# Part I - Exploring the Prosper Loan Dataset

## By Bessan Hussein


My Github: @Bessan Hussein

My LinkedIn: @Bessan Hussein


## Table of Contents



## 1. Introduction

On November 24, 2008, the SEC found Prosper to be in violation of the Securities Act of 1933. As a result of these findings, the SEC imposed a cease and desist order on Prosper ... In July 2009, Prosper reopened their website for lending ("investing") and borrowing after having obtained SEC registration for its loans ("notes"). After the relaunch, bidding on loans was restricted to residents of 28 U.S. states and the District of Columbia. Borrowers may reside in any of 47 states, with residents of three states (Iowa, Maine, and North Dakota) not permitted to borrow through Prosper

In this notebook, the analysis is done on the Prosper Datatset which is collected from a Loan company. The dataset includes customers who have paid off their loans, who have been past due and put into collection without paying back their loan and interests, and who have paid off only after they were put in collection. The original dataset contains 113937 rows and 81 columns out of which 12 features of intrest were selected.

### Data Dictionary

The following data dictionary shows each variable of the dataset and the corresponding description:

| Variable | Description |
| --- | --- |
| ListingKey | Unique key for each listing, same value as the 'key' used in the listing object in the API. |
| ListingNumber | The number that uniquely identifies the listing to the public as displayed on the website. |

| Variable | Description |
|---|---|
| ListingCreation Date | The date the listing was created. |
| CreditGrade | The Credit rating that was assigned at the time the listing went live. Applicable for listings pre-2009 period and will only be populated for those listings. |
| Term | The length of the loan expressed in months. |
| LoanStatus | The current status of the loan: Cancelled, Chargedoff, Completed, Current, Defaulted, FinalPaymentInProgress, PastDue. The PastDue status will be accompanied by a delinquency bucket. |
| ClosedDate | Closed date is applicable for Cancelled, Completed, Chargedoff and Defaulted loan statuses. |
| BorrowerAPR | The Borrower's Annual Percentage Rate (APR) for the loan. |
| BorrowerRate | The Borrower's interest rate for this loan. |
| LenderYield | The Lender yield on the loan. Lender yield is equal to the interest rate on the loan less the servicing fee. |
| EstimatedEffec tiveYield | Effective yield is equal to the borrower interest rate (i) minus the servicing fee rate, (ii) minus estimated uncollected interest on charge-offs, (iii) plus estimated collected late fees. Applicable for loans originated after July 2009. |
| EstimatedLoss | Estimated loss is the estimated principal loss on charge-offs. Applicable for loans originated after July 2009. |
| EstimatedRetur n | The estimated return assigned to the listing at the time it was created. Estimated return is the difference between the Estimated Effective Yield and the Estimated Loss Rate. Applicable for loans originated after July 2009. |
| ProsperRating (numeric) | The Prosper Rating assigned at the time the listing was created: 0 - N/A, 1 - HR, 2 - E, 3 - D, 4 - C, 5 - B, 6 - A, 7 - AA. Applicable for loans originated after July 2009. |
| ProsperRating (Alpha) | The Prosper Rating assigned at the time the listing was created between AA - HR. Applicable for loans originated after July 2009. |
| ProsperScore | A custom risk score built using historical Prosper data. The score ranges from 1-10, with 10 being the best, or lowest risk score. Applicable for loans originated after July 2009. |
| ListingCategory | The category of the listing that the borrower selected when posting their listing: 0 - Not Available, 1 - Debt Consolidation, 2 - Home Improvement, 3 - Business, 4 - Personal Loan, 5 - Student Use, 6 - Auto, 7 - Other, 8 - Baby&Adoption, 9 - Boat, 10 - Cosmetic Procedure, 11 - Engagement Ring, 12 - Green Loans, 13 - Household Expenses, 14 - Large Purchases, 15 - Medical/Dental, 16 - Motorcycle, 17 - RV, 18 - Taxes, 19 - Vacation, 20 - Wedding Loans |
| BorrowerState | The two letter abbreviation of the state of the address of the borrower at the time the Listing was created. |
| Occupation | The Occupation selected by the Borrower at the time they created the |

| Variable | Description |
|---|---|
| | listing. |
| EmploymentStatus | The employment status of the borrower at the time they posted the listing. |
| EmploymentStatusDuration | The length in months of the employment status at the time the listing was created. |
| IsBorrowerHomeowner | A Borrower will be classified as a homeowner if they have a mortgage on their credit profile or provide documentation confirming they are a homeowner. |
| CurrentlyInGroup | Specifies whether or not the Borrower was in a group at the time the listing was created. |
| GroupKey | The Key of the group in which the Borrower is a member of. Value will be null if the borrower does not have a group affiliation. |
| DateCreditPulled | The date the credit profile was pulled. |
| CreditScoreRangeLower | The lower value representing the range of the borrower's credit score as provided by a consumer credit rating agency. |
| CreditScoreRangeUpper | The upper value representing the range of the borrower's credit score as provided by a consumer credit rating agency. |
| FirstRecordedCreditLine | The date the first credit line was opened. |
| CurrentCreditLines | Number of current credit lines at the time the credit profile was pulled. |
| OpenCreditLines | Number of open credit lines at the time the credit profile was pulled. |
| TotalCreditLinespast7years | Number of credit lines in the past seven years at the time the credit profile was pulled. |
| OpenRevolvingAccounts | Number of open revolving accounts at the time the credit profile was pulled. |
| OpenRevolvingMonthlyPayment | Monthly payment on revolving accounts at the time the credit profile was pulled. |
| InquiriesLast6Months | Number of inquiries in the past six months at the time the credit profile was pulled. |
| TotalInquiries | Total number of inquiries at the time the credit profile was pulled. |
| CurrentDelinquencies | Number of accounts delinquent at the time the credit profile was pulled. |
| AmountDelinquent | Dollars delinquent at the time the credit profile was pulled. |
| DelinquenciesLast7Years | Number of delinquencies in the past 7 years at the time the credit profile was pulled. |
| PublicRecordsLast10Years | Number of public records in the past 10 years at the time the credit profile was pulled. |

| Variable | Description |
|---|---|
| PublicRecordsLast12Months | Number of public records in the past 12 months at the time the credit profile was pulled. |
| RevolvingCreditBalance | Dollars of revolving credit at the time the credit profile was pulled. |
| BankcardUtilization | The percentage of available revolving credit that is utilized at the time the credit profile was pulled. |
| AvailableBankcardCredit | The total available credit via bank card at the time the credit profile was pulled. |
| TotalTrades | Number of trade lines ever opened at the time the credit profile was pulled. |
| TradesNeverDelinquent | Number of trades that have never been delinquent at the time the credit profile was pulled. |
| TradesOpenedLast6Months | Number of trades opened in the last 6 months at the time the credit profile was pulled. |
| DebtToIncomeRatio | The debt to income ratio of the borrower at the time the credit profile was pulled. This value is Null if the debt to income ratio is not available. This value is capped at 10.01 (any debt to income ratio larger than 1000% will be returned as 1001%). |
| IncomeRange | The income range of the borrower at the time the listing was created. |
| IncomeVerifiable | The borrower indicated they have the required documentation to support their income. |
| StatedMonthlyIncome | The monthly income the borrower stated at the time the listing was created. |
| LoanKey | Unique key for each loan. This is the same key that is used in the API. |
| TotalProsperLoans | Number of Prosper loans the borrower at the time they created this listing. This value will be null if the borrower had no prior loans. |
| TotalProsperPaymentsBilled | Number of on time payments the borrower made on Prosper loans at the time they created this listing. This value will be null if the borrower had no prior loans. |
| OnTimeProsperPayments | Number of on time payments the borrower had made on Prosper loans at the time they created this listing. This value will be null if the borrower has no prior loans. |
| ProsperPaymentsLessThanOneMonthLate | Number of payments the borrower made on Prosper loans that were less than one month late at the time they created this listing. This value will be null if the borrower had no prior loans. |
| ProsperPaymentsOneMonthPlusLate | Number of payments the borrower made on Prosper loans that were greater than one month late at the time they created this listing. This value will be null if the borrower had no prior loans. |
| ProsperPrincipalBorrowed | Total principal borrowed on Prosper loans at the time the listing was created. This value will be null if the borrower had no prior loans. |
| ProsperPrincipalOutstanding | Principal outstanding on Prosper loans at the time the listing was created. This value will be null if the borrower had no prior loans. |
| ScorexChangeA | Borrower's credit score change at the time the credit profile was pulled. |

| Variable | Description |
| --- | --- |
| tTimeOfListing | This will be the change relative to the borrower's last Prosper loan. This value will be null if the borrower had no prior loans. |
| LoanCurrentDaysDelinquent | The number of days delinquent. |
| LoanFirstDefaultedCycleNumber | The cycle the loan was charged off. If the loan has not charged off the value will be null. |
| LoanMonthsSinceOrigination | Months since the loan originated. |
| LoanNumber | The number that uniquely identifies the loan to the public as displayed on the website. |
| LoanOriginalAmount | The original amount of the loan. |
| LoanOriginationDate | The date the loan originated. |
| LoanOriginationQuarter | The quarter in which the loan originated. |
| MemberKey | Unique key for each member. This is the same key that is used in the API. |
| MonthlyLoanPayment | The monthly payment (principal and interest) the borrower is required to make for this loan. |
| LP_CustomerPayments | The total payments (principal + interest) that have been made on the loan by the borrower. |
| LP_CustomerPrincipalPayments | The total principal payments that have been made on the loan by the borrower. |
| LP_InterestandFees | Interest and fees paid by the borrower. |
| LP_ServiceFees | The servicing fees paid by the borrower. |
| LP_CollectionFees | The collection fees paid by the borrower. |
| LP_GrossPrincipalLoss | Gross principal loss on the loan. |
| LP_NetPrincipalLoss | Net principal loss on the loan. |
| LP_NonPrincipalRecoverypayments | Non-principal recovery payments on the loan. |
| PercentFunded | The percentage of the loan that was funded. |
| Recommendations | Number of recommendations the borrower had at the time they created the listing. |
| InvestmentFromFriendsCount | Number of investments that were made by friends at the time the listing was created. |

| Variable | Description |
|---|---|
| InvestmentFromFriendsAmount | The dollar amount of investments that were made by friends at the time the listing was created. |
| Investors | The number of investors that funded the loan. |

# 2. Objectives

**1. Loan Performance Analysis**

**2. Credit Score and Borrower Analysis**

**3. Geographic and Demographic Analysis**

# 3. Preliminary Wrangling

- In this section, a preliminary data wrangling is done on the dataset.

```
## import all packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# suppress warnings from final output
import warnings
warnings.simplefilter("ignore")
```

## Loading the dataset

Let's Load in the dataset into a pandas dataframe:

```
df = pd.read_csv("./data/prosperLoanData.csv") ## Load the csv into
pandas dataframe
df.sample(10) ## Looking at a random sample of 10 rows.
```

```
                     ListingKey  ListingNumber
ListingCreationDate  \
18639   FD6235799041823063F13A0         794771  2013-06-03
09:20:18.070000000
50370   D3363541493021983 70EF4E        570423  2012-03-20
07:30:24.940000000
110175  2897358983943074764 36D8        902805  2013-09-17
08:01:21.233000000
40179   9352353244762308 69E8408        540013  2011-11-18
16:51:27.080000000
112082  33DF3583781490640CADE61         835311  2013-07-11
11:06:28.953000000
24027   2B3C354676945166149 56BC        589050  2012-05-15
11:09:54.097000000
```

```
49349   543D34568778950159D6C2E        416988  2009-07-17
11:32:41.037000000
33801   3E9D3583177317128AA1710        835796  2013-07-11
16:32:05.027000000
31780   C80B3365206111418538B4C         25303  2006-07-18
12:51:20.287000000
90175   26013583091014354E0A546        820473  2013-06-25
03:10:48.177000000

       CreditGrade  Term             LoanStatus
ClosedDate  \
18639          NaN    60                Current                  NaN

50370          NaN    36              Completed  2012-06-14 00:00:00

110175         NaN    36                Current                  NaN

40179          NaN    36             Chargedoff  2012-12-22 00:00:00

112082         NaN    60                Current                  NaN

24027          NaN    60  Past Due (31-60 days)                  NaN

49349          NaN    36              Completed  2012-09-14 00:00:00

33801          NaN    36                Current                  NaN

31780            E    36              Completed  2009-06-30 00:00:00

90175          NaN    60                Current                  NaN


        BorrowerAPR  BorrowerRate  LenderYield  ...  LP_ServiceFees  \
18639       0.20593        0.1819       0.1719  ...          -96.95
50370       0.33973        0.2999       0.2899  ...           -8.96
110175      0.08325        0.0699       0.0599  ...          -23.87
40179       0.35797        0.3177       0.3077  ...          -25.03
112082      0.15629        0.1334       0.1234  ...          -72.96
24027       0.35838        0.3304       0.3204  ...          -59.03
49349       0.22761        0.1900       0.1800  ...          -16.84
33801       0.13138        0.1034       0.0934  ...          -81.82
31780       0.29525        0.2875       0.2825  ...          -24.65
90175       0.20081        0.1769       0.1669  ...          -96.89

        LP_CollectionFees  LP_GrossPrincipalLoss  LP_NetPrincipalLoss
\
18639                 0.0                   0.00                 0.00

50370                 0.0                   0.00                 0.00

110175                0.0                   0.00                 0.00
```

```
40179                     0.0                  3420.73                3420.73
112082                    0.0                     0.00                   0.00
24027                     0.0                     0.00                   0.00
49349                     0.0                     0.00                   0.00
33801                     0.0                     0.00                   0.00
31780                     0.0                     0.00                   0.00
90175                     0.0                     0.00                   0.00

       LP_NonPrincipalRecoverypayments  PercentFunded  Recommendations  \
18639                              0.0            1.0                0
50370                              0.0            1.0                0
110175                             0.0            1.0                0
40179                              0.0            1.0                0
112082                             0.0            1.0                0
24027                              0.0            1.0                0
49349                              0.0            1.0                0
33801                              0.0            1.0                0
31780                              0.0            1.0                0
90175                              0.0            1.0                0

       InvestmentFromFriendsCount InvestmentFromFriendsAmount
Investors
18639                           0                         0.0
1
50370                           0                         0.0
23
110175                          0                         0.0
114
40179                           0                         0.0
8
112082                          0                         0.0
237
```

```
24027                               0                          0.0
29
49349                               0                          0.0
44
33801                               0                          0.0
261
31780                               0                          0.0
32
90175                               0                          0.0
1

[10 rows x 81 columns]
```

## Dataset Structure

```
df.shape ## showing the shape of the dataset

(113937, 81)
```

This dataset has 113,937 rows and 81 columns. Which is a relatively big dataset.

## Dataset Assessment and Cleaning

### Duplicated Records

```
df.duplicated(subset='LoanKey').sum()

np.int64(871)
```

Let's identify the duplicated records based on the LoanKey and see if wwe should handle this

```
duplicates = df[df.duplicated(subset='LoanKey', keep=False)]
display(duplicates.head(10))

                  ListingKey  ListingNumber
ListingCreationDate  \
8    0F043596202561788EA13D5        1023355  2013-12-02
10:43:39.117000000
9    0F043596202561788EA13D5        1023355  2013-12-02
10:43:39.117000000
29   0F563597161095613517437        1051243  2013-12-17
09:18:33.220000000
176  106335993636414276CB477        1119836  2014-01-08
14:27:50.320000000
313  09233589620788733CFB8CE         930842  2013-09-25
08:03:11.860000000
349  313635901230654318A9030         931467  2013-09-26
18:50:29.053000000
442  09AD35918712001025AC1BD         969821  2013-10-24
```

```
13:21:31.607000000
444  09CD3592594126374FB0A7C         986199  2013-10-18
08:28:03.610000000
455  31C73597152310464749E00        1092437  2013-12-23
13:47:35.500000000
461  44F2358557406858060EBDE         870200  2013-08-15
07:12:49.410000000

    CreditGrade  Term LoanStatus ClosedDate  BorrowerAPR  BorrowerRate  
\
8           NaN    36    Current        NaN      0.07620        0.0629   

9           NaN    36    Current        NaN      0.07620        0.0629   

29          NaN    36    Current        NaN      0.15223        0.1239   

176         NaN    36    Current        NaN      0.32446        0.2850   

313         NaN    36    Current        NaN      0.19144        0.1550   

349         NaN    36    Current        NaN      0.17090        0.1349   

442         NaN    36    Current        NaN      0.20524        0.1685   

444         NaN    36    Current        NaN      0.22773        0.1905   

455         NaN    36    Current        NaN      0.17151        0.1355   

461         NaN    60    Current        NaN      0.18965        0.1660   


     LenderYield  ...  LP_ServiceFees  LP_CollectionFees  \
8         0.0529  ...          -16.77                0.0   
9         0.0529  ...          -16.77                0.0   
29        0.1139  ...          -29.73                0.0   
176       0.2750  ...           -3.40                0.0   
313       0.1450  ...          -36.97                0.0   
349       0.1249  ...          -15.40                0.0   
442       0.1585  ...           -8.41                0.0   
444       0.1805  ...          -42.03                0.0   
455       0.1255  ...           -6.40                0.0   
461       0.1560  ...          -49.69                0.0   

     LP_GrossPrincipalLoss  LP_NetPrincipalLoss  \
8                      0.0                  0.0   
9                      0.0                  0.0   
29                     0.0                  0.0   
176                    0.0                  0.0   
313                    0.0                  0.0   
349                    0.0                  0.0   
```

```
442                    0.0                    0.0
444                    0.0                    0.0
455                    0.0                    0.0
461                    0.0                    0.0

    LP_NonPrincipalRecoverypayments  PercentFunded  Recommendations  \
8                               0.0            1.0                0
9                               0.0            1.0                0
29                              0.0            1.0                0
176                             0.0            1.0                0
313                             0.0            1.0                0
349                             0.0            1.0                0
442                             0.0            1.0                0
444                             0.0            1.0                0
455                             0.0            1.0                0
461                             0.0            1.0                0

    InvestmentFromFriendsCount InvestmentFromFriendsAmount Investors
8                            0                         0.0         1
9                            0                         0.0         1
29                           0                         0.0         1
176                          0                         0.0         5
313                          0                         0.0       169
349                          0                         0.0         1
442                          0                         0.0         1
444                          0                         0.0         1
455                          0                         0.0         1
461                          0                         0.0         1

[10 rows x 81 columns]
```

```
df.drop_duplicates(subset='LoanKey', keep='first', inplace=True) ##
Dropping the duplicated records and keeping only the first record

print(df.duplicated(subset='LoanKey').sum()) ## Checking the drop of
duplicted
df.shape
```

```
0

(113066, 81)
```

Let's check for the duplicated records based on the `ListingKey` based on teh documentation it has to be unique too.

```
df.duplicated(subset='ListingKey').sum() ## checking for duliated
based on the listing key
```

```
np.int64(0)
```

Data types Validity

- Assessment: Let's look at the data types of these variables and assess them using `.info`:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 113066 entries, 0 to 113936
Data columns (total 81 columns):
 #   Column                               Non-Null Count   Dtype
---  ------                               --------------   -----
 0   ListingKey                           113066 non-null  object
 1   ListingNumber                        113066 non-null  int64
 2   ListingCreationDate                  113066 non-null  object
 3   CreditGrade                          28953 non-null   object
 4   Term                                 113066 non-null  int64
 5   LoanStatus                           113066 non-null  object
 6   ClosedDate                           55076 non-null   object
 7   BorrowerAPR                          113041 non-null  float64
 8   BorrowerRate                         113066 non-null  float64
 9   LenderYield                          113066 non-null  float64
 10  EstimatedEffectiveYield              83982 non-null   float64
 11  EstimatedLoss                        83982 non-null   float64
 12  EstimatedReturn                      83982 non-null   float64
 13  ProsperRating (numeric)              83982 non-null   float64
 14  ProsperRating (Alpha)                83982 non-null   object
 15  ProsperScore                         83982 non-null   float64
 16  ListingCategory (numeric)            113066 non-null  int64
 17  BorrowerState                        107551 non-null  object
 18  Occupation                           109537 non-null  object
 19  EmploymentStatus                     110811 non-null  object
 20  EmploymentStatusDuration             105441 non-null  float64
 21  IsBorrowerHomeowner                  113066 non-null  bool
 22  CurrentlyInGroup                     113066 non-null  bool
 23  GroupKey                             13339 non-null   object
 24  DateCreditPulled                     113066 non-null  object
 25  CreditScoreRangeLower                112475 non-null  float64
 26  CreditScoreRangeUpper                112475 non-null  float64
 27  FirstRecordedCreditLine              112369 non-null  object
 28  CurrentCreditLines                   105462 non-null  float64
 29  OpenCreditLines                      105462 non-null  float64
 30  TotalCreditLinespast7years           112369 non-null  float64
 31  OpenRevolvingAccounts                113066 non-null  int64
 32  OpenRevolvingMonthlyPayment          113066 non-null  float64
 33  InquiriesLast6Months                 112369 non-null  float64
 34  TotalInquiries                       111907 non-null  float64
 35  CurrentDelinquencies                 112369 non-null  float64
 36  AmountDelinquent                     105444 non-null  float64
 37  DelinquenciesLast7Years              112076 non-null  float64
 38  PublicRecordsLast10Years             112369 non-null  float64
 39  PublicRecordsLast12Months            105462 non-null  float64
```

```
 40  RevolvingCreditBalance                 105462 non-null  float64
 41  BankcardUtilization                    105462 non-null  float64
 42  AvailableBankcardCredit                105522 non-null  float64
 43  TotalTrades                            105522 non-null  float64
 44  TradesNeverDelinquent (percentage)     105522 non-null  float64
 45  TradesOpenedLast6Months                105522 non-null  float64
 46  DebtToIncomeRatio                      104594 non-null  float64
 47  IncomeRange                            113066 non-null  object
 48  IncomeVerifiable                       113066 non-null  bool
 49  StatedMonthlyIncome                    113066 non-null  float64
 50  LoanKey                                113066 non-null  object
 51  TotalProsperLoans                      21923 non-null   float64
 52  TotalProsperPaymentsBilled             21923 non-null   float64
 53  OnTimeProsperPayments                  21923 non-null   float64
 54  ProsperPaymentsLessThanOneMonthLate    21923 non-null   float64
 55  ProsperPaymentsOneMonthPlusLate        21923 non-null   float64
 56  ProsperPrincipalBorrowed               21923 non-null   float64
 57  ProsperPrincipalOutstanding            21923 non-null   float64
 58  ScorexChangeAtTimeOfListing            18912 non-null   float64
 59  LoanCurrentDaysDelinquent              113066 non-null  int64
 60  LoanFirstDefaultedCycleNumber          16952 non-null   float64
 61  LoanMonthsSinceOrigination             113066 non-null  int64
 62  LoanNumber                             113066 non-null  int64
 63  LoanOriginalAmount                     113066 non-null  int64
 64  LoanOriginationDate                    113066 non-null  object
 65  LoanOriginationQuarter                 113066 non-null  object
 66  MemberKey                              113066 non-null  object
 67  MonthlyLoanPayment                     113066 non-null  float64
 68  LP_CustomerPayments                    113066 non-null  float64
 69  LP_CustomerPrincipalPayments           113066 non-null  float64
 70  LP_InterestandFees                     113066 non-null  float64
 71  LP_ServiceFees                         113066 non-null  float64
 72  LP_CollectionFees                      113066 non-null  float64
 73  LP_GrossPrincipalLoss                  113066 non-null  float64
 74  LP_NetPrincipalLoss                    113066 non-null  float64
 75  LP_NonPrincipalRecoverypayments        113066 non-null  float64
 76  PercentFunded                          113066 non-null  float64
 77  Recommendations                        113066 non-null  int64
 78  InvestmentFromFriendsCount             113066 non-null  int64
 79  InvestmentFromFriendsAmount            113066 non-null  float64
 80  Investors                              113066 non-null  int64
dtypes: bool(3), float64(50), int64(11), object(17)
memory usage: 68.5+ MB
```

We have the columns of `ClosedDate`, `LoanOriginationDate`, `DateCreditPulled`, and the `ListingCreationDate` has an object type and it has to be a datetime type.

```
## listing the date columns
date_columns = ['ClosedDate', 'LoanOriginationDate',
'DateCreditPulled', 'ListingCreationDate']

### Loopong through the list and coverting to datetime data type
for col in date_columns:
    df[col] = pd.to_datetime(df[col], format='mixed') ## using the
format as mixed to infer the format for each element individually

df.info() ### check if that is successful
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 113066 entries, 0 to 113936
Data columns (total 81 columns):
 #   Column                              Non-Null Count   Dtype

---  ------                              --------------   -----

 0   ListingKey                          113066 non-null  object

 1   ListingNumber                       113066 non-null  int64

 2   ListingCreationDate                 113066 non-null
datetime64[ns]
 3   CreditGrade                         28953 non-null   object

 4   Term                                113066 non-null  int64

 5   LoanStatus                          113066 non-null  object

 6   ClosedDate                          55076 non-null
datetime64[ns]
 7   BorrowerAPR                         113041 non-null  float64

 8   BorrowerRate                        113066 non-null  float64

 9   LenderYield                         113066 non-null  float64

 10  EstimatedEffectiveYield             83982 non-null   float64

 11  EstimatedLoss                       83982 non-null   float64

 12  EstimatedReturn                     83982 non-null   float64

 13  ProsperRating (numeric)             83982 non-null   float64

 14  ProsperRating (Alpha)               83982 non-null   object

 15  ProsperScore                        83982 non-null   float64

 16  ListingCategory (numeric)           113066 non-null  int64
```

```
 17  BorrowerState                          107551 non-null  object
 18  Occupation                             109537 non-null  object
 19  EmploymentStatus                       110811 non-null  object
 20  EmploymentStatusDuration               105441 non-null  float64
 21  IsBorrowerHomeowner                    113066 non-null  bool
 22  CurrentlyInGroup                       113066 non-null  bool
 23  GroupKey                               13339 non-null   object
 24  DateCreditPulled                       113066 non-null
datetime64[ns]
 25  CreditScoreRangeLower                  112475 non-null  float64
 26  CreditScoreRangeUpper                  112475 non-null  float64
 27  FirstRecordedCreditLine                112369 non-null  object
 28  CurrentCreditLines                     105462 non-null  float64
 29  OpenCreditLines                        105462 non-null  float64
 30  TotalCreditLinespast7years             112369 non-null  float64
 31  OpenRevolvingAccounts                  113066 non-null  int64
 32  OpenRevolvingMonthlyPayment            113066 non-null  float64
 33  InquiriesLast6Months                   112369 non-null  float64
 34  TotalInquiries                         111907 non-null  float64
 35  CurrentDelinquencies                   112369 non-null  float64
 36  AmountDelinquent                       105444 non-null  float64
 37  DelinquenciesLast7Years                112076 non-null  float64
 38  PublicRecordsLast10Years               112369 non-null  float64
 39  PublicRecordsLast12Months              105462 non-null  float64
 40  RevolvingCreditBalance                 105462 non-null  float64
 41  BankcardUtilization                    105462 non-null  float64
 42  AvailableBankcardCredit                105522 non-null  float64
```

```
 43  TotalTrades                          105522 non-null  float64
 44  TradesNeverDelinquent (percentage)   105522 non-null  float64
 45  TradesOpenedLast6Months              105522 non-null  float64
 46  DebtToIncomeRatio                    104594 non-null  float64
 47  IncomeRange                          113066 non-null  object
 48  IncomeVerifiable                     113066 non-null  bool
 49  StatedMonthlyIncome                  113066 non-null  float64
 50  LoanKey                              113066 non-null  object
 51  TotalProsperLoans                    21923 non-null   float64
 52  TotalProsperPaymentsBilled           21923 non-null   float64
 53  OnTimeProsperPayments                21923 non-null   float64
 54  ProsperPaymentsLessThanOneMonthLate  21923 non-null   float64
 55  ProsperPaymentsOneMonthPlusLate      21923 non-null   float64
 56  ProsperPrincipalBorrowed             21923 non-null   float64
 57  ProsperPrincipalOutstanding          21923 non-null   float64
 58  ScorexChangeAtTimeOfListing          18912 non-null   float64
 59  LoanCurrentDaysDelinquent            113066 non-null  int64
 60  LoanFirstDefaultedCycleNumber        16952 non-null   float64
 61  LoanMonthsSinceOrigination           113066 non-null  int64
 62  LoanNumber                           113066 non-null  int64
 63  LoanOriginalAmount                   113066 non-null  int64
 64  LoanOriginationDate                  113066 non-null
datetime64[ns]
 65  LoanOriginationQuarter               113066 non-null  object
 66  MemberKey                            113066 non-null  object
 67  MonthlyLoanPayment                   113066 non-null  float64
 68  LP_CustomerPayments                  113066 non-null  float64
```

```
 69   LP_CustomerPrincipalPayments           113066 non-null   float64
 70   LP_InterestandFees                     113066 non-null   float64
 71   LP_ServiceFees                         113066 non-null   float64
 72   LP_CollectionFees                      113066 non-null   float64
 73   LP_GrossPrincipalLoss                  113066 non-null   float64
 74   LP_NetPrincipalLoss                    113066 non-null   float64
 75   LP_NonPrincipalRecoverypayments        113066 non-null   float64
 76   PercentFunded                          113066 non-null   float64
 77   Recommendations                        113066 non-null   int64
 78   InvestmentFromFriendsCount             113066 non-null   int64
 79   InvestmentFromFriendsAmount            113066 non-null   float64
 80   Investors                              113066 non-null   int64
dtypes: bool(3), datetime64[ns](4), float64(50), int64(11), object(13)
memory usage: 68.5+ MB
```

Now the rest of the data types of the variables are valid.

## Data Completness

```
def get_percent_null(df):
    """
    Args:
        - df (pd.Dataframe)
    Returns:
        - The percentages of missing values
    """
    null_counts = df.isnull().sum()
    null_counts = null_counts[null_counts > 0].sort_values()
    return (null_counts/df.shape[0])*100

get_percent_null(df)
```

```
BorrowerAPR                                0.022111
CreditScoreRangeLower                      0.522704
CreditScoreRangeUpper                      0.522704
PublicRecordsLast10Years                   0.616454
InquiriesLast6Months                       0.616454
CurrentDelinquencies                       0.616454
```

```
TotalCreditLinespast7years              0.616454
FirstRecordedCreditLine                 0.616454
DelinquenciesLast7Years                 0.875595
TotalInquiries                          1.025065
EmploymentStatus                        1.994410
Occupation                              3.121186
BorrowerState                           4.877682
TotalTrades                             6.672209
AvailableBankcardCredit                 6.672209
TradesNeverDelinquent (percentage)      6.672209
TradesOpenedLast6Months                 6.672209
RevolvingCreditBalance                  6.725276
CurrentCreditLines                      6.725276
PublicRecordsLast12Months               6.725276
BankcardUtilization                     6.725276
OpenCreditLines                         6.725276
AmountDelinquent                        6.741195
EmploymentStatusDuration                6.743849
DebtToIncomeRatio                       7.492969
ProsperRating (Alpha)                  25.723029
ProsperRating (numeric)                25.723029
ProsperScore                           25.723029
EstimatedLoss                          25.723029
EstimatedEffectiveYield                25.723029
EstimatedReturn                        25.723029
ClosedDate                             51.288628
CreditGrade                            74.392833
ProsperPrincipalOutstanding            80.610440
TotalProsperLoans                      80.610440
TotalProsperPaymentsBilled             80.610440
OnTimeProsperPayments                  80.610440
ProsperPaymentsLessThanOneMonthLate    80.610440
ProsperPaymentsOneMonthPlusLate        80.610440
ProsperPrincipalBorrowed               80.610440
ScorexChangeAtTimeOfListing            83.273486
LoanFirstDefaultedCycleNumber          85.006987
GroupKey                               88.202466
dtype: float64
```

There are many columns that has null values, some of which have a very high count of null values. The following columns were the highest percentage of null such as higher than 50%. These columns are droppped.

```
high_missing_percent = ['CreditGrade', 'ScorexChangeAtTimeOfListing',
'LoanFirstDefaultedCycleNumber', 'ClosedDate', 'TotalProsperLoans',
 'TotalProsperPaymentsBilled', 'OnTimeProsperPayments',
'ProsperPaymentsLessThanOneMonthLate',
'ProsperPaymentsOneMonthPlusLate',
```

```
 'ProsperPrincipalBorrowed', 'ProsperPrincipalOutstanding'] ## list of columns with missing values higher than 50%

df.drop(columns= high_missing_percent, inplace=True) ## Drop the list of high percentage columns

df.shape ## Check if the dropping was successful
```

```
(113066, 70)
```

The drop is checked now that the variables went from 81 to 70.

Handling the other null values by dropping NA values and rows:

```
get_percent_null(df)
```

```
BorrowerAPR                             0.022111
CreditScoreRangeUpper                   0.522704
CreditScoreRangeLower                   0.522704
FirstRecordedCreditLine                 0.616454
CurrentDelinquencies                    0.616454
PublicRecordsLast10Years                0.616454
InquiriesLast6Months                    0.616454
TotalCreditLinespast7years              0.616454
DelinquenciesLast7Years                 0.875595
TotalInquiries                          1.025065
EmploymentStatus                        1.994410
Occupation                              3.121186
BorrowerState                           4.877682
TradesOpenedLast6Months                 6.672209
AvailableBankcardCredit                 6.672209
TradesNeverDelinquent (percentage)      6.672209
TotalTrades                             6.672209
CurrentCreditLines                      6.725276
BankcardUtilization                     6.725276
OpenCreditLines                         6.725276
PublicRecordsLast12Months               6.725276
RevolvingCreditBalance                  6.725276
AmountDelinquent                        6.741195
EmploymentStatusDuration                6.743849
DebtToIncomeRatio                       7.492969
EstimatedReturn                        25.723029
EstimatedLoss                          25.723029
ProsperRating (numeric)                25.723029
ProsperRating (Alpha)                  25.723029
ProsperScore                           25.723029
EstimatedEffectiveYield                25.723029
GroupKey                               88.202466
dtype: float64
```

These columns are identifiers, mostly unique between each loan.They are irrelevant for the task and therefore they should be dropped too.

```
df.columns

Index(['ListingKey', 'ListingNumber', 'ListingCreationDate', 'Term',
       'LoanStatus', 'BorrowerAPR', 'BorrowerRate', 'LenderYield',
       'EstimatedEffectiveYield', 'EstimatedLoss', 'EstimatedReturn',
       'ProsperRating (numeric)', 'ProsperRating (Alpha)',
'ProsperScore',
       'ListingCategory (numeric)', 'BorrowerState', 'Occupation',
       'EmploymentStatus', 'EmploymentStatusDuration',
'IsBorrowerHomeowner',
       'CurrentlyInGroup', 'GroupKey', 'DateCreditPulled',
       'CreditScoreRangeLower', 'CreditScoreRangeUpper',
       'FirstRecordedCreditLine', 'CurrentCreditLines',
'OpenCreditLines',
       'TotalCreditLinespast7years', 'OpenRevolvingAccounts',
       'OpenRevolvingMonthlyPayment', 'InquiriesLast6Months',
'TotalInquiries',
       'CurrentDelinquencies', 'AmountDelinquent',
'DelinquenciesLast7Years',
       'PublicRecordsLast10Years', 'PublicRecordsLast12Months',
       'RevolvingCreditBalance', 'BankcardUtilization',
       'AvailableBankcardCredit', 'TotalTrades',
       'TradesNeverDelinquent (percentage)',
'TradesOpenedLast6Months',
       'DebtToIncomeRatio', 'IncomeRange', 'IncomeVerifiable',
       'StatedMonthlyIncome', 'LoanKey', 'LoanCurrentDaysDelinquent',
       'LoanMonthsSinceOrigination', 'LoanNumber',
'LoanOriginalAmount',
       'LoanOriginationDate', 'LoanOriginationQuarter', 'MemberKey',
       'MonthlyLoanPayment', 'LP_CustomerPayments',
       'LP_CustomerPrincipalPayments', 'LP_InterestandFees',
'LP_ServiceFees',
       'LP_CollectionFees', 'LP_GrossPrincipalLoss',
'LP_NetPrincipalLoss',
       'LP_NonPrincipalRecoverypayments', 'PercentFunded',
'Recommendations',
       'InvestmentFromFriendsCount', 'InvestmentFromFriendsAmount',
       'Investors'],
      dtype='object')

## List of identifiers and other irrelevant columns
identifiers = ["ListingKey", "ListingNumber", "GroupKey", "LoanKey",
"LoanNumber", "MemberKey", "DateCreditPulled"]
df.drop(columns= identifiers, axis= 1, inplace= True ) ## Dropping
irrelevant columns

df.shape ##
```

```
(113066, 63)
```

The drop is done, since the features count went down to 63.

```
get_percent_null(df)
```

```
BorrowerAPR                             0.022111
CreditScoreRangeLower                   0.522704
CreditScoreRangeUpper                   0.522704
FirstRecordedCreditLine                 0.616454
PublicRecordsLast10Years                0.616454
CurrentDelinquencies                    0.616454
InquiriesLast6Months                    0.616454
TotalCreditLinespast7years              0.616454
DelinquenciesLast7Years                 0.875595
TotalInquiries                          1.025065
EmploymentStatus                        1.994410
Occupation                              3.121186
BorrowerState                           4.877682
AvailableBankcardCredit                 6.672209
TotalTrades                             6.672209
TradesOpenedLast6Months                 6.672209
TradesNeverDelinquent (percentage)      6.672209
CurrentCreditLines                      6.725276
RevolvingCreditBalance                  6.725276
PublicRecordsLast12Months               6.725276
BankcardUtilization                     6.725276
OpenCreditLines                         6.725276
AmountDelinquent                        6.741195
EmploymentStatusDuration                6.743849
DebtToIncomeRatio                       7.492969
ProsperRating (numeric)                25.723029
EstimatedReturn                        25.723029
EstimatedEffectiveYield                25.723029
ProsperRating (Alpha)                  25.723029
ProsperScore                           25.723029
EstimatedLoss                          25.723029
dtype: float64
```

Let's handle the lower percentages by only dropping the NA values instead of the whole columns.

```
df.dropna(inplace=True)

get_percent_null(df)
```

```
Series([], dtype: float64)
```

No missing NA values left.

```
df.shape
```

```
(75486, 63)
```

After handling missing values and dropping unneccary columns the shape of the data is 76,216 rows and 63 columns

We want to create a simplified version of this columns, since there are multiple values for the Due Past values.

```
df['LoanStatus'].unique()
```

```
array(['Current', 'Past Due (1-15 days)', 'Defaulted', 'Completed',
       'Chargedoff', 'Past Due (16-30 days)', 'Past Due (61-90 days)',
       'Past Due (31-60 days)', 'Past Due (91-120 days)',
       'FinalPaymentInProgress', 'Past Due (>120 days)'],
dtype=object)
```

```
df['SimplifiedLoanStatus'] = df['LoanStatus'].apply(lambda x: 'Past
Due' if 'Past Due' in x else x)
df.head(10)
```

```
       ListingCreationDate  Term            LoanStatus  BorrowerAPR  \
1  2014-02-27 08:28:07.900    36               Current      0.12016
3  2012-10-22 11:02:35.010    36               Current      0.12528
4  2013-09-14 18:38:39.097    36               Current      0.24614
5  2013-12-14 08:26:37.093    60               Current      0.15425
6  2013-04-12 09:52:56.147    36               Current      0.31032
7  2013-05-05 06:49:27.493    36               Current      0.23939
8  2013-12-02 10:43:39.117    36               Current      0.07620
10 2012-05-10 07:04:01.577    60               Current      0.27462
12 2013-12-15 20:01:10.757    36  Past Due (1-15 days)      0.17969
13 2013-07-15 16:28:28.087    36               Current      0.13138

    BorrowerRate  LenderYield  EstimatedEffectiveYield  EstimatedLoss
\
1         0.0920       0.0820                  0.07960         0.0249

3         0.0974       0.0874                  0.08490         0.0249

4         0.2085       0.1985                  0.18316         0.0925

5         0.1314       0.1214                  0.11567         0.0449

6         0.2712       0.2612                  0.23820         0.1275

7         0.2019       0.1919                  0.17830         0.0799

8         0.0629       0.0529                  0.05221         0.0099

10        0.2489       0.2389                  0.23320         0.0890
```

```
12        0.1435       0.1335                  0.12640        0.0524

13        0.1034       0.0934                  0.09050        0.0274


    EstimatedReturn  ProsperRating (numeric)  ... LP_CollectionFees  \
1           0.05470                      6.0  ...               0.0
3           0.06000                      6.0  ...               0.0
4           0.09066                      3.0  ...               0.0
5           0.07077                      5.0  ...               0.0
6           0.11070                      2.0  ...               0.0
7           0.09840                      4.0  ...               0.0
8           0.04231                      7.0  ...               0.0
10          0.14420                      4.0  ...               0.0
12          0.07400                      5.0  ...               0.0
13          0.06310                      6.0  ...               0.0

    LP_GrossPrincipalLoss  LP_NetPrincipalLoss  \
1                     0.0                  0.0
3                     0.0                  0.0
4                     0.0                  0.0
5                     0.0                  0.0
6                     0.0                  0.0
7                     0.0                  0.0
8                     0.0                  0.0
10                    0.0                  0.0
12                    0.0                  0.0
13                    0.0                  0.0

   LP_NonPrincipalRecoverypayments PercentFunded Recommendations  \
1                              0.0           1.0               0
3                              0.0           1.0               0
4                              0.0           1.0               0
5                              0.0           1.0               0
6                              0.0           1.0               0
7                              0.0           1.0               0
8                              0.0           1.0               0
10                             0.0           1.0               0
12                             0.0           1.0               0
13                             0.0           1.0               0

    InvestmentFromFriendsCount  InvestmentFromFriendsAmount  Investors
\
1                            0                          0.0          1

3                            0                          0.0        158

4                            0                          0.0         20
```

```
5                             0                          0.0           1

6                             0                          0.0           1

7                             0                          0.0           1

8                             0                          0.0           1

10                            0                          0.0          19

12                            0                          0.0           1

13                            0                          0.0         171


    SimplifiedLoanStatus
1                Current
3                Current
4                Current
5                Current
6                Current
7                Current
8                Current
10               Current
12              Past Due
13               Current

[10 rows x 64 columns]
```

## Main Features of interest

- All the left columns are the features of intrest they will be devided based on the objective into three subsets.
- Objective 1: Loan Performance

```
loan_performance_columns = [ 'LoanStatus','SimplifiedLoanStatus' ,
'LoanCurrentDaysDelinquent','LoanMonthsSinceOrigination',
    'LoanOriginalAmount', 'BorrowerAPR', 'DebtToIncomeRatio',
'CreditScoreRangeLower','CreditScoreRangeUpper',
    'ProsperScore', 'EmploymentStatus','IsBorrowerHomeowner']

loan_performance_df = df[loan_performance_columns]
print(loan_performance_df.shape)
loan_performance_df.head()
```

```
(75486, 12)

  LoanStatus SimplifiedLoanStatus  LoanCurrentDaysDelinquent  \
1    Current              Current                          0
3    Current              Current                          0
4    Current              Current                          0
```

```
5    Current              Current                          0
6    Current              Current                          0

   LoanMonthsSinceOrigination  LoanOriginalAmount  BorrowerAPR  \
1                           0               10000      0.12016
3                          16               10000      0.12528
4                           6               15000      0.24614
5                           3               15000      0.15425
6                          11                3000      0.31032

   DebtToIncomeRatio  CreditScoreRangeLower  CreditScoreRangeUpper  \
1               0.18                  680.0                  699.0
3               0.15                  800.0                  819.0
4               0.26                  680.0                  699.0
5               0.36                  740.0                  759.0
6               0.27                  680.0                  699.0

   ProsperScore EmploymentStatus  IsBorrowerHomeowner
1           7.0         Employed                False
3           9.0         Employed                 True
4           4.0         Employed                 True
5          10.0         Employed                 True
6           2.0         Employed                False
```

- Objective 2: Credit Score and Borrower Analysis

```
credit_borrower_columns = [ 'CreditScoreRangeLower',
'CreditScoreRangeUpper', 'ProsperRating (numeric)',
    'ProsperRating
(Alpha)','ProsperScore','IncomeRange','EmploymentStatus','IsBorrowerHo
meowner',
    'BorrowerAPR', 'LoanOriginalAmount']

credit_borrower_df = df[credit_borrower_columns]
print(credit_borrower_df.shape)
credit_borrower_df.head()

(75486, 10)

   CreditScoreRangeLower  CreditScoreRangeUpper  ProsperRating
(numeric)  \
1                  680.0                  699.0
6.0
3                  800.0                  819.0
6.0
4                  680.0                  699.0
3.0
5                  740.0                  759.0
5.0
6                  680.0                  699.0
```

```
2.0

  ProsperRating (Alpha)  ProsperScore     IncomeRange EmploymentStatus  \
1                     A           7.0  $50,000-74,999         Employed

3                     A           9.0  $25,000-49,999         Employed

4                     D           4.0        $100,000+        Employed

5                     B          10.0        $100,000+        Employed

6                     E           2.0  $25,000-49,999         Employed


   IsBorrowerHomeowner  BorrowerAPR  LoanOriginalAmount
1                False      0.12016               10000
3                 True      0.12528               10000
4                 True      0.24614               15000
5                 True      0.15425               15000
6                False      0.31032                3000
```

- Objective 3: Geographic and Demographic Analysis

```
geo_demo_columns = [ 'BorrowerState', 'Occupation',
'EmploymentStatus', 'IncomeRange',
    'IsBorrowerHomeowner', 'LoanOriginalAmount', 'ProsperRating
(Alpha)', 'CreditScoreRangeLower',
    'CreditScoreRangeUpper']

geo_demo_df = df[geo_demo_columns]
print(geo_demo_df.shape)
geo_demo_df.head()

(75486, 9)

  BorrowerState      Occupation EmploymentStatus     IncomeRange  \
1            CO    Professional         Employed  $50,000-74,999
3            GA   Skilled Labor         Employed  $25,000-49,999
4            MN       Executive         Employed       $100,000+
5            NM    Professional         Employed       $100,000+
6            KS  Sales - Retail         Employed  $25,000-49,999

   IsBorrowerHomeowner  LoanOriginalAmount ProsperRating (Alpha)  \
1                False               10000                     A
3                 True               10000                     A
4                 True               15000                     D
5                 True               15000                     B
6                False                3000                     E

   CreditScoreRangeLower  CreditScoreRangeUpper
```

```
1                     680.0                   699.0
3                     800.0                   819.0
4                     680.0                   699.0
5                     740.0                   759.0
6                     680.0                   699.0
```

**Store the cleaned dataset**

```
df.to_csv('./data/prosperLoanDataCleaned.csv') ## Load the csv into
pandas dataframe
```

# 4. Univariate Exploration

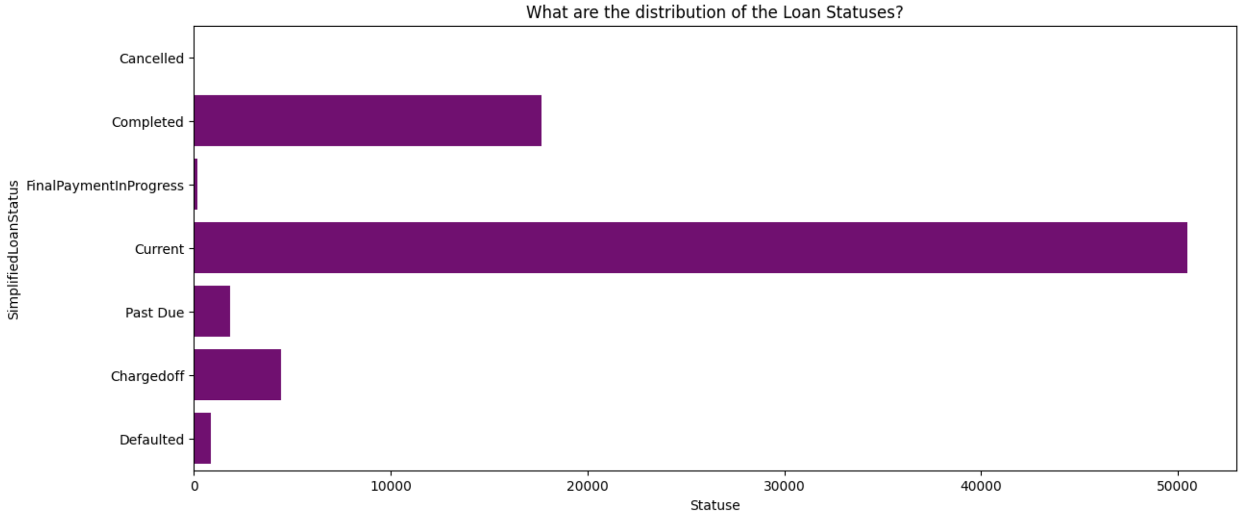
In this section, we are investigating distributions of individual variables. To see unusual points or outliers, take a deeper look to clean things up and prepare yourself to look at relationships between variables. the "Question-Visualization-Observations" framework is used throughout the exploration, it involves asking a question from the data, creating a visualization to find answers, and then recording observations after each visualisation.

**1. Loan Status**

**Question**: What are the distribution of the Loan Statuses?

```
## add orderring to the loan status and include the cancelled ones
loan_status_order = ['Cancelled', 'Completed',
'FinalPaymentInProgress', 'Current',
                     'Past Due', 'Chargedoff', 'Defaulted']

plt.figure(figsize=(14, 6))
sns.countplot(data = loan_performance_df, y = 'SimplifiedLoanStatus',
color = 'purple', order=loan_status_order)
plt.xlabel('Statuse')
plt.title('What are the distribution of the Loan Statuses?');
```




It seems that the majority status among the loan statuses is the `current` which is 50462, While the completed are success story they are `17675`. The chargedoff however are 4444 cases they are failed loans. While the `deafulted` are the cases in danger of `chargedoff` they reach 885. And the past due are slightly larger 1835 but still not in danger of charge off.

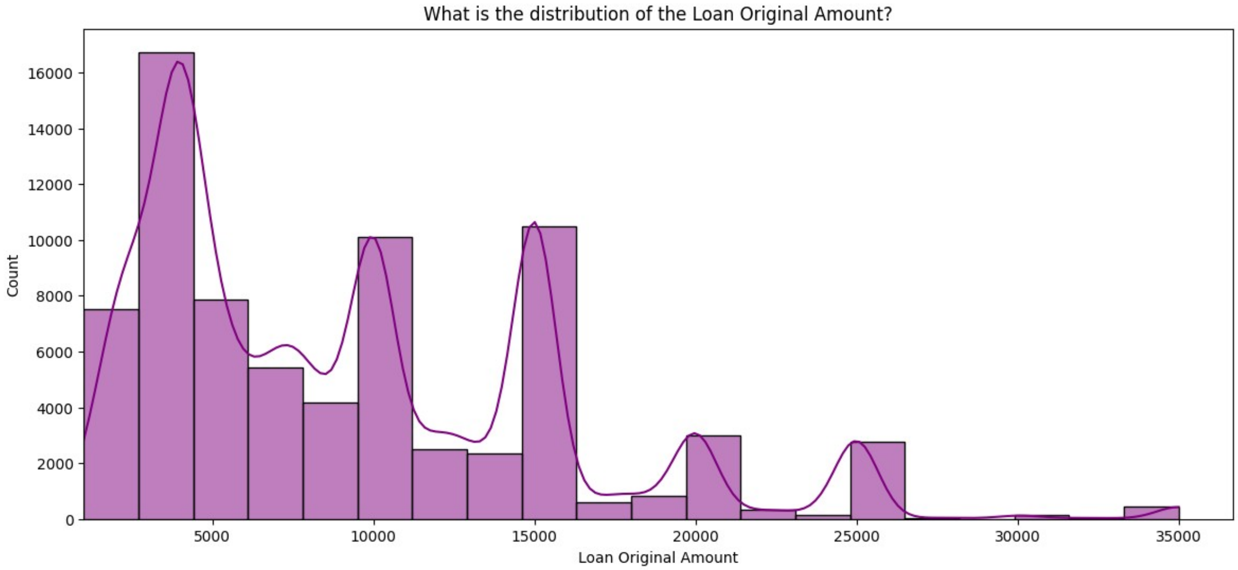
The chraged off and the deafulted represntts the actual risks for these loans.

Nothing unusual with this distribution.

### 2. The Loan Original Amount

Let's take a look into the Loan Original Amounts.

```
plt.figure(figsize=(14, 6))
sns.histplot(data= credit_borrower_df,  x = 'LoanOriginalAmount',
bins=20, kde= True, color = 'purple')
plt.title('What is the distribution of the Loan Original Amount?')
plt.ylabel('Count')
plt.xlim(1000)
plt.xlabel('Loan Original Amount');
```



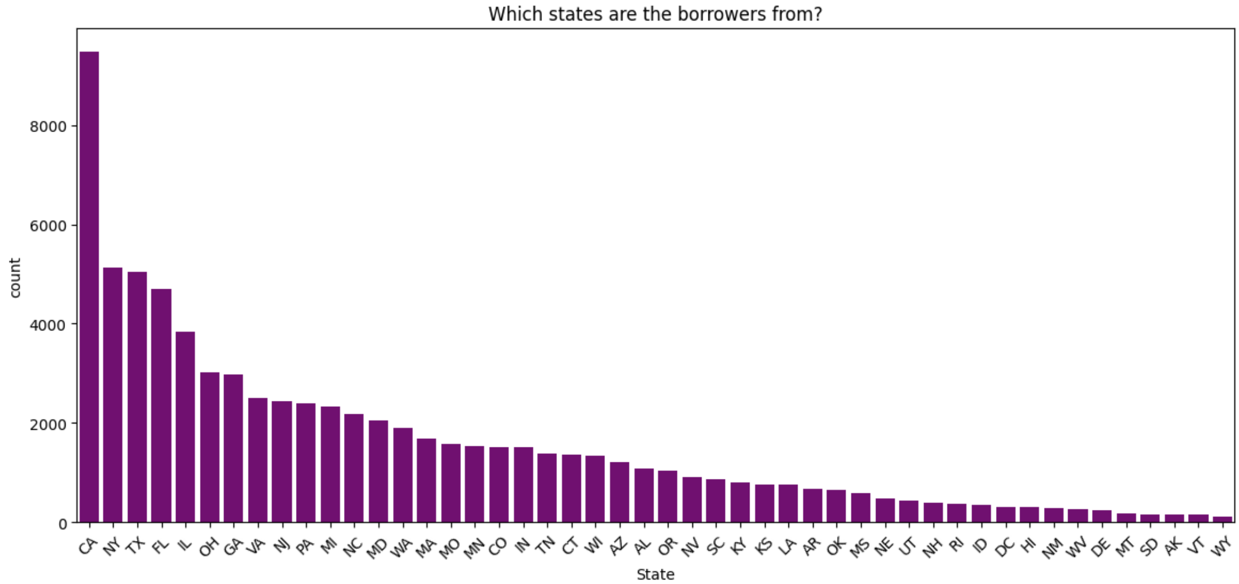

The distribution of the Loan Original Amount is right skewed, with multiple peaks suggesting distinct borrower groups. Most loans are relatively small, but there are outliers indicating larger, less frequent loan amounts. However these are numerical outliers but valid values for a given loan amount, so doesn't need to be handled.

### 3. Borrower State

Which states are the borrowers from?

```
borrower_states_counts = geo_demo_df['BorrowerState'].value_counts()

plt.figure(figsize=(14, 6))
ax = sns.barplot(borrower_states_counts, color = 'purple')
plt.title('Which states are the borrowers from?')
plt.xlabel('State')
plt.xticks(rotation = 45);
```




California boasts the highest number of borrowers among all US states, followed closely by Texas, New York, and Florida. On the other end of the spectrum, states like South Dakota, Alaska, Vermont, and Wyoming have significantly lower borrower counts.

California is an unpper bound outlier, however this is numerically and valid values that will not be handled.

# 5. Bivariate Exploration

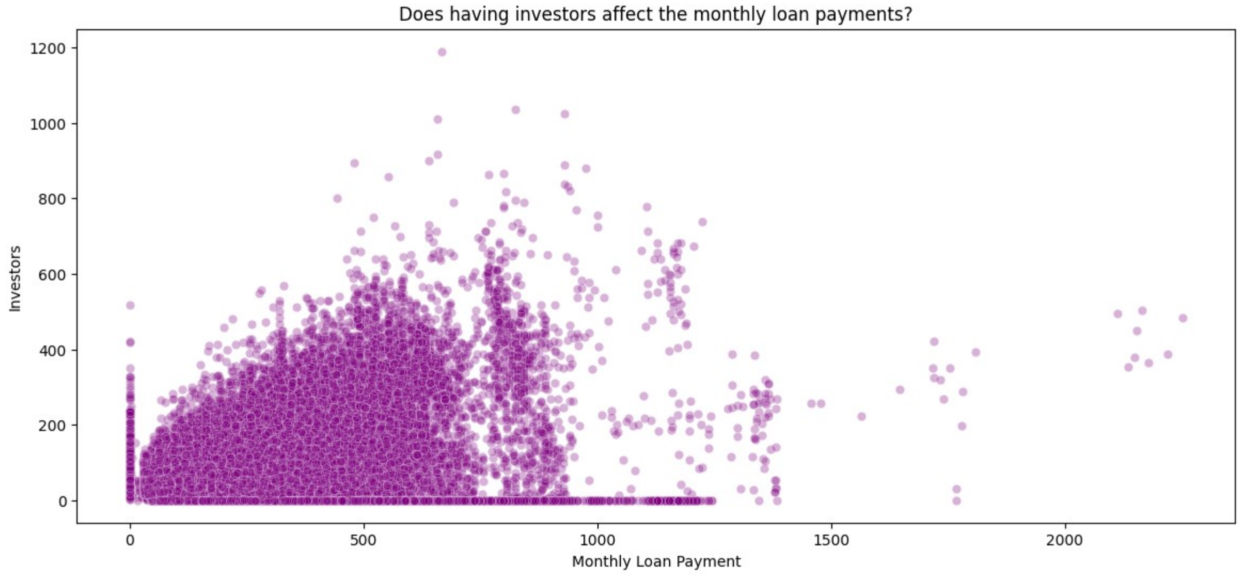
In this section, relationships between pairs of variables in the data are investigated. The variables of intreset that have been introduced in the previous sections. Questions are asked.

**Question:** Does having investors affect the monthly loan payments?

```
df['Investors'].corr(df['MonthlyLoanPayment'])
```

```
np.float64(0.307614290001082)
```

```
plt.figure(figsize=(14, 6))
sns.scatterplot(data= df, x='MonthlyLoanPayment', y='Investors',
alpha=0.3, color = 'purple')
plt.title('Does having investors affect the monthly loan payments?')
plt.xlabel('Monthly Loan Payment')
plt.ylabel('Investors');
```




There is a positive correlation between the number of investors and the monthly loan payments, with a correlation coefficient of 0.3. This indicates that as the number of investors increases, the monthly loan payments also tend to increase.

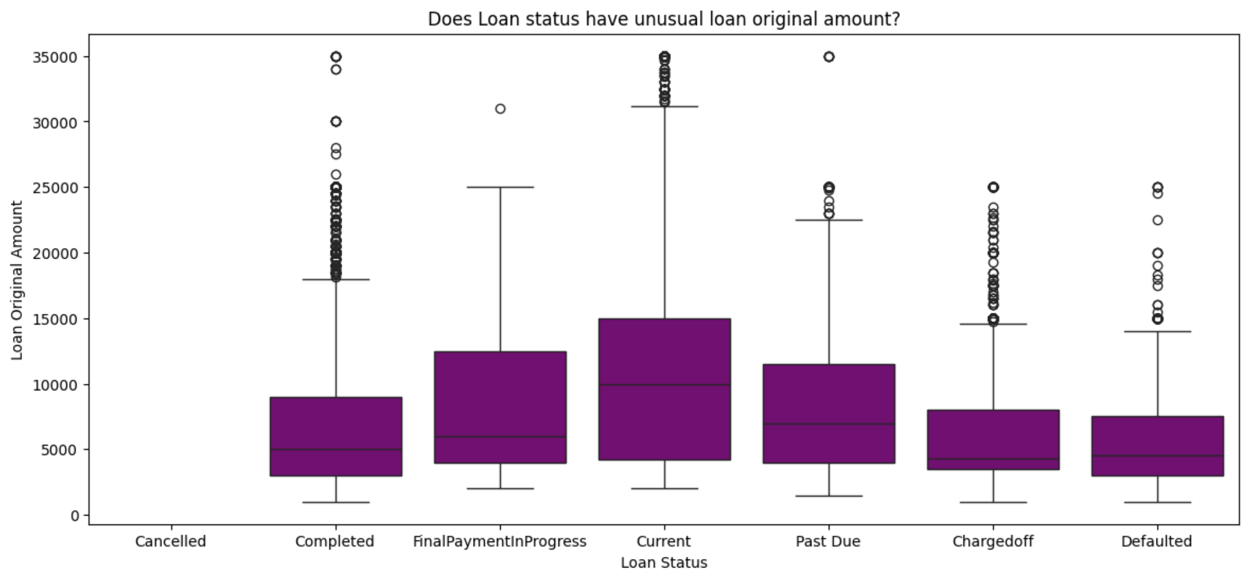
Some outliers are observed, with loans having over 600 investors and monthly payments exceeding 1000.

A few loans with very high monthly payments (above 2000) are also present, indicating significant loan amounts or terms.

The plot shows a dense cluster of points with fewer than 200 investors and monthly payments below 500, suggesting this is the most common scenario.

**Question:** Does Loan status have unusual loan original amount?

```
plt.figure(figsize=(14, 6))
sns.boxplot(data=loan_performance_df, x='SimplifiedLoanStatus',
y='LoanOriginalAmount', color='purple', order= loan_status_order)
plt.title('Does Loan status have unusual loan original amount?')
plt.xlabel('Loan Status')
plt.ylabel('Loan Original Amount');
```




All loan statuses have outliers, indicating some loans are significantly larger than most others in their respective categories.

Completed loans exhibit the most outliers, suggesting a wide variation in loan amounts within this category.

Charged-off loans also show a considerable number of outliers.

Current Loans, The highest upper outliers are observed in the current loan status, indicating some very large loans are currently active.

**Question:** How does the Original Amount of Loan change over time?

```
plt.figure(figsize=(14, 6))
sns.lineplot(data= df, x = 'LoanOriginationDate', y
='LoanOriginalAmount', color = 'purple')
plt.ylabel('Amount')
plt.xlabel('Year')
plt.xlim(pd.to_datetime('2009-07-20'), pd.to_datetime('2014-03-12'))
## the max and min dates of the Loan Origination Date
plt.title('How does the Original Loan Amount changes over time?');
```

There is a noticeable increase in loan amounts over the years, suggesting that borrowers have been taking out larger loans as time progresses.

Interestingly, the year 2009 stands out with some of the highest loan amounts, indicating that during this period, there were notably larger loans compared to other years.

**Question:** Does employment status have unuaual credit score lower range?

```
employment_status_oredr = ['Employed', 'Full-time', 'Part-time',
'Self-employed', 'Retired', 'Not employed', 'Other']
```

```
plt.figure(figsize=(14, 6))
sns.boxplot(data=credit_borrower_df, x='EmploymentStatus',
y='CreditScoreRangeLower', color='purple',
order=employment_status_oredr)
plt.title('Does employment status have unuaual Credit Score Range
Lower?')
plt.xlabel('Employment Status')
plt.ylabel('Credit Score Range Lower');
```

Employed borrowers have a wide range of credit scores, with some higher outliers. This indicates that while most employed borrowers have moderate to high credit scores, there are a few with exceptionally high scores.

Borrowers in the "Other" and "Full-time" categories have similar distributions of credit scores, generally higher than those of employed borrowers. These groups also include some high outliers, suggesting that a subset of these borrowers have excellent credit.

Self-employed borrowers tend to have lower credit scores compared to other groups, with a narrower range and fewer outliers.

Not employed borrowers have the lowest and least variable credit scores, indicating financial instability or limited credit history.

**Question:** What are the employment statuses of students who are taking loans?

```
## List of students occupations ordered by their natural order
study_occupations = ['Student - College Freshman',  'Student - College
Sophomore',  'Student - College Junior',
 'Student - Community College', 'Student - Technical School']

## Labels for the visualization
labels = ['College Freshman', 'College Sophomore', 'College Junior',
'Community College', 'Technical School']

## the natural order of employment staus
employment_status_hue_oredr = ['Employed', 'Full-time', 'Part-time',
'Not employed']

plt.figure(figsize=(14, 6))
sns.countplot(data=geo_demo_df[geo_demo_df['Occupation'].isin(study_oc
cupations)], x='Occupation', palette=['purple','blue', 'yellow',
'grey'],
              hue='EmploymentStatus',
hue_order=employment_status_hue_oredr,  order = study_occupations,
edgecolor = 'white')
plt.title('What are the employment statuses of students who are taking
loans?')
plt.xlabel('Studnets Occupation')
plt.ylabel('Count')
```

```
plt.xticks(ticks= study_occupations, labels= labels)
plt.legend(title='Employment Status');
```

Employment is widespread among students taking loans, with the majority falling into the "Employed" category across all educational levels.

Part-time work appears to be the most common arrangement, indicated by the higher count of "Part-time" compared to "Full-time" in most categories. Expected since they are students.

Community college students exhibit a notably higher proportion of full-time employment compared to other groups.

The "Technical School" category shows a lower overall employment rate and a higher percentage of students who are not employed.

**Question:** What are intresting relationships between variables in the dataset?

```
intrest_vars = ['LoanOriginalAmount', 'BorrowerAPR', 'BorrowerRate',
    'CreditScoreRangeLower', 'CreditScoreRangeUpper',
    'DebtToIncomeRatio', 'StatedMonthlyIncome']
subset = df[intrest_vars]
sns.heatmap(subset.corr(), annot=True)
plt.xticks(rotation = 45)
plt.title("Correlation Matrix");

grid = sns.PairGrid(data = df, vars = intrest_vars )
grid.map_diag(plt.hist, bins = 20,color='purple')
grid.map_offdiag(plt.scatter,color='purple')
grid.tight_layout();
```

Positive correlation between `borrowerAPR` and `BorrowerRate`.

`Debit Income Ratio` has a low positive correlation with bith the `credit scire range lower and the upper range`.

`LoanOriginalAmount` and `StatedMonthlyIncome` have a moderate positive correlation (0.3). This indicates that individuals with higher incomes tend to borrow larger amounts.

`BorrowerAPR` and `CreditScoreRangeLower` have a moderate negative correlation (-0.55). This suggests that borrowers with higher credit scores tend to have lower APRs.

There is no other intresting correlations between the variables.

# 6. Multivariate Exploration

In this section, plots of three or more variables are created to investigate the data even further.

**Question:** How does the original loan amount relate to the borrower's APR across different loan statuses?

```
## this list is the natural order of loan status
loan_status_order = [ 'Completed','FinalPaymentInProgress','Current',
'Past Due','Chargedoff','Defaulted']

## Scatterplot with multiple encodings, color encoding with ordered
categories
plt.figure(figsize=(14, 6))
sns.scatterplot(data=df, x='LoanOriginalAmount', y='BorrowerAPR',
hue='SimplifiedLoanStatus',
                hue_order=loan_status_order, palette='RdYlGn_r', s =
10)
plt.title('How does the original loan amount relate to the borrowers
APR across different loan statuses?')
plt.ylabel('Borrower APR')
plt.xlabel('Loan Original Amount')
plt.legend(title = 'Loan Status', loc='upper right');
```

The majority of data points cluster in the lower left region, indicating that most loans have lower original amounts and APRs.

A few data points appear as outliers, situated away from the main cluster. These represent loans with significantly higher original amounts or APRs compared to the majority.

The current loans have different and varying APRs and loan amounts. While most of the completed, dedfaulted, and chargedoff are in the lower region.

**Question:** How does the original loan amount vary across different income ranges and loan statuses?

```
## Facet Plotting three variables SimplifiedLoanStatus,
LoanOriginalAmount and IncomeRange
grid = sns.FacetGrid(data = df, col = 'SimplifiedLoanStatus', col_wrap
= 3, col_order= loan_status_order)
grid.map(sns.barplot, 'LoanOriginalAmount', 'IncomeRange', color =
'Purple')
grid.set_titles("{col_name}")
grid.set_axis_labels('Loan Original Amount', 'Income Range')
plt.suptitle('How does the original loan amount vary across different
income ranges and loan statuses?', y=1.02)
grid.figure.set_size_inches(14, 6);
```

As income range increases, the loan original amount also tends to increase. This is evident in the length of the bars across different income ranges.

The "Current" and "Completed" statuses, where the highest loan original amount is not always associated with the highest income range.

Higher income ranges might be associated with fewer defaults, which is worth investigating.

Across all loan statuses, the highest Loans amounts are associated with the 100,000+ income range.

**Question:** How do different terms group with the selected variables?

```
selected_columns = ['LoanOriginalAmount',
'BorrowerAPR','DebtToIncomeRatio']

## Plot Matrix
grid = sns.pairplot(data=df, vars=selected_columns, hue='Term',
palette=['purple', 'yellow', 'b'], plot_kws = {'s':2})
plt.suptitle('How do different terms group with the selected
variables?', y=1.02)
grid.figure.set_size_inches(14, 6);
```

Longer terms (36 and 60 months) tend to be associated with higher loan original amounts.

There doesn't seem to be a strong relationship between term and borrower APR or debt-to-income ratio.

Loan Original Amount vs. Borrower APR have a weak positive correlation, suggesting that larger loans might have slightly higher APRs.

# 7. Conclusions

Finally, In the conclusion section, three different visualizations are done in order to explore univariate, bivariate, and multivariate analysis of loan attributes. Here is the list of summary for the findings:

- A majority of loans are currently performing well, with a significant portion successfully completed.
- California and Texas dominate the borrower base, while other states have significantly lower participation.
- Borrower income is a key driver of loan amounts, with higher-income borrowers tending to take larger loans.
- Credit scores vary significantly across employment types, with employed and full-time borrowers generally exhibiting higher scores.
- Students, particularly those attending community colleges, are more likely to be employed than those in technical schools whom they took a loan with Prosper.
- Loan terms influence loan amounts, with longer terms associated with larger loans.
- The loan market has seen increasing loan amounts over time, with a notable spike in 2009.
- Investor participation correlates with monthly loan payments, indicating a relationship between loan size and investor involvement.
- Outliers in investor count and monthly payments suggest potential high-value or complex loan structures.
- Borrower APR is negatively correlated with credit score, indicating a reward for good credit.

- Loan original amount and stated monthly income exhibit a moderate positive correlation, suggesting income as a key factor in borrowing capacity.