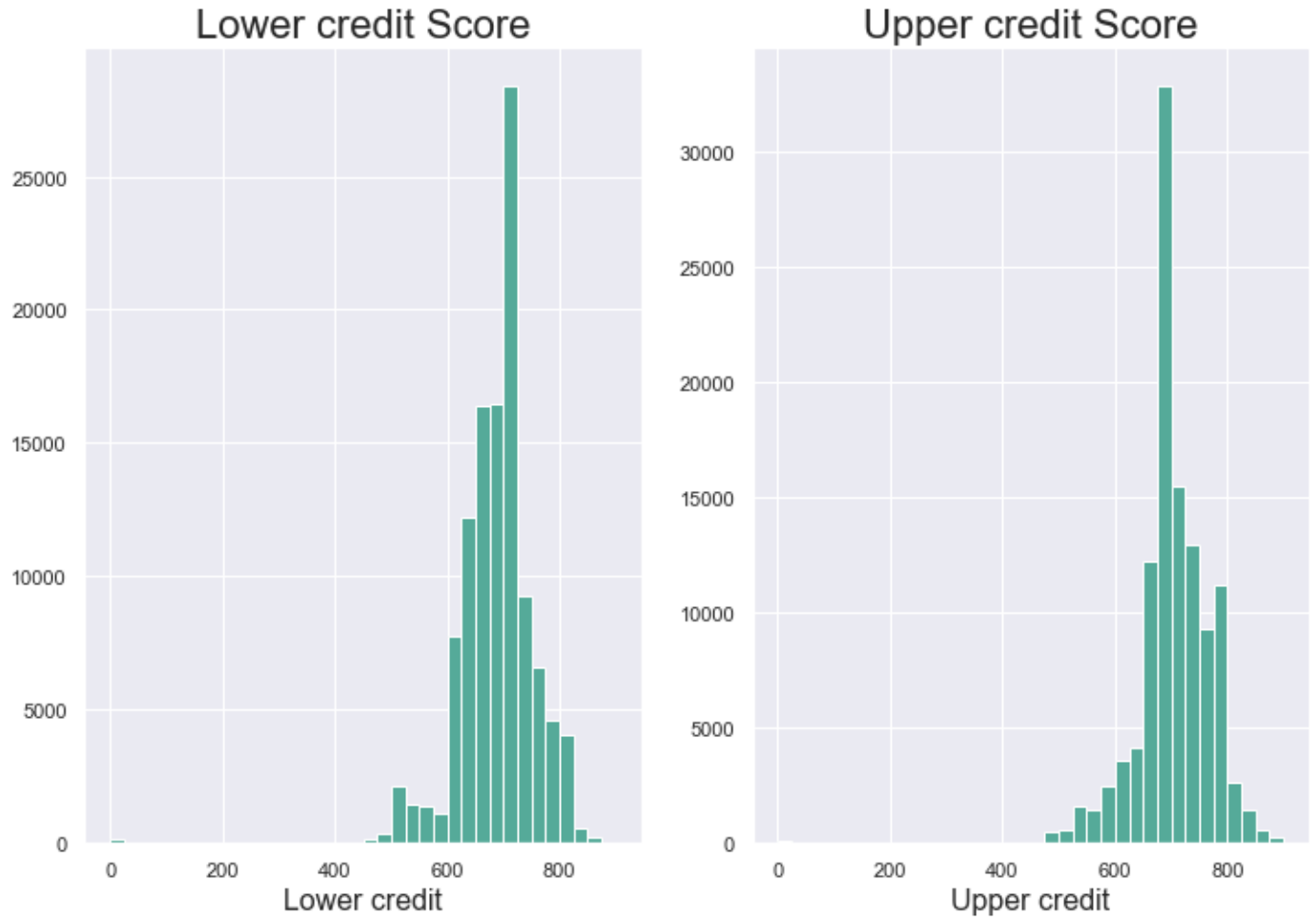


The highest number of loans were given out in 2013. However, a drop of over 50% was observed in 2014.

January was observed to be the month with highest loan collection over the years. The highest rated loan collection purpose was Debt consolidation. After the winter holidays and the numerous bills it brings, there will be a likely need for money to sustain an individual or family till the next paycheck.

What variations or similiarities exist between the lower and upper credit score?

In [60]:
```
#Plot two subplots containing the lower credit score and upper credit score respectively
plot_histogram2(subplot= 1, x='CreditScoreRangeLower', title= 'Lower credit Score', xlab
plot_histogram2(subplot= 2, x='CreditScoreRangeUpper', title= 'Upper credit Score', xlab
```

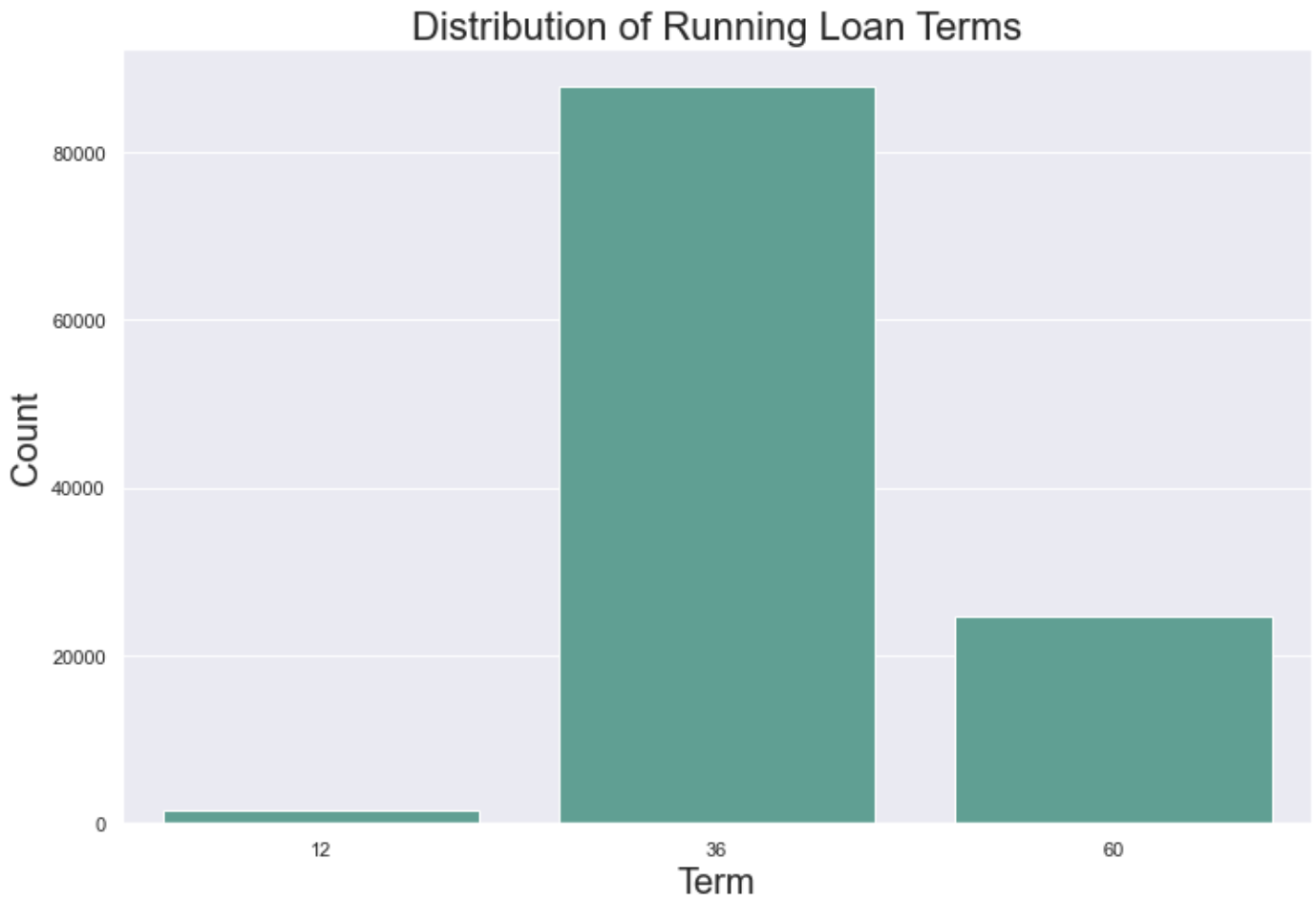


Suprisingly, The lower credit scores peak at 700 - 725. This is slightly higher than the upper credit which peaks at 675-700.

Both scores are almost within the same range

What is the duration of loans taken on the platform?

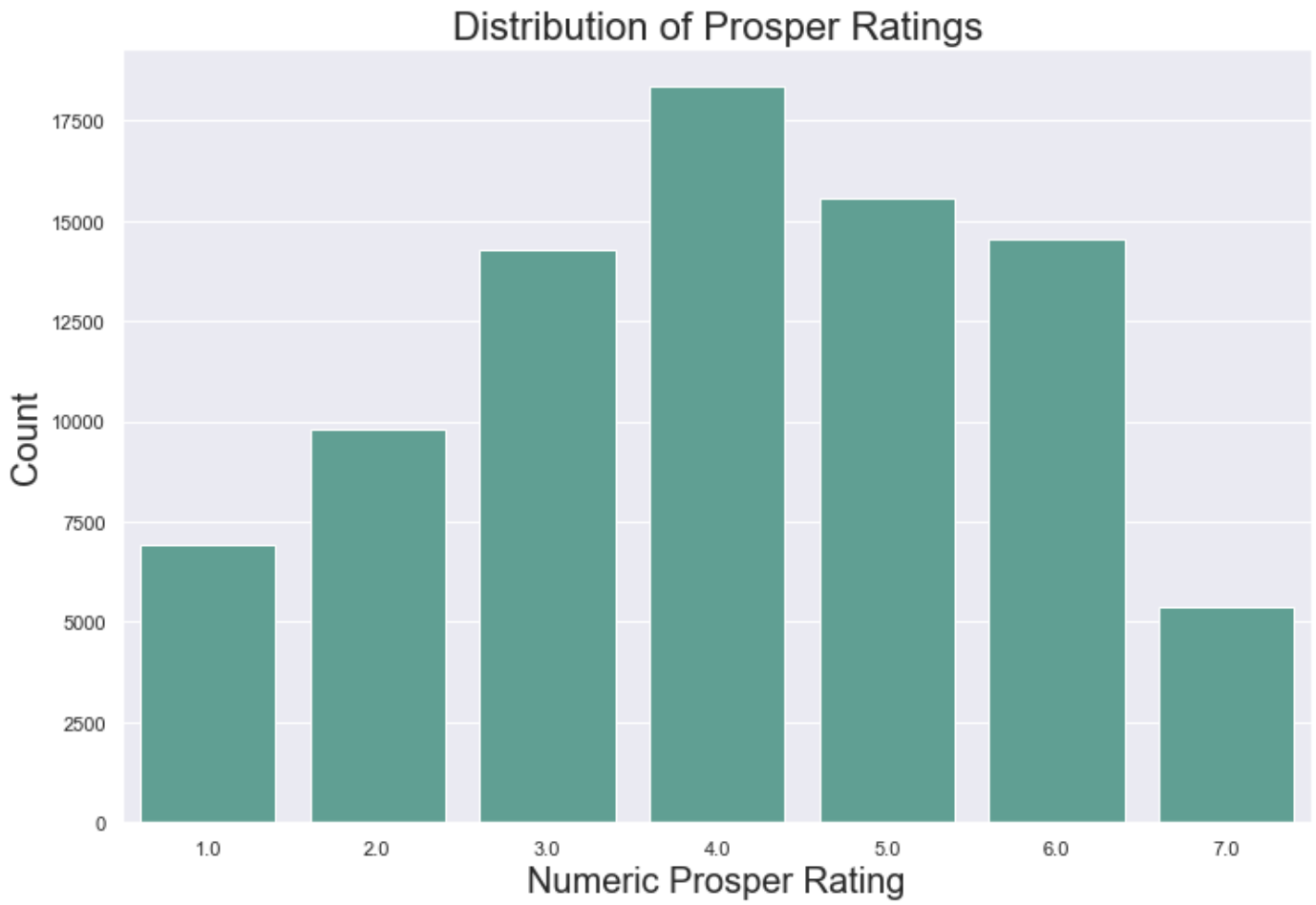
In [61]:
```
#Barchart showing loan terms
plot_bar1(x = 'Term', title= "Distribution of Running Loan Terms", xlabel= "Term", ylabe
```



Three loan terms are approved by Prosper company. 12 months, 36 months and 60 months. More than 50% of loans are to run for 36 months. The least favoured term is the 12 months, likely because of its short duration. Investors are likely to make more money on longer termed loans. The 36 month option is great for investors as it is most favoured by prosper clients.

What are the highest and lowest prosper ratings by count?

In [62]:
```
#Bar chart showing prosper rating count
plot_bar1(x = 'ProsperRating (numeric)', title= "Distribution of Prosper Ratings", xlabe
```

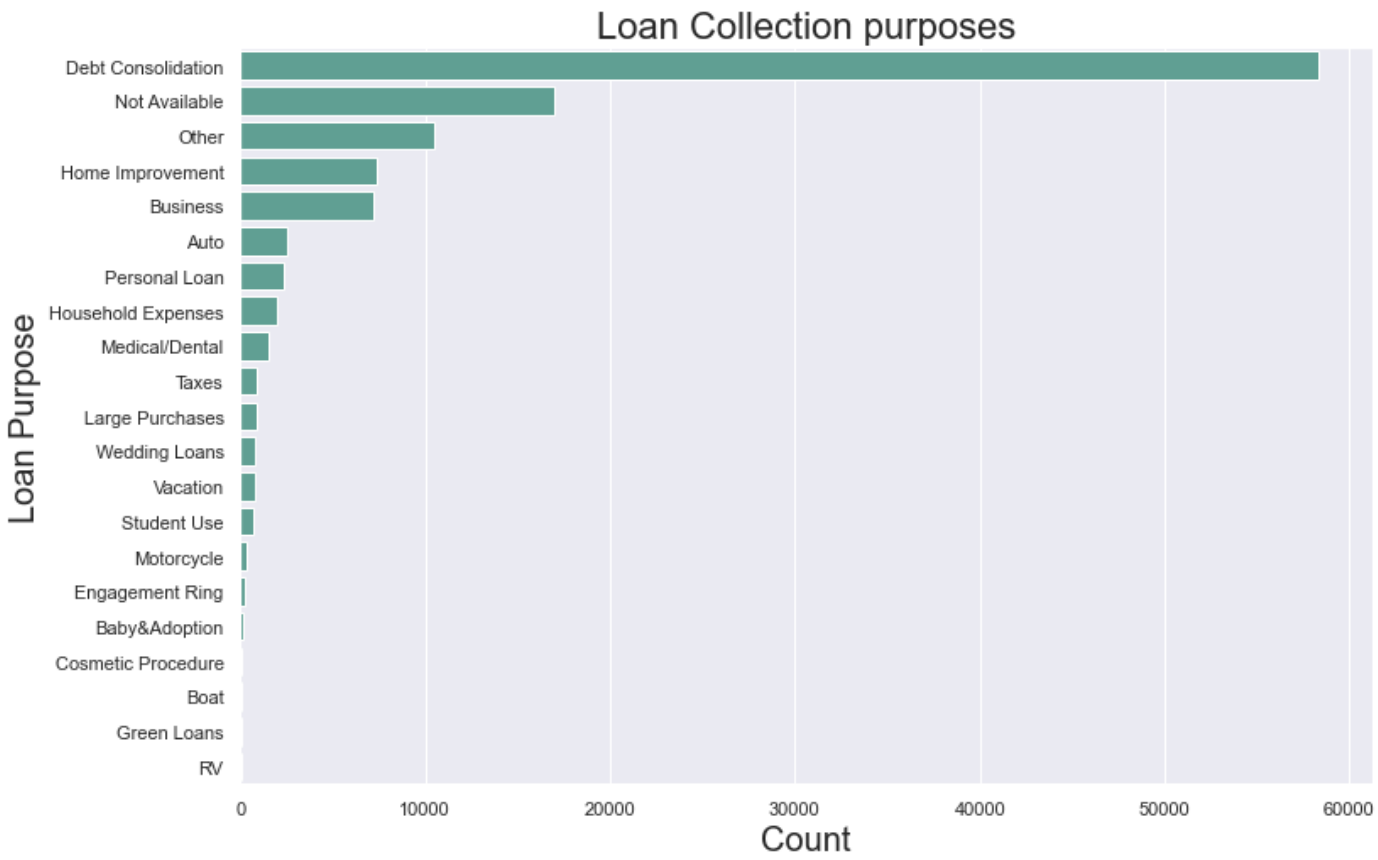


The prosper rating grows in idealty in ascending order. 1.0 being poorly rated and 7.0 being highly rated. Highly rated status was most difficult to achieve as fewer individuals fall in this category compared to others. Most individuals performed averagely well

What purposes are most loans taken for?

In [63]:
```
#Replace the numeric values with their corresponding meanings
Listing_Categories = {0: 'Not Available', 1: 'Debt Consolidation', 2: 'Home Improvement'
                      8:'Baby&Adoption', 9:'Boat', 10:'Cosmetic Procedure', 11:'Engage
                      15:'Medical/Dental', 16: 'Motorcycle', 17:'RV', 18:'Taxes', 19:'

loan_clean['ListingCategory (numeric)'].replace(Listing_Categories, inplace = True)

#Rename the column, taking out the numeric tag
loan_clean.rename(columns={'ListingCategory (numeric)':'ListingCategory'}, inplace = Tru
```

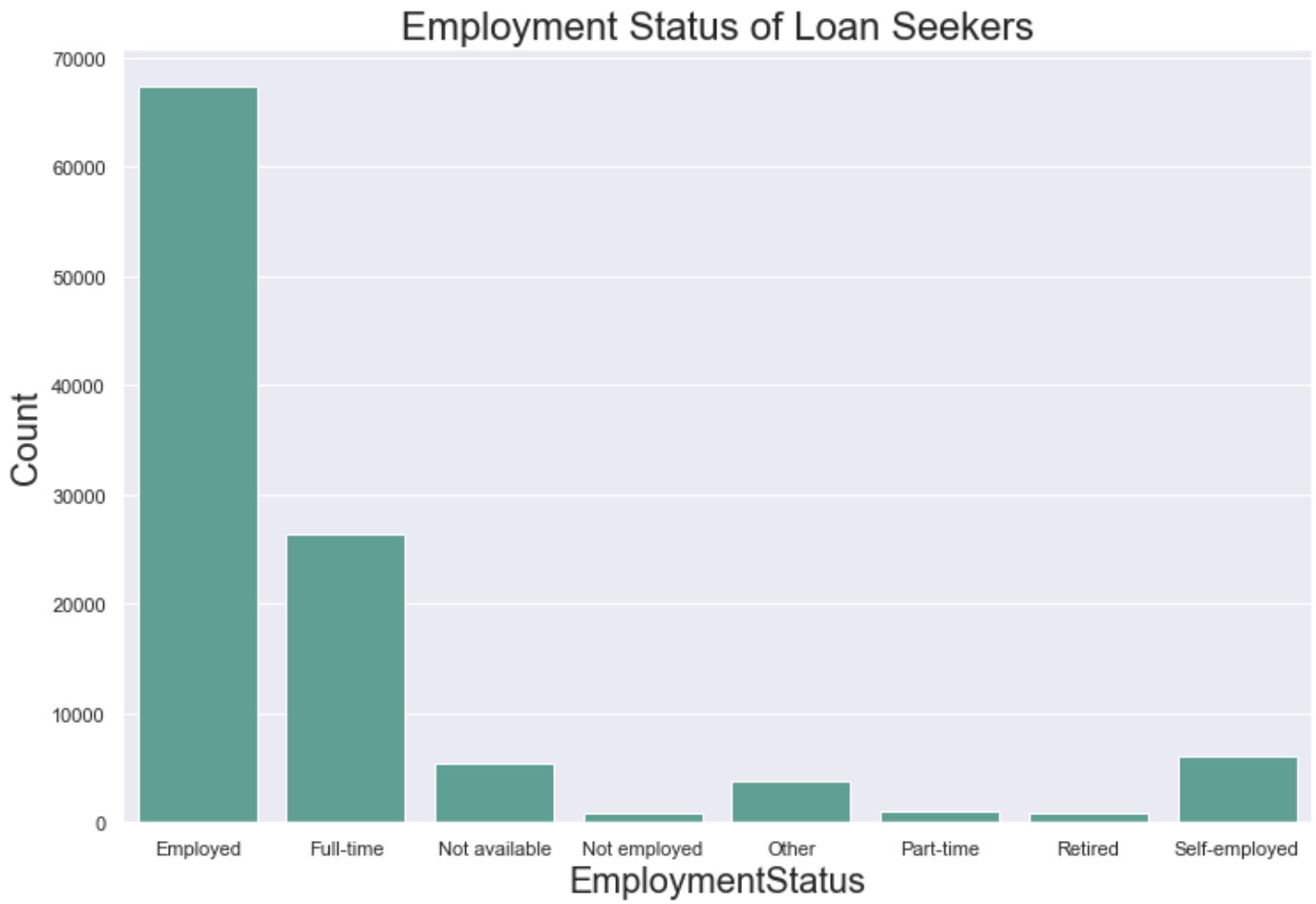
In [64]:
```
#Horizontal barchart showing Listing categories
plot_barh1(y = loan_clean['ListingCategory'], title= "Loan Collection purposes", xlabel=
```



Debt consolidation was the foremost reason for prosper loan collection. This is a little disturbing as more debt is collected to fund other debts. Money making ventures such as Business or student use did not take the forefront.

What is the employment status of individuals seeking loans?

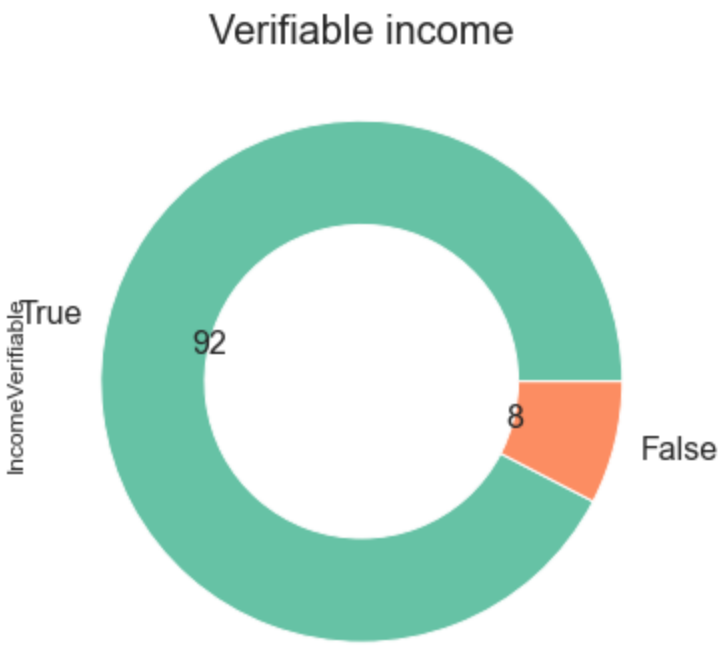
In [65]:
```
#Barchart showing Employment status distibution.
plot_bar1(x = 'EmploymentStatus', title= "Employment Status of Loan Seekers", xlabel= "E
```



Most individuals taking these loans are employed. Loan recovery will be easier amongst the employed compared to the not employed.

What proportion of clients have a verifiable income?

In [66]:
```
#create donut chart
plt.figure(figsize=[8,6])
loan_clean['IncomeVerifiable'].value_counts().plot(kind="pie",  autopct='%0.0f', colors

plt.title('Verifiable income', fontsize=20)
plt.show()
```



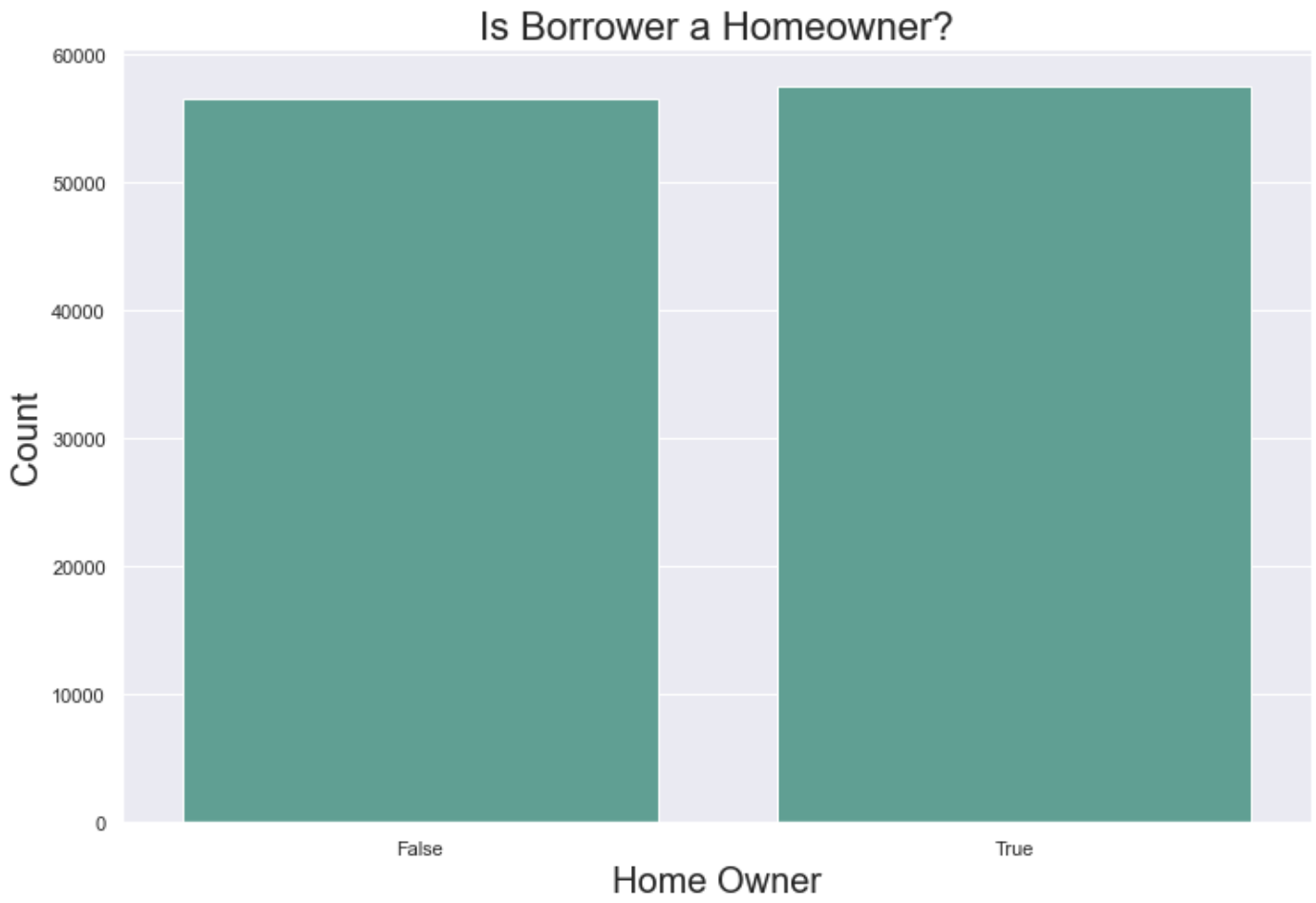
About 92 percent of individuals given a prosper loan have a verifiable source of income

What proportion of borrowers own homes?

In [67]:
```
loan_clean['IsBorrowerHomeowner'].value_counts()
```

Out[67]:
```
True     57478
False    56459
Name: IsBorrowerHomeowner, dtype: int64
```

In [68]:
```
#Barchart showing count of home owners and non homeowners.
plot_bar1(x = 'IsBorrowerHomeowner', title= "Is Borrower a Homeowner?", xlabel= "Home Ow
```



About 57478 individuals own homes while about 56459 individuals do not own homes. I will like to discover if this is an important metric which should be considered before approving or investing in a loan.

Bivariate Exploration

In this section, i investigate relationships between pairs of variables in the prosper loan dataset.

Is one's credit rating impacted by his/her monthly income?

In [69]:
```
#Summary statistics on the StatedMonthlyIncome column
loan_clean['StatedMonthlyIncome'].describe()
```

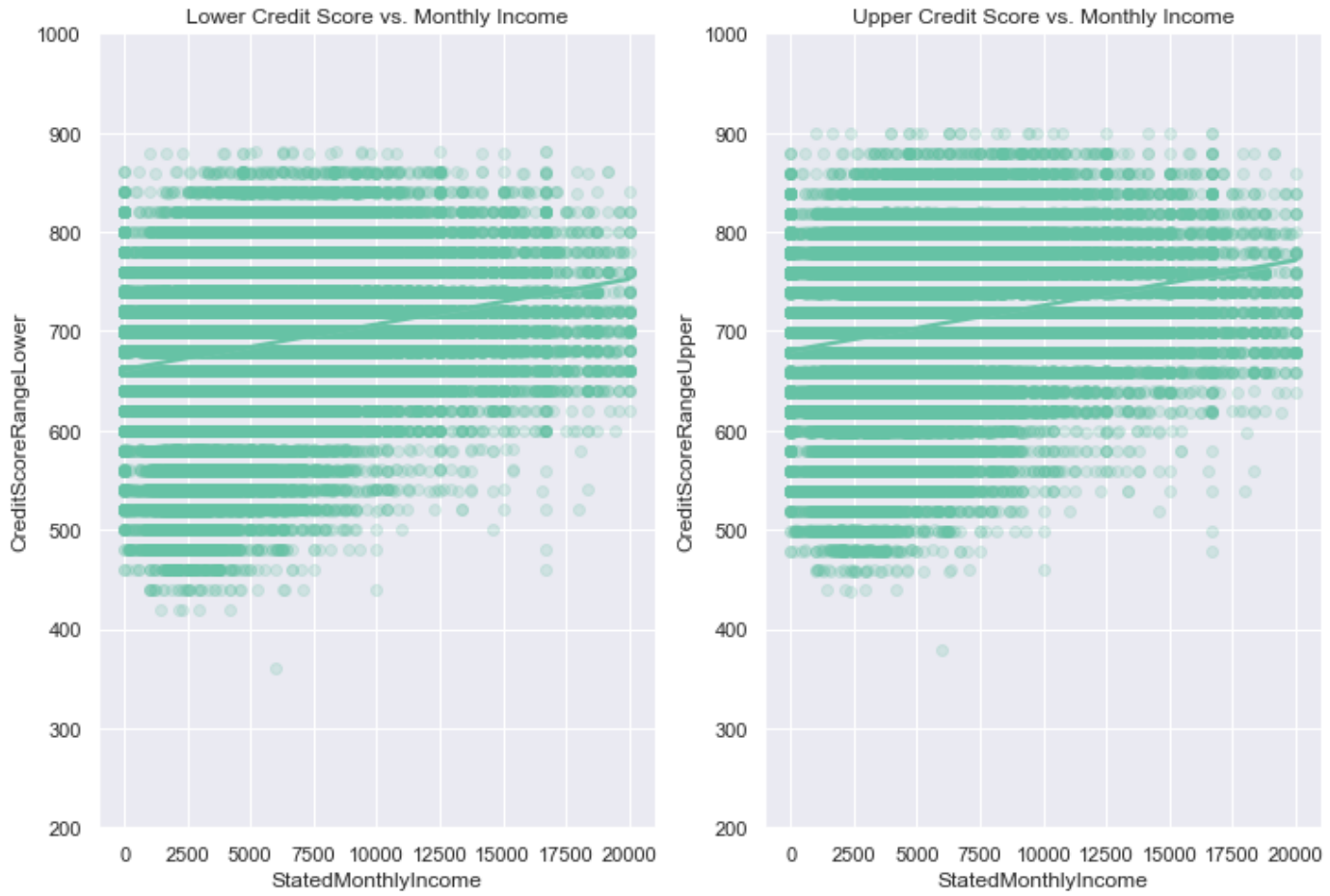
Out[69]:
```
count    1.139370e+05
mean     5.608028e+03
std      7.478497e+03
min      0.000000e+00
25%      3.200000e+03
50%      4.667000e+03
75%      6.825000e+03
max      1.750003e+06
Name: StatedMonthlyIncome, dtype: float64
```

The maximum value of 1,750,000 dollars is an outlier which will affect our results. With a mean of 5,608 dollars, a standard deviation of 7,478 dollars and 75% of the population falling below a 6,825 dollar monthly salary, i will work with 20,000 dollars as the maximum value.

In [70]:
```
#replace all value above $15000 with the median of the data to avoid outliers
loan_clean['StatedMonthlyIncome'].values[loan_clean['StatedMonthlyIncome'] > 20000] = lo
```

In [71]:
```
#create function to plot scatterplot
def plot_scatter1(x, y, title, xlabel, ylabel, transparency):
    '''This function plots a single scatterplot'''

    #regression plot
    sns.regplot(data = loan_clean, x=x, y=y, x_jitter=0.3, y_jitter=0.3, scatter_kws={'a

    #display graph labels
    plt.xlabel(xlabel, fontsize=20)
    plt.ylabel(ylabel, fontsize=20)
    plt.title(title, fontsize=22)
```

In [72]:
```
# subplot 1: Monthly income vs Lower credit score
plt.subplot(1, 2, 1)
sns.regplot(data = loan_clean, x='StatedMonthlyIncome', y='CreditScoreRangeLower',x_jitt
plt.ylim (200, 1000)
plt.title('Lower Credit Score vs. Monthly Income', fontsize=12);

# subplot 2: Monthly income vs Upper credit score
plt.subplot(1, 2, 2)
sns.regplot(data = loan_clean, x='StatedMonthlyIncome', y='CreditScoreRangeUpper', x_jit
plt.ylim (200, 1000)
plt.title('Upper Credit Score vs. Monthly Income', fontsize=12);
```

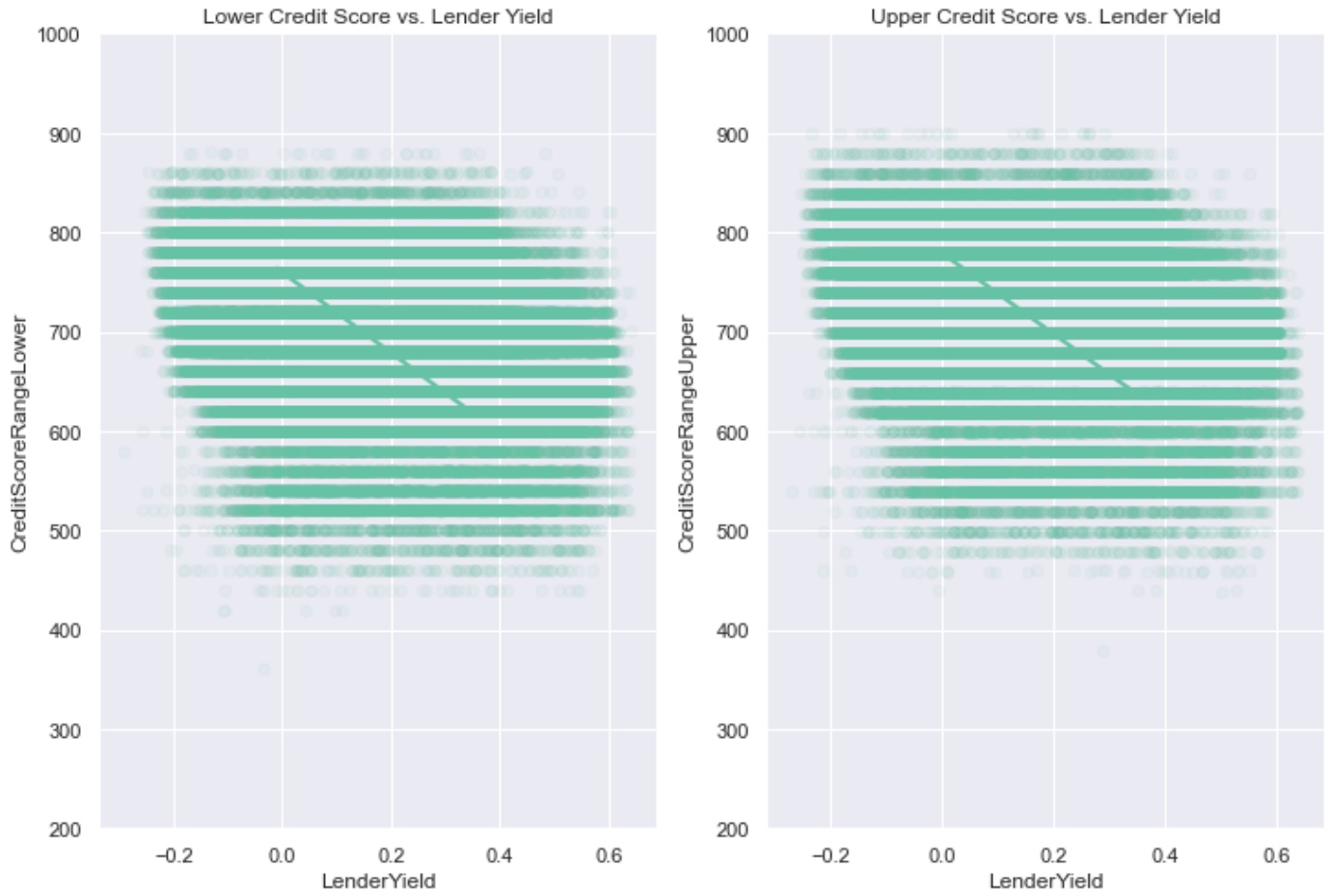


A positive correlation can be observed between the monthly income and both lower and upper credit score range. This means that the higher the stated income, the higher the probabaity of having a high credit score.

Is there a relationship between one's credit rating

and the expected lenders yield?

In [74]:
```
# subplot 1: Monthly income vs Lower credit score
plt.subplot(1, 2, 1)
sns.regplot(data = loan_clean, x='LenderYield', y='CreditScoreRangeLower',x_jitter=0.3,
plt.ylim (200, 1000)
plt.title('Lower Credit Score vs. Lender Yield', fontsize=12);

# subplot 2: Monthly income vs Upper credit score
plt.subplot(1, 2, 2)
sns.regplot(data = loan_clean, x='LenderYield', y='CreditScoreRangeUpper', x_jitter=0.3,
plt.ylim (200, 1000)
plt.title('Upper Credit Score vs. Lender Yield', fontsize=12);
```

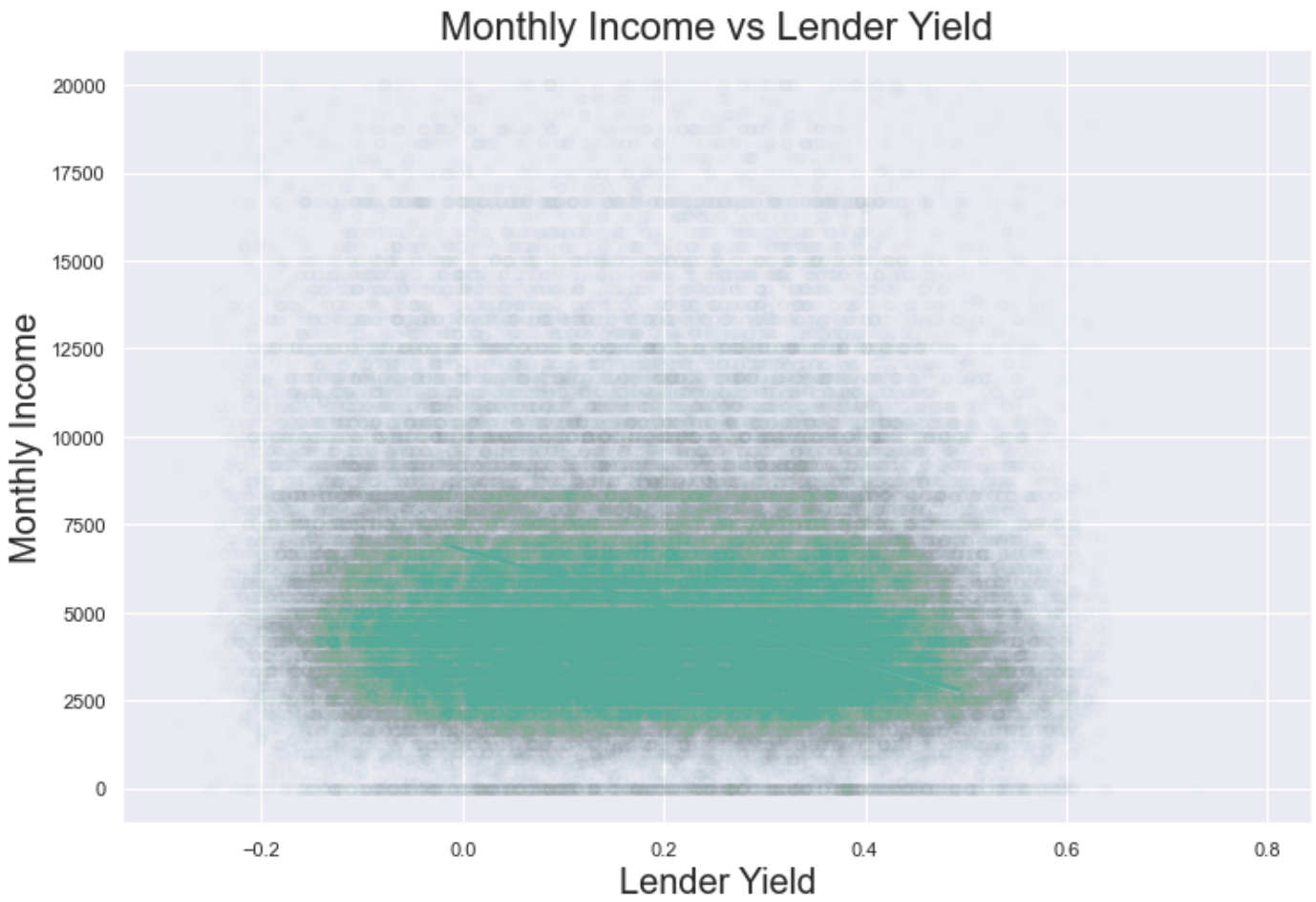


A negative correlation between credit rating and lender's yield is observed. The previous analysis showed a positive relationship between ones stated credit rating and monthly stated income. We further explore the relationship between monthly income and lender yield.

Is there a relationship between one's stated income and investor's lender yield?

In [343…

```
#Scatter plot showing Monthly Income against Lender Yield
plot_scatter1(x= 'LenderYield', y= 'StatedMonthlyIncome', title= 'Monthly Income vs Lend
```



Suprisingly, A negative correlation is observed between stated monthly income and Lender's yield. This means that the investor is likely to have a lower percentage yield as the monthly income of the client increases. This does not however directly translate to lower monies.

Does an individual's monthly income affect the loan amount that can be collected ?

In [345…

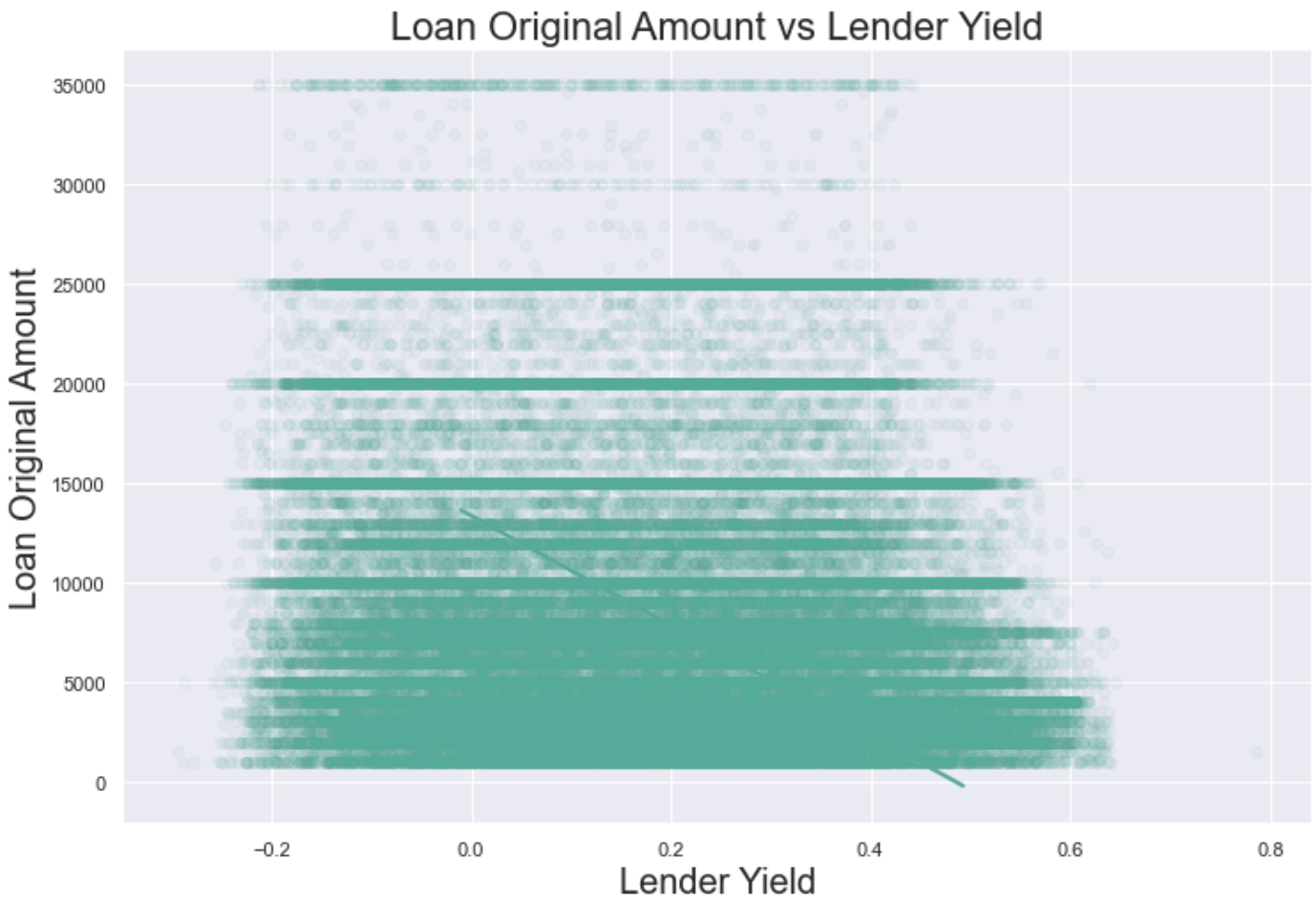
```
#Scatter plot showing Loan Original Amount against Monthly Income
plot_scatter1(x= 'StatedMonthlyIncome', y= 'LoanOriginalAmount', title= 'Loan Original A
```

A positive relationship can be observed between the loan amount collected and the stated monthly income. This means with higher income, one has the possibility to access higher loans. To understand why the lender's yield negatively correlates with stated income, we further observe the relationship between the loan amount collected and the lender's yield.

How does the Loan amount affect Lender yield?

In [346…

```
#Scatter plot showing Loan Original Amount against Lender Yield
plot_scatter1(x= 'LenderYield', y= 'LoanOriginalAmount', title= 'Loan Original Amount vs
```



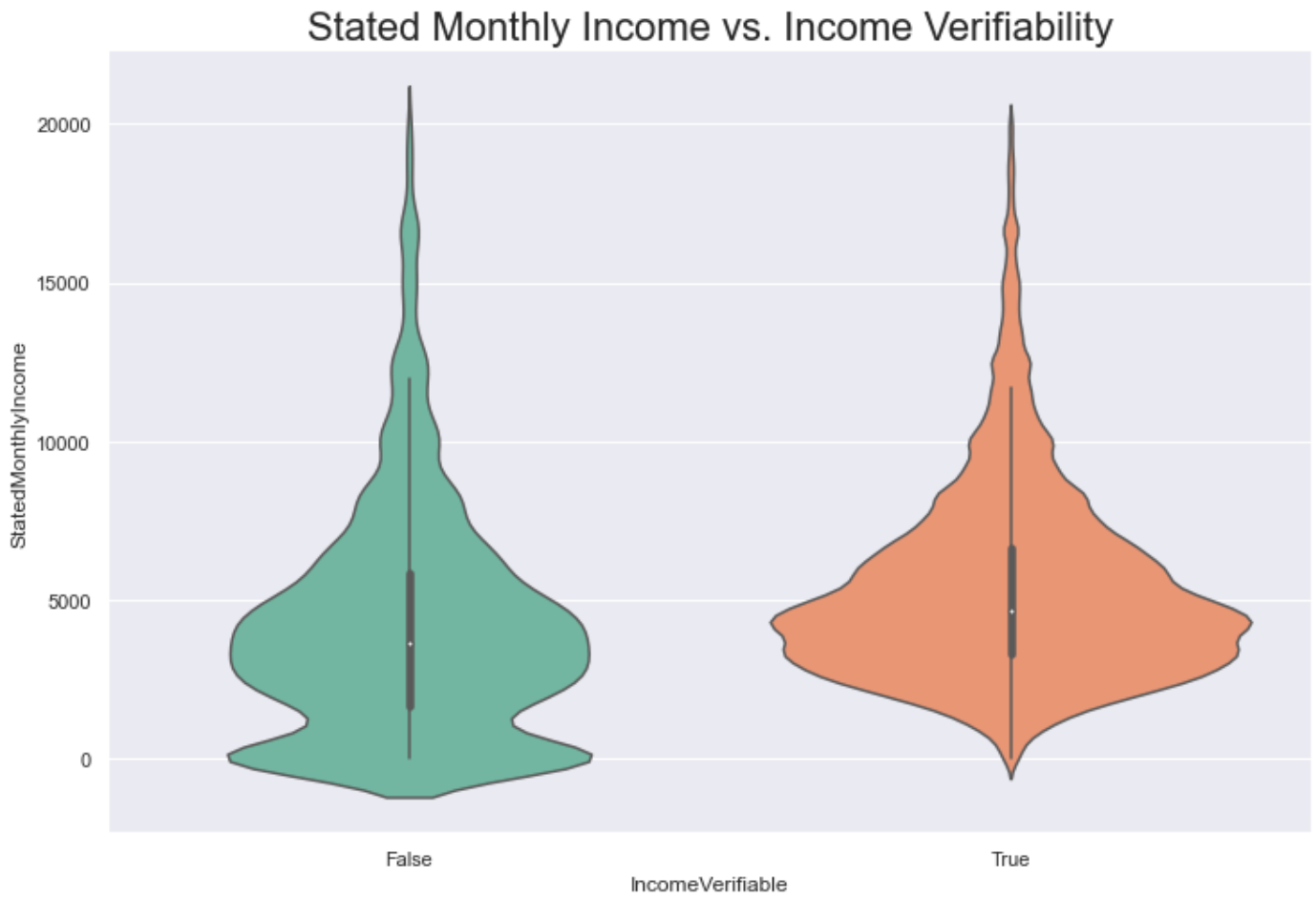
The loan amount and lender yield are negatively correlated. This means that as the loan amount increases, the percentage charged the client, and gained by the investor on that loan reduces.

This also explains the negative relationship between stated monthly income and lender yield. A higher income creates the opportunity for higher loan amount collection.

Although the expected percentage yield drops as loan amount increases, this does not mean the actual amount is smaller. The investor needs to identify the exact rate and loan amount to know what is to be expected by the Lender

Does any significant difference exist between the monthly income of those with verifiable income and those which cannot be verified?

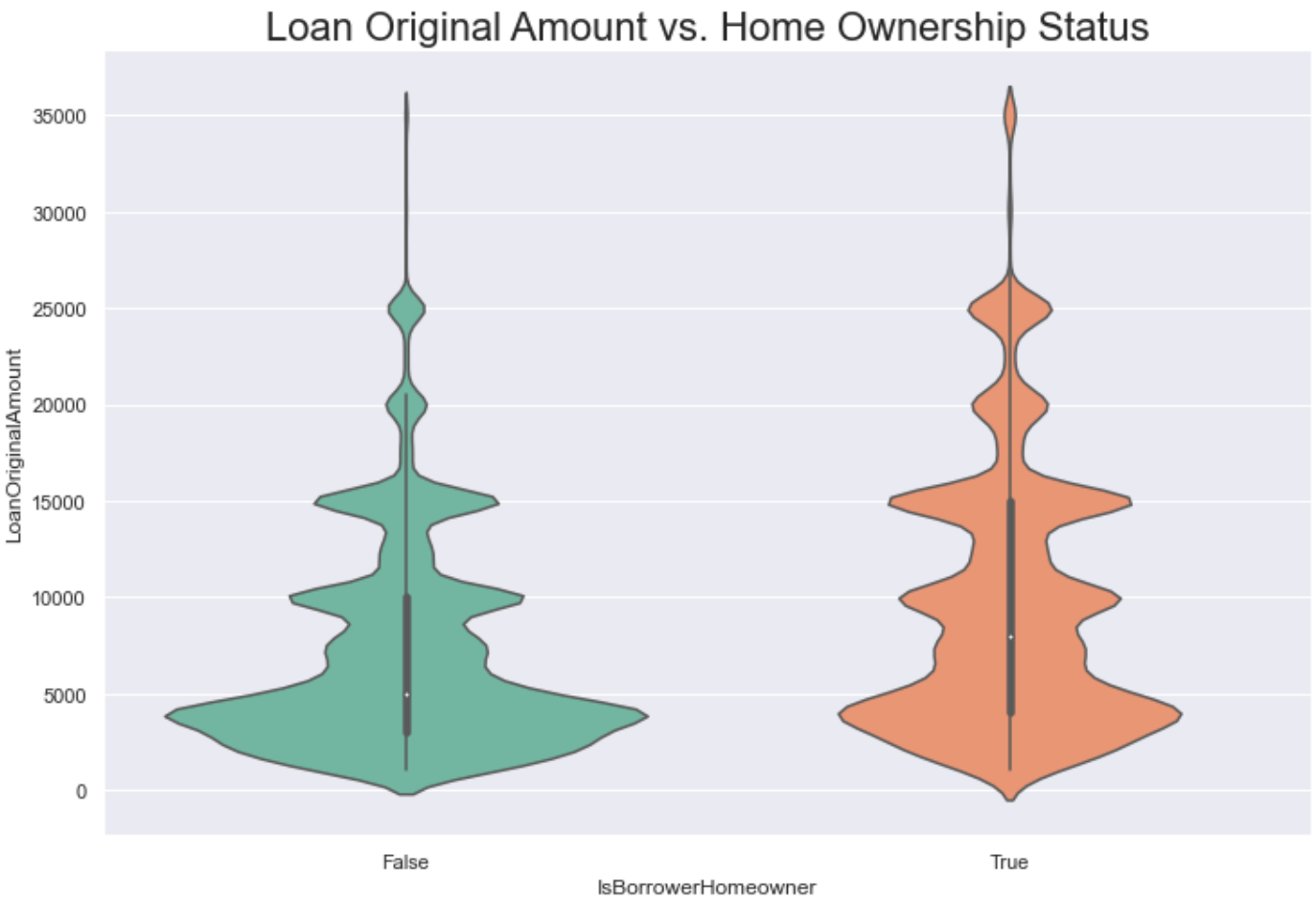
In [75]:
```
#Violin plot comparing income verifiability and monthly income
sns.violinplot(data = loan_clean, x='IncomeVerifiable', y= 'StatedMonthlyIncome')
plt.title('Stated Monthly Income vs. Income Verifiability', fontsize=22);
```



A multimodal distribution is observed amongst individuals without a verifiable income. The average stated income of both distributions significantly peak below $5000. A second peak for the unverifiable is observed at zero. This makes sense as a good number of these individuals do not have an income, hence the need for loans to survive. It also attempts to explain why the highest use for these prosper loans is to pay off debts.

What behaviour can be observed amongst home owners and non-homeowners in relation to loan amounts collected?

In [76]:
```
#Violin plot comparing home ownership and loan amount
sns.violinplot(data = loan_clean, x='IsBorrowerHomeowner', y= 'LoanOriginalAmount')
plt.title('Loan Original Amount vs. Home Ownership Status', fontsize=22);
```

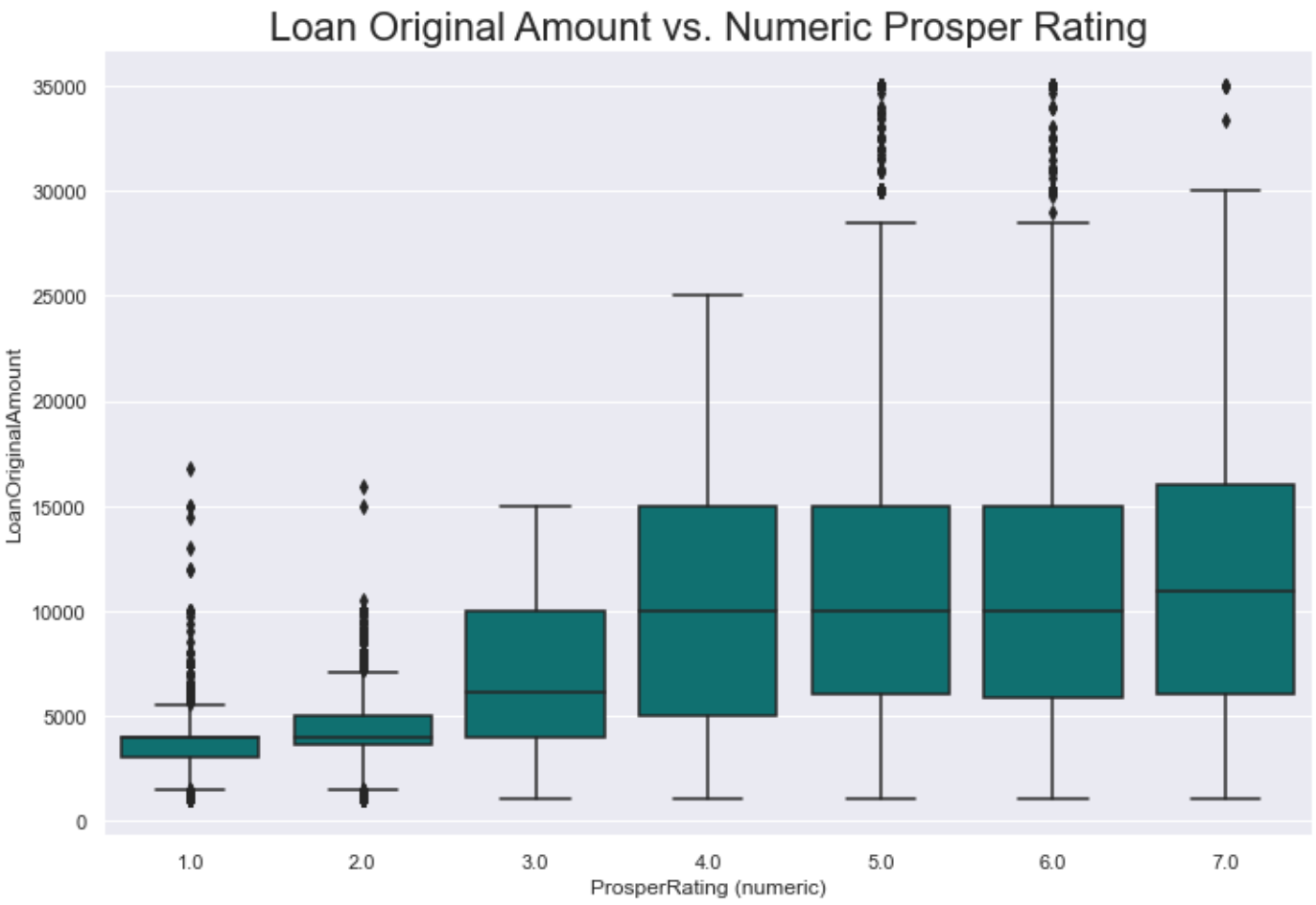


Home owners are observed to collect significantly higher loan amounts on average, compared to non home owners. Both distributions are multimodal, with clusters found at different loan areas. Most non home owners collected loans below 5000 dollars.

The proportion of house owners who collected higher loan amounts is greater than those who do not own homes. However, getting a high loan amount is still possible, without owning a home.

How does the prosper rating affect possible loan amount?

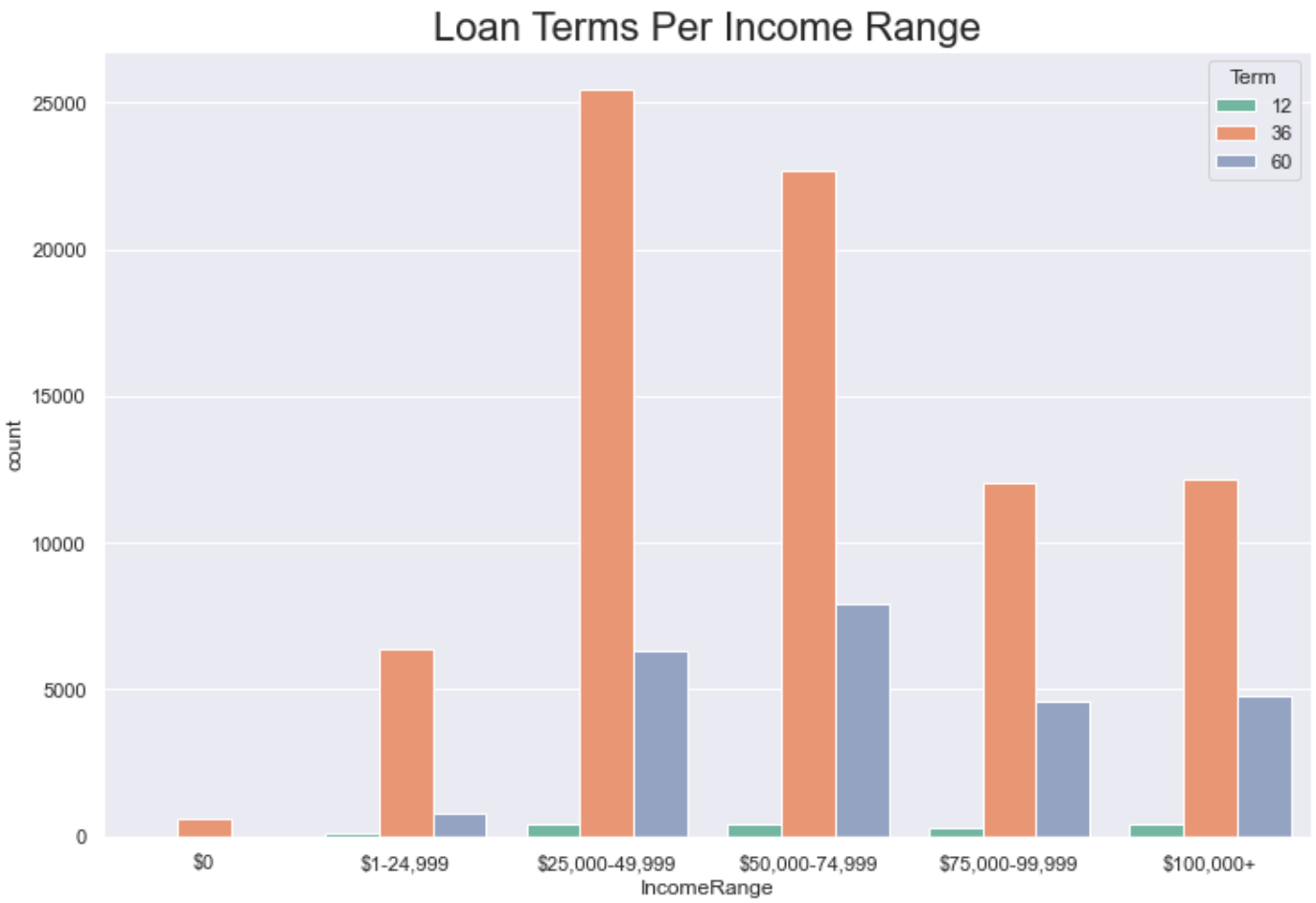
In [77]:
```
#Boxplot plot comparing Prosper rating and loan amount
sns.boxplot(data = loan_clean, x='ProsperRating (numeric)', y= 'LoanOriginalAmount', col
plt.title('Loan Original Amount vs. Numeric Prosper Rating', fontsize=22);
```



High risk rated(1.0) individuals were given significantly lower loan amounts compared to higher scored individuals. More individuals with a prosper score of 4.0 and above enjoyed loans from above $10,000. For the investor, this is a good thing as Prosper Company can be trusted to keep risks at bare minimum.

How long does each income range need to pay off loans?

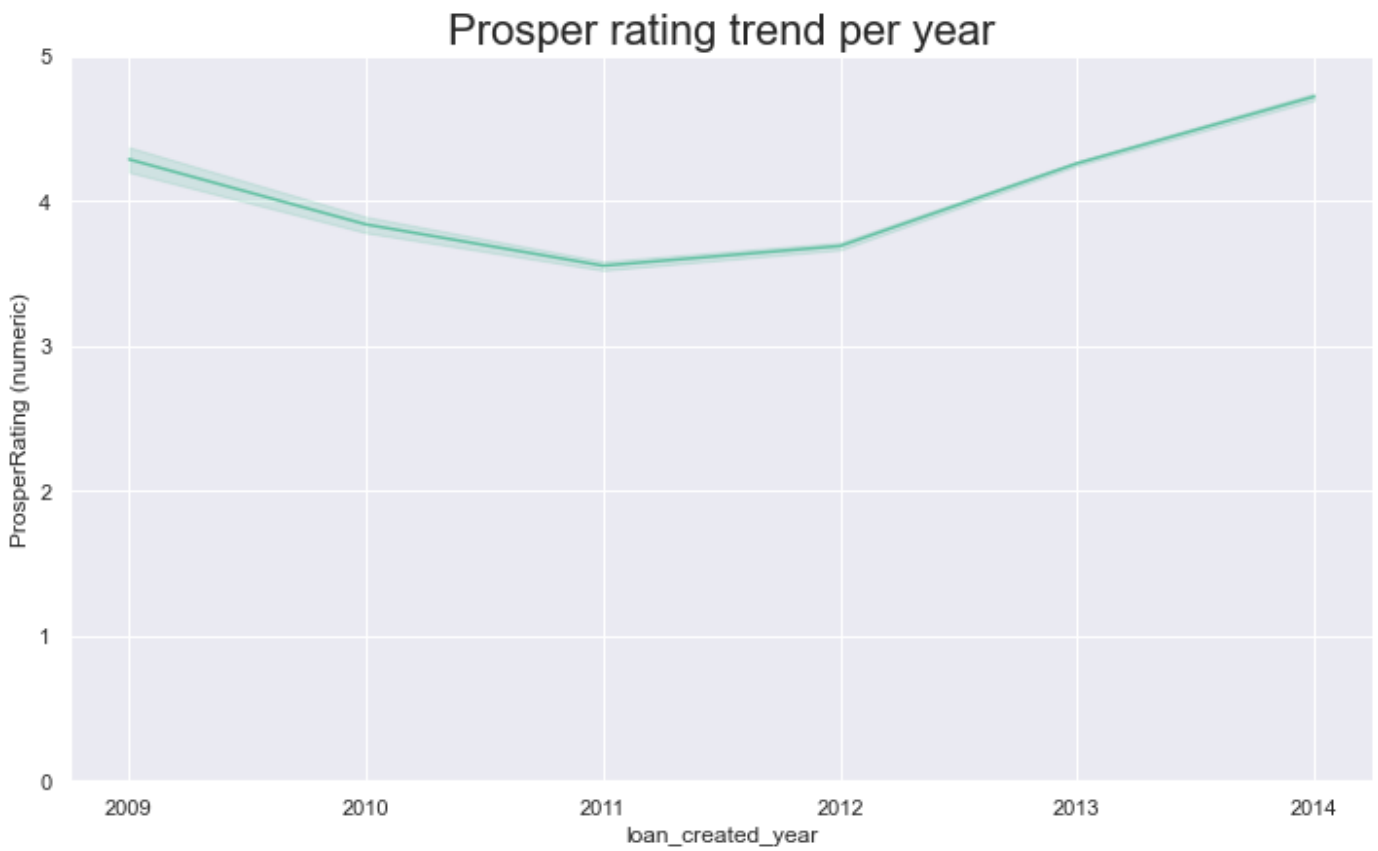
In [78]:
```
#Clustered barchart showing the distribution of income range over loan term
sns.countplot(data= loan_clean, x= 'IncomeRange', hue='Term')
plt.title('Loan Terms Per Income Range', fontsize=22);
```



Interestingly, amongst all income ranges, the 36 month loan duration was prefered. 60 months duration was most observed in individuals earning between 50000 and 75000 dollars. If an investor will prefer longer term monthly profits, its better to focus on individuals earning within this range.

What is the average Prosper rating trend per year?

In [79]:
```
#Line chart showing Prosper rating trend per year
sns.relplot(data=loan_clean, x='loan_created_year', y='ProsperRating (numeric)', kind="l

#set y axis to start from Zero
plt.ylim(0,5)
plt.title('Prosper rating trend per year', fontsize=22);
```

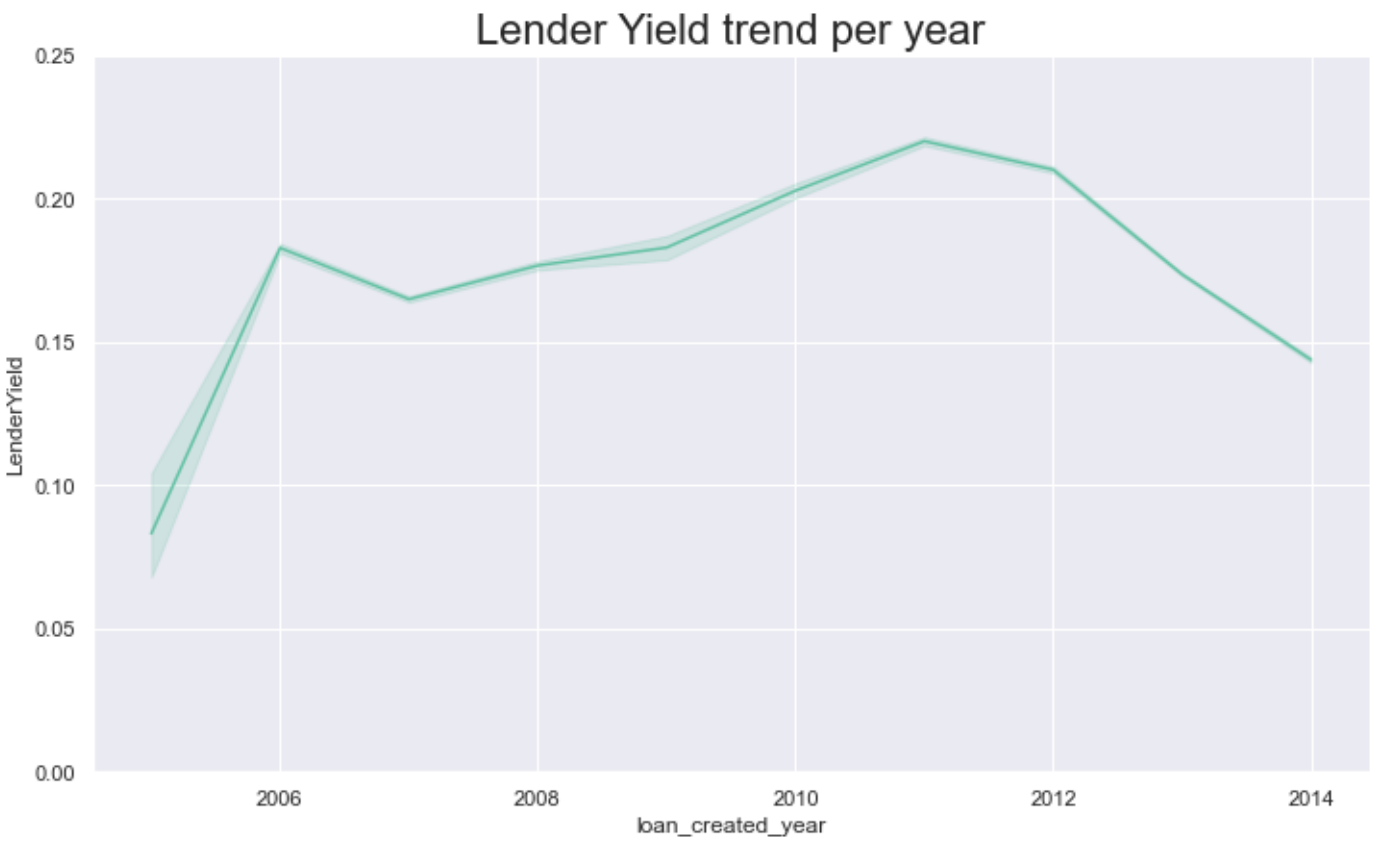


In 2009, the average prosper rating was above 4.0. This value dropped in 2010 through 2012. It started rising in 2013 and hit its highest in 2014.

This shows that in recent times, Prosper company is attracting and giving loans to better rated individuals on average, compared to previous years.

What is the average lender yield trend over time?

In [80]:
```
#Line chart showing average lender yield trend over time
sns.relplot(data=loan_clean, x='loan_created_year', y='LenderYield', kind="line", height
#set y axis to start from Zero
plt.ylim(0,0.25)
plt.title('Lender Yield trend per year', fontsize=22);
```



Before 2006, the average lender yield was less than 0.1 percent. It peaked at about 0.17 in 2006, slightly dropped in 2007 then steadily increased till 2011. However, as at 2014, the average lender yield had dropped to pre-2006 levels. This is great for loan clients but discouraging for loan investors.

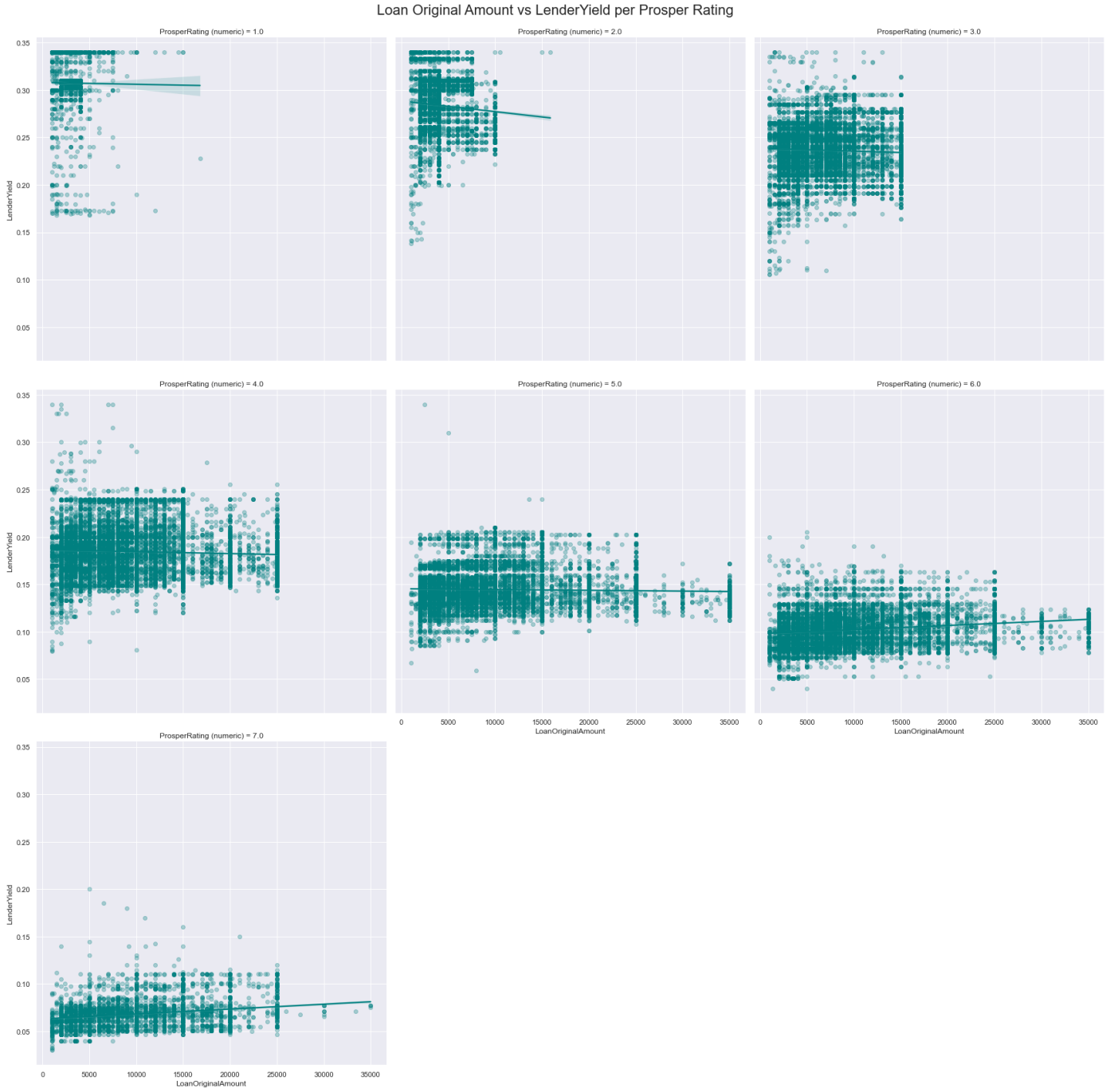
Prosper Company will need to balance the expectations of both clients and investors as none can do without the other.

Multivariate Exploration

In this section, i investigate relationships between three or more variables in the prosper loan dataset. I use this to understand results from previous setions and further investigate this dataset.

How does the prosper rating affect both the loaned amount and the lender's yield?

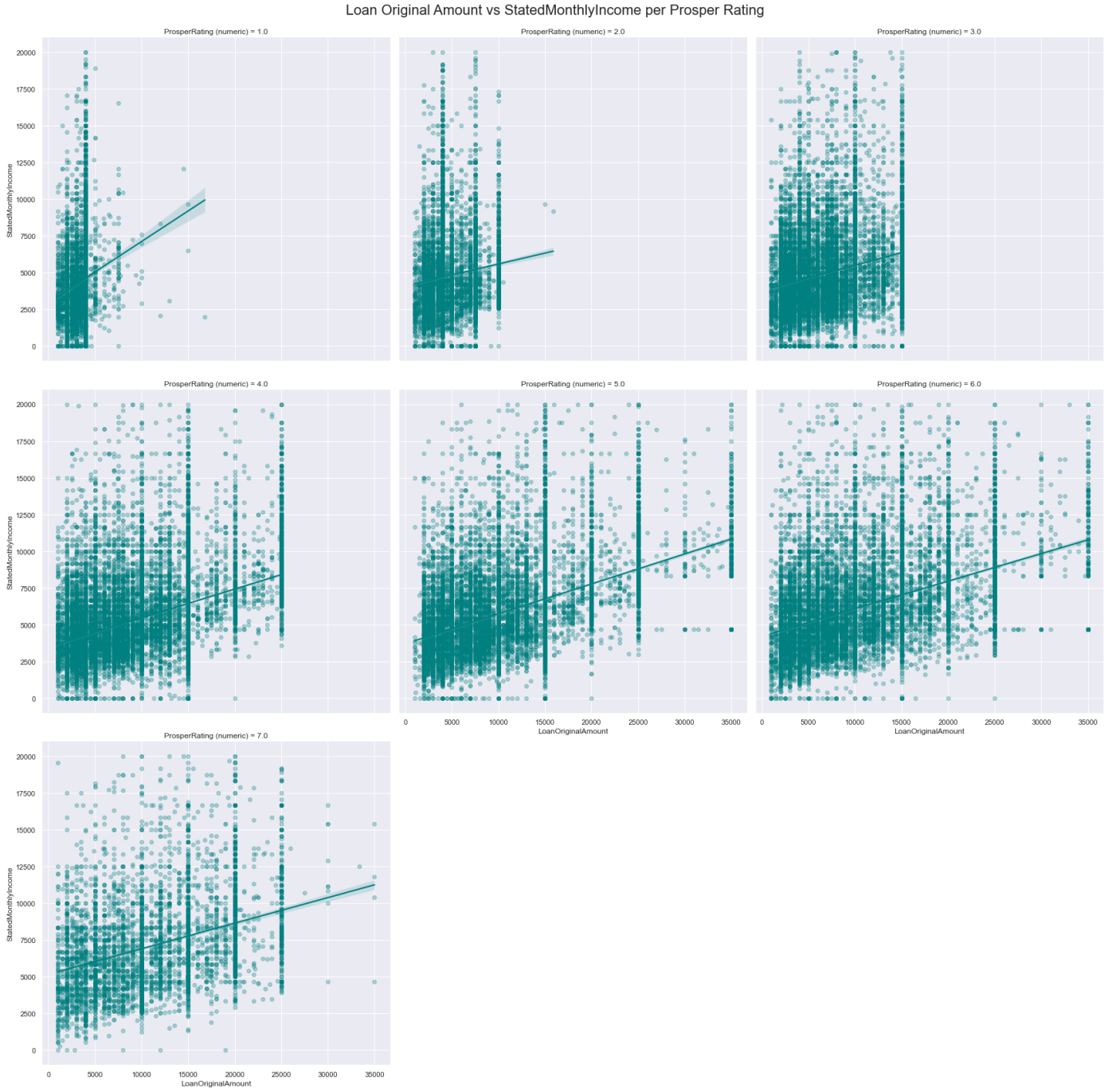
In [81]:
```
#create a facet grid to accommodate the graph and legend
g= sns.FacetGrid(data=loan_clean, col='ProsperRating (numeric)', col_wrap=3, height=8)

#plot scatter graph
g.map(sns.regplot, 'LoanOriginalAmount', 'LenderYield', scatter_kws={'alpha': 0.3}, colo

#display graph labels
plt.suptitle('Loan Original Amount vs LenderYield per Prosper Rating', fontsize=22)
plt.subplots_adjust(top=0.95);
```



A negative correlation between the loan amount and the lender's yield is observed in prosper ratings 1 - 5 and a positive correlation in prosper ratings 6 and 7. This matrix shows that loans of individuals with rating 6 and 7 are the best for loan investment opportunities.

Does the prosper rating affect both the loaned amount and the lender's yield?

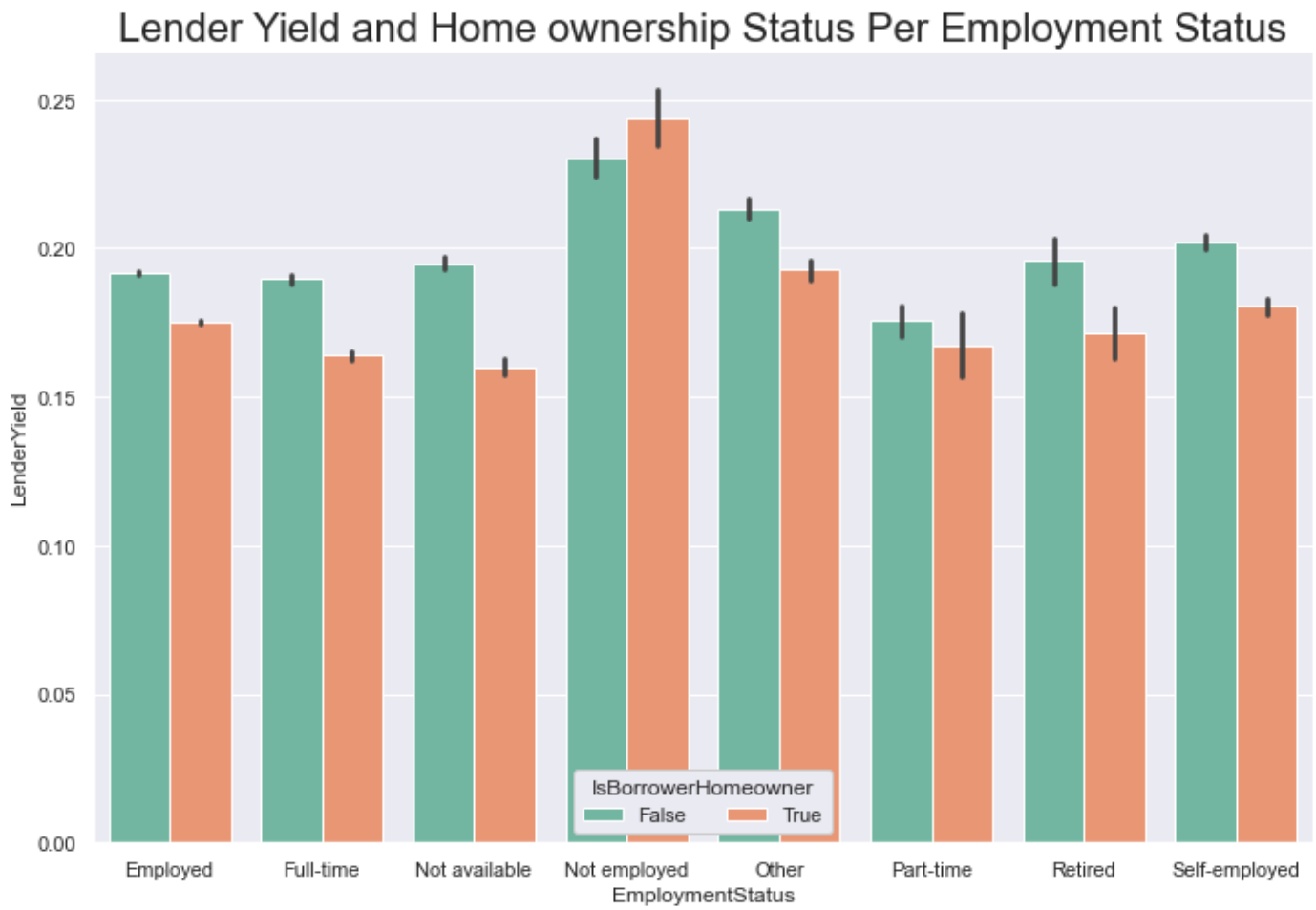
In [82]:
```
#create a facet grid to accommodate the graph and legend
g= sns.FacetGrid(data=loan_clean, col='ProsperRating (numeric)', col_wrap=3, height=8)

#plot scatter graph
g.map(sns.regplot, 'LoanOriginalAmount', 'StatedMonthlyIncome', scatter_kws={'alpha': 0.

#display graph labels
plt.suptitle('Loan Original Amount vs StatedMonthlyIncome per Prosper Rating', fontsize=
plt.subplots_adjust(top=0.95);
```



Across all prosper rating levels, there is a positive correlation between the loaned amount and the stated monthly income. This implies that the higher an individual's income, the higher the loan that can be possibly accessed

What is the lender yield like for the employed and unemployed home owners and none homeowners?

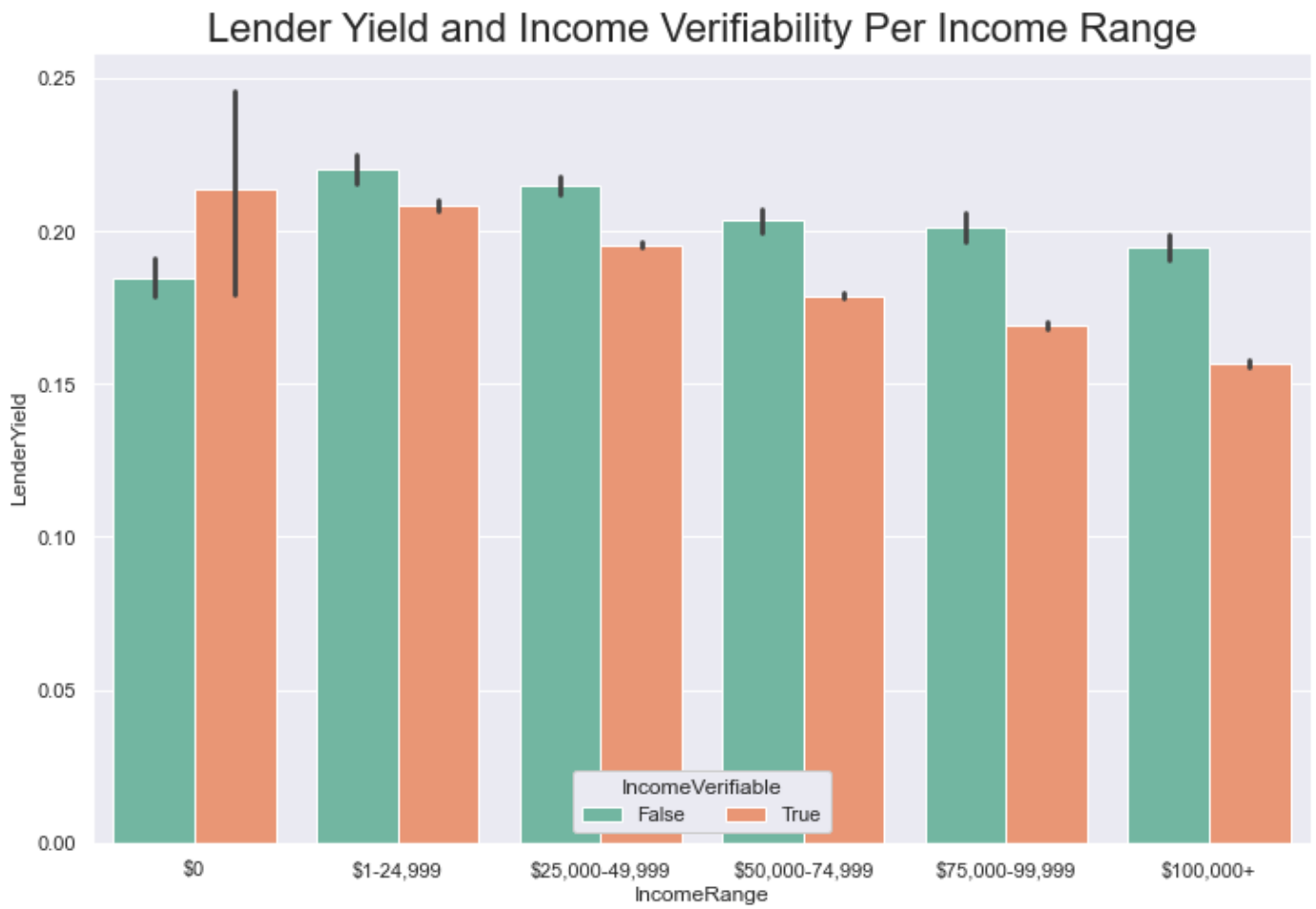
In [84]:
```
#Clustered barchart exploring lender yield, employment status and home ownership.
ax= sns.barplot(data = loan_clean, x= 'EmploymentStatus', y= 'LenderYield', hue='IsBorro
ax.legend(loc=8, ncol = 3, framealpha=1, title = 'IsBorrowerHomeowner' )
plt.title('Lender Yield and Home ownership Status Per Employment Status', fontsize=22);
```



The not employed individuals are actually generating a higher lenders yield because they are collecting smaller amounts which generally has higher rates on them. In all categories except the unemployed, home owners are generating a lower lenders yield compared to their counterparts who do not own homes

What is the lender yield like across the income ranges for both verifiable and unverifiable incomes ?

In [85]:
```
#Clustered barchart exploring lender yield, income range and income verifiability.
ax= sns.barplot(data = loan_clean, x= 'IncomeRange', y= 'LenderYield', hue='IncomeVerifi
ax.legend(loc=8, ncol = 3, framealpha=1, title = 'IncomeVerifiable' )
plt.title('Lender Yield and Income Verifiability Per Income Range', fontsize=22);
```

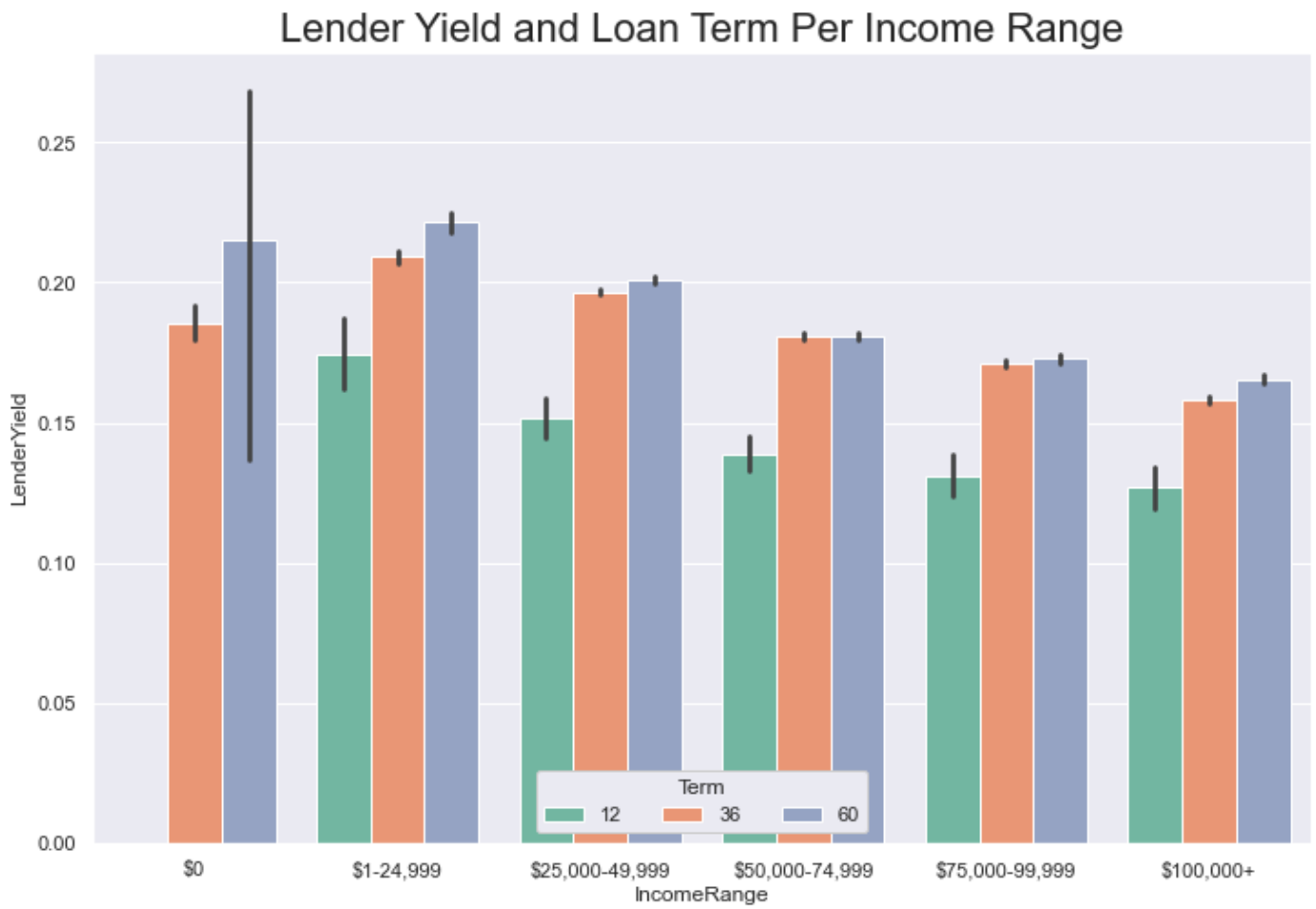


The highest lender's yield was observed in the 'unemployed with no verifiable income' group. For those with verifiable income, the $0 group recorded the highest lender's yield.

This confirms that looking at only the lender's yield as a metric for investment in a loan may not generate the best output possible. A lower percentage on a higher loan amount may give better investment returns.

Does a relationship between lender's yield, income range, loan term and income verificability exist?

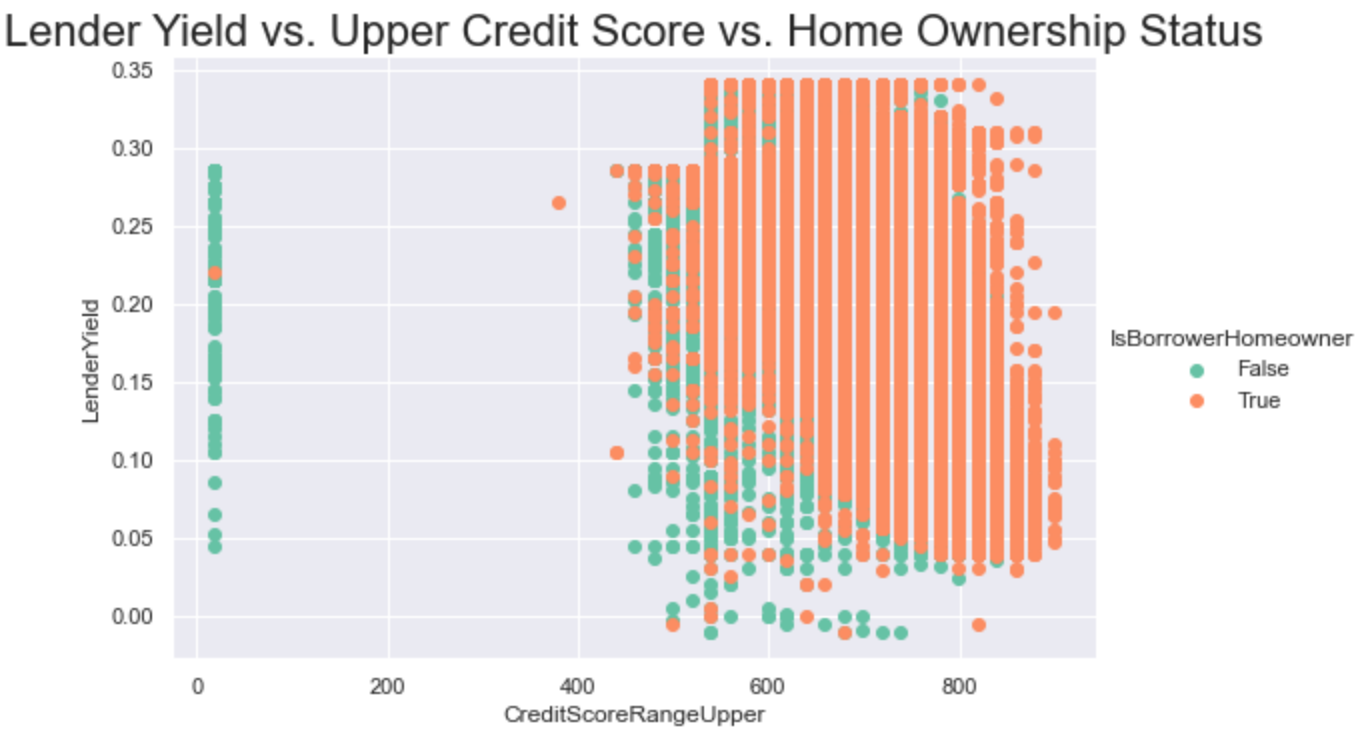
In [87]:
```
#Clustered barchart exploring lender yield, income range, loan term and income verifiabi
ax= sns.barplot(data = loan_clean, x= 'IncomeRange', y= 'LenderYield', hue='Term' )
ax.legend(loc=8, ncol = 3, framealpha=1, title = 'Term' )
plt.title('Lender Yield and Loan Term Per Income Range', fontsize=22);
```



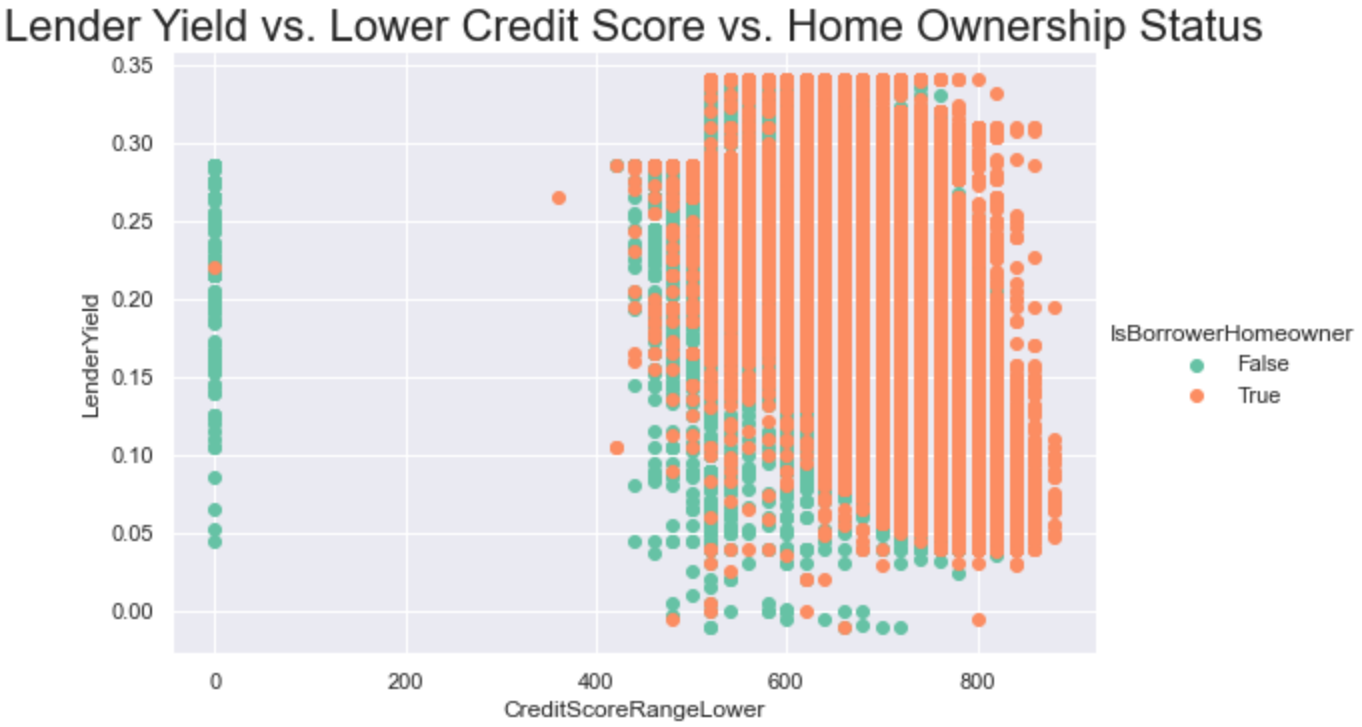
Clients earning between 25000 dollars and over 100000 dollars require shorter termed loans, although this reduces the lender yield owed.

How does owning a home and having a great credit score affect lender's yield?

In [88]:
```
#Scatter plot showing relationships existing between lender yield, home ownership and up
g = sns.FacetGrid(data = loan_clean, aspect=1.5, height=5, hue='IsBorrowerHomeowner')
g.map(plt.scatter, 'CreditScoreRangeUpper', 'LenderYield'  )
g.add_legend(title='IsBorrowerHomeowner')
plt.title('Lender Yield vs. Upper Credit Score vs. Home Ownership Status', fontsize=22);
```



In [89]:
```
#Scatter plot showing relationships existing between lender yield, home ownership and Lo
g = sns.FacetGrid(data = loan_clean, aspect=1.5, height=5, hue='IsBorrowerHomeowner')
g.map(plt.scatter, 'CreditScoreRangeLower', 'LenderYield'  )
g.add_legend(title='IsBorrowerHomeowner')
plt.title('Lender Yield vs. Lower Credit Score vs. Home Ownership Status', fontsize=22);
```



For both upper and lower credit score, owning a home and having a high credit score range was associated with a better lender yield, compared to individuals who do not own homes and have a lower credit range score.

This can be the first metric wich can identify better output for the investor and Prosper Company.

Limitation

- Due to the large nature of the dataset, finding the right information was a difficult process
- The dataset did not contain enough information on repaid loans. This would have helped me further my research in other important areas.

Conclusions

I analysed the prosper loan dataset and discovered the following insights:

- The highest possible yield on any loan is 0.492500% on the loaned amount. 0.3% has been most earned by investors, followed closely by 0.15%.
- On some loans, investors earned -0.010%. This means that as an investor, one could end up with losses.
- Individuals who earn between 25,000 and 75,000 dollars have most access to prosper loans.
- The highest rated loan collection purpose was Debt consolidation.
- 92 percent of individuals given a prosper loan have a verifiable source of income.
- An investor is likely to have a lower percentage yield as the monthly income of the client increases.
- Home owners are observed to collect significantly higher loan amounts on average, compared to non home owners, however, getting a high loan amount is still possible, without owning a home.
- High risk rated(1.0) individuals were given significantly lower loan amounts compared to higher scored individuals.
- The 36 month loan duration is prefered by individuals across all income ranges.
- In 2009, the average prosper rating was above 4.0. This value dropped in 2010 through 2012. It started rising in 2013 and hit its highest in 2014.
- Before 2006, the average lender yield was less than 0.1 percent. It peaked at about 0.17 in 2006, slightly dropped in 2007 then steadily increased till 2011. However, as at 2014, the average lender yield had dropped to pre-2006 levels.
- The loans of individuals with prosper rating 6 and 7 are the best for loan investment opportunities.
- The higher an individual's income, the higher the loan amount that can be possibly accessed.
- Using the lender's yield as the only metric for investment in a loan may not generate the best output possible.
- For both upper and lower credit score, owning a home and having a high credit score range was associated with a better lender yield, compared to individuals who do not own homes and have a lower credit range score.

The lender yield as a single metric may not inform an investor about the best prosper loan to invest in. Combining the lender yield with other metrics such as credit score, prosper rating, income level and employment status will help an investor make better choices on what loan to invest in and help the prosper company fund low risk loans.

In [90]:
```
loan_clean.to_csv('loan_clean.csv', index=False,  encoding = 'utf-8')
```

In [5]:
```
#!jupyter nbconvert --to webpdf --allow-chromium-download Part_1_exploration .ipynb
```