EXPLORING PROSPER LOAN DATASET

by - NDUKA, Anthony

Table of Contents

1.0 Introduction

> As the British Mathematician, Clive Humby, would say, “Data is the new oil.” As such, data analysis has become a vital venture which no business can overlook. As a subject matter, Data analysis has been described as a process of inspecting, cleaning (wrangling), transforming and visualizing data with the view of deducing useful pieces of information that aid decision making. By analysis, it refers to diving of a dataset into units of sets of units for individual exploration, and correlations between these units. It is therefore necessary that Prosper Loan Company analysis their dataset for scientific and informed decision making.

1.1 Objective

> The purpose of this project is to show mastery in using Python for data analysis through the use of summary statistics and data visualizations in describing the features of variables that might impact loan status. Furthermore, it seeks to get some

> insight into the correlations between various variables of Dataset and also aims at solving some vital queries in the real-world of data. To do this, we used univariate, bivariate and multivariate analysis – their statistics and visualizations, to elucidate correlations between various variables of the dataset.

1.2 About Dataset

> Prosper Marketplace – a loan company, was founded in 2005 as the first peer-to-peer lending marketplace in the United States. Since then, Prosper has facilitated more than 23 billion USD in loans to more than 1,400,000 people. Borrowers apply online for a fixed-rate, fixed-term loan between 2,000 USD and 50,000 USD. Individuals and institutions can invest in the loans and earn attractive returns. The Prosper Loan Dataset used in this project, contains 113937 loan listing with 81 variables showing various relationships between different loan variables.

2.0 Importing Libraries

> To start with, we have to import relevant libraries and packages that will be help us vectorize and visualise our dataset. Thus, import the **Pandas** and **Numpy** for the vectors and then **Matplotlib**, **Plotly**, and **Seaborn** for the visuals.

In [3]:
```
# import all packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

3.0 Preliminary Wrangling

> Here we load our dataset `Prosper Loan.csv` and then read it into dataframe

In [4]:
```
df=pd.read_csv('Prosper Loan.csv')
```

In [5]:
```
#Displaying a purview of the dataframe
df
```

Out[5]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	LoanStatus	ClosedDate	BorrowerAPR
0	1021339766868145413AB3B	193129	09:29.3	C	36	Completed	14/08/2009 00:00	0.16516
1	10273602499503308B223C1	1209647	28:07.9	NaN	36	Current	NaN	0.12016
2	0EE9337825851032864889A	81716	00:47.1	HR	36	Completed	17/12/2009 00:00	0.28269
3	0EF5356002482715299901A	658116	02:35.0	NaN	36	Current	NaN	0.12528
4	0F023589499656230C5E3E2	909464	38:39.1	NaN	36	Current	NaN	0.24614
...	...	...	...	...	...	...	...	...
113932	E6D9357655724827169606C	753087	55:02.7	NaN	36	Current	NaN	0.22354
113933	E6DB353036033497292EE43	537216	42:55.3	NaN	36	FinalPaymentInProgress	NaN	0.13220
113934	E6E13596170052029692BB1	1069178	49:12.7	NaN	60	Current	NaN	0.23984
113935	E6EB3531504622671970D9E	539056	18:26.6	NaN	60	Completed	13/08/2013 00:00	0.28408
113936	E6ED3600409833199F711B7	1140093	27:37.7	NaN	36	Current	NaN	0.13189

113937 rows × 81 columns

In [6]:
```
#Data structures are “containers” that organize and group data according to type.
# Herein, we explore the structure of our dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 81 columns):
 #   Column                               Non-Null Count   Dtype  
---  ------                               --------------   -----  
 0   ListingKey                           113937 non-null  object 
 1   ListingNumber                        113937 non-null  int64  
 2   ListingCreationDate                  113937 non-null  object 
 3   CreditGrade                          28953 non-null   object 
 4   Term                                 113937 non-null  int64  
 5   LoanStatus                           113937 non-null  object 
 6   ClosedDate                           55089 non-null   object 
 7   BorrowerAPR                          113912 non-null  float64
 8   BorrowerRate                         113937 non-null  float64
 9   LenderYield                          113937 non-null  float64
 10  EstimatedEffectiveYield              84853 non-null   float64
 11  EstimatedLoss                        84853 non-null   float64
 12  EstimatedReturn                      84853 non-null   float64
 13  ProsperRating (numeric)              84853 non-null   float64
 14  ProsperRating (Alpha)                84853 non-null   object 
 15  ProsperScore                         84853 non-null   float64
 16  ListingCategory (numeric)            113937 non-null  int64  
 17  BorrowerState                        108422 non-null  object 
 18  Occupation                           110349 non-null  object 
 19  EmploymentStatus                     111682 non-null  object 
 20  EmploymentStatusDuration             106312 non-null  float64
 21  IsBorrowerHomeowner                  113937 non-null  bool   
 22  CurrentlyInGroup                     113937 non-null  bool   
 23  GroupKey                             13341 non-null   object 
 24  DateCreditPulled                     113937 non-null  object 
 25  CreditScoreRangeLower                113346 non-null  float64
 26  CreditScoreRangeUpper                113346 non-null  float64
 27  FirstRecordedCreditLine              113240 non-null  object 
 28  CurrentCreditLines                   106333 non-null  float64
 29  OpenCreditLines                      106333 non-null  float64
 30  TotalCreditLinespast7years           113240 non-null  float64
 31  OpenRevolvingAccounts                113937 non-null  int64  
 32  OpenRevolvingMonthlyPayment          113937 non-null  int64  
 33  InquiriesLast6Months                 113240 non-null  float64
 34  TotalInquiries                       112778 non-null  float64
 35  CurrentDelinquencies                 113240 non-null  float64
 36  AmountDelinquent                     106315 non-null  float64
 37  DelinquenciesLast7Years              112947 non-null  float64
 38  PublicRecordsLast10Years             113240 non-null  float64
 39  PublicRecordsLast12Months            106333 non-null  float64
 40  RevolvingCreditBalance               106333 non-null  float64
 41  BankcardUtilization                  106333 non-null  float64
 42  AvailableBankcardCredit              106393 non-null  float64
 43  TotalTrades                          106393 non-null  float64
 44  TradesNeverDelinquent (percentage)   106393 non-null  float64
 45  TradesOpenedLast6Months              106393 non-null  float64
 46  DebtToIncomeRatio                    105383 non-null  float64
 47  IncomeRange                          113937 non-null  object 
 48  IncomeVerifiable                     113937 non-null  bool   
 49  StatedMonthlyIncome                  113937 non-null  float64
 50  LoanKey                              113937 non-null  object 
 51  TotalProsperLoans                    22085 non-null   float64
 52  TotalProsperPaymentsBilled           22085 non-null   float64
 53  OnTimeProsperPayments                22085 non-null   float64
 54  ProsperPaymentsLessThanOneMonthLate  22085 non-null   float64
 55  ProsperPaymentsOneMonthPlusLate      22085 non-null   float64
 56  ProsperPrincipalBorrowed             22085 non-null   float64
 57  ProsperPrincipalOutstanding          22085 non-null   float64
 58  ScorexChangeAtTimeOfListing          18928 non-null   float64
 59  LoanCurrentDaysDelinquent            113937 non-null  int64  
 60  LoanFirstDefaultedCycleNumber        16952 non-null   float64
 61  LoanMonthsSinceOrigination           113937 non-null  int64  
 62  LoanNumber                           113937 non-null  int64  
 63  LoanOriginalAmount                   113937 non-null  int64  
 64  LoanOriginationDate                  113937 non-null  object 
 65  LoanOriginationQuarter               113937 non-null  object 
 66  MemberKey                            113937 non-null  object 
 67  MonthlyLoanPayment                   113937 non-null  float64
 68  LP_CustomerPayments                  113937 non-null  float64
 69  LP_CustomerPrincipalPayments         113937 non-null  float64
 70  LP_InterestandFees                   113937 non-null  float64
 71  LP_ServiceFees                       113937 non-null  float64
 72  LP_CollectionFees                    113937 non-null  float64
 73  LP_GrossPrincipalLoss                113937 non-null  float64
 74  LP_NetPrincipalLoss                  113937 non-null  float64
 75  LP_NonPrincipalRecoverypayments      113937 non-null  float64
 76  PercentFunded                        113937 non-null  float64
 77  Recommendations                      113937 non-null  int64  
 78  InvestmentFromFriendsCount           113937 non-null  int64  
 79  InvestmentFromFriendsAmount          113937 non-null  float64
 80  Investors                            113937 non-null  int64  
dtypes: bool(3), float64(49), int64(12), object(17)
memory usage: 68.1+ MB
```

3.1 Dataset Structure

> Insights from the dataset show that there are **113937 entries** and **81 columns** in or dataset. Its data types are Boolean - 4%, Float - 60%, Obect - 21% and Integers - 15%. This makes our numberic data to be 75% of the dataset.

3.2 Features of Interest

> After an analytical look at the dataset, there are some features of interest that could be dedcued from the dataset above.This has to do with what motivates a client's(borrower) interest to seek a loan and also some factors that might affect favorable loan scheme for a client. Thus, we consider some factors such as these below:
>
> - **Term:** This is the length of the loan expressed in months.
> - **LoanStatus:** This has to do with the current status of the loan. It could be: Cancelled, Chargedoff, Completed, Current, Defaulted, FinalPaymentInProgress, PastDue. The PastDue status will be accompanied by a delinquency bucket.
> - **ListingCreationDate:** It has to do with the date the loan listing was created.
> - **ListingCategory (numeric):** 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.
> - **EmploymentStatusDuration:** The length in months of the employment status at the time the listing was created.
> - **IncomeRange:** This is 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.
> - **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.
> - **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.
> - **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.
> - **ProsperRating (Alpha):** The Prosper Rating assigned at the time the listing was created between AA - HR. Applicable for loans originated after July 2009.
> - **BorrowerAPR:** The Borrower's Annual Percentage Rate (APR) for the loan.
>
> **Note:** Efforts will be made to measure loan favourability using the **Prosper Rating** and the **Borrower's APR** as they will help support our investigation into the above features of interest.

4.0 Assessment of Dataset

> Here we will undergo a data exploration of our dataframe with keen attention to our features of interest.To do that, we will first create a listing that contains our features of intetrest as already listed in 3.2 above. Effort is also made to assess the dataset in Excel spreedsheet for more accurate observation.

In [7]:

```
# Creating a sub-dataset keeping only the main features of interest.

main_features = ['Term','LoanStatus','ListingCreationDate','ListingCategory (numeric)','EmploymentStatus',
                 'EmploymentStatusDuration','IncomeRange','IncomeVerifiable','IsBorrowerHomeowner','DebtToIncom
                 'StatedMonthlyIncome','LoanOriginalAmount','ProsperScore','ProsperRating (Alpha)','BorrowerAPR
```

In [8]:

```
df[main_features].sample(n = 10)
```

Out[8]:

	Term	LoanStatus	ListingCreationDate	ListingCategory (numeric)	EmploymentStatus	EmploymentStatusDuration	IncomeRange	IncomeVerifiabl
91849	36	Current	47:45.3	1	Employed	153.0	$50,000-74,999	Tru
93836	36	Current	27:25.2	7	Employed	25.0	$25,000-49,999	Tru
40234	36	Current	46:05.9	1	Employed	159.0	$50,000-74,999	Tru
108206	36	Completed	33:47.2	0	Not available	NaN	Not displayed	Tru
75443	36	Current	12:40.4	1	Employed	32.0	$25,000-49,999	Tru
78210	36	Completed	34:21.1	7	Retired	232.0	$25,000-49,999	Tru
83869	36	Current	42:52.1	2	Self-employed	183.0	$75,000-99,999	Fals
112511	60	Completed	53:15.9	1	Employed	91.0	$100,000+	Tru
27224	60	Current	58:59.8	1	Employed	106.0	$75,000-99,999	Tru
35385	36	Completed	47:12.0	3	Full-time	46.0	$25,000-49,999	Tru

In [9]:

```
# Examining a concise summary of our main_features subset:
df[main_features].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 15 columns):
 #   Column                     Non-Null Count   Dtype  
---  ------                     --------------   -----  
 0   Term                       113937 non-null  int64  
 1   LoanStatus                 113937 non-null  object 
 2   ListingCreationDate        113937 non-null  object 
 3   ListingCategory (numeric)  113937 non-null  int64  
 4   EmploymentStatus           111682 non-null  object 
 5   EmploymentStatusDuration   106312 non-null  float64
 6   IncomeRange                113937 non-null  object 
 7   IncomeVerifiable           113937 non-null  bool   
 8   IsBorrowerHomeowner        113937 non-null  bool   
 9   DebtToIncomeRatio          105383 non-null  float64
 10  StatedMonthlyIncome        113937 non-null  float64
 11  LoanOriginalAmount         113937 non-null  int64  
 12  ProsperScore               84853 non-null   float64
 13  ProsperRating (Alpha)      84853 non-null   object 
 14  BorrowerAPR                113912 non-null  float64
dtypes: bool(2), float64(5), int64(3), object(5)
memory usage: 11.5+ MB
```

In [10]:

```
#Checking for duplicates in our subset
duplicates = df[main_features].duplicated().sum()
print('There are {} duplicate records in the dataset'.format(duplicates))
```

```
There are 0 duplicate records in the dataset
```

In [11]:

```
# View the disctiptive statistics
df[main_features].describe()
```

Out[11]:

	Term	ListingCategory (numeric)	EmploymentStatusDuration	DebtToIncomeRatio	StatedMonthlyIncome	LoanOriginalAmount	ProsperSc
count	113937.000000	113937.000000	106312.000000	105383.000000	1.139370e+05	113937.00000	84853.000
mean	40.830248	2.774209	96.071582	0.275947	5.608026e+03	8337.01385	5.950
std	10.436212	3.996797	94.480605	0.551759	7.478497e+03	6245.80058	2.376
min	12.000000	0.000000	0.000000	0.000000	0.000000e+00	1000.00000	1.000
25%	36.000000	1.000000	26.000000	0.140000	3.200333e+03	4000.00000	4.000
50%	36.000000	1.000000	67.000000	0.220000	4.666667e+03	6500.00000	6.000
75%	36.000000	3.000000	137.000000	0.320000	6.825000e+03	12000.00000	8.000
max	60.000000	20.000000	755.000000	10.010000	1.750003e+06	35000.00000	11.000

4.1 Summary of Assessment

> 1. Main features need to be isolated from the dataset.
>
> 2. There are 0 duplicate records in the dataset.

3. **ListingCategory (numeric)** and **ProsperRating (Alpha)** can be reassigned with a different column names for easy analysis of dataset.

4. The numeric information in **ListingCategory (numeric)** could be changed to pandas object to reflect actual reasons for the loan. Accuring from the information in the metadata dictionary.

5. **ListingCreationDate** is stored with the wrong datatype. It should be a pandas datetime object.

6. The *Not employed* entries in **IncomeRange** could be replaced with **0**.

7. The **BorrowerAPR**, **DebtToIncomeRatio**, **EmploymentStatus**, **ProsperRating (Alpha)**,**EmploymentStatusDuration** columns contain null values.

8. **ProsperRating** and **IncomeRange** could be converted to ordinal categorical variables.

5.0 Data Cleaning

> We will begin by creating a copy of the original dataframe, and then address some points identified in the cleaning process.

5.1 Create a copy of the original dataframe

In [12]:
```
clean_df = df.copy()
```

5.2 Isolate main features from the dataframe

In [13]:
```
# Filter out the key features from the original dataframe
clean_df = clean_df[main_features]


# Verify the changes made
assert len(clean_df.columns) == len(main_features)
```

In [15]:
```
clean_df
```

Out[15]:

	Term	LoanStatus	ListingCreationDate	ListingCategory (numeric)	EmploymentStatus	EmploymentStatusDuration	IncomeRange	Inco
0	36	Completed	09:29.3	0	Self-employed	2.0	$25,000-49,999	
1	36	Current	28:07.9	2	Employed	44.0	$50,000-74,999	
2	36	Completed	00:47.1	0	Not available	NaN	Not displayed	
3	36	Current	02:35.0	16	Employed	113.0	$25,000-49,999	
4	36	Current	38:39.1	2	Employed	44.0	$100,000+	
...	...	...	...	...	...	...	...	
113932	36	Current	55:02.7	1	Employed	246.0	$50,000-74,999	
113933	36	FinalPaymentInProgress	42:55.3	7	Employed	21.0	$75,000-99,999	
113934	60	Current	49:12.7	1	Employed	84.0	$25,000-49,999	
113935	60	Completed	18:26.6	2	Full-time	94.0	$25,000-49,999	
113936	36	Current	27:37.7	1	Employed	244.0	$50,000-74,999	

113937 rows × 15 columns

5.3 Reassigning the ListingCategory (numeric) and ProsperRating (Alpha) columns to a different name for easy of analysis.

In [16]:
```
# Rename the columns
clean_df = clean_df.rename(columns = {'ListingCategory (numeric)': 'ListingCategory', 'ProsperRating (Alpha)':

# verify code results
for col_name in ['ListingCategory', 'ProsperRating']:
    assert col_name in clean_df.columns
```

5.4 Putting the ListingCategory column to the right category titles

5.4 Putting the ListingCategory column to the right category titles

In [17]:
```
category_lisitng = {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 or Dental', 16 : 'Motorcycle
                   17 : 'RV', 18 : 'Taxes', 19 : 'Vacation', 20 : 'Wedding Loans'}

clean_df.ListingCategory = clean_df.ListingCategory.map(category_lisitng)

# Check the results
clean_df.ListingCategory.unique()
```

Out[17]:
```
array(['Not Available', 'Home Improvement', 'Motorcycle',
       'Debt Consolidation', 'Other', 'Household Expenses', 'Auto',
       'Medical or Dental', 'Wedding Loans', 'Vacation', 'Business',
       'Taxes', 'Baby & Adoption', 'Personal Loan', 'Engagement Ring',
       'Large Purchases', 'Student Use', 'Boat', 'RV',
       'Cosmetic Procedure', 'Green Loans'], dtype=object)
```

5.5 Converting ListingCreationDate to DateTime object

In [18]:
```
clean_df['ListingCreationDate'] = pd.to_datetime((clean_df.ListingCreationDate), errors ='coerce')
clean_df.dtypes[0:3]
```

Out[18]:
```
Term                            int64
LoanStatus                     object
ListingCreationDate    datetime64[ns]
dtype: object
```

5.6 Replacing 'Not employed' entries in IncomeRange with column with $0

In [19]:
```
clean_df.IncomeRange = clean_df.IncomeRange.str.replace('Not employed', '$0')

# Verify changes
assert 'Not employed' not in clean_df.IncomeRange
```

5.7 Handling the null values in the BorrowerAPR, DebtToIncomeRatio, EmploymentStatus, ProsperRating (Alpha) and EmploymentStatusDuration columns

> - To avoid taking chances, we have to see the amount of null values in our working dataframe
> - We will have to drop them if they are not so high, else we leave them.

In [20]:
```
# Finding the sum of null values in the columns
clean_df.isnull().sum()
```

Out[20]:
```
Term                            0
LoanStatus                      0
ListingCreationDate         68512
ListingCategory                 0
EmploymentStatus             2255
EmploymentStatusDuration     7625
IncomeRange                     0
IncomeVerifiable                0
IsBorrowerHomeowner             0
DebtToIncomeRatio            8554
StatedMonthlyIncome             0
LoanOriginalAmount              0
ProsperScore                29084
ProsperRating               29084
BorrowerAPR                    25
dtype: int64
```

In [21]:
```
# Finding the total proportion of null values in the dataframe
clean_df.isnull().sum() / clean_df.shape[0]
```

Out[21]:
```
Term                        0.000000
LoanStatus                  0.000000
ListingCreationDate         0.601315
ListingCategory             0.000000
EmploymentStatus            0.019792
EmploymentStatusDuration    0.066923
IncomeRange                 0.000000
IncomeVerifiable            0.000000
IsBorrowerHomeowner         0.000000
DebtToIncomeRatio           0.075077
StatedMonthlyIncome         0.000000
LoanOriginalAmount          0.000000
ProsperScore                0.255264
ProsperRating               0.255264
BorrowerAPR                 0.000219
dtype: float64
```

In [22]:
```
# Since they are not so high, we will drop them.
clean_df = clean_df.dropna()
```

```
# Check the result
clean_df.isnull().sum()
```

Out[22]:

```
Term                         0
LoanStatus                   0
ListingCreationDate          0
ListingCategory              0
EmploymentStatus             0
EmploymentStatusDuration     0
IncomeRange                  0
IncomeVerifiable             0
IsBorrowerHomeowner          0
DebtToIncomeRatio            0
StatedMonthlyIncome          0
LoanOriginalAmount           0
ProsperScore                 0
ProsperRating                0
BorrowerAPR                  0
dtype: int64
```

5.8 Converting ProsperRating and IncomeRange columns to ordered categorical types

In [23]:

```
# Firstly, we store the correct variable orders in a dictionary
order_dict = {'ProsperRating': ['HR', 'E', 'D', 'C', 'B', 'A', 'AA'],
              'IncomeRange': ['$0', '$1-24,999', '$25,000-49,999',
                              '$50,000-74,999', '$75,000-99,999', '$100,000+']}

# Assign each column to the proper order
for key, value in order_dict.items():
    correct_order = pd.api.types.CategoricalDtype(categories=value, ordered=True)
    clean_df[key] = clean_df[key].astype(correct_order)

# Verify changes
clean_df[order_dict.keys()].dtypes
```

Out[23]:

```
ProsperRating    category
IncomeRange      category
dtype: object
```

> Now let us see our **final working dataframe**.

In [24]:

```
clean_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 31037 entries, 3 to 113935
Data columns (total 15 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Term                      31037 non-null  int64
 1   LoanStatus                31037 non-null  object
 2   ListingCreationDate       31037 non-null  datetime64[ns]
 3   ListingCategory           31037 non-null  object
 4   EmploymentStatus          31037 non-null  object
 5   EmploymentStatusDuration  31037 non-null  float64
 6   IncomeRange               31037 non-null  category
 7   IncomeVerifiable          31037 non-null  bool
 8   IsBorrowerHomeowner       31037 non-null  bool
 9   DebtToIncomeRatio         31037 non-null  float64
 10  StatedMonthlyIncome       31037 non-null  float64
 11  LoanOriginalAmount        31037 non-null  int64
 12  ProsperScore              31037 non-null  float64
 13  ProsperRating             31037 non-null  category
 14  BorrowerAPR               31037 non-null  float64
dtypes: bool(2), category(2), datetime64[ns](1), float64(5), int64(2), object(3)
memory usage: 3.0+ MB
```

> We can see that we have **31037** rows and **15** columns to work with.

6.0 Data Analysis

> - In this section, We will explore our data systematically by building univariate and bivariate visualizations.
> - To have an idea of how the numeric values are distributed, we will first compute the descriptive statistics of the relevant numeric columns. Doing this will be helpful in creating our histogram bins for univariate explorations.

In [25]:

```
# Determining the discriptive statistics
clean_df.describe()
```

Out[25]:

	Term	EmploymentStatusDuration	DebtToIncomeRatio	StatedMonthlyIncome	LoanOriginalAmount	ProsperScore	BorrowerAPR
count	31037.000000	31037.000000	31037.000000	31037.000000	31037.000000	31037.000000	31037.000000
mean	42.818700	105.641621	0.258661	5940.875251	9306.250797	6.047782	0.224232
std	11.705389	97.322442	0.300263	4548.009095	6377.819266	2.355436	0.079140
min	12.000000	0.000000	0.000000	2.416667	1000.000000	1.000000	0.045830
25%	36.000000	32.000000	0.150000	3513.500000	4000.000000	4.000000	0.163040
50%	36.000000	77.000000	0.220000	5000.000000	8000.000000	6.000000	0.215660
75%	60.000000	151.000000	0.320000	7166.666667	14800.000000	8.000000	0.287800
max	60.000000	755.000000	10.010000	394400.000000	35000.000000	11.000000	0.423950

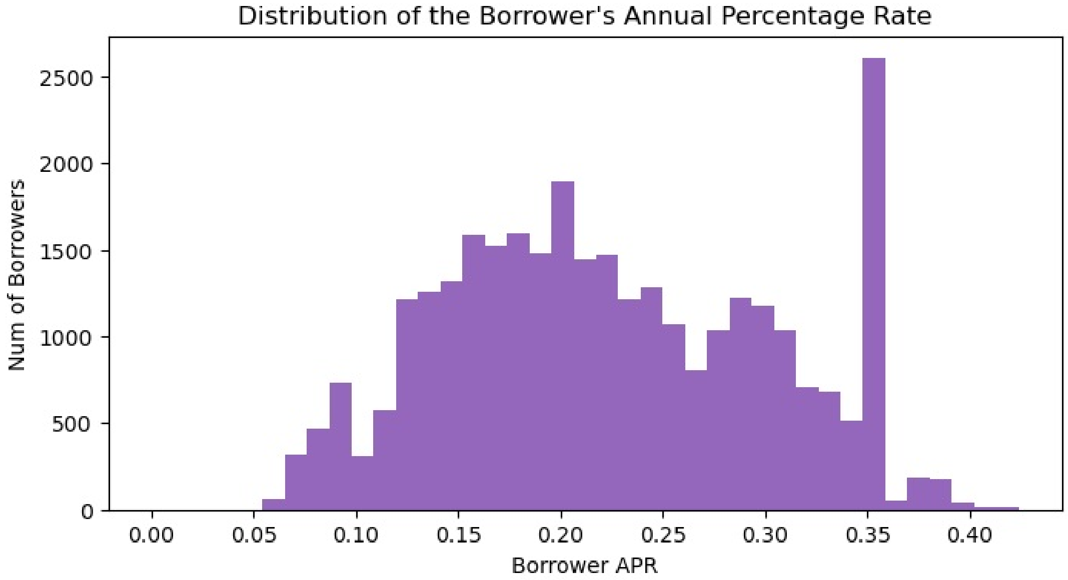
6.1 Univariate Exploration

> We shall use the "Question-Visualization-Observations" framework throughout our exploration. This framework will involve **asking a question from the data, creating a visualization to find answers, and then recording observations after each visualisation.**

Question A: How are Borrower APR values distributed in the dataset?

`Visualization`

In [26]:
```
# Pre our seaborn color for our visuals
colors = sns.color_palette('tab10')


# Create 40 evenly spaced bins for Borrower APR from zero to the maximum value
bins = np.linspace(0, clean_df.BorrowerAPR.max(), 40)

plt.figure(figsize=(8, 4))
plt.hist(data=clean_df, x='BorrowerAPR', bins=bins, color = colors[4]);
plt.xticks(np.arange(0, 0.45, 0.05))
plt.xlabel('Borrower APR');
plt.ylabel('Num of Borrowers')
plt.title("Distribution of the Borrower's Annual Percentage Rate");
```



`Observation`

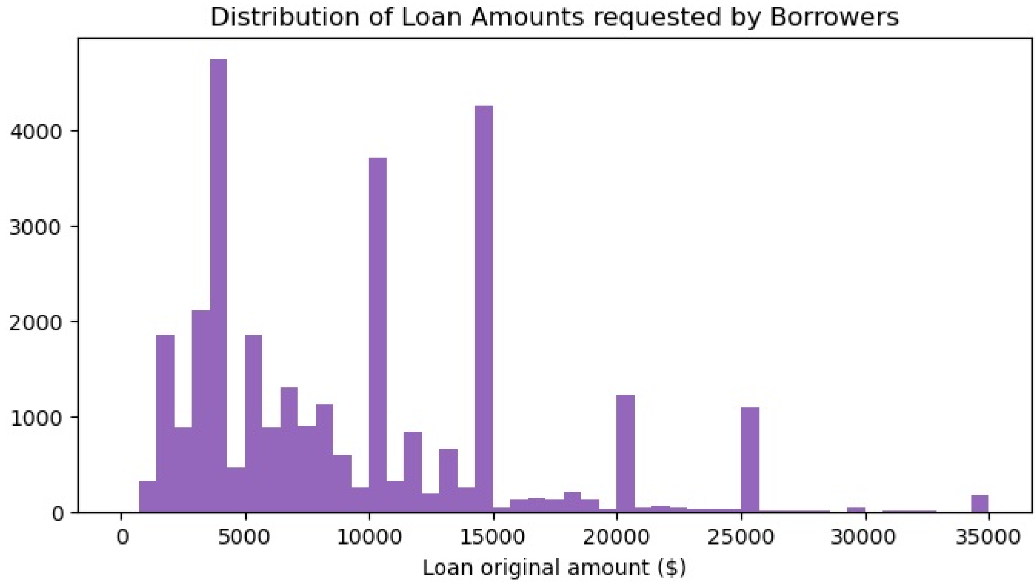
> As we can observe, distribution of Borrower APR looks multimodal with small peaks around the interval of 0.1 and a bigger ones around the interval of 0.2. Afterward, it goes on a downward trend with a peaks at 0.3 and at 0.37. Observably, Only a very few loans have APR greater than 0.43.

Question B. What is the distribution of loan amounts requested by borrowers?

`Visualization`

In [27]:
```
plt.figure(figsize=(8, 4))
bins = np.linspace(0, clean_df.LoanOriginalAmount.max(), 50)
```

```
plt.hist(data=clean_df, x='LoanOriginalAmount', bins=bins, color = colors[4])
plt.xlabel('Loan original amount ($)');
plt.title("Distribution of Loan Amounts requested by Borrowers");
```



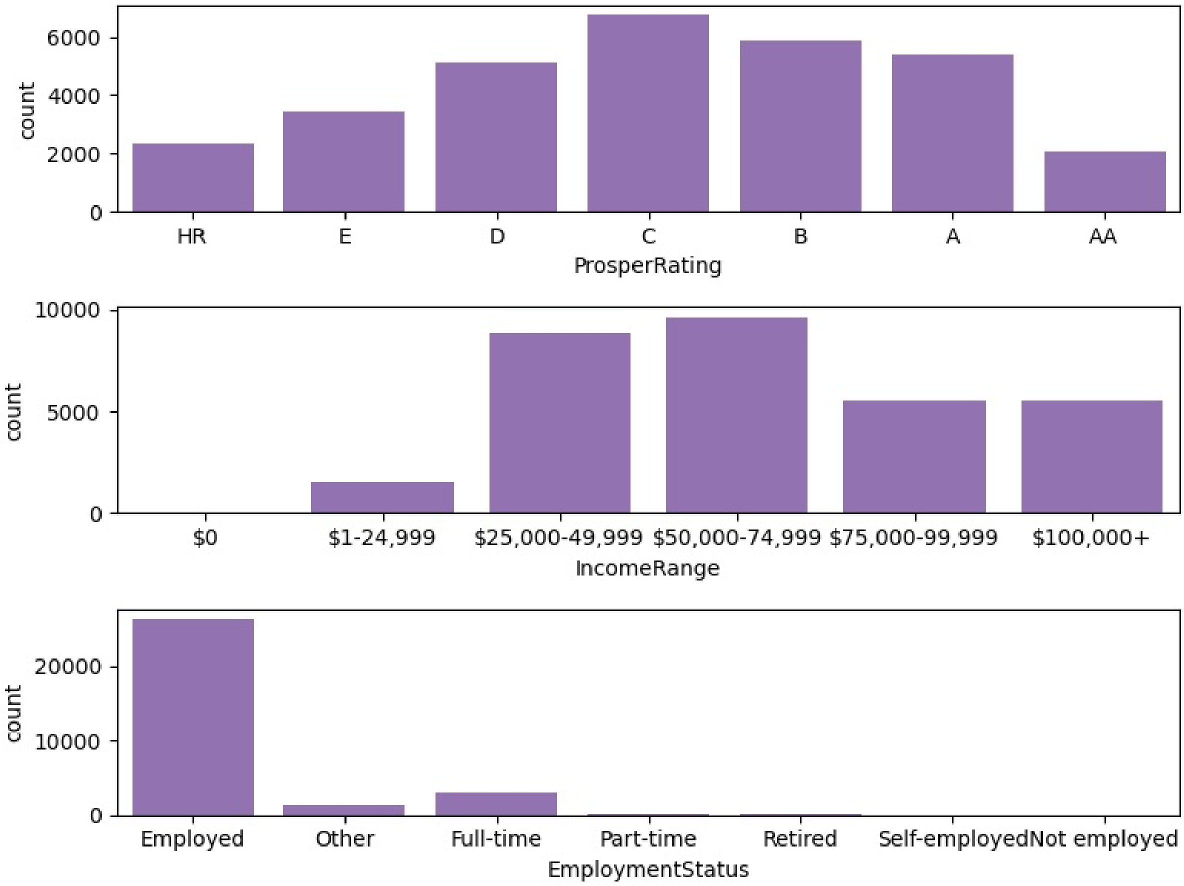
Observation

> One can observe that very few customers asked for loans that are more than 15000USD. A good number request for loans of 5000USD, or in round figure of 5000USD for easy of loan repayment.

Question C. Determine the distribution of Prosper rating, Income range and Employment Status

Visualization

In [28]:
```
columns = ['ProsperRating', 'IncomeRange', 'EmploymentStatus']

fig, ax = plt.subplots(nrows=3, figsize=(8, 6))

for col, index in zip(columns, range(3)):
    sns.countplot(data=clean_df, x=clean_df[col],color = colors[4], ax= ax[index])
plt.tight_layout();
```



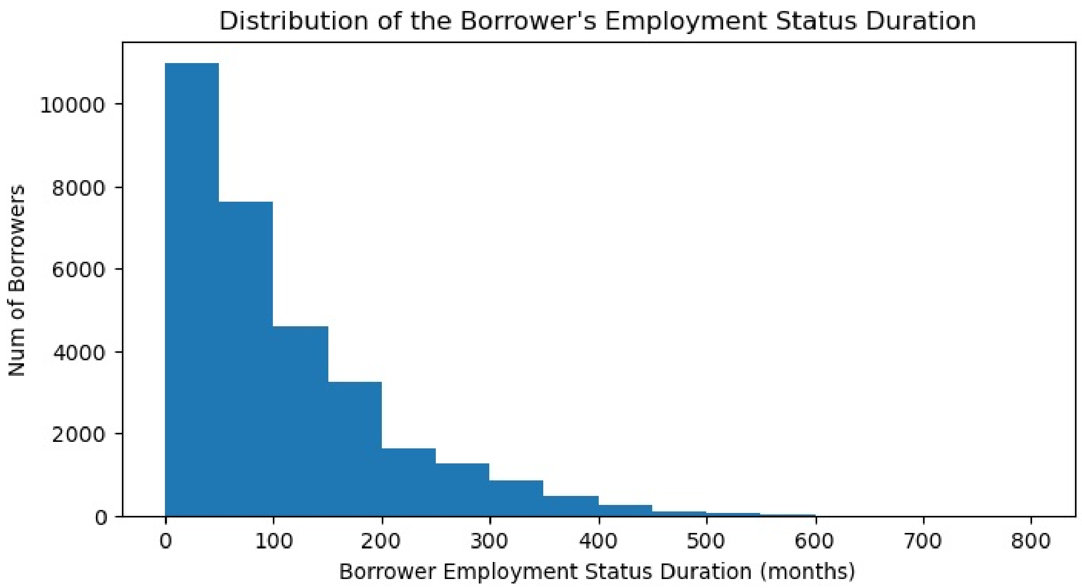
Observations

> - The distribution of **prosper ratings** of the borrowers ranges mostly from A to D, with most of the borowers falling under the C category. It can also be observed that the Listings with very high prosper ratings (AA) are the least common.
> - The **Income range** show that most of the borrowers earn between 25,000USD and 74,999USD per annum. A quite handful earn above 74,999USD, then a very few propertion of the borrowers earn below 25,000USD annually.
> - The **Employment Status** of the borrowers indicate that majority of them are employed. None of them is umemployed. This is understandable owning to the fact that it is difficult to obtain a loan without a means of loan repayment. Afterall, **Prosper Marketplace** is not a charity organization.

Question D. What is the distribution of Borrower's Employment Status Duration?

Visualization

In [29]:
```
# Plot the distribution of the EmploymentstatusDuration
rate_bins = np.arange(clean_df.EmploymentStatusDuration.min(), clean_df.EmploymentStatusDuration.max()+50, 50)

plt.figure(figsize = [8, 4])

plt.hist(data=clean_df, x='EmploymentStatusDuration', bins=rate_bins)
plt.xlabel('Borrower Employment Status Duration (months)');
plt.ylabel('Num of Borrowers')
plt.title("Distribution of the Borrower's Employment Status Duration");
```



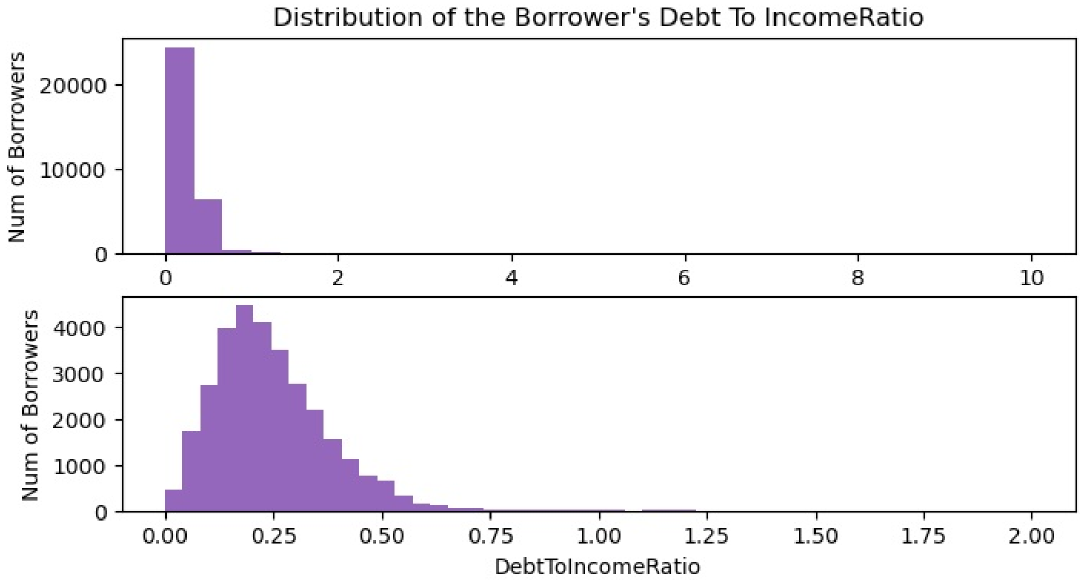
Observations

> - The distribution of Employment Status Duration is right skewed.
> - A good maority of the borrowers have an Employment duration around 4 years.

Question E. What is the values of Debt-to-income ratio of the borrowers?

Visualization

In [30]:
```
plt.figure(figsize=(8, 4))

# PLotting a distribution with 30 bins.
plt.subplot(2,1,1)
plt.hist(data=clean_df, x='DebtToIncomeRatio', bins=30, color = colors[4])
plt.xlabel('DebtToIncomeRatio');
plt.ylabel('Num of Borrowers')
plt.title("Distribution of the Borrower's Debt To IncomeRatio")

# PLoting with a closer and spaced bins
bins = np.linspace(0, 2, 50)
plt.subplot(2,1,2)
plt.hist(data=clean_df, x='DebtToIncomeRatio', bins=bins, color = colors[4])
plt.xlabel('DebtToIncomeRatio');
plt.ylabel('Num of Borrowers');
```



Observations

> - The Debt-to-Income ratio is right skewed
> - Most of the values are distributed btween 0 and 1. We can infer that most borrowrers apply for loans they can be able to repay.
> - The zoomed in analysis show that the distribution is mostly within 0.20 and 0.30. It is as if most of the borrowers maintain a debt ratio of one-fifth of their salary.

In [ ]:

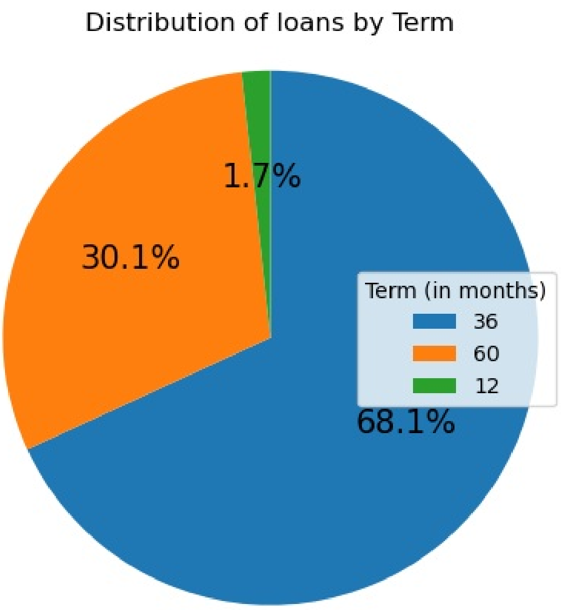
Question **. What is the distribution of loan terms requested by borrowers?

Visualization

In [31]:
```
# A view of the Term category
clean_df.Term.value_counts()
```

Out[31]:
```
36    21149
60     9353
12      535
Name: Term, dtype: int64
```

In [32]:
```
# Set dtype of 'LoanStatus' to category
term_order = clean_df['Term'].value_counts().index
ordered_var = pd.api.types.CategoricalDtype(ordered = True,
                                            categories = term_order)
clean_df['Term'] = clean_df['Term'].astype(ordered_var)

# Print the proportion below the bars
plt.figure(figsize=[10, 5])

sorted_term = pd.DataFrame(clean_df['Term'].value_counts().reset_index())
labels = sorted_term['index']
textprops = {"fontsize":15}

plt.pie(data = sorted_term, x = 'Term', labels = None, startangle = 90,
        counterclock = False, autopct='%.1f%%', textprops = textprops)
plt.axis('square')
plt.title('Distribution of loans by Term')
plt.legend(labels, title = 'Term (in months)',loc=5);
```



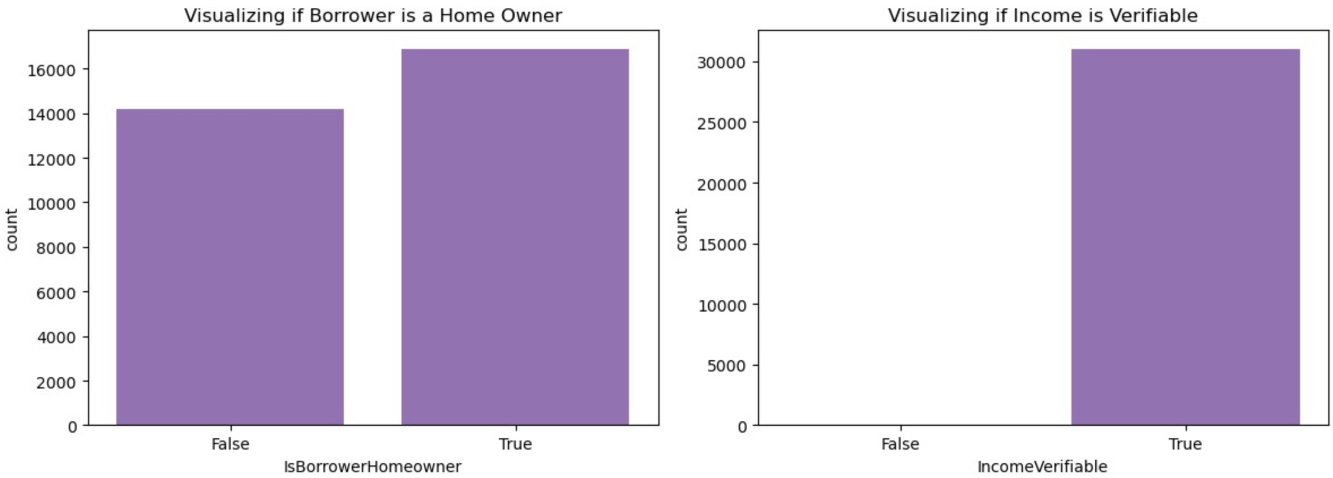
`Observation`

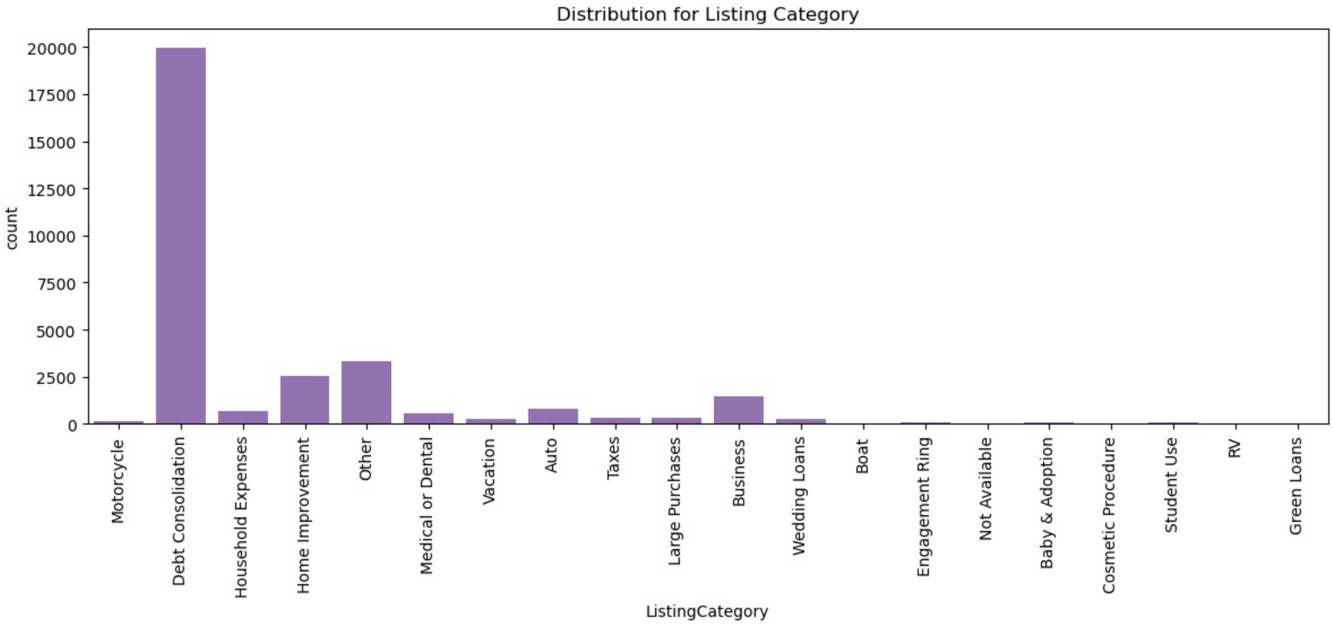
> From the pie, about 68.1% of the borrowers sought for a 36 months (3 years) loan term. About 30.1% sought for a 60 months (5years) loan term, while only a few 1.7% of the borrowers sought for a 12 months (1 year) loan term. This data futhers shows that there are only three loan terms in **Prosper MarketPlace**.

Question G. Visualize the pattern of distribution of loan by IsBorrowerHomeowner, IncomeVerifiable and ListingCategory columns.

`Visualization`

In [33]:
```
fig = plt.figure(figsize=(12, 10))
# IsBorrowerHomeowner
plt.subplot(2,2,1)
sns.countplot(data=clean_df, x='IsBorrowerHomeowner', color = colors[4]);
plt.title("Visualizing if Borrower is a Home Owner")

# IncomeVerifiable
plt.subplot(2,2,2)
sns.countplot(data=clean_df, x='IncomeVerifiable', color = colors[4]);
plt.title("Visualizing if Income is Verifiable")

# ListingCategory
plt.subplot(2,1,2)
sns.countplot(data=clean_df, x='ListingCategory', color = colors[4]);
plt.title("Distribution for Listing Category")
plt.xticks(rotation=90)
plt.tight_layout();
```





`Observations`

> - The **Home Owners** visual shows that greater number of borrowers are home owners.
> - In **Income Verification**, it shows that almost all of the borrowrs have verfiable income. It lays credence of Employment status above (Question C) that demonstrates taht most borrowers are employed.
> - In the **Listing Category**, the Debt consolidation is the most popular. One could deduce that it seems that borrowers use loans to repay older loans. This category is so much wen compared to other groups like Household Expenses, Medical or Dental, Auto, etc.

6.1 Answering Some Questions on our Univariate Exploration

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

> - The variable that caught my interest the more is the `Distribution for Listing Category`. I was surprised to notice that rather than take loans to establish businesses or purchase valuable assets, most of the borowers used their loans to service their debts. Borrowing for consumption has been seriously discouraged by prominent economist as being self-indulgent.
> - Again, in the `LoanStatus`, I observed that maority of the loans are marked 'Current'. This goes to point that many of the borrowers are still in debt. There are other insights from the visuals, but there is nothing very unsual.

B. 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?

> To help for a quality data exploration, we put together our main features into another dataframe called `clean_df`. Other operations happened along the cause of this exposition:

- The LoanOriginalAmout is right skewed. There were large amounts ranging from 5000USD to 25000USD which were the common ones people took.
- The ProsperRating (numeric), ListingCategory (numeric) and ProsperRating (Alpha) column names were changed to facilitate our exploration.
- No duplicate was found.
- We changed the LoanOriginaionDate variable type to datatime obect.
- There were other operations that were made as seen in our visualization's observations.

6.2 Bivariate Exploration

> In this section, we will investigate relationships between pairs of variables in our dataframe. The variables that are covered here have been introduced in the previous section (univariate exploration) above.
>
> - We will begin by exploring correlations between our numerical variables, as it will provide a lucid insight into our target variables which are Borrower APR and Prosper rating.

In [34]:
```
#A look at our dataframe
clean_df
```

Out[34]:

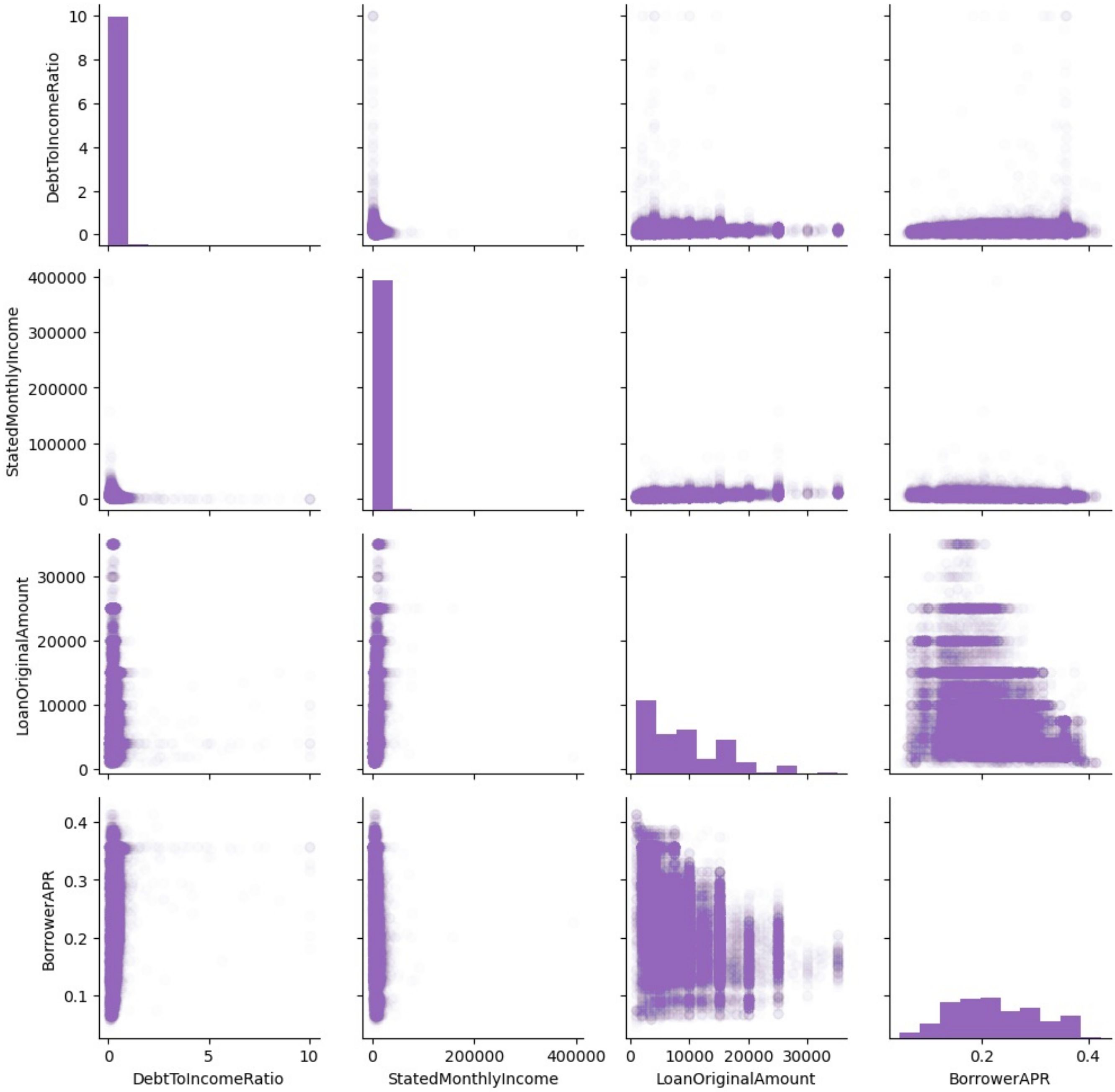
	Term	LoanStatus	ListingCreationDate	ListingCategory	EmploymentStatus	EmploymentStatusDuration	IncomeRange	IncomeVerifiabl
3	36	Current	2023-02-12 02:35:00	Motorcycle	Employed	113.0	$25,000-49,999	Tru
10	60	Current	2023-02-12 04:01:36	Debt Consolidation	Employed	300.0	$75,000-99,999	Tru
12	36	Past Due (1-15 days)	2023-02-12 01:10:48	Debt Consolidation	Employed	1.0	$25,000-49,999	Tru
14	60	Current	2023-02-12 17:41:42	Debt Consolidation	Employed	35.0	$100,000+	Tru
15	36	Defaulted	2023-02-12 14:46:18	Household Expenses	Other	121.0	$50,000-74,999	Tru
...	...	...	...	...	...	...	...	.
113916	36	Current	2023-02-12 07:36:36	Household Expenses	Employed	149.0	$50,000-74,999	Tru
113924	60	Current	2023-02-12 15:52:42	Household Expenses	Employed	56.0	$25,000-49,999	Tru
113928	36	Completed	2023-02-12 02:44:24	Business	Full-time	22.0	$25,000-49,999	Tru
113931	60	Current	2023-02-12 13:08:00	Business	Employed	12.0	$75,000-99,999	Tru
113935	60	Completed	2023-02-12 18:26:36	Home Improvement	Full-time	94.0	$25,000-49,999	Tru

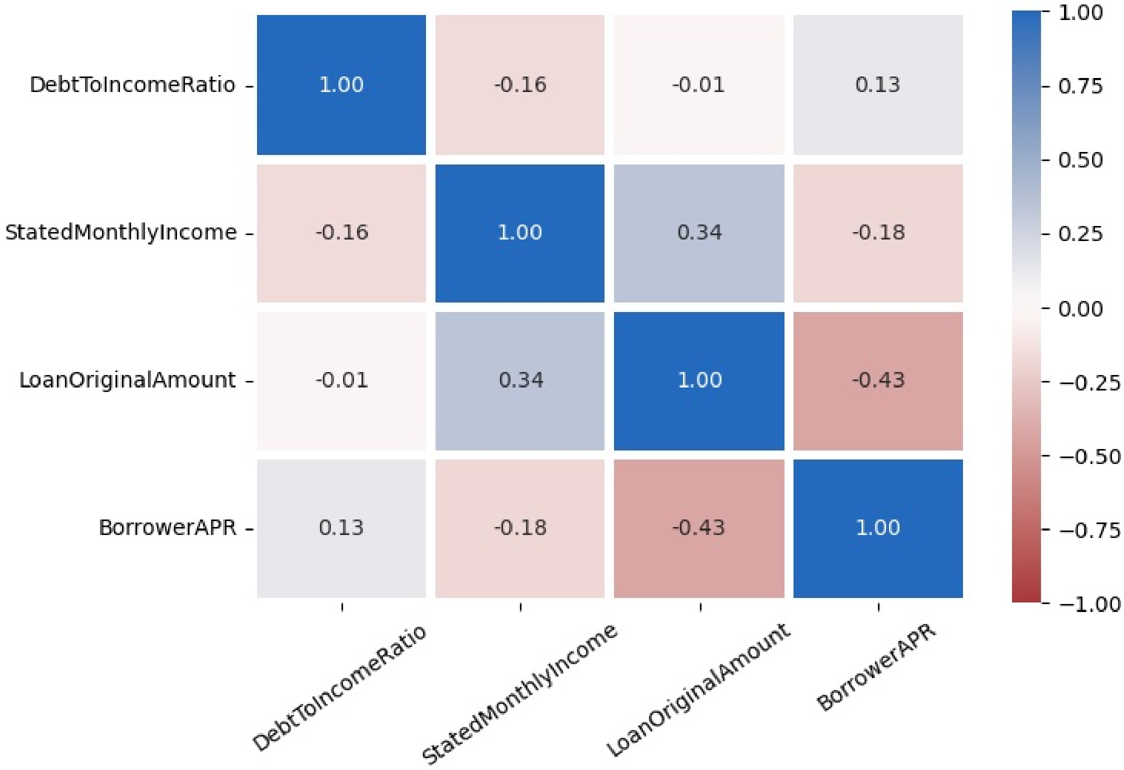
31037 rows × 15 columns

Question A. What is the relationship you observed amongst the numeric variables [DebtToIncomeRatio, StatedMonthlyIncome, LoanOriginalAmount, BorrowerAPR and Term]. Do they show any correlation?

`Visualization`

In [37]:
```
#we will use the seaborn PairGrid and Heatmat

# creating a list of numeric variables
numeric_vars = ['DebtToIncomeRatio', 'StatedMonthlyIncome', 'LoanOriginalAmount', 'BorrowerAPR', 'Term']

# Creating a scatterplot of each numeric variable against the other
fig = sns.PairGrid(clean_df[numeric_vars])
fig.map_diag(plt.hist, color = colors[4])
fig.map_offdiag(plt.scatter,color = colors[4], alpha=0.01);

# Visualizing the correlation between the numeric variables with a heatmap
plt.figure(figsize= (7.5, 5))
sns.heatmap(data=clean_df[numeric_vars].corr(), annot=True, fmt='.2f',
            cmap='vlag_r', vmin=-1, vmax=1, linewidth=3)
plt.xticks(rotation=35);
```





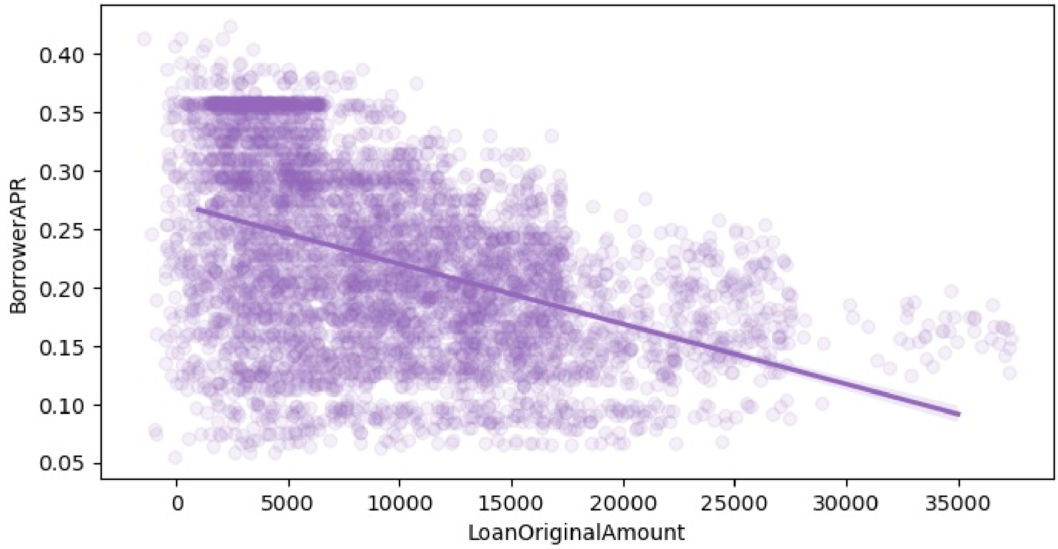
Observations

> - Our chart shows that there is a negative correlation (-0.43) between loan original amount and Borrower APR. From this, we could deduce that larger loans may attract lesser annual percentage rates than smaller loans.
> - When it comes to Loan term and LoanOriginalAmount, we could notice a positive correlation (0.34). This demonstrates that borrowers may need a longer term to pay off higher loans.

Question B. Can you visualize the relationship between BorrowerAPR and LoanOriginalAmount only.

`Visualization`

In [38]:
```
# using regplot and a sample of 5000 to avoid overplotting

plt.figure(figsize=(8, 4))
sns.regplot(data=clean_df.sample(5000, random_state=1), x='LoanOriginalAmount', y='BorrowerAPR',
            x_jitter=2500, color = colors[4], scatter_kws={'alpha': 0.1});
```



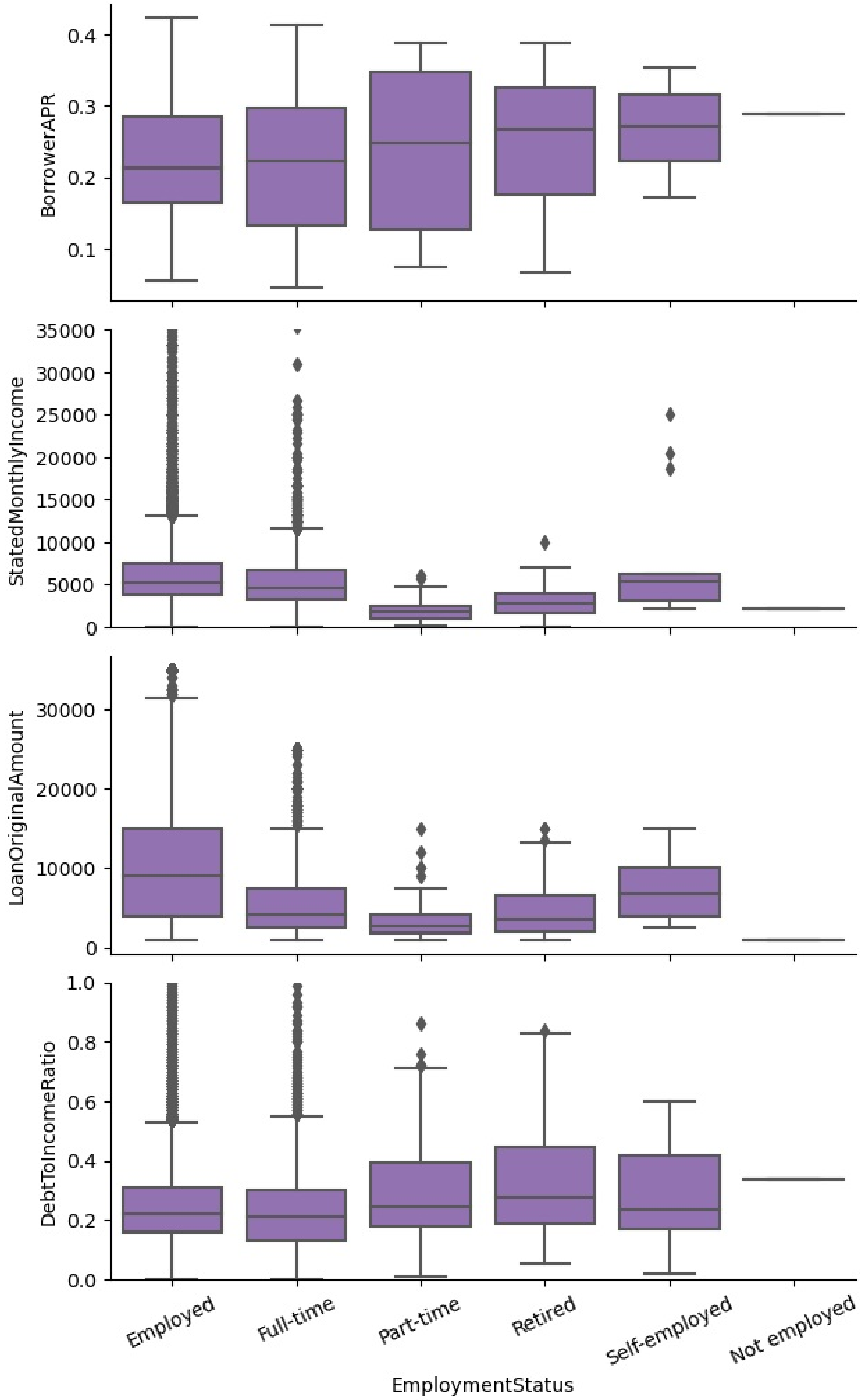
`Observations`

> - There exist a relationship between BorrowerAPR and LoanOriginalAmount variables. We could deduce from it that the interest rate on loans with a large amount is lower, and the reverse is the case with lower amounts.

Question C. Explore the bivariate relationships between Employment status and the numerical variables: BorrowerAPR, StatedMonthlyIncome, LoanOriginalAmount, DebtToIncomeRatio?

`Visualization`

In [42]:
```
# Create a list for the y_col columns to plot on each pairgrid axis
y_cols = ['BorrowerAPR', 'StatedMonthlyIncome', 'LoanOriginalAmount','DebtToIncomeRatio']

# Then, sieve out entries where employment status is 'other', for the x_var
employment_filter = clean_df.query('EmploymentStatus != "Other"')

fig = sns.PairGrid(data=employment_filter, y_vars= y_cols, x_vars='EmploymentStatus', aspect=2.5)
fig.map(sns.boxplot, color=colors[4])

# Limit the y axis of StatedMonthlyIncome to 0 - 50000
fig.axes[1][0].set_ylim(0, 35000)
# Limit the y axis of DebtToIncome ratio to 0 - 1
fig.axes[3][0].set_ylim(0, 1)
plt.xticks(rotation=25);
```



Observations

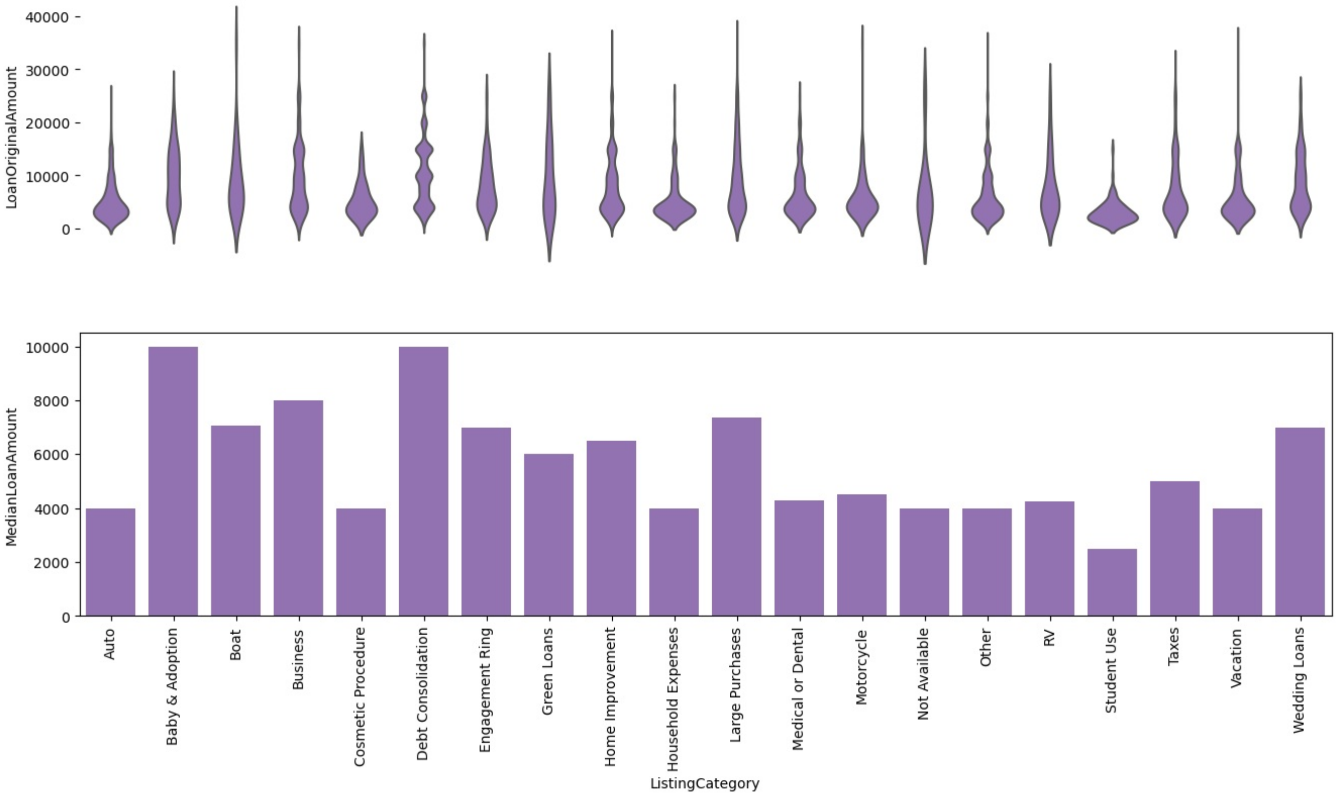
> - We can see from the chart that the Employed and Fulltime category with a higher montly income receive a higer loan amonts and thereby enjoy a lower BorrowerAPR than other borrowers - retires, part-time, etc.
> - It is observed that the Employed and full-time borrowers have lower debt to income ratios compared to others - part-time, retired, etc.

Question D. Why do borrowers apply for loan on the average? Using the relationship between ListingCategory and LoanOriginalAmount.

Visualization

In [44]:
```
# First use the median loan amount as the measure of our average value,
#then compute it by listing category

table = clean_df.groupby('ListingCategory')['LoanOriginalAmount'].median()
table = table.reset_index(name='MedianLoanAmount')
```

```python
# Visualize with Violinplot
plt.figure(figsize=(16, 8))
plt.subplot(2,1,1)
sns.violinplot(data=clean_df.sort_values(by='ListingCategory'), x='ListingCategory', y='LoanOriginalAmount', in
               color=colors[4])
# Remove violinplot ticks, labels and spines
plt.xticks([])
plt.xlabel('')
sns.despine(left=True, bottom=True)

# Use barplot
plt.subplot(2,1,2)
sns.barplot(data=table, x='ListingCategory', y='MedianLoanAmount', ci=None, color=colors[4])
plt.xticks(rotation= 90);
```



Observation

> - The pattern here show that the borrowers take loan more for Debt Consolidation, Baby & Adoption, Business and Wedding loans. Students apply for the least loan on average. One could deduce that the borrowers seem to need the loan more for simply personal expenditures than for investments. Sadly, they basically borrow to consume.

Question D. Consider the relationship ProsperRating and IncomeRange vs Home onwership

Visualization

> To make use of a clustered bar chart, we will have to define two functions - *the compute_proportions()* which will compute the proportion of the y-axis within the x-axis variable, wile the second function, *plot_proportions()* will plot a column bar chart that will be based on the computed proportions.

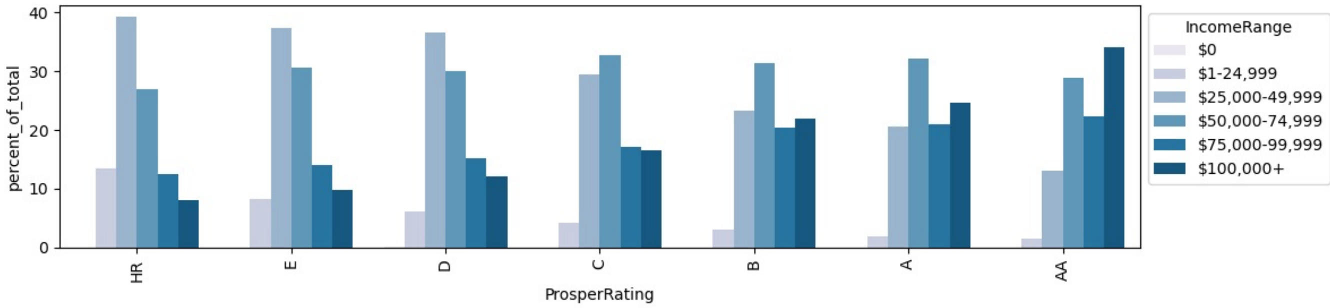
In [66]:

```python
# defining the two proportions
def compute_proportions(df, group_col, proportion_col):
    """
        Computes the proportions of proportion_col within group_col
        Params:
            :df (dataframe): dataframe of interest
            :group_col (string): name of grouping column
            :proportion_col (string): name of column to compute proportions for
        Output:
            returns a dataframe with relative frequencies of proportion_col within group_col
    """
    # First group dataframe by group_col and proportion_col
    result = df.groupby([group_col, proportion_col]).agg({proportion_col: 'size'})

    # Compute the proportions of proportion_col within group_col
    result = result.groupby(level=0).apply(lambda x: 100 * x / float(x.sum()))

    # Rename proportion results and reset dataframe index
```

```
    result.rename(columns={proportion_col: 'percent_of_total'}, inplace=True)
    result = result.reset_index()
    return result

def plot_proportions(df, group_col, proportion_col, cmap):
    """ Creates a clustered bar chart  of prop_col and group_col"""
    # Call the compute proportion function
    table = compute_proportions(df, group_col, proportion_col)
    # Create Column bar chart
    sns.barplot(data=table, x= group_col, y= 'percent_of_total', hue=proportion_col, palette=cmap, color=colors
    plt.legend(bbox_to_anchor=(1,1), loc="upper left", title=proportion_col)
```

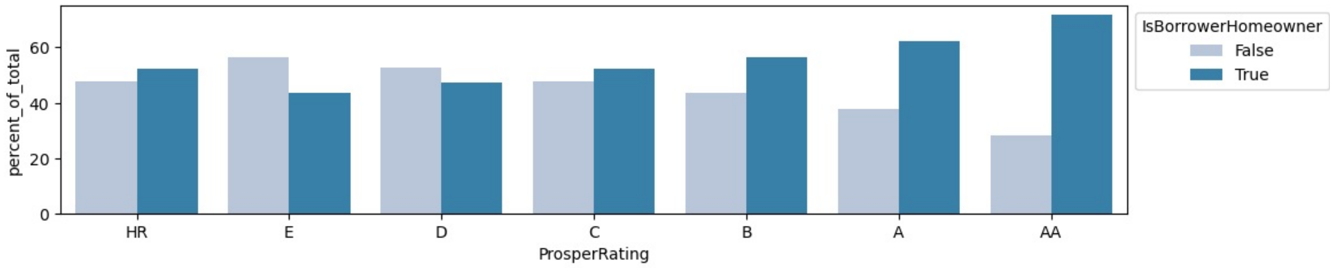
> - **Is there any Relationship between Propser Rating and Income Range?**

In [72]:
```
plt.figure(figsize=(12,8))

# ProsperRating and IncomeRange
plt.subplot(3,1,1)
plot_proportions(clean_df, 'ProsperRating', 'IncomeRange','PuBu')

plt.xticks(rotation=90)
plt.tight_layout();
```



> - **Visualize the Relationship between Propser Rating and Home Owership**

In [77]:
```
plt.figure(figsize=(12,8))
# Prosper Rating and IsBorrowerHomeowner
plt.subplot(3,1,2)
plot_proportions(clean_df, 'ProsperRating', 'IsBorrowerHomeowner', 'PuBu')
```



Observation

> - It is observed that as ProsperRating increases, the proportion of high income earners also increase.
> - ProsperRating also has a postive correlation with HouseOwnership. So, The top ProsperRatings (AA, A and B) have more proportions of homeowners, while the lower prosper ratings have lesser proportions of homeowners. Then, the lowest ProsperRating, (HR), is dominant among unemployed borrowers.

6.4 Observations from the Bivariate Exploration

> From our key points of interest, we can observe that:
>
> - The BorrowerAPR shows an inverse correlation between Prosper rating and LoanOriginalAmount. Thus, the interest rate on loans with higher amounts is less.
> - While the ProsperRating shows an inverese correlation with the DebtToIncomeRatio, and the positive correlation with the IncomeRange, IncomeVerifiability, HomeOwenership.

7.0 Conclusion

> From the foregoing, the Prosper loan dataset has demonstrated different motivation of borrowers as well as some factors that influence their loan favorability. We noticed that most of the borrowers take loans to service their debts. Thus, Debt cosolidation turned out to be the hightes motivation of the borrowers. Other motivations that topped the chart is that of Child& adoptions, Wedding and then Business. It seems that our borrowers have less need of investments.
>
> Furthermore, measuring loan favourability to the Borrowers Annual Percentage Rate, we noticed a negative correlation between BorrowerAPR and variables like Loan Original Amount, Loan Term and Prosper Rating. The Prosper Rating on its own seem to have been influnced by High and Verifiable incomes, Home onwership , low debt to income ration as well as presence of a measn of employment.

7.1 Limitation

> As a limitation to our analysis, we much as we exmined the correlations between our main features of interest, we cannot firmly say that one feature influenced other with certainty. Thus, this study was only an observatory from the dataset at hand.

7.2 Saving our work

In [84]:

```
# Save our finished cleaned data locally
clean_df.to_csv('Prosper_Project.csv')
```

In [ ]:

```

```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js