

(Loan Data from Prosper)

by (Ahmed Tarek)

Investigation Overview

In this presentation I will attempt to answer some questions:

- What attributes affect the borrower's annual percentage rate?

This question is helpful for borrowers to decide on taking loans.

- What attributes affect a loan's outcome status?

This is very critical for banks to minimize the risk and to set the right interest rate.

- insight on increasing Prosper profits from loans.

Dataset Overview

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

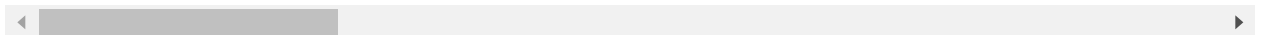
%matplotlib inline
```

```
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

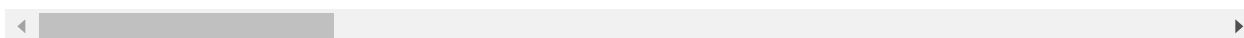
```

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

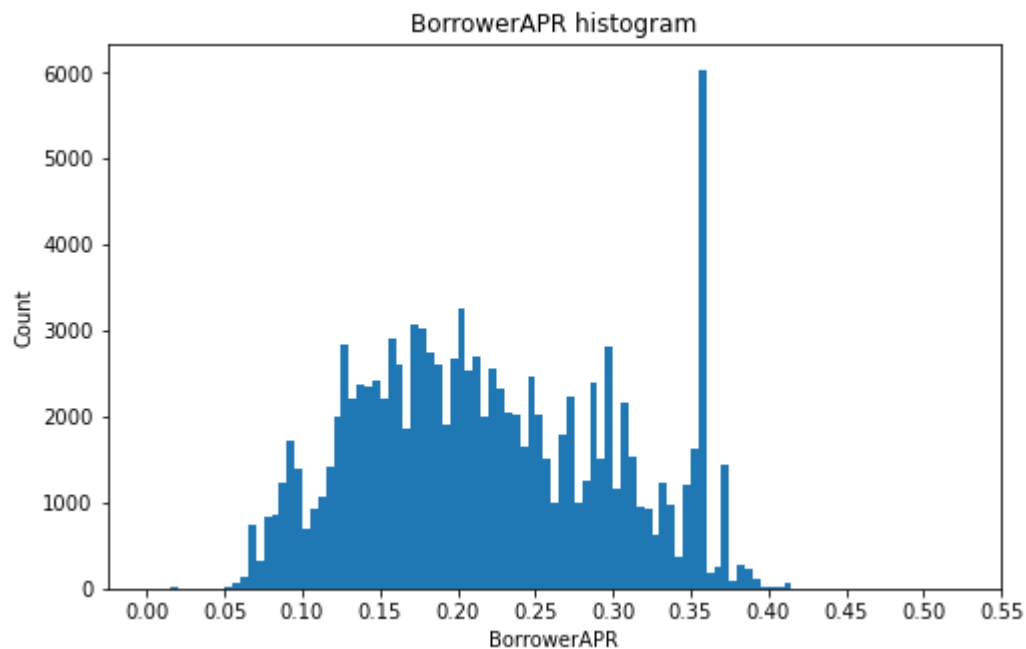


Univariate Exploration

Lets take a look at the BorrowerAPR

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.

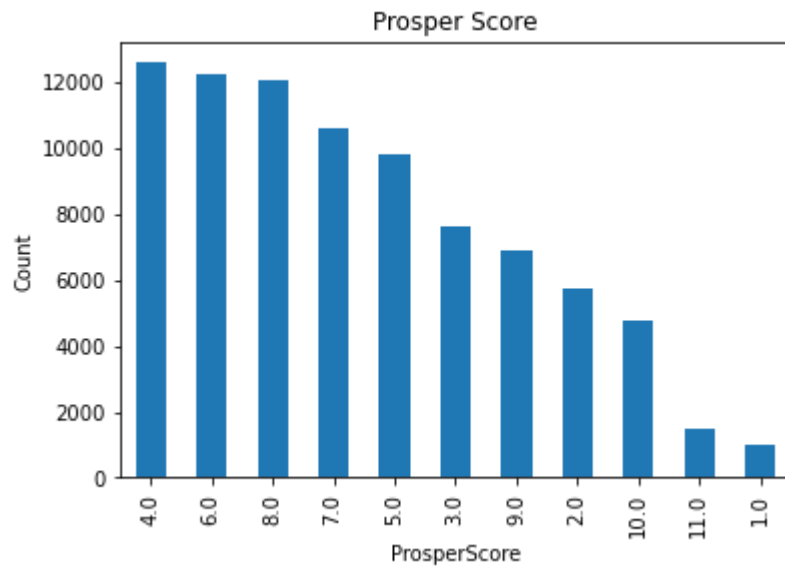
```
In [5]: bins = np.arange(0, df_loans.BorrowerAPR.max(), 0.005)
plt.figure(figsize=[8, 5])
plt.hist(data = df_loans, x = 'BorrowerAPR', bins = bins)
plt.xticks(np.arange(0, df_loans.BorrowerAPR.max()+0.05, 0.05))
plt.xlabel('BorrowerAPR')
plt.ylabel('Count')
plt.title('BorrowerAPR histogram');
```



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

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.

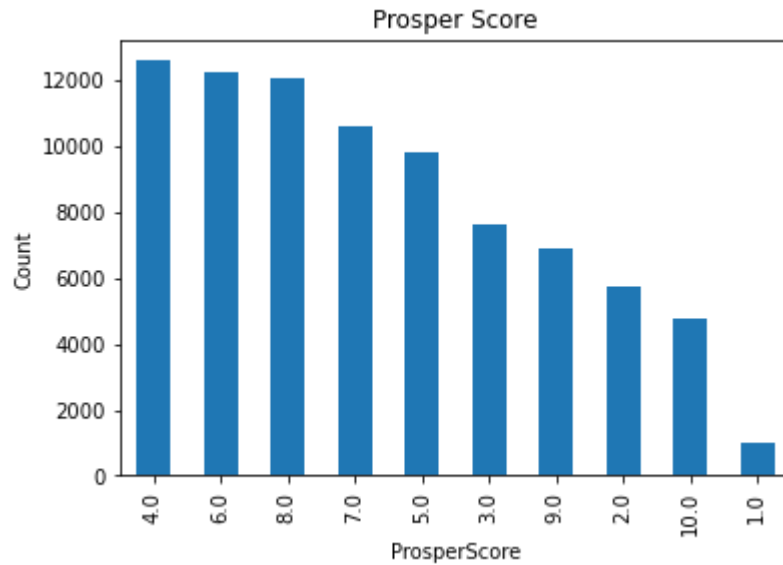
```
In [6]: df_loans.ProsperScore.value_counts().plot(kind='bar')  
plt.xlabel('ProsperScore')  
plt.ylabel('Count')  
plt.title('Prosper Score');
```



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]: df_loans = df_loans[df_loans.ProsperScore != 11]
```

```
In [8]: df_loans.ProspersScore.value_counts().plot(kind='bar')
plt.xlabel('ProspersScore')
plt.ylabel('Count')
plt.title('Prospers Score');
```



Now lets look at ProsperRating (Alpha) and Occupation

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

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

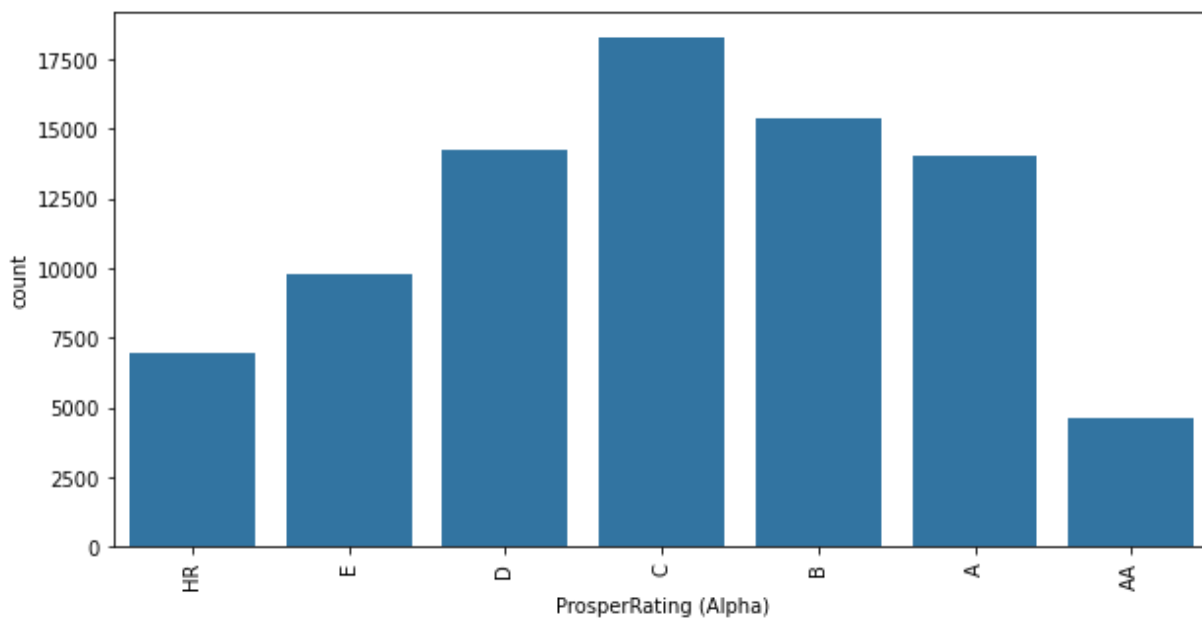
```
In [9]: rate_order = ['HR', 'E', 'D', 'C', 'B', 'A', 'AA']
ordered_var = pd.api.types.CategoricalDtype(ordered = True, categories = rate_order)
df_loans['ProsperRating (Alpha)'] = df_loans['ProsperRating (Alpha)'].astype(ordered_var)

<ipython-input-9-983a76435ecb>:3: 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
df_loans['ProsperRating (Alpha)'] = df_loans['ProsperRating (Alpha)'].astype(ordered_var)
```

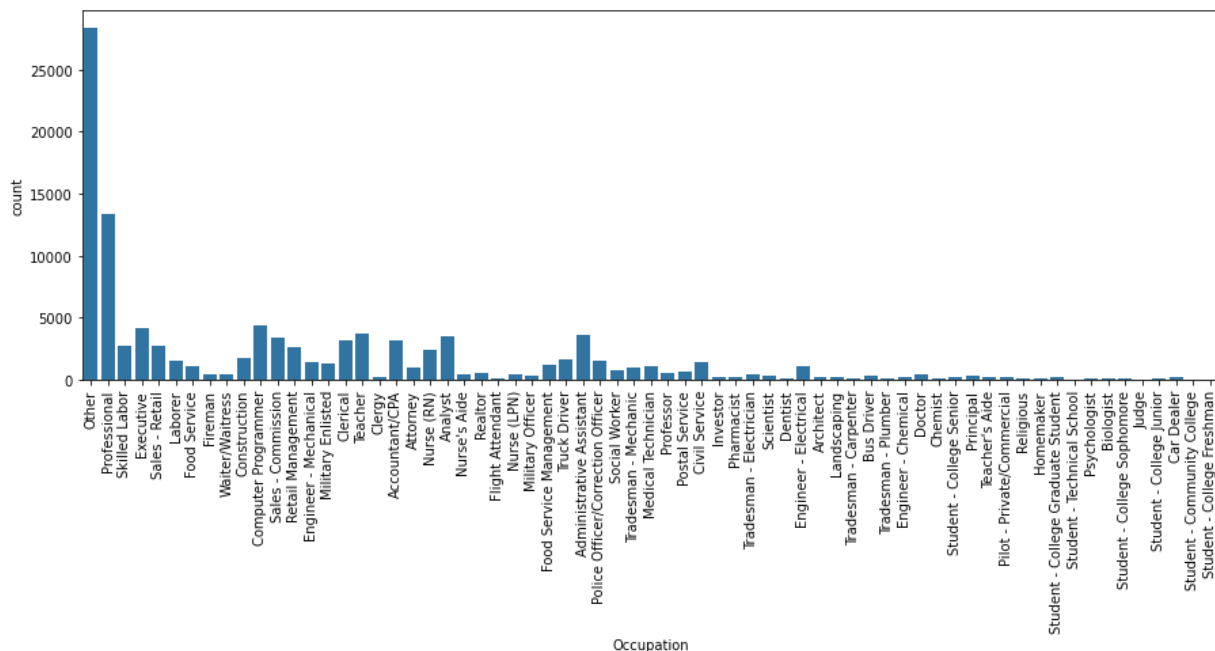
```
In [10]: fig = plt.subplots(figsize = [10, 5])

default_color = sb.color_palette()[0]
sb.countplot(data = df_loans, x = 'ProsperRating (Alpha)', color = default_color)
plt.xticks(rotation=90);
```




```
In [11]: fig = plt.subplots(figsize = [15, 5])

default_color = sb.color_palette()[0]
sb.countplot(data = df_loans, x = 'Occupation', color = default_color)
plt.xticks(rotation=90);
```

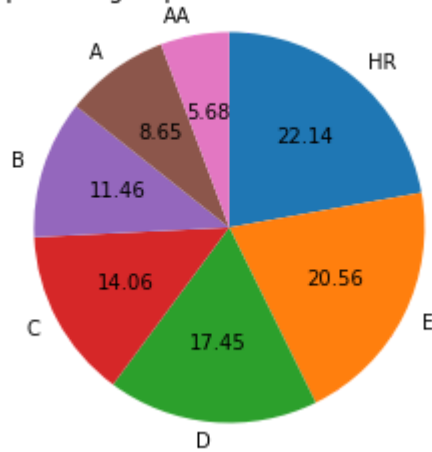


Now, let's compare the Prosper rating (Alpha) mean with the Borrower APR mean

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.

```
In [12]: 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



I am interested in knowing more about the LoanOriginalAmount

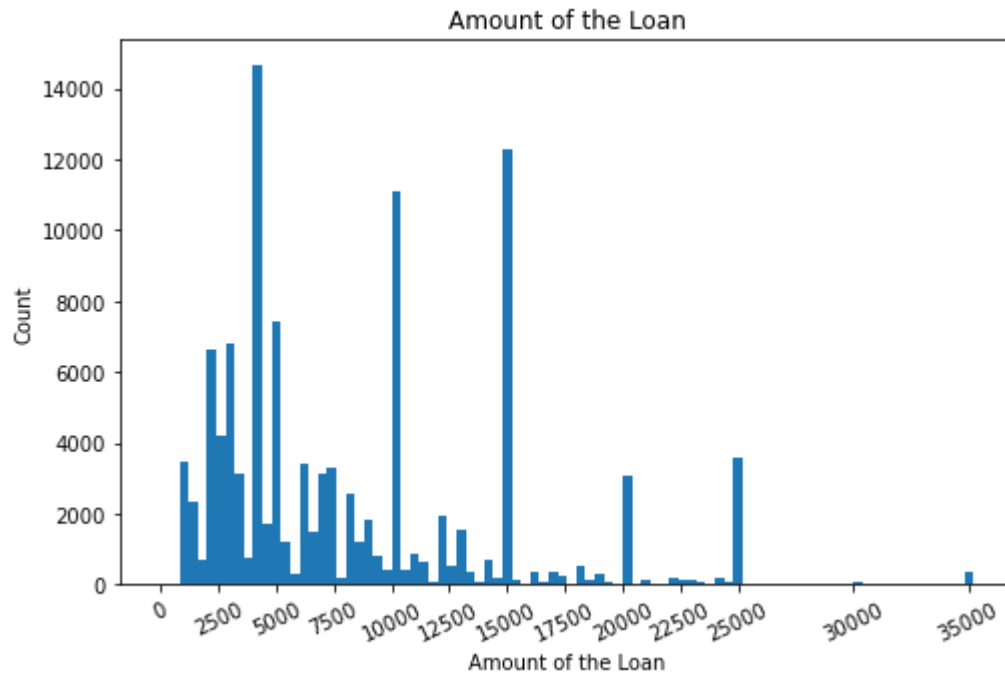
The histogram has several peaks at around 4,000, 10,000, and 15,000. But most of the values are in the lower end between 2500 and 10,000. The most loaned amounts are 4,000 and 15,000. The mean loan amount is between 8200.

```
In [13]: df_loans.LoanOriginalAmount.mean()
```

```
Out[13]: 8252.601132635735
```

```
In [14]: binsize = 400
bins = np.arange(0, df_loans.LoanOriginalAmount.max()+binsize, binsize)

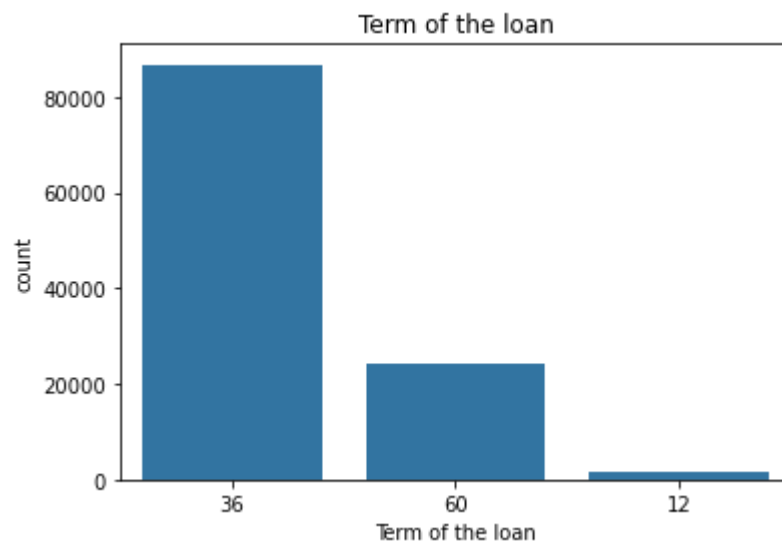
plt.figure(figsize=[8, 5])
plt.hist(data = df_loans, x = 'LoanOriginalAmount', bins = bins)
plt.xlabel('Amount of the Loan')
plt.ylabel('Count')
plt.title('Amount of the Loan')
plt.xticks([0,2500,5000,7500,10000,12500,15000,17500,20000,22500,25000,30000,35000])
```



Now lets see if there is a relation between the LoanOriginalAmount and Term

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.

```
In [15]: 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');
```



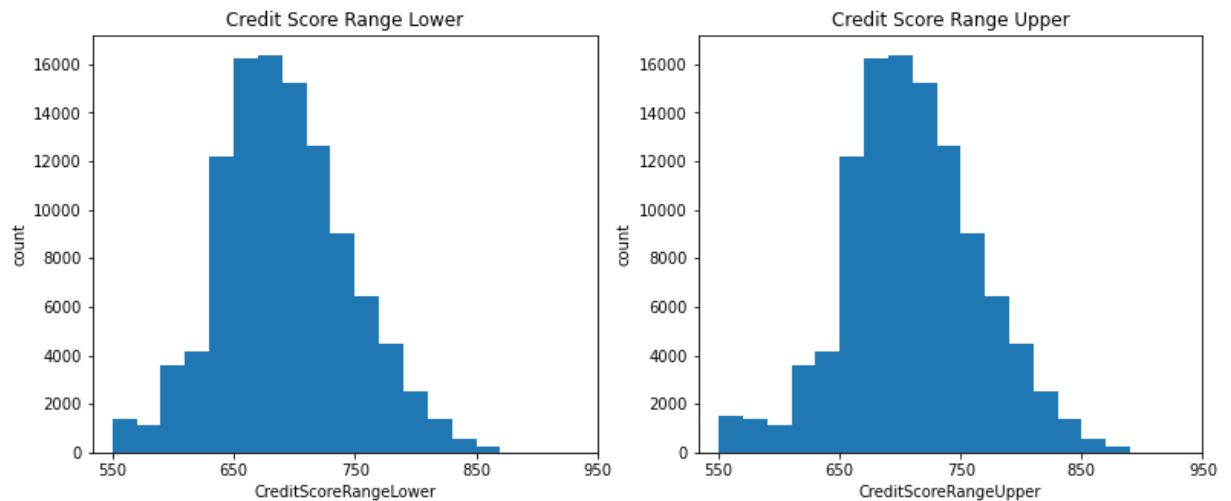
Lets look at the CreditScoreRangeLower and CreditScoreRangeUpper

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

```
In [16]: 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');

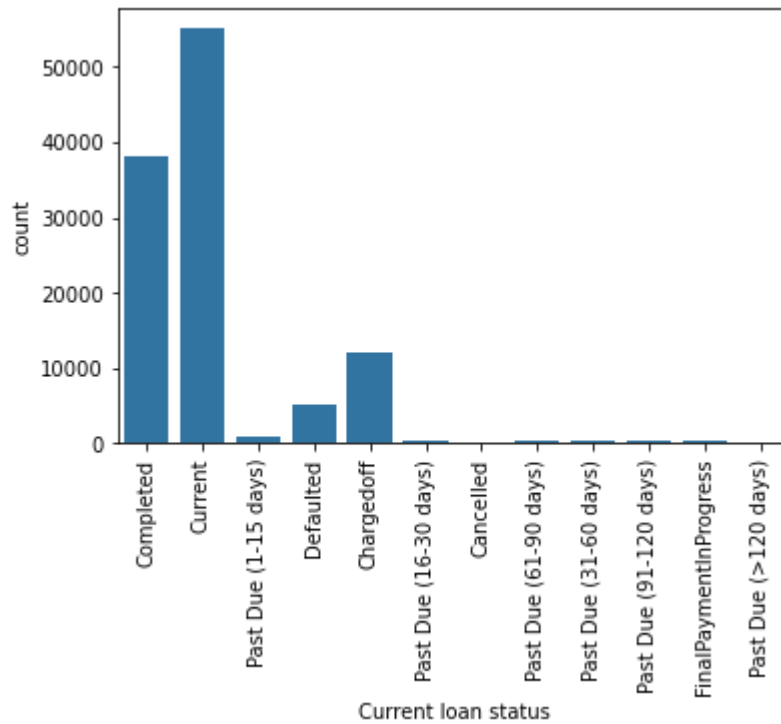
plt.subplot(1, 2, 2)
bins = np.arange(550, df_loans.CreditScoreRangeUpper.max(), 20)
plt.hist(data = df_loans, x = 'CreditScoreRangeUpper', bins = bins)
plt.xticks(np.arange(550, 1000, 100))
plt.title('Credit Score Range Upper')
plt.xlabel('CreditScoreRangeUpper')
plt.ylabel('count');
```



Now lets take a look at the loan status

Most of the loans are current or completed.

```
In [17]: 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 [18]: df_loans.LoanStatus = df_loans.LoanStatus.replace(['Past Due (1-15 days)', 'Past Due (16-30 days)', 'Past Due (31-60 days)', 'Past Due (61-90 days)', 'Past Due (91-120 days)', 'Past Due (>120 days)'], 'Past Due')

df_loans.LoanStatus.value_counts()
```

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

A value is trying to be set on a copy of a slice from a DataFrame.

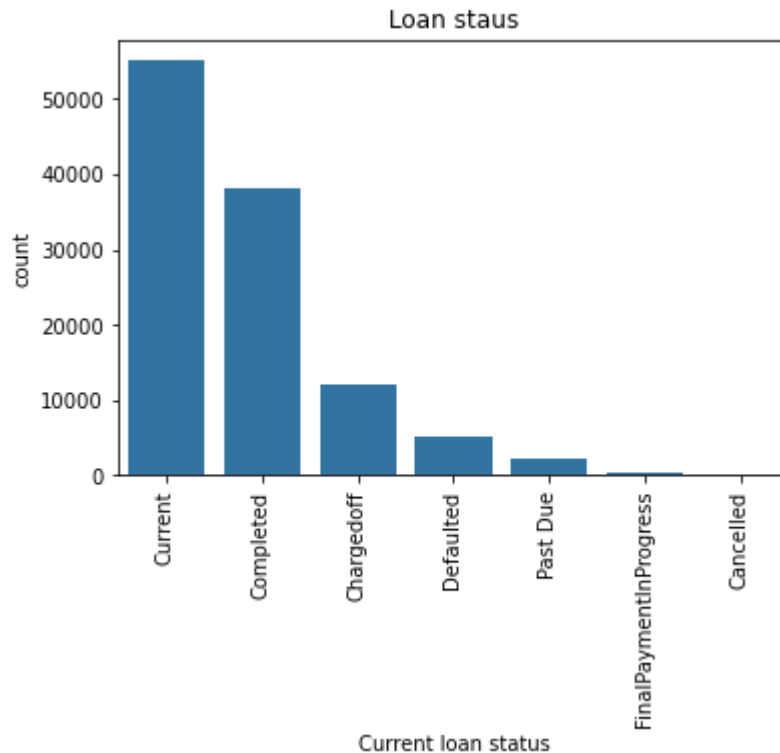
Try using `.loc[row_indexer,col_indexer] = value` instead

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)

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

```
Out[18]: Current          55157
Completed        38043
Chargedoff       11992
Defaulted        5017
Past Due         2065
FinalPaymentInProgress  202
Cancelled         5
Name: LoanStatus, dtype: int64
```

```
In [19]: base_color = sb.color_palette()[0]
sb.countplot(data = df_loans, x = 'LoanStatus', color = base_color, order = df_loans['LoanStatus'].value_counts().index)
plt.xlabel('Current loan status')
plt.xticks(rotation = 90)
plt.title('Loan status');
```

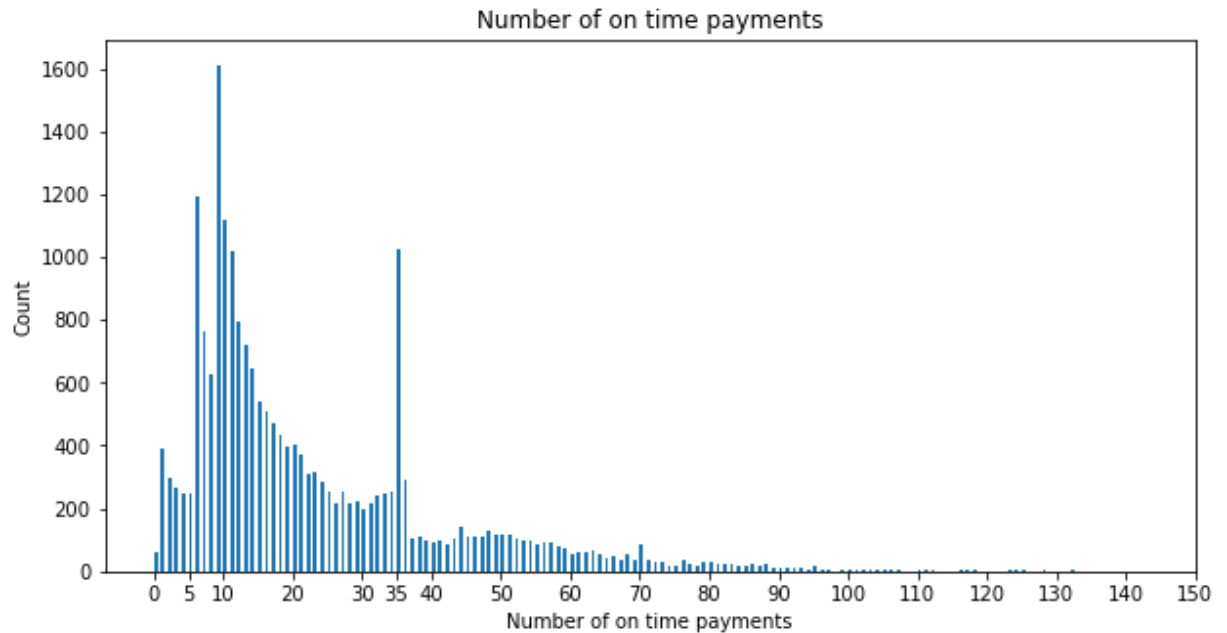


Now lets look at the TotalProsperPaymentsBilled (Number of on time payments)

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.


```
In [20]: binsize = 0.5
bins = np.arange(df_loans.TotalProsperPaymentsBilled.min(), df_loans.TotalProsperPaymentsBilled.max(), binsize)

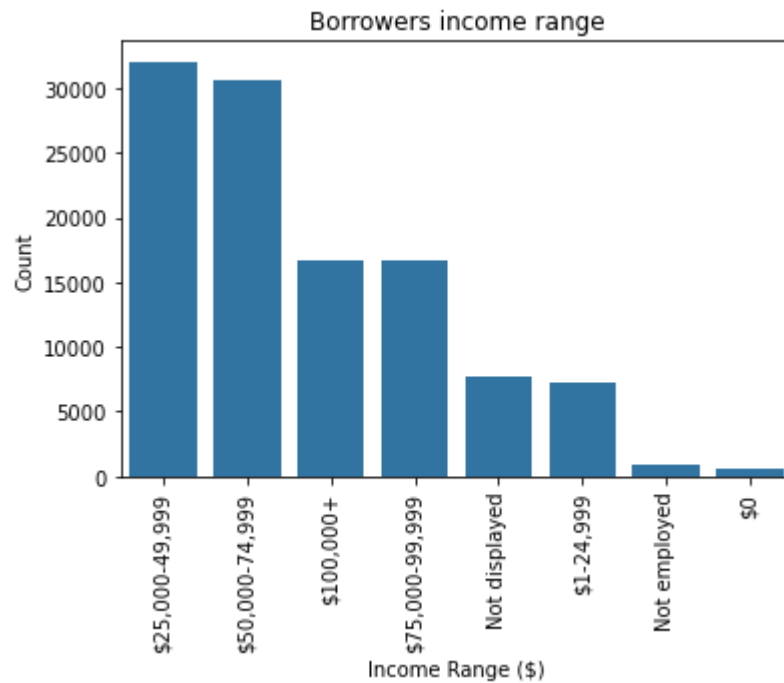
plt.figure(figsize=[10, 5])
plt.hist(data = df_loans, x = 'TotalProsperPaymentsBilled', bins = bins)
plt.xlabel('Number of on time payments')
plt.ylabel('Count')
plt.title('Number of on time payments')
plt.xticks([0,5,10,20,30,35,40,50,60,70,80,90,100,110,120,130,140,150])
plt.show()
```



Now the IncomeRange

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.

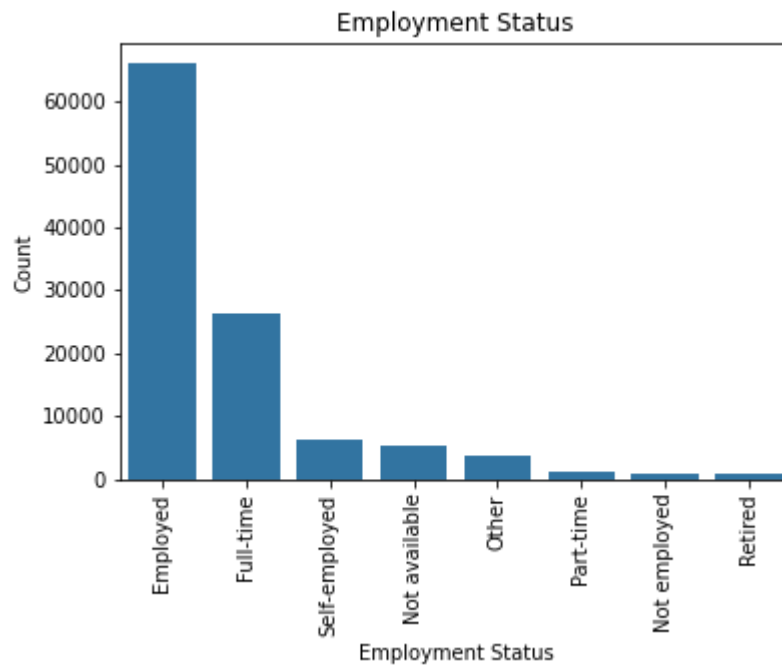
```
In [21]: sb.countplot(data= df_loans, x= 'IncomeRange', color= base_color, order = df_loar
plt.xticks(rotation=90)
plt.xlabel('Income Range ($)')
plt.ylabel('Count')
plt.title('Borrowers income range');
```



Lets check the EmploymentStatus to find out

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

```
In [22]: 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');
```

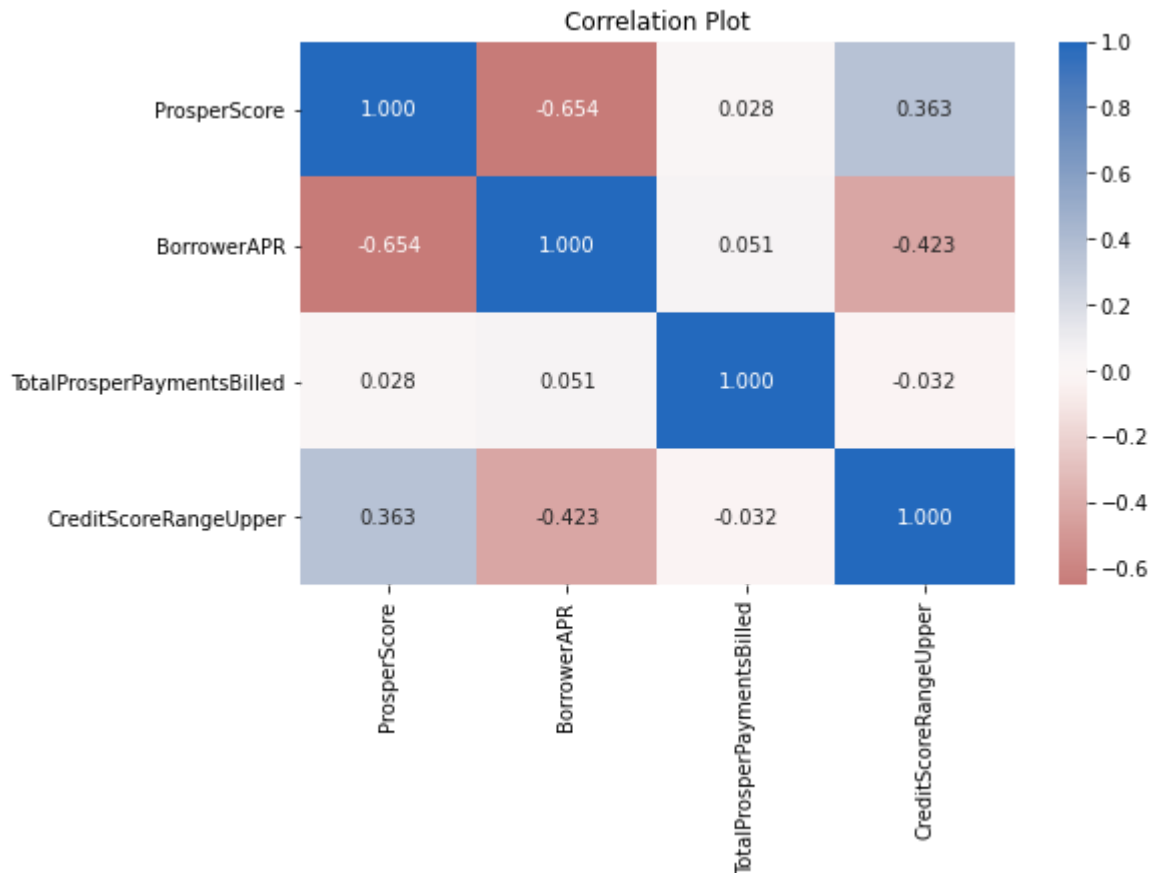


Bivariate Exploration

Lets look at the numeric variables

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.

```
In [23]: 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');
```

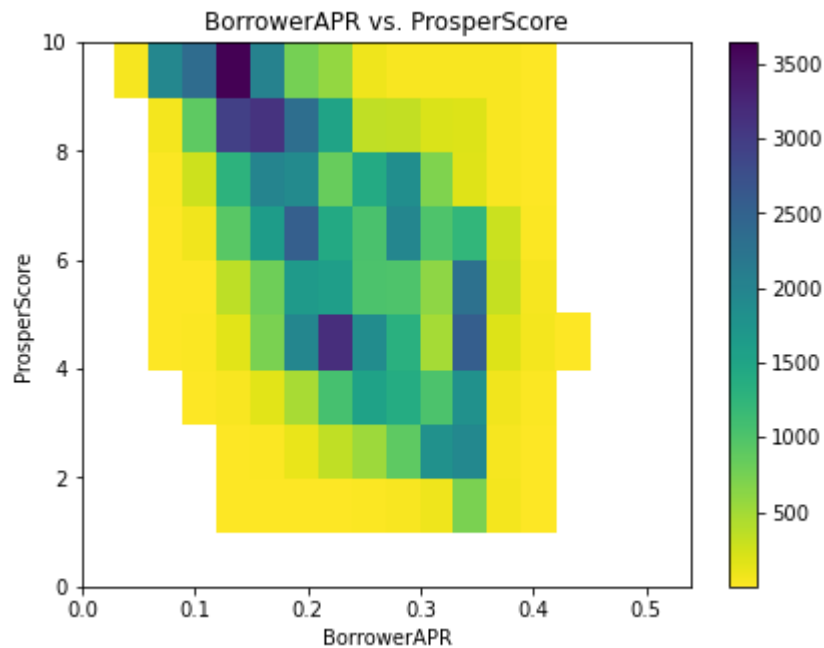


Lets look more closely at ProsperScore vs BorrowerAPR

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

```
In [24]: 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');
```

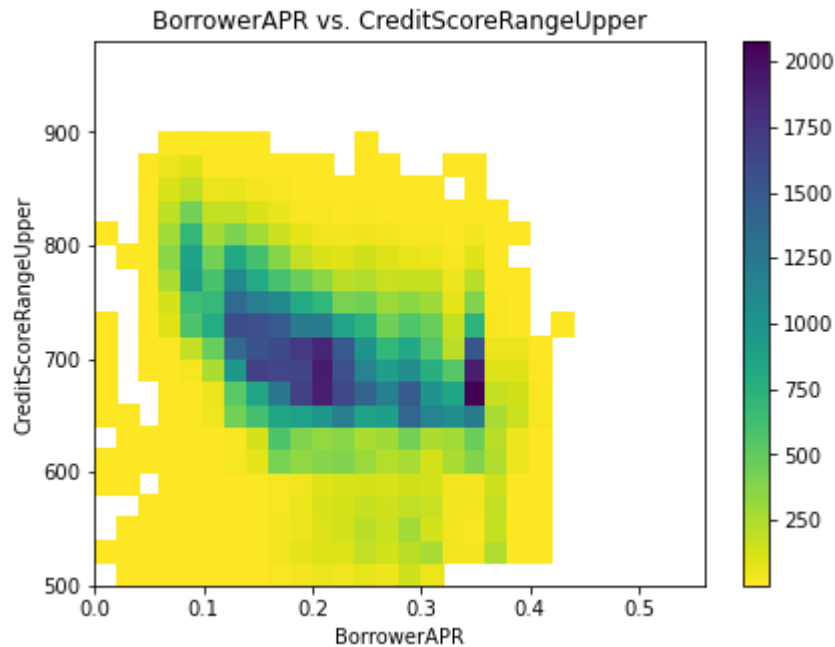


Now BorrowerAPR vs. CreditScoreRangeUpper

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

```
In [25]: 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');
```

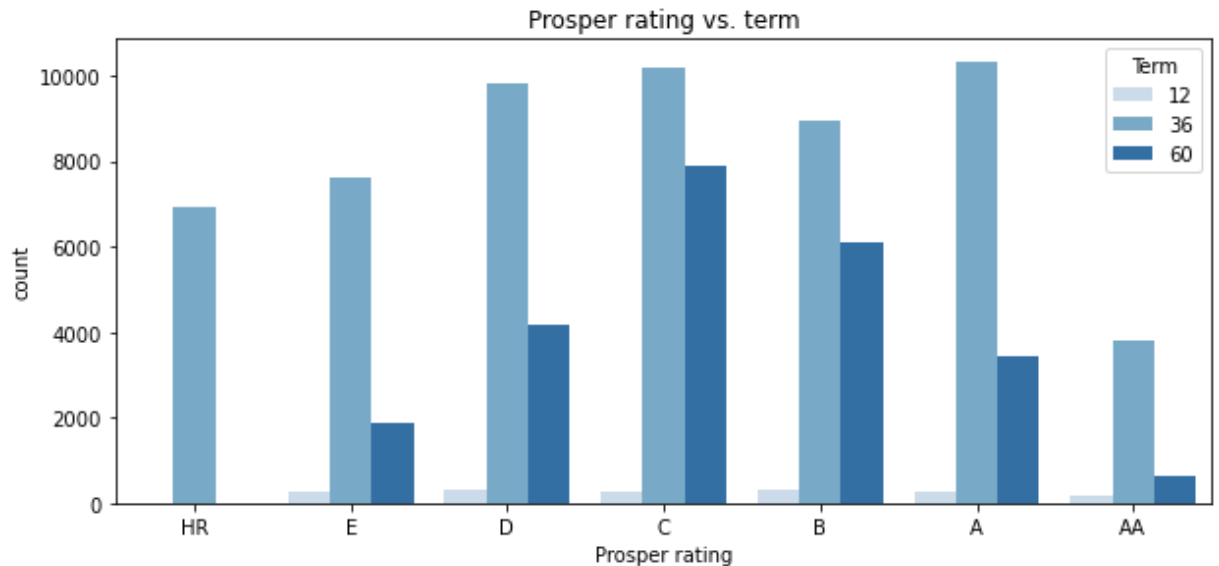


let's look at relationships between the categorical features.

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.

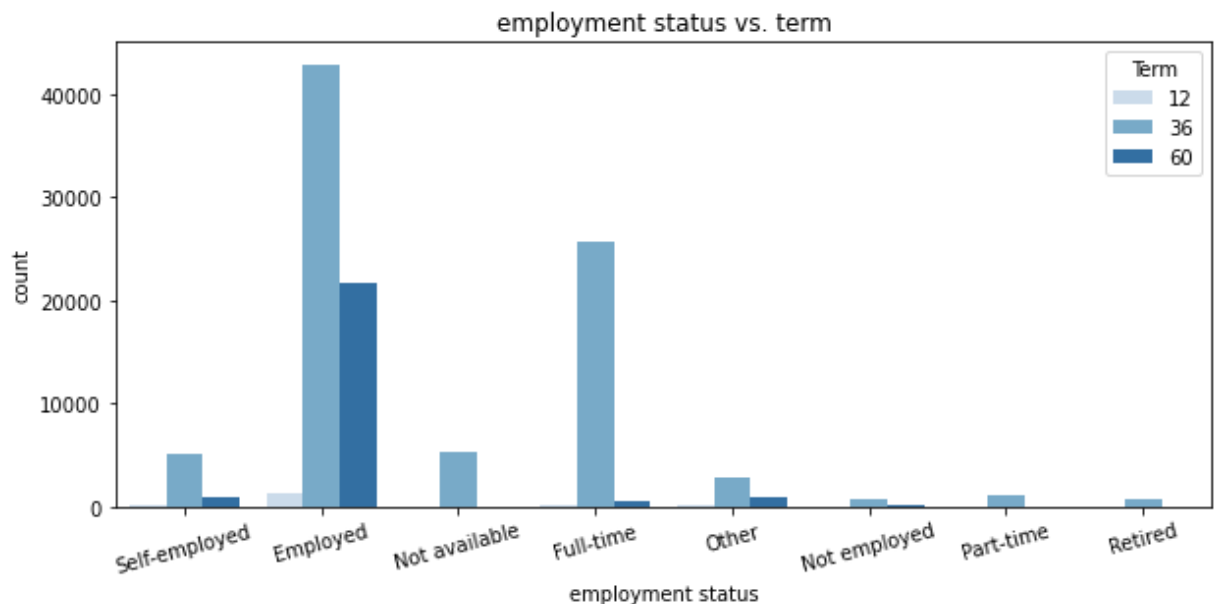
```
In [26]: plt.figure(figsize = [10, 20])

# 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');
```

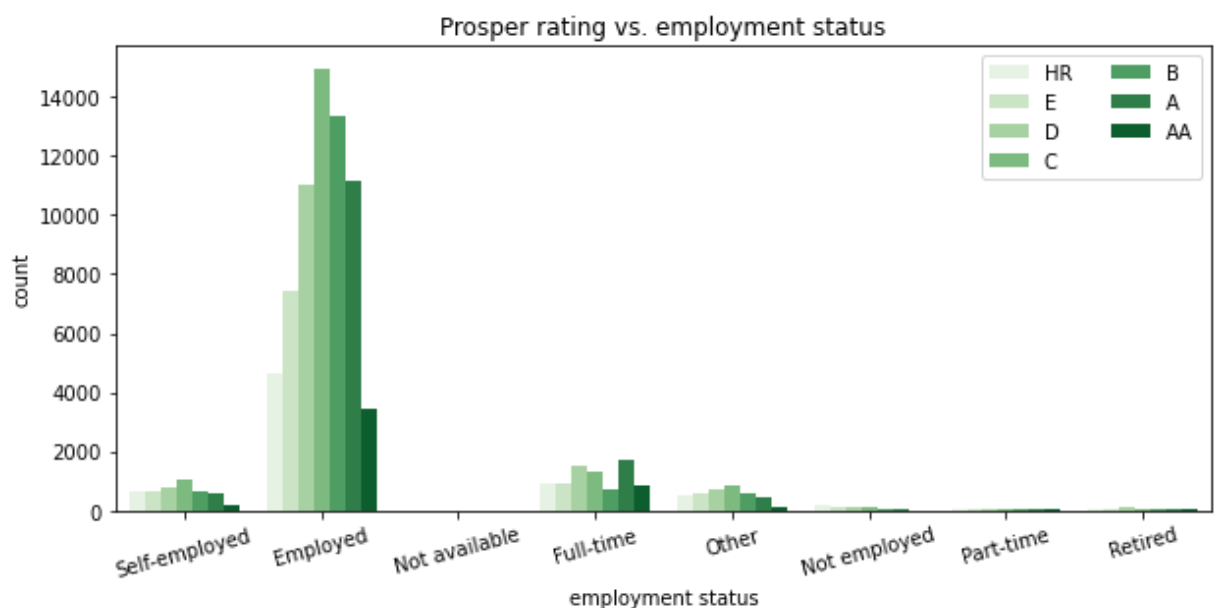


```
In [27]: # employment status vs. term
plt.figure(figsize = [10, 20])

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



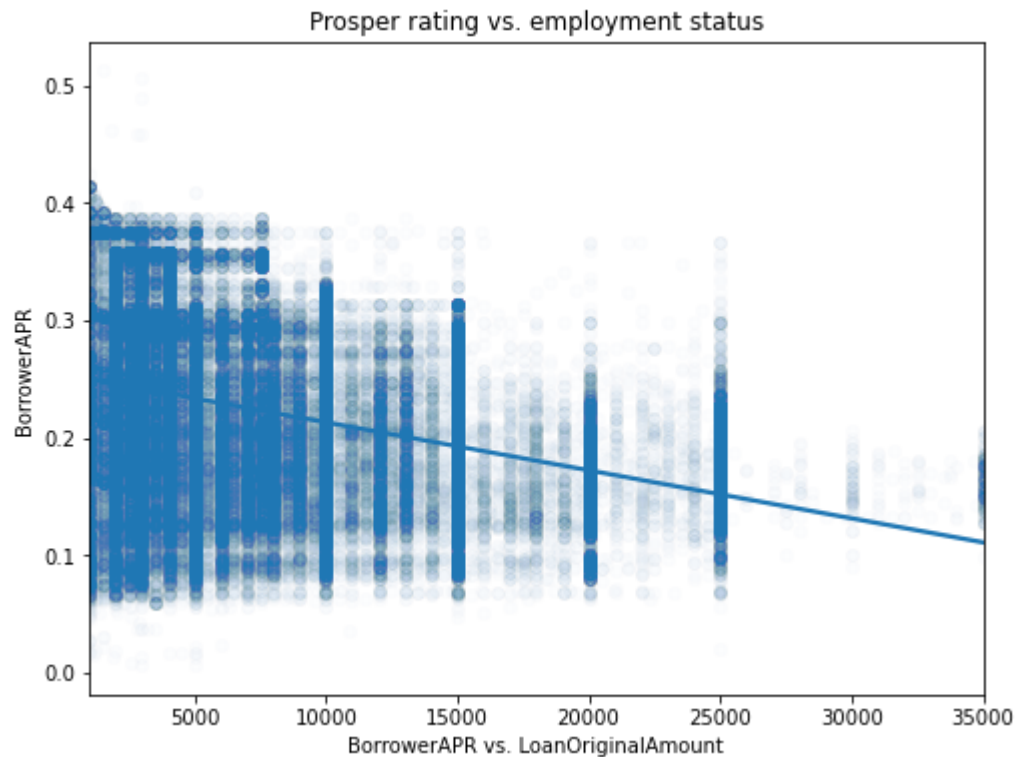
```
In [28]: # Prosper rating vs. employment status
plt.figure(figsize = [10, 20])
ax = plt.subplot(4, 1, 3)
sb.countplot(data = df_loans, x = 'EmploymentStatus', hue = 'ProsperRating (Alpha', palette = 'magma')
ax.legend(loc = 1, ncol = 2)
plt.xticks(rotation = 15)
plt.xlabel('employment status')
plt.title('Prosper rating vs. employment status');
```



Lets see how borrower APR and loan original amount are related

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.

```
In [29]: 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');
```

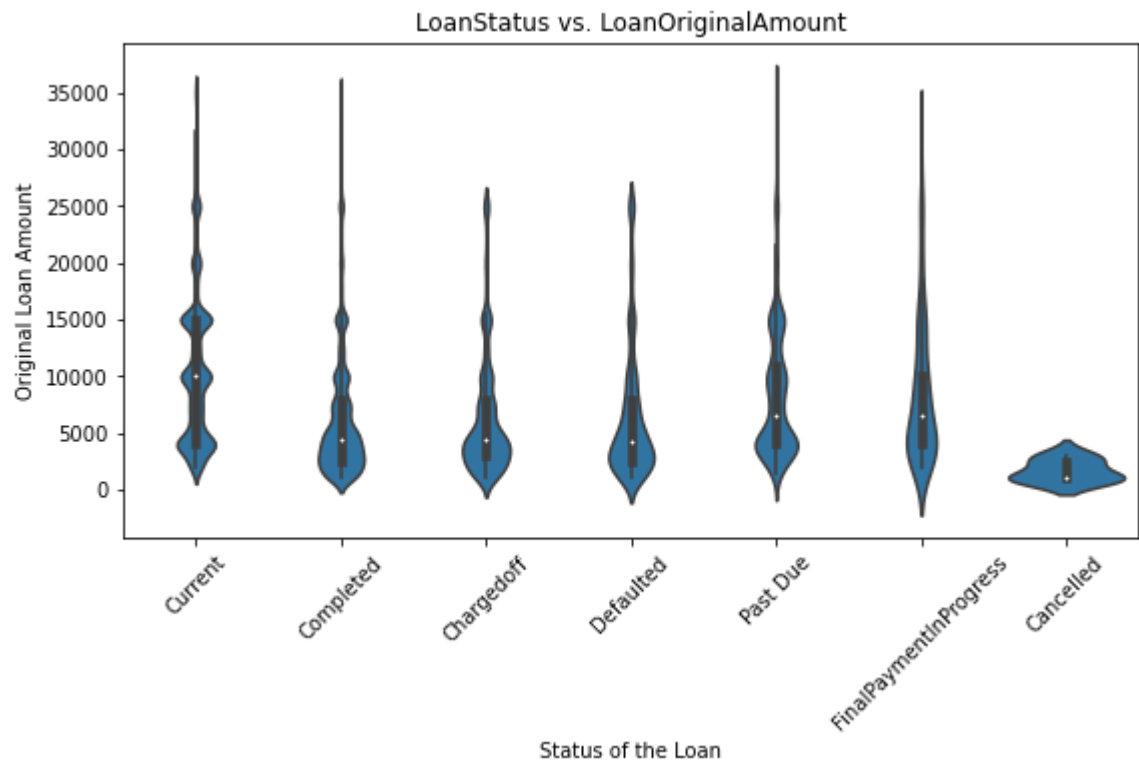


Now the relation between LoanStatus and LoanOriginalAmount

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.

```
In [30]: plt.figure(figsize = [20, 10])

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');
```

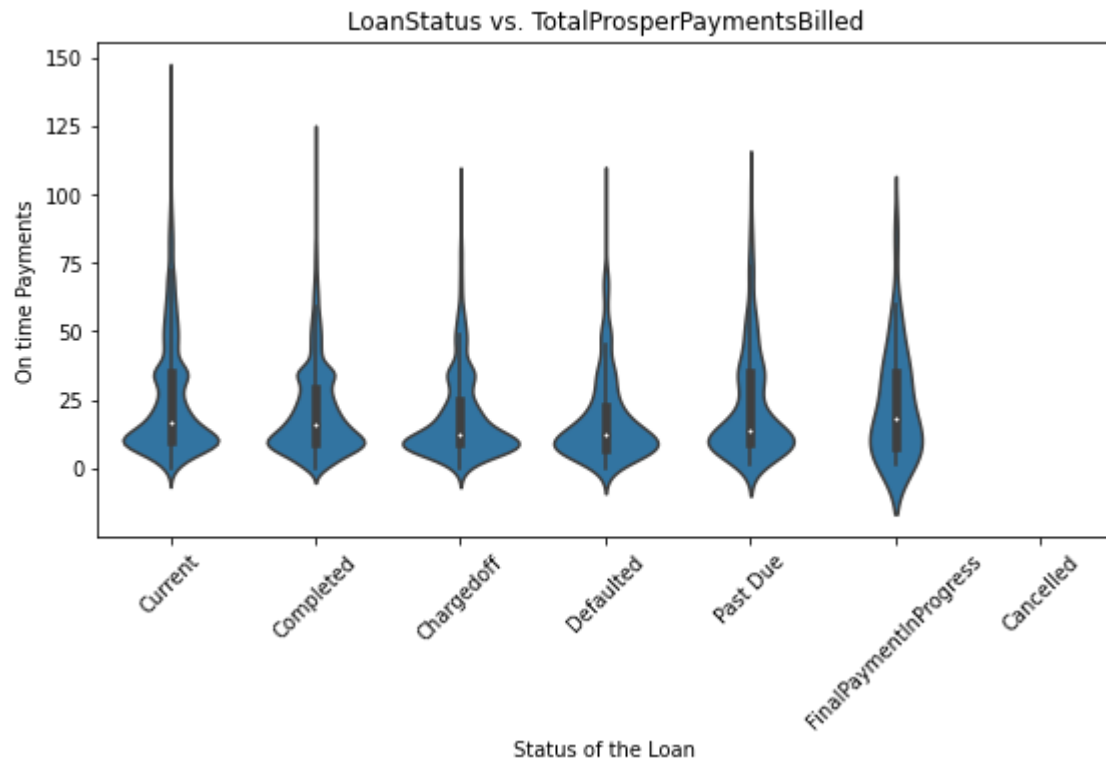


Now the relation between loan status and loan total prosper payments billed

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

```
In [31]: plt.figure(figsize = [20, 10])

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');
```

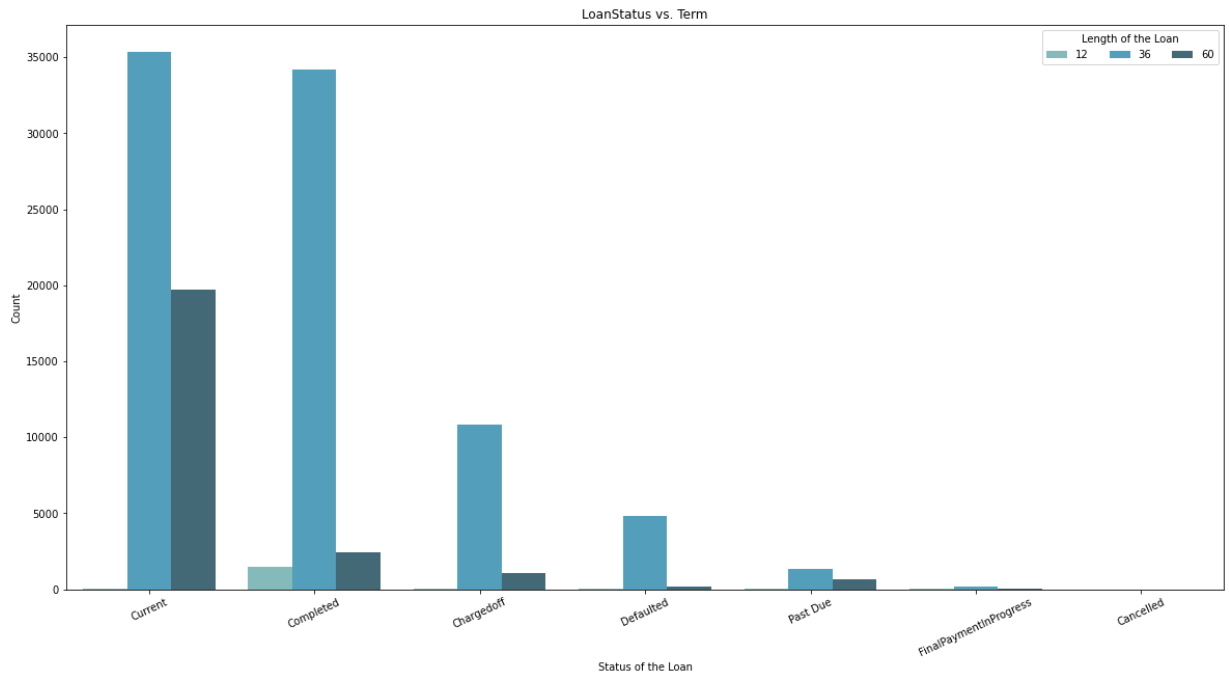


Relation between loan status and term

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

```
In [32]: 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');
```

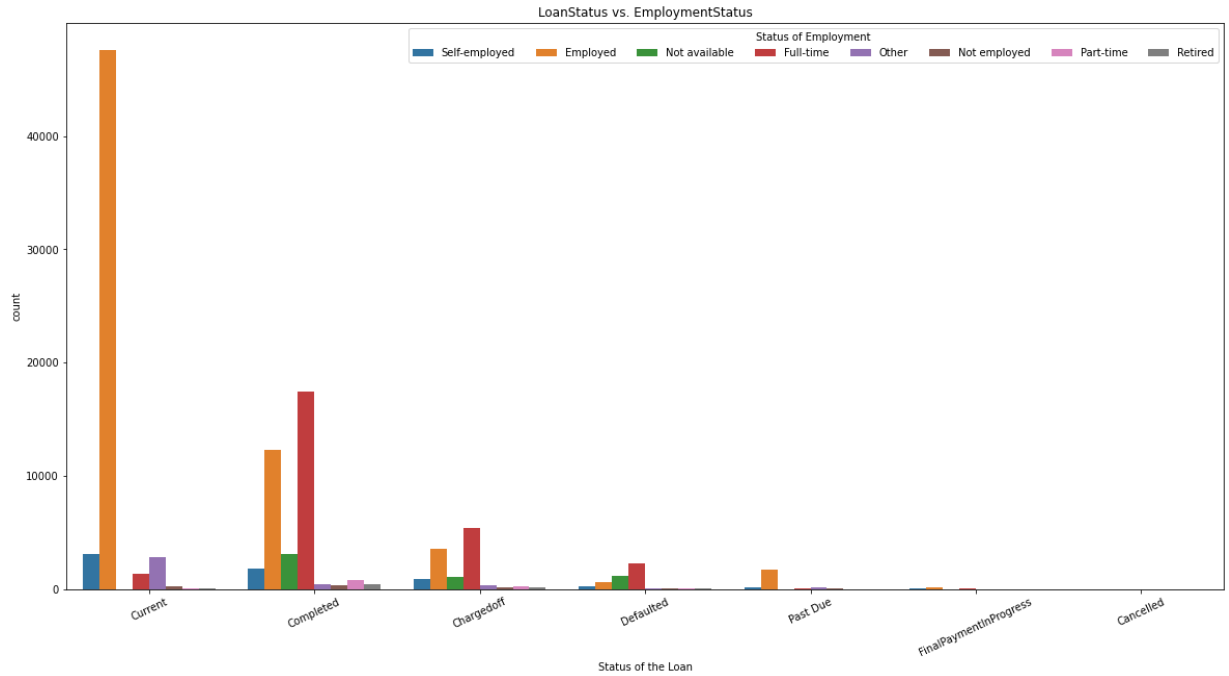


Relation between loan status and employment status

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.

```
In [33]: 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');
```

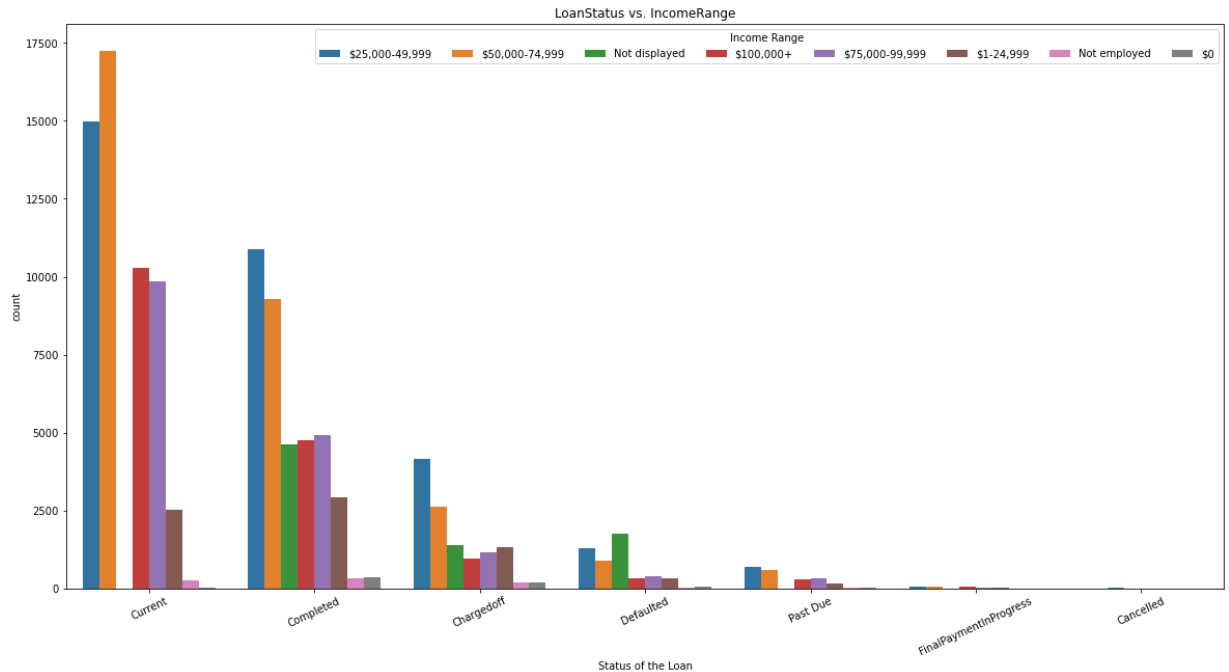


Relation between loan status and income range

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

```
In [34]: plt.figure(figsize = [20,10])

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



Relation between prosper score and loan status

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.

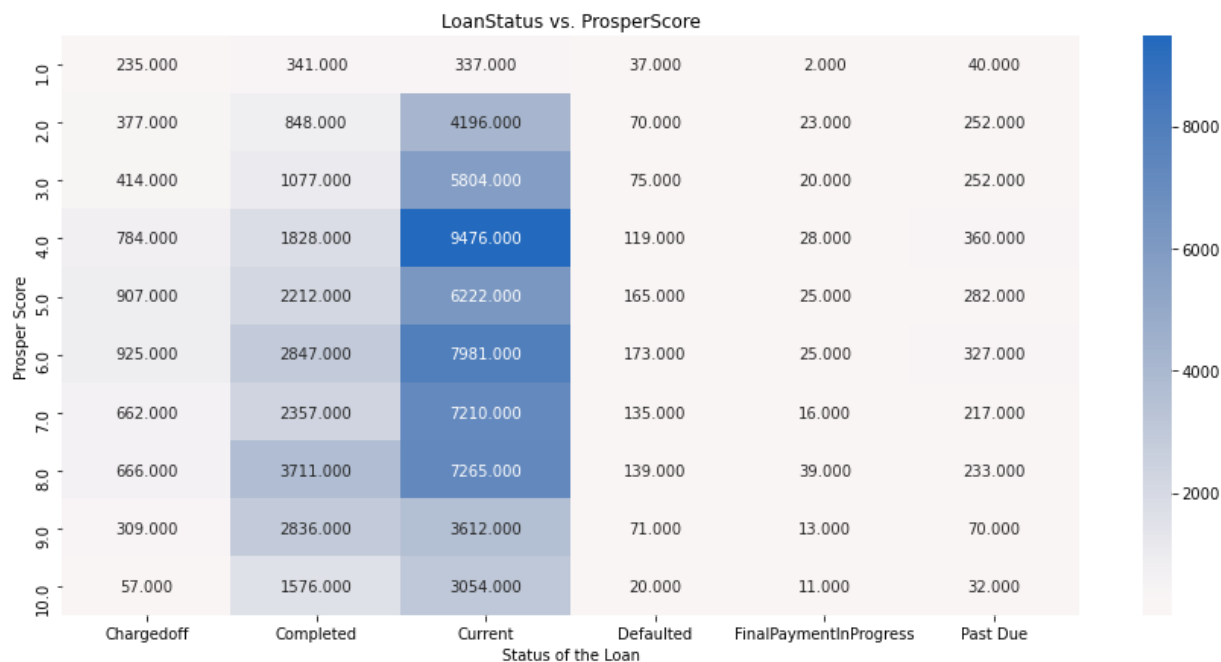
```
In [35]: # first I need to reshape the data
reshape = df_loans.groupby(['LoanStatus', 'ProsperScore']).size()
reshape = reshape.reset_index(name = 'count')
reshape = reshape.pivot(index = 'ProsperScore', columns = 'LoanStatus', values =
reshape
```

Out[35]:

	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 [36]: 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');
```



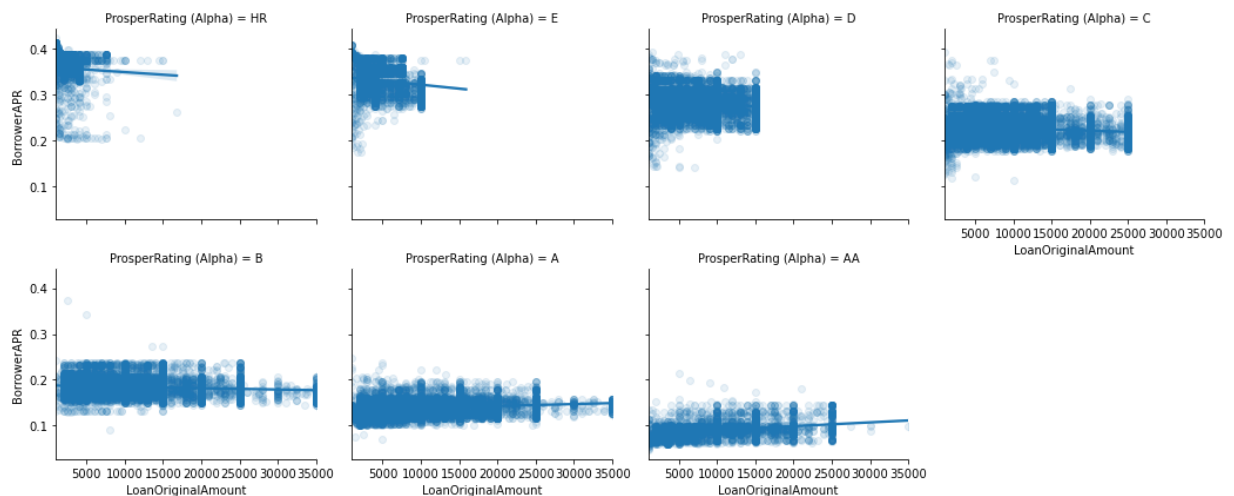
Multivariate Exploration

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

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.

```
In [37]: 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={
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)

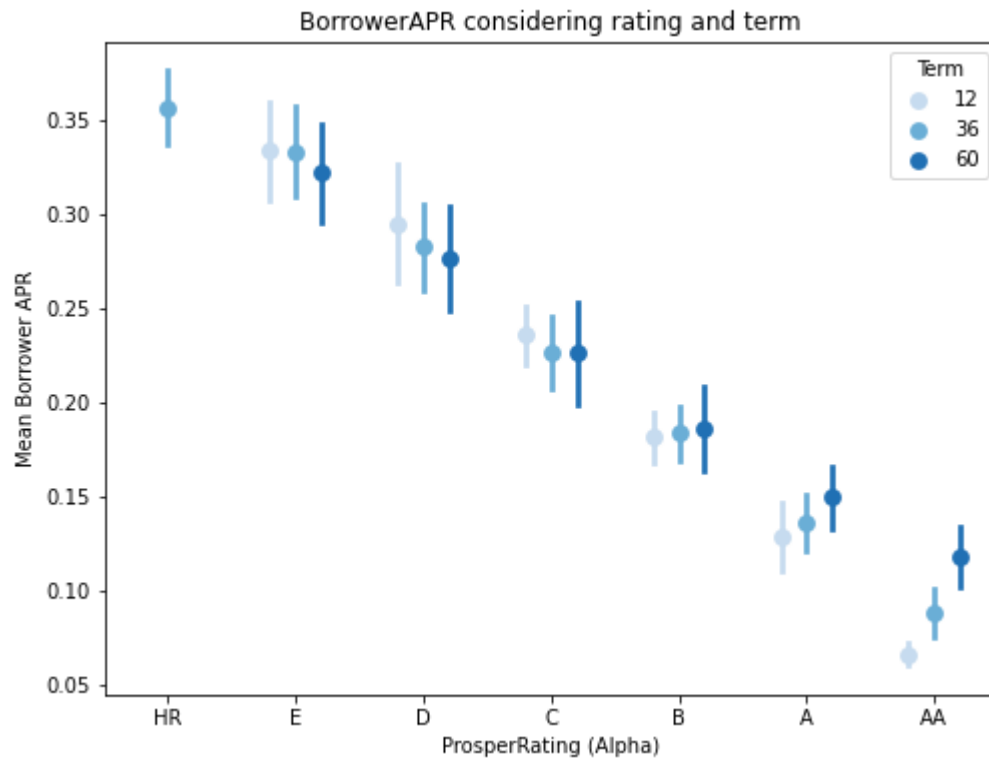


BorrowerAPR considering rating and term

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.


```
In [38]: 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);
```

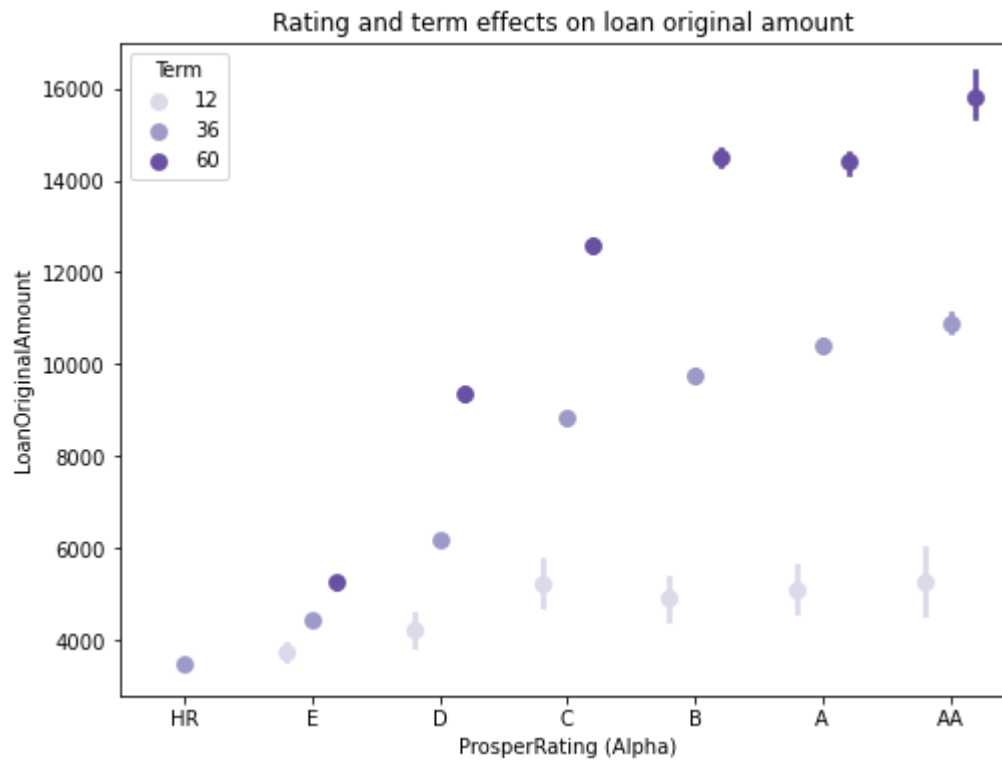


The rating and term effects on loan original amount

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

```
In [39]: fig, ax = plt.subplots(figsize=[8,6])

sb.pointplot(data = df_loans, x = 'ProsperRating (Alpha)', y = 'LoanOriginalAmount',
plt.title('Rating and term effects on loan original amount '));
```

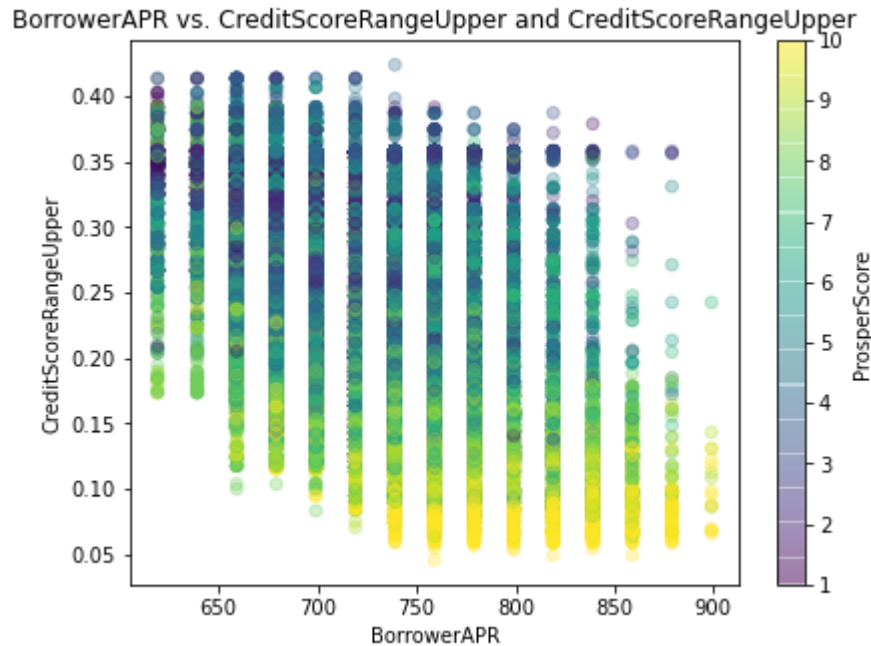


BorrowerAPR considering CreditScoreRangeUpper and ProsperScore

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

```
In [40]: 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');
```



Conclusion

After the demonstration, we answer the previously answered questions as follows:

- Prosper seem to give loans to all level of borrowers, but more is given to borrowers with a score of (C to A).
- We can see that the higher the borrowers rating the lower the annual rate of interest is charged (APR).

- APR is affected directly with the following attributes:

- Loan amount: when the amount increases the APR decrease.
- Term: as the most popular term is 36, it is also the most completed term which means it gets a higher score rating so as well the APR of the borrower decreases.
- Income: borrowers with an income range of (25000 - 74999) get more loans and also complete their loans on time, therefore gets a prosper rating of 8, so as well the APR is low for them.

- The APR also is low for borrowers who get large loans, but also make more than 100000+ as well as complete their loans on time.
- The most popular loans amount are between 2500 - 10000 as well as the term of 36 months, which is reasonable because small amounts are paid in a short time and preferable between borrowers.
- Borrowers who are employed tend to get more loans and pay on time and complete their loans therefore get higher scores.

Few insights on how Prosper could increase their profits from loans:

- Notice that most borrowers' income is between (25000 - 74000), so I think that prosper should focus their marketing plan on the middle-class segment.
- The most popular term of loans is the short ones, as the average loan amount is between 8200 dollars. I think that a marketing promotion plan should include activities the borrowers do like shopping for example to encourage more clients of that sort of loan.

In [41]: !jupyter nbconvert slide_deck_Loan_Data_from_Prosper.ipynb --to slides --post ser

```
[NbConvertApp] WARNING | Config option `kernel_spec_manager_class` not recogniz
ed by `NbConvertApp`.
[NbConvertApp] Converting notebook slide_deck_Loan_Data_from_Prosper.ipynb to s
lides
Traceback (most recent call last):
  File "C:\Users\Ahmed\anaconda3\Scripts\jupyter-nbconvert-script.py", line 10,
in <module>
    sys.exit(main())
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\jupyter_core\application.p
y", line 270, in launch_instance
    return super(JupyterApp, cls).launch_instance(argv=argv, **kwargs)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\traitlets\config\application
n.py", line 845, in launch_instance
    app.start()
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\nbconvertapp.py",
line 350, in start
    self.convert_notebooks()
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\nbconvertapp.py",
line 524, in convert_notebooks
    self.convert_single_notebook(notebook_filename)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\nbconvertapp.py",
line 489, in convert_single_notebook
    output, resources = self.export_single_notebook(notebook_filename, resource
s, input_buffer=input_buffer)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\nbconvertapp.py",
line 418, in export_single_notebook
    output, resources = self.exporter.from_filename(notebook_filename, resource
s=resources)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\exporters\exporte
r.py", line 181, in from_filename
    return self.from_file(f, resources=resources, **kw)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\exporters\exporte
r.py", line 199, in from_file
    return self.from_notebook_node(nbformat.read(file_stream, as_version=4), re
sources=resources, **kw)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\exporters\html.p
y", line 119, in from_notebook_node
    return super().from_notebook_node(nb, resources, **kw)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\exporters\template
exporter.py", line 384, in from_notebook_node
    output = self.template.render(nb=nb_copy, resources=resources)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\jinja2\environment.py", line
1090, in render
    self.environment.handle_exception()
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\jinja2\environment.py", line
832, in handle_exception
    reraise(*rewrite_traceback_stack(source=source))
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\jinja2_compat.py", line 28,
in reraise
    raise value.with_traceback(tb)
  File "C:\Users\Ahmed\Udacity visualization project\output_toggle.tpl", line
5, in top-level template code
    {% extends 'slides_reveal.tpl' %}
jinja2.exceptions.TemplateNotFound: slides_reveal.tpl
```

In []: