Prosper Loan Data Exploration

by Nathaniel Essilfie-Conduah

Introduction

> This project explores and explains the loan data sets from prosper. The data will be assessed explored and explained with visuals. Prosper is a peer to peer lending platform based in San Francisco and offers personal loans for borrowers with good credit (source:https://www.nerdwallet.com/reviews/loans/personal-loans/prosper-personal-loans)
>
> The focus for this data analysis is to know what factors influence or affect a loan's outcome application. What also affects a borrower's Annual Percentage rate (APR) or interest rate and differences between loans as a result of the amount.

Preliminary Wrangling

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

%matplotlib inline
```

In [2]:
```
# Loading the dataset into a pandas dataframe, print statistics
Prosper_df = pd.read_csv('prosperLoanData.csv')
```

In [3]:
```
# Overview of data
Prosper_df.head()
```

Out[3]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	LoanStatus
0	1021339766868145413AB3B	193129	2007-08-26 19:09:29.263000000	C	36	Completed
1	10273602499503308B223C1	1209647	2014-02-27 08:28:07.900000000	NaN	36	Current
2	0EE9337825851032864889A	81716	2007-01-05 15:00:47.090000000	HR	36	Completed
3	0EF5356002482715299901A	658116	2012-10-22 11:02:35.010000000	NaN	36	Current
4	0F023589499656230C5E3E2	909464	2013-09-14 18:38:39.097000000	NaN	36	Current

5 rows × 81 columns

In [4]:
```
Prosper_df.shape
```

Out[4]:
```
(113937, 81)
```

In [5]:
```
Prosper_df.dtypes
```

Out[5]:
```
ListingKey                     object
ListingNumber                   int64
ListingCreationDate            object
CreditGrade                    object
Term                            int64
                                ...
PercentFunded                 float64
Recommendations                 int64
InvestmentFromFriendsCount      int64
InvestmentFromFriendsAmount   float64
Investors                       int64
Length: 81, dtype: object
```

In [6]:
```
list(Prosper_df )
```

```
Out[6]: ['ListingKey',
 'ListingNumber',
 'ListingCreationDate',
 'CreditGrade',
 'Term',
 'LoanStatus',
 'ClosedDate',
 'BorrowerAPR',
 'BorrowerRate',
 'LenderYield',
 'EstimatedEffectiveYield',
 'EstimatedLoss',
 'EstimatedReturn',
 'ProsperRating (numeric)',
 'ProsperRating (Alpha)',
 'ProsperScore',
 'ListingCategory (numeric)',
 'BorrowerState',
 'Occupation',
 'EmploymentStatus',
 'EmploymentStatusDuration',
 'IsBorrowerHomeowner',
 'CurrentlyInGroup',
 'GroupKey',
 'DateCreditPulled',
 'CreditScoreRangeLower',
 'CreditScoreRangeUpper',
 'FirstRecordedCreditLine',
 'CurrentCreditLines',
 'OpenCreditLines',
 'TotalCreditLinespast7years',
 'OpenRevolvingAccounts',
 'OpenRevolvingMonthlyPayment',
 'InquiriesLast6Months',
 'TotalInquiries',
 'CurrentDelinquencies',
 'AmountDelinquent',
 'DelinquenciesLast7Years',
 'PublicRecordsLast10Years',
 'PublicRecordsLast12Months',
 'RevolvingCreditBalance',
 'BankcardUtilization',
 'AvailableBankcardCredit',
 'TotalTrades',
 'TradesNeverDelinquent (percentage)',
 'TradesOpenedLast6Months',
 'DebtToIncomeRatio',
 'IncomeRange',
 'IncomeVerifiable',
 'StatedMonthlyIncome',
 'LoanKey',
 'TotalProsperLoans',
 'TotalProsperPaymentsBilled',
 'OnTimeProsperPayments',
 'ProsperPaymentsLessThanOneMonthLate',
 'ProsperPaymentsOneMonthPlusLate',
 'ProsperPrincipalBorrowed',
 'ProsperPrincipalOutstanding',
 'ScorexChangeAtTimeOfListing',
 'LoanCurrentDaysDelinquent',
 'LoanFirstDefaultedCycleNumber',
 'LoanMonthsSinceOrigination',
 'LoanNumber',
 'LoanOriginalAmount',
```

```
'LoanOriginationDate',
'LoanOriginationQuarter',
'MemberKey',
'MonthlyLoanPayment',
'LP_CustomerPayments',
'LP_CustomerPrincipalPayments',
'LP_InterestandFees',
'LP_ServiceFees',
'LP_CollectionFees',
'LP_GrossPrincipalLoss',
'LP_NetPrincipalLoss',
'LP_NonPrincipalRecoverypayments',
'PercentFunded',
'Recommendations',
'InvestmentFromFriendsCount',
'InvestmentFromFriendsAmount',
'Investors']
```

What is the structure of your dataset?

> There are 81 coulmns and 113937 entries. The variables in the data set consists of floats, objects, and integers

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

> I am interested in finding out the factors that affects one's eligibility to a certain amount of loan

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

> I expect the Credit score, Borrower APR, employment status, Estimated return and the loan Term to have major effects

Define

The listing category here is numeric. From the data dictionary we need to replace it with 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
```

Code

In [7]:

```
Prosper_df['ListingCategory (numeric)'] = Prosper_df['ListingCategory (numeric)'].
    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'})
```

Test

In [8]:

```
Prosper_df['ListingCategory (numeric)'].value_counts()
```

Out[8]:

```
Debt Consolidation    58308
Not Available         16965
Other                 10494
Home Improvement       7433
Business               7189
Auto                   2572
Personal Loan          2395
Household Expenses     1996
Medical/Dental         1522
Taxes                   885
Large Purchases         876
Wedding Loans           771
Vacation                768
Student Use             756
Motorcycle              304
Engagement Ring         217
Baby&Adoption           199
Cosmetic Procedure       91
Boat                     85
Green Loans              59
RV                       52
Name: ListingCategory (numeric), dtype: int64
```

Define

Renaming the ListingCategory (numeric) to ListingCategory

Code

```
In [9]: Prosper_df.rename(columns = {'ListingCategory (numeric)':'ListingCategory'}, inpla
```

Test

```
In [10]: Prosper_df['ListingCategory']
```

```
Out[10]: 0              Not Available
1           Home Improvement
2              Not Available
3                 Motorcycle
4           Home Improvement
                  ...
113932    Debt Consolidation
113933                 Other
113934    Debt Consolidation
113935      Home Improvement
113936    Debt Consolidation
Name: ListingCategory, Length: 113937, dtype: object
```

Define

We can tell the names of all the columns within the data set by using the list function. Some selected columns will be used for the data analysis

Code

```
In [11]: # Selecting columns to be used for the data analysis
Prosper_clean = Prosper_df.copy()
Prosper_clean = Prosper_clean[['ListingKey',  'ListingCreationDate', 'Term', 'Loan
```

```
In [12]: # Checking to see the new list of columns for our data
Prosper_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 17 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   ListingKey                  113937 non-null  object 
 1   ListingCreationDate         113937 non-null  object 
 2   Term                        113937 non-null  int64  
 3   LoanStatus                  113937 non-null  object 
 4   BorrowerAPR                 113912 non-null  float64
 5   LenderYield                 113937 non-null  float64
 6   EstimatedReturn             84853 non-null   float64
 7   ListingCategory             113937 non-null  object 
 8   BorrowerState               108422 non-null  object 
 9   EmploymentStatus            111682 non-null  object 
 10  IsBorrowerHomeowner         113937 non-null  bool   
 11  CreditScoreRangeLower       113346 non-null  float64
 12  IncomeRange                 113937 non-null  object 
 13  MonthlyLoanPayment          113937 non-null  float64
 14  LoanMonthsSinceOrigination  113937 non-null  int64  
 15  LoanOriginalAmount          113937 non-null  int64  
 16  Recommendations             113937 non-null  int64  
dtypes: bool(1), float64(5), int64(4), object(7)
memory usage: 14.0+ MB
```

Test

In [13]:
```
Prosper_clean.shape
```

Out[13]:
```
(113937, 17)
```

The new data frame which is the Prosper_clean has 113937 rows and 17 columns. I chose 17 columns out of the 81 because I believe these variables are very crucial and influential with respect to one's laon application outcome

In [14]:
```
Prosper_clean
```

Out[14]:

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAP
0	1021339766868145413AB3B	2007-08-26 19:09:29.263000000	36	Completed	0.1651
1	10273602499503308B223C1	2014-02-27 08:28:07.900000000	36	Current	0.1201
2	0EE9337825851032864889A	2007-01-05 15:00:47.090000000	36	Completed	0.2826
3	0EF5356002482715299901A	2012-10-22 11:02:35.010000000	36	Current	0.1252
4	0F023589499656230C5E3E2	2013-09-14 18:38:39.097000000	36	Current	0.2461
...	...	...	...	...	
113932	E6D9357655724827169606C	2013-04-14 05:55:02.663000000	36	Current	0.2235
113933	E6DB353036033497292EE43	2011-11-03 20:42:55.333000000	36	FinalPaymentInProgress	0.1322
113934	E6E13596170052029692BB1	2013-12-13 05:49:12.703000000	60	Current	0.2398
113935	E6EB3531504622671970D9E	2011-11-14 13:18:26.597000000	60	Completed	0.2840
113936	E6ED3600409833199F711B7	2014-01-15 09:27:37.657000000	36	Current	0.1318

113937 rows × 17 columns

In [15]:
```
Prosper_clean.sample(50)
```

Out[15]:

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAPR	Lende
49076	B2CE359699018428779CC79	2013-12-16 15:09:54.180000000	36	Current	0.21342	0
40631	89A7338987296234O343412	2007-05-16 10:04:00.417000000	36	Completed	0.16697	0
68396	360E35991455766782A3278	2013-12-30 06:11:18.157000000	60	Current	0.23267	0
20618	A0DB359586065129248F394	2013-11-26 07:34:01.270000000	36	Current	0.11563	0
63113	0E973419172256137F5843C	2008-04-21 12:38:15.087000000	36	Completed	0.21679	0
41303	DDFA3551319611630F660F7	2012-07-05 07:45:04.693000000	36	Completed	0.12427	0
38058	8B073594787175427D1B5D3	2013-11-08 00:52:07.857000000	36	Completed	0.22108	0
107116	64DF3502071131683DAA28F	2010-12-09 20:45:39.833000000	36	Chargedoff	0.35858	0
15069	07E93549515845549B0901B	2012-06-12 13:31:02.103000000	36	Current	0.25259	0
44310	4D80356160226819813119A	2012-10-16 19:27:01.433000000	36	Current	0.35797	0
61153	C38D3391846253249F0B194	2007-06-09 17:58:38.297000000	36	Chargedoff	0.11696	0
73137	0B893587807498875F4F261	2013-08-30 14:05:05.930000000	60	Current	0.24589	0
97502	AF34349231479421495B7C4	2010-08-17 23:43:20.693000000	36	Completed	0.08591	0
64	0F1B35892502944590155D8	2013-09-26 11:16:17.373000000	36	Current	0.30899	0
57569	5190360226194186933061C	2014-02-04 12:46:43.637000000	36	Current	0.12081	0
20979	F748355002262941991D1B4	2012-06-22 13:31:02.793000000	60	Completed	0.17849	0
101851	6DD33555558966501A830DE	2012-08-23 08:40:00.833000000	60	Current	0.15936	0
84730	3D5D3530843097281AC7D1B	2011-11-01 09:22:01.730000000	36	Current	0.35244	0
14166	0F983420346910834EE10F7	2008-05-08 11:16:38.660000000	36	Completed	0.16461	0
10843	13A435515238528345F5783	2012-07-04 05:08:26.420000000	36	Completed	0.26395	0
2351	2BB73515078872837E2F2D6	2011-05-08 16:03:02.437000000	60	Defaulted	0.27322	0
112637	CB7D3484329678355B6E913	2010-05-20 15:15:38.670000000	36	Completed	0.08591	0
113693	E2193428655642974D34A0B	2008-07-31 21:40:08.397000000	36	Defaulted	0.27103	0

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAPR	Lender
88920	AE6035656431517591370FF	2012-12-05 12:45:38.717000000	36	Chargedoff	0.35797	0
69536	40023493625216988507E8E	2010-08-24 15:52:59.587000000	36	Completed	0.37699	0
84469	21C93396540024942BAC5BA	2007-08-13 06:15:17.180000000	36	Chargedoff	0.21480	0
38433	735E35343319087267136F6	2011-12-27 06:37:50.830000000	60	Completed	0.32989	0
41357	789B358308304267155EB14	2013-07-16 14:26:15.890000000	60	Current	0.17522	0
59344	FF823575742981280A45E99	2013-04-20 06:15:27.967000000	36	Current	0.14857	0
14492	0F75342572091879597E09A	2008-07-17 12:00:08.270000000	36	Chargedoff	0.37453	0
8548	3F6336027715644199F2CA2	2014-02-12 20:03:06.487000000	36	Current	0.22505	0
34553	92763600678326143496CEB6	2014-02-01 15:28:14.737000000	60	Current	0.21166	0
11602	66FD3508735355060137253	2011-02-22 12:07:02.773000000	36	Completed	0.35643	0
47591	8E26358442092315009690EE	2013-07-21 17:55:45.217000000	36	Past Due (31-60 days)	0.23121	0
74929	B35F33650920601385A7AA2	2006-05-04 08:32:53.953000000	36	Completed	0.24502	0
75162	41C13597651905858BB97A8	2013-12-10 09:03:49.707000000	36	Current	0.18275	0
43809	969F3415964018461D08A37	2008-03-11 07:30:41.930000000	36	Defaulted	0.23504	0
77344	F6103401029183293AF753C	2007-10-02 20:40:11.747000000	36	Completed	0.15932	0
34655	ADF935831311942917D0FAE	2013-06-24 11:18:29.130000000	60	Current	0.13942	0
24936	BA20360083955710791A00A	2014-01-16 14:49:37.600000000	36	Current	0.20217	0
65484	CAF53501939151112454B99	2010-12-21 05:51:26.503000000	36	Completed	0.22872	0
57939	5F423408728042901212E40	2007-12-26 17:38:31.127000000	36	Chargedoff	0.18816	0
84390	21AB3584259963178677003	2013-07-22 14:22:34.673000000	60	Current	0.21566	0
102421	AB86354886146383363D794	2012-06-08 13:24:14.097000000	36	Current	0.26681	0
38578	76ED35521199666669EBBD0	2012-07-02 08:46:50	60	Current	0.17849	0

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAPR	Lende
35205	B57B3546436796203A888E4	2012-05-15 09:21:27.857000000	60	Current	0.31375	(
72364	F9333496305196102326772	2010-10-04 22:49:19.860000000	36	Completed	0.38058	(
19072	1D55360048100111156E7C4	2014-01-31 16:57:55.937000000	36	Current	0.30848	(
78286	D86A3407108240307D9BDD2	2007-11-25 12:03:34.943000000	36	Defaulted	0.10991	(
9774	16CC354525216306170EF29	2012-05-02 18:23:30.977000000	36	Current	0.21372	(

In [16]:
```
# Assessing the Recommendation column to identify the values within it.

Prosper_clean.Recommendations
```

Out[16]:
```
0         0
1         0
2         0
3         0
4         0
         ..
113932    0
113933    0
113934    0
113935    0
113936    0
Name: Recommendations, Length: 113937, dtype: int64
```

In [17]:
```
Prosper_clean.Recommendations.value_counts()
```

Out[17]:
```
0     109678
1       3516
2        568
3        108
4         26
5         14
9          6
7          5
6          4
8          3
18         2
16         2
14         1
21         1
24         1
19         1
39         1
Name: Recommendations, dtype: int64
```

In [18]:
```
#Checking to see if there are any null values within the dataset

Prosper_clean.isnull()
```

Out[18]:

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAPR	LenderYield	Estimated
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
...	...	...	...	...	...	...	
113932	False	False	False	False	False	False	
113933	False	False	False	False	False	False	
113934	False	False	False	False	False	False	
113935	False	False	False	False	False	False	
113936	False	False	False	False	False	False	

113937 rows × 17 columns

In [19]:

```
Prosper_clean.isnull().sum()
```

Out[19]:

```
ListingKey                     0
ListingCreationDate            0
Term                           0
LoanStatus                     0
BorrowerAPR                   25
LenderYield                    0
EstimatedReturn            29084
ListingCategory                0
BorrowerState               5515
EmploymentStatus            2255
IsBorrowerHomeowner            0
CreditScoreRangeLower        591
IncomeRange                    0
MonthlyLoanPayment             0
LoanMonthsSinceOrigination     0
LoanOriginalAmount             0
Recommendations                0
dtype: int64
```

Define

The table depicts EstimatedReturn, BorrowerState, EmploymentStatus, CreditScoreRangeLower having null values in the dataset.

Code

In [20]:

```
# drop null values in BorrowerState, EmploymentStatus, CreditScoreRangeLower and Le

Prosper_clean= Prosper_clean.dropna(subset = ['BorrowerState', 'CreditScoreRangeLo
Prosper_clean.isnull().sum()
```

Out[20]:
```
ListingKey                         0
ListingCreationDate                0
Term                               0
LoanStatus                         0
BorrowerAPR                        0
LenderYield                        0
EstimatedReturn                22701
ListingCategory                    0
BorrowerState                      0
EmploymentStatus                   0
IsBorrowerHomeowner                0
CreditScoreRangeLower              0
IncomeRange                        0
MonthlyLoanPayment                 0
LoanMonthsSinceOrigination         0
LoanOriginalAmount                 0
Recommendations                    0
dtype: int64
```

Test

In [21]:
```
Prosper_clean.EstimatedReturn
```

Out[21]:
```
0             NaN
1         0.05470
2             NaN
3         0.06000
4         0.09066
           ...
113932    0.09500
113933    0.08070
113934    0.08578
113935    0.15950
113936    0.06081
Name: EstimatedReturn, Length: 107554, dtype: float64
```

In [22]:
```
Prosper_clean.sample(50)
```

Out[22]:

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAPR	Lender'
103842	6F6D3574428755227FFAA0E	2013-03-20 07:31:30.750000000	36	Completed	0.22354	0.
74477	873D3579067992952ED51D7	2013-05-14 00:26:56.320000000	36	Current	0.23530	0.
65693	71EB34297447580213C9A16	2008-09-03 15:14:58.557000000	36	Completed	0.37453	0.
102094	40D83595302073547B7E132	2013-11-29 07:39:58.707000000	36	Current	0.07922	0.
22726	EBE835209935108762ED006	2011-07-26 15:40:21.853000000	36	Completed	0.22362	0.
32969	0071352742552723510F6E5	2011-10-06 14:14:50.817000000	12	Completed	0.34105	0.
64700	0CFD36035364614796192E7	2014-02-20 18:02:18.483000000	60	Current	0.16686	0.
86486	43533570658650186 5E6DE1	2013-02-08 08:31:22.550000000	36	Current	0.08325	0.
68751	FFAC354811044392378CE9F	2012-06-06 10:42:24.437000000	36	Chargedoff	0.35797	0.
59375	7585358042149754 82EE8ED	2013-06-06 12:55:43.050000000	60	Current	0.33040	0.
108875	265D3400873933759E194FC	2007-09-26 05:09:42.640000000	36	Completed	0.21739	0.
2779	0D7F35491337371222F9914	2012-06-13 12:39:02.337000000	36	Completed	0.12427	0.
21656	64703571975073324F61D1B	2013-03-05 10:36:30.060000000	36	Current	0.21025	0.
68444	39153580061577347A97B6E	2013-05-30 08:16:35.607000000	36	Current	0.31790	0.
28894	2F993590951510863B8498A	2013-10-01 08:33:08.517000000	36	Current	0.26917	0.
71649	88D3352879939554124CC7E	2011-10-21 18:51:57.210000000	36	Completed	0.35132	0.
52729	DB69338674499453078768F	2007-04-05 07:21:09.943000000	36	Completed	0.15713	0.
5146	9CE4354451015381412AF81	2012-04-12 18:14:13.200000000	36	Completed	0.26681	0.
87226	E7683585689045898 7472E7	2013-08-07 18:37:01.863000000	36	Current	0.31790	0.
83011	710D3597043824784958829	2013-12-11 09:50:51.573000000	60	Current	0.16662	0.
103483	E05C340450194922211414E	2007-11-15 10:06:24.147000000	36	Completed	0.16717	0.
47855	33793425663589312650BE4	2008-06-29 21:43:53.943000000	36	Completed	0.37453	0.
104094	0F933389844535057B95969	2007-05-17 16:17:01.303000000	36	Chargedoff	0.30564	0.

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAPR	Lender
29658	921035885758242923750B3	2013-09-18 08:43:53.493000000	36	Current	0.30899	0.
83769	CBB7355498735637628690A	2012-08-15 23:53:45.690000000	60	Past Due (1-15 days)	0.18545	0.
94652	F71F36034333911576B4B37	2014-03-05 04:43:01.950000000	36	Current	0.16259	0.
80202	F8BD3603093000950CED557	2014-02-25 12:24:24.993000000	36	Current	0.23375	0.
24412	77CB355440305520937F907	2012-08-05 17:57:32.697000000	36	Current	0.28339	0.
54641	FF013560637147541A27F10	2012-10-22 19:21:43.067000000	60	Chargedoff	0.28848	0.
45840	A4223583683051691A1D853	2013-06-30 13:17:53.767000000	36	Current	0.21945	0.
27649	638F357264549710416F390	2013-03-15 12:34:28.583000000	36	Current	0.13138	0.
34200	91DA3396741561245FF65F5	2007-08-12 13:49:28.760000000	36	Chargedoff	0.18927	0.
81749	431A34783231873755B7713	2010-03-12 10:03:07.870000000	36	Completed	0.11296	0.
23260	768834091131951903 8E1B0	2007-12-23 05:04:22.820000000	36	Completed	0.07799	0.
71243	C734339027946997216923 1	2007-05-22 08:42:45.047000000	36	Completed	0.14709	0.
105134	7FD036029863227068577FA	2014-03-01 09:41:12.093000000	36	Current	0.09065	0.
90654	A9543581484964768AD6794	2013-06-03 13:04:29.393000000	36	Completed	0.20053	0.
81122	A1C53599787103275A18710	2014-01-22 19:19:21.840000000	36	Current	0.09434	0.
54892	58A5354254300456572CDC3	2012-03-15 07:50:00.840000000	60	Current	0.27246	0.
2680	04F53389408045160C52C93	2007-05-20 20:22:14.123000000	36	Completed	0.30271	0.
26693	BCDF3515754796575769 9AC	2011-05-28 09:46:38.310000000	36	Current	0.13524	0.
50640	AD8E351096622253714C189	2011-03-20 06:16:07.277000000	36	Current	0.29510	0.
89200	7CD83604707943482929 8A7	2014-03-07 05:37:55.593000000	36	Current	0.26383	0.
80514	D76C3473897113041127AF1	2010-01-23 19:03:21.513000000	36	Completed	0.20716	0.
72109	C8233518807001260C1195F	2011-06-24 10:55:40.093000000	36	Current	0.13524	0.
48826	672D3596293311667AD1880	2013-12-07 17:22:17.060000000	36	Current	0.15223	0.

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAPR	Lender
62188	367A3585298531623483BFB	2013-08-07 15:23:58.423000000	60	Current	0.16499	0.
67193	701D34052057787615215F4	2007-11-16 07:21:59.810000000	36	Completed	0.18866	0.
76975	98593601409445431427F63	2014-01-29 19:23:11.143000000	36	Current	0.15223	0.
13115	48063404176553389366E24	2007-11-03 09:19:24.687000000	36	Completed	0.13705	0.

In [23]:
```
Prosper_clean.Term.value_counts()
```

Out[23]:
```
36    81395
60    24545
12     1614
Name: Term, dtype: int64
```

In [24]:
```
Prosper_clean.IncomeRange.value_counts()
```

Out[24]:
```
$25,000-49,999    31520
$50,000-74,999    30638
$100,000+         17175
$75,000-99,999    16737
$1-24,999          7040
Not displayed      3060
Not employed        781
$0                  603
Name: IncomeRange, dtype: int64
```

In [25]:
```
Prosper_clean.CreditScoreRangeLower.value_counts()
```

Out[25]:
```
680.0    16119
660.0    15995
700.0    15205
720.0    12643
640.0    11585
740.0     9076
760.0     6432
780.0     4505
620.0     3618
600.0     3032
800.0     2562
820.0     1355
520.0     1214
560.0     1025
540.0      941
580.0      874
840.0      547
500.0      264
860.0      194
480.0      174
460.0       72
0.0         69
880.0       26
440.0       24
420.0        3
Name: CreditScoreRangeLower, dtype: int64
```

In [26]:
```
Prosper_clean.LoanStatus.value_counts()
```

```
Out[26]: Current                     56576
         Completed                   33781
         Chargedoff                  10833
         Defaulted                    4091
         Past Due (1-15 days)          806
         Past Due (31-60 days)         363
         Past Due (61-90 days)         313
         Past Due (91-120 days)        304
         Past Due (16-30 days)         265
         FinalPaymentInProgress        205
         Past Due (>120 days)           16
         Cancelled                       1
         Name: LoanStatus, dtype: int64
```

```
In [27]: Prosper_clean.head()
```

Out[27]:

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAPR	LenderYield
0	1021339766868145413AB3B	2007-08-26 19:09:29.263000000	36	Completed	0.16516	0.1380
1	10273602499503308B223C1	2014-02-27 08:28:07.900000000	36	Current	0.12016	0.0820
2	0EE9337825851032864889A	2007-01-05 15:00:47.090000000	36	Completed	0.28269	0.2400
3	0EF5356002482715299901A	2012-10-22 11:02:35.010000000	36	Current	0.12528	0.0874
4	0F023589499656230C5E3E2	2013-09-14 18:38:39.097000000	36	Current	0.24614	0.1985

Define

Converting these columns into categories.

```
In [28]: score_ranges = [-1, 350, 540, 600, 700, 800, 900]


         score_names = ['Worse', 'Average', 'Satisfactory', 'Good', 'Very Good', 'Exception


         Prosper_clean.insert(loc=12, column="Credit_Score", value = pd.cut(Prosper_clean['
```

```
In [29]: Prosper_clean.head()
```

Out[29]:

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAPR	LenderYield
0	1021339766868145413AB3B	2007-08-26 19:09:29.263000000	36	Completed	0.16516	0.1380
1	10273602499503308B223C1	2014-02-27 08:28:07.900000000	36	Current	0.12016	0.0820
2	0EE9337825851032864889A	2007-01-05 15:00:47.090000000	36	Completed	0.28269	0.2400
3	0EF5356002482715299901A	2012-10-22 11:02:35.010000000	36	Current	0.12528	0.0874
4	0F023589499656230C5E3E2	2013-09-14 18:38:39.097000000	36	Current	0.24614	0.1985

In [30]:

```
ordinal_var_dict = {
                    'Term': [12,36,60],
                    'IncomeRange': ['Not employed', 'Not displayed', '$0', '$1-24,
                    'Credit_Score':['Worse', 'Average', 'Satisfactory', 'Good', 'Ve
                    'LoanStatus':['Cancelled', 'FinalPaymentInProgress', 'Past Due
                    }

for var in ordinal_var_dict:
    ordered_var = pd.api.types.CategoricalDtype(ordered = True,
                                                categories = ordinal_var_dict[var])
    Prosper_clean[var] = Prosper_clean[var].astype(ordered_var)
```

```
C:\Users\Owner\AppData\Local\Temp\ipykernel_14444\711241954.py:11: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  Prosper_clean[var] = Prosper_clean[var].astype(ordered_var)
C:\Users\Owner\AppData\Local\Temp\ipykernel_14444\711241954.py:11: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  Prosper_clean[var] = Prosper_clean[var].astype(ordered_var)
C:\Users\Owner\AppData\Local\Temp\ipykernel_14444\711241954.py:11: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  Prosper_clean[var] = Prosper_clean[var].astype(ordered_var)
C:\Users\Owner\AppData\Local\Temp\ipykernel_14444\711241954.py:11: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  Prosper_clean[var] = Prosper_clean[var].astype(ordered_var)
```

```
Prosper_clean.sample(50)
```

In [31]:

Out[31]:

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAPR	Lender
81202	DF223599554973329AD9245	2013-12-30 18:47:28.127000000	36	Current	0.09030	0
88261	9E443382199498395718BAB	2007-02-21 06:45:47.917000000	36	Completed	0.21480	0
38085	596B3576940477033D03CEE	2013-04-22 17:28:14.770000000	36	Current	0.15324	0
24034	F979355086482234878 2ABC	2012-06-23 14:34:20.753000000	60	Current	0.27462	0
76478	CAAA3510318600640168B98	2011-03-07 08:13:36.200000000	60	Completed	0.24763	0
105986	B6A3359364991197 10652B2	2013-11-14 07:40:37.707000000	36	Current	0.17090	0
9396	F4C6357009362362796C3A4	2013-02-07 10:06:54.953000000	36	Current	0.14348	0
47146	55753600796540668BA7102	2014-01-22 11:23:40.137000000	36	Current	0.14206	0
80662	4B823428322698920C7341D	2008-07-29 19:43:19.600000000	36	Completed	0.10285	0
40973	5FAE3598708813157C15049	2014-01-05 16:27:32.977000000	36	Current	0.17151	0
69905	D74B358150776158904CA67	2013-06-12 03:20:38.863000000	36	Current	0.13138	0
56948	74EA3424087917917B92E66	2008-06-12 12:37:47.467000000	36	Chargedoff	0.37453	0
29336	BF4B347031313195181A02B	2009-12-19 06:13:35.257000000	36	Completed	0.24163	0
107479	6E7033884382608652A1E6C	2007-05-04 18:20:54.563000000	36	Chargedoff	0.30564	0
73333	79633602264665196258308	2014-02-07 13:26:14.680000000	36	Current	0.31975	0
41226	E1143588901047579051734	2013-09-17 11:27:44.240000000	36	Current	0.26047	0
30303	332734017020176051CD316	2007-10-15 13:18:53.670000000	36	Completed	0.08935	0
36516	52F13602887215496C575DA	2014-02-12 15:37:04.547000000	36	Current	0.14243	0
86924	AC9E3583233748678D20112	2013-06-24 13:23:37.763000000	60	Current	0.33040	0
42691	D1DF3549828150961A20AC7	2012-06-08 15:11:41.607000000	36	Past Due (16-30 days)	0.35797	0
106159	DC0935522376299129E47CB	2012-07-05 13:50:01.633000000	36	Completed	0.28851	0
13140	0D2335382625943981E5D47	2012-01-26 16:25:27.427000000	36	Completed	0.28339	0

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAPR	Lender
13397	E2753599122255675220F6D	2013-12-31 08:01:19.683000000	36	Current	0.19501	0
28108	B7B435778293106810D549B	2013-04-28 15:32:05.040000000	60	Current	0.22283	0
28892	2F8935913718589497E8FF8	2013-10-15 15:19:03.400000000	36	Current	0.35356	0
95723	5B073581048692414C42559	2013-06-04 13:39:02.860000000	36	Current	0.28780	0
8230	9E3D3596327187261A0B119	2013-11-25 10:11:43.557000000	36	Current	0.28595	0
65925	D6923558358541695E416D3	2012-09-17 13:38:10.607000000	36	Current	0.12528	0
59434	ACF03564841207696BCDBEB	2012-12-02 14:02:57.330000000	36	Completed	0.32538	0
97685	B1373599951088414D8A7BB	2014-01-25 09:29:48.907000000	36	Current	0.30131	0
67822	87373515316013679292F2F	2011-05-23 11:37:12.433000000	36	Completed	0.30532	0
19954	6D0D359556124999958511D	2013-12-05 09:43:32.403000000	36	Current	0.22773	0
12902	0F2F3519307863404AA14F4	2011-06-30 01:16:32.040000000	36	Past Due (1-15 days)	0.34621	0
98954	256334154329703123AB4D7	2008-03-19 17:18:46.893000000	36	Defaulted	0.17677	0
55089	B79D3596272519632864E38	2013-12-06 09:01:30.147000000	36	Current	0.20524	0
94195	F0653418185931941A1E7DB	2008-04-18 17:21:17.450000000	36	Chargedoff	0.17677	0
1500	30C0358084995188901B36F	2013-06-07 08:03:22.830000000	36	Current	0.28032	0
31761	8DEF35849771146586102E4	2013-07-23 08:43:51.603000000	36	Current	0.28032	0
31338	85293411175280510D71C8B	2008-01-10 13:18:23.037000000	36	Completed	0.34550	0
35233	19603587314899245988A96	2013-08-13 11:14:38.937000000	60	Completed	0.14965	0
25567	06AE3525765886727036F33	2011-09-13 18:56:15.053000000	36	Completed	0.16056	0
72501	3C383505032733454FFF490	2011-01-12 08:08:09.800000000	36	Completed	0.20321	0
41532	F9FF3595342727955302689	2013-12-06 11:52:30.633000000	60	Current	0.18197	0
32874	008436022372007020491E5	2014-02-03 09:50:31.690000000	36	Current	0.12691	0
106077	CC503422661216443201FD3	2008-06-11 08:09:06.663000000	36	Defaulted	0.09186	0

	ListingKey	ListingCreationDate	Term	LoanStatus	BorrowerAPR	Lender
91439	42BE3593530416080C7532B	2013-11-09 12:56:37.443000000	36	Current	0.15223	0
100626	E08B3407566247108370AE0	2007-12-11 14:32:01.407000000	36	Completed	0.26762	0
57589	09E536001477674668E2C37	2014-01-14 11:44:28.573000000	60	Current	0.13636	0
110987	2B943561375831190E84942	2012-10-14 20:32:55.360000000	60	Current	0.17982	0
19952	6CFF35260950453573C75C6	2011-09-13 19:50:46.827000000	36	Current	0.30532	0

In [32]:
```
Prosper_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 107554 entries, 0 to 113936
Data columns (total 18 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   ListingKey                  107554 non-null  object
 1   ListingCreationDate         107554 non-null  object
 2   Term                        107554 non-null  category
 3   LoanStatus                  107554 non-null  category
 4   BorrowerAPR                 107554 non-null  float64
 5   LenderYield                 107554 non-null  float64
 6   EstimatedReturn             84853 non-null   float64
 7   ListingCategory             107554 non-null  object
 8   BorrowerState               107554 non-null  object
 9   EmploymentStatus            107554 non-null  object
 10  IsBorrowerHomeowner         107554 non-null  bool
 11  CreditScoreRangeLower       107554 non-null  float64
 12  Credit_Score                107554 non-null  category
 13  IncomeRange                 107554 non-null  category
 14  MonthlyLoanPayment          107554 non-null  float64
 15  LoanMonthsSinceOrigination  107554 non-null  int64
 16  LoanOriginalAmount          107554 non-null  int64
 17  Recommendations             107554 non-null  int64
dtypes: bool(1), category(4), float64(5), int64(3), object(5)
memory usage: 12.0+ MB
```

Univariate Exploration

> Will start by looking at the variables of interests in the dataframe such as:
>
> 1. Recommendation
> 2. Borrowers APR
> 3. Estimated returns
> 4. LoanOriginalAmount
> 5. Monthly Loan Payment

Estimated Returns

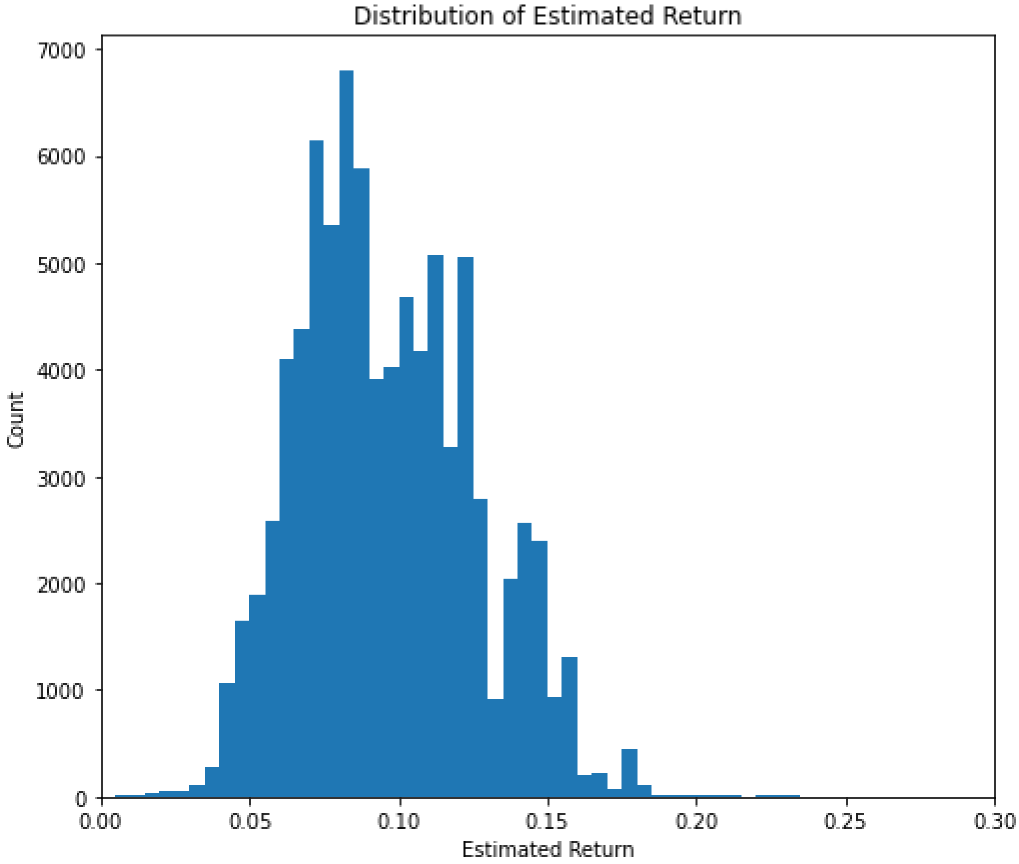
In [33]:
```
def x_y_t(xl,yl,title):
```

```
    plt.title(title)
    plt.xlabel(xl)
    plt.ylabel(yl)
```

In [34]:
```
binsize = 0.005
bins = np.arange(0, Prosper_clean.EstimatedReturn.max()+binsize, binsize)

plt.figure(figsize=[8, 7])
plt.hist(data = Prosper_clean, x= 'EstimatedReturn', bins = bins);
print(x_y_t('Estimated Return', 'Count', 'Distribution of Estimated Return'))
plt.xlim(0, 0.30)
```

```
None
```

Out[34]:
```
(0.0, 0.3)
```



In [35]:
```
# We are going to look out for the mean, std etc
Prosper_clean.EstimatedReturn.describe()
```

Out[35]:
```
count    84853.000000
mean         0.096068
std          0.030403
min         -0.182700
25%          0.074080
50%          0.091700
75%          0.116600
max          0.283700
Name: EstimatedReturn, dtype: float64
```

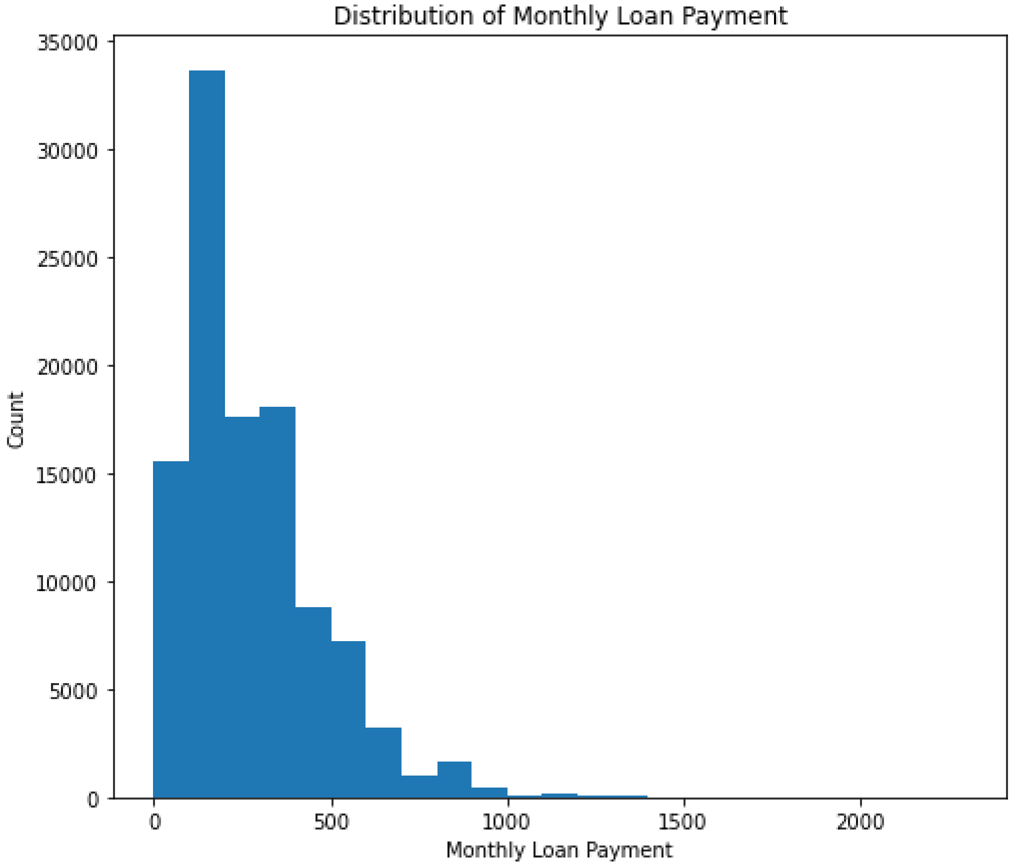
The Estimated return is between 0.05 to 0.17. An estimted return from 0.20 to 0.26 is very rare and hard to come by from the distribution

Monthly Payment

In [64]:
```
binsize = 100
bins = np.arange(0, Prosper_clean.MonthlyLoanPayment.max()+binsize, binsize)
```

```
plt.figure(figsize=[8, 7])
plt.hist(data = Prosper_clean, x= 'MonthlyLoanPayment', bins = bins);
print(x_y_t('Monthly Loan Payment', 'Count', 'Distribution of Monthly Loan Payment
```

None



In [37]:

```
Prosper_clean.MonthlyLoanPayment.describe()
```

Out[37]:

```
count    107554.000000
mean        277.678866
std         192.564168
min           0.000000
25%         136.980000
50%         226.205000
75%         376.700000
max        2251.510000
Name: MonthlyLoanPayment, dtype: float64
```

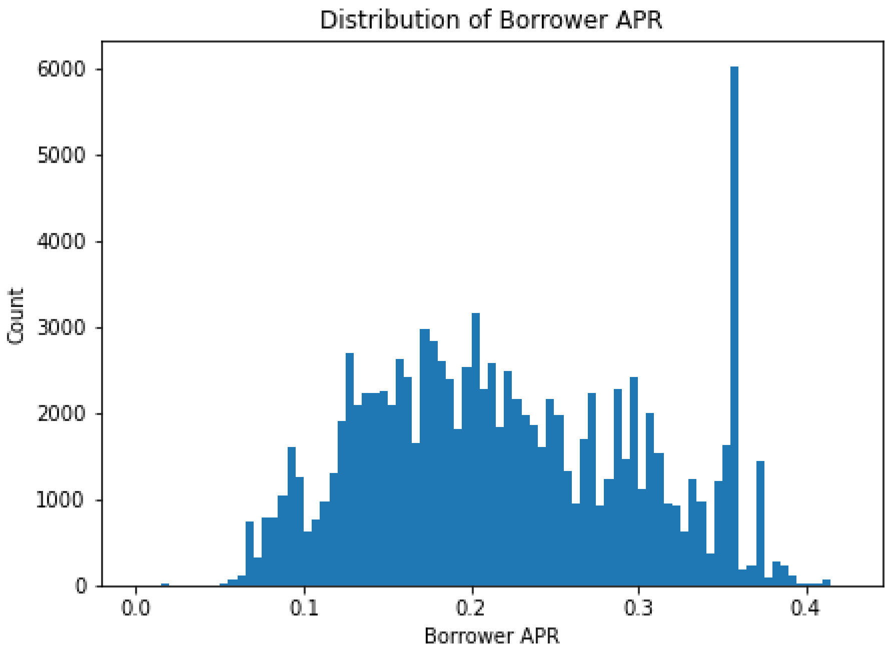
From the viualization you could tell that the amount of money paid monthly is within the 10's and 100's region. Not so much within the 1000's region. Also looking at the percentiles, much more people pay their monthly loan above a 136.98

Borrower APR

In [65]:

```
binsize = 0.005
bins = np.arange(0, Prosper_clean.BorrowerAPR.max()+binsize, binsize)

plt.figure(figsize=[7, 5])
plt.hist(data = Prosper_clean, x= 'BorrowerAPR', bins = bins);
print(x_y_t('Borrower APR', 'Count', 'Distribution of Borrower APR'))
```

None



In [39]:
```
Prosper_clean.BorrowerAPR.describe()
```

Out[39]:
```
count    107554.000000
mean          0.220352
std           0.080820
min           0.006530
25%           0.157130
50%           0.211220
75%           0.285950
max           0.423950
Name: BorrowerAPR, dtype: float64
```

APR is comprised of the interest rate stated on a loan plus fees, origination charges, discount points, and agency fees paid to the lender(Source :https://www.investopedia.com/ask/answers/100314/what-difference-between-interest-rate-and-annual-percentage-rate-apr.asp). Looking at this distribution we can have a look at each borrowers API with the highest been 0.42 .

LoanOriginalAmount

In [40]:
```
Prosper_clean.LoanOriginalAmount.value_counts()
```

Out[40]:
```
4000     14101
15000    12270
10000    10852
5000      6401
2000      5773
         ...
10593        1
11446        1
3136         1
4256         1
4292         1
Name: LoanOriginalAmount, Length: 2362, dtype: int64
```

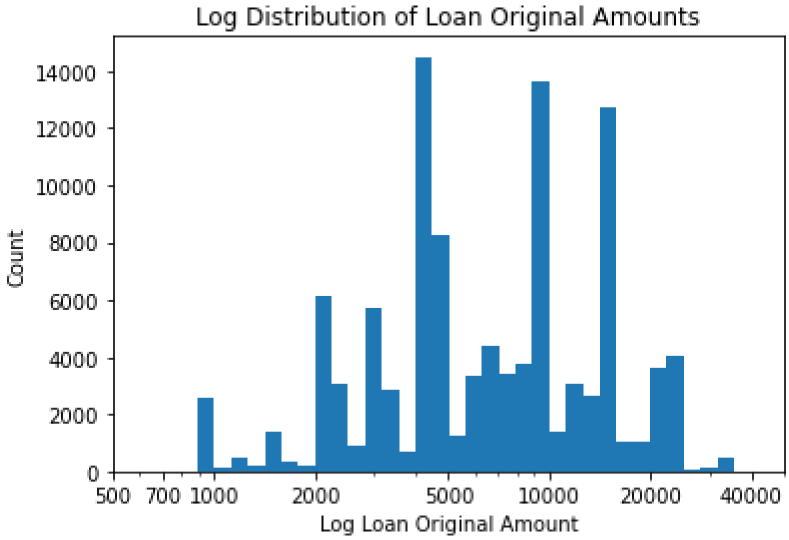
We will be using a log Distribution for thr Loan Original Amount due to the large number counts

In [41]:
```
log_binsize = 0.05
bins = 10 ** np.arange(1.5, np.log(Prosper_clean.LoanOriginalAmount.max())+log_bin
```

```
ticks = [500, 700, 1000, 2000, 5000, 10000, 20000, 40000]

plt.hist(data = Prosper_clean, x= 'LoanOriginalAmount', bins = bins);

plt.xscale('log')
plt.xticks(ticks, ticks)
plt.xlabel('Log Loan Original Amount')
plt.ylabel('Count')
plt.title('Log Distribution of Loan Original Amounts')
plt.xlim(500, 50000)
```

Out[41]:

```
(500.0, 50000.0)
```



The limits on the x axis was set to have a proper distribution of the curve.

In [42]:

```
Prosper_clean.LoanOriginalAmount.describe()
```

Out[42]:

```
count    107554.000000
mean       8518.340917
std        6272.171802
min        1000.000000
25%        4000.000000
50%        7000.000000
75%       12000.000000
max       35000.000000
Name: LoanOriginalAmount, dtype: float64
```

The amounts loaned out ranges from $1000 to 35000$, This also indicates that more than 75% had their loan amount above $4000

Recommendation

Under this visual. We will like to see how often are people recommended for a loan

In [43]:

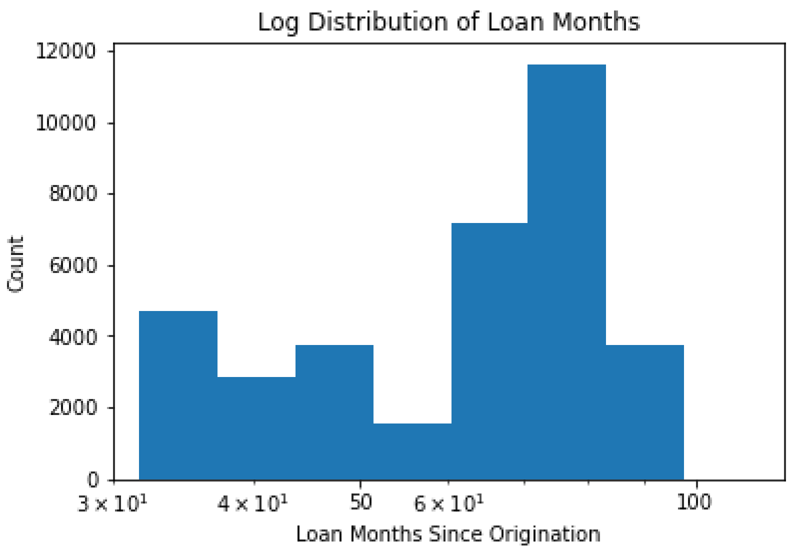
```
Prosper_clean.LoanMonthsSinceOrigination.value_counts()
```

```
Out[43]: 2      5865
3      5215
5      4899
1      4485
4      4336
       ...
90      304
55      248
92      163
56       20
58       13
Name: LoanMonthsSinceOrigination, Length: 86, dtype: int64
```

Log Distribution of the loan months since origination

```
In [44]: log_binsize = 0.07
bins = 10 ** np.arange(1.5, np.log(Prosper_clean.LoanMonthsSinceOrigination.max())
ticks = [50, 100, 1000, 2000, 5000, 6000]

plt.hist(data = Prosper_clean, x= 'LoanMonthsSinceOrigination', bins = bins);

plt.xscale('log')
plt.xticks(ticks, ticks)
plt.xlabel('Loan Months Since Origination')
plt.ylabel('Count')
plt.title('Log Distribution of Loan Months')
plt.xlim(30, 120)
```

```
Out[44]: (30.0, 120.0)
```




```
In [45]: Prosper_clean.LoanMonthsSinceOrigination.describe()
```

```
Out[45]: count    107554.000000
mean         28.558733
std          27.411125
min           0.000000
25%           6.000000
50%          19.000000
75%          45.000000
max          92.000000
Name: LoanMonthsSinceOrigination, dtype: float64
```

The number of month since the loan originated visual skewed toward the right meaning the data has recorded loans that have been disperesed beyond a month or two

Define

Condisdering the data set that have been set to categories under the datatype. We have the:

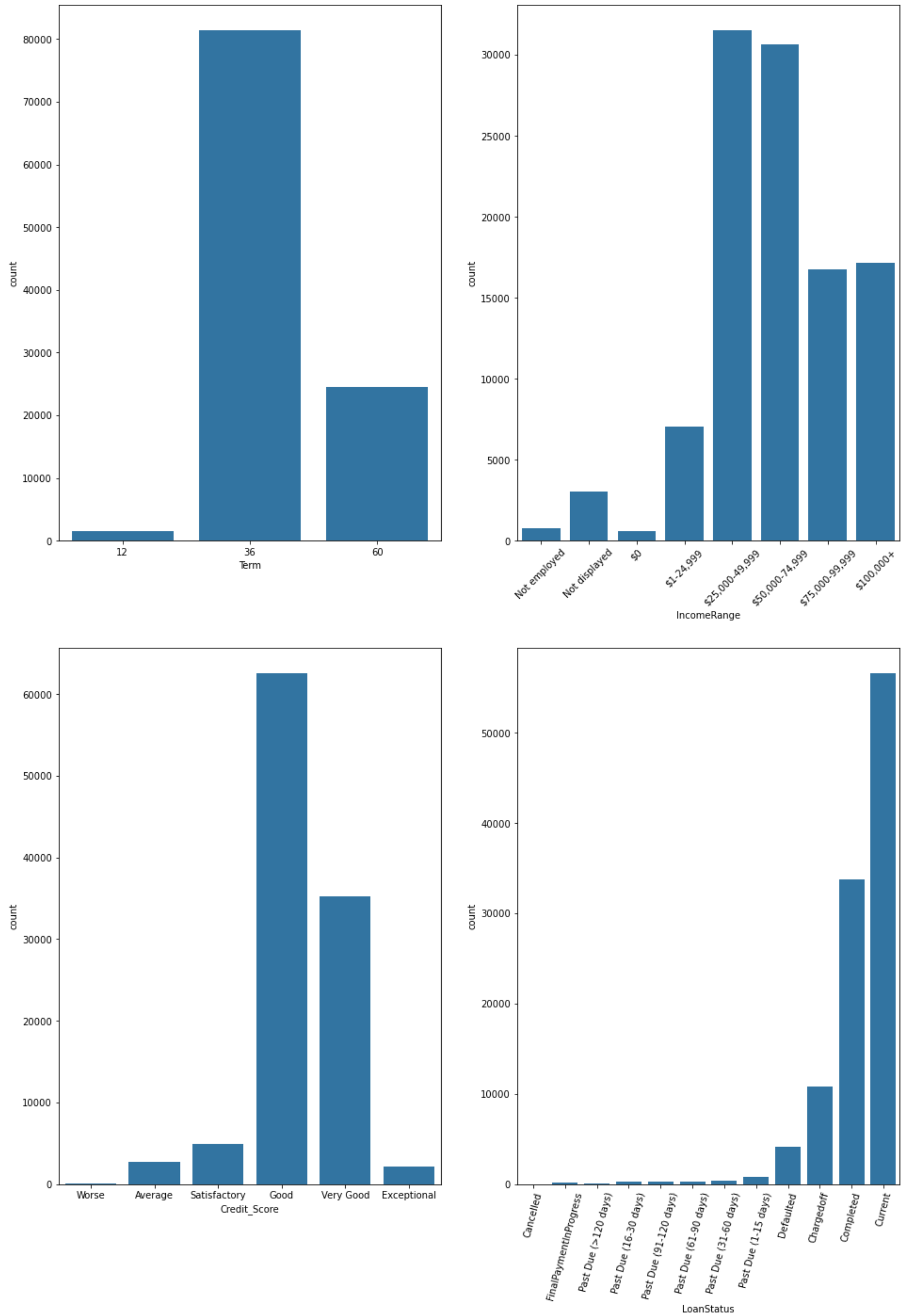
1. Term : The length of the loan expressed in months
2. LoanStatus: The current status of the loan: Cancelled, Chargedoff, Completed, Current, Defaulted, FinalPaymentInProgress, PastDue.
3. Credit_Score: Consumer credit score rating
4. IncomeRange: The income range of the borrower at the time the listing was created.

Code

Plotting the variables with datatype as category

In [46]:
```
#Setting the labels for the Loanstatus and Income range
LoanStatus_labels = ['Cancelled', 'FinalPaymentInProgress', 'Past Due (>120 days)',
IncomeRange_labels = ['Not employed', 'Not displayed', '$0', '$1-24,999', '$25,000-

fig, ax = plt.subplots(nrows=2 , ncols = 2, figsize = [16,20])

default_color = sb.color_palette()[0]
sb.countplot(data = Prosper_clean, x = 'Term', color = default_color, ax = ax[0, 0]
sb.countplot(data = Prosper_clean, x = 'IncomeRange', color = default_color, ax = a
sb.countplot(data = Prosper_clean, x = 'Credit_Score', color = default_color, ax =
sb.countplot(data = Prosper_clean, x = 'LoanStatus', color = default_color, ax = ax

ax[0, 1].set_xticklabels(labels = IncomeRange_labels, rotation = 45)
ax[1, 1].set_xticklabels(labels = LoanStatus_labels, rotation = 75)

plt.subplots_adjust(bottom = 0.01)
fig.suptitle("Data type Category Distribution", fontsize=15)

plt.show()
```

Data type Category Distribution



From the visualization we can realise that the loan status is skewed towards current. This shows the data had current loan acquisitions more than those that were cancelled or past due.

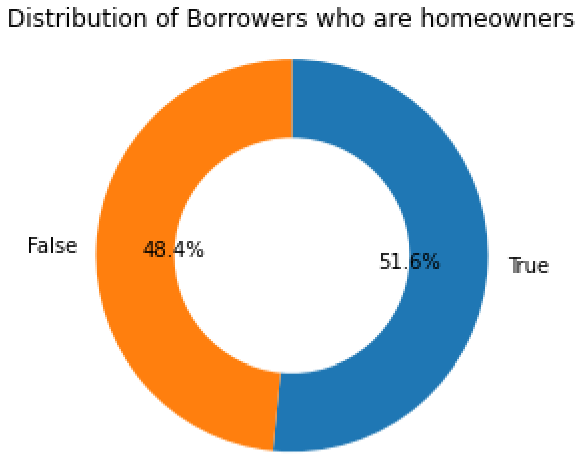
Define

The variable IsBorrowerHomeowner is a boolean value as such we will look at it with a pie chart and donut chart.

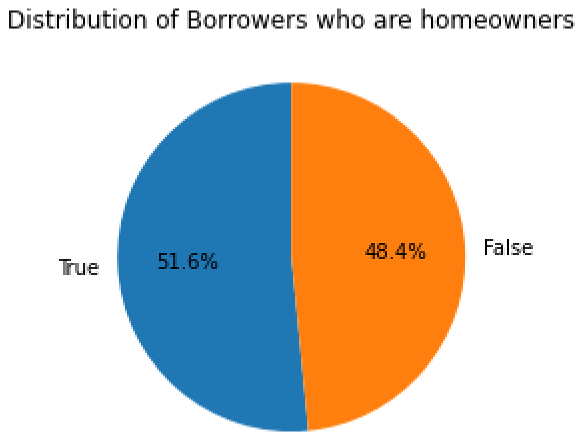
Code

In [47]:
```
sorted_counts = Prosper_clean['IsBorrowerHomeowner'].value_counts()

plt.pie(sorted_counts, labels = sorted_counts.index, startangle = 90,
        counterclock = False, wedgeprops = {'width' : 0.4}, autopct='%0.01f%%');
plt.axis('square')
plt.title('Distribution of Borrowers who are homeowners')
```

Out[47]:
```
Text(0.5, 1.0, 'Distribution of Borrowers who are homeowners')
```



In [48]:
```
plt.pie(Prosper_clean.IsBorrowerHomeowner.value_counts().values, labels = Prosper_
        startangle = 90, autopct='%0.01f%%');

plt.title('Distribution of Borrowers who are homeowners')

plt.show()
```



We can tell from the charts that approximately 51% borrowers are homeowners and 48% are not homeowners

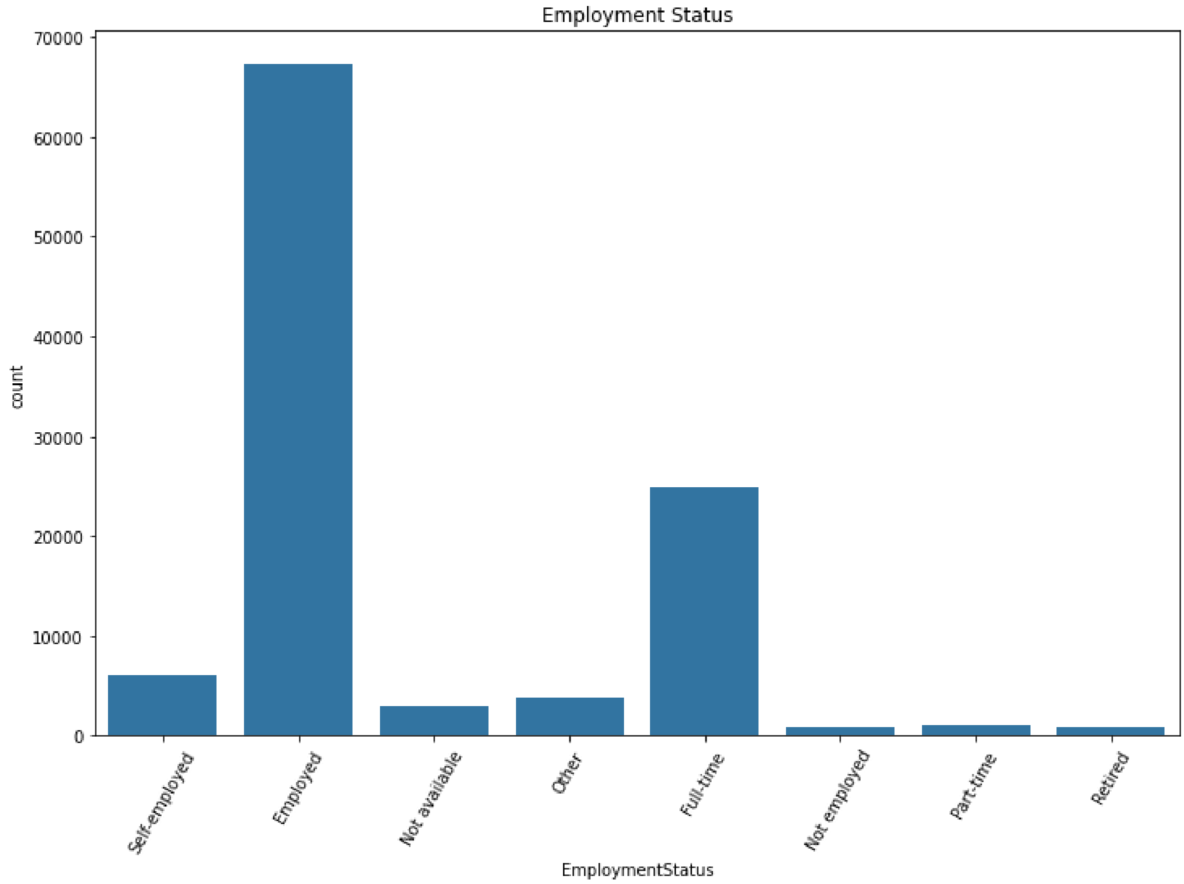
Define

Drawing the charts for employment status

Code

In [49]:
```
plt.figure(figsize = [12, 8])
default_color = sb.color_palette()[0]
sb.countplot(data = Prosper_clean, x = 'EmploymentStatus', color = default_color)
plt.title('Employment Status')
plt.xticks(rotation = 60)
```

Out[49]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7]),
 [Text(0, 0, 'Self-employed'),
  Text(1, 0, 'Employed'),
  Text(2, 0, 'Not available'),
  Text(3, 0, 'Other'),
  Text(4, 0, 'Full-time'),
  Text(5, 0, 'Not employed'),
  Text(6, 0, 'Part-time'),
  Text(7, 0, 'Retired')])
```



From the chart, more borrowers are employed and more are in full time

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

> The LoanMonthsSinceOrigination variable took on a large range of values, so I opted for a log transform. The number of month since the loan originated visual skewed toward the right meaning the data has recorded loans that have been disperesed beyond a month or two

> Most people who went in for a loan are employed. The retired, part-time and unemployed workers were the least people who went in for a loan

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?

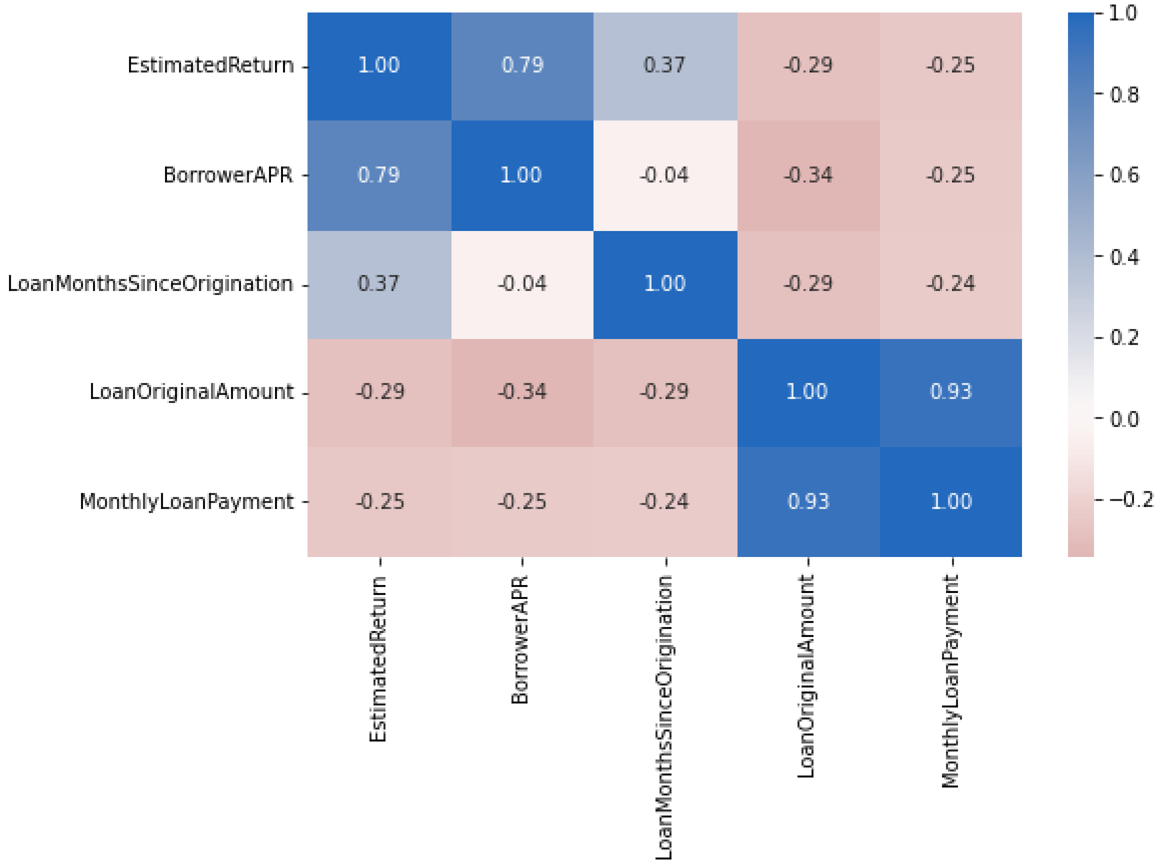
> I converted the scor names, score ranges into a credit_score column to correspond to the CreditScoreRangeLower variables. Also the ListingCategory (numeric) variables had to be changed into a categorical type for it to be useful in the data assessment.

Bivariate Exploration

To start off with, I will want to look at the pairwise correlations present between features in the data i.e (numeric and categories)

In [50]:
```
numeric_vars = ['EstimatedReturn', 'BorrowerAPR', 'LoanMonthsSinceOrigination', 'Lo
categoric_vars = ['IncomeRange', 'Credit_Score', 'EmploymentStatus']
```

In [51]:
```
# correlation plot
plt.figure(figsize = [8, 5])
sb.heatmap(Prosper_clean[numeric_vars].corr(), annot = True, fmt = '.2f',
           cmap = 'vlag_r', center = 0)
plt.show()
```



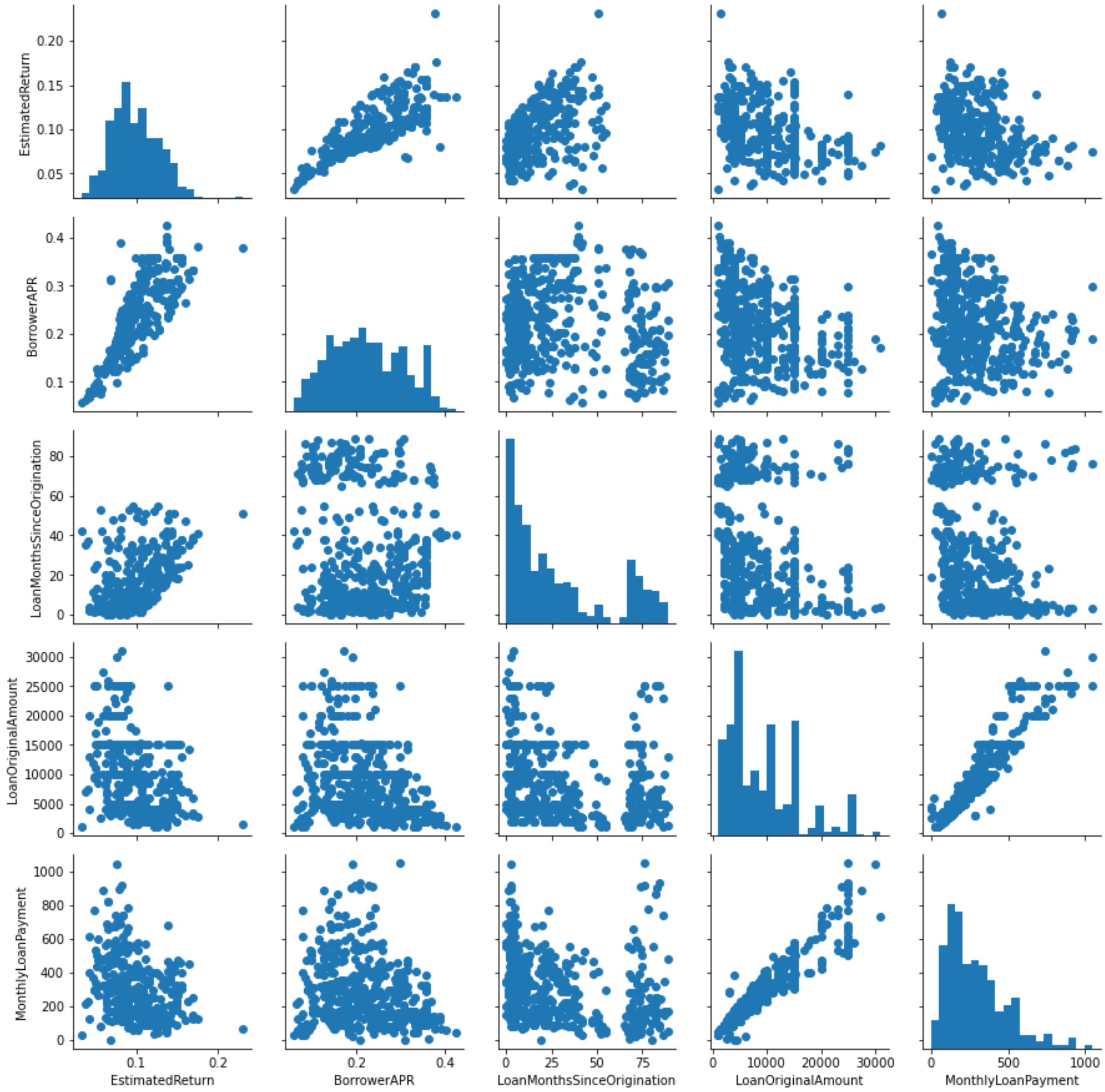
In [52]:
```
# plot matrix: sample 500 Prosper_clean so that plots are clearer and they render j
print("Prosper_clean.shape=",Prosper_clean.shape)
```

```
Prosper_clean_samp = Prosper_clean.sample(n=500, replace = False)
print("Prosper_clean_samp.shape=",Prosper_clean_samp.shape)

g = sb.PairGrid(data = Prosper_clean_samp, vars = numeric_vars)
g = g.map_diag(plt.hist, bins = 20);
g.map_offdiag(plt.scatter)
```

```
Prosper_clean.shape= (107554, 18)
Prosper_clean_samp.shape= (500, 18)
```

Out[52]:

```
<seaborn.axisgrid.PairGrid at 0x293300a0af0>
```



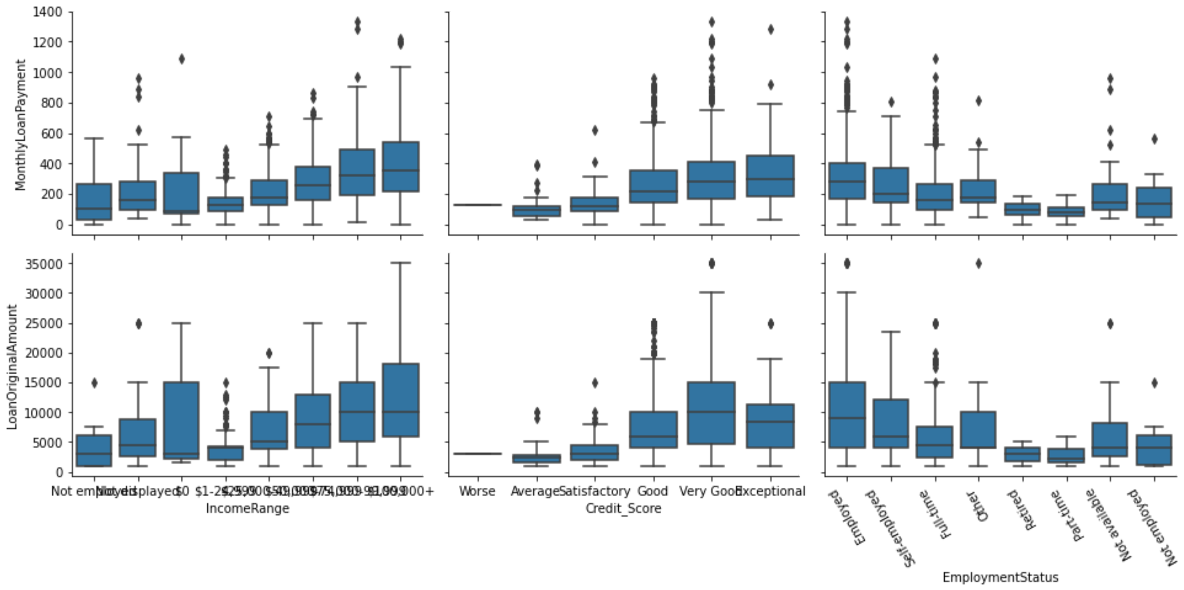
Monthly Loan Payment and LoanOriginalAmount against the Credit_Score, Income Range and Employment Status

In [53]:

```
# Box plot matrix of numeric features against categorical features.

Prosper_clean_samp = Prosper_clean.sample(n=2000, replace = False)


def boxgrid(x, y, **kwargs):
    """ Quick hack for creating box plots with seaborn's PairGrid. """
    default_color = sb.color_palette()[0]
    sb.boxplot(x=x, y=y, color=default_color)

plt.figure(figsize = [15, 15])
g = sb.PairGrid(data = Prosper_clean_samp, y_vars = ['MonthlyLoanPayment', 'LoanOr:
```

```
                    height = 3, aspect = 1.5)
g.map(boxgrid)
plt.xticks(rotation = 120)
plt.show();
```

```
<Figure size 1080x1080 with 0 Axes>
```



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?

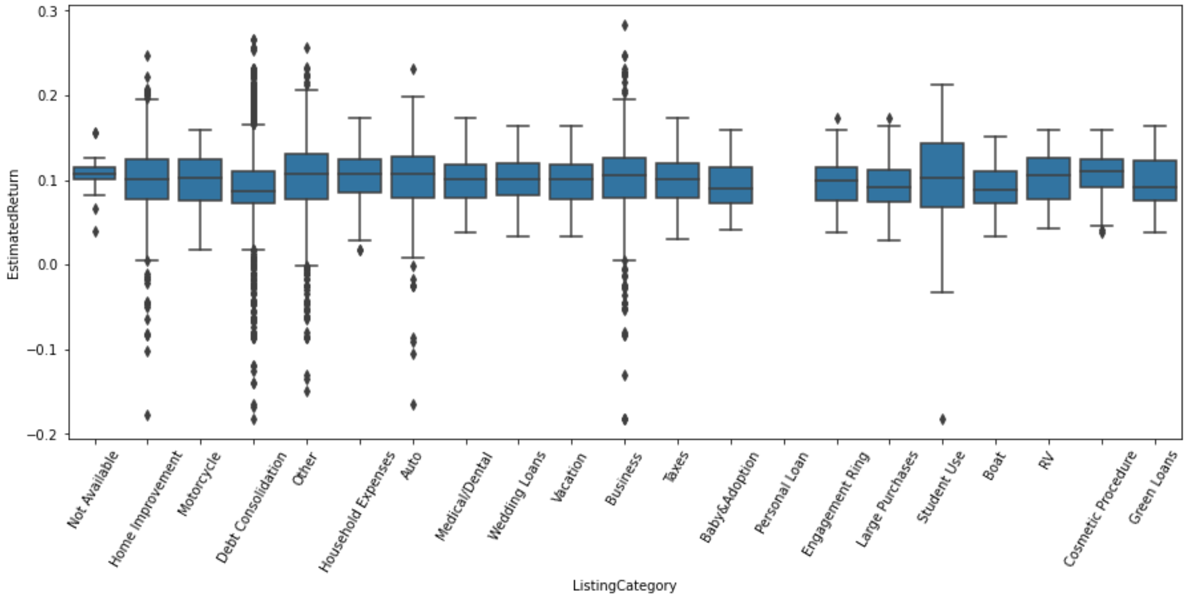
> The Loan original amount are highly correlated with the monthly loan payment. Also the correlation between the Estimated return and Borrower APR is very high. This shows the estimated return is affected by the Borrowers APR.
>
> The loan original amount increases per your employment status. Borrowers who are employed are able to obtain high loan amounts, than the unemployed and part time workers. Also, those with excellent and good credit scoreshad great loan amounts. Theose with high income ranges also obtain good loan amounts.

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

> Monthly loan payments across the categories also show how much each category is contributing. Under employment status we can tell that those employed are able to pay their monthly loan payments than those who aren't employed or part time. Also those with good credit scores are able to make thier monthly payments compared to those with poor ones

In [54]:

```
plt.figure(figsize = [15, 6])

sb.boxplot(data = Prosper_clean, x = 'ListingCategory', y = 'EstimatedReturn', col
plt.xticks(rotation = 60);
```



The Estimated returns of investors on the various loans listed categories. Every investor will consider their returns on the usage of the loan by loan borrowers.

In [55]:
```
Prosper_clean.info()
```

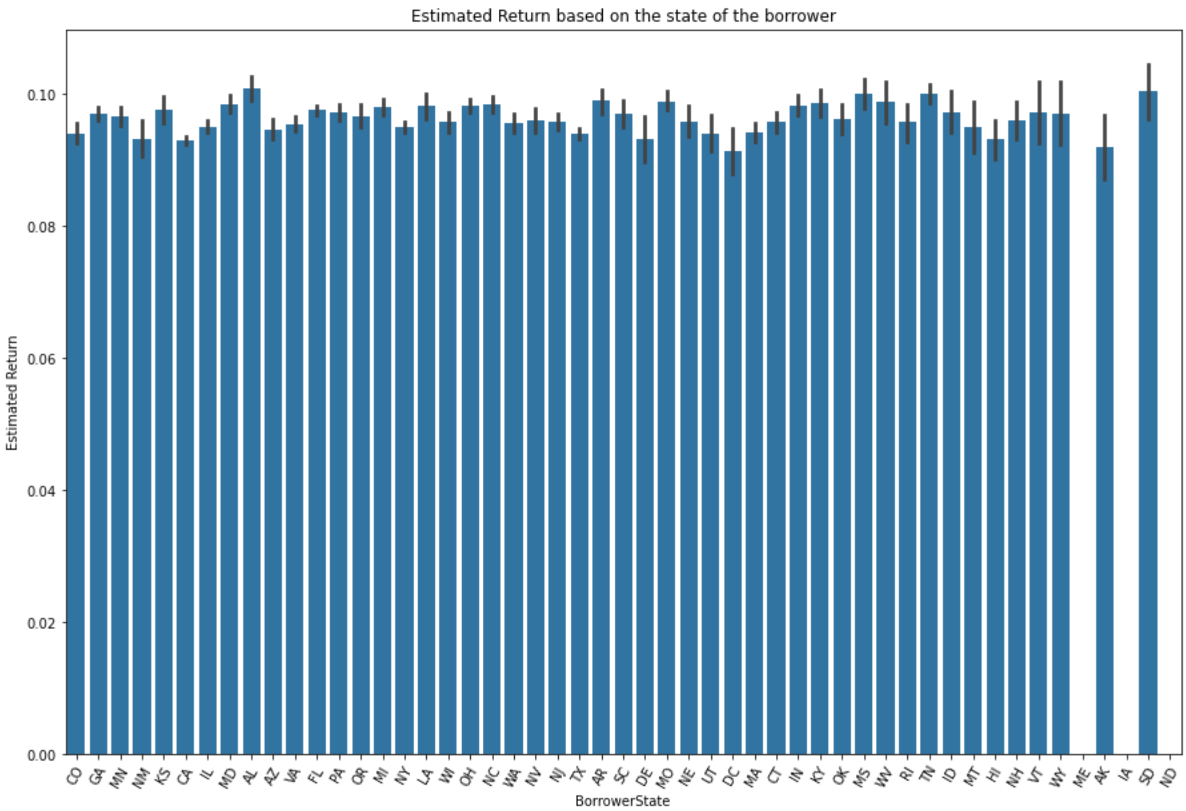
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 107554 entries, 0 to 113936
Data columns (total 18 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   ListingKey                    107554 non-null  object
 1   ListingCreationDate           107554 non-null  object
 2   Term                          107554 non-null  category
 3   LoanStatus                    107554 non-null  category
 4   BorrowerAPR                   107554 non-null  float64
 5   LenderYield                   107554 non-null  float64
 6   EstimatedReturn               84853 non-null   float64
 7   ListingCategory               107554 non-null  object
 8   BorrowerState                 107554 non-null  object
 9   EmploymentStatus              107554 non-null  object
 10  IsBorrowerHomeowner           107554 non-null  bool
 11  CreditScoreRangeLower         107554 non-null  float64
 12  Credit_Score                  107554 non-null  category
 13  IncomeRange                   107554 non-null  category
 14  MonthlyLoanPayment            107554 non-null  float64
 15  LoanMonthsSinceOrigination    107554 non-null  int64
 16  LoanOriginalAmount            107554 non-null  int64
 17  Recommendations               107554 non-null  int64
dtypes: bool(1), category(4), float64(5), int64(3), object(5)
memory usage: 12.0+ MB
```

In [56]:
```
Prosper_clean.BorrowerState.value_counts()
```

Out[56]:
```
CA    14577
TX     6699
NY     6692
FL     6674
IL     5898
GA     4904
OH     4196
MI     3543
VA     3272
NJ     3093
NC     3055
WA     3008
PA     2968
MD     2812
MO     2583
MN     2316
MA     2224
CO     2188
IN     2067
AZ     1878
WI     1837
OR     1787
TN     1734
AL     1667
CT     1625
SC     1118
NV     1090
KS     1050
KY      983
OK      966
LA      950
UT      870
AR      853
MS      785
NE      671
ID      595
NH      547
NM      466
RI      435
HI      408
WV      385
DC      382
MT      325
DE      300
VT      207
AK      200
SD      189
IA      186
WY      150
ME       97
ND       49
Name: BorrowerState, dtype: int64
```

In [57]:
```
base_color = sb.color_palette()[0]
plt.figure(figsize = [15, 10])
sb.barplot(data = Prosper_clean, x = 'BorrowerState', y = 'EstimatedReturn', color
plt.title('Estimated Return based on the state of the borrower')
plt.xlabel('BorrowerState')
plt.ylabel('Estimated Return')
plt.xticks(rotation = 60)
```

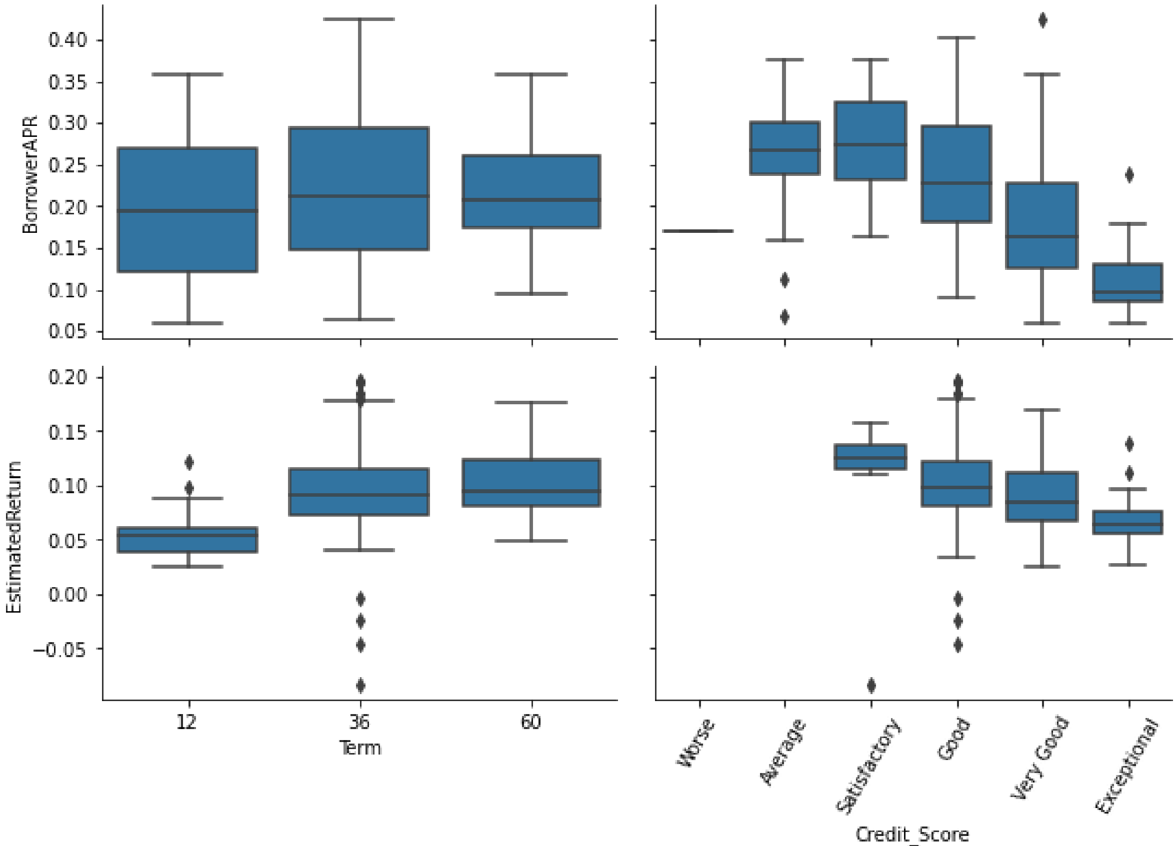
```
Out[57]: (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50]),
 [Text(0, 0, 'CO'),
  Text(1, 0, 'GA'),
  Text(2, 0, 'MN'),
  Text(3, 0, 'NM'),
  Text(4, 0, 'KS'),
  Text(5, 0, 'CA'),
  Text(6, 0, 'IL'),
  Text(7, 0, 'MD'),
  Text(8, 0, 'AL'),
  Text(9, 0, 'AZ'),
  Text(10, 0, 'VA'),
  Text(11, 0, 'FL'),
  Text(12, 0, 'PA'),
  Text(13, 0, 'OR'),
  Text(14, 0, 'MI'),
  Text(15, 0, 'NY'),
  Text(16, 0, 'LA'),
  Text(17, 0, 'WI'),
  Text(18, 0, 'OH'),
  Text(19, 0, 'NC'),
  Text(20, 0, 'WA'),
  Text(21, 0, 'NV'),
  Text(22, 0, 'NJ'),
  Text(23, 0, 'TX'),
  Text(24, 0, 'AR'),
  Text(25, 0, 'SC'),
  Text(26, 0, 'DE'),
  Text(27, 0, 'MO'),
  Text(28, 0, 'NE'),
  Text(29, 0, 'UT'),
  Text(30, 0, 'DC'),
  Text(31, 0, 'MA'),
  Text(32, 0, 'CT'),
  Text(33, 0, 'IN'),
  Text(34, 0, 'KY'),
  Text(35, 0, 'OK'),
  Text(36, 0, 'MS'),
  Text(37, 0, 'WV'),
  Text(38, 0, 'RI'),
  Text(39, 0, 'TN'),
  Text(40, 0, 'ID'),
  Text(41, 0, 'MT'),
  Text(42, 0, 'HI'),
  Text(43, 0, 'NH'),
  Text(44, 0, 'VT'),
  Text(45, 0, 'WY'),
  Text(46, 0, 'ME'),
  Text(47, 0, 'AK'),
  Text(48, 0, 'IA'),
  Text(49, 0, 'SD'),
  Text(50, 0, 'ND')])
```



Drawing a chart of Borrower APR and Estimated return against the Term and also the Credit_Score

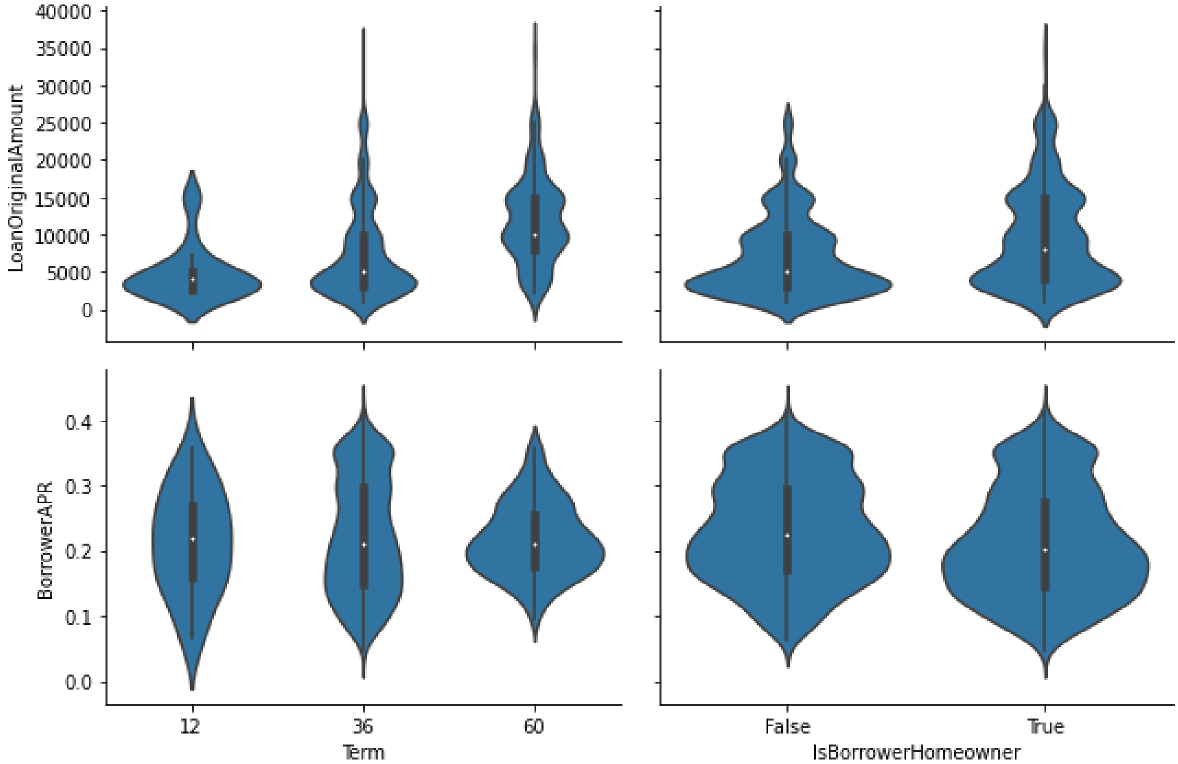
In [58]:
```
# Box plot matrix of numeric features against categorical features.

Prosper_clean_samp = Prosper_clean.sample(n=2000, replace = False)


def boxgrid(x, y, **kwargs):
    """ Quick hack for creating box plots with seaborn's PairGrid. """
    default_color = sb.color_palette()[0]
    sb.boxplot(x=x, y=y, color=default_color)

plt.figure(figsize = [15, 15])
g = sb.PairGrid(data = Prosper_clean_samp, y_vars = ['BorrowerAPR', 'EstimatedRetur
                height = 3, aspect = 1.5)
g.map(boxgrid)
plt.xticks(rotation = 60)
plt.show();
```

```
<Figure size 1080x1080 with 0 Axes>
```



In [59]:
```
Prosper_clean_samp = Prosper_clean.sample(n=2000, replace = False)


def boxgrid(x, y, **kwargs):
    """ Quick hack for creating box plots with seaborn's PairGrid. """
    default_color = sb.color_palette()[0]
    sb.violinplot(x=x, y=y, color=default_color)

plt.figure(figsize = [10, 10])
g = sb.PairGrid(data = Prosper_clean_samp, y_vars = ['LoanOriginalAmount', 'Borrowe
                height = 3, aspect = 1.5)
g.map(boxgrid)
plt.show();
```

```
<Figure size 720x720 with 0 Axes>
```



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?

> The higher the term the higher the estimated return. But with the APR, with a term of 12 has higher APR than an APR with a term of 60.
>
> The estimated return increases acroos the credit_score and the median for the APR across the credit_score decreases

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

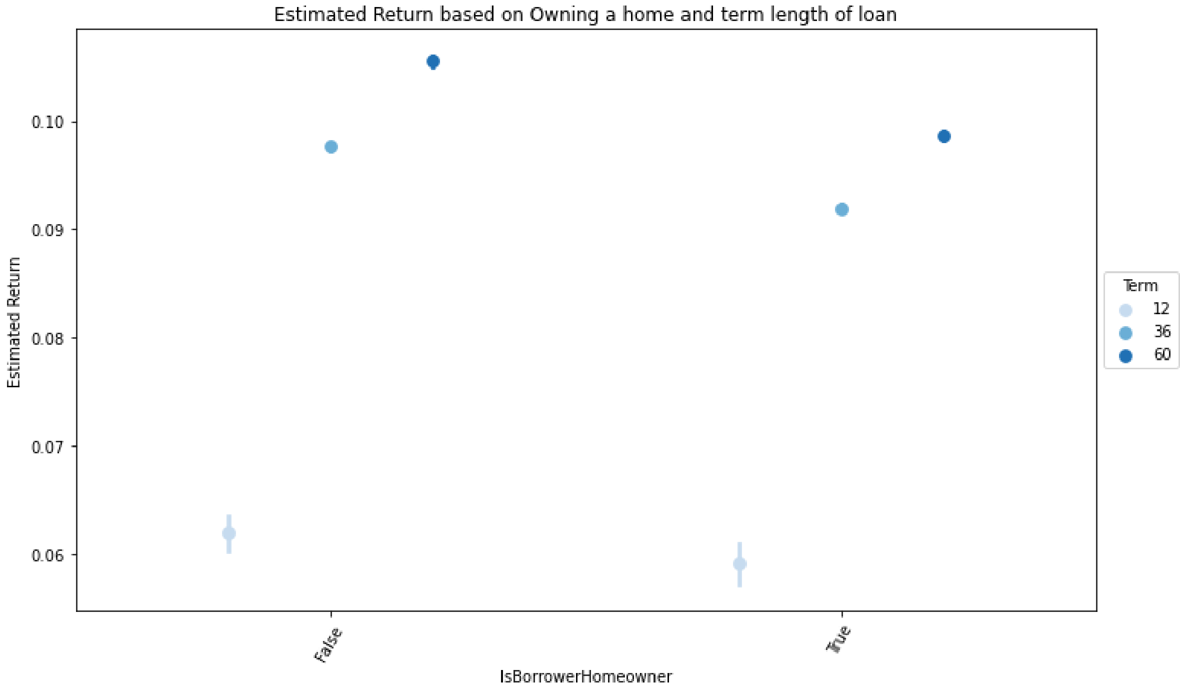
> The borrower state didn't have any major influence on the estimated return by investors. Meaning estimated return is not influenced by the state in which one lives

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.

In [60]:
```
fig = plt.figure(figsize = [12,7])
ax = sb.pointplot(data = Prosper_clean, x = 'IsBorrowerHomeowner', y = 'EstimatedR
           palette = 'Blues', linestyles = '', dodge = 0.4)

plt.title('Estimated Return based on Owning a home and term length of loan')
plt.xlabel('IsBorrowerHomeowner')
plt.ylabel('Estimated Return')
```

```
plt.xticks(rotation = 60)
# plot legend outside of figure
ax.legend(loc='center left', title='Term', bbox_to_anchor=(1, 0.5))
plt.show();
```



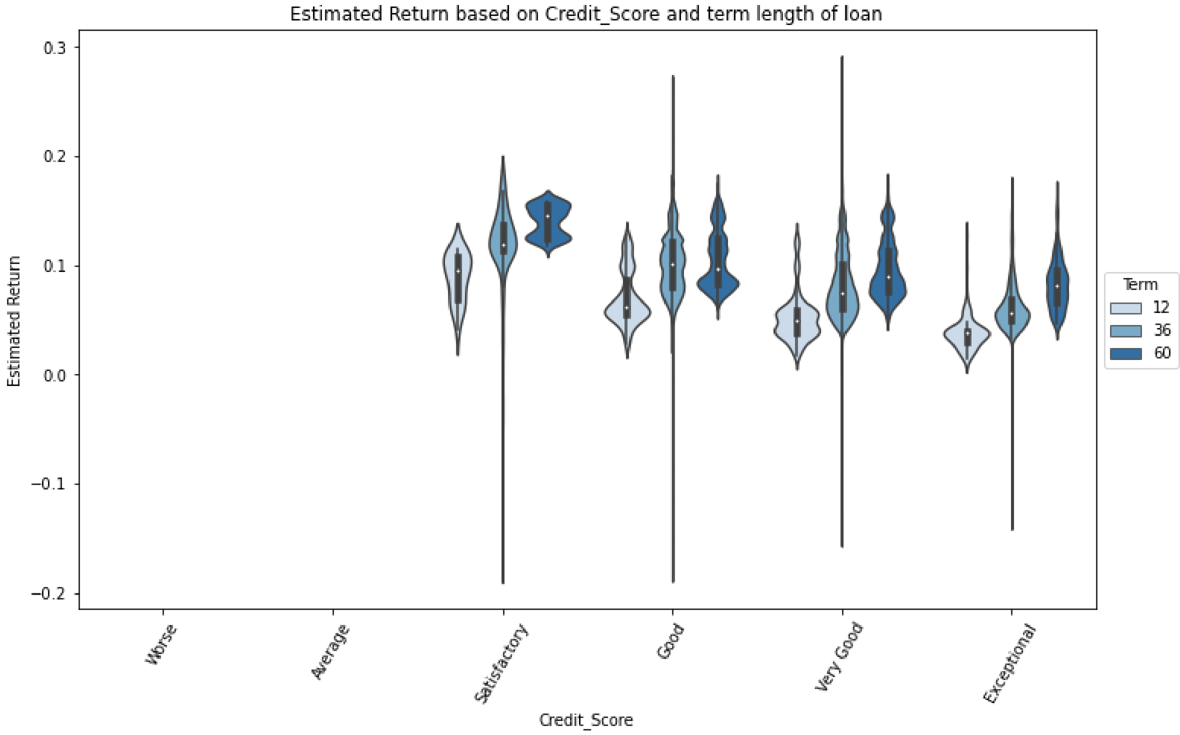
Under the Estimated return based on whether a borrower is a home owner and considering the length of the loan.

I observed that whether one is a homeowner or not doesn't really have a major impact to the Estimated return. Nevertheless the term length did but just a marginal difference.

The Estimated return seems to for home owners under the various term length seems less than those without homes. But as said the difference is minimal

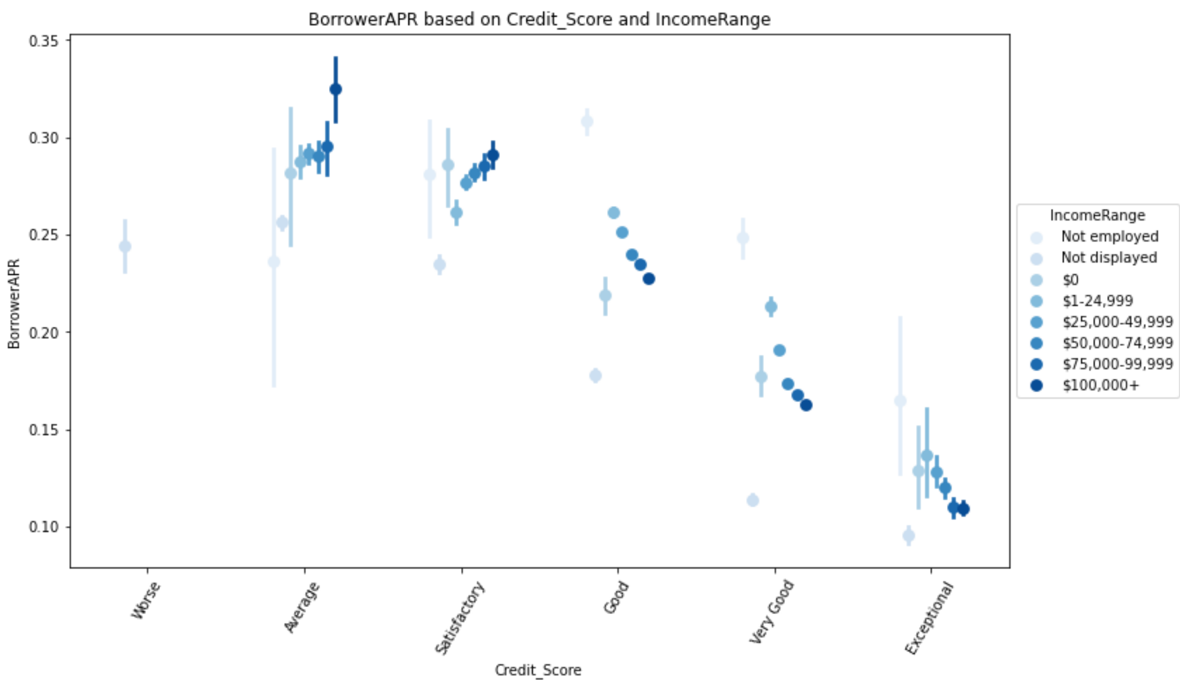
In [61]:

```
fig = plt.figure(figsize = [12,7])
ax = sb.violinplot(data = Prosper_clean, x = 'Credit_Score', y = 'EstimatedReturn',
           palette = 'Blues', linestyles = '', dodge = 0.4)

plt.title('Estimated Return based on Credit_Score and term length of loan')
plt.xlabel('Credit_Score')
plt.ylabel('Estimated Return')
plt.xticks(rotation = 60)
# plot legend outside of figure
ax.legend(loc='center left', title='Term', bbox_to_anchor=(1, 0.5))
plt.show();
```



Having a look at this visualisation, which considered whether the Estimated return is affected by credit_score and term length. One can easily recognise that yes the credit score and term length affect the Estimated return in a directly proportional relationship.

In [66]:
```
fig = plt.figure(figsize = [12,7])
ax = sb.pointplot(data = Prosper_clean, x = 'Credit_Score', y = 'BorrowerAPR',  hue
           palette = 'Blues', linestyles = '', dodge = 0.4)

plt.title('BorrowerAPR based on Credit_Score and IncomeRange')
plt.xlabel('Credit_Score')
plt.ylabel('BorrowerAPR')
plt.xticks(rotation = 60)
# plot legend outside of figure
ax.legend(loc='center left', title='IncomeRange', bbox_to_anchor=(1, 0.5))
plt.show();
```



With an exceptional credit score, even if you are not employed your Borrower APR will be lower than someone with an average or satisfactory credit_score and earning $100,000+. This clearly shows one's credit score is important and directly affects the Borrowers APR

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?

> The estimated return wasn't much affected if one was a homeowner or not. It was marginally high for those who had no ownership of homes than those who had ownership of homes.
>
> The term length and credit score also influenced one's estimated return.
>
> Been unemployed affect one's APR no matter the credit score one had
>
> ### Were there any interesting or surprising interactions between features?
>
> The term length and the credit score had a linear relation. The borrower APR decreased with a good credit score.

Conclusions

It was insightful and educative as well. Some key points taking are:

1. Credit Score is a greatfactor with respect to the amount you can obtain for a loan amount.
2. Employment history also affects the loan amount as well as your ability to pay monthly.
3. The country or place of stay doesn't influence the loan and APR that much.
4. Estimated returns on loans are also influenced by the use of the loan in activities with respect to the listing category in the project.
5. The term of your loan also influences your APR.

In [67]:

```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Part_I_exploration_template.ipynb'])
```

Out[67]:

```
1
```

In [ ]: