

# Part\_I\_exploration\_template

March 15, 2023

## 1 Part I - (Prosper Loan Dataset Exploration)

### 1.1 by (Sussan Omeruah)

### 1.2 Introduction

Prosper is a lending platform in the U.S. that offers a variety of resources people can use to try and improve their financial health, regardless of their financial situation. Users can consolidate debt, improve their home, or finance healthcare costs with personal loans among many other purposes. Investors who may be looking for new opportunities to diversify their portfolio can invest in personal loans.

This data set contains 113,937 loans with 81 variables on each loan, with duration between november 2005 to march 2014, including loan amount, borrower rate (or interest rate), current loan status, borrower income, and many others. See this data [dictionary](#) to understand the dataset's variables.

### 1.3 Preliminary Wrangling

```
In [16]: # import all packages and set plots to be embedded inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
%matplotlib inline
```

```
In [17]: # Upgrade seaborn
!pip install seaborn --upgrade
```

```
Requirement already up-to-date: seaborn in /opt/conda/lib/python3.6/site-packages (0.11.2)
Requirement already satisfied, skipping upgrade: matplotlib>=2.2 in /opt/conda/lib/python3.6/site-packages
Requirement already satisfied, skipping upgrade: scipy>=1.0 in /opt/conda/lib/python3.6/site-packages
Requirement already satisfied, skipping upgrade: numpy>=1.15 in /opt/conda/lib/python3.6/site-packages
Requirement already satisfied, skipping upgrade: pandas>=0.23 in /opt/conda/lib/python3.6/site-packages
Requirement already satisfied, skipping upgrade: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /opt/conda/lib/python3.6/site-packages
Requirement already satisfied, skipping upgrade: cyclo in /opt/conda/lib/python3.6/site-packages
Requirement already satisfied, skipping upgrade: kiwisolver>=1.0.1 in /opt/conda/lib/python3.6/site-packages
```

Requirement already satisfied, skipping upgrade: python-dateutil>=2.1 in /opt/conda/lib/python3.6/site-packages  
Requirement already satisfied, skipping upgrade: pillow>=6.2.0 in /opt/conda/lib/python3.6/site-packages  
Requirement already satisfied, skipping upgrade: pytz>=2011k in /opt/conda/lib/python3.6/site-packages  
Requirement already satisfied, skipping upgrade: six in /opt/conda/lib/python3.6/site-packages

Load in your dataset and describe its properties through the questions below. Try and motivate your exploration goals through this section.

```
In [82]: # import dataset
        loans = pd.read_csv('prosperLoanData.csv')
```

### 1.3.1 View/explore dataset

```
In [19]: #view data
        loans.head()
```

```
Out[19]:
```

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

	CreditGrade	Term	LoanStatus	ClosedDate	BorrowerAPR	\
0	C	36	Completed	2009-08-14 00:00:00	0.16516	
1	NaN	36	Current	NaN	0.12016	
2	HR	36	Completed	2009-12-17 00:00:00	0.28269	
3	NaN	36	Current	NaN	0.12528	
4	NaN	36	Current	NaN	0.24614	

	BorrowerRate	LenderYield	...	LP_ServiceFees	LP_CollectionFees	\
0	0.1580	0.1380	...	-133.18	0.0	
1	0.0920	0.0820	...	0.00	0.0	
2	0.2750	0.2400	...	-24.20	0.0	
3	0.0974	0.0874	...	-108.01	0.0	
4	0.2085	0.1985	...	-60.27	0.0	

	LP_GrossPrincipalLoss	LP_NetPrincipalLoss	LP_NonPrincipalRecoverypayments	\
0	0.0	0.0	0.0	
1	0.0	0.0	0.0	
2	0.0	0.0	0.0	
3	0.0	0.0	0.0	
4	0.0	0.0	0.0	

	PercentFunded	Recommendations	InvestmentFromFriendsCount	\
0	1.0	0	0	
1	1.0	0	0	
2	1.0	0	0	

3	1.0	0	0
4	1.0	0	0

	InvestmentFromFriendsAmount	Investors
0	0.0	258
1	0.0	1
2	0.0	41
3	0.0	158
4	0.0	20

[5 rows x 81 columns]

```
In [20]: # View columns and data types
        loans.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 81 columns):
ListingKey                113937 non-null object
ListingNumber             113937 non-null int64
ListingCreationDate       113937 non-null object
CreditGrade              28953 non-null object
Term                     113937 non-null int64
LoanStatus                113937 non-null object
ClosedDate                55089 non-null object
BorrowerAPR               113912 non-null float64
BorrowerRate              113937 non-null float64
LenderYield               113937 non-null float64
EstimatedEffectiveYield   84853 non-null float64
EstimatedLoss              84853 non-null float64
EstimatedReturn           84853 non-null float64
ProsperRating (numeric)   84853 non-null float64
ProsperRating (Alpha)     84853 non-null object
ProsperScore              84853 non-null float64
ListingCategory (numeric) 113937 non-null int64
BorrowerState             108422 non-null object
Occupation                110349 non-null object
EmploymentStatus          111682 non-null object
EmploymentStatusDuration  106312 non-null float64
IsBorrowerHomeowner       113937 non-null bool
CurrentlyInGroup           113937 non-null bool
GroupKey                  13341 non-null object
DateCreditPulled          113937 non-null object
CreditScoreRangeLower     113346 non-null float64
CreditScoreRangeUpper     113346 non-null float64
FirstRecordedCreditLine   113240 non-null object
CurrentCreditLines         106333 non-null float64
OpenCreditLines           106333 non-null float64
```

TotalCreditLinespast7years	113240	non-null	float64
OpenRevolvingAccounts	113937	non-null	int64
OpenRevolvingMonthlyPayment	113937	non-null	float64
InquiriesLast6Months	113240	non-null	float64
TotalInquiries	112778	non-null	float64
CurrentDelinquencies	113240	non-null	float64
AmountDelinquent	106315	non-null	float64
DelinquenciesLast7Years	112947	non-null	float64
PublicRecordsLast10Years	113240	non-null	float64
PublicRecordsLast12Months	106333	non-null	float64
RevolvingCreditBalance	106333	non-null	float64
BankcardUtilization	106333	non-null	float64
AvailableBankcardCredit	106393	non-null	float64
TotalTrades	106393	non-null	float64
TradesNeverDelinquent (percentage)	106393	non-null	float64
TradesOpenedLast6Months	106393	non-null	float64
DebtToIncomeRatio	105383	non-null	float64
IncomeRange	113937	non-null	object
IncomeVerifiable	113937	non-null	bool
StatedMonthlyIncome	113937	non-null	float64
LoanKey	113937	non-null	object
TotalProsperLoans	22085	non-null	float64
TotalProsperPaymentsBilled	22085	non-null	float64
OnTimeProsperPayments	22085	non-null	float64
ProsperPaymentsLessThanOneMonthLate	22085	non-null	float64
ProsperPaymentsOneMonthPlusLate	22085	non-null	float64
ProsperPrincipalBorrowed	22085	non-null	float64
ProsperPrincipalOutstanding	22085	non-null	float64
ScorexChangeAtTimeOfListing	18928	non-null	float64
LoanCurrentDaysDelinquent	113937	non-null	int64
LoanFirstDefaultedCycleNumber	16952	non-null	float64
LoanMonthsSinceOrigination	113937	non-null	int64
LoanNumber	113937	non-null	int64
LoanOriginalAmount	113937	non-null	int64
LoanOriginationDate	113937	non-null	object
LoanOriginationQuarter	113937	non-null	object
MemberKey	113937	non-null	object
MonthlyLoanPayment	113937	non-null	float64
LP_CustomerPayments	113937	non-null	float64
LP_CustomerPrincipalPayments	113937	non-null	float64
LP_InterestandFees	113937	non-null	float64
LP_ServiceFees	113937	non-null	float64
LP_CollectionFees	113937	non-null	float64
LP_GrossPrincipalLoss	113937	non-null	float64
LP_NetPrincipalLoss	113937	non-null	float64
LP_NonPrincipalRecoverypayments	113937	non-null	float64
PercentFunded	113937	non-null	float64
Recommendations	113937	non-null	int64

```
InvestmentFromFriendsCount      113937 non-null int64
InvestmentFromFriendsAmount     113937 non-null float64
Investors                       113937 non-null int64
dtypes: bool(3), float64(50), int64(11), object(17)
memory usage: 68.1+ MB
```

```
In [21]: # View data summary statistics
         loans.describe()
```

```
Out[21]:
```

	ListingNumber	Term	BorrowerAPR	BorrowerRate \
count	1.139370e+05	113937.000000	113912.000000	113937.000000
mean	6.278857e+05	40.830248	0.218828	0.192764
std	3.280762e+05	10.436212	0.080364	0.074818
min	4.000000e+00	12.000000	0.006530	0.000000
25%	4.009190e+05	36.000000	0.156290	0.134000
50%	6.005540e+05	36.000000	0.209760	0.184000
75%	8.926340e+05	36.000000	0.283810	0.250000
max	1.255725e+06	60.000000	0.512290	0.497500

	LenderYield	EstimatedEffectiveYield	EstimatedLoss	EstimatedReturn \
count	113937.000000	84853.000000	84853.000000	84853.000000
mean	0.182701	0.168661	0.080306	0.096068
std	0.074516	0.068467	0.046764	0.030403
min	-0.010000	-0.182700	0.004900	-0.182700
25%	0.124200	0.115670	0.042400	0.074080
50%	0.173000	0.161500	0.072400	0.091700
75%	0.240000	0.224300	0.112000	0.116600
max	0.492500	0.319900	0.366000	0.283700

	ProsperRating (numeric)	ProsperScore	...	LP_ServiceFees \
count	84853.000000	84853.000000	...	113937.000000
mean	4.072243	5.950067	...	-54.725641
std	1.673227	2.376501	...	60.675425
min	1.000000	1.000000	...	-664.870000
25%	3.000000	4.000000	...	-73.180000
50%	4.000000	6.000000	...	-34.440000
75%	5.000000	8.000000	...	-13.920000
max	7.000000	11.000000	...	32.060000

	LP_CollectionFees	LP_GrossPrincipalLoss	LP_NetPrincipalLoss \
count	113937.000000	113937.000000	113937.000000
mean	-14.242698	700.446342	681.420499
std	109.232758	2388.513831	2357.167068
min	-9274.750000	-94.200000	-954.550000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000

max	0.000000	25000.000000	25000.000000
-----	----------	--------------	--------------

	LP_NonPrincipalRecoverypayments	PercentFunded	Recommendations \
count	113937.000000	113937.000000	113937.000000
mean	25.142686	0.998584	0.048027
std	275.657937	0.017919	0.332353
min	0.000000	0.700000	0.000000
25%	0.000000	1.000000	0.000000
50%	0.000000	1.000000	0.000000
75%	0.000000	1.000000	0.000000
max	21117.900000	1.012500	39.000000

	InvestmentFromFriendsCount	InvestmentFromFriendsAmount	Investors
count	113937.000000	113937.000000	113937.000000
mean	0.023460	16.550751	80.475228
std	0.232412	294.545422	103.239020
min	0.000000	0.000000	1.000000
25%	0.000000	0.000000	2.000000
50%	0.000000	0.000000	44.000000
75%	0.000000	0.000000	115.000000
max	33.000000	25000.000000	1189.000000

[8 rows x 61 columns]

```
In [22]: print(loans.ListingCreationDate.min())
```

2005-11-09 20:44:28.847000000

```
In [23]: print(loans.ListingCreationDate.max())
```

2014-03-10 12:20:53.760000000

```
In [24]: # Checking for missing data
         loans.isna().sum()
```

```
Out[24]: ListingKey          0
         ListingNumber       0
         ListingCreationDate  0
         CreditGrade         84984
         Term                0
         LoanStatus          0
         ClosedDate          58848
         BorrowerAPR         25
         BorrowerRate        0
         LenderYield         0
         EstimatedEffectiveYield 29084
         EstimatedLoss       29084
```

EstimatedReturn	29084
ProsperRating (numeric)	29084
ProsperRating (Alpha)	29084
ProsperScore	29084
ListingCategory (numeric)	0
BorrowerState	5515
Occupation	3588
EmploymentStatus	2255
EmploymentStatusDuration	7625
IsBorrowerHomeowner	0
CurrentlyInGroup	0
GroupKey	100596
DateCreditPulled	0
CreditScoreRangeLower	591
CreditScoreRangeUpper	591
FirstRecordedCreditLine	697
CurrentCreditLines	7604
OpenCreditLines	7604
...	
TotalProsperLoans	91852
TotalProsperPaymentsBilled	91852
OnTimeProsperPayments	91852
ProsperPaymentsLessThanOneMonthLate	91852
ProsperPaymentsOneMonthPlusLate	91852
ProsperPrincipalBorrowed	91852
ProsperPrincipalOutstanding	91852
ScoreExchangeAtTimeOfListing	95009
LoanCurrentDaysDelinquent	0
LoanFirstDefaultedCycleNumber	96985
LoanMonthsSinceOrigination	0
LoanNumber	0
LoanOriginalAmount	0
LoanOriginationDate	0
LoanOriginationQuarter	0
MemberKey	0
MonthlyLoanPayment	0
LP_CustomerPayments	0
LP_CustomerPrincipalPayments	0
LP_InterestandFees	0
LP_ServiceFees	0
LP_CollectionFees	0
LP_GrossPrincipalLoss	0
LP_NetPrincipalLoss	0
LP_NonPrincipalRecoverypayments	0
PercentFunded	0
Recommendations	0
InvestmentFromFriendsCount	0
InvestmentFromFriendsAmount	0

```
Investors                                0
Length: 81, dtype: int64
```

```
In [25]: #copy dataset before wrangling
df = loans.copy()
```

```
In [26]: # Select necessary columns
```

```
columns=['ListingNumber', 'ListingCreationDate', 'Term', 'LoanStatus', 'BorrowerRate', 'BorrowerState',
          'EmploymentStatus', 'IncomeRange', 'MonthlyLoanPayment', 'Investors', 'CurrentDelinquencies', 'AmountDelinquent']
df=df[columns]
```

```
In [27]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 21 columns):
ListingNumber          113937 non-null int64
ListingCreationDate    113937 non-null object
Term                   113937 non-null int64
LoanStatus             113937 non-null object
BorrowerRate           113937 non-null float64
BorrowerState          108422 non-null object
IsBorrowerHomeowner    113937 non-null bool
LoanOriginationDate    113937 non-null object
LenderYield            113937 non-null float64
LoanOriginalAmount     113937 non-null int64
ListingCategory (numeric) 113937 non-null int64
StatedMonthlyIncome    113937 non-null float64
ProsperRating (Alpha)   84853 non-null object
ProsperRating (numeric) 84853 non-null float64
DebtToIncomeRatio      105383 non-null float64
EmploymentStatus       111682 non-null object
IncomeRange            113937 non-null object
MonthlyLoanPayment     113937 non-null float64
Investors              113937 non-null int64
CurrentDelinquencies    113240 non-null float64
AmountDelinquent       106315 non-null float64
dtypes: bool(1), float64(8), int64(5), object(7)
memory usage: 17.5+ MB
```

```
In [30]: # Check for missing data
df.isna().sum()
```

```
Out[30]: ListingNumber          0
          ListingCreationDate    0
          Term                   0
          LoanStatus             0
```



```

BorrowerRate          0
BorrowerState         5515
IsBorrowerHomeowner   0
LoanOriginationDate    0
LenderYield            0
LoanOriginalAmount     0
ListingCategory (numeric) 0
StatedMonthlyIncome    0
ProsperRating (Alpha)  29084
ProsperRating (numeric) 29084
DebtToIncomeRatio      0
EmploymentStatus       2255
IncomeRange            0
MonthlyLoanPayment     0
Investors              0
CurrentDelinquencies   697
AmountDelinquent       7622
dtype: int64

```

### 1.3.2 Handling Missing Data

```

In [31]: #fill the missing values of DebtToIncomeRatio with the mean
         df.DebtToIncomeRatio = df.DebtToIncomeRatio.fillna(df.DebtToIncomeRatio.mean())

```

```

In [32]: #drop rows of missing data in ProsperRating
         df = df[df['ProsperRating (Alpha)'].notnull()]

```

```

In [33]: # Test for missing data
         df.isna().sum()

```

```

Out[33]: ListingNumber          0
         ListingCreationDate     0
         Term                   0
         LoanStatus              0
         BorrowerRate            0
         BorrowerState           0
         IsBorrowerHomeowner     0
         LoanOriginationDate     0
         LenderYield             0
         LoanOriginalAmount       0
         ListingCategory (numeric) 0
         StatedMonthlyIncome      0
         ProsperRating (Alpha)    0
         ProsperRating (numeric)  0
         DebtToIncomeRatio        0
         EmploymentStatus         0
         IncomeRange              0
         MonthlyLoanPayment       0
         Investors                0

```

```

CurrentDelinquencies      0
AmountDelinquent          0
dtype: int64

```

```
In [34]: #Adjust Datetime columns
```

```

df['ListingCreationDate'] = pd.to_datetime(df['ListingCreationDate'])
df['LoanOriginationDate'] = pd.to_datetime(df['LoanOriginationDate'])
df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 84853 entries, 1 to 113936
Data columns (total 21 columns):
ListingNumber      84853 non-null int64
ListingCreationDate 84853 non-null datetime64[ns]
Term               84853 non-null int64
LoanStatus         84853 non-null object
BorrowerRate       84853 non-null float64
BorrowerState      84853 non-null object
IsBorrowerHomeowner 84853 non-null bool
LoanOriginationDate 84853 non-null datetime64[ns]
LenderYield        84853 non-null float64
LoanOriginalAmount 84853 non-null int64
ListingCategory (numeric) 84853 non-null int64
StatedMonthlyIncome 84853 non-null float64
ProsperRating (Alpha) 84853 non-null object
ProsperRating (numeric) 84853 non-null float64
DebtToIncomeRatio  84853 non-null float64
EmploymentStatus    84853 non-null object
IncomeRange         84853 non-null object
MonthlyLoanPayment  84853 non-null float64
Investors           84853 non-null int64
CurrentDelinquencies 84853 non-null float64
AmountDelinquent    84853 non-null float64
dtypes: bool(1), datetime64[ns](2), float64(8), int64(5), object(5)
memory usage: 13.7+ MB

```

```
In [35]: # Extract year and month from ListingCreationDate
```

```

df["ListingYear"] = df["ListingCreationDate"].dt.year
df["ListingMonth"] = df["ListingCreationDate"].dt.month

```

```
In [36]: # Convert ProsperRating & IncomeRange to ordinal categorical
```

```

var_dict = {'ProsperRating (Alpha)': ['HR', 'E', 'D', 'C', 'B', 'A', 'AA'],
            'IncomeRange': ['$1-24,999', '$25,000-49,999', '$50,000-74,999', '$75,000-99,999', '$100,000-149,999', '$150,000-199,999', '$200,000-249,999', '$250,000-299,999', '$300,000-349,999', '$350,000-399,999', '$400,000-449,999', '$450,000-499,999', '$500,000-549,999', '$550,000-599,999', '$600,000-649,999', '$650,000-699,999', '$700,000-749,999', '$750,000-799,999', '$800,000-849,999', '$850,000-899,999', '$900,000-949,999', '$950,000-999,999', '$1,000,000-1,499,999', '$1,500,000-1,999,999', '$2,000,000-2,499,999', '$2,500,000-2,999,999', '$3,000,000-3,499,999', '$3,500,000-3,999,999', '$4,000,000-4,499,999', '$4,500,000-4,999,999', '$5,000,000-5,499,999', '$5,500,000-5,999,999', '$6,000,000-6,499,999', '$6,500,000-6,999,999', '$7,000,000-7,499,999', '$7,500,000-7,999,999', '$8,000,000-8,499,999', '$8,500,000-8,999,999', '$9,000,000-9,499,999', '$9,500,000-9,999,999', '$10,000,000-10,499,999', '$10,500,000-10,999,999', '$11,000,000-11,499,999', '$11,500,000-11,999,999', '$12,000,000-12,499,999', '$12,500,000-12,999,999', '$13,000,000-13,499,999', '$13,500,000-13,999,999', '$14,000,000-14,499,999', '$14,500,000-14,999,999', '$15,000,000-15,499,999', '$15,500,000-15,999,999', '$16,000,000-16,499,999', '$16,500,000-16,999,999', '$17,000,000-17,499,999', '$17,500,000-17,999,999', '$18,000,000-18,499,999', '$18,500,000-18,999,999', '$19,000,000-19,499,999', '$19,500,000-19,999,999', '$20,000,000-20,499,999', '$20,500,000-20,999,999', '$21,000,000-21,499,999', '$21,500,000-21,999,999', '$22,000,000-22,499,999', '$22,500,000-22,999,999', '$23,000,000-23,499,999', '$23,500,000-23,999,999', '$24,000,000-24,499,999', '$24,500,000-24,999,999', '$25,000,000-25,499,999', '$25,500,000-25,999,999', '$26,000,000-26,499,999', '$26,500,000-26,999,999', '$27,000,000-27,499,999', '$27,500,000-27,999,999', '$28,000,000-28,499,999', '$28,500,000-28,999,999', '$29,000,000-29,499,999', '$29,500,000-29,999,999', '$30,000,000-30,499,999', '$30,500,000-30,999,999', '$31,000,000-31,499,999', '$31,500,000-31,999,999', '$32,000,000-32,499,999', '$32,500,000-32,999,999', '$33,000,000-33,499,999', '$33,500,000-33,999,999', '$34,000,000-34,499,999', '$34,500,000-34,999,999', '$35,000,000-35,499,999', '$35,500,000-35,999,999', '$36,000,000-36,499,999', '$36,500,000-36,999,999', '$37,000,000-37,499,999', '$37,500,000-37,999,999', '$38,000,000-38,499,999', '$38,500,000-38,999,999', '$39,000,000-39,499,999', '$39,500,000-39,999,999', '$40,000,000-40,499,999', '$40,500,000-40,999,999', '$41,000,000-41,499,999', '$41,500,000-41,999,999', '$42,000,000-42,499,999', '$42,500,000-42,999,999', '$43,000,000-43,499,999', '$43,500,000-43,999,999', '$44,000,000-44,499,999', '$44,500,000-44,999,999', '$45,000,000-45,499,999', '$45,500,000-45,999,999', '$46,000,000-46,499,999', '$46,500,000-46,999,999', '$47,000,000-47,499,999', '$47,500,000-47,999,999', '$48,000,000-48,499,999', '$48,500,000-48,999,999', '$49,000,000-49,499,999', '$49,500,000-49,999,999', '$50,000,000-50,499,999', '$50,500,000-50,999,999', '$51,000,000-51,499,999', '$51,500,000-51,999,999', '$52,000,000-52,499,999', '$52,500,000-52,999,999', '$53,000,000-53,499,999', '$53,500,000-53,999,999', '$54,000,000-54,499,999', '$54,500,000-54,999,999', '$55,000,000-55,499,999', '$55,500,000-55,999,999', '$56,000,000-56,499,999', '$56,500,000-56,999,999', '$57,000,000-57,499,999', '$57,500,000-57,999,999', '$58,000,000-58,499,999', '$58,500,000-58,999,999', '$59,000,000-59,499,999', '$59,500,000-59,999,999', '$60,000,000-60,499,999', '$60,500,000-60,999,999', '$61,000,000-61,499,999', '$61,500,000-61,999,999', '$62,000,000-62,499,999', '$62,500,000-62,999,999', '$63,000,000-63,499,999', '$63,500,000-63,999,999', '$64,000,000-64,499,999', '$64,500,000-64,999,999', '$65,000,000-65,499,999', '$65,500,000-65,999,999', '$66,000,000-66,499,999', '$66,500,000-66,999,999', '$67,000,000-67,499,999', '$67,500,000-67,999,999', '$68,000,000-68,499,999', '$68,500,000-68,999,999', '$69,000,000-69,499,999', '$69,500,000-69,999,999', '$70,000,000-70,499,999', '$70,500,000-70,999,999', '$71,000,000-71,499,999', '$71,500,000-71,999,999', '$72,000,000-72,499,999', '$72,500,000-72,999,999', '$73,000,000-73,499,999', '$73,500,000-73,999,999', '$74,000,000-74,499,999', '$74,500,000-74,999,999', '$75,000,000-75,499,999', '$75,500,000-75,999,999', '$76,000,000-76,499,999', '$76,500,000-76,999,999', '$77,000,000-77,499,999', '$77,500,000-77,999,999', '$78,000,000-78,499,999', '$78,500,000-78,999,999', '$79,000,000-79,499,999', '$79,500,000-79,999,999', '$80,000,000-80,499,999', '$80,500,000-80,999,999', '$81,000,000-81,499,999', '$81,500,000-81,999,999', '$82,000,000-82,499,999', '$82,500,000-82,999,999', '$83,000,000-83,499,999', '$83,500,000-83,999,999', '$84,000,000-84,499,999', '$84,500,000-84,999,999', '$85,000,000-85,499,999', '$85,500,000-85,999,999', '$86,000,000-86,499,999', '$86,500,000-86,999,999', '$87,000,000-87,499,999', '$87,500,000-87,999,999', '$88,000,000-88,499,999', '$88,500,000-88,999,999', '$89,000,000-89,499,999', '$89,500,000-89,999,999', '$90,000,000-90,499,999', '$90,500,000-90,999,999', '$91,000,000-91,499,999', '$91,500,000-91,999,999', '$92,000,000-92,499,999', '$92,500,000-92,999,999', '$93,000,000-93,499,999', '$93,500,000-93,999,999', '$94,000,000-94,499,999', '$94,500,000-94,999,999', '$95,000,000-95,499,999', '$95,500,000-95,999,999', '$96,000,000-96,499,999', '$96,500,000-96,999,999', '$97,000,000-97,499,999', '$97,500,000-97,999,999', '$98,000,000-98,499,999', '$98,500,000-98,999,999', '$99,000,000-99,499,999', '$99,500,000-99,999,999', '$100,000,000-100,499,999', '$100,500,000-100,999,999', '$101,000,000-101,499,999', '$101,500,000-101,999,999', '$102,000,000-102,499,999', '$102,500,000-102,999,999', '$103,000,000-103,499,999', '$103,500,000-103,999,999', '$104,000,000-104,499,999', '$104,500,000-104,999,999', '$105,000,000-105,499,999', '$105,500,000-105,999,999', '$106,000,000-106,499,999', '$106,500,000-106,999,999', '$107,000,000-107,499,999', '$107,500,000-107,999,999', '$108,000,000-108,499,999', '$108,500,000-108,999,999', '$109,000,000-109,499,999', '$109,500,000-109,999,999', '$110,000,000-110,499,999', '$110,500,000-110,999,999', '$111,000,000-111,499,999', '$111,500,000-111,999,999', '$112,000,000-112,499,999', '$112,500,000-112,999,999', '$113,000,000-113,499,999', '$113,500,000-113,999,999', '$114,000,000-114,499,999', '$114,500,000-114,999,999', '$115,000,000-115,499,999', '$115,500,000-115,999,999', '$116,000,000-116,499,999', '$116,500,000-116,999,999', '$117,000,000-117,499,999', '$117,500,000-117,999,999', '$118,000,000-118,499,999', '$118,500,000-118,999,999', '$119,000,000-119,499,999', '$119,500,000-119,999,999', '$120,000,000-120,499,999', '$120,500,000-120,999,999', '$121,000,000-121,499,999', '$121,500,000-121,999,999', '$122,000,000-122,499,999', '$122,500,000-122,999,999', '$123,000,000-123,499,999', '$123,500,000-123,999,999', '$124,000,000-124,499,999', '$124,500,000-124,999,999', '$125,000,000-125,499,999', '$125,500,000-125,999,999', '$126,000,000-126,499,999', '$126,500,000-126,999,999', '$127,000,000-127,499,999', '$127,500,000-127,999,999', '$128,000,000-128,499,999', '$128,500,000-128,999,999', '$129,000,000-129,499,999', '$129,500,000-129,999,999', '$130,000,000-130,499,999', '$130,500,000-130,999,999', '$131,000,000-131,499,999', '$131,500,000-131,999,999', '$132,000,000-132,499,999', '$132,500,000-132,999,999', '$133,000,000-133,499,999', '$133,500,000-133,999,999', '$134,000,000-134,499,999', '$134,500,000-134,999,999', '$135,000,000-135,499,999', '$135,500,000-135,999,999', '$136,000,000-136,499,999', '$136,500,000-136,999,999', '$137,000,000-137,499,999', '$137,500,000-137,999,999', '$138,000,000-138,499,999', '$138,500,000-138,999,999', '$139,000,000-139,499,999', '$139,500,000-139,999,999', '$140,000,000-140,499,999', '$140,500,000-140,999,999', '$141,000,000-141,499,999', '$141,500,000-141,999,999', '$142,000,000-142,499,999', '$142,500,000-142,999,999', '$143,000,000-143,499,999', '$143,500,000-143,999,999', '$144,000,000-144,499,999', '$144,500,000-144,999,999', '$145,000,000-145,499,999', '$145,500,000-145,999,999', '$146,000,000-146,499,999', '$146,500,000-146,999,999', '$147,000,000-147,499,999', '$147,500,000-147,999,999', '$148,000,000-148,499,999', '$148,500,000-148,999,999', '$149,000,000-149,499,999', '$149,500,000-149,999,999', '$150,000,000-150,499,999', '$150,500,000-150,999,999', '$151,000,000-151,499,999', '$151,500,000-151,999,999', '$152,000,000-152,499,999', '$152,500,000-152,999,999', '$153,000,000-153,499,999', '$153,500,000-153,999,999', '$154,000,000-154,499,999', '$154,500,000-154,999,999', '$155,000,000-155,499,999', '$155,500,000-155,999,999', '$156,000,000-156,499,999', '$156,500,000-156,999,999', '$157,000,000-157,499,999', '$157,500,000-157,999,999', '$158,000,000-158,499,999', '$158,500,000-158,999,999', '$159,000,000-159,499,999', '$159,500,000-159,999,999', '$160,000,000-160,499,999', '$160,500,000-160,999,999', '$161,000,000-161,499,999', '$161,500,000-161,999,999', '$162,000,000-162,499,999', '$162,500,000-162,999,999', '$163,000,000-163,499,999', '$163,500,000-163,999,999', '$164,000,000-164,499,999', '$164,500,000-164,999,999', '$165,000,000-165,499,999', '$165,500,000-165,999,999', '$166,000,000-166,499,999', '$166,500,000-166,999,999', '$167,000,000-167,499,999', '$167,500,000-167,999,999', '$168,000,000-168,499,999', '$168,500,000-168,999,999', '$169,000,000-169,499,999', '$169,500,000-169,999,999', '$170,000,000-170,499,999', '$170,500,000-170,999,999', '$171,000,000-171,499,999', '$171,500,000-171,999,999', '$172,000,000-172,499,999', '$172,500,000-172,999,999', '$173,000,000-173,499,999', '$173,500,000-173,999,999', '$174,000,000-174,499,999', '$174,500,000-174,999,999', '$175,000,000-175,499,999', '$175,500,000-175,999,999', '$176,000,000-176,499,999', '$176,500,000-176,999,999', '$177,000,000-177,499,999', '$177,500,000-177,999,999', '$178,000,000-178,499,999', '$178,500,000-178,999,999', '$179,000,000-179,499,999', '$179,500,000-179,999,999', '$180,000,000-180,499,999', '$180,500,000-180,999,999', '$181,000,000-181,499,999', '$181,500,000-181,999,999', '$182,000,000-182,499,999', '$182,500,000-182,999,999', '$183,000,000-183,499,999', '$183,500,000-183,999,999', '$184,000,000-184,499,999', '$184,500,000-184,999,999', '$185,000,000-185,499,999', '$185,500,000-185,999,999', '$186,000,000-186,499,999', '$186,500,000-186,999,999', '$187,000,000-187,499,999', '$187,500,000-187,999,999', '$188,000,000-188,499,999', '$188,500,000-188,999,999', '$189,000,000-189,499,999', '$189,500,000-189,999,999', '$190,000,000-190,499,999', '$190,500,000-190,999,999', '$191,000,000-191,499,999', '$191,500,000-191,999,999', '$192,000,000-192,499,999', '$192,500,000-192,999,999', '$193,000,000-193,499,999', '$193,500,000-193,999,999', '$194,000,000-194,499,999', '$194,500,000-194,999,999', '$195,000,000-195,499,999', '$195,500,000-195,999,999', '$196,000,000-196,499,999', '$196,500,000-196,999,999', '$197,000,000-197,499,999', '$197,500,000-197,999,999', '$198,000,000-198,499,999', '$198,500,000-198,999,999', '$199,000,000-199,499,999', '$199,500,000-199,999,999', '$200,000,000-200,499,999', '$200,500,000-200,999,999', '$201,000,000-201,499,999', '$201,500,000-201,999,999', '$202,000,000-202,499,999', '$202,500,000-202,999,999', '$203,000,000-203,499,999', '$203,500,000-203,999,999', '$204,000,000-204,499,999', '$204,500,000-204,999,999', '$205,000,000-205,499,999', '$205,500,000-205,999,999', '$206,000,000-206,499,999', '$206,500,000-206,999,999', '$207,000,000-207,499,999', '$207,500,000-207,999,999', '$208,000,000-208,499,999', '$208,500,000-208,999,999', '$209,000,000-209,499,999', '$209,500,000-209,999,999', '$210,000,000-210,499,999', '$210,500,000-210,999,999', '$211,000,000-211,499,999', '$211,500,000-211,999,999', '$212,000,000-212,499,999', '$212,500,000-212,999,999', '$213,000,000-213,499,999', '$213,500,000-213,999,999', '$214,000,000-214,499,999', '$214,500,000-214,999,999', '$215,000,000-215,499,999', '$215,500,000-215,999,999', '$216,000,000-216,499,999', '$216,500,000-216,999,999', '$217,000,000-217,499,999', '$217,500,000-217,999,999', '$218,000,000-218,499,999', '$218,500,000-218,999,999', '$219,000,000-219,499,999', '$219,500,000-219,999,999', '$220,000,000-220,499,999', '$220,500,000-220,999,999', '$221,000,000-221,499,999', '$221,500,000-221,999,999', '$222,000,000-222,499,999', '$222,500,000-222,999,999', '$223,000,000-223,499,999', '$223,500,000-223,999,999', '$224,000,000-224,499,999', '$224,500,000-224,999,999', '$225,000,000-225,499,999', '$225,500,000-225,999,999', '$226,000,000-226,499,999', '$226,500,000-226,999,999', '$227,000,000-227,499,999', '$227,500,000-227,999,999', '$228,000,000-228,499,999', '$228,500,000-228,999,999', '$229,000,000-229,499,999', '$229,500,000-229,999,999', '$230,000,000-230,499,999', '$230,500,000-230,999,999', '$231,000,000-231,499,999', '$231,500,000-231,999,999', '$232,000,000-232,49
```

```
In [79]: #Adjust datatype for all other categorical columns
df['BorrowerState'] = df['BorrowerState'].astype('category')
df['LoanStatus'] = df['LoanStatus'].astype('category')
df['ListingCategory (numeric)'] = df['ListingCategory (numeric)'].astype('category')

In [80]: # Check size of df
df.shape

Out[80]: (84853, 25)
```

### 1.3.3 What is the structure of your dataset?

The dataset contains columns with information on date, purpose, amount, rates and status of loans. The size is (84853, 28)

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

I am interested in exploring the factors that affect loan outcome

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

I am guessing that loan amount, borrower rate and loan status will help my investigation. I also think that some borrower characteristics like loan purpose(ListingCategory) and Loan Term will affect the outcome of a loan

## 1.4 Univariate Exploration

I will start from Loan Amount

### 1.4.1 Question

What is the distribution of loan amounts?

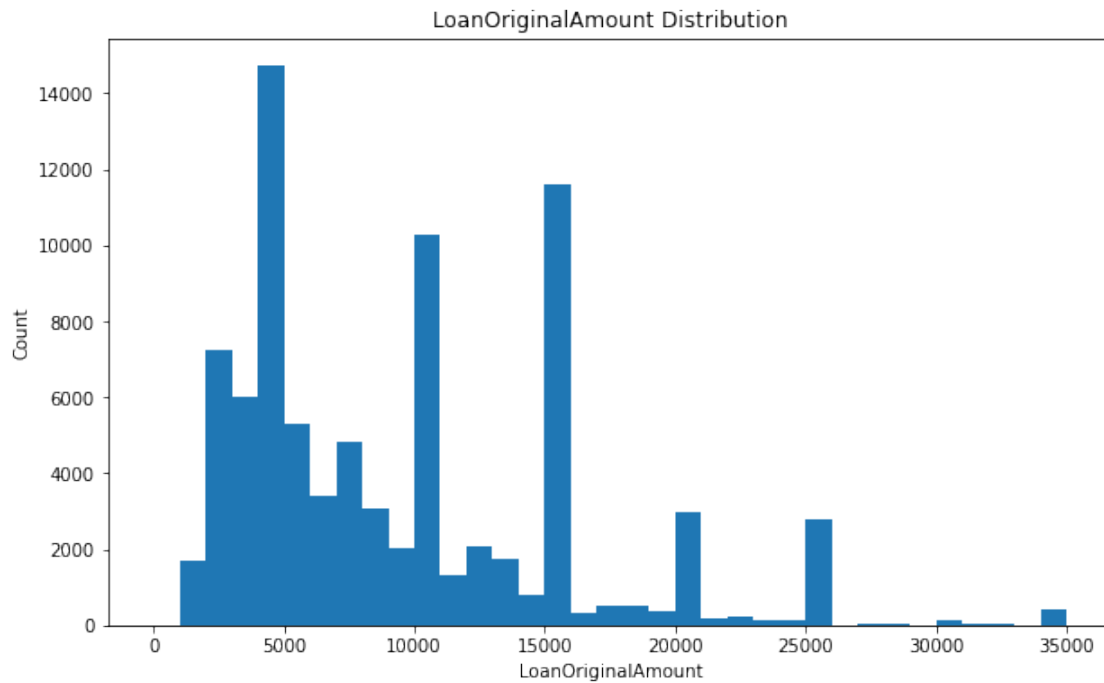
### 1.4.2 Visualization

```
In [39]: def plot_histogram(df, column, bin_size):
# Create bins using specified bin size
bins = np.arange(0, df[column].max()+bin_size, bin_size)

# Create a histogram of the specified column
base_color = sns.color_palette()[0]
fig, ax = plt.subplots(figsize=(10, 6))
ax.hist(df[column], bins=bins, color= base_color)

# Set the chart title and axes labels
ax.set_title(f'{column} Distribution')
ax.set_xlabel(column)
ax.set_ylabel('Count')
```

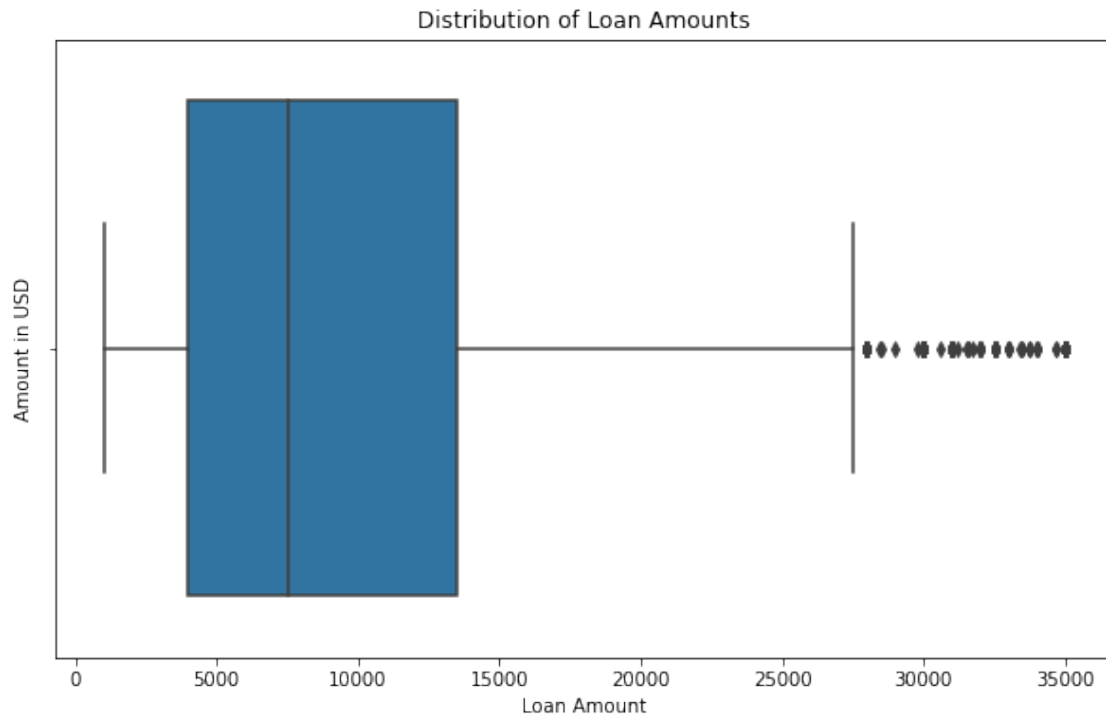
```
In [40]: plot_histogram(df, 'LoanOriginalAmount', 1000)
```



```
In [41]: # Create a boxplot to examine outliers
fig, ax = plt.subplots(figsize=(10, 6))
sns.boxplot(df["LoanOriginalAmount"])

# Add labels and title
plt.xlabel("Loan Amount")
plt.ylabel("Amount in USD")
plt.title("Distribution of Loan Amounts");
```

/opt/conda/lib/python3.6/site-packages/seaborn/\_decorators.py:43: FutureWarning: Pass the following arguments as keyword arguments: `ax=` and `fig=`. From version 0.12, passing `ax` and `fig` to the `plot` function is deprecated and may result in an error in the future.



### 1.4.3 Observation

The loan original Amount is multimodal with various peaks, with loan amount of about 5000 having the most peak

The histogram appears to be positively skewed, meaning that there are relatively more loans with lower amounts and fewer loans with higher amounts. The long tail towards higher loan amounts suggests that there are some loans with very high amounts that are considered outliers.

### 1.4.4 Observation

The median loan amount is around \$7500, which is represented by the line in the middle of the box.

The height of the box indicates that half of the loan amounts in the dataset fall between approximately \$4000 and 14000

From the boxplot, we can also see that the distribution of loan amounts in the dataset is positively skewed, with a long tail towards higher values, but the highest loan amounts in the dataset are much higher than the median loan amount.

### 1.4.5 Question

What are the common reasons for taking loans?

### 1.4.6 Visualization

```
In [42]: # Count the number of loans for each purpose
purpose_counts = loans['ListingCategory (numeric)'].value_counts().sort_index()

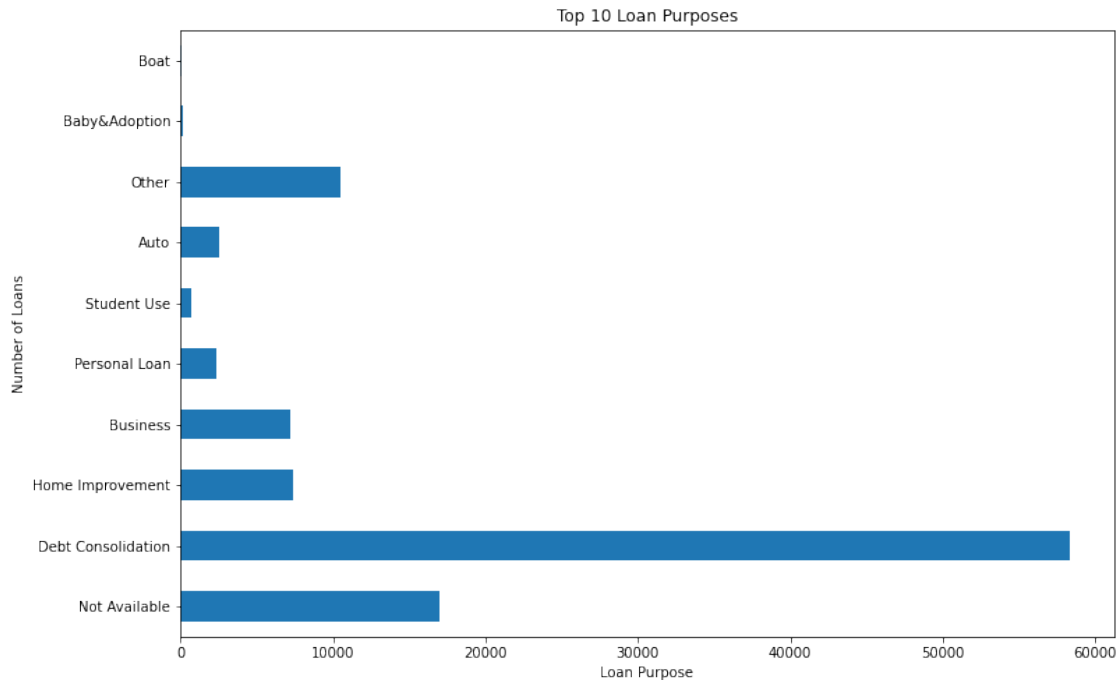
# Map the numeric categories to their corresponding string values

listing_category = {
    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'
}

purpose_counts.index = purpose_counts.index.map(listing_category)

# Create a bar chart of the loan purposes
base_color = sns.color_palette()[0]
fig, ax = plt.subplots(figsize=(12, 8))
purpose_counts.head(10).plot(kind='barh', color= base_color)

# Set the chart title and axes labels
ax.set_title('Top 10 Loan Purposes')
ax.set_xlabel('Loan Purpose')
ax.set_ylabel('Number of Loans');
```



### 1.4.7 Observation

The most common loan purpose is debt consolidation

### 1.4.8 Question

What is the common duration for loans?

### 1.4.9 Visualization

```
In [43]: # Set figsize
plt.figure(figsize=[6, 4])

#plot barchart
sns.countplot(x='Term', data=df, color = base_color)

# Add labels and title
plt.xlabel('Loan Term (Months)')
plt.ylabel('Number of Loans')
plt.title('Distribution of Loan Term');
```



#### 1.4.10 Observation

Most of the loan duration is 3years and 2years

#### 1.4.11 Question

What is the distribution of LoanStatus?

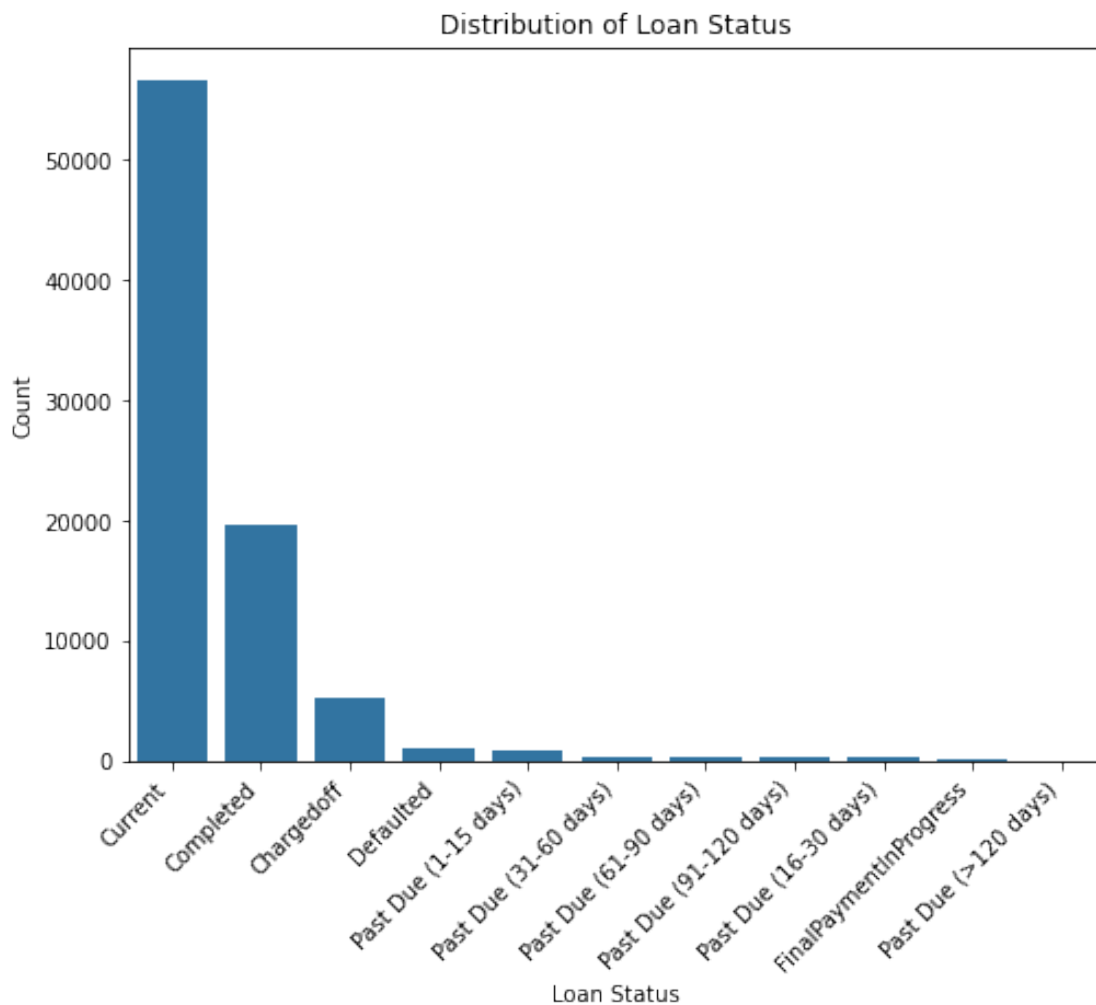
#### 1.4.12 Visualization

```
In [44]: df["LoanStatus"].value_counts()
```

```
Out[44]: Current          56576
Completed        19664
Chargedoff        5336
Defaulted         1005
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
Name: LoanStatus, dtype: int64
```



```
In [45]: # Create countplot
plt.figure(figsize=(8, 6))
sns.countplot(x='LoanStatus', data=df, order=df['LoanStatus'].value_counts().index, col
plt.xticks(rotation=45, ha='right')
plt.title('Distribution of Loan Status')
plt.xlabel('Loan Status')
plt.ylabel('Count');
```



#### 1.4.13 Observation

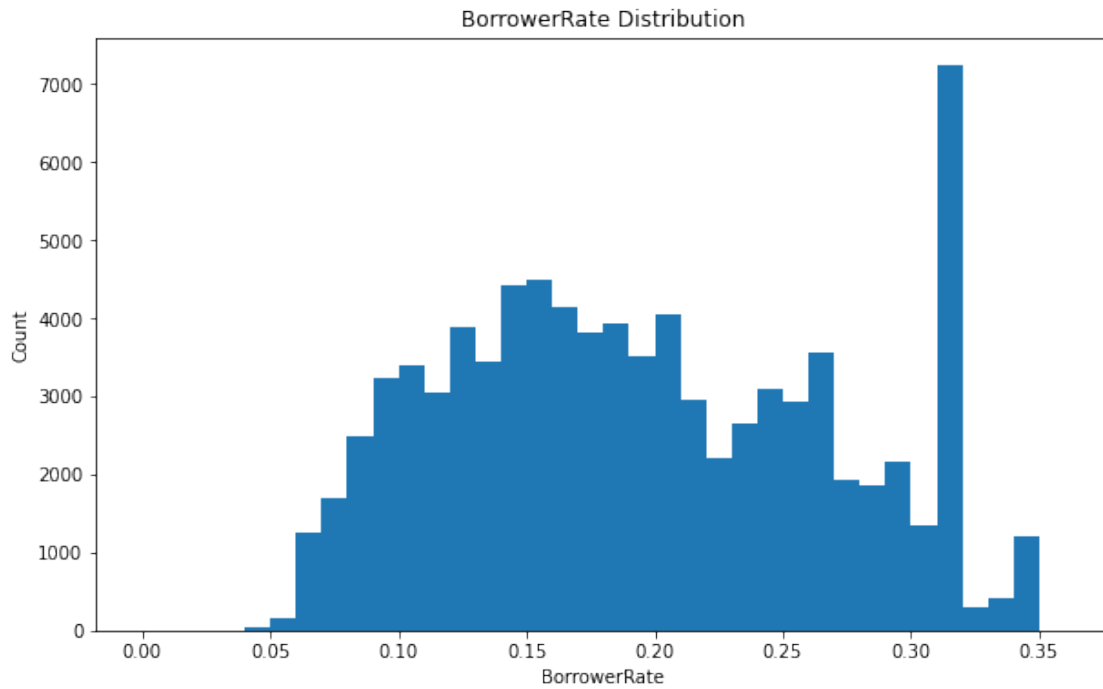
Majority of the loans fall under 'Current' status

#### 1.4.14 Question

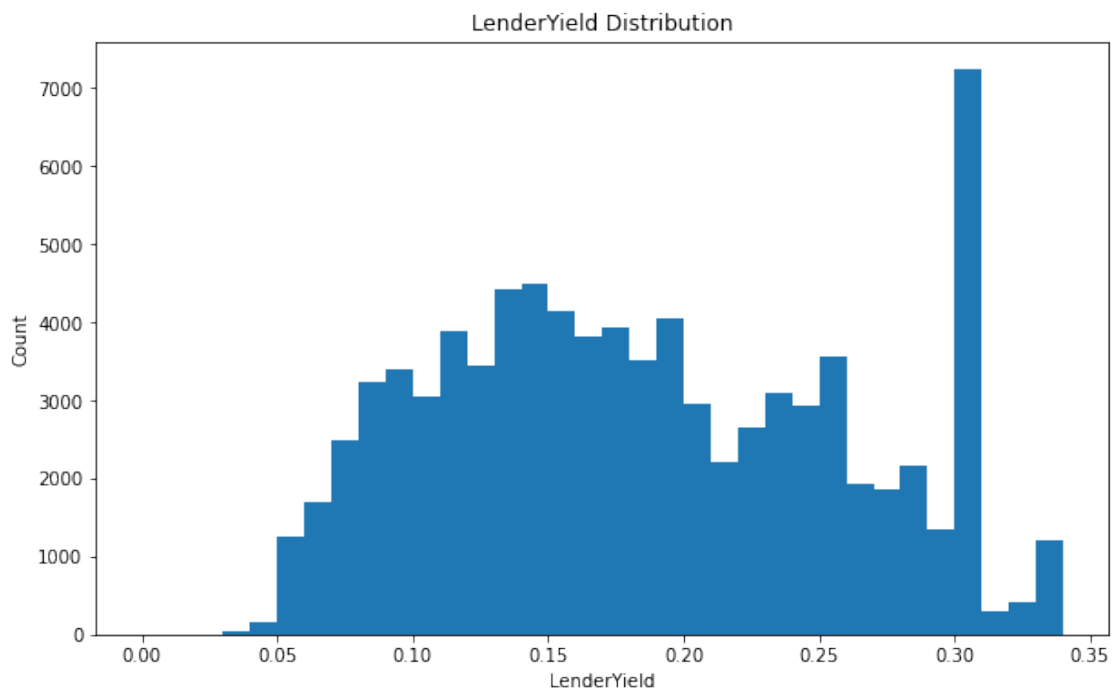
What is the distribution of BorrowerRate, LenderYield?

### 1.4.15 Visualization

```
In [46]: plot_histogram(df, 'BorrowerRate', 0.01)
```



```
In [47]: plot_histogram(df, 'LenderYield', 0.01)
```



### 1.4.16 Observation

Borrower Rate distribution is multimodal. The majority of loans have APRs between approximately 0.05 and 0.25. There are also some loans with APRs greater than 0.3 but they are relatively few compared to those in the lower APR ranges. We have a shape peak between 0.15 and 0.2 and outlier in 0.3

Lender Yield follows the same distribution as borrower rate

### 1.4.17 Question

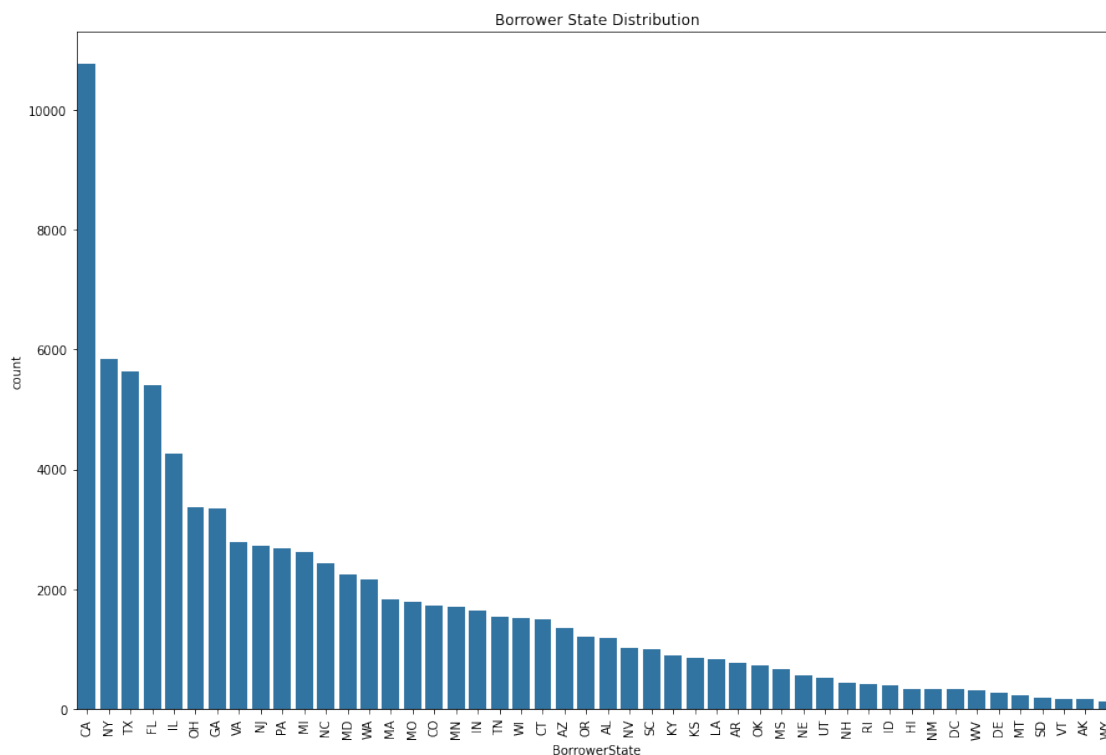
What are the distributions of borrower characteristics like State, employment status, income range, prosper rating, homeowner?

### 1.4.18 Visualization

In [48]: *#Borrower State Distribution*

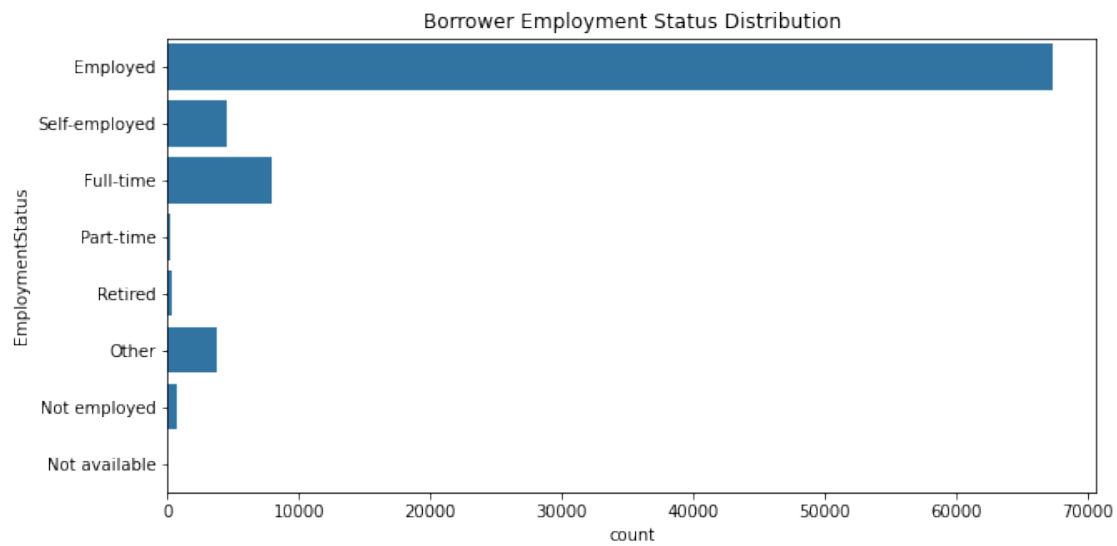
```
state_order = df['BorrowerState'].value_counts().index

base_color = sns.color_palette()[0]
plt.figure(figsize=[15, 10])
sns.countplot(data=df, x='BorrowerState', color=base_color, order=state_order);
plt.title('Borrower State Distribution');
plt.xticks(rotation=90);
```



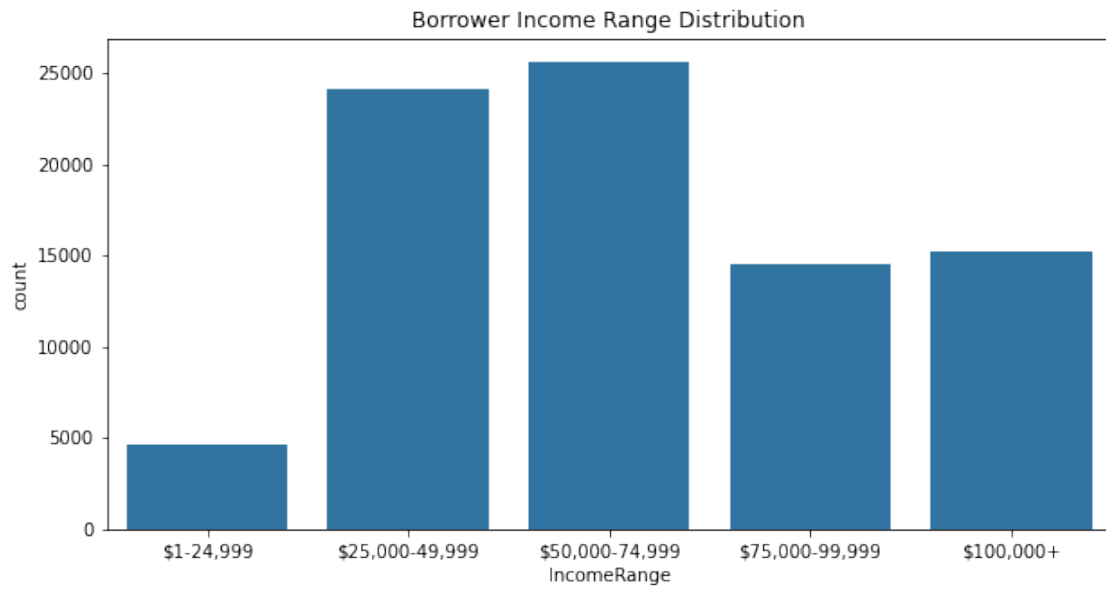
```
In [49]: # Borrower Employment Status Distribution
```

```
plt.figure(figsize=[10, 5])
sns.countplot(data = df, y = 'EmploymentStatus', color = base_color)
plt.title('Borrower Employment Status Distribution');
```



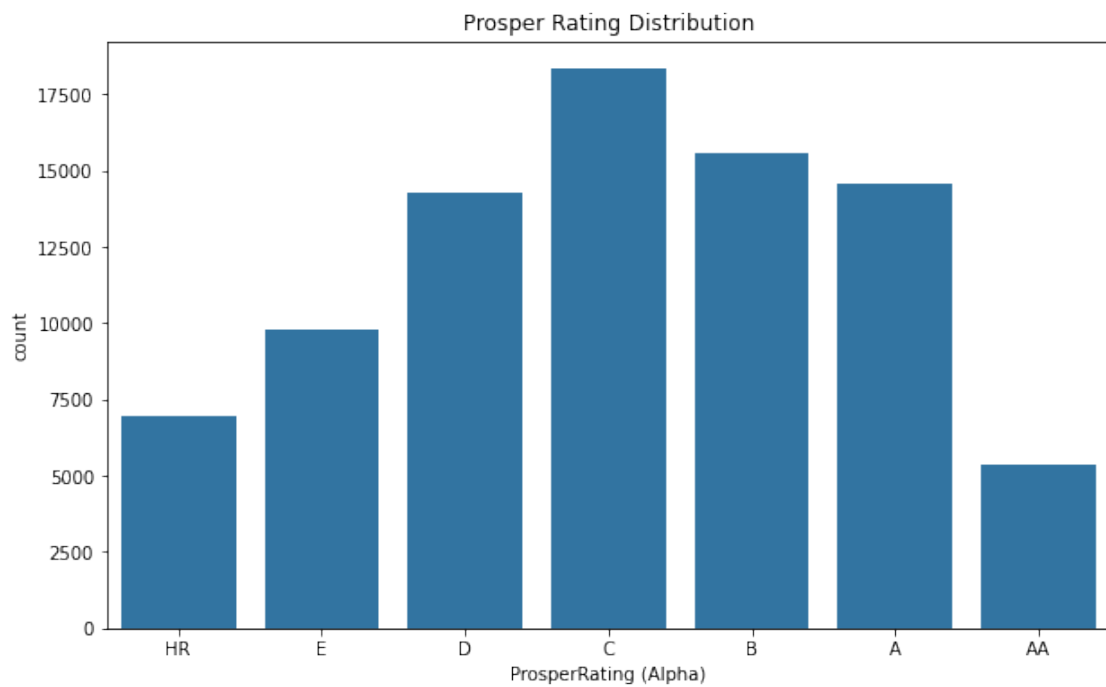
```
In [50]: #Borrower Income Range Distrobution
```

```
plt.figure(figsize=[10, 5])
sns.countplot(data=df,x='IncomeRange',color=base_color);
plt.title('Borrower Income Range Distribution');
```

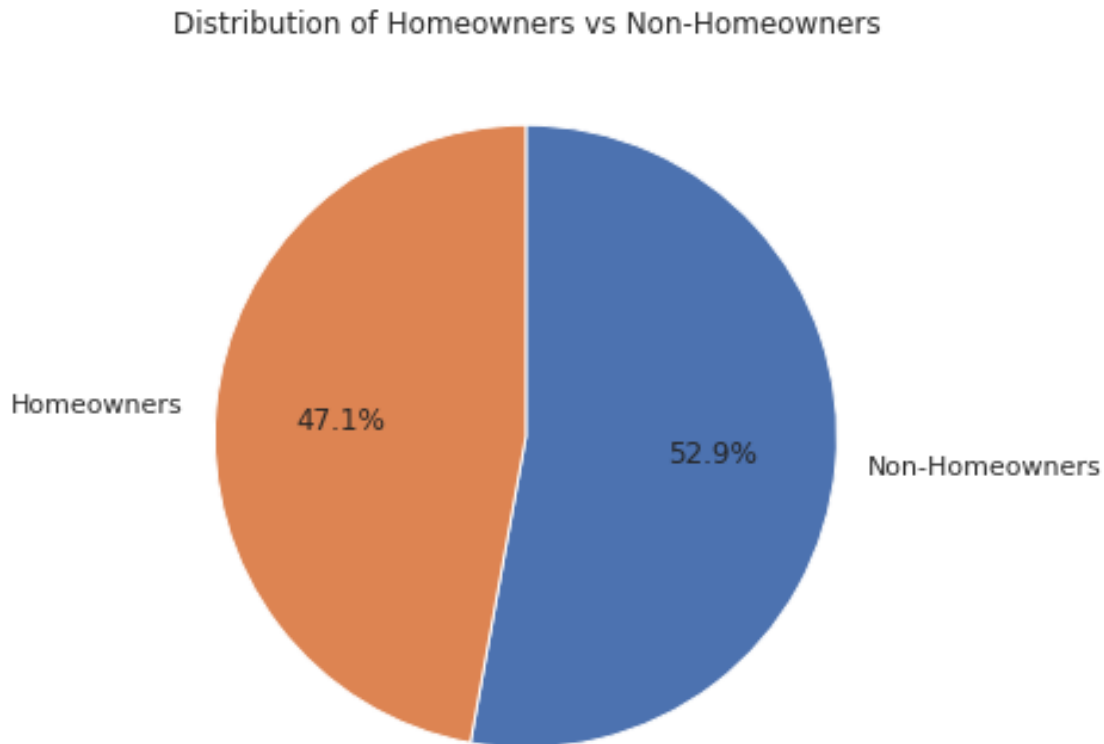


In [51]: # Prosper Rating Distribution

```
plt.figure(figsize=[10, 6]);  
sns.countplot(data=df, x='ProsperRating (Alpha)', color=base_color);  
plt.title('Prosper Rating Distribution');
```



```
In [85]: # Create a pie chart of IsBorrowerHomeowner
plt.figure(figsize=[10, 6])
plt.pie(df['IsBorrowerHomeowner'].value_counts(), labels=['Non-Homeowners', 'Homeowners'],
        autopct='%1.1f%%', startangle = 90,
        counterclock = False)
plt.title('Distribution of Homeowners vs Non-Homeowners');
```



#### 1.4.19 Observation

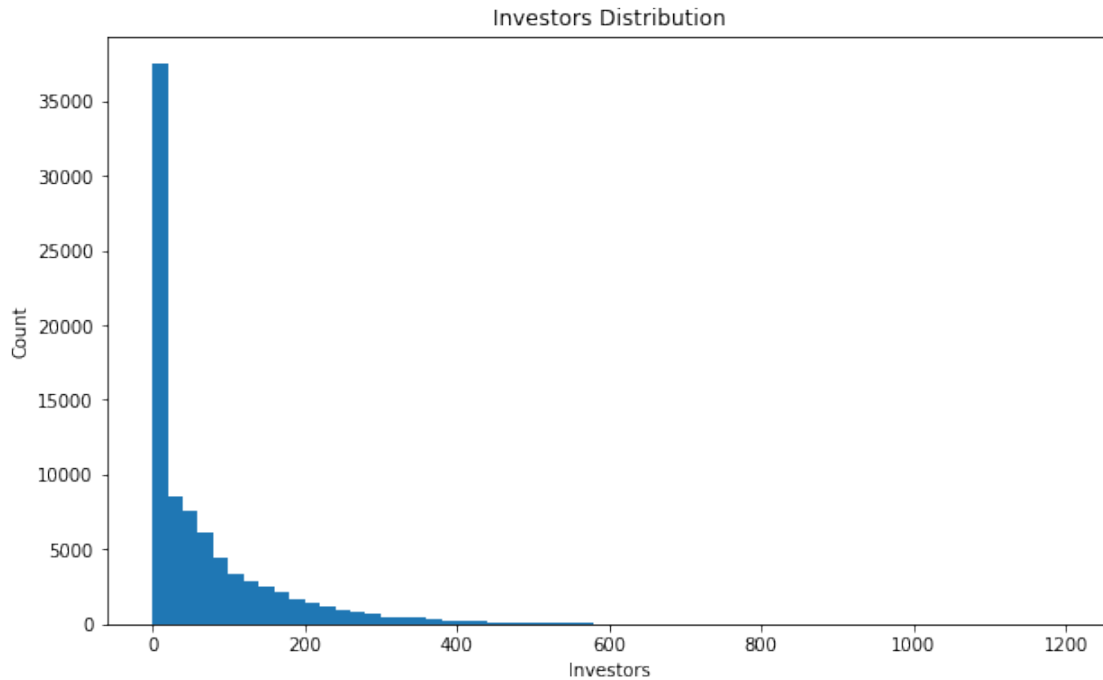
Majority of the borrowers come from California(CA), they're employed, earn between 25,000 to 74,000 dollars, have a 'C' Prosper rating and often own a house

#### 1.4.20 Question

How are Investors spread out across loans?

#### 1.4.21 Visualization

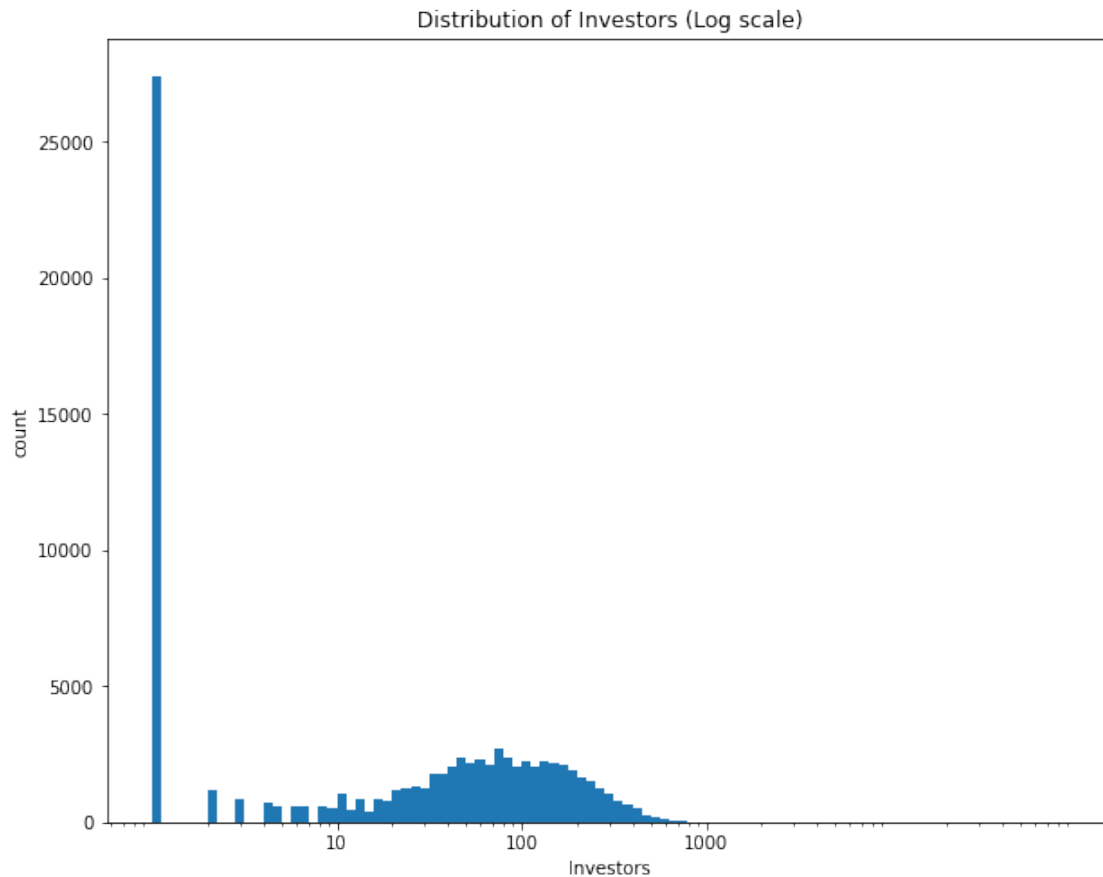
```
In [53]: plot_histogram(df, 'Investors', 20)
```



This chart is not very informative. Using a logarithmic scale would help to display the smaller values in more detail

```
In [54]: # create histogram of Investors using log scale
log_binsize = 0.05
bins = 10 ** np.arange(0, 5, log_binsize)

plt.figure(figsize=[10, 8])
plt.hist(data = df, x = 'Investors', bins = bins)
plt.title('Distribution of Investors (Log scale)')
plt.xscale('log')
plt.xticks([1e1, 1e2, 1e3], ['10', '100', '1000'])
plt.xlabel('Investors')
plt.ylabel('count');
```



#### 1.4.22 Observation

we can see that the majority of the loans have less than 10 investors, with a peak at around 90-100 investors.

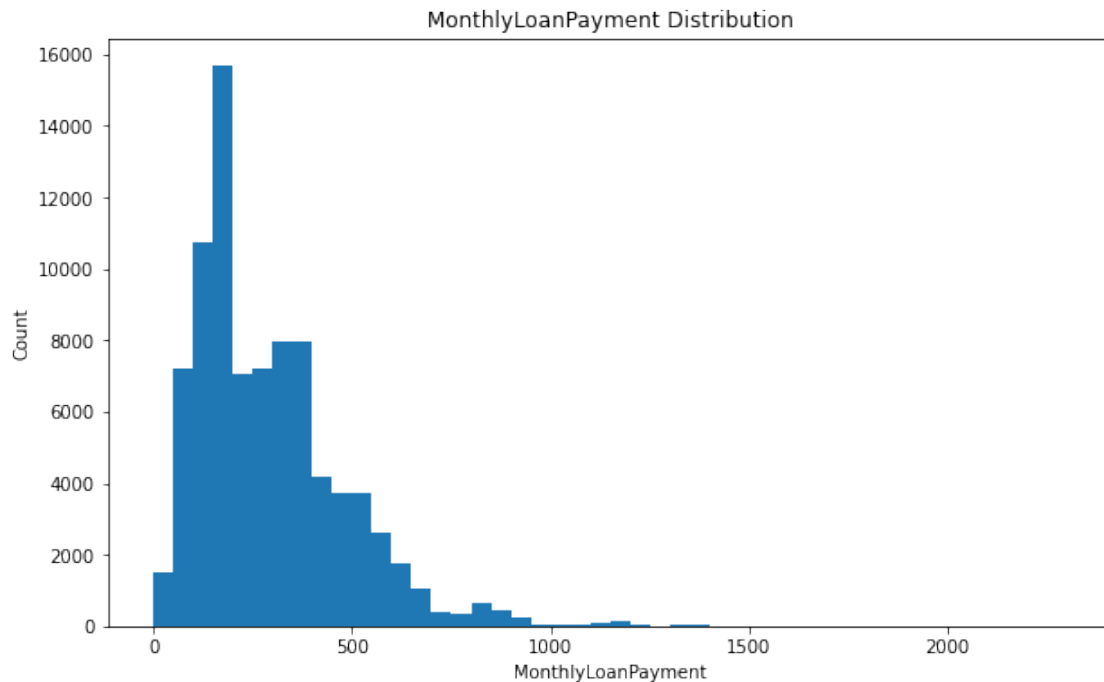
#### 1.4.23 Question

What is the distribution of loan monthly payments?

#### 1.4.24 Visualization

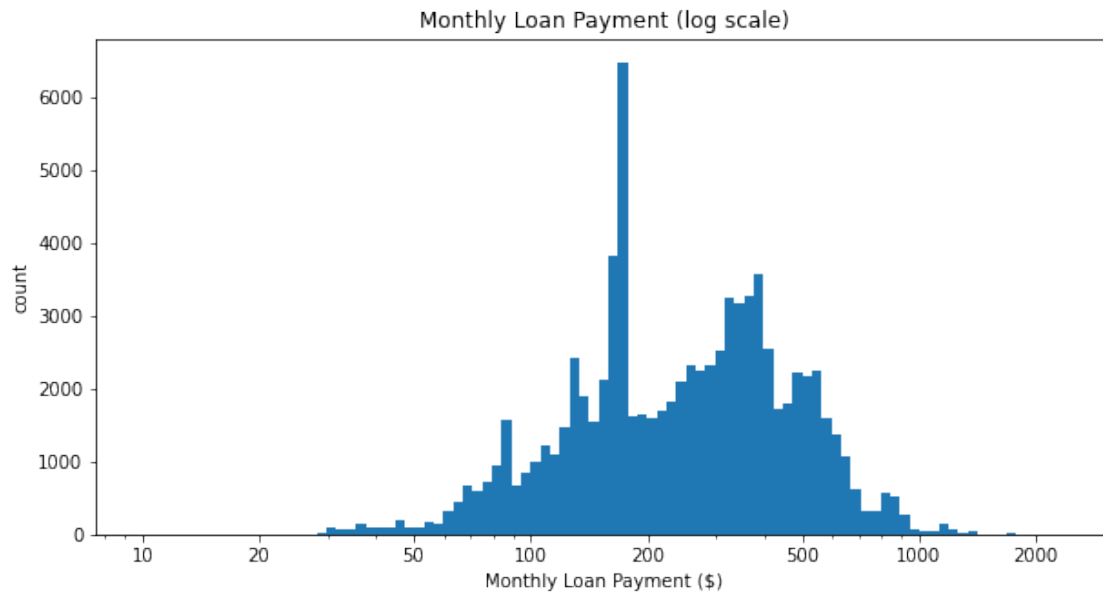
```
In [55]: plot_histogram(df, 'MonthlyLoanPayment', 50)
```





```
In [56]: # Taking a closer look using log-scale
log_binsize = 0.025
bins = 10 ** np.arange(1, np.log10(df['MonthlyLoanPayment'].max())+log_binsize, log_binsize)

plt.figure(figsize=[10, 5])
plt.hist(data = df, x = 'MonthlyLoanPayment', bins = bins)
plt.xscale('log')
plt.xticks([10, 20, 50, 100, 200, 500, 1e3, 2e3], ['10', '20', '50', '100', '200', '500', '1000', '2000'])
plt.xlabel('Monthly Loan Payment ($)')
plt.ylabel('count')
plt.title('Monthly Loan Payment (log scale)');
```



#### 1.4.25 Observation

Most of the borrowers are scheduled to pay about \$180 monthly, although a considerable amount are also scheduled to pay between 300 to 500 dollars monthly

#### 1.4.26 Question

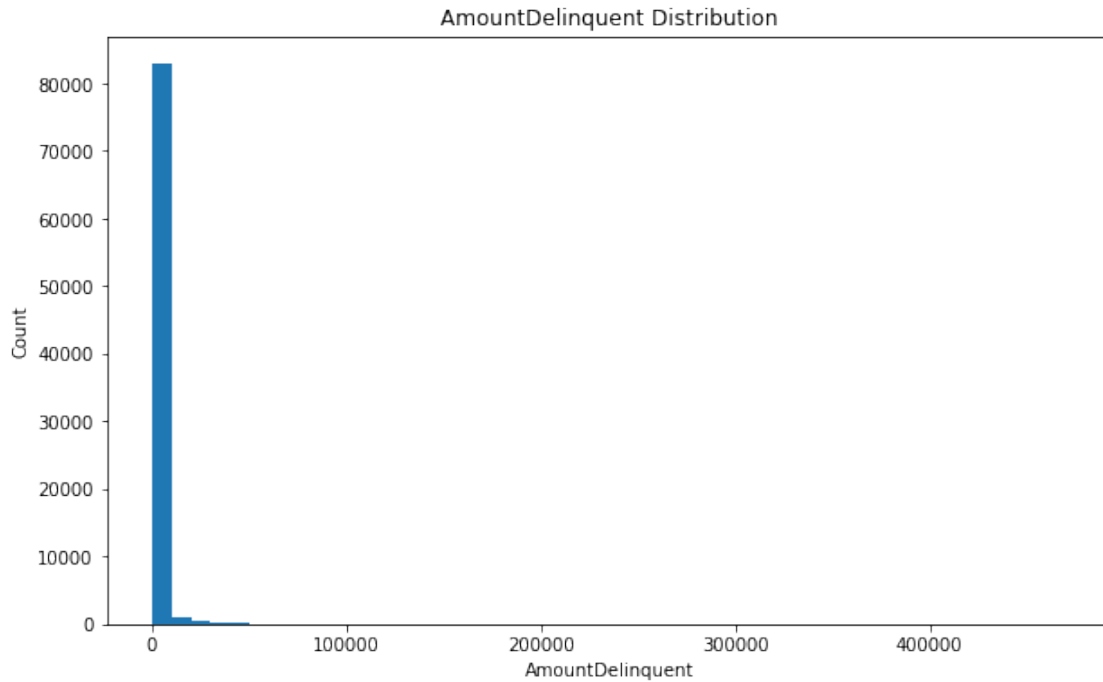
What is the distribution of delinquent loans?

#### 1.4.27 Visualization

```
In [57]: df['AmountDelinquent'].describe()
```

```
Out[57]: count      84853.000000
         mean        950.773797
         std         7419.574684
         min           0.000000
         25%           0.000000
         50%           0.000000
         75%           0.000000
         max        463881.000000
         Name: AmountDelinquent, dtype: float64
```

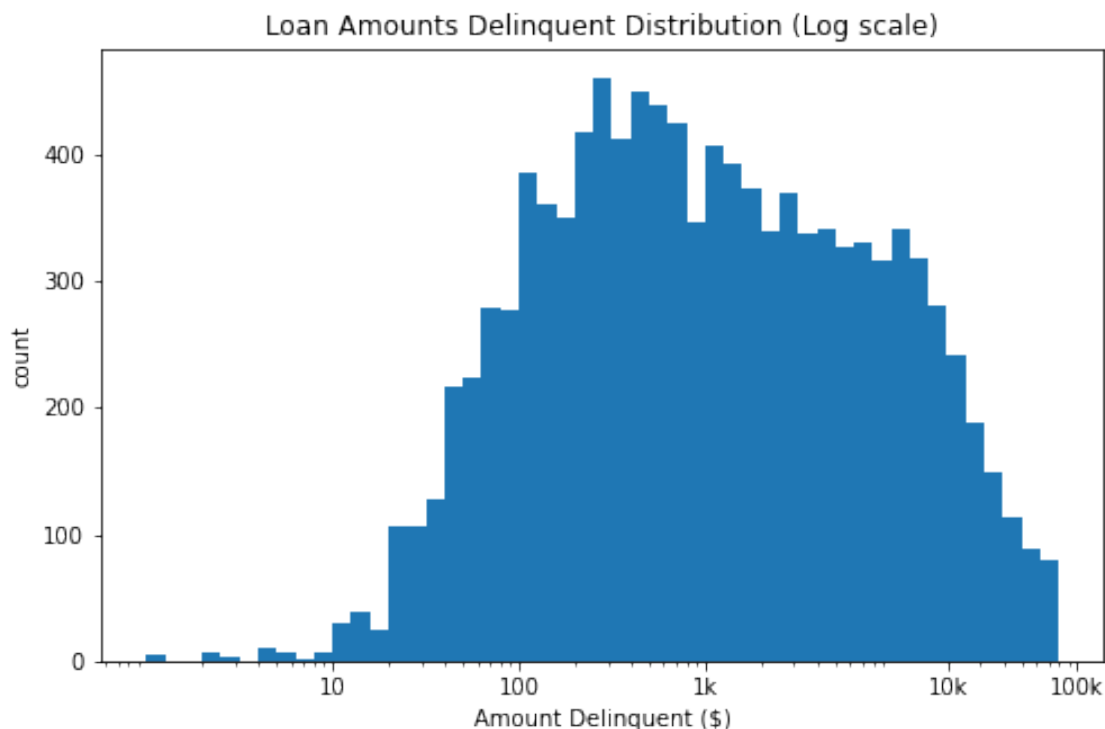
```
In [58]: plot_histogram(df, 'AmountDelinquent', 10000)
```



In [59]: *#Amount Delinquent Distribution on a log scale*

```
log_binsize = 0.1
bins = 10 ** np.arange(0,5, log_binsize)

plt.figure(figsize=[8, 5])
plt.hist(data = df, x = 'AmountDelinquent', bins = bins)
plt.title('Loan Amounts Delinquent Distribution (Log scale)')
plt.xscale('log')
plt.xticks([1e1, 1e2, 1e3, 2e4, 1e5], ['10', '100', '1k', '10k', '100k'])
plt.xlabel('Amount Delinquent ($)')
plt.ylabel('count');
```



#### 1.4.28 Observation

Most loan amounts delinquent falls around \$950 and it is normally distributed

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

Loan Original Amount distribution is right skewed and Borrower Rate distribution is multimodal

#### 1.4.30 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 performed log transformation on the Investors, MonthlyLoanPayments and Amount-Delinquent columns in order to take a closer look at the smaller values and interpret the visualization

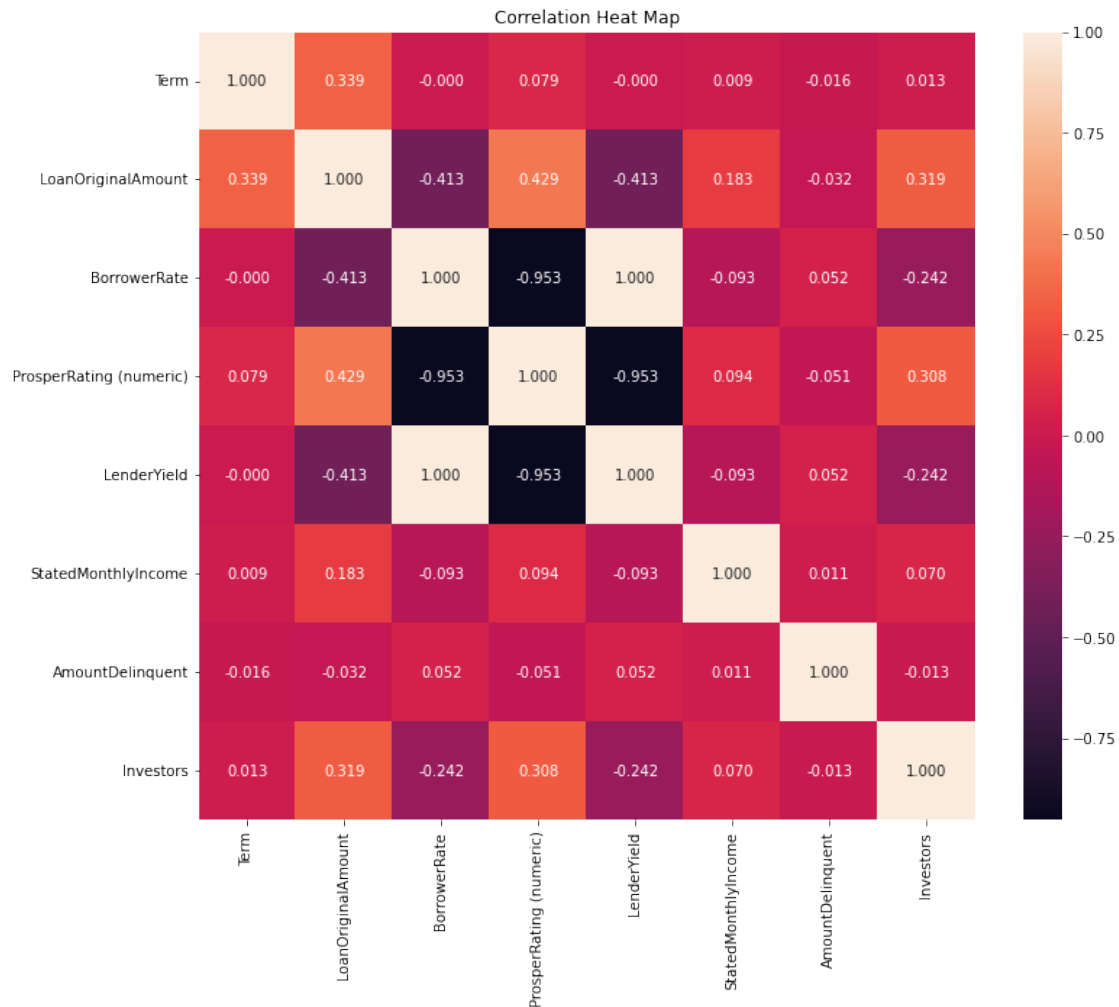
### 1.5 Bivariate Exploration

I'd start by looking at the relationships between Numerical variables

```
In [60]: numerical_vars = ['Term', 'LoanOriginalAmount', 'BorrowerRate', 'ProsperRating (numeric)',
                          'LenderYield', 'StatedMonthlyIncome', 'AmountDelinquent', 'Investors']
```

```
categorical_vars = ['Term','ProsperRating (Alpha)', 'EmploymentStatus','Occupation',
                    'IncomeRange','LoanStatus','BorrowerState','ListingCategory (numeri
```

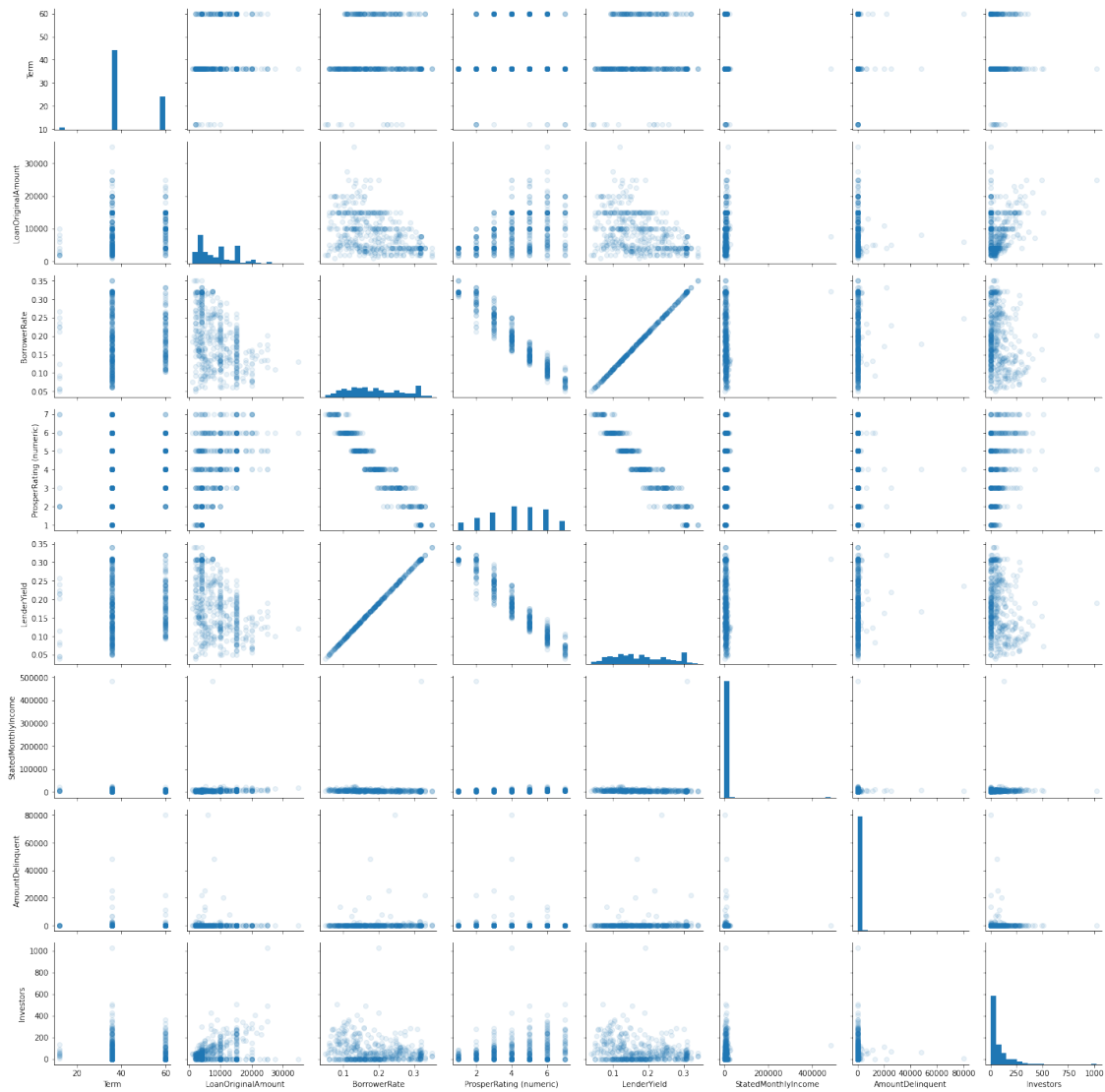
```
In [61]: plt.figure(figsize = [12, 10])
sns.heatmap(df[numerical_vars].corr(), annot = True, fmt = '.3f',
            cmap = 'rocket', center = 0)
plt.title('Correlation Heat Map');
```



```
In [62]: # plot matrix: sample of 600
print("df.shape=",df.shape)
df_samp = df.sample(n=500, replace = False)
print("df_samp.shape=",df_samp.shape)

g = sns.PairGrid(data = df_samp, vars = numerical_vars)
g = g.map_diag(plt.hist, bins = 20);
g.map_offdiag(plt.scatter,alpha=0.1);
```

```
df.shape= (84853, 23)
df_samp.shape= (500, 23)
```



### 1.5.1 Observation

- LoanOriginalAmount and BorrowerRate are negatively correlated with coefficient of -0.413 which suggests that larger loans have lower borrower rate compared to lesser loans
- Strong positive correlations between Lender yield and Borrower Rate which is logical, higher borrower rates should result in higher lender yield
- Strong negative correlation between Prosper rating and borrower rate, which suggests that high prosper ratings have lower borrower rates

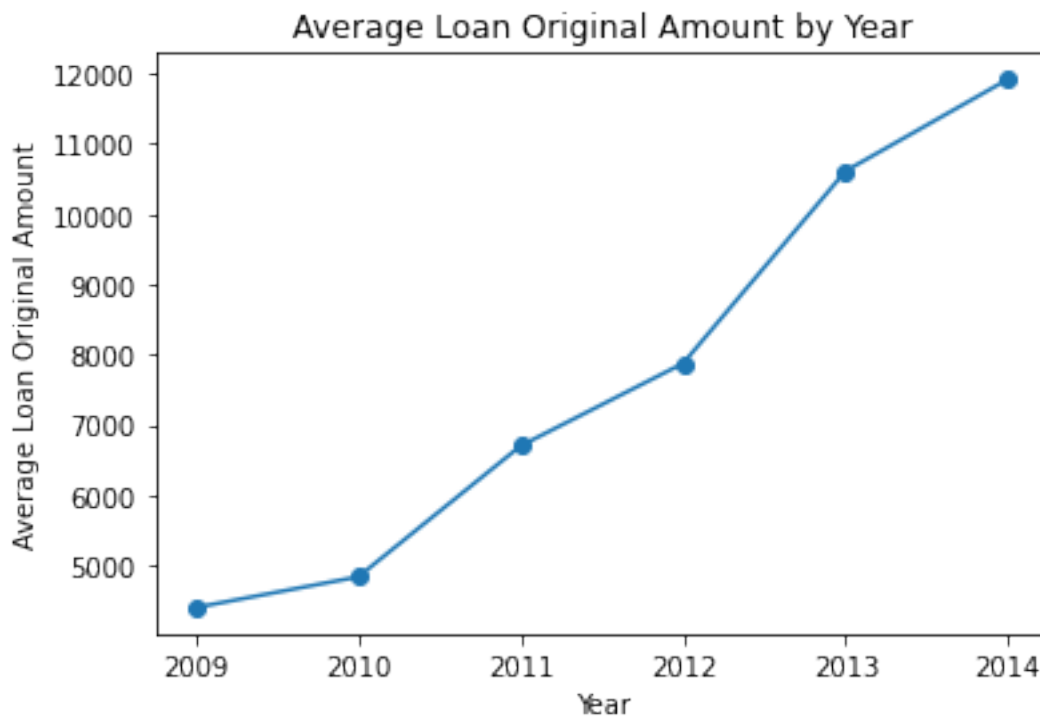
### 1.5.2 Question

How has Loan amount changed over time?

### 1.5.3 Visualization

```
In [63]: # Calculate the average loan original amount by year
avg_loan_by_year = df.groupby('ListingYear')['LoanOriginalAmount'].mean()

# Create a line plot of the average loan original amount by year
plt.plot(avg_loan_by_year.index, avg_loan_by_year.values, marker='o')
plt.xlabel('Year')
plt.ylabel('Average Loan Original Amount')
plt.title('Average Loan Original Amount by Year');
```



```
In [64]: # How are loans distributed across each month?
```

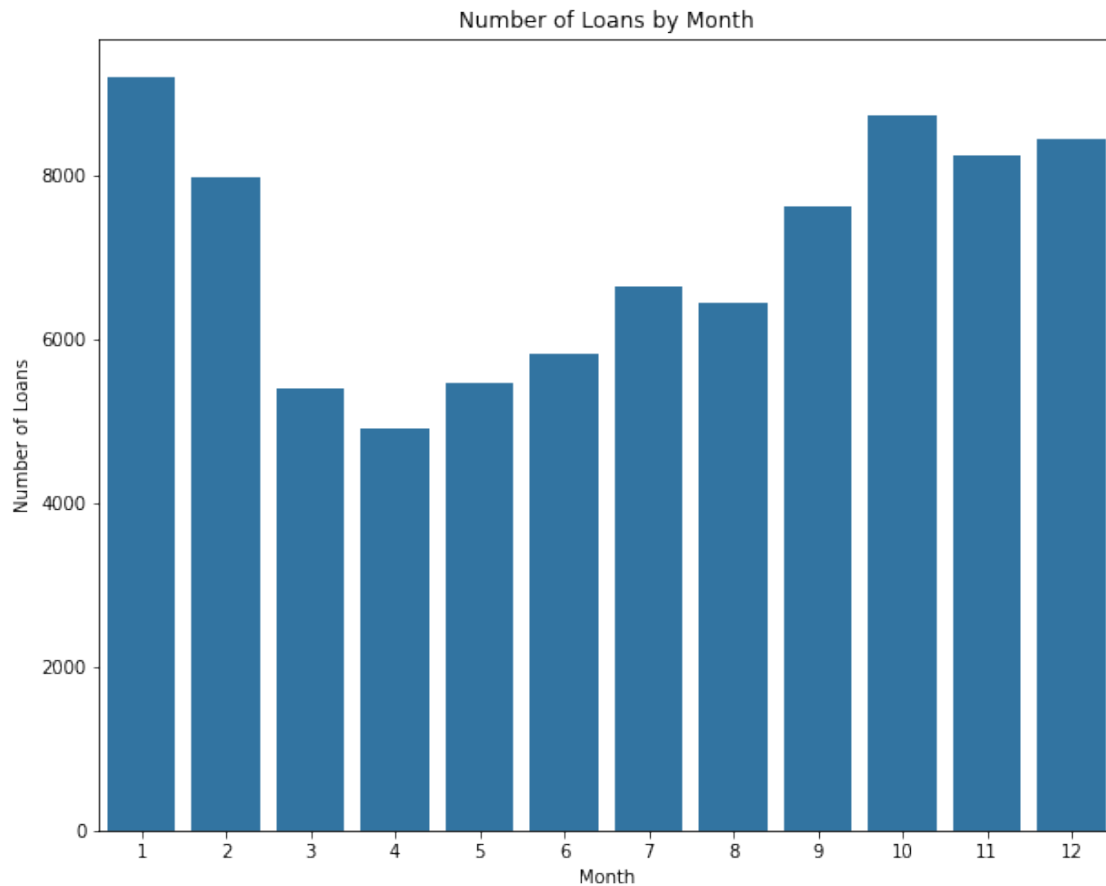
```
# Count the number of loans in each month
month_counts = df["ListingMonth"].value_counts()

# set figsize
fig, ax = plt.subplots(figsize=(10, 8))

# Create the bar chart
```

```
sns.barplot(x=month_counts.index, y=month_counts.values, color = base_color)

# Add labels and title
plt.xlabel("Month")
plt.ylabel("Number of Loans")
plt.title("Number of Loans by Month");
```



#### 1.5.4 Observation

- Loan amount has been on a continuous uptrend over the years
- January has the most number of loans

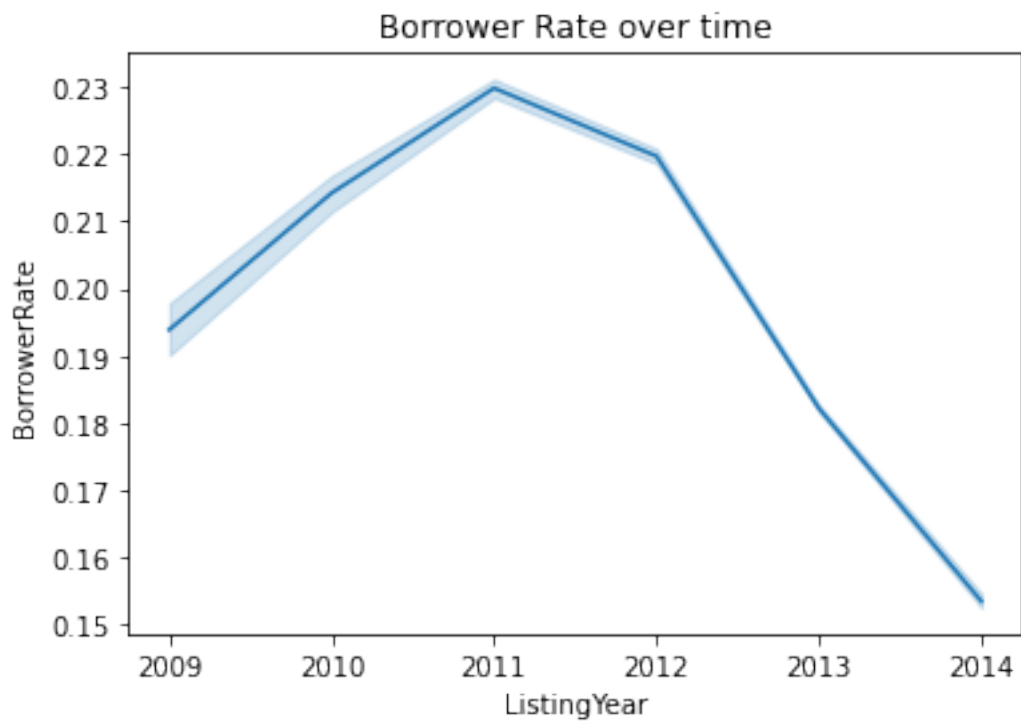
#### 1.5.5 Question

How has borrower rate changed over time?

#### 1.5.6 Visualization

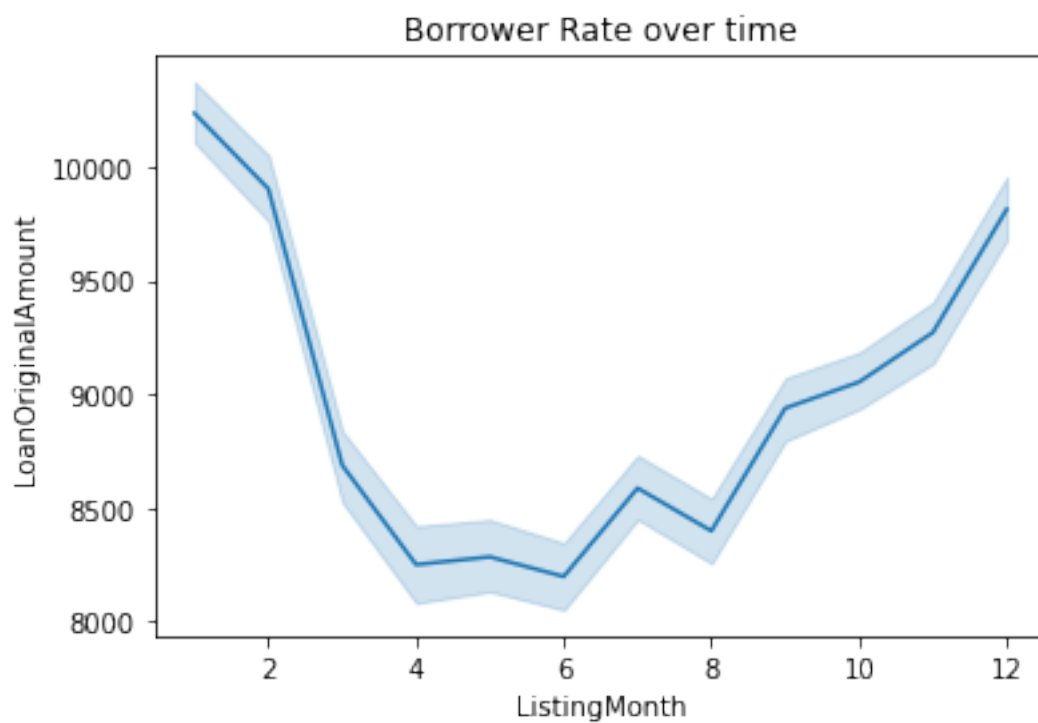
```
In [65]: sns.lineplot(x="ListingYear", y="BorrowerRate", data=df).set(title='Borrower Rate over
```





In [66]: # Taking a closer look at the borrower rate over months

```
sns.lineplot(x="ListingMonth", y="LoanOriginalAmount", data=df).set(title='Borrower Rate
```



### 1.5.7 Observation

On yearly timeframe, Borrower rate peaked at 2011 and has been on a downtrend since then

On monthly timeframe, rates are highest during january and february(which might be because the most number of loans are taken in january) but drop towards April to june

### 1.5.8 Question

How has loan status changed over time?

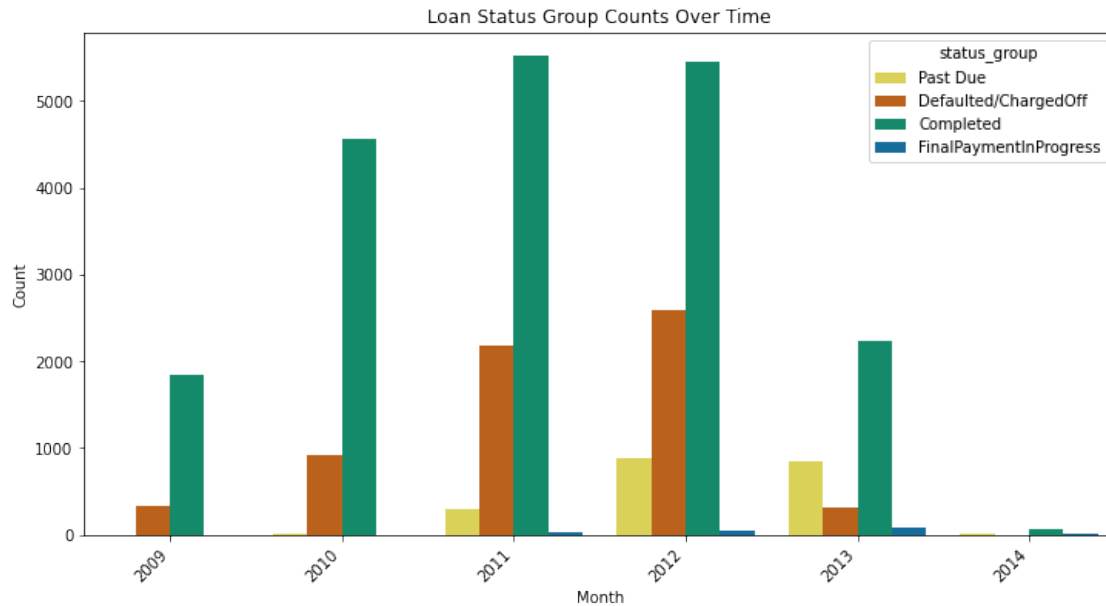
### 1.5.9 Visualization

```
In [67]: # create a dictionary to map loan statuses to their corresponding group
status_groups = {
    'Completed': 'Completed',
    'Defaulted': 'Defaulted/ChargedOff',
    'Chargedoff': 'Defaulted/ChargedOff',
    'Past Due (1-15 days)': 'Past Due',
    'Past Due (16-30 days)': 'Past Due',
    'Past Due (31-60 days)': 'Past Due',
    'Past Due (61-90 days)': 'Past Due',
    'Past Due (91-120 days)': 'Past Due',
    'Past Due (>120 days)': 'Past Due',
    'FinalPaymentInProgress': 'FinalPaymentInProgress'
}

# create a new column for the loan status group
df['status_group'] = df['LoanStatus'].map(status_groups)

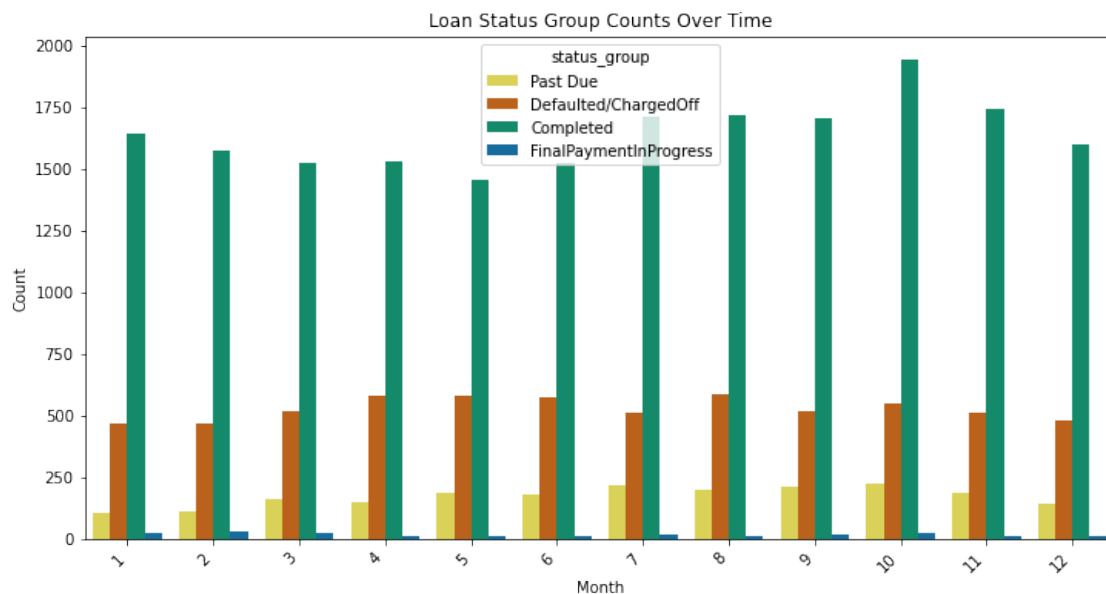
# plot the loan status group counts over the years
colored = ['#F0E442', '#D55E00', '#009E73', '#0072B2']

plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='ListingYear', hue='status_group', palette=colored)
plt.title('Loan Status Group Counts Over Time')
plt.xlabel('Month')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right');
```



In [68]: # Taking a closer look on a monthly timeframe

```
# plot the loan status group counts over months
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='ListingMonth', hue='status_group', palette=colored)
plt.title('Loan Status Group Counts Over Time')
plt.xlabel('Month')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right');
```



### 1.5.10 Observation

- 2011 has the highest number of completed loans
- The year with the highest delinquent loans is 2012
- October has the highest completed loans

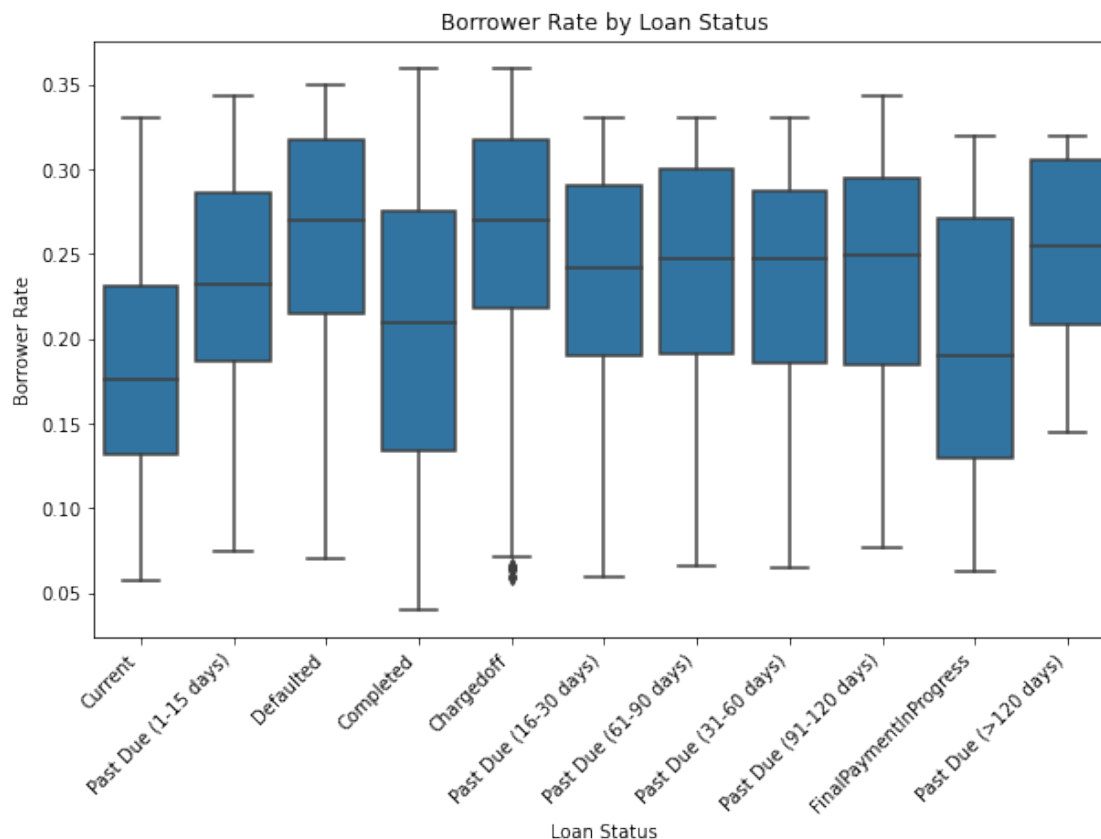
### 1.5.11 Question

What is the relationship between Borrower Rate and Loan Status?

### 1.5.12 Visualization

In [69]: *# create a box plot of borrower rate by loan status*

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='LoanStatus', y='BorrowerRate', color = base_color)
plt.xticks(rotation=45, ha='right')
plt.title('Borrower Rate by Loan Status')
plt.xlabel('Loan Status')
plt.ylabel('Borrower Rate')
plt.show()
```



### 1.5.13 Observation

We can see that the median borrower rate is generally higher for loans that are defaulted, charged-off or past due compared to completed loans. Completed and current loans have median borrower rate of 0.22 and 0.18 respectively.

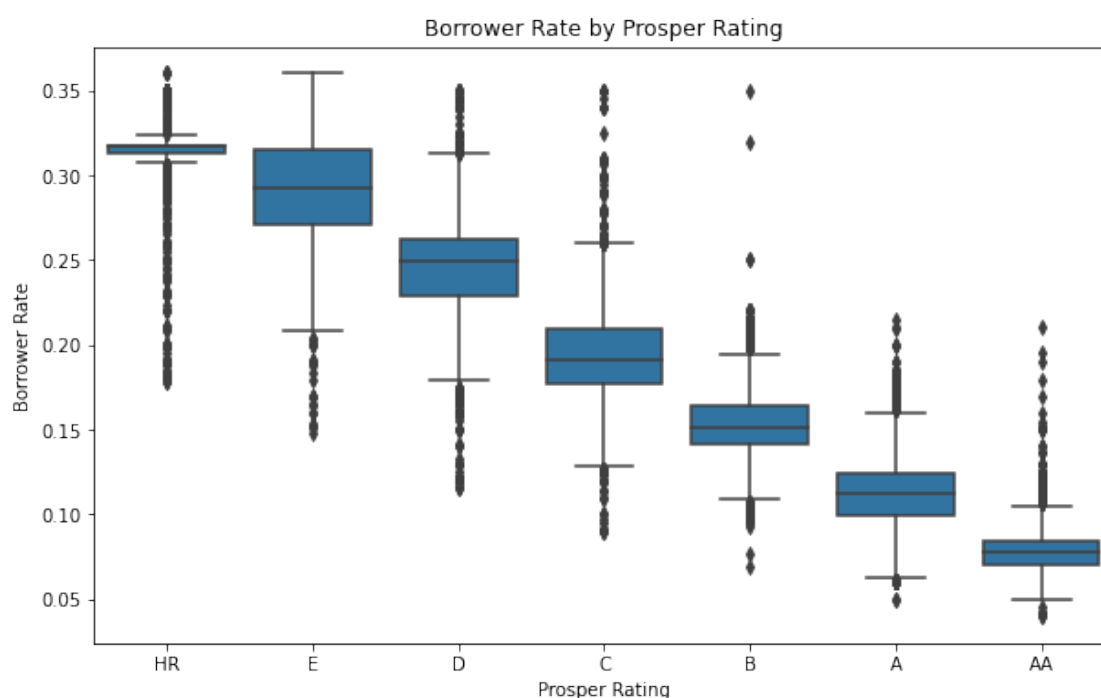
This suggests that lower borrower rates are associated with higher completion rates, while higher borrower rates are associated with higher rates of default, charge off, and past due status.

### 1.5.14 Question

What is the relationship between Borrower rate and Prosper rating?

### 1.5.15 Visualization

```
In [70]: # Create box plot
plt.figure(figsize=(10, 6))
sns.boxplot(x='ProsperRating (Alpha)', y='BorrowerRate', data=df, color = base_color)
plt.title('Borrower Rate by Prosper Rating')
plt.xlabel('Prosper Rating')
plt.ylabel('Borrower Rate');
```



### 1.5.16 Observation

Borrower rates are highest for HR(High risk) Prosper rating and lowest for the highest Prosper rating AA This shows that borrowers with better Prosper rating get lower rates

### 1.5.17 Question

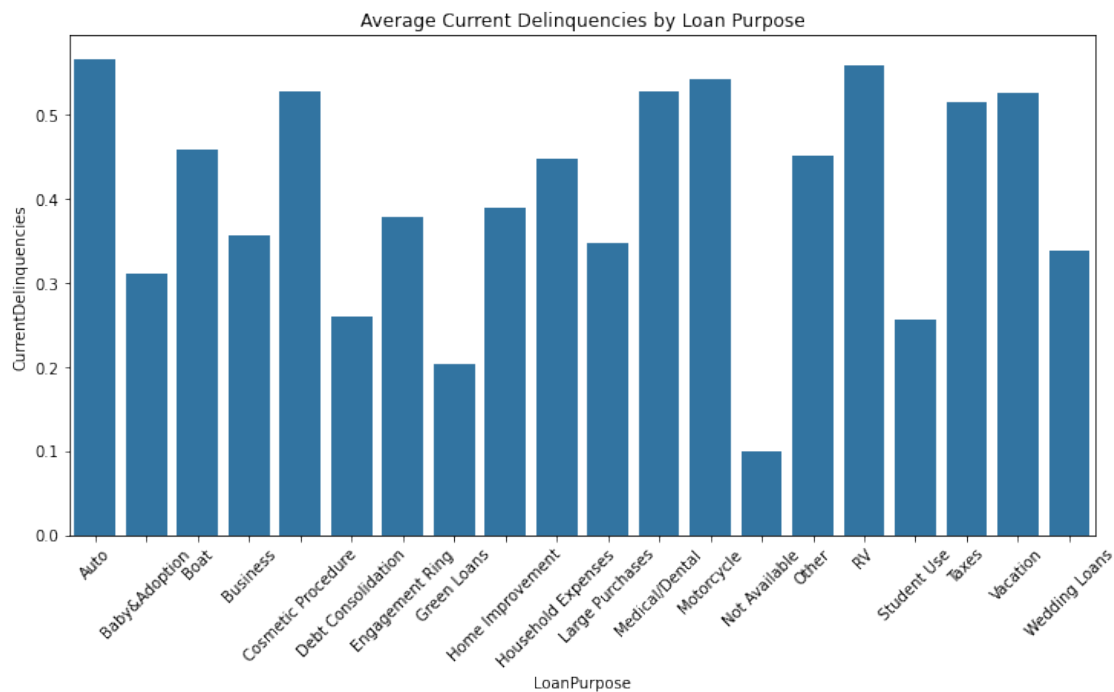
What listing category is likely to (1) default (2) Be Completed

### 1.5.18 Visualization

```
In [71]: # create a new column with the loan purposes
df['LoanPurpose'] = df['ListingCategory (numeric)'].map(listing_category)

# group the data by loan purpose and calculate the mean current delinquencies
delinquencies_by_purpose = df.groupby('LoanPurpose')['CurrentDelinquencies'].mean().reset_index()

# create a barplot of loan purposes and current delinquencies
plt.figure(figsize=(12, 6))
sns.barplot(x='LoanPurpose', y='CurrentDelinquencies', data=delinquencies_by_purpose, color='blue')
plt.title('Average Current Delinquencies by Loan Purpose')
plt.xticks(rotation=45);
```



```
In [72]: # create a subset dataframe with only completed loans
completed_loans = df[df['LoanStatus'] == 'Completed']
```

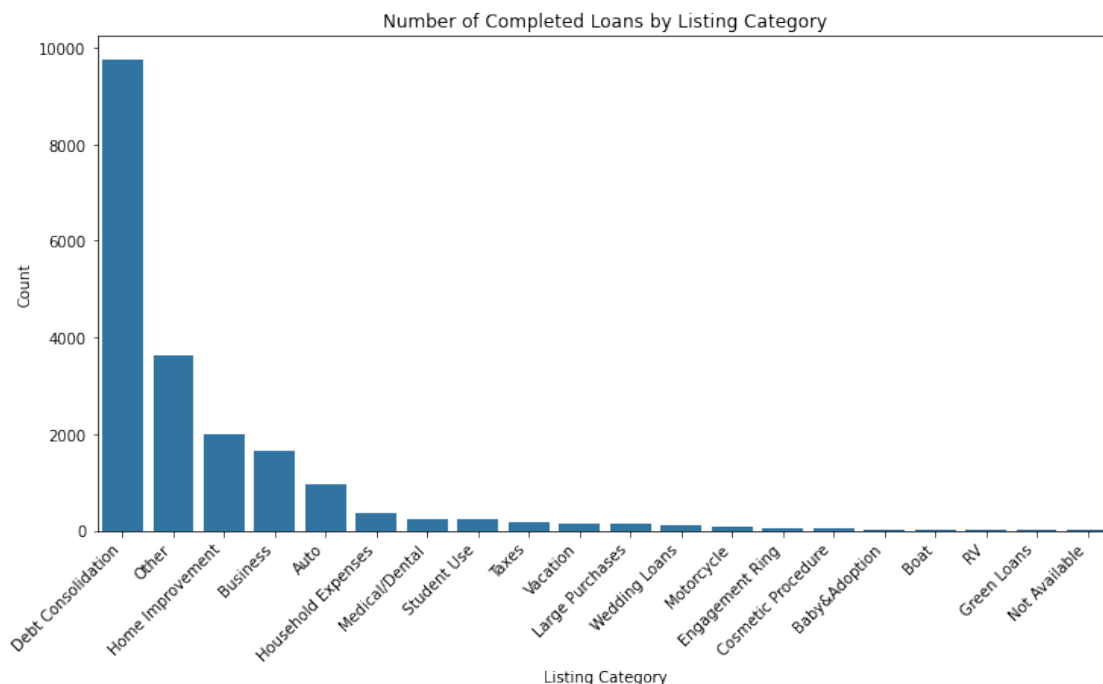
```

# count the number of completed loans for each listing category
listing_counts = completed_loans['ListingCategory (numeric)'].value_counts()

# map the numeric listing categories to their corresponding names
listing_counts.index = listing_counts.index.map(listing_category)

# create a bar plot of completed loans by listing category
plt.figure(figsize=(12, 6))
sns.barplot(x=listing_counts.index, y=listing_counts.values, color= base_color)
plt.title('Number of Completed Loans by Listing Category')
plt.xlabel('Listing Category')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right');

```



### 1.5.19 Observation

- Loans requested for the purpose of Auto have the highest delinquencies It suggests that loans for Auto, Business, Large purchases, Medical/ dental and RV may be riskier than other purposes like debt consolidation and student use.
- Loans taken for debt consolidation are most likely to be completed

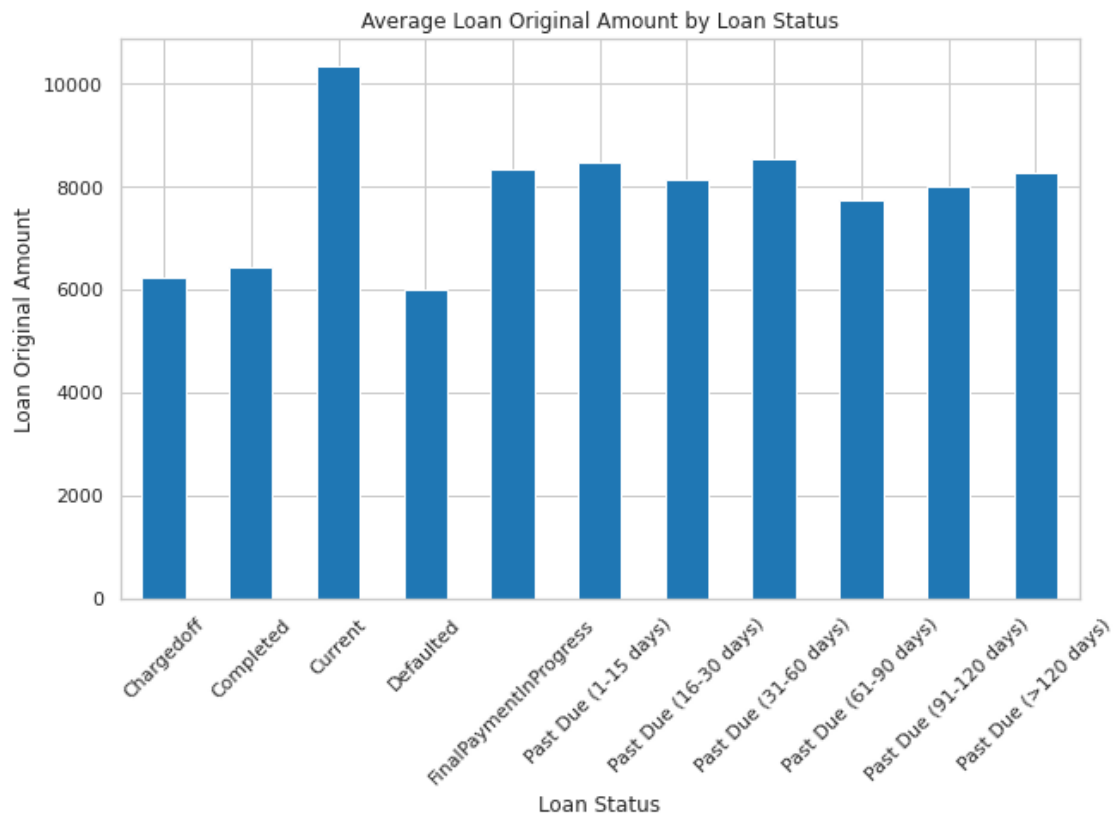
### 1.5.20 Question

What is the relationship between LoanOriginalAmount and LoanStatus?

### 1.5.21 Visualization

```
In [86]: # Group the dataset by LoanStatus and calculate the mean loan original amount for each
grouped = df.groupby('LoanStatus')['LoanOriginalAmount'].mean()

# Create a bar chart of the mean loan original amount for each loan status
grouped.plot(kind='bar', figsize=(10,6), color = base_color)
plt.title('Average Loan Original Amount by Loan Status')
plt.xlabel('Loan Status')
plt.xticks(rotation=45)
plt.ylabel('Loan Original Amount');
```



### 1.5.22 Observation

Larger loans are more likely to be in the "Current" status, while smaller loans are more likely to be in the "Defaulted" and "Chargedoff" categories.

### 1.5.23 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?

Strong positive correlations between Lender yield and Borrower Rate which is logical, higher borrower rates should result in higher lender yield



Strong negative correlation between Prosper rating and borrower rate, which suggests that high prosper ratings have lower borrower rates

2011 had the highest borrower rates and highest completed loans  
Loans taken for debt consolidation are most likely to be completed

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

Loans taken for the purpose of Auto have the highest delinquency

## 1.6 Multivariate Exploration

### 1.6.1 Question

What is the distribution of Loan amounts by Borrower rate and Prosper rating?

### 1.6.2 Visualization

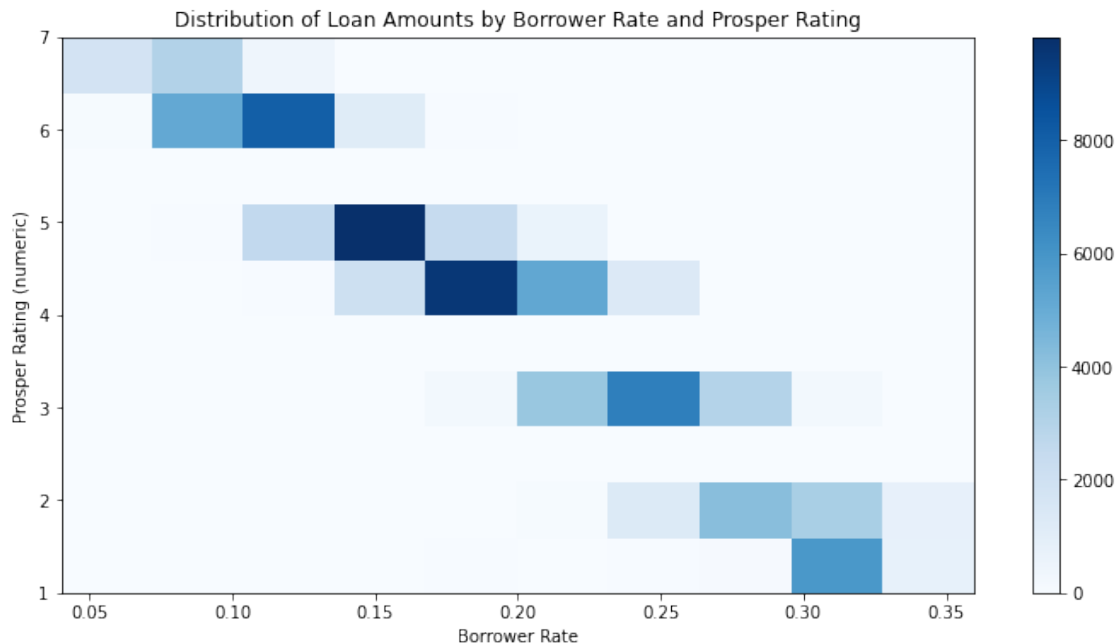
```
In [74]: median_loan_by_rating_rate = df.pivot_table(index='BorrowerRate', columns='ProsperRating', values='LoanAmount', aggfunc='median')

# plot 2D histogram
plt.figure(figsize=(12, 6))
plt.hist2d(x=df['BorrowerRate'], y=df['ProsperRating (numeric)'], bins=(10,10), cmap=pl

# add a colorbar
plt.colorbar()

# set the title and axis labels

plt.title('Distribution of Loan Amounts by Borrower Rate and Prosper Rating')
plt.xlabel('Borrower Rate')
plt.ylabel('Prosper Rating (numeric)');
```



### 1.6.3 Observation

We can see that prosper ratings of 4 (same as C- rating) and above have the largest Loan amounts but lowest borrower rates

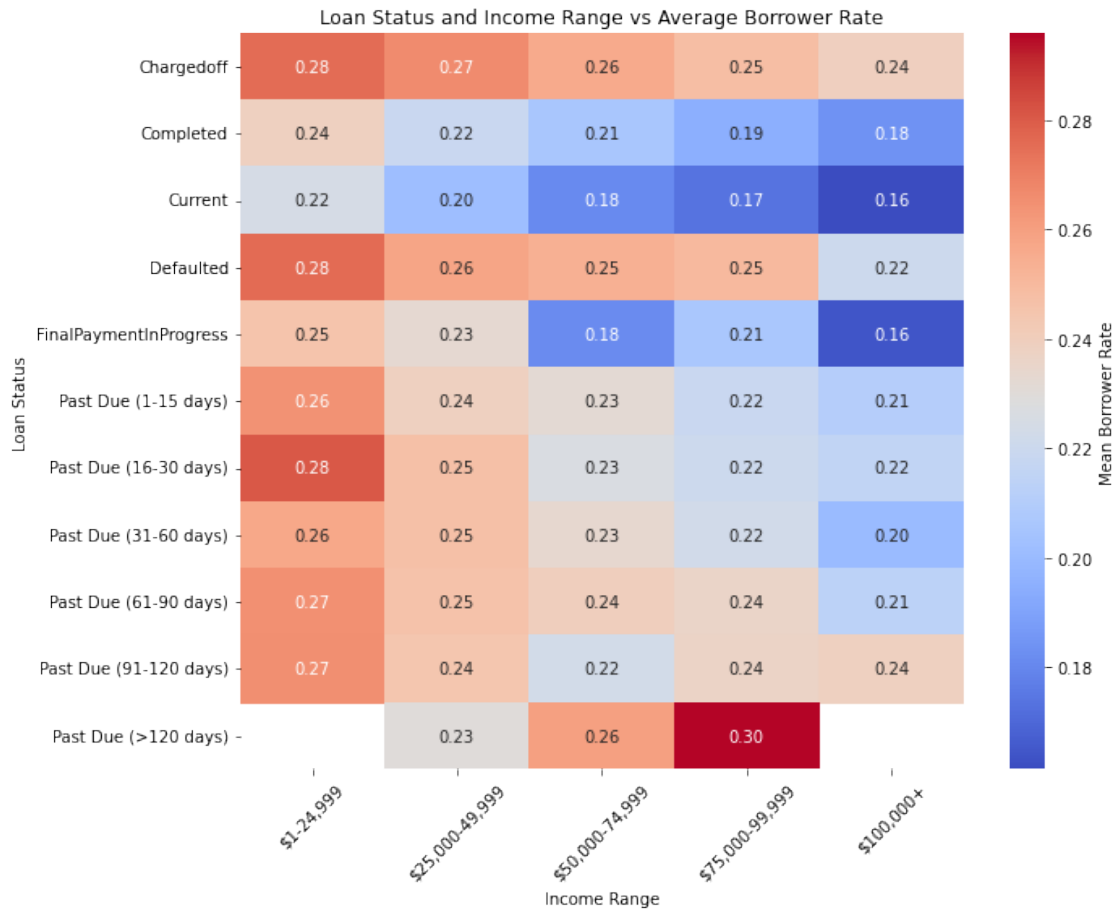
### 1.6.4 Question

What is the average borrower rate across Loan status and income range?

### 1.6.5 Visualization

In [75]: # create heatmap of Mean Borrower rate across Loan status and Income Range

```
plt.figure(figsize=(10, 8))
pivot_table = pd.pivot_table(df, values='BorrowerRate', index='LoanStatus', columns='Income Range')
sns.heatmap(pivot_table, cmap='coolwarm', annot=True, fmt='.2f', cbar_kws={'label': 'Mean Borrower Rate'})
plt.title('Loan Status and Income Range vs Average Borrower Rate')
plt.xlabel('Income Range')
plt.xticks(rotation=45)
plt.ylabel('Loan Status')
plt.tight_layout();
```



### 1.6.6 Observation

- borrowers in higher income ranges tend to have lower rates across all loan statuses, with the lowest rates concentrated in the \$100,000+ income range
- Loans that are charged off or defaulted tend to have higher rates across all income ranges, while completed loans tend to have the lowest rates

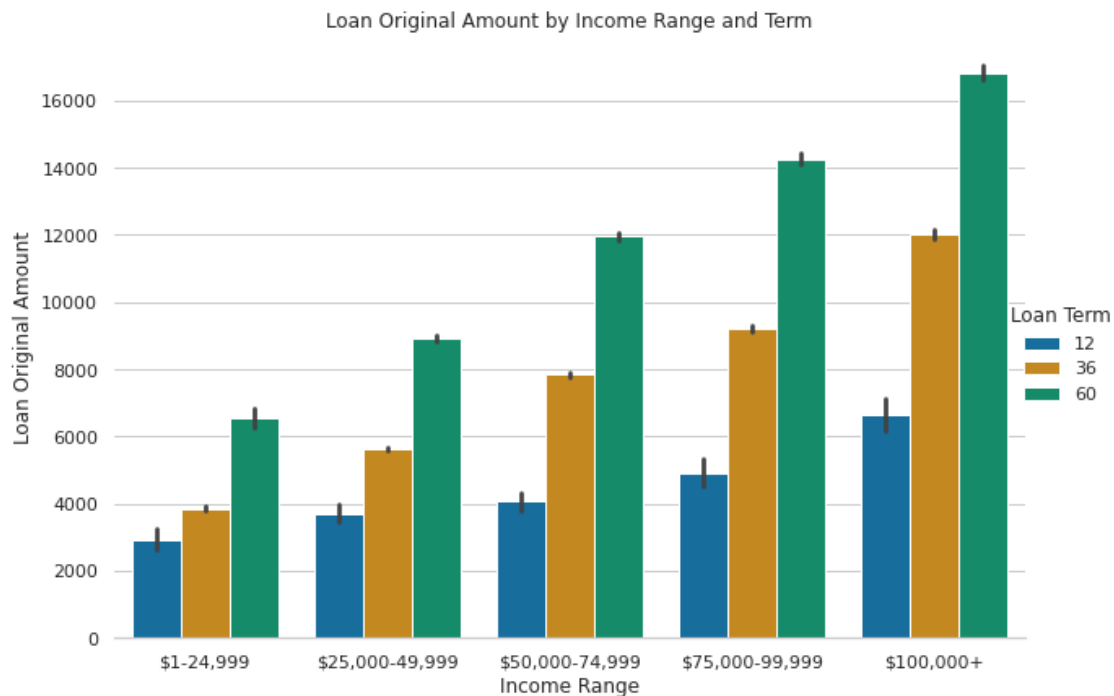
### 1.6.7 Question

How does Loan amount interact with Income range and Loan Term?

### 1.6.8 Visualization

```
In [76]: # create a stacked bar chart of LoanOriginalAmount by IncomeRange and Term
sns.set(style="whitegrid")
g = sns.catplot(x="IncomeRange", y="LoanOriginalAmount", hue="Term", data=df,
                kind="bar", palette="colorblind", height=6, aspect=1.5)
g.despine(left=True)
```

```
g.set_axis_labels("Income Range", "Loan Original Amount")
g.legend.set_title("Loan Term")
plt.title('Loan Original Amount by Income Range and Term');
```



### 1.6.9 Observation

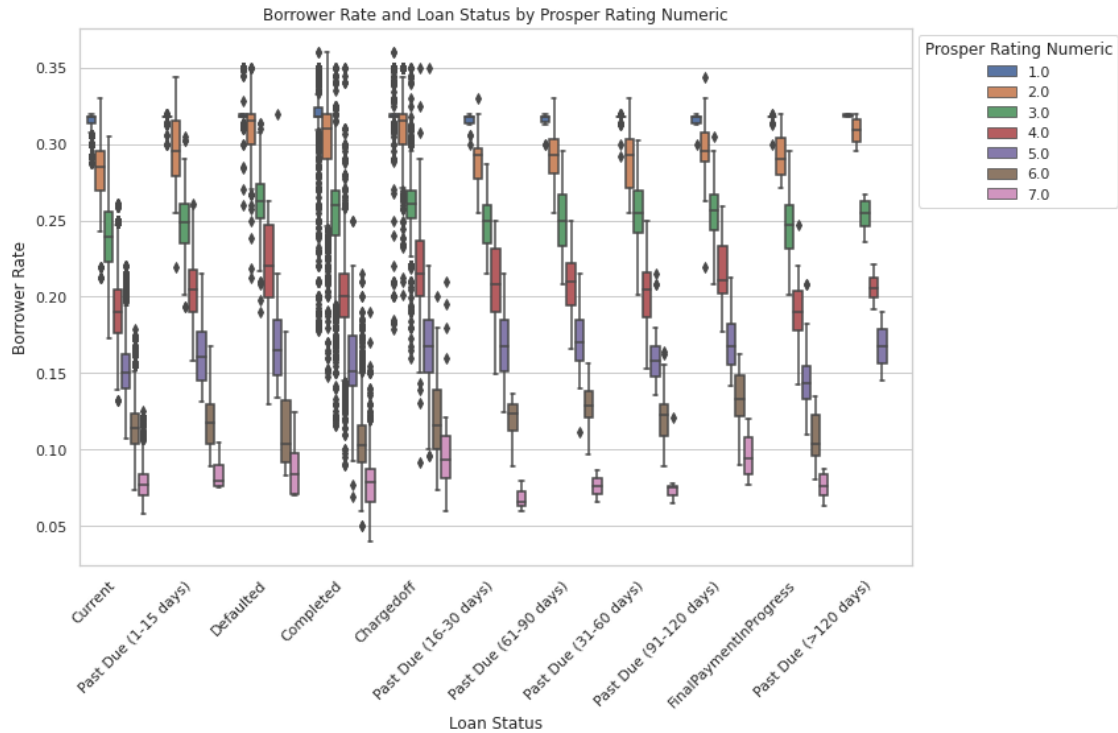
- The amount of loans increases as income range increases
- The chart suggests that Prosper tends to give larger loans to individuals with higher incomes and the 60-month term is the most popular among borrowers.

### 1.6.10 Question

What is the distribution of Borrower rate by Loan status and Prosper rating?

### 1.6.11 Visualization

```
In [77]: # create a box plot to show the distribution of borrower rate by loan status and prosper rating
plt.figure(figsize=(12, 8))
sns.boxplot(data=df, x='LoanStatus', y='BorrowerRate', hue='ProsperRating (numeric)')
plt.title('Borrower Rate and Loan Status by Prosper Rating Numeric')
plt.xlabel('Loan Status')
plt.ylabel('Borrower Rate')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Prosper Rating Numeric', bbox_to_anchor=(1, 1))
plt.tight_layout()
```



### 1.6.12 Observation

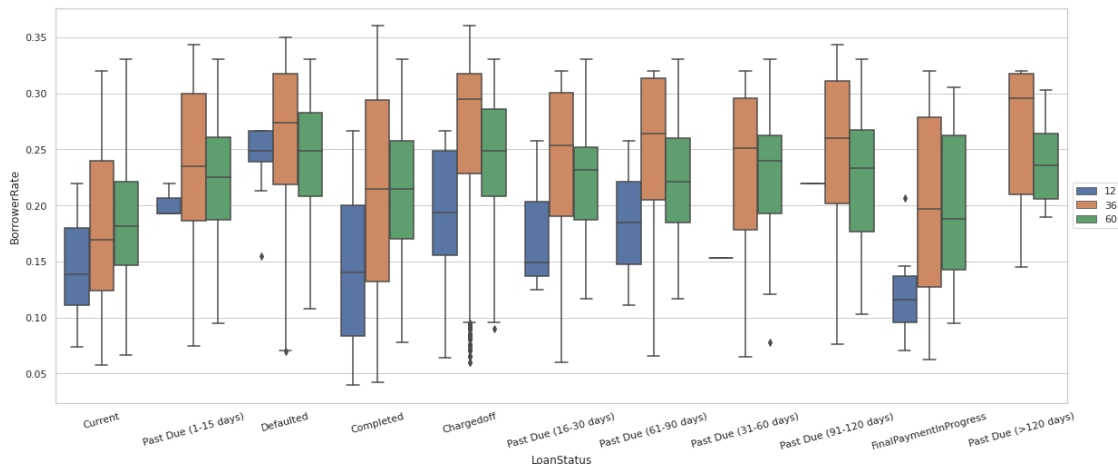
- Borrower rate tends to be higher for loans that are charged off or defaulted compared to loans that are current or have been paid off.
- The borrower rate tends to decrease as the Prosper rating increases, with lower ratings associated with higher rates
- Borrower rate is also lower for completed loans

### 1.6.13 Question

How does Borrower rate and Loan status interact with loan term?

### 1.6.14 Visualization

```
In [78]: #Creating a boxplot for BorrowerRate, LoanStatus, and Term with the Term as color encoding
plt.figure(figsize=[20,8])
sns.boxplot(data = df, y = 'BorrowerRate', x = 'LoanStatus', hue = 'Term')
plt.legend(loc = 6, bbox_to_anchor = (1.0, 0.5)) # legend to right of figure
plt.xticks(rotation = 15);
```



### 1.6.15 Observation

- The median borrower rate is generally higher for loans that have defaulted or charged off compared to loans that are current or completed
- The median borrower rate tends to increase as the loan term gets longer, with the 60-month term having the highest median rate.

### 1.6.16 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?

Prosper rating increased borrower amount but had negative relationship with borrower rates

Borrower amount increased with income range

As the Loan term increased, borrower rates generally increased

Borrowers in higher income ranges tend to have lower rates across all loan statuses

### 1.6.17 Were there any interesting or surprising interactions between features?

Loan Amount increased over the years while borrower rate reduced over the years

## 1.7 Conclusions

### 1.7.1 Steps Taken

I started by exploring the distribution of the various variables including the main variables of interest which include distribution of Loan amount, Listing Category, Term, Loan status, borrower rate and lender yield.

I looked at some borrower characteristics which are Borrower state, employment status, Income range, prosper rating and homeowners. I also viewed the distribution of investors, monthly loan payments and amount delinquent.

Some of the charts(Investors, MonthlyLoanPayments and AmountDelinquent) were not informative at first hence I visualized on a log-scale for a more detailed look.

Next I created a heatmap and plot matrix to visualize numerical variables before proceeding to check for the relationship between various variables

In the Multivariate section, I visualized how a combination of some of the main variables (Loan Amount, Borrower rate and Loan status) interacted with other variables like Prosper rating, Term, Income range,

## 1.8 Main Findings

- Average loan amount is about 5000 dollars and there are relatively more loans with lower amounts and fewer loans with higher amounts with an average borrower rate of 0.2
- Majority of the borrowers come from California(CA), they're employed, earn between 25,000 to 74,000 dollars, have a 'C' Prosper rating and often own a house
- LoanOriginalAmount and BorrowerRate are negatively correlated which suggests that larger loans have lower borrower rate compared to lesser loans
- Strong positive correlations between Lender yield and Borrower Rate which is logical, higher borrower rates should result in higher lender yield
- Strong negative correlation between Prosper rating and borrower rate, which suggests that high prosper ratings have lower borrower rates
- Loan amounts increased over the years with january recording the most loans while borrower rates dropped over the years
- Lower borrower rates are associated with higher completion rates, while higher borrower rates are associated with higher rates of default, charge off, and past due status
- Borrower rates are highest for HR(High risk) Prosper rating and lowest for the highest Prosper rating AA This shows that borrowers with better Prosper rating get lower rates
- Loans for Auto, Business, Large purchases, Medical/dental and RV may be riskier than other purposes like debt consolidation and student use. Loans taken for debt consolidation are most likely to be completed
- We can see that prosper ratings of 4 (same as C- rating) and above have the largest Loan amounts but lowest borrower rates
- Borrowers in higher income ranges tend to have lower rates across all loan statuses, with the lowest rates concentrated in the \$100,000+ income range
- Prosper tends to give larger loans to individuals with higher incomes and the 60-month term is the most popular among borrowers

- Borrower rate tends to be higher for loans that are charged off or defaulted compared to loans that are current or have been paid off.

In [ ]: