Part I - (Loan Data from Prosper)

by Chia-Hung Lee

Introduction

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

Preliminary Wrangling

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

%matplotlib inline
```

In [2]:
```
df = pd.read_csv('Loan-Data-from-Prosper.csv')
print('Dimensions of the dataset = {}'.format(df.shape))
print('-----------------------------------------------')
print(df.dtypes)
```

```
Dimensions of the dataset = (113937, 81)
-----------------------------------------------
ListingKey                      object
ListingNumber                    int64
ListingCreationDate             object
CreditGrade                     object
Term                             int64
                                 ...
PercentFunded                  float64
Recommendations                  int64
InvestmentFromFriendsCount       int64
InvestmentFromFriendsAmount    float64
Investors                        int64
Length: 81, dtype: object
```

In [3]:
```
df.head()
```

Out[3]:

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

5 rows × 81 columns

In [4]:

```
print(df.dtypes.unique())
print('-------------------------------------------------')
print('Object features: {}'.format(df.dtypes[df.dtypes=='O'].count()))
print(df.dtypes[df.dtypes=='O'])
print('-------------------------------------------------')
print('Object features: {}'.format(df.dtypes[df.dtypes=='int64'].count()))
print(df.dtypes[df.dtypes=='int64'])
print('-------------------------------------------------')
print('Object features: {}'.format(df.dtypes[df.dtypes=='float64'].count()))
print(df.dtypes[df.dtypes=='float64'])
print('-------------------------------------------------')
print('Object features: {}'.format(df.dtypes[df.dtypes=='bool'].count()))
print(df.dtypes[df.dtypes=='bool'])
```

```
[dtype('O') dtype('int64') dtype('float64') dtype('bool')]
---------------------------------------------
Object features: 17
ListingKey                 object
ListingCreationDate        object
CreditGrade                object
LoanStatus                 object
ClosedDate                 object
ProsperRating (Alpha)      object
BorrowerState              object
Occupation                 object
EmploymentStatus           object
GroupKey                   object
DateCreditPulled           object
FirstRecordedCreditLine    object
IncomeRange                object
LoanKey                    object
LoanOriginationDate        object
LoanOriginationQuarter     object
MemberKey                  object
dtype: object
---------------------------------------------
Object features: 11
ListingNumber                 int64
Term                          int64
ListingCategory (numeric)     int64
OpenRevolvingAccounts         int64
LoanCurrentDaysDelinquent     int64
LoanMonthsSinceOrigination    int64
LoanNumber                    int64
LoanOriginalAmount            int64
Recommendations               int64
InvestmentFromFriendsCount    int64
Investors                     int64
dtype: object
---------------------------------------------
Object features: 50
BorrowerAPR                            float64
BorrowerRate                           float64
LenderYield                            float64
EstimatedEffectiveYield                float64
EstimatedLoss                          float64
EstimatedReturn                        float64
ProsperRating (numeric)                float64
ProsperScore                           float64
EmploymentStatusDuration               float64
CreditScoreRangeLower                  float64
CreditScoreRangeUpper                  float64
CurrentCreditLines                     float64
OpenCreditLines                        float64
TotalCreditLinespast7years             float64
OpenRevolvingMonthlyPayment            float64
InquiriesLast6Months                   float64
TotalInquiries                         float64
CurrentDelinquencies                   float64
AmountDelinquent                       float64
DelinquenciesLast7Years                float64
PublicRecordsLast10Years               float64
PublicRecordsLast12Months              float64
RevolvingCreditBalance                 float64
BankcardUtilization                    float64
AvailableBankcardCredit                float64
TotalTrades                            float64
TradesNeverDelinquent (percentage)     float64
TradesOpenedLast6Months                float64
```

```
DebtToIncomeRatio                        float64
StatedMonthlyIncome                      float64
TotalProsperLoans                        float64
TotalProsperPaymentsBilled               float64
OnTimeProsperPayments                    float64
ProsperPaymentsLessThanOneMonthLate      float64
ProsperPaymentsOneMonthPlusLate          float64
ProsperPrincipalBorrowed                 float64
ProsperPrincipalOutstanding              float64
ScorexChangeAtTimeOfListing              float64
LoanFirstDefaultedCycleNumber            float64
MonthlyLoanPayment                       float64
LP_CustomerPayments                      float64
LP_CustomerPrincipalPayments             float64
LP_InterestandFees                       float64
LP_ServiceFees                           float64
LP_CollectionFees                        float64
LP_GrossPrincipalLoss                    float64
LP_NetPrincipalLoss                      float64
LP_NonPrincipalRecoverypayments          float64
PercentFunded                            float64
InvestmentFromFriendsAmount              float64
dtype: object
-----------------------------------------------
Object features: 3
IsBorrowerHomeowner     bool
CurrentlyInGroup        bool
IncomeVerifiable        bool
dtype: object
```

Note on data description

> - ListingKey, ListingNumber: are ID for listing, could be used as index.
> - CreditGrade: Only for listings in and before 2009.
> - LoanStatus: [Cancelled, Chargedoff, Completed, Current, Defaulted, FinalPaymentInProgress, PastDue.]
> - [EstimatedEffectiveYield, EstimatedLoss, EstimatedReturn]: Only for listings after July 2009.
> - [ProsperRating(numeric), ProsperRating(Alpha), ProsperScore]: Only for listings after July 2009.
> - [CurrentlyInGroup, GroupKey]: affiliation to specific groups.

What is the structure of your dataset?

> There are 113,937 load data in the dataset with 81 features (Object: 17; Int64: 11; Float64: 50; Boolean:3). The majority (61 variables) are numeric in nature.

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

While the dataset offers an array of features for exploration, this analysis focuses primarily on investigating the BorrowerAPR and BorrowerRate variables, in addition to other relevant attributes. We are trying to answer the following questions:

> - What affects the borrower’s APR or interest rate?
> - What affects the original loan amount?

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

> The key attributes that will play a pivotal role in supporting the analysis of BorrowerAPR and BorrowerRate include:

Column	Description

|**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. |**BorrowerAPR** | The Borrower's Annual Percentage Rate (APR) for the loan. |**BorrowerRate** | The Borrower's interest rate for this loan. |**ProsperRating (Alpha)**| The Prosper Rating assigned at the time the listing was created between AA - HR. 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 |**EmploymentStatus** | The employment status of the borrower at the time they posted the listing. |**IsBorrowerHomeowner** | A Borrower will be classified as a homowner if they have a mortgage on their credit profile or provide documentation confirming they are a homeowner. TRUE or FALSE |**IncomeRange** | The income range of the borrower at the time the listing was created. |**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%). |**StatedMonthlyIncome** | The monthly income the borrower stated at the time the listing was created. |**LoanOriginalAmount** | The original amount of the loan. | **MonthlyLoanPayment** | The scheduled monthly loan payment.

Constructing the Modified Dataset to for Analysis

In [5]:
```
df_mod = df[['LoanKey','Term','LoanStatus','BorrowerAPR','BorrowerRate',
             'ProsperRating (Alpha)','ListingCategory (numeric)','EmploymentStatus',
             'IsBorrowerHomeowner','IncomeRange','DebtToIncomeRatio',
             'LoanOriginalAmount','StatedMonthlyIncome','MonthlyLoanPayment']].copy()
df_mod.head()
```

Out[5]:

	LoanKey	Term	LoanStatus	BorrowerAPR	BorrowerRate	ProsperRating (Alpha)
0	E33A3400205839220442E84	36	Completed	0.16516	0.1580	NaN
1	9E3B37071505919926B1D82	36	Current	0.12016	0.0920	A
2	6954337960046817851BCB2	36	Completed	0.28269	0.2750	NaN
3	A0393664465886295619C51	36	Current	0.12528	0.0974	A
4	A180369302188889200689E	36	Current	0.24614	0.2085	D

In [6]:
```
# Rename columns
df_mod.rename(columns={'ProsperRating (Alpha)':'ProsperRating', 'ListingCategory (num
df_mod.head()
```

Out[6]:

	LoanKey	Term	LoanStatus	BorrowerAPR	BorrowerRate	ProsperRating
0	E33A3400205839220442E84	36	Completed	0.16516	0.1580	NaN
1	9E3B37071505919926B1D82	36	Current	0.12016	0.0920	A
2	6954337960046817851BCB2	36	Completed	0.28269	0.2750	NaN
3	A0393664465886295619C51	36	Current	0.12528	0.0974	A
4	A180369302188889200689E	36	Current	0.24614	0.2085	D

In [7]:
```
df_mod.duplicated().value_counts()
```

Out[7]:
```
False    113066
True        871
dtype: int64
```

In [8]:
```
# Remode duplicate data
df_mod = df_mod.drop_duplicates()
df_mod.duplicated().value_counts()
```

Out[8]:
```
False    113066
dtype: int64
```

In [9]:
```
df_mod.isnull().sum()
```

Out[9]:
```
LoanKey                   0
Term                      0
LoanStatus                0
BorrowerAPR              25
BorrowerRate              0
ProsperRating         29084
ListingCategory           0
EmploymentStatus       2255
IsBorrowerHomeowner       0
IncomeRange               0
DebtToIncomeRatio      8472
LoanOriginalAmount        0
StatedMonthlyIncome       0
MonthlyLoanPayment        0
dtype: int64
```

In [10]:
```
df_mod=df_mod.dropna()
df_mod.isnull().sum()
```

Out[10]:
```
LoanKey               0
Term                  0
LoanStatus            0
BorrowerAPR           0
BorrowerRate          0
ProsperRating         0
ListingCategory       0
EmploymentStatus      0
IsBorrowerHomeowner   0
IncomeRange           0
DebtToIncomeRatio     0
LoanOriginalAmount    0
StatedMonthlyIncome   0
MonthlyLoanPayment    0
dtype: int64
```

In [11]:
```
df_mod.set_index('LoanKey',inplace=True)
```

In [12]:
```
df_mod.shape
```

Out[12]:
```
(76768, 13)
```

Structure of my dataset (df_mod)

> Number of Rows in dataset are 76768 and Number of Columns in dataset are 13.

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

> - LoanKey (Index)
> - Term
> - LoanStatus
> - BorrowerAPR
> - BorrowerRate
> - ProsperRating (Alpha)
> - ListingCategory (numeric)
> - EmploymentStatus
> - IsBorrowerHomeowner
> - IncomeRange
> - DebtToIncomeRatio
> - LoanOriginalAmount
> - StatedMonthlyIncome
> - MonthlyLoanPayment

In [13]:
```
df_mod.head()
```

Out[13]:

	Term	LoanStatus	BorrowerAPR	BorrowerRate	ProsperRating	Li
LoanKey						
9E3B37071505919926B1D82	36	Current	0.12016	0.0920	A	
A0393664465886295619C51	36	Current	0.12528	0.0974	A	
A180369302188889200689E	36	Current	0.24614	0.2085	D	
C3D63702273952547E79520	60	Current	0.15425	0.1314	B	
CE963680102927767790520	36	Current	0.31032	0.2712	E	

Univariate Exploration

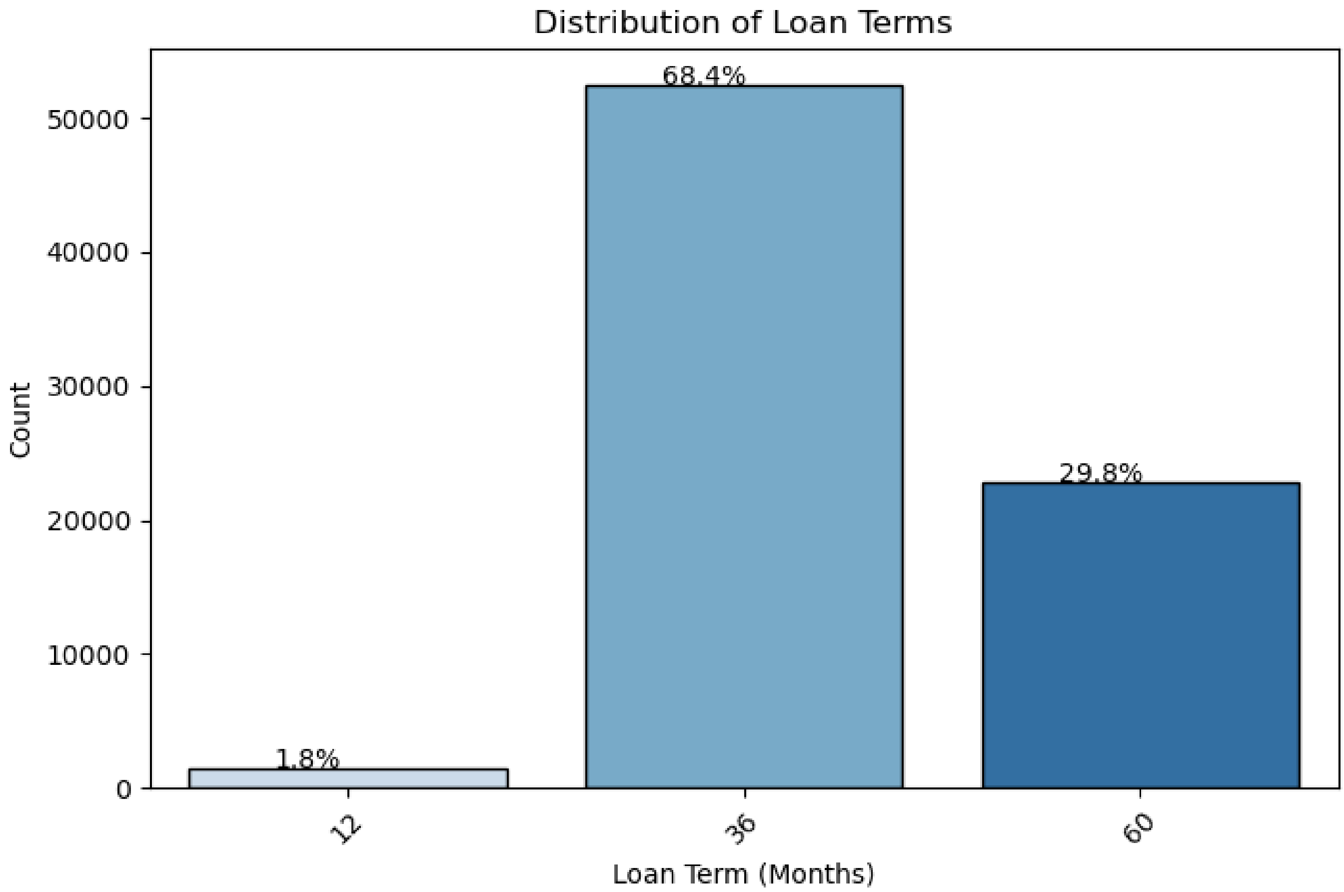
Term

In [14]:
```
df_mod.Term.value_counts()
```

Out[14]:
```
36    52505
60    22849
12     1414
Name: Term, dtype: int64
```

In [15]:
```
# Plot a countplot for the distribution of loan terms
plt.figure(figsize=[8,5])
```

```
ax = sns.countplot(data=df_mod, x='Term', palette = 'Blues', edgecolor='black')

plt.xlabel('Loan Term (Months)')  # Label for the x-axis
plt.ylabel('Count')  # Label for the y-axis
plt.title('Distribution of Loan Terms')  # Title for the plot

# Show percentage labels
total_count = len(df_mod)
for i in ax.patches:
    percentage = '{:.1f}%'.format(100 * i.get_height() / total_count)
    x = i.get_x() + i.get_width() / 2 - 0.1
    y = i.get_height() + 20
    ax.annotate(percentage, (x, y), fontsize=10, ha='center')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

plt.show()  # Display the plot
```



> - There are **3 available loan term**. The most frequently selected term is **36 month**, although there is also a notable preference for the **60 month** term among some borrowers.

LoanStatus

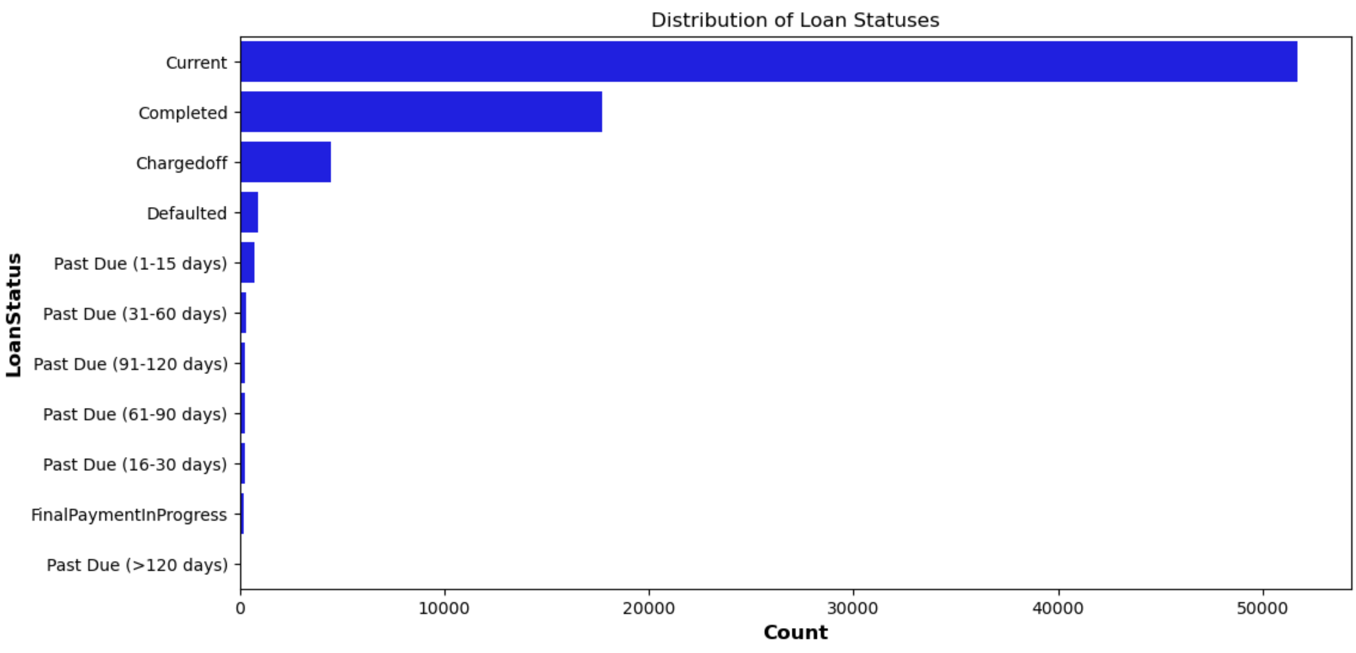
- Which categories do majority of loans fall in ?

In [16]:
```
df_mod['LoanStatus'].value_counts()
```

```
Out[16]: Current                    51712
         Completed                  17691
         Chargedoff                  4445
         Defaulted                    885
         Past Due (1-15 days)         716
         Past Due (31-60 days)        325
         Past Due (91-120 days)       277
         Past Due (61-90 days)        274
         Past Due (16-30 days)        242
         FinalPaymentInProgress       187
         Past Due (>120 days)          14
         Name: LoanStatus, dtype: int64
```

In [17]:
```
def BarPlot(df, var, order=None, figsize=[12,6], title=''):
    # Create the figure and axes
    fig, ax = plt.subplots(figsize=figsize)

    # Plot the barplot
    type_count = df[var].value_counts()
    sns.barplot(x=type_count, y=type_count.index, order=order, color='blue')

    # Add title
    ax.set_title(title)

    # Add X and Y labels and format them
    ax.set_xlabel('Count', fontsize=12, weight="bold")
    ax.set_ylabel(var, fontsize=12, weight="bold")


    plt.show()
```

In [18]:
```
BarPlot(df_mod, 'LoanStatus', title='Distribution of Loan Statuses')
```



> - The plot above illustrates that the majority of loans within the dataset belong to the categories of "Current," followed by "Completed," "Charged-off," "Defaulted," and "Past Due (1-15 days)."

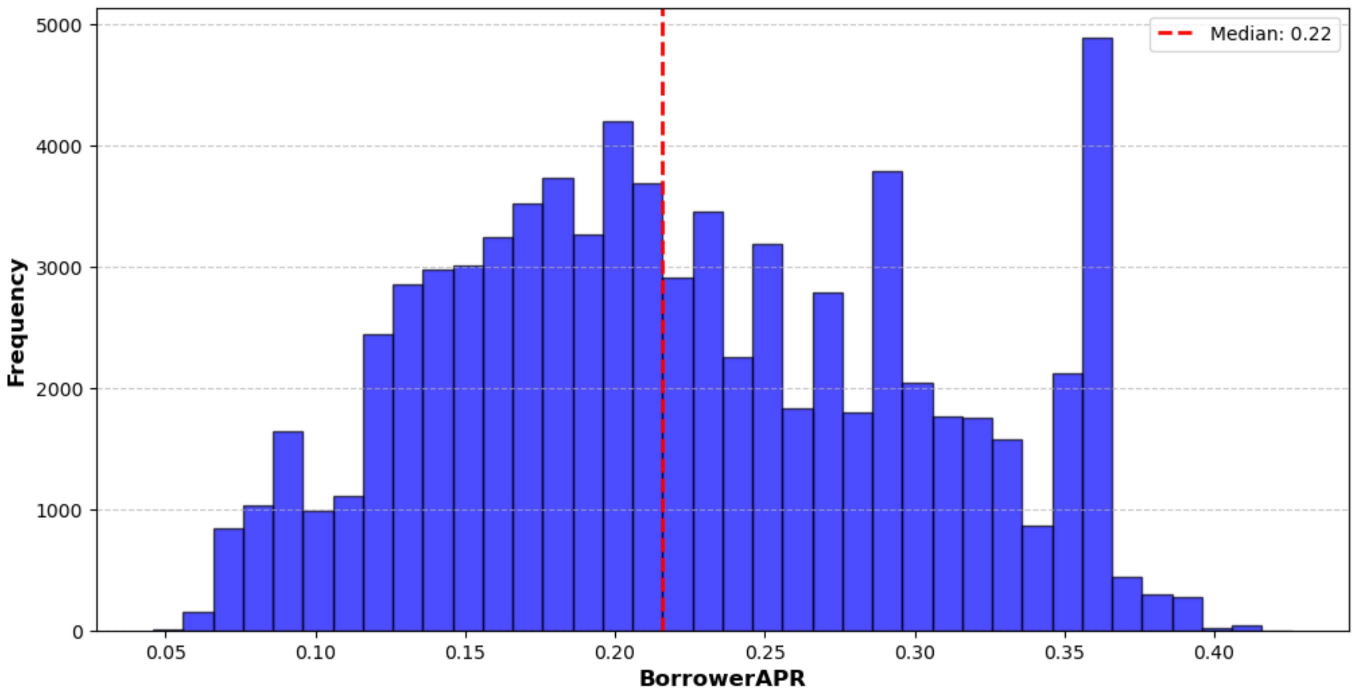
BorrowerAPR

In [19]:
```
df_mod.BorrowerAPR.describe()
```

Out[19]:
```
count    76768.000000
mean         0.223978
std          0.079291
min          0.045830
25%          0.162590
50%          0.215660
75%          0.287800
max          0.423950
Name: BorrowerAPR, dtype: float64
```

In [20]:
```python
# Function to plot Histogram distribution
def HistPlot(df, var, interval=30, figsize=[12,6], title=''):
    # Set intervals for bins
    bins = np.arange(df[var].min(), df[var].max() + interval, interval)

    # Create the figure and axes
    fig, ax = plt.subplots(figsize=figsize)

    # Plot the histogram
    ax.hist(df[var], bins=bins, edgecolor='black', color='Blue', alpha=0.7)

    # Add a vertical line for the median
    median = df[var].median()
    ax.axvline(median, color='red', linestyle='dashed', linewidth=2, label=f'Median:

    # Add title
    ax.set_title(title)

    # Add X and Y labels and format them
    ax.set_xlabel(var, fontsize=12, weight="bold")
    ax.set_ylabel('Frequency', fontsize=12, weight="bold")

    # Add grid lines for better readability
    ax.grid(axis='y', linestyle='--', alpha=0.7)

    # Add a legend for the median line
    ax.legend(loc='upper right')

    plt.show()
```

In [21]:
```python
# Call plot function (using all data)
HistPlot(df_mod, 'BorrowerAPR', 0.01)
```



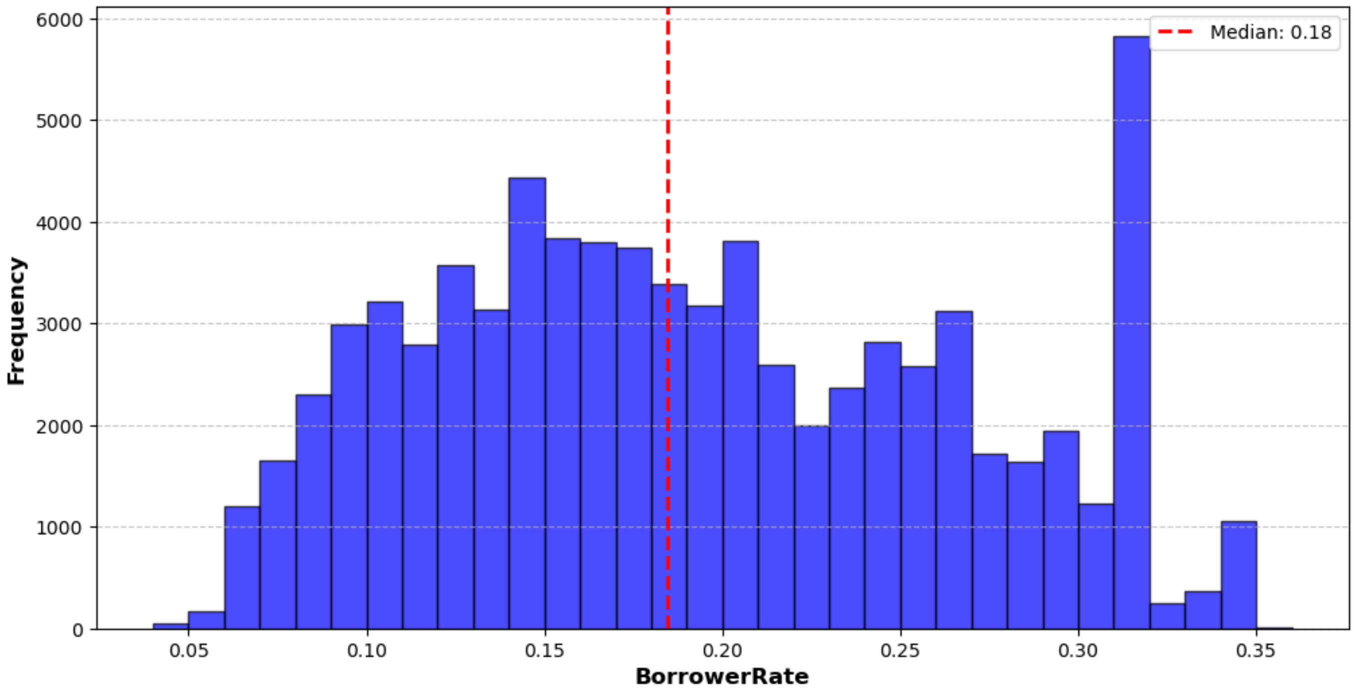
- **Higher BorrowerAPR** indicates a greater interest rate on borrowed funds, resulting in increased interest payments.
- In the distribution plot of BorrowerAPR, there is a prominent peak around 0.09, 0.23, 0.25, 0.27, 0.29, a minor peak at approximately 0.18, and a significant peak around 0.36. Very few individuals have an APR exceeding 0.4.

BorrowerRate

In [22]:
```
df_mod.BorrowerRate.describe()
```

Out[22]:
```
count    76768.000000
mean         0.193653
std          0.074018
min          0.040000
25%          0.134900
50%          0.184500
75%          0.254900
max          0.360000
Name: BorrowerRate, dtype: float64
```

In [23]:
```
HistPlot(df_mod, 'BorrowerRate', 0.01)
```



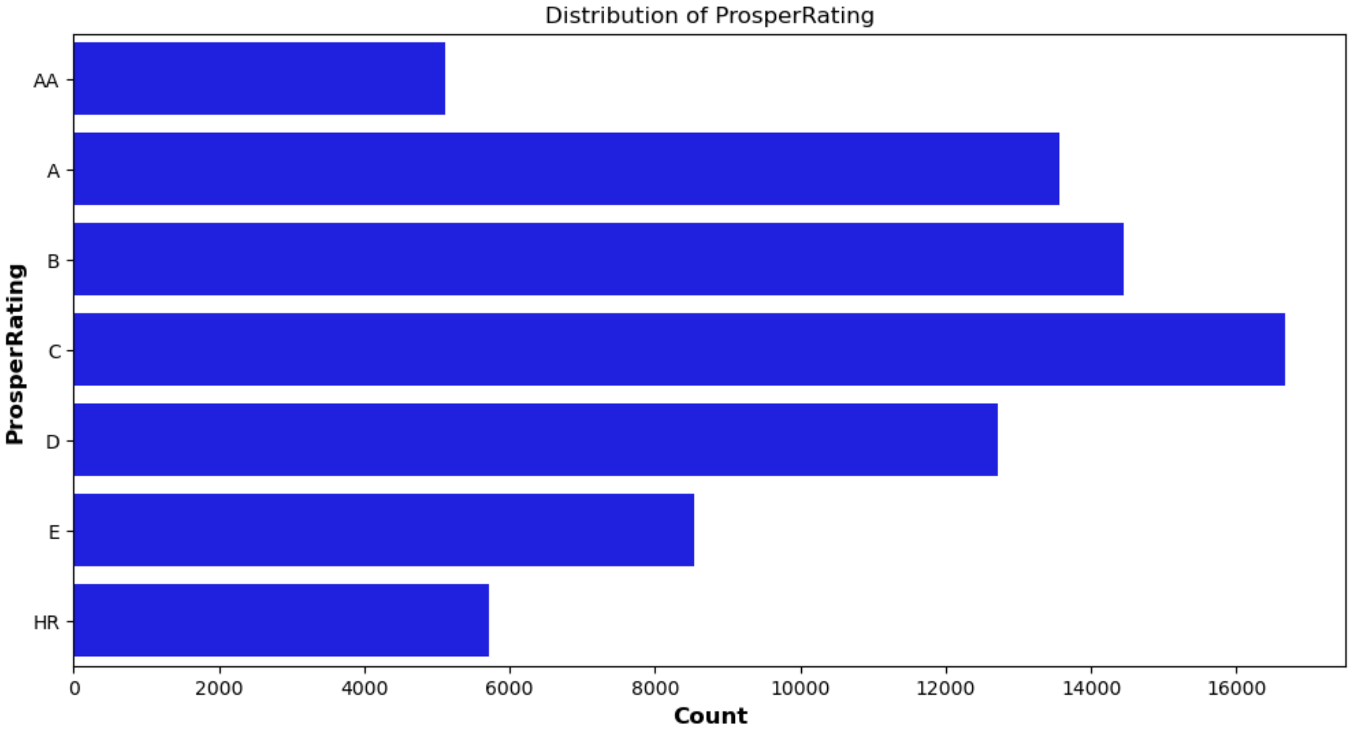
- The distribution of **BorrowerRate** is multimodal, with a prominent peak at approximately 0.32. Rates exceeding 0.35 are rare, with a median rate of 0.18.

ProsperRating

In [24]:
```
df_mod.ProsperRating.value_counts()
```

Out[24]:
```
C     16671
B     14444
A     13555
D     12724
E      8543
HR     5722
AA     5109
Name: ProsperRating, dtype: int64
```

In [25]:
```
order=['AA','A','B','C','D','E','HR']
BarPlot(df_mod, 'ProsperRating', order=order, title='Distribution of ProsperRating')
```



> - The most common **ProsperRating** are C, B, A, and D

ListingCategory

In [26]:
```
df_mod.ListingCategory.value_counts()
```

Out[26]:
```
1     49099
7      8334
2      6326
3      3626
6      2038
13     1779
15     1390
14      794
18      785
20      724
19      718
16      289
5       201
11      198
8       188
9        83
10       82
17       50
12       45
0        19
Name: ListingCategory, dtype: int64
```

In [27]:
```
listing_categories = {0 : 'Not Available',
                      1 : 'Debt Consolidation',
                      2 : 'Home Improvement',
                      3 : 'Business',
                      4 : 'Personal Loan',
                      5 : 'Student Use',
                      6 : 'Auto',
                      7 : 'Other',
                      8 : 'Baby&Adoption',
```

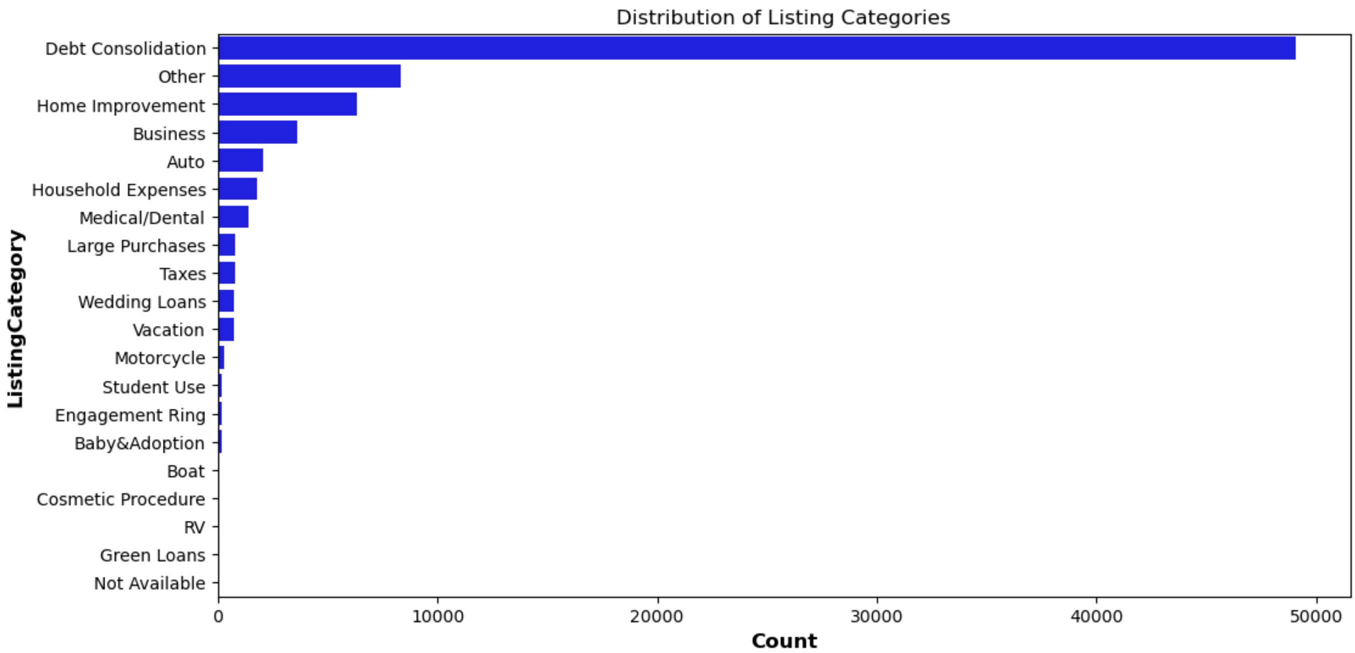
```
                 9 : 'Boat',
                 10 : 'Cosmetic Procedure',
                 11 : 'Engagement Ring',
                 12 : 'Green Loans',
                 13 : 'Household Expenses',
                 14 : 'Large Purchases',
                 15 : 'Medical/Dental',
                 16 : 'Motorcycle',
                 17 : 'RV',
                 18 : 'Taxes',
                 19 : 'Vacation',
                 20 : 'Wedding Loans'}
```

In [28]:
```
df_mod['ListingCategory'] = df_mod['ListingCategory'].replace(to_replace=listing_cate
```

In [29]:
```
df_mod.ListingCategory.value_counts()
```

Out[29]:
```
Debt Consolidation    49099
Other                  8334
Home Improvement       6326
Business               3626
Auto                   2038
Household Expenses     1779
Medical/Dental         1390
Large Purchases         794
Taxes                   785
Wedding Loans           724
Vacation                718
Motorcycle              289
Student Use             201
Engagement Ring         198
Baby&Adoption           188
Boat                     83
Cosmetic Procedure       82
RV                       50
Green Loans              45
Not Available            19
Name: ListingCategory, dtype: int64
```

In [30]:
```
BarPlot(df_mod, 'ListingCategory', title='Distribution of Listing Categories')
```



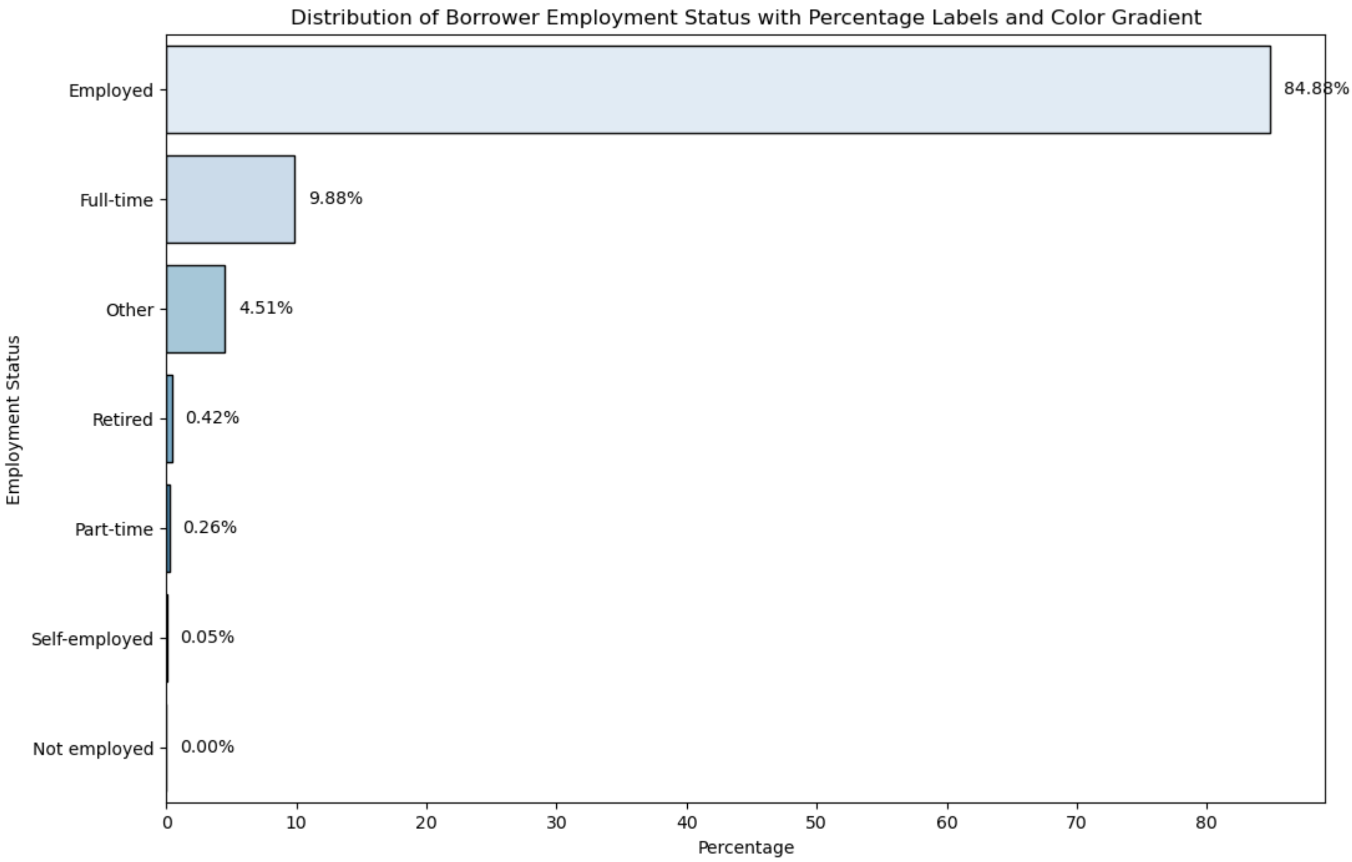
- From the chart above, it is evident that a significant number of individuals acquire loans for the purpose of **"Debt Consolidation"** primarily to manage and consolidate their existing debt obligations.

EmploymentStatus

In [31]:
```
df_mod.EmploymentStatus.value_counts()
```

Out[31]:
```
Employed         65160
Full-time         7584
Other             3462
Retired            320
Part-time          199
Self-employed       42
Not employed         1
Name: EmploymentStatus, dtype: int64
```

In [32]:
```
# Calculate the percentage of borrowers in each employment status category
percentage_data = (df_mod['EmploymentStatus'].value_counts() / len(df_mod)) * 100

# Create a color palette with a gradient
color_palette = sns.color_palette("Blues", len(percentage_data))

# Sort the data by count to maintain color consistency
percentage_data = percentage_data.sort_values(ascending=False)

# Create the bar chart with color gradient
plt.figure(figsize=(12, 8))
bars = sns.barplot(y=percentage_data.index, x=percentage_data.values, palette=color_p

# Add percentage labels on the bars
for bar, percentage in zip(bars.patches, percentage_data):
    width = bar.get_width()
    plt.text(width + 1, bar.get_y() + bar.get_height() / 2, f'{percentage:.2f}%', ha=

plt.xlabel('Percentage')
plt.ylabel('Employment Status')
plt.title('Distribution of Borrower Employment Status with Percentage Labels and Colo
plt.show()
```



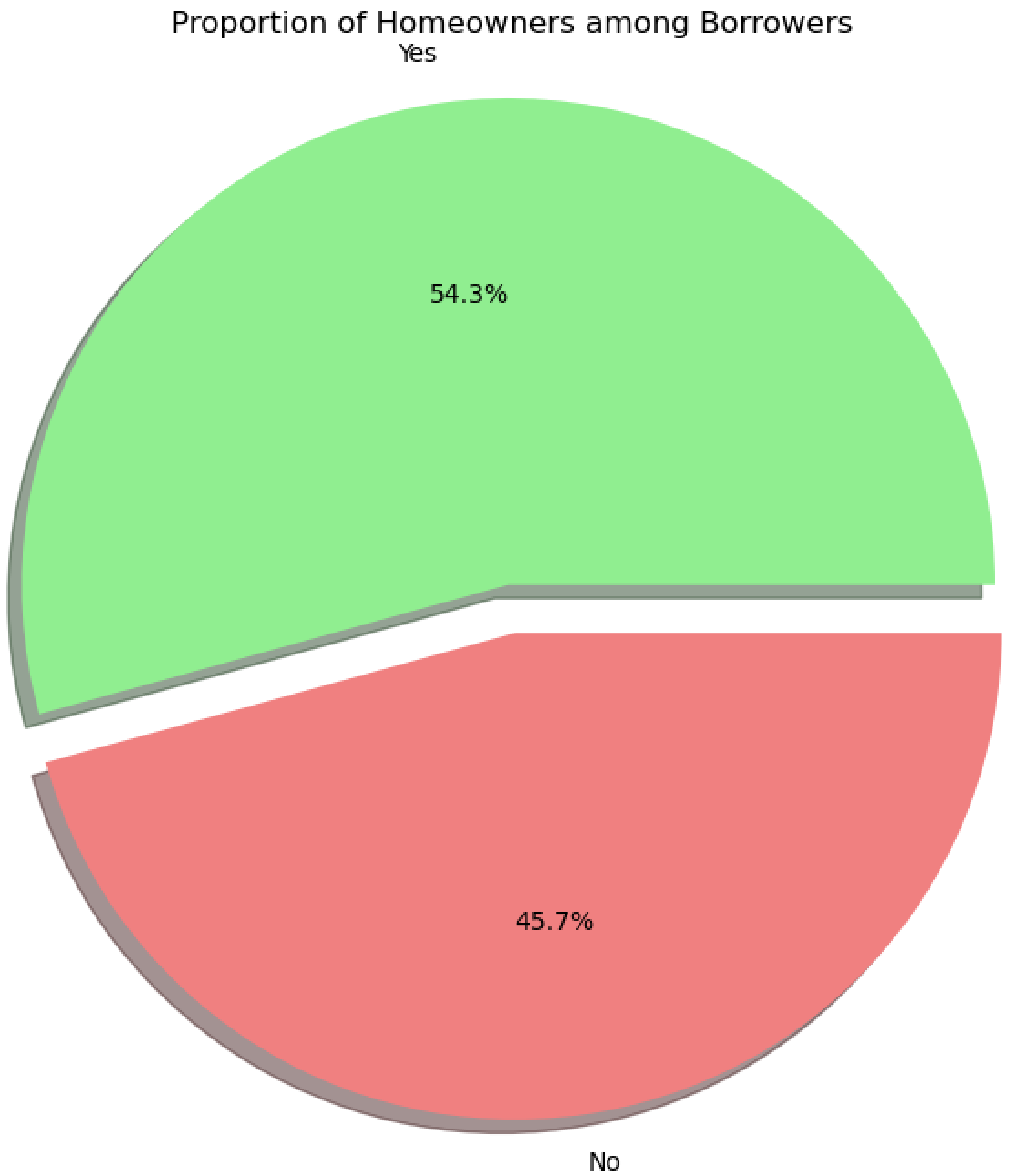
- The majority of loan borrowers are **Employed**, which is a logical trend as a source of income is necessary to repay the loans.

IsBorrowerHomeowner

In [33]:
```
df_mod.IsBorrowerHomeowner.value_counts()
```

Out[33]:
```
True     41676
False    35092
Name: IsBorrowerHomeowner, dtype: int64
```

In [34]:
```
# Pie chart to show the proportion of people who are homeowners
labels = ['Yes', 'No']  # Define the labels
plt.figure(figsize=(8, 8))
colors = ['lightgreen', 'lightcoral']
explode = (0.1, 0)  # Explode the 'Yes' slice for emphasis

plt.pie(df_mod['IsBorrowerHomeowner'].value_counts(), labels=labels, autopct='%0.1f%%
plt.title('Proportion of Homeowners among Borrowers')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```



- A slight majority of borrowers are homeowners.

IncomeRange

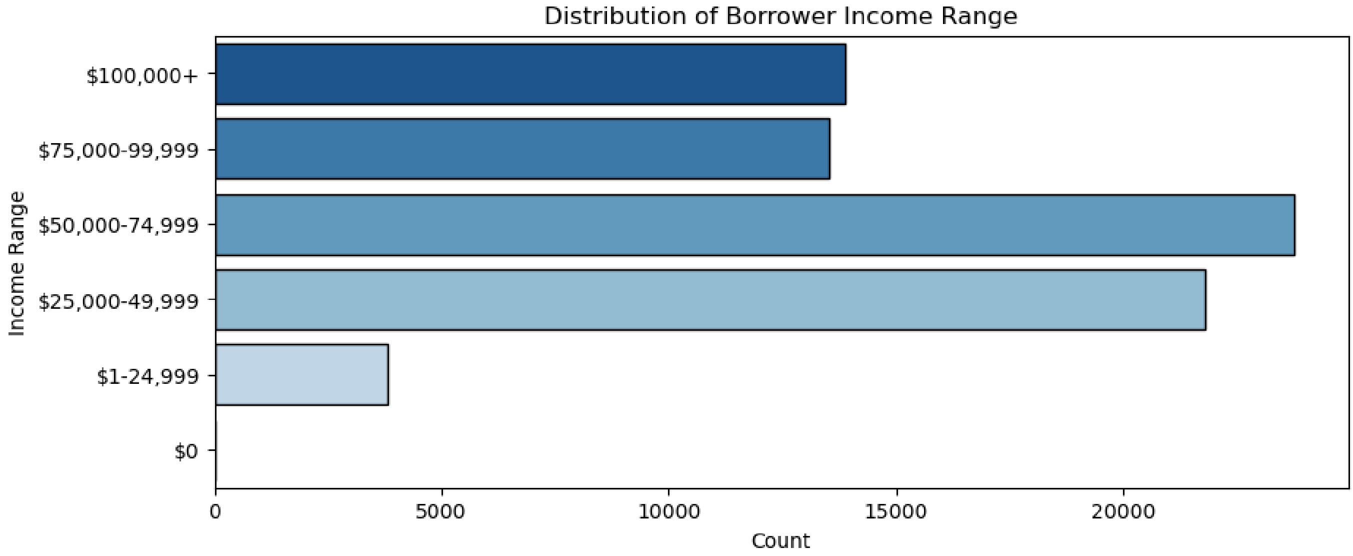
In [35]:
```
df_mod.IncomeRange.value_counts()
```

Out[35]:
```
$50,000-74,999    23756
$25,000-49,999    21795
$100,000+         13889
$75,000-99,999    13519
$1-24,999          3808
Not employed          1
Name: IncomeRange, dtype: int64
```

In [36]:
```
# Create ordinal categories for income ranges
ordinal_income_ranges = ['$100,000+', '$75,000-99,999', '$50,000-74,999', '$25,000-49
income_order = pd.api.types.CategoricalDtype(ordered=True, categories=ordinal_income_
df_mod['IncomeRange'] = df_mod['IncomeRange'].astype(income_order)
df_mod['IncomeRange']
```

Out[36]:
```
LoanKey
9E3B37071505919926B1D82    $50,000-74,999
A0393664465886295619C51    $25,000-49,999
A180369302188889200689E         $100,000+
C3D63702273952547E79520         $100,000+
CE963680102927767790520    $25,000-49,999
                                ...
9BD7367919051593140DB62    $50,000-74,999
62D93634569816897D5A276    $75,000-99,999
DD1A370200396006300ACA0    $25,000-49,999
589536350469116027ED11B    $25,000-49,999
00AF3704550953269A64E40    $50,000-74,999
Name: IncomeRange, Length: 76768, dtype: category
Categories (6, object): ['$100,000+' < '$75,000-99,999' < '$50,000-74,999' < '$25,00
0-49,999' < '$1-24,999' < '$0']
```

In [37]:
```
# Plot a horizontal bar chart for the distribution of LoanStatus
plt.figure(figsize=[10, 4])

sns.countplot(data=df_mod, y='IncomeRange', palette='Blues_r', edgecolor='black')  #

plt.xlabel('Count')  # Label for the x-axis
plt.ylabel('Income Range')  # Label for the y-axis
plt.title('Distribution of Borrower Income Range')  # Title for the plot

# Display the plot
plt.show()


# Median income
median_income = df_mod['IncomeRange']
median_income = median_income.sort_values().reset_index()['IncomeRange'][76768/2]
print('The median income category = {}'.format(median_income))
```




```
The median income category = $50,000–74,999
```

> - The **median IncomeRange** (Also the most common) is `$50,000` - `$74,999`, and follows by `$25,000` - `$49,999`.

DebtToIncomeRatio

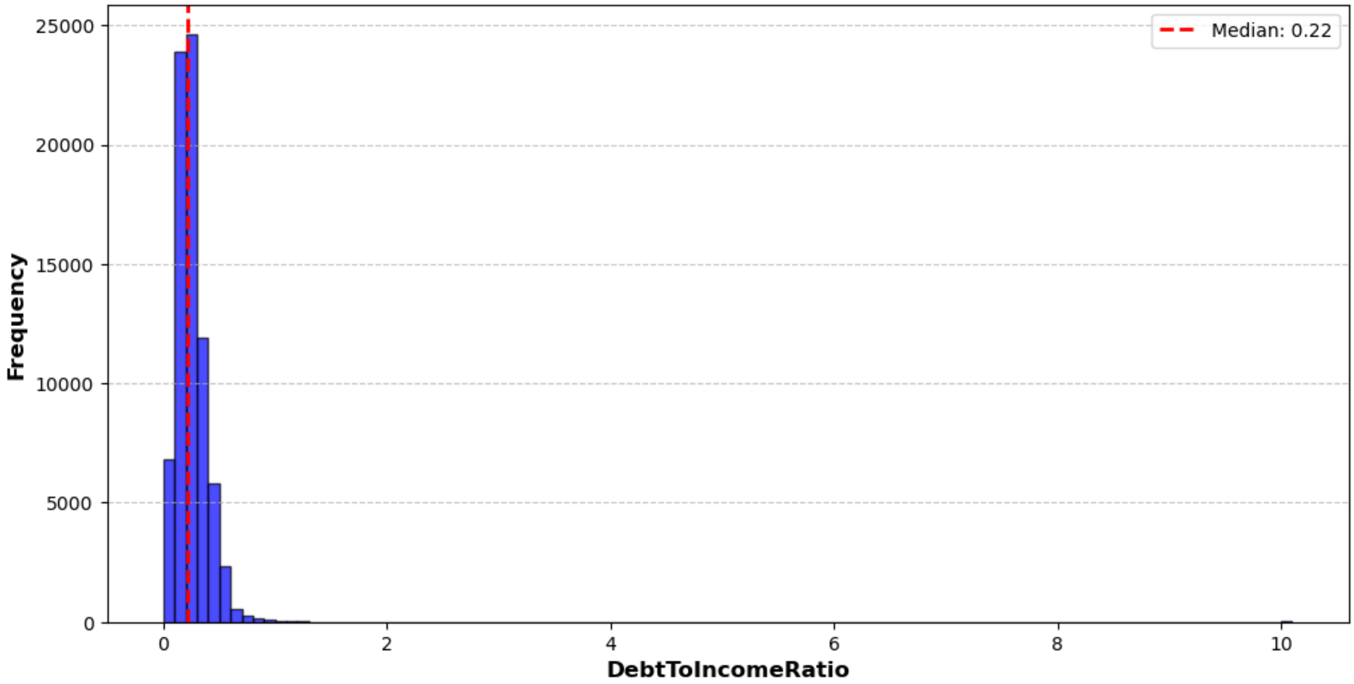
In [38]:
```
df_mod.DebtToIncomeRatio.value_counts()
```

Out[38]:
```
0.18    3184
0.22    2974
0.17    2728
0.14    2681
0.21    2545
        ...
1.54       1
3.81       1
4.54       1
1.93       1
2.53       1
Name: DebtToIncomeRatio, Length: 259, dtype: int64
```

In [39]:
```
df_mod.DebtToIncomeRatio.describe()
```

Out[39]:
```
count    76768.000000
mean         0.258692
std          0.319727
min          0.000000
25%          0.150000
50%          0.220000
75%          0.320000
max         10.010000
Name: DebtToIncomeRatio, dtype: float64
```

In [40]:
```
HistPlot(df_mod, 'DebtToIncomeRatio', 0.1)
```



In [41]:
```
df_mod['DebtToIncomeRatio'].quantile(0.99)
```

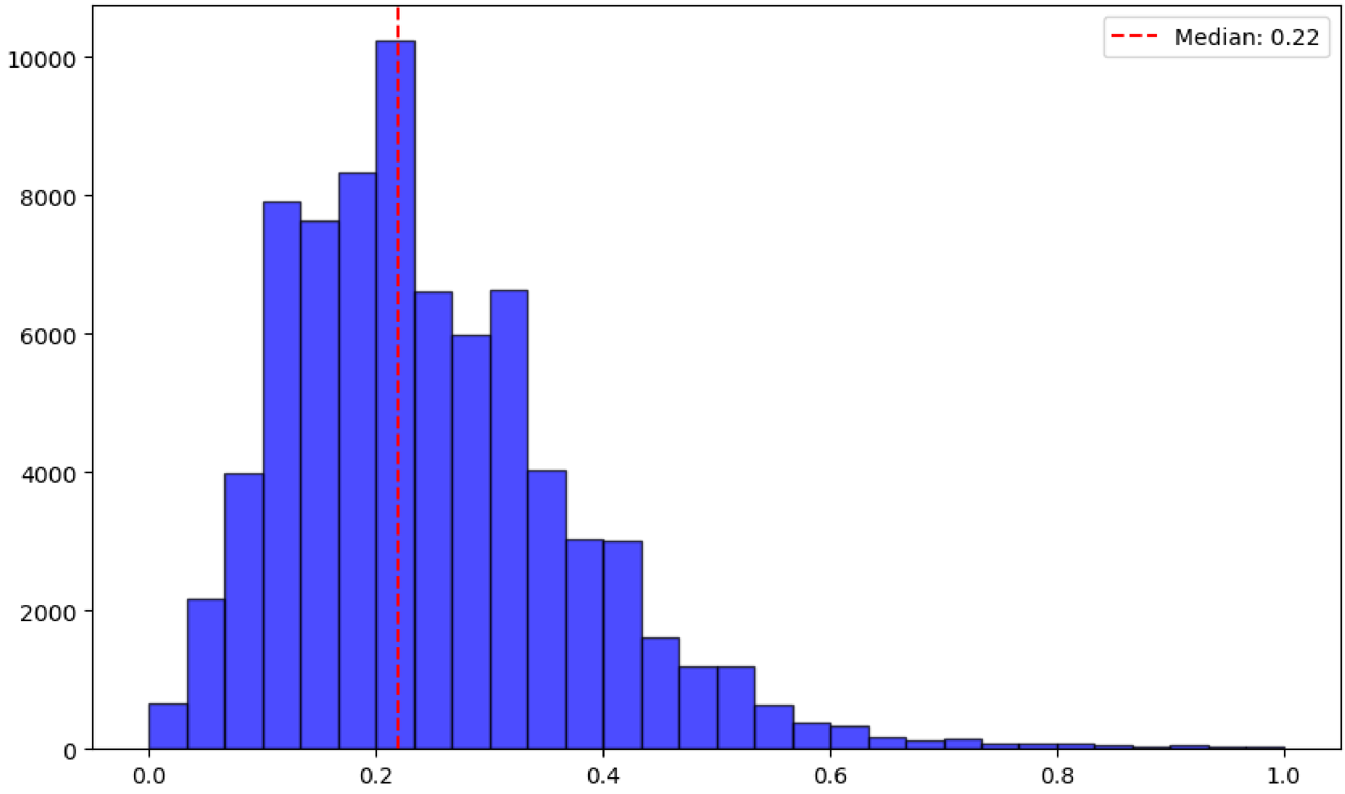
Out[41]: 0.72

In [42]:
```
x=df_mod[df_mod.DebtToIncomeRatio==10.01]
print('There are {} data entry with DebtToIncomeRatio = 10.01'.format(x.shape[0]))
```

```
There are 46 data entry with DebtToIncomeRatio = 10.01
```

> - From the plot above, the majority (99%) of the **DebtToIncomeRatio** are within 1.
> - There are 46 data entry with maximun DebtToIncomeRatio = 10.01 (should not be manual error)
> - We will look into the majority data of the DebtToIncomeRatio

In [43]:
```
fig, ax = plt.subplots(figsize=[10,6])
ax.hist(df_mod['DebtToIncomeRatio'], range=(0,1),
        bins=30, edgecolor='black', color='Blue', alpha=0.7)
# Add a vertical line at the median (50th percentile) in red
median=df_mod['DebtToIncomeRatio'].median()
plt.axvline(df_mod['DebtToIncomeRatio'].median(), color='red', linestyle='--', label=

# Remove the legend
plt.legend().set_visible(True)

plt.show()
```



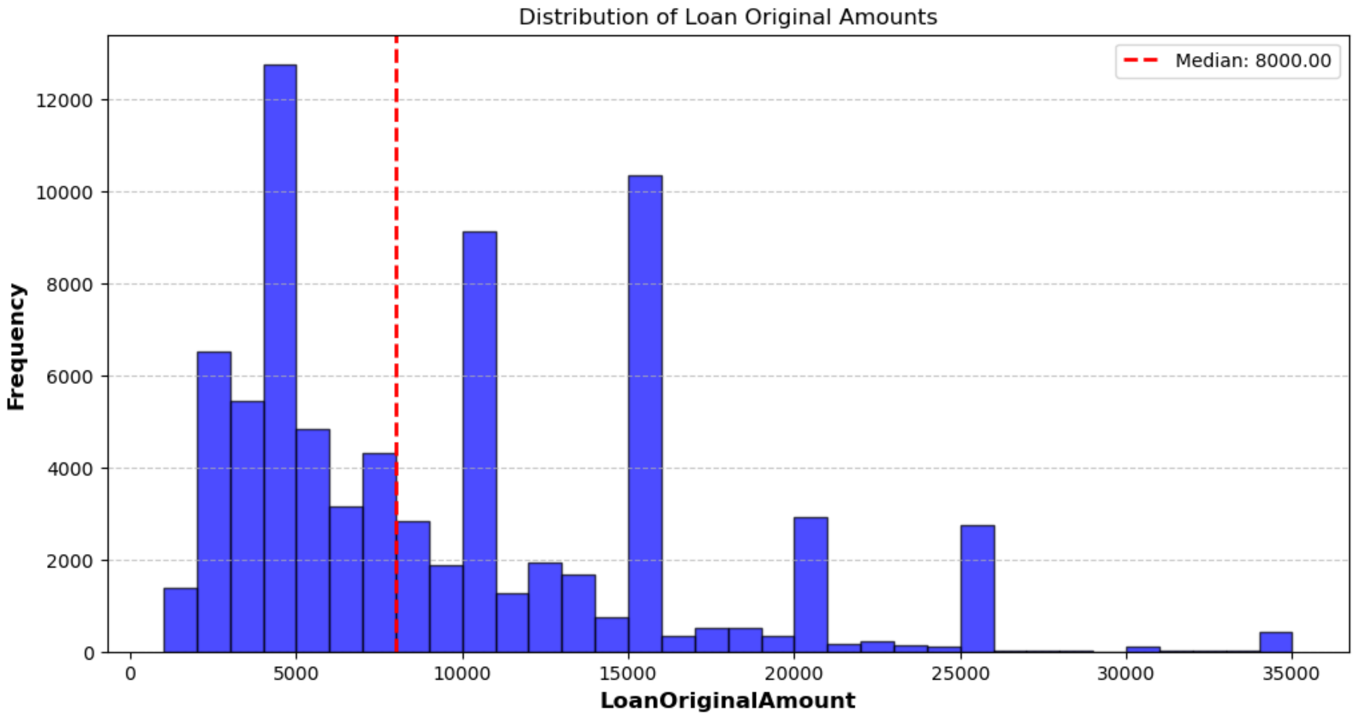
> - **The majority of DebtToIncomeRatios fall below 50%**, indicating that most borrowers have a relatively low debt-to-income ratio. However, there is a notable minority with higher debt-to-income ratios. A lower ratio is generally favorable for borrowers as it signifies their ability to manage and repay their loans effectively.

OriginalLoanAmount

In [44]:
```
df_mod.LoanOriginalAmount.describe()
```

Out[44]:
```
count    76768.000000
mean      9248.961416
std       6389.782292
min       1000.000000
25%       4000.000000
50%       8000.000000
75%      14000.000000
max      35000.000000
Name: LoanOriginalAmount, dtype: float64
```

In [45]:
```
HistPlot(df_mod, 'LoanOriginalAmount', 1000, title="Distribution of Loan Original Amo
```



> - The most common **OriginalLoanAmount** occur at 4k, followed by 15k and 10k.

StatedMonthlyIncome

In [46]:
```
df_mod.StatedMonthlyIncome.sort_values()
```

Out[46]:
```
LoanKey
DCA9366929635721086C17C          0.250000
A2E63643277665696D47A5E          1.416667
DA623646827555431CA659F          1.833333
C9893607029050217F845B3          1.833333
B1D236532712404897E17A1          1.916667
                                   ...
1B313703263370099FC7B30     158333.333333
C6C536749708412328A7D15     394400.000000
B7F13618099831699F10189     416666.666667
5D0136156133365609F840F     466666.666667
77AC3617940949299F18FAF     483333.333333
Name: StatedMonthlyIncome, Length: 76768, dtype: float64
```

In [47]:
```
df_mod.StatedMonthlyIncome.describe()
```

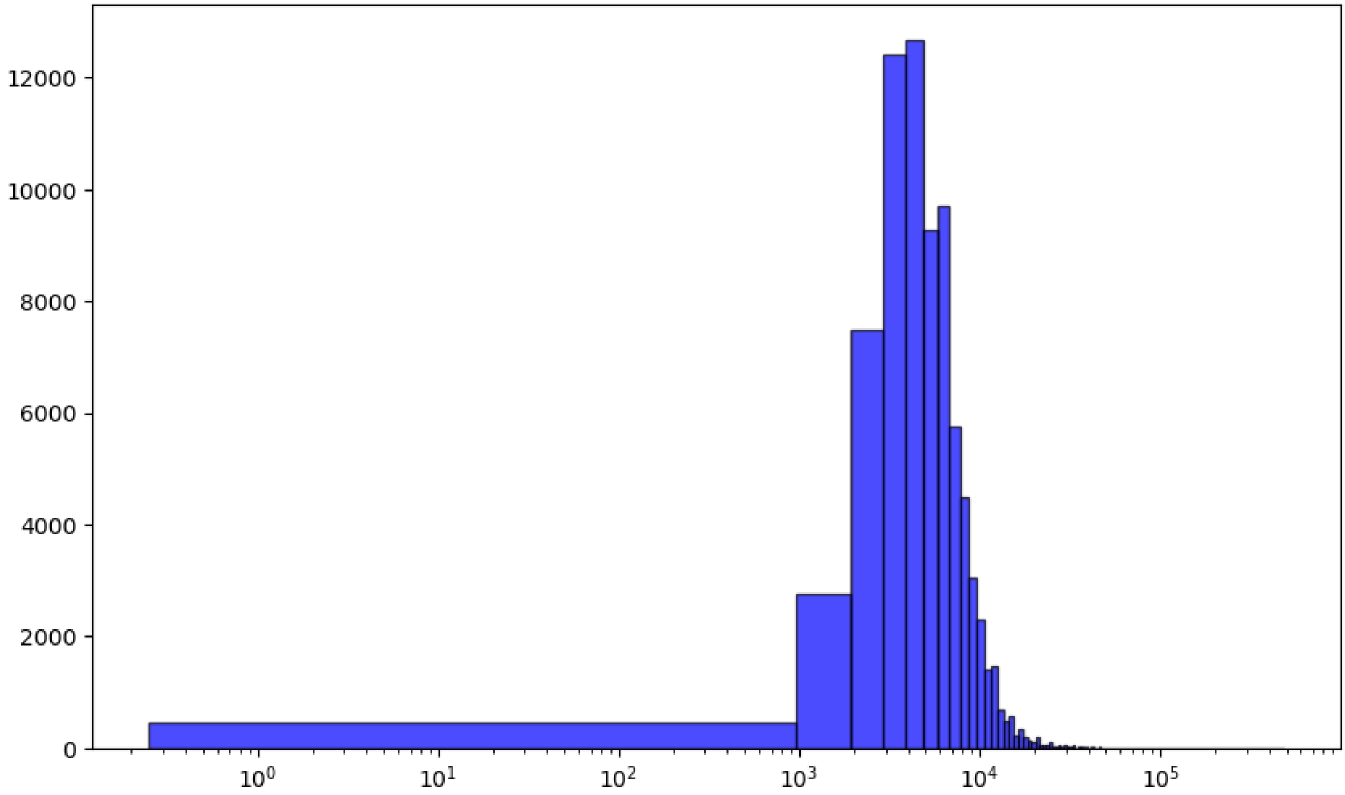
Out[47]:
```
count     76768.000000
mean       5964.256138
std        5089.682309
min           0.250000
25%        3528.895833
50%        5000.000000
75%        7166.666667
max      483333.333333
Name: StatedMonthlyIncome, dtype: float64
```

In [48]:
```
df_mod['StatedMonthlyIncome'].quantile(0.99)
```

Out[48]:
```
20416.666667
```

In [49]:
```
fig, ax = plt.subplots(figsize=[10,6])
ax.hist(df_mod['StatedMonthlyIncome'],
        bins=500, edgecolor='black', color='Blue', alpha=0.7)
```

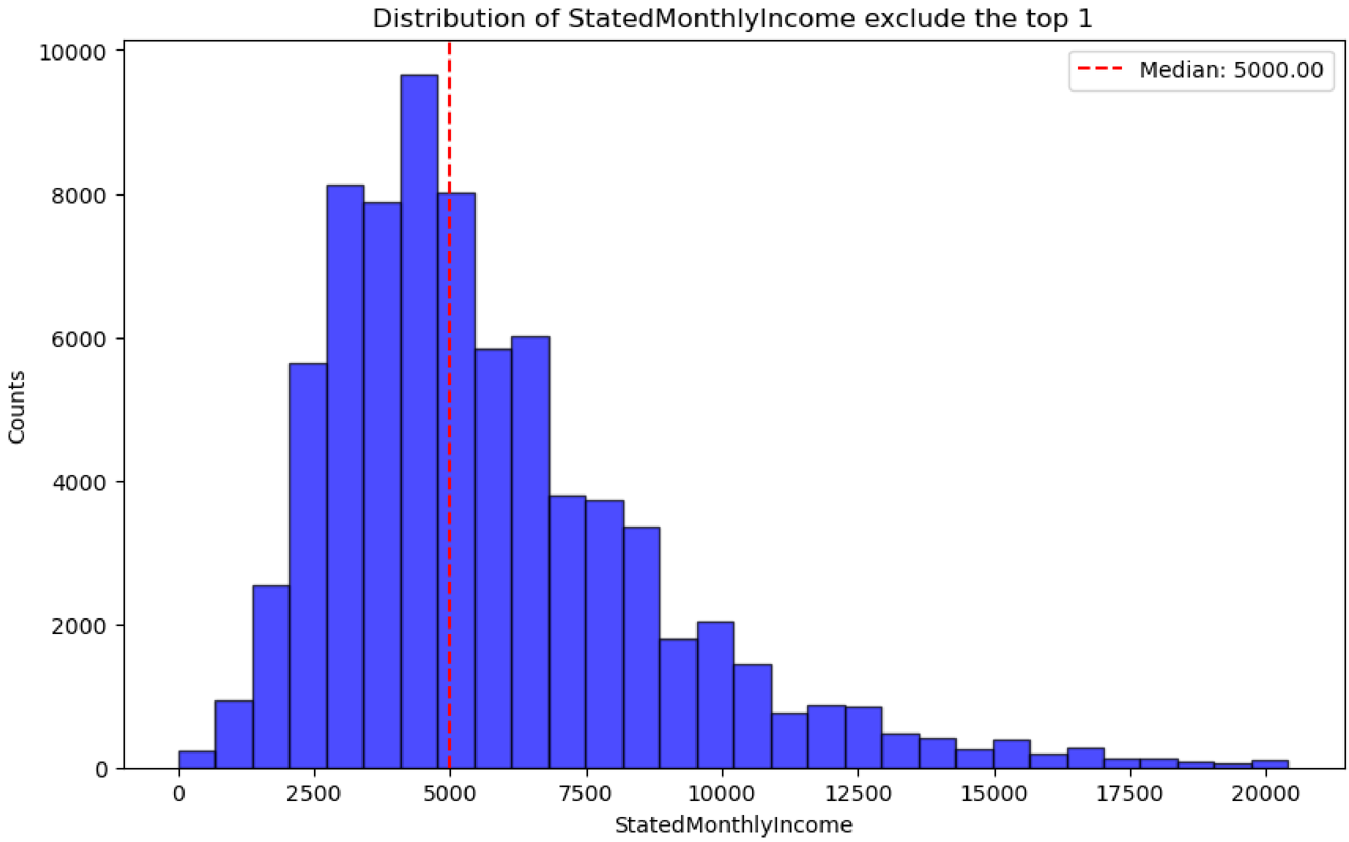
```
plt.xscale('log')
plt.show()
```



- **StatedMonthlyIncome** is higly skewed to the right.
- 99% of data are within 0 - 20526

In [50]:

```
# Distribution of StatedMonthlyIncome exclude the top 1%
fig, ax = plt.subplots(figsize=[10,6])
ax.hist(df_mod['StatedMonthlyIncome'], range=(0,df_mod['StatedMonthlyIncome'].quantil
        bins=30, edgecolor='black', color='Blue', alpha=0.7)
# Add a vertical line at the median (50th percentile) in red
median=df_mod['StatedMonthlyIncome'].median()
plt.axvline(df_mod['StatedMonthlyIncome'].median(), color='red', linestyle='--', labe

# Remove the legend
plt.legend().set_visible(True)
plt.title("Distribution of StatedMonthlyIncome exclude the top 1")
plt.xlabel('StatedMonthlyIncome')
plt.ylabel('Counts')
plt.show()
```



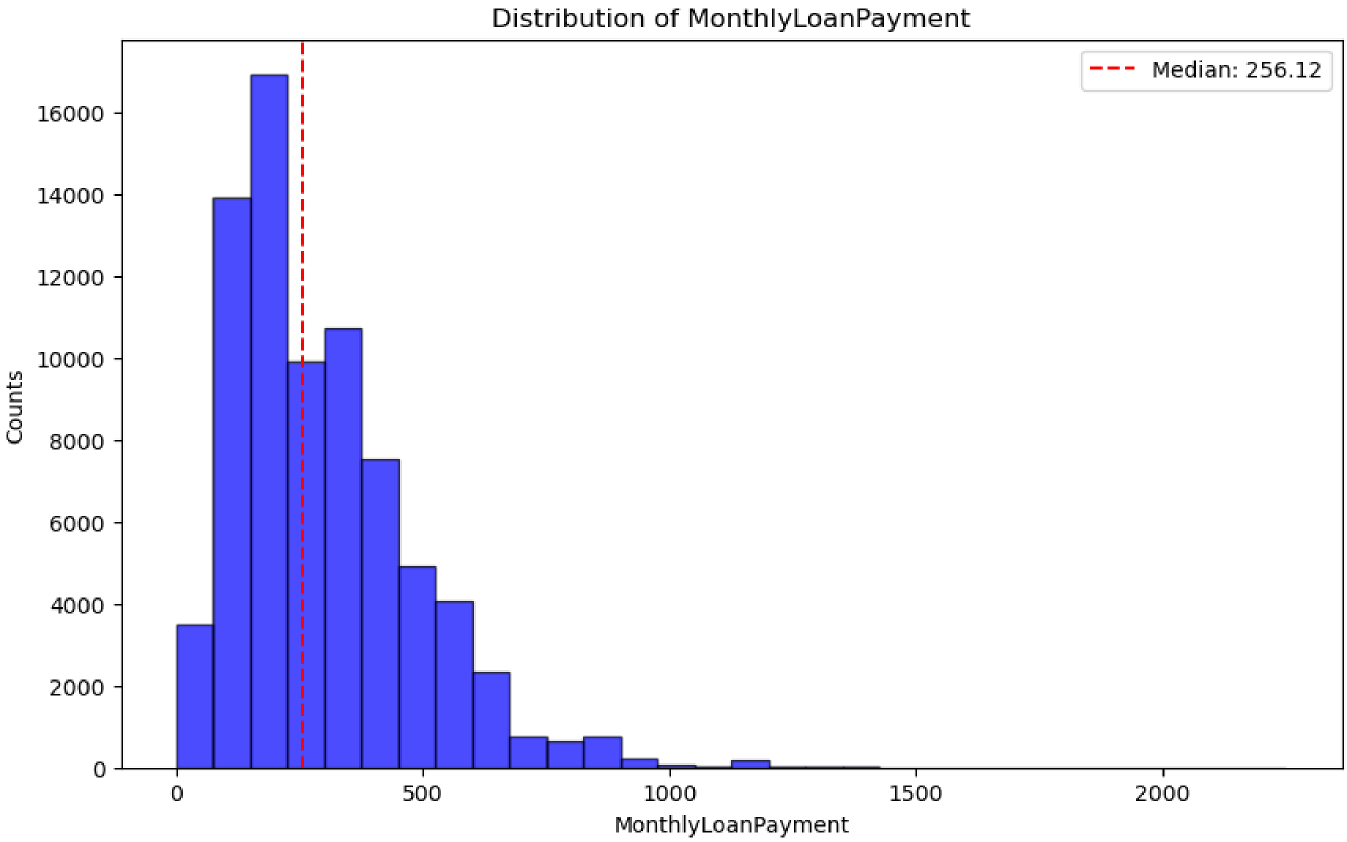
> - Most borrowers have a **StatedMonthlyIncome** that is less than 10k
> - There is a peak slightly below 5k

MonthlyLoanPayment

In [51]:
```
df_mod.MonthlyLoanPayment.describe()
```

Out[51]:
```
count    76768.000000
mean       295.275039
std        189.109061
min          0.000000
25%        158.330000
50%        256.120000
75%        392.010000
max       2251.510000
Name: MonthlyLoanPayment, dtype: float64
```

In [52]:
```
fig, ax = plt.subplots(figsize=[10,6])
ax.hist(df_mod['MonthlyLoanPayment'],
        bins=30, edgecolor='black', color='Blue', alpha=0.7)
# Add a vertical line at the median (50th percentile) in red
median=df_mod['MonthlyLoanPayment'].median()
plt.axvline(df_mod['MonthlyLoanPayment'].median(), color='red', linestyle='--', label

# Remove the legend
plt.legend().set_visible(True)
plt.title("Distribution of MonthlyLoanPayment")
plt.xlabel('MonthlyLoanPayment')
plt.ylabel('Counts')
plt.show()
```



> - Majority of **MonthlyLoanPayment** are below 500 USD

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

> - There are 46 data entry with maximun DebtToIncomeRatio = 10.01 while 99% of data have DebtToIncomeRatio < 1.
> - Consider that there are 46 data entry, it should not be manual error so I will not remove the data.

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?

> - StatedMonthlyIncome is higly skewed to the right while 99% of data entry are within 0 to 20526. I did not perform any adjust to the dataset but look into the majority data.

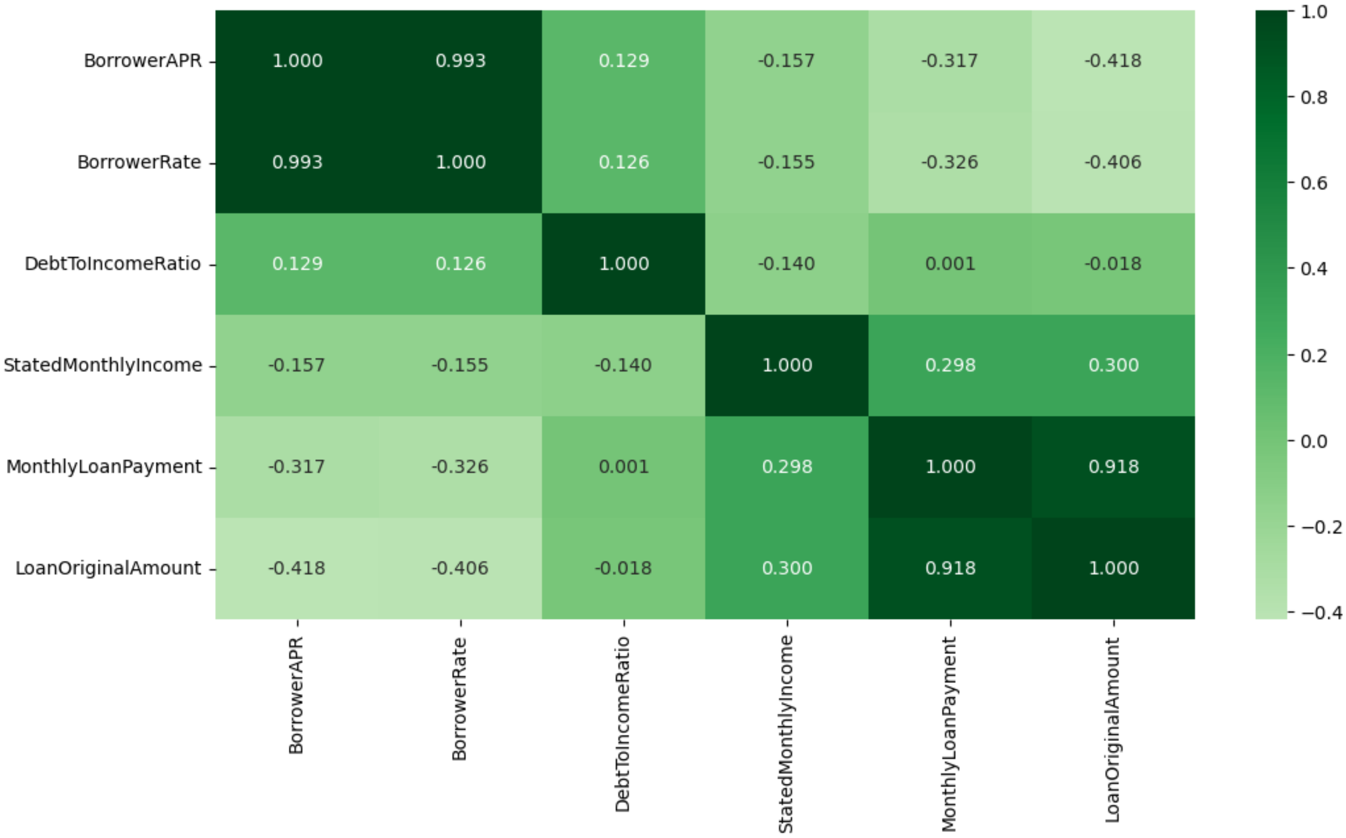
Bivariate Exploration

Examining the Relationship Between Essential Features. To facilitate visualization, I will classify the variables of concern into Numeric and Categorical categories."

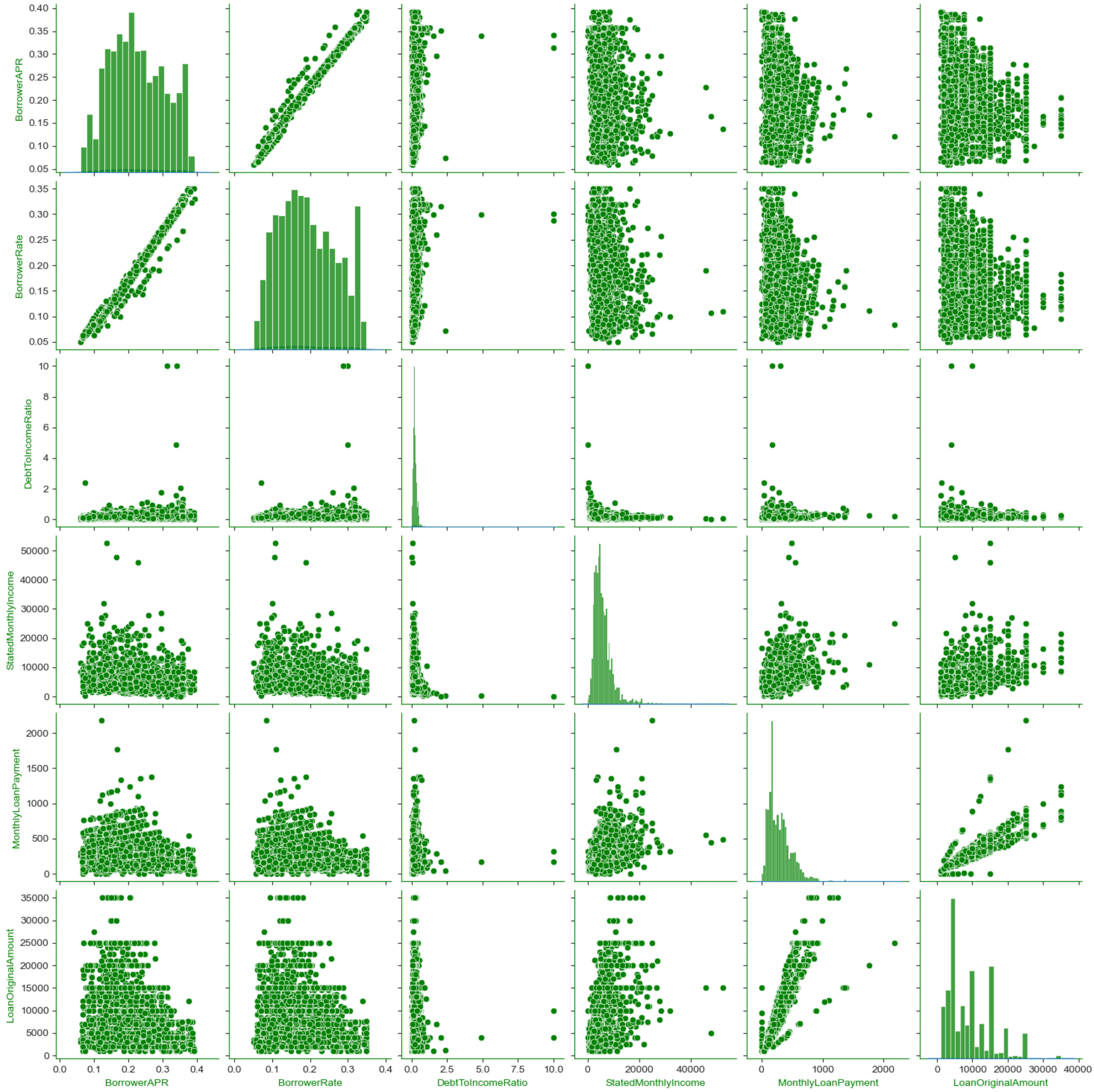
In [53]:
```
# Separate the dataset into numeric and categorical features
numeric_features = ['BorrowerAPR', 'BorrowerRate', 'DebtToIncomeRatio',
                    'StatedMonthlyIncome', 'MonthlyLoanPayment',
                    'LoanOriginalAmount']

categorical_features = ['Term', 'LoanStatus', 'EmploymentStatus',
                        'IsBorrowerHomeowner', 'IncomeRange',
                        'ProsperRating', 'ListingCategory']
```

1. Quantitative Vs. Quantitave

In [54]:
```python
# Correlation plot for numeric features
plt.figure(figsize=[12, 6])

# Heatmap of the correlation matrix, annotating values.
sns.heatmap(df_mod[numeric_features].corr(), annot=True,
            fmt='.3f', center=0, cmap='Greens')

# Display the plot
plt.show()
```



In [55]:
```python
# Take a sample of 3000 loans to plot
samples = np.random.choice(df_mod.index, 3000, replace=False)
loans_sample = df_mod.loc[samples, :]

# Plot pairwise relationships between all the numeric variables of interest
sns.set_style("ticks")
g = sns.pairplot(data=loans_sample, vars=numeric_features, diag_kind='kde')

# Set the color to green for all the plots
for ax in g.axes.flat:
    ax.spines['bottom'].set_color('green')
    ax.spines['top'].set_color('green')
    ax.spines['left'].set_color('green')
    ax.spines['right'].set_color('green')
    ax.xaxis.label.set_color('green')
    ax.yaxis.label.set_color('green')

# Add a grid to the plots
g.map_upper(sns.scatterplot, color='green')
g.map_lower(sns.scatterplot, color='green')
g.map_diag(sns.histplot, color='green')

plt.show()
```



- **Borrower APR** and **Borrower Rate** exhibit a robust positive correlation, with a coefficient of 0.99, which is expected since a higher APR typically leads to borrowers paying a greater amount of interest on their loans.

- **Borrower APR** and the **Loan Original Amount** demonstrate an inverse correlation with a coefficient of -0.32. The accompanying scatter plot visually corroborates this negative association, illustrating that as the loan amount rises, the APR tends to decline.

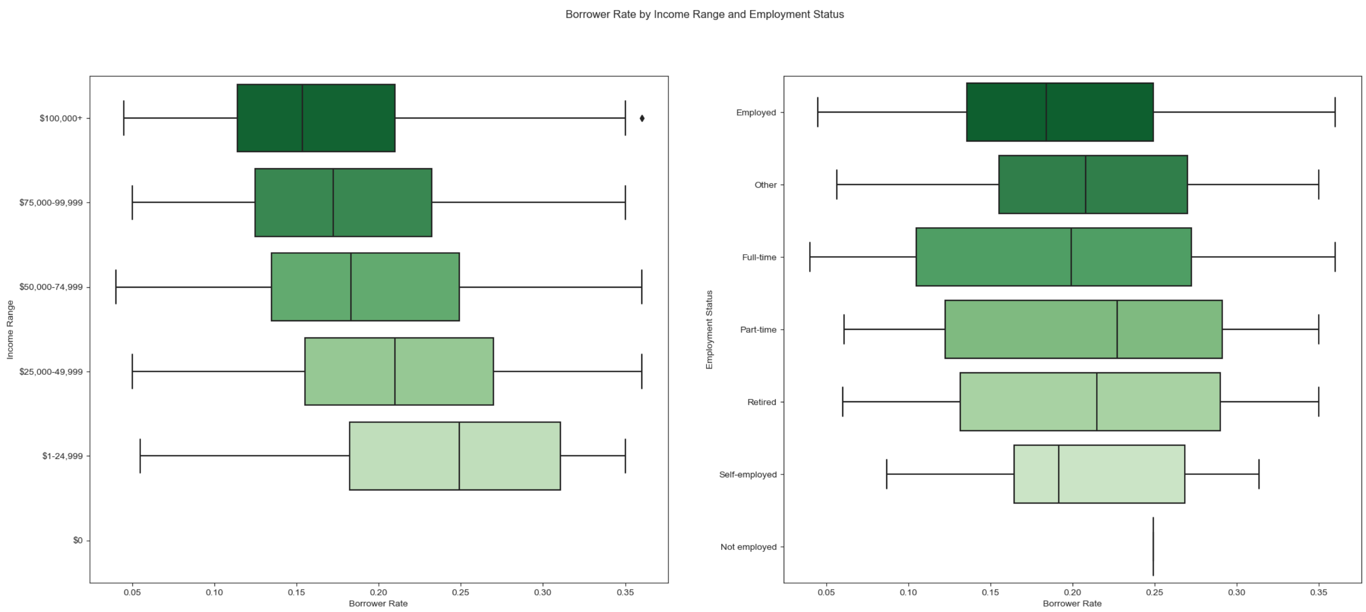
- **Loan Original Amount** and **Stated Monthly Income** shows a slight positive correlation, as denoted by a correlation coefficient of 0.30.

- As the **Loan Original Amount** rises, there is a concurrent growth in the **Monthly Loan Payment**.

2. Quantitative vs. Qualitative

In [56]:
```
# Set a larger figure size for subplots
plt.figure(figsize=[25, 10])
```

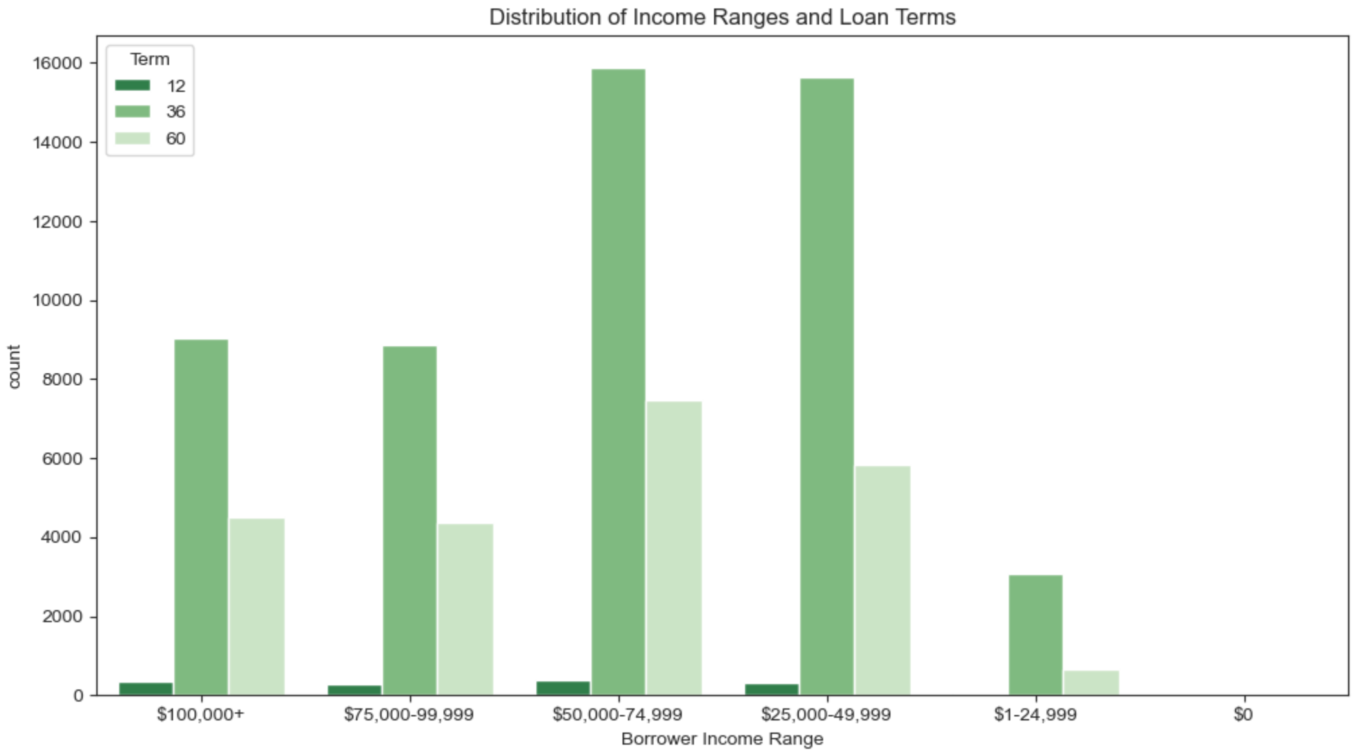
```
# Subplot 1
plt.subplot(1, 2, 1)
sns.boxplot(data=df_mod, y='IncomeRange', x='BorrowerRate', palette='Greens_r')
plt.ylabel('Income Range')
plt.xlabel('Borrower Rate')

# Subplot 2
plt.subplot(1, 2, 2)
sns.boxplot(data=df_mod, y='EmploymentStatus', x='BorrowerRate', palette='Greens_r')
plt.ylabel('Employment Status')
plt.xlabel('Borrower Rate')

plt.suptitle('Borrower Rate by Income Range and Employment Status')
plt.show()
```

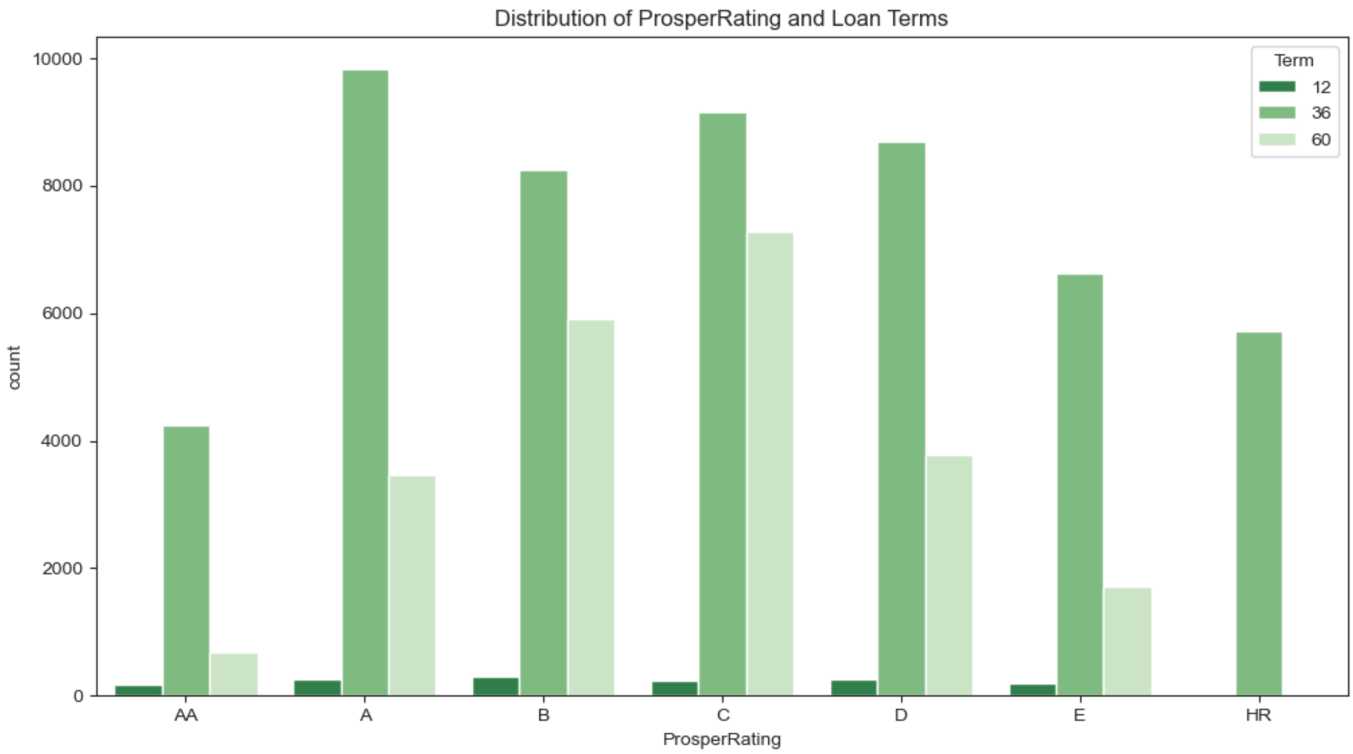


- The boxplots reveal that the median borrower rate decrease with higher income leves. The income range of 1-24,999k exhibits the highest median borrower rate.
- Within the employment status categories, the "unemployed" group displays the highest median borrower rate, indicating that unemployed individuals tend to incur higher interest costs on their loans.

3. Qualitative vs. Qualitiative

In [57]:
```
# Clustered bar chart of income range and loan term
plt.figure(figsize=[12, 14])

# Subplot 1: Income range and term distribution
plt.subplot(2, 1, 1)
sns.countplot(data=df_mod, x='IncomeRange', hue='Term', palette='Greens_r')
plt.title('Distribution of Income Ranges and Loan Terms')
plt.xlabel('Borrower Income Range')

# Subplot 2: Prosper rating and term distribution
plt.subplot(2, 1, 2)
sns.countplot(data=df_mod, x='ProsperRating', hue='Term', palette='Greens_r',order=or
plt.title('Distribution of ProsperRating and Loan Terms')
plt.xlabel('ProsperRating');
```





- Borrowers with an income range of `$50,000` - `$74,999` predominantly opt for 36-month loan terms, followed closely by those with income ranges of `$25,000` - `$49,999`. Borrowers with no reported income primarily choose 36-month loan terms.
- Among Prosper rating categories, B and C ratings have a higher proportion of 60-month loans. In contrast, HR-rated borrowers exclusively opt for 36-month terms, while A-rated borrowers show a preference for 36-month loans as well.

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?

> - **Borrower APR** and **Borrower Rate** exhibit a robust positive correlation, with a coefficient of 0.99, which is expected since a higher APR typically leads to borrowers paying a greater amount of interest on their loans.

> - **Borrower APR** and the **Loan Original Amount** demonstrate an inverse correlation with a coefficient of -0.32. The accompanying scatter plot visually corroborates this negative association, illustrating that as the loan amount rises, the APR tends to decline.

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

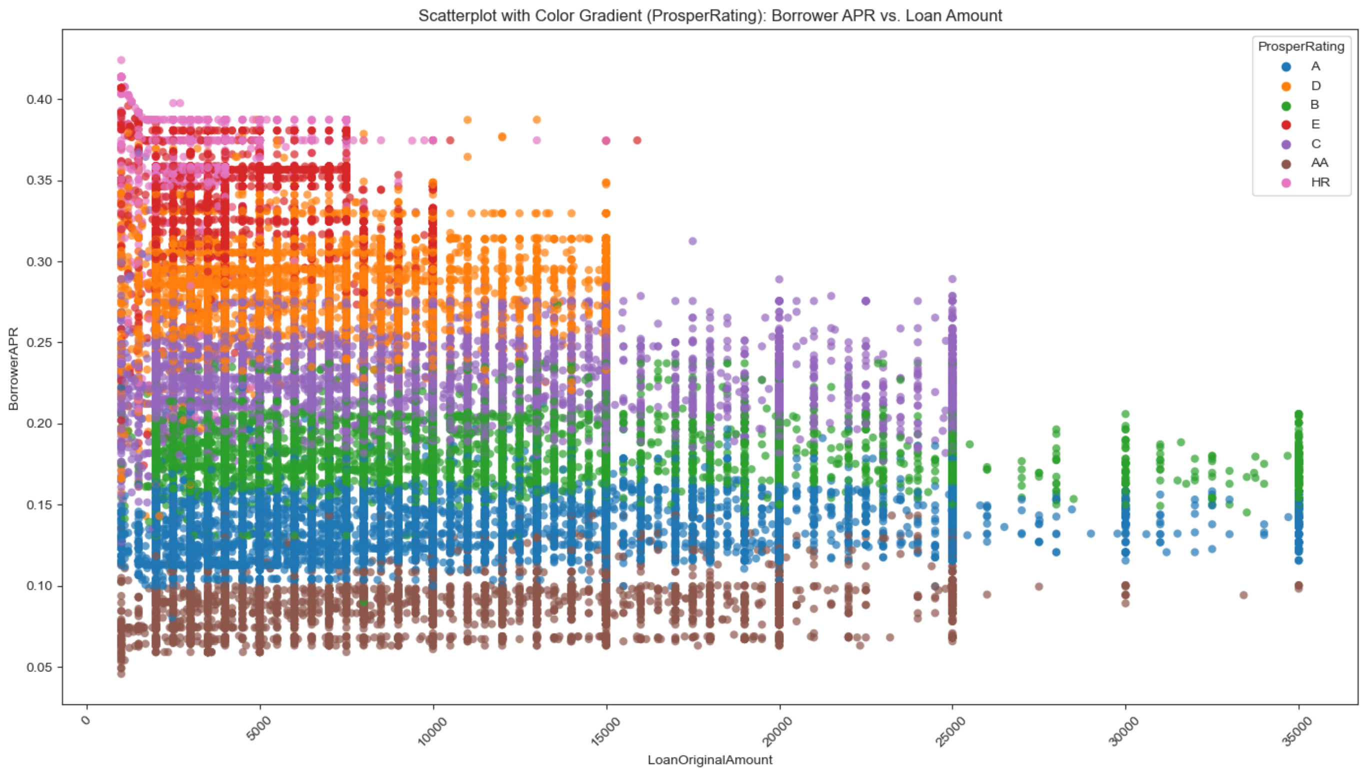
> - As the **Loan Original Amount** rises, there is a concurrent growth in the **Monthly Loan Payment** (positive correlation).
> - **Loan Original Amount** and **Stated Monthly Income** shows a slight positive correlation, as denoted by a correlation coefficient of 0.30.
> - Borrowers within the income range of `$50,000` `$74,999` primarily opt for 36-month loan terms, followed by those in the `$25,000` - `$49,999` income range. Borrowers with zero reported income exclusively select 36-month terms.
> - Notably, individuals categorized as unemployed demonstrate the highest median borrower rate, signifying that they pay higher interest rates on loans.

Multivariate Exploration

In [58]:
```
df_mod.BorrowerAPR.describe()
```

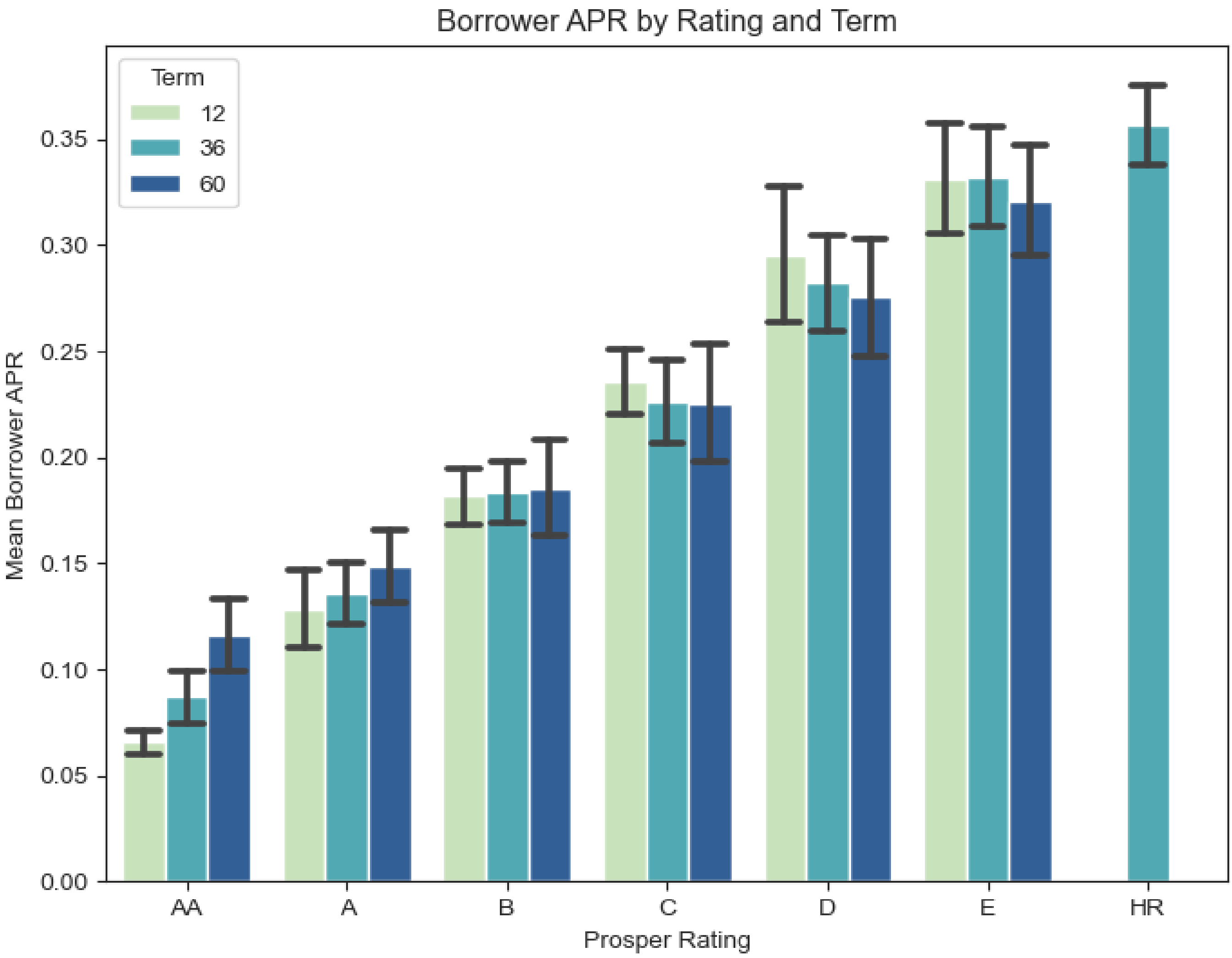
Out[58]:
```
count    76768.000000
mean         0.223978
std          0.079291
min          0.045830
25%          0.162590
50%          0.215660
75%          0.287800
max          0.423950
Name: BorrowerAPR, dtype: float64
```

In [59]:
```
# Create a scatter plot to show the distribution of LoanOriginalAmount by BorrowerAPR
plt.figure(figsize=(14, 8))
sns.scatterplot(data=df_mod, x='LoanOriginalAmount', y='BorrowerAPR',hue='ProsperRati
               alpha=0.7, edgecolor='none')

plt.xlabel('LoanOriginalAmount')
plt.ylabel('BorrowerAPR')
plt.title('Scatterplot with Color Gradient (ProsperRating): Borrower APR vs. Loan Amo
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



- As the **ProsperRating** improves, the **LoanOriginalAmount** tends to increase.
- With higher **ProsperRating**, borrowers generally experience lower **BorrowerAPR**.
- The correlation between **LoanOriginalAmount** and **BorrowerAPR** transitions from negative to slightly positive as **ProsperRating** improve.

In [60]:
```
# Create a figure for the point plot
fig = plt.figure(figsize=[8, 6])

# Create a point plot to visualize Borrower APR across Prosper Rating and Loan Term
sns.barplot(data=df_mod, x='ProsperRating', y='BorrowerAPR', hue='Term', order=order,
              palette='YlGnBu', errorbar='sd', capsize=0.2)

# Set the title and labels for the plot
plt.title('Borrower APR by Rating and Term')
plt.xlabel('Prosper Rating')
plt.ylabel('Mean Borrower APR');
```



> - **Higher ProsperRating (AA-B)** experience increasing APRs as the **Term** extends.
> - Conversely, **lower ProsperRating (C-HR)** observe decreasing APRs as the **Term** extends.

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?

> - As borrowers have better ratings, we observe that the loan amount tends to increase, while the borrower APR decreases. This suggests that borrowers with higher ratings receive larger loans at more favorable interest rates.
> - When we specifically analyze the relationship between loan term and APR, interesting patterns emerge. Borrowers with excellent ratings (AA-B) experience increasing APRs as the loan term extends, which may seem counterintuitive. However, for lower-rated borrowers (C-HR), APRs tend to decrease as the loan term becomes longer. This suggests that loan terms interact differently with borrower ratings, potentially reflecting risk assessment variations by lenders.

Were there any interesting or surprising interactions between features?

> - Notably, we found a negative correlation between BorrowerAPR and LoanOriginalAmount. This implies that as borrowers request larger loan amounts, the associated APR tends to decrease. This observation could be significant for both lenders and borrowers, as it indicates a potential benefit for borrowers seeking higher loan amounts in terms of lower interest rates.

- These relationships and interactions provide valuable insights into how borrower ratings, loan terms, loan amounts, and APRs are interconnected, allowing for a more comprehensive understanding of the lending dynamics in the dataset.

Conclusions

- **Borrower APR** and **Borrower Rate** exhibit a robust positive correlation, with a coefficient of 0.99, which is expected since a higher APR typically leads to borrowers paying a greater amount of interest on their loans.
- **Borrower APR** and the **Loan Original Amount** demonstrate an inverse correlation with a coefficient of -0.32. The accompanying scatter plot visually corroborates this negative association, illustrating that as the loan amount rises, the APR tends to decline.
- **Higher ProsperRating (AA-B)** experience increasing **Borrower APR** as the **Term** extends.
- Conversely, **lower ProsperRating (C-HR)** observe decreasing **Borrower APR** as the **Term** extends.
- With higher **ProsperRating**, borrowers generally experience lower **BorrowerAPR**.
- The correlation between **LoanOriginalAmount** and **BorrowerAPR** transitions from negative to slightly positive as **ProsperRating** improve.

In [ ]:

In [ ]: