

(Loan Data from Prosper) ¶

by (Ahmed Tarek)

Preliminary Wrangling

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

```
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]: df_loans = pd.read_csv('prosperLoanData.csv')
df_loans
```

Out[2]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	
0	1021339766868145413AB3B	193129	2007-08-26 19:09:29.263000000	C	36	
1	10273602499503308B223C1	1209647	2014-02-27 08:28:07.900000000	NaN	36	
2	0EE9337825851032864889A	81716	2007-01-05 15:00:47.090000000	HR	36	
3	0EF5356002482715299901A	658116	2012-10-22 11:02:35.010000000	NaN	36	
4	0F023589499656230C5E3E2	909464	2013-09-14 18:38:39.097000000	NaN	36	
...	
113932	E6D9357655724827169606C	753087	2013-04-14 05:55:02.663000000	NaN	36	
113933	E6DB353036033497292EE43	537216	2011-11-03 20:42:55.333000000	NaN	36	FinalPa
113934	E6E13596170052029692BB1	1069178	2013-12-13 05:49:12.703000000	NaN	60	
113935	E6EB3531504622671970D9E	539056	2011-11-14 13:18:26.597000000	NaN	60	
113936	E6ED3600409833199F711B7	1140093	2014-01-15 09:27:37.657000000	NaN	36	

113937 rows × 81 columns



```
In [3]: df_loans.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

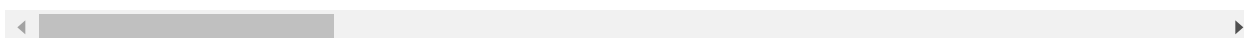
```

In [4]: `df_loans.describe()`

Out[4]:

	ListingNumber	Term	BorrowerAPR	BorrowerRate	LenderYield	EstimatedEffect
count	1.139370e+05	113937.000000	113912.000000	113937.000000	113937.000000	84853
mean	6.278857e+05	40.830248	0.218828	0.192764	0.182701	0
std	3.280762e+05	10.436212	0.080364	0.074818	0.074516	0
min	4.000000e+00	12.000000	0.006530	0.000000	-0.010000	-0
25%	4.009190e+05	36.000000	0.156290	0.134000	0.124200	0
50%	6.005540e+05	36.000000	0.209760	0.184000	0.173000	0
75%	8.926340e+05	36.000000	0.283810	0.250000	0.240000	0
max	1.255725e+06	60.000000	0.512290	0.497500	0.492500	0

8 rows × 61 columns



What is the structure of your dataset?

- 113937 loans (rows)
- 81 attributes (columns)

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

- The Borrower's Annual Percentage Rate (APR) for the loan
- The loan statuses and what attributes affect it

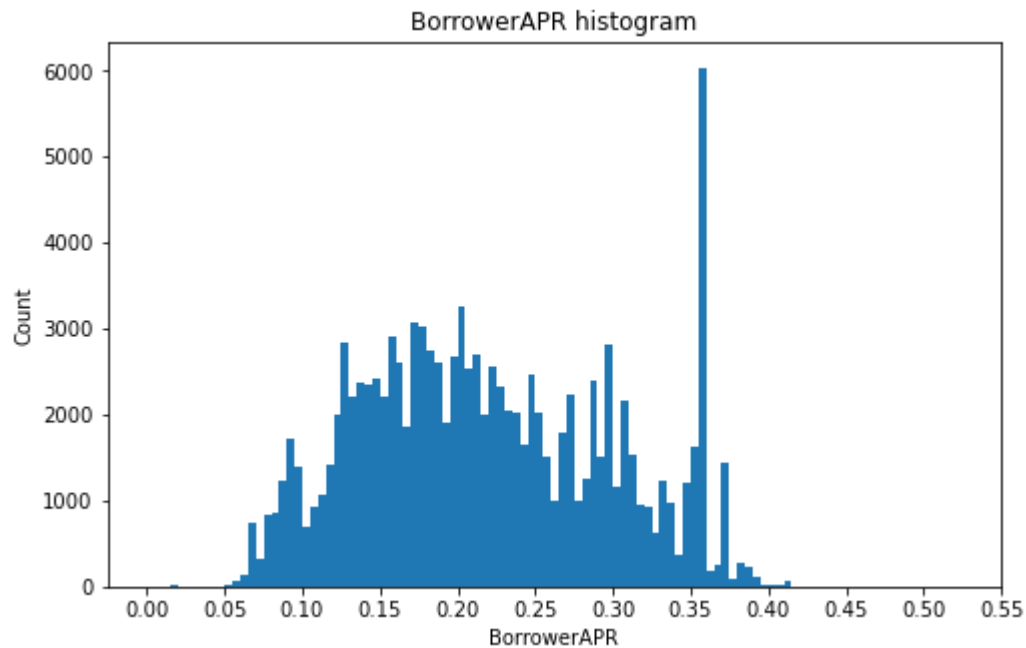
What features in the dataset do you think will help support your investigation into your feature(s) of interest?

- ProsperScore
- ProsperRating (Alpha)
- Occupation
- CreditScoreRangeLower and CreditScoreRangeUpper
- Term
- LoanOriginalAmount
- TotalProsperPaymentsBilled
- IncomeRange
- EmploymentStatus

Univariate Exploration

Lets take a look at the BorrowerAPR

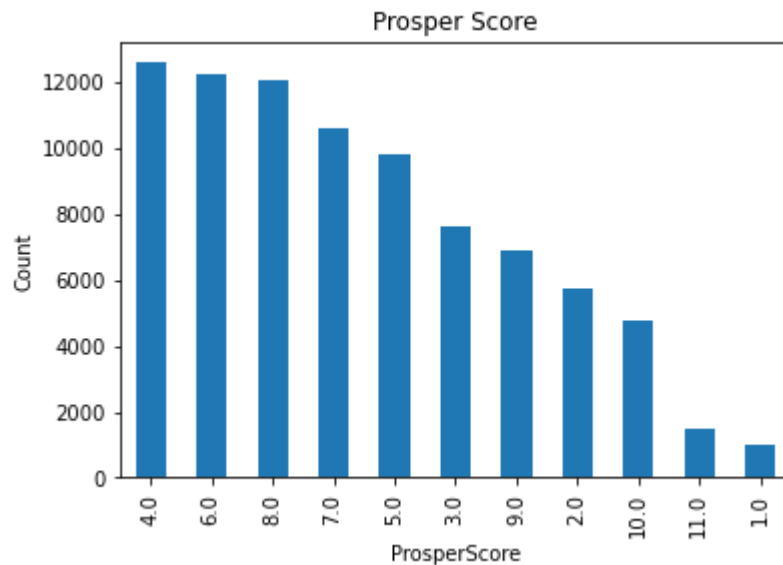
```
In [5]: # counts for BorrowerAPR values
bins = np.arange(0, df_loans.BorrowerAPR.max(), 0.005)
plt.figure(figsize=[8, 5])
plt.hist(data = df_loans, x = 'BorrowerAPR', bins = bins)
plt.xticks(np.arange(0, df_loans.BorrowerAPR.max()+0.05, 0.05))
plt.xlabel('BorrowerAPR')
plt.ylabel('Count')
plt.title('BorrowerAPR histogram');
```



It appears that this distribution is multimodal with several peaks. A peak at 0.08, 0.2, 0.3, and an exceptionally high peak at 0.36.

Let's look at Prosper Score which is a custom risk score built using historical Prosper data.

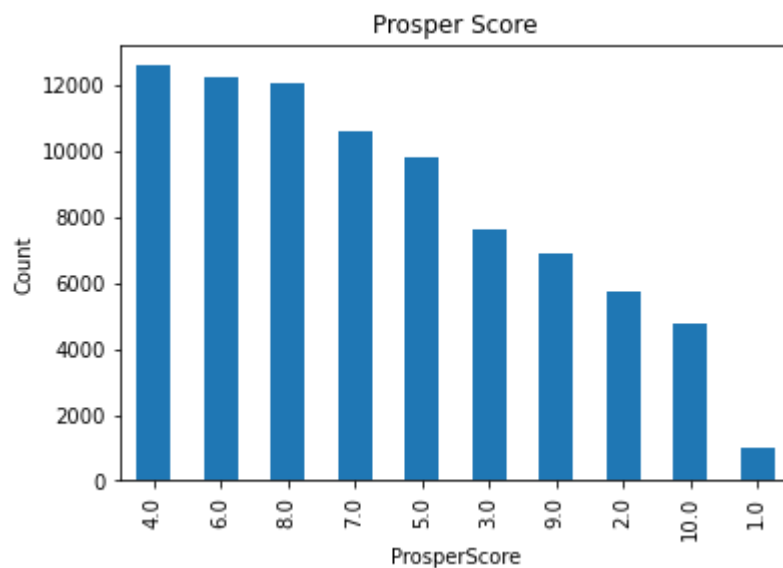
```
In [6]: # Bar Chart ProsperScore
df_loans.ProspersScore.value_counts().plot(kind='bar')
plt.xlabel('ProsperScore')
plt.ylabel('Count')
plt.title('Prosper Score');
```



This shows that there are scores above 10 which is not possible since the score is from 1-10. This data must be removed.

```
In [7]: # excluding all borrowers with a Prosper Score of 11
df_loans = df_loans[df_loans.ProspersScore != 11]
```

```
In [8]: # plotting the bar chart again
df_loans.ProspersScore.value_counts().plot(kind='bar')
plt.xlabel('ProsperScore')
plt.ylabel('Count')
plt.title('Prosper Score');
```



Most of the borrower got low Prosper Score of 4 that means they are risky to loan. Notice that even customers with a low risks score of 1 or 2 did get a loan. Not many borrowers received the highest score of 10.

Now lets look at ProsperRating (Alpha) and Occupation

Before plotting, the ProsperRating (Alpha) should be ordered from low to high so there won't be any misleading visualization about the rating order

```
In [9]: # Convert ProsperRating into ordered categorical types
rate_order = ['HR', 'E', 'D', 'C', 'B', 'A', 'AA']
ordered_var = pd.api.types.CategoricalDtype(ordered = True, categories = rate_order)
df_loans['ProsperRating (Alpha)'] = df_loans['ProsperRating (Alpha)'].astype(ordered_var)
```

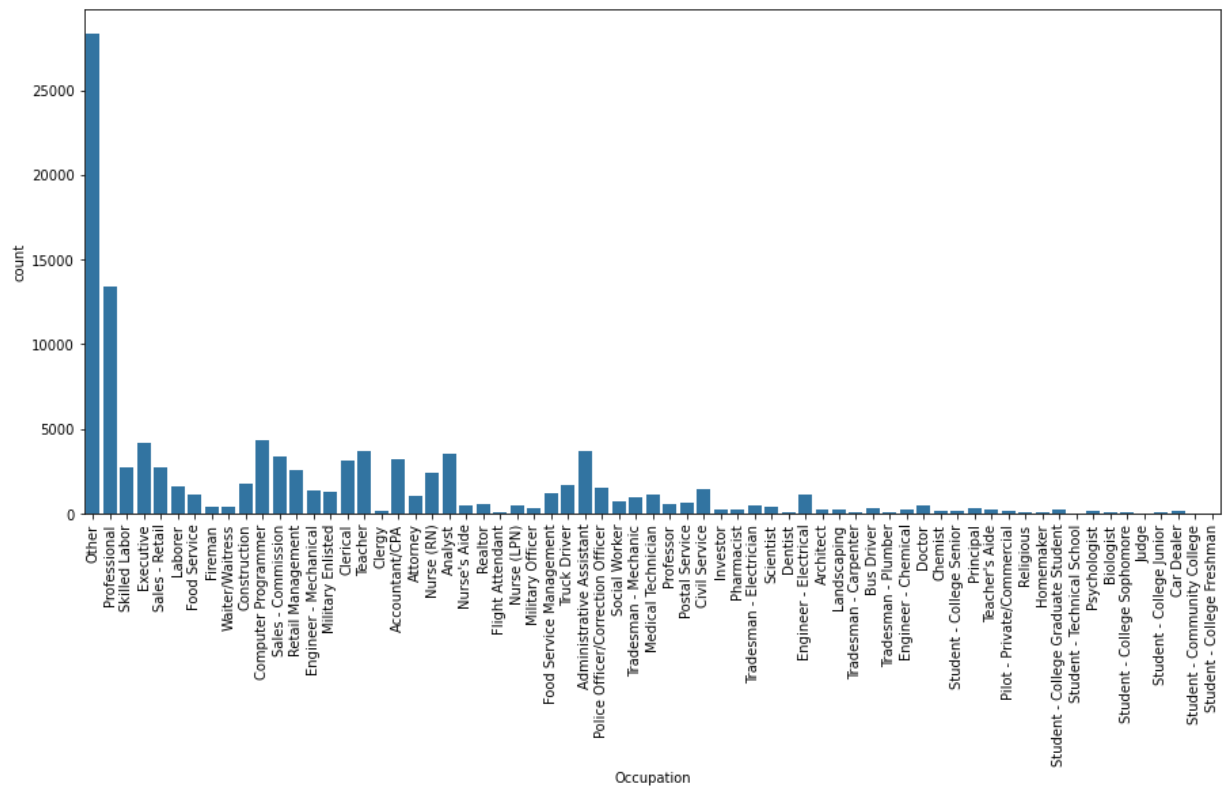
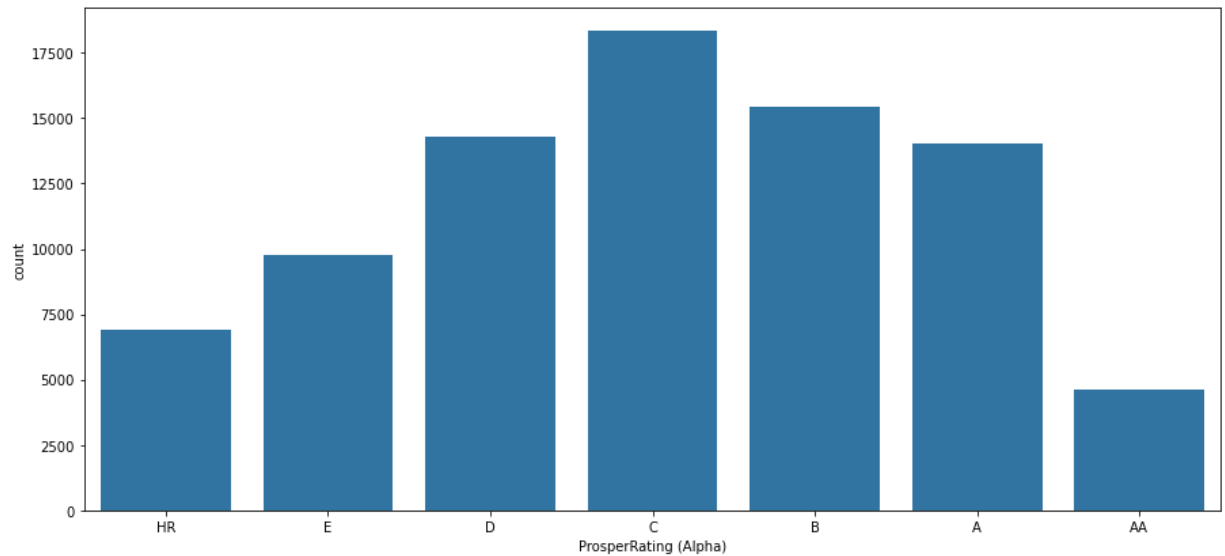
<ipython-input-9-9876739eac83>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df_loans['ProsperRating (Alpha)'] = df_loans['ProsperRating (Alpha)'].astype(ordered_var)
```



```
In [10]: fig, ax = plt.subplots(nrows=2, figsize = [15,15])
default_color = sb.color_palette()[0]
sb.countplot(data = df_loans, x = 'ProsperRating (Alpha)', color = default_color,
sb.countplot(data = df_loans, x = 'Occupation', color = default_color, ax = ax[1]
plt.xticks(rotation=90);
```



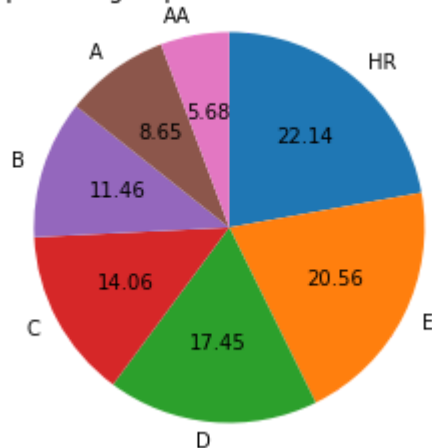
It appears that most borrowers were rated from C to A, and students are the least to take loans

Now, let's compare the Prosper rating (Alpha) mean with the Borrower APR mean

```
In [11]: # plotting bar chart for ProsperRating vs BorrowerAPR mean
ProsperRatingAlpha_mean = df_loans.groupby('ProsperRating (Alpha)').BorrowerAPR.mean()

plt.pie(ProsperRatingAlpha_mean, labels = ProsperRatingAlpha_mean.index, startangle=90)
plt.axis('square')
plt.title('ProsperRating (Alpha) VS. BorrowerAPR mean');
```

ProsperRating (Alpha) VS. BorrowerAPR mean



Notice that the highest rating of AA received the lowest BorrowerAPR (5.61), while the lowest rating of HR received the highest BorrowerAPR (22.17). This shows that borrowers with higher ratings received lower BorrowerAPR.

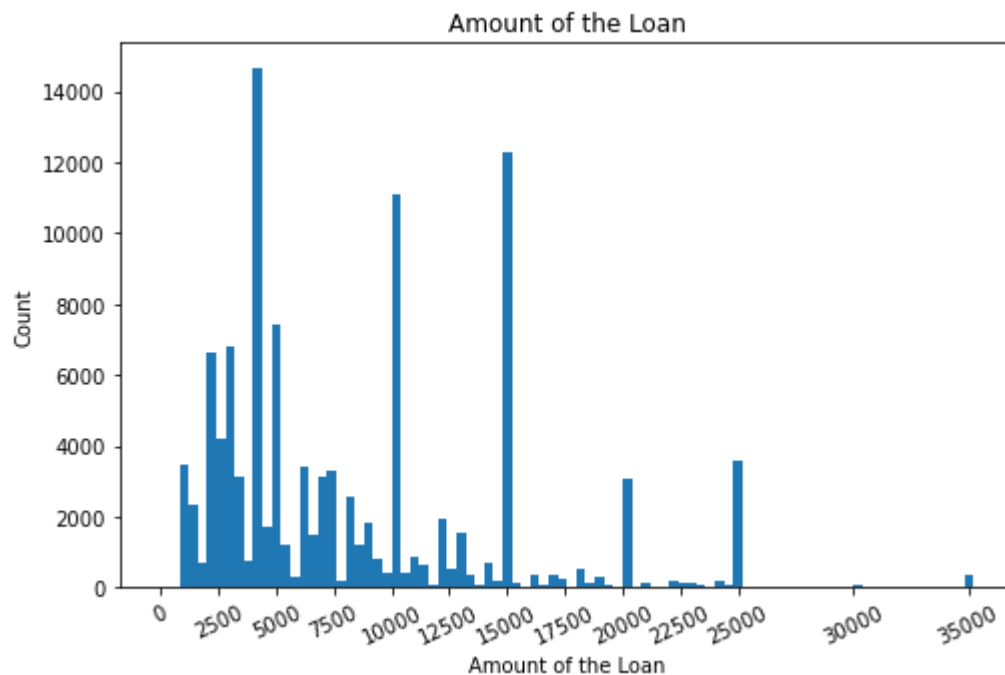
I am interested in knowing more about the LoanOriginalAmount

```
In [12]: # getting overview over the Loan amount
df_loans.LoanOriginalAmount.describe()
```

```
Out[12]: count    112481.000000
mean       8252.601133
std        6172.040285
min        1000.000000
25%        4000.000000
50%        6001.000000
75%       11500.000000
max       35000.000000
Name: LoanOriginalAmount, dtype: float64
```

```
In [13]: # Histogramm for The origination amount of the Loan
binsize = 400
bins = np.arange(0, df_loans.LoanOriginalAmount.max()+binsize, binsize)

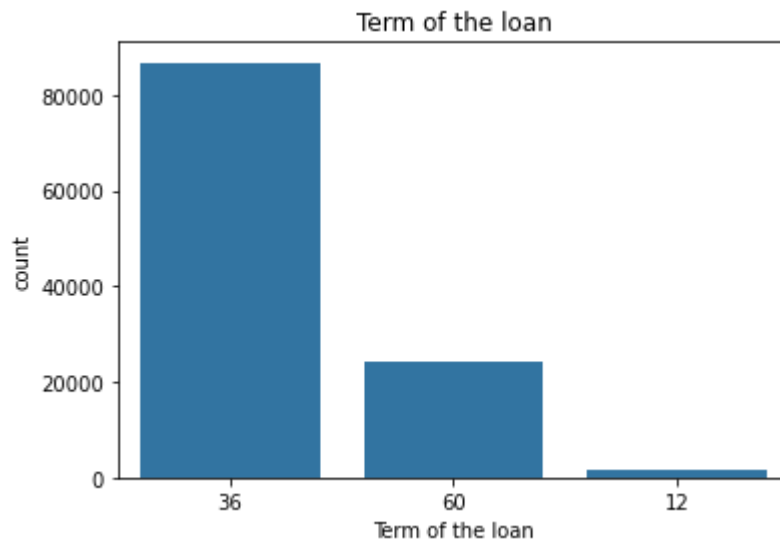
plt.figure(figsize=[8, 5])
plt.hist(data = df_loans, x = 'LoanOriginalAmount', bins = bins)
plt.xlabel('Amount of the Loan')
plt.ylabel('Count')
plt.title('Amount of the Loan')
plt.xticks([0,2500,5000,7500,10000,12500,15000,17500,20000,22500,25000,30000,35000])
```



The histogram has several peaks at around 4,000, 10,000, and 15,000. But most of the values are in the lower end between 2,500 and 10,000. The most loaned amounts are 4,000 and 15,000.

Now let's see if there is a relation between the LoanOriginalAmount and Term

```
In [14]: # Bar chart of the Length of the Loan in months
base_color = sb.color_palette()[0]
sb.countplot(data = df_loans, x='Term', color = base_color, order = df_loans.Term
plt.xlabel('Term of the loan')
plt.title('Term of the loan');
```



Most of the loans have a length of 36 months. The fact that most of the loans have a length of just 36 months might be correlated to the fact the average loan amount is around 8200 which can be paid back quickly.

Lets look at the CreditScoreRangeLower and CreditScoreRangeUpper

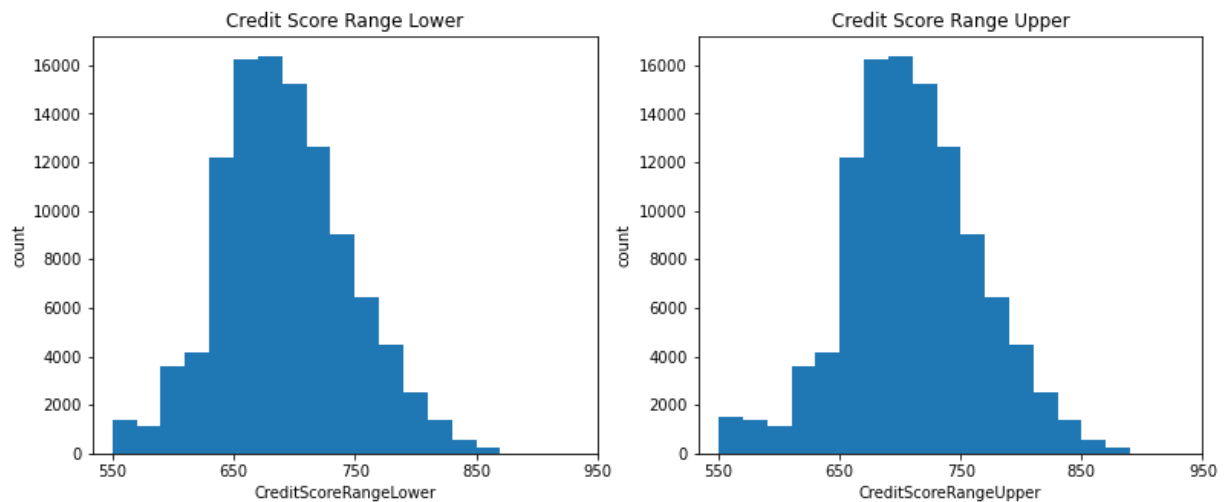
```

In [15]: # Histogram for CreditScoreRangeLower and CreditScoreRangeUpper
plt.figure(figsize = [13, 5])

plt.subplot(1, 2, 1)
bins = np.arange(550, df_loans.CreditScoreRangeLower.max(), 20)
plt.hist(data = df_loans, x = 'CreditScoreRangeLower', bins = bins)
plt.xticks(np.arange(550, 1000, 100))
plt.title('Credit Score Range Lower')
plt.xlabel('CreditScoreRangeLower')
plt.ylabel('count');

plt.subplot(1, 2, 2)
bins = np.arange(550, df_loans.CreditScoreRangeUpper.max(), 20)
plt.hist(data = df_loans, x = 'CreditScoreRangeUpper', bins = bins)
plt.xticks(np.arange(550, 1000, 100))
plt.title('Credit Score Range Upper')
plt.xlabel('CreditScoreRangeUpper')
plt.ylabel('count');

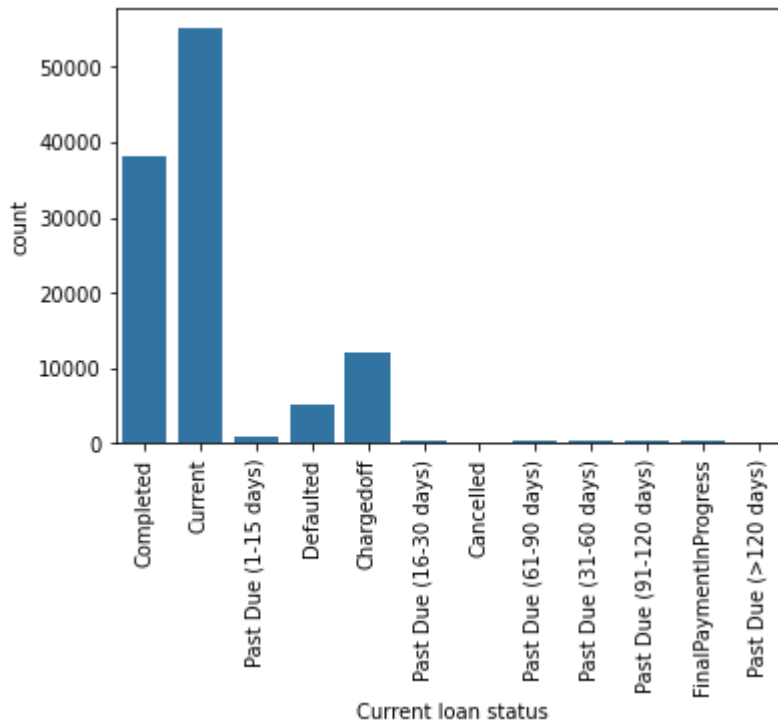
```



The two histograms show similar trends, and there are no outliers that fall out of the range.

Now lets take a look at the loan status

```
In [16]: # bar chart LoanStatus for the current status of the loan
base_color = sb.color_palette()[0]
sb.countplot(data = df_loans, x = 'LoanStatus', color = base_color)
plt.xlabel('Current loan status')
plt.xticks(rotation = 90);
```



Combining the past dues into one column since it is not important to show that much information

```
In [17]: # replacing Past Due (1-15 days)etc. with just Past Due
df_loans.LoanStatus = df_loans.LoanStatus.replace(['Past Due (1-15 days)', 'Past
                                                    'Past Due (91-120 days)', 'Past
                                                    'Past Due (>120 days)'], 'Past

df_loans.LoanStatus.value_counts()
```

C:\Users\Ahmed\anaconda3\lib\site-packages\pandas\core\generic.py:5168: Setting WithCopyWarning:

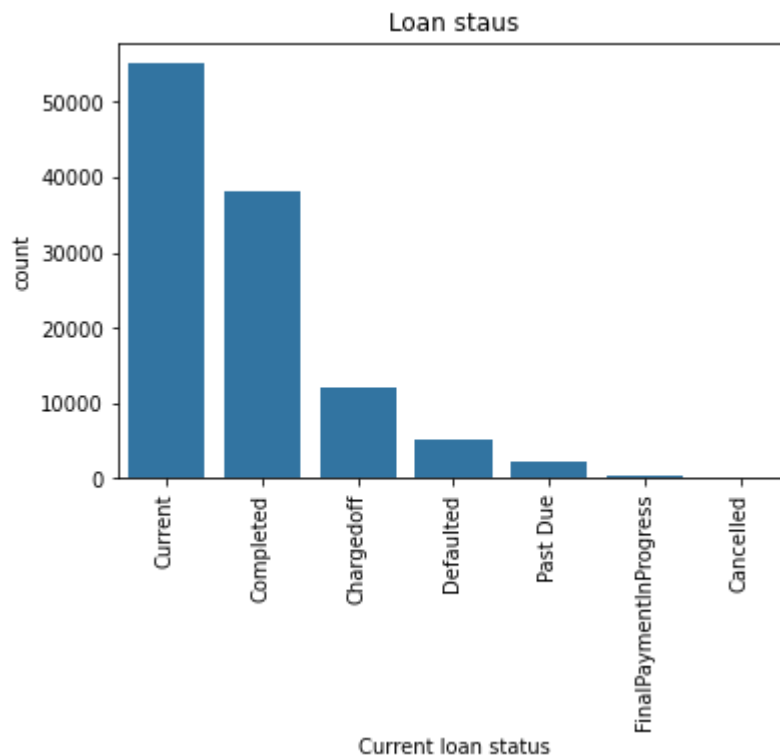
A value is trying to be set on a copy of a slice from a DataFrame.
Try using `.loc[row_indexer,col_indexer] = value` instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
self[name] = value
```

```
Out[17]: Current          55157
Completed        38043
Chargedoff       11992
Defaulted        5017
Past Due         2065
FinalPaymentInProgress    202
Cancelled         5
Name: LoanStatus, dtype: int64
```

```
In [18]: # plotting the bar chart of LoanStatus again
base_color = sb.color_palette()[0]
sb.countplot(data = df_loans, x = 'LoanStatus', color = base_color, order = df_lo
plt.xlabel('Current loan status')
plt.xticks(rotation = 90)
plt.title('Loan staus');
```

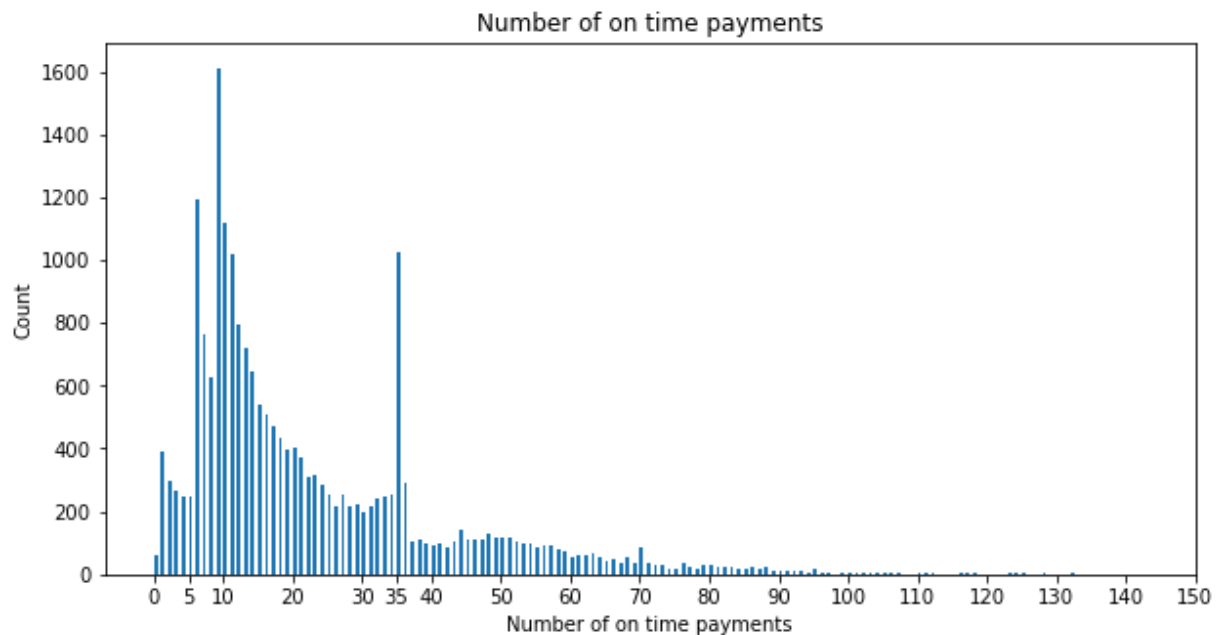


Most of the loans are current or completed.

Now lets look at the TotalProsperPaymentsBilled (Number of on time payments)


```
In [19]: # Histogramm for the number of on time payments
binsize = 0.5
bins = np.arange(df_loans.TotalProsperPaymentsBilled.min(), df_loans.TotalProsperPaymentsBilled.max(), binsize)

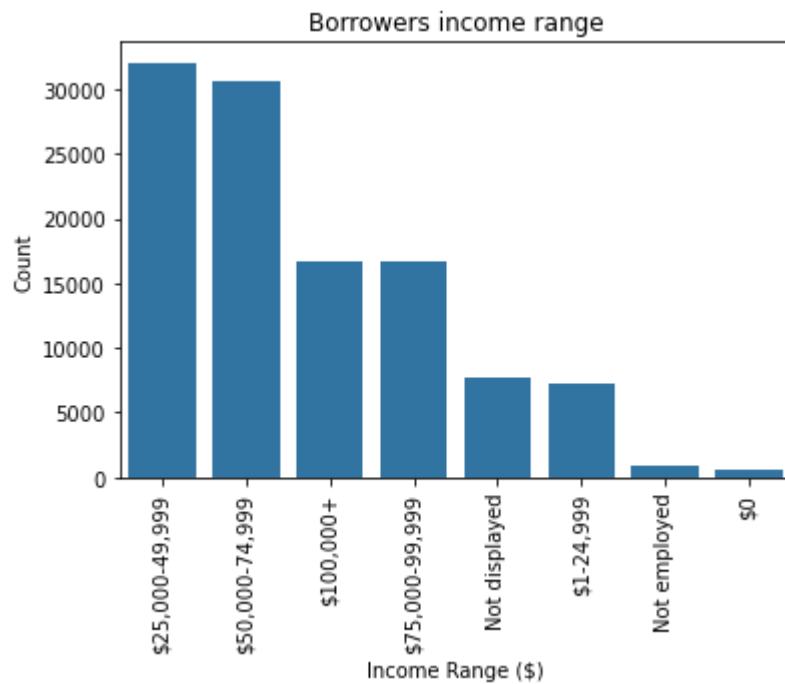
plt.figure(figsize=[10, 5])
plt.hist(data = df_loans, x = 'TotalProsperPaymentsBilled', bins = bins)
plt.xlabel('Number of on time payments')
plt.ylabel('Count')
plt.title('Number of on time payments')
plt.xticks([0,5,10,20,30,35,40,50,60,70,80,90,100,110,120,130,140,150])
plt.show()
```



The distribution of the number of on-time payments has two peaks 9 and 35. Notice that the distribution is right-skewed with most of the values on the lower end and fewer values on the higher end. This would make the distribution multimodal. It seems like that most of the borrowers had missed paying some of the monthly payments on time.

Now the IncomeRange

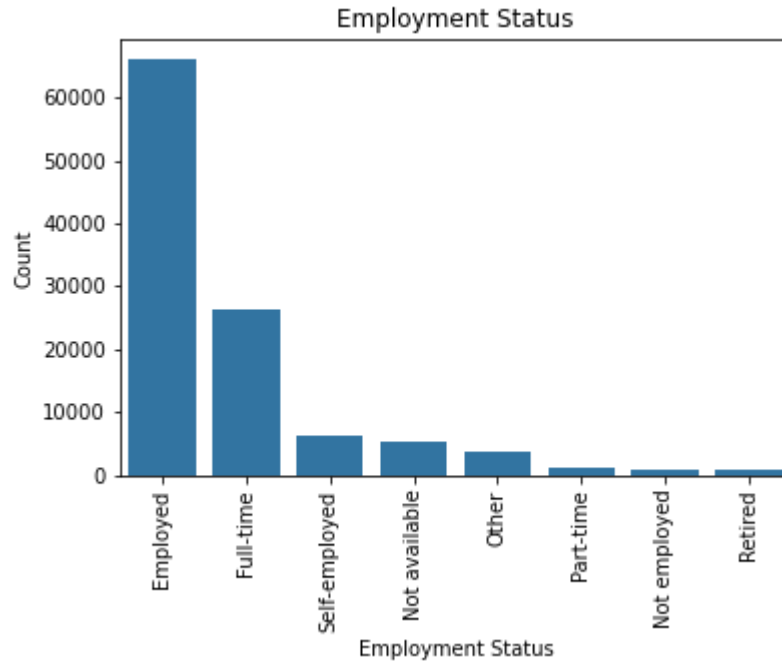
```
In [20]: # Bar chart of the borrowers income range
sb.countplot(data= df_loans, x= 'IncomeRange', color= base_color, order = df_loar
plt.xticks(rotation=90)
plt.xlabel('Income Range ($)')
plt.ylabel('Count')
plt.title('Borrowers income range');
```



The income range of the borrowers shows that most of the loans were given to customers with an income between 25,000 and 74,999. Notice that people that are not employed, or have an income of 0 received a loan as well. They might be other criteria than the income that qualifies one to get a loan like being a student.

Lets check the EmploymentStatus to find out

```
In [21]: # plotting the borrowers employment status
sb.countplot(data= df_loans, x= 'EmploymentStatus', color= base_color, order = df_loans['EmploymentStatus'].value_counts().index)
plt.xticks(rotation=90)
plt.xlabel('Employment Status')
plt.ylabel('Count')
plt.title('Employment Status');
```



Of course, most of the borrowers are employed, but the data shows that retired persons got a loan too.

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

- The distribution of borrowers' APR most of the values are between 0.05 and 0.4. but there is one value that was higher than the rest there might be a reason that this value is used that much. Therefore, there is no need to perform any transformations.
- Most of the loans are current or completed. However, there are a fairly significant number of loans that are charged off, defaulted, or past due.
- The income range of the borrowers shows that most of the loans were to give to customers with an income between 25,000 and 74,999.
- Most of the borrower got low Prosper Score of 4 that means they are risky to loan. You can also see that even customers with a low risks score of 1 or 2 did get a loan. Not many borrowers received the highest score of 10.
- The distribution of the number of on-time payments has two peaks 9 and 35. Notice that the distribution is right-skewed with most of the values on the lower end and fewer values on the higher end. This would make the distribution multimodal. It seems like that most of the borrowers had missed paying some of the monthly payments on time.
- Most of the loans have a length of 36 months. The fact that most of the loans have a length of just 36 months might be correlated to the fact the average loan amount is around 8200 which can be paid back quickly.

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?

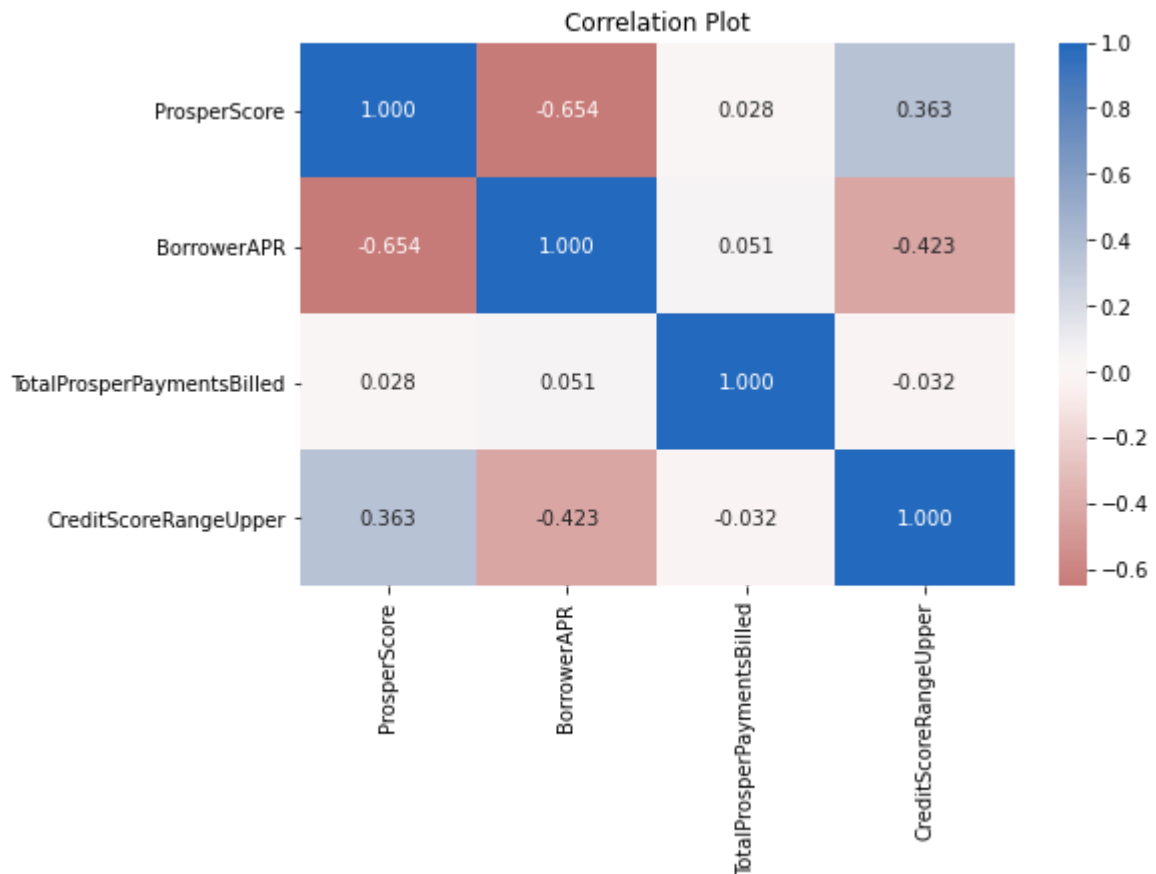
- Notice in the distribution of the Employment status that people that are not employed, or have an income of 0 received a loan. There might be other criteria than that qualifies one to get a loan like being a student or retired.
- The bar chart of the Prosper Score showed that there are customers in the data set that have a score of 11 which is not possible since the score is from 1-10, this data was removed.
- Combining the past dues into one column since it is not important to show that much information.

Bivariate Exploration

Lets look at the numeric variables

```
In [22]: # correlation plot for numeric variables
num_var = ['ProsperScore', 'BorrowerAPR', 'TotalProsperPaymentsBilled', 'CreditScoreRangeUpper']

plt.figure(figsize = [8, 5])
sb.heatmap(df_loans[num_var].corr(), annot = True, fmt = '.3f', cmap = 'vlag_r',
plt.title('Correlation Plot');
```

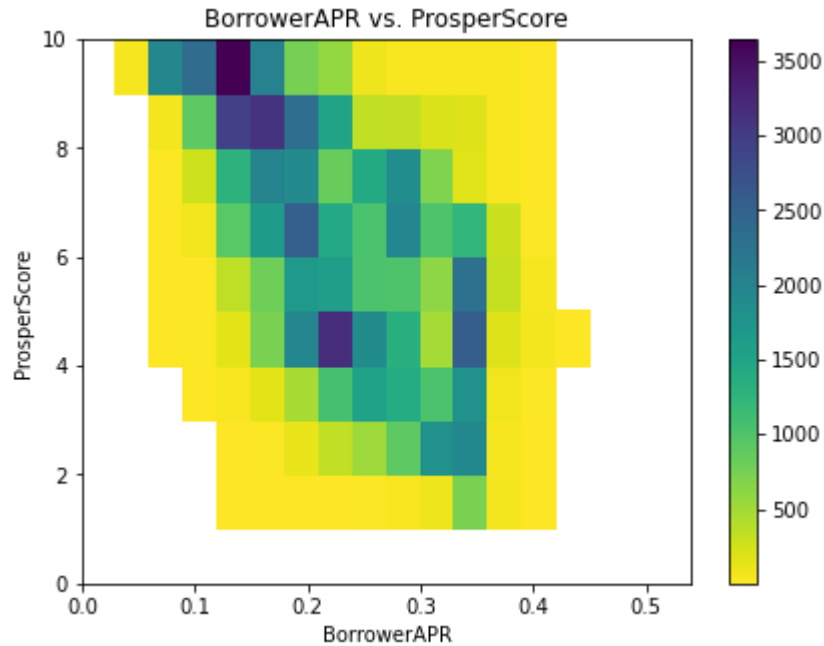


There are no strong positive relationships between any pairs. BorrowerAPR and ProsperScore are negative because borrowers with a lower score are more likely to pay higher APR. CreditScore and BorrowerAPR are also negative because the higher the borrowers CreditScore the more trustworthy they are, therefore they received lower APR.

Lets look more closely at ProsperScore vs BorrowerAPR

```
In [23]: # heat plot for comparing ProsperScore and BorrowerAPR
plt.figure(figsize = [15, 5])

plt.subplot(1, 2, 2)
bins_x = np.arange(0, df_loans.BorrowerAPR.max()+0.05, 0.03)
bins_y = np.arange(0, df_loans.ProspersScore.max()+1, 1)
plt.hist2d(data = df_loans, x = 'BorrowerAPR', y = 'ProspersScore', bins = [bins_x, bins_y])
plt.colorbar()
plt.title('BorrowerAPR vs. ProspersScore')
plt.xlabel('BorrowerAPR')
plt.ylabel('ProspersScore');
```

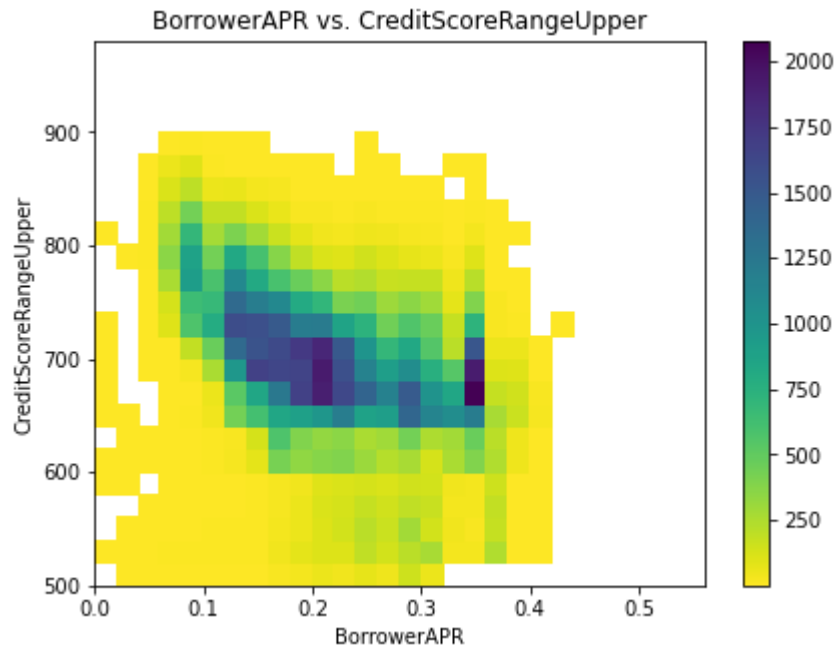


This also proves that people with higher ratings tend to be more trustworthy and therefore given lower BorrowerAPR.

Now BorrowerAPR vs. CreditScoreRangeUpper

```
In [24]: # heat plot for comparing ProsperScore and BorrowerAPR
plt.figure(figsize = [15, 5])

plt.subplot(1, 2, 2)
bins_x = np.arange(0, df_loans.BorrowerAPR.max()+0.05, 0.02)
bins_y = np.arange(500, df_loans.CreditScoreRangeUpper.max()+100, 20)
plt.hist2d(data = df_loans, x = 'BorrowerAPR', y = 'CreditScoreRangeUpper', bins
plt.colorbar()
plt.title('BorrowerAPR vs. CreditScoreRangeUpper')
plt.xlabel('BorrowerAPR')
plt.ylabel('CreditScoreRangeUpper');
```



We can see the trend that the higher the CreditScore the lower the APR.

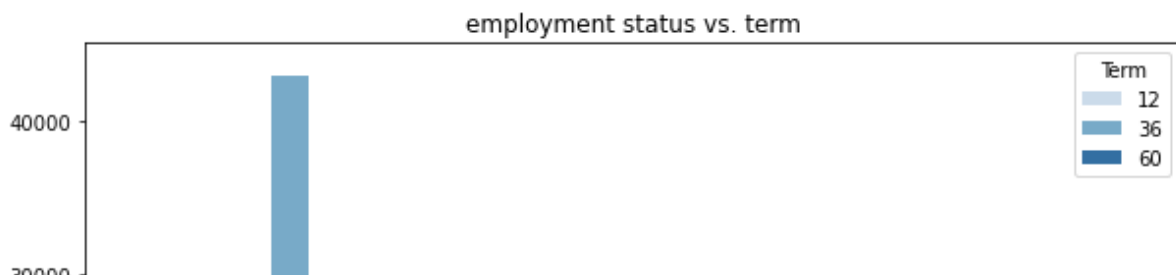
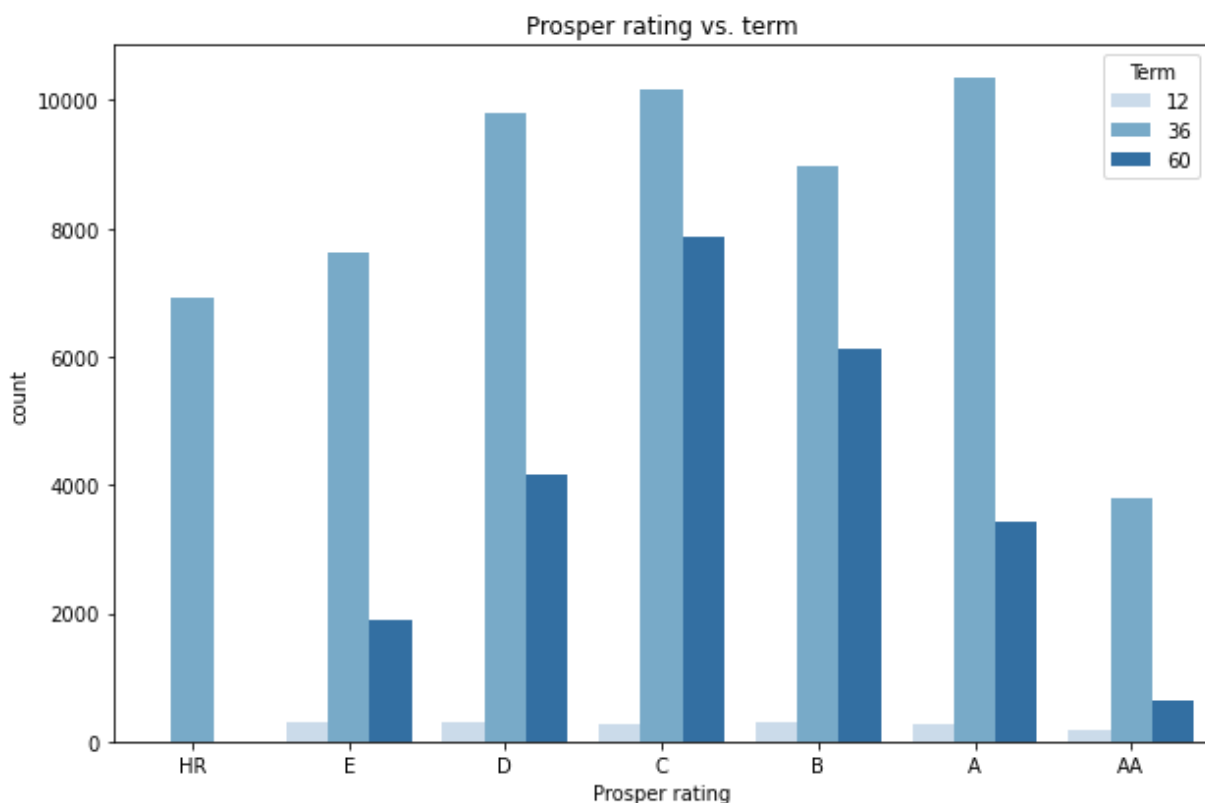
let's look at relationships between the categorical features.

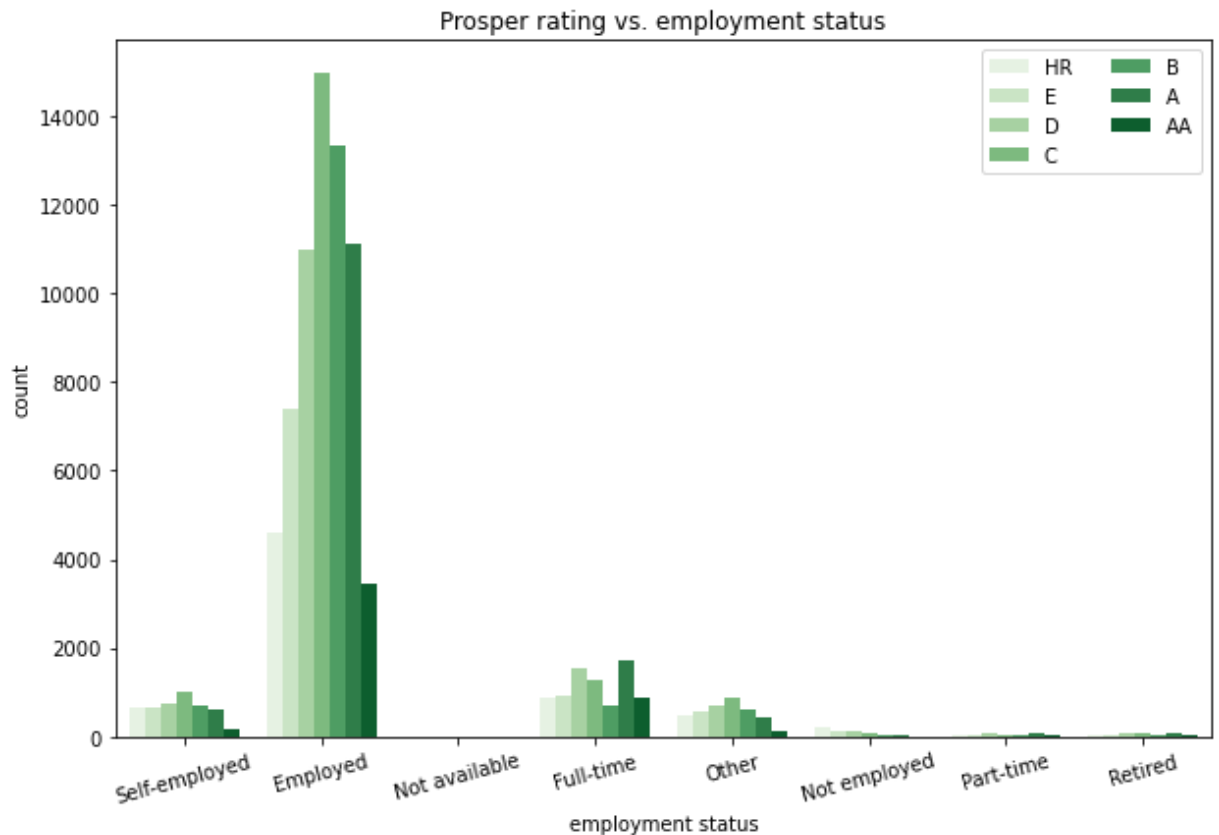
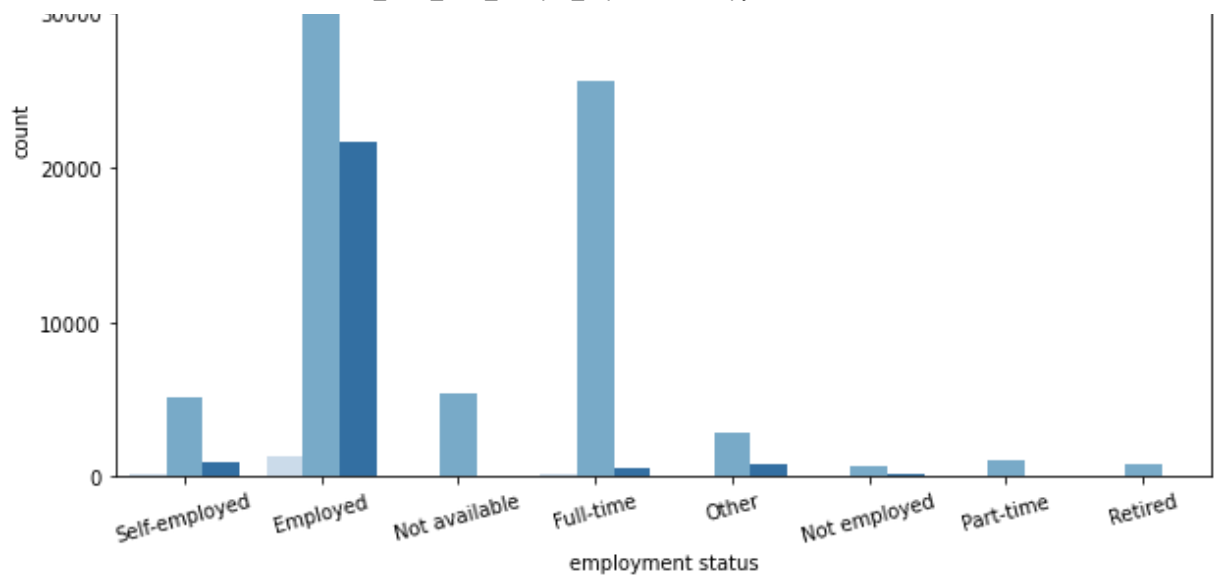
```
In [25]: plt.figure(figsize = [10, 30])

# Prosper rating vs. term
plt.subplot(4, 1, 1)
sb.countplot(data = df_loans, x = 'ProsperRating (Alpha)', hue = 'Term', palette
plt.xlabel('Prosper rating')
plt.title('Prosper rating vs. term')

# employment status vs. term
ax = plt.subplot(4, 1, 2)
sb.countplot(data = df_loans, x = 'EmploymentStatus', hue = 'Term', palette = 'B
plt.xticks(rotation = 15)
plt.xlabel('employment status')
plt.title('employment status vs. term')

# Prosper rating vs. employment status
ax = plt.subplot(4, 1, 3)
sb.countplot(data = df_loans, x = 'EmploymentStatus', hue = 'ProsperRating (Alpha
ax.legend(loc = 1, ncol = 2)
plt.xticks(rotation = 15)
plt.xlabel('employment status')
plt.title('Prosper rating vs. employment status');
```

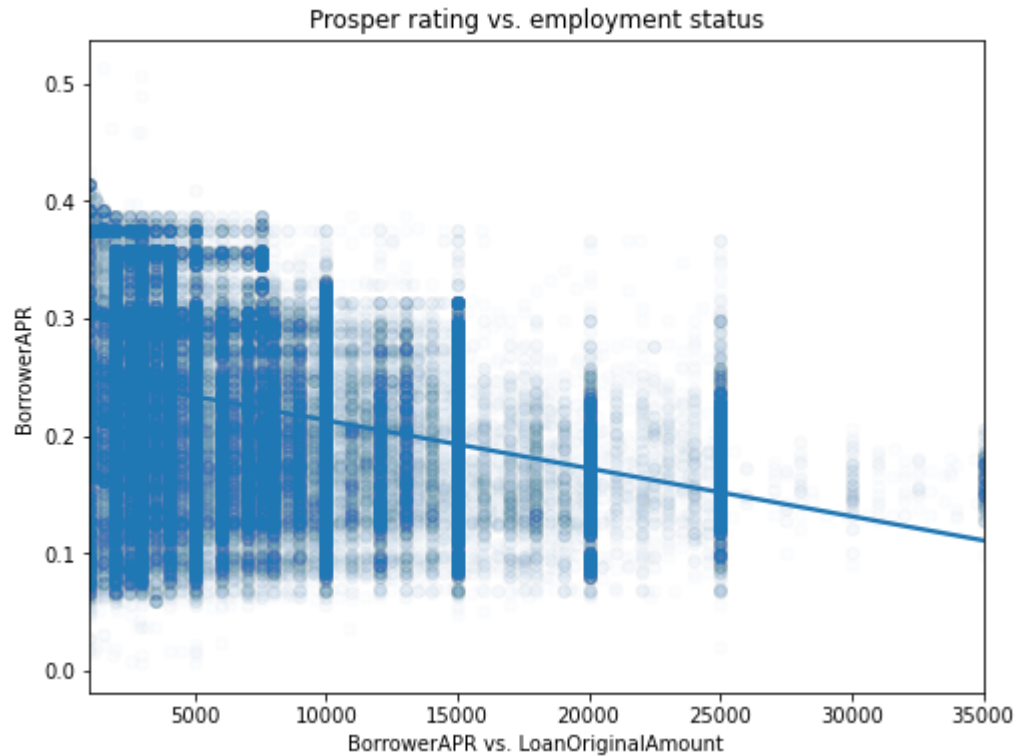




We can see that there is an interaction between term and Prosper rating. The most popular term was 36 and most of the employed especially with full-time jobs took that kind of loan. And of course, the higher proper ratings were given to the employed personnel.

Lets see how borrower APR and loan original amount are related

```
In [26]: # how borrower APR and Loan original amount are related
plt.figure(figsize = [8, 6])
sb.regplot(data = df_loans, x = 'LoanOriginalAmount', y = 'BorrowerAPR', scatter_
plt.xlabel('BorrowerAPR vs. LoanOriginalAmount')
plt.title('Prosper rating vs. employment status');
```

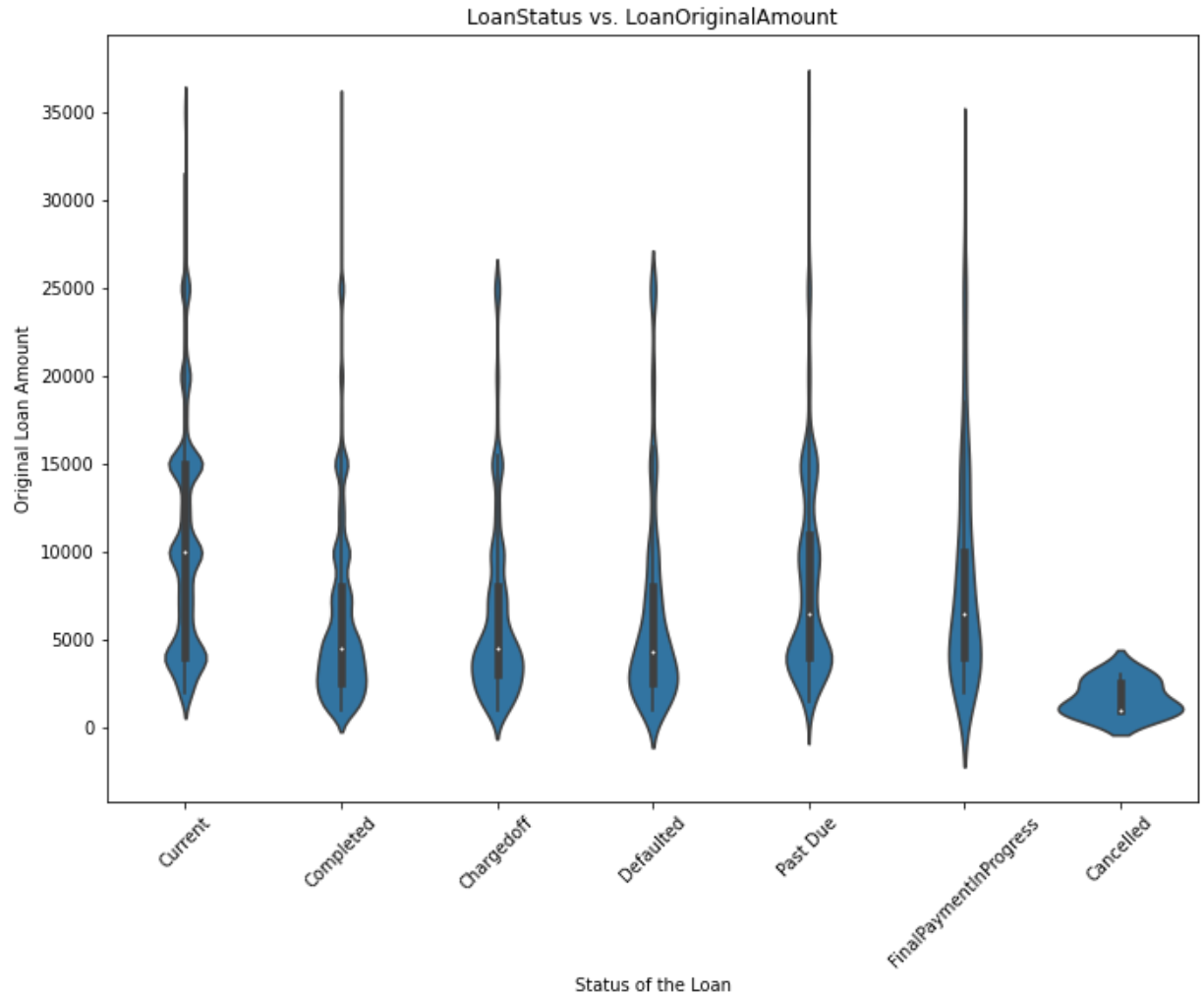


This relation shows that the range of APR decreases with the increase in the loan amount. Overall, the borrower's APR is negatively correlated with the loan amount.

Now the relation between LoanStatus and LoanOriginalAmount

```
In [27]: # plotting original loan amount with a violinplot against LoanStatus
plt.figure(figsize = [25, 18])

plt.subplot(2, 2, 2)
sb.violinplot(data = df_loans, x = 'LoanStatus', y = 'LoanOriginalAmount', color
plt.xticks(rotation=45)
plt.xlabel('Status of the Loan')
plt.ylabel('Original Loan Amount')
plt.title('LoanStatus vs. LoanOriginalAmount');
```



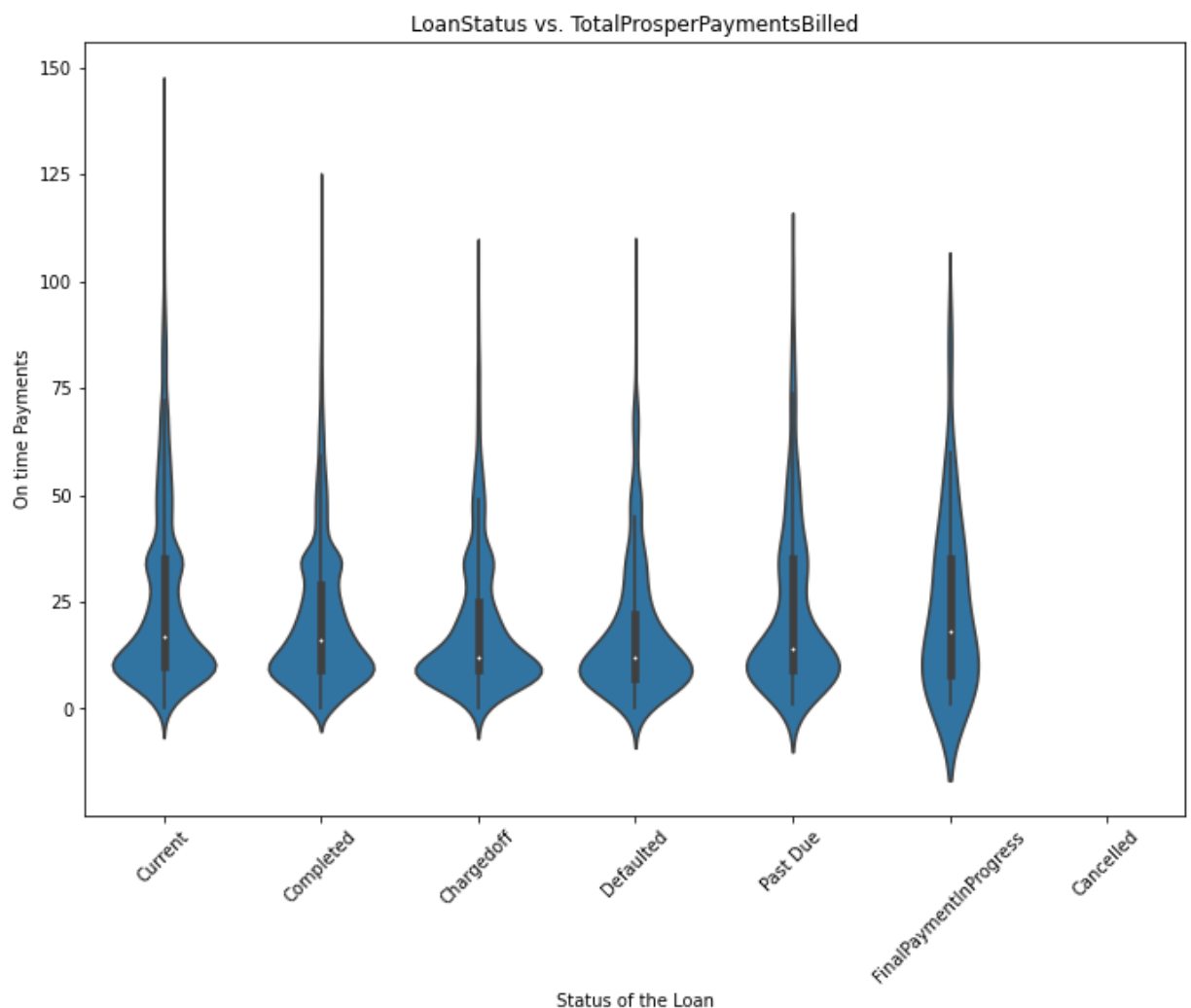
The original loan amount is about the same on average for loans that are completed, charged-off,

or defaulted. However, loans with past due payments have on average a higher original loan amount.

Now the relation between loan status and loan total prosper payments billed

```
In [28]: # plotting TotalProsperPaymentsBilled (On time Payments) with a violinplot against
plt.figure(figsize = [25, 18])

plt.subplot(2, 2, 2)
sb.violinplot(data = df_loans, x = 'LoanStatus', y = 'TotalProsperPaymentsBilled')
plt.xticks(rotation=45)
plt.xlabel('Status of the Loan')
plt.ylabel('On time Payments')
plt.title('LoanStatus vs. TotalProsperPaymentsBilled');
```

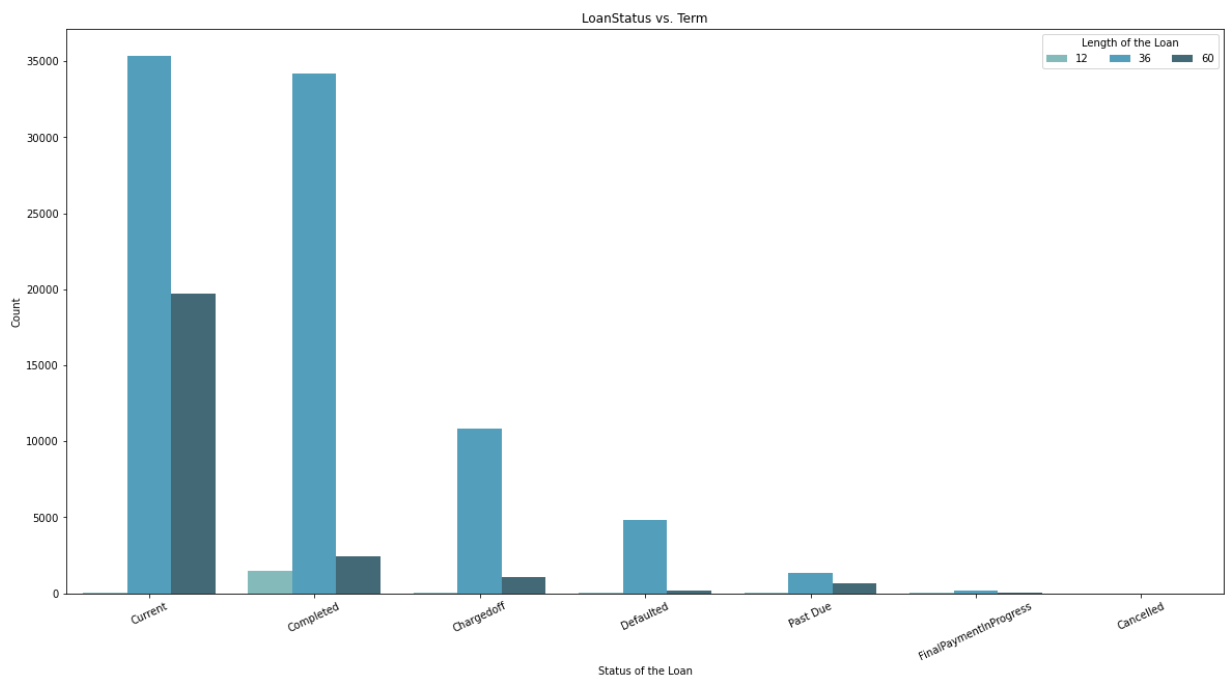


Complete loans have on average the highest number of on time payments while loans with the status charged-off and defaulted have the lowest.

Relation between loan status and term

```
In [29]: # plotting Length of the Loans against LoanStatus a clustered bar chart
plt.figure(figsize = [20,10])

ax = sb.countplot(data = df_loans, x = 'LoanStatus', hue = 'Term', palette = "GnBu")
plt.legend(loc = 1, ncol = 3, title = 'Length of the Loan')
plt.xticks(rotation = 25)
plt.xlabel('Status of the Loan')
ax.set_ylabel('Count')
plt.title('LoanStatus vs. Term');
```

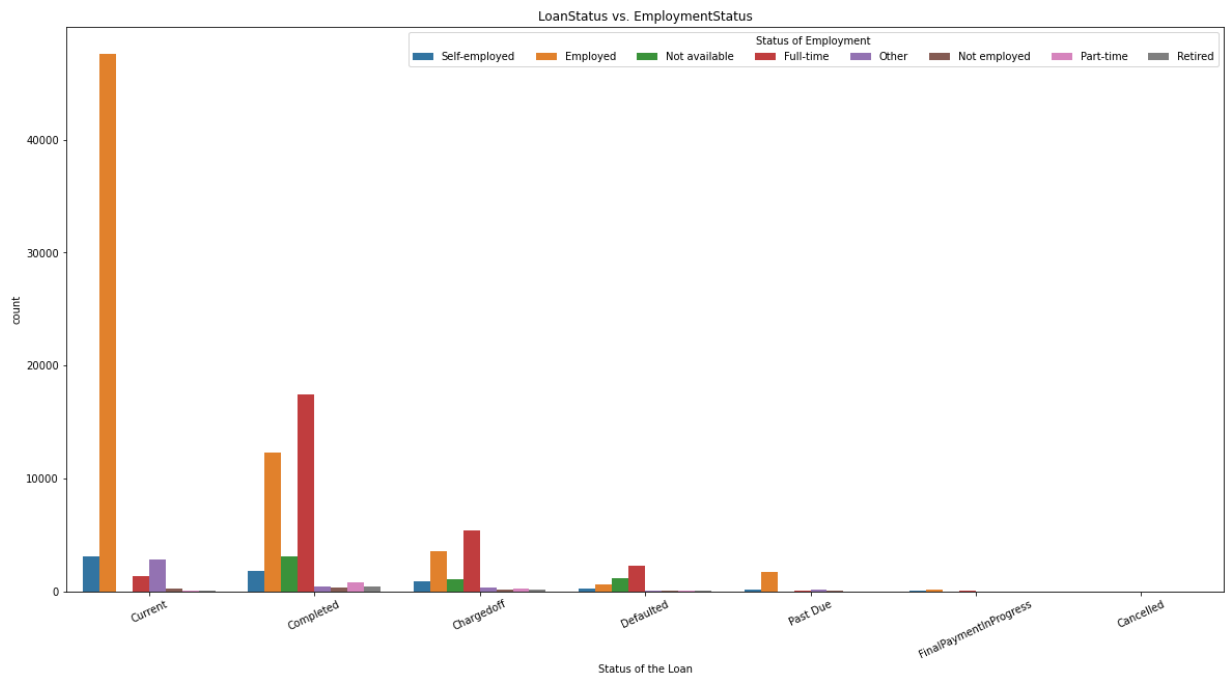


No matter what status a loan has, the most common length is 36 months and the least common is 60 months.

Relation between loan status and employment status

```
In [30]: # plotting EmploymentStatus against LoanStatus
plt.figure(figsize = [20,10])

sb.countplot(data = df_loans, x = 'LoanStatus', hue = 'EmploymentStatus', order = 
plt.legend(loc = 1, ncol = 8, title = 'Status of Employment')
plt.xticks(rotation = 25)
plt.xlabel('Status of the Loan')
plt.title('LoanStatus vs. EmploymentStatus');
```

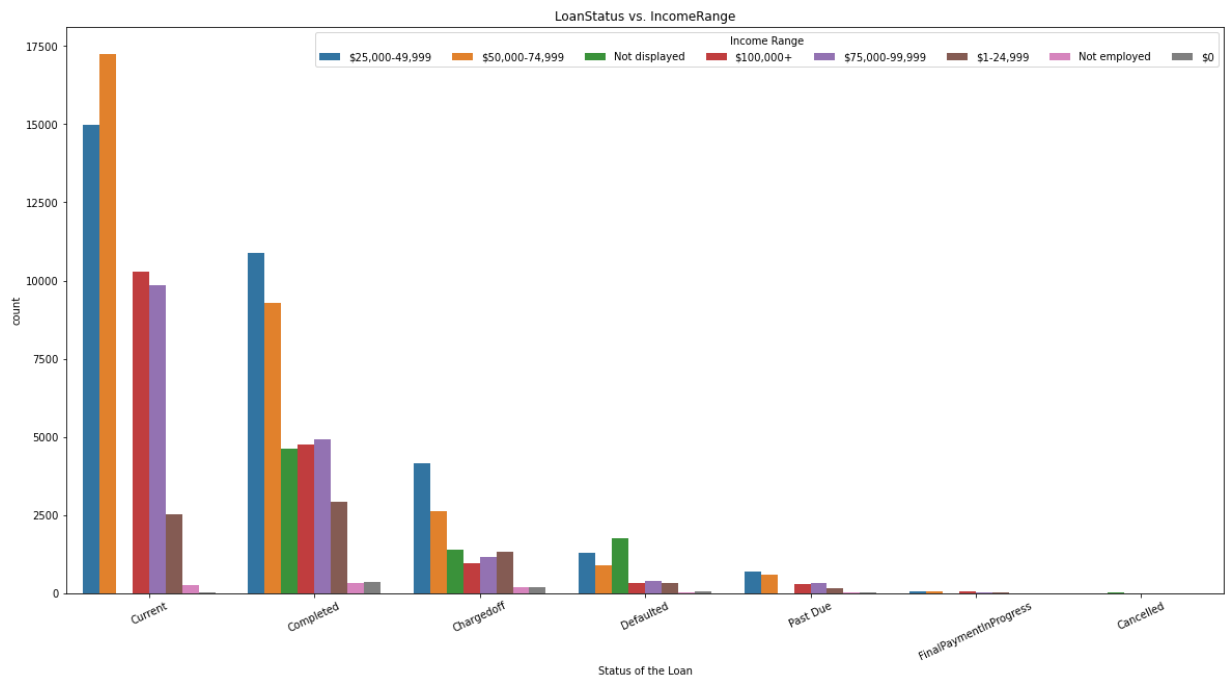


The status of the employment of the borrower seems not to have an impact on the outcome of the loan. but again the ones who take loans mostly are employed as also the ones who completed their loans are the full-time employed.

Relation between loan status and income range

```
In [31]: # plotting Income Range against LoanStatus
plt.figure(figsize = [20,10])

sb.countplot(data = df_loans, x = 'LoanStatus', hue = 'IncomeRange', order = df_loans['LoanStatus'].order)
plt.legend(loc = 1, ncol = 8, title = 'Income Range')
plt.xticks(rotation = 25)
plt.xlabel('Status of the Loan')
plt.title('LoanStatus vs. IncomeRange');
```



The borrowers who have an Income Range of (25000 - 74999) seem to be the ones who get more loans and pay them on time.

Relation between prosper score and loan status

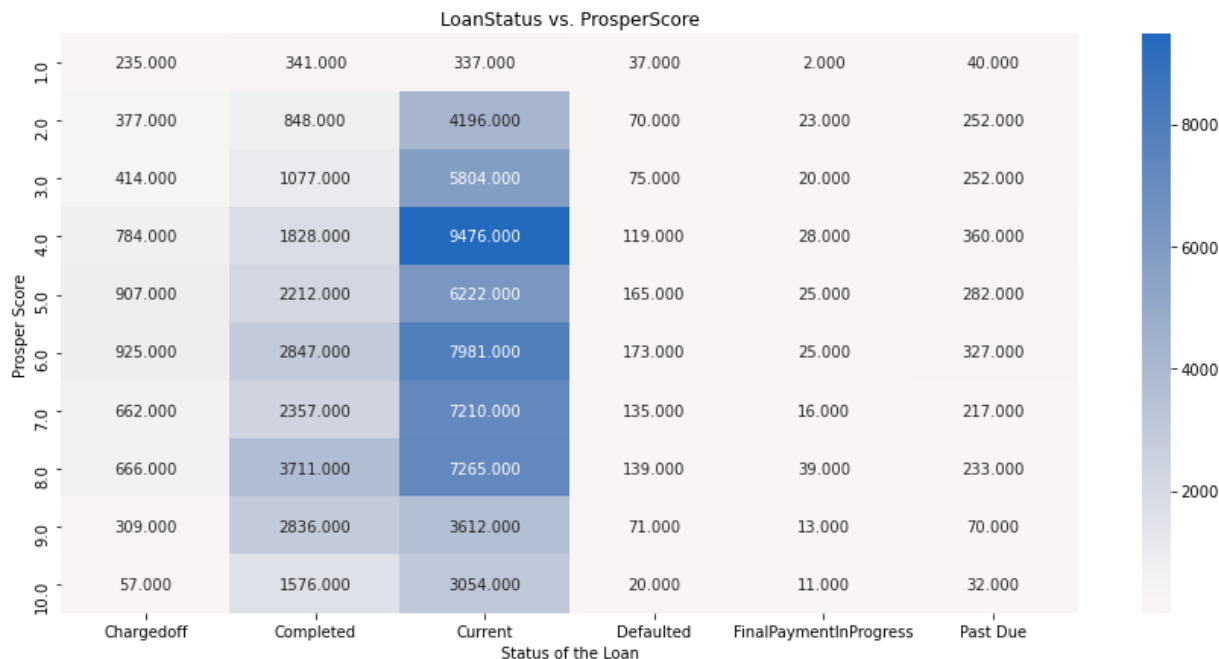
```
In [32]: # first I need to reshape the data
reshape = df_loans.groupby(['LoanStatus', 'ProsperScore']).size()
reshape = reshape.reset_index(name = 'count')
reshape = reshape.pivot(index = 'ProsperScore', columns = 'LoanStatus', values =
reshape
```

Out[32]:

LoanStatus	Chargedoff	Completed	Current	Defaulted	FinalPaymentInProgress	Past Due
ProsperScore						
1.0	235	341	337	37	2	40
2.0	377	848	4196	70	23	252
3.0	414	1077	5804	75	20	252
4.0	784	1828	9476	119	28	360
5.0	907	2212	6222	165	25	282
6.0	925	2847	7981	173	25	327
7.0	662	2357	7210	135	16	217
8.0	666	3711	7265	139	39	233
9.0	309	2836	3612	71	13	70
10.0	57	1576	3054	20	11	32


```
In [33]: # plotting ProsperScore against LoanStatus with a heat map
plt.figure(figsize = [15,7])

sb.heatmap(reshape, annot = True, fmt = '.3f', cmap = 'vlag_r', center = 0)
plt.xlabel('Status of the Loan')
plt.ylabel('Prosper Score')
plt.title('LoanStatus vs. ProsperScore');
```



The Prosper Score seems to affect the outcome of the loan. So the highest number of borrowers with completed loans has a prosper score of 8, while the highest number of borrowers with defaulted and charged-off loans have a prosper score of 6. Notice that the most common prosper score for borrowers with loans that are past due payments is 4.

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?

- BorrowerAPR and ProsperScore are negative because borrowers with a lower score are more likely to pay higher APR. CreditScore and BorrowerAPR are also negative because the higher the borrowers CreditScore the more trustworthy they are, therefore they received lower APR.
- People with higher ratings tend to be more trustworthy and therefore given lower BorrowerAPR.
- We can see the trend that the higher the CreditScore the lower the APR.
- We can see that there is an interaction between term and Prosper rating. The most popular term was 36 and most of the employed especially with full-time jobs took that kind of loan. And of course, the higher proper ratings were given to the employed personnel.
- The range of APR decreases with the increase of loan amount. Overall, the borrower's APR is negatively correlated with the loan amount. This means the more the loan amount the lower the APR
- The original loan amount is about the same on average for loans that are completed, charged-off or defaulted. However, loans with past due payments have on average a higher original loan amount.
- Complete loans have on average the highest number of on time payments while loans with the status charged-off and defaulted have the lowest.
- No matter what status a loan has, the most common length is 36 months, and the least common is 60 months.
- The status of the employment of the borrower seems not to have an impact on the outcome of the loan. but again the ones who take loans mostly are employed as also the ones who completed their loans are the full-time employed.
- The Prosper Score seems to affect the outcome of the loan. So the highest number of borrowers with completed loans has a prosper score of 8, while the highest number of borrowers with defaulted and charged-off loans have a prosper score of 6. Notice that the most common prosper score for borrowers with loans that are past due payments is 4.
- The borrowers who have an Income Range of (25000 - 74999) seem to be the ones who get more loans and pay them on time.

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

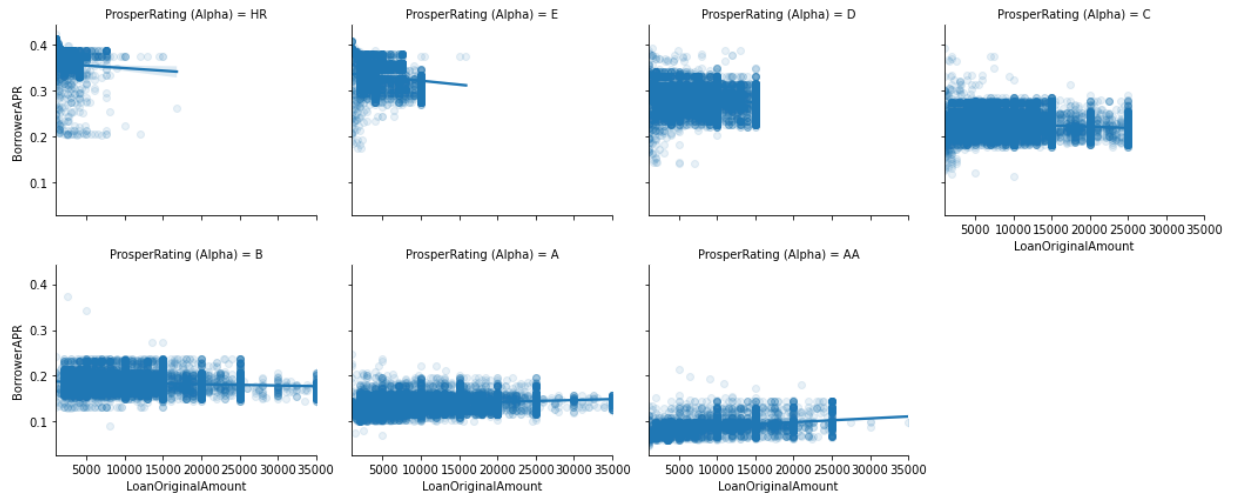
- There is an interaction between prospering rating and term. There are more 60-month loans on B and C ratings, while there are only 36 months loans for HR rating.

Multivariate Exploration

Prosper rating effect on relationship between borrower APR and loan original amount

```
In [34]: # Prosper rating effect on relationship of APR and Loan amount
g=sb.FacetGrid(data = df_loans, aspect = 1.2, height = 5, col = 'ProsperRating (Alpha)')
g.map(sb.regplot, 'LoanOriginalAmount', 'BorrowerAPR', x_jitter=0.04, scatter_kws={'s': 100})
g.add_legend();
```

C:\Users\Ahmed\anaconda3\lib\site-packages\seaborn\axisgrid.py:316: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
warnings.warn(msg, UserWarning)

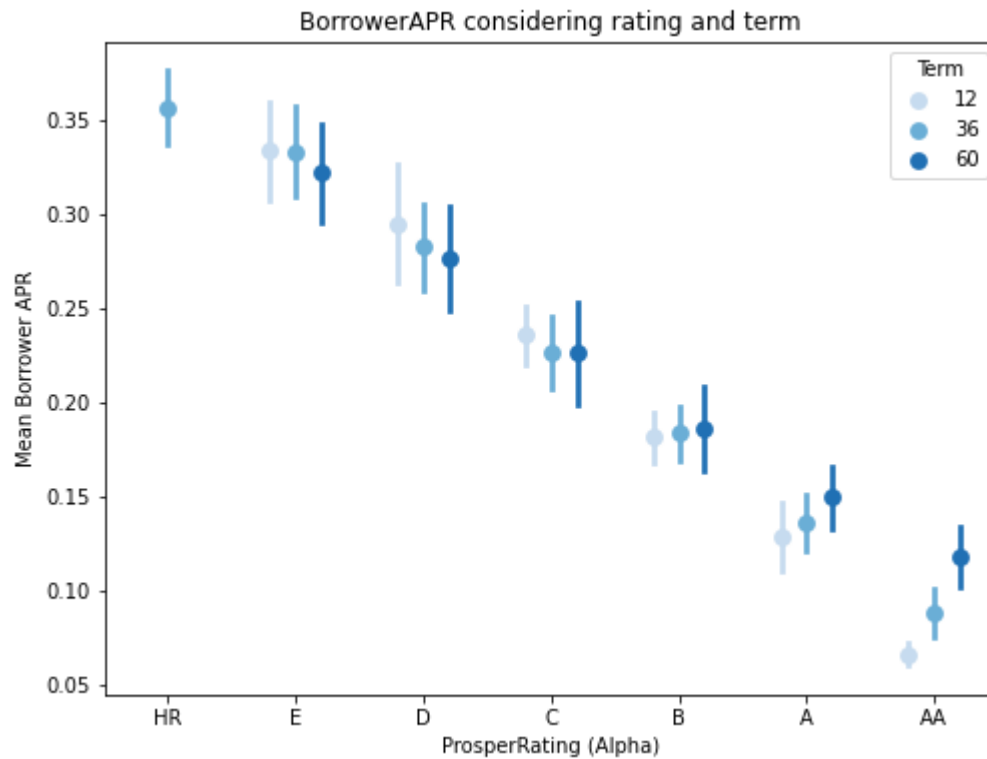


The loan amount increases with a better rating, the borrower's APR decreases with a better rating. The relationship between borrower APR and loan amount raises from negative to slightly positive when the prosper ratings are increased from HR to A or better. Maybe because people with A or AA ratings tend to borrow more money, and pay on time.

BorrowerAPR considering rating and term

```
In [35]: # BorrowerAPR considering rating and term
fig = plt.figure(figsize = [8,6])

ax = sb.pointplot(data = df_loans, x = 'ProsperRating (Alpha)', y = 'BorrowerAPR')
plt.title('BorrowerAPR considering rating and term')
plt.ylabel('Mean Borrower APR')
ax.set_yticklabels([],minor = True);
```

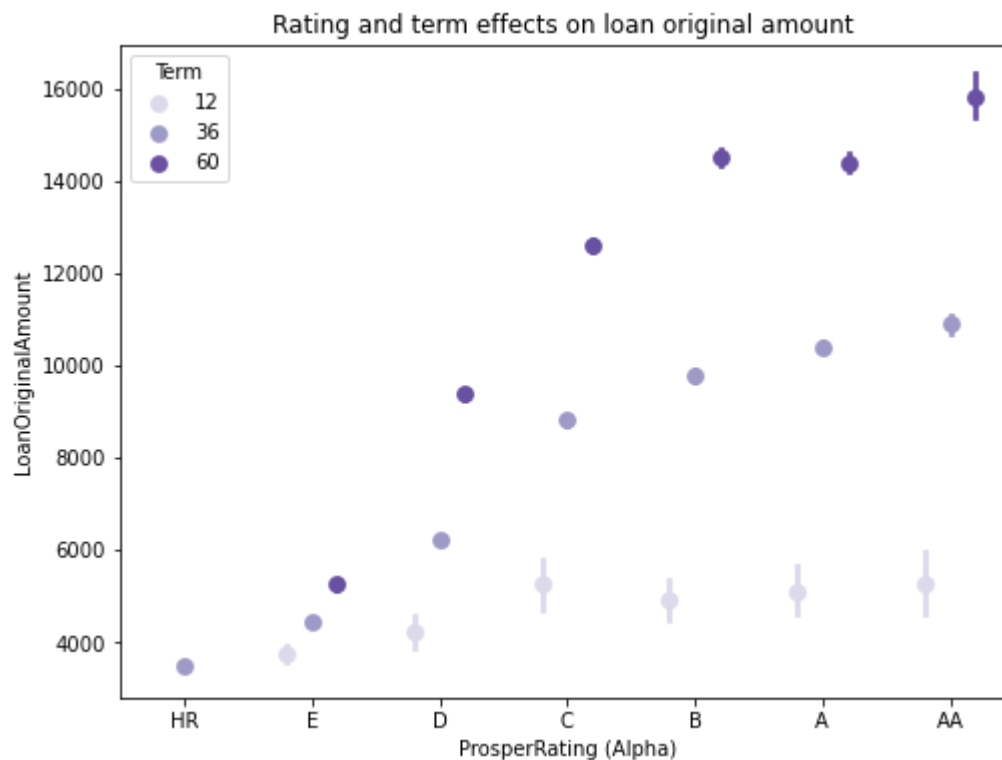


Notice that for prosper rating from HR to D the borrower APR increases with the decrease of borrow term, then it starts to shift from C to AA rating.

The rating and term effects on loan original amount

```
In [36]: # Rating and term effects on loan original amount
fig, ax = plt.subplots(figsize=[8,6])

sb.pointplot(data = df_loans, x = 'ProsperRating (Alpha)', y = 'LoanOriginalAmount',
             errorbar = 'ci', errorbar_kwargs = {'ci': 'ci.j', 'ci_low': 'ci_low', 'ci_high': 'ci_high'})
plt.title('Rating and term effects on loan original amount');
```

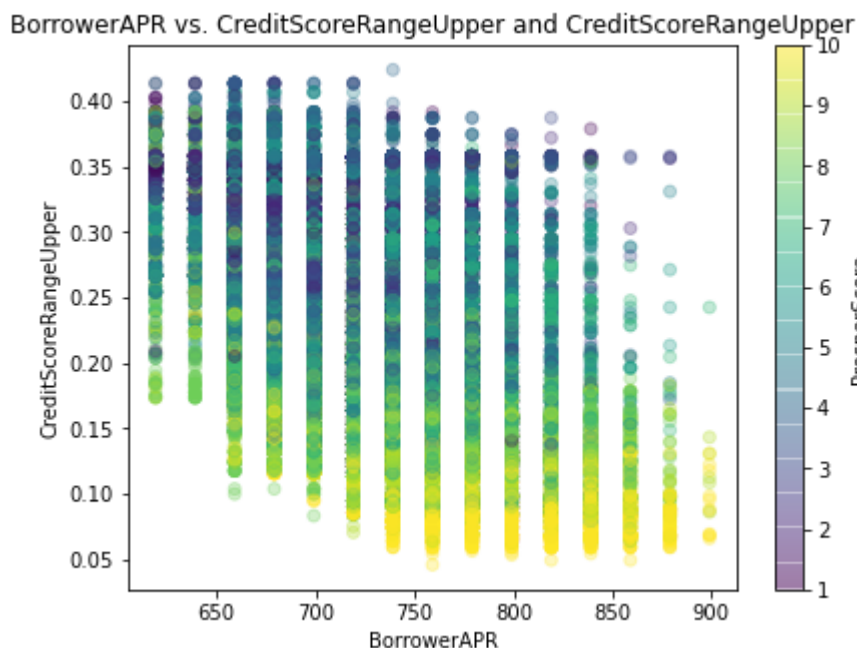


Notice that with a better prosper rating, the loan amount of all three terms increases.

BorrowerAPR considering CreditScoreRangeUpper and ProsperScore

```
In [37]: # BorrowerAPR vs. CreditScoreRangeUpper and CreditScoreRangeUpper
plt.figure(figsize = [15, 5])

plt.subplot(1, 2, 2)
plt.scatter(data = df_loans, x = 'CreditScoreRangeUpper', y = 'BorrowerAPR', c = 'ProsperScore')
plt.colorbar(label = 'ProsperScore')
plt.title('BorrowerAPR vs. CreditScoreRangeUpper and CreditScoreRangeUpper')
plt.xlabel('BorrowerAPR')
plt.ylabel('CreditScoreRangeUpper');
```



Notice that CreditScoreRangeUpper increase as BorrowerAPR decrease in the plots, this proves that CreditScoreRangeUpper and ProsperScore negatively correlated to BorrowerAPR.

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 loan amount increases with a better rating, the borrower APR decreases with a better rating. The relationship between borrower APR and loan amount raises from negative to slightly positive when the prosper ratings are increased from HR to A or better and pay on time.
- Notice that for prosper rating from HR to D the borrower APR increases with the decrease of borrow term, then it starts to shift from C to AA rating.
- Notice that with a better prosper rating, the loan amount of all terms increases.
- Notice that CreditScoreRangeUpper increase as BorrowerAPR decrease in the plots, This proves that CreditScoreRangeUpper and ProsperScore negatively correlated to BorrowerAPR.

Were there any interesting or surprising interactions between features?

- The borrower APR increases with the decrease of borrow term for people with HR to D ratings. Then it starts to shift for people with C to AA ratings, the APR decreases with the increase of borrow term.

In []: